ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

«Έξυπνη» εφαρμογή ηλεκτρονικού εμπορίου για Ταμπλέτες με λειτουργικό σύστημα Android

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

Στέφανου - Μάριου Γ. Χουρδάκη

Επιβλέπων : Ιωάννης Βασιλείου,

Καθηγητής Ε.Μ.Π.

Αθήνα, Δεκέμβριος 2013

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

«Έξυπνη» εφαρμογή ηλεκτρονικού εμπορίου για Ταμπλέτες με λειτουργικό σύστημα Android

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

Στέφανου - Μάριου Γ. Χουρδάκη

Επιβλέπων : Ιωάννης Βασιλείου,

Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 18η Δεκέμβρη 2013

.....	.....	.....
Ιωάννης Βασιλείου, καθηγητής Ε.Μ.Π.	Νεκτάριος Κοζύρης , αναπληρωτής καθηγητής Ε.Μ.Π.	Κώστας Κοντογιάννης, αναπληρωτής καθηγητής Ε.Μ.Π.

Αθήνα, Δεκέμβριος 2013

.....

Στέφανος - Μάριος Γ. Χουρδάκης

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Ο σκοπός της παρούσας διπλωματικής εργασίας είναι η δημιουργία μιας εφαρμογής που θα επιτρέπει στον χρήστη να ψωνίζει σε υπάρχοντα μεγάλα ηλεκτρονικά καταστήματα πιο εύκολα και πιο έξυπνα μέσα από την ταμπλέτα του. Η εφαρμογή αναπτύχθηκε σε γλώσσα προγραμματισμού Java για χρήση σε περιβάλλον Android.

Ο χρήστης μέσω της εφαρμογής έχει τη δυνατότητα να περιηγηθεί ταυτόχρονα σε διάφορα υπάρχοντα μεγάλα ηλεκτρονικά καταστήματα, να εξετάσει αν υπάρχουν διαθέσιμα ειδικά ηλεκτρονικά εκπτωτικά κουπόνια τα οποία μπορεί να ανήκουν στον ίδιο ή να τα μοιράζεται με άλλους χρήστες και εν τέλη να πλοηγηθεί μέχρι την πραγματική ηλεκτρονική σελίδα κάποιου υπάρχοντος προϊόντος για να κάνει την αγορά του.

Συγκεκριμένα, η εφαρμογή επικοινωνεί σε πραγματικό χρόνο με κεντρικό διακομιστή, ενημερώνεται για υπάρχοντα δεδομένα (προϊόντα, εκπτωτικά κουπόνια κλπ) σε αυτόν και τα παρουσιάζει με ελκυστικό και έξυπνο τρόπο στον χρήστη κάνοντας την χρήση πιο εύκολη και διαδραστική.

Η εφαρμογή μπορεί να εγκατασταθεί και να χρησιμοποιηθεί σε κάθε ταμπλέτα που χρησιμοποιεί λειτουργικό σύστημα Android, ανεξαρτήτως μεγέθους, με μόνη προϋπόθεση η έκδοση του λειτουργικού να είναι η 4.2 ή και νεότερη.

Λέξεις Κλειδιά

Ηλεκτρονικό εμπόριο, ταμπλέτες, Android, πραγματικού χρόνου, οθόνη αφής, προϊόντα, καταστήματα, κουπόνια

Abstract

The purpose of this thesis is the development of an application that enables the user to shop easier and smarter on big existing online stores through his tablet. The application is developed on Java programming language for use on Android operating systems.

The user has the ability, through the application, to browse different big existing online stores at the same time, verify if there are any special electronic discount coupons available that may belong to the user or are shared by other users and finally to browse all the way to the real existing internet page of an existing product so he can purchase it.

Specifically, the application communicates in real time with a main server, gets updated on existing data (products, discount coupons etc.) in the server and displays them in an appealing and smart way, thus making the use of the application easier and more interactive.

The application can be installed and used on every tablet that runs on Android operating system, independently of its screen size, with lone requirement that the version of the operating system be 4.2 or higher.

Keywords

Online shopping, tablets, Android, real time, touch screen, products, stores, coupons

Πίνακας Περιεχομένων

1. Εισαγωγή στις Κινητές Συσκευές (Mobile Devices)

1.1 Γενική Περιγραφή

Ο όρος **κινητές συσκευές** (mobile devices) ή **υπολογιστές χειρός** (handheld computers) αναφέρεται σε μικρές υπολογιστικές συσκευές οι οποίες συνήθως έχουν μια οθόνη αφής και/ή ένα μικρό πληκτρολόγιο και ζυγίζουν λιγότερο από 1 κιλό. Εταιρίες όπως οι Nokia, Samsung, Apple, HTC, LG είναι μερικές μόνο από τις εταιρίες που ασχολούνται παγκοσμίως με την παραγωγή και πώληση κινητών συσκευών.

Μια κινητή συσκευή βασίζεται σε ένα λειτουργικό σύστημα (OS) και μπορεί να τρέξει διάφορων ειδών λογισμικά εφαρμογών τα οποία είναι γνωστά ως εφαρμογές (apps). Οι περισσότερες κινητές συσκευές είναι εξοπλισμένες με Wi-Fi, Bluetooth, GPS και έχουν την δυνατότητα σύνδεσης στο διαδίκτυο. Επίσης, διαθέτουν συνήθως ειδική κάμερα με δυνατότητα λήψης φωτογραφιών και βίντεο καθώς και ενσωματωμένο media player για αναπαραγωγή αρχείων βίντεο και ήχου.

Η πρόσφατη ανάπτυξη της τεχνολογίας στο πεδίο των κινητών συνεργατικών συστημάτων (mobile collaboration systems) έχει δώσει τη δυνατότητα παραγωγής κινητών συσκευών με τη δυνατότητα βίντεο, ήχου και σχεδιασμού στην-οθόνη (on-screen drawing) έτσι ώστε να καθιστά δυνατή την συνδιάσκεψη πολλών ατόμων, ανεξαρτήτως τοποθεσίας, σε πραγματικό χρόνο.

Οι κινητές συσκευές είναι διαθέσιμες σε ένα μεγάλο εύρος μορφών και ειδών, όπως είναι τα **«έξυπνα» τηλέφωνα** (**smartphones**), PDA χειρός, Ultra-Mobile PC και **Tablet PC**. Στην παρούσα διπλωματική θα γίνει αναφορά στα **«έξυπνα» τηλέφωνα** (**smartphones**) και στα **Tablet PC** όπου και θα είναι διαθέσιμη η εφαρμογή.

1.2 «Έξυπνα» Τηλέφωνα (Smartphones)

Ένα **«έξυπνο» τηλέφωνο** (**smartphone**) είναι ένα κινητό τηλέφωνο σχεδιασμένο πάνω σε ένα φορητό λειτουργικό σύστημα (mobile operating system) με πιο εξελιγμένη υπολογιστική ικανότητα από ένα απλό κινητό τηλέφωνο (feature phone). Τα πρώτα «έξυπνα» τηλέφωνα συνδύαζαν τις δυνατότητες ενός «προσωπικού ψηφιακού βοηθού» (PDA – personal digital assistant), συμπεριλαμβανομένης της δυνατότητας ηλεκτρονικού ταχυδρομείου (e-mail), με αυτές ενός κινητού τηλεφώνου. Οι επόμενες εκδόσεις προσέθεσαν την λειτουργικότητα φορητών media player, χαμηλής τεχνολογίας ψηφιακών φωτογραφικών μηχανών, συμπαγών

video camera και μονάδων πλοήγησης GPS. Τα περισσότερα σύγχρονα «έξυπνα» τηλέφωνα περιέχουν οθόνες αφής υψηλής ανάλυσης και περιηγητές διαδικτύου (web browsers) με τη δυνατότητα περιήγησης και απεικόνισης διαδικτυακών ιστοσελίδων. Υψηλή ταχύτητα πρόσβασης σε δεδομένα (High speed data access) παρέχεται μέσω Wi-Fi, κινητής ευρυζωνικότητας (mobile broadband) και Bluetooth.

Τα φορητά λειτουργικά συστήματα (mobile operating systems, OS) που χρησιμοποιούνται από τα σύγχρονα «έξυπνα τηλέφωνα» περιλαμβάνουν το **Android** της Google, το iOS της Apple, το Symbian της Nokia, το BlackBerry 10 της Blackberry Ltd, το Bada της Samsung, το Windows Phone της Microsoft, το webOS της Hewlett-Packard και ενσωματωμένες διανομές Linux όπως τα Maemo και MeeGo. Μερικά πολυαναμενόμενα λειτουργικά συστήματα κατά της διάρκεια της συγγραφής της παρούσας διπλωματικής είναι το Firefox OS της Mozilla, το Ubuntu Phone της Canonical Ltd., και το Tizen.

Εικόνα 2.1 : Το Nexus 4 της Google

Αξίζει να αναφερθεί ότι στην αρχή του 2013 οι πωλήσεις των «έξυπνων» τηλεφώνων ξεπέρασαν αυτές των απλών κινητών τηλεφώνων.

1.3 Ταμπλέτες (Tablet computers)

Ένας υπολογιστής ταμπλέτα, ή απλά **ταμπλέτα** (**tablet**), είναι ένας φορητός υπολογιστής με οθόνη, κύκλωμα και μπαταρία όλα σε μία μονάδα. Οι ταμπλέτες είναι εξοπλισμένες με αισθητήρες, συμπεριλαμβανομένων κάμερας, μικροφώνου, επιταχυνσιομέτρου (accelerometer) και οθόνης αφής, με τις κινήσεις του δαχτύλου ή της ειδικής ακίδας να αντικαθιστούν την λειτουργία του ποντικιού και του πληκτρολογίου. Οι ταμπλέτες μπορεί να έχουν και φυσικά πλήκτρα για να εκτελούν βασικές λειτουργίες όπως είναι η ενεργοποίηση/απενεργοποίηση της ταμπλέτας, η ρύθμιση της ένταση του ήχου και θύρες για επικοινωνία και για φόρτιση της μπαταρίας. Ένα εικονικό πληκτρολόγιο στην-οθόνη (virtual on-screen keyboard) χρησιμοποιείται συνήθως για την πληκτρολόγηση. Οι ταμπλέτες είναι τυπικά μεγαλύτερες σε μέγεθος από ότι είναι τα «έξυπνα» τηλέφωνα (Παράγραφος 1.2) ή τα PDA, με το μέγεθος τους να ξεκινάει από τις 7 ίντσες (18 εκατοστά) και πάνω, μετρούμενες διαγώνια.

Η ιδέα πρωτοξεκίνησε στα μέσα του 20^{ου} αιώνα με τα πρώτα πρωτότυπα και την ανάπτυξη τους να έρχονται στις τελευταίες δύο δεκαετίες εκείνου του αιώνα. Οι ταμπλέτες όμως άρχισαν να γίνονται δημοφιλείς μόλις το 2010. Εμφανίστηκαν κυρίως ως εξέλιξη και πρόσθετο «αξεσουάρ» (gadget) στα δημοφιλή «έξυπνα» τηλέφωνα αλλά άργησαν να κατακτήσουν την αγορά. Με την σταθερή εγκαθίδρυση όμως των πρώτων και την αντικατάσταση των απλών τηλεφώνων με «έξυπνα», οι ταμπλέτες τα τελευταία δύο χρόνια έχουν γίνει αρκετά δημοφιλείς, ειδικά στο νεανικό καταναλωτικό κοινό. Τον Μάρτιο του 2013, το 31% των χρηστών του διαδικτύου στις Η.Π.Α ανέφερε πως κατέχει ταμπλέτα την

οποία χρησιμοποιεί κυρίως για να βλέπει δημοσιευμένο περιεχόμενο όπως βίντεο και ειδήσεις. Ανάμεσα σε όλες τις ταμπλέτες που βρίσκονταν σε κυκλοφορία το 2012, η Apple με το iPad είχε πουλήσει 100 εκατομμύρια κομμάτια από τον Απρίλη του 2010 μέχρι τον Οκτώβρη του 2012.

Οι σημερινές ταμπλέτες χρησιμοποιούν χωρητικές οθόνες αφής (capacitive touchscreens) με πολυ-επαφή (multi-touch) σε αντίθεση με παλαιότερες που χρησιμοποιούσαν μόνο οθόνες «αντίστασης» (resistive) με χρήση ακίδας (stylus). Αυτά τα δύο είναι και τα μόνα βασικά είδη που χρησιμοποιούνται στην τεχνολογία οθονών αφής.

- Οι οθόνες αντίστασης (resistive touchscreens) είναι παθητικές και ανταποκρίνονται στην άσκηση πίεσης στην οθόνη. Επιτρέπουν ακρίβεια υψηλού επιπέδου και για τη λειτουργία τους χρησιμοποιείται συνήθως μια ακίδα (stylus).
- Οι χωρητικές οθόνες (capacitive touchscreens) συνήθως είναι λιγότερο ακριβής αλλά ανταποκρίνονται πιο άμεσα από ότι οι οθόνες αντίστασης. Επειδή απαιτούν ένα αγώγιμο υλικό, όπως είναι το δάχτυλο, για είσοδο, δεν συνηθίζονται για χρήση με συσκευές που χρησιμοποιούν ακίδα αλλά είναι διαδεδομένες σε συσκευές ευρείας κατανάλωσης.

Όπως και με τα «έξυπνα» τηλέφωνα, οι περισσότερες εφαρμογές (apps) παρέχονται μέσω διαδικτυακής διανομής (online distribution). Η διανομή αυτή γίνεται μέσα από εξειδικευμένα διαδικτυακά «καταστήματα», τα οποία είναι γνωστότερα ως «καταστήματα εφαρμογών» (app stores). Τα καταστήματα αυτά παρέχουν κεντρικούς καταλόγους με λογισμικό και επιτρέπουν την αγορά, την εγκατάσταση και την ενημέρωση του λογισμικού με «ένα-κλικ» μέσα από την συσκευή. Το «κατάστημα εφαρμογών» είναι συνήθως το ίδιο για τα «έξυπνα» τηλέφωνα και για τις ταμπλέτες που χρησιμοποιούν το ίδιο λειτουργικό σύστημα.

Εικόνα 2.2 : Το iPad της Apple

Εικόνα 2.3 : Το Galaxy Tab της Samsung

2. Το λειτουργικό σύστημα Android

2.1 Τι είναι το Android

Εικόνα 2.4 : Το λογότυπο του Android

Το **Android** είναι ένα λειτουργικό σύστημα το οποίο βασίζεται στον πυρήνα Linux (Linux kernel). Αρχικά αναπτύχθηκε από μια startup εταιρία με το ίδιο όνομα, την Android, Inc. την οποία χρηματοδότησε και έπειτα εξαγόρασε η Google το 2005 σε μια προσπάθεια της να εισέλθει στον χώρο των κινητών συσκευών. Το λειτουργικό παρουσιάστηκε επίσημα το 2007 και το πρώτο κινητό βασισμένο στο λειτουργικό σύστημα Android πωλήθηκε τον Οκτώβρη του 2008.

Η διεπιφάνεια χρήστη (user interface) βασίζεται στον άμεσο χειρισμό (direct manipulation). Χρησιμοποιώντας σαν είσοδο την οθόνη αφής, αντιστοιχίζει ως γεγονότα εισόδου πραγματικά γεγονότα αφής όπως είναι το πέρασμα (swipe) του δαχτύλου, το κτύπημα (tap), το «τσίμπημα» (pinch) κ.α., με σκοπό τον χειρισμό αντικειμένων στην οθόνη. Εσωτερικό υλικό όπως επιταχυνσιόμετρα (accelerometers), γυροσκόπια (gyroscopes) και αισθητήρες χρησιμοποιούνται για να ανταποκριθούν σε πρόσθετες ενέργειες του χρήστη όπως η εναλλαγή τους προσανατολισμού της οθόνης.

Το Android είναι ένα ελεύθερο λειτουργικό ανοιχτού κώδικα (open source) και η Google κυκλοφορεί τον κώδικα του βάσει της Apache License (άδεια ελεύθερου λογισμικού). Αυτό σημαίνει ότι οποιοσδήποτε θέλει μπορεί να κατεβάσει (download) τον πλήρη πηγαίο κώδικα (source code) του Android, είτε είναι πωλητής, κατασκευαστής, ή προγραμματιστής. Οι περισσότεροι πωλητές (οι οποίοι συνήθως είναι και κατασκευαστές συσκευών και υλικού (hardware)) μπορούν να προσθέσουν τις δικές τους αποκλειστικές επεκτάσεις στο Android και να το τροποποιήσουν έτσι ώστε να διαφοροποιήσουν το προϊόν τους από άλλα. Αυτό το απλό μοντέλο ανάπτυξης και προγραμματισμού έχει κάνει το Android πολύ ελκυστικό, στρέφοντας πάνω του το ενδιαφέρον πολλών πωλητών και κατασκευαστών. Αυτό είχε ιδιαίτερο αντίκτυπο σε αρκετές εταιρίες οι οποίες είχαν επηρεαστεί από το φαινόμενο του iPhone της Apple, ένα ιδιαίτερα δημοφιλές προϊόν το οποίο άλλαξε άρδην τον χώρο της βιομηχανίας των «έξυπνων» τηλεφώνων. Εταιρίες όπως η Motorola και η Sony Ericsson, οι

οποίες για πάρα πολλά χρόνια είχαν αναπτύξει το δικό τους λειτουργικό σύστημα για κινητές συσκευές, μόλις κυκλοφόρησε το iPhone έπρεπε να προσπαθήσουν πάρα πολύ για να βρουν νέους τρόπους ώστε να ανανεώσουν τα προϊόντα τους. Αυτοί οι κατασκευαστές λοιπόν, βλέπουν το Android σαν μια λύση, καθώς μπορούν να συνεχίσουν να κατασκευάζουν κινητές συσκευές και να χρησιμοποιούν το Android για λειτουργικό σύστημα.

Το Android έχει και μια πολύ μεγάλη κοινότητα προγραμματιστών (developers) οι οποίοι δημιουργούν εφαρμογές (apps) οι οποίες επεκτείνουν την λειτουργικότητα των συσκευών. Τον Οκτώβρη του 2012 υπήρχαν 700.000 εφαρμογές διαθέσιμες και ο εκτιμώμενος αριθμός εφαρμογών που είχαν κατέβει από το Google Play, το οποίο είναι το κύριο «κατάστημα εφαρμογών» (app store) του Android, ήταν 25 δισεκατομμύρια. Μια έρευνα η οποία πραγματοποιήθηκε σε προγραμματιστές από τον Απρίλιο έως τον Μάιο του 2013 έδειξε ότι το Android είναι η πιο δημοφιλής πλατφόρμα για προγραμματιστές, καθώς χρησιμοποιείται από το 71% του συνόλου των προγραμματιστών για κινητές συσκευές.

Το Android είναι η πιο ευρέως χρησιμοποιούμενη πλατφόρμα για «έξυπνα» τηλέφωνα στον κόσμο, ξεπερνώντας το Symbian το τέταρτο τρίμηνο του 2010. Παρά το ότι αρχικά σχεδιάστηκε για χρήση με «έξυπνα» τηλέφωνα και ταμπλέτες, έχει επίσης χρησιμοποιηθεί σε τηλεοράσεις, κονσόλες παιχνιδιών, ψηφιακές κάμερες και άλλες ηλεκτρονικές συσκευές. Είναι αξιοσημείωτο ότι τον Σεπτέμβρη του 2013 είχαν ενεργοποιηθεί ένα δισεκατομμύριο συσκευές με Android και τον Νοέμβρη του 2013 το μερίδιο του Android στην παγκόσμια αγορά «έξυπνων» τηλεφώνων έφτασε το 80%.

2.2 Χαρακτηριστικά

Επειδή το Android είναι ανοιχτού κώδικα και διατίθεται ελεύθερα στους κατασκευαστές για προσαρμογές και τροποποιήσεις, δεν υπάρχουν σταθερά δομικά χαρακτηριστικά τόσο στο υλικό όσο και στο λογισμικό. Παρ' όλα αυτά, το Android από μόνο του διαθέτει και υποστηρίζει τα παρακάτω χαρακτηριστικά:

- **Επικοινωνία μέσω μηνυμάτων** (**Messaging**)
  Υποστηρίζει τόσο SMS όσο και MMS καθώς και Android Cloud To Device Messaging (C2DM) καθώς και μια ενισχυμένη έκδοση του C2DM, το Android Google Cloud Messaging (GCM).

- **Περιηγητής Διαδικτύου** (**Web Browser**)
  Ο περιηγητής διαδικτύου που είναι διαθέσιμος στο Android βασίζεται στη μηχανή διάταξης (layout engine) του Blink (πρώην WebKit) σε συνδυασμό με την V8 Javascript engine του Chrome.

- **Χαρακτηριστικά βασισμένα στην φωνή** (**Voice based features**)
  Η φωνητική αναζήτηση στο Google ήταν διαθέσιμη από την αρχική κυκλοφορία του Android. Φωνητικές εντολές για κλήσεις, μηνύματα, πλοήγηση κλπ υποστηρίζονται από

την έκδοση 2.2 και έπειτα. Από την έκδοση 4.1 και έπειτα η Google έχει επεκτείνει της φωνητικές εντολές, προσθέτοντας την δυνατότητα ομιλίας και ανάγνωσης απαντήσεων μέσα από το Knowledge Graph της Google όταν χρησιμοποιούνται κατάλληλες ερωτήσεις με συγκεκριμένες εντολές. Η δυνατότητα ελέγχου του υλικού μέσω της φωνής δεν έχει εφαρμοστεί ακόμα.

- **Πολυ-επαφή** (**Multi-touch**)
  Το Android έχει υπάρχουσα υποστήριξη για πολυ-επαφή η οποία έγινε αρχικά διαθέσιμη σε κινητές συσκευές όπως το HTC Hero. Το χαρακτηριστικό αυτό ήταν αρχικά απενεργοποιημένο σε επίπεδο πυρήνα.

- **Πολυδιεργασία** (**Multitaksing**)
  Η πολυδιεργασία εφαρμογών είναι διαθέσιμη σε συνδυασμό με ειδικό χειρισμό καταμερισμού μνήμης.

- **«Σύλληψη» Οθόνης** (**Screen Capture**)
  Η «σύλληψη» οθόνης είναι η αποτύπωση σε αρχείο της εικόνας της οθόνης όπως αυτή εμφανίζεται σε μια συγκεκριμένη χρονική στιγμή. Το Android παρέχει αυτή τη δυνατότητα με το ταυτόχρονο πάτημα του πλήκτρου ενεργοποίησης (power button) και του πλήκτρου μείωσης της έντασης (volume-down button). Πριν από την έκδοση 4.0 του Android ο μόνος τρόπος να κάνει ο χρήστης «σύλληψη» οθόνης, ήταν μέσω προσαρμογών του κατασκευαστή ή χρησιμοποιώντας σύνδεση με υπολογιστή και τα κατάλληλα εργαλεία (όπως το DDMS developer tool).

- **Video Κλήσεις** (**Video calling**)
  Το Android δεν υποστηρίζει προϋπάρχουσα δυνατότητα για video κλήσεις, αλλά κάποιες κινητές συσκευές έχουν μια τροποποιημένη έκδοση του λειτουργικού συστήματος που το επιτρέπει. Αυτό γίνεται είτε μέσω του δικτύου UMTS, είτε μέσω IP. Οι video κλήσεις μέσω τους Google Talk είναι διαθέσιμες από την έκδοση 2.3.4 και μετά. Εφαρμογές όπως το Skype από την έκδοση του Android 2.3 και μετά προσφέρουν video κλήσεις. Επίσης, οι χρήστες που διαθέτουν την εφαρμογή Google+ μπορούν να κάνουν video συνομιλίες μέσω της δυνατότητας hangout.

- **Πολλαπλή γλωσσική υποστήριξη** (**Multiple language support**)
  Το Android προσφέρει υποστήριξη και χρήση πολλαπλών διαφορετικών γλωσσών.

- **Προσβασιμότητα** (**Accessibility**)
  Το Android προσφέρει ενσωματομένη δυνατότητα μετατροπής κειμένου σε ήχο (text to speech) για ανθρώπους με μειωμένη ή καθόλου όραση. Ακόμα, είναι διαθέσιμες διάφορες προσθήκες και βοηθήματα για άτομα με προβλήματα ακοής.

- **Συνδεσιμότητα** (**Connectivity**)
  Το Android υποστηρίζει διάφορες τεχνολογίες συνδεσιμότητας όπως GSM/EDGE, Wi-Fi, Bluetooth, LTE, CDMA, EV-DO, UMTS, NFC, IDEN και WiMAX.

- **Bluetooth**
  Η υποστήριξη Bluetooth στο Android δίνει τη δυνατότητα φωνητικής κλήσης, αποστολής επαφών μέσω των τηλεφώνων, αποστολή αρχείων, πρόσβαση στον τηλεφωνικό κατάλογο κ.α.

- **Tethering**
  Μέσω της υποστήριξης tethering το Android δίνει τη δυνατότητα να χρησιμοποιηθεί ένα τηλέφωνο σαν ένα Wi-Fi hotspot. Πριν την έκδοση 2.2 αυτό ήταν δυνατό μόνο μέσω εφαρμογών ή μέσω τροποποιήσεων του κατασκευαστή.

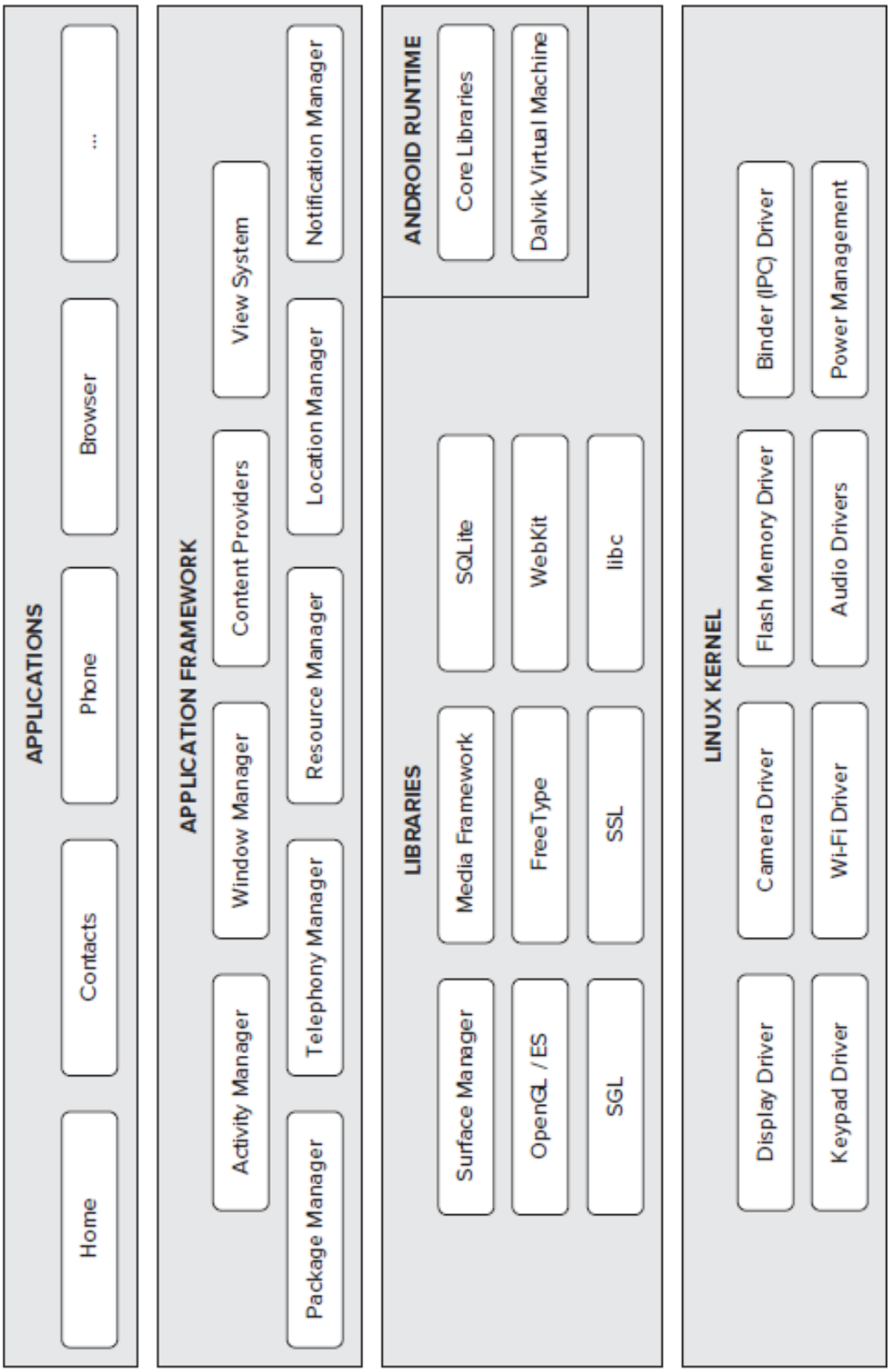
- **Υποστήριξη Media** (**Media Support)**
  Το Android υποστηρίζει τα εξής πρότυπα ήχου/video/εικόνας :
  WebM, H.263, H.264, AAC, HE-AAC (σε 3GP ή MP4 container), MPEG-4 SP, AMR, AMR-WB (σε 3GP container), MP3, MIDI, Ogg Vorbis, FLAC, WAV, JPEG, PNG, GIF, BMP, WebP.

- **Εσωτερική αποθήκευση** (**Storage**)
  Για την αποθήκευση δεδομένων. το Android χρησιμοποιεί την SQLite, μια ελαφριά σχεσιακή βάση δεδομένων.

- **Εξωτερική αποθήκευση** (**External storage**)
  Οι περισσότερες συσκευές με Android περιέχουν θύρες microSD οι οποίες μπορούν να διαβάσουν κάρτες microSD μορφοποιημένες με αρχεία συστήματος FAT32, Ext3 ή Ext4. Για να μπορέσουν να επιτρέψουν την χρήση μέσων αποθήκευσης υψηλής χωρητικότητας όπως USB flash drives και USB HDD, πολλές ταμπλέτες με Android περιέχουν έναν υποδοχέα USB ‘A’.

2.3 Αρχιτεκτονική

Για να καταλάβει κάποιος με ποιον ακριβώς τρόπο δουλεύει το Android θα ήταν καλό να κοιτάξει και να μελετήσει την **Εικόνα 2.5** στην οποία φαίνονται τα διαφορετικά στρώματα από τα οποία αποτελείται το λειτουργικό σύστημα του Android. Η εικόνα φαίνεται παρακάτω και έπειτα αναλύονται οι βασικοί τομείς στους οποίου χωρίζεται το λειτουργικό σύστημα.



Εικόνα 2.5 : Η Αρχιτεκτονική του Android

Όπως παρατηρείται, το λειτουργικό σύστημα του Android χωρίζεται σε 5 τομείς, σε 4 διαφορετικά στρώματα. Αναφέρονται συνοπτικά οι 5 βασικοί τομείς και η λειτουργία που επιτελεί ο καθένας.

- **Linux Kernel** (**Πυρήνας Linux**)
  Είναι ο πυρήνας στον οποίο βασίζεται το λειτουργικό σύστημα Android. Στο στρώμα αυτό περιέχονται όλοι οι χαμηλού επιπέδου οδηγοί της συσκευής (low level device drivers) που χρειάζονται για τα διάφορα υλικά μέρη της συσκευής.

- **Βιβλιοθήκες** (**Libraries**)
  Οι βιβλιοθήκες περιέχουν όλο τον κώδικα ο οποίος παρέχει τα βασικά χαρακτηριστικά ενός λειτουργικού συστήματος Android (όπως αυτά αναφέρθηκαν στην παράγραφο 2.2).

- **Android Runtime**
  Στο ίδιο στρώμα με τις βιβλιοθήκες, ο τομέας Android runtime παρέχει ένα σύνολο από βασικές βιβλιοθήκες που δίνουν τη δυνατότητα στους προγραμματιστές να γράψουν εφαρμογές για το Android σε γλώσσα προγραμματισμού Java. Ακόμα, περιέχει την εικονική μηχανή (virtual machine) Dalvik η οποία επιτρέπει σε κάθε εφαρμογή να τρέξει μέσα σε δική της διαδικασία, με δική της ξεχωριστή εικονική μηχανή Dalvik. Η Dalvik είναι μια ειδική εικονική μηχανή, ειδικά σχεδιασμένη για Android και βελτιστοποιημένη για χρήση με κινητές συσκευές που λειτουργούν με μπαταρία και διαθέτουν περιορισμένες δυνατότητες σε μνήμη και CPU.

- **Πλαίσιο Εφαρμογών** (**Application Framework**)
  Παραθέτει τις διάφορες δυνατότητες του λειτουργικού συστήματος στους προγραμματιστές εφαρμογών ώστε να μπορέσουν να τις χρησιμοποιήσουν στις εφαρμογές τους.

- **Εφαρμογές** (**Applications**)
  Σε αυτό το υψηλότερο στρώμα βρίσκονται τόσο οι εφαρμογές οι οποίες διατίθενται μαζί με την συσκευή (όπως οι Επαφές (Contacts), το Τηλέφωνο (Phone), ο Περιηγητής (Browser) κλπ) όσο και οι εφαρμογές που μπορεί να κατεβάσει ο χρήστης από το κεντρικό «κατάστημα εφαρμογών», το Google Play.

2.4 Εκδόσεις του Android

Το Android από τότε που κυκλοφόρησε μέχρι και σήμερα έχει περάσει από πολλές αλλαγές και ανανεώσεις. Τον Σεπτέμβρη του 2008 κυκλοφόρησε η πρώτη έκδοση με όνομα Android 1.0. Από τον Απρίλη του 2009 και μετά όμως, κάθε διαφορετική έκδοση είχε και μια κωδική ονομασία και πάντα ακολουθούσε αλφαβητική σειρά. Στον **Πίνακα 2.1** εμφανίζονται όλες οι βασικές εκδόσεις μέχρι σήμερα, μαζί με την ημερομηνία κυκλοφορίας και την κωδική τους

ονομασία. Στον **Πίνακα 2.2** φαίνονται αναλυτικά όλες τις εκδόσεις, μαζί με το επίπεδο του API (application programming interface) τους.

ΕΚΔΟΣΗ ANDROID	ΗΜΕΡΟΜΗΝΙΑ ΚΥΚΛΟΦΟΡΙΑΣ	ΚΩΔΙΚΗ ΟΝΟΜΑΣΙΑ
1.0	23-Σεπ-08	
1.1	9-Φεβ-09	
1.5	30-Απρ-09	Cupcake
1.6	15-Σεπ-09	Donut
2.0/2.1	26-Οκτ-09	Eclair
2.2	20-Μαϊ-10	Froyo
2.3	6-Δεκ-10	Gingerbread
3.0/3.1/3.2	22-Φεβ-11	Honeycomb
4.0	19-Οκτ-11	Ice Cream Sandwich
4.1/4.2/4.3	9-Ιουλ-12	Jelly Bean
4.4	31-Οκτ-13	KitKat

Πίνακας 2.1 : Οι εκδόσεις του Android

Android 1.0 (API level 1)
Android 1.1 (API level 2)
Android 1.5 Cupcake (API level 3)
Android 1.6 Donut (API level 4)
Android 2.0 Eclair (API level 5)
Android 2.0.1 Eclair (API level 6)
Android 2.1 Eclair (API level 7)
Android 2.2–2.2.3 Froyo (API level 8)
Android 2.3–2.3.2 Gingerbread (API level 9)
Android 2.3.3–2.3.7 Gingerbread (API level 10)
Android 3.0 Honeycomb (API level 11)
Android 3.1 Honeycomb (API level 12)
Android 3.2 Honeycomb (API level 13)
Android 4.0–4.0.2 Ice Cream Sandwich (API level 14)
Android 4.0.3–4.0.4 Ice Cream Sandwich (API level 15)
Android 4.1 Jelly Bean (API level 16)
Android 4.2 Jelly Bean (API level 17)
Android 4.3 Jelly Bean (API level 18)
Android 4.4 KitKat (API level 19)

Πίνακας 2.2 : Αναλυτικά οι εκδόσεις του Android με βάση το API τους

3. Προγραμματισμός και ανάπτυξη εφαρμογών σε περιβάλλον Android

3.1 Προγραμματιστικά Εργαλεία

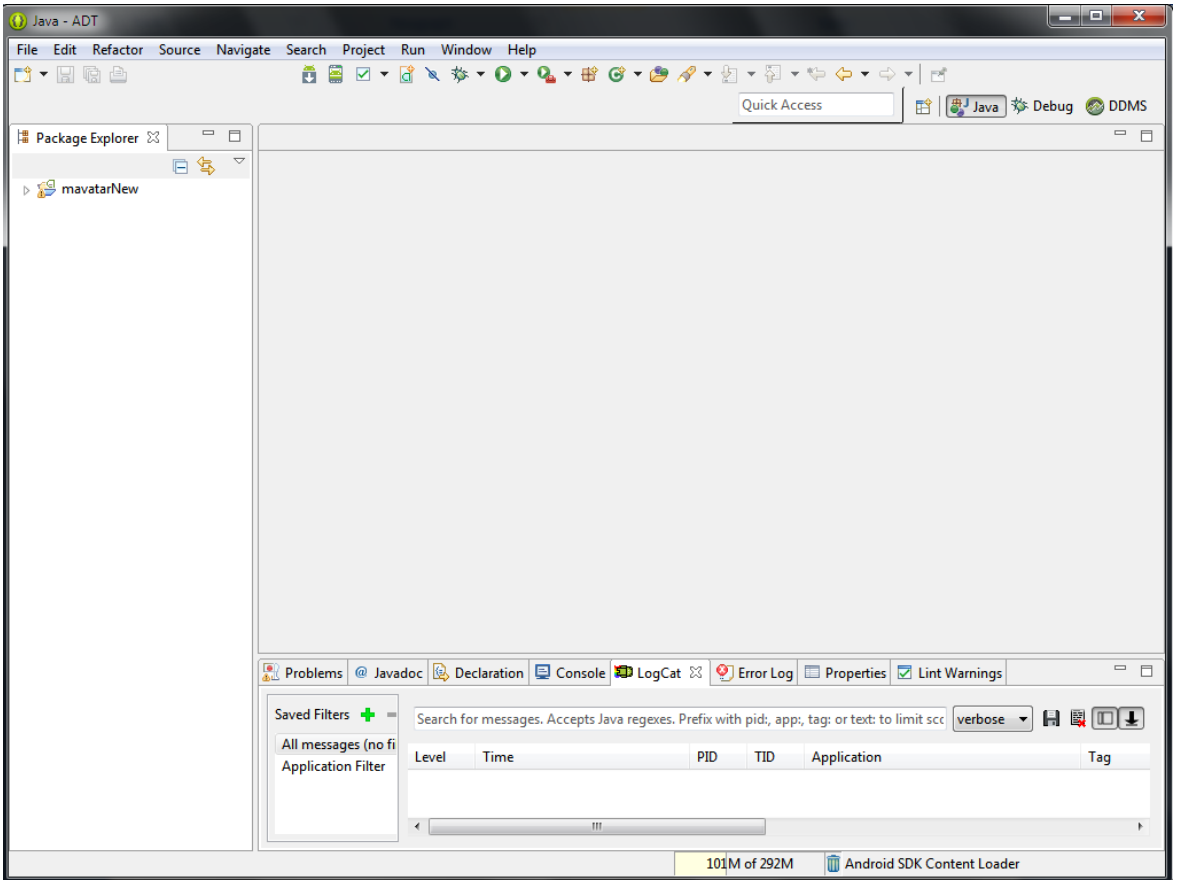
Η κύρια γλώσσα προγραμματισμού που χρησιμοποιείται στην ανάπτυξη εφαρμογών σε Android είναι η Java. Θα πρέπει λοιπόν να υπάρχει στον υπολογιστή το Java Development Kit (JDK) το οποίο και περιέχει όλα τα απαραίτητα εργαλεία της γλώσσας. Επίσης, θα χρειαστεί ένα ολοκληρωμένο περιβάλλον ανάπτυξης (IDE), στο οποίο θα γράφονται, θα μεταγλωττίζονται και θα εκτελούνται τα προγράμματα. Ένα τέτοιο IDE είναι το Eclipse (**Εικόνα 3.1**), το οποίο υποστηρίζεται επίσημα και είναι αυτό το οποίο χρησιμοποιήθηκε στην ανάπτυξη της παρούσας εφαρμογής.

Για να μπορέσει τώρα κάποιος να γράψει κώδικα συγκεκριμένα για εφαρμογές σε περιβάλλον Android, το πιο βασικό εργαλείο που θα χρειαστεί είναι το Android SDK (software development kit). Το επόμενο βήμα είναι να συνδέσει το Android SDK με το IDE. Αυτό επιτυγχάνεται μέσω του ADT (Android Development Tools) Plugin, το οποίο είναι ουσιαστικά μια επέκταση η οποία εγκαθίσταται στο Eclipse.

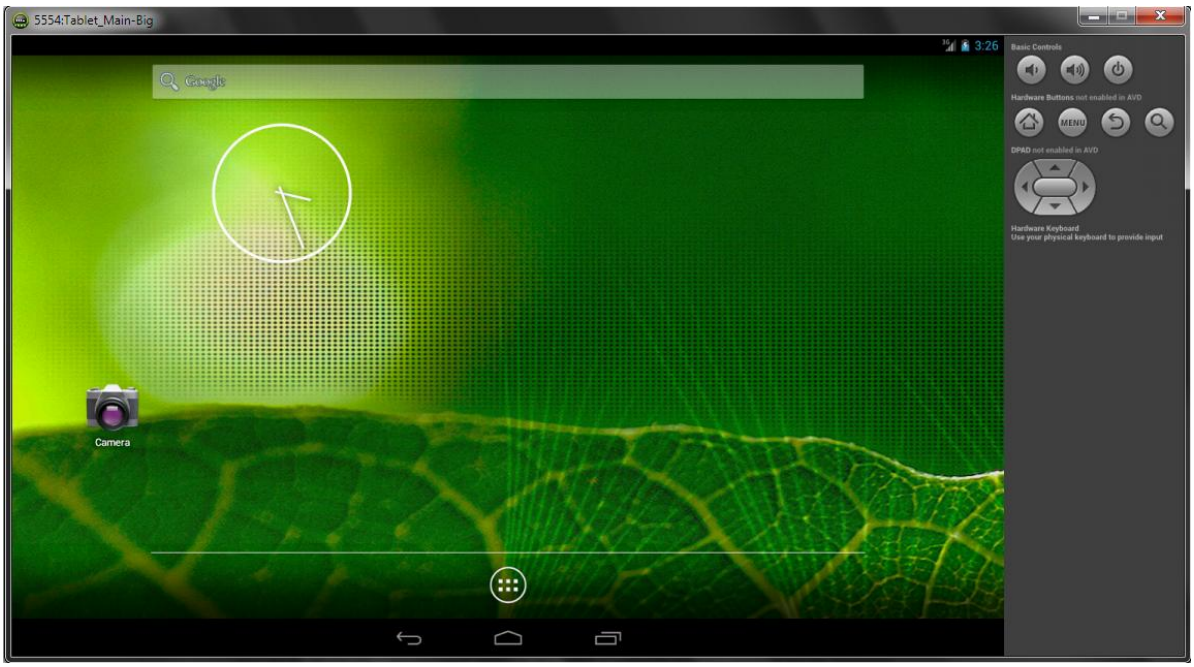
Όπως αναφέρθηκε στην παράγραφο 2.4, το Android κυκλοφορεί σε πολλές διαφορετικές εκδόσεις. Επειδή κάθε έκδοση είναι διαφορετική από τις άλλες, χρειάζεται τα δικά της προγραμματιστικά εργαλεία. Για να μπορέσει λοιπόν ο ενδιαφερόμενος να έχει τη δυνατότητα να προγραμματίσει για τις εκδόσεις που θέλει, θα πρέπει να χρησιμοποιήσει το ADT Plugin, μέσω του οποίου μπορεί να κατεβάσει και να εγκαταστήσει όλα τα απαραίτητα εργαλεία για κάθε έκδοση. Έτσι, αν θέλει μια εφαρμογή να είναι συμβατή με πολλές διαφορετικές εκδόσεις (συνήθως παλαιότερες), αρκεί να εγκαταστήσει τα κατάλληλα εργαλεία από το ADT και να δοκιμάσει την εφαρμογή στις εκδόσεις αυτές.

Επειδή όμως όλη η διαδικασία ανάπτυξης γίνεται μέσω υπολογιστή, ο οποίος και χρησιμοποιεί κάποιο λειτουργικό σύστημα (Windows, Linux, MacOS) το οποίο φυσικά και δεν είναι το Android, θα χρειαστεί ένα τρόπο για να μπορέσει να τρέξει και να «δει» την εφαρμογή σε χρήση. Τη λύση δίνει μια κατάλληλα κατασκευασμένη εικονική συσκευή Android (Android Virtual Device - AVD) η οποία ουσιαστικά προσομοιώνει ένα κινητό τηλέφωνο ή μια ταμπλέτα με λειτουργικό σύστημα Android, τόσο σε επίπεδο υλικού όσο και λογισμικού (**Εικόνα 3.2**). Η συσκευή αυτή εγκαθίσταται μέσω του ADT Plugin και είναι παραμετροποιήσιμη. Μπορεί να ρυθμιστεί η έκδοση Android που χρησιμοποιεί, το μέγεθος της RAM, το μέγεθος της κάρτας SD, την δυνατότητα για χρήση κάμερας κλπ. Αξίζει να αναφερθεί ότι ο προσομοιωτής αντλεί πάρα πολλούς υπολογιστικούς πόρους με αποτέλεσμα να καθιστά κάποιες φορές αργή την εκτέλεση εφαρμογών, ειδικά αν ο χρήστης δεν διαθέτει ισχυρό υπολογιστικό σύστημα. Σε κάθε περίπτωση, ο χρήστης μπορεί να εξάγει το εκτελέσιμο αρχείο της εφαρμογής και να το εγκαταστήσει σε κάποια πραγματική κινητή συσκευή με Android και να την τρέξει εκεί.

Όλα τα παραπάνω προγραμματιστικά εργαλεία διατίθενται ελεύθερα στο διαδίκτυο από τους κατασκευαστές τους και μπορεί ο καθένας να τα κατεβάσει και να τα χρησιμοποιήσει χωρίς κόστος.



Εικόνα 3.1 : Το περιβάλλον ανάπτυξης λογισμικού Eclipse



Εικόνα 3.2 : Εικονική συσκευή με λειτουργικό σύστημα Android (AVD). Προσομοίωση ταμπλέτας μεγέθους 10.1".

3.2 Ανάλυση βασικών API για την ανάπτυξη μιας εφαρμογής σε Android

Η ανάπτυξη μιας εφαρμογής σε περιβάλλον Android μπορεί να χωριστεί σε πολλά διαφορετικά τμήματα ανάλογα με το ποιο κομμάτι της εφαρμογής καλείται ο προγραμματιστής να υλοποιήσει κάθε φορά. Από τα βασικά δομικά στοιχεία μιας εφαρμογής στο προγραμματιστικό επίπεδο, το βασικό σχεδιασμό της διάταξης που θα βλέπει και θα χειρίζεται ο χρήστης, μέχρι τα περιφερειακά συστήματα της συσκευής στα οποία θα έχει πρόσβαση η εφαρμογή, ο προγραμματιστής θα κληθεί να ανταποκριθεί στα προβλήματα που θα προκύψουν σε κάθε κομμάτι.

Ονομαστικά, αυτή τη διαδικασία θα μπορούσαμε να χωριστεί στα παρακάτω τμήματα, καθένα από τα οποία μπορεί να αναλυθεί στις δικές του ξεχωριστές κατηγορίες:

- **App Components**
- **User Interface**
- **App Resources**
- **Animation and Graphics**
- **Computation**
- **Media and Camera**
- **Location and Sensors**
- **Connectivity**
- **Text and Input**
- **Data Storage**

Στα πλαίσια αυτής της διπλωματικής εργασίας δεν θα ήταν δυνατό αλλά και δε κρίνεται απαραίτητο να αναλυθούν όλα τα παραπάνω. Θα αναφερθούν τα πιο βασικά από αυτά και θα αναλυθούν ιδιαίτερα τα τμήματα που έπαιξαν καθοριστικό ρόλο στην ανάπτυξη της συγκεκριμένης εφαρμογής.

3.2.1 Συστατικά στοιχεία της εφαρμογής (App Components)

Τα συστατική στοιχεία μιας εφαρμογής είναι τα δομικά εκείνα μέρη που είναι απαραίτητα για την κατασκευή της. Κάθε στοιχείο είναι και ένας διαφορετικός τρόπος με τον οποίο το λειτουργικό μπορεί να εισέλθει στην εφαρμογή. Τα στοιχεία αυτά μπορούν να διακριθούν σε τέσσερα βασικά είδη. Κάθε είδος επιτελεί τον δικό του συγκεκριμένο ρόλο και έχει ένα ξεχωριστό κύκλο ζωής σύμφωνα με τον οποίο δημιουργείται και καταστρέφεται.

Τα τέσσερα αυτά είδη είναι:

- **Activities**
- **Services**
- **Content Providers**
- **Broadcast Receivers**

3.2.1.1 Activities

Οι activities αποτελούν το βασικότερο στοιχείο μιας εφαρμογής. Κάθε activity ουσιαστικά αντιπροσωπεύει μια οθόνη μαζί με την διεπιφάνεια χρήστη. Μια εφαρμογή συνήθως αποτελείται από πολλές activities οι οποίες είναι χαλαρά δεμένες μεταξύ τους. Συνηθέστερα, μια activity αποτελεί την main (βασική) activity η οποία είναι και η οθόνη που βλέπει ο χρήστης όταν ανοίγει την εφαρμογή για πρώτη φορά. Κάθε activity μπορεί να καλέσει μια νέα activity για να εκτελέσει διαφορετικές λειτουργίες. Κάθε φορά που μια νέα activity ξεκινάει, η προηγούμενη activity σταματάει. Το σύστημα όμως την διατηρεί και την αποθηκεύει σε μια στοίβα (LIFO), το λεγόμενο "back stack". Όταν λοιπόν μια νέα activity ξεκινάει, τότε εμφανίζεται στην οθόνη και αποθηκεύεται τελευταία στο back stack. Έτσι, όταν ο χρήστης τελειώσει με τη χρήση της και πατήσει το πλήκτρο Back, τότε αυτή γίνεται pop από την στοίβα, καταστρέφεται και η οθόνη επανέρχεται στην προηγούμενη activity. Αν πατηθεί το πλήκτρο Back ενώ ο χρήστης βρίσκεται στην main activity τότε εξέρχεται από την εφαρμογή.

Για να κατασκευαστεί μια activity, πρέπει να δημιουργηθεί μια υποκλάση της βασικής κλάσης Activity. Σε αυτή την υποκλάση πρέπει να εφαρμοστούν συγκεκριμένες μεθόδοι τις οποίες καλεί το σύστημα όταν η activity μεταβαίνει σε διαφορετικά στάδια του κύκλου ζωής της. Τέτοια στάδια είναι όταν η activity δημιουργείται (create), σταματάει (stop), επανέρχεται (resume) ή καταστρέφεται (destroy).

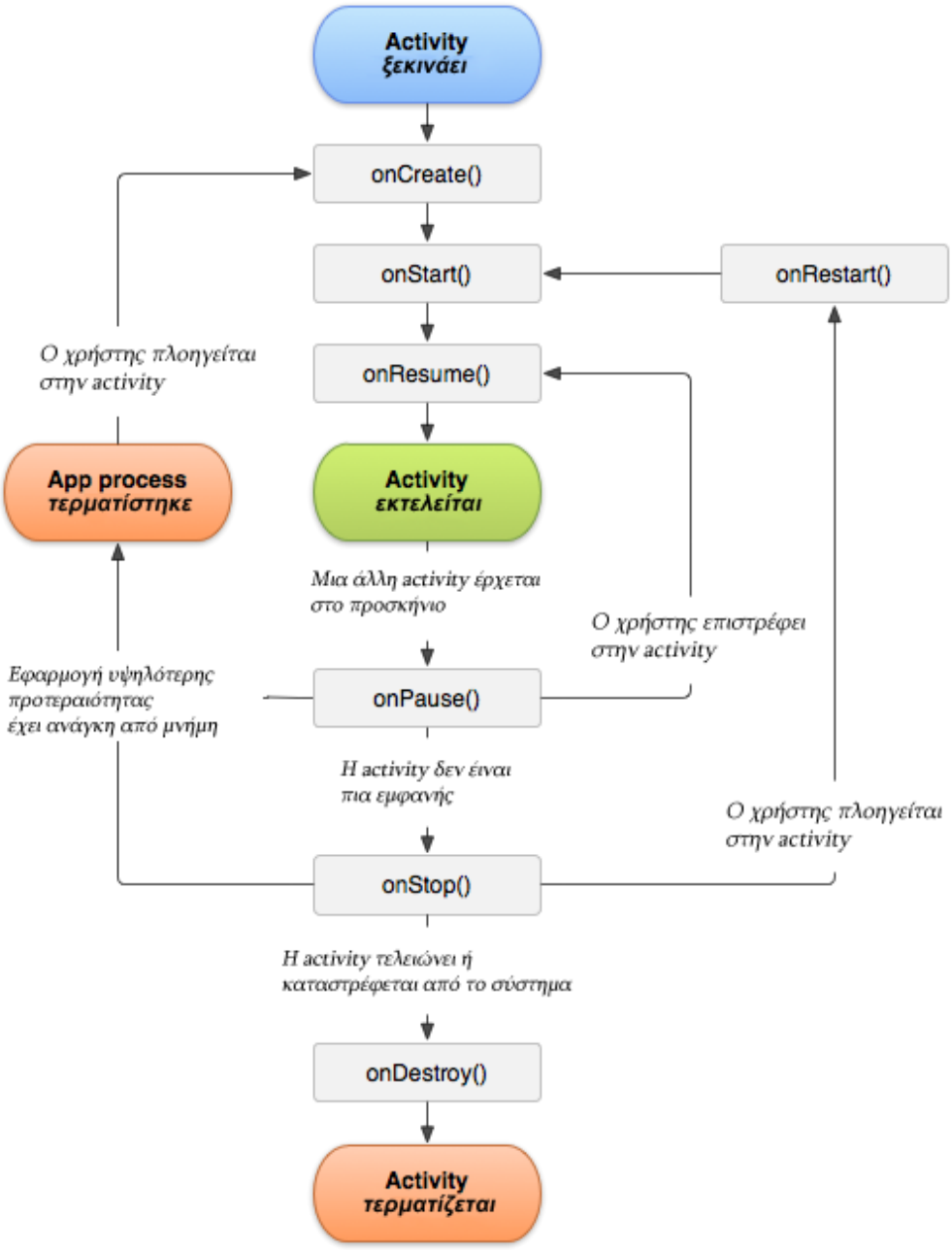
Οι δύο πιο βασικές τέτοιες μέθοδοι είναι:

- **onCreate( )**
  Πρέπει αναγκαστικά να εφαρμοστεί αυτή η μέθοδος. Το σύστημα την καλεί όταν δημιουργεί την activity. Εδώ πρέπει να αρχικοποιηθούν τα απαραίτητα στοιχεία της activity. Το πιο σημαντικό είναι να γίνει κλήση σε αυτό το σημείο στην μέθοδο setContentView( ) με την οποία ορίζεται η διάταξη (layout) της διεπιφάνειας χρήστη (user interface) αυτής της activity.

