ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Ανάπτυξη και εγκατάσταση διαδικτυακής εφαρμογής διαμοιρασμού χρεών

Κλέων Χηράκης
Α.Μ. 71343878

Εισηγητής: Δρ. Γεώργιος Πρεζεράκος, Καθηγητής

ΑΘΗΝΑ
ΜΑΡΤΙΟΣ 2022

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Ανάπτυξη και εγκατάσταση διαδικτυακής εφαρμογής διαμοιρασμού χρεών

Κλέων Χηράκης
Α.Μ. 71343878

Εισηγητής:

Δρ. Γεώργιος Πρεζεράκος, Καθηγητής

Εξεταστική Επιτροπή:

Δρ. Γεώργιος Πρεζεράκος
Καθηγητής

Georgios Prezerakos
Digitally signed by Georgios Prezerakos
Date: 2022.03.08 20:39:44 +02'00'

Δρ. Ιωάννης Βογιατζής
Καθηγητής

Ioannis Voyiatzis
Digitally signed by Ioannis Voyiatzis
Date: 2022.03.09 07:12:36 +02'00'

Δρ. Νικόλαος Ζάχαρης
Καθηγητής

Digitally signed by Nikolaos Zacharis
Date: 2022.03.08 21:30:38 +02'00'

Ημερομηνία εξέτασης 4/3/2022

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ

Ο κάτωθι υπογεγραμμένος Κλέων Χηράκης του Γεωργίου, με αριθμό μητρώου 71343878 φοιτητής του Τμήματος Μηχανικών Πληροφορικής και Υπολογιστών της Σχολής Μηχανικών του Πανεπιστημίου Δυτικής Αττικής, δηλώνω ότι:

«Είμαι συγγραφέας αυτής της διπλωματικής εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, οι όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε ακριβώς είτε παραφρασμένες, αναφέρονται στο σύνολό τους, με πλήρη αναφορά στους συγγραφείς, τον εκδοτικό οίκο ή το περιοδικό, συμπεριλαμβανομένων και των πηγών που ενδεχομένως χρησιμοποιήθηκαν από το διαδίκτυο. Επίσης, βεβαιώνω ότι αυτή η εργασία έχει συγγραφεί από μένα αποκλειστικά και αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο δικής μου, όσο και του Ιδρύματος.

Παράβαση της ανωτέρω ακαδημαϊκής μου ευθύνης αποτελεί ουσιώδη λόγο για την ανάκληση του πτυχίου μου».

Ο Δηλών

Χηράκης Κλέων

ΕΥΧΑΡΙΣΤΙΕΣ

Η παρούσα διπλωματική εργασία ολοκληρώθηκε εντός μιας απαιτητικής περιόδου, σε ενδιαφέροντα και ακανθώδη αντικείμενα, όπως αυτό του διαμοιρασμού χρεών και της ανάπτυξης πολλαπλών εφαρμογών σε διαφορετικά περιβάλλοντα με σκοπό την ενοποίηση σε σύστημα. Την προσπάθειά μου αυτή υποστήριξε ο επιβλέπων καθηγητής μου, κ. Γεώργιο Πρεζεράκο, τον οποίο θα ήθελα να ευχαριστήσω θερμά. Ακόμα θα ήθελα να ευχαριστήσω την οικογένειά μου, καθώς και τον Δημοσθένη Καστρινάκη για τις πολύτιμες συμβουλές του.

ΠΕΡΙΛΗΨΗ

Η παρούσα διπλωματική εργασία ασχολείται με την σχεδίαση, ανάπτυξη και εγκατάσταση μιας διαδικτυακής εφαρμογής, η οποία θα συνεργάζεται με μια εφαρμογή για έξυπνα κινητά Android με σκοπό τον συλλογικό διαμοιρασμό χρεών μεταξύ μιας ομάδας. Σε αυτό το πεδίο έχουν αναπτυχθεί πλήθος εφαρμογών, διαφέροντας ως προς τις απαιτήσεις και συνεπώς, την υλοποίηση. Απαιτήσεις της παρούσας διπλωματικής αποτελούν η βέλτιστη και δίκαιη κατανομή των χρεών, η ποιοτική εμπειρία χρήσης, καθώς και το χαμηλό κόστος συντήρησης.

ABSTRACT

The present thesis concerns the design, development and installation of a web application, which along with an Android smartphone application, will have as purpose the collective debt sharing of a group. A number of applications have been developed for this field, while their requirements and thus, their implementation, differ. The requirements of this thesis include the optimal and fair distribution of debts, the qualitative user experience, as well as keeping maintenance costs low.

ΠΕΡΙΕΧΟΜΕΝΑ

ΣΥΝΤΟΜΟΓΡΑΦΙΕΣ

REST Representational state transfer

API Application Programmable Interface

JSON JavaScript Object Notation

URI Uniform Resource Identifier

MVC Model-View-Controller

MVVM Model-View-ViewModel

ORM Object-Relational Mapping

DAO Data Access Object

ERM Entity–relationship modeling

SSOT Single source of truth

JWT JSON Web Token

CSRF Cross Site Request Forgery

XSS Cross Site Scripting

DTO Data Transfer Object

MITM man-in-the-middle

SDK Software Development Kit

QR code Quick Response code

UTC Coordinated Universal Time

XML Extensible Markup Language

OCR Optical Character Recognition

DML Data Manipulation Language

ΚΕΦΑΛΑΙΟ 1

ΕΙΣΑΓΩΓΗ

Σε αυτό το κεφάλαιο αναλύεται το αντικείμενο της διπλωματικής εργασίας και καταγράφονται οι απαιτήσεις που η εφαρμογή καλείται να ικανοποιήσει.

