ΔΙΕΘΝΕΣ ΠΑΝΕΠΙΣΤΗΜΙΟ ΤΗΣ ΕΛΛΑΔΟΣ

Παράρτημα Σερρών

Τμήμα Μηχανικών Πληροφορικής

Πτυχιακή εργασία – Μιχάλης Αναγνώστου

ΑΕΜ: 3620

Θέμα:

Εφαρμογή σε android για διαχείριση εσόδων-εξόδων (Android application for income-outcome management)

Επιβλέπων καθηγητής: Αλκιβιάδης Τσιμπίρης

ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα να ευχαριστήσω τον Επίκουρο καθηγητή Τσιμπίρη Αλκιβιάδη, για την εμπιστοσύνη του, την στηριξή του καθώς και για την δυνατότητα που μου έδωσε να εκπονήσω την συγκεκριμένη πτυχιακή εργασία. Μια εργασία καινοτόμα, ενδιαφέρουσα η οποία συνέβαλε στην προσωπική μου εξέλιξη όσο και στις γνώσεις που έλαβα.

Επίσης, να ευχαριστήσω όλους τους καθηγητές του τμήματος για την διδασκαλία τους και την υπομονή τους όλα αυτά τα χρόνια. Για την προσπάθεια που καταβάλουν καθημερινά και τις γνώσεις που μοιράζονται.

Τέλος να ευχαριστήσω την διοίκηση και την γραμματεία του τμήματος για την βοήθεια και την συνεισφορά τους στους φοιτητές.

Περιεχόμενα

1.ΕΙΣΑΓΩΓΗ

1.1)Τι είναι μία εφαρμογή κινητού

Μία εφαρμογή κινητού είναι ενα προϊόν λογισμικού εκτελέσιμο σε κινητά smartphones , tablets και φορητές συσκευές. Για την εγκατάστασή τους χρησιμοποιούνται ειδικές πλατφόρμες διανομής εφαρμογών(Play Store, App Store, Google Play).

Οι εφαρμογές κινητού έχουν ως κύριο στόχο την διευκόλυνση του χρήστη στην καθημερινότητα του. Εφαρμογές για email, ημερολόγιο, αριθμομηχανής, καιρό είχαν στόχο την προσφορά στο γενικό σύνολο. Ωστόσο, η αύξηση της ζήτησης είχε ως αποτέλεσμα την συνεχή παραγωγή καινοτόμων εφαρμογών. Με γρήγορους ρυθμούς αναπτύχθηκαν εφαρμογές location based(GPS), αυτοματοποιημένες εφαρμογές για γρήγορες συναλλαγές όπως αγορές.

Εφαρμογές κοινωνικής δικτύωσης(social media) είναι πλέον από τα δημοφιλέστερα μέσα επικοινωνίας(Facebook, Twitter, Instagram). Η έννοια του Κοινωνικού Δικτύου μας παραπέμπει σε μια Κοινωνική Δομή (Social Structure), που στη βασική της θεώρηση αποτελείται από Κόμβους (Nodes) και Δεσμούς (Ties) μεταξύ των Κόμβων. Οι Κόμβοι μπορεί να είναι άνθρωποι ή οργανισμοί και οι Δεσμοί μπορεί να είναι κάθε είδους σχέσεις μεταξύ των Κόμβων αλλά και κάθε είδους αλληλεξαρτήσεις.

1.2) Λίγα λόγια για το Android

Το Android είναι λειτουργικό σύστημα για συσκευές κινητής τηλεφωνίας το οποίο τρέχει τον πυρήνα του λειτουργικού linux. Δημιουργήθηκε από την Google και αργότερα απ την Open Handset Alliance. Επιτρέπει στους προγραμματιστές την ανάπτυξη εφαρμογών με κώδικα της γλώσσας προγραμματισμού JAVA, C#, Kotlin. Είναι σχεδιασμένο για συσκευές με οθόνη αφής, αυτοκίνητα , τηλεοράσεις και ρολόγια χειρός.

Το Android είναι το πιο δημοφιλής λογισμικό στον κόσμο. Έχει τις περισσότερες πωλήσεις ευρέως ξεπερνώντας Windows, iOS και Mac OS μαζί.

Η πρώτη παρουσίαση έγινε στις 5 Νοεμβρίου το 2007. Η Google δημοσίευσε το μεγαλύτερο μέρος του κώδικα υπό τους όρους της Apache Alliance, μιας ελεύθερης άδειας λογισμικού.

Εικόνα 1: Το λογότυπο του Android

1.3) Πλεονεκτήματα Android

Το Android έκανε επανάσταση, με την στρατηγική της Google να διανέμει στην αγορά ανοικτό και ελεύθερο λειτουργικό σύστημα. Η χρήση του ελεύθερου λογισμικού έχει μειώσει δραματικά το τελικό κόστος ενός κινητού τηλεφώνου. Διαφέρει από τα άλλα λειτουργικά συστήματα στην προσέγγιση του στην καθολικότητα. Τα πλεονεκτήματα είναι τα ακόλουθα

- Είναι ανοικτή πλατφόρμα. Η εγκατάσταση των εφαρμογών είναι δωρεάν καθώς και η αναπτυξη τους είναι open source.

- Εφαρμογή συστήματος συγχρονισμού. Ο χρήστης έχει ένα πανομοιότυπο σύνολο προγραμμάτων σε διαφορετικές συσκευές χωρίς να καταβάλει κάποια προσπάθεια.

- Υποστήριξη κάρτας μνήμης. Εξοικονόμηση αποθηκευτικού χώρου.

- Απεριόριστα προσαρμόσιμη. Οι εφαρμογές Android παρέχουν μεγαλύτερη ευελιξία και προσαρμογή από άλλες(iOS).

- Υποστήριξη ασύρματου πρωτοκόλλου Bluetooth.

- Τεράστια ποικιλία επιλογής κινητού με λογισμικό Android , καθώς είναι ανοιχτό λογισμικό και ελεύθερο, οπότε υπάρχει μεγάλος ανταγωνισμός στην κατασκευαστική αγορά.

1.3.1) Τρέχουσες Εκδόσεις

Ένα λογισμικό με πολλές εκδόσεις για την ικανοποιήση των πολλαπλών αναγκών του χρήστη και του κατασκευαστή εφαρμογών. Η ιστορία εκδόσεων του Android του λειτουργικού συστήματος των κινητών ξεκίνησε με την κυκλοφορία του Android beta το Νοέμβριο του 2007.

Έκδοση	Νούμερο έκδοσης	Ημερομηνία κυκλοφορίας	Επίπεδο API
Alpha	1.0	23/9/2008	1
Beta	1.1	9/2/2009	2
Cupcake	1.5	27/4/2009	3
Donut	1.6	15/9/2009	4
Éclair	2.0 - 2.1	26/10/2009	5-7
Froyo	2.2 - 2.2.3	20/5/2010	8
Gingerbread	2.3 - 2.3.7	6/12/2010	9-10
Honeycomb	3.0 - 3.2.6	22/2/2011	11-13
Ice cream	4.0 - 4.0.4	18/10/2011	14-15

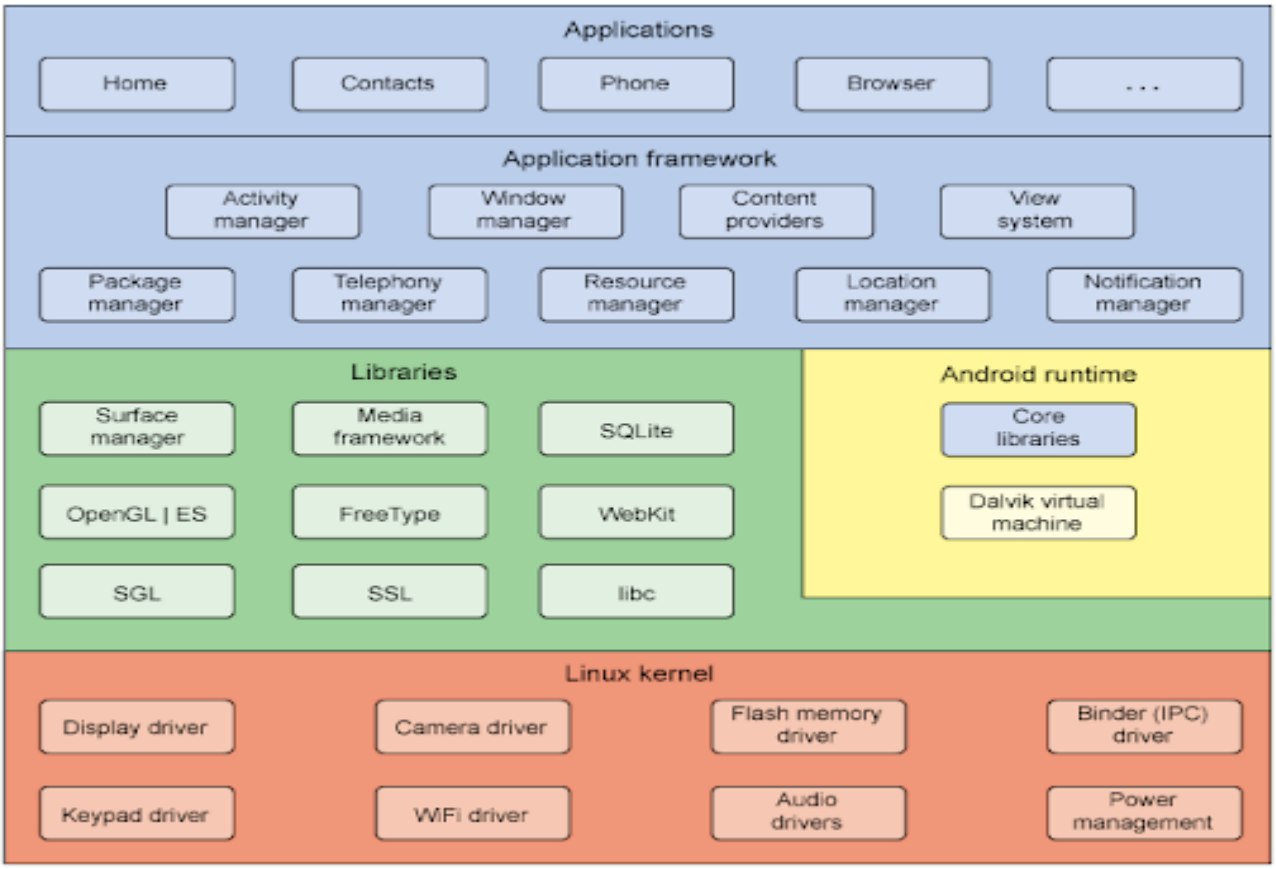
sandwich			
Jelly Bean	4.1 - 4.3.1	9/7/2012	16-18
KitKat	4.4 – 4.4.4	31/10/2013	19-20
Lollipop	5.0 – 5.1.1	12/11/14	21-22
Marshmallow	6.0 – 6.0.1	5/10/2015	23
Nougat	7.0 – 7.1.2	22/8/2016	24-25
Oreo	8.0 – 8.1	21/8/2017	26-27
Pie	9.0	6/8/2018	28
Q	10.0	3/9/2019	29
R	11.0	19/2/2020	30

Πίνακας εκδόσεων Android

Πλέον οι εκδόσεις που χρησιμοποιουνται είναι από την KitKat(4.4) και έπειτα.

1.4) Αρχιτεκτονική Android

Παρουσιάζεται η αρχιτεκτονική του Android όπως φαίνεται στην Εικόνα 2. Αποτελείται από 5 επίπεδα : Linux kernel, Libraries, Android runtime, Application framework, Applications.



Εικόνα 2: Αρχιτεκτονική Android

1.4.1) Linux kernel

Ένας πυρήνας λειτουργικού συστήματος της οικογένεις Linux. Το Linux είναι ένα ευρεώς γνωστό ελεύθερο λογισμικό. Οι αρμοδιότητες του πυρηνα είναι:

1. Διαχείριση μνήμης. Παρακολουθεί πόση μνήμη χρησιμοποιείται, τι θα αποθηκεύσει και που.
2. Διαχείριση εργασιών. Προσδιορίζει ποιές διαδικασίες μπορούν να χρησιμοποιήσουν την κεντρική μονάδα επεξεργασίας, πότε και για πόσο καιρό.
3. Προγράμματα οδήγησης συσκευών. Ενεργεί ως διερμηνέας μεταξύ του υλικού και των διαδικασιών.
4. Κλήσεις συστήματος και ασφάλεια. Λαμβάνει αιτήματα για υπηρεσία απο τις διαδικασίες.

1.4.2) Libraries

Οι βιβλιοθήκες που χρησιμοποιούνται στις εφαρμογές Android περιλαμβάνουν όλα όσα χρειάζονται για την δημιουργία μιας εφαρμογής, τον πηγαίο κώδικα , resource files, Android Manifest. Είναι γραμμένες συνήθως σε γλώσσα c/c++ και χρησιμοποιούνται απο δίαφορα components της εφαρμογής.

Κάποιες απ τις βασικές βιβλιοθήκες ειναι:

- System C Library
- SQLite
- Media Libraries
- SGL(2D graphics)
- 3D Libraries

1.4.3) Android Runtime

Είναι το περιβάλλον εκτέλεσης εφαρμογών που χρησιμοποιείται απ το λειτουργικό σύστημα Android.

Εφαρμογές που είναι γραμμένες σε κώδικα Java, μετατρέπονται σε κώδικα byte, συσκευάζονται ως apk και εκτελούνται στο Runtime. Αυτος ο χρόνος εκτέλεσης μπορεί να είναι είτε ART είτε DVM. Το Dalvik virtual machine(DVM) ήταν ο προεπιλεγμένος χρόνος εκτέλεσης μέχρι την έκδοση KitKat. Από το Lollipop και μετά, το Android Runtime είναι η προεπιλεγμένη πλατφόρμα για την εκτέλεση των εφαρμογών Android. Το ART ισχυρίζεται ότι είναι γρηγορότερος χρόνος εκτέλεσης απο το DVM, καθώς προχωρά στην προετοιμασία που μετατρέπει τα Android apks σε odex για να βελτιώσει την απόδοση της εφαρμογής.

1.4.4) Application Framework

Το επίπεδο Application Framework παρέχει πληροφορίες υψηλού επιπέδου σε εφαρμογές με την μορφή κλάσεων Java. Οι προγραμματιστές μπορούν να κάνουν χρήση αυτών των υπηρεσιών στις εφαρμογές τους.

Οι βασικές υπηρεσίες είναι οι ακόλουθες:

- Activity Manager. Ελέγχει όλες τις πτυχές ζωής της εφαρμογής.
- Content Providers. Επιτρέπει στις εφαρμογές τη δημοσίευση και την κοινή χρήση με άλλες εφαρμογές.
- Resource Manager. Παρέχει πρόσβαση σε μη ενσωματομένους πόρους, όπως συμβολοσειρές, χρώματα και διεπαφή χρήστη.
- Notifications Manager. Επιτρέπει στις εφαρμογές την εμφάνιση ειδοποιήσεων στο χρήστη.
- View System. Ένα επεκτάσιμο σύνολο προβολών που χρησιμοποιούνται για τη δημιουργία διεπαφών με το χρήστη.

1.4.5) Applications

Το τελευταίο επίπεδο όπου βρίσκεται η εφαρμογή. Το μόνο απαραίτητο που χρειάζεται είναι η αίτηση για εγκατάσταση. Παραδείγματα τέτοιων εφαρμογών είναι τα παιχνίδια, προγράμματα περιήγησης κτλπ.

1.5) Κύκλος ζωής δραστηριότητας

Μία δραστηριότητα είναι η αρχική οθόνη στο Android. Με τη βοήθεια της δραστηριότητας μπορείτε να τοποθετήσετε όλες τις λειτουργίες και τα γραφικά στοιχεία της διεπαφής του χρήστη σε μία οθόνη. Υπάρχουν 7 μεθόδοι που περιγράφουν πως συμπεριφέρεται η δραστηριότητα σε διαφορετικές καταστάσεις.

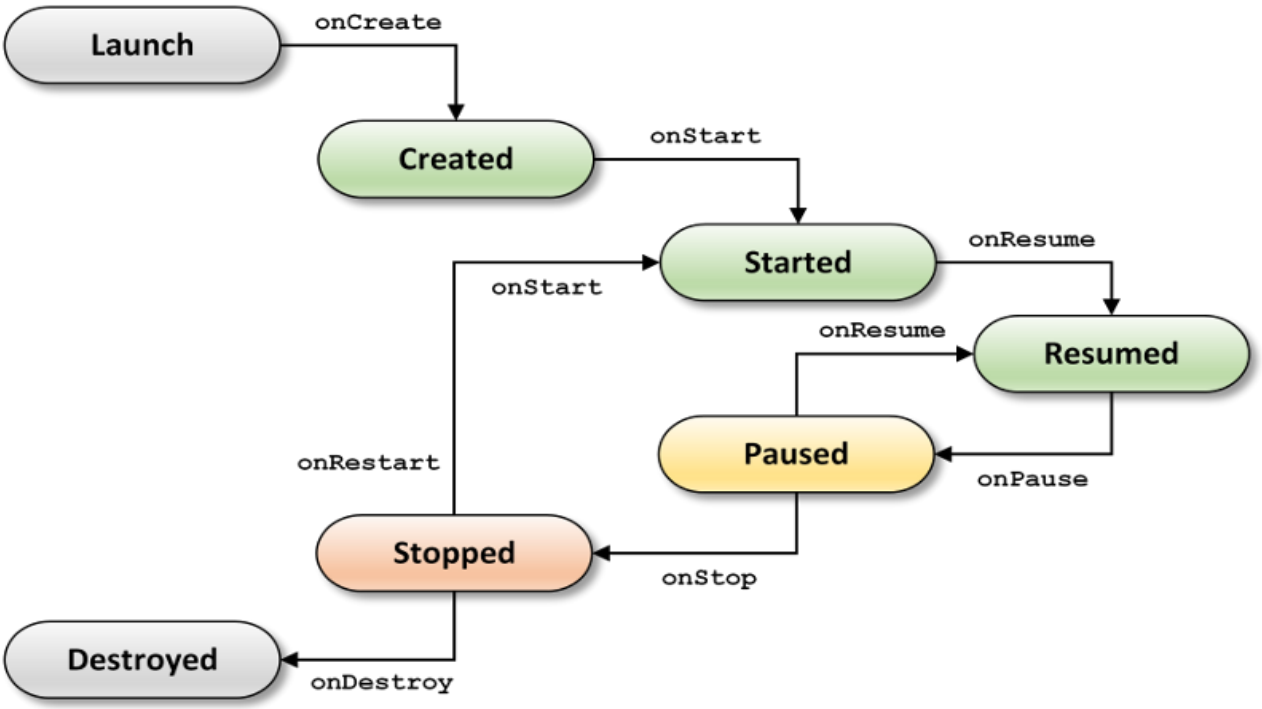
- onCreate() – Καλείται όταν δημιουργείται η δραστηριότητα.
- onStart() – Καλείται οταν η δραστηριότητα εμφανίζεται στο χρήστη.
- onResume() – Καλείται όταν η δραστηριότητα αρχίζει να αλληλεπιδρά με τον χρήστη.

- onPause() – Καλείται όταν η δραστηριότητα δεν είναι εμφανής στονχρήστη.

- onStop() – Καλείται όταν η δραστηριότητα δεν είναι πλέον ορατή στο χρήστη.

- onRestart() – Καλείται αφού η δραστηριότητα έχει σταματήσει, για να ξεκινήσει πάλι.

- onDestroy() – Καλείται πριν καταστραφεί η δραστηριότητα.



Εικόνα 3 : Android lifecycle

2.ΠΑΡΟΜΟΙΕΣ ΕΦΑΡΜΟΓΕΣ

Σε αυτήν την ενότητα θα παρουσιαστούν πανομοιότυπες εφαρμογές που διανέμονται δωρέαν στα κινητά Android. Μπορέι κανείς να τις κατεβάσει απο την

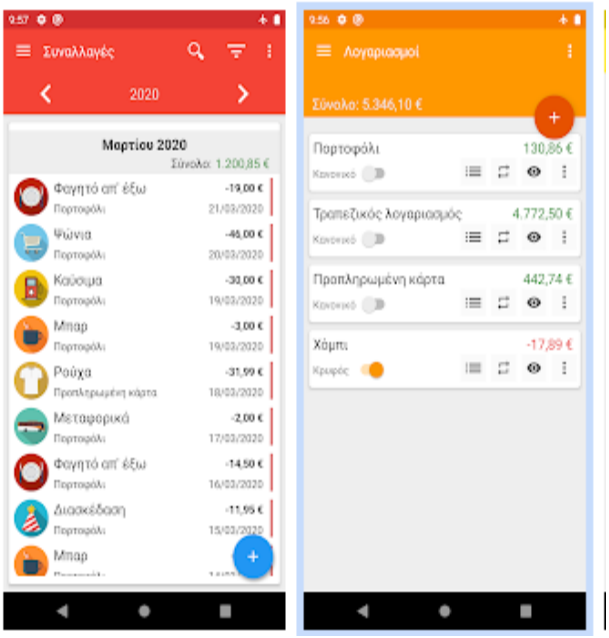
πλατφόρμα Google Play. Στόχος τους είναι η εύκολη διαχείριση οικονομικών του χρήστη.

2.1) Γρήγορος προϋπολογισμός- Διαχειριστής εξόδων

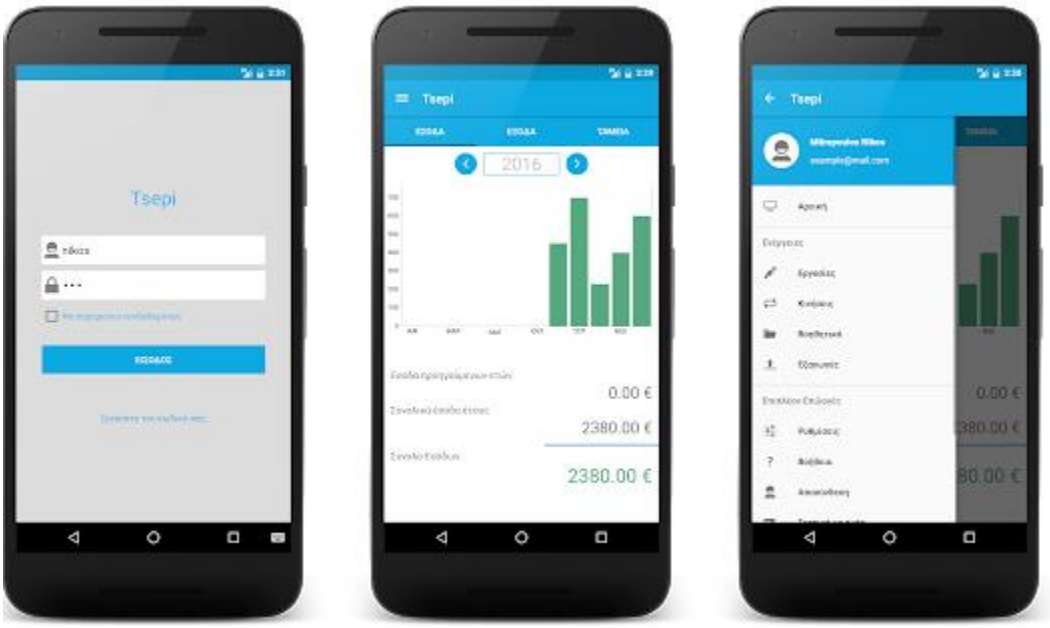
Εικόνα 3: Διαχειριστής εξόδων

Μία εφαρμογή για την γρήγορη αποθήκευση προσωπικών ή οικογενειακών οικονομικών δεδομένων. Διαθέτει χρήσιμα εργαλεία όπως διαγράμματα και ημερολόγιο. Επίσης, υπάρχει δυνατότητα εισαγωγής πιστωτικής ή χρεωστικής κάρτας για την επίβλεψη εικονικών συναλλαγών.

2.2) Tsepi (Έσοδα-έξοδα-πελάτες-ταμεία)

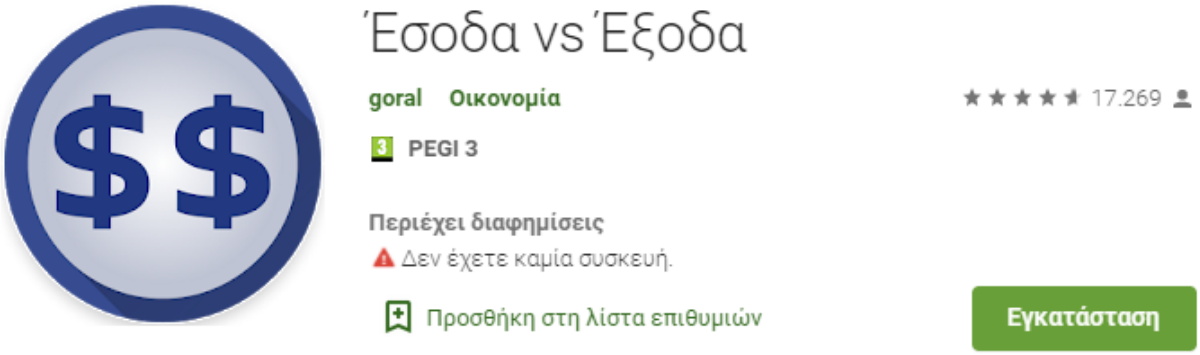


Εικόνα 4: Tsepi

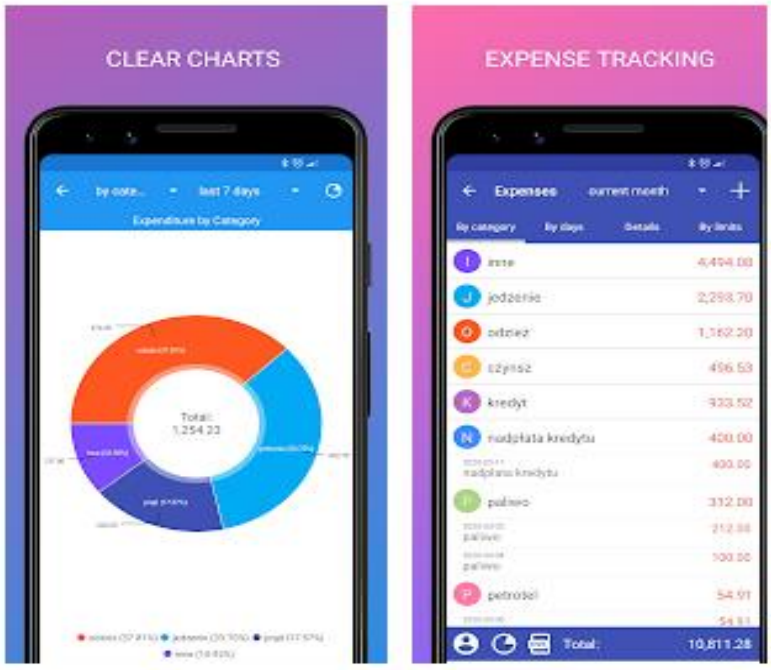


Η εφαρμογή Tsepi είναι ένα πολύγραμμα που μπορεί να εξυπηρετήσει τον χρήστη για εξωλογιστική παρακολούθηση κάποιων εργασιών χωρίς ενοχλητικές διαφημίσεις.

2.3) Έσοδα vs Έξοδα

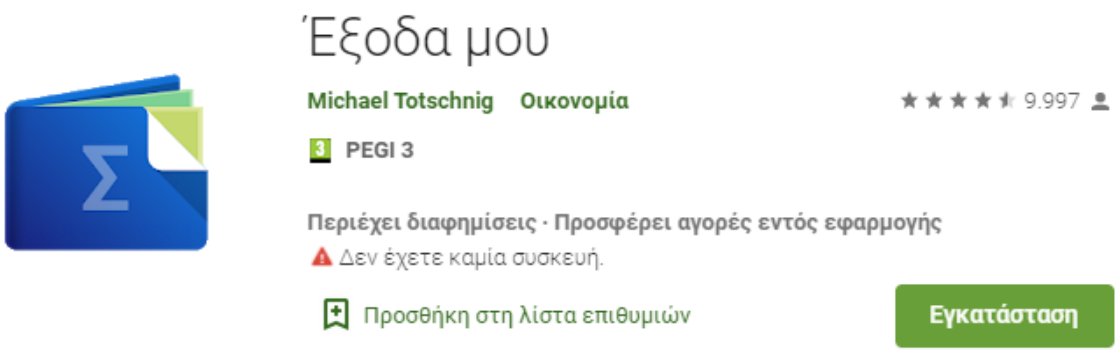


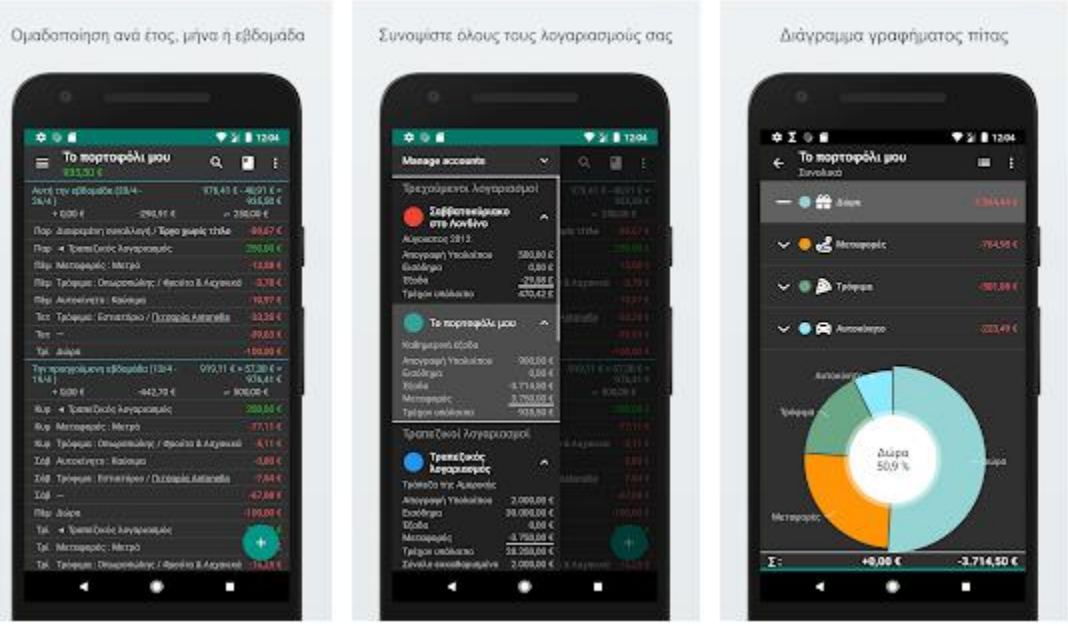
Εικόνα 4: Έσοδα vs Έξοδα



Η εφαρμογή έχει ως στόχο τον έλεγχο των οικονομικών. Επιτρέπει στον χρήστη να αποθηκεύσει τα καθημερινά έσοδα και έξοδα σε διάφορες κατηγορίες και να καθορίσει εάν είναι σταθερό ή μεταβλητό κόστος. Υπάρχει δυνατότητα προβολής εσόδων και δαπανών σε σχέση με τις κατηγορίες, τις ημέρες και τις μελλοντικές δαπάνες. Backup δεδομένων ασφαλή απί πιθανή απώλεια, ειδοποιήσεις για την υπενθύμιση πληρωμής ληξιπρόθεσμων οφειλών.

2.4) Έξοδα μου





Μία εφαρμογή για παρακολούθηση και διαχείρηση εσόδων – εξόδων. Διαθέτει δύο επίπεδα κατηγοριών, διαιρεμένη συναλλαγή, προστασία με κωδικό πρόσβασης, ολοκληρωμένη βοήθεια, εξαγωγή δεδομένων τύπου Excel. Υπάρχει επίσης δυνατότητα μελλοντικών πλάνων, επαναλαμβανόμενων συναλλαγών όπως και συγχρονισμό συσκευών μέσω Cloud.

2.5) Προϋπολογισμός

Προϋπολογισμός

SIRI **Οικονομία**

★★★★☆ 23.989

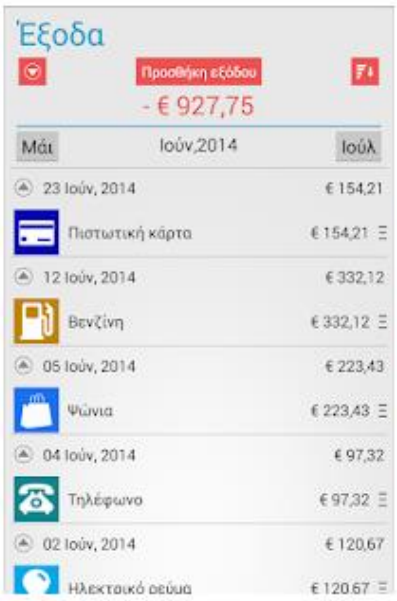
PEGI 3

Προσφέρει αγορές εντός εφαρμογής

Δεν έχετε καμία συσκευή.

Προσθήκη στη λίστα επιθυμιών

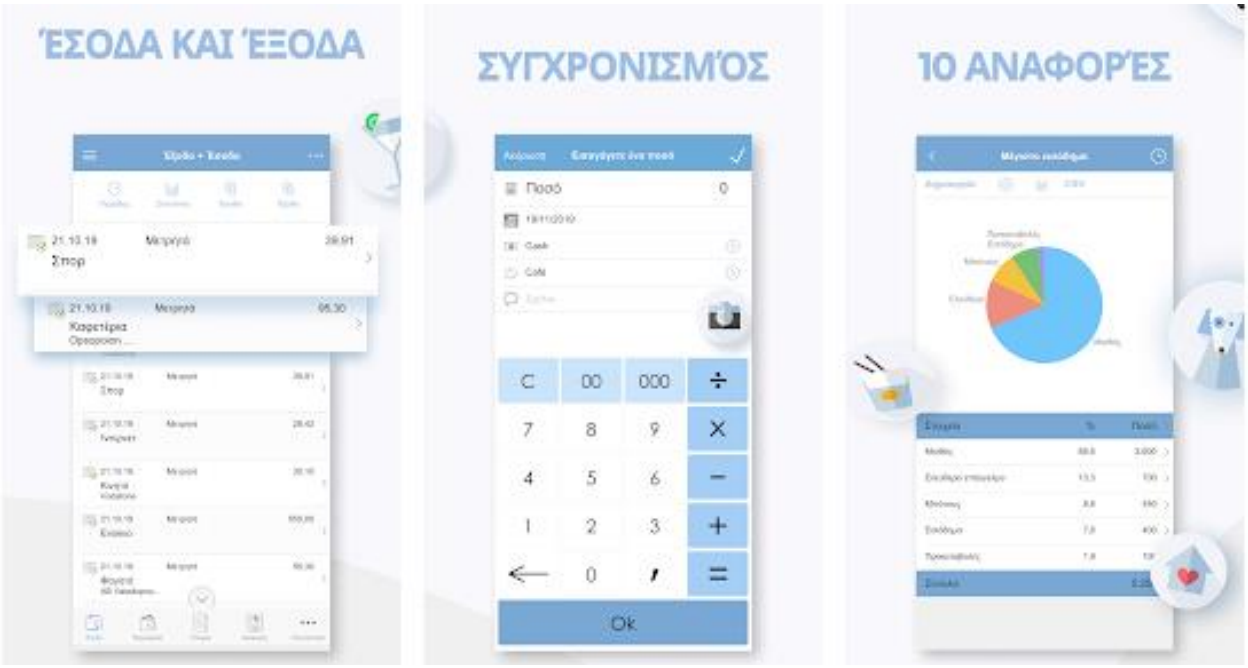
Εγκατάσταση



Σε αυτήν την εφαρμογή μπορείτε να δείτε το σύνολο των λογαριασμών. Έλεγχος συναλλαγών απο τον λογαριασμό, ρύθμιση επανάληψης εσόδων – εξόδων, απεριόριστοι λογαριασμοί, υπενθύμιση γραμμάτιων, προστασία με κωδικό πρόσβασης, έλεγχος φόρων που χρησιμοποιείται στην κατηγορία εξόδων. Τέλος, θα βρείτε στατιστικά εσόδων και δαπανών καθώς και εξαγωγή δεδομένων σε Excel.

Η εφαρμογή είναι επι πληρωμής με 30 μέρες δωρεάν λειτουργίας.

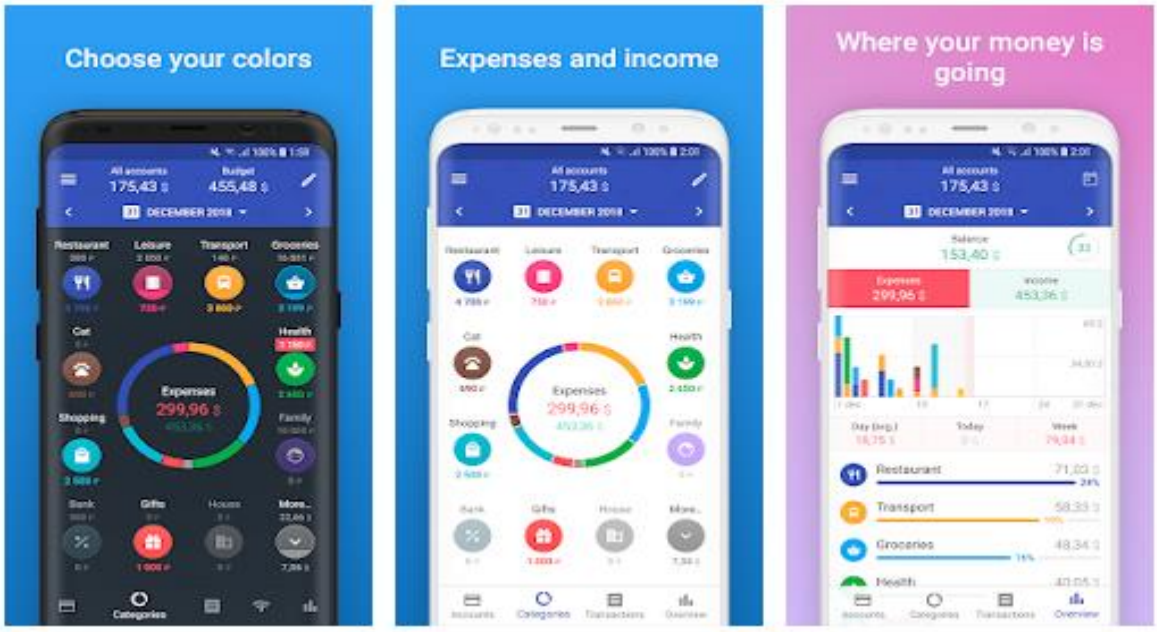
2.6) Έσοδα και Έξοδα



Το πρόγραμμα έσοδα και έξοδα είναι εργαλείο για την καταγραφή και ανάλυση προσωπικών οικονομικών δεδομένων. Διαχειρίζει και αναλύει τις ταμειακές ροές, δείχνει τις μηνιαίες δαπάνες, το ποσό των εσόδων και εξόδων για την τρέχουσα περίοδο καθώς και το διαθέσιμο υπόλοιπο για σπατάλη.

Μία εφαρμογή επί πληρωμής 1.09- 21,99 ευρώ ανα στοιχείο.

2.7) 1Money – Διαχείριση οικονομικών προυπολογισμού



Η εφαρμογή επιτρέπει την άμεση προσθήκη συναλλαγών, εμφάνιση εσόδων, οικονομικό προυπολογισμό, παρακολούθηση χρεών – αποταμιεύσεων και συγχρονισμό συσκευών μέσω του cloud.

Διανέμεται επι πληρωμής 0.50 – 17,99 ευρώ ανά στοιχείο.

3.ΣΧΕΔΙΑΣΗ ΕΦΑΡΜΟΓΗΣ

3.1) Σκοπός πτυχιακής

Ο σκοπός της εργασίας αυτής είναι η ανάπτυξη μιας εφαρμογής για λογισμικό Android η οποία επιτρέπει την εισαγωγή, επεξεργασία οικονομικών δεδομένων ταξινομημένα σε κατηγορίες καθώς και γραφήματα ανα μέρα, μήνα, έτος. Έυκολη διαχείρηση, εύκολη πλοήγηση, χωρίς διαφημίσεις, πλήρως λειτουργική και διανέμεται δωρεάν σε όποιον θέλει να την χρησιμοποιήσει.

Ο κώδικας της εφαρμογής είναι open source και μπορεί ο καθένας να τον κατεβάσει και να τον χρησιμοποιήσει ανάλογα με τις ανάγκες του. Το link για να το βρει κάποιος αναφέρεται στα χρήσιμα Links του εγγράφου.

3.2) Απαιτήσεις - Λειτουργικότητα

Η εφαρμογή θα πρέπει να είναι σε θέση να εκτελεί ένα πλήθος λειτουργιών εισαγωγής – επεξεργασίας καθώς και να παρουσιάζει στατιστικά των δεδομένων. Οι απαιτήσεις είναι οι εξής:

1. *Εισαγωγή ποσού εσόδου – εξόδου*. Ο χρήστης θα μπορεί να εισάγει το ποσό που επιθυμεί είτε σαν έσοδο είτε σαν έξοδο στο αντίστοιχο πεδίο.

2. *Επιλογή ημερομηνίας καταχώρησης ποσού*. Ο χρήστης θα μπορεί να επιλέγει την ημερομηνία που έγινε η συναλλαγή μέσω ενός ημερολογίου.

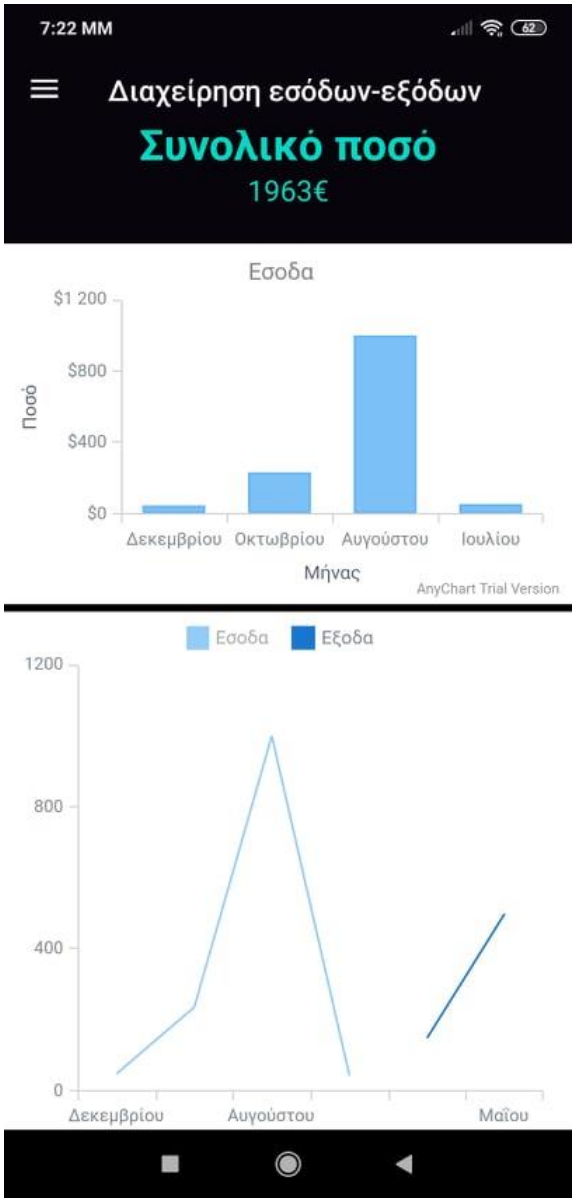
3. *Επιλογή κατηγορίας ποσού(μισθός, τράπεζα, χαρτζιλίκι κτλπ)*. Ο χρήστης μπορεί να επιλέξει μια απο τις υπάρχοντες κατηγορίες, ταξινομώντας τα ποσά του έτσι ώστε να γνωρίζει που δαπανήθηκαν ή από που προήλθαν.

4. *Εμφάνιση διαγράμματος ανάλογο με τις ακριβείς συναλλαγές του χρήστη*. Ο χρήστης θα έχει την δυνατότητα να δει τα ποσά που καταχώρησε και την ημερομηνία τους μέσω ενός interactive διάγραμμα τύπου πίτας.

5. *Προβολή ιστορικού συναλλαγών*. Ο χρήστης μπορεί να δει σε μια νέα καρτέλα τις συναλλαγές που πραγματοποίησε αναλυτικά. Ποσό, ημερομηνία, κατηγορία.

6. *Στατιστικο διάγραμμα ροής εσόδων εξόδων*. Ενα ευκρινές μηνιαίο διάγραμμα για τα έσοδα και έξοδα του, καθώς και ένα στατιστικό μηνιαίο διάγραμμα για την κίνηση των εσόδων – εξόδων ταυτόχρονα.

7. *Συνολικό ποσό που απομένει*. Ο χρήστης έχει την δυνατότητα να δεί το συνολικό ποσό που του απομένει.



3.3) Δεδομένα

Για την ικανοποίηση του συστήματος η εφαρμογή θα πρέπει να αποθηκεύει τα δεδομένα. Για τον σκοπό αυτό θα χρησιμοποιηθεί μια βάση δεδομένων SQLite, ευρέως γνωστή και η πιο συνηθής για κινητά Android. Οι βιβλιοθήκες SQLite βρίσκονται ενσωματωμένες στο Android SDK.

Για την αποθήκευση των δεδομένων θα χρειαστούν 2 πίνακες.

Πίνακας εσόδων

- ID Integer auto-increment Primary-key
- DATES Date
- AMOUNT INT
- CATEGORY VARCHAR

Table: income_table

	ID	DATES	AMOUNT	CATEGORY
	Filter	Filter	Filter	Filter
1	1	08-04-2020	121	Salary
2	2	17-9-2020	1000	Salary
3	3	30-10-2020	5000	Salary
4	4	31-05-2020	30	Bonus
5	5	31-05-2020	30	Bonus
6	6	31-05-2020	30	Bonus
7	7	31-05-2020	30	Bonus
8	8	31-05-2020	30	Salary

Πίνακας εξόδων

- ID Integer auto-increment Primary-key

- DATES Date
- AMOUNT INT
- CATEGORY VARCHAR

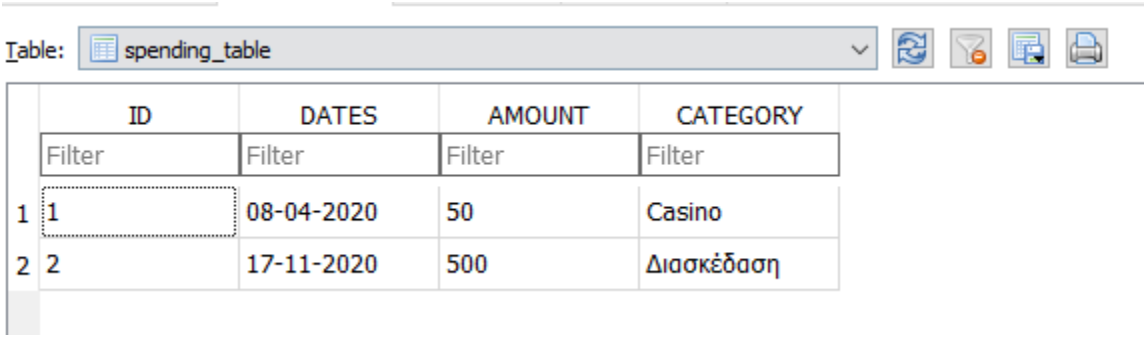
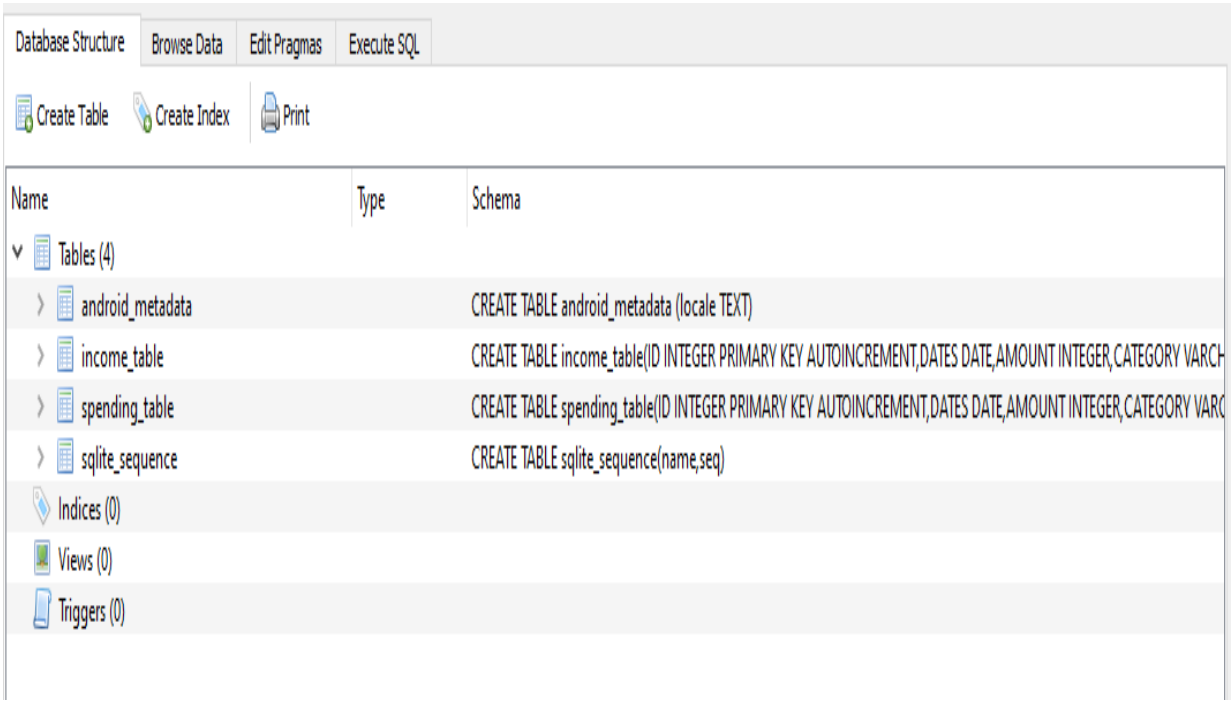


Table: spending_table

	ID	DATES	AMOUNT	CATEGORY
	Filter	Filter	Filter	Filter
1	1	08-04-2020	50	Casino
2	2	17-11-2020	500	Διασκέδαση

Δομή βάσης δεδομένων



4.ΥΛΟΠΟΙΗΣΗ

4.1) Γλώσσες προγραμματισμού - Βιβλιοθήκες

4.1.1) Java

Ένας από τους μεγαλύτερους λόγους για τους οποίους η Java είναι τόσο δημοφιλής είναι η ανεξαρτησία της πλατφόρμας. Τα προγράμματα μπορούν να εκτελεστούν σε διάφορους τύπους υπολογιστών, εφόσον διαθέτει Java Runtime Environment (JRE).

Οι περισσότεροι τύποι υπολογιστών είναι συμβατοί με ένα JRE, συμπεριλαμβανομένων υπολογιστών που λειτουργούν με Windows, Macintosh, Unix, Linux καθώς και κινητά τηλέφωνα.

Δεδομένου ότι υπάρχει εδώ και πολύ καιρό, ορισμένοι από τους μεγαλύτερους οργανισμούς στον κόσμο έχουν κατασκευαστεί χρησιμοποιώντας την Java. Για παράδειγμα, πολλές τράπεζες, έμποροι λιανικής, ασφαλιστικές εταιρείες, επιχειρήσεις κοινής ωφέλειας και κατασκευαστές χρησιμοποιούν Java.

Θεωρείται μια εξελισσόμενη γλώσσα που συνδυάζει σχεδόν μοναδικά τη σταθερότητα με την καινοτομία. Ακόμη, ο κώδικας που είναι γραμμένος 15 χρόνια πριν, θα μπορεί να εκτελείται στα πιο ενημερωμένα JVM και θα αποκτήσει το πλεονέκτημα ταχύτητας, τη μετάφραση εγγενή κώδικα και τη διαχείριση μνήμης.

Η Java είναι γλώσσα object-oriented. Ο κώδικας είναι τόσο ισχυρός επειδή τα αντικείμενα Java δεν περιέχουν αναφορές σε δεδομένα που είναι εξωτερικά. Θεωρείται απλή σαν γλώσσα, ωστόσο, συνοδεύεται από μια βιβλιοθήκη κλάσεων που προσφέρουν κοινά χρησιμοποιούμενες λειτουργίες, χωρις τις οποίες δεν μπορούν να λειτουργήσουν τα περισσότερα προγράμματα Java.

Το Java API, η βιβλιοθήκη κλάσης, είναι τόσο μέρος της Java όσο και η ίδια η γλώσσα. Στην πραγματικότητα η πρόκληση είναι η εκμάθηση του API και όχι η γλώσσα. Η γλώσσα αποτελείται από 50 λέξεις – κλειδιά, αλλά το API περιέχει χιλιάδες κλάσεις με δεκάδες χιλιάδες μεθόδους για να χρησιμοποιηθούν στην ανάπτυξη προγραμμάτων. Παρ ‘όλα αυτά, οι προγραμματιστές δεν αναμένεται να μάθουν όλο το Java API.

Η Java χρησιμοποιείται για:

- *Ανάπτυξη εφαρμογών Android*. Οι περισσότερες εφαρμογές Android είναι γραμμένες σε Java με την χρήση του Google Android API.

- *Ανάπτυξη επιστημονικών εφαρμογών*. Η Java είναι συχνά η προεπιλεγμένη γλώσσα για επιστημονικές εφαρμογές. Είναι ασφαλής, φορητή, συντηρήσιμη και διαθέτει καλύτερα εργαλεία υψηλού επιπέδου από οποιαδήποτε άλλη γλώσσα.

- *Ανάπτυξη web εφαρμογών*. Υπάρχουν πολλές και σημαντικές web εφαρμογές στην καθημερινότητα μας. Ενα σημαντικό παράδειγμα είναι το Gmail της Google.

- *Ανάπτυξη εργαλείων λογισμικού*. Πολλά λογισμικά και εργαλεία ανάπτυξης λογισμικού είναι γραμμένα σε Java. Μερικά από αυτά, Intellij IDEA, Eclipse, NetBeans IDE.

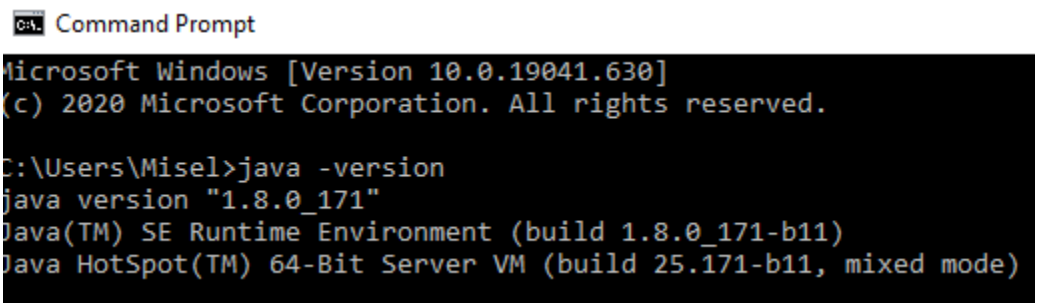
Ο λόγος που πολλοί επιλέγουν Java είναι επειδή είναι εύκολο να γραφτεί και να εκτελεστεί. Μπορεί να εκτελεστεί σχεδόν οπουδήποτε και ανα πάσα στιγμή. Έχει την δυνατότητα δημιουργίας ολοκληρωμένων εφαρμογών που μπορούν να εκτελεστούν σε έναν μόνο υπολογιστή ή να διανεμηθούν σε διακομιστές και πελάτες σε ένα δίκτυο.

Διαθέτει συστήματα GUI(Γραφικό περιβάλλον εργασίας χρήστη), τρόπους επικοινωνίας – σύνδεσης σε βάσεις δεδομένων μέσω του JDBC(Java Database Connectivity).

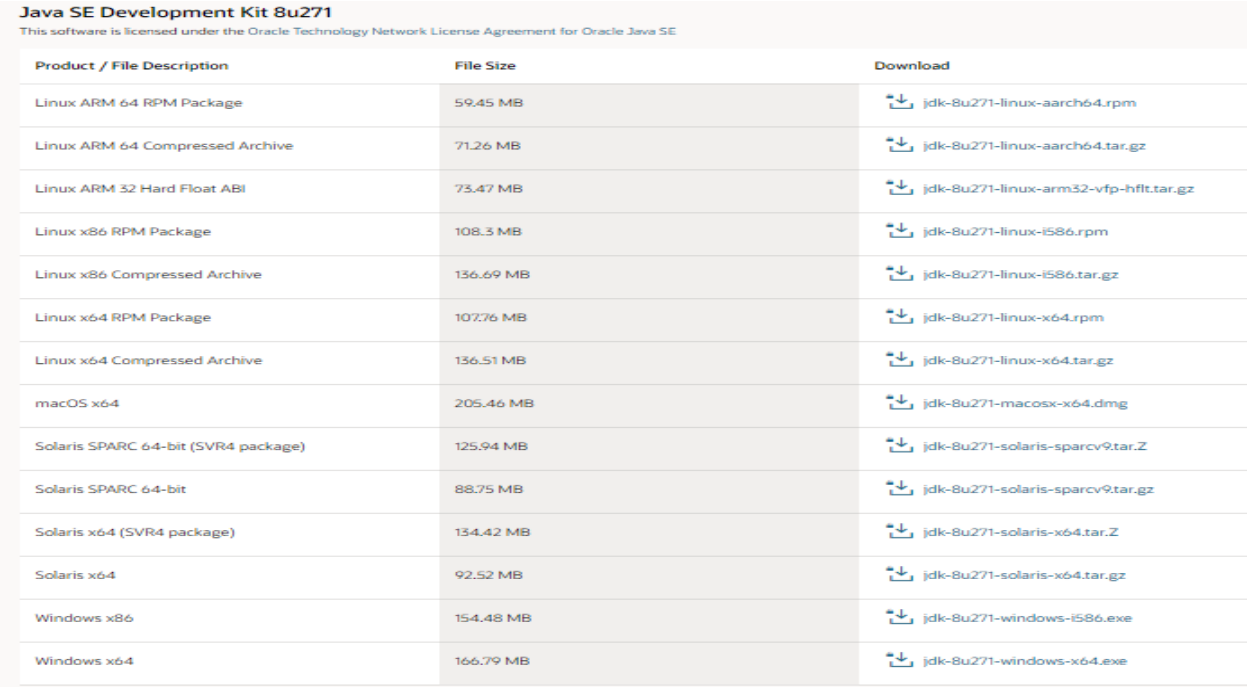
Τέλος, η Java συνεχίζει να δημιουργεί πολλές θέσεις εργασίας στον κλάδο της τεχνολογίας. Το OpenJDK είναι μια δωρεάν και ανοιχτή πηγή υλοποίησης της γλώσσας προγραμματισμού Java. Δεδομένου ότι είναι μια γλώσσα προγραμματισμού ανοιχτού κώδικα, είναι ανεξάρτητη πλατφόρμα σε περιβάλλον λειτουργικού συστήματος και μπορεί να χρησιμοποιηθεί παντου.

Εγκατάσταση Java

Η γλώσσα προγραμματισμού που χρησιμοποιήθηκε για την ανάπτυξη της εφαρμογής είναι η Java. Έκδοση της γλώσσας Java είναι 1.8.0_171 γνωστή και ως Java 8.



Για να εγκαταστήσει κάποιος την Java μπορεί να περιηγηθεί στο επίσημο site της Oracle https://www.oracle.com/java/technologies/javase/javase-jdk8-downloads.html και να κατεβάσει το jdk που αντιστοιχεί στο λειτουργικό του σύστημα.



Java SE Development Kit 8u271

This software is licensed under the Oracle Technology Network License Agreement for Oracle Java SE

Product / File Description	File Size	Download
Linux ARM 64 RPM Package	59.45 MB	jdk-8u271-linux-aarch64.rpm
Linux ARM 64 Compressed Archive	71.26 MB	jdk-8u271-linux-aarch64.tar.gz
Linux ARM 32 Hard Float ABI	73.47 MB	jdk-8u271-linux-arm32-vfp-hflt.tar.gz
Linux x86 RPM Package	108.3 MB	jdk-8u271-linux-i586.rpm
Linux x86 Compressed Archive	136.69 MB	jdk-8u271-linux-i586.tar.gz
Linux x64 RPM Package	107.76 MB	jdk-8u271-linux-x64.rpm
Linux x64 Compressed Archive	136.51 MB	jdk-8u271-linux-x64.tar.gz
macOS x64	205.46 MB	jdk-8u271-macosx-x64.dmg
Solaris SPARC 64-bit (SVR4 package)	125.94 MB	jdk-8u271-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	88.75 MB	jdk-8u271-solaris-sparcv9.tar.gz
Solaris x64 (SVR4 package)	134.42 MB	jdk-8u271-solaris-x64.tar.Z
Solaris x64	92.52 MB	jdk-8u271-solaris-x64.tar.gz
Windows x86	154.48 MB	jdk-8u271-windows-i586.exe
Windows x64	166.79 MB	jdk-8u271-windows-x64.exe

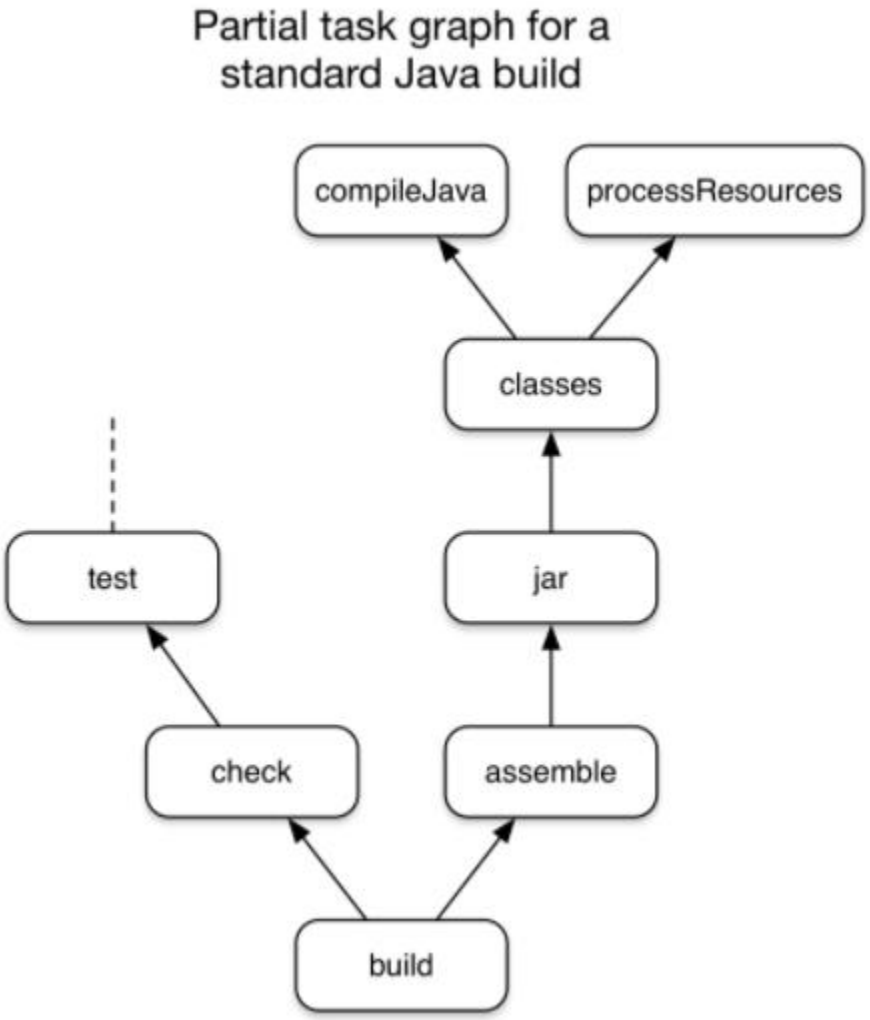
4.1.2) Gradle - build system

Το βασικό εργαλείο που χρησιμοποιήθηκε για την κατασκευή του κώδικα είναι το Gradle.Ένα εργαλείο αυτοματοποίησης ανοιχτού κώδικα που έχει σχεδιαστεί αρκετά ευέλικτο ώστε να δημιουργεί οποιοδήποτε τύπο λογισμικού. Ελέγχει την διαδικασία ανάπτυξης του κώδικα, τα συλλέγει για testing και τα δημοσιεύει.

Το Gradle εκτελείται στο JVM προϋποθέτοντας ότι είναι εγκατεστημένο το κατάλληλο Java Development Kit(JDK). Διαθέτει για χρήση το βασικό Java API στη λογική κατασκευής, όπως προσαρμοσμένους τύπους εργασιών και προσθήκες(plugins). Είναι σχετικά μια νέα λύση διαχείρισης πακέτων που έχει αναπτυχθεί τα τελευταία χρόνια για ταχύτερους χρόνους κατασκευής.

Επίσης, διευκολύνει την κατασκευή κοινών τύπων έργων(βιβλιοθήκες Java) προσθέτοντας ένα επίπεδο συμβάσεων και προκαθορισμένης λειτουργικότητας μέσω προσθηκών. Ακόμη, μπορεί κάποιος να δημιουργήσει τα δικά του plugins να τα προσθέσει στο πρόγραμμα και να τα δημοσιεύσει.

Το Gradle μοντελοποιεί τις κατασκευές του ως Directed Acyclic Graphs(DAGs)(μονάδες εργασίας). Αυτό σημαίνει ότι ένα build διαμορφώνει ένα σύνολο εργασιών και τα ενώνει μαζί, για να δημιουργήσει το DAG. Μόλις δημιουργηθεί το γράφημα εργασιών, το Gradle καθορίζει ποιές εργασίες πρέπει να εκτελεστούν, με ποιά σειρά και στη συνέχεια προχωρά στην εκτέλεση τους.



Το Gradle αξιολογεί και εκτελεί τα scripts σε 3 φάσεις:

1. *Αρχικοποίηση.* Ρυθμίζει το περιβάλλον για την κατασκευή και καθορίζει ποιά έργα θα λάβουν μέρος σε αυτό.

2. *Διαμόρφωση.* Κατασκευάζει και διαμορφώνει το γράφημα εργασιών για το build, καθορίζει ποιές εργασίες πρέπει να εκτελεστούν και με ποιά σειρά.

3. *Εκτέλεση.* Εκτελεί τις εργασίες που έχουν επιλεγεί στο τέλος της διαμόρφωσης

Εγκατάσταση του Gradle

Εάν θέλει κάποιος να εκτελέσει μια υπάρχουσα έκδοση Gradle και το build διαθέτει Gradle Wrapper, τότε δεν χρειάζεται να γίνει εγκατάσταση. Απλά πρέπει το σύστημα να ικανοποιεί τις προυποθέσεις του Gradle. Το Android Studio διαθέτει μια λειτουργική εγκατάσταση του Gradle.

Για να δημιουργήσει κάποιος μια νέα έκδοση, θα πρέπει να εγκαταστήσει το Gradle. Προϋποθέτει την εγκατάσταση του Java Development Kit που δείξαμε παραπάνω και στην συνέχεια λήψη και εγκατάστση της τελευταίας έκδοσης Gradle. Το τελευταίο βήμα που απομένει είναι η δημιουργία καταλόγου C:\ Gradle και η μετακίνηση του περιεχομένου zip με την έκδοση Gradle που κατέβηκε προηγουμένως.

Τις οδηγίες εγκατάστασης καθώς και όλες τις πληροφορίες σχετικά με το Gradle μπορεί να τις βρει κάποιος μπαίνοντας στο εξής σύνδεσμο https://docs.gradle.org/.

4.1.3) SQLite - database

Η SQLite είναι μια βιβλιοθήκη γλώσσας C που περιέχει μια μικρή, γρήγορη, αυτόνομη, υψηλής αξιοπιστίας μηχανή βάσης δεδομένων. Η πιο διάσημη βάση δεδομένων στον κόσμο. Είναι ενσωματωμένη σε όλα τα κινητά τηλέφωνα και τους περισσότερους υπολογιστές και συνοδεύεται απο αμέτρητες εφαρμογές που χρησιμοποιούνται καθημερινά.

Μια συμπαγής βιβλιοθήκη με μέγιστο μέγεθος 600 KB. Όσο περισσότερη μνήμη υπάρχει τόσο γρηγορότερη είναι η εκτέλεση της. Μία αξιόπιστη, πολύ προσεγμένη βάση με ανοιχτό κώδικα του οποίου το μεγαλύτερο μέρος προορίζεται αποκλειστικά για δοκιμές και επαλήθευση.

Ο πηγαίος κώδικας SQLite βρίσκεται στο public domain και είναι δωρεάν σε όλους για χρήση για οποιονδήποτε σκοπό.

Γιατι SQLite:

Οι κύριοι λόγοι που προτιμήθηκε η SQLite στην ανάπτυξη της εφαρμογής μας είναι

- Η ταχύτητα που προσφέρει.

- Η αξιοπιστία.
- Απλή και έυκολη στη χρήση.

Εγκατάσταση SQLite

Για να εγκαταστήσει κάποιος την SQLite αρκεί να μπεί στο site https://www.sqlite.org να κατεβάσει το zip αρχείο ανάλογα με το λειτουργικό σύστημα που διαθέτει, να εκτελέσει το αρχείο που βρίσκεται στο zip φάκελο και στην συνέχεια να εκτελέσει στο command prompt τις παρακάτω εντολές:

1. *C:\cd c:\sqlite*
2. *C:\sqlite>*

4.1.3) Android studio

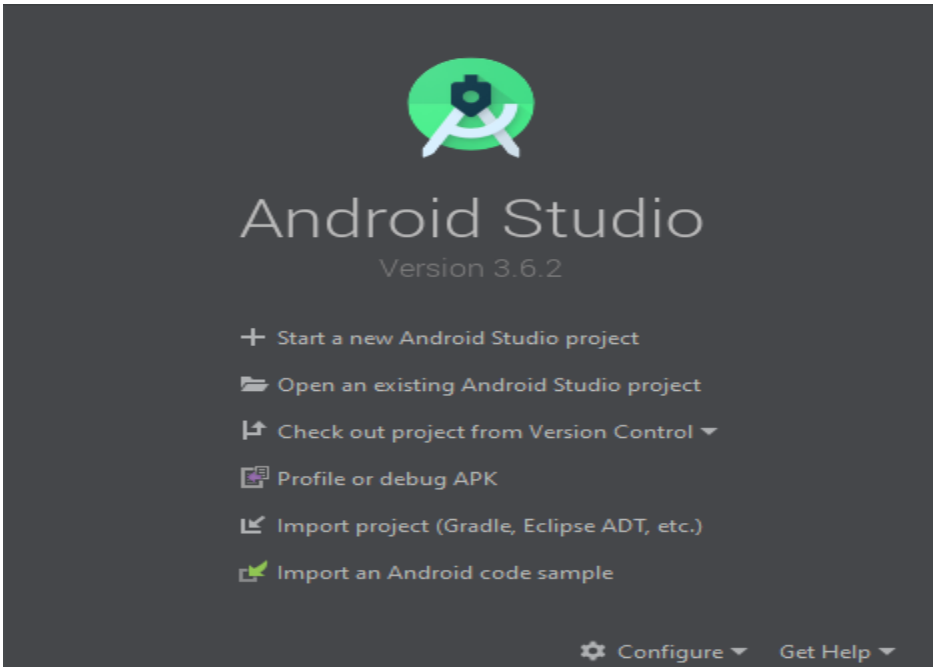
Το περιβάλλον που χρησιμοποιήθηκε για την ανάπτυξη κώδικα της εφαρμογής είναι το Android studio. Παρέχει γρήγορα εργαλεία για σχεδίαση και ανάπτυξη κάθε τύπου Android συσκευής.

Εγκατάσταση Android studio

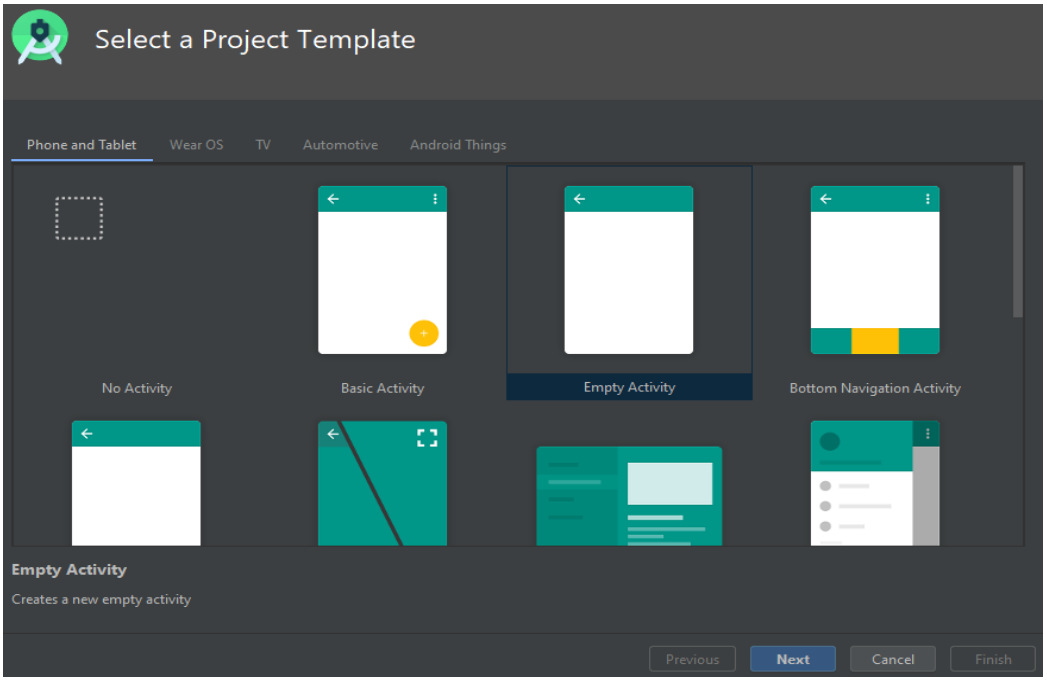
Για την εγκατασταση του μπορεί κάποιος να κατεβάσει το Android studio από το επίσημο site της android.

Το σχετικό link είναι το εξής: https://developer.android.com/studio. Αφού έχει εγκατασταθεί το περιβάλλον μας, ανοίγουμε το Android studio και εκτελούμε τα ακόλουθα βήματα.

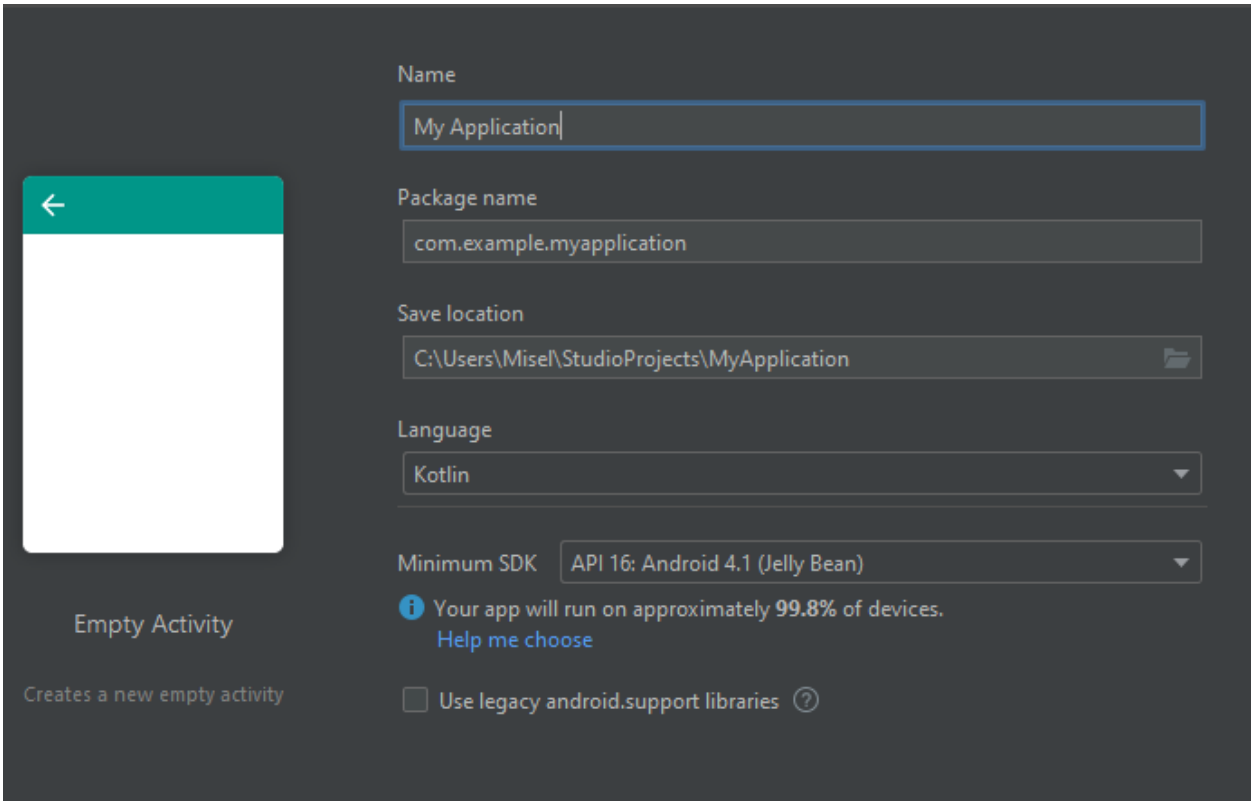
1. Ξεκινάμε ένα νέο Android project.



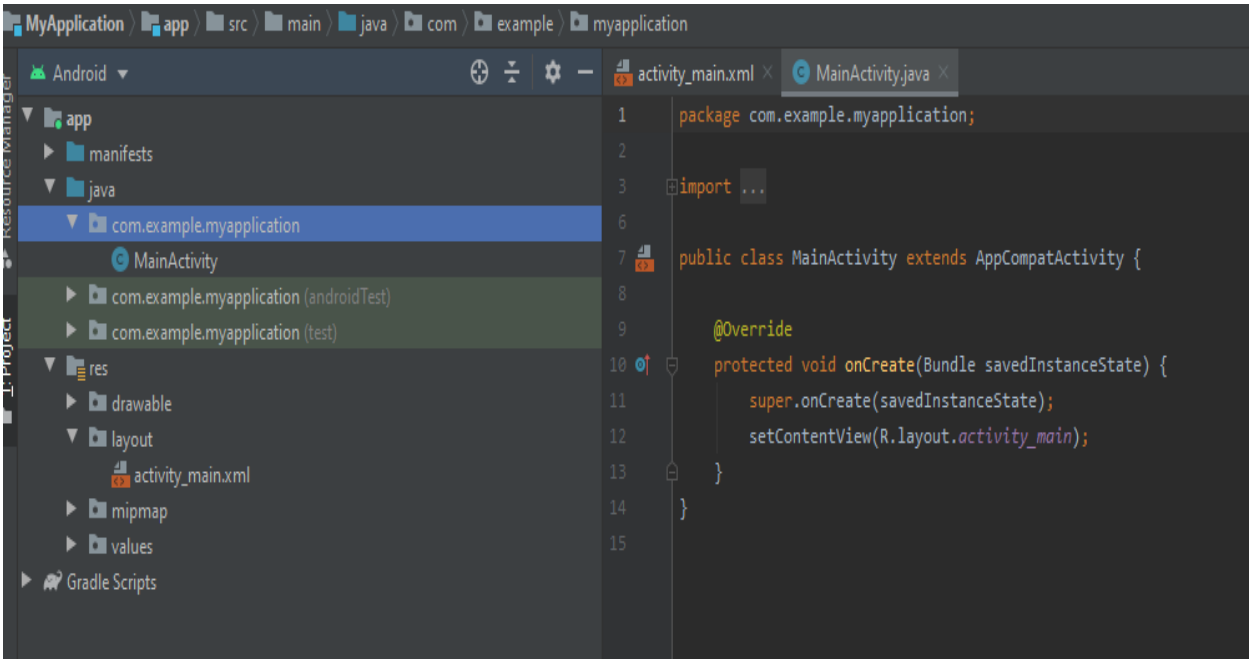
2. Επιλέγουμε μια άδεια δραστηριότητα(Empty Activity).

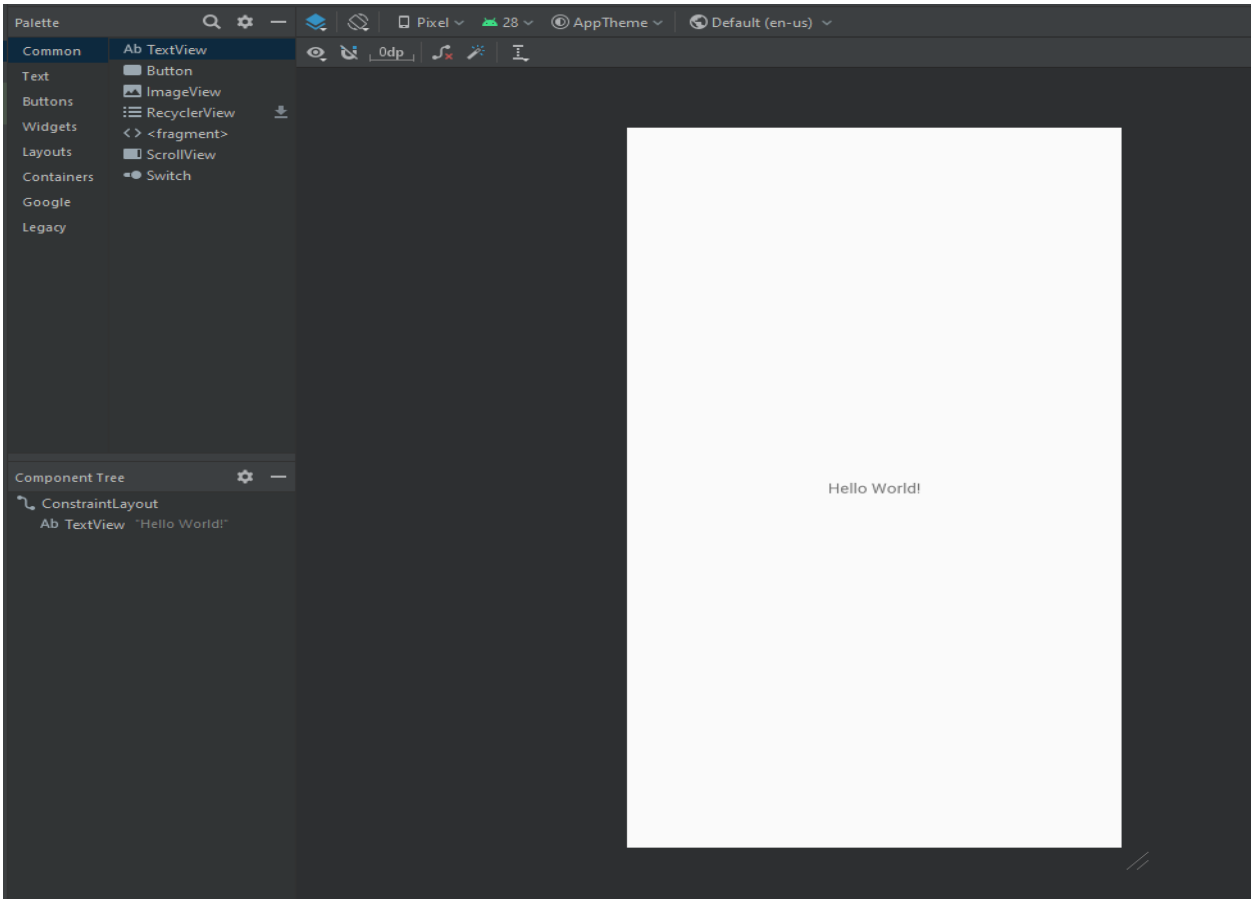


3. Εισάγουμε το όνομα του project, την γλώσσα(Java, Kotlin) και την τοποθεσία αποθήκευσης.



4. Το project έχει δημιουργηθεί και μπορούμε να ξεκινήσουμε την ανάπτυξη κωδικα.





4.2) Περιγραφή κώδικα

Στην ενότητα αυτή περιγράφεται η υλοποίηση των λειτουργιών της εφαρμογής. Μερικές από τις βασικότερες λειτουργίες είναι οι παρακάτω.

- Εισαγωγή ποσού εσόδων – εξόδων.
- Επιλογή ημερομηνίας, κατηγορίας
- Προβολή ιστορικού συναλλαγών.
- Προβολή διαγραμμάτων πίτας, στήλης, γραμμής.
- Διαγραφή, προβολή συναλλαγών.

4.2.1) Εισαγωγή βάσης δεδομένων

Δημιουργία κλάσης DatabaseHelper υπεύθυνη για την δημιουργία της βάσης δεδομένων, των πινάκων και όλες τις λειτουργίες, επεξεργασία δεδομένων που εκτελούνται μέσα στην βάση.

```
public class DatabaseHelper extends SQLiteOpenHelper {
    public static final String DATABASE_NAME = "money.db";
    public static final String TABLE_NAME = "income_table";
    public static final String TABLE2_NAME = "spending_table";
    public static final String COL_1 = "ID";
    public static final String COL_2 = "DATES";
    public static final String COL_3 = "AMOUNT";
    public static final String COL_4 = "CATEGORY";
    public static final String S_COL_1 = "ID";
    public static final String S_COL_2 = "DATES";
    public static final String S_COL_3 = "AMOUNT";
    public static final String S_COL_4 = "CATEGORY";

    public DatabaseHelper( Context context) {
        super(context, DATABASE_NAME,  factory: null, version: 1);

    }
    @Override
    public void onCreate(SQLiteDatabase db) {
        db.execSQL(" create table " + TABLE_NAME + "(ID INTEGER PRIMARY KEY AUTOINCREMENT,DATES DATE,AMOUNT INTEGER,CATEGORY VARCHAR)"  );
        db.execSQL(" create table " + TABLE2_NAME + "(ID INTEGER PRIMARY KEY AUTOINCREMENT,DATES DATE,AMOUNT INTEGER,CATEGORY VARCHAR)" );

    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        db.execSQL(" DROP TABLE IF EXISTS "+TABLE_NAME);
        db.execSQL(" DROP TABLE IF EXISTS "+TABLE2_NAME);
        onCreate(db);
    }
```

Εικόνα: DatabaseHelper

-Σε static strings καταχωρούνται τα ονόματα της βάσης, των δύο πινάκων και των πεδίων του κάθε πίνακα.

-Στην συνέχεια εκτελείται η onCreate μέθοδος, αρχικοποιούνται οι πίνακες, τα πεδία και οι τύποι πεδίων.

-Η onUpgrade μέθοδος ουσιαστικά σβήνει και αντικαθιστά τους παλιούς πίνακες εφόσον υπάρχουν.

4.2.2) Μενού περιήγησης

Δημιουργία ενός μενού περιήγησης από την αρχική οθόνη στις καρτέλες εισόδου, εξόδου, διαγράμματος, πληροφοριών.

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    if(myToggle.onOptionsItemSelected(item))
    {
        return true;
    }
    return super.onOptionsItemSelected(item);
}

@Override
public boolean onNavigationItemSelected(@NonNull MenuItem menuItem) {
    switch (menuItem.getItemId())
    {
        case R.id.nav_income:{
            Intent incomeIntent = new Intent( packageContext: MainActivity.this,IncomeActivity.class);
            startActivity(incomeIntent);
            break;
        }
        case R.id.nav_outgoing:{
            Intent outIntent = new Intent( packageContext: MainActivity.this,ExpenditureActivity.class);
            startActivity(outIntent);
            break;
        }
        case R.id.nav_info:{
            Intent infoIntent = new Intent( packageContext: MainActivity.this,InfoActivity.class);
            startActivity(infoIntent);
            break;
        }
        case R.id.nav_diagrams: {
            Intent graphIntent = new Intent( packageContext: MainActivity.this, DiagramsActivity.class);
            startActivity(graphIntent);
            break;
        }
    }
    myDrawerLayout.closeDrawer(GravityCompat.START);
    return true;
}
```

Εικόνα: Main Activity Navigation

-Στην πρώτη συνάρτηση εκτελείται η επιλογή του μενού.

-Στην δεύτερη συνάρτηση γίνεται η μετάβαση περιεχομένου στην καρτέλα του επιλέξαμε. Η μετάβαση επιτυγχάνεται με την προϋπάρχουσα κλάση Intent.

```xml
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@+id/nav_income"
        android:icon="@drawable/income_menu"
        android:title="Εσοδα"/>

    <item android:id="@+id/nav_outgoing"
        android:icon="@drawable/expenditure_menu"
        android:title="Εξοδα"/>

    <item android:id="@+id/nav_diagrams"
        android:icon="@drawable/expenditure_menu"
        android:title="Διαγράμματα"/>

    <item android:id="@+id/nav_info"
        android:icon="@drawable/info"
        android:title="App Info"/>

</menu>
```

Εικόνα: Menu layout xml file

-Το αρχείο που περιέχει τον σχεδιασμό και την ύπαρξη του μενού περιήγησης.

4.2.3) Εισαγωγή εσόδων- Income Activity

```
public void AddData() {
    btnAddData.setOnClickListener((v) → {
            boolean isInserted = myDb.insertData(editAmount.getText().toString(),textView.getText().toString(),categoryText.getText().toString());
            if(isInserted = true)
                Toast.makeText( context: IncomeActivity.this, text: "Επιτυχής καταχώρηση",Toast.LENGTH_LONG).show();
            else
                Toast.makeText( context: IncomeActivity.this, text: "Απέτυχε",Toast.LENGTH_LONG).show();
    });

}
```

```
btnAddData = findViewById(R.id.but_addData);
```

-Εισαγωγή ποσού μέσω ενός component editText.

-Εισαγωγή ημερομηνίας, κατηγορίας όπως θα δούμε παρακάτω.

-Όλα τα δεδομένα που δίνει ο χρήστης εισάγονται μέσα στην βάση δεδομένων, καθώς, στην συνάρτηση addData() καλούμε σε μια μεταβλητή την συνάρτηση insertData() από το Database Helper.

```
public boolean insertData(String amount, String date, CharSequence text){

    SQLiteDatabase db = this.getWritableDatabase();
    ContentValues contentValues = new ContentValues();
    contentValues.put(COL_2,date);
    contentValues.put(COL_3,amount);
    contentValues.put(COL_4, (String) text);
    long result = db.insert(TABLE_NAME, nullColumnHack: null,contentValues);
    if(result == -1)
        return false;
    else
        return true;
}
```

-Η insertData() εκτελεί ένα sql ερώτημα τύπου insert και καταχωρεί στον πίνακα, τις τιμές στα πεδία που ορίστηκαν.

Επιλογή ημερομηνίας

```
public void GetDate(){
    SimpleDateFormat sdf = new SimpleDateFormat( pattern: "dd/MM/yyyy");
    String dates = sdf.format(new Date());
    textView.setText(dates);

    btnDate.setOnClickListener((v) -> {
            Calendar calendar = Calendar.getInstance();
            int year = calendar.get(Calendar.YEAR);
            int month = calendar.get(Calendar.MONTH) ;
            int dayOfMonth = calendar.get(Calendar.DAY_OF_MONTH);

            datePickerDialog = new DatePickerDialog( context: IncomeActivity.this,
                    (datePicker, year, month, day) -> {
                            int selectedMonth = month+1;
                            textView.setText(day + "/" + selectedMonth + "/" + year);
                    }, year, month , dayOfMonth);
            datePickerDialog.show();
    });
}
```

-Συνάρτηση getDate(). Χρησιμοποιείται ένα date picker dialog, ανοίγει το ημερολόγιο και επιστρέφει την ημερομηνία στις μεταβλητές year, month, day αντίστοιχα. Στην συνέχεια μπαίνει η ημερομηνία σε ένα text view και αποθηκεύεται στην βάση με την μέθοδο που κλήθηκε παραπάνω.

Επιλογή κατηγορίας

```
ArrayAdapter<CharSequence> adapter = ArrayAdapter.createFromResource( context: this,R.array.Categories, android.R.layout.simple_spinner_item);
adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
catSpinner.setAdapter(adapter);
catSpinner.setOnItemSelectedListener(this);
```

-Για την επιλογή κατηγορίας χρησιμοποιείται ένα spinner. Αρχικοποιείται με την βοήθεια ενός αντάπτορα.

```
@Override
public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {
    String text = parent.getItemAtPosition(position).toString();
    categoryText.setText(text);
}
```

-Στην συνέχεια το μετατρέπει σε string και βάζει την τιμή στο text view(κατηγορία) που έχουμε ορίσει

```
<string-array name="Categories">
    <item>Μισθός</item>
    <item>Μετοχές</item>
    <item>Μπόνους</item>
    <item>Επενδύσεις</item>
    <item>Φιλοδώρημα</item>
    <item>Φόροι</item>
    <item>Δάνειο</item>
    <item>Επίδομα</item>
    <item>Χαρτζιλίκι</item>
</string-array>
```

-Ενας πίνακας με strings περιέχει όλες τις κατηγορίες που μπορεί κάποιος να επιλέξει.

4.2.4) Προβολή ιστορικού εσόδων – IncomeShowData Activity

```
public void ViewData(){
    Cursor res = myDb.getAllData();

    if(res.getCount() == 0){
        ShowMessage( title: "Error", message: "No data");
        return;
    }

        while (res.moveToNext()) {
            amount.add(res.getString( columnIndex: 2));
            date.add(res.getString( columnIndex: 1));
            category.add(res.getString( columnIndex: 3));
        }

        for(int i=0 ; i<amount.size();i++){
            String poso = amount.get(i);
            String imerominia = date.get(i);
            String katigoria = category.get(i);
          appendData.add("Ποσό:"+poso+"\n"+"Ημερομηνία:"+imerominia+"\n"+"Κατηγορία:"+katigoria);
        }
}

public void ShowMessage(String title, String message){
    AlertDialog.Builder builder = new AlertDialog.Builder( context: this);
    builder.setTitle(title);
    builder.setMessage(message);
    builder.show();
}
```

-Η συνάρτηση viewData() χρησιμοποιεί την κλάση cursor για να καταφέρει να συλλέξει απο την βάση δεδομένων, τα δεδομένα ιστορικού. Στην συνέχεια αποθηκεύει σε λίστες τα δεδομένα που συλλέγει απο τα πεδία του πίνακα. Γίνεται προσπέλαση του πίνακα και αποθηκεύεται σε μια λίστα τύπου string, το ιστορικό της κάθε συναλλαγής.

```
public Cursor getAllData(){
    SQLiteDatabase db = this.getWritableDatabase();
    Cursor res = db.rawQuery( sql: "select * FROM " + TABLE_NAME , selectionArgs: null);
    return res;
}
```

-Αντίστοιχα, φαίνεται η συνάρτηση τύπου Cursor getAllData() που κλήθηκε παραπάνω. Βρίσκεται στο Database Helper και εκτελεί ένα sql ερώτημα που επιστρέφει όλα τα στοιχεία του πίνακα.

-Το ιστορικό συναλλαγών εμφανίζεται ομοιόμορφα σε μια λίστα(list view).

Προβολή μεμονομένης συναλλαγής

```
listView.setOnItemClickListener((parent, view, position, id) → {
        String amount1 = parent.getItemAtPosition(position).toString();
        String part3= "";
        if(amount1.contains(":") ){
           String[] parts = amount1.split( regex: ":", limit: 2);
            String part1 = parts[0];
            String part2 = parts[1];
            if(part2.contains("\n")){
                String parts1[] =  part2.split( regex: "\n", limit: 2);
                part3 = parts1[0];
                String part4 = parts1[1];
            }
        }

        Cursor data = myDb.getItemID(part3);
        int itemID = -1;
        while (data.moveToNext()) {
            itemID = data.getInt( columnIndex: 0);
        }
            if(itemID>-1){
                Intent deleteScreen = new Intent( packageContext: IncomeShowDataActivity.this, DeleteItemActivity.class);
                deleteScreen.putExtra( name: "id",itemID);
                deleteScreen.putExtra( name: "amount1",amount1);
                deleteScreen.putExtra( name: "part3",part3);
                startActivity(deleteScreen);
            }
});
```

-Πατώντας σε κάθε στοιχείο της λίστας ιστορικού, εκτελείται ο παραπάνω κώδικας. Γίνεται μια επεξεργασία διαχωρισμού της γραμμής, που αποθηκεύτηκε προηγουμένως στην λίστα, για να ελεγχθεί ότι το ποσό αντιστοιχεί στο ID που χρειαζόμαστε. Έπειτα καλείται η συνάρτηση τύπου Cursor getItemID(), και γίνεται μετάβαση σε νέα καρτέλα.

```
public Cursor getItemID(String amount){
    SQLiteDatabase db = this.getWritableDatabase();
    String query = "SELECT " + COL_1 + " FROM " + TABLE_NAME +
            " WHERE " + COL_3 + " = '" + amount + "'";
    Cursor data = db.rawQuery(query,  selectionArgs: null);
    return data;
}
```

-Ο cursor getItemID() εκτελεί και επιστρέφει ένα ερώτημα sql. Επιστρέφει το ID primary key, όπου το ποσό είναι ίδιο με αυτό που επιλέχθηκε.

4.2.5) Διαγραφή επιλεγμένης συναλλαγής - DeleteItem Activity

```
Intent receiveIntent = getIntent();

selectedID = receiveIntent.getIntExtra( name: "id", defaultValue: -1);

selectedAmount = receiveIntent.getStringExtra( name: "part3");

selectedText = receiveIntent.getStringExtra( name: "amount1");

textAmount.setText(selectedText);

deleteBtn.setOnClickListener((v) -> {
        myDb.deleteItem(selectedID,selectedAmount);
        textAmount.setText("");
});
```

-Σε αυτή την δραστηριότητα εμφανίζεται το περιεχόμενο μίας συνναλαγής που επιλέχθηκε, μέσω του receive intent. Τέλος, υπάρχει η δυνατότητα διαγραφής συναλλαγής με την υλοποίηση της μεθόδου DeleteItem().

```
public void deleteItem(int id, String amount){
    SQLiteDatabase db = this.getWritableDatabase();
    String query = "DELETE FROM " + TABLE_NAME + " WHERE "
            + COL_1 + " = '" + id + "'" +
            " AND " + COL_3 + " = '" + amount + "'";
    db.execSQL(query);
}
```

-Εκτελείται ένα ερώτημα sql, διαγράφει από τον πίνακα εσόδων την συναλλαγή που επιλέξαμε.

4.2.6) Διαγράμματα – Diagrams Activity

- Διάγραμμα στήλης

```
public void setupColumnChart() throws ParseException {
    Cartesian cartesian = AnyChart.column();
    Cursor incBalance = myDb.getAllData();

    if(incBalance.getCount() ==0){
        Toast.makeText(this.getContext(), text: "Δεν υπαρχουν δεδομένα διαθέσιμα για προβολή",Toast.LENGTH_LONG).show();
        return;
    }

    while(incBalance.moveToNext()){
        months.add(incBalance.getString( columnIndex: 1));
        amount.add(incBalance.getDouble( columnIndex: 2));
    }
    List<DataEntry> dataEntries = new ArrayList<>();
    for(int i=0; i<amount.size();i++){
        SimpleDateFormat sdf = new SimpleDateFormat( pattern: "dd/MM/yyyy");
        Date parse = sdf.parse(months.get(i));
        Calendar c = Calendar.getInstance();
        c.setTime(parse);
        String month = c.getDisplayName(Calendar.MONTH,Calendar.LONG, Locale.getDefault());
        dataEntries.add(new ValueDataEntry(month, amount.get(i)));
    }
```

```
    Column column = cartesian.column(dataEntries);

    column.tooltip()
            .titleFormat("{%X}")
            .position(Position.CENTER_BOTTOM)
            .anchor(Anchor.CENTER_BOTTOM)
            .offsetX(0d)
            .offsetY(5d)
            .format("${%Value}{groupsSeparator: }");
    cartesian.animation(true);
    cartesian.title("Εσοδα");

    cartesian.yScale().minimum(0d);

    cartesian.yAxis( index: 0).labels().format("${%Value}{groupsSeparator: }");

    cartesian.tooltip().positionMode(TooltipPositionMode.POINT);
    cartesian.interactivity().hoverMode(HoverMode.BY_X);

    cartesian.xAxis( index: 0).title("Μήνας");
    cartesian.yAxis( index: 0).title("Ποσό");

    anyChartView.setChart(cartesian);
}
```

-Η μέθοδος setupColumnChart(), δημιουργεί ένα διάγραμμα στήλης. Γίνεται κλήση της μεθόδου getAllData(), για την λήψη δεδομένων από την βάση. Επιλέγονται τα πεδία

ποσό, ημερομηνία για τους άξονες x,y. Ακολουθεί επεξεργασία των δεδομένων για την εμφάνιση διαγράμματος με βάση τον μήνα.

Τέλος, προστίθενται κάποια γραφικά για την εμφάνιση του τίτλου διαγράμματος, του μήνα, το μέγεθος και της διαδραστικότητας.

- Διάγραμμα πίτας

```
public void setupPieChart() throws ParseException {
    Pie pie = AnyChart.pie();
    Cursor values = myDb.getAllData();

    if(values.getCount() ==0 ){
        Toast.makeText(this.getContext(), text: "Δεν υπαρχουν δεδομένα διαθέσιμα για προβολή",Toast.LENGTH_LONG).show();
        return;
    }

    while(values.moveToNext()){
        months.add(values.getString( columnIndex: 1));
        amount.add(values.getDouble( columnIndex: 2));
    }

    List<DataEntry> dataEntries = new ArrayList<>();
    for(int i=0; i<months.size();i++){
        SimpleDateFormat sdf = new SimpleDateFormat( pattern: "dd/MM/yyyy");
        Date parse = sdf.parse(months.get(i));
        Calendar c = Calendar.getInstance();
        c.setTime(parse);
        String month = c.getDisplayName(Calendar.MONTH,Calendar.LONG, Locale.getDefault());
        dataEntries.add(new ValueDataEntry(month, amount.get(i)));
    }
    pie.title("Εσοδα");
    pie.labels().position("outside");

    pie.legend().title().enabled(true);
    pie.legend().title()
            .text("Μηνιαίο Διάγραμμα")
            .padding(0d, 0d, 10d, 0d);

    pie.legend()
            .position("center-bottom")
            .itemsLayout(LegendLayout.HORIZONTAL)
            .align(Align.CENTER);
    pie.data(dataEntries);
    anyChartView.setChart(pie);
}
```

Κάτι αντίστοιχο με προηγουμένως συμβαίνει και στο διάγραμμα πίτας. Γίνεται λήψη των δεδομένων απο την βάση και έπειτα εκτελείται κώδικας για τα γραφικά του διαγράμματος.

- Διάγραμμα γραμμής

```
public void setupBalanceChart() throws ParseException {
    Cartesian cartesian = AnyChart.line();
    Cursor balance = myDb.getAllData();
    Cursor balance1 = myDb.getExpenditure();

    if(balance.getCount() ==0 || balance1.getCount()==0){
        Toast.makeText(this.getContext(), text: "Δεν υπαρχουν δεδομένα διαθέσιμα για προβολή",Toast.LENGTH_LONG).show();
        return;
    }

    while(balance.moveToNext()){
        amount.add(balance.getDouble( columnIndex: 2));
        months.add(balance.getString( columnIndex: 1));
    }

    while (balance1.moveToNext()){
        amountExp.add(balance1.getDouble( columnIndex: 2));
        months1.add(balance1.getString( columnIndex: 1));
    }

    cartesian.animation(true);

    List<DataEntry> dataEntries = new ArrayList<>();
    for(int y=0;y<amountExp.size();y++) {
    for(int i=0; i<amount.size();i++){
        SimpleDateFormat sdf = new SimpleDateFormat( pattern: "dd/MM/yyyy");
        Date parse = sdf.parse(months.get(i));
        Date parse1 = sdf.parse(months1.get(y));
        Calendar c = Calendar.getInstance();
        Calendar calendar = Calendar.getInstance();
        c.setTime(parse);
        calendar.setTime(parse1);
        String month = c.getDisplayName(Calendar.MONTH,Calendar.LONG, Locale.getDefault());
        String month1 = calendar.getDisplayName(Calendar.MONTH,Calendar.LONG,Locale.getDefault());
            dataEntries.add(new CustomDataEntry(month,month1 ,amount.get(i), amountExp.get(y)));
        }
    }
    Set set = Set.instantiate();
    set.data(dataEntries);
```

```
    Mapping series1Mapping = set.mapAs( mapping: "{ x: 'x', value: 'value' }");
    Mapping series2Mapping = set.mapAs( mapping: "{ x: 'x2', value: 'value2' }");

    Line series1 = cartesian.line(series1Mapping);
    series1.name("Εσοδα");
    series1.hovered().markers().enabled(true);
    series1.hovered().markers()
            .type(MarkerType.CIRCLE)
            .size(4d);
    series1.tooltip()
            .position("right")
            .anchor(Anchor.LEFT_CENTER)
            .offsetX(5d)
            .offsetY(5d);

    Line series2 = cartesian.line(series2Mapping);
    series2.name("Εξοδα");
    series2.hovered().markers().enabled(true);
    series2.hovered().markers()
            .type(MarkerType.CIRCLE)
            .size(4d);
    series2.tooltip()
            .position("right")
            .anchor(Anchor.LEFT_CENTER)
            .offsetX(5d)
            .offsetY(5d);

    cartesian.legend().enabled(true);
    cartesian.legend().fontSize(13d);
    cartesian.legend().padding(0d, 0d, 10d, 0d);

    anyChartView.setChart(cartesian);
}
```

-Στο διάγραμμα αυτό χρησιμοποιούνται τα δεδομένα από τους πίνακες εσόδων,εξόδων. Επιλέγονται τα πεδία ποσό, μήνας από τους δύο πίνακες, γίνεται προσπέλαση πινάκων και εισάγονται σε μια custom λίστα, οι συναλλαγές σχετικά με τον μήνα που δημιουργήθηκαν.

-Έπειτα, εκτελείται κώδικας για το γραφικό περιβάλλον του διαγράμματος όπως η θέση, ο τίτλος, το μέγεθος, το σχήμα και η διαδραστικότητα.

```
private class CustomDataEntry extends ValueDataEntry {
    CustomDataEntry(String x,String x2, Number value, Number value2) {
        super(x, value);
        setValue("value2", value2);
        setValue("x2",x2);
    }

}
```

-Μία κλάση που δημιουργήθηκε, για την εισαγωγή περαιτέρω δεδομένων στην custom λίστα dataEntries.

4.2.6) Εισαγωγή εξόδων – Expenditure Activity

Στην εισαγωγή εξόδων, ο κώδικας που χρησιμοποιείται είναι πανομοιότυπος με αυτόν που χρησιμοποιήσαμε στην εισαγωγή εσόδων, κεφάλαιο 4.2.3) Εισαγωγή εσόδων Income Activity. Αυτό διότι, οι πίνακες της βάσης δεδομένων περιέχουν τα ίδια πεδία, καθώς και η καρτέλα του χρήστη είναι ίδια με την καρτέλα εσόδων.

```
public boolean insertSpendingData(String amount, String dates, CharSequence text){
    SQLiteDatabase db = this.getWritableDatabase();
    ContentValues contentValues = new ContentValues();
    contentValues.put(S_COL_2,dates);
    contentValues.put(S_COL_3,amount);
    contentValues.put(S_COL_4, (String) text);
    long result = db.insert(TABLE2_NAME, nullColumnHack: null,contentValues);
    if(result == -1)
        return false;
    else
        return true;
}
```

```
public Cursor getExpenditure(){
    SQLiteDatabase db = this.getWritableDatabase();
    Cursor result = db.rawQuery( sql: "select * from " + TABLE2_NAME, selectionArgs: null);
    return result;
}
```

```
public Cursor getItemIDSpending(String amount){
    SQLiteDatabase db = this.getWritableDatabase();
    String query = "SELECT " + S_COL_1 + " FROM " + TABLE2_NAME +
            " WHERE " + S_COL_3 + " = '" + amount + "'";
    Cursor data = db.rawQuery(query, selectionArgs: null);
    return data;
}

public void deleteItemSpending(int id, String amount){
    SQLiteDatabase db = this.getWritableDatabase();
    String query = "DELETE FROM " + TABLE2_NAME + " WHERE "
            + S_COL_1 + " = '" + id + "'" +
            " AND " + S_COL_3 + " = '" + amount + "'";
    db.execSQL(query);
}
```

```
public void AddSpending(){
    btnAddData.setOnClickListener((view) → {
            boolean isInserted = myDb.insertSpendingData(editAmount.getText().toString(),dateText.getText().toString(),CategoryText.getText().toString());
            if(isInserted = true)
                Toast.makeText( context: ExpenditureActivity.this, text: "Επιτυχής καταχώρηση",Toast.LENGTH_LONG).show();
            else
                Toast.makeText( context: ExpenditureActivity.this, text: "Απέτυχε", Toast.LENGTH_LONG).show();
    });

}

public void GetDate(){
    SimpleDateFormat sdf = new SimpleDateFormat( pattern: "dd/MM/yyyy");
    String dates = sdf.format(new Date());
    dateText.setText(dates);

    btnDate.setOnClickListener((v) → {
            Calendar calendar = Calendar.getInstance();
            int year = calendar.get(Calendar.YEAR);
            int month = calendar.get(Calendar.MONTH);
            int dayOfMonth = calendar.get(Calendar.DAY_OF_MONTH);

            datePickerDialog = new DatePickerDialog( context: ExpenditureActivity.this,
                    (datePicker, year, month, day) → {
                            int selectedMonth = month+1;
                            dateText.setText(day + "/" + selectedMonth + "/" + year);
                    }, year, month, dayOfMonth);
            datePickerDialog.show();
    });

}
```

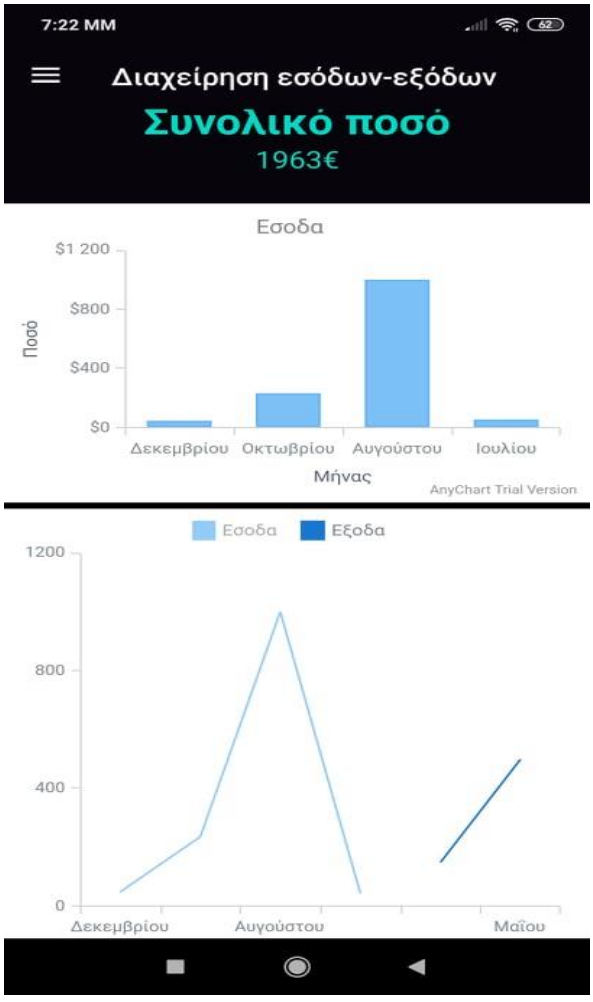
5.ΠΑΡΟΥΣΙΑΣΗ ΕΦΑΡΜΟΓΗΣ

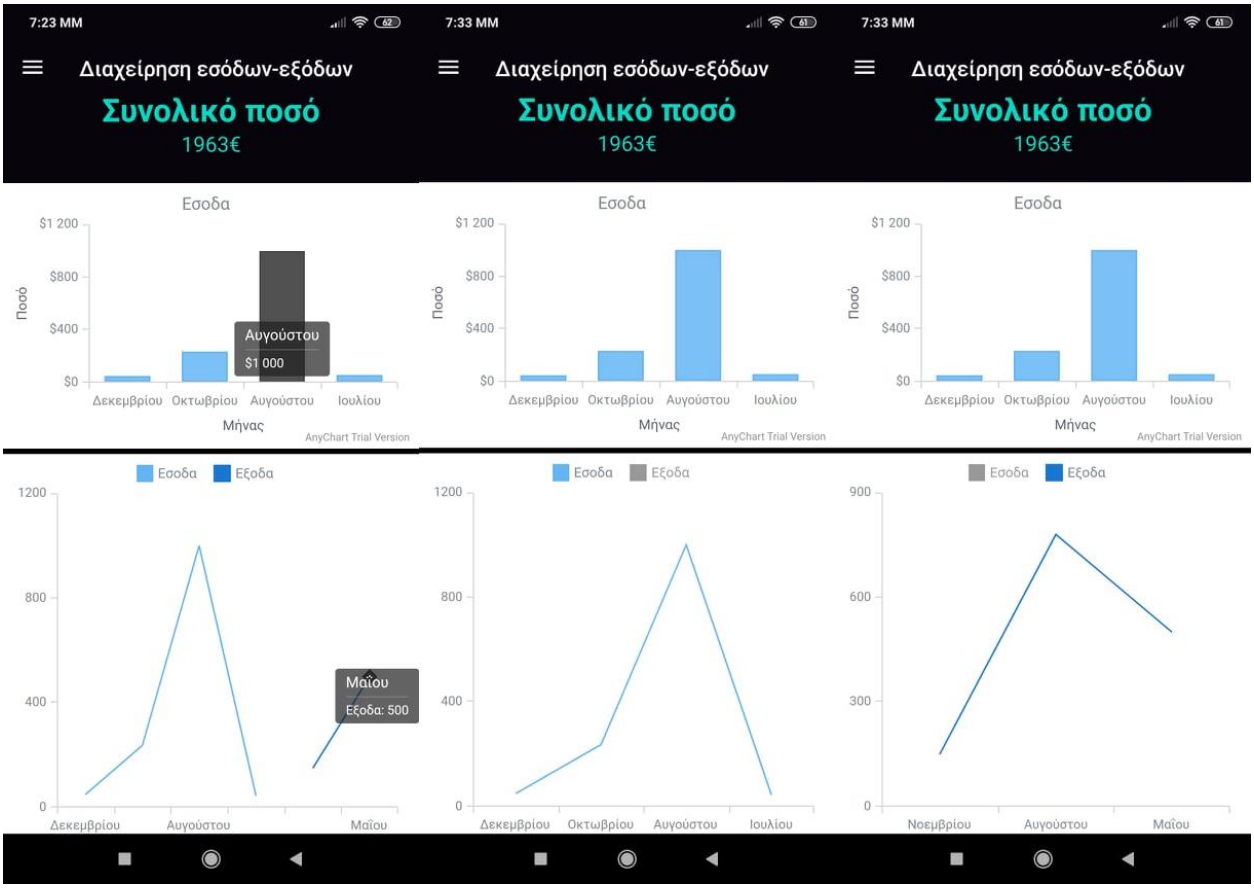
Στην ενότητα αυτή θα γίνει παρουσίαση της εφαρμογής αναλυτικά με στιγμιότυπα οθόνης και επεξήγηση των λειτουργιών. Τα στιγμιότυπα οθόνης είναι από αληθινή συσκευή στην οποία χρησιμοποιείται η εφαρμογή(Redmi note 6 pro).

5.1) Λογότυπο εφαρμογής

-Το λογότυπο της εφαρμογής που εμφανίζεται, αφού πρώτα κάποιος την κατεβάσει.

5.2) Αρχική οθόνη





Αρχική οθόνη εφαρμογής:

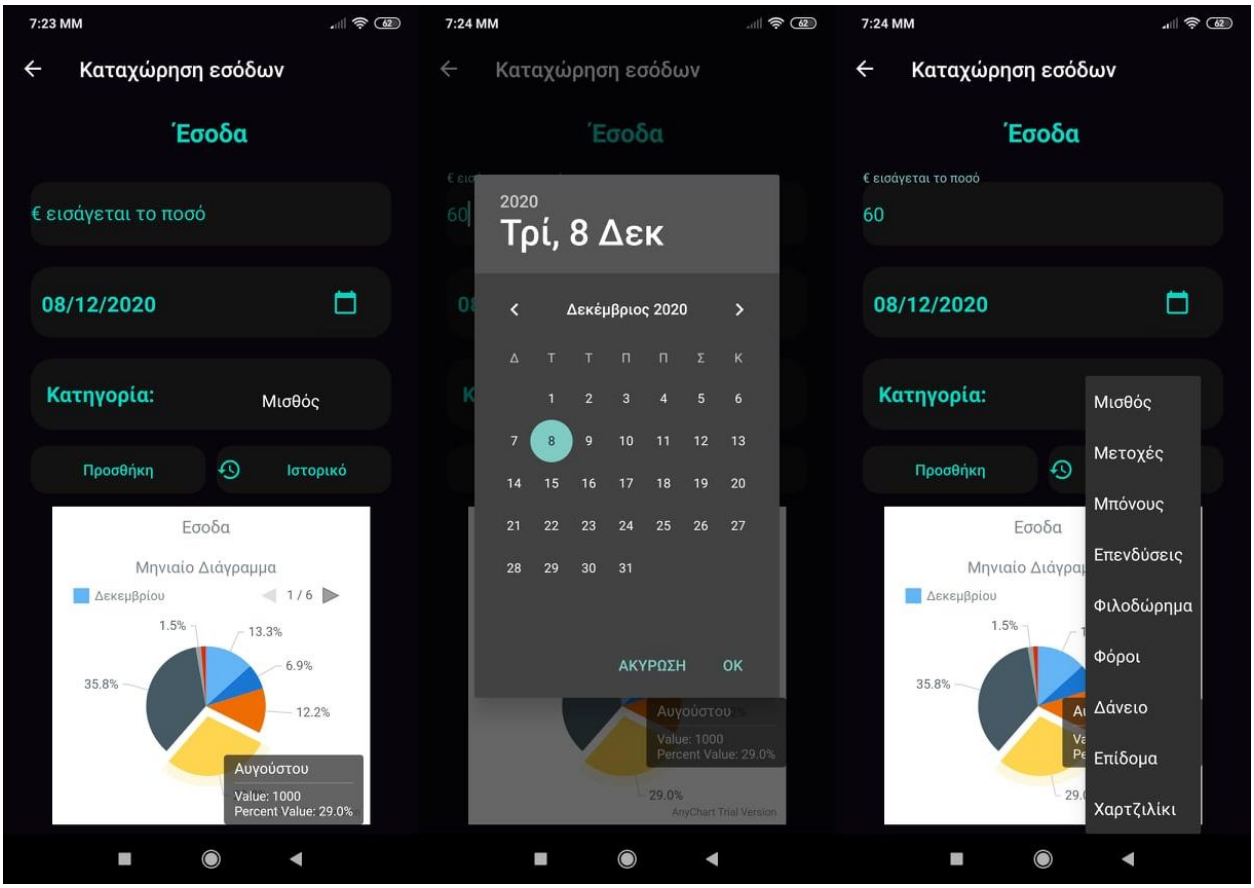
- Συνολικό ποσό χρήστη που απομένει.
- Διάγραμμα στήλης, μήνα – ποσού εσόδων.
- Διάγραμμα γραμμής, μήνα – ποσού εσόδων και εξόδων.

-Τα διαγράμματα είναι διαδραστικά. Πατώντας πάνω στα διαγράμματα εμφανίζεται το ποσό και ο μήνας συναλλαγής.

-Στο δεύτερο διάγραμμα εσόδων – εξόδων, πατώντας το μπλέ τετράγωνο(Έξοδα) γίνεται απόκρυψη των εξόδων και εμφανίζεται το διάγραμμα ροής εσόδων.

- Αντίστοιχα, πατώντας το γαλάζιο τετράγωνο(Έσοδα) γίνεται απόκρυψη των εσόδων και εμφανίζεται το διάγραμμα ροής εξόδων.

5.3) Καταχώρηση εσόδων

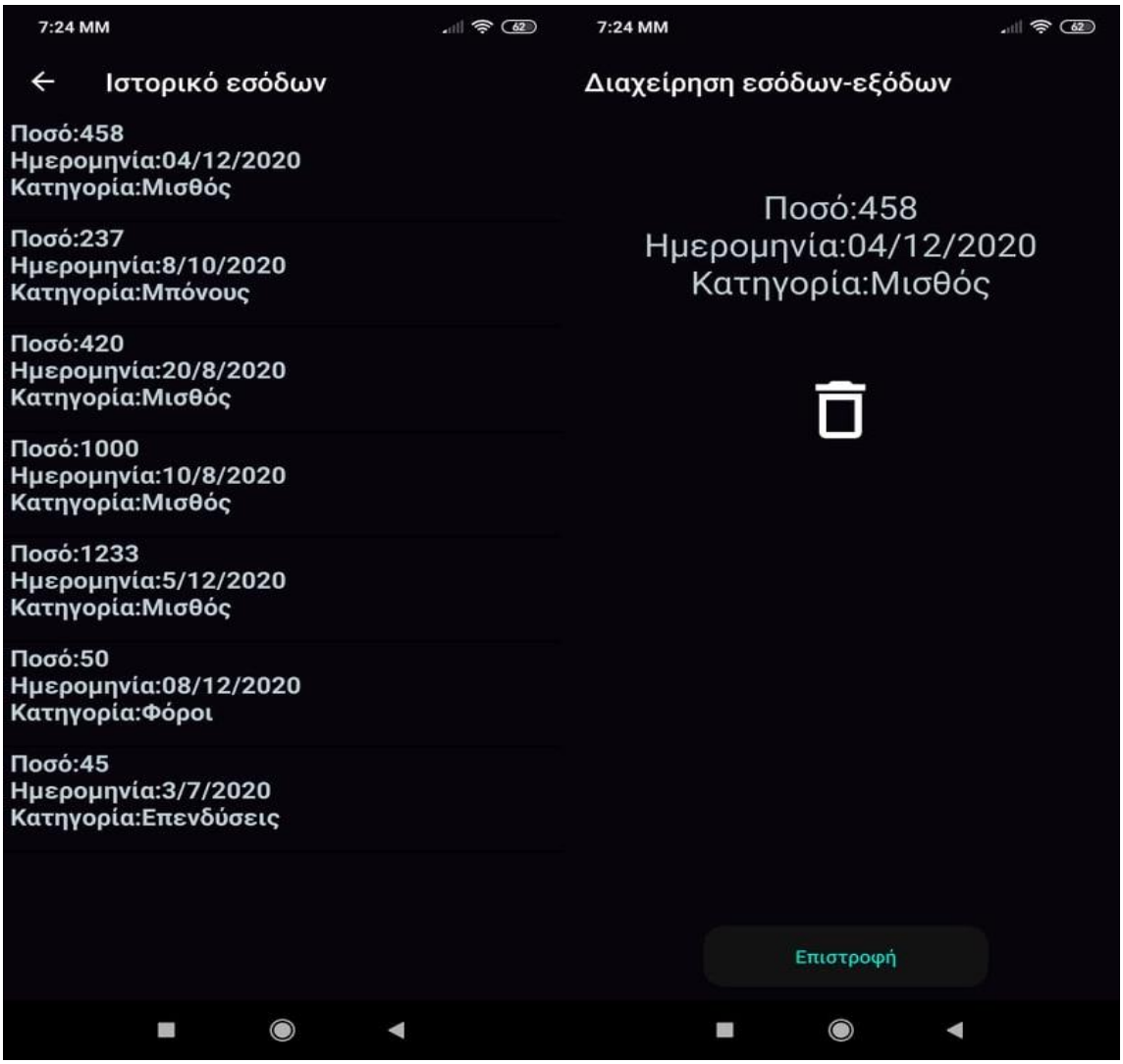


Καρτέλα εσόδων:

- Τίτλος έσοδα.
- Πεδίο εισαγωγής ποσού.
- Κουμπί επιλογής ημερομηνίας.
- Μενού επιλογής κατηγορίας.
- Κουμπί προσθήκης συναλλαγής.
- Κουμπί προβολής ιστορικού.
- Μηνιαίο Διάγραμμα πίτας.

-Στο διάγραμμα πίτας εμφανίζονται οι συναλλαγές που έκανε ο χρήστης. Πατώντας επάνω σε κάθε χρώμα εμφανίζεται το ποσό, ο μήνας και το ποσοστό των εσόδων.

5.4) Ιστορικό εσόδων



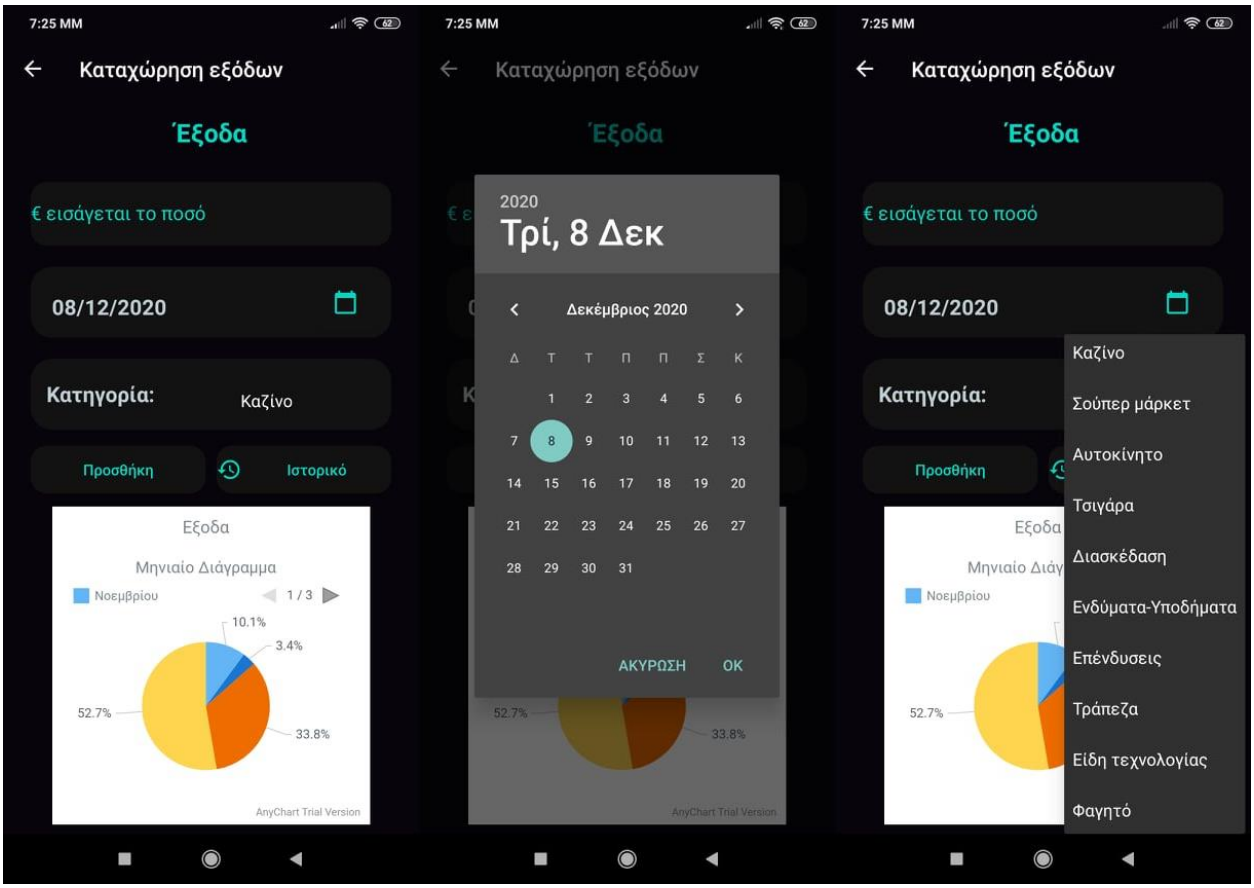
Ιστορικό εσόδων:

- Λίστα με το ιστορικό συναλλαγών.
- Εμφάνιση ποσού, ημερομηνίας, κατηγορίας.

-Πατώντας πάνω στην συναλλαγή, εμφανίζεται μια νέα καρτέλα με τα στοιχεία της συναλλαγής.

- Κουμπί διαγραφής συναλλαγής.
- Κουμπί επιστροφής στην αρχική οθόνη.

5.5) Καταχώρηση εξόδων

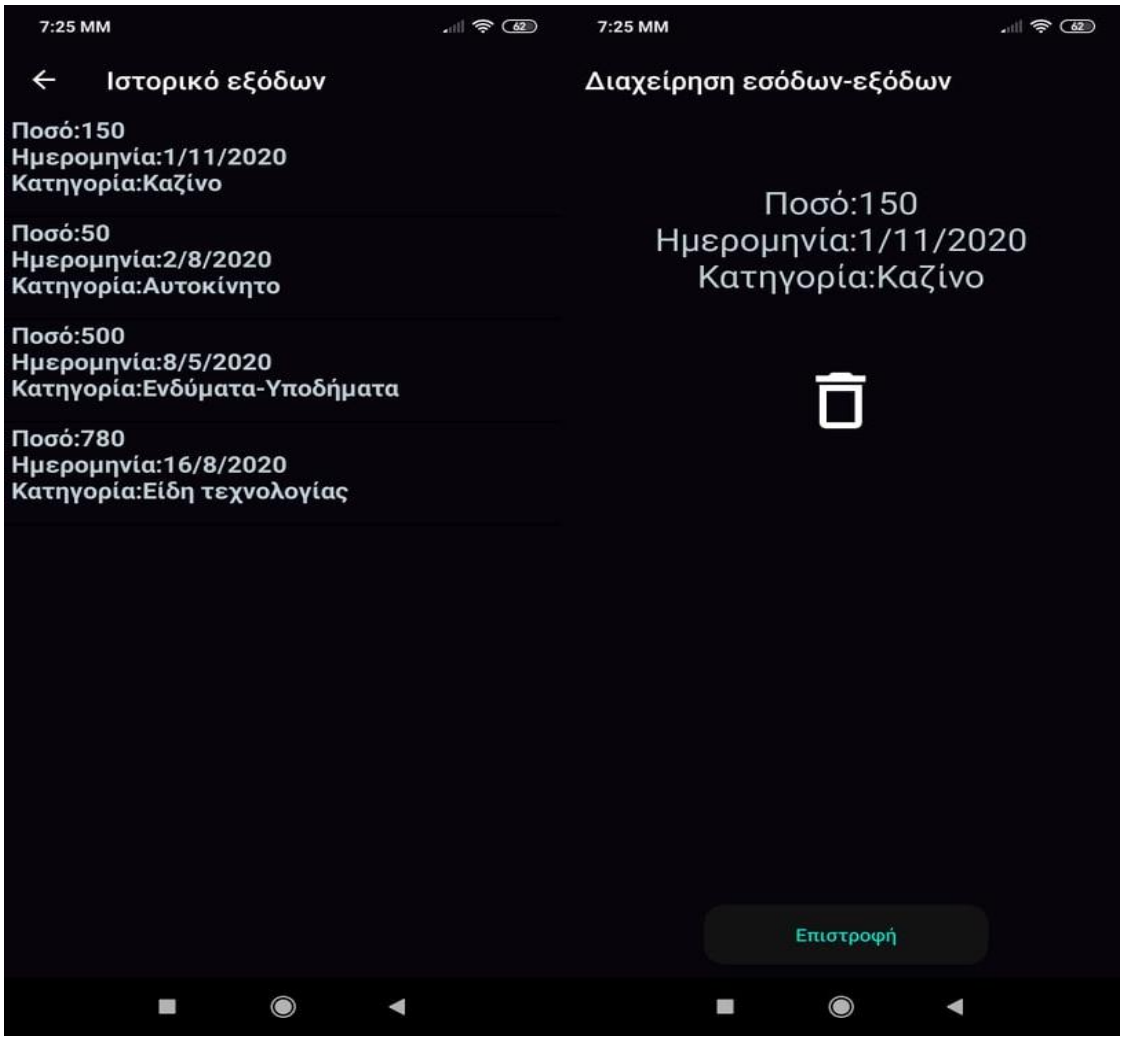


Καρτέλα εξόδων:

- Τίτλος έξοδα.
- Πεδίο εισαγωγής ποσού.
- Κουμπί επιλογής ημερομηνίας.
- Μενού επιλογής κατηγορίας.
- Κουμπί προσθήκης συναλλαγής.
- Κουμπί προβολής ιστορικού.
- Μηνιαίο Διάγραμμα πίτας.

-Στο διάγραμμα πίτας εμφανίζονται οι συναλλαγές που έκανε ο χρήστης. Πατώντας επάνω σε κάθε χρώμα εμφανίζεται το ποσό, ο μήνας και το ποσοστό των εξόδων.

5.6) Ιστορικό εξόδων



Ιστορικό εσόδων:

- Λίστα με το ιστορικό συναλλαγών.
- Εμφάνιση ποσού, ημερομηνίας, κατηγορίας.

-Πατώντας πάνω στην συναλλαγή, εμφανίζεται μια νέα καρτέλα με τα στοιχεία της συναλλαγής.

- Κουμπί διαγραφής συναλλαγής.
- Κουμπί επιστροφής στην αρχική οθόνη.

5.7) Διαγράμματα

Καρτέλα διαγράμματα:

- Διάγραμμα στήλης εσόδων.
- Διάγραμμα γραμμής εσόδων – εξόδων.

- Διάγραμμα πίτας εσόδων.
- Διάγραμμα πίτας εξόδων.

-Τα διαγράμματα είναι διαδραστικά. Πατώντας πάνω στα διαγράμματα εμφανίζεται το ποσό και ο μήνας συναλλαγής.

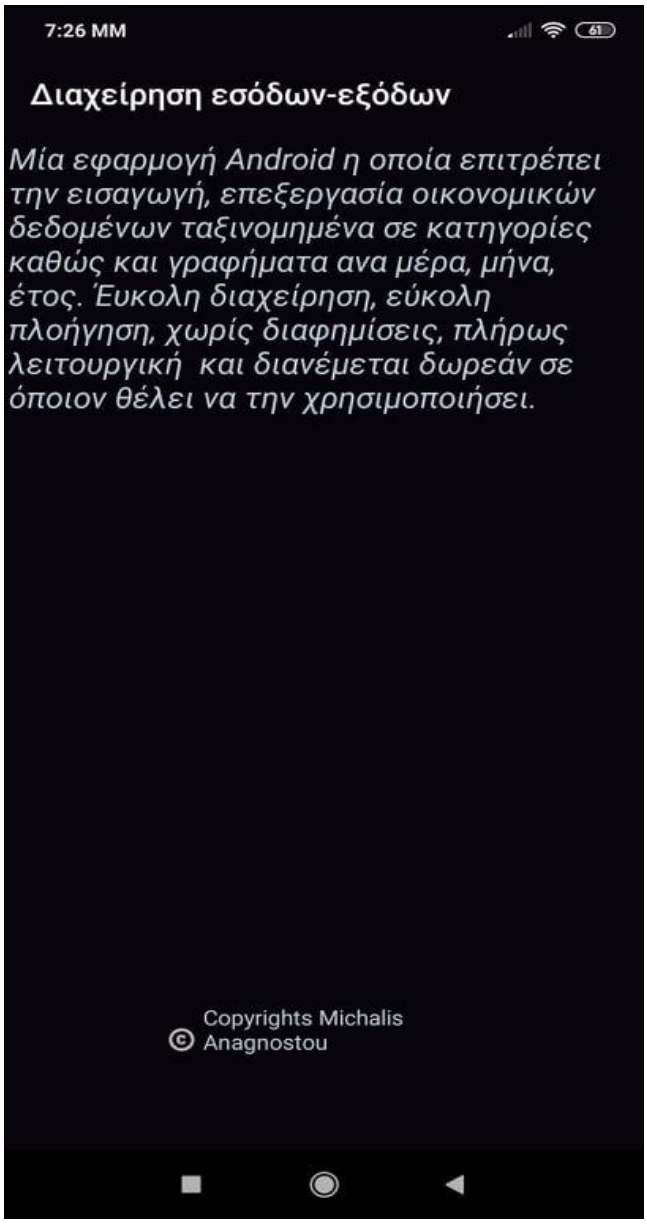
-Στο δεύτερο διάγραμμα εσόδων – εξόδων, πατώντας το μπλέ τετράγωνο(Έξοδα) γίνεται απόκρυψη των εξόδων και εμφανίζεται το διάγραμμα ροής εσόδων.

- Αντίστοιχα, πατώντας το γαλάζιο τετράγωνο(Έσοδα) γίνεται απόκρυψη των εσόδων και εμφανίζεται το διάγραμμα ροής εξόδων.

-Στα διαγράμματα πίτας εμφανίζονται οι συναλλαγές που έκανε ο χρήστης. Πατώντας επάνω σε κάθε χρώμα εμφανίζεται το ποσό, ο μήνας και το ποσοστό των εσόδων ή εξόδων αντίστοιχα.

5.8) Πληροφορίες

-Μια καρτέλα που περιγράφει τον σκοπό και την λειτουργικότητα της εφαρμογής.

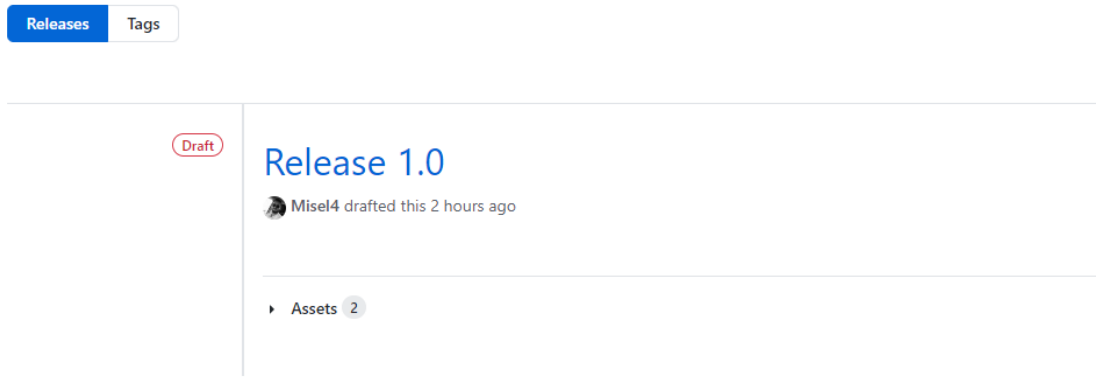


6.Εγκατάσταση εφαρμογής

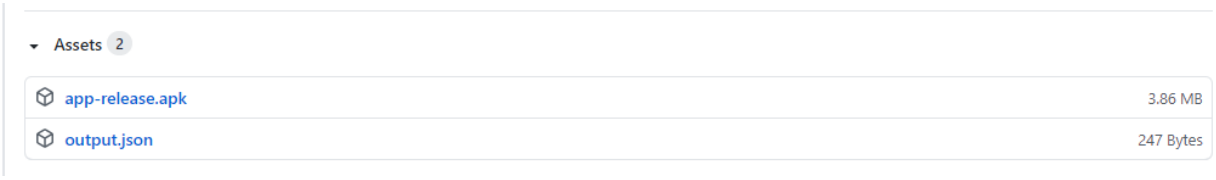
Για την εγκατάσταση της εφαρμογής ή την λήψη του κώδικα ακολουθήστε τον παρακάτω σύνδεσμο:

https://github.com/Misel4/MoneyManager.

-Μετάβαση στα releases



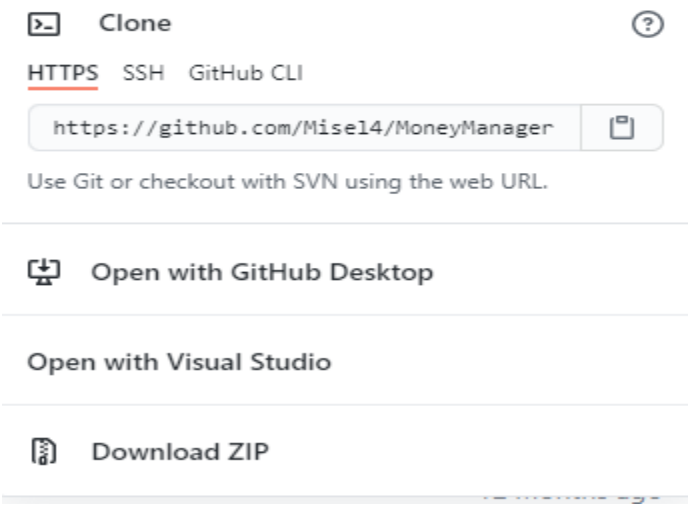
-Λήψη του apk



-Εγκατάσταση του apk στο κινητό android.

Για την λήψη του κώδικα:

$git clone https://github.com/Misel4/MoneyManager.git



7.Πηγές

- *https://www.oracle.com/java*
- *https://developer.android.com*
- *https://www.sqlite.org*
- *https://developer.android.com/studio*
- *https://el.wikipedia.org/wiki/Android*
- *https://el.techinfus.com/cifrovaya/smartfon/s-os-android.html*
- *https://www.redhat.com/en/topics/linux/what-is-the-linux-kernel*
- *https://www.tutorialspoint.com/android/android_architecture.htm*
- *https://www.frgconsulting.com/blog/why-is-java-so-popular-developers/*
- *https://github.com/AnyChart/AnyChart-Android*