- **onPause( )**
  Το σύστημα καλεί αυτή τη μέθοδο όταν έχει ενδείξεις ότι ο χρήστης φεύγει από την activity (χωρίς να σημαίνει η activity καταστρέφεται). Εδώ συνήθως ο προγραμματιστής πρέπει να κάνει όποιες αλλαγές θα χρειαστεί να παραμείνουν και μετά το πέρας αυτής της περιόδου γιατί ο χρήστης μπορεί να μην επιστρέψει.

Υπάρχουν και άλλες βασικές μέθοδοι πέρα από αυτές, οι οποίες χρησιμοποιούνται από το σύστημα και που ο προγραμματιστής θα πρέπει να χρησιμοποιεί ώστε να προσφέρει στον χρήστη μια ομαλή εμπειρία χρήσης. Ο προγραμματιστής θα πρέπει να έχει χρησιμοποιήσει όλες τις κατάλληλες μεθόδους ώστε καθώς ο χρήστης θα πλοηγείται ανάμεσα στις διάφορες activities, η εφαρμογή να μπορεί να χειριστεί απρόσμενες διακοπές οι οποίες θα αναγκάζουν

τα activities να σταματούν ή ακόμα και να καταστρέφονται. Ο βασικός κύκλος ζωής μιας activity φαίνεται στην **Εικόνα 3.3**.



Εικόνα 3.3 : Ο κύκλος ζωής μιας Activity

3.2.1.1.1 Fragments

Τα Fragments παρά το ότι είναι ένα εξειδικευμένο στοιχείο στην ανάπτυξη εφαρμογών για Android, είναι αναγκαίο να αναφερθούν στην παρούσα διπλωματική διότι έχουν χρησιμοποιηθεί κατά κόρον στην παρούσα εφαρμογή. Επίσης, είναι ένα αρκετά νέο στοιχείο στον κόσμο του Android καθ' ότι έγινε διαθέσιμο από το API 11 και μετά (Πίνακα 2.2), δηλαδή από την έκδοση 3.0. Από τότε όμως χρησιμοποιείται ευρύτατα από τους προγραμματιστές μιας και είναι πάρα πολύ χρήσιμο και ευέλικτο.

Τα Fragments αναπαριστούν τη συμπεριφορά ή ένα μέρος της διεπιφάνειας χρήστη μέσα σε μια Activity. Μπορούν να συνδυαστούν πολλά fragments σε μια μόνο activity ώστε να κατασκευαστεί ένα UI με πολλά παράθυρα και επίσης μπορεί να επαναχρησιμοποιηθεί το ίδιο fragment σε πολλές activities. Ουσιαστικά, ένα fragment είναι κατασκευαστικό κομμάτι μιας activity, με το δικό του κύκλο ζωής, την δική του είσοδο δεδομένων και το οποίο μπορεί να προστεθεί ή να αφαιρεθεί στη διάρκεια που μια εφαρμογή εκτελείται.

Ένα fragment πρέπει πάντα να είναι ενσωματωμένο μέσα σε μια activity και ο κύκλος ζωής του επηρεάζεται άμεσα από τον κύκλο ζωής αυτής της activity. Για παράδειγμα, όταν η activity σταματάει, το ίδιο συμβαίνει και σε όλα τα fragments μέσα σε αυτή. Όταν η activity καταστρέφεται, καταστρέφονται και όλα τα fragments σε αυτήν. Όμως, το σημαντικό κομμάτι είναι ότι, καθώς η activity εκτελείται, ο προγραμματιστής μπορεί να χειριστεί κάθε fragment ξεχωριστά, να προσθέσει ή να αφαιρέσει διαφορετικά fragments, ακόμα και να αντικαταστήσει fragments με άλλα fragments – αυτό ονομάζεται fragment transaction. Όπως και με τις activities, έτσι και τα fragments, μπορούν να τα προστεθούν στο back stack. Το back stack αυτό χρησιμοποιείται από την activity και δίνει τη δυνατότητα στο χρήστη να αντιστρέψει μια fragment transaction, να πλοηγηθεί δηλαδή προς τα πίσω, με τη χρήση του πλήκτρου Back.

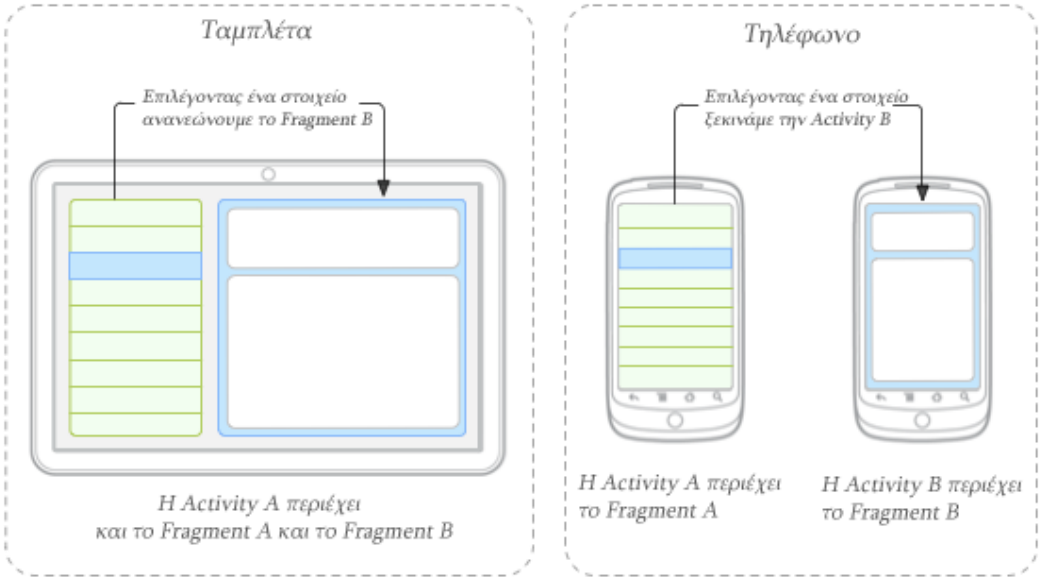
Σχεδιαστική Φιλοσοφία:

Το Android εισήγαγε τα Fragments κυρίως για να υποστηρίξει πιο δυναμικά και ευέλικτα UI σε μεγάλες οθόνες. Επειδή η οθόνη μιας ταμπλέτας είναι αρκετά πιο μεγάλη από αυτή ενός τηλεφώνου, υπάρχει περισσότερος χώρος ώστε να συνδυαστούν περισσότερα γραφικά στοιχεία. Τα Fragments επιτρέπουν αυτόν τον σχεδιασμό χωρίς να πρέπει να μπει ο προγραμματιστής σε πολύπλοκες αλλαγές. Χωρίζοντας την διάταξη (layout) μιας activity σε fragments, μπορεί να αλλάξει την εμφάνιση μιας activity ενώ αυτή εκτελείται.

Για να γίνει αυτό πιο κατανοητό, θα αναφερθεί ένα παράδειγμα. Μια ειδησεογραφική εφαρμογή μπορεί να χρησιμοποιήσει ένα fragment για να παρουσιάσει μια λίστα με άρθρα στα αριστερά και ένα άλλο fragment για να δείξει αυτό το άρθρο στα δεξιά. Και τα δύο αυτά fragments είναι εμφανή μέσα στην ίδια activity, το ένα δίπλα στο άλλο, αλλά το καθένα από αυτά έχει το δικό του κύκλο ζωής και χειρίζεται μόνο του τα δεδομένα εισόδου που το αφορούν. Έτσι, αντί ο χρήστης να χρησιμοποιήσει δύο activities, μία για τη λίστα με τα άρθρα

και μια άλλη για την εμφάνιση του άρθρου, μπορεί να κάνει και τα δύο μαζί, στην ίδια activity (και κατ επέκταση στην ίδια οθόνη).

Αυτό έχει ακόμα περισσότερες προεκτάσεις και εφαρμογές όταν μια εφαρμογή πρέπει να είναι διαθέσιμη για διαφορετικά μεγέθη οθονών και κυρίως να είναι διαθέσιμη τόσο για ταμπλέτες όσο και για τηλέφωνα. Συνεχίζοντας με το παραπάνω παράδειγμα, όταν η εφαρμογή τρέχει σε ταμπλέτα, μπορεί να χρησιμοποιεί μια Activity A η οποία να περιέχει και τα δύο fragments. Όταν όμως η εφαρμογή τρέχει σε κινητό, δεν υπάρχει αρκετός χώρος για να παρουσιαστούν και τα δύο fragments το ένα δίπλα στο άλλο. Έτσι, θα χρειαστεί να υπάρχουν δύο activities. Η Activity A θα έχει μόνο το ένα fragment και όταν ο χρήστης επιλέξει το άρθρο που θέλει να διαβάσει, θα καλεί την Activity B η οποία περιέχει το άλλο fragment, ώστε ο χρήστης να διαβάσει το άρθρο. Όλα τα παραπάνω φαίνονται καλύτερα στην **Εικόνα 3.4**.



Εικόνα 3.4 : Ένα παράδειγμα που δείχνει πως δύο γραφικά στοιχεία που ορίζονται από Fragments μπορούν να συνδυαστούν σε μια activity για σχεδιασμό σε ταμπλέτα και να χωριστούν για σχεδιασμό σε τηλέφωνο.

Τα fragments, όπως και οι activities, για να κατασκευαστούν θα πρέπει να δημιουργηθεί μια υποκλάση της βασικής κλάσης Fragment και να εφαρμοστούν οι κατάλληλες μέθοδοι. Ο κώδικας αυτός μοιάζει πάρα πολύ με τον κώδικα των activities και περιέχει παρόμοιες μεθόδους, όπως είναι οι onCreate( ), onStart( ), onPause( ), onStop( ).

Επίσης, υπάρχουν υποκλάσεις οι οποίες μπορούν να επεκταθούν αντί της βασικής κλάσης Fragment. Τέτοιες είναι οι DialogFragment, ListFragment, PreferenceFragment.

Στην παρούσα εφαρμογή γίνεται χρήση της κλάσης DialogFragment η οποία παρουσιάζει στον χρήστη ένα παράθυρο – διάλογο (παράγραφος 3.2.2.3).

3.2.1.2 Services

Ένα service είναι ένα στοιχείο το οποίο τρέχει στο παρασκήνιο (background) και εκτελεί χρονοβόρες διαδικασίες. Ένα service δεν παρέχει διεπιφάνεια χρήστη (user interface). Για παράδειγμα, ένα service μπορεί να παίζει μουσική στο παρασκήνιο, ενώ ο χρήστης βρίσκεται σε μια άλλη εφαρμογή. Ένα service εφαρμόζεται ως υποκλάση της βασικής κλάσης Service.

3.2.1.3 Content Providers

Ένας content provider διαχειρίζεται ένα μοιραζόμενο σύνολο από δεδομένα εφαρμογών. Μέσω αυτών μπορούν να αποθηκευθούν δεδομένα στο σύστημα αρχείων, σε μια βάση δεδομένων SQLite, στο δίκτυο, ή σε οποιαδήποτε άλλο αποθηκευτικό χώρο έχει πρόσβαση η εφαρμογή.

3.2.1.4 Broadcast Receivers

Ένας broadcast receiver είναι ένα στοιχείο το οποίο αποκρίνεται σε γεγονότα τα οποία «ανακοινώνονται» από το ίδιο το λειτουργικό σύστημα. Τέτοια γεγονότα μπορεί να είναι το ότι έσβησε η οθόνη, το ότι η στάθμη της μπαταρίας είναι χαμηλή ή ότι ο χρήστης τράβηξε μια φωτογραφία. Για παράδειγμα, αρκετές εφαρμογές λήψης φωτογραφιών που χρησιμοποιούν flash, όταν η στάθμη της μπαταρίας είναι χαμηλή το απενεργοποιούν για να εξοικονομήσουν ενέργεια. Ενημερώνονται δηλαδή μέσω ενός broadcast receiver για το γεγονός και αντιδρούν ανάλογα.

3.2.1.5 Processes and Threads (Διαδικασίες και Νήματα)

Όταν ένα συστατικό στοιχείο της εφαρμογής ξεκινάει να εκτελείται και η εφαρμογή δεν έχει κάποιο άλλο στοιχείο να τρέχει, τότε το Android ξεκινάει μια νέα Linux process (διαδικασία) για την εφαρμογή, με ένα μόνο thread (νήμα) εκτέλεσης. Όλα τα στοιχεία της ίδιας εφαρμογής τρέχουν στην ίδια διαδικασία και στο ίδιο νήμα. Αν δηλαδή ένα στοιχείο μιας εφαρμογής ξεκινήσει να εκτελείται και υπάρχει ήδη μια διαδικασία για αυτή την εφαρμογή, τότε το νέο στοιχείο ξεκινάει να εκτελείται μέσα σε αυτή την διαδικασία και χρησιμοποιεί το ίδιο thread εκτέλεσης. Παρ’ όλα αυτά, ο προγραμματιστής μπορεί να επέμβει έτσι ώστε διαφορετικά στοιχεία να τρέχουν σε διαφορετικές διαδικασίες και να δημιουργήσει νέα thread για κάθε διαδικασία.

Δεν θα αναλυθούν παραπάνω όλα τα περιεχόμενα αυτής της κατηγορίας, αλλά θα γίνει αναφορά σε μια συγκεκριμένη λειτουργία η οποία και χρησιμοποιείται στην εφαρμογή που αναπτύχθηκε στα πλαίσια της διπλωματικής και παίζει σημαντικό ρόλο στη λειτουργία της.

Η λειτουργία αυτή ονομάζεται **AsyncTask** και επιτρέπει στον προγραμματιστεί να εκτελεί ασύγχρονες λειτουργίες στο user interface (Παράγραφος 3.2.2). Επειδή το Android χρησιμοποιεί μόνο ένα thread όπως περιγράφηκε παραπάνω, είναι ζωτικής σημασίας για την απόκριση και τη λειτουργία των γραφικών στοιχείων της εφαρμογής (UI) να μην μπλοκάρεται ποτέ το UI thread. Αν πρέπει να εκτελεστούν λειτουργίες οι οποίες δεν είναι στιγμιαίες αλλά είναι χρονοβόρες, τότε θα πρέπει να εκτελεστούν σε ξεχωριστό thread (ονομαζόμενο και "worker-thread" (νήμα εργασίας)).

Η λειτουργία AsyncTask λοιπόν, αναλαμβάνει να εκτελέσει όλες τις χρονοβόρες λειτουργίες (Task) σε ένα worker thread και, μόλις αυτές εκτελεστούν, να ενημερώσει το UI thread για τα αποτελέσματα. Έτσι, σε όλη τη διάρκεια εκτέλεσης τους, το UI thread παραμένει ανεπηρέαστο, δεν μπλοκάρεται, και ο χρήστης μπορεί να συνεχίσει να χρησιμοποιεί την εφαρμογή καθώς οι λειτουργίες αυτές εκτελούνται πλέον σε άλλο thread.

Για να χρησιμοποιηθεί το AsyncTask πρέπει να δημιουργηθεί μια υποκλάση της κλάσης AsyncTask και να εφαρμοστούν κάποιες βασικές μέθοδοι. Οι μέθοδοι αυτές είναι:

- **onPreExecute( )** – καλείται στο UI thread πριν την εκτέλεση της λειτουργίας (Task). Συνήθως χρησιμοποιείται για να αρχικοποιήσει την λειτουργία, για παράδειγμα δείχνοντας μια μπάρα προόδου στο user interface.

- **doInBackground(Params…)** – Η πιο σημαντική μέθοδος καθώς είναι η μέθοδος η οποία αναλαμβάνει την εκτέλεση της λειτουργίας. Καλείται αμέσως μετά το πέρας της onPreExecute( ). Μέσω αυτής περνιούνται οι παράμετροι που θα χρειαστούν για τη λειτουργία. Επίσης, είναι και η μέθοδος που είναι υπεύθυνη για την ανακοίνωση των αποτελεσμάτων της λειτουργίας. Ακόμα, μπορεί να ενημερώνει κατά τη διάρκεια της εκτέλεσης για την πρόοδο της λειτουργίας μέσω της μεθόδου publishProgress(Progress…)

- **onProgressUpdate(Progress…)** – Αυτή η μέθοδος καλείται μόνο μέσω της publishProgress(Progress…) και μπορεί να εκτελεστεί όσο ακόμα η λειτουργία είναι σε εξέλιξη. Χρησιμοποιείται συνήθως για να ενημερώνει και να απεικονίζει την πρόοδο της λειτουργίας σε μια μπάρα προόδου.

- **onPostExecute(Result) –** καλείται από το UI thread μετά το τέλος της λειτουργίας. Το αποτέλεσμα της λειτουργίας περνάει σε αυτήν ως παράμετρος.

3.2.1.6 Android Manifest

Κάθε εφαρμογή πρέπει να έχει ένα αρχείο **AndroidManifest.xml** (ακριβώς με αυτό το όνομα) στο βασικό φάκελο του project. Το manifest παρέχει στο λειτουργικό σύστημα όλες τις απαραίτητες πληροφορίες χωρίς τις οποίες το σύστημα δεν μπορεί να εκτελέσει τον κώδικα της εφαρμογής. Οι σημαντικότερες πληροφορίες που παρέχει είναι:

- Η ονομασία του πακέτου Java της εφαρμογής
- Την έκδοση της εφαρμογής (1.0, 2.1 κλπ)
- Την ελάχιστη έκδοση του λειτουργικού συστήματος (min sdk version) που απαιτεί η εφαρμογή για να λειτουργήσει. Για παράδειγμα, στην παρούσα εφαρμογή, έχει δηλωθεί min sdk version = 14 το οποίο αντιστοιχεί στην έκδοση 4.0
- Την στοχευόμενη έκδοση του λειτουργικού συστήματος (target sdk version). Αυτό σημαίνει ότι ενώ μπορεί να δουλέψει και σε παλαιότερες εκδόσεις, έχει τεσταριστεί λεπτομερώς με την έκδοση που αναφέρεται εδώ. Για παράδειγμα, στην παρούσα εφαρμογή, έχει δηλωθεί target sdk version = 17 το οποίο αντιστοιχεί στην έκδοση 4.2
- Το όνομα της εφαρμογής και το εικονίδιο της
- Τις άδειες που απαιτούνται για να εκτελέσει η εφαρμογή συγκεκριμένες λειτουργίες. Τέτοιες λειτουργίες μπορεί να είναι η χρήση της φωτογραφικής μηχανής, η πρόσβαση στην κάρτα SD, η πρόσβαση στο διαδίκτυο, η χρήση Wi-Fi κ.α.
- Όλα τα συστατικά στοιχεία της εφαρμογής (activities, services, content providers, broadcast receivers). Αν προσπαθήσει κάποιος να δημιουργήσει μια activity η οποία δεν είναι δηλωμένη στο Manifest τότε αυτή δε θα μπορεί να λειτουργήσει.
- Τις εξωτερικές βιβλιοθήκες με τις οποίες θα πρέπει να συνδεθεί.

Ακολουθεί ένα παράδειγμα ενός AndroidManifest.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.javapapers.android"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk android:minSdkVersion="7" />

    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name" >
        <activity
            android:label="@string/app_name"
            android:name=".HelloWorld" >
            <intent-filter >
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

3.2.2 User Interface (Διεπιφάνεια Χρήστη)

Η διεπιφάνεια χρήστη μιας εφαρμογής είναι οτιδήποτε με το οποίο μπορεί να αλληλεπιδράσει και να δει ο χρήστης. Το Android προσφέρει μια μεγάλη ποικιλία προκατασκευασμένων γραφικών στοιχείων που δίνουν την δυνατότητα στον προγραμματιστή να σχεδιάσει και να φτιάξει την διεπιφάνεια χρήστη.

Όλα τα στοιχεία της διεπιφάνειας σε μια εφαρμογή κατασκευάζονται χρησιμοποιώντας αντικείμενα των κλάσεων View (όψη) και ViewGroup. Μια View είναι ένα αντικείμενο το οποίο σχεδιάζει κάτι στην οθόνη, με το οποίο μπορεί να αλληλεπιδράσει ο χρήστης. Ένα ViewGroup είναι ένα αντικείμενο το οποίο περιέχει άλλα αντικείμενα τύπου View ή ViewGroup, με σκοπό να ορίσει την διάταξη (layout) της εφαρμογής.

Το user interface είναι ουσιαστικά ένα από τα πιο βασικά στοιχεία της εφαρμογής καθώς είναι άρρηκτα συνδεδεμένο με τη λειτουργία της. Αποτελεί την εικόνα που βλέπει ο χρήστης και το περιβάλλον μέσα στο οποίο μπορεί να πλοηγηθεί και να χρησιμοποιήσει τις λειτουργίες της εφαρμογής. Είναι πολύ σημαντικό το user interface να είναι σχεδιασμένο με όμορφο, εύχρηστο και λειτουργικό τρόπο ώστε να ωθεί και να βοηθάει τον χρήστη να χρησιμοποιεί την εφαρμογή. Πολλές εφαρμογές στην αγορά μπορεί να επιτελούν ακριβώς τις ίδιες λειτουργίες, ο χρήστης όμως θα επιλέξει εκείνη η οποία είναι πιο όμορφα, έξυπνα και λειτουργικά σχεδιασμένη ώστε να κάνει την δουλειά του πιο εύκολα και διασκεδαστικά. Καθίσταται σαφές, λοιπόν, ότι πολλές φορές είναι πιο σημαντικό και πιο δύσκολο να σχεδιαστεί το κατάλληλο περιβάλλον εργασίας από ότι η ίδια η λειτουργικότητα της εφαρμογής.

3.2.2.1 Layout

Ένα layout ορίζει την δομή και τη διάταξη των γραφικών στοιχείων ενός user interface.

Ένα layout μπορεί να δηλωθεί με δύο διαφορετικούς τρόπους.

- **Δηλώνοντας τα γραφικά στοιχεία σε ένα αρχείο XML**
  Τα αρχεία xml αποτελούν ένα στατικό τρόπο δημιουργίας των γραφικών στοιχείων της εφαρμογής. Αποθηκεύονται σε ειδικό φάκελο του project και καλούνται μέσα από την εφαρμογή για τη δημιουργία του γραφικού περιβάλλοντος. Είναι πολύ εύκολα στη συγγραφή και κατανόηση του περιεχομένου τους. Παρουσιάζουν μια δενδρική δομή και η κατασκευή τους μοιάζει πάρα πολύ με την χρήση HTML για κατασκευή ιστοσελίδων. Παρακάτω φαίνεται ένα παράδειγμα ενός αρχείου xml το οποίο περιέχει έναν βασικό τύπο Layout και διάφορα γραφικά στοιχεία:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
              android:layout_width="fill_parent"
              android:layout_height="fill_parent"
              android:orientation="vertical" >
    <TextView android:id="@+id/text"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content"
              android:text="Hello, I am a TextView" />
    <Button android:id="@+id/button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Hello, I am a Button" />
</LinearLayout>
```

- **Αρχικοποιώντας τα γραφικά στοιχεία κατά την εκτέλεση**
  Πολλές φορές δεν είναι γνωστή η διάταξη που θα χρησιμοποιηθεί καθώς αυτή μπορεί να εξαρτάται από επιλογές του χρήστη. Σε αυτές τις περιπτώσεις ο προγραμματιστής δημιουργεί τα layouts προγραμματιστικά μέσα στα activities της εφαρμογής.

Το Android δίνει τη δυνατότητα να χρησιμοποιηθούν οποιοσδήποτε από τους παραπάνω τρόπους, ή και οι δύο μαζί.

Υπάρχουν διάφορα είδη Layout. Τα πιο βασικά είναι τα:

- **Linear Layout**
- **Relative Layout**
- **Frame Layout**

Όταν το περιεχόμενο το οποίο πρέπει παρουσιαστεί στον χρήστη είναι δυναμικό και δεν είναι προκαθορισμένο, μπορούν να χρησιμοποιηθούν ειδικά layouts τα οποία είναι υποκλάσεις της κλάσης AdapterView. Μια τέτοια υποκλάση χρησιμοποιεί έναν **Adapter** για να «δέσει» τα δεδομένα με την γραφική διάταξη. Ουσιαστικά ο Adapter είναι ο μεσάζοντας ανάμεσα στα δεδομένα και στην διάταξη. Αυτός λαμβάνει τα δεδομένα από μία πηγή (έναν πίνακα, μια βάση δεδομένων κλπ) και τα παρέχει ένα ένα στην διάταξη. Αξίζει να αναφερθούν δύο πολύ βασικά στοιχεία που χρησιμοποιούνται. Αυτά είναι τα:

- **List View**
- **Grid View**

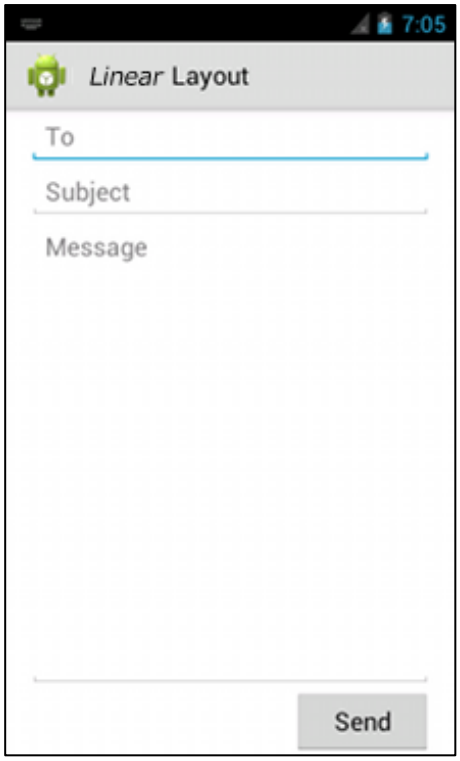
3.2.2.1.1 Linear Layout

Το Linear Layout είναι ένα ViewGroup το οποίο στοιχίζει όλα τα παιδιά τα οποία περιέχει, σε μια κατεύθυνση. Είτε οριζόντια, είτε κάθετα. Όλα τα παιδιά ενός Linear Layout στοιβάζονται το ένα μετά το άλλο. Έτσι, μια κάθετη λίστα θα έχει μόνο ένα στοιχείο σε κάθε σειρά, ανεξάρτητα από το πλάτος της σειράς.

Μια ξεχωριστή δυνατότητα των linear layout είναι η χρήση **weight** (βάρους). Το weight είναι ένα χαρακτηριστικό το οποίο αποδίδεται στα παιδιά ενός linear layout και αντικατοπτρίζει το «πόσο σημαντικό» είναι ένα παιδί για αυτό το layout. Χρησιμοποιώντας αριθμητικές τιμές, το weight επηρεάζει τον τρόπο με τον οποίο εμφανίζονται τα στοιχεία μέσα στη διάταξη και συγκεκριμένα το μέγεθος κάθε στοιχείου σε σχέση με τα υπόλοιπα.

Ακολουθεί ένα παράδειγμα αρχείου xml με χρήση Linear Layout καθώς και η οθόνη την οποία αυτό κατασκευάζει (**Εικόνα 3.5**).

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:orientation="vertical" >
    <EditText
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:hint="@string/to" />
    <EditText
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:hint="@string/subject" />
    <EditText
        android:layout_width="fill_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:gravity="top"
        android:hint="@string/message" />
    <Button
        android:layout_width="100dp"
        android:layout_height="wrap_content"
        android:layout_gravity="right"
        android:text="@string/send" />
</LinearLayout>
```



Εικόνα 3.5 : Μια οθόνη κατασκευασμένη με τη χρήση Linear Layout.

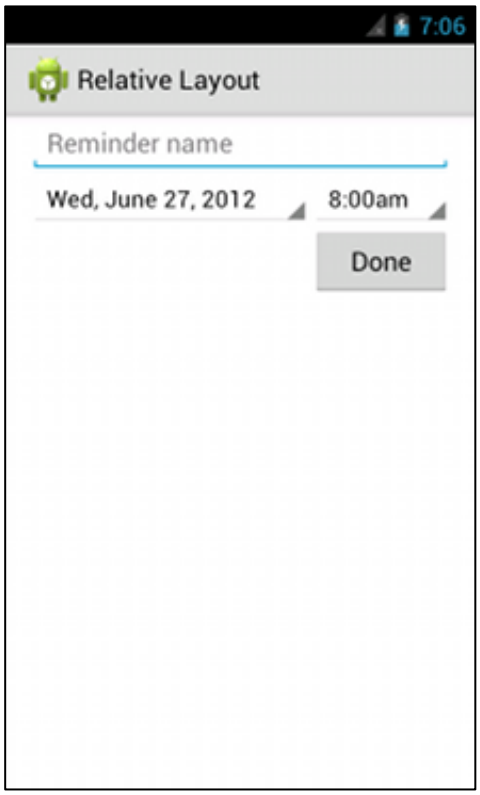
3.2.2.1.2 Relative Layout

Το Relative Layout είναι ένα ViewGroup το οποίο παρουσιάζει όλα τα παιδιά τα οποία περιέχει σε σχετικές μεταξύ τους θέσεις. Για παράδειγμα, μπορεί να οριστεί ένα στοιχείο να εμφανίζεται δεξιά από ένα άλλο, από πάνω ή από κάτω του. Αυτό δίνει τη δυνατότητα να αποφευχθεί η χρήση πολλών φωλιασμένων layout, το ένα μέσα στο άλλο, αυξάνοντα έτσι την απόδοση της εφαρμογής.

Ακολουθεί ένα παράδειγμα αρχείου xml με χρήση Relative Layout καθώς και η οθόνη την οποία αυτό κατασκευάζει (**Εικόνα 3.6**).

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:paddingLeft="16dp"
    android:paddingRight="16dp" >
    <EditText
        android:id="@+id/name"
```

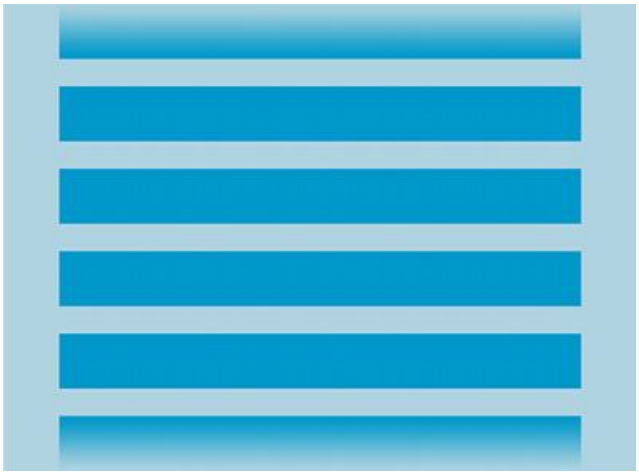
```
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:hint="@string/reminder" />
    <Spinner
        android:id="@+id/dates"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_below="@id/name"
        android:layout_alignParentLeft="true"
        android:layout_toLeftOf="@+id/times" />
    <Spinner
        android:id="@id/times"
        android:layout_width="96dp"
        android:layout_height="wrap_content"
        android:layout_below="@id/name"
        android:layout_alignParentRight="true" />
    <Button
        android:layout_width="96dp"
        android:layout_height="wrap_content"
        android:layout_below="@id/times"
        android:layout_alignParentRight="true"
        android:text="@string/done" />
</RelativeLayout>
```



Εικόνα 3.6 : Μια οθόνη κατασκευασμένη με τη χρήση Relative Layout.

3.2.2.1.3 List View

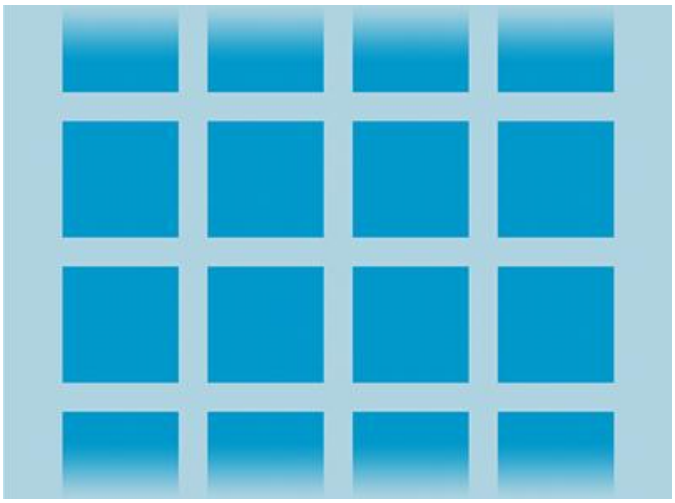
Μια List View είναι ένα ViewGroup το οποίο παρουσιάζει μια λίστα από scrollable αντικείμενα. Τα στοιχεία της λίστας παρέχονται από έναν Adapter ο οποίος τα αντλεί από την πηγή τους, τα μετατρέπει σε κατάλληλα view και τα τοποθετεί στη λίστα.



Εικόνα 3.7 : Μια αφηρημένη εικόνα μιας List View.

3.2.2.1.4 Grid View

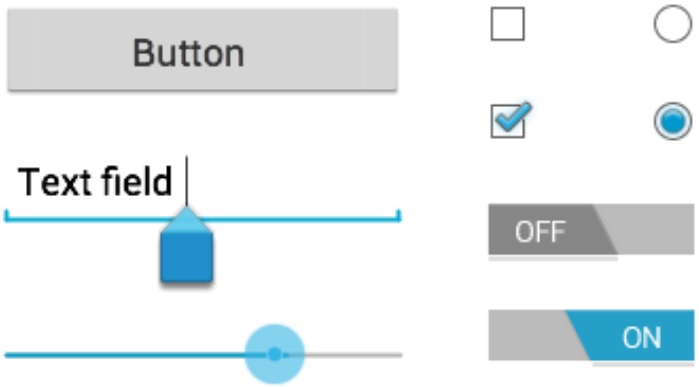
Ένα Grid View είναι ένα ViewGroup το οποίο παρουσιάζει ένα σύνολο από αντικείμενα σε ένα δισδιάστατο, scrollable πλέγμα (grid). Τα στοιχεία του πλέγματος παρέχονται, όπως και στο List View, από έναν Adapter.



Εικόνα 3.8 : Μια αφηρημένη εικόνα ενός Grid View.

3.2.2.2 Input Controls

Τα input controls είναι διαδραστικά στοιχεία στο user interface της εφαρμογής. Το Android προσφέρει μια πληθώρα από τέτοια στοιχεία, όπως πεδία κειμένου (text fields), πλήκτρα (buttons), μπάρες αναζήτησης (seek barks) και πολλά άλλα.



Εικόνα 3.9 : Ένα σύνολο από πολλά και διαφορετικά είδη input controls.

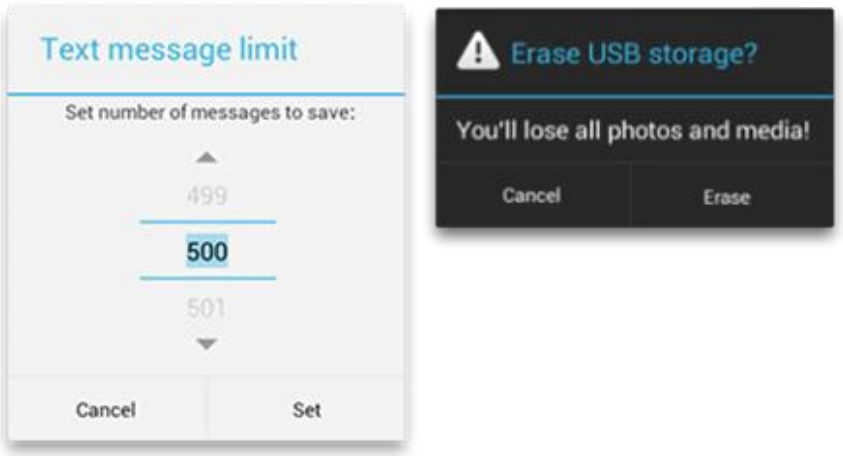
3.2.2.3 Dialogs (Διάλογοι)

Ένας διάλογος (dialog), είναι ένα μικρό παράθυρο το οποίο προτρέπει τον χρήστη να πάρει μια απόφαση ή να εισάγει επιπρόσθετες πληροφορίες. Ένας διάλογος δεν γεμίζει όλη την οθόνη και συνήθως χρησιμοποιείται όταν ο χρήστης πρέπει να κάνει κάποια ενέργεια για να μπορέσει να συνεχίσει. Όταν ο διάλογος εμφανίζεται, τότε η activity χάνει την εστίαση (focus) της και η εφαρμογή εστιάζει σε αυτόν. Έτσι, ο χρήστης μπορεί να αλληλεπιδράσει μόνο με τον διάλογο. Τα δύο πιο βασικά είδη διαλόγου είναι:

- Alert Dialog – Ένας διάλογος ο οποίος αποτελείται από ένα τίτλο, μερικά κουμπιά, ένα κείμενο ή μια λίστα επιλογών.
- Progress Dialog – Ένας διάλογος προόδου. Χρησιμοποιείται για να δείξει στον χρήστη την πρόοδο μιας διαδικασίας. Όταν εκτελούνται χρονοβόρες διαδικασίες, όπως για παράδειγμα φόρτωση εικόνων ή σύνδεση με κάποιον εξυπηρετητή μέσω διαδικτύου, για να μην βλέπει ο χρήστης μια παγωμένη ή κενή οθόνη, χρησιμοποιείται το progress dialog για να ενημερωθεί για την πρόοδο της διαδικασίας.

Ένα ακόμα ειδικό είδος διαλόγου είναι και τα Dialog Fragment, τα οποία ουσιαστικά είναι ένας διάλογος αλλά έχει ως βάση ένα Fragment (Παράγραφος 3.2.1.1.1) και καλείται μέσα από την activity που το περιέχει.

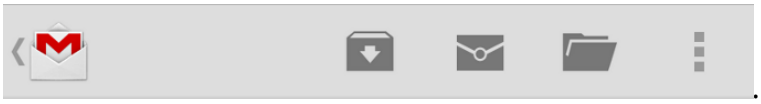
Στην παρακάτω **Εικόνα 3.10** φαίνονται δύο παραδείγματα διαλόγων.



Εικόνα 3.10 : Παραδείγματα διαλόγων.

3.2.2.4 Action Bar

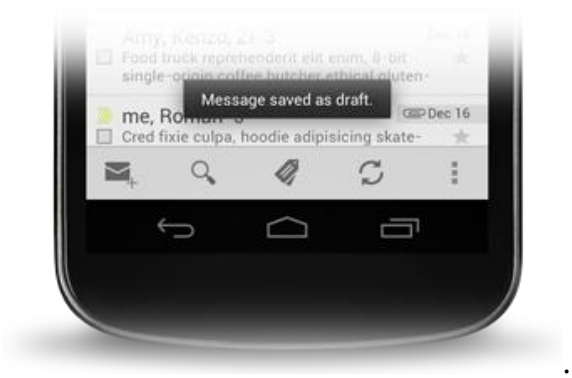
Η Action Bar είναι μια μπάρα η οποία εμφανίζεται συνήθως στην κορυφή του παραθύρου και δίνει τη δυνατότητα στο χρήστη να επιλέξει να εκτελέσει συγκεκριμένες ενέργειες. Μπορεί να χρησιμοποιηθεί από όλες τις εφαρμογές και έχει ένα ενιαίο στυλ κάνοντας την ένα χαρακτηριστικό στοιχείο του Android. Είναι παραμετροποιήσιμη, μπορεί δηλαδή ο προγραμματιστής να προσθέσει τον τίτλο που επιθυμεί, πλήκτρα και άλλα input controls ανάλογα με την εφαρμογή που αναπτύσσει.



Εικόνα 3.11 : Μια action bar με τρία πλήκτρα.

3.2.2.5 Toasts

Τα toast προσφέρουν ένα πολύ απλό και εύκολο τρόπο ειδοποίησης του χρήστη σε ένα μικρό pop-up παράθυρο. Εμφανίζονται για πολύ μικρό χρονικό διάστημα και έπειτα εξαφανίζονται. Ο χρήστης δε μπορεί να αλληλεπιδράσει μαζί τους, παρά μόνο να ενημερωθεί για κάποιο συμβάν. Για παράδειγμα, σε μια φόρμα εισαγωγής δεδομένων, αν ο χρήστης εισάγει μη αποδεκτά δεδομένα, η εφαρμογή μπορεί να τον ενημερώσει με ένα toast που θα λέει απλώς «Εισάγατε λάθος δεδομένα», ώστε ο χρήστης να κάνει τις απαραίτητες διορθώσεις.



Εικόνα 3.12 : Ένα παράδειγμα toast όπου μια εφαρμογή αποστολής μηνυμάτων ειδοποιεί τον χρήστη ότι το μήνυμα του αποθηκεύτηκε στα πρόχειρα.

3.2.3 App Resources

Ο όρος App Resources αναφέρεται στους πόρους που χρησιμοποιεί η εφαρμογή, όπως είναι οι εικόνες ή τα strings. Όλοι οι πόροι πρέπει να είναι ανεξάρτητοι από τον κώδικα της εφαρμογής και να δηλώνονται σε διαφορετικό μέρος από τις κλάσεις. Η εξωτερίκευση των πόρων, επιτρέπει την παροχή εναλλακτικών πόρων για διαφορετικές ρυθμίσεις (configurations) της κάθε συσκευής, όπως χρήση διαφορετικής γλώσσας ή μέγεθος οθόνης. Καθώς όλο και περισσότερες διαφορετικές συσκευές με Android κυκλοφορούν στην αγορά, με διαφορετικά χαρακτηριστικά και ρυθμίσεις μεταξύ τους, καθίσταται απαραίτητο η εφαρμογή να μπορεί να είναι συμβατή με κάθε συσκευή ανεξάρτητα από τα χαρακτηριστικά της. Ακόμα, θα πρέπει η εφαρμογή να ανταποκρίνεται και να είναι συμβατή με τις διαφορετικές ρυθμίσεις στην ίδια συσκευή.

Οι βασικές ρυθμίσεις και εναλλαγές ρυθμίσεων μπορούν να συνοψιστούν σε τρεις βασικές κατηγορίες

- **Προσανατολισμός οθόνης**
- **Μέγεθος και ανάλυση οθόνης**
- **Γλώσσα**

3.2.3.1 Προσανατολισμός οθόνης

Οι συσκευές με λειτουργικό σύστημα Android μπορούν να τοποθετηθούν είτε σε προσανατολισμό πορτραίτου (portrait mode) είτε σε προσανατολισμό τοπίου (landscape mode). Η συσκευή βρίσκεται σε προσανατολισμό πορτραίτου όταν είναι σε όρθια θέση, όταν δηλαδή το μεγαλύτερο μέρος της οθόνης είναι κατακόρυφη (όπως κρατάει κάποιος συνήθως ένα κινητό τηλέφωνο). Σε προσανατολισμό τοπίου βρίσκεται όταν η μεγαλύτερη πλευρά της

οθόνης είναι οριζόντια. Είναι προφανές ότι θα ήταν επιθυμητό σε κάποιες περιπτώσεις να εμφανίζονται τα γραφικά στοιχεία της εφαρμογής με διαφορετικό τρόπο, ανάλογα με τον προσανατολισμό της οθόνης. Για παράδειγμα, σε μια εφαρμογή παρουσίασης φωτογραφιών, όταν ο χρήστης κρατάει την συσκευή σε κατακόρυφη θέση, είναι επιθυμητό να εμφανίζεται ένα πλέγμα με δυο φωτογραφίες σε κάθε σειρά. Όταν όμως ο χρήστης αλλάξει τον προσανατολισμό σε οριζόντια θέση, τώρα θα ήταν επιθυμητό να εμφανίζονται τρεις φωτογραφίες για να γίνει εκμετάλλευση του μεγαλύτερο πλάτους της οθόνης. Ένα τέτοιο παράδειγμα φαίνεται στην **Εικόνα 3.13**.

Εικόνα 3.13 : Ένα παράδειγμα όπου κατά την αλλαγή του προσανατολισμού της οθόνης, η διάταξη των στοιχείων αλλάζει.

Εικόνα 3.14 : Ένα παράδειγμα όπου κατά την αλλαγή του προσανατολισμού της οθόνης, η διάταξη των στοιχείων παραμένει η ίδια.

Όμως, ο προεπιλεγμένος τρόπος (default) με τον οποίο το Android χειρίζεται τις εναλλαγές του προσανατολισμού είναι να διατηρεί την ίδια διάταξη. Με αυτό τον τρόπο και συνεχίζοντας το προηγούμενο παράδειγμα, ο χρήστης θα έβλεπε και στις δύο περιπτώσεις δύο φωτογραφίες. Αυτό φαίνεται στην **Εικόνα 3.14**

Όπως αναφέρθηκε στην παράγραφο 3.2.2 για το user interface, η διάταξη (layout) των γραφικών στοιχείων της εφαρμογής δηλώνεται σε αρχεία xml. Συγκεκριμένα, σε ένα project, τα αρχεία αυτά τοποθετούνται στον φάκελο res/layout. Το res αναφέρεται στο resources και είναι ένας φάκελος μέσα στον οποίο ουσιαστικά τοποθετούνται όλοι οι απαραίτητοι πόροι της εφαρμογής, σε κατάλληλους υποφακέλους.

Για να επιτευχθεί λοιπόν το αποτέλεσμα της **Εικόνας 3.13**, θα κατασκευαστούν δύο διαφορετικά layout, δηλαδή δύο διαφορετικά αρχεία xml. Στο ένα θα δημιουργηθεί ένα layout που θα περιέχει ένα Grid View (Παράγραφο 3.2.2.1.4) με 2 στήλες και στο άλλο ένα με 3 στήλες. Έπειτα, θα πρέπει να τοποθετηθούν αυτά τα δύο xml στους κατάλληλους φακέλους. Το Android παρέχει κατάλληλη ονοματολογία για τους υποφακέλους αυτούς, ώστε να τους αναγνωρίζει και να αντλεί αυτόματα μέσα από αυτούς τους πόρους που χρειάζεται. Το πρώτο xml θα τοποθετηθεί στον φάκελο res/layout όπως τοποθετούνται όλα τα xml, όμως το δεύτερο θα τοποθετηθεί στον φάκελο res/layout-land (δηλαδή landscape) που αναφέρεται στον προσανατολισμό τοπίου. Όταν λοιπόν ο χρήστης κρατάει την συσκευή σε προσανατολισμό

πορτραίτου, το Android θα φορτώσει τη διάταξη από τον φάκελο res/layout ενώ όταν ο χρήστης κρατάει τη συσκευή σε προσανατολισμό τοπίου, θα φορτώσει τη διάταξη από τον φάκελο res/layout-land.

3.2.3.2 Μέγεθος και ανάλυση οθόνης

Ακριβώς όπως και με τον προσανατολισμό οθόνης, έτσι και σε αυτή τη παράγραφο θα αναλυθούν οι τρόποι με τους οποίους μπορούν να παρουσιαστούν στον χρήστη τα γραφικά στοιχεία με διαφορετικό τρόπο ανάλογα με το μέγεθος και την ανάλυση της οθόνης της συσκευής του.

Για παράδειγμα, απαιτείται να εμφανιστεί μια εικόνα σε ένα τηλέφωνο με μέγεθος οθόνης 4 ίντσες και σε μια ταμπλέτα με μέγεθος οθόνης 10 ίντσες. Τότε, το σωστό θα ήταν να χρησιμοποιηθούν εικόνες διαφορετικής ανάλυσης στην κάθε περίπτωση. Αλλιώς, αν χρησιμοποιηθεί η ίδια εικόνα που εμφανίζεται σε ένα Χ μέγεθος στο τηλέφωνο, στην ταμπλέτα η ίδια εικόνα θα καταλαμβάνει πολύ μικρότερο χώρο με αποτέλεσμα και να φαίνεται μικρότερη στο χρήστη και να δημιουργεί προβλήματα στη διάταξη. Αυτό μπορεί να συμβεί όχι μόνο με εικόνες, αλλά με κάθε γραφικό στοιχείο που χρησιμοποιεί η εφαρμογή.

Το πρόβλημα λύνεται, όπως και με τον προσανατολισμό οθόνης, χρησιμοποιώντας διαφορετικούς πόρους για διαφορετικά μεγέθη οθόνης (πχ την ίδια εικόνα σε διαφορετικά μεγέθη/αναλύσεις, μια για κάθε περίπτωση μεγέθους οθόνης). Αρχικά, οι φάκελοι στους οποίους έπρεπε να τοποθετηθούν ήταν υποφάκελοι του res (πχ res/drawable, res/layout κλπ) ακολουθούμενοι από το μέγεθος: *small, normal, large, and xlarge*. (πχ res/drawable-small). Από το Android 3.2 όμως και μετά χρησιμοποιείται μια διαφορετική και πιο έξυπνη τεχνική. Αντί να χωρίζει το Android τις οθόνες σε σύνολα μεγεθών, χρησιμοποιεί την τεχνική του μικρότερου πλάτους.

Αυτό καθιερώθηκε με τη διάδοση και τη χρήση των ταμπλετών, διότι μέχρι τότε ο μόνος φάκελος που μπορούσαν οι προγραμματιστές να χρησιμοποιήσουν για να τοποθετήσουν τους πόρους για χρήση σε ταμπλέτα ήταν ο φάκελος ακολουθούμενος από το μέγεθος xlarge (πχ res/drawable-xlarge). Αυτό είχε ως αποτέλεσμα ότι, όλες οι ταμπλέτες, είτε ήταν 8 ίντσες, είτε 10, είτε 12 θα έπρεπε να χρησιμοποιούν τους ίδιους πόρους. Η νέα λοιπόν αυτή τεχνική χρησιμοποιεί νέους υποφακέλους οι οποίοι έχουν ως χαρακτηριστικό στοιχείο το μικρότερο δυνατό πλάτος.

Για παράδειγμα, κατασκευάζονται δύο φάκελοι, ο **res/layout-sw600dp** (smallest width 600dp) και ο **res/layout-sw720dp**. Ο φάκελος res/layout-sw600dp περιέχει πόρους οι οποίοι θα χρησιμοποιηθούν από το σύστημα μόνο αν το μικρότερο δυνατό πλάτος (και κατά συνέπεια μέγεθος) της οθόνης είναι 600dp (density pixels). Ο res/layout-sw720dp, περιέχει πόρους οι οποίοι θα χρησιμοποιηθούν μόνο αν το μικρότερο δυνατό μέγεθος είναι 720dp.

Αυτό σημαίνει ότι μέχρι 600dp (πχ τηλέφωνο) θα χρησιμοποιήσει τον προκαθορισμένο φάκελο res/layout μιας και δεν έχει οριστεί άλλο μικρότερο δυνατό πλάτος, για συσκευές με πλάτος από 600dp έως 720dp (πχ ταμπλέτα 7") θα χρησιμοποιήσει τον πρώτο φάκελο και για συσκευές με πλάτος από 720dp και πάνω (πχ ταμπλέτα 10") θα χρησιμοποιήσει τον δεύτερο φάκελο.

Η αντιστοίχηση ανάμεσα σε μέγεθος οθόνης και density pixels φαίνεται παρακάτω:

- 320dp: μια τυπική οθόνη τηλεφώνου (240x320 ldpi, 320x480 mdpi, 480x800 hdpi, κλπ).
- 480dp: μια μικρή ταμπλέτα (480x800 mdpi).
- 600dp: μια ταμπλέτα 7" (600x1024 mdpi).
- 720dp: μια ταμπλέτα 10" (720x1280 mdpi, 800x1280 mdpi, κλπ).

Ισχύει **px = dp * (dpi / 160)**.

3.2.3.3 Γλώσσα

Όταν ένας προγραμματιστής αναπτύσσει μια εφαρμογή Android πρέπει να έχει στο μυαλό του ότι η εφαρμογή αυτή δε θα χρησιμοποιηθεί μόνο από χρήστες της χώρας του, αλλά μέσω του Google Play η εφαρμογή μπορεί να είναι διαθέσιμη σε όλο τον κόσμο. Γι αυτό και πρέπει να κάνει την εφαρμογή εύχρηστη σε χρήστες από διαφορετικές χώρες, όπου χρησιμοποιούν στη συσκευή τους διαφορετική γλώσσα. Το Android δίνει αυτή τη δυνατότητα με πολύ εύκολο τρόπο.

Όλες οι λέξεις που εμφανίζονται στις οθόνες μιας εφαρμογής, είτε είναι κείμενο, είτε εντολές, είτε λέξεις μέσα σε πλήκτρα κλπ, δεν είναι τίποτα άλλο από ένα σύνολο από strings. Γι αυτό και δεν πρέπει ποτέ να δίνεται μια στατική τιμή string σε κάποιο γραφικό στοιχείο αλλά να χρησιμοποιούνται ειδικές μεταβλητές οι οποίες περιέχουν την τιμή που πρέπει να δοθεί στο στοιχείο. Αυτό επιτυγχάνεται με τη χρήση ενός αρχείου xml, του strings.xml και την τοποθέτηση του μέσα στο φάκελο res/values/ . Μέσα σε αυτό το xml αρχείο μπορούν να δηλωθούν μεταβλητές οι οποίες θα περιέχουν τιμές τύπου string. Αυτό γίνεται με τον εξής τρόπο: `<string name = "foobar"> Foo Bar </string>`.

Η παραπάνω δήλωση ορίζει την μεταβλητή foobar η οποία έχει ως τιμή το string "Foo Bar". Πώς όμως η χρήση του αρχείου string.xml και η ανάθεση τιμών σε μεταβλητές βοηθάει στην χρήση διαφορετικών γλωσσών; Την απάντηση έρχεται να δώσει το ίδιο το Android με τη χρήση επεκτάσεων στο όνομα των φακέλων όπως ακριβώς έκανε και με τα μεγέθη της οθόνης και τον προσανατολισμό της. Αν, για παράδειγμα, είναι ανάγκη να υπάρχει ένας φάκελος με τα strings που θα χρησιμοποιηθούν όταν ο χρήστης διαθέτει μια συσκευή στην ελληνική γλώσσα, αρκεί να δημιουργηθεί το αρχείο strings.xml και να τοποθετηθεί αυτή τη φορά στον φάκελο res/values-**el**. Η επέκταση –el σημαίνει ότι το λειτουργικό θα χρησιμοποιήσει αυτούς τους πόρους μόνο αν η γλώσσα της συσκευής είναι τα ελληνικά.

Έτσι, με τη δημιουργία ξεχωριστών φακέλων για κάθε γλώσσα, μπορεί πολύ εύκολα η εφαρμογή να γίνει συμβατή με πολλές γλώσσες. Για να ολοκληρωθεί η επίλυση του προβλήματος, θα πρέπει να τονιστεί ότι πρέπει η μεταβλητή να έχει το ίδιο όνομα στις διάφορες γλώσσες και να αλλάζει μόνο το περιεχόμενο της. Για παράδειγμα, για χρήση με μια ελληνική συσκευή, υπάρχει στον φάκελο res/values-el, το αρχείο strings.xml που περιέχει την μεταβλητή `<string name = "hello"> Γεια! </string>`. Ο φάκελος res/values-en, για χρήση με συσκευή που χρησιμοποιεί την αγγλική γλώσσα, θα πρέπει στο δικό του xml να έχει την ίδια μεταβλητή με άλλο περιεχόμενο, δηλαδή: `<string name = "hello"> Hello! </string>`. Αντίστοιχη εργασία απαιτείται και για άλλες γλώσσες, στους φακέλους res/values-de για γερμανικά, res/values-fr για γαλλικά κλπ. Αν λοιπόν υπάρχει μια εφαρμογή η οποία τυπώνει ένα μήνυμα στην οθόνη, και αυτό το μήνυμα παίρνει το περιεχόμενο του από την μεταβλητή hello, γίνεται εύκολα κατανοητό ότι καθένας από τους παραπάνω χρήστες θα δει στην οθόνη του το μήνυμα τυπωμένο στη δική του γλώσσα!

Πρέπει να σημειώσουμε ότι όλα όσα αναφέρθηκαν στην Παράγραφο 3.2.2 για τους πόρους της εφαρμογής, μπορούν να δουλέψουν και συνδυαστικά. Για παράδειγμα, αν δηλωθεί το user interface μιας οθόνης στον φάκελο res/layout-en-land, τότε αυτό θα χρησιμοποιηθεί μόνο αν η συσκευή χρησιμοποιεί την αγγλική γλώσσα και είναι σε προσανατολισμό τοπίου.

Γίνεται κατανοητό, λοιπόν, ότι η σωστή χρήση των app resources και των κατάλληλων φακέλων είναι απαραίτητη για να είναι μια εφαρμογή συμβατή με όσο το δυνατόν περισσότερες συσκευές.

4. Αρχιτεκτονική της εφαρμογής

4.1 Εισαγωγή

Στο παρόν κεφάλαιο θα περιγραφεί αναλυτικά η εφαρμογή που αναπτύχθηκε για περιβάλλον Android. Θα αναλυθεί η εξωτερική δομή της, δηλαδή οι φάκελοι στους οποίους οργανώνεται παράλληλα με τα αρχεία που τους αποτελούν. Η αναλυτική λειτουργία των κλάσεων θα περιγραφεί στο επόμενο κεφάλαιο.

4.2 Περιγραφή

Ο κύριος σκοπός της εφαρμογής είναι η δυνατότητα του χρήστη να ψωνίζει ηλεκτρονικά (e-shopping) σε διαφορετικά υπάρχοντα ηλεκτρονικά καταστήματα, ταυτόχρονα, χωρίς να χρειάζεται κάθε φορά να επισκεφτεί το αντίστοιχο κατάστημα ξεχωριστά. Ακόμα, παρέχει ένα σύστημα εκπτωτικών κουπονιών για προϊόντα, ώστε ο χρήστης να μπορεί να δει ποια κουπόνια είναι διαθέσιμα και για ποια προϊόντα. Τα κουπόνια αυτά είναι δημόσια και μπορούν να χρησιμοποιηθούν από τον καθένα, υπάρχουν όμως και προσωπικά κουπόνια τα οποία μπορεί να ανήκουν είτε στον χρήστη αποκλειστικά ή να ανήκουν σε κάποιον άλλον χρήστη ο οποίος τα μοιράζεται με τον χρήστης της εφαρμογής.

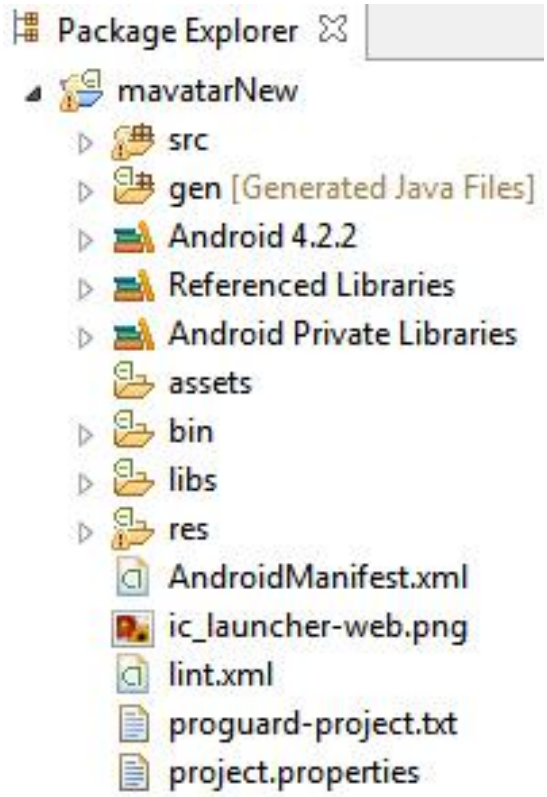
Μέσα από την εφαρμογή, ο χρήστης έχει τη δυνατότητα να περιηγηθεί, μέσω μιας πληθώρας διαφορετικών κατηγοριών, σε ένα σύνολο από προϊόντα καθένα από τα οποία μπορεί να ανήκει σε διαφορετικό ηλεκτρονικό κατάστημα. Όταν ο χρήστης επιλέξει το προϊόν για το οποίο θέλει να ενημερωθεί, η εφαρμογή παρουσιάζει αρχικά μια συνοπτική περιγραφή του προϊόντος μαζί με τα κουπόνια που είναι διαθέσιμα για το κατάστημα του προϊόντος και του δίνει τη δυνατότητα να μεταβεί στην ιστοσελίδα του καταστήματος για να κάνει την αγορά του. Η ιστοσελίδα αυτή, ανοίγει μέσα στην εφαρμογή και όχι σε ξεχωριστό περιηγητή (browser) κάνοντας έτσι την χρήση πιο εύκολη και ομαλή και δίνει ταυτόχρονα τη δυνατότητα χρήσης λειτουργιών της εφαρμογής μέσα στην ιστοσελίδα. Μια τέτοια λειτουργία είναι η χρήση των εκπτωτικών κουπονιών του χρήστη κατά την αγορά του προϊόντος.

Όλες οι παραπάνω πληροφορίες (κατηγορίες, προϊόντα, κουπόνια) αποστέλλονται στην εφαρμογή σε πραγματικό χρόνο μέσω της σύνδεσης με έναν κεντρικό εξυπηρετητή (server) ο οποίος διαθέτει μια κατάλληλη βάση δεδομένων. Μέσω κατάλληλων αιτήσεων (request) η εφαρμογή ζητά τα δεδομένα από τον εξυπηρετητή και έπειτα χειρίζεται κατάλληλα την απάντηση που ο εξυπηρετητής αποστέλλει. Ύστερα, εμφανίζει αυτά τα δεδομένα με έναν απλό και ελκυστικό τρόπο στον χρήστη, συνήθως μέσω ενός πλέγματος (grid) εικόνων και δεδομένων που αναπαριστούν τις κατηγορίες, τα προϊόντα ή τα κουπόνια.

4.3 Δομή

Η δομή της παρούσας εφαρμογής, όπως και κάθε εφαρμογής για Android, αποτελείται από συγκεκριμένους βασικούς φακέλους και αρχεία σε δενδρική δομή. Όπως αναφέρθηκε και στο κεφάλαιο 3, ένας προγραμματιστής πρέπει να τοποθετεί τις κλάσεις και τα αρχεία της εφαρμογής του στους κατάλληλα σχεδιασμένους φακέλους που του δίνονται.

Στο προγραμματιστικό περιβάλλον Eclipse στο οποίο αναπτύχθηκε η εφαρμογή, η δομή της εμφανίζεται όπως φαίνεται παρακάτω:

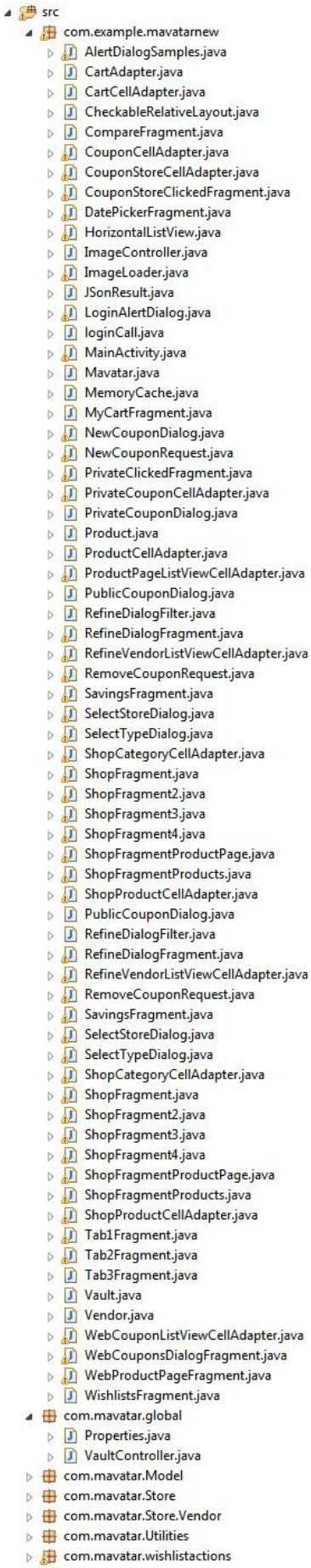


Εικόνα 4.1 : Η δομή της εφαρμογής

Θα περιγραφεί η χρησιμότητα και η σημασία των παραπάνω φακέλων και αρχείων καθώς και των περιεχομένων τους.

- **mavatarNew –** Ο βασικός φάκελος και το όνομα του project (και κατ' επέκταση και της εφαρμογής). Μέσα σε αυτόν περιέχονται όλα τα αρχεία της εφαρμογής.

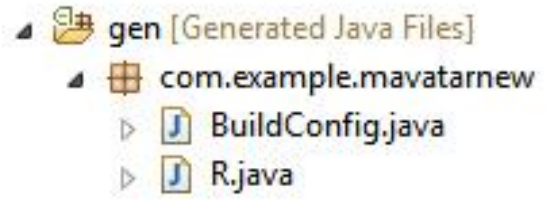
- **src/** - Ένας από τους πιο βασικούς φακέλους της εφαρμογής. Περιέχει όλα τα πακέτα στα οποία έχει δηλωθεί η εφαρμογή. Κάθε πακέτο περιέχει τις κλάσεις σε Java μέσα στις οποίες υπάρχει ο λειτουργικός κώδικας της εφαρμογής.



Εικόνα 4.2 : Ο φάκελος src και τα περιεχόμενα του

Το πακέτο com.example.mavatarnew είναι και το πιο σημαντικό όπως φαίνεται και από το μέγεθος και το πλήθος των κλάσεων που περιέχει. Οι κλάσεις αυτές είναι απαραίτητες για την λειτουργία της εφαρμογής καθώς περιέχουν τον κώδικα που περιγράφει την λειτουργία της. Το πακέτο com.mavatar.global περιέχει μόνο δύο κλάσεις οι οποίες περιέχουν ως επί το πλείστον global μεταβλητές που χρησιμοποιούνται από την εφαρμογή. Τα υπόλοιπα πακέτα προϋπήρχαν και δεν χρησιμοποιήθηκαν στα πλαίσια της διπλωματικής, περιέχουν όμως σημαντικά αρχεία που μπορούν να χρησιμοποιηθούν για να προστεθούν στο μέλλον επιπλέον λειτουργίες και γενικότερα για να γίνει μια επέκταση της εφαρμογής.

- **gen/**



Εικόνα 4.3 : Ο φάκελος gen και τα περιεχόμενα του

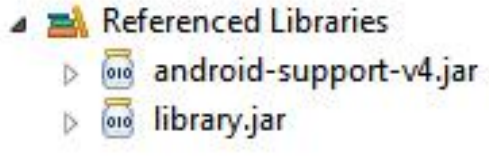
Ο φάκελος gen περιέχει δύο κλάσεις. Η BuildConfig.java δημιουργείται αυτόματα από το σύστημα και χρησιμοποιείται από αυτό για debuging. Η R.java κατασκευάζεται και αυτή αυτόματα και περιέχει όλα τα ID για όλα τα resources που χρησιμοποιούνται από την εφαρμογή. Φροντίζει να γίνεται η σύνδεση με τα resources που υπάρχουν στους αντίστοιχους φακέλους (όπως αναφέρθηκε στην παράγραφο 3.2.3) ώστε να είναι δυνατό μέσω αυτής της κλάσης να γίνεται αναφορά σε αυτά.

- **Android 4.2.2**

Εικόνα 4.4 : Ο φάκελος Android 4.2.2 και τα περιεχόμενα του

Ο φάκελος αυτός περιέχει τις βιβλιοθήκες που δημιουργούνται αυτόματα από το σύστημα. Καθορίζονται από την ελάχιστη έκδοση Android που έχει οριστεί ότι θα είναι συμβατή με την εφαρμογή. Όπως έχει αναφερθεί και όπως φαίνεται και στην παραπάνω εικόνα, η ελάχιστη έκδοση Android με την οποία είναι συμβατή η εφαρμογή είναι η 4.2.

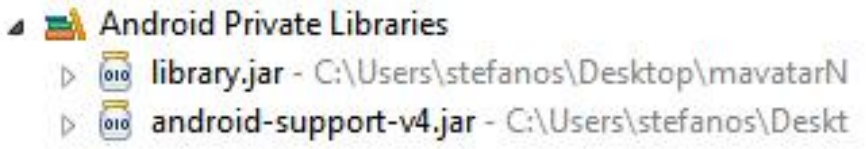
- **Referenced Libraries**



Εικόνα 4.4 : Ο φάκελος Referenced Libraries και τα περιεχόμενα του

Ο φάκελος αυτός περιέχει τις βιβλιοθήκες που χρησιμοποιούνται από την εφαρμογή και τις οποίες έχει προσθέσει ο προγραμματιστής και δεν είναι ενσωματωμένες στις βιβλιοθήκες τους λειτουργικού συστήματος. Και οι δύο παραπάνω βιβλιοθήκες περιέχουν επιπρόσθετες λειτουργίες μέσω της βιβλιοθήκης android-support η οποία χρησιμοποιείται ευρέως στην ανάπτυξη εφαρμογών.

- **Android Private Libraries**



Εικόνα 4.5 : Ο φάκελος Android Private Libraries και τα περιεχόμενα του

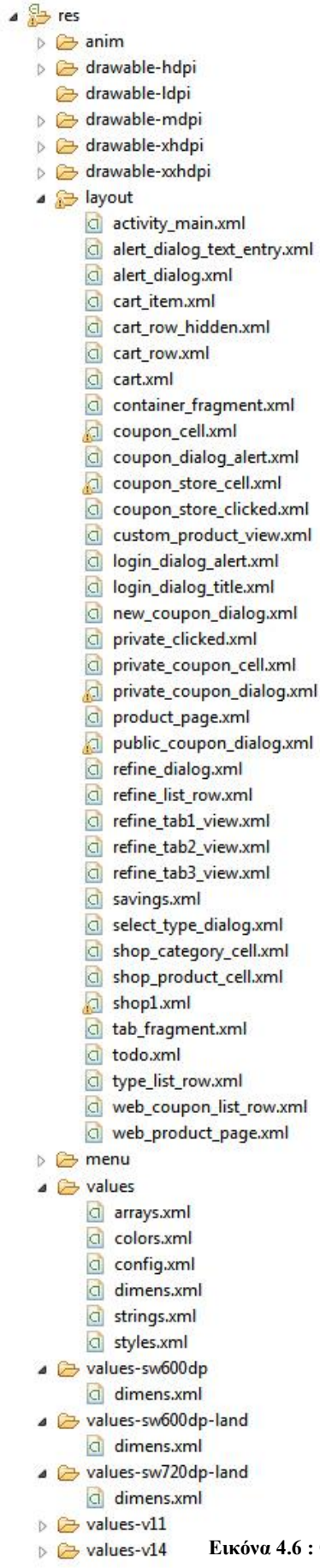
Ο φάκελος αυτός περιέχει τις βιβλιοθήκες που χρησιμοποιούνται αποκλειστικά από την εφαρμογή. Παρατηρείται ότι είναι ίδιες με αυτές του φακέλου Referenced Libraries. Αυτό συμβαίνει διότι τα .jar αρχεία περιέχουν πακέτα τα οποία αναφέρονται (refer) σε άλλες βιβλιοθήκες. Γι αυτό το λόγο και περιέχονται εκ νέου και στο φάκελο Referenced Libraries.

- **assets/ -** Στο φάκελο αυτό μπορεί να αποθηκευθεί οποιοδήποτε αρχείο. Βέβαια, όπως έχει αναφερθεί, τα Application Resources αποθηκεύονται σε ειδικό φάκελο, με αποτέλεσμα αυτός ο φάκελος συνήθως να είναι άδειος – όπως και στην παρούσα εφαρμογή. Ουσιαστικά εδώ ο προγραμματιστής μπορεί να αποθηκεύσει όποιο αρχείο δεν μπορεί να αποθηκευθεί κατάλληλα στους resources φακέλους.

- **bin/** - Ο φάκελος αυτός δημιουργείται από το σύστημα κατά τη δημιουργία του project και ανανεώνεται κάθε φορά που το project γίνεται build επιτυχώς και τρέχει. Μέσα σε αυτό το φάκελο, μετά από ένα επιτυχές build, μπορεί να βρεθεί και το εκτελέσιμο αρχείο .apk της εφαρμογής.

- **libs/ -** Ο φάκελος αυτός περιέχει την φυσική τοποθεσία των βιβλιοθηκών του φακέλου Referenced Libraries που αναφέρθηκαν παραπάνω.

- **res/ -** Ο φάκελος στον οποίο αποθηκεύονται τα Application Resources (Παράγραφος 3.2.3)



Εικόνα 4.6 : Ο φάκελος res και τα περιεχόμενα του

- Στους φακέλους drawable αποθηκεύονται οι εικόνες που χρησιμοποιεί η εφαρμογή. Ανάλογα με το αντίστοιχο πρόσθετο (πχ –hdpi) τοποθετούνται οι εικόνες που πρέπει να χρησιμοποιεί το σύστημα για την αντίστοιχη ανάλυση οθόνης της συσκευής.
- Στον φάκελο layout αποθηκεύονται τα Layouts (παράγραφος 3.2.2.1) της εφαρμογής σε κατάλληλα αρχεία .xml.
- Στον φάκελο values αποθηκεύονται κατάλληλα αρχεία που περιέχουν ειδικές μεταβλητές που χρησιμοποιεί η εφαρμογή (παράγραφοι 3.2.3.1-3). Αντίστοιχα στους φακέλους values με το αντίστοιχο πρόσθετο (πχ values-sw600dp-land) περιέχονται οι μεταβλητές που χρησιμοποιεί το σύστημα σε αντίστοιχες περιπτώσεις (πχ συσκευή με smallest width 600dp και προσανατολισμός τοπίου).
- Στον φάκελο anim περιέχονται τα αρχεία xml που περιγράφουν κάποια κίνηση animation που χρησιμοποιείται από την εφαρμογή.

Τέλος, παρατηρείται το αρχείο AndroidManifest.xml το οποίο και αναλύθηκε στην παράγραφο 3.2.1.6, ένα αρχείο εικόνας .jpg το οποίο είναι η εικόνα της εφαρμογής καθώς και τρία αρχεία που δημιουργούνται αυτόματα κατά την δημιουργία του project.

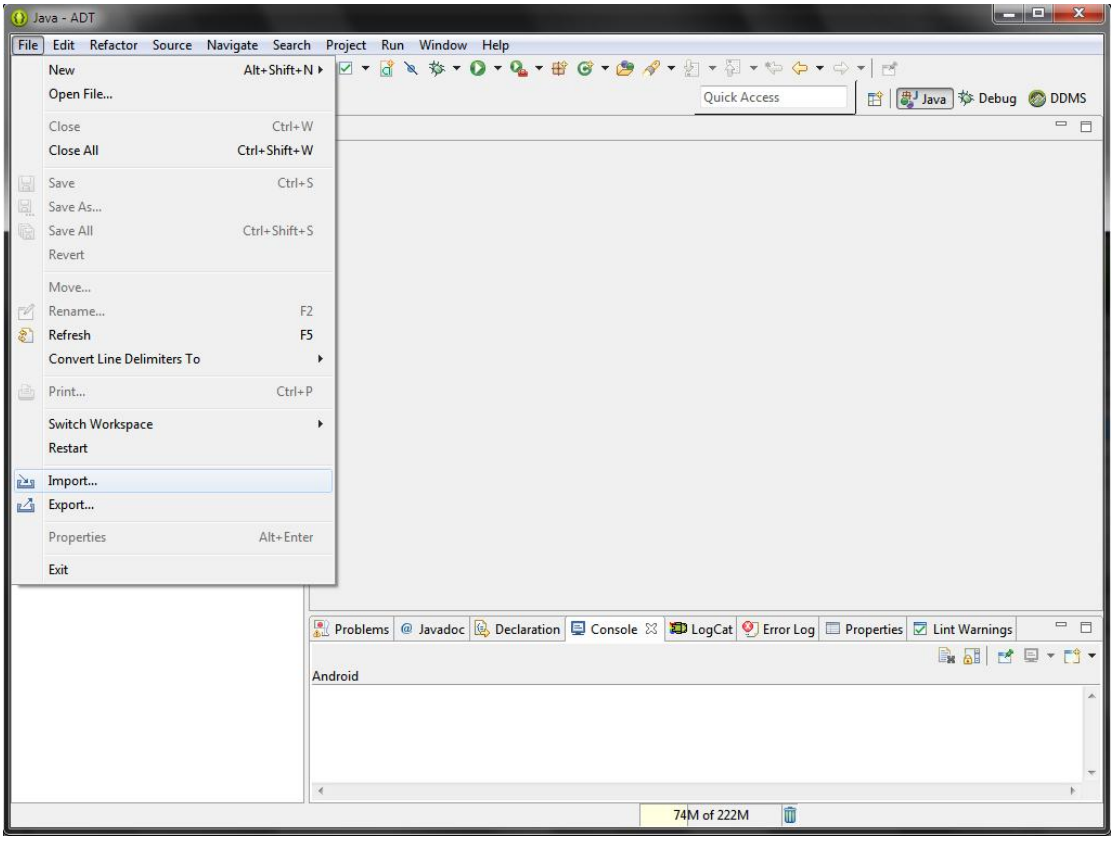
5. Αναλυτική λειτουργία και χρήση της εφαρμογής

5.1 Οδηγίες εκτέλεσης της εφαρμογής

Σε αυτή τη παράγραφο επεξηγείται αναλυτικά ο τρόπος με τον οποίο μπορεί κάποιος να εκτελέσει την παρούσα εφαρμογή τόσο σε περιβάλλον Eclipse όσο και σε μια συσκευή Android.

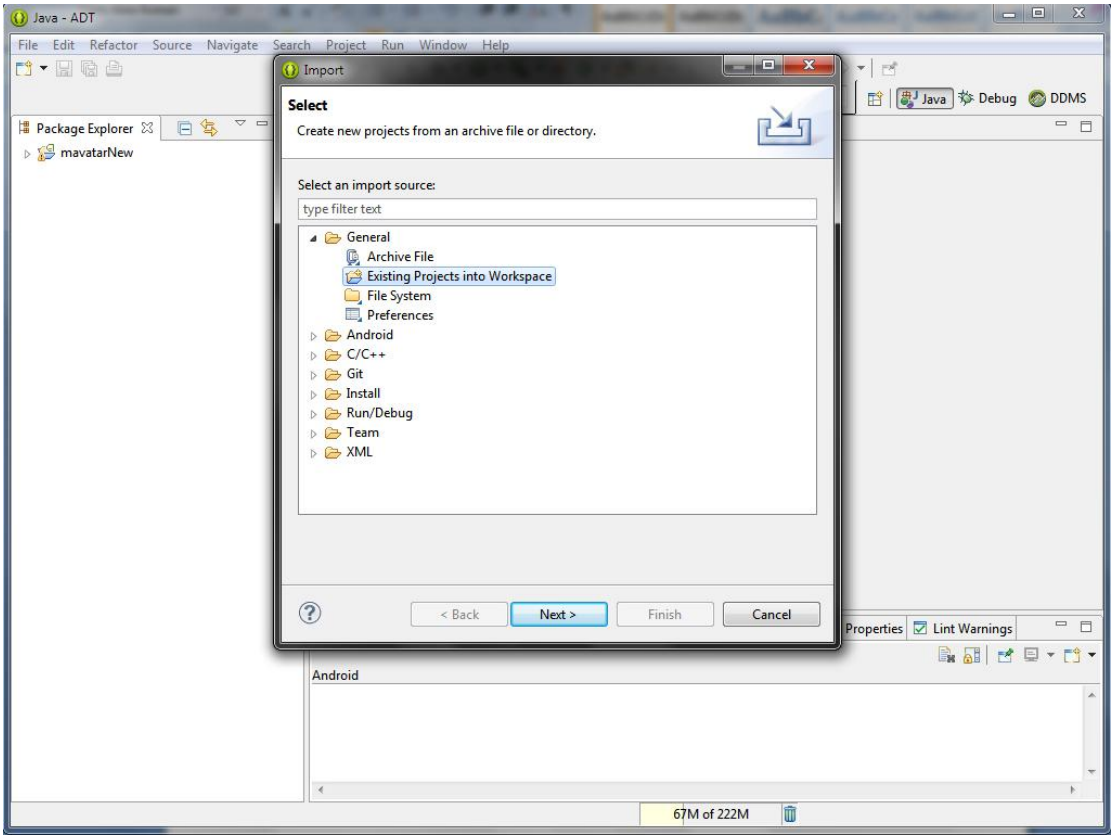
5.1.1 Εκτέλεση σε προγραμματιστικό περιβάλλον Eclipse

Αρχικά, ο ενδιαφερόμενος θα πρέπει να έχει στην κατοχή του τον φάκελο mavatarNew όπως αυτός βρίσκεται στο CD με την διπλωματική εργασία. Επίσης, θα πρέπει να έχει ήδη εγκατεστημένο στον υπολογιστή του το προγραμματιστικό περιβάλλον Eclipse. Στην συνέχεια, αφού ανοίξει το Eclipse, θα πρέπει να βρει στην καρτέλα File την επιλογή Import. Αυτό φαίνεται στην παρακάτω εικόνα:



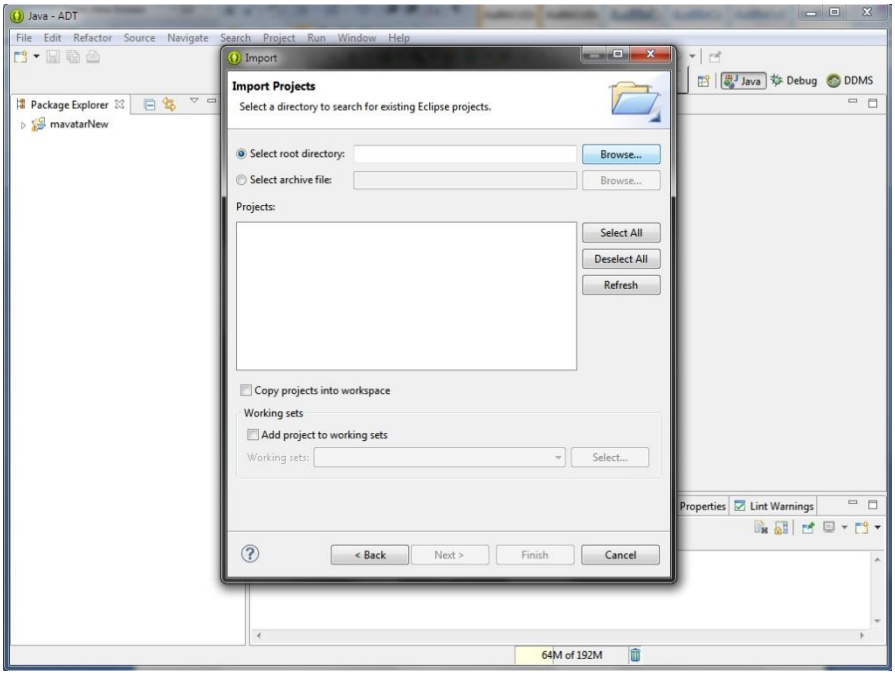
Εικόνα 5.1 : Η επιλογή Import στην καρτέλα File του Eclipse

Έπειτα, στο παράθυρο που θα εμφανιστεί, πρέπει να επιλέξει την επιλογή Existing Project Into Workspace όπως φαίνεται παρακάτω, και να πατήσει Next:



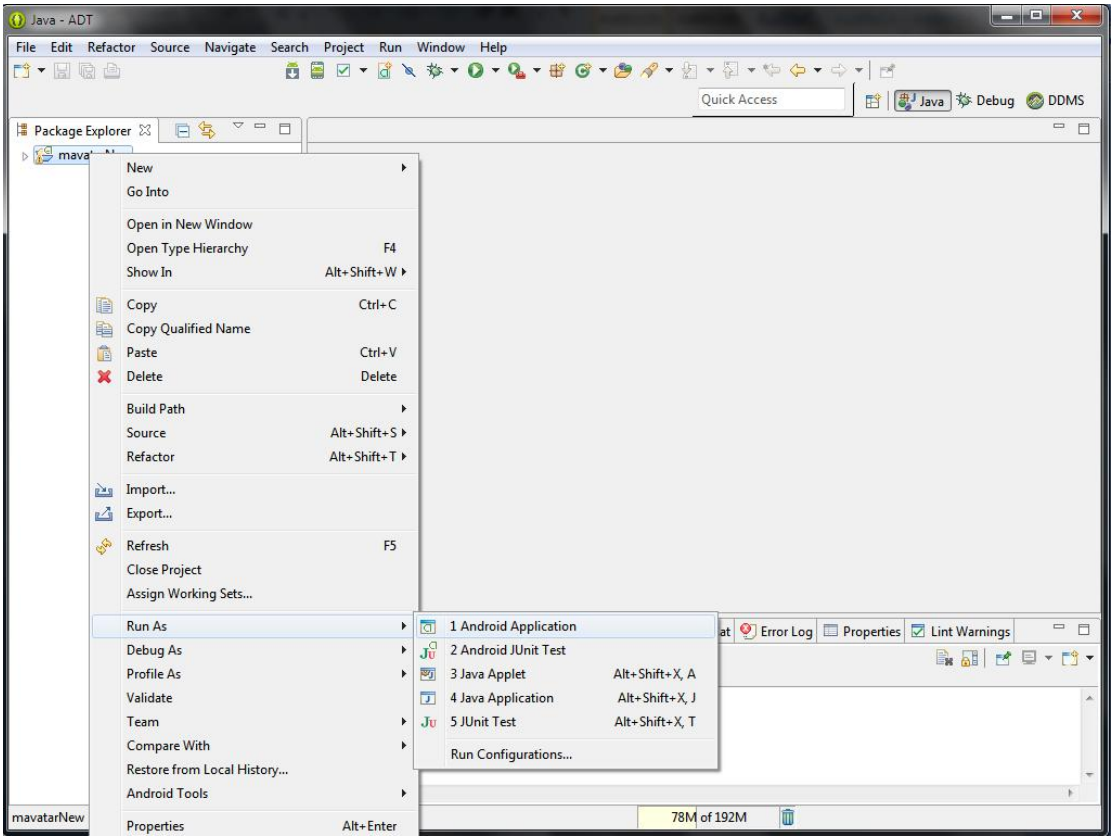
Εικόνα 5.2 : Η επιλογή Existing Project Into Workspace στο παράθυρο Import

Στο επόμενο παράθυρο που θα εμφανιστεί, στην αυτόματα επιλεγμένη επιλογή Select root directory, αρκεί να πατήσει το πλήκτρο Browse και να βρει μέσα στον υπολογιστή του το αρχείο mavatarNew της εφαρμογής. Έπειτα, αυτό θα εισαχθεί μέσα στο Eclipse. Αυτά φαίνονται παρακάτω:



Εικόνα 5.3 : Η επιλογή Select root directory και το πλήκτρο Browse στο παράθυρο Import

Αφού το project εισαχθεί στο Eclipse τότε για να εκτελεστεί, αρκεί ο χρήστης να κάνει δεξί κλικ πάνω του και να επιλέξει την επιλογή Run as – Android application. Αυτό φαίνεται παρακάτω:



Εικόνα 5.4 : Η επιλογή Run as – Android Application μετά το δεξί κλικ στην εφαρμογή

Μετά από αυτό, θα εμφανιστεί η επιλογή να διαλέξει σε ποια εικονική συσκευή επιθυμεί να εκτελέσει την εφαρμογή. Αφού κάνει την επιλογή (κατά προτίμηση συσκευή Tablet) τότε θα εκτελεστεί αυτόματα ο προσομοιωτής της συσκευής με λειτουργικό σύστημα Android (**Εικόνα 3.2**) και μετά από μερικά λεπτά θα δούμε σε λειτουργία την εφαρμογή.

5.1.2 Εκτέλεση σε συσκευή με λειτουργικό σύστημα Android

Σε κανονικές συνθήκες, αν η παρούσα εφαρμογή προοριζόταν για εμπορική χρήση, θα έπρεπε να ανέβει στο Play Store της Google και από εκεί ο κάθε χρήστης θα είχε τη δυνατότητα να την κατεβάσει στην συσκευή του. Επειδή όμως η εφαρμογή αναπτύχθηκε στα πλαίσια διπλωματικής εργασίας και δεν έχει σκοπό να βγει στο εμπόριο, θα χρησιμοποιηθεί ένας άλλος τρόπος για να μπορέσει ο χρήστης να την περάσει σε κάποια συσκευή με Android. Για να το κάνει αυτό, αρκεί να βρει το εκτελέσιμο αρχείο της εφαρμογής .apk και να το περάσει στη συσκευή. Το αρχείο αυτό μπορεί είτε να το εξάγει μέσα από το Eclipse, είτε, ευκολότερα, μετά από μια επιτυχή εκτέλεση της εφαρμογής στο Eclipse, αυτό δημιουργείται αυτόματα μέσα στον φάκελο bin/, όπως αναφέρθηκε και στην παράγραφο 4.3. Οπότε, αφού ο

ενδιαφερόμενος εκτελέσει την εφαρμογή στο προγραμματιστικό περιβάλλον Eclipse, πρέπει να ανοίξει τον φάκελο mavatarNew της εφαρμογής, να βρει τον υποφάκελο bin/ και εκεί μέσα βρίσκεται το εκτελέσιμο αρχείο **mavatarNew.apk**. Έπειτα, αρκεί να περάσει αυτό το αρχείο στη συσκευή του (πχ με κάποιο usb καλώδιο σύνδεσης με τον υπολογιστή) και με ένα κλικ να την εγκαταστήσει.

5.2 Αρχική οθόνη της εφαρμογής

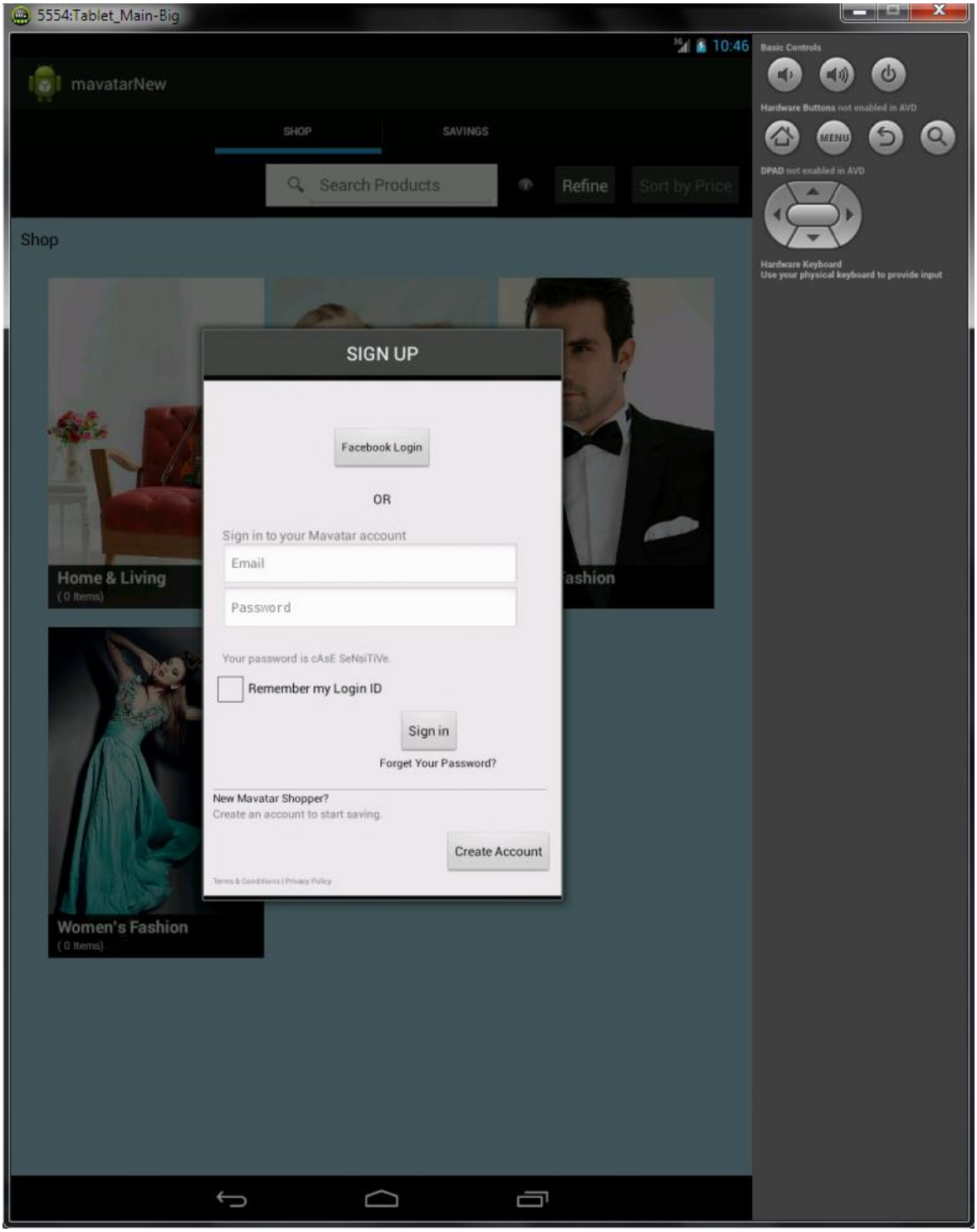
Όταν η εφαρμογή εκτελεστεί, η κλάση που ουσιαστικά εκτελείται πρώτη είναι η MainActivity.java η οποία, όπως λέει και το όνομα της, περιέχει την Main Activity της εφαρμογής. Μέσα στο αρχείο αυτό κατασκευάζονται τα 2 βασικά Tabs της εφαρμογής, το **Shop** Tab και το **Savings** Tab. Αυτό επιτυγχάνεται με τη βοήθεια της Action Bar του Android πάνω στην οποία προστίθενται τα Tabs. Ακολουθεί το αντίστοιχο κομμάτι κώδικα:

```
ActionBar bar = getSupportActionBar();
        bar.setNavigationMode(ActionBar.NAVIGATION_MODE_TABS);
        ActionBar.Tab tab1 = bar.newTab();
        ActionBar.Tab tab5 = bar.newTab();
        tab1.setText("Shop");
        tab5.setText("Savings");
        tab1.setTabListener(new MyTabListener());
        tab5.setTabListener(new MyTabListener());
        bar.addTab(tab1);
        bar.addTab(tab5);
```

Έπειτα, η πρώτη λειτουργία που επιτελείται είναι η εμφάνιση ενός pop-up παραθύρου το οποίο χρησιμοποιείται για την σύνδεση (Log In) του χρήστη στο σύστημα. Το παράθυρο αυτό είναι ένας διάλογος (dialog) τύπου Alert Dialog (αναλυτικά παράγραφος 3.2.2.3). Ο διάλογος αυτός αποκτά εστίαση (focus) και ο χρήστης δεν μπορεί να χρησιμοποιήσει καμία άλλη λειτουργία της εφαρμογής αν δεν συνδεθεί πρώτα. Ο κώδικας του παραθύρου βρίσκεται στην κλάση **LoginAlertDialog**.java μέσα στην οποία δημιουργούνται και αρχικοποιούνται όλα τα στοιχεία του παραθύρου. Η κλήση του διαλόγου από την MainActivity φαίνεται παρακάτω:

```
//login pop up
Intent i = new Intent(this, LoginAlertDialog.class);
startActivityForResult(i, 1);
```

Η αρχική οθόνη της εφαρμογής όπως αναλύθηκε παραπάνω φαίνεται στην παρακάτω **Εικόνα 5.5**.



Εικόνα 5.5 : Η αρχική οθόνη της εφαρμογής και το παράθυρο Log In.

Για να συνδεθεί στην εφαρμογή, ο χρήστης αρκεί να εισάγει το email και τον κωδικό του στα κατάλληλα πεδία όπως φαίνεται παραπάνω και να πιέσει το πλήκτρο Sign In (οι λειτουργίες για τα υπόλοιπα πλήκτρα και επιλογές δεν υλοποιήθηκαν στα πλαίσια της διπλωματικής). Τότε, η κλάση LoginAlertDialog.java ενεργοποιεί τον listener που έχει συνδεθεί με το πλήκτρο Sign In, παίρνει τα δεδομένα που εισήγαγε ο χρήστης και καλεί την κλάση

loginCall.java η οποία είναι αρμόδια για την σύνδεση με τον server. Η κλήση αυτή φαίνεται παρακάτω:

```
signinButton.setOnClickListener(new OnClickListener(){
      @Override
      public void onClick(View v) {

            String emailtext = email.getText().toString();
            String passtext = pass.getText().toString();

            loginCall lc = new loginCall(mContext,activity);
            lc.execute(emailtext,passtext);
      }
      });
```

Όπως φαίνεται, το email του χρήστη αποθηκεύεται στην μεταβλητή emailtext και ο κωδικός στην μεταβλητή passtext. Έπειτα, αυτές οι δύο μεταβλητές περνάνε ως ορίσματα στην κλήση της κλάσης loginCall.

Στην κλάση loginCall γίνεται χρήση της λειτουργίας AsyncTask η οποία επεξηγήθηκε αναλυτικά στην παράγραφο 3.2.1.5. Στην doInBackground() μέθοδο δημιουργείται η σύνδεση με τον server και κατασκευάζεται το αρχείο JSON το οποίο θα αποσταλεί σε αυτόν μέσω μιας Http σύνδεσης τύπου POST. Όλα αυτά φαίνονται στο τμήμα του κώδικα που παρατίθεται παρακάτω:

```
@Override
      protected String doInBackground(String... params) {
          try{

              HttpPost httppost = new HttpPost("https://stage-
mavatar.com/login/api");
              // Depends on your web service
              httppost.setHeader("Content-type", "application/json");
              JSONObject json = new JSONObject();
              json.put("email", params[0]);
              json.put("password",params[1]);
              StringEntity se = new StringEntity(json.toString());
                se.setContentType(new BasicHeader(HTTP.CONTENT_TYPE,
"application/json"));
                httppost.setEntity(se);
              InputStream inputStream = null;
              // create object of DefaultHttpClient
              DefaultHttpClient   httpclient = new
      DefaultHttpClient(httpParameters);

              HttpResponse response = httpclient.execute(httppost);
```

Παρατηρείται ότι στο JSON αρχείο περνάνε ως παράμετροι τα πεδία email και password οι τιμές των οποίων είναι οι παράμετροι params[0] και params[1]. Οι τιμές αυτές είναι οι τιμές που περάστηκαν στην loginCall μέσω της LoginAlertDialog.

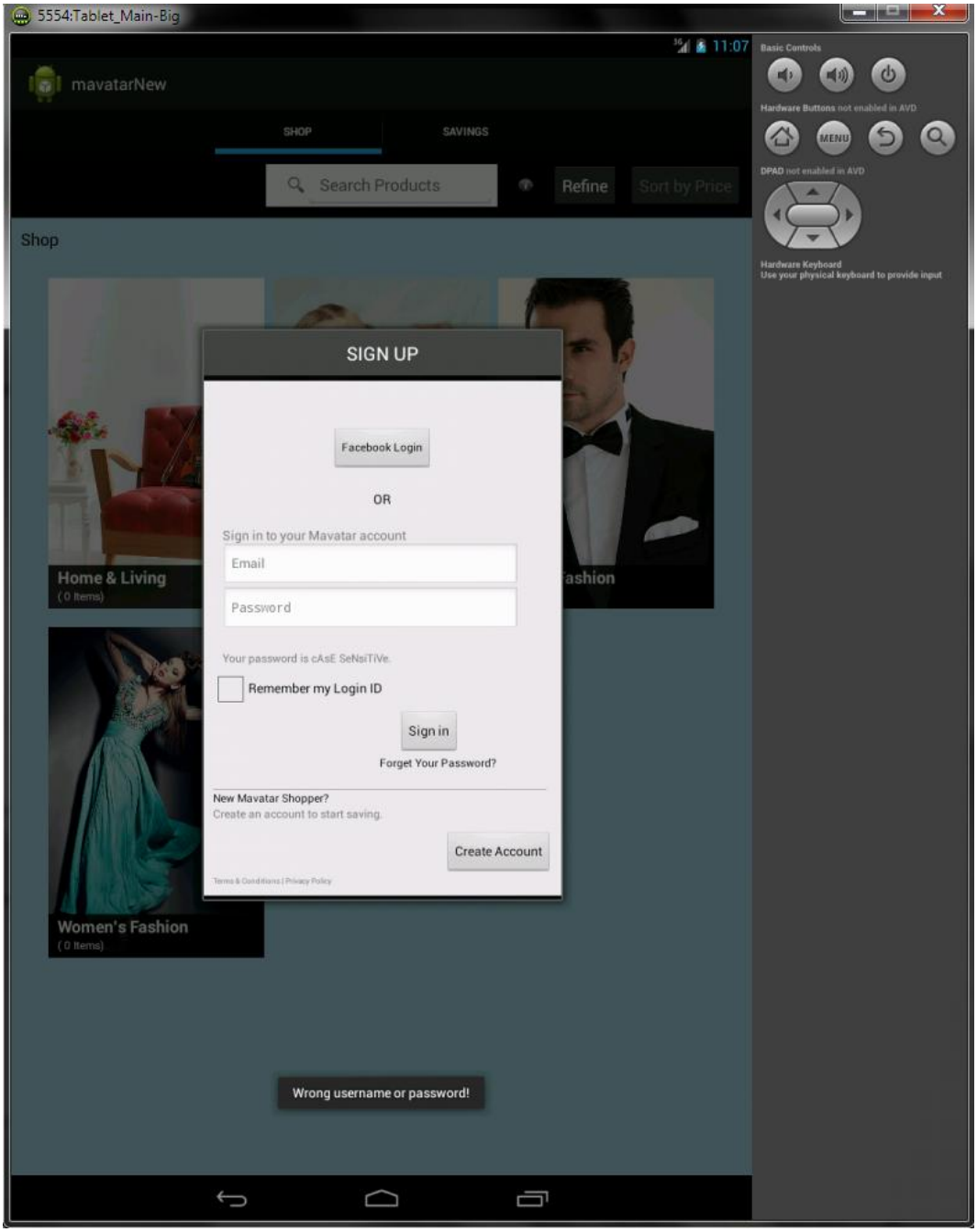
Αν ο χρήστης εισάγει σωστά δεδομένα, τότε το παράθυρο θα εξαφανιστεί αφήνοντας την εφαρμογή να εστιάσει στο Shop Tab που φαίνεται από πίσω. Αυτό ελέγχεται στην onPostExecute() μέθοδο με τον εξής απλό τρόπο:

```
if (status.equals("Success")){
                activity.finish();
}
```

Σε περίπτωση εισαγωγής λανθασμένων δεδομένων, που ελέγχεται με τον ίδιο τρόπο:

```
else{
      CharSequence text = "Wrong username or password!";
      int duration = Toast.LENGTH_LONG;
      Toast toast = Toast.makeText(mContext, text, duration);
      toast.show();
}
```

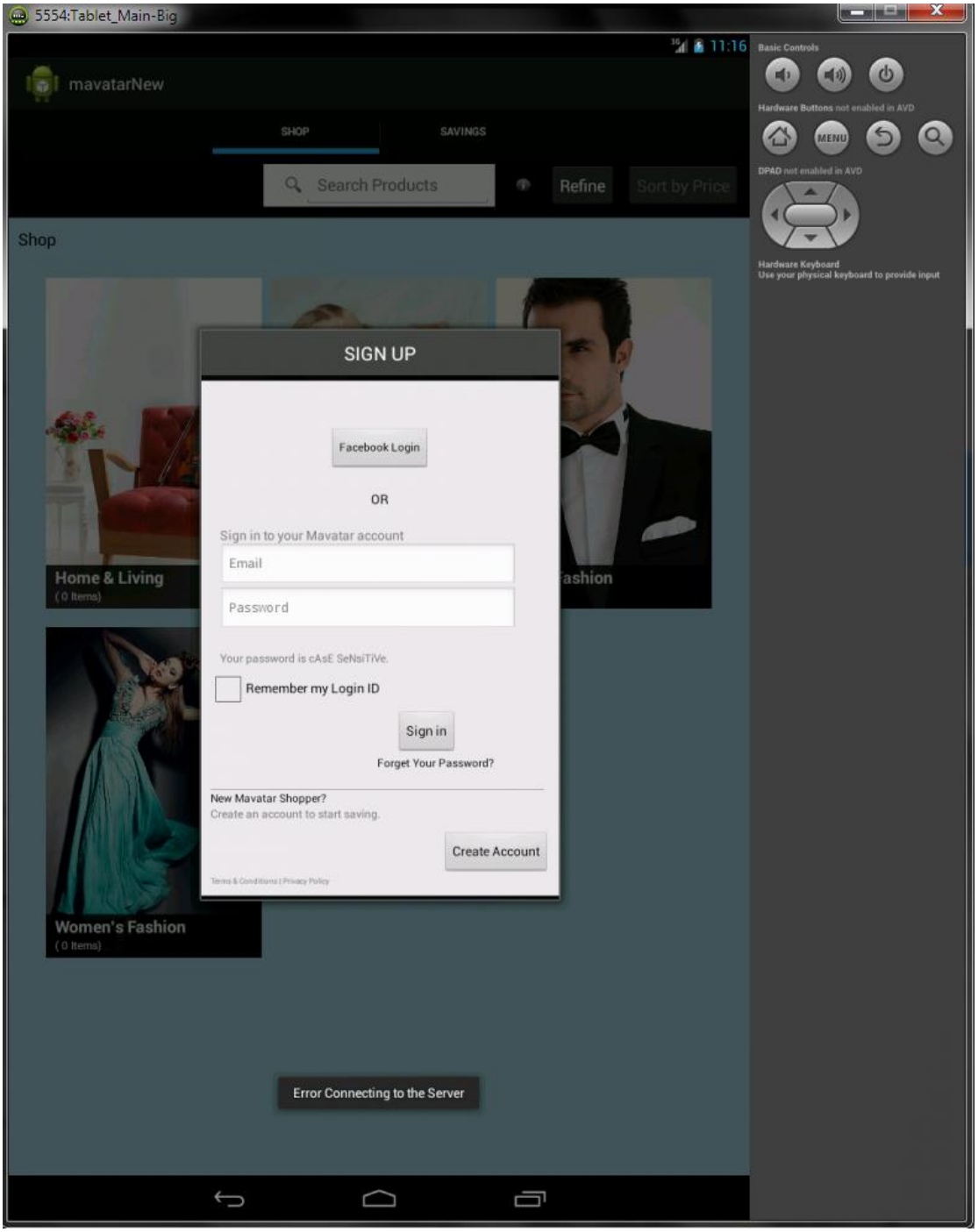
εμφανίζεται μια ειδοποίηση Toast (παράγραφος 3.2.2.5) που ενημερώνει τον χρήστη ότι εισήγαγε λάθος στοιχεία και το παράθυρο παραμένει στην θέση του αναμένοντας νέα στοιχεία. Αυτό φαίνεται στην παρακάτω **Εικόνα 5.6** όπου εισήχθησαν κενά δεδομένα.



Εικόνα 5.6 : Μήνυμα ειδοποίησης Toast σε περίπτωση εισαγωγής λάθος δεδομένων.

Σε περίπτωση που η συσκευή του χρήστη δε διαθέτει σύνδεση με το διαδίκτυο ή η σύνδεση αυτή έχει χαθεί, ο χρήστης ειδοποιείται με ακριβώς τον ίδιο τρόπο αλλά με το κατάλληλο μήνυμα. Παρακάτω φαίνεται ο αντίστοιχος κώδικας και ένα παράδειγμα στην **Εικόνα 5.7**:

```
if (result==null){
      CharSequence text = "Error Connecting to the Server";
      int duration = Toast.LENGTH_LONG;
      Toast toast = Toast.makeText(mContext, text, duration);
      toast.show();
}
```



Εικόνα 5.7 : Μήνυμα ειδοποίησης Toast σε περίπτωση απώλειας σύνδεσης με το διαδίκτυο

5.3 Το Savings Tab

5.3.1 Το SavingsFragment

Μετά την επιτυχή σύνδεση του χρήστη στην εφαρμογή ο χρήστης παρατηρεί στην οθόνη του το Shop Tab. Για την καλύτερη κατανόηση της εφαρμογής όμως, θα αναλυθεί πρώτα το Savings Tab.

Το Savings Tab όπως λέει και το όνομα του, είναι ουσιαστικά το Tab στο οποίο περιέχονται όλα τα εκπτωτικά κουπόνια που είναι διαθέσιμα για τον χρήστη. Για να περιηγηθεί σε αυτό το Tab ο χρήστης αρκεί να πατήσει πάνω στο Savings στην κύρια μπάρα της εφαρμογής. Τότε, μέσω του Main Activity καλείται το **SavingsFragment**.java το οποίο είναι ένα Fragment (αναλυτικά παράγραφος 3.2.1.1.1) το οποίο θα πάρει την θέση του Shop στην οθόνη. Αυτό γίνεται μέσω ενός TabListener που είναι αρμόδιος για τον χειρισμό γεγονότων που αφορούν τα Tabs της εφαρμογής:

```
private class MyTabListener implements ActionBar.TabListener{
      ShopFragment shopTab;
  @Override
      public void onTabSelected(Tab tab, FragmentTransaction ft) {
```

[...]

```
      else if(tab.getPosition()==1){
          SavingsFragment frag = new SavingsFragment();
          ft.replace(android.R.id.content,frag);
      }
```

Βάσει της θέσης του Tab στην μπάρα είναι εύκολη η εύρεση του Tab που επέλεξε ο χρήστης και η φόρτωση του κατάλληλο αρχείου. Όταν λοιπόν ο χρήστης επιλέξει το Savings Tab τότε το αρχείο SavingsFragment.java αναλαμβάνει τις λειτουργίες της εφαρμογής. Αρχικά, κατασκευάζει ένα GridView (παράγραφος 3.2.2.1.4) με τη βοήθεια ενός Adapter (κλάση **CouponStoreCellAdapter**.java) μέσα στο οποίο θα φορτωθούν τα καταστήματα που περιέχουν τα αντίστοιχα κουπόνια. Οι πληροφορίες για τα καταστήματα και τα κουπόνια όμως βρίσκονται στον server. Έτσι, αρχικά το GridView κατασκευάζεται άδειο και στη συνέχεια γίνεται και πάλι χρήση της λειτουργίας AsyncTask.

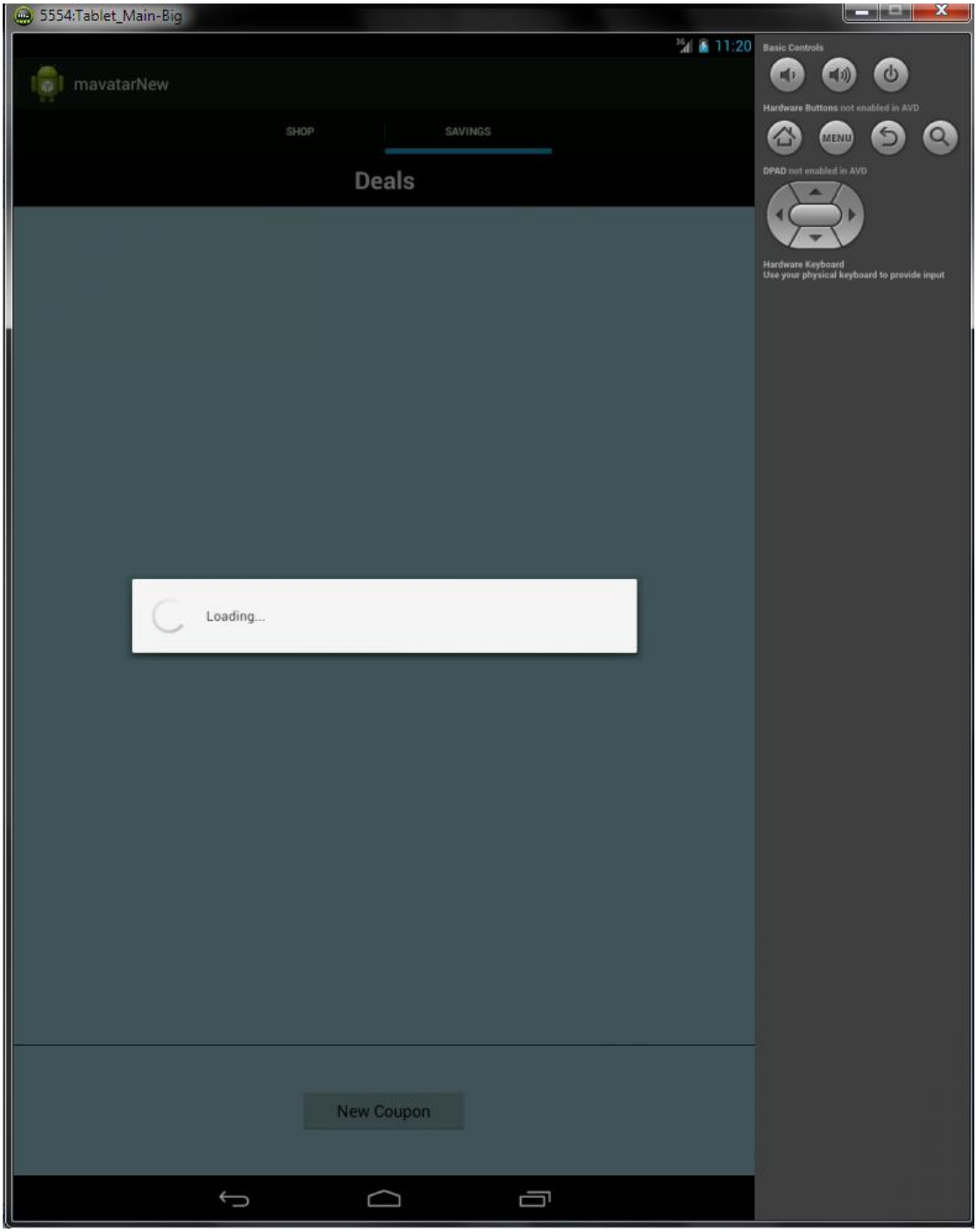
Καθώς η επικοινωνία με τον server και η επεξεργασία των δεδομένων που αυτός θα αποστείλει είναι χρονοβόρες διαδικασίες, σε όλη αυτή τη διάρκεια για να μη βλέπει ο χρήστης μια άδεια οθόνη (δηλαδή το άδειο GridView) γίνεται χρήση της μεθόδου onPreExecute(). Εκεί δημιουργείται ένας διάλογος τύπου Progress Dialog ο οποίος απλώς ενημερώνει τον χρήστη ότι η διαδικασία βρίσκεται σε εξέλιξη. Αυτό φαίνεται στην **Εικόνα 5.8**.

Έπειτα, δημιουργείται και πάλι στη μέθοδο doInBackground() μια Http σύνδεση, τύπου GET αυτή τη φορά καθώς θα γίνει λήψη δεδομένων. Όταν τα δεδομένα επιστρέψουν από τον server, είναι σε μορφή JSON. Αποθηκεύονται σε κατάλληλα HashMap και όχι σε στατικούς

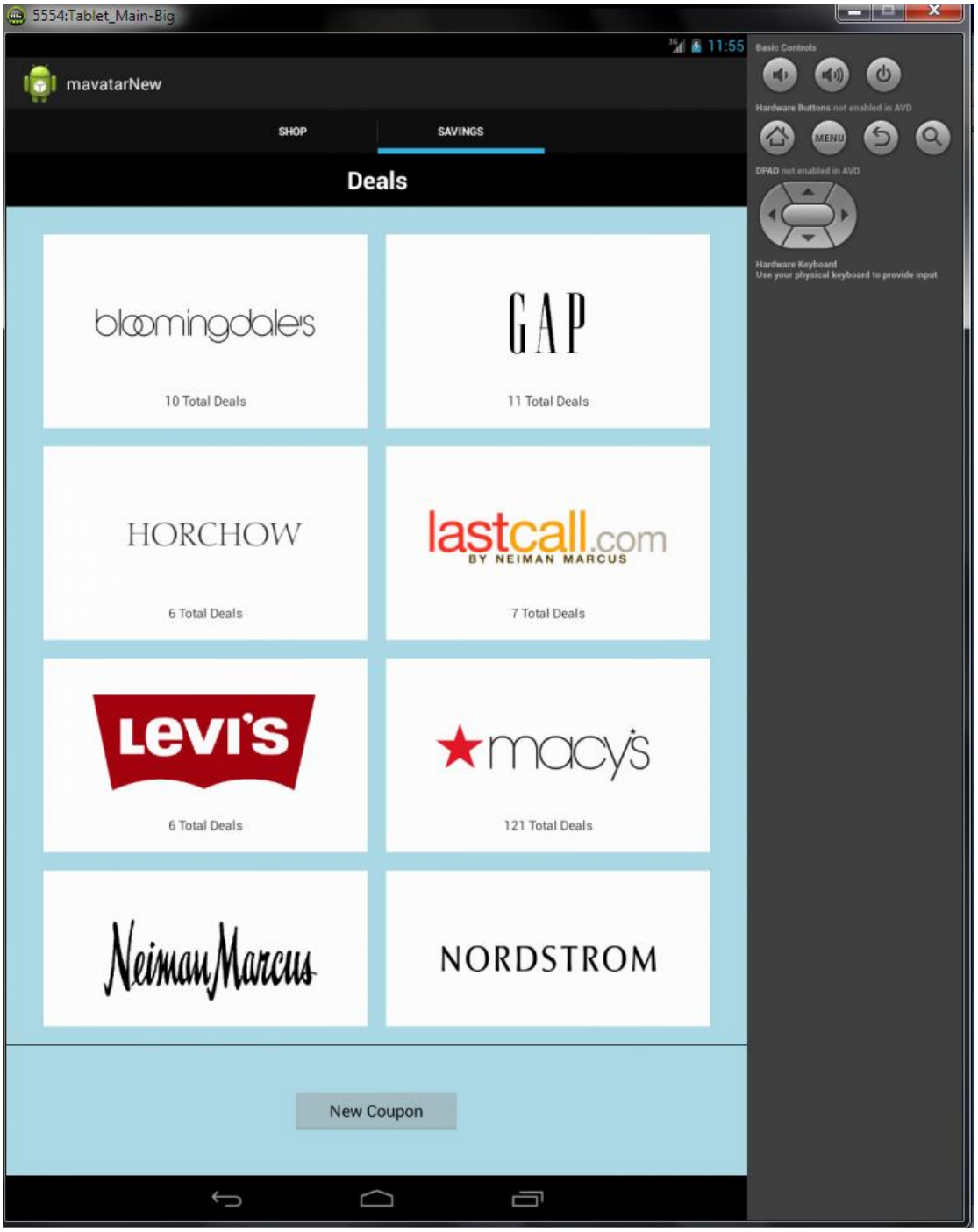
πίνακες για να γίνεται χρήση αποδοτικών δομών δεδομένων. Όταν γίνει και αυτή η επεξεργασία και αποθήκευση μπορούν να εμφανιστούν τα καταστήματα στην οθόνη.

Τότε, στην μέθοδο onPostExecute() που καλείται στο τέλος της διαδικασίας, αρχικά εξαφανίζεται ο διάλογος ProgressDialog που δημιουργήθηκε προηγουμένως και έπειτα ενημερώνεται ο Adapter ότι υπάρχουν νέα στοιχεία προς εμφάνιση ώστε να ανανεώσει την οθόνη. Αυτό φαίνεται στην **Εικόνα 5.9**. Αντίστοιχα με το παράθυρο LogIn που περιγράφηκε στην προηγούμενη παράγραφο, έτσι και εδώ υπάρχει ειδικό μήνυμα Toast για την περίπτωση απώλειας σύνδεσης με το διαδίκτυο ή σε περίπτωση που ο server δεν ανταποκρίνεται.

```
@Override
protected void onPostExecute(JSonResult result) {
      dialog.dismiss();
      if (result!=null){
          Properties.result=result;
          csca.setCoupons(result.getmain());
          csca.notifyDataSetChanged();
       }
       else{
          CharSequence text = "Error Connecting to the Server";
          int duration = Toast.LENGTH_LONG;
          Toast toast = Toast.makeText(mContext, text, duration);
          toast.show();
      }
}
```



Εικόνα 5.8 : Το Savings Tab με άδειο GridView και ένα Progress Dialog που δείχνει στον χρήστη ότι υπάρχει κάποια διαδικασία φόρτωσης σε εξέλιξη.



Εικόνα 5.9 : Το Savings Tab μετά την κλήση στον server και την ανανέωση του περιεχομένου του

Θα αναλυθεί τώρα η **Εικόνα 5.9** η οποία ουσιαστικά αποτυπώνει την πρώτη βασική οθόνη που εμφανίζεται στο Savings Tab μέσω του SavingsFragment. Παρατηρείται ότι στο πλέγμα (grid) τώρα υπάρχουν εικόνες με λογότυπα καταστημάτων και κάτω από κάθε εικόνα αναφέρεται το πλήθος των κουπονιών (Deals) που διαθέτει το κάθε κατάστημα. Επίσης, στο κάτω μέρος της οθόνης υπάρχει ένα πλήκτρο, το New Coupon. Ο χρήστης λοιπόν, τώρα μπορεί να προβεί σε δύο ενέργειες. Είτε να πατήσει σε κάποιο κατάστημα για να μπορέσει να δει τα κουπόνια που αυτό περιέχει, είτε να πατήσει το πλήκτρο New Coupon για να δημιουργήσει και να αποστείλει στον server ένα νέο κουπόνι που μπορεί να έχει στην κατοχή

του. Αρχικά, θα αναλυθεί η επιλογή ενός καταστήματος και το New Coupon θα αναλυθεί στη συνέχεια, καθώς εμφανίζεται και σε επόμενες οθόνες της εφαρμογής.

Πατώντας λοιπόν ο χρήστης σε κάποιο κατάστημα, το SavingsFragment μέσω ενός Listener αναγνωρίζει σε ποιο κελί του πλέγματος έκανε κλικ ο χρήστης και έπειτα καλεί ένα νέο Fragment να πάρει τη θέση του. Αυτό είναι ένα **CouponStoreClickedFragment** του οποίου η λειτουργία περιγράφεται στην κλάση CouponStoreClickedFragment.java.

```
gridView.setOnItemClickListener(new OnItemClickListener() {

            @Override
            public void onItemClick(AdapterView<?> parent, View v,
                                    int position, long id) {

              /* When clicked we replace our Fragment with a new one based
on the store clicked */

              CouponStoreClickedFragment frag = new
CouponStoreClickedFragment();
```

[...]

```
               FragmentManager fragmentManager = getFragmentManager();
              FragmentTransaction ft = fragmentManager.beginTransaction();
              ft.replace(android.R.id.content, frag);
              ft.addToBackStack(null);
              ft.commit();
```

Παρατηρείται ότι μαζί με την αντικατάσταση του SavingsFragment από το CouponStoreClickedFragment το πρώτο τοποθετείται στο backstack ώστε να μπορέσει ο χρήστης να επιστρέψει σε αυτό με την χρήση του Back Button.

5.3.2 Το CouponStoreClickedFragment

Το CouponStoreClickedFragment ακριβώς όπως και το προηγούμενο, αποτελείται από ένα GridView στο πλέγμα του οποίου θα φορτωθούν τα κουπόνια του αντίστοιχου καταστήματος. Αυτή τη φορά δεν χρειάζεται να γίνει σύνδεση με κάποιον server καθώς στην κλήση που έγινε στο SavingsFragment αντλήθηκαν όλες οι απαραίτητες πληροφορίες για όλα τα κουπόνια όλων των καταστημάτων. Περνώντας αυτές οι πληροφορίες, μέσω των HashMap που κατασκευάστηκαν, μέσα στο νέο Fragment, είναι πολύ εύκολο να γεμίσει το grid με τα αντίστοιχα κουπόνια. Χρησιμοποιείται και πάλι ένας Adapter (ο **CouponCellAdapter**) για να φορτώσει στο grid τα κουπόνια στην κατάλληλη μορφή. Για παράδειγμα, αν ο χρήστης κάνει κλικ στο κατάστημα Macy's τότε η νέα οθόνη που εμφανίζεται φαίνεται στην παρακάτω **Εικόνα 5.10**.

Εικόνα 5.10 : Το Savings Tab αφού ο χρήστης πάτησε πάνω στο κατάστημα Macy's.

Θα γίνει ανάλυση της παραπάνω **Εικόνας 5.10**. Σε αυτή την οθόνη, φαίνονται όλα τα Public (δημόσια) κουπόνια του καταστήματος Macy's. Παρατηρείται ότι το πλήκτρο Public φαίνεται ήδη πατημένο ώστε να δίνει στον χρήστη να καταλάβει πως βρίσκεται στην οθόνη με τα public κουπόνια. Επίσης, ο χρήστης δεν μπορεί να το αποεπιλέξει. Αυτό επιτυγχάνεται με τον παρακάτω κώδικα:

```
Button publicbutton = (Button)
getActivity().findViewById(R.id.PublicButton);
      publicbutton.setClickable(false);
      publicbutton.setPressed(true);
```

Στην γαλάζια μπάρα ο χρήστης μπορεί να δει πόσα δημόσια κουπόνια περιέχει το κατάστημα. Σε κάθε κελί του grid υπάρχει μια εικόνα με το λογότυπο του καταστήματος, μια ημερομηνία λήξης του κουπονιού και μια σύντομη περιγραφή του.

Σε αυτή την οθόνη ο χρήστης έχει αρκετές επιλογές.

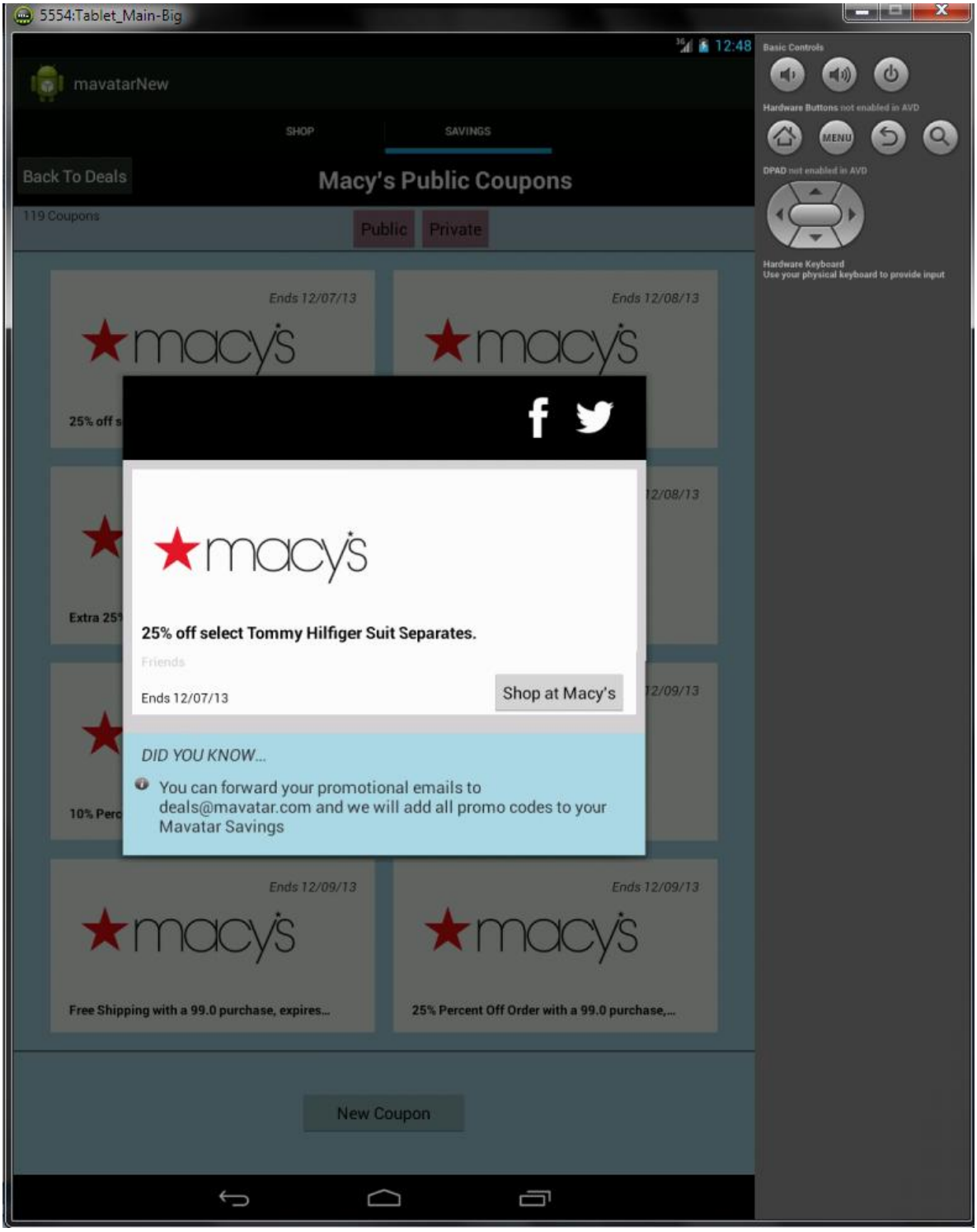
- Μπορεί να κάνει κλικ **σε κάποιο κουπόνι** ώστε να δει αναλυτικές λεπτομέρειες για αυτό
- Μπορεί να κάνει κλικ στο πλήκτρο **Private** ώστε να περιηγηθεί στα προσωπικά κουπόνια που έχει για αυτό το κατάστημα
- Μπορεί να πατήσει το πλήκτρο **Back To Deals** το οποίο θα τον επιστρέψει στην αρχική οθόνη του Savings Tab
- Μπορεί να κάνει κλικ στο πλήκτρο **New Coupon** το οποίο εκτελεί ακριβώς την ίδια λειτουργία με αυτή που είχε στην προηγούμενη οθόνη.

5.3.2.1 Πατώντας σε κάποιο κουπόνι – Το PublicCouponDialog

Αν ο χρήστης πατήσει σε κάποιο κουπόνι, τότε το CouponStoreClickedFragment μέσω ενός Listener εντοπίζει ποιο κουπόνι πάτησε ο χρήστης και καλεί ένα νέο Fragment, το **PublicCouponDialog**. Αυτή τη φορά όμως δεν γίνεται replace, δεν αντικαθίσταται δηλαδή η οθόνη με μια άλλη, αλλά απλά γίνεται show, καθώς το PublicCouponDialog είναι ένα Dialog Fragment. Είναι δηλαδή ένας διάλογος που έχει ως βάση ένα Fragment.

```
/** On Click event for Single Gridview Item **/
gridView.setOnItemClickListener(new OnItemClickListener() {

      @Override
       public void onItemClick(AdapterView<?> parent, View v,
                                int position, long id) {

          PublicCouponDialog publicdialog= new PublicCouponDialog();
[…]
           FragmentManager fm = getFragmentManager();

           publicdialog.show(fm,"public_coupon_dialog");
        }
});
```

Αν για παράδειγμα ο χρήστης πατήσει στο πρώτο κουπόνι της **Εικόνας 5.10** τότε το παράθυρο που θα ανοίξει φαίνεται παρακάτω στην **Εικόνα 5.11**.



Εικόνα 5.11 : Ο διάλογος που εμφανίζεται αν ο χρήστης πατήσει το πρώτο κουπόνι

Στην **Εικόνα 5.11** φαίνεται ο διάλογος που κατασκευάστηκε μέσω του PublicCouponDialog και ο οποίος είναι στην πραγματικότητα ένα ακόμα Fragment. Σε αυτό το παράθυρο εμφανίζεται:

- το λογότυπο του καταστήματος
- η περιγραφή του (με μαύρα bold γράμματα)

- ο κωδικός του (με ανοιχτό γκρι – Friends εδώ)
- Η ημερομηνία λήξης του κουπονιού
- Το πλήκτρο Shop At το οποίο δεν επιτελεί κάποια λειτουργία καθώς η υλοποίηση του δεν αναπτύχθηκε στα πλαίσια της διπλωματικής
- Μια ενημερωτική περιγραφή στο κάτω μέρος του παραθύρου
- Δύο εικόνες-λογότυπα, μια του Facebook και μια του Twitter, οι οποίες στην πραγματικότητα είναι πλήκτρα. Η λειτουργία τους αφορά τη σύνδεση του κουπονιού με αυτές τις πλατφόρμες social media όμως η υλοποίηση αυτή δεν αναπτύχθηκε στα πλαίσια της διπλωματικής

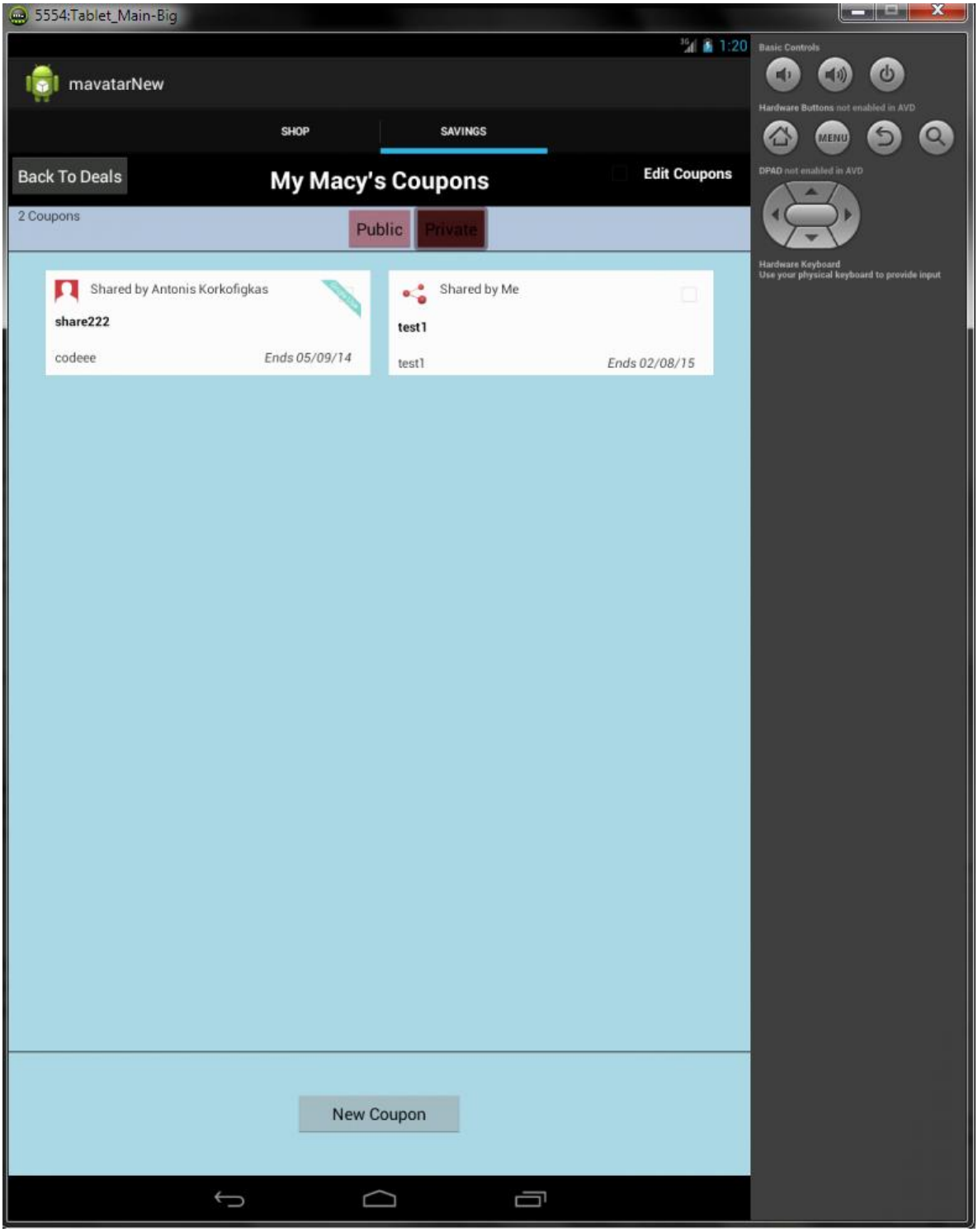
Αν ο χρήστης κάνει κλικ κάπου έξω από το παράθυρο ή αν πατήσει το πλήκτρο Back τότε το παράθυρο εξαφανίζεται.

5.3.2.2 Πατώντας στο πλήκτρο Private – Το PrivateClickedFragment

Πατώντας στο πλήκτρο Private ο χρήστης μεταφέρεται σε μια νέα οθόνη, αυτή με τα Private (προσωπικά) κουπόνια. Αυτό γίνεται για μια ακόμη φορά, με την κλήση ενός νέου Fragment, του **PrivateClickedFragment** το οποίο αντικαθιστά το CouponStoreClickedFragment. Σε αυτό το transaction το CouponStoreClickedFragment μπαίνει στο back stack και την λειτουργία αναλαμβάνει ο κώδικας της κλάσης PrivateClickedFragment.java.

```
/** On Click event for clicking the Private Button **/
privatebutton.setOnClickListener(new View.OnClickListener(){

      @Override
       public void onClick(View v) {

          PrivateClickedFragment frag = new PrivateClickedFragment();
[…]
           FragmentManager fragmentManager = getFragmentManager();
           FragmentTransaction ft = fragmentManager.beginTransaction();
           ft.replace(android.R.id.content, frag);
           ft.addToBackStack(null);
           ft.commit();
       }
});
```

Αυτό το Fragment όπως και τα προηγούμενα, περιλαμβάνει ένα GridView με ένα πλέγμα το οποία περιέχει τα προσωπικά κουπόνια του χρήστη. Αυτό γίνεται και πάλι μέσω ενός Adapter (του **PrivateCouponCellAdapter**). Η νέα οθόνη με τα Private κουπόνια φαίνεται παρακάτω στην **Εικόνα 5.12**.



Εικόνα 5.12 : Το Savings Tab αφού ο χρήστης πατήσει στο πλήκτρο Private μέσα στο κατάστημα Macy's

Παρατηρώντας την **Εικόνα 5.12** μπορεί κανείς να δει ότι η οθόνη με τα Private κουπόνια μοιάζει πάρα πολύ με την οθόνη με τα Pubic κουπόνια. Έχει όμως μερικές αλλαγές και κάποια επιπλέον χαρακτηριστικά. Αρχικά, κάποιος μπορεί εύκολα να παρατηρήσει την διαφορά στα κουπόνια που εμφανίζονται στο πλέγμα. Έχουν τελείως διαφορετική μορφή από τα Public κουπόνια. Επίσης, έχουν διαφορές ακόμα και μεταξύ τους. Θα αναλυθούν τα δύο αυτά κουπόνια που εμφανίζονται παραπάνω.

Το πρώτο κουπόνι περιέχει μια εικόνα προφίλ και δίπλα αναφέρει ότι είναι Shared από τον χρήστη Antonis Korkofigkas. Αυτό σημαίνει ότι η εικόνα προφίλ ανήκει σε αυτόν τον χρήστη και επίσης ότι το κουπόνι αυτό δεν ανήκει στον χρήστη της παρούσας εφαρμογής, αλλά του έχει γίνει share (έχει μοιραστεί) από τον χρήστη αυτόν. Επίσης, στην δεξιά πλευρά υπάρχει μια γαλάζια «κορδέλα» η οποία γράφει Single Use. Αυτό σημαίνει ότι το κουπόνι αυτό μπορεί να χρησιμοποιηθεί μόνο μια φορά. Ακόμα, φαίνεται η περιγραφή του κουπονιού με μαύρα γράμματα (share222) και ο κωδικός του (codeee). Τέλος, αναφέρεται η ημερομηνία λήξης του κουπονιού.

Το δεύτερο κουπόνι περιέχει μια κόκκινη εικόνα με 3 μπάλες (η οποία γενικότερα στο διαδίκτυο υποδηλώνει το sharing) και αναφέρει ότι είναι Shared By Me. Αυτό σημαίνει ότι το κουπόνι ανήκει στον χρήστη της παρούσας εφαρμογής και ότι το μοιράζεται (αν δεν το μοιραζόταν, εμφανίζεται η ίδια εικόνα με τις 3 μπάλες, σε γκρι χρώμα). Δεν υπάρχει επισήμανση Single Use που σημαίνει ότι το κουπόνι μπορεί να χρησιμοποιηθεί παραπάνω από μια φορές. Τα υπόλοιπα δεδομένα είναι ίδια με το προηγούμενο (περιγραφή, κωδικός, ημερομηνία λήξης).

Σε αυτή την οθόνη ο χρήστης έχει τις εξής επιλογές.

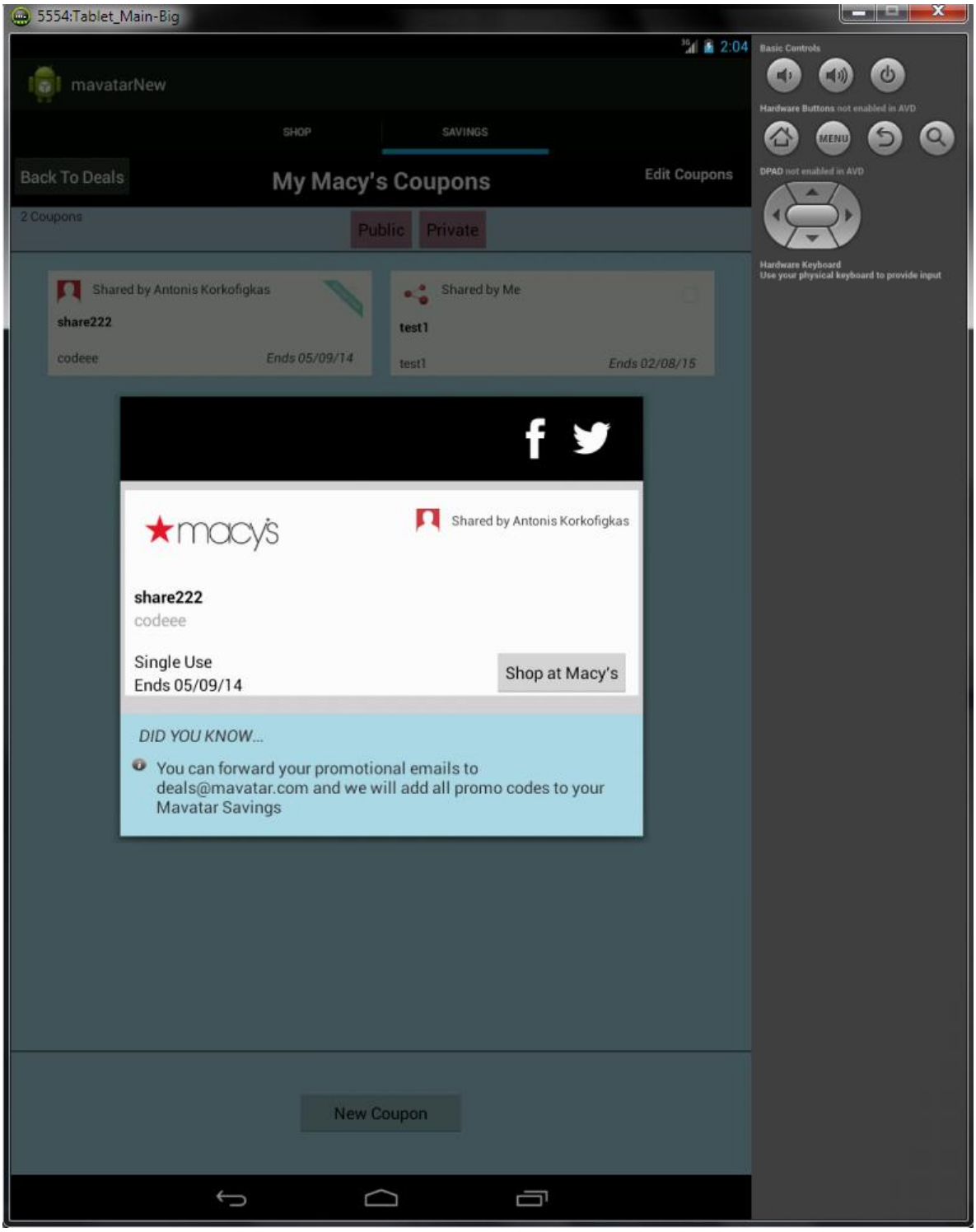
- Μπορεί να κάνει κλικ σε κάποιο κουπόνι ώστε να δει αναλυτικές λεπτομέρειες για αυτό
- Μπορεί να κάνει κλικ στο πλήκτρο Public ώστε να επιστρέψει στα Public κουπόνια
- Μπορεί να πατήσει το πλήκτρο Back To Deals το οποίο θα τον επιστρέψει στην αρχική οθόνη του Savings Tab
- Μπορεί να κάνει κλικ στο πλήκτρο New Coupon
- Μπορεί να κάνει κλικ στην επιλογή Edit Coupons ώστε να επεξεργαστεί τα κουπόνια του

5.3.2.2.1 Πατώντας σε κάποιο κουπόνι – Το PrivateCouponDialog

Ακριβώς με τον ίδιο τρόπο που χρησιμοποιήθηκε ένα Dialog Fragment για την εμφάνιση ενός Public κουπονιού (παράγραφος 5.3.2.1), έτσι και εδώ όταν ο χρήστης πατήσει σε κάποιο κουπόνι, το PrivateClickedFragment μέσω ενός Listener βρίσκει ποιο κουπόνι πατήθηκε και το εμφανίζει σε ένα νέο παράθυρο Dialog Fragment μέσω της κλάσης **PrivateCouponDialog**.

```
/** On Click event for Single Gridview Item **/
gridView.setOnItemClickListener(new OnItemClickListener() {
      @Override
      public void onItemClick(AdapterView<?> parent, View v,
                                          int position, long id) {
                           PrivateCouponDialog privatedialog= new
PrivateCouponDialog();
[…]
                           FragmentManager fm = getFragmentManager();

                           privatedialog.show(fm,"private_coupon_dialog");
      }
});
```

Αν για παράδειγμα ο χρήστης πατήσει στο πρώτο κουπόνι, το παράθυρο που δημιουργείται φαίνεται στην παρακάτω **Εικόνα 5.13**.



Εικόνα 5.13 : Ο διάλογος που εμφανίζεται αν ο χρήστης πατήσει το πρώτο Private κουπόνι

Το παράθυρο είναι σχεδόν πανομοιότυπο με το παράθυρο για τα Public κουπόνια. Σε αυτό το παράθυρο εμφανίζεται:

- το λογότυπο του καταστήματος

- Η εικόνα προφίλ του χρήστη από τον οποίο το μοιραζόμαστε μαζί με το κατάλληλο κείμενο Shared by και το όνομα του χρήστη
- η περιγραφή του (με μαύρα bold γράμματα)
- ο κωδικός του (με ανοιχτό γκρι – codeee εδώ)
- Αν είναι Single Use αυτό επισημαίνεται (όπως εδώ)
- Η ημερομηνία λήξης του κουπονιού
- Το πλήκτρο Shop At το οποίο δεν επιτελεί κάποια λειτουργία καθώς η υλοποίηση του δεν αναπτύχθηκε στα πλαίσια της διπλωματικής
- Μια ενημερωτική περιγραφή στο κάτω μέρος του παραθύρου
- Δύο εικόνες-λογότυπα, μια του Facebook και μια του Twitter, οι οποίες στην πραγματικότητα είναι πλήκτρα. Η λειτουργία τους αφορά τη σύνδεση του κουπονιού με αυτές τις πλατφόρμες social media όμως η υλοποίηση αυτή δεν αναπτύχθηκε στα πλαίσια της διπλωματικής

Αν ο χρήστης κάνει κλικ κάπου έξω από το παράθυρο ή αν πατήσει το πλήκτρο Back τότε το παράθυρο εξαφανίζεται.

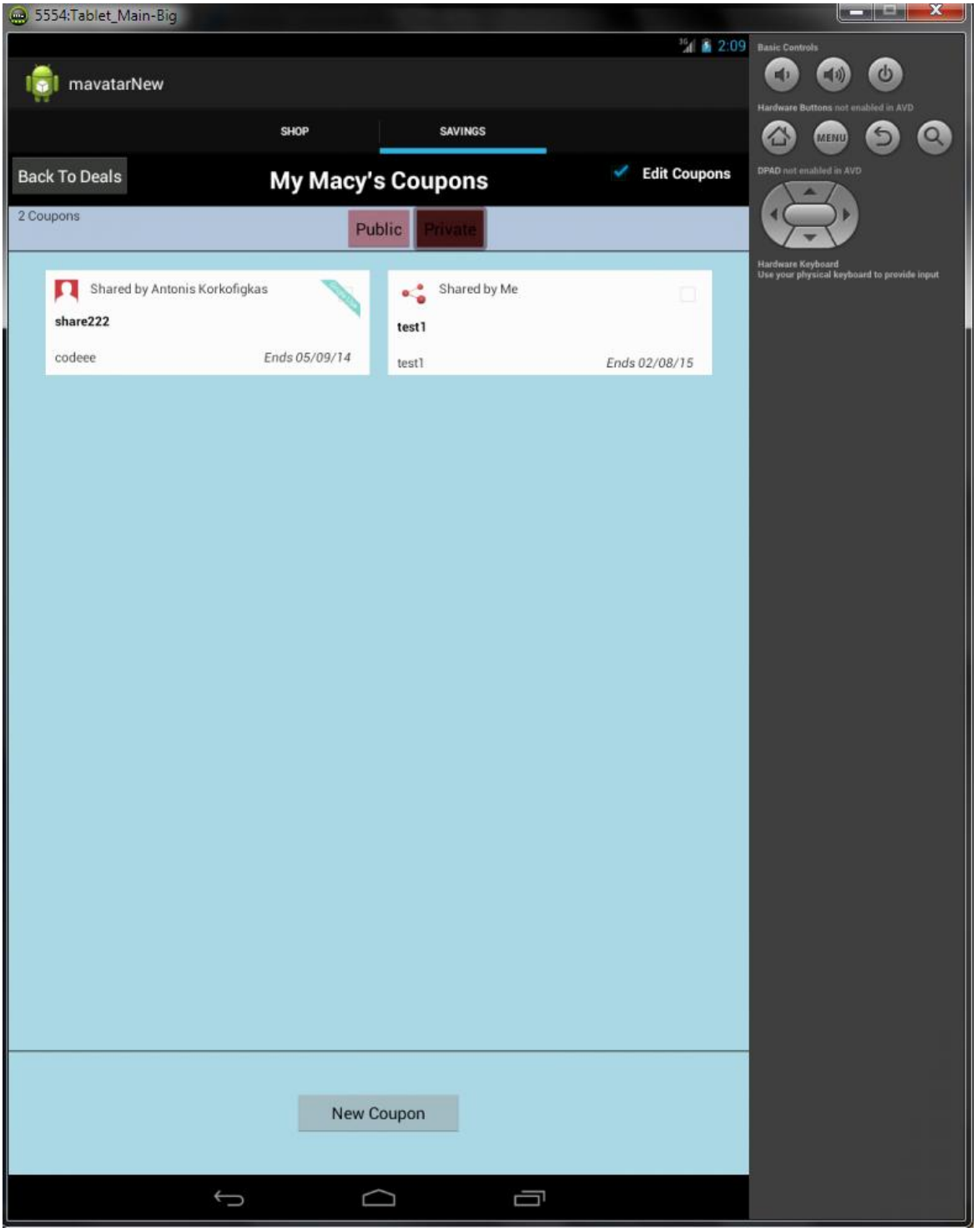
5.3.2.2.2 Πατώντας την επιλογή Edit Coupons – Η Context Action Bar

Αν ο χρήστης πατήσει πάνω στην επιλογή Edit Coupons τότε ενεργοποιείται μια ειδική λειτουργία. Αυτή της επεξεργασίας των κουπονιών. Μέσω αυτής της λειτουργίας ο χρήστης έχει τη δυνατότητα να επιλέξει πολλαπλά κουπόνια (δυνατότητα πολυεπιλογής - multichoice mode) και να τα διαγράψει από την εφαρμογή (και τον server αντίστοιχα).

Όταν ο χρήστης ενεργοποιήσει τη λειτουργία, τότε δίπλα στο Edit Coupons εμφανίζεται ένα γαλάζιο «τσεκ» ώστε να ενημερωθεί ο χρήστης ότι η λειτουργία είναι σε χρήση. Αυτό φαίνεται στην **Εικόνα 5.14**. Αντίστοιχα, αν ο χρήστης ξαναπατήσει στο Edit Coupons το «τσεκ» εξαφανίζεται και η λειτουργία σταματάει.

Όταν η λειτουργία είναι σε ισχύ, τότε ο χρήστης μπορεί να επιλέξει τα κουπόνια που θέλει να διαγράψει. Όταν πατήσει πάνω σε κάποιο κουπόνι συμβαίνουν δύο πράγματα. Πρώτον, ένα πράσινο «τσεκ» εμφανίζεται στο κουπόνι για να ενημερώσει το χρήστη ότι το κουπόνι έχει επιλεγεί. Δεύτερον, στο πάνω μέρος της εφαρμογής εμφανίζεται η Context Action Bar (παράγραφος 3.2.2.4). Αυτό φαίνεται στην **Εικόνα 5.15**. Επίσης, όσο η λειτουργία είναι ενεργή, ο χρήστης δεν μπορεί να πατήσει στα πλήκτρα Public, Private, Back To Deals παρά μόνο αν απενεργοποιήσει την λειτουργία:

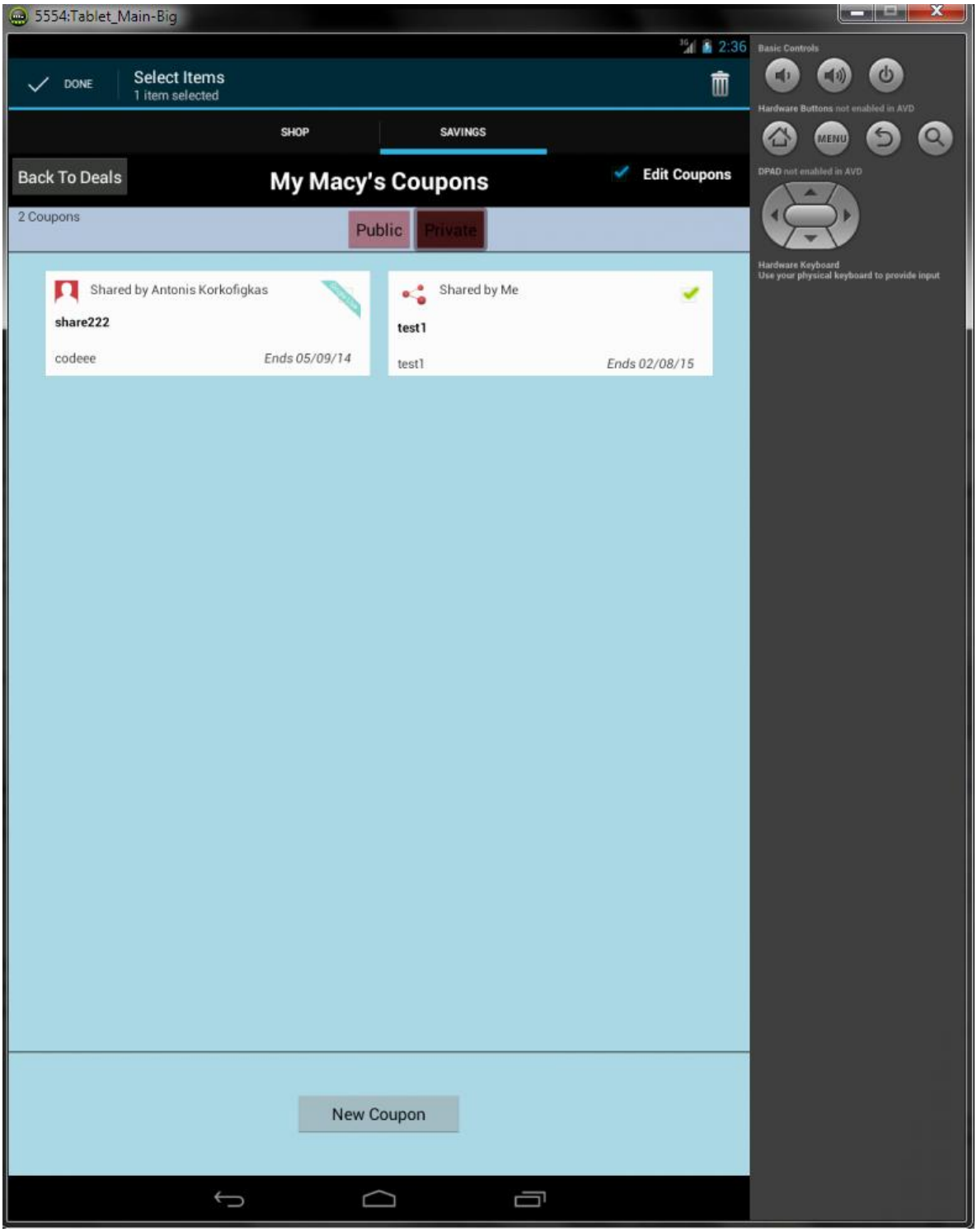
```
if (editbox.isChecked()){
      privatebutton.setClickable(false);
      publicbutton.setClickable(false);
      dealsbutton.setClickable(false);
```



Εικόνα 5.14 : Ο χρήστης ενεργοποιεί την λειτουργία Edit Coupons

Ο κώδικας που ενεργοποιεί την Context Action Bar:

```
gridView.setOnItemClickListener(new OnItemClickListener() {
  @Override
  public void onItemClick(AdapterView<?> parent, View v,
                                          int position, long id) {
      editbox.setClickable(false);
      gridView.setChoiceMode(GridView.CHOICE_MODE_MULTIPLE_MODAL);
      gridView.setMultiChoiceModeListener(new MultiChoiceModeListener(){
```



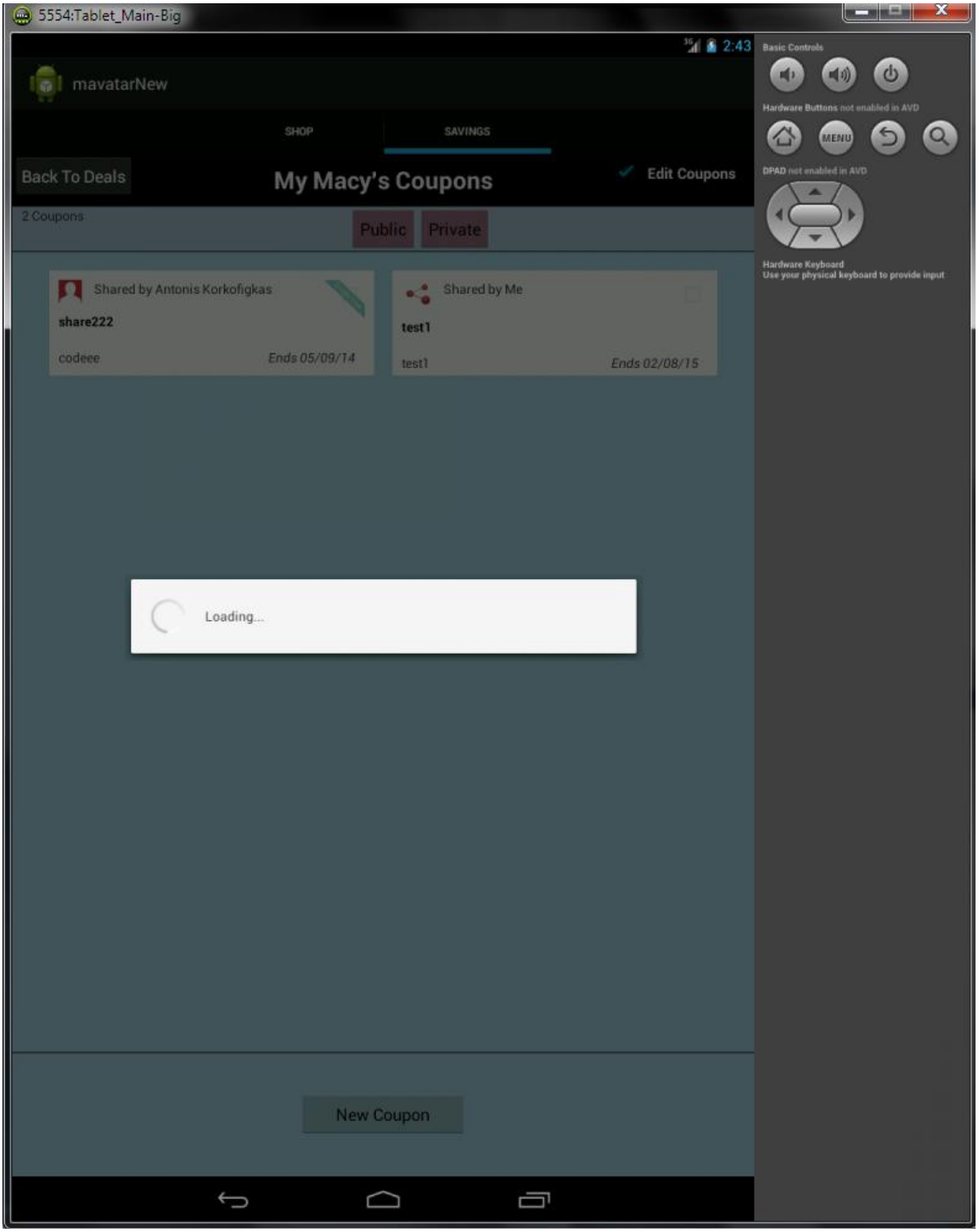
Εικόνα 5.15 : Ο χρήστης επιλέγει ένα κουπόνι κατά την λειτουργία Edit Coupons

Η Context Action Bar δείχνει τα εξής. Κάτω από τον τίτλο Select Items φαίνονται πόσα κουπόνια είναι επιλεγμένα. Στην παραπάνω περίπτωση είναι μόνο ένα γι αυτό και φαίνεται "1 item selected". Αν ήταν δύο, θα έλεγε 2 items κ.ο.κ. Επίσης, η Context Action Bar έχει δύο πλήκτρα. Το πρώτο βρίσκεται στα αριστερά και γράφει DONE μαζί με ένα μεγάλο «τσεκ». Αν ο χρήστης το πατήσει, τότε αποεπιλέγονται όλα τα κουπόνια χωρίς να διαγραφούν, η

Context Action Bar εξαφανίζεται, αλλά η λειτουργία Edit Coupons παραμένει ενεργή. Το δεύτερο και βασικό κουμπί είναι η εικόνα με τον κάδο που βρίσκεται στη δεξιά πλευρά. Αν ο χρήστης πατήσει σε αυτή την εικόνα-πλήκτρο τότε τα κουπόνια που έχει επιλέξει θα διαγραφούν. Για την ακρίβεια, το PrivateClickedFragment καλεί την κλάση RemoveCouponRequest.java η οποία είναι αρμόδια για την διαγραφή των κουπονιών.

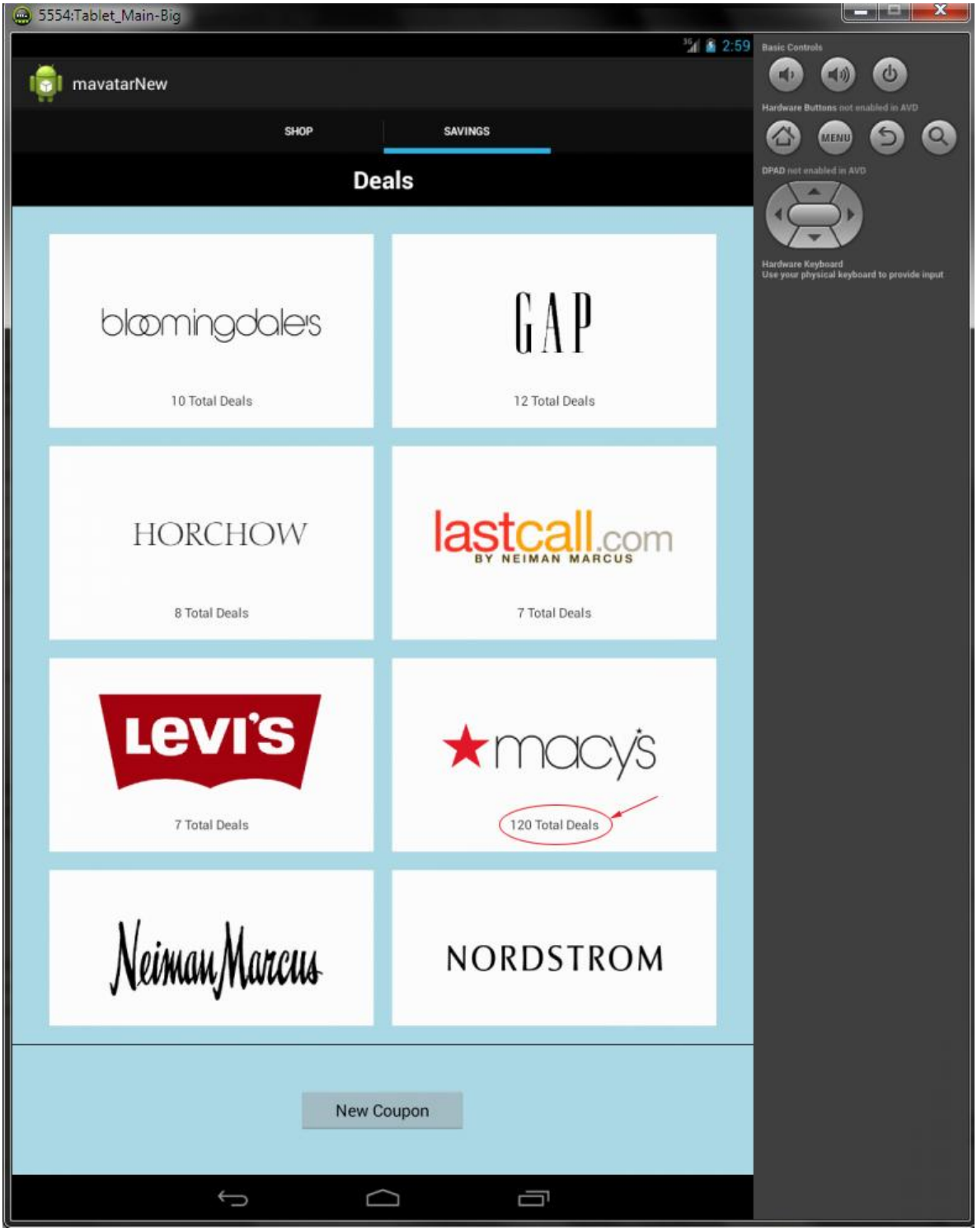
```
RemoveCouponRequest req = new RemoveCouponRequest(mContext,activity,fm);
req.execute(mymap.get("personal").get("id"));
```

Η κλάση αυτή χρησιμοποιεί την λειτουργία AsyncTask η οποία έχει περιγραφεί πολλές φορές μέχρι τώρα. Δημιουργεί μια Http σύνδεση τύπου GET στην διάρκεια της οποίας αναλαμβάνει να εμφανίσει ένα Progress Dialog στην οθόνη. Αυτό φαίνεται στην **Εικόνα 5.16** όπου ο χρήστης επέλεξε να διαγράψει το προηγουμένως επιλεγμένο κουπόνι.



Εικόνα 5.16 : Ο χρήστης επέλεξε να διαγράψει το κουπόνι. Εμφανίζεται ένα Progress Dialog που τον ενημερώνει για την εξέλιξη της διαδικασίας.

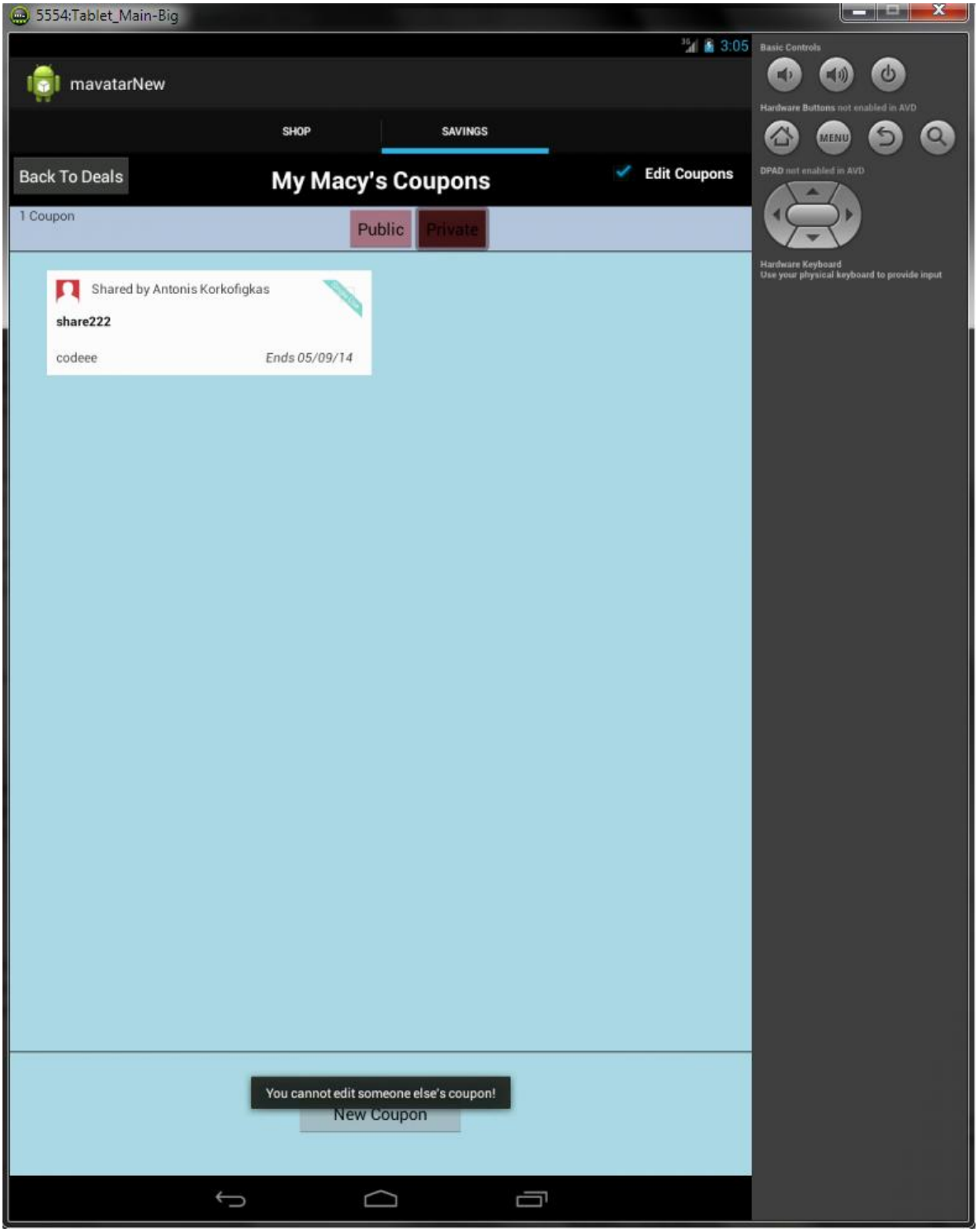
Μόλις η διαγραφή ολοκληρωθεί, η εφαρμογή μας επιστρέφει αυτόματα στην αρχική σελίδα του Savings Tab. Αυτό φαίνεται στην **Εικόνα 5.17**.



Εικόνα 5.17 : Η αρχική σελίδα του Savings Tab μετά τη διαγραφή

Αν δει κάποιος την παραπάνω εικόνα και την συγκρίνει με την **Εικόνα 5.9** μπορεί εύκολα να παρατηρήσει ότι ο αριθμός των συνολικών κουπονιών του καταστήματος Macy's μειώθηκε κατά 1. Αυτό το 1 είναι και το κουπόνι που μόλις διαγράφηκε. Δηλαδή ο server ενημερώθηκε κανονικά και διέγραψε από τη βάση δεδομένων του το κουπόνι. Έπειτα, όταν το SavingsFragment έκανε την νέα κλήση στον server το κουπόνι είχε πλέον διαγραφεί.

Όσον αφορά την λειτουργία Edit Coupons, πρέπει να σημειωθεί μια ακόμα περίπτωση. Αν ο χρήστης προσπαθήσει να επιλέξει ένα κουπόνι που δεν ανήκει σε αυτόν, τότε εμφανίζεται ειδικό μήνυμα Toast που τον ενημερώνει ότι δεν μπορεί να επεξεργαστεί κάποιο κουπόνι που δεν του ανήκει. Αυτό φαίνεται παρακάτω στην **Εικόνα 5.18**.



Εικόνα 5.18 : Μήνυμα Toast που ενημερώνει τον χρήστη ότι δε μπορεί να επεξεργαστεί κουπόνι που δεν του ανήκει

Αξίζει επίσης να αναφερθεί ότι το Android δεν προσφέρει κάποιο layout το οποίο να εμφανίζει και να εξαφανίζει τα «τσεκ» πάνω σε επιλεγμένα αντικείμενα. Έτσι,

κατασκευάστηκε ένα νέο layout, το CheckableRelativeLayout.java το οποίο ουσιαστικά επεκτείνει την λειτουργία ενός απλού RelativeLayout (παράγραφος 3.2.2.1.2) και προσθέτει την λειτουργία της πολυεπιλογής. Αυτό γίνεται μέσω μια λίστας με αντικείμενα της κλάσης Checkable και κατάλληλων μεθόδων που εξετάζουν αν ένα αντικείμενο είναι επιλεγμένο (checked) ή όχι και ενεργοποιούν της αντίστοιχες εικόνες «τσεκ». Ακολουθεί ένα μικρό μόνο κομμάτι της κλάσης αυτής:

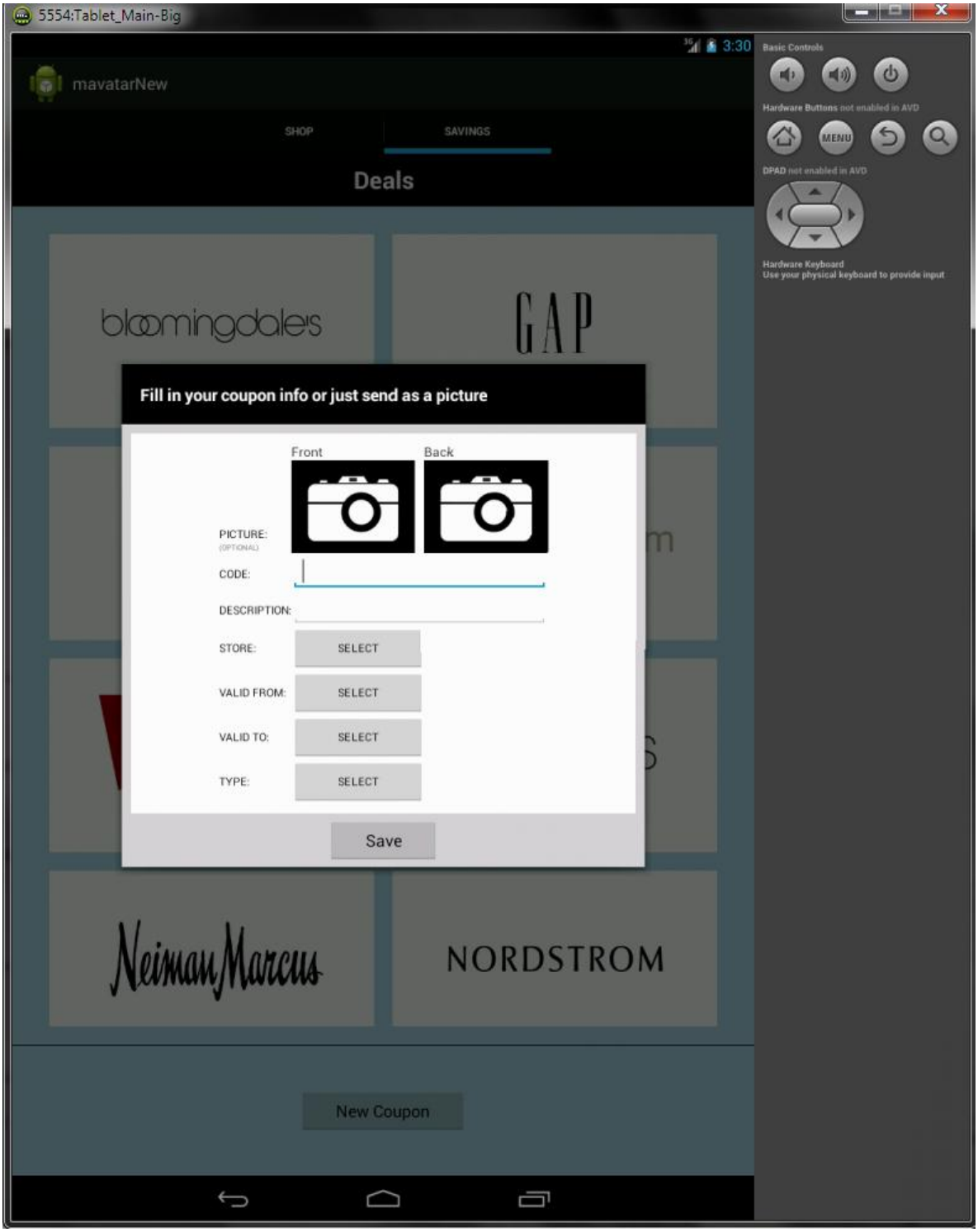
```
* Extension of a relative layout to provide a checkable behaviour */
public class CheckableRelativeLayout extends RelativeLayout implements
        Checkable {

    private boolean isChecked;
    private List<Checkable> checkableViews;
[…]
    public void toggle() {
        this.isChecked = !this.isChecked;
        for (Checkable c : checkableViews) {
            c.toggle();
        }
    }
```

5.3.2.3 Πατώντας στο πλήκτρο New Coupon

Όταν ο χρήστης πατήσει στο πλήκτρο New Coupon (ανεξάρτητα από ποια οθόνη θα το κάνει), θα εμφανιστεί ένα νέο παράθυρο-διάλογος με το οποίο ο χρήστης μπορεί να δημιουργήσει στον server ένα νέο κουπόνι που έχει στην κατοχή του. Το Fragment στο οποίο βρίσκεται ο χρήστης εκείνη τη στιγμή θα καλέσει την κλάση **NewCouponDialog** η οποία υλοποιεί και τον διάλογο, ο οποίος για μια ακόμη φορά είναι ένα Dialog Fragment.

```
/** On Click event for clicking the New Coupon Button **/
newcouponbutton.setOnClickListener(new View.OnClickListener(){
      @Override
      public void onClick(View v) {
          NewCouponDialog newdialog= new NewCouponDialog();
          fm = getFragmentManager();
           newdialog.show(fm,"new_coupon_dialog");
       }
});
```

Το παράθυρο για τη δημιουργία νέου κουπονιού φαίνεται παρακάτω στην **Εικόνα 5.19**.



Εικόνα 5.19 : Το παράθυρο που εμφανίζεται με το πάτημα του New Coupon.

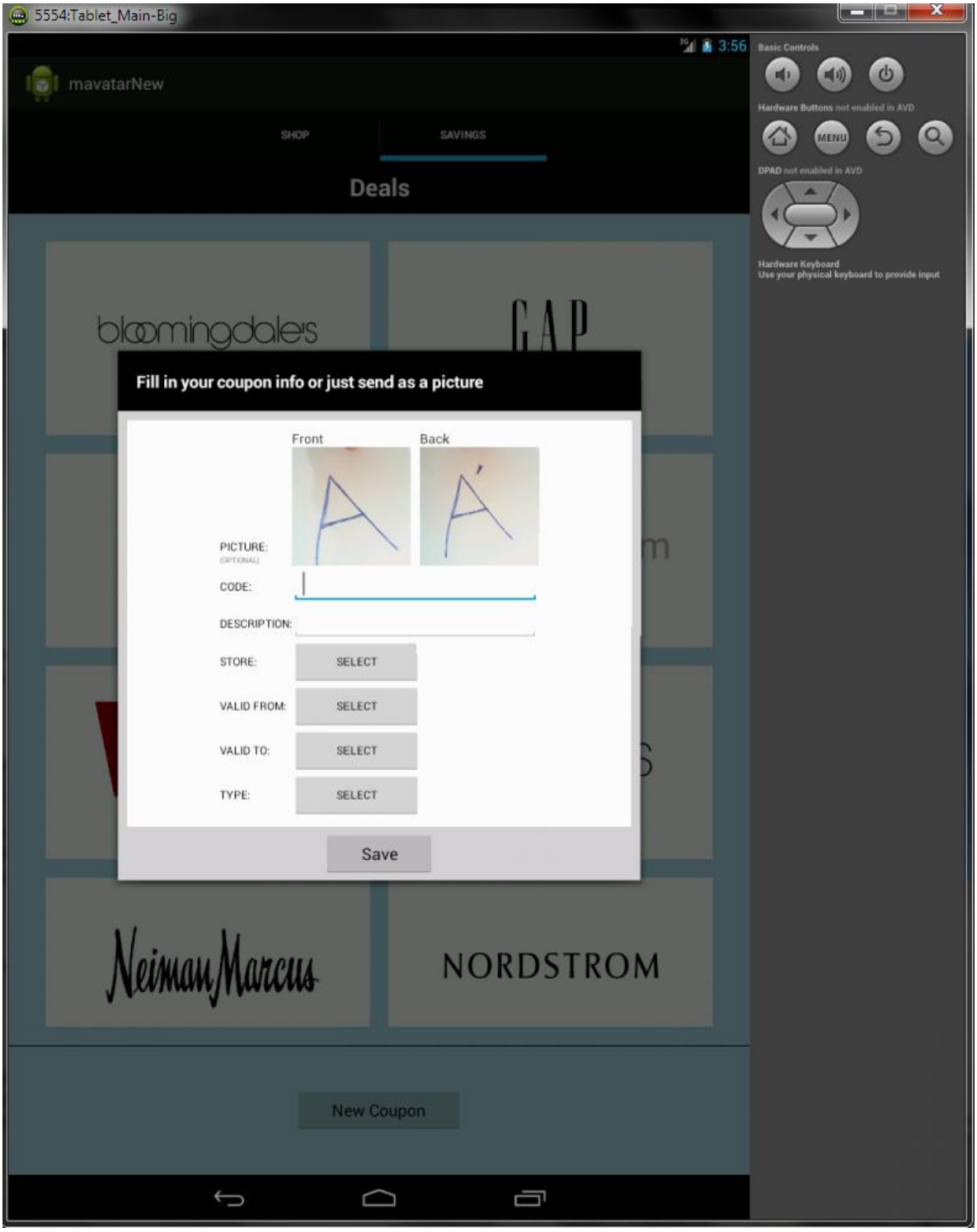
Το παράθυρο – διάλογος για τη δημιουργία ενός νέου κουπονιού, ουσιαστικά περιέχει όλα εκείνα τα πεδία που είναι απαραίτητα για την αναγνώριση ενός κουπονιού. Ο χρήστης, καλείται είτε να τα συμπληρώσει όλα, είτε να τραβήξει μια φωτογραφία με την κάμερα της συσκευής του και να την στείλει στον server.

Όταν ο χρήστης πατήσει σε ένα από τα πλήκτρα Front ή Back, τότε ενεργοποιείται αυτόματα η κάμερα της συσκευής. Αυτό ουσιαστικά είναι ένα νέο Activity το οποίο ενεργοποιείται με

την χρήση intent. Πριν από αυτό όμως, η εφαρμογή αρχικά ελέγχει αν υπάρχει διαθέσιμη κάρτα εξωτερικής αποθήκευσης ώστε να αποθηκευτεί εκεί η φωτογραφία που θα τραβηχτεί. Αν όχι, εμφανίζεται κατάλληλο μήνυμα.

```
/** On Click event for clicking the Front Picture Button **/
frontpicturebutton.setOnClickListener(new View.OnClickListener(){

      @Override
       public void onClick(View v) {

          boolean mExternalStorageAvailable = false;
          boolean mExternalStorageWriteable = false;
          String state = Environment.getExternalStorageState();

          if (Environment.MEDIA_MOUNTED.equals(state)) {
              // We can read and write the media
              mExternalStorageAvailable = mExternalStorageWriteable = true;
          } else if (Environment.MEDIA_MOUNTED_READ_ONLY.equals(state)) {
              // We can only read the media
              mExternalStorageAvailable = true;
              mExternalStorageWriteable = false;
          } else {
                 // Something else is wrong. It may be one of many other
states, but all we need
                 //  to know is we can neither read nor write
                 mExternalStorageAvailable = mExternalStorageWriteable =
false;
          }

          if(mExternalStorageAvailable && mExternalStorageWriteable ){

              Intent intent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
              imageflagfront=1;
              fileUri = getOutputMediaFileUri(MEDIA_TYPE_IMAGE,
"MavatarPictureFront"); // create a file to save the image
              intent.putExtra(MediaStore.EXTRA_OUTPUT, fileUri); // set the
image file name
              intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
              intent.setFlags(Intent.FLAG_GRANT_READ_URI_PERMISSION);

              // start the image capture Intent
              getActivity().startActivityForResult(intent,
CAPTURE_IMAGE_ACTIVITY_REQUEST_CODE);

          }
          else{
              CharSequence text = "SD Storage Unavailable!";
              int duration = Toast.LENGTH_LONG;
              Toast toast = Toast.makeText(mContext, text, duration);
              toast.show();
          }

        }
});
```

Στην παρακάτω **Εικόνα 5.20** φαίνεται το παράθυρο μετά τη λήψη και των δύο φωτογραφιών.

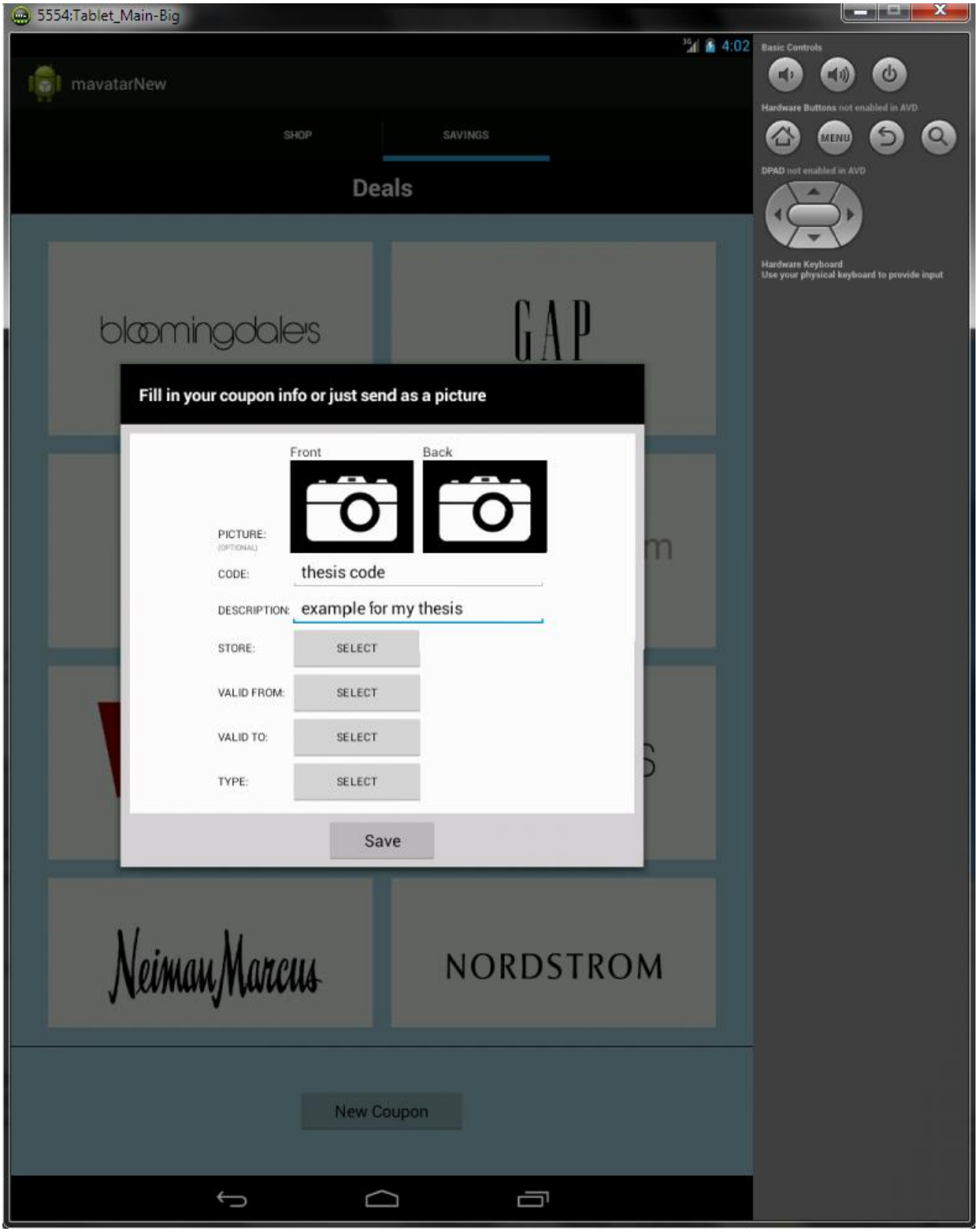


Εικόνα 5.20: Το παράθυρο New Coupon μετά τη λήψη των δύο φωτογραφιών.

Παρατηρείται ότι οι δύο φωτογραφίες αντικατέστησαν τα πλήκτρα Front και Back. Αυτό επιτεύχθηκε με την δημιουργία δύο thumbnail (προεπισκοπήσεις) των φωτογραφιών και την τοποθέτηση τους ως εικόνες στη θέση των αρχικών εικόνων των πλήκτρων. Παρ' όλα αυτά, τα πλήκτρα είναι ακόμα ενεργά και αν ο χρήστης πατήσει σε ένα thumbnail θα ενεργοποιηθεί εκ νέου η κάμερα. Τώρα, αν ο χρήστης πατήσει στο πλήκτρο Save η φωτογραφία θα σταλεί

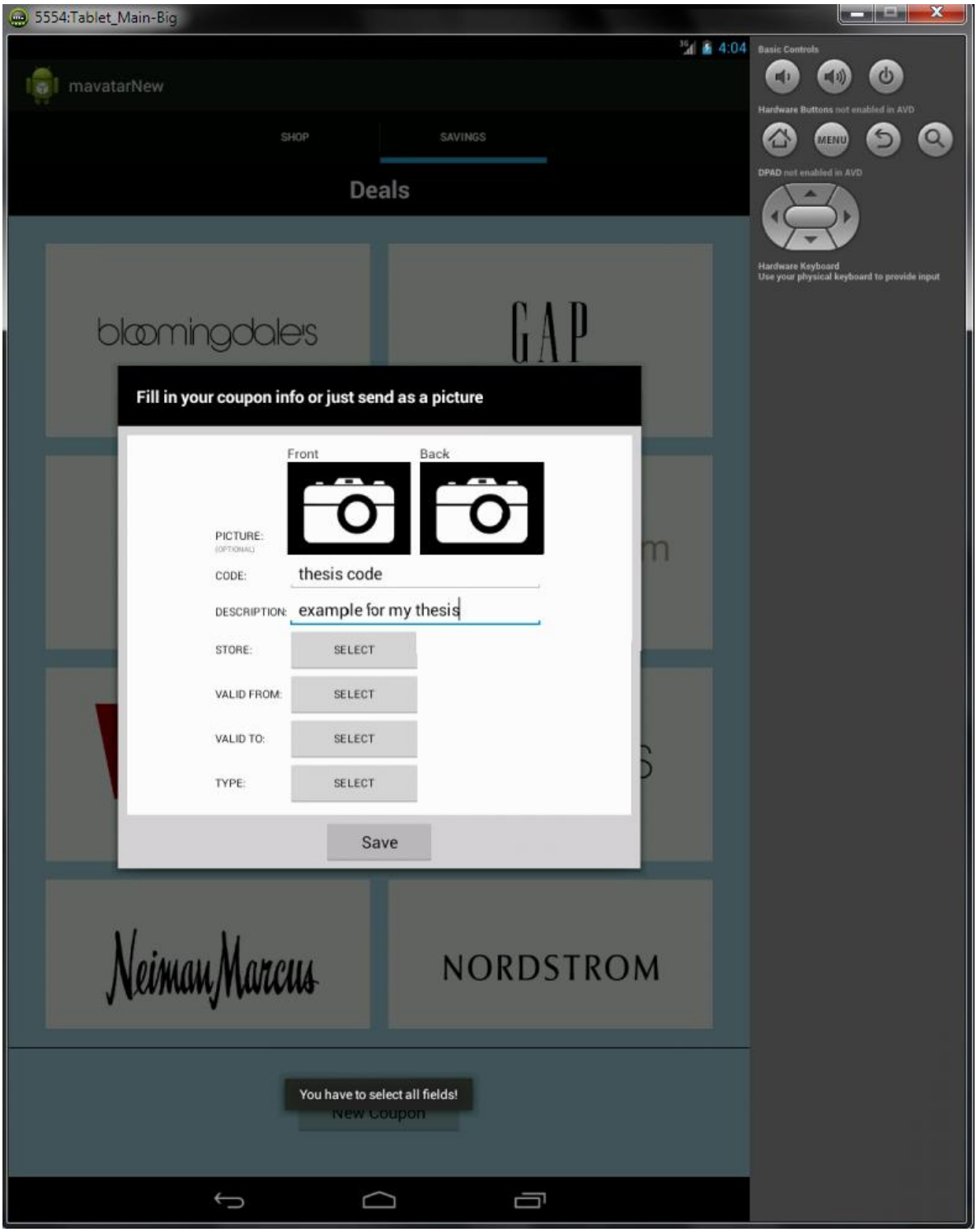
στον server (αυτό το τελευταίο σκέλος δεν πραγματοποιήθηκε στα πλαίσια της διπλωματικής).

Αν τώρα ο χρήστης επιλέξει να εισάγει χειροκίνητα τα στοιχεία του κουπονιού, θα πρέπει να συμπληρώσει ένα ένα τα πεδία που ζητούνται. Αρχικά, θα πρέπει να συμπληρώσει τα πεδία Code και Description. Αυτό φαίνεται στην παρακάτω **Εικόνα 5.21**.



Εικόνα 5.21: Ο χρήστης έχει εισάγει τιμές για τα πρώτα δύο πεδία στο παράθυρο New Coupon.

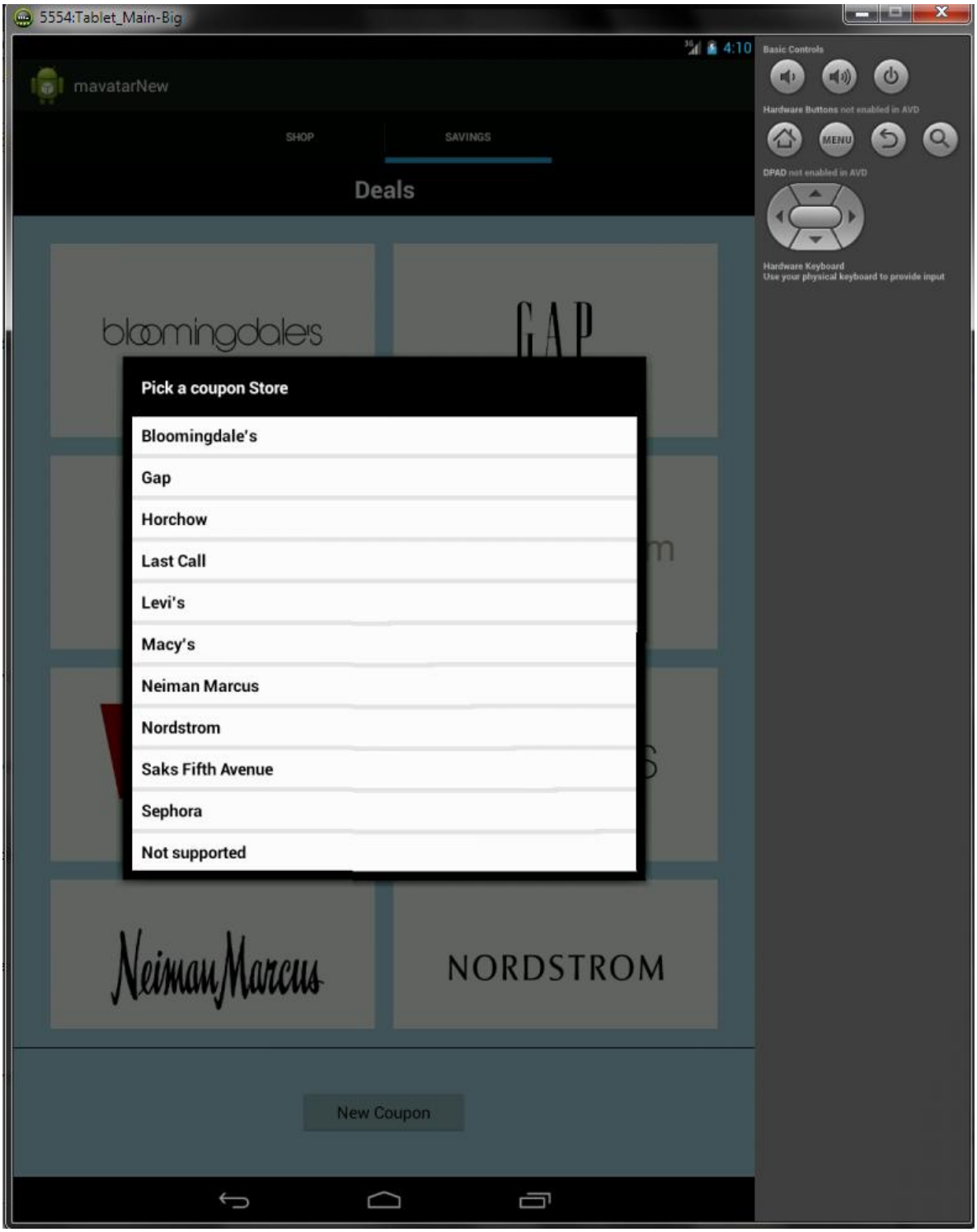
Αν ο χρήστης τώρα προσπαθήσει να δημιουργήσει το κουπόνι με ελλιπή στοιχεία, η εφαρμογή εμφανίζει κατάλληλο μήνυμα. Ένα παράδειγμα φαίνεται στην **Εικόνα 5.22**.



Εικόνα 5.21: Κατάλληλο μήνυμα Toast όταν ο χρήστης προσπαθεί να δημιουργήσει κουπόνι με ελλιπή στοιχεία.

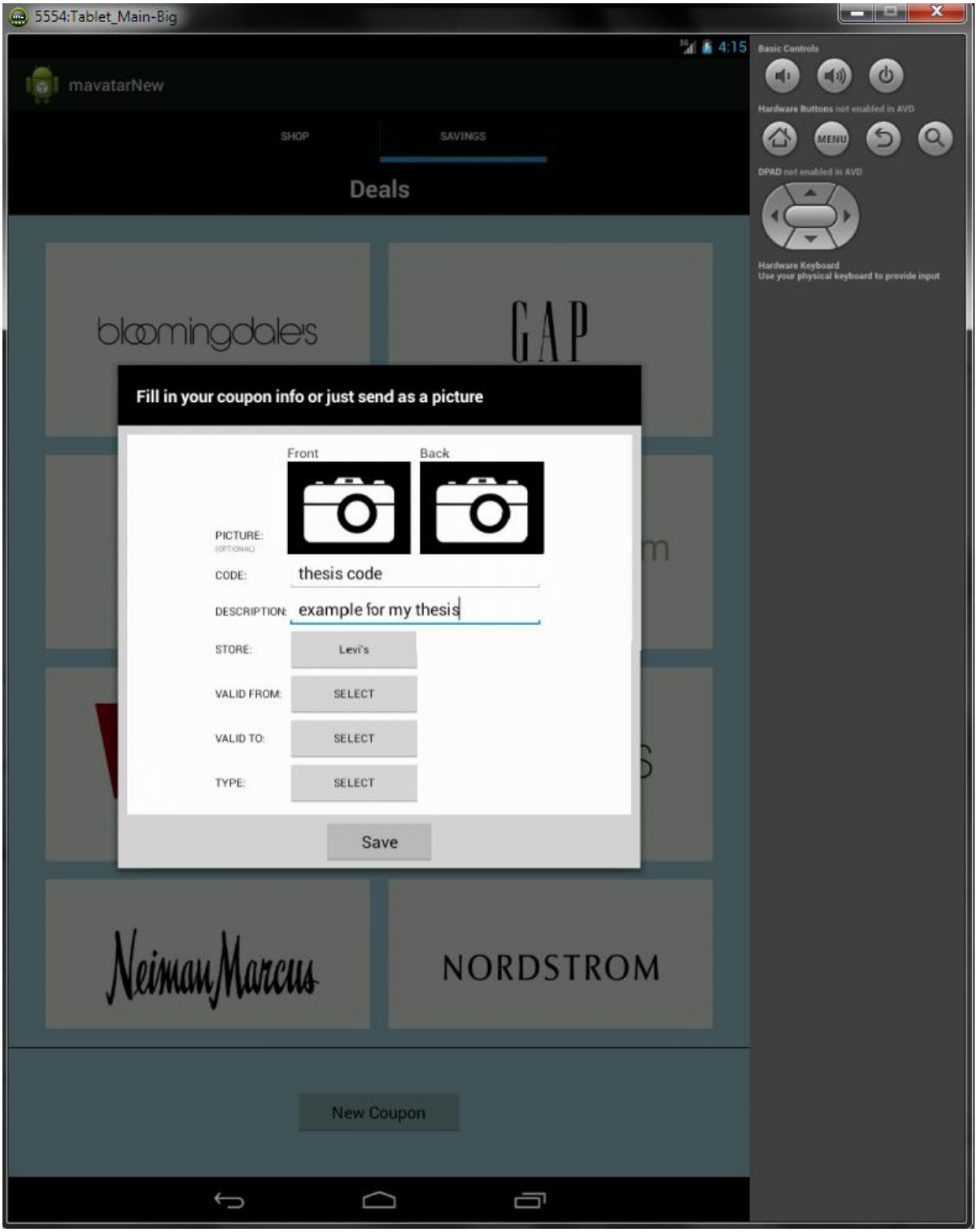
Όταν ο χρήστης πατήσει στο πλήκτρο SELECT στην γραμμή STORE εμφανίζεται ένα νέο παράθυρο στο οποίο πρέπει να επιλέξει σε ποιο κατάστημα ανήκει το κουπόνι. Αυτό το παράθυρο είναι ένα ακόμα Dialog Fragment το **SelectStoreDialog**. Το παράθυρο αποτελείται

από μια λίστα με τα ονόματα όλων των διαθέσιμων καταστημάτων. Αυτό φαίνεται στην **Εικόνα 5.22**.



Εικόνα 5.22: Το παράθυρο SelectStoreDialog στο οποίο ο χρήστης καλείται να επιλέξει ένα από τα διαθέσιμα καταστήματα.

Όταν ο χρήστης επιλέξει το κατάστημα που επιθυμεί, το παράθυρο εξαφανίζεται αυτόματα και το κατάστημα που επέλεξε εμφανίζεται μέσα στο πλήκτρο που προηγουμένως έγραφε SELECT. Αν για παράδειγμα ο χρήστης επιλέξει το κατάστημα Levi's το αποτέλεσμα φαίνεται στην **Εικόνα 5.23**.

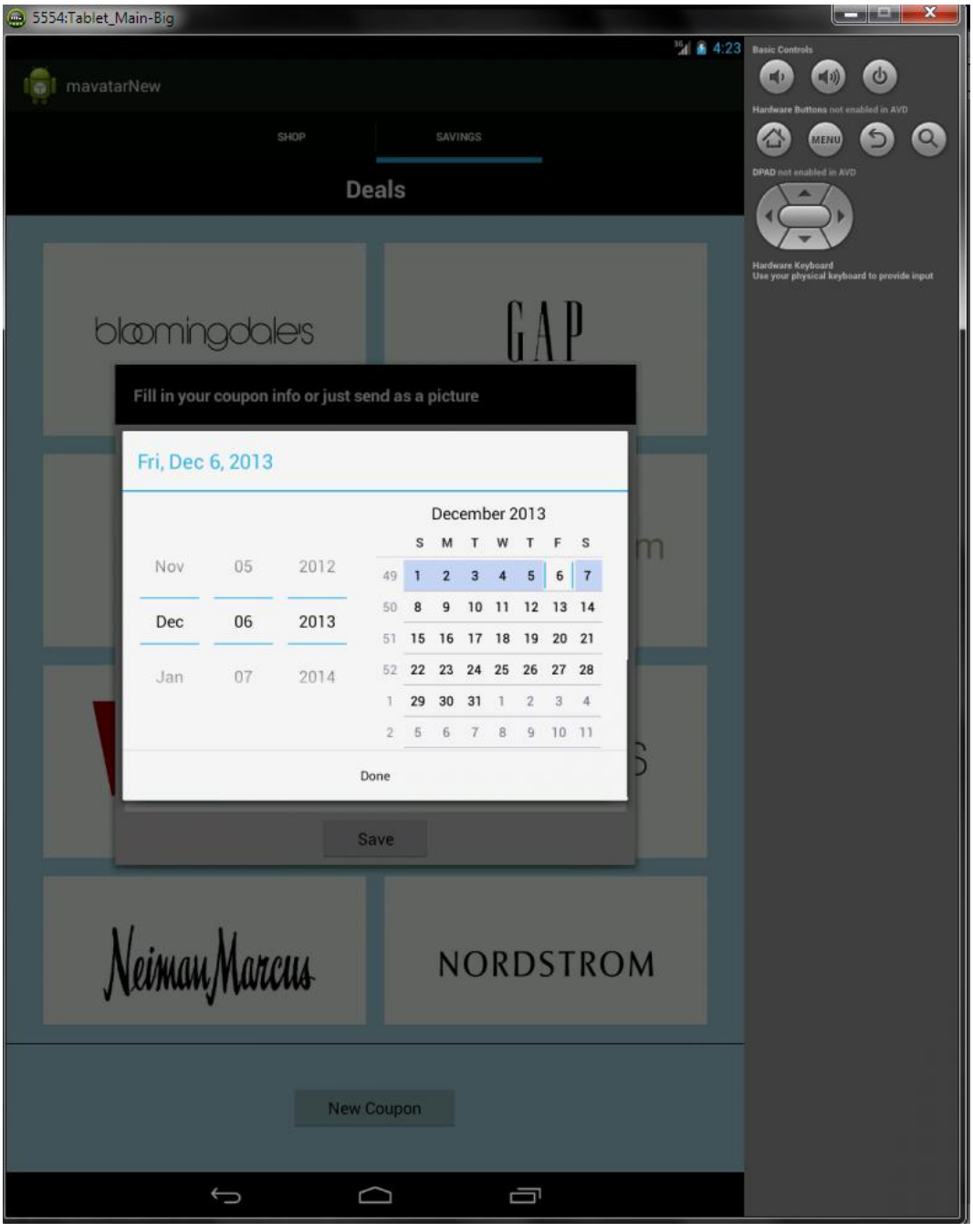


Εικόνα 5.23: Το παράθυρο NewCouponDialog μετά την επιλογή του καταστήματος Levi's

Τα πεδία Valid From και Valid To εκτελούν πανομοιότυπες λειτουργίες καθώς και τα δύο αναφέρονται σε ημερομηνίες. Έτσι, ο χρήστης καλείται να εισάγει από ποια ημερομηνία και έπειτα είναι έγκυρο το κουπόνι και ποια είναι η ημερομηνία λήξης του. Όταν ο χρήστης πατήσει σε ένα από αυτά τα πλήκτρα SELECT τότε εμφανίζεται ένα ακόμα Dialog Fragment, το **DatePickerFragment**. Το νέο παράθυρο αυτό περιέχει ένα ημερολόγιο με το οποίο ο χρήστης μπορεί να επιλέξει την ημερομηνία που επιθυμεί. Ακολουθεί τμήμα του κώδικα:

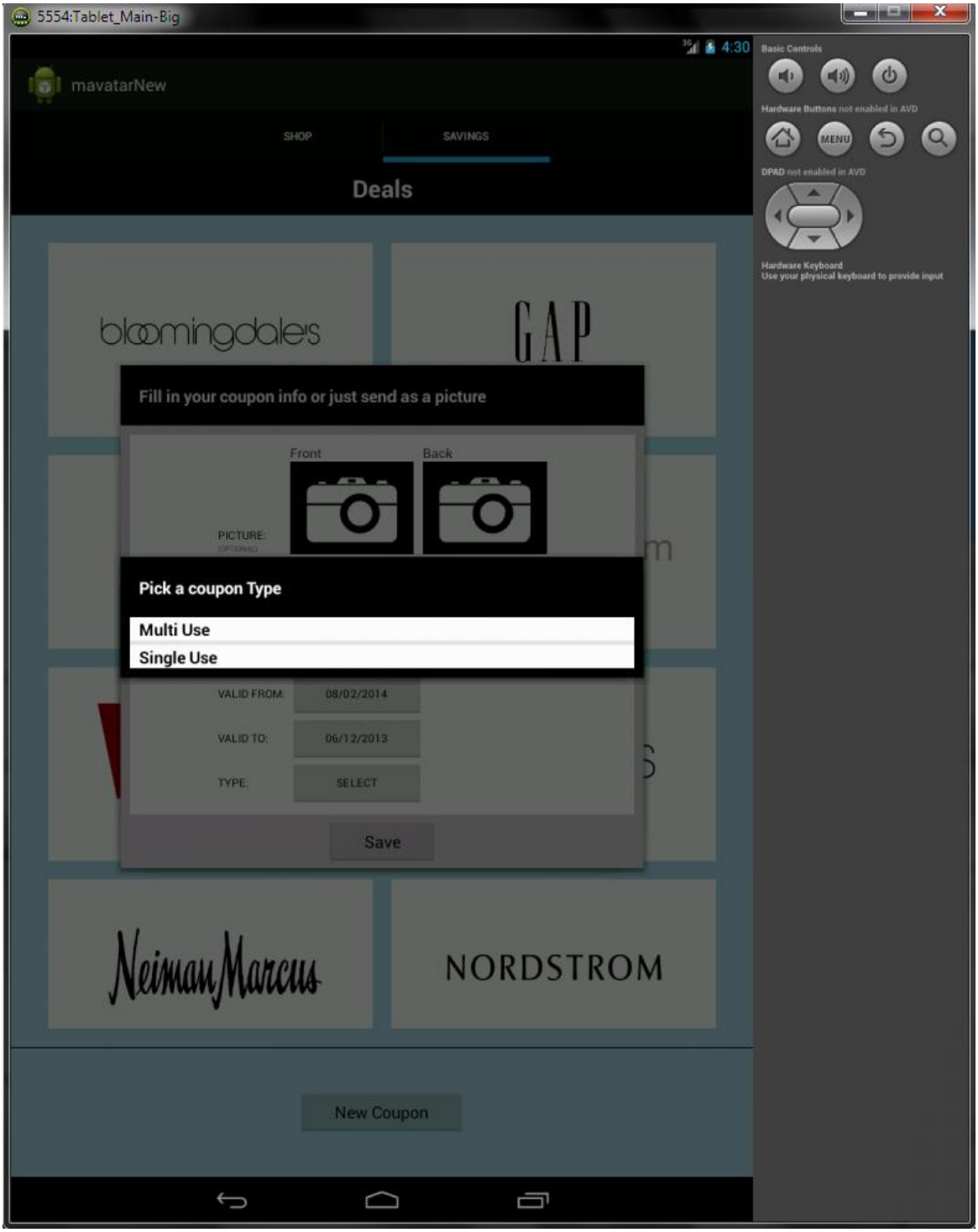
```
public class DatePickerFragment extends DialogFragment implements
DatePickerDialog.OnDateSetListener {
  @Override
  public Dialog onCreateDialog(Bundle savedInstanceState) {
      // Use the current date as the default date in the picker
      final Calendar c = Calendar.getInstance();
      int year = c.get(Calendar.YEAR);
      int month = c.get(Calendar.MONTH);
      int day = c.get(Calendar.DAY_OF_MONTH);
      // Create a new instance of DatePickerDialog and return it
      return new DatePickerDialog(getActivity(), this, year, month, day);
  }
```

Το παράθυρο αυτό φαίνεται στην παρακάτω **Εικόνα 5.24**.



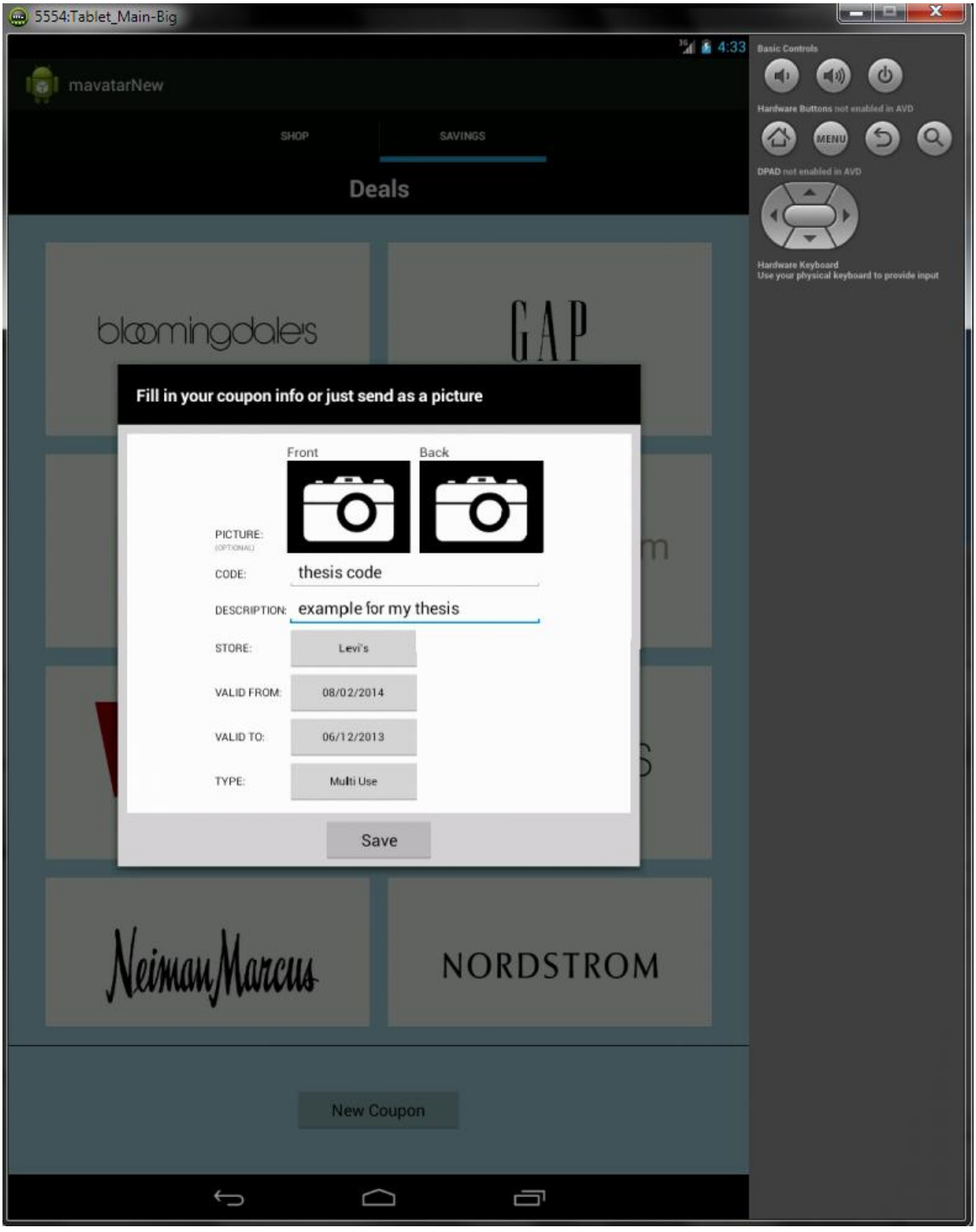
Εικόνα 5.24: Το παράθυρο επιλογής ημερομηνίας, DatePickerFragment

Πατώντας το πλήκτρο Done, το παράθυρο εξαφανίζεται και η ημερομηνία που επιλέχθηκε εμφανίζεται στη θέση του SELECT στο αντίστοιχο πλήκτρο. Πατώντας έπειτα στο πλήκτρο Type εμφανίζεται ένα ακόμα παράθυρο Dialog Fragment το **SelectTypeDialog** το οποίο όπως και το SelectStoreDialog περιέχει μια λίστα. Εδώ όμως τα πράγματα είναι πιο απλά καθ' ότι ο χρήστης καλείται να επιλέξει ανάμεσα σε δύο επιλογές, Single Use και Multi Use (παράγραφος 5.3.2.2) οι οποίες αναφέρονται στην δυνατότητα ή όχι πολλαπλής χρήσης του κουπονιού. Το παράθυρο αυτό φαίνεται στην **Εικόνα 5.25**.



Εικόνα 5.25: Το παράθυρο επιλογής τύπου κουπονιού, SelectTypeDialog

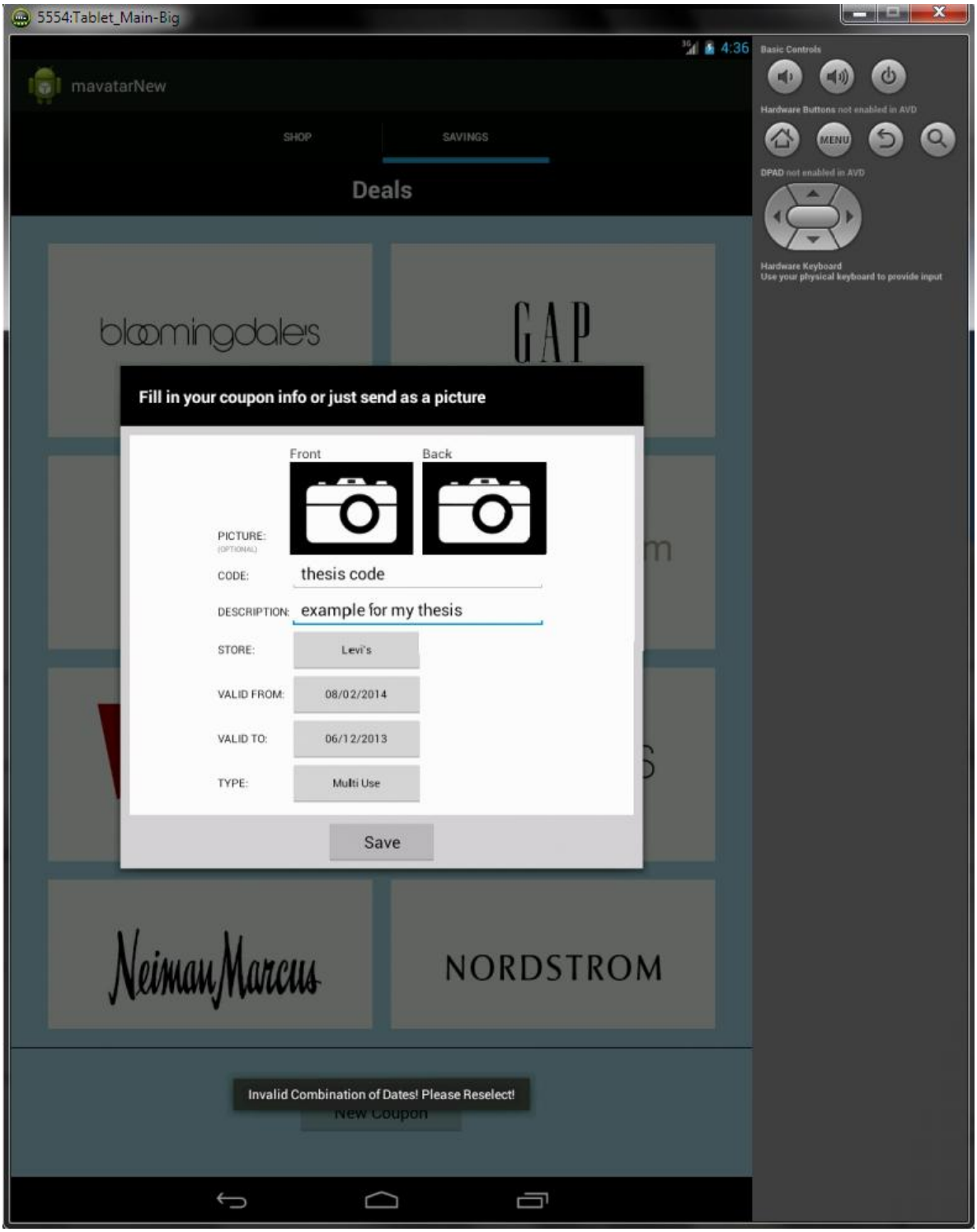
Μόλις ο χρήστης επιλέξει τον τύπο του κουπονιού, το παράθυρο εξαφανίζεται και η επιλογή του εμφανίζεται μέσα στο αντίστοιχο πλήκτρο που πριν έγραφε SELECT. Μετά από την εισαγωγή όλων των απαραίτητων δεδομένων ο χρήστης είναι έτοιμος να καταχωρήσει το κουπόνι του. Ένα τέτοιο κουπόνι φαίνεται στην **Εικόνα 5.26**.



Εικόνα 5.26: Το παράθυρο NewCouponDialog με όλα τα πεδία συμπληρωμένα

Αν προσέξει κάποιος την **Εικόνα 5.26** θα παρατηρήσει ότι ο χρήστης έχει εισάγει ημερομηνία λήξης προγενέστερη της ημερομηνίας έναρξης. Αυτό φυσικά και είναι λάθος και η εφαρμογή

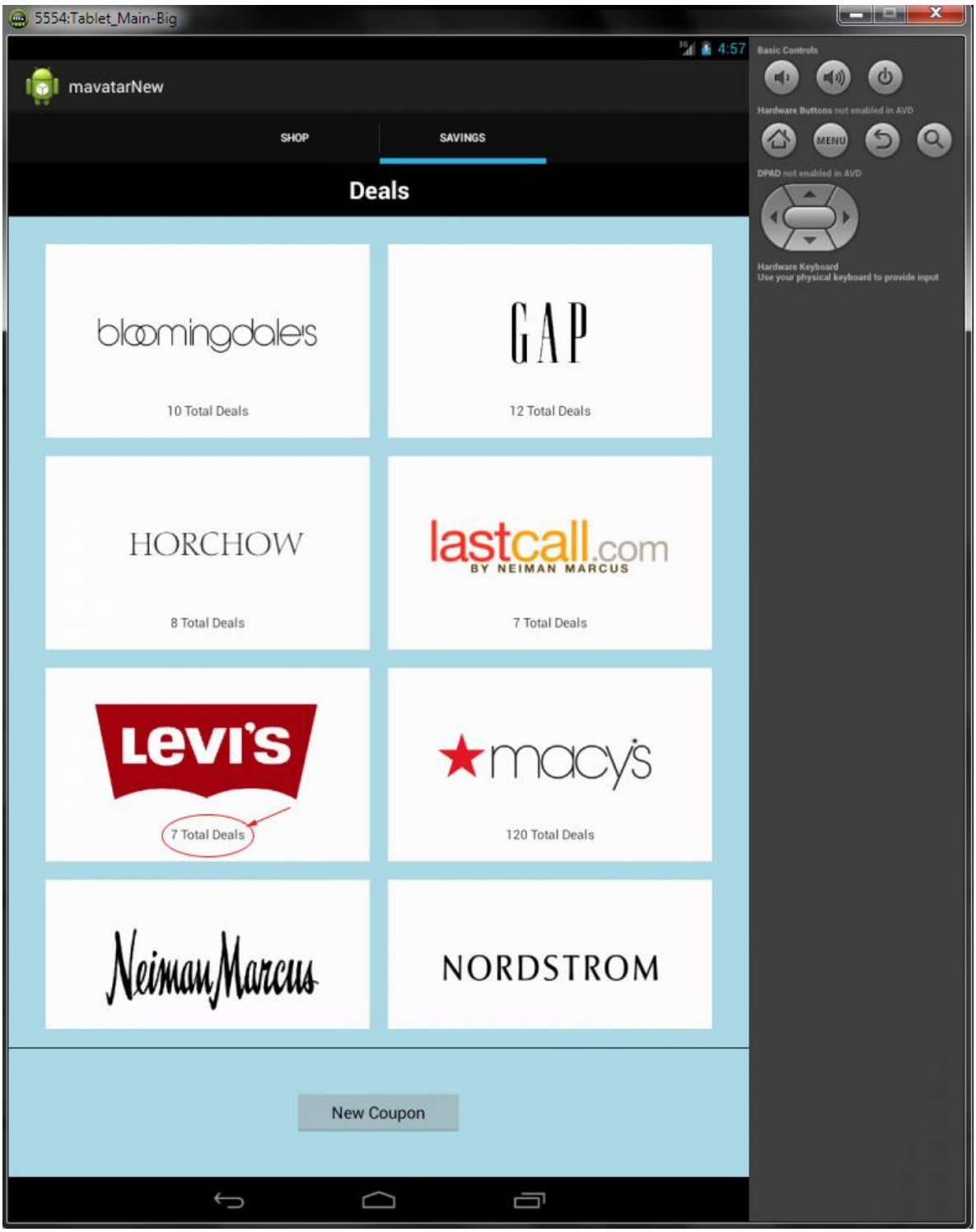
θα πρέπει να το λαμβάνει υπ' όψιν. Γι αυτό και σε αυτές τις περιπτώσεις ενημερώνεται ο χρήστης με κατάλληλο Toast. Ένα παράδειγμα φαίνεται στην **Εικόνα 5.27**.



Εικόνα 5.27: Μήνυμα Toast που ενημερώνει τον χρήστη για λάθος επιλογή ημερομηνιών

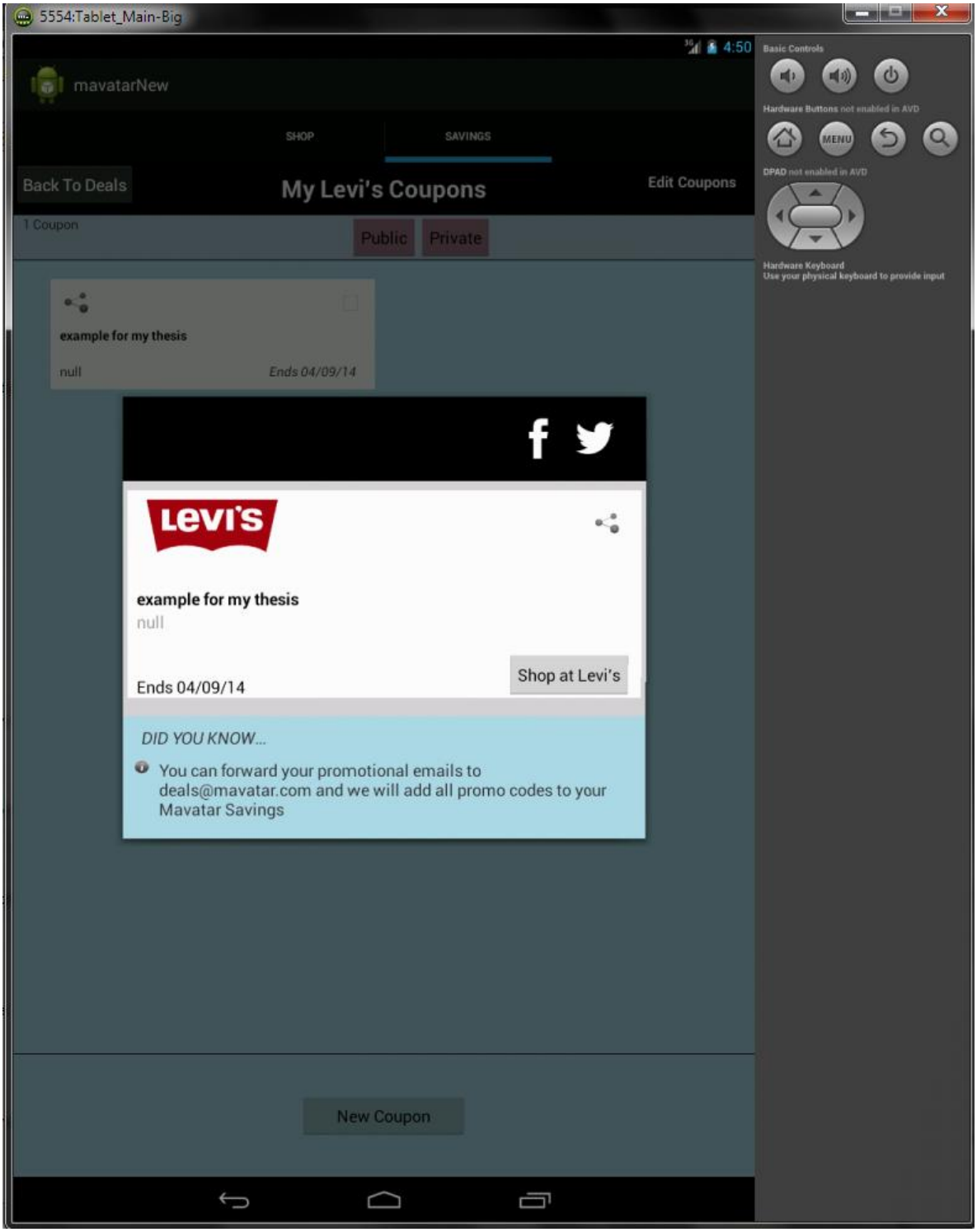
Αφού ο χρήστης διορθώσει τις ημερομηνίες και πατήσει ξανά το πλήκτρο Save τότε καλείται η κλάση NewCouponRequest η οποία, όπως και η RemoveCouponRequest που περιγράφηκε στην παράγραφο 5.3.2.2.2, αναλαμβάνει να επικοινωνήσει με τον server και να του αποστείλει τα απαραίτητα δεδομένα. Αυτό γίνεται και πάλι μέσω της λειτουργίας AsyncTask

και μέσω Http σύνδεσης τύπου POST. Δημιουργείται και στέλνεται κατάλληλο αρχείο JSON με όλα τα δεδομένα που εισήγαγε ο χρήστης. Εμφανίζεται και πάλι ένα Progress Dialog μέχρι να ολοκληρωθεί η διαδικασία και όταν αυτή ολοκληρωθεί η εφαρμογή μεταφέρει τον χρήστη στην αρχική οθόνη του Savings Tab. Ακολουθεί η **Εικόνα 5.28** στην οποία φαίνεται η αρχική οθόνη μετά την εισαγωγή του νέου κουπονιού.



Εικόνα 5.28: Η αρχική οθόνη του Savings Tab μετά την προσθήκη του νέου κουπονιού

Συγκρίνοντας και πάλι την **Εικόνα 5.28** με την **Εικόνα 5.9** παρατηρούμε ότι έχουν αυξηθεί τα κουπόνια του καταστήματος Levi's κατά 1, το οποίο είναι και αυτό που μόλις δημιουργήθηκε. Το κουπόνι αυτό φαίνεται αναλυτικά στην οθόνη Private του καταστήματος Levi's και πατώντας επάνω του ανοίγει σε νέο παράθυρο-διάλογο. Αυτό φαίνεται στην παρακάτω **Εικόνα 5.29**.



Εικόνα 5.29: Το κουπόνι που δημιουργήθηκε προηγουμένως.

5.3.2.4 Πατώντας στο πλήκτρο Back To Deals

Το πλήκτρο Back To Deals το μόνο που κάνει είναι να επιστρέφει τον χρήστη στην αρχική οθόνη του Savings Tab. Ουσιαστικά καλεί ξανά το SavingsFragment το οποίο ξανακάνει την αντίστοιχη κλήση στον Server. Είναι αρκετά χρήσιμο όταν ο χρήστης έχει πλοηγηθεί ανάμεσα σε πολλές οθόνες και θα χρειαζόταν να πατήσει πολλές φορές το πλήκτρο Back της συσκευής για να επανέλθει στην αρχική οθόνη του Savings Tab.

5.4 Το Shop Tab

5.4.1 Το ShopFragment

Όπως αναφέρθηκε και προηγουμένως, μετά την επιτυχή σύνδεση του χρήστη στην εφαρμογή ο χρήστης παρατηρεί στην οθόνη του το Shop Tab. Το Shop Tab είναι το βασικό Tab της εφαρμογής. Σε αυτό το Tab ο χρήστης έχει τη δυνατότητα να περιηγηθεί μέσα σε διάφορες κατηγορίες προϊόντων, να αναζητήσει συγκεκριμένα προϊόντα μέσω της λειτουργίας της αναζήτησης, να δει λεπτομερώς τα προϊόντα που τον ενδιαφέρουν καθώς και να μεταβεί στην ιστοσελίδα του καταστήματος για να κάνει την αγορά του. Η εφαρμογή παρέχει δεκάδες διαφορετικές κατηγορίες προϊόντων, οι οποίες και μέσω του server ανανεώνονται συνεχώς, από δεκάδες διαφορετικά καταστήματα.

Τα περιεχόμενα της αρχικής οθόνης του Shop Tab παρέχονται μέσω ενός Fragment, του **ShopFragment**. Αυτό το Fragment, όπως και τα περισσότερα που έχουν περιγραφεί μέχρι στιγμής, περιέχει ένα GridView μέσα στο οποίο παρουσιάζονται οι αρχικές βασικές κατηγορίες προϊόντων. Το GridView κατασκευάζεται αρχικά άδειο και έπειτα, μέσω της λειτουργίας AsyncTask, γίνεται μια σύνδεση Http τύπου POST στον server για να αντληθούν τα απαραίτητα δεδομένα. Τα δεδομένα αυτά περιέχουν τις βασικές πληροφορίες της κάθε κατηγορίας που είναι η εικόνα που την χαρακτηρίζει καθώς και το όνομα της. Μόλις η αποστολή των δεδομένων από τον server ολοκληρωθεί τότε ο Adapter (ShopCategoryCellAdapter) ενημερώνεται και ανανεώνει το πλέγμα με τις κατάλληλες κατηγορίες. Η αρχική οθόνη του Shop Tab φαίνεται στην **Εικόνα 5.30**.

Σε αυτό το σημείο, θεωρείται απαραίτητο να αναφερθεί ο **Image Loader** που έχει κατασκευαστεί και έχει ως λειτουργία την φόρτωση των κατάλληλων εικόνων για τις κατηγορίες (και για τα προϊόντα) καθώς και των κατάλληλων ενδείξεων φόρτωσης.

5.4.1.1 Ο Image Loader

Το πιο σημαντικό σε μια εφαρμογή η οποία βασίζεται στην διαδικτυακή επικοινωνία με server και που εκτελεί χρονοβόρες διαδικασίες, είναι να μην αποσπά το χρήστη από την χρήση της εφαρμογής όσο αυτές οι διαδικασίες βρίσκονται σε εξέλιξη. Σε προηγούμενα σημεία

αναφέρθηκε η χρήση των διαλόγων Progress Dialog για την ενημέρωση του χρήστη. Όμως, τα δεδομένα που ανταλλάσσονταν με τον server ήταν κυρίως ένα αρχείο JSON με text δεδομένα, πράγμα που σημαίνει ότι ο χρόνος που χρειαζόταν η διαδικασία ήταν μηδαμινός.

Στην περίπτωση των εικόνων όμως τα πράγματα είναι αρκετά πιο πολύπλοκα. Ο server δεν αποστέλλει την ίδια την εικόνα (πράγμα που θα ήταν και πρακτικά ασύμφορο να αποστέλλει δεδομένα εικόνας για δεκάδες εικόνες) αλλά παρέχει το κατάλληλο url link (διαδικτυακό σύνδεσμο) στο οποίο αυτή βρίσκεται. Έπειτα, η εφαρμογή πρέπει να συνδεθεί με αυτή την διεύθυνση και να κατεβάσει από εκεί την εικόνα. Γίνεται εμφανές, ότι αν το μέγεθος της εικόνας είναι πολύ μεγάλο η διαδικασία «κατεβάσματος» της εικόνας θα είναι αρκετά χρονοβόρα. Σε όλο αυτό το διάστημα, δε θα ήταν δυνατό απλώς να εμφανιστεί ένα Progress Dialog και να αφεθεί ο χρήστης να περιμένει. Αυτό θα ήταν πρακτικά καταστροφικό και για έναν ακόμα λόγο: η εικόνα μπορεί να μην υπάρχει πλέον στον διαδικτυακό σύνδεσμο, ή ο σύνδεσμος να μην δουλεύει. Ειδικά όσων αφορά τα προϊόντα που η φυσική παρουσία των πληροφοριών τους βρίσκεται σε εκατοντάδες διαφορετικούς server από διαφορετικά καταστήματα, είναι σύνηθες φαινόμενο η απουσία εικόνων και ανενεργών διαδικτυακών συνδέσμων.

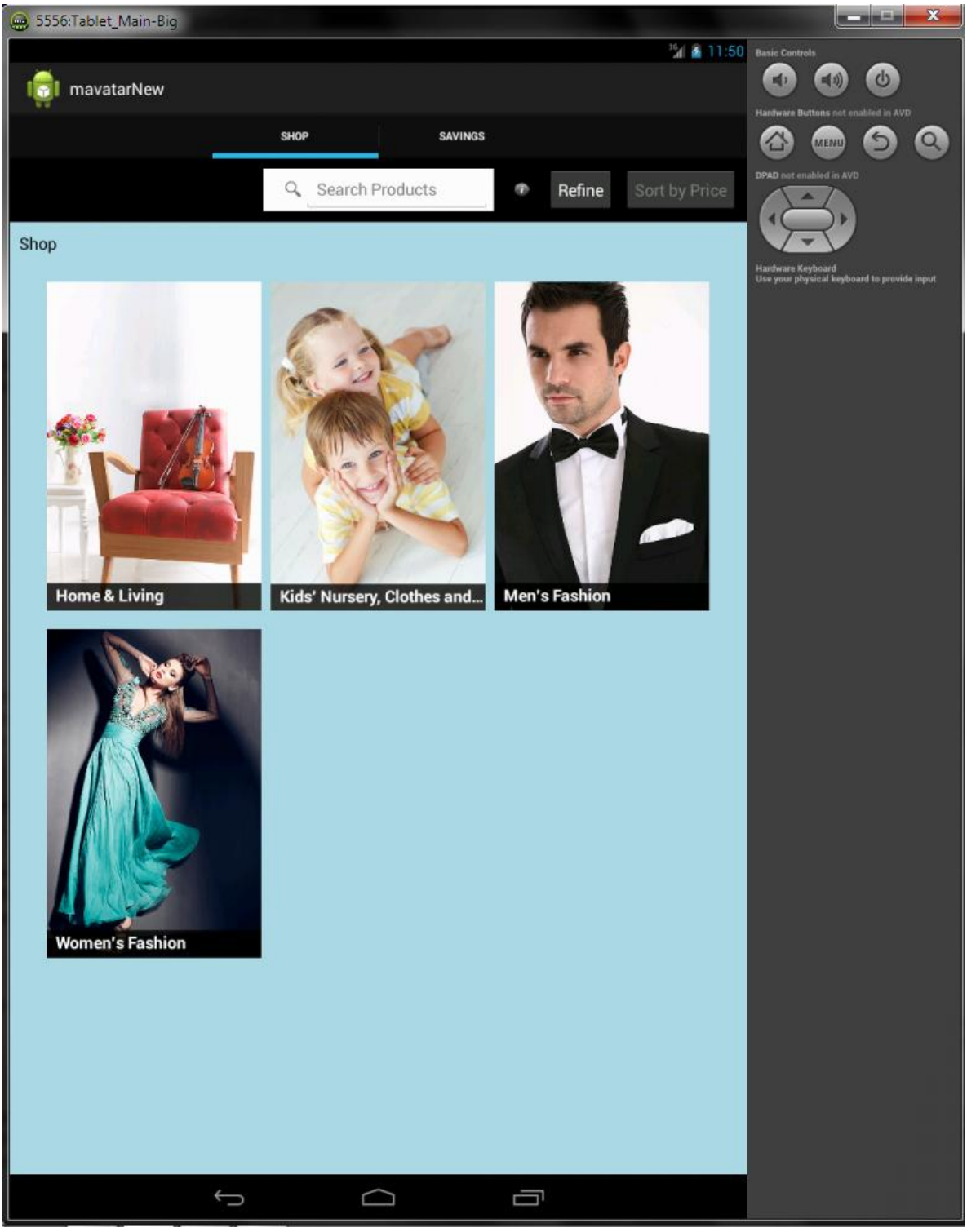
Το πρόβλημα αυτό έρχεται να λύσει ο Image Loader. Η λειτουργία που επιτελεί είναι απλή αλλά πολύ σημαντική. Αρχικά, εμφανίζει στο πλέγμα και στη θέση που θα έπρεπε να εμφανιστεί η εικόνα, ένα animation που δείχνει ότι η εικόνα «φορτώνεται». Το animation αποτελείται από ένα απλό spinner που γυρίζει. Έπειτα, ξεκινάει την διαδικασία κατεβάσματος της εικόνας. Μόλις η εικόνα κατέβει, απλά αντικαθιστά το spinner με την εικόνα. Το σημαντικό σε αυτή τη διαδικασία είναι ότι ο χρήστης μπορεί κανονικά να πατήσει πάνω στο spinner και να ανοίξει μια νέα οθόνη ακόμα και αν η εικόνα δεν έχει εμφανιστεί. Φυσικά, αν ο σύνδεσμος είναι ανενεργός ή δεν υπάρχει η αντίστοιχη εικόνα, απλά δεν θα ανανεωθεί ποτέ και θα παραμείνει το spinner.

Για να γίνει ακόμα πιο αποδοτική η εφαρμογή, χρησιμοποιήθηκε μια Memory Cache στην οποία αποθηκεύτηκαν οι εικόνες που έχουν κατέβει, ώστε να μην ξαναχρειαστεί να ξανακατέβουν όταν ο χρήστης επιστρέψει στην ίδια οθόνη.

Ο Image Loader κατασκευάζεται στην κλάση ImageLoader.java και ένα τμήμα του κώδικα φαίνεται στη συνέχεια:

```
final int stub_id=R.drawable.loading2;//The loading animation
public void DisplayImage(String url, ImageView imageView){
        imageViews.put(imageView, url);
        Bitmap bitmap=memoryCache.get(url);
        if(bitmap!=null){
            imageView.setImageBitmap(bitmap);
          imageView.clearAnimation();
        }else{
            queuePhoto(url, imageView);
            imageView.setImageResource(stub_id);
            imageView.startAnimation(loadingAnim);
        }
}
```

Κατάλληλες εικόνες που κάνουν εμφανή τη λειτουργία του Image Loader βρίσκονται στη συνέχεια του κεφαλαίου.



Εικόνα 5.30: Η αρχική οθόνη του Shop Tab με τις βασικές κατηγορίες προϊόντων .

Θα αναλυθεί τώρα η αρχική οθόνη του Shop Tab όπως αυτή φαίνεται παραπάνω στην **Εικόνα 5.30**. Σε αυτή την οθόνη ο χρήστης έχει αρκετές επιλογές:

- Να πατήσει πάνω σε κάποια κατηγορία ώστε να περιηγηθεί στα προϊόντα αυτής της κατηγορίας
- Να χρησιμοποιήσει την λειτουργία αναζήτησης μέσω του ειδικού πεδίου Search, για να αναζητήσει συγκεκριμένα προϊόντα
- Να πατήσει το πλήκτρο Refine ώστε να φιλτράρει με κατάλληλο τρόπο τα προϊόντα που θα εμφανιστούν στην συνέχεια

5.4.1.2 Πατώντας σε κάποια κατηγορία

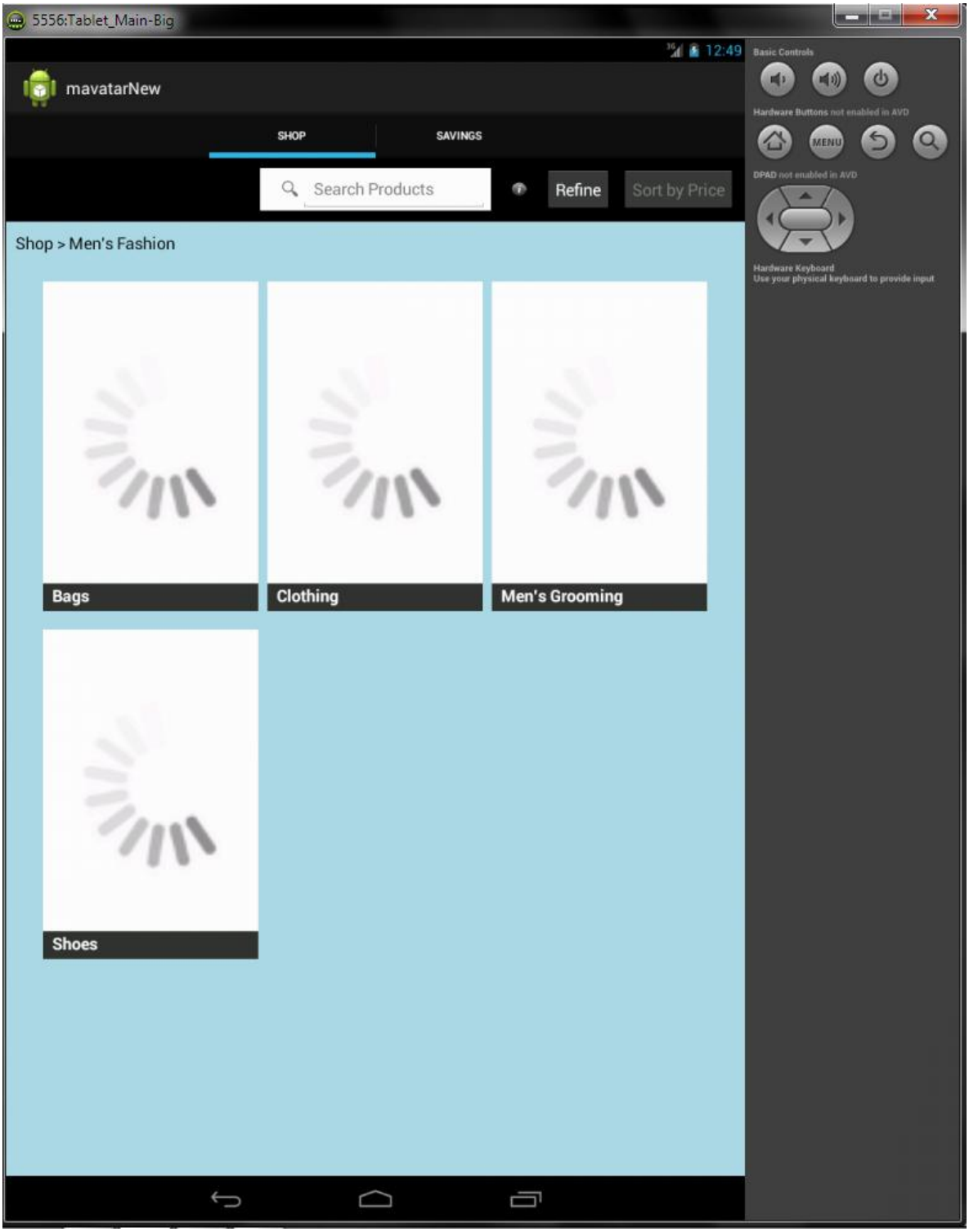
Όταν ο χρήστης πατήσει πάνω σε κάποια κατηγορία τότε το ShopFragment μέσω ενός Listener εντοπίζει ποια κατηγορία πατήθηκε και έπειτα αντικαθίσταται από το **ShopFragment2** το οποίο περιέχει τις υποκατηγορίες αυτής της κατηγορίας (είτε τα προϊόντα). Πριν γίνει αυτό, μέσω της λειτουργίας AsyncTask, η εφαρμογή επικοινωνεί με τον server μέσω μιας σύνδεσης Http τύπου POST, όπου λαμβάνει τα δεδομένα για την επόμενη οθόνη. Το ShopFragment μπαίνει στο back stack ώστε ο χρήστης να μπορεί να επιστρέψει σε αυτό με τη χρήση του πλήκτρου Back. Παρατίθενται κάποια χρήσιμα τμήματα κώδικα.

```
gridView.setOnItemClickListener(new OnItemClickListener() {

      @Override
       public void onItemClick(AdapterView<?> parent, View v,
                                int position, long id) {
```

[...]

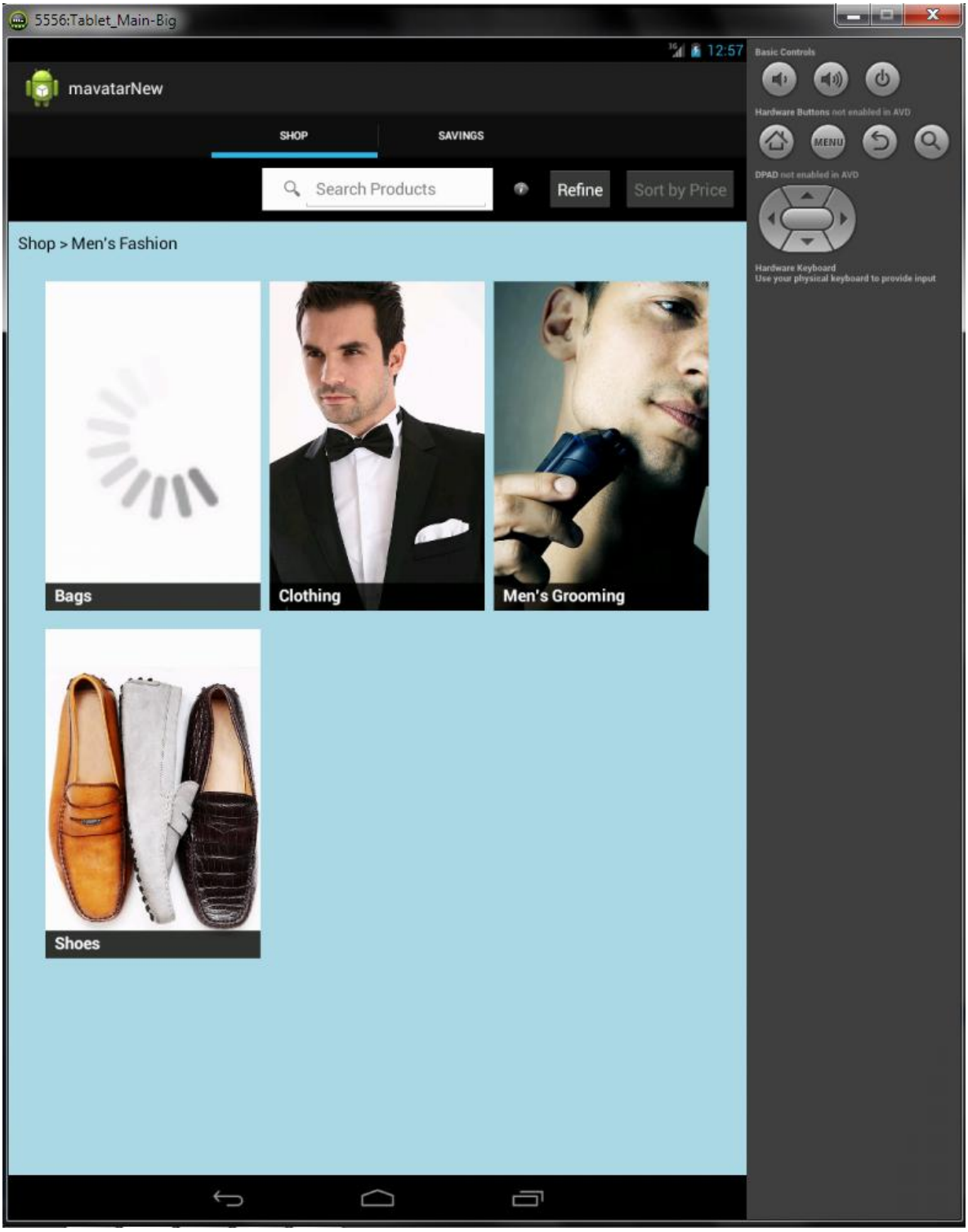
```
/* When clicked we replace our Fragment with a new one based on the
category clicked */
      frag = new ShopFragment2();

      FragmentManager fragmentManager = getFragmentManager();
      FragmentTransaction ft = fragmentManager.beginTransaction();
      ft.replace(android.R.id.content, frag);
      ft.addToBackStack(null);
      ft.commit();
```

Αν για παράδειγμα ο χρήστης πατήσει στην κατηγορία Men's Fashion η οθόνη που εμφανίζεται φαίνεται παρακάτω στην **Εικόνα 5.31** (και εν συνεχεία στην **Εικόνα 5.32**).



Εικόνα 5.31: Η οθόνη του Shop Tab όταν ο χρήστης πατήσει στην κατηγορία Men's Fashion

Εδώ γίνεται εμφανής η λειτουργία του Image Loader που αναλύθηκε στην προηγούμενη παράγραφο 5.4.1.1. Στην **Εικόνα 5.31** φαίνεται η οθόνη όταν ακόμα οι εικόνες φορτώνουν, ο χρήστης όμως έχει τη δυνατότητα να χρησιμοποιήσει κανονικά όλες τις λειτουργίες της εφαρμογής. Όταν οι εικόνες φορτώσουν (κάτι το οποίο γίνεται πολύ γρήγορα), τα spinner αντικαθίστανται και η οθόνη φαίνεται παρακάτω στην **Εικόνα 5.32**.



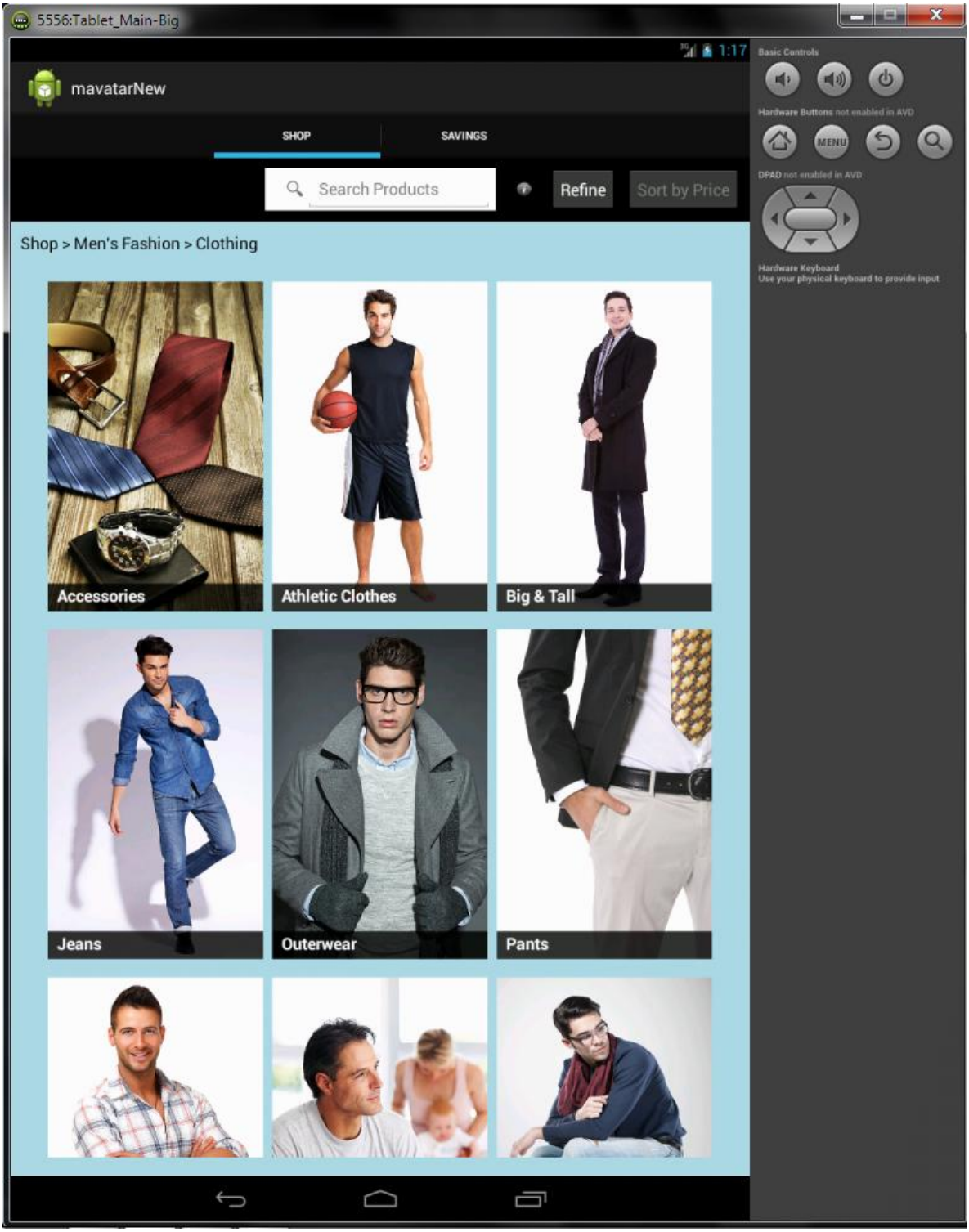
Εικόνα 5.32: Η οθόνη του Shop Tab όταν ο χρήστης πατήσει στην κατηγορία Men's Fashion και αφού οι εικόνες έχουν φορτωθεί.

Στην παραπάνω **Εικόνα 5.32** παρατηρείται ότι τα κελιά με τις κατηγορίες ανανεώθηκαν με τις εικόνες τους. Επίσης, φαίνεται ότι στη πρώτη κατηγορία (Bags) παραμένει το spinner. Αυτό συμβαίνει διότι η συγκεκριμένη εικόνα δεν είναι διαθέσιμη. Ο χρήστης όμως μπορεί κανονικά να την επισκεφθεί.

Το ShopFragment2 ουσιαστικά δεν διαφέρει σε τίποτα από το ShopFragment, αυτό υποδεικνύεται και από τα παρόμοια ονόματα. Ο σχεδιασμός του είναι ακριβώς ο ίδιος, αλλάζει απλά το περιεχόμενο του πλέγματος, στο οποίο φορτώνονται τα περιεχόμενα της κατηγορίας που επιλέχθηκε στην προηγούμενη οθόνη. Επίσης, στο πάνω μέρος του πλέγματος ανανεώνεται ο τίτλος ενημερώνοντας τον χρήστη για το «δένδρο» των κατηγοριών (φαίνεται ότι το Shop έγινε Shop > Men's Fashion).

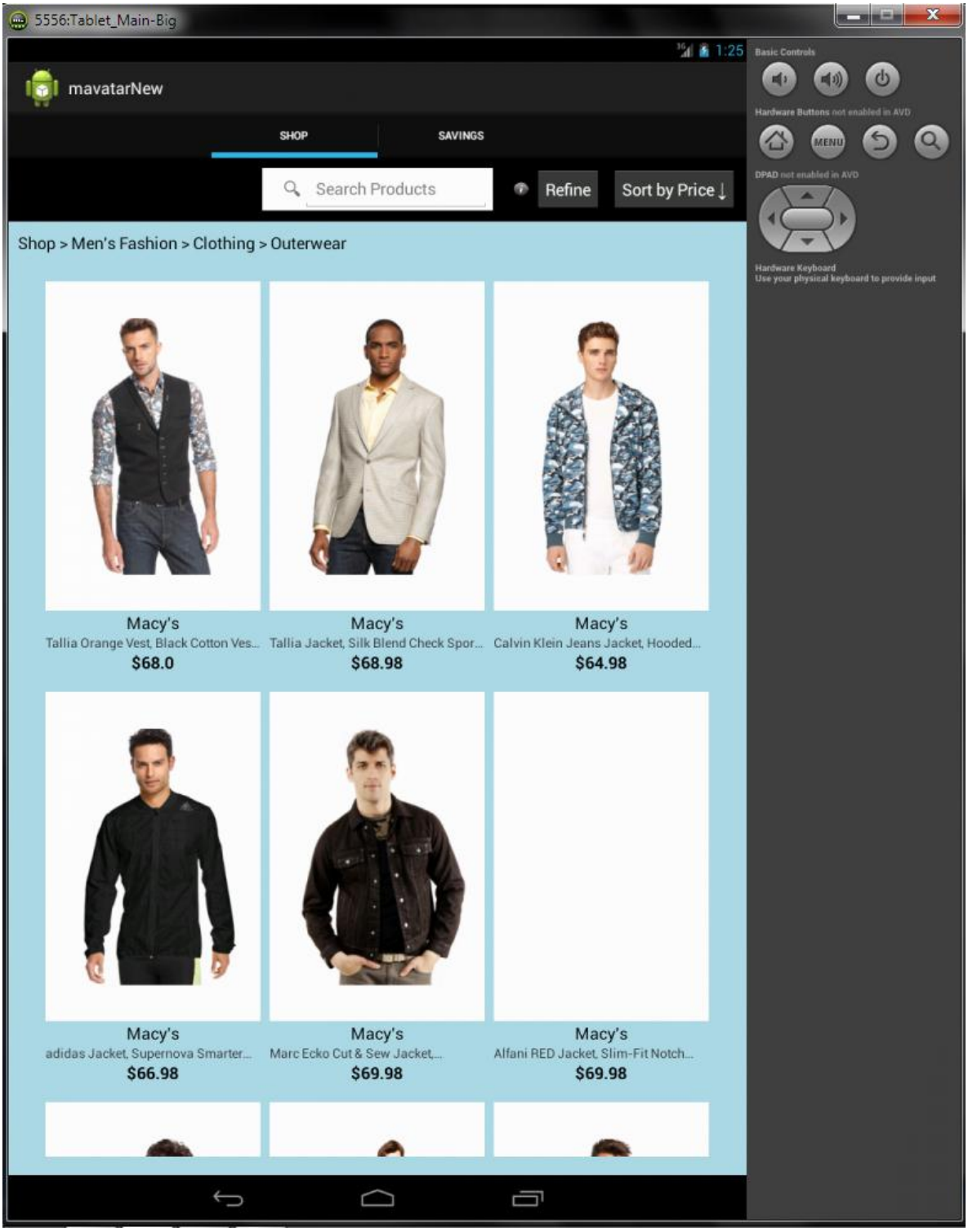
Για να γίνει πιο κατανοητός ο σχεδιασμός και η ροή της λειτουργίας της εφαρμογής, αναφέρεται ότι αν ο χρήστης και σε αυτή την οθόνη επιλέξει κάποια κατηγορία, θα φορτωθεί ένα νέο Fragment, το οποίο είναι το **ShopFragment3** με τα περιεχόμενα της κατηγορίας. Είναι ακριβώς η ίδια διαδικασία που ακολουθήθηκε κατά την μετάβαση από το ShopFragment στο ShopFragment2. Αυτή η εναλλαγή οθονών μέσα στις κατηγορίες ουσιαστικά συνεχίζεται μέχρι ο χρήστης να φτάσει στο τελευταίο επίπεδο, όπου είναι και τα προϊόντα της συγκεκριμένης κατηγορίας. Επειδή διαφορετικές κατηγορίες έχουν διαφορετικό «βάθος» επιπέδων, έχουν κατασκευαστεί 4 Fragment όπου 4 είναι και το μέγιστο δυνατό βάθος. Αν η κατηγορία περιέχει και άλλες κατηγορίες τότε φορτώνεται το αντίστοιχο Fragment (ShopFragment > ShopFragment2 > ShopFragment3 > **ShopFragment4**). Αν η κατηγορία περιέχει προϊόντα, τότε φορτώνεται ένα άλλο, ειδικό Fragment, το **ShopFragmentProducts** το οποίο και αναλύεται στη συνέχεια του κεφαλαίου.

Η ροή αυτή μπορεί εύκολα να γίνει κατανοητή μέσα από τις εικόνες που ακολουθούν. Αν για παράδειγμα στην οθόνη της **Εικόνας 5.32** ο χρήστης πατήσει στην κατηγορία Clothing η οθόνη που εμφανίζεται φαίνεται στην **Εικόνα 5.33** παρακάτω.



Εικόνα 5.33: Η οθόνη του Shop Tab όταν ο χρήστης πατήσει στην κατηγορία Clothing (μέσα από την κατηγορία (Men's Fashion).

Στην παραπάνω **Εικόνα 5.33** φαίνονται οι υποκατηγορίες της κατηγορίας Clothing που πατήθηκε. Αυτή ουσιαστικά είναι η οθόνη που εμφανίζεται με τη λειτουργία του ShopFragment3. Αν σε αυτή την οθόνη ο χρήστης πατήσει στην κατηγορία Outerwear η οθόνη που εμφανίζεται φαίνεται παρακάτω στην **Εικόνα 5.34**.



Εικόνα 5.34: Η οθόνη του Shop Tab με τα προϊόντα της κατηγορίας Outerwear (μέσα από την κατηγορία Men's Fashion > Clothing)

Στην παραπάνω **Εικόνα 5.34** παρατηρείται ότι ο χρήστης έφτασε στο τελευταίο επίπεδο αυτής της κατηγορίας και για πρώτη φορά έχει πρόσβαση σε προϊόντα. Η οθόνη αυτή εμφανίζεται με την λειτουργία του **ShopFragmentProducts**, το οποίο αναλύεται παρακάτω.

5.4.2 Το ShopFragmentProducts

Σε αυτή την οθόνη ο χρήστης έχει τη δυνατότητα να δει τα προϊόντα της κατηγορίας που επέλεξε. Τα προϊόντα φορτώνονται σε 9-άδες και εμφανίζονται μέσα στο πλέγμα. Ένα από τα μεγάλα προβλήματα που έπρεπε να αντιμετωπιστεί ήταν ο όγκος δεδομένων. Μια κατηγορία μπορεί και να περιέχει εκατοντάδες ή και χιλιάδες προϊόντα (και όσο θα προστίθενται στον server περισσότερα προϊόντα ο αριθμός μπορεί να αγγίζει και εκατομμύρια). Αν γινόταν προσπάθεια να φορτωθούν όλα αυτά τα δεδομένα με μια μόνο κλήση στον server, είναι κατανοητό ότι η διαδικασία δεν θα τελείωνε ποτέ. Γι αυτό εφαρμόστηκε η τεχνική του On Scroll Load.

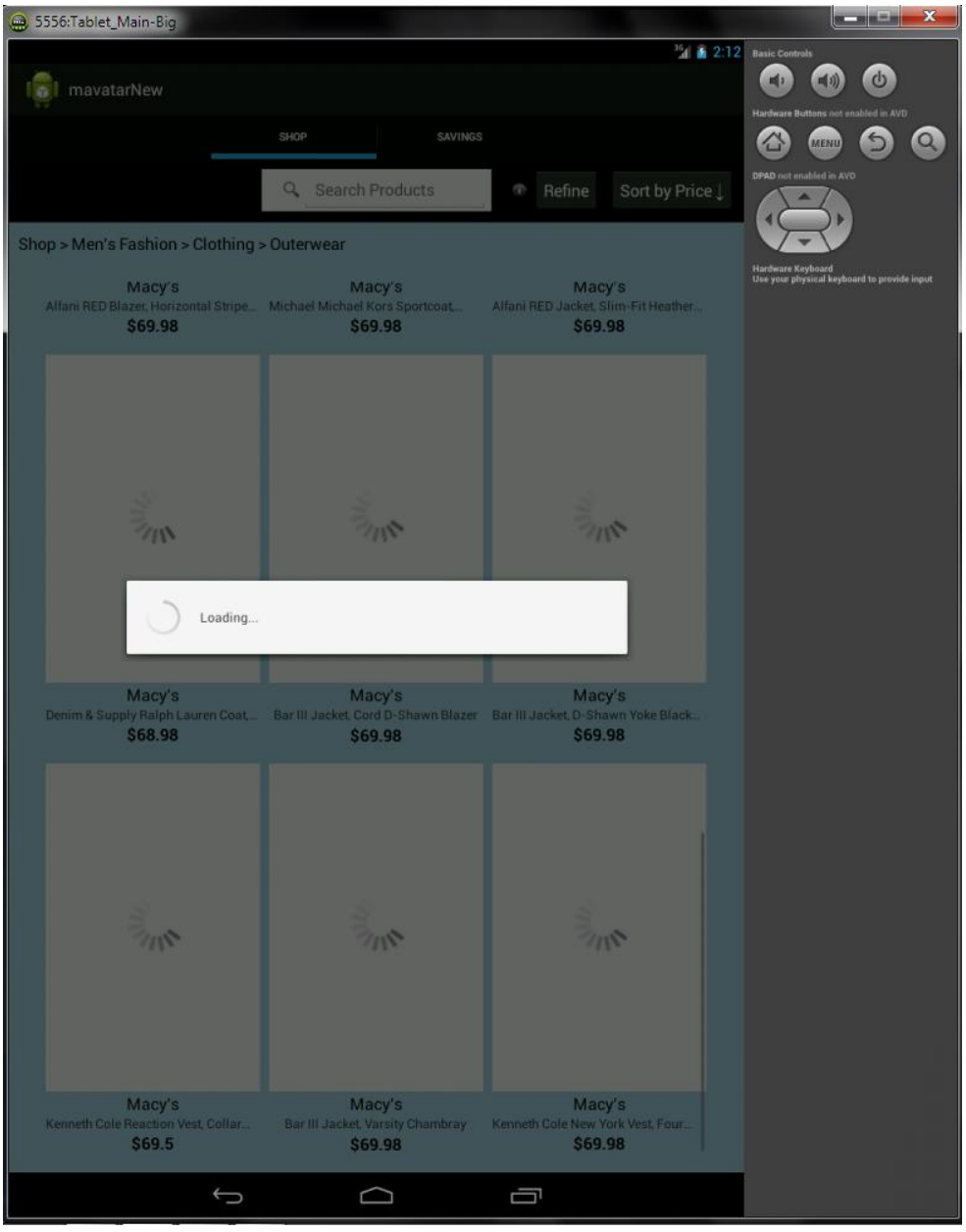
Ουσιαστικά γίνεται μια κλήση στον server μέσω AsyncTask και μιας Http σύνδεσης τύπου POST, και ζητείται μια 9άδα από προϊόντα. Όταν ο χρήστης κάνει scroll down και φτάσει στο τέλος της 9άδας, τότε γίνεται εκ νέου κλήση και φορτώνεται μια ακόμα 9άδα και τοποθετείται στο τέλος της προηγούμενης. Έτσι, ο χρήστης μπορεί να χειριστεί ομαλά την εφαρμογή και να κάνει ελεύθερα scroll αναζητώντας νέα προϊόντα με ελάχιστη έως καθόλου καθυστέρηση. Επίσης, ο κώδικας της εφαρμογής είναι πολύ εύκολα παραμετροποιήσιμος και μπορεί με μια απλή αλλαγή να αλλάξει ο αριθμός των προϊόντων που φορτώνονται από 9 σε όσα χρειάζονται. Βέβαια είναι καλύτερο να έρχονται σε πολλαπλάσια του 3 καθώς το πλέγμα περιέχει 3 στήλες (το οποίο με τη σειρά του είναι και αυτό εύκολα παραμετροποιήσιμο και μπορεί εύκολα να αλλάξει. Για παράδειγμα, μια επέκταση της εφαρμογής σε κινητά τηλέφωνα, που έχουν μικρότερη οθόνη, θα μπορούσε να έχει ένα διαφορετικό σχεδιασμό με πλέγμα 2 στηλών και φόρτωση προϊόντων σε 4άδες.). Παρατίθεται ένα τμήμα του κώδικα που εκτελεί αυτή τη λειτουργία:

```
gridView.setOnScrollListener(new AbsListView.OnScrollListener() {
          @Override
          public void onScrollStateChanged(AbsListView view, int
scrollState) {
              //
          }
          @Override
          public void onScroll(AbsListView view, int firstVisibleItem, int
visibleItemCount, int totalItemCount) {
              /*Implementing an EndlessScrollListener!*/
              if (mLoading) {
                   if (totalItemCount > mPreviousTotal) {
                       mLoading = false;
                       mPreviousTotal = totalItemCount;
                       mCurrentPage++;
                       // Condition in order to know when we
                       // have finished displaying all items
                       if (totalItemCount + 1 >= total) {
                          System.out.println(totalItemCount);
                           mLastPage = true;
                       }
                   }
           }
           if (!mLastPage && !mLoading &&
                       (totalItemCount - visibleItemCount) <=
(firstVisibleItem + mVisibleThreshold)) {
```

Ουσιαστικά χρησιμοποιούνται οι εξής παράμετροι:

```
  /*Endless Scrolling parameters */
  private int mVisibleThreshold = 0; /*The minimum amount of items to have
below current scroll position, before loading more.*/
    private int mCurrentPage = 0; /*The current page of data you have
loaded*/
    private int mPreviousTotal = 0; /*The total number of items in the
dataset after the last load*/
    private boolean mLoading = true; /*True if we are still waiting for the
last set of data to load.*/
    private boolean mLastPage = false; /*True if we are on the last page*/
```

Με αυτές γίνεται έλεγχος για το πλήθος των προϊόντων που έχουν φορτωθεί, αν υπάρχουν προϊόντα που περιμένουν να φορτωθούν εκείνη τη στιγμή και αν έχουν φορτωθεί όλα τα προϊόντα. Η διαδικασία φόρτωσης νέων προϊόντων φαίνεται στην παρακάτω **Εικόνα 5.35**.



Εικόνα 5.35: Η οθόνη του Shop Tab κατά τη διάρκεια φόρτωσης των επόμενων 9 προϊόντων

Στην οθόνη των προϊόντων ο χρήστης πέρα από το να δει τα προϊόντα και να κάνει scroll φορτώνοντας περισσότερα, έχει αρκετές ακόμα επιλογές.

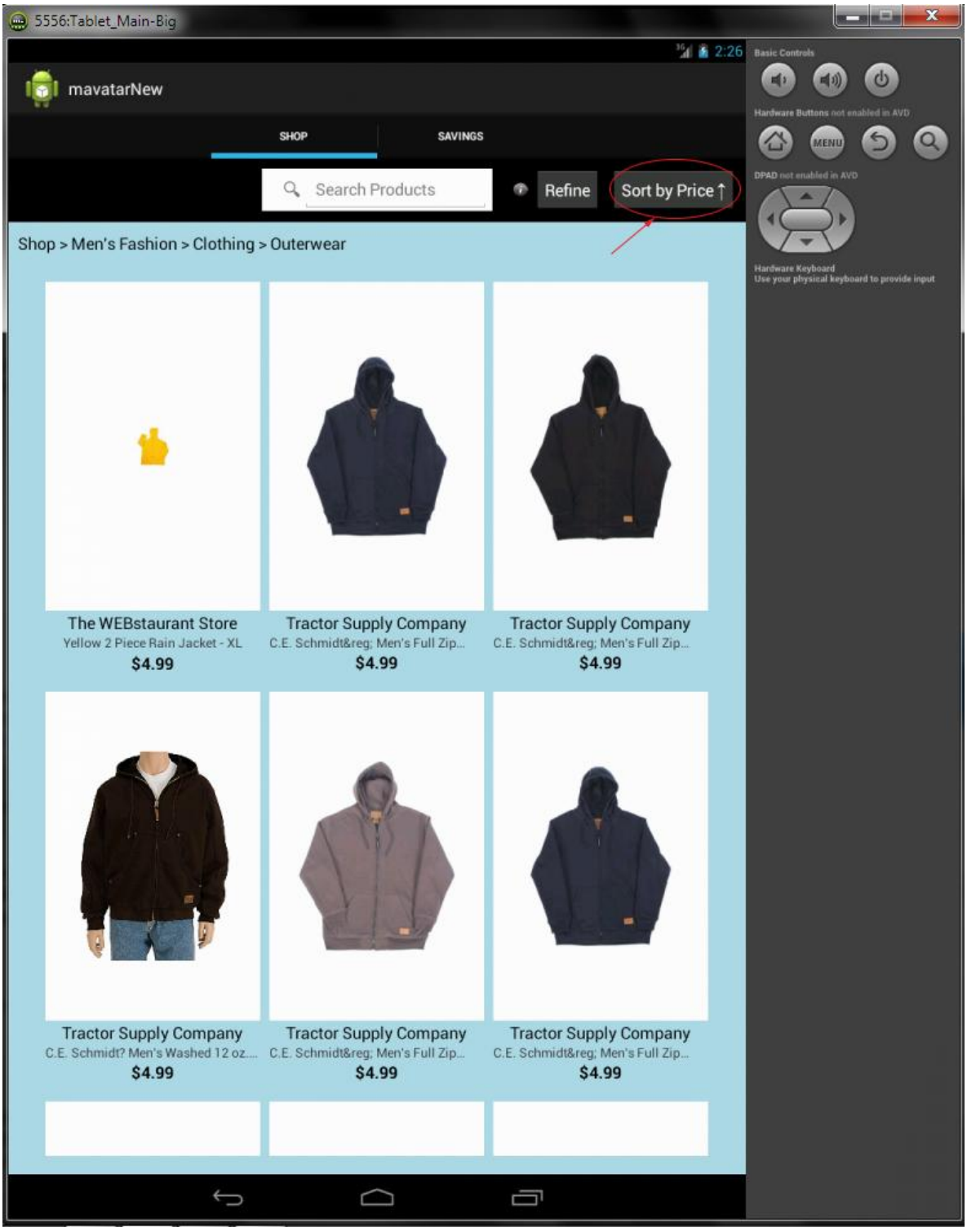
- Να πατήσει στο πλήκτρο Sort By Price για να ταξινομήσει τα προϊόντα κατά τιμή (αύξουσα αρχικά)
- Να πατήσει πάνω σε κάποιο προϊόν για να δει αναλυτικά την περιγραφή του
- Να χρησιμοποιήσει την λειτουργία αναζήτησης για να αναζητήσει προϊόντα
- Να πατήσει στο πλήκτρο Refine για να φιλτράρει τα προϊόντα

5.4.2.1 Πατώντας το πλήκτρο Sort By Price

Όταν ο χρήστης πατήσει στο πλήκτρο Sort By Price τότε γίνεται εκ νέου κλήση στον server και ανανεώνονται τα προϊόντα στο πλέγμα με βάση την τιμή τους σε αύξουσα σειρά. Επίσης, το πλήκτρο Sort By Price αλλάζει μορφή με το βέλος αυτή τη φορά να κοιτάει προς τα πάνω (δείχνοντας στο χρήστη ότι αν το ξαναπατήσει θα ξαναταξινομηθούν τα προϊόντα σε φθίνουσα σειρά). Ακολουθεί ένα τμήμα του κώδικα:

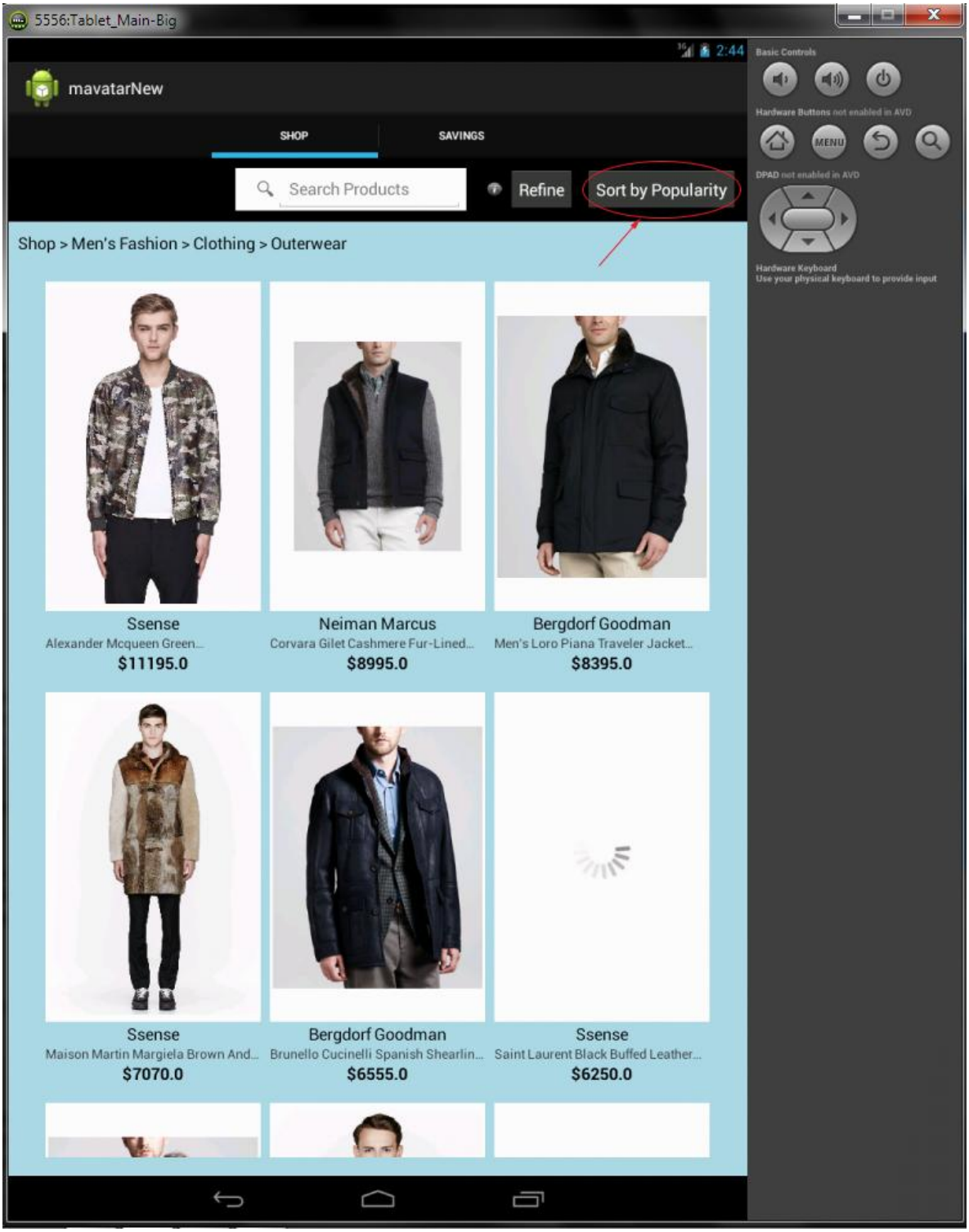
```
/*What to do when user clicks on the Sort Button*/
sortbutton.setOnClickListener(new View.OnClickListener(){

      @Override
      public void onClick(View v) {

              String buttontext = (String) sortbutton.getText();

              productlist.clear();
              spca.notifyDataSetChanged();

              /*Reinitialize every parameter*/
              mCurrentPage = 0;
              mPreviousTotal = 0;
              mLoading = true;
              mLastPage = false;
              total=0;

              if(buttontext.equals("Sort by Price \u2193")){
                  […]
              }
              else if(buttontext.equals("Sort by Price \u2191")){
                  […]
              }
              else if(buttontext.equals("Sort by Popularity")){
                  […]
              }

       }

});
```

Η οθόνη που εμφανίζεται μετά το πάτημα του Sort By Price φαίνεται παρακάτω στην **Εικόνα 5.36**.



Εικόνα 5.36: Η οθόνη του Shop Tab μετά το πάτημα του πλήκτρου Sort By Price

Αν ο χρήστης ξαναπατήσει τώρα στο (νέο) πλήκτρο Sort By Price τα προϊόντα ξαναταξινομούνται με βάση την τιμή σε φθίνουσα σειρά. Αυτό φαίνεται στην παρακάτω **Εικόνα 5.37**.



Εικόνα 5.37: Η οθόνη του Shop Tab μετά το δεύτερο πάτημα του πλήκτρου Sort By Price

Το πλήκτρο Sort By Price πλέον γράφει Sort By Popularity και αν ο χρήστης το ξαναπατήσει τα προϊόντα ξαναταξινομούνται με βάση την δημοτικότητα τους. Αυτή είναι και η αρχική επιλογή με την οποία εμφανίζονται τα προϊόντα και ουσιαστικά επιστρέφει σε αυτήν. Θα επιστρέψει δηλαδή στην οθόνη που φαίνεται στην **Εικόνα 5.34**.

5.4.2.2 Πατώντας πάνω σε κάποιο προϊόν – Το ShopFragmentProductPage

Μόλις ο χρήστης πατήσει σε κάποιο προϊόν τότε τη θέση του ShopFragmentProducts παίρνει ένα νέο Fragment, το ShopFragmentProductPage. Σε αυτό το Fragment ο χρήστης μπορεί να δει αναλυτικές πληροφορίες για το προϊόν που επέλεξε.

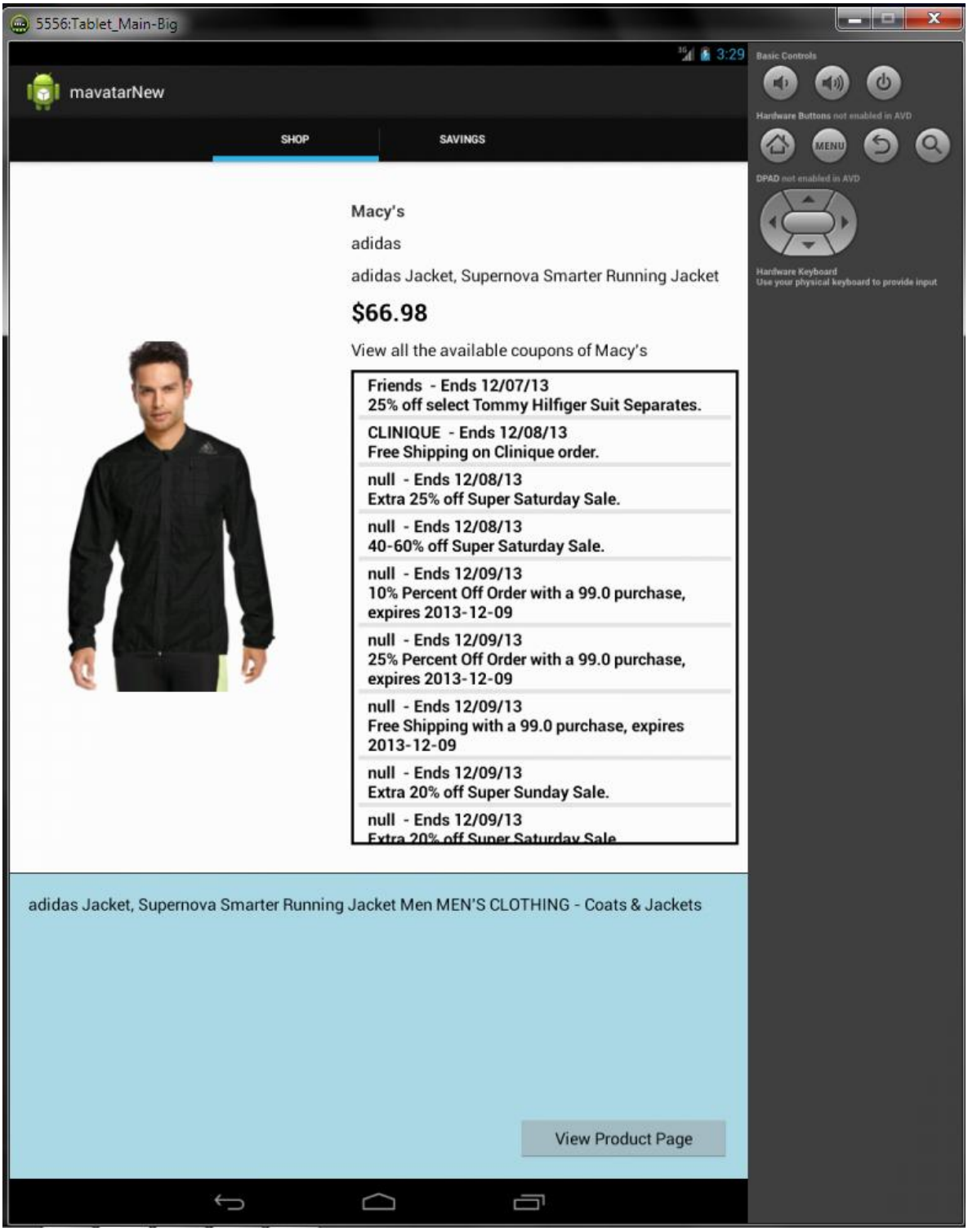
Μπορεί να δει:

- Την εικόνα του προϊόντος
- Το κατάστημα στο οποίο ανήκει το προϊόν
- Την μάρκα του προϊόντος
- Την τιμή του προϊόντος
- Αναλυτική περιγραφή του προϊόντος
- Όλα τα διαθέσιμα κουπόνια του καταστήματος στο οποίο ανήκει το προϊόν (όπως αυτά περιγράφηκαν αναλυτικά στην παράγραφο 5.3)

Επίσης, μπορεί να πατήσει στο ειδικό πλήκτρο View Product Page για να μεταφερθεί στην διαδικτυακή σελίδα του προϊόντος.

```
gridView.setOnItemClickListener(new OnItemClickListener() {

            @Override
            public void onItemClick(AdapterView<?> parent, View v,
                                    int position, long id) {

              /* When clicked we replace our Fragment with a new one with *
               * products based on the category clicked */
              ShopFragmentProductPage frag = new ShopFragmentProductPage();

              HashMap<String, String> mymap = new HashMap<String, String>();
              mymap=(HashMap<String, String>)
gridView.getAdapter().getItem(position);
[…]
              FragmentManager fragmentManager = getFragmentManager();
              FragmentTransaction ft = fragmentManager.beginTransaction();
              ft.replace(android.R.id.content, frag);
              ft.addToBackStack(null);
              ft.commit();
            }
      });
```

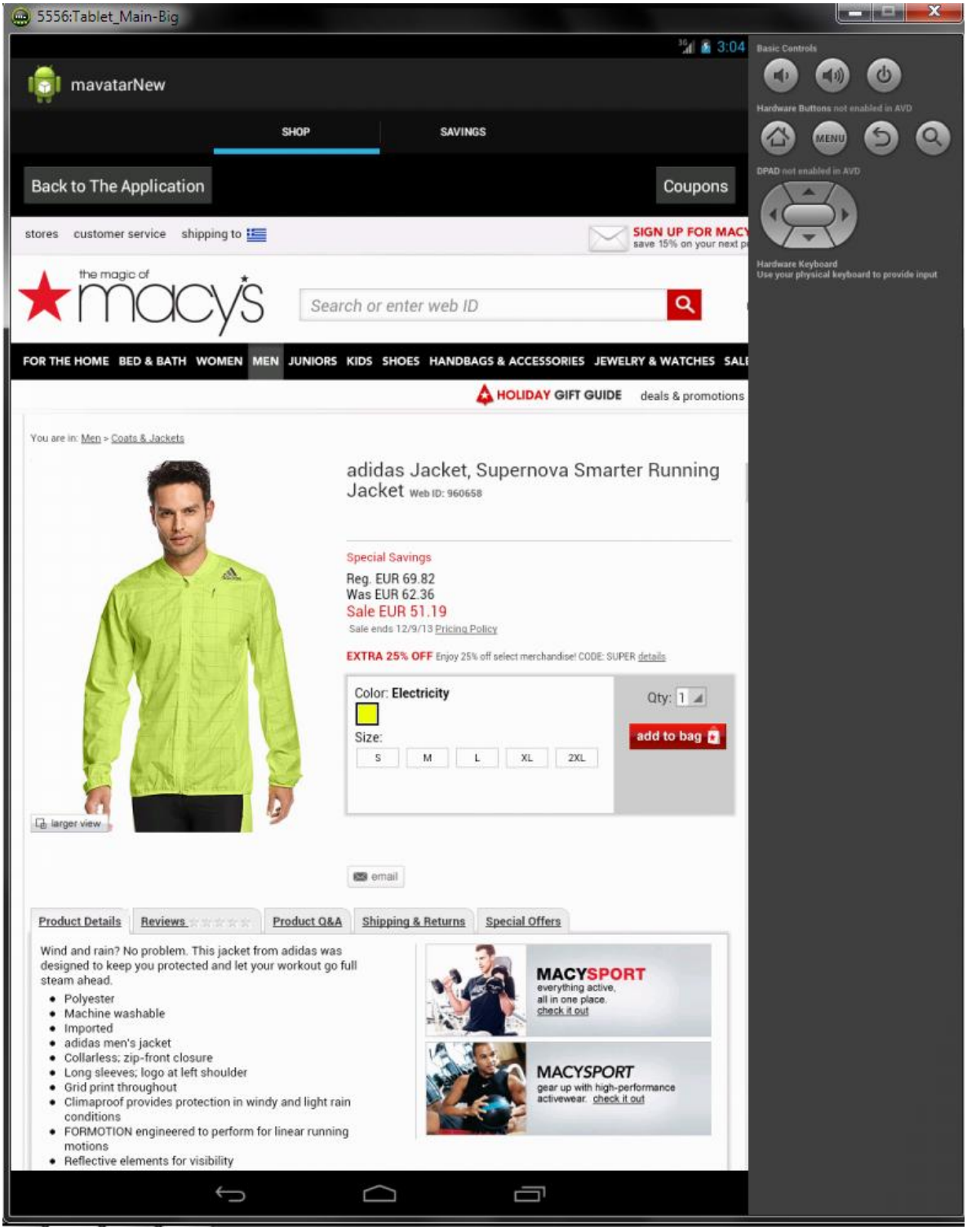
Αν για παράδειγμα, στην οθόνη που φαίνεται στην **Εικόνα 5.34** ο χρήστης πατήσει στο πρώτο προϊόν της δεύτερης σειράς, η οθόνη που θα εμφανιστεί εμφανίζεται παρακάτω στην **Εικόνα 5.38**.



Εικόνα 5.38: Η οθόνη του Shop Tab όταν ο χρήστης πατήσει πάνω σε κάποιο προϊόν

Αφού ο χρήστης μελετήσει αναλυτικά τα χαρακτηριστικά του προϊόντος και δει και τα διαθέσιμα κουπόνια του καταστήματος στο οποίο ανήκει, μπορεί να πατήσει στο πλήκτρο View Product Page για να κάνει την αγορά του. Αυτό φαίνεται παρακάτω στην **Εικόνα 5.39** και ακολουθούν εικόνες με όλη την ροή της εφαρμογής μέχρι και την τελική αγορά.

5.4.2.2.1 Πατώντας το πλήκτρο View Product Page – Το WebProductPageFragment και σενάριο αγοράς προϊόντος



Εικόνα 5.39: Η ιστοσελίδα του προϊόντος η οποία έχει ανοίξει μέσα στην εφαρμογή

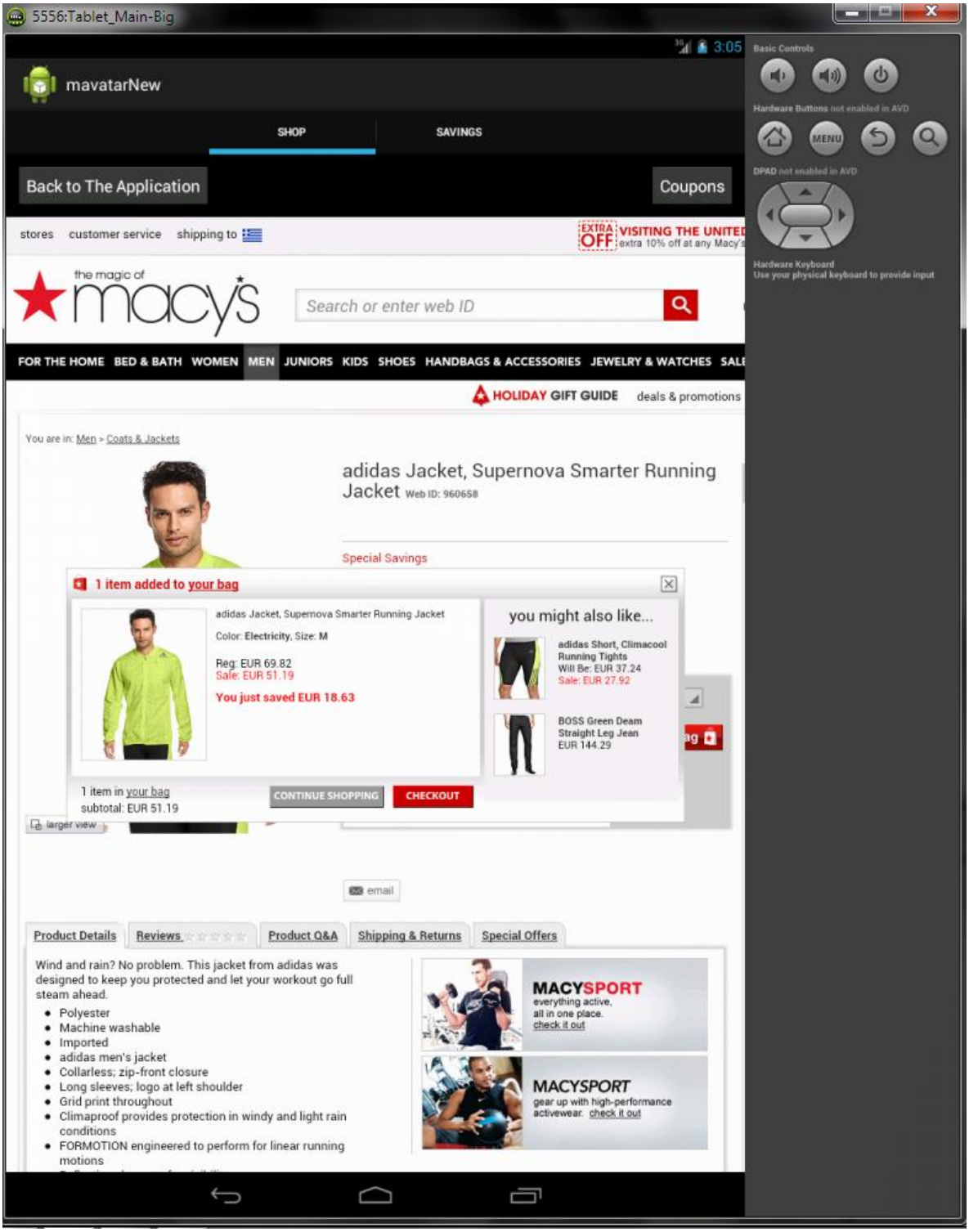
Στην **Εικόνα 5.39** παρατηρείται ότι άνοιξε η διαδικτυακή ιστοσελίδα του προϊόντος μέσα στην εφαρμογή. Όταν ο χρήστης πατήσει το View Product Page το

ShopFragmentProductPage δίνει τη θέση του στο WebProductPageFragment. Αυτό το Fragment περιέχει μια μπάρα στο πάνω μέρος με δύο πλήκτρα. Το πλήκτρο Back To The Application που επιστρέφει το χρήστη στην αρχική οθόνη του Shop Tab και το πλήκτρο Coupons με το οποίο εμφανίζει τα διαθέσιμα κουπόνια του καταστήματος. Από κάτω υπάρχει η ιστοσελίδα του προϊόντος η οποία έχει ανοίξει μέσα στην εφαρμογή. Αυτό επιτυγχάνεται μέσω ενός WebView.

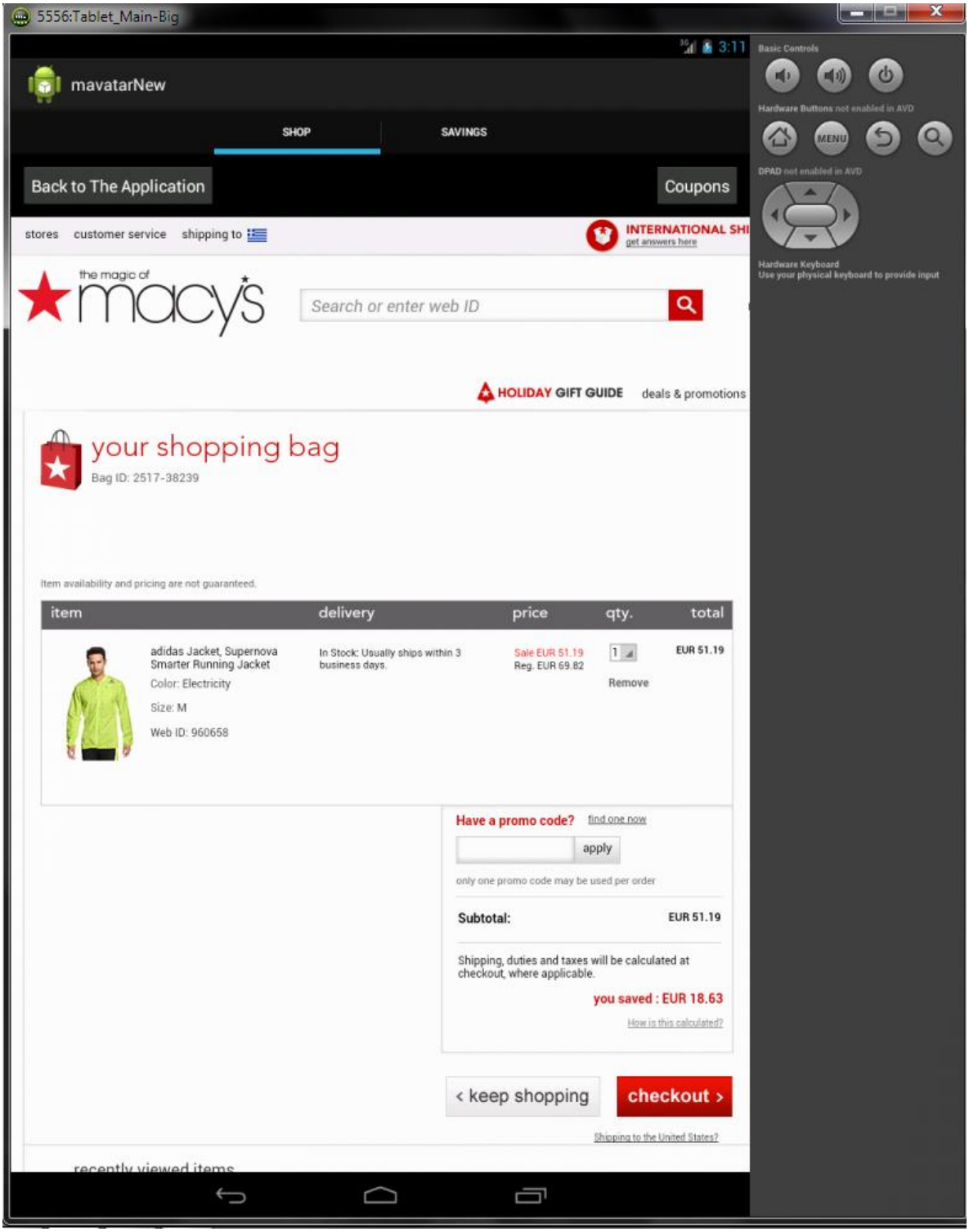
Το WebView είναι ένα View το οποίο χρησιμοποιείται για να φορτώνει διαδικτυακές ιστοσελίδες μέσα σε ένα Layout. Αν δεν είχε γίνει χρήση αυτού του μέσου, η ιστοσελίδα θα άνοιγε σε ξεχωριστό browser της συσκευής του χρήστη. Για την καλύτερη χρήση της εφαρμογής και για να μπορεί ο χρήστης να χρησιμοποιήσει κατευθείαν τα κουπόνια μέσα από αυτήν κατά την αγορά του, επιλέχθηκε η χρήση του WebView. Ακολουθεί ένα μικρό κομμάτι κώδικα με την αρχικοποίηση του WebView:

```
  myWebView = (WebView)
getActivity().findViewById(R.id.ProductPageWebView);

  myWebView.clearHistory();
  myWebView.setWebViewClient(new MyWebViewClient());
  WebSettings webSettings = myWebView.getSettings();
  webSettings.setJavaScriptEnabled(true);
  webSettings.setDisplayZoomControls(true);
  webSettings.setBuiltInZoomControls(true);

  myWebView.loadUrl(vendor_catalog_url);
```

Ο χρήστης τώρα για να προχωρήσει στην αγορά του προϊόντος που φαίνεται στην **Εικόνα 5.39** θα πρέπει να πατήσει στην επιλογή add to bag. Έπειτα αφού πατήσει στην επιλογή Checkout (**Εικόνα 5.40**) θα μεταφερθεί στην σελίδα αγοράς (**Εικόνα 5.41**).



Εικόνα 5.40: Η ιστοσελίδα του προϊόντος η οποία έχει ανοίξει μέσα στην εφαρμογή – Επιλογή αγοράς

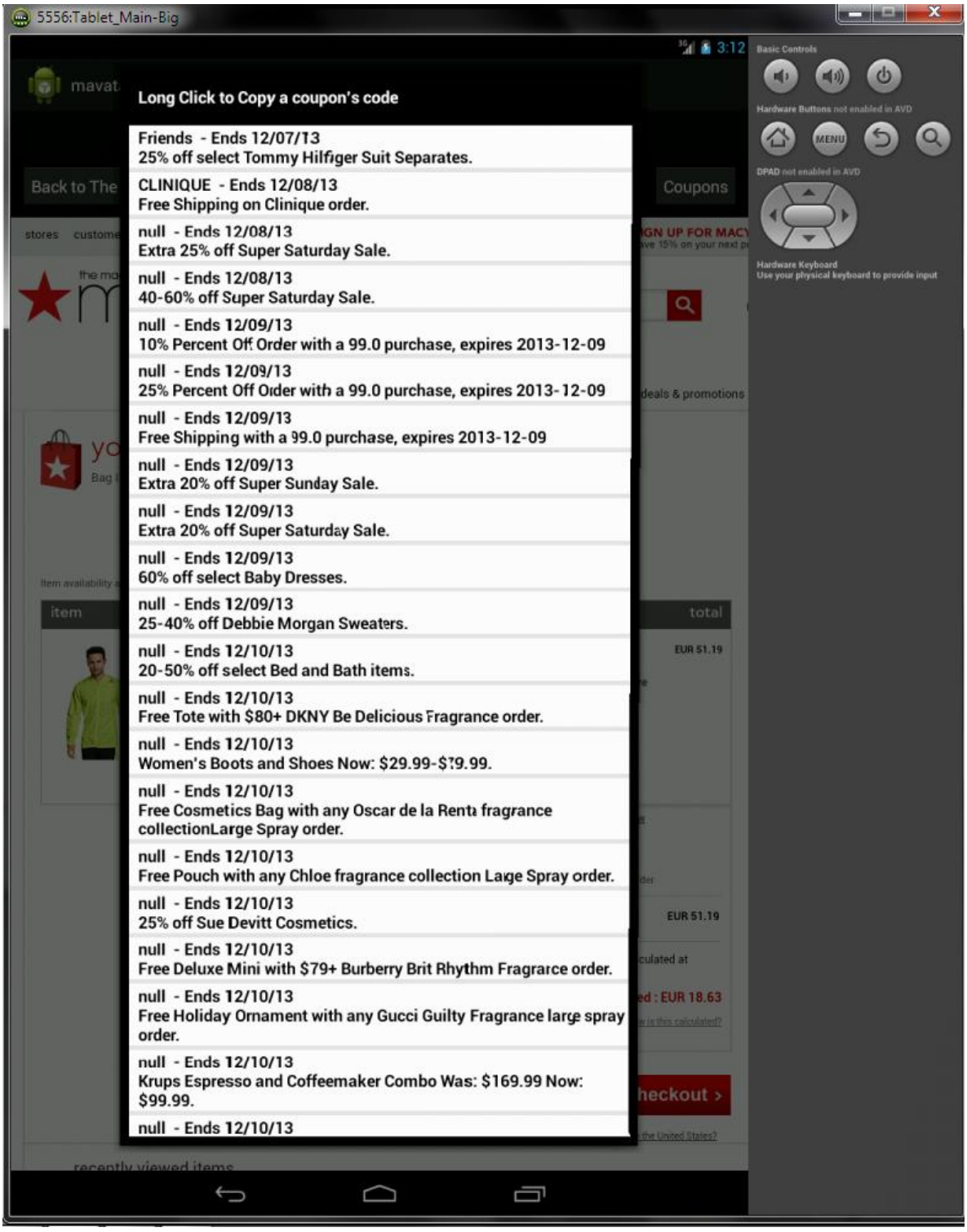


Εικόνα 5.41: Η ιστοσελίδα του προϊόντος η οποία έχει ανοίξει μέσα στην εφαρμογή – Η τελική σελίδα για την αγορά

Στην **Εικόνα 5.41** φαίνεται η σελίδα στην οποία ο χρήστη μπορεί να κάνει την αγορά του. Κανονικά, θα πατούσε στο κόκκινο πλήκτρο checkout και θα πλήρωνε το ποσό το οποίο εμφανίζεται. Εδώ, θα γίνει εμφανής η λειτουργία των εκπτωτικών κουπονιών.

Αν ο χρήστης πατήσει στο πλήκτρο Coupons, θα εμφανιστεί ένας διάλογος Dialog Fragment, ο **WebCouponDialogFragment**. Εκεί, αφού γίνει η σύνδεση με τον server μέσω AsyncTask

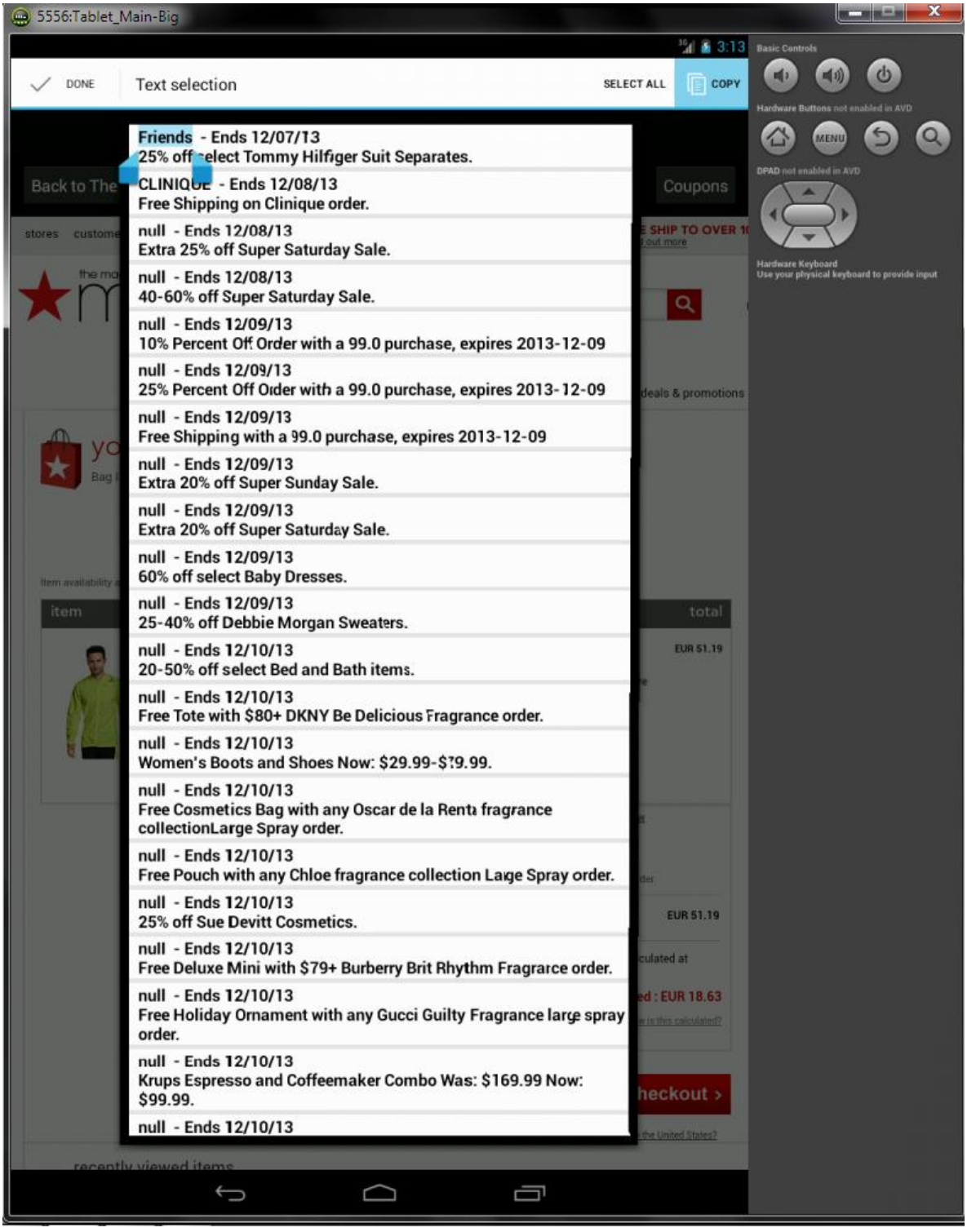
για να αντληθούν οι πληροφορίες για τα κουπόνια του καταστήματος, εμφανίζονται σε μια λίστα όλα τα διαθέσιμα κουπόνια. Αυτό φαίνεται παρακάτω στην **Εικόνα 5.42**.



Εικόνα 5.41: Το WebCouponDialogFragment με όλα τα διαθέσιμα κουπόνια του καστήματος Macy's

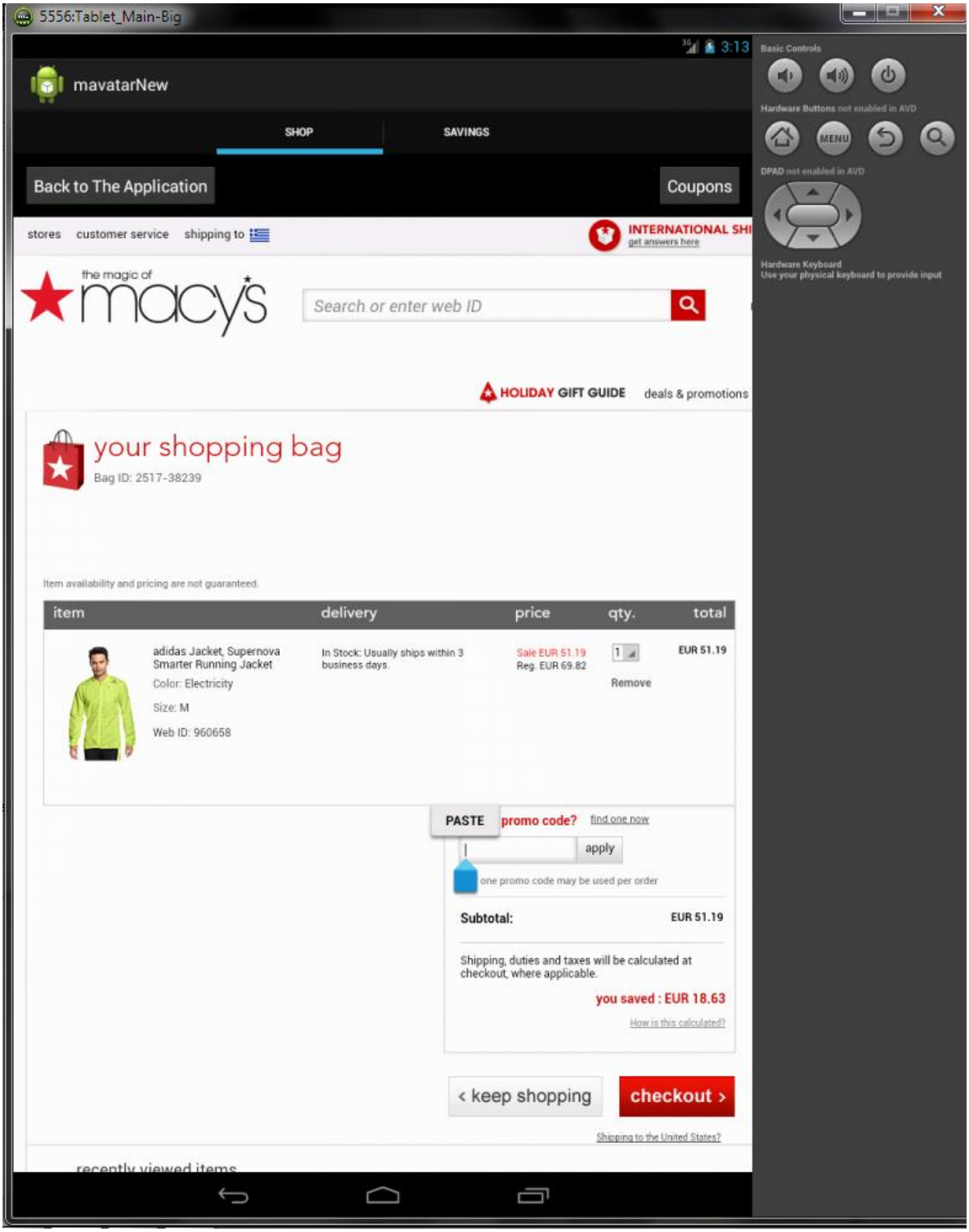
Μιας και το κατάστημα του προϊόντος είναι το Macy's, μπορεί κανείς να παρατηρήσει ότι τα κουπόνια που φαίνονται στη λίστα της **Εικόνα 5.41** είναι τα ίδια με αυτά της **Εικόνας 5.10** στην παράγραφο 5.3. Τώρα, ο χρήστης μπορεί, όπως τον προτρέπει και το παράθυρο, να κάνει

Long Click, δηλαδή να πατήσει παρατεταμένα πάνω σε κάποιο κουπόνι. Τότε, εμφανίζεται η Action Bar η χρήση της οποίας περιγράφηκε σε προηγούμενη παράγραφο. Αυτή τη φορά η Action Bar δεν έχει παραμετροποιηθεί, αλλά είναι όπως την διαθέτει το Android και εμφανίζεται αυτόματα με συγκεκριμένες ενέργειες – όπως το Long Click. Αυτό φαίνεται στην παρακάτω **Εικόνα 5.42**.

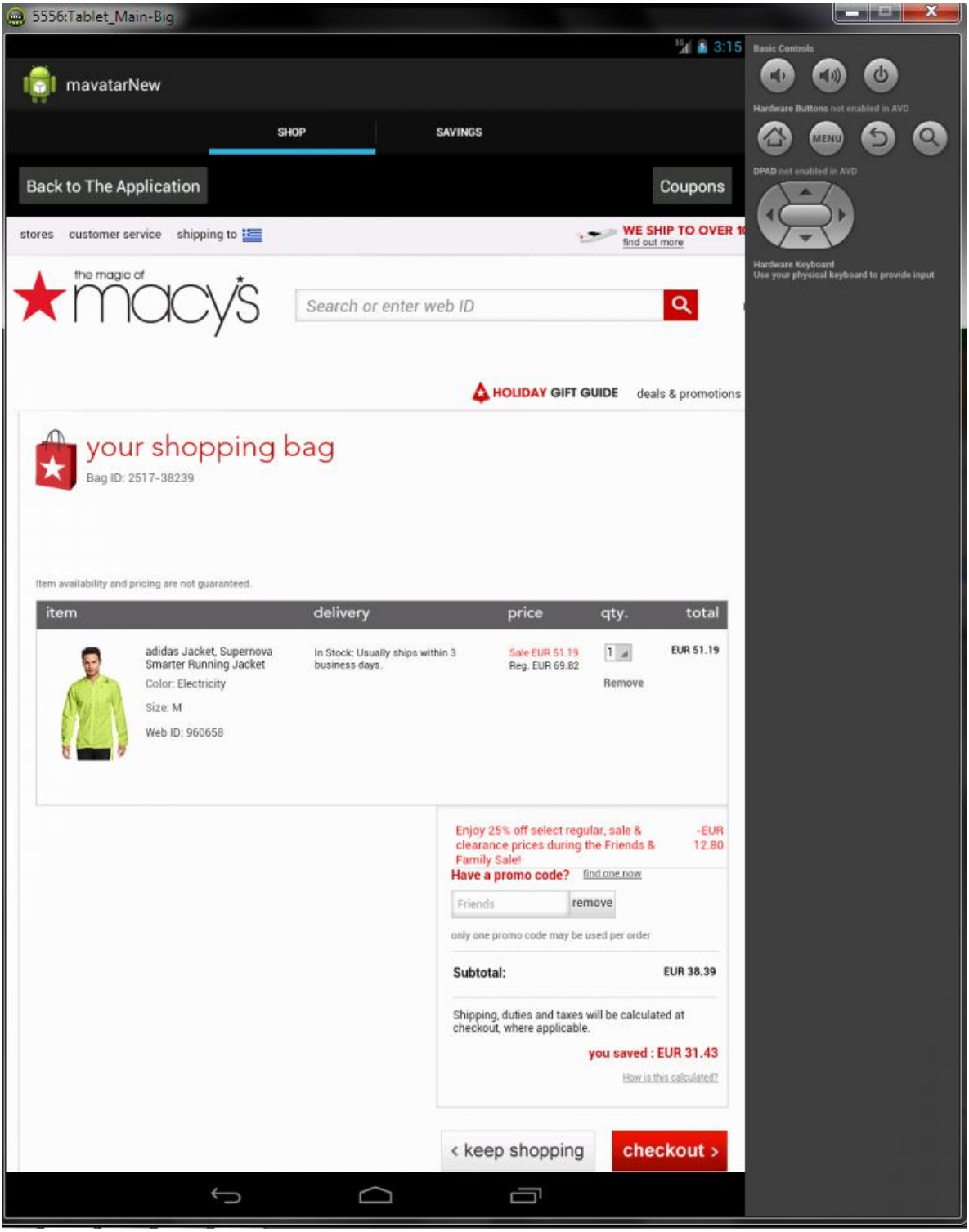


Εικόνα 5.42: Το WebCouponDialogFragment και η Action Bar για αντιγραφή του κωδικού του κουπονιού

Αφού ο χρήστης κάνει Long Click και αντιγράψει τον κωδικό του κουπονιού που επιθυμεί (πχ Friends από το πρώτο κουπόνι στην περίπτωση της **Εικόνας 5.42**) μπορεί να επικολλήσει αυτόν τον κωδικό στο κατάλληλο πεδίο για να ενεργοποιήσει την έκπτωση. Η επικόλληση γίνεται και πάλι με τη χρήση Long Click. Αυτό φαίνεται παρακάτω στις **Εικόνες 5.43 – 5.44**.



Εικόνα 5.43: Η ιστοσελίδα για την αγορά του προϊόντως – επικόλληση του κωδικού του κουπονιού



Εικόνα 5.44: Η ιστοσελίδα για την αγορά του προϊόντως – ενεργοποίηση μέσω του κωδικού του κουπονιού

Όπως φαίνεται στην παραπάνω **Εικόνα 5.44** ο κωδικός που αντέγραψε ο χρήστης έχει επικολληθεί στο κατάλληλο πεδίο και μάλιστα έχει προσμετρηθεί η κατάλληλη έκπτωση, μειώνοντας την τελική τιμή του προϊόντος. Ο χρήστης μπορεί τώρα ελεύθερα να κάνει την αγορά του με μειωμένη τιμή.

5.4.3 Η λειτουργία αναζήτησης

Όπως αναφέρθηκε στις προηγούμενες παραγράφους, τόσο στα ShopFragment όσο και στο ShopFragmentProducts ο χρήστης είχε τη δυνατότητα, μέσω της λειτουργίας αναζήτησης, να αναζητήσει συγκεκριμένα προϊόντα. Αυτό επιτυγχάνεται με τη χρήση ενός SearchView, ενός πεδίου στο οποίο ο χρήστης εισάγει αυτό που θέλει να αναζητήσει και έπειτα η εφαρμογή μας ενεργοποιεί την λειτουργία αναζήτησης.

Ουσιαστικά, η λειτουργία αναζήτησης είναι μια ακόμα κλήση στον server. Στην κλήση αυτή που γίνεται μέσω μιας Http σύνδεσης τύπου POST ζητούνται προϊόντα, όπως και στην απλή κλήση για προϊόντα του ShopFragmentProducts, με την προσθήκη στο αρχείο JSON ενός κατάλληλου πεδίου fts (free text search) το οποίο περιέχει τις λέξεις κλειδιά που χρησιμοποίησε ο χρήστης στην αναζήτηση του. Επίσης, είναι προφανές ότι μόλις ο χρήστης ενεργοποιήσει την λειτουργία αναζήτησης, καλείται στην οθόνη το ShopFragmentProducts εκτός και αν η αναζήτηση γίνει από εκεί, τότε απλώς ανανεώνει το περιεχόμενο του.

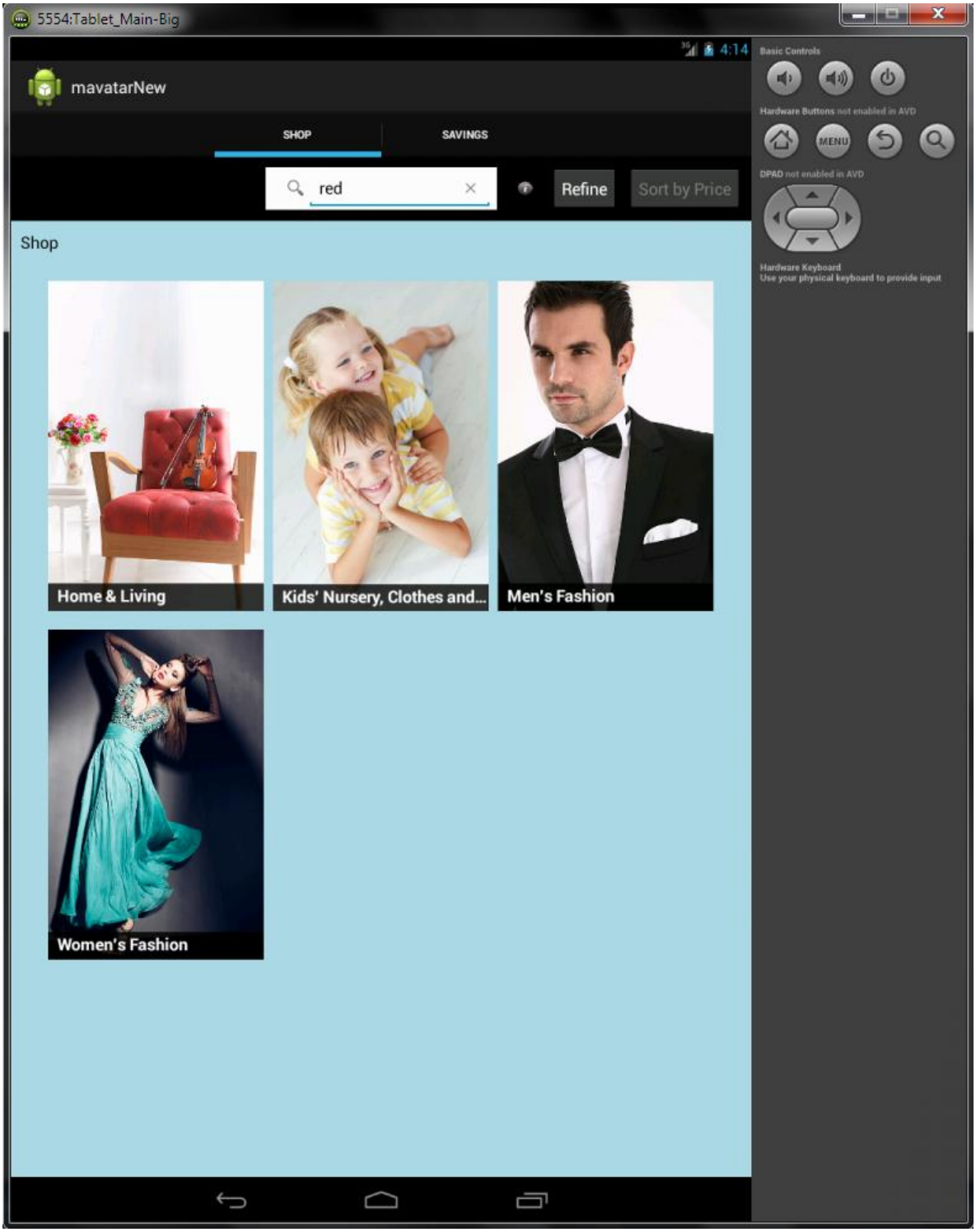
Η μέθοδος και ο Listener που χρησιμοποιείται για να ενεργοποιήσει την λειτουργία αναζήτησης:

```
      /*Method defining what to do when user uses the search function*/
      final SearchView.OnQueryTextListener queryTextListener = new
SearchView.OnQueryTextListener() {
          @Override
          public boolean onQueryTextChange(String newText) {
              // Do something
              return true;
          }

          @Override
          public boolean onQueryTextSubmit(String query) {
```

Η κλήση προς τον server μαζί με το κατάλληλο πεδίο αναζήτησης:

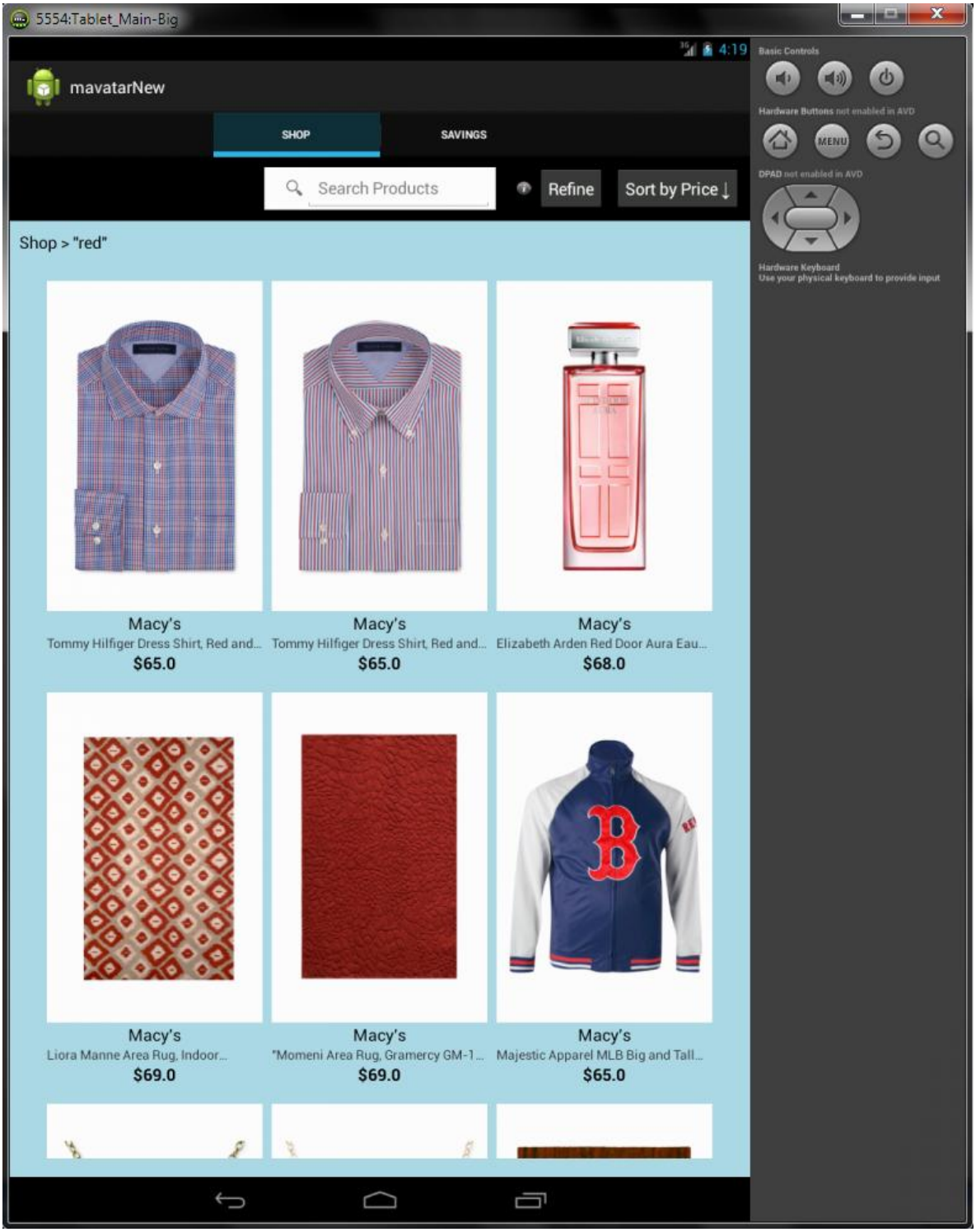
```
HttpPost httppost = new HttpPost();
JSONObject json = new JSONObject();
if (params[1].equals("search")){
      httppost = new HttpPost("http://vm9.stage-
mavatar.com/category_products");
      json.put("fts", query);
```

Στην **Εικόνα 5.45** που ακολουθεί φαίνεται η εισαγωγή δεδομένων προς αναζήτηση από τον χρήστη. Ο χρήστης εισάγει την λέξη-κλειδί red (κόκκινο) για να αναζητήσει μόνο προϊόντα με κόκκινο χρώμα.



Εικόνα 5.45: Ο χρήστης εισάγει δεδομένα στο πεδίο αναζήτησης

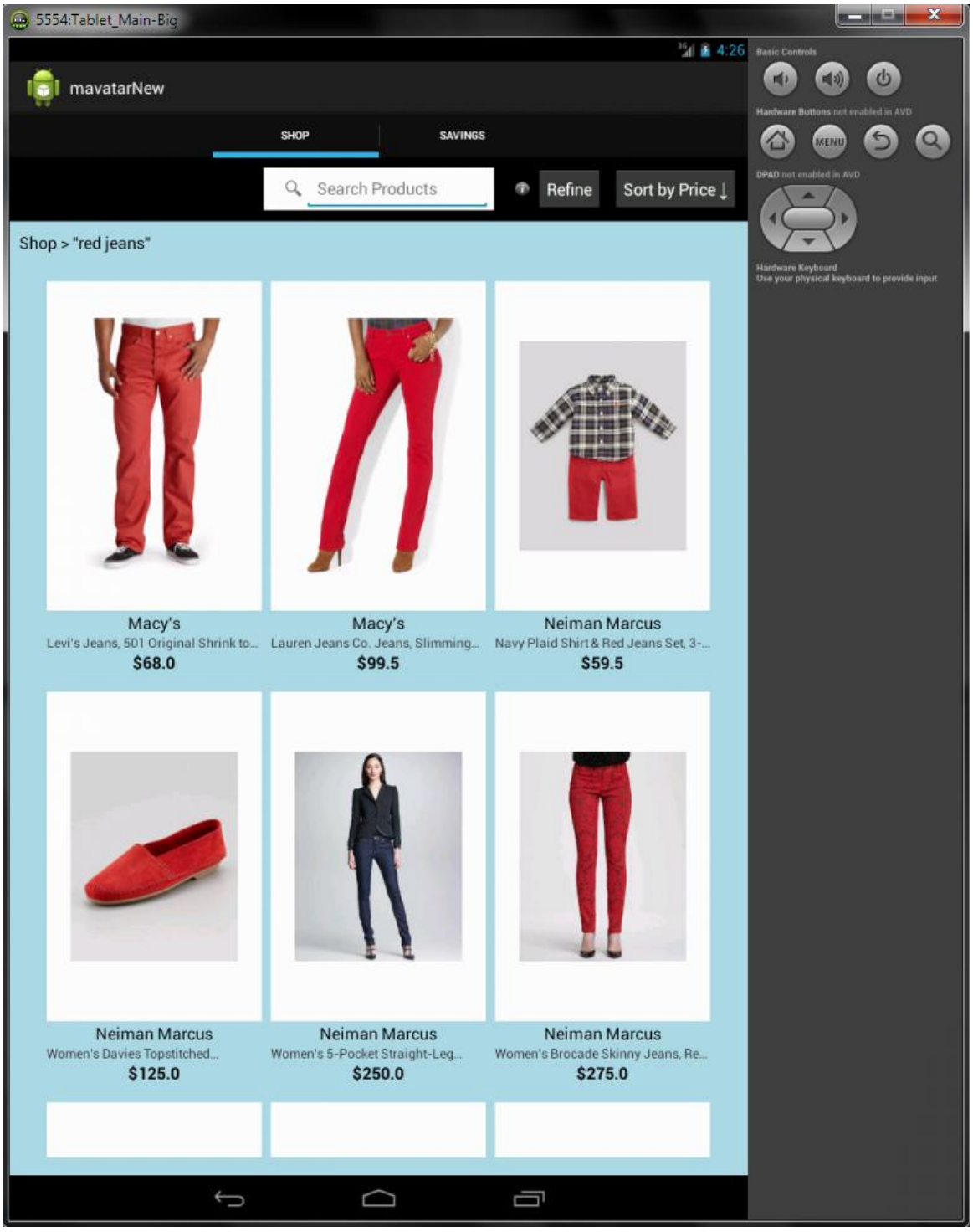
Στην **Εικόνα 5.46** που ακολουθεί φαίνονται τα αποτελέσματα της παραπάνω αναζήτησης.



Εικόνα 5.46: Τα αποτελέσματα της αναζήτησης με λέξη κλειδί red (κόκκινο)

Παρατηρείται ότι όντως τα προϊόντα που εμφανίζονται έχουν άμεση σχέση με το κόκκινο χρώμα και η αναζήτηση ήταν επιτυχής. Επίσης, στο πάνω μέρος του πλέγματος φαίνεται ότι το δένδρο κατηγοριών έχει ανανεωθεί με την προσθήκη της λέξης αναζήτησης (Shop > "red"). Πρέπει επίσης να αναφερθεί ότι η λειτουργία της αναζήτησης δουλεύει σε «τοπικό» επίπεδο. Στο παραπάνω παράδειγμα η αναζήτηση έγινε από την αρχική σελίδα του Shop Tab οπότε αναζητήθηκαν κόκκινα προϊόντα σε όλες τις διαθέσιμες κατηγορίες. Αν η αναζήτηση

είχε γίνει για παράδειγμα μέσα στην κατηγορία Women's Fashion θα γινόταν αναζήτηση για κόκκινα προϊόντα μόνο μέσα στην κατηγορία αυτή (άρα δε θα εμφανίζονταν τα αντρικά πουκάμισα κλπ που φαίνονται στην παραπάνω εικόνα, αλλά πχ κόκκινα γυναικεία ρούχα και το δένδρο κατηγοριών θα γινόταν Shop > Women's Fashion > "jeans"). Ακόμα, η αναζήτηση δεν είναι αναγκαίο να περιέχει μόνο μια λέξη αλλά μπορεί να γίνει ακόμα πιο συγκεκριμένη. Για παράδειγμα, αν ο χρήστης θέλει να ψάξει πιο συγκεκριμένα για κόκκινα τζιν αρκεί να εισάγει "red jeans". Το αποτέλεσμα μιας τέτοιας αναζήτησης φαίνεται στην παρακάτω **Εικόνα 5.47**.



Εικόνα 5.47: Τα αποτελέσματα της αναζήτησης με λέξεις κλειδία red jeans

5.4.4 Η λειτουργία Refine

Όπως αναφέρθηκε στις προηγούμενες παραγράφους, τόσο στα ShopFragment όσο και στο ShopFragmentProducts ο χρήστης είχε τη δυνατότητα να φιλτράρει τα προϊόντα μέσω του πλήκτρου Refine. Το πλήκτρο Refine, όταν πατηθεί, εμφανίζει ένα παράθυρο – διάλογο τύπου Dialog Fragment, το **RefineDialogFragment**. Το παράθυρο αυτό περιέχει τρία διαφορετικά Tabs με συγκεκριμένες επιλογές στο καθένα. Το κάθε Tab είναι και αυτό από μόνο του ένα ξεχωριστό Fragment, είναι δηλαδή εμφωλιασμένα Fragment μέσα σε ένα Dialog Fragment.

- Το **Store** Tab (Tab1Fragment) το οποίο περιέχει όλα τα διαθέσιμα καταστήματα.
- Το **Manufacturer** Tab (Tab2Fragment) το οποίο περιέχει όλες τις διαθέσιμες μάρκες (κατασκευαστές) προϊόντων.
- Το **Price** Tab (Tab3Fragment) το οποίο περιέχει δύο κατάλληλα πεδία ώστε ο χρήστης να αναζητήσει προϊόντα σε ένα επιθυμητό εύρος τιμών.

Το παράθυρο του RefineDialogFragment περιέχει επίσης τρία βασικά πλήκτρα:

- Το πλήκτρο Done – Όταν πατηθεί καταχωρούνται οι επιλογές του χρήστη
- Το πλήκτρο Clear Tab – Όταν πατηθεί καθαρίζουν όλες οι επιλογές του χρήστη σε αυτό το συγκεκριμένο Tab. Δεν καταχωρούνται αν δεν πατηθεί το πλήκτρο Done
- Το πλήκτρο Clear All – Όταν πατηθεί καθαρίζονται όλες οι επιλογές από όλα τα Tab, το παράθυρο κλείνει και η εφαρμογή ανανεώνει τα δεδομένα.

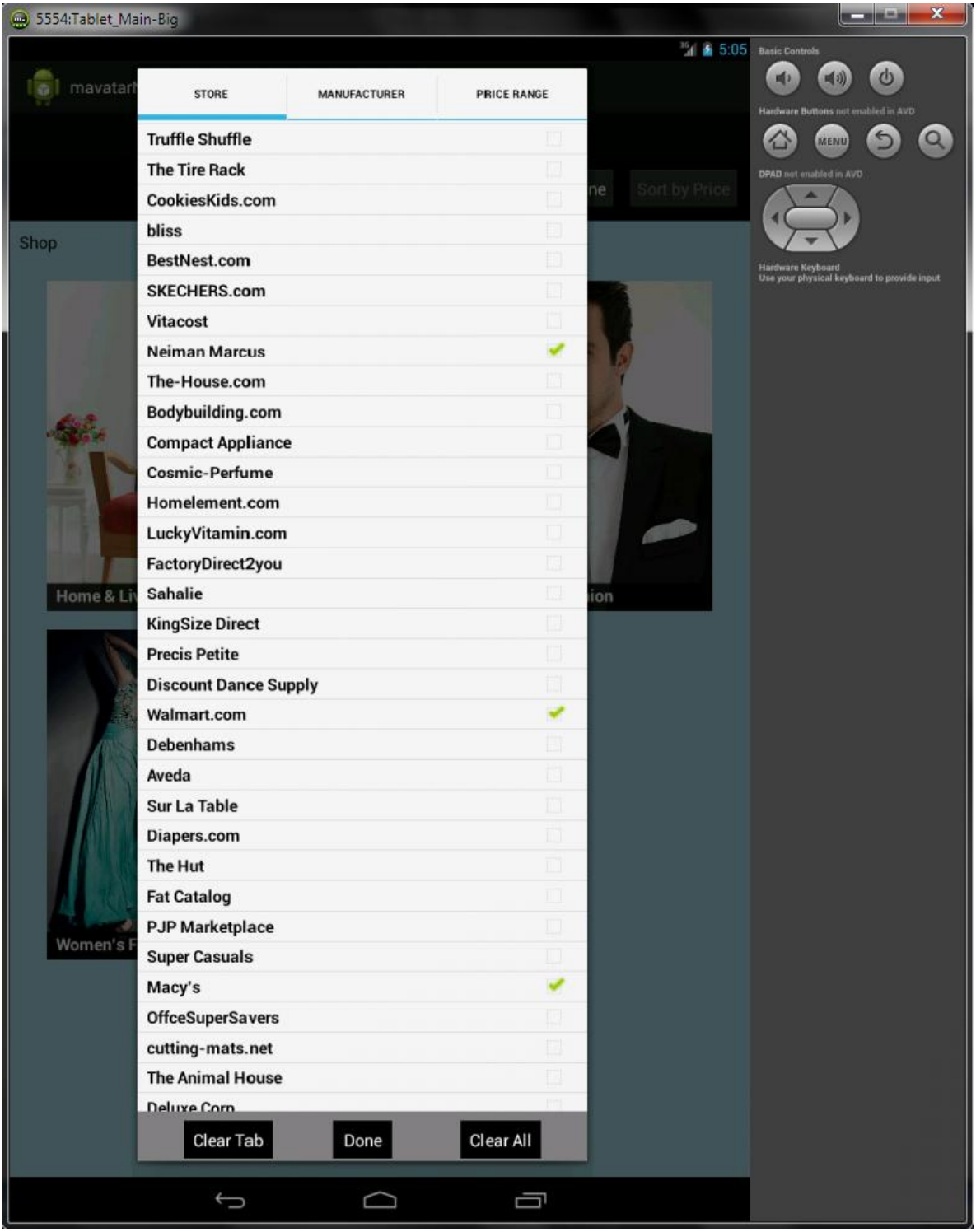
Σε αυτό το σημείο πρέπει να αναφερθεί ότι αν ο χρήστης πατήσει το πλήκτρο Done τότε οι επιλογές καταχωρούνται σε ειδικές global μεταβλητές τις οποίες χειρίζεται η κλάση **RefineDialogFilter**. Αυτό είναι χρήσιμο διότι όταν ο χρήστης ξαναπατήσει στο πλήκτρο Refine, στην λίστα που θα εμφανιστεί, τα προηγουμένως επιλεγμένα καταστήματα θα είναι ακόμα επιλεγμένα. Η λειτουργία Refine δηλαδή «θυμάται» τις επιλογές του χρήστη. Επίσης, η λειτουργία Refine είναι ανεξάρτητη της οθόνης στην οποία βρίσκεται ο χρήστης. Αν δηλαδή ο χρήστης πατήσει σε κάποια κατηγορία και προχωρήσει σε κάποια επόμενη οθόνη, οι επιλογές που έχει κάνει θα είναι ακόμα ενεργές. Έτσι, όταν καταλήξει σε οθόνη με προϊόντα (ShopFragmentProducts) τότε τα προϊόντα που θα φορτωθούν θα είναι σύμφωνα με τις επιλογές που έχει κάνει ο χρήστης. Γι αυτό και η λειτουργία Refine ονομάζεται και «φιλτράρισμα». Ο χρήστης λοιπόν μπορεί είτε να ενεργοποιήσει την επιλογή Refine εξ αρχής και να περιηγηθεί στις κατηγορίες και τα προϊόντα που θα εμφανιστούν να είναι σύμφωνα με τις επιλογές του, είτε να αναζητήσει πρώτα κάποια προϊόντα και έπειτα στην σελίδα με τα προϊόντα να φιλτράρει όλα τα αποτελέσματα αυτά όπως επιθυμεί.

5.4.4.1 To Store Tab

Το Store Tab είναι το πρώτο και προεπιλεγμένο Tab που εμφανίζεται μόλις ο χρήστης πατήσει το πλήκτρο Refine. Το Store Tab περιέχει μια λίστα με όλα τα διαθέσιμα καταστήματα τα οποία περιέχει ο server. Οι πληροφορίες για τα καταστήματα αυτά αντλούνται μέσω μιας Http σύνδεσης τύπου POST με τον server και η οποία γίνεται μέσω της λειτουργίας AsyncTask. Τα κελιά της λίστας αυτής περιέχουν το όνομα του κάθε καταστήματος αλλά ταυτόχρονα είναι σχεδιασμένα με βάση το CheckableRelativeLayout το οποίο αναφέρθηκε στην παράγραφο 5.3.2.2.2. Έτσι, όταν ο χρήστης πατήσει σε κάποιο κατάστημα, ένα πράσινο «τσεκ» εμφανίζεται ώστε να ενημερώσει το χρήση ότι το κατάστημα επιλέχθηκε. Ο χρήστης μπορεί να επιλέξει όσα καταστήματα επιθυμεί και έπειτα να καταχωρήσει την επιλογή του με το πλήκτρο Done. Ακολουθεί ένα μέρος του κώδικα:

```
/* If user has opened the dialog again, previously,
* set and show the previously checked items */
if(RefineDialogFilter.checkedItemsfilter!=null){
   for (int i = 0; i < (RefineDialogFilter.checkedItemsfilter).size(); ++i)
{

   listView.setItemChecked((RefineDialogFilter.checkedItemsfilter).keyAt(i),
                       (RefineDialogFilter.checkedItemsfilter).valueAt(i));
   }
}

listView.setOnItemClickListener(new OnItemClickListener() {

           @Override
           public void onItemClick(AdapterView<?> arg0, View arg1, int arg2,
                   long arg3) {

               checkedItems = listView.getCheckedItemPositions();
               int checkedItemsCount = checkedItems.size();
```

Στην **Εικόνα 5.48** που ακολουθεί φαίνεται η οθόνη που εμφανίζεται όταν είναι επιλεγμένο το Store Tab και ο χρήστης έχει επιλέξει διάφορα καταστήματα.

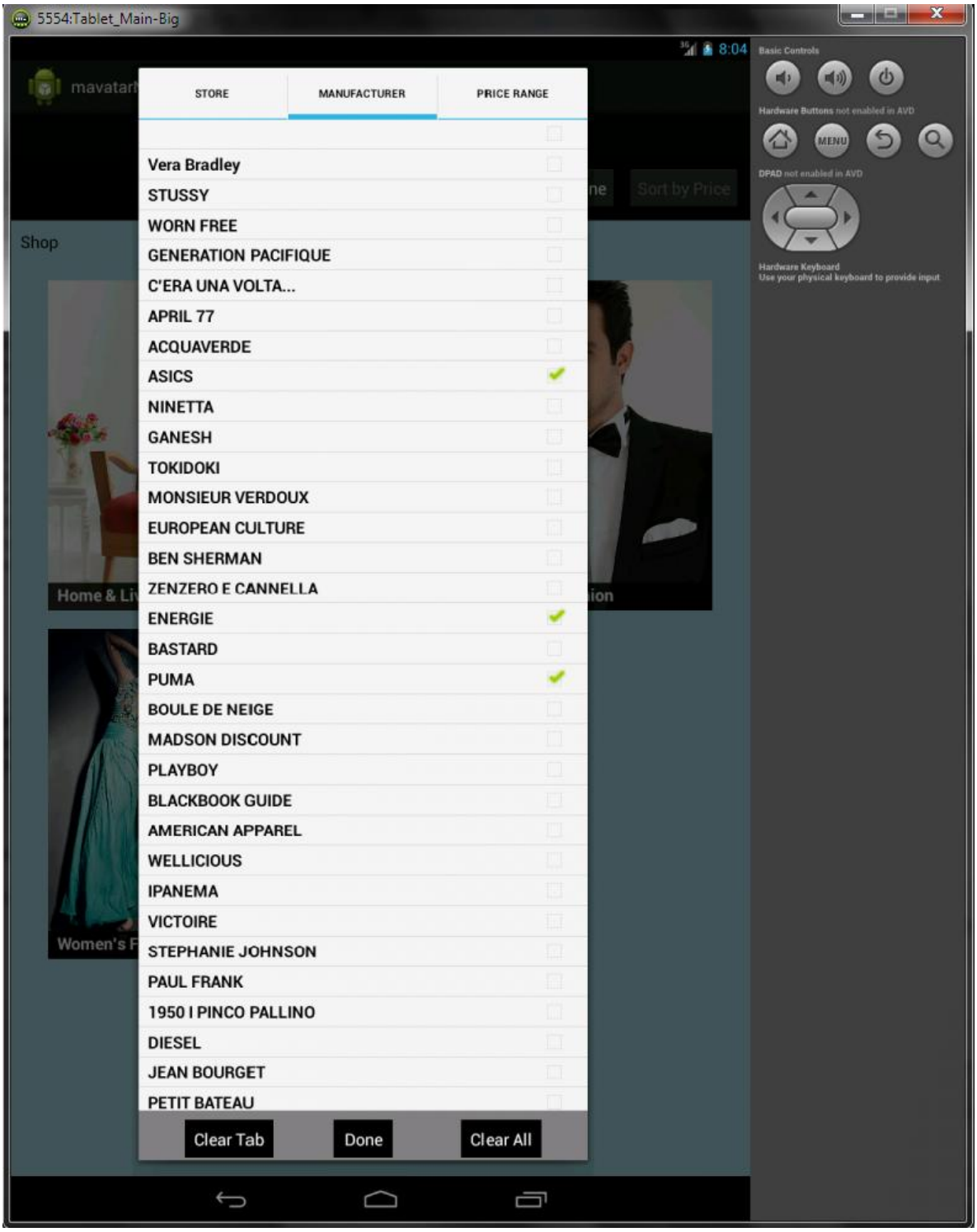


Εικόνα 5.48: Το Store Tab του RefineDialogFragment με διάφορα καταστήματα επιλεγμένα

5.4.4.2 Το Manufacturer Tab

Το Manufacturer Tab περιέχει μια λίστα με όλες τις διαθέσιμες μάρκες προϊόντων που περιέχει ο server. Οι πληροφορίες για τις μάρκες αντλούνται και πάλι μέσω μιας Http

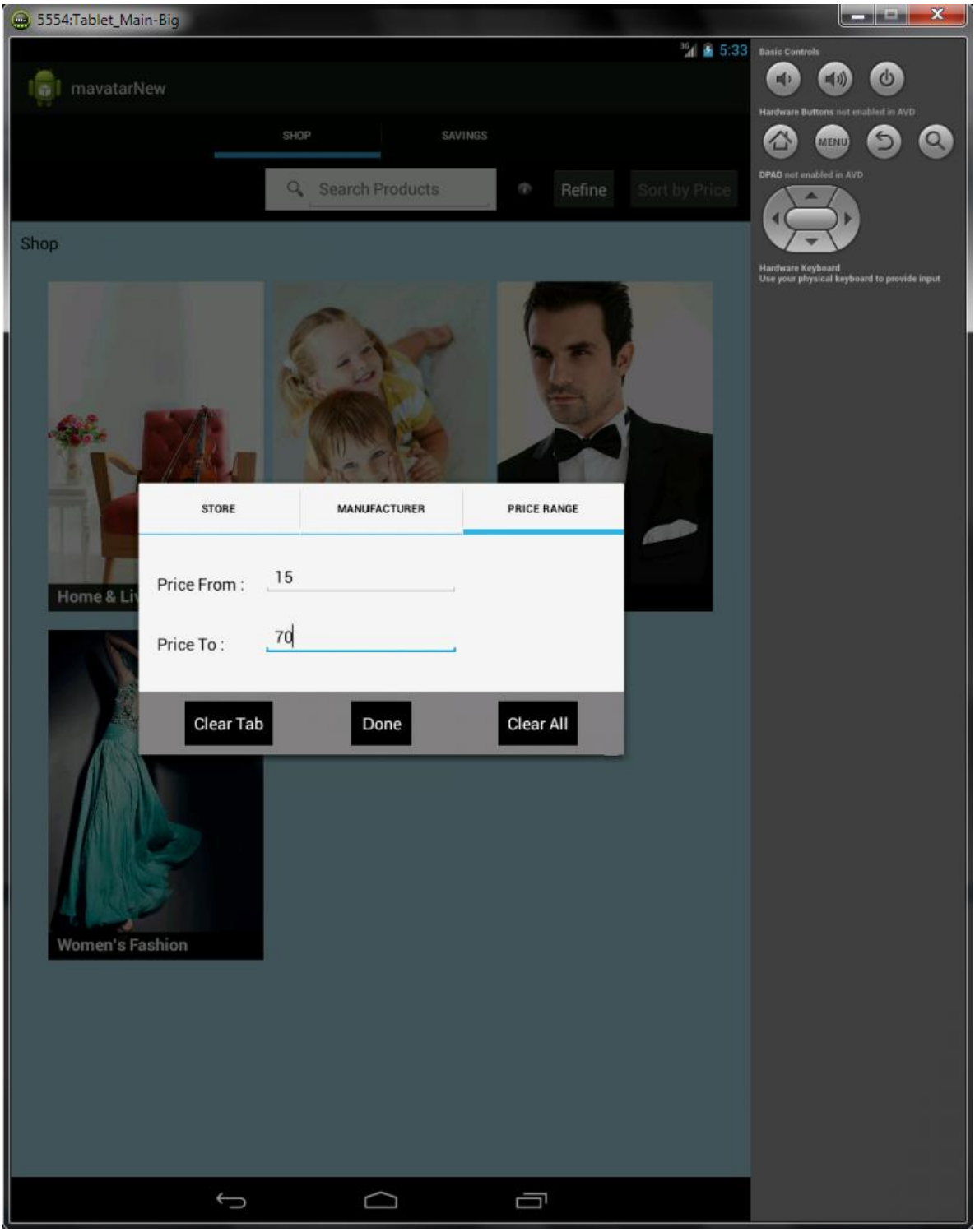
σύνδεσης τύπου POST με τον server και η οποία γίνεται μέσω της λειτουργίας AsyncTask. Οι μάρκες εμφανίζονται και αυτές σε μια λίστα με βάση το CheckableRelativeLayout. Ο χρήστης μπορεί να επιλέξει όσες μάρκες επιθυμεί και έπειτα να καταχωρήσει την επιλογή του με το πλήκτρο Done. Ο κώδικας του είναι ουσιαστικά ο ίδιος με του Store Tab με μόνη διαφορά την κλήση προς τον server. Στην **Εικόνα 5.49** που ακολουθεί φαίνεται το Manufacturer Tab μαζί με διάφορες επιλογές του χρήστη.



Εικόνα 5.49: Το Manufacturer Tab του RefineDialogFragment με διάφορες μάρκες επιλεγμένες

5.4.4.3 Το Price Tab

Το Price Tab χρησιμοποιείται για να θέσει ο χρήστης ένα επιθυμητό εύρος τιμών για τα προϊόντα που θέλει. Περιέχει δύο κατάλληλα πεδία Price From (Τιμή Από) και Price To (Τιμή Έως) στα οποία ο χρήστης εισάγει αριθμητικά τις τιμές που επιθυμεί. Ένα παράδειγμα φαίνεται στην παρακάτω **Εικόνα 5.50**.



Εικόνα 5.50: Το Price Tab του RefineDialogFragment όπου ο χρήστης έχει εισάγει δύο τιμές.

Επίσης, υπάρχει κατάλληλος έλεγχος έτσι ώστε αν ο χρήστης εισάγει Τιμή Από μεγαλύτερη από την Τιμή Έως να εμφανίζεται κατάλληλο μήνυμα Toast για να τον ενημερώνει ότι δεν έχει εισάγει αποδεκτό συνδυασμό τιμών.

5.4.4.4 Παράδειγμα χρήσης της λειτουργίας Refine

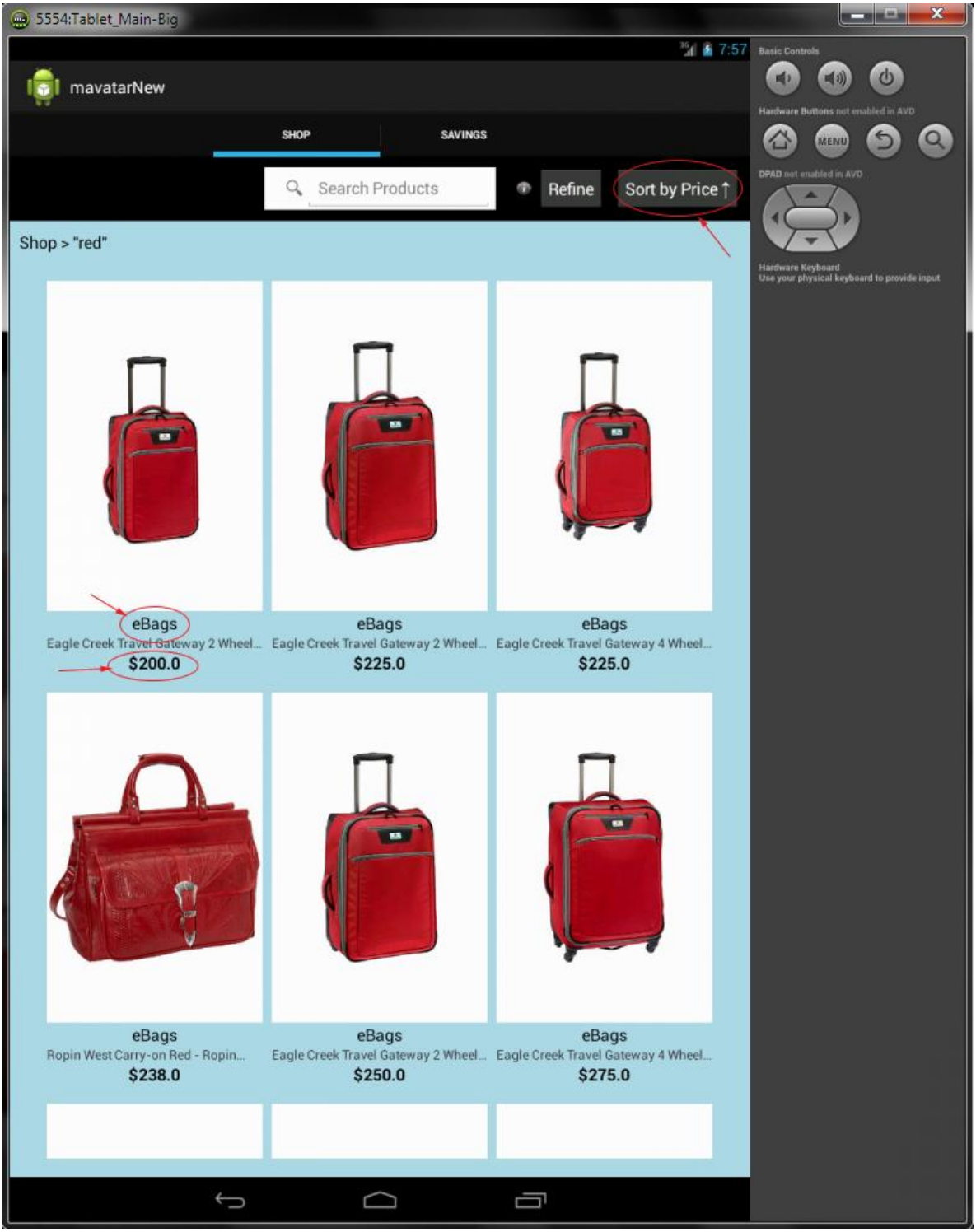
Παρατηρώντας την **Εικόνα 5.46** όπου ο χρήστης έκανε αναζήτηση με την λέξη κλειδί "red", θα παρουσιαστεί ένα παράδειγμα στο οποίο ο χρήστης κάνει και πάλι την ίδια αναζήτηση αλλά αυτή τη φορά έχοντας ενεργοποιήσει την λειτουργία Refine.

- Από το Store Tab επιλέγει το κατάστημα eBags
- Από το Manufacturer Tab δεν επιλέγει τίποτα
- Στο Price Tab εισάγει εύρος τιμών από 200 έως 500

Ακολουθεί το βασικότερο τμήμα του κώδικα στο οποίο γίνονται οι απαραίτητοι έλεγχοι και έπειτα εισάγονται τα κατάλληλα πεδία στο αρχείο JSON που θα σταλεί στον server, με τιμές τα δεδομένα που εισήγαγε ο χρήστης κατά τη λειτουργία Refine.

```
if(pricefromfilter!=null){
      if(!pricefromfilter.equals("")){
          json.put("price_from", Integer.parseInt(pricefromfilter));
      }
}

if(pricetofilter!=null ){
      if(!pricetofilter.equals("")){
          json.put("price_to", Integer.parseInt(pricetofilter));
      }
}

if(storenamesfilter!=null){
      JSONArray refine_vendor = new JSONArray();
      for(int i=0; i<storenamesfilter.length; i++){
          refine_vendor.put(storenamesfilter[i]);
      }
      json.put("vendors", refine_vendor);
}

if(manufacturernamesfilter!=null){
      JSONArray manuf_vendor = new JSONArray();
      for(int i=0; i<manufacturernamesfilter.length; i++){
          manuf_vendor.put(manufacturernamesfilter[i]);
      }
      json.put("mfgs",manuf_vendor);
}
```

Τα αποτελέσματα του παραδείγματος που περιγράφηκε παραπάνω φαίνονται παρακάτω στην **Εικόνα 5.51** όπου έχει ρυθμιστεί και το Sort By Price σε αύξουσα σειρά ώστε να φανούν πιο συγκεκριμένα τα αποτελέσματα της επιλογής του Price Tab.



Εικόνα 5.51: Τα αποτελέσματα της αναζήτησης με λέξη κλειδί red (κόκκινο) μαζί με την λειτουργία Refine ενεργοποιημένη

Είναι προφανές κοιτάζοντας την **Εικόνα 5.51** ότι η λειτουργία Refine έχει αλλάξει άρδην τα αποτελέσματα σε σχέση με την **Εικόνα 5.46**. Παρατηρείται ότι εμφανίζονται προϊόντα μόνο του καταστήματος eBags και από τιμές από 200$ και πάνω (αν ο χρήστης πατήσει άλλη μια φορά στο πλήκτρο Sort By Price θα παρατηρήσει ότι και η μέγιστη τιμή θα είναι τα 500$ όπως και επέλεξε).

6. Επίλογος

Στο παρόν κεφάλαιο σχολιάζεται το αποτέλεσμα της διπλωματικής εργασίας και αναφέρονται δυνατότητες επέκτασης.

6.1 Σύνοψη και συμπεράσματα

Στην παρούσα διπλωματική εργασία αναπτύχθηκε μια εφαρμογή για ταμπλέτες που χρησιμοποιούν το λειτουργικό σύστημα Android, το οποίο όπως αναλύθηκε στο πρώτο και δεύτερο κεφάλαιο, είναι ένα ταχέως αναπτυσσόμενο λογισμικό το οποίο αυτή τη στιγμή κατέχει το μεγαλύτερο μέρος της αγοράς κινητών συσκευών. Επίσης, οι υπολογιστές ταμπλέτες καθιερώνονται καθημερινά όλο και περισσότερο τόσο στην καταναλωτική αγορά όσο και στην καθημερινή χρήση δεκάδων διαφορετικών χρηστών. Αυτό καθιστά την επιλογή της ανάπτυξης της συγκεκριμένης εφαρμογής επιτυχή καθώς ανταποκρίνεται απόλυτα στις τρέχουσες τεχνολογικές εξελίξεις.

Η ενασχόληση με το πεδίο ανάπτυξης εφαρμογών για κινητές συσκευές και πιο συγκεκριμένα για συσκευές με λογισμικό Android κρίνεται άκρως ενδιαφέρουσα. Αποκομοίθηκαν πολύ σημαντικές γνώσεις τόσο για την σχεδίαση όσο και για τον προγραμματισμό εφαρμογών για κινητές συσκευές και έγιναν αντιληπτές οι διαφορές από τον παραδοσιακό προγραμματισμό για σταθερούς υπολογιστές.

Συστήνεται ανεπιφύλαχτα η ενασχόληση με το αντικείμενο της ανάπτυξης εφαρμογών για κινητές συσκευές, σε όλους τους ενδιαφερόμενους φοιτητές. Είναι ένα πεδίο το οποίο έχει την πολυτέλεια να βελτιώνεται, να αναπτύσσεται και να ανανεώνεται συνεχώς με την πάροδο του χρόνου. Αυτό, πέρα από την δημιουργία νέων θέσεων εργασίας, τόσο σε εταιρικό όσο και σε ανεξάρτητο προσωπικό επίπεδο, δημιουργεί και την ανάγκη για εύρεση νέων τεχνολογικών λύσεων στα νέα προβλήματα που θα προκύπτουν. Λύσεις που θα κληθούν να δώσουν οι νέοι προγραμματιστές και μηχανικοί.

6.2 Μελλοντικές επεκτάσεις

Παρά το ότι σε αυτό το στάδιο η εφαρμογή αναπτύχθηκε στα πλαίσια μιας διπλωματικής εργασίας και δεν είχε εμπορικό στόχο, θα μπορούσε εύκολα στο μέλλον μαζί με τις κατάλληλες τροποποιήσεις και επεκτάσεις να γίνει διαθέσιμη μέσω του καταστήματος Play Store.

Η εφαρμογή αναπτύχθηκε με τέτοιο τρόπο που καθιστά την επέκταση της πολύ εύκολη χωρίς να χρειαστούν αλλαγές στον υπάρχον κώδικα, παρά μόνο δημιουργία νέων κλάσεων και

κώδικα που θα περιγράφει αυτές τις προσθήκες. Ακόμα, έχουν δημιουργηθεί ήδη οι βάσεις για συγκεκριμένες λειτουργίες που μπορούν εύκολα να προστεθούν στο μέλλον. Επεκτάσεις που προτείνονται και που είναι εύκολα υλοποιήσιμες είναι:

- Η σύνδεση με social media όπως το Facebook και το Twitter μέσω κατάλληλων πλήκτρων που έχουν ήδη δημιουργηθεί
- Η επέκταση αγορών στις σελίδες ηλεκτρονικών καταστημάτων μέσω πλήκτρων που έχουν ήδη δημιουργηθεί
- Η επέκταση της λειτουργίας των κουπονιών με εισαγωγή της λειτουργίας Share ώστε να μπορεί ο χρήστης να μοιραστεί τα κουπόνια του με συγκεκριμένους χρήστες τους οποίους θα αναζητεί μέσα από την εφαρμογή με τη χρήση κατάλληλου server
- Η δημιουργία επιπλέον Layout με διαφορετικό σχεδιασμό για χρήση της εφαρμογής και σε κινητές συσκευές πέρα από ταμπλέτες
- Η δημιουργία επιπλέον Tab η βάση των οποίων προϋπάρχει σε αρχικό στάδιο. Τέτοια Tab μπορεί να είναι: Tab για σύγκριση προϊόντων, Tab για προσωπικό καλάθι αγορών, Tab με λίστα επιθυμητών προϊόντων προς αγορά κ.α

7. Βιβλιογραφία – Ιστοσελίδες

1. Bray, Tim (2010, 28 Απρίλη). *Multitasking the Android Way*. Android Developers. http://android-developers.blogspot.gr/2010/04/multitasking-android-way.html

2. Brian X. Chen (2012, 23 Οκτωβρίου). *Apple, Facing Competition, Introduces a Smaller iPad of no significant change*. The New York Times. http://www.nytimes.com/2012/10/24/technology/apple-facing-competition-introduces-a-smaller-ipad.html?ref=technology&r=0

3. Deitel P., Deitel H., Deitel A., Morgano M., *Android For Programmers An App-Driven Approach*. 2012.

4. Duarte M., Fulcher R., Nurik R. et al, *Designing and Implementing Android UIs for Phones and Tablets*.

5. Elgin, Ben (2005, 17 Αυγούστου). *Google Buys Android for Its Mobile Arsenal*. Bloomberg Businessweek. Bloomberg. http://www.webcitation.org/5wk7sIvVb

6. Gohring, Nancy (2011, 19 Οκτωβρίου). *Samsung, Google Unveil Latest Android OS, Phone*. PCWorld. http://www.pcworld.com/article/242128/samsung_google_unveil_latest_android_os_phone.html

7. Vic Gundotra - Google+ - *Just back from a whirlwind trip to Asia visiting our….* Plus.google.com. https://plus.google.com/+VicGundotra/posts/8CVJ79nPQwN

8. Moscaritolo, Angela (2012, 18 Ιουνίου). *31 Percent of U.S. Internet Users Own Tablets* "Survey: 31 Percent of U.S. Internet Users Own Tablets". PC Magazine. http://www.pcmag.com/article2/0,2817,2405972,00.asp

9. Robbins, Renee (2009, 28 Μαΐου). *Mobile video system visually connects global plant floor engineers*. Control Engineering.

10. Lee, Wei-Meng, *Beginning Android 4 Application Development*. Wrox Press. 2012.

11. *Android Developers, The Developer's Guide*. http://developer.android.com/guide/index.html

12. *Android Supported Media Formats*. Android Developers http://developer.android.com/guide/appendix/media-formats.html

13. *Android 4.1 (Jelly Bean) Voice Actions explained*. Geek.com. http://www.geek.com/mobile/googles-new-jelly-bean-voice-actions-1499437/

14. *Feature Phone*. Phone Scoop. http://www.phonescoop.com/glossary/term.php?gid=310

15. (2011, 31 Ιανουαρίου) *Google's Android becomes the world's leading smart phone platform*. Canalys. http://www.canalys.com/newsroom/google%E2%80%99s-android-becomes-world%E2%80%99s-leading-smart-phone-platform

16. *Google Play hits 25 billion downloads*.Official Android Blog http://officialandroid.blogspot.ca/2012/09/google-play-hits-25-billion-downloads.html

17. *Google Play Matches Apple's iOS With 700,000 Apps* http://www.tomsguide.com/us/Google-Play-Android-Apple-iOS,news-16235.html

18. *Smartphone*. Phone Scoop. http://www.phonescoop.com/glossary/term.php?gid=131

19. (2013, 29 Απριλίου) *Smartphones now outsell 'dumb' phones*.3 News NZ. http://www.3news.co.nz/Smartphones-now-outsell-dumb-phones/tabid/412/articleID/295878/Default.aspx#.UoPJNfk712I

20. *T-Mobile G1 Spec*.Infosite and comparisons. GSM Arena. http://www.gsmarena.com/t_mobile_g1-2533.php

21. *Voice Actions for Android*. google.com. http://www.google.com/mobile/search/