1.1 Περιγραφή του αντικειμένου της διπλωματικής εργασίας

Αντικείμενο της παρούσας διπλωματικής εργασίας είναι η ανάπτυξη μιας ολοκληρωμένης διαδικτυακής εφαρμογής με στόχο τον επιμερισμό των χρεών που προκύπτουν από τις συναλλαγές μιας παρέας. Συγκεκριμένα γίνεται προσπάθεια να αποφεύγεται η ανάγκη του κάθε μέλους, να υπολογίζει, αλλά και να συγκεντρώνει το ποσό που καλείται να πληρώσει ανά συναλλαγή. Αυτή η διαδικασία είναι χρονοβόρα, επίπονη, δύσχρηστη και επιρρεπής σε σφάλματα. Αντ' αυτού τα μέλη, προαιρετικά επιλέγουν να δώσουν όποιο πόσο επιθυμούν και η εφαρμογή αναλαμβάνει την διαχείριση των χρεών για όλη την ομάδα. Με τον τρόπο αυτό δίνεται η δυνατότητα στους χρήστες να εκτελούν τις συναλλαγές τους γρήγορα, εύκολα και αξιόπιστα, ενώ όποτε θελήσουν μπορούν να δώσουν/λάβουν στα/από τα υπόλοιπα μέλη της ομάδας τα ποσά που ορίζει η εφαρμογή και προέκυψαν βάσει των χρεών.

1.2 Απαιτήσεις

Η εφαρμογή καλείται να ικανοποιεί κάποιες απαιτήσεις. Αυτές περιγράφονται παρακάτω μαζί με κάποιες συνοπτικές επεξηγήσεις:

- Βέλτιστη απόσβεση χρεών, απαιτώντας τον μικρότερο δυνατό αριθμό συναλλαγών
- Όσο το δυνατόν πιο δίκαιη κατανομή χρεών, όπως στις περιπτώσεις που η διαίρεση των χρεών δεν έχει ως αποτέλεσμα ποσά πληρωτέα με χρήση πραγματικού χρήματος (πχ. επαναλαμβανόμενος δεκαδικός)
- Ικανότητα προσθήκης πολλαπλών αντικειμένων σε μια συναλλαγή, επιτελώντας λεπτομερή καταγραφή των μεριδίων και συνεισφέροντας στο δίκαιο καταμερισμό
- Αξιοποίηση δεδομένων με παραγωγή στατιστικών

- Καταμερισμός στη χρήση πόρων, ώστε να συμμετέχουν τόσο οι πόροι του εξυπηρετητή, όσο και του πελάτη, επιτυγχάνοντας μικρότερα κόστη και υψηλότερη ανταποκρισιμότητα μεταξύ πολλαπλών πελατών
- Εμπλοκή όλων των μελών στις ενέργειες μιας ομάδας. Για παράδειγμα ένα μέλος Α μπορεί να δημιουργήσει καταχωρήσεις για τα μέλη Β και Γ
- Οι ομάδες θα πρέπει να έχουν περιορισμένο αριθμό μελών, ώστε να είναι εύκολα διαχειρίσιμες από το σύστημα, αλλά και από ένα μόνο μέλος. Για παράδειγμα, όταν ένας μέλος προσθέσει μια συναλλαγή, στην χειρότερη περίπτωση, θα πρέπει να εισάγει δεδομένα που αφορούν το σύνολο της ομάδας. Ο μέγιστος αριθμός μελών ορίστηκε να είναι δεκαπέντε (15)
- Δεν αποτελεί όργανο επίλυσης διαφορών ή δεσμεύει τα μέλη του με οποιοδήποτε τρόπο
- Σχεδιασμένη για έξυπνα τηλεφωνά με περιβάλλον Android
- Εύκολη και γρήγορη για υψηλής ποιότητας εμπειρία χρήσης

ΚΕΦΑΛΑΙΟ 2

ΤΕΧΝΟΛΟΓΙΕΣ ΑΝΑΠΤΥΞΗΣ

2.1 Λογισμικό ανάπτυξης

2.1.1 Λογισμικό εξυπηρετητή

Το κομμάτι του εξυπηρετητή κάνει χρήση της γλώσσας προγραμματισμού TypeScript. Σχεδιασμένη και αναπτυγμένη από τη Microsoft, είναι μια γλώσσα προγραμματισμού με σκοπό την ανάπτυξη μεγάλων εφαρμογών και έκανε την πρώτη της εμφάνιση το 2012. Αποτελεί υπερσύνολο της JavaScript, της μονονηματικής (single threaded), μη μπλοκαριζόμενης (non-blocking) και ασύγχρονης γλώσσας προγραμματισμού, προσθέτοντας χαρακτηριστικά ισχυρού τύπου (strongly typed), καθώς και επιπλέον λειτουργίες. Ο κώδικας TypeScript μεταφράζεται σε JavaScript με χρήση μεταγλωττιστή. Αποτέλεσμα της μετάφρασης είναι αρχεία JavaScript, τα οποία με τη σειρά τους θα εκτελεστούν μέσω ενός περιβάλλοντος εκτέλεσης JavaScript, όπως είναι το Node.js. (TypeScript, 2021)

Ο εξυπηρετητής στηρίζεται στο framework Nest (NestJS) το οποίο υπαγορεύει τις βασικές κατευθυντήριες γραμμές αρχιτεκτονικής μιας διαδικτυακής εφαρμογής, παρέχοντας εργαλεία, βιβλιοθήκες και προτεινόμενες πρακτικές. (Documentation, n.d.)

Για τη βάση δεδομένων επιλέχθηκε η PostgreSQL, μια σχεσιακή βάση δεδομένων ανοικτού λογισμικού. Η σχεσιακή βάση επιτρέπει τη δημιουργία σχέσεων μεταξύ των οντοτήτων της εφαρμογής, όπως είναι οι χρήστες και οι συναλλαγές. Αυτές οι σχέσεις με τη σειρά τους επιτρέπουν την κλήση πολύπλοκων αιτημάτων (queries). Συμμορφούμενη στο ACID παρέχει τις απαραίτητες εγγυήσεις σχετικά με την εγκυρότητα των δεδομένων κατά την εκτέλεση των συναλλαγών της βάσης, κάτι το οποίο καθίσταται αναγκαίο λόγω της ευαισθησίας των δεδομένων της εφαρμογής, καθώς αυτά περιλαμβάνουν χρηματικά ποσά.

2.1.2 Λογισμικό πελάτη

Το λογισμικό του πελάτη εκτελείται σε περιβάλλον Android, το λειτουργικό σύστημα ανοικτού λογισμικού, βασισμένο στον πυρήνα του Linux. Προωθείται από

τη Google και προορίζεται κυρίως για κινητές συσκευές. (Ι. Ν. Έλληνας, 2014, 2017, p. 29)

Γίνεται χρήση της γλώσσας προγραμματισμού Kotlin, καθώς και περιορισμένη χρήση Java. Η πρώτη είναι γλώσσα προγραμματισμού ανοιχτού λογισμικού και στατικού τύπου (statically typed) με υποστήριξη πολλαπλών πλατφορμών. Διαθέτει χαρακτηριστικά αντικειμενοστραφούς και συναρτησιακού προγραμματισμού, ενώ παράγει πιο συνεκτικό κώδικα. Αναπτύχθηκε από την JetBrains και παρουσιάστηκε επίσημα το 2016. Η δεύτερη είναι από τις πιο δημοφιλείς γλώσσες αντικειμενοστραφούς προγραμματισμού, ενώ μεγάλο πλεονέκτημα αποτελεί η ικανότητά της να εκτελείται σε πολλαπλές πλατφόρμες χωρίς αλλαγές στον κώδικα. (FAQ, 22) (What is Java?, n.d.)

Επιλέχθηκε να χρησιμοποιηθεί η σχεσιακή βάση δεδομένων ανοιχτού λογισμικού SQLite που αποτελεί και την προεπιλογή του Android. Είναι μια βιβλιοθήκη γραμμένη σε γλώσσα C που περιέχει μια αυτόνομη μηχανή σχεσιακής βάσης δεδομένων. Επίσης συμμορφούμενη στο ACID, με πληθώρα χαρακτηριστικών, μικρό μέγεθος και γρήγορη εκτέλεση καθίσταται ιδανική επιλογή για κινητές συσκευές. (SQLite Home Page, n.d.)

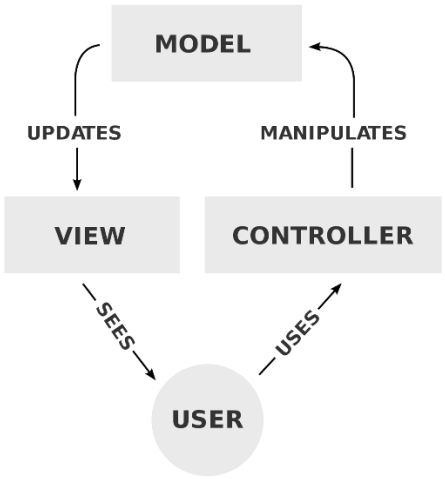
Εκμεταλλεύεται τις δυνατότητες του Android Jetpack, το οποίο αποτελεί μια συλλογή βιβλιοθηκών. Σκοπός του είναι η εφαρμογή των βέλτιστων πρακτικών κατά τη συγγραφή εφαρμογών.

2.2 Αρχιτεκτονική

2.2.1 Αρχιτεκτονική εξυπηρετητή

Το μοντέλο αρχιτεκτονικής που επιλέχθηκε για τον εξυπηρετητή είναι το MVC. Αυτό ορίζει ότι η εφαρμογή διαιρείται σε τρία κύρια συστατικά στοιχεία, τα οποία είναι (Model-view-controller, 2020):

- Model: Επιφορτίζεται με τη διαχείριση των δεδομένων
- View: Αποτελεί το επίπεδο της παρουσίασης των δεδομένων
- Controller: Δέχεται εισόδους και αλληλοεπιδρά με τα δύο παραπάνω επίπεδα με στόχο την μεταβολή της κατάστασης του μοντέλου ή/και την προβολή των δεδομένων στο View



Εικόνα 2.1: Αρχιτεκτονική MVC (Πηγή: wikipedia.com, 2009)

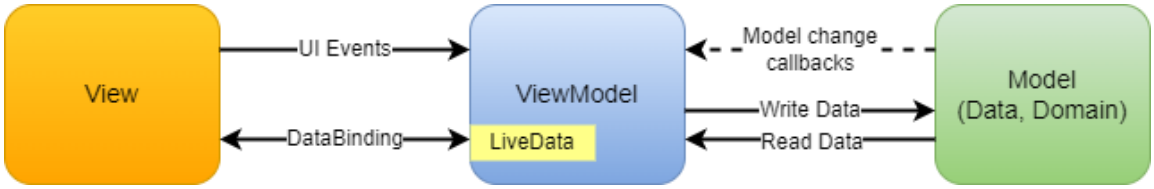
Σημειώνεται, ότι στη δική μας περίπτωση το επίπεδο View παραλείπεται από τον εξυπηρετητή, καθώς η απεικόνιση των δεδομένων επιβαρύνει την εφαρμογή για κινητά. Επιπλέον γίνεται χρήση του αρχιτεκτονικού μοτίβου REST, το οποίο ορίζει τις προδιαγραφές για την δημιουργία web APIs (RESTful web APIs). Στην περίπτωση των web APIs σε εξυπηρετητές, αυτά αποτελούν δημόσια εκτεθειμένα στο διαδίκτυο endpoints τα οποία επιστρέφουν πληροφορίες σε μια προκαθορισμένη μορφή, όπως JSON, η οποία και επιλέχθηκε ως το μορφότυπο για την ανταλλαγή δεδομένων της εφαρμογής. Η λήψη των πληροφοριών γίνεται με κλήση μεθόδων HTTP σε συγκεκριμένα URIs, καθώς αποτελεί το πιο κοινό πρωτόκολλο επικοινωνίας. Επιπλέον, επειδή αυτό το πρωτόκολλο είναι stateless, δίνει την δυνατότητα να επιτευχθούν υψηλά επίπεδα ταχύτητας, αξιοπιστίας και υποστήριξης συστημάτων υψηλής κλίμακας. Στην περίπτωση της παρούσας διπλωματικής, γίνεται χρήση sessions για την αυθεντικοποίηση και κυρίως για την υποστήριξη απομακρυσμένης αποσύνδεσης, οπότε η επικοινωνία καθίσταται stateful.

Για τη δομή του κώδικα του εξυπηρετητή επιλέχθηκε η μέθοδος package by feature, η οποία ορίζει ότι τα συστατικά στοιχεία ενός χαρακτηριστικού θα πρέπει να συγκεντρώνονται εντός του ίδιου πακέτου.

2.2.2 Αρχιτεκτονική πελάτη

Για την αρχιτεκτονική του πελάτη, επιλέχθηκε η μεθοδολογία MVVM. Αποτελείται από τα παρακάτω κύρια στοιχεία (Model–view–viewmodel, 2021):

- Model: Συνήθως αναφέρεται στο Domain Model ή στο επίπεδο πρόσβασης δεδομένων (DAL). Το domain model αποτελεί μια δομημένη απεικόνιση σημαντικών διασυνδεδεμένων εννοιών (concepts) συναφή με το επίπεδο domain που θα πρέπει να μοντελοποιηθούν στο λογισμικό. Αυτές περιλαμβάνουν δεδομένα, κανόνες, συμπεριφορές.
- View: Αποτελεί το επίπεδο της παρουσίασης των δεδομένων. Προωθεί τις αλληλεπιδράσεις του χρήστη στο παρακάτω επίπεδο.
- ViewModel (View Model): Αποτελεί τον κάτοχο της κατάστασης των δεδομένων του Model. Αποτελεί μια αφαίρεση του View, το οποίο και συνδέεται (bind) στις δημόσιες ιδιότητες του View Model. Η σύνδεση γίνεται αυτόματα μέσω του binder, ενώ όπου αυτό δεν είναι δυνατό, υπάρχει η δυνατότητα να γραφεί κώδικας που θα τον υποκαθιστά, δημιουργώντας την απαραίτητη επικοινωνία.
- Binder: Συγχρονίζει το Model με το View, παρέχοντας τα δεδομένα προς παρουσίαση.



Εικόνα 2.2: Αρχιτεκτονική MVVM (Δημιουργήθηκε από draw.io, πηγές: wikipedia.com/2021, geeksforgeeks.org/2021, medium.com/2017)

Η δομή του κώδικα που ακολουθείται είναι η μέθοδος package by layer, η οποία ορίζει ότι κάθε πακέτο συγκεντρώνει τα συστατικά στοιχεία ενός επιπέδου της εφαρμογής.

2.2.3 Επίπεδο Δεδομένων

Σε κάθε εφαρμογή, το επίπεδο των δεδομένων (Data Layer) αποτελεί βασικό κομμάτι της. Παρακάτω περιγράφεται ο τρόπος με τον οποίο γίνεται η πρόσβαση και η διαχείριση αυτών των δεδομένων.

Στο κομμάτι του εξυπηρετητή γίνεται χρήση της τεχνικής (μοτίβου σχεδίασης) ORM. Όπως δηλώνει και η ονομασία της, αφορά αντικειμενοστραφής γλώσσες

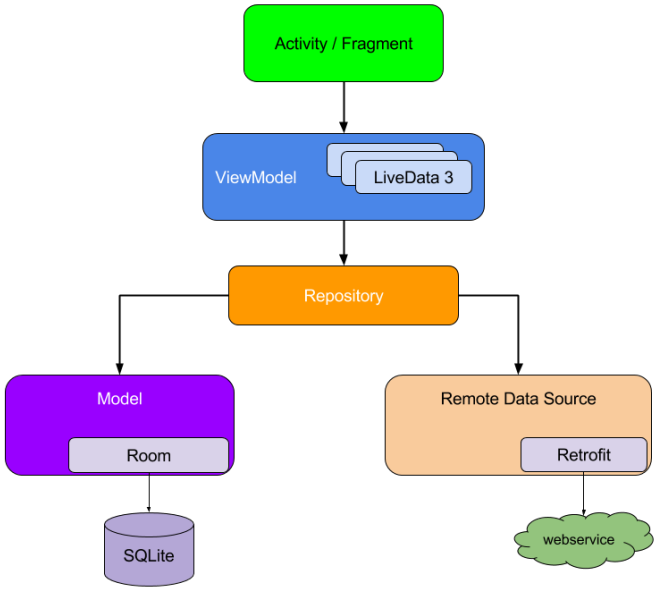
προγραμματισμού και χρησιμοποιείται για τη μετατροπή των δεδομένων που διατηρούν οι βάσεις δεδομένων. Πολλές σχεσιακές βάσεις δεδομένων χρησιμοποιούν απλούς τύπους δεδομένων ως περιεχόμενο των στηλών τους, το οποίο μπορεί να περιγράφει πιθανές σχέσεις μεταξύ πινάκων. Αυτές οι σχέσεις, στον αντικειμενοστραφή προγραμματισμό, δεν μπορούν να περιγράφουν από απλούς τύπους, αλλά απαιτούν πιο περίπλοκες δομές. Ευθύνη του ORM είναι η μετατροπή των απλών δεδομένων της βάσης σε πιο περίπλοκα αντικείμενα και το αντίστροφο, ενώ παράλληλα διατηρούνται οι σχέσεις και οι ιδιότητές τους. (Object–relational mapping, 2021)

Επίσης επιτρέπουν την εκτέλεση αιτημάτων στη βάση δεδομένων απλά και μόνο με τη χρήση της γλώσσας προγραμματισμού που αναπτύσσεται μια εφαρμογή, χωρίς να απαιτείται η γνώση SQL. Συγκρινόμενο με παραδοσιακότερες τεχνικές επικοινωνίας με τη βάση δεδομένων, το ORM συνήθως οδηγεί σε αισθητά λιγότερο κώδικα. Η βιβλιοθήκη που επιλέχθηκε είναι η TypeORM που προσφέρει πλήρη υποστήριξη της TypeScript και συνεργάζεται με το Nest framework.

Όπως προαναφέρθηκε παραπάνω, γίνεται χρήση του Android Jetpack, το οποίο περιλαμβάνει την βιβλιοθήκη Room για την διαχείριση της βάσης δεδομένων SQLite. Ο προτεινόμενος τρόπος αλληλεπίδρασης με τα αποθηκευμένα δεδομένα είναι μέσω DAO. Το DAO αποτελεί ένα μοτίβο σχεδίασης παρέχοντας μια αφηρημένη διεπαφή με τη βάση δεδομένων. Η διεπαφή περιέχει κλήσεις, ώστε να δίνεται πρόσβαση στο επίπεδο δεδομένων, χωρίς να εκθέτονται πληροφορίες της βάσης δεδομένων. Ουσιαστικά, ευθύνη του είναι η μεταφορά δεδομένων ανάμεσα στην εφαρμογή (αντικείμενα) και τη βάση δεδομένων (καταχωρήσεις). Σημειώνεται ότι η διεπαφή αποτελεί το σχέδιο των κλήσεων, όπως είναι η περιγραφή των τύπων δεδομένων, ενώ η υλοποίηση τους αποτελεί ξεχωριστό κομμάτι. Αυτό επιτρέπει την εύκολη μετατροπή κάποιας υλοποίησης αν προκύψει κάποια αλλαγή και διευκολύνει τους ελέγχους μονάδων (Unit tests). (Data access object, 2020)

Επιπλέον, γίνεται χρήση και του μοτίβου σχεδίασης Repository. Ένα Repository επιφορτίζεται με τη λογική πρόσβασης διαφορετικών πηγών δεδομένων (data sources), συγκεντρώνοντας την πρόσβαση κοινών δεδομένων σε ένα μέρος. Για παράδειγμα, ένα Repository μπορεί να περιέχει δύο πηγές δεδομένων, μία τοπική βάση δεδομένων και άλλη μία απομακρυσμένη πηγή που τα δεδομένα της λαμβάνονται με κλήσεις σε κάποιο service. Το Repository εμπεριέχει τη λογική του ποια πηγή θα επιλεχθεί. Αυτό το μοτίβο επιτρέπει την εύκολη συντήρηση του

κώδικα, αφού «αποσυνδέει» (decouple) την υποδομή πρόσβασης των δεδομένων από το επίπεδο domain model. (Design the infrastructure persistence layer, 2018)



Εικόνα 2.3: Μοτίβο σχεδίασης Repository με τοπική και απομακρυσμένη πηγή δεδομένων (Πηγή: developer.android.com, χ.χ.)

2.2.4 Dependency Injection

Το Dependency Injection αποτελεί μια τεχνική για την παροχή αντικειμένων σε κάποιο άλλο αντικείμενο που εξαρτάται από αυτά. Για το λόγο αυτό, ονομάζονται εξαρτήσεις (dependencies). Πελάτης ονομάζεται το αντικείμενο που λαμβάνει τις εξαρτήσεις, ενώ τα παρεχόμενα αντικείμενα ονομάζονται υπηρεσίες (services). Η παροχή των εξαρτήσεων γίνεται μέσω του injector που μεσολαβεί ανάμεσα στις δύο πλευρές. Ο πελάτης που θέλει να εκτελέσει κλήση σε κάποια υπηρεσία δεν θα πρέπει να δημιουργήσει αυτήν την υπηρεσία, ούτε καν να προσπαθήσει να την βρει. Η πρόσβαση και πιθανώς και η κατασκευή της υπηρεσίας παρέχονται από τον injector. Ο πελάτης δεν γνωρίζει τίποτα, ούτε ακόμα και το ποιες υπηρεσίες αποζητά, καθώς αυτές παρέχονται από τον injector. Το μόνο που γνωρίζει είναι η διεπαφή της υπηρεσίας, ώστε να είναι σε θέση να την χρησιμοποιήσει. Ο διαχωρισμός της κατασκευής και της χρήσης, πετυχαίνει επαναχρησιμοποιήσιμο και ευανάγνωστο κώδικα. (Dependency injection, 2021)

Τόσο η εφαρμογή του εξυπηρετητή, όσο και του πελάτη εκμεταλλεύονται τις δυνατότητες του Dependency Injection. Ο εξυπηρετητής στηρίζεται στα ενσωματωμένα εργαλεία του Framework, ενώ η εφαρμογή πελάτη στηρίζεται στη βιβλιοθήκη Hilt.

ΚΕΦΑΛΑΙΟ 3

BACK END

3.1 Βάση δεδομένων

Το πρώτο βήμα στη σχεδίαση του εξυπηρετητή ήταν η σχεδίαση της βάσης δεδομένων. Αυτή αποτελεί το θεμέλιο λίθο για τα υπόλοιπα επίπεδα. Η σχεδίαση έγινε με τη μέθοδο ERM. Αρχικό μέλημα της σχεδίασης είναι η συγκέντρωση των απαραίτητων οντοτήτων (model entities) και εν συνεχεία η καταγραφή των σχέσεων μεταξύ τους. Οι οντότητες αποτελούν λογικό κατασκεύασμα και μεταφράζονται σε πίνακες, οι οποίοι αποτελούν φυσικό κατασκεύασμα.

3.1.1 Data Model

Η εφαρμογή απαιτεί την είσοδο των χρηστών με χρήση των διαπιστευτηρίων τους ή μέσω τρίτου παρόχου (social login). Για να επιτευχθεί αυτός ο στόχος επιλέχθηκε η δημιουργία μιας οντότητας που αναπαριστά ένα γενικό χρήστη (User). Ο πίνακας users περιέχει γενικές πληροφορίες για τον χρήστη, όπως το όνομα, τον πάροχο σύνδεσης και την σημαία του αν ο λογαριασμός είναι διαγραμμένος. Επειδή ανάλογα με τον τρόπο σύνδεσης, χρειάζονται και διαφορετικές πληροφορίες, αποφασίστηκε ότι κάθε πάροχος θα έχει και την αντίστοιχη οντότητα. Πάροχος θεωρείται και το τοπικό σύστημα λογαριασμών. Ως τρίτος πάροχος επιλέχθηκε η υπηρεσία Facebook Login. Επομένως, πέρα από τον αρχικό πίνακα, προστέθηκαν ο πίνακας των τοπικών χρηστών (users_kchap) και των χρηστών που συνδέθηκαν μέσω Facebook (users_fb). Και οι δύο πίνακες έχουν ξένο κλειδί που δείχνει στον κεντρικό πίνακα users. Επιλέχθηκε η χρήση σημαίας διαγραφής λογαριασμού έναντι της διαγραφής της εγγραφής του χρήστη από τον πίνακα, λόγω εξαρτήσεων των ξένων κλειδιών.

Σύμφωνα με το RFC 3696, οι διευθύνσεις email έχουν πεπερασμένο μήκος 320 χαρακτήρων. Παρ’ όλα αυτά το RFC 5321 δηλώνει ότι ένα path είναι μέγιστου μήκους 256 χαρακτήρων, ενώ συμπληρώνει ότι μέσα στο path συμπεριλαμβάνονται σημεία στίξης και διαχωριστικά στοιχείων. Στην παράγραφο 4.1.2 του κειμένου RFC δίνεται η δομή του path, η οποία είναι:

```
Path = "<" [ A-d-l ":" ] Mailbox ">"
```

Βλέπουμε ότι το path περιλαμβάνει γωνιακές αγκύλες στην έναρξη και τη λήξη του. Επομένως, από το μέγιστο μήκος του path, αφαιρούμε τις δύο αγκύλες και το αποτέλεσμα είναι το μέγιστο μήκος μιας διεύθυνσης email, δηλαδή 254 χαρακτήρες.

Η διεύθυνση ηλεκτρονικού ταχυδρομείου αποτελεί προϋπόθεση για την εγγραφή και σύνδεση των χρηστών μέσω του τοπικού παρόχου. Στους χρήστες του Facebook κατά την εγγραφή τους στο μέσο κοινωνικής δικτύωσης τους δίνεται η επιλογή χρήσης τηλεφωνικού αριθμού ή email. Αυτό σημαίνει ότι υπάρχει πιθανότητα οι χρήστες που συνδέονται μέσω Facebook να μην έχουν καταχωρημένη διεύθυνση ηλεκτρονικού ταχυδρομείου και γι' αυτό δεν αποτελεί πεδίο στον αντίστοιχο πίνακα. Το πεδίο password του πίνακα users_kchap είναι ένα αλφαριθμητικό και περιέχει τον κατακερματισμένο (hashed) κωδικό πρόσβασης του κάθε τοπικού χρήστη.

Το επόμενο βήμα ήταν να σχεδιαστεί η οντότητα της ομάδας (groups), η οποία αποθηκεύει τις πληροφορίες που την αφορούν, δηλαδή το όνομά της, τον πιθανό κωδικό διαμοιρασμού για ένταξη στην ομάδα και τον αριθμό έκδοσης του πόρου. Το πεδίο της έκδοσης θα το συναντήσουμε και σε άλλες οντότητες και αποτελεί ένδειξη για τον αν ο πόρος που λαμβάνουμε και ο πόρος που είχαμε είναι διαφορετικοί, καθώς κάθε ενημέρωση του πόρου προκαλεί την αύξηση του αριθμού κατά μία μονάδα. Το πεδίο overwrite, αποτελεί ένδειξη για το αν κάποιος χρήστης μπορεί να αντικαταστήσει ήδη αλλαγμένους πόρους και συγκεκριμένα αφορά τα λογιστικά συμβάντα της εφαρμογής. Επιπλέον διατηρείται και η χρονοσφραγίδα τη στιγμή της δημιουργίας της.

Ως (λογιστικά) συμβάντα θεωρούνται τα γεγονότα που επιφέρουν αλλαγές στο λογιστικό υπόλοιπο (balance) των μελών. Το συγκεκριμένο υπόλοιπο καλείται και πρωτεύων ή απλά υπόλοιπο. Οι τύποι των συμβάντων είναι οι συναλλαγές (transactions) και οι μεταφορές (transfers). Συναλλαγές είναι οι διάφορες χρηματικές δραστηριότητες που συμμετέχουν τα μέλη, όπως μια έξοδος για φαγητό. Μεταφορά είναι η μετακίνηση χρηματικών ποσών από ένα μέλος σε κάποιο άλλο, όπως για παράδειγμα η μεταφορά χρημάτων για αποπληρωμή χρεών ή ο δανεισμός. Οι πίνακες των συμβάντων μοιράζονται αρκετά πεδία, όπως το αναγνωριστικό, το ξένο κλειδί της ομάδας και του δημιουργού, η περιγραφή (αλλά με διαφορετικό μέγιστο μήκος ανά πίνακα), η έκδοση, η χρονική σήμανση του συμβάντος (dateTime), η τοποθεσία, το πότε δημιουργήθηκε στο κινητό (created_on), το πότε δημιουργήθηκε στον εξυπηρετητή (uploaded_on) και το πότε μεταβλήθηκε στον εξυπηρετητή (modified_on).

Τα επιπλέον πεδία ανά πίνακα είναι τα πεδία του τίτλου και της συνολικής τιμής για τον πίνακα των συναλλαγών. Ενώ για τον πίνακα των μεταφορών τα επιπλέον πεδία είναι το ποσό μεταφοράς, τα πεδία του παραλήπτη ή δικαιούχο (to_user_id) και του αποστολέα ή εντολέα (from_user_id). Βλέποντας τα κοινά χαρακτηριστικά των δύο πινάκων μια προσέγγιση θα ήταν η δημιουργία ενός πίνακα με στήλες τα κοινά χαρακτηριστικά και η σύνδεση του στους πίνακες των συμβάντων μέσω ξένου κλειδιού. Παρ' όλα αυτά, προτιμήθηκε κάθε πίνακας να διατηρεί τις δικές του στήλες, ώστε να αποφευχθούν joins τα οποία εν δυνάμει βλάπτουν την απόδοση της εφαρμογής.

Η τοποθεσία είναι τύπου JSONB και αποτελεί ένα JSON δύο προαιρετικών πεδίων. Αυτά είναι το αλφαριθμητικό του ονόματος και το αντικείμενο των συντεταγμένων της τοποθεσίας. Το τελευταίο αποτελείται από δύο υποχρεωτικά πεδία για το γεωγραφικό μήκος και πλάτος. Σημειώνεται ότι αν και τα δύο πεδία της τοποθεσίας είναι κενά, τότε αντί αντικειμένου JSONB αναθέτεται η κενή τιμή (null). Η χρήση του τύπου JSONB επιλέχθηκε σε σημεία που τα δεδομένα δεν είχαν τόσο ενδιαφέρον, όπως για εξόρυξη συγκεντρωτικών πληροφορίων ή οι ανάγκες για βελτιστοποίηση της πρόσβασής τους με χρήση ευρετηρίου (indexing) ήταν μικρές έως μηδαμινές. Αυτό επέτρεψε μια απλούστερη προσέγγιση για την αποθήκευση τους, βοηθώντας παράλληλα στην μείωση της πολυπλοκότητας της εφαρμογής, ενώ επιτρέπει και την ευκολότερη μεταβολή της δομής σε σχέση με ένα πίνακα της βάσης.

Τα παραπάνω συμβάντα θα έχουν θετικό ή αρνητικό αντίκτυπο στο πρόσημο των υπολοίπων των μελών της ομάδας, τα οποία αποθηκεύονται στον ενδιάμεσο πίνακα που αντιστοιχίζει τους χρήστες με τις ομάδες (groups_users) και ουσιαστικά αντιπροσωπεύουν τα ποσά που πρέπει να πληρωθούν ή να πληρώσουν οι χρήστες σε σχέση με μια ομάδα. Θετικό πρόσημο σημαίνει ότι στον χρήστη χρωστάνε χρήματα, ενώ αρνητικό σημαίνει ότι ο χρήστης χρωστάει. Με βάση αυτά τα υπόλοιπα θα πραγματοποιηθεί και ο υπολογισμός των χρεών μεταξύ των μελών. Στην εισαγωγή αναφέρθηκε, ότι απαίτηση της εφαρμογής είναι η όσο το δυνατόν πιο δίκαιη κατανομή των χρεών. Για το λόγο αυτό δημιουργήθηκε το πεδίο ep_balance (εφεξής καλείται και δευτερεύων υπόλοιπο), το οποίο συμβουλεύεται η εφαρμογή για να διαμοιράσει ποσά τα οποία δεν αντανακλούν πραγματικά χρήματα (μη πληρωτέα ποσά), όπως είναι το ισόποσο μοίρασμα ενός λεπτού του ευρώ σε δύο άτομα. Το

πεδίο joined_on πληροφορεί για την ημερομηνία που ο χρήστης εντάχθηκε στην ομάδα.

Οι αλλαγές στα υπόλοιπα που προκαλούνται από κάθε λογιστικό συμβάν καταγράφονται στον πίνακα balance_change. Μέσω αυτού του πίνακα γίνεται δυνατός ο συγχρονισμός μεταξύ του εξυπηρετητή και των κινητών συσκευών των χρηστών. Τα πεδία του είναι το ξένο κλειδί της ομάδας (group_id), ο τύπος οντότητας που ευθύνεται για τη μεταβολή του υπολοίπου (gen_for) και η ενέργεια που εκτελέστηκε. Τα δύο τελευταία πεδία έχουν enumerated τύπους και συγκεκριμένα το balance_change_gen_for_enum και το balance_change_gen_for_action_enum αντίστοιχα. Οι enumerated τύποι δέχονται προκαθορισμένες τιμές και συγκεκριμένα:

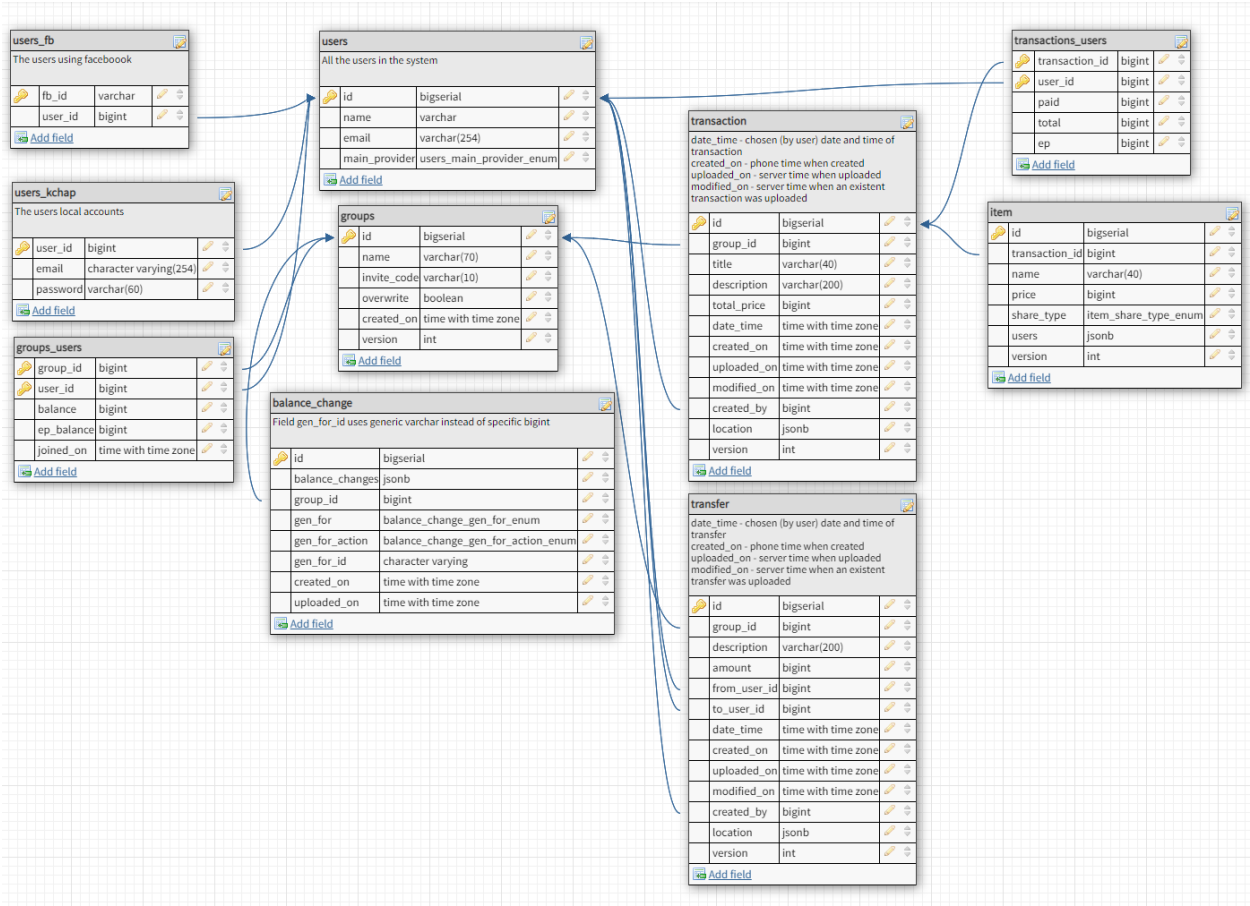
- balance_change_gen_for_enum: TRANSACTION, TRANSFER, GROUP
- balance_change_gen_for_action_enum: CREATE, UPDATE, DELETE

Το πεδίο gen_for_id υποδεικνύει το αναγνωριστικό της οντότητας που οφείλεται η καταχώρηση και επιλέχθηκε ένας γενικός τύπος δεδομένων (αλφαριθμητικές τιμές) ώστε σε περίπτωση που μελλοντικά προστεθούν και άλλες οντότητες/συμβάντα, το αναγνωριστικό να παραμένει συμβατό (δεκτές αριθμητικές και αλφαριθμητικές τιμές). