ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΛΟΠΟΝΝΗΣΟΥ
ΣΧΟΛΗ ΟΙΚΟΝΟΜΙΑΣ, ΔΙΟΙΚΗΣΗΣ ΚΑΙ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

Εργαστήριο Γνώσης και Αβεβαιότητας

Μεταπτυχιακή Εργασία

MARKETPAL - Πλατφόρμα Διαχείρισης Προϊόντων Σούπερ Μάρκετ με Σκοπό την Εύρεση Οικονομικότερης Αγοράς

Χρήστος Παναγιώτης Χαμπηλωμάτης
2022202302019

Επιβλέπων:

Εμμανουήλ Γουάλλες
Αναπληρωτής Καθηγητής

Τρίπολη, Νοέμβριος, 2024

Εγκρίθηκε από την εξεταστική επιτροπή την.

Εμμανουήλ Γουάλλες	Κώστας Πέππας	Βασίλης Πουλόπουλος
Αναπληρωτής Καθηγητής	Αναπληρωτής Καθηγητής	Αναπληρωτής Καθηγητής

.....

Χρήστος Παναγιώτης Χαμπηλωμάτης
Τμήμα Πληροφορικής και Τηλεπικοινωνιών
Πανεπιστήμιο Πελοποννήσου

Η παρούσα εργασία εκπονήθηκε σε συνεργασία με το Εργαστήριο Γνώσης και Αβεβαιότητας (ΓΑΒ LAB).

To my family.

Περίληψη

Η εφαρμογή Marketpal αποτελεί μια εξελιγμένη λύση για τους καταναλωτές που αναζητούν τις καλύτερες προσφορές σε προϊόντα από διάφορες αλυσίδες σούπερ μάρκετ στην Ελλάδα. Η Android εφαρμογή προσφέρει πολλαπλές διεπαφές που διευκολύνουν τους καταναλωτές. Οι βασικές λειτουργίες σύγκρισης τιμών στις διάφορες αυτές αλυσίδες είναι προσβάσιμες μέσω της διεπαφής της εφαρμογής. Στον πυρήνα της εφαρμογής βρίσκονται πέντε διαφορετικές διεπαφές και χωρίζονται ως εξής. Α) Αρχική Διεπαφή: Προσφέρει μια επισκόπηση των διαθέσιμων κατηγοριών προϊόντων. Β) Αναζήτηση: Επιτρέπει στους χρήστες να εντοπίζουν συγκεκριμένα προϊόντα. Γ) Προσφορές: Δίνει τη δυνατότητα στους χρήστες να βρίσκουν τρέχουσες προσφορές και εκπτώσεις. Δ) Καλάθι: Επιτρέπει τη δημιουργία, διαχείριση, και επισκόπηση του καλαθιού αγορών, συμπεριλαμβανομένων των βέλτιστων επιλογών. Ε) Προφίλ: Επιτρέπει τη διαχείριση του προσωπικού προφίλ, συμπεριλαμβανομένης της προσθήκης και αφαίρεσης αγαπημένων προϊόντων. Η εφαρμογή Marketpal προσφέρει μια πρακτική και ευέλικτη λύση που απλοποιεί τη διαδικασία σύγκρισης τιμών και αγορών για τους καταναλωτές. Κατασκευάστηκε για να καλύψει τις ανάγκες των χρηστών και να τους βοηθήσει να εντοπίζουν τις καλύτερες δυνατές προσφορές από τα σούπερ μάρκετ στην περιοχή τους.

Λέξεις-κλειδιά: Κινητές Συσκευές, Εφαρμογή Android, Java, Διεπαφές, Σύγκριση Τιμών, Αλυσίδες Σούπερ Μάρκετ

Abstract

The Marketpal application is a sophisticated solution for consumers looking for the best deals on products from various supermarket chains in Greece. The Android application offers multiple interfaces that facilitate the consumers. The basic price comparison functions in these various chains are accessible through the application interface. At the core of the application are five different interfaces and they are divided as follows. A) Home Interface: Offers an overview of available product categories. B) Search: Allows users to locate specific products. C) Offers: Enables users to find current offers and discounts. D) Cart: Allows the creation, management, and overview of the shopping cart, including the best options. E) Profile: Allows the management of the personal profile, including adding and removing favorite products. The Marketpal app offers a practical and flexible solution that simplifies the price comparison and shopping process for consumers. It was built to meet the needs of users and help them find the best possible offers from supermarkets in their area.

Keywords: Mobile Devices, Android Application, Java, Interfaces, price comparison, supermarket

Πίνακας περιεχομένων

Πίνακας σχημάτων

Εισαγωγή

Η παρούσα εργασία αντιπροσωπεύει τη συλλογική προσπάθεια προσδιορισμού και υλοποίησης της εφαρμογής Marketpal, μιας εξελιγμένης Android εφαρμογής που ανταποκρίνεται στις ανάγκες των σύγχρονων καταναλωτών στην Ελλάδα. Μέσα από πέντε διαφορετικές διεπαφές, η Marketpal προσφέρει στους χρήστες τη δυνατότητα σύγκρισης τιμών, ανεύρεσης προσφορών, δημιουργίας καλαθιών αγορών, και διαχείρισης προφίλ. Αυτή η εφαρμογή αναδεικνύει μια πρακτική και ευέλικτη λύση που διευκολύνει την καθημερινή ζωή των καταναλωτών, επιτρέποντάς τους να εντοπίζουν τις καλύτερες δυνατές προσφορές από διάφορα σούπερ μάρκετ στην περιοχή τους. Στο μέλλον, η εφαρμογή αυτή έχει το δυναμικό να επεκταθεί και σε άλλες πλατφόρμες, όπως το web, γεγονός που θα δώσει τη δυνατότητα σε ακόμα περισσότερους χρήστες να επωφεληθούν από τις υπηρεσίες της. Η δομή της εργασίας είναι η εξής: το επόμενο κεφάλαιο προσφέρει μια σύντομη ανασκόπηση σε αντίστοιχα συστήματα όπως το e-Καταναλωτής, το Pockee και το Trolley.co.uk. Αυτά τα συστήματα παρέχουν σχεδόν ίδιες λειτουργίες με την Marketpal. Στην συνέχεια εξετάζονται οι τεχνολογίες που χρησιμοποιήθηκαν για την δημιουργία της εφαρμογής, όπου σημαντικό ρόλο έπαιξε η Java. Στην συνέχεια γίνεται ανάλυση του λογισμικού που αναπτύχθηκε, η ανάλυση λογισμικού είναι η διαδικασία της κατανόησης και καταγραφής των απαιτήσεων που πρέπει να πληροί το λογισμικό που θα αναπτυχθεί. Συνεχίζοντας στην Αναλυτική Περιγραφή Λογισμικού, όπου παρέχονται εικόνες από όλες τις διαθέσιμες διεπαφές της εφαρμογής καθώς και ο τρόπος χρήσης της. Ακολουθεί η σύνοψη στην οποία διευκρινίζονται οι δυσκολίες που αντιμετωπίστηκαν στην υλοποίηση της εφαρμογής καθώς και το τι είδαμε στην συγκεκριμένη διπλωματική εργασία. Τέλος, παρουσιάζονται κάποια παραρτήματα κώδικα. Ένα από αυτά είναι το πως έγινε η εγκατάσταση διαφημίσεων μέσω της Google AdMob στην Marketpal.

Κεφάλαιο 1

Αντίστοιχα Συστήματα

Αντίστοιχα συστήματα παρόμοια με την εφαρμογή Marketpal είναι το e-Καταναλωτής (https://e-katanalotis.gov.gr/) ο οπόιος αντιπροσωπεύει μια πρωτοποριακή πρωτοβουλία του υπουργείου, παρέχοντας μια εξαιρετικά χρήσιμη υπηρεσία στους πολίτες. Αυτή η διαδικτυακή πλατφόρμα διευκολύνει τους καταναλωτές στη σύγκριση των τιμών σε ελληνικά σούπερ μάρκετ, προωθώντας έτσι τη διαφάνεια και την ενεργό συμμετοχή στην αγορά. Αντίστοιχα και το Pockee (https://pockee.com/) παρέχει χρήσιμες πληροφορίες για τα προϊόντα στα ελληνικά σούπερ μάρκετς. Αξίζει να αναφερθεί επίσης και ένα σύστημα εκτός Ελλάδας το Trolley.co.uk που απευθύνεται στο Ηνωμένο Βασίλειο και παρέχει εξίσου λειτουργίες σύγκρισης στα σούπερ μάρκετ. Παρακάτω γίνεται μία ανάλυση αυτών των συστημάτων.

1.1 e-Καταναλωτής

Στην πλατφόρμα e-Καταναλωτής [eka] μπορείτε να ενημερωθείτε για τιμές προϊόντων που διατίθενται από υπερκαταστήματα τροφίμων (super markets), για τις τιμές καυσίμων πρατηρίων και για τις τιμές της ενέργειας. Για κάθε προϊόν μπορείτε να δείτε την τιμή του σε κάθε διαφορετικό κατάστημα καθώς και την εξέλιξη της τιμής του από την προηγούμενη εβδομάδα (ελάχιστη, μέγιστη και μέση τιμή). Επιπλέον μπορείτε να τοποθετήσετε στο «καλάθι» της εφαρμογής προϊόντα που σας ενδιαφέρουν για να υπολογιστεί αυτόματα το συνολικό τους κόστος για κάθε διαφορετικό super market. Η πλατφόρμα σας δίνει τη δυνατότητα να αναζητήσετε σε διαδραστικό χάρτη τα καταστήματα και να ενημερωθείτε για τα δικαιώματα και την προστασία σας ως καταναλωτής. Στην πλατφόρμα e-Καταναλωτής μπορείτε επίσης να ενημερωθείτε για την πρωτοβουλία «Το καλάθι του νοικοκυριού», το οποίο περιέχει τις εβδομαδιαίες τιμές 51 βασικών προϊόντων, όπως διαμορφώνονται σε κάθε super market. Για τις κατηγορίες καυσίμων μπορείτε να δείτε τη μέση τιμή πώλησης τους για κάθε διαφορετικό πρατήριο καυσίμων. Επίσης, η πλατφόρμα σας δίνει τη δυνατότητα να αναζητήσετε σε διαδραστικό χάρτη τα πρατήρια καυσίμων. Για κάθε πάροχο ενέργειας μπορείτε να δείτε τη μέση μηνιαία τιμή για το οικιακό τιμολόγιο ρεύματος καθώς και την εξέλιξή της ανά μήνα.

Εικόνα 1.1. Ιστοσελίδα e-Καταναλωτής

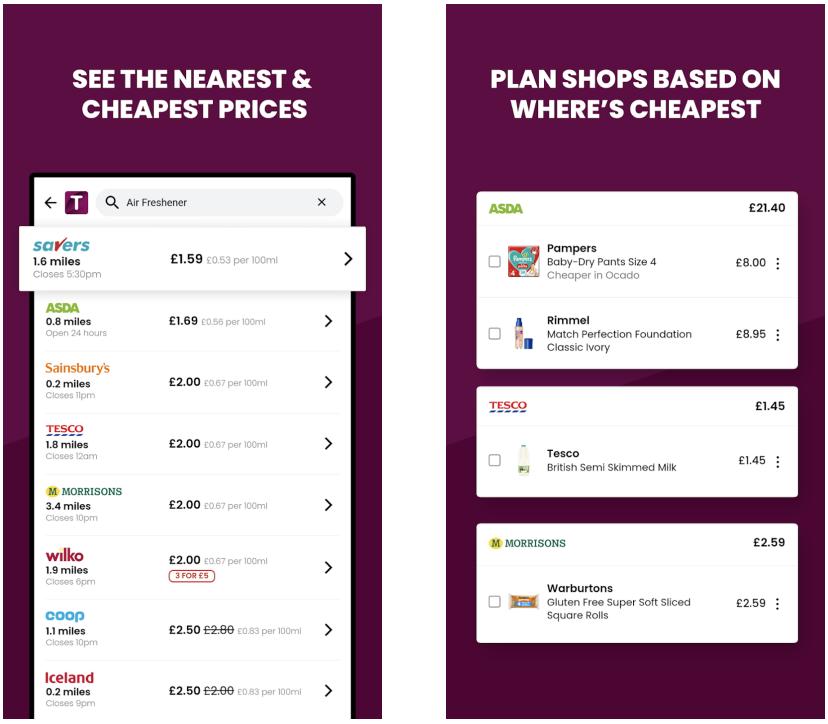
1.2 Pockee

Το Pockee είναι μια δωρεάν mobile εφαρμογή διαθέσιμη για Android και iOS που σας βοηθάει να κάνετε έξυπνες αγορές στο σούπερ μάρκετ. Μέσω αυτής της εφαρμογής μπορείτε να βρίσκετε τις πιο συμφέρουσες προσφορές σούπερ μάρκετ, να ανακαλύπτετε προϊόντα με αποκλειστικές επιστροφές Pockee και να δημιουργείτε τη λίστα αγορών σας, ενώ ενημερώνεστε απευθείας για το εκτιμώμενο κέρδος των αγορών σας. Ακόμα, αφού ολοκληρώσετε τις αγορές σας, μπορείτε να καταχωρείτε την απόδειξη σας στο Pockee. Τα χρήματα των επιστροφών Pockee τοποθετούνται στο Pockee πορτοφόλι σας, δίνοντάς σας τη δυνατότητα να επιλέξετε πώς θα τα λάβετε. Παρέχει πολλαπλές διεπαφές που διευκολύνουν τους καταναλωτές στην προσπέλαση των πληροφοριών σχετικά με τα προϊόντα και τις τιμές τους.

Εικόνα 1.2. https://pockee.com/

1.3 Trolley.co.uk

Το Trolley.co.uk βοηθά τους καταναλωτές να συγκρίνουν τιμές σε δημοφιλή σούπερ μάρκετ όπως τα Asda, Tesco, Aldi και πολλά άλλα. Συστήνεται από ειδικούς στις δαπάνες, όπως είναι η The Sun και η Mirror. Με αυτήν την υπηρεσία, μπορείτε να συγκρίνετε τις τιμές για πάνω από 130.000 προϊόντα από πάνω από 7.000 διαφορετικές εμπορικές επωνυμίες. Οι καταναλωτές έχουνε την δυνατότα να εξοικονομήσουνε έως και 30% στις εβδομαδιαίες τους αγορές και επωφεληθούν από κάθε λίρα και δεκάρα που κερδίζουνε. Παρέχονται πάνω από δεκατέσσερα καταστήματα του Ηνωμένου Βασιλείου. Κάποια από αυτά είναι τα: Asda, Tesco, Aldi, Ocado, Home Bargains, Morrisons, Iceland, B&M, Waitrose Co-op Sainsbury's, Superdrug Boots και πολλά άλλα! Το Trolley.co.uk είναι διαθέσιμο στο διαδίκτυο, σε συσκευές android και iOs καθιστώντας το προσβάσιμο για όλους τους χρήστες.



Εικόνα 1.3. Η εφαρμογή Trolley.co.uk

Κεφάλαιο 2

Τεχνολογίες που Χρησιμοποιήθηκαν

Οι τεχνολογίες που χρησιμοποιήθηκαν για την υλοποίηση της εφαρμογής περιλαμβάνουν τη γλώσσα προγραμματισμού Java, η οποία χρησιμοποιήθηκε για τη δημιουργία των διεπαφών και όλων των βασικών λειτουργιών της εφαρμογής. Επιπλέον, η βιβλιοθήκη JUnit χρησιμοποιείται για την εκτέλεση μοναδικών δοκιμών στην Android εφαρμογή. Η Espresso, από την άλλη, αποτελεί μια βιβλιοθήκη που βοηθά στην αυτοματοποίηση δοκιμών διεπαφής χρήστη για την Android. Προκειμένου να διασφαλιστεί η συμβατότητα της εφαρμογής με διάφορες εκδόσεις του Android, χρησιμοποιήθηκε η AndroidX AppCompat. Αξιοποιώντας τη βιβλιοθήκη Material Design, δημιουργήθηκε μια καλαίσθητη διεπαφή χρήστη. Τέλος, οι βιβλιοθήκες Glide και Picasso χρησιμοποιήθηκαν για τη φόρτωση και την εμφάνιση εικόνων, ενώ η βιβλιοθήκη Shimmer προσέδωσε εντυπωσιακά εφέ στις διαδικασίες φόρτωσης. Για την ενσωμάτωση διαφημίσεων στην εφαρμογή, χρησιμοποιήθηκε η βιβλιοθήκη Google Play Services Ads. Επιπλέον, η AndroidX Concurrent Futures παρέχει μηχανισμούς για τη διαχείριση συντονισμένων εργασιών, ενώ η OkHttp είναι υπεύθυνη για την αλληλεπίδραση με διάφορες υπηρεσίες στο διαδίκτυο και το μορφότυπο ανταλλαγής δεδομένων JSON. Όλες αυτές οι τεχνολογίες συνεργάστηκαν για τη δημιουργία μιας προηγμένης εφαρμογής που προσφέρει ευελιξία, εξυπηρέτηση των αναγκών των χρηστών και ενισχύει την καθημερινή ζωή των καταναλωτών. Παρακάτω γίνεται μια σύντομη περιγραφή αυτών των τεχνολογιών.

2.1 Αντικειμενοστραφής Προγραμματισμός (Object-Oriented Programming)

Ο αντικειμενοστραφής προγραμματισμός (Object-Oriented Programming - OOP) είναι μια μεθοδολογία προγραμματισμού που βασίζεται στην έννοια των αντικειμένων. Στον αντικειμενοστραφή προγραμματισμό ένα πρόγραμμα αναπαριστά τον κόσμο ως συλλογή από αντικείμενα που αλληλεπιδρούν μεταξύ τους. Κάθε αντικείμενο έχει ιδιότητες (attributes) που περιγράφουν την κατάστασή του και μεθόδους (methods) που περιγράφουν τις ενέργειες που μπορεί να πραγματοποιήσει. Τα αντικείμενα μπορούν να κληρονομούνται, δηλαδή να παίρνουν ιδιότητες και μεθόδους από άλλα αντικείμενα, και μπορούν να αλληλεπιδρούν μεταξύ τους μέσω μηνυμάτων. Ο αντικειμενοστραφής προγραμματισμός έχει πολλά πλεονεκτήματα, όπως η δυνατότητα επαναχρησιμοποίησης κώδικα, η αναγνώριση και αντιμετώπιση σφαλμάτων κατά τη διάρκεια της ανάπτυξης, και η δυνατότητα ανάπτυξης πιο περίπλοκων και δομημένων εφαρμογών, ο κώδικας οργανώνεται σε κλάσεις, οι οποίες είναι πρότυπα για τη δημιουργία αντικειμένων. Κάθε κλάση περιγράφει τα χαρακτηριστικά και τις λειτουργίες που μπορούν να εκτελεστούν από τα αντικείμενα που δημιουργούνται από αυτήν. Οι περισσότερες σύγχρονες γλώσσες προγραμματισμού υποστηρίζουν τον αντικειμενοστραφή προγραμματισμό [Ren82].

2.1.1 Γλώσσα Προγραμματισμού Java

Η Java [AGH05] είναι μια γλώσσα προγραμματισμού υψηλού επιπέδου η οποία έχει προέλθει από την εταιρεία Sun Microsystems. Χρησιμοποιείται τόσο σε εφαρμογές διαδικτύου όσο και σε αυτόνομες εφαρμογές. Μερικά από τα βασικά της χαρακτηριστικά είναι η αντικειμενοστρέφεια, η επαναχρησιμοποίηση κώδικα και η φορητότητα. Αντικειμενοστρέφεια είναι μία προσέγγιση στην ανάπτυξη λογισμικού που οργανώνει τόσο το πρόβλημα όσο και τη λύση του ως μία συλλογή από διακριτά αντικείμενα. Η επαναχρησιμοποίηση κώδικα επιτυγχάνεται με τη χρησιμοποίηση κ' εισαγωγή ήδη υπαρχόντων προγραμμάτων των λεγόμενων κλάσεων οι οποίες επιτρέπουν στον προγραμματιστή να φτάσει γρήγορα στο επιθυμητό τελικό αποτέλεσμα. Η φορητότητα της Java έγκειται στην ανεξαρτησία λειτουργικού συστήματος και πλατφόρμας. Τα προγράμματα που είναι γραμμένα σε Java τρέχουνε ακριβώς το ίδιο σε Windows και Linux, χωρίς να χρειαστεί να ξαναγίνει μεταγλώττιση (compiling) ή να αλλάξει ο πηγαίος κώδικας για κάθε διαφορετικό λειτουργικό σύστημα. Άλλα ιδιαίτερα χαρακτηριστικά της Java είναι ο πολυνηματισμός, η ιδιαίτερη τεχνολογία δικτύου καθώς και το χαρακτηριστικό της ότι μπορεί να χρησιμοποιηθεί σε οποιουδήποτε μεγέθους εφαρμογή. Το Android Studio σε συνδυασμό με τη Java αποτελούν μια ισχυρή πλατφόρμα για τη δημιουργία εντυπωσιακών Android εφαρμογών. Η Java παρέχει ένα ισχυρό σύστημα αντικειμενοστραφούς προγραμματισμού, ενώ το Android Studio προσφέρει εργαλεία και πόρους που επιτρέπουν στους προγραμματιστές να δημιουργήσουν λειτουργικές και ευέλικτες Android εφαρμογές με εξαίσιες γραφικές διεπαφές χρήστη.

2.2 JUnit

Το JUnit [MH03] είναι ένα δημοφιλές πλαίσιο δοκιμών για τη γλώσσα προγραμματισμού Java. Χρησιμοποιείται ευρέως από προγραμματιστές για τη δημιουργία και εκτέλεση δοκιμών μονάδων σε εφαρμογές Java. Οι δοκιμές JUnit είναι αυτόματες, ανεξάρτητες και επαναλαμβανόμενες, βοηθώντας τους προγραμματιστές να εντοπίζουν σφάλματα και να διασφαλίσουν τη σωστή λειτουργία του κώδικά τους. Το JUnit παρέχει αναλυτικές αναφορές σχετικά με τα αποτελέσματα των δοκιμών και τη δυνατότητα επαναλαμβανόμενων ελέγχων στην ανάπτυξη. Είναι ένα ισχυρό εργαλείο για τη βελτιστοποίηση της ποιότητας του λογισμικού και την επιτάχυνση της διαδικασίας ανάπτυξης.

2.3 Espresso Testing Framework

Η Espresso [andc] είναι μια προηγμένη τεχνολογία για τον έλεγχο και τη δοκιμή των διεπαφών χρήστη σε εφαρμογές Android. Επιτρέπει στους προγραμματιστές να δημιουργούν αυτοματοποιημένες δοκιμές για την αλληλεπίδραση με την εφαρμογή ακριβώς όπως θα έκανε ένας πραγματικός χρήστης. Με αυτήν την τεχνολογία, μπορείτε να ελέγχετε διάφορες δραστηριότητες, κουμπιά, πεδία εισόδου, και άλλα στοιχεία της εφαρμογής Android. Οι δοκιμές Espresso είναι γρήγορες, αξιόπιστες και προσφέρουν λεπτομερείς αναφορές σχετικά με την επίδοση και τη συμπεριφορά της εφαρμογής. Αυτή η τεχνολογία διευκολύνει τη διασφάλιση ότι η εφαρμογή Android λειτουργεί σωστά και προσφέρει μια άριστη εμπειρία χρήστη.

2.4 AndroidX AppCombat

Το AndroidX AppCompat είναι μια ευέλικτη βιβλιοθήκη για την ανάπτυξη εφαρμογών Android. Παρέχει συμβατότητα με διάφορες εκδόσεις του Android OS και ενσωματώνει τον υλικό σχεδιασμό (Material Design) για μια σύγχρονη αισθητική. Επιτρέπει την προσαρμογή των εμφανίσεων και των χρωμάτων, ενσωματώνει δράσεις εργαλείων για απλή πλοήγηση, υποστηρίζει διεθνοποίηση και σκοτεινό θέμα. Είναι συμβατό με το AndroidX και διαχειρίζεται διάφορες εκδόσεις Android. Αποτελεί χρήσιμο εργαλείο για τη δημιουργία εφαρμογών Android με εντυπωσιακή εμφάνιση και λειτουργικότητα [andb].

2.5 Glide

Το Glide [gli] είναι μια ισχυρή βιβλιοθήκη διαχείρισης φορτωμένων εικόνων σχεδιασμένη ειδικά για την πλατφόρμα Android. Αποτελώντας μια εύχρηστη και αποδοτική λύση, το Glide προσφέρει απλότητα στη χρήση και εκτεταμένη λειτουργικότητα για τη φόρτωση, την εμφάνιση και τη διαχείριση εικόνων. Με δυνατότητες όπως caching, scaling, animation και πολλά άλλα, το Glide καλύπτει εκτενώς τις ανάγκες των εφαρμογών Android που απαιτούν επίδοση στην εμφάνιση εικόνων. Με τη συνεχή υποστήριξη και τη συνεχή εξέλιξη, το Android Glide αποτελεί μια αξιόπιστη επιλογή για τους προγραμματιστές Android που επιδιώκουν την αποτελεσματική διαχείριση εικόνων στις εφαρμογές τους.

2.6 Picasso

Το Picasso [pic] αποτελεί μια αξιοσημείωτη βιβλιοθήκη διαχείρισης εικόνων στο πεδίο της ανάπτυξης Android εφαρμογών. Σχεδιασμένο με έμφαση στην ευκολία χρήσης και την αποδοτικότητα, το Picasso προσφέρει μια απλή και κατανοητή διεπαφή για το φορτώσιμο και την προβολή εικόνων στις εφαρμογές Android. Με δυνατότητες όπως caching, resizing, και την αυτόματη διαχείριση μνήμης, το Picasso βοηθά τους προγραμματιστές να δημιουργήσουν εφαρμογές με γρήγορη απόκριση και ομαλή εμπειρία χρήστη. Με τη στήριξη της κοινότητας ανοικτού κώδικα, το Picasso συνεχίζει να εξελίσσεται, παρέχοντας ένα αξιόπιστο εργαλείο για τη διαχείριση εικόνων σε εφαρμογές Android.

2.7 Facebook Shimmer

Το Shimmer [fbs] αναπτύχθηκε με σκοπό να προσφέρει εντυπωσιακά εφέ φωτεινότητας σε στοιχεία UI στον κόσμο της ανάπτυξης εφαρμογών. Είναι μια βιβλιοθήκη που επιτρέπει στους προγραμματιστές να προσθέτουν όμορφα και εκλεπτυσμένα εφέ shimmer, δηλαδή λάμψης, σε στοιχεία όπως τα RecyclerViews, τα TextViews και άλλα στοιχεία UI. Η εμφάνιση του Shimmer είναι εξαιρετικά εντυπωσιακή, δημιουργώντας την εντύπωση κίνησης και φωτεινότητας στα στοιχεία της διεπαφής, προσθέτοντας έναν δυναμικό χαρακτήρα στις εφαρμογές. Το Shimmer αποτελεί ένα χρήσιμο εργαλείο για τη δημιουργία εντυπωσιακών και ευχάριστων εμπειριών χρήστη σε Android εφαρμογές.

2.8 Google Play Services Ads

Οι Google Play Services Ads αποτελούν ένα σύνολο υπηρεσιών από τη Google που παρέχουν δυνατότητες διαφήμισης στις εφαρμογές Android. Η υπηρεσία αυτή περιλαμβάνει το Google Mobile Ads SDK, το οποίο επιτρέπει στους προγραμματιστές να ενσωματώσουν διάφορες μορφές διαφήμισης στις εφαρμογές τους, όπως banner ads, interstitial ads και rewarded video ads.

Μέσω των Google Play Services Ads, οι προγραμματιστές έχουν πρόσβαση σε πολυάριθμες λειτουργίες, όπως τον προσαρμοσμένο στοχευμένο έλεγχο των διαφημίσεων, τη δυνατότητα παρακολούθησης της απόδοσης των διαφημίσεων με μετρήσεις όπως οι εμφανίσεις, οι κλικ και οι εσοχές, καθώς και τη δυνατότητα ενσωμάτωσης της AdMob, μιας πλατφόρμας διαφημίσεων της Google, για ακόμη περισσότερες επιλογές διαφημιστικής προβολής.

Οι Google Play Services Ads [goo] συμβάλλουν στη δημιουργία βιώσιμων μοντέλων εσόδων για τους προγραμματιστές εφαρμογών, παρέχοντας τους πρόσβαση σε ισχυρά εργαλεία για την αποτελεσματική και κερδοφόρα διαφήμιση.

2.9 AndroidX Concurrent Futures

AndroidX Concurrent Futures αποτελεί μια σύγχρονη βιβλιοθήκη στον χώρο της ανάπτυξης Android, η οποία επιτρέπει στους προγραμματιστές να διαχειρίζονται τις ασύγχρονες εργασίες

με έναν αποτελεσματικό και καθολικό τρόπο. Χρησιμοποιώντας την βιβλιοθήκη αυτή, οι προγραμματιστές μπορούν να δημιουργήσουν και να διαχειριστούν Futures, που αντιπροσωπεύουν ασύγχρονες εργασίες, με μεγαλύτερη ευκολία.

Η AndroidX Concurrent Futures [jet] παρέχει μηχανισμούς για την παρακολούθηση, τον έλεγχο και τον συγχρονισμό των ασύγχρονων εργασιών, επιτρέποντας στους προγραμματιστές να διαχειρίζονται τις απαιτήσεις των εφαρμογών τους με αποδοτικό τρόπο. Με τη χρήση αυτής της βιβλιοθήκης, η αντιμετώπιση των ασύγχρονων διεργασιών στο Android γίνεται πιο ευέλικτη, ενθαρρύνοντας τον κώδικα που είναι ευανάγνωστος, συντηρήσιμος και αποτελεσματικός.

2.10 Βιβλιοθήκη HTTP Client OkHTTP

Η OkHttp [Squ] είναι μια προηγμένη βιβλιοθήκη HTTP client για τη γλώσσα προγραμματισμού Java, σχεδιασμένη για αποτελεσματική επικοινωνία δικτύου σε εφαρμογές Android και Java. Αναπτύσσεται από τη Square και προσφέρει μια ευέλικτη και εύκολη στη χρήση διεπαφή για τη διαχείριση HTTP/HTTPS requests και των σχετικών απαντήσεων. Ο OkHttp παρέχει χαρακτηριστικά όπως συνδυασμός συνδέσμων (connection pooling), διαχείριση cookies, διακριτικός χειρισμός σφαλμάτων και προώθηση διεργασιών για τη βέλτιστη απόδοση. Επιπλέον, υποστηρίζει την ασύγχρονη λειτουργία μέσω του πακέτου OkHttp's Call, επιτρέποντας στους προγραμματιστές να εκτελούν αιτήσεις δικτύου χωρίς να αποκλείουν το νήμα UI. Με την υψηλή του απόδοση, την εκτενή τεκμηρίωση και την ενεργή κοινότητα, το OkHttp αποτελεί μια προτιμητέα επιλογή για τους προγραμματιστές που αναζητούν αξιόπιστη λύση για τις δικτυακές ανάγκες των εφαρμογών τους..

2.11 Javascript Object Notation (JSON)

Το JSON [Cro06] , ή JavaScript Object Notation, αποτελεί ένα απλό, ελαφρύ και ευανάγνωστο φορμάτ ανταλλαγής δεδομένων, που έχει καθιερωθεί ως ένα από τα κύρια πρότυπα για τον τρόπο ανταλλαγής πληροφοριών στο web. Σχεδιασμένο αρχικά για χρήση σε συνδυασμό με τη γλώσσα προγραμματισμού JavaScript, το JSON αποτελείται από απλά ζεύγη "κλειδί-τιμή", που διευκολύνουν την αναπαράσταση δεδομένων. Η σύνταξή του είναι ευανάγνωστη από ανθρώπους και εύκολα αναλυόμενη από μηχανές, καθιστώντας το ιδανικό για τη μεταφορά δεδομένων ανάμεσα σε εφαρμογές και υπηρεσίες στο πλαίσιο της ανάπτυξης λογισμικού και του web. Επιπλέον, το JSON έχει επεκταθεί πέραν του αρχικού πλαισίου της JavaScript, καθιστώντας το πανεπιστήμιο στοιχείο στη διασύνδεση των συστημάτων και την ανταλλαγή δεδομένων σε πολυάριθμες πλατφόρμες και τεχνολογίες.

2.12 Extensible Markup Language (XML)

Η XML [xml], ή Extensible Markup Language, αποτελεί ένα πρότυπο μεταδεδομένων που χρησιμοποιείται ευρέως για τον ανταλλαγή και την αποθήκευση δεδομένων. Αναπτύχθηκε με σκοπό να παρέχει μια γενική μέθοδο για τη δημιουργία προσαρμοσμένων γλωσσών σήμανσης για διάφορες εφαρμογές. Το XML βασίζεται σε ένα σύστημα ετικετών που περιβάλλουν τα δεδομένα και καθορίζουν τη δομή τους. Η ευελιξία της XML επιτρέπει την καθορισμένη οργάνωση των πληροφοριών, καθιστώντας το κατάλληλο για χρήση σε διάφορες εφαρμογές όπως ανταλλαγή δεδομένων στον τομέα του ιστού, διαμοιρασμός δεδομένων μεταξύ συστημάτων και διατήρηση δομής σε αρχεία.

Κεφάλαιο 3

Ανάλυση Λογισμικού

Η ανάλυση απαιτήσεων λογισμικού (software requirement analysis) είναι η διαδικασία της κατανόησης και καταγραφής των απαιτήσεων που πρέπει να πληροί το λογισμικό που θα αναπτυχθεί. Οι απαιτήσεις αυτές αφορούν τις λειτουργικές και μη λειτουργικές ανάγκες των χρηστών και των συστημάτων που θα αλληλοεπιδρούν με το λογισμικό. Η ανάλυση απαιτήσεων συνήθως περιλαμβάνει τη συλλογή, την περιγραφή, την ταξινόμηση και την οργάνωση των απαιτήσεων, καθώς και τη διερεύνηση της σχέσης μεταξύ τους και με τις λειτουργικές απαιτήσεις του συστήματος στο οποίο το λογισμικό θα χρησιμοποιείται. Ο στόχος της ανάλυσης απαιτήσεων είναι η ανάπτυξη ενός κατανοητού, συνεκτικού και πλήρους συνόλου απαιτήσεων για το λογισμικό, το οποίο θα χρησιμοποιηθεί στη συνέχεια ως βάση για τον σχεδιασμό, την υλοποίηση και τον έλεγχο του λογισμικού.

3.1 Απαιτήσεις Εφαρμογής

3.1.1 Αρχική Διεπαφή

Στην αρχική διεπαφή ο χρήστης έχει την δυνατότητα επισκόπησης των διαθέσιμων προϊόντων, να πλοηγηθεί σε διάφορες κατηγορίες προϊόντων και να προβάλει τον αριθμό προϊόντων του καλαθιού του καθώς και τις ανακοινώσεις συστήματος.

3.1.2 Διεπαφή Αναζήτησης

Στην διεπαφή αναζήτησης ο χρήστης έχει την δυνατότητα να αναζητήσει κάποιο προϊόν με βάση το όνομά του.

3.1.3 Διεπαφή Προσφορών

Στην διεπαφή προσφορών ο χρήστης έχει την δυνατότητα να ερευνήσει τα προϊόντα τα οποία έχουν κάποια προσφορά, καθώς και διάφορες κατηγορίες τους.

3.1.4 Διεπαφή Καλαθιου

Στην διεπαφή καλαθιού παρέχονται στον χρήστη πολλές δυνατότες, όπως η αφαίρεση-/πρόσθεση του προϊόντος από/στο καλάθι, η ποσότητα του προϊόντος, η επισκόπηση τιμών ανά κατάστημα και τέλος η επισκόπηση των προτάσεων.

3.1.5 Διεπαφή Προϊόντος

Στην διεπαφή προϊόντος, ο χρήστης έχει την δυνατότητα επισκόπησης όλων τον διαθέσιμων τιμών ανά κατάστημα, την αφαίρεση/προσθήκη του προϊόντος από/στο καλάθι, την προσθήκη του προϊόντος στην λίστα αγαπημένων και τέλος την επισκόπηση περιγραφής του προϊόντος.

3.1.6 Διεπαφή Προφίλ

Στην διεπαφή προφίλ ο χρήστης έχει την δυνατότητα επισκόπησης των αγαπημένων προϊόντων του, τα πρόσφατα προβεβλημένα προϊόντα, τα προτεινόμενα προϊόντα της ημέρας, τα cashback προϊόντα και προϊόντα 1+1 δώρο καθώς και να ρυθμίσει ποιες αλυσίδες επιθυμεί να παρέχονται από την εφαρμογή.

3.2 Application Programming Interface (API)

Η Διεπαφή Προγραμματισμού Εφαρμογών είναι η διεπαφή των προγραμματιστικών διαδικασιών που παρέχει ένα λειτουργικό σύστημα, ή κάποιες εφαρμογές προκειμένου να γίνονται αιτήσεις από άλλα προγράμματα προς αυτά [api].

3.2.1 Web Application Programming Interface (Web API)

Το Web API (Διαδικτυακό Πρόγραμμα Εφαρμογών Προγραμματισμού) είναι ένα σύνολο κανόνων και πρωτοκόλλων που επιτρέπουν σε διάφορες εφαρμογές λογισμικού να επικοινωνούν μεταξύ τους μέσω του διαδικτύου. Αποτελεί έναν τρόπο για τις εφαρμογές να ανταλλάσσουν δεδομένα και να εκτελούν λειτουργίες μεταξύ τους, ανεξάρτητα από τον τρόπο και τη γλώσσα προγραμματισμού που χρησιμοποιούν.

Τα Web APIs επιτρέπουν την ενσωμάτωση διαφορετικών υπηρεσιών και λογισμικού μεταξύ τους, καθιστώντας δυνατή την ανάπτυξη εφαρμογών που χρησιμοποιούν λειτουργίες και δεδομένα από διάφορες πηγές. Τα Web APIs χρησιμοποιούν διάφορα πρωτόκολλα επικοινωνίας, όπως το HTTP, για τη μεταφορά δεδομένων μεταξύ του διακομιστή και του πελάτη [web].

3.2.2 API και Δεδομένα Εφαρμογής

Η Marketpal χρησιμοποιεί ένα Web API 3.2.1 για να λάβει πληροφορίες για τις κατηγορίες προϊόντων και για αυτά. Κάνει HTTP αιτήματα προς αυτό και λαμβάνει μία απάντηση HTTP σε μορφή JSON 2.11. Έχοντας τις πληροφορίες αυτές, δημιουργεί το γραφικό περιβάλλον και εμπλουτίζει τις διεπαφές της με αυτές. Το συγκεκριμένο API παρέχεται από την εφαρμογή Pockee. Ακολουθεί παράδειγμα HTTP Response.

https://v8api.pockee.com/api/v8/public/products/31041


```
{
    "data": {
        "id": 31041,
        "category_id": 367,
        "brand_id": 1348,
        "name": "Milk 170gr",
        "description": "",
        "image_versions": {},
        "is_weighted": false,
        "qty_step": 1,
        "qty_max": 6,
        "uom": "g",
        "uom_value": 170,
        "suggested_price": 0.79,
        "active": true,
        "delisted_at": null,
        "images": [],
        "brand": {},
        "category": {},
        "coupons": [],
        "assortments": []
```

Κεφάλαιο 4

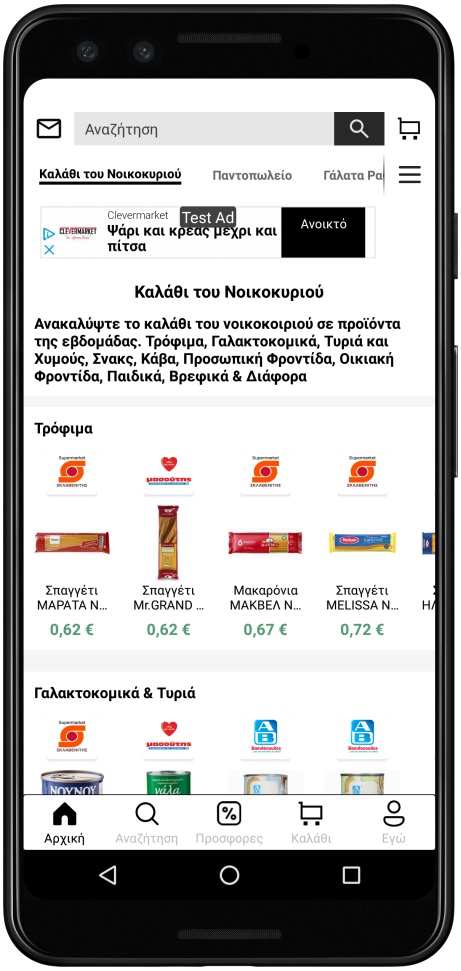
Αναλυτική Περιγραφή Λογισμικού

Παρακάτω θα επιχειρήσουμε να περιγράψουμε τις επιμέρους λειτουργίες του λογισμικού που αναπτύχθηκε.

4.1 Αρχική Διεπαφή

Όπως αναφέρθηκε στην ενότητα 3.1.1, η αρχική διεπαφή προσφέρει διάφορες λειτουργίες στον χρήστη, μία από αυτές είναι η επισκόπηση όλων των προϊόντων ανά κατηγορία και μάρκα που προσφέρονται από το API.

Στο header της διεπαφής παρέχονται τρεις λειτουργίες, η επισκόπηση ανακοινώσεων της εφαρμογής, η ανακατεύθηση στην διεπαφή αναζήτησης και του καλαθιού. Παρακάτω παρέχεται μια λίστα που δηλώνει όλες τις διαθέσιμες κατηγορίες προϊόντων όπου ο χρήστης μπορεί να επιλέξει και να κατευθυνθεί σε αυτή. Στην συνέχεια παρέχονται όλες οι κατηγορίες προϊόντων σε αύξουσα σειρά με βάση την τιμή τους. Στο footer παρέχονται όλες οι διαθέσιμες και κύριες διεπαφές της εφαρμογής Αρχική, Αναζήτηση, Προσφορές, Καλάθι και Προφίλ. Με αυτήν τη διαρρύθμιση, ο χρήστης έχει εύκολη πρόσβαση στις βασικές λειτουργίες της εφαρμογής σας, προσφέροντας έναν αποτελεσματικό τρόπο πλοήγησης και αλληλεπίδρασης.

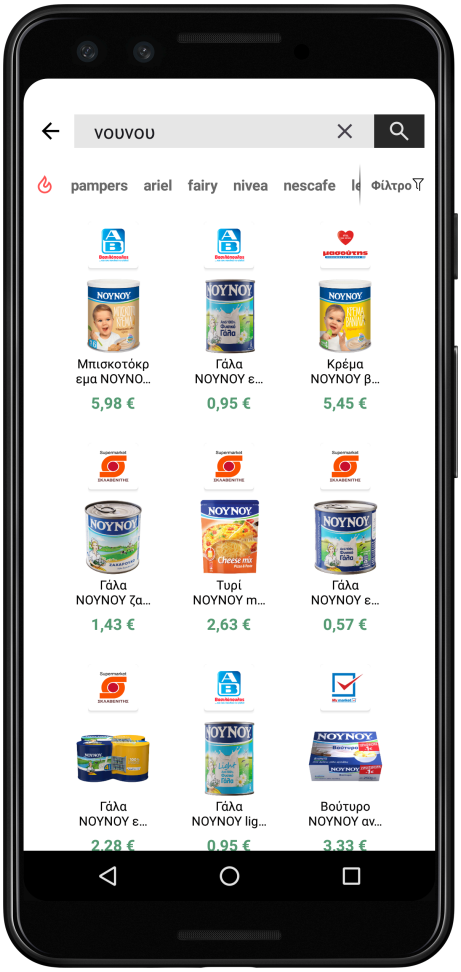


Εικόνα 4.1. Αρχική Διεπαφή

4.2 Διεπαφή Αναζήτησης

Η συγκεκριμένη διεπαφή επιτρέπει στον χρήστη να ψάξει ένα προϊόν με βάση το όνομά του. Αυτό επιτυγχάνεται μέσω της χρήσης του OkHTTP για να κάνει μια αίτηση HTTP στο Web API του Pockee, ώστε να λάβει σχετικές πληροφορίες σχετικά με το προϊόν που αναζητεί ο χρήστης.

Στο header παρέχεται η ανακατεύθυνση του χρήστη στην προηγούμενη διεπαφή που βρισκόταν καθώς και η μπάρα αναζήτησης. Παρακάτω παρέχεται μια λίστα με τις πιο δημοφιλές κατηγορίες αναζήτησης καθώς και το φίλτρο το οποίο επιτρέπει στον χρήστη να επιλέξει είτε την απλή αναζήτηση, αναζήτηση προϊόντων με προσφορά, με cashback και 1+1 δώρο.

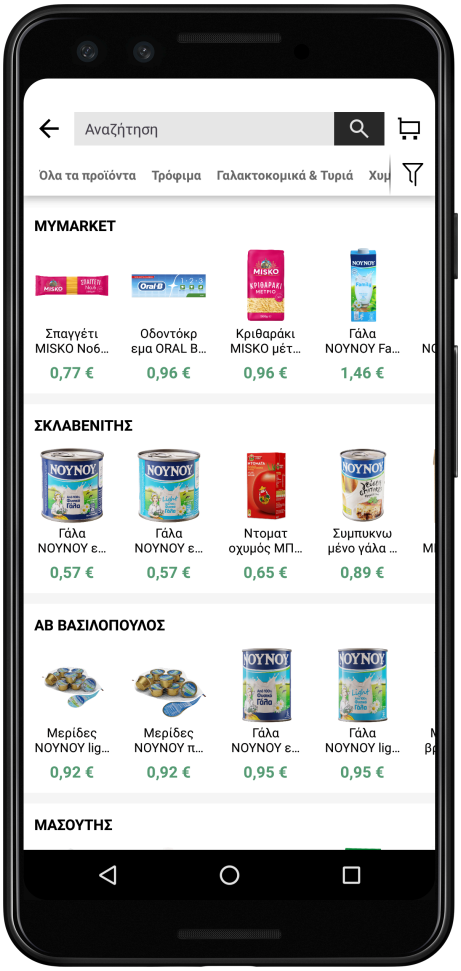


Εικόνα 4.2. Διεπαφή Αναζήτησης

4.3 Διεπαφή Προσφορών

Η διεπαφή προσφορών παρέχει λεπτομερείς πληροφορίες σχετικά με προϊόντα που ανήκουν σε διάφορες κατηγορίες και έχουν εφαρμοσμένη μια μειωμένη τιμή.

Το header παρέχει εξίσου ανακατεύθυνση του χρήστη στην προηγούμενη διεπαφή που βρισκόταν, ανακατεύθυνση στην διεπαφή αναζήτησης και του καλαθιού. Παρακάτω παρέχεται μια λίστα με τις διάφορες κατηγορίες προϊόντων όπου επιθυμεί να ερευνήσει ο χρήστης. Τα προϊόντα που προβάλλονται σε κάθε μία από τις διαθέσιμες αλυσίδες παρέχονται από αυτές με την ελάχιστη δυνατή τιμή επιτρέποντας πιο γρήγορη την αναζήτηση.



Εικόνα 4.3. Διεπαφή Προσφορών

4.4 Διεπαφή Προϊόντος

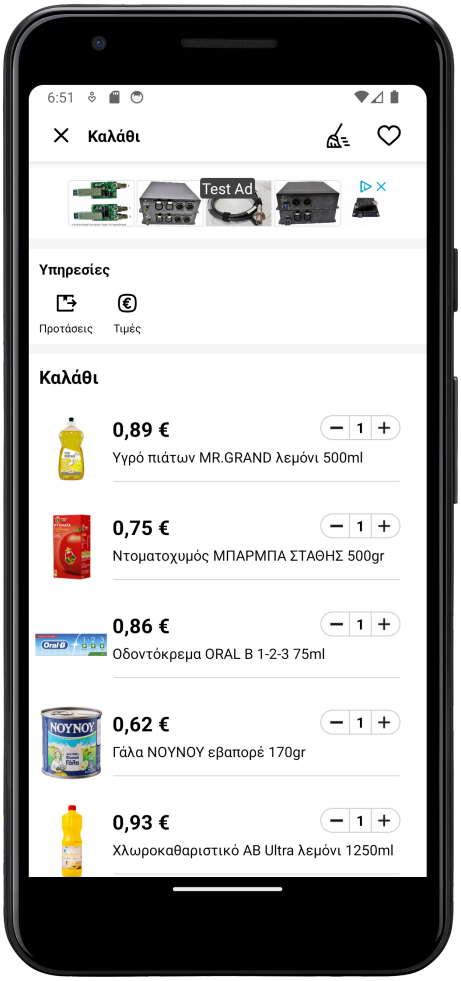
Η διεπαφή προϊόντος παρέχει σχετικές πληροφορίες για ένα συγκεκριμένο προϊόν. Η πρόσβαση στην συγκεκριμένη διεπαφή παρέχεται σχεδόν από όλες τις διαπαφές όταν ο χρήστης πατήσει πάνω στο προϊόν.

Παρέχεται η επίσκοπηση του προϊόντος καθώς και η περιγραφή του. Η λειτουργία πρόσθεσης και αφαίρεσής του στο και από το καλάθι και τέλος ο χρήστης προβάλει όλες τις τιμές που παρέχονται από τις αλυσίδες. Υπάρχει επίσης η λειτουργία πρόσθεσης στην λίστα αγαπημένων όπου συνδέεται με την διεπαφή προφίλ 4.6. Επιπλέον παρέχονται σχετικά προϊόντα.

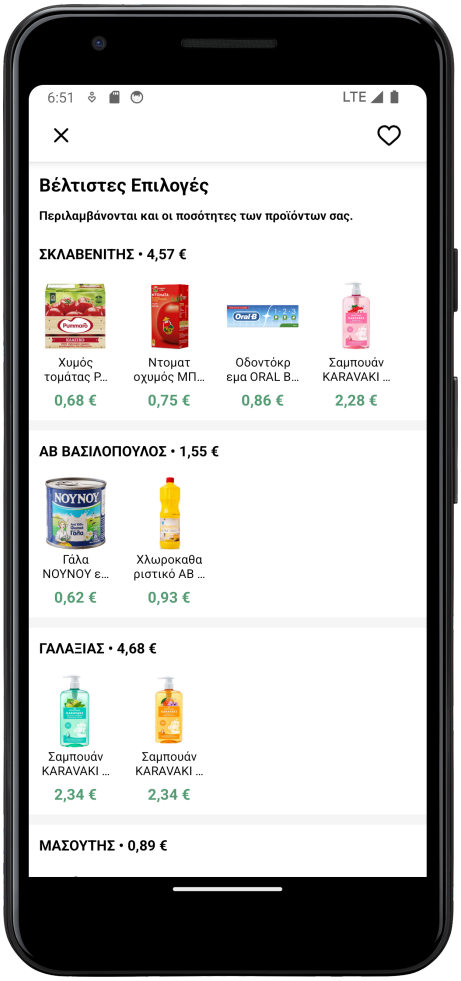
Εικόνα 4.4. Διεπαφή Προϊόντος

4.5 Διεπαφή Καλαθιού

Η διεπαφή καλαθιού είναι μια από τις σημαντικότερες διεπαφής της εφαρμογής καθώς είναι αυτή που παρέχει πληροφορίες στον χρήστη και το συμβουλεύει για την αγορά των προϊόντων του. Παρέχονται λειτουργίες όπως η αύξηση ποσότητας του προϊόντος, αφαίρεσης, εκκάθαρισης ολόκληρου το καλαθιού καθώς και λειτουργίες επισκόπησης για τη συνολική τιμή με βάση τις ποσότητες ανά αλυσίδα. Επιπλέον παρέχεται η επισκόπηση βέλτιστων επιλογών όπου παρέχεται η πληροφόρια στο χρήστη για το ποια θα είναι η βέλτιστη αγορά των συγκεκριμένων προϊόντων ώστε να εξοικονομήσει χρήματα.



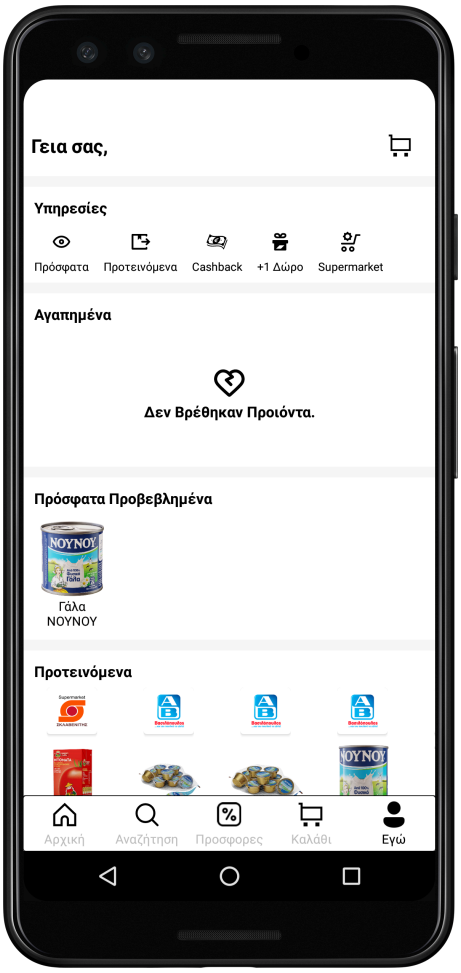
Εικόνα 4.5. Διεπαφή Καλαθιού



Εικόνα 4.6. Βέλτιστες Επιλογές Καλαθιού

4.6 Διεπαφή Προφίλ

Η διεπαφή προφίλ παρέχει στον χρήστη διάφορες υπηρεσίες, μία από αυτές είναι η επισκόπηση των αγαπημένων προϊόντων όπου ο χρήστης μπορεί να ελέγχει καθημερινά τις τιμές τους, τα προσφάτα προβεβλημένα, τα προτεινόμενα προϊόντα της ημέρας, τα προϊόντα με κάποιο cashback, τα προϊόντα με 1+1 δώρο. Επίσης μία κύρια υπηρεσία είναι η διαχείριση των επιθυμητών καταστημάτων όπου επιτρέπει στον χρήστη να επιλέξει για το ποιες αλυσίδες θα είναι διαθέσιμες στην εφαρμογή.

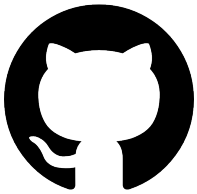


Εικόνα 4.7. Διεπαφή Προφίλ

Κεφάλαιο 5

Online Παρουσία

5.1 Github και Google Play



https://github.com/Xristosxmp/marketpal
https://play.google.com/store/apps/details?id=com.app.marketpal

Κεφάλαιο 6

Σύνοψη

Στην συγκεκριμένη εργασία παρουσιάστηκε η διαδικασία σχεδιασμού και υλοποίησης της εφαρμογής Marketpal. Συγκεκριμένα παρουσιάστηκαν οι τεχνολογίες που χρησιμοποιήθηκαν για την δημιουργια της. Όλες οι διεπαφές της, καθώς και το τι προσφέρει η κάθε μία από αυτές. Για την δημιουργία αυτών των διεπαφών χρειάστηκε να συνδιαστούν πολλές τεχνολογίες και υπήρξαν αρκετές τεχνολογικές προκλήσεις. Μία από αυτές ήταν η χρήση του ενσωματωμένου Recycler Viewer στην αρχική διεπαφή, το οποίο προσφέρει καλύτερη απόδοση κατά την εκτέλεση της εφαρμογής, καθώς ανακυκλώνει τα αντικείμενα εκτός οθόνης. Άλλη μία πρόκληση, ήταν η κατανόηση του Pockee API. Το σύστημα έχει υλοποιηθέι σε επίπεδο που μπορεί να είναι σε πλήρη χρήση, ωστόσο υπάρχουν αρκετά στοιχεία τα οποία θα μπορούσαν να βελτιωθούν. Κλείνοντας, η εφαρμογή μπορεί να αναπτυχτεί και σε άλλες πλατφόρες όπως το web και iOs καθιστώντας την διαθέσιμη σε περισσότερους χρήστες.

Βιβλιογραφία

[adm] admob. *Google AdMob - Earn More With Mobile App Monetization — admob.google.com*. https://admob.google.com/home/. [Accessed 29-12-2023].

[AGH05] Ken Arnold, James Gosling και David Holmes. *The Java programming language*. Addison Wesley Professional, 2005.

[anda] google android. *Get Started | Android | Google for Developers — developers.google.com*. https://developers.google.com/admob/android/quick-start. [Accessed 29-12-2023].

[andb] androidAppcompatJetpack. *Appcompat | Jetpack | Android Developers — developer.android.com*. https://developer.android.com/jetpack/androidx/releases/appcompat. [Accessed 19-12-2023].

[andc] androidEspressoAndroid. *Espresso | Android Developers — developer.android.com*. https://developer.android.com/training/testing/espresso. [Accessed 19-12-2023].

[api] api. *Διεπαφή προγραμματισμού εφαρμογών - Βικιπαίδεια — el.wikipedia.org*. https://en.wikipedia.org/wiki/API. [Accessed 20-12-2023].

[Cro06] Douglas Crockford. *The application/json media type for javascript object notation (json)*. Αδημοσίευτη ερευνητική εργασία. 2006.

[Dev23a] Android Developers. *AsyncTask Class Reference*. Accessed: October 26, 2023. 2023.

[Dev23b] Android Developers. *RecyclerView Class Reference*. Accessed: October 26, 2023. 2023.

[eka] ekat. *e-Καταναλωτής | Παρατηρητήριο Αγοράς — e-katanalotis.gov.gr*. https://www.gov.gr/ipiresies/polites-kai-kathemerinoteta/enemerose-katanaloton/e-katanalotes. [Accessed 4-7-2024].

[fbs] fbshimmer. *GitHub - facebookarchive/shimmer-android: An easy, flexible way to add a shimmering effect to any view in an Android app. — github.com*. https://github.com/facebookarchive/shimmer-android. [Accessed 19-12-2023].

[gli] glide. *GitHub - bumptech/glide: An image loading and caching library for Android focused on smooth scrolling — github.com*. https://github.com/bumptech/glide. [Accessed 19-12-2023].

[goo] googleads. *Overview of Google Play services | Google for Developers — developers.google.com*. https://developers.google.com/android/guides/overview. [Accessed 19-12-2023].

[jet] jetpack. *Concurrent | Jetpack | Android Developers — developer.android.com*. https://developer.android.com/jetpack/androidx/releases/concurrent. [Accessed 19-12-2023].

[MH03] Vincent Massol και Ted Husted. *JUnit in Action*. USA: Manning Publications Co., 2003. ISBN: 1930110995.

[pic] picasso. *GitHub - square/picasso: A powerful image downloading and caching library for Android — github.com*. https://github.com/square/picasso. [Accessed 19-12-2023].

[poc] pockeecashback. *Κουπόνια cashback - pockee — pockee.com*. https://pockee.com/. [Accessed 17-12-2023].

[Ren82] Tim Rentsch. «Object oriented programming». Στο: *ACM Sigplan Notices* 17.9 (1982), σσ. 51–57.

[Squ] Inc. Square. *Overview - OkHttp — square.github.io*. https://square.github.io/okhttp/. [Accessed 16-03-2024].

[web] webapi. *Web API - Wikipedia — en.wikipedia.org*. https://en.wikipedia.org/wiki/Web_API. [Accessed 20-12-2023].

[xml] xml. *XML - Wikipedia — en.wikipedia.org*. https://en.wikipedia.org/wiki/XML. [Accessed 29-12-2023].

Παραρτήματα

Αναλυτική Εξέταση Τεχνολογιών

Ο κύριος στόχος αυτού του παραρτήματος είναι η λεπτομερής ανάλυση ενός συγκεκριμένου κομματιού του πηγαίου κώδικα της εφαρμογής Marketpal που υλοποιείται χρησιμοποιώντας τη βιβλιοθήκη OkHTTP, τις AsyncTasks, και άλλες σύγχρονες τεχνολογίες ανάπτυξης εφαρμογών Android. Επικεντρώνεται στην εξήγηση της λειτουργίας και του ρόλου αυτού του κομματιού του κώδικα, καθώς και στην ανάδειξη των πλεονεκτημάτων που προσφέρει στην εφαρμογή. Επιπλέον, προσπαθεί να διαφωτίσει την εφαρμογή της θεωρίας μέσα από την πρακτική εφαρμογή των νέων τεχνολογιών στον κώδικα. Μέσω αυτής της ανάλυσης, αναδεικνύονται οι τρόποι με τους οποίους η εφαρμογή Marketpal εκμεταλλεύεται τη συνδυαστική χρήση των παραπάνω εργαλείων και τεχνολογιών για τη βελτίωση της εμπειρίας των χρηστών και την αποτελεσματική υλοποίηση των λειτουργιών της εφαρμογής.

Εγκατάσταση και Χρήση της βιβλιοθήκης OkHTTP στο Android Studio

Όπως αναφέρθηκε στην ενότητα 2.10. Το OkHttp είναι μια προηγμένη βιβλιοθήκη HTTP client για τη γλώσσα προγραμματισμού Java, σχεδιασμένη για αποτελεσματική επικοινωνία δικτύου σε εφαρμογές Android και Java. Ακολουθεί παράδειγμα για την εγκατάσταση της συγκεκριμένης βιβλιοθήκης:

- **Επεξεργασία του αρχείου build.gradle**
  Ανοίξτε το αρχείο build.gradle του έργου Android Studio σας. Βρίσκεται συνήθως στον φάκελο ”app”. Προσθέστε την εξής γραμμή στην εξάρτηση του αρχείου build.gradle.
  implementation 'com.squareup.okhttp3:okhttp:[version]'

- **Συγχρονισμός των Εξαρτήσεων**
  Μετά την προσθήκη της εξάρτησης, εκτελέστε τον συγχρονισμό των εξαρτήσεων για να ληφθούν τα απαραίτητα αρχεία. Κάντε κλικ στο κουμπί ”Sync Now” που εμφανίζεται στο πάνω μέρος του Android Studio.

- **Χρήση της OkHTTP**
  Μόλις εγκατασταθεί η βιβλιοθήκη, μπορείτε να τη χρησιμοποιήσετε στον κώδικά σας. Ανοίξτε τον κατάλογο όπου θέλετε να χρησιμοποιήσετε τη βιβλιοθήκη **OkHTTP** και προσθέστε τον ακόλουθο κώδικα.

HttpClient.class

```
public class HttpClient{
  OkHttpClient client = new OkHttpClient();
  public String run(String URL) throws IOException{
    Request request = new Request.Builder().url(URL).build();
    try(Response response = client.newCall(request).execute()){
      return response.body().string();
    }
  }
}
```

Αυτό είναι ένα απλό παράδειγμα για το πώς μπορείτε να χρησιμοποιήσετε τη βιβλιοθήκη OkHTTP για να κάνετε ένα αίτημα HTTP GET. Μπορείτε να προσαρμόσετε τον κώδικα σύμφωνα με τις ανάγκες σας.

Η Κλάση AsyncTask

Η AsyncTask [Dev23a] είναι μία κλάση που χρησιμοποείται για την εκτέλεση διεργασιών που δεν πρέπει να εκτελούνται στο κύριο νήμα της εφαρμογής. Στο Android, η διαχείριση της διεπαφής γίνεται συνήθως στο κύριο νήμα, και εάν εκτελέσουμε μακροχρόνια ή επιβαρυντικές εργασίες εκεί, προκαλούντε διακοπές στην εφαρμογή. Γι' αυτό τον λόγο η AsyncTask επιτρέπει την δημιουργία ενός νήματος φόντου και παρέχει μεθόδους για την αλληλεπίδραση με το κύριο νήμα με ασύγχρονο τρόπο. Έτσι, μπορούμε να εκτελέσουμε μακροχρόνιες εργασίες, όπως η λήψη δεδομένων από το διαδίκτυο χωρίς να παγώσουμε το κύριο νήμα της εφαρμογής. Ακολουθεί παράδειγμα χρήσης αυτής της κλάσης.

AsyncTast.java

```
private class MyAsyncTask extends AsyncTask<Void , Void , String>{
    @override
    protected String doInBackground(Void params){
        return "Apotelesma apo to nhma fontou";
    }
    @override
    protected void onPostExecute(String result){
        // Update UI
    }
}
```

Recycler Viewer και Adapter

Το Recycler View [Dev23b] είναι ένα στοιχείο του Android SDK που χρησιμοποιείται για τη δημιουργία και εμφάνιση λιστών στις εφαρμογές Android. Επιτρέπει την αποτελεσματική και ευέλικτη εμφάνιση μεγάλων συνόλων δεδομένων στο γραφικό περιβάλλον χρήστη, όπως λίστες με στοιχεία. Ένα RecyclerView επιτρέπει την ανακύκλωση των στοιχείων που εμφανίζονται στην οθόνη κατά την κύλιση, βελτιώνοντας την απόδοση και εξοικονομώντας πόρους μνήμης. Το Recycler View Adapter είναι ένα σημαντικό στοιχείο όταν χρησιμοποιείτε το Recycler View στο Android. Αποτελεί το μέσο με το οποίο ορίζετε το περιεχόμενο και την εμφάνιση των στοιχείων σας σε ένα Recycler View.

Ο βασικός στόχος ενός **RecyclerView.Adapter** είναι να συνδέσει τα δεδομένα σας με τα αντικείμενα που εμφανίζονται στο RecyclerView. Αυτό γίνεται με την υλοποίηση των παρακάτω τριών μεθόδων:

onCreateViewHolder(): Αυτή η μέθοδος χρησιμοποιείται για τη δημιουργία νέων αντικειμένων ViewHolder, τα οποία αντιπροσωπεύουν τα στοιχεία που εμφανίζονται στη λίστα. Καλείται λίγες φορές, ανάλογα με τον αριθμό των ορατών στοιχείων.

onBindViewHolder(): Αυτή η μέθοδος χρησιμοποιείται για την ανάθεση των δεδομένων από τη θέση του στο RecyclerView στο αντίστοιχο αντικείμενο ViewHolder. Καλείται κατά την προσπέλαση των δεδομένων.

getItemCount(): Αυτή η μέθοδος επιστρέφει τον συνολικό αριθμό των στοιχείων στη λίστα. Χρησιμοποιείται για την ενημέρωση του RecyclerView για τον αριθμό των στοιχείων.

Ο **RecyclerView.Adapter** σάς επιτρέπει να προσαρμόσετε την εμφάνιση και τη συμπεριφορά των στοιχείων στο RecyclerView, όπως τοποθέτηση πλαισίων, τίτλων, εικόνων, κουμπιών και άλλων στοιχείων σε κάθε στοιχείο της λίστας. Είναι ένα σημαντικό εργαλείο για τη διαχείριση και την προσαρμογή των στοιχείων που εμφανίζονται σε μια λίστα στο Android.

Έστω πως θέλουμε να εμφανίσουμε μία οριζόντια λίστα από προϊόντα με την μορφή που εμφανίζονται στην εφαρμογή. Δηλαδή δύο αρχικές εικόνες ακολουθόντας από δύο προβολές κειμένου. Το πρώτο βήμα είναι η μορφοποίηση της συγκεκριμένης δομής σε ένα xml αρχείο μέσα στο ευρετήριο layout του android. Παρακάτω παρέχεται η μορφοποίηση του αρχείου xml.

Εικόνα 6.1. Δομή Προϊόντος

Δομή Προϊόντος στο xml.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginLeft="10dp"
    android:layout_marginRight="10dp"
    android:layout_marginBottom="10dp">
    <ImageView
        android:id="@+id/market_product_txt"
        android:layout_width="48dp"
        android:layout_height="48dp"
        android:layout_gravity="center"
        android:background="@drawable/rect"/>
    <ImageView
        android:id="@+id/prouduct_txt"
        android:layout_width="72dp"
        android:layout_height="74dp"
        android:layout_gravity="center"
        android:scaleType="centerInside"
        android:layout_marginTop="5dp"></ImageView>
    <TextView
        android:id="@+id/product_name_txt"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textColor="#070606"
        android:gravity="center"
```

```
        android:ellipsize="end"
        android:maxLines="2"
        android:minLines="2"
        android:textSize="12dp"/>
    <TextView
        android:layout_marginTop="5dp"
        android:id="@+id/product_price_txt"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:textSize="15dp"
        android:textStyle="bold"
        android:textColor="@color/light_green"></TextView>
</LinearLayout>
```

Καθώς έχουμε ολοκληρώσει την μορφοποίηση, θα δημιουργήσουμε μία κλάση στην Java ονομάζοντάς την ProductClass η οποία θα έχει τις απαραίτητες μεταβλητές, Setters & Getters μαζί με έναν άδειο Constructor ή έναν Constructor που θα αρχικοποιεί και θα προσαρμόζει τις μεταβλητές. Ακολουθεί κώδικας στην Java της κλάσης ProductClass.

ProductClass.class

```
public class ProductClass implements Serializable {

    String market;
    String url;
    String price;
    String name;
    String id;
    String ASSORTEMTNS_DATA[][];
    String desc;
    String original_name;
    String brand_id;
    String coupon_value;
    String value_discount;

    public String getCoupon_value() {
        return coupon_value;
    }

    public void setCoupon_value(String coupon_value) {
        this.coupon_value = coupon_value;
    }

    public String getValue_discount() {
        return value_discount;
    }

    public void setValue_discount(String value_discount) {
        this.value_discount = value_discount;
    }

    public ProductClass(){}

    public String getMarket() {
        return market;
    }
    public void setMarket(String market) {
        this.market = market;
    }

    public String getUrl() {
```

```
41          return url;
42      }
43      public void setUrl(String url) {
44          this.url = url;
45      }
46
47      public String getPrice() {
48          return price;
49      }
50      public void setPrice(String price) {
51          this.price = price;
52      }
53
54      public String getName() {
55          return name;
56      }
57      public void setName(String name) {
58          this.name = name;
59      }
60
61      public void setID(String id) {this.id = id;}
62      public String getID() {return id;}
63
64      public void setASSORTEMTNS_DATA(String[][] ASSORTEMTNS_DATA) {this.
            ASSORTEMTNS_DATA = ASSORTEMTNS_DATA;}
65      public String[][] getASSORTEMTNS_DATA() {return ASSORTEMTNS_DATA;}
66
67      public String getBrand_id() {return brand_id;}
68
69      public void setBrand_id(String brand_id) {this.brand_id = brand_id;}
70
71      public void setDesc(String desc) {this.desc = desc;}
72      public String getDesc() {return desc;}
73
74      public void setOrigianlName(String original_name) {this.original_name =
            original_name;}
75      public String getOrigianlName() {return original_name;}
76
77      ProductClass(String coupon_value , String value_discount , String brand_id
            ,String market , String url , String price , String name , String id ,
            String ASSORTEMTNS_DATA[][] , String desc , String original_name){
78          this.market = market;
79          this.url = url;
80          this.price = price;
81          this.name = name;
82          this.id = id;
83          this.ASSORTEMTNS_DATA =  ASSORTEMTNS_DATA;
84          this.desc = desc;
85          this.original_name = original_name;
86          this.brand_id = brand_id;
87          this.coupon_value = coupon_value;
88          this.value_discount = value_discount;
89      }
90  }
```

Καθώς υλοποιήσαμε την συγκεκριμένη κλάση, συνεχίζουμε την υλοποίηση για το Recycler Viewer και το Adapter.

Recycler Viewer και Adapter

```
public class Adaptery extends RecyclerView.Adapter<Adaptery.MyViewHolder> {
    private Context mContext;
    private List<ProductClass> mData;
    private Adaptery.OnClickListener onClickListener;
    private ActivityType activityType;
    public Adaptery(Context mContext , List<ProductClass> mData, ActivityType
        activityType){
        this.mContext = mContext;
        this.mData = mData;
        this.activityType = activityType;
    }

    @NonNull
    @Override
    public MyViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int
        viewType) {
        View v;
        LayoutInflater inflater = LayoutInflater.from(mContext);
        if(activityType == ActivityType.MAIN_ACTIVITY)  v = inflater.inflate(R.
            layout.product_item , parent , false);
        else v = inflater.inflate(R.layout.product_item_search , parent , false
            );
        return new MyViewHolder(v);
    }


    @Override
    public void onBindViewHolder(@NonNull MyViewHolder holder, int position) {
        holder.txt_01.setText(mData.get(position).getName());
        holder.txt_02.setText(mData.get(position).getPrice().replace("." , ",")
            );
        Glide.with(mContext).load(mData.get(position).getUrl())
                .diskCacheStrategy(DiskCacheStrategy.AUTOMATIC)
                .skipMemoryCache(false)
                .placeholder(R.drawable.product_placeholder)
                .dontAnimate()
                .error(R.drawable.product_not_found)
                .priority(Priority.HIGH)
                .fitCenter()
                .into(holder.img_02);
        switch (mData.get(position).getMarket()) {
            case "MYMARKET": holder.img_01.setImageResource(R.drawable.mymarket
                ); break;
            case "ΜΑΣΟΥΤΗΣ": holder.img_01.setImageResource(R.drawable.
                masouths); break;
            case "ΓΑΛΑΞΙΑΣ":  holder.img_01.setImageResource(R.drawable.
                galaxias); break;
            case "ΑΒ ΒΑΣΙΛΟΠΟΥΛΟΣ": holder.img_01.setImageResource(R.drawable
                .ab); break;
            case "ΣΚΛΑΒΕΝΙΤΗΣ": holder.img_01.setImageResource(R.drawable.
                sklavenitis_logo); break;
            case "NULL": holder.img_01.setVisibility(View.GONE); break;
            default:  break;
        }
        holder.itemView.setOnClickListener(v -> { if (onClickListener != null)
             onClickListener.onClick(position , mData.get(position)); });
    }
    public void setOnClickListener(OnClickListener onClickListener) { this.
        onClickListener = onClickListener; }
    public interface OnClickListener { void onClick(int position , ProductClass
```

```
        model); }
49      @Override public int getItemCount() { return mData == null ? 0 : mData.size
            (); }
50      @Override public long getItemId(int position) { return position; }
51
52      public static class MyViewHolder extends RecyclerView.ViewHolder{
53          ImageView img_01;
54          TextView txt_01;
55          ImageView img_02;
56          TextView txt_02;
57          public MyViewHolder(@NonNull View itemView) {
58              super(itemView);
59               img_01 = itemView.findViewById(R.id.market_product_txt);
60               txt_01 = itemView.findViewById(R.id.product_name_txt);
61               img_02 = itemView.findViewById(R.id.prouduct_txt);
62               txt_02 = itemView.findViewById(R.id.product_price_txt);
63          }
64      }
65  }
```

Στην αρχή του κώδικα, πραγματοποιούνται οι απαραίτητες εισαγωγές βιβλιοθηκών, ορίζονται οι μεταβλητές της κλάσης, και ορίζεται η κλάση Adaptery ως υποκλάση της RecyclerView.Adapter. Ο Constructor της κλάσης δέχεται δύο παραμέτρους: το Context και μια λίστα αντικειμένων τύπου ProductClass. Οι παράμετροι αυτές αποθηκεύονται στις αντίστοιχες μεταβλητές της κλάσης. Η μέθοδος onCreateViewHolder χρησιμοποιείται για τη δημιουργία νέας καρτέλας για κάθε αντικείμενο της λίστας. Εδώ, χρησιμοποιείται η βιβλιοθήκη LayoutInflater για να δημιουργήσει την δομή της κάθε καρτέλας. Η μέθοδος onBindViewHolder χρησιμοποιείται για την εμφάνιση των δεδομένων στην καρτέλα. Εδώ, προσαρμόζονται τα δεδομένα από τη λίστα στα αντίστοιχα στοιχεία της καρτέλας. Επίσης, χρησιμοποιείται η βιβλιοθήκη Glide για τη φόρτωση εικόνων και η εμφάνιση τους στην καρτέλα. Στον κώδικα, υπάρχει και η δυνατότητα αλληλεπίδρασης με τον χρήστη. Μέσω της δήλωσης OnClickListener, παρέχεται η δυνατότητα κάποια ενέργειας όταν ο χρήστης κάνει κλικ σε ένα αντικείμενο της λίστας. Τέλος, η εσωτερική κλάση MyViewHolder χρησιμοποιείται για την καταχώρηση των στοιχείων της καρτέλας (εικόνες και κείμενο). Ορίζονται οι μεταβλητές που αντιστοιχούν στα στοιχεία της καρτέλας και αρχικοποιούνται μέσω της μεθόδου findViewById.

Η Async Task Κλάση CollectData

Οι τεχνολογίες που αναφέρθηκαν προηγουμένος παίζουν σημαντικό ρόλο στην κατανόηση του κώδικα της εφαρμογής που θα δούμε στην συνέχεια. Το σημαντικό κομμάτι κώδικα της εφαρμογής είναι μία κλάση Async Task σε συνδιασμό με τα προηγούμενα με σκοπό την λήψη δεδομένων σε μορφή JSON από το API του Pockee και την οπτικοποίηση με ομοιόμορφο τρόπο στην εφαρμογή. Ακολουθεί παράδειγμα πηγαίου κώδικα.

Η Κλάση CollectData

```
1  private class CollectData extends AsyncTask<String,String,Void>{
2  private List<ProductClass> PRODUCTS;
3  private String API;
4  private CategoryClass c;
5  private Adaptery adp;
6  private String TITLE;
7  private String DESC;
8  private String BRAND;
9  CollectData(String API ,
```

```
10              String TITLE,
11              String DESC,
12              String BRAND,
13              List<ProductClass> PRODUCTS){
14      this.API = API;
15      this.PRODUCTS = PRODUCTS;
16      CategorySelector = TITLE;
17      this.TITLE = TITLE;
18      this.DESC = DESC;
19      this.BRAND = BRAND;
20  }
21
22  CollectData(String API , String BRAND , List<ProductClass> PRODUCTS){
23      this.API = API;
24      this.PRODUCTS = PRODUCTS;
25      this.BRAND = BRAND;
26      this.TITLE = "NULL";
27      this.DESC = "NULL";
28  }
29
30  @Override
31  protected void onPreExecute() {
32      super.onPreExecute();
33      c = new CategoryClass();
34      c.setCategory_title(TITLE);
35      c.setCategory_desc(DESC);
36      c.setStatus(false);
37      c.setCategory_brand(BRAND);
38  }
39
40  List<ProductClass> productList = new ArrayList<>();
41  @Override
42  protected List<ProductClass> doInBackground(String... strings) {
43      try {
44          OkHttpClient client = new OkHttpClient.Builder()
45                  .connectTimeout(15, TimeUnit.SECONDS)
46                  .readTimeout(15, TimeUnit.SECONDS)
47                  .build();
48          Request request = new Request.Builder()
49                  .url(API)
50                  .addHeader("Content-Type", "application/json")
51                  .addHeader("Accept", "application/json")
52                  .build();
53          Response response = client.newCall(request).execute();
54          if (response.isSuccessful()) {
55              ResponseBody responseBody = response.body();
56              if (responseBody != null) {
57                  JSONArray data = new JSONObject(responseBody.string()).
                        getJSONArray("data");
58                  if (data.length() == 0) return productList;
59                  c.setCategory_brand(BRAND);
60                  DecimalFormat decimalFormat = new DecimalFormat("#0.00");
61                  for (int i = 0; i < data.length(); i++) {
62                      ProductClass model = new ProductClass();
63                      JSONObject product = data.getJSONObject(i);
64                      double final_price = Double.MAX_VALUE;
65                      JSONArray assortments = product.getJSONArray("assortments")
                            ;
66                      String[][] assortmentsData = new String[assortments.length
                            ()][2];
67
```

```
68                      if (product.has("coupons") && product.getJSONArray("coupons
                            ").length() > 0) {
69                          JSONObject coupon = product.getJSONArray("coupons").
                                getJSONObject(0);
70                          if (coupon.getDouble("value") > 0) model.
                                setCoupon_value(decimalFormat.format(coupon.
                                getDouble("value")));
71                          else model.setCoupon_value("null");
72                          if (coupon.getDouble("value_discount") > 0) model.
                                setValue_discount(decimalFormat.format(coupon.
                                getDouble("value_discount")));
73                          else model.setValue_discount("null");
74                      }
75
76                      if (!product.isNull("image_versions")) model.setUrl(product
                            .getJSONObject("image_versions").getString("thumb"));
77                      else model.setUrl("https://d3kdwhwrhuoqcv.cloudfront.net/
                            uploads/products/product-image-404.png");
78
79
80                      for (int j = 0; j < assortments.length(); j++) {
81                          JSONObject item = assortments.getJSONObject(j);
82                          JSONObject retailer = item.getJSONObject("retailer");
83                          JSONObject productPivot = item.getJSONObject("
                                product_pivot");
84                          String name = retailer.getString("name");
85                          if (!supermarkets.contains(name)) continue;
86
87                          double current_price = productPivot.isNull("final_price
                                ")
88                                  ? productPivot.getDouble("start_price")
89                                  : productPivot.getDouble("final_price");
90                          if (current_price < final_price) {
91                              model.setMarket(name.trim());
92                              final_price = current_price;
93                              model.setPrice(current_price + " €");
94                          }
95                          assortmentsData[j][0] = name;
96                          assortmentsData[j][1] = String.valueOf(current_price);
97                      }
98                      if (final_price == Double.MAX_VALUE) continue;
99
100                     model.setName(product.getString("name"));
101                     model.setID(product.getString("id"));
102                     model.setASSORTEMTNS_DATA(assortmentsData);
103                     model.setDesc(product.getString("description"));
104                     model.setOrigianlName(product.getString("name"));
105                     model.setBrand_id( product.getString("brand_id"));
106                     productList.add(model);
107                 }
108             }
109         }
110         response.close();
111     } catch (Exception e) {
112         // Handle exception
113     }
114     return productList;
115 }
116
117 @Override
118 protected void onPostExecute(List<ProductClass> productList) {
119     super.onPostExecute(productList);
```

```
120     try {
121         if(!productList.isEmpty()){
122             if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.N) productList.
                    sort(Comparator.comparingDouble(product -> Double.parseDouble(
                    product.getPrice().replace(" €", ""))));
123             else {
124                 Collections.sort(productList, new Comparator<ProductClass>() {
125                     @Override
126                     public int compare(ProductClass p1, ProductClass p2) {
127                         double price1 = Double.parseDouble(p1.getPrice().
                                replace(" €", ""));
128                         double price2 = Double.parseDouble(p2.getPrice().
                                replace(" €", ""));
129                         return Double.compare(price1, price2);
130                     }
131                 });
132             }
133             adp = new Adaptery(getBaseContext(), productList, ActivityType.
                    MAIN_ACTIVITY);
134             adp.setHasStableIds(true);
135             adp.setOnClickListener((position, model) -> {
136                 Intent intent = new Intent(MainActivity.this, ProductView.class
                        );
137                 intent.putExtra("PRODUCT_OBJ", model);
138                 startActivity(intent);
139                 overridePendingTransition(0, 0);
140             });
141             c.setCategory_holder(adp);
142             NotifyAdapter(c);
143         } else c.setCategory_holder(null);
144     } catch (Exception e) {
145         Log.e("MAIN_ACTIVITY" , e.getLocalizedMessage());
146     }
147 }
```

Αυτή η εσωτερική κλάση εκτελεί τις εργασίες που απαιτούνται για την ανάκτηση, επεξεργασία και παρουσίαση δεδομένων. Υπάρχουν δύο κατασκευαστές, ανάλογα με τη χρήση: Κατηγορία (Category): Χρησιμοποιείται για την ανάκτηση δεδομένων σχετικά με μια κατηγορία προϊόντων. Λαμβάνει την API διεύθυνση, τον τίτλο, την περιγραφή και το όνομα της μάρκας, καθώς και μια λίστα προϊόντων. Μεμονωμένο (Solo): Χρησιμοποιείται για την ανάκτηση δεδομένων ενός μεμονωμένου προϊόντος χωρίς να περιλαμβάνει κεντρική κατηγορία από πάνω. Λαμβάνει την API διεύθυνση, τη μάρκα και μια λίστα προϊόντων. Οι δύο αυτοί κατασκευαστές παίζουν ρόλο στην εμφάνιση της κατηγορίας και τον προϊόντων της εφαρμογής. onPreExecute, σε αυτήν τη μέθοδο πραγματοποιείται η προετοιμασία πριν από την εκτέλεση των εργασιών στο παρασκήνιο. Δημιουργείται ένα αντικείμενο της κλάσης CategoryClass, ονομασμένο c, και γίνονται κάποιες ρυθμίσεις. doInBackground, αυτή η μέθοδος εκτελείται στο παρασκήνιο και περιλαμβάνει τις εξής εργασίες: Ενημερώνει την API διεύθυνση ανάλογα με τις απαιτήσεις. Ανακτά δεδομένα από την API και τα μετατρέπει σε μορφή JSON. Επεξεργάζεται τα δεδομένα JSON για να αποσπάσει πληροφορίες προϊόντων. Δημιουργεί αντικείμενα τύπου ProductClass για κάθε προϊόν. Ενημερώνει τη λίστα PRODUCTS με τα προϊόντα και την ταξινομεί βάσει της τιμής. Δημιουργεί ένα αντικείμενο τύπου Adaptery για την παρουσίαση των προϊόντων.

Ορίζει έναν ακροατή για τα στοιχεία της λίστας προϊόντων. Ολοκληρώνει τη διαδικασία, ενημερώνοντας τη διεπαφή χρήστη. onPostExecute, σε αυτήν τη μέθοδο πραγματοποιείται η ολοκλήρωση της διαδικασίας. Ανάλογα με την κατηγορία, ενημερώνονται συγκεκριμένα στοιχεία στη διεπαφή χρήστη.

Ένα απλό παράδειγμα χρήσης είναι το εξής.

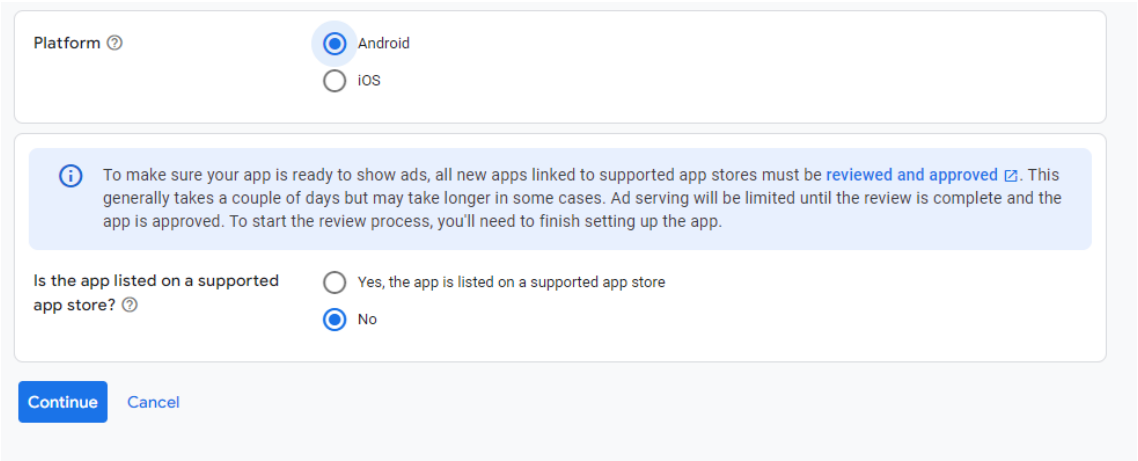
Χρήση CollectData Κλάσης

```
1 new CollectData("www.API.gr/data.json", "Category", "Description", "Product Brand", new ArrayList<>()).executeOnExecutor(pool)
2
3 new CollectData("www.API.gr/data.json", "Product Brand", new ArrayList<>()).executeOnExecutor(pool)
```

Εικόνα 6.2. Αποτέλεσμα της Async Task

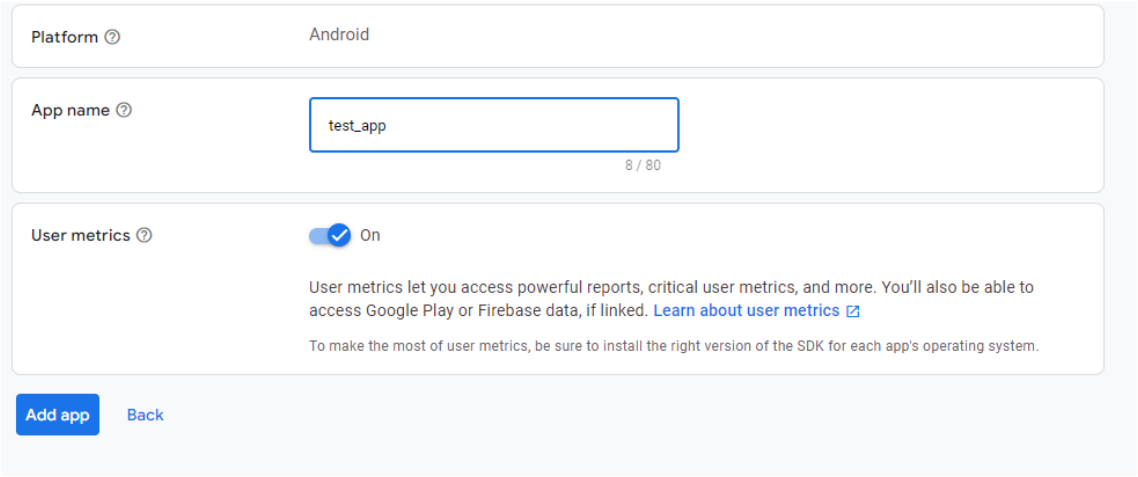
Εγκατάσταση διαφημίσεων μέσω της Google AdMob

Το Google AdMob [adm] [anda] αποτελεί μια διαφημιστική πλατφόρμα που ανήκει στη Google και επιτρέπει στις εφαρμογές κινητών να ενσωματώνουν διαφημίσεις και να κερδίζουν έσοδα μέσω αυτών. Είναι σχεδιασμένο για προγραμματιστές εφαρμογών που θέλουν να μονετάρουν τις εφαρμογές τους μέσω διαφημίσεων. Το AdMob υποστηρίζει διάφορες μορφές διαφημίσεων, συμπεριλαμβανομένων banner ads, interstitial ads, video ads και native ads. Οι διαφημιζόμενοι πληρώνουν για τις διαφημίσεις με βάση το μοντέλο CPC (Cost Per Click) ή CPM (Cost Per Mille), ενώ οι εφαρμογές λαμβάνουν ένα μερίδιο των εσόδων. Το Google AdMob παρέχει επίσης αναλυτικά στατιστικά στοιχεία και εργαλεία για την παρακολούθηση της απόδοσης των διαφημίσεων και των εσόδων.
Στην συγκεκριμένη ενότητα θα δούμε αναλυτικά τα βήματα τα οποία θα μας επιτρέψουν να προσθέσουμε τέτοιες διαφημίσεις στην εφαρμογή μας. Αρχικά απαιτείται η εγκατάσταση και ο συγχρονισμός της υπηρεσίας **gms-play-services-ads**.
implementation 'com.google.android.gms:play-services-ads:[version]'
Έχοντας δημιουργήσει έναν λογαριασμό AdMob, το πρώτο βήμα είναι να καταχωρίσετε την εφαρμογή σας στο AdMob. Αυτό το βήμα δημιουργεί μια εφαρμογή AdMob με ένα μοναδικό αναγνωριστικό εφαρμογής AdMob που απαιτείται στην συνέχεια εντός της εφαρμογής.



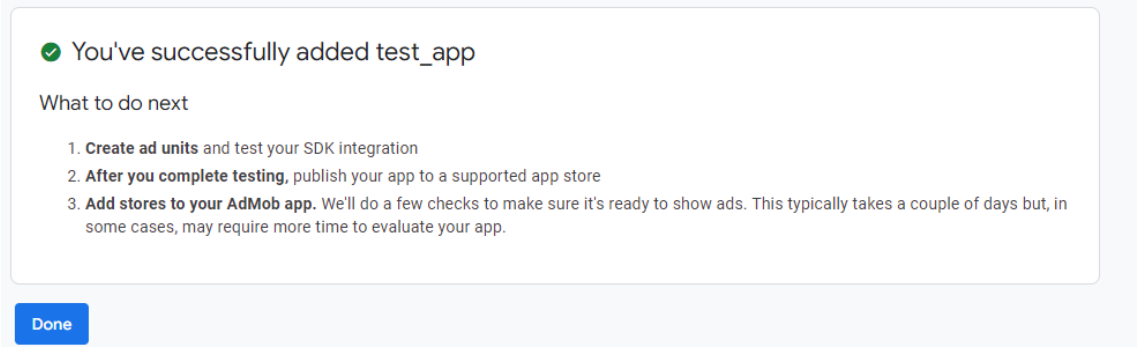
Εικόνα 6.3. Εγγραφή εφαρμογής στο Admob 1ο Βήμα

Επιλέγουμε το όνομα της εφαρμογής μας καθώς και την πλατφόρμα, παρέχει επίσης την δυνατότητα για iOs λειτουργικό σύστημα σε περίπτωση που δημιουργείται εφαρμογή για αυτο. Τέλος πατάμε το ***add app*** για να προσθέσουμε την εφαρμογή στο Ad Mob και να συνεχίσουμε με την σύνδεση των διαφημίσεων.



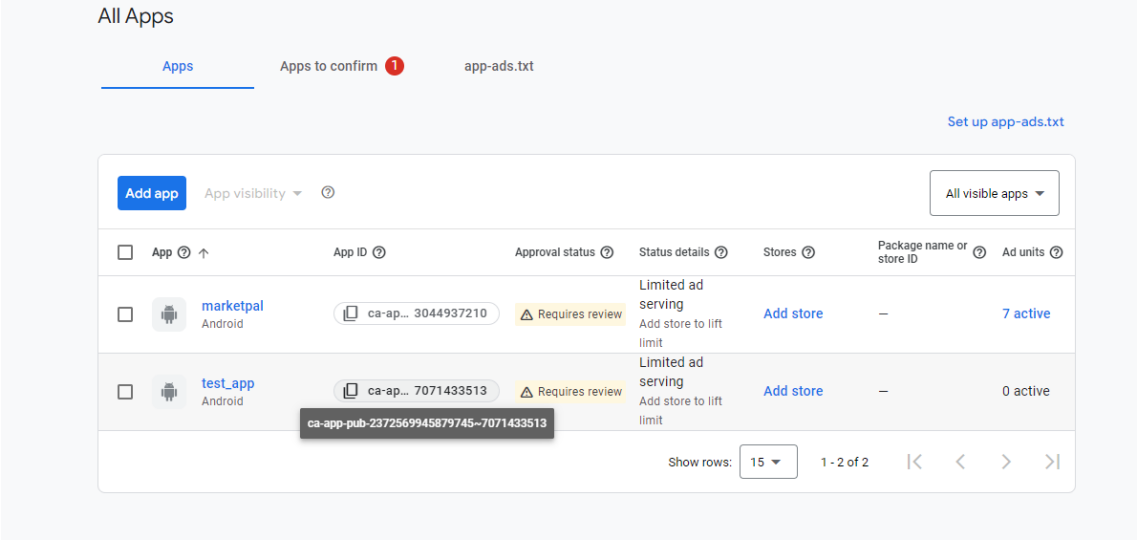
Εικόνα 6.4. Εγγραφή εφαρμογής στο Admob 2ο Βήμα

Καθώς η εγγραφή της εφαρμογής είναι επιτυχής θα λάβουμε το εξής μήνυμα.



Εικόνα 6.5. Εγγραφή εφαρμογής στο Admob 3ο Βήμα

Κατά την επιτυχής εγγραφή μας παρέχεται ένα μοναδικό κλειδί. Μπορείται να δείτε το δικό σας κλειδί ακολουθόντας τα βήματα **Apps** → **view all apps**.



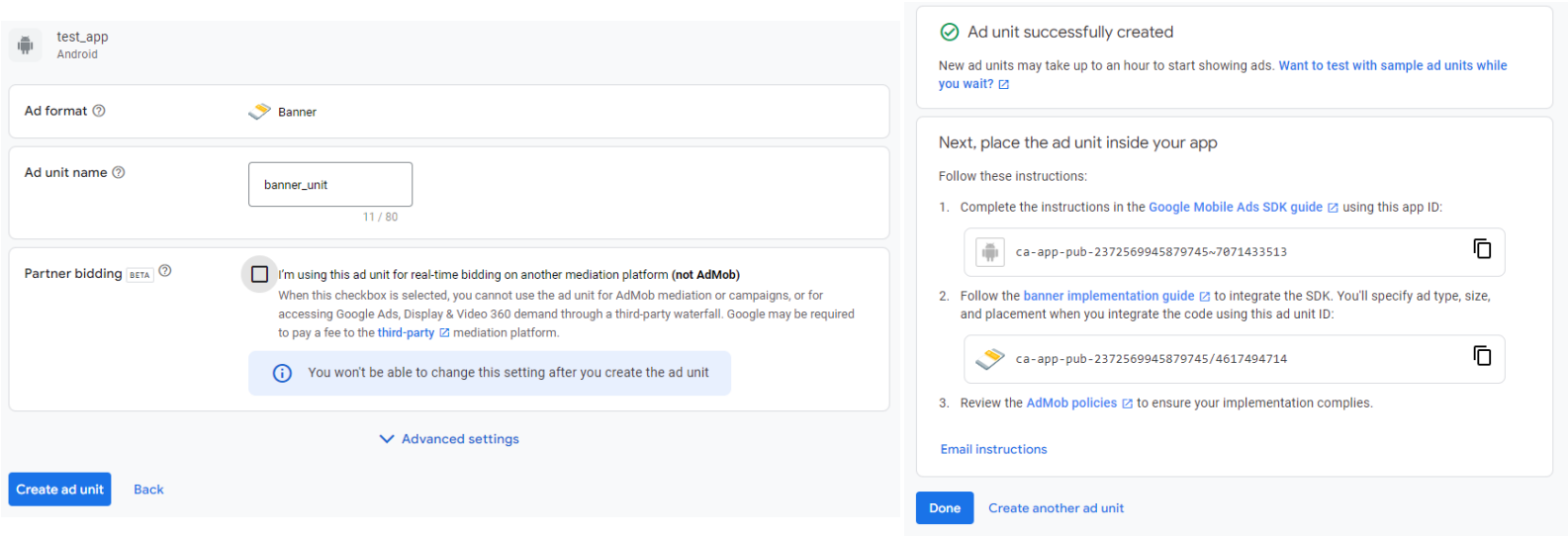
Εικόνα 6.6. AdMob - Μοναδικός Αριθμός Εφαρμογής

Καθώς κάναμε την εγγραφή της εφαρμογής μας στο AdMob, θα πάμε στο AndroidManifest.xml της εφαρμογής μας και στο Application θα προσθέσουμε την εξής εντολή:

```
<meta-data android:name="com.google.android.gms.ads.APPLICATION_ID"
           android:value="ID"/>
```

όπου ID είναι ο μοναδικός αριθμός της εφαρμογής που προσθέσαμε. Το επόμενο βήμα είναι η δημιουργία ενός Ad Unit όπου θα μας επιτρέψει να επιλέξουμε την δομή και την ενσωμάτωση της διαφήμισης. Επιλέγοντας την εφαρμογής μας από τα **Apps**. Στο αριστερό μέρος θα μας βγάλει την επιλογή **Ad Units**, εκεί θα μπορέσουμε να δημιουργήσουμε το Ad Unit. **Ad Unit** → **Add ad unit**.

Ακολουθεί παράδειγμα δημιουργίας ad unit.



Εικόνα 6.7. Δημιουργία Banner και ολοκλήρωση του Ad Unit

Καθώς η δημιουργία Ad Unit ήταν επιτυχής συνεχίζουμε στο βήμα της ενσωμάτωσης, όπου θα χρησιμοποιήσουμε το μοναδικό αριθμό του ad unit για να δημιουργήσουμε στην διάταξη μας ενα AdView. Αυτό μπορεί να γίνει προγραμματιστικά ή κατευθειάν στην διάταξη .xml μας.

Προγραμματιστικά

```
1    AdView adView = new AdView(this);
2
3    adView.setAdSize(AdSize.BANNER);
4
5    adView.setAdUnitId("ca-app-pub-3940256099942544/6300978111");
```

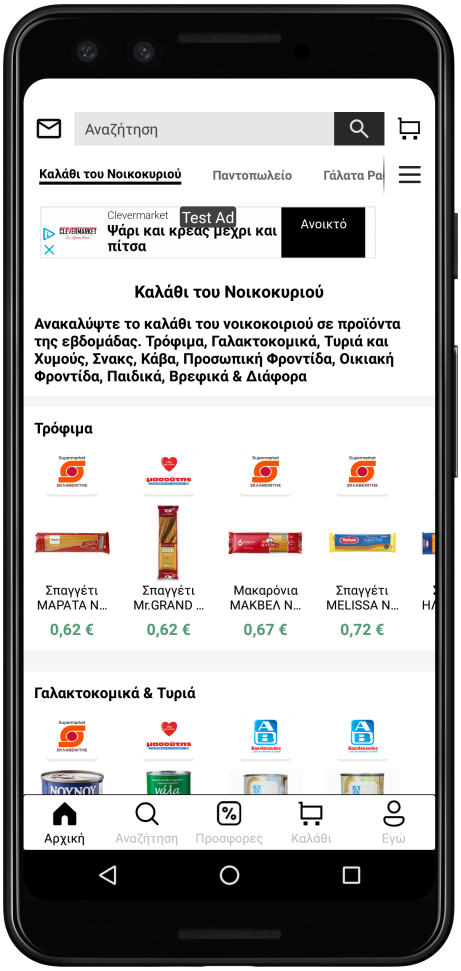
xml

```
1  <com.google.android.gms.ads.AdView
2      xmlns:ads="http://schemas.android.com/apk/res-auto"
3      android:id="@+id/adView"
4      android:layout_width="wrap_content"
5      android:layout_height="wrap_content"
6      android:layout_centerHorizontal="true"
7      android:layout_alignParentBottom="true"
8      ads:adSize="BANNER"
9      ads:adUnitId="ca-app-pub-ad-unit-id">
10 </com.google.android.gms.ads.AdView>
```

Ολοκληρώνοντας τα παραπάνω βήματα, η εφαρμογή μας θα είναι σε θέση να εμφανίσει διαφημίσεις. Ο πλήρης οδηγός εγκατάστασης παρέχεται στον παρακάτω σύνδεσμο. (https://developers.google.com/admob/android/quick-start)

Μετά την ολοκλήρωση των παραπάνω βημάτων και την επιτυχημένη εγκατάσταση της εφαρμογής, η δυνατότητα εμφάνισης διαφημίσεων ανοίγει νέες προοπτικές για τη χρηματοδότηση και την ανάπτυξη της εφαρμογής σας. Το AdMob, ως πλατφόρμα διαφημίσεων της Google, παρέχει έναν εύκολο και αποτελεσματικό τρόπο για την ενσωμάτωση διαφημίσεων στην εφαρμογή σας. Με τον πλήρη οδηγό εγκατάστασης που παρέχεται στον παραπάνω σύνδεσμο, μπορείτε να προχωρήσετε σε εξατομικευμένες ρυθμίσεις και προσαρμογές σύμφωνα με τις ανάγκες σας. Είναι σημαντικό να διαβάσετε προσεκτικά τον οδηγό για να εξασφαλίσετε ότι οι διαφημίσεις ενσωματώνονται σωστά και ότι η εμπειρία του χρήστη διατηρείται

υψηλή. Επιπλέον, μπορείτε να εξερευνήσετε τις διάφορες επιλογές διαφημίσεων που προσφέρει το AdMob, όπως τις ενσωματωμένες διαφημίσεις, τις διαφημίσεις με προσαρμοσμένα μεγέθη, και άλλες. Επίσης, μπορείτε να παρακολουθήσετε την απόδοση των διαφημίσεων σας μέσω του AdMob Dashboard, προσφέροντας έτσι πλήρη εικόνα του πώς οι διαφημίσεις συμβάλλουν στην επιτυχία της εφαρμογής σας. Τέλος, μην ξεχνάτε να τηρείτε τους όρους χρήσης και τις πολιτικές του AdMob για να αποφεύγετε προβλήματα και να διασφαλίσετε τη συνεχή συνεργασία με την πλατφόρμα. Με αυτόν τον τρόπο, η εφαρμογή σας θα είναι έτοιμη να εκμεταλλευτεί πλήρως τις δυνατότητες των διαφημίσεων και να προσφέρει μια ενισχυμένη εμπειρία χρήστη.



Εικόνα 6.8. Παράδειγμα Διαφήμισης

Υλοποίηση Σύγκρισης Βέλτιστων Επιλογών στην Διεπαφή

Ο παρακάτω κώδικας εκτελεί μια επανάληψη μέσω όλων των εγγραφών ενός αντικειμένου shopping.cart, το οποίο αντιπροσωπεύει το καλάθι αγορών με προϊόντα. Κάθε εγγραφή αντιστοιχεί σε ένα προϊόν, με το όνομα του προϊόντος να λειτουργεί ως κλειδί και η τιμή του να αντιπροσωπεύει την τιμή του προϊόντος.

Εντός του βρόχου επανάληψης, ο κώδικας εξάγει τις απαραίτητες πληροφορίες για κάθε προϊόν:

1. Λαμβάνει το όνομα του προϊόντος.
2. Λαμβάνει την τιμή του προϊόντος και την αποθηκεύει στη μεταβλητή PRICE.
3. Υπολογίζει το συνολικό κόστος του προϊόντος (PRICE.TYPECAST.DOUBLE) πολλαπλασιάζοντας την τιμή του με τον συνολικό αριθμό του προϊόντος.
4. Αναζητά και αποθηκεύει την αγορά (market) με τη χαμηλότερη τιμή.
5. Λαμβάνει το URL της εικόνας του προϊόντος.

Συνολικά, ο κώδικας αυτός εκτελεί λογική για κάθε προϊόν στο καλάθι, προσδιορίζοντας τις απαραίτητες πληροφορίες και τοποθετώντας το προϊόν στην κατάλληλη κατηγορία, με βάση τη χαμηλότερη τιμή ανάμεσα στις διαθέσιμες αλυσίδες.

Κώδικας Βέλτιστων Επιλογών

```
        Map<String, ?> allEntries = shopping_cart.getAll();
        for (Map.Entry<String, ?> entry : allEntries.entrySet()) {
            String key = entry.getKey();

            String PRICE = // GET PRICE
            Double PRICE_TYPECAST_DOUBLE = // GET PRICE * TOTAL NUMBER OF THE
                PRODUCT

            String market = // GET MARKET WITH LOWEST VALUE
            String url = // IMAGES OF PRODUCT

            ProductClass model = new ProductClass();
            model.setName(key);
            model.setUrl(url);
            model.setMarket("NULL");
            model.setOrigianlName(key);

            if(market.equals("S1")){
                 // Add to MYMARKET.
            }else if(market.equals("S2")){
                // ...
            }else if(market.equals("S3")){
                // ...
            }else if(market.equals("S3")){
                // ...
            }else if(market.equals("S4")){
                // ...
            }
        }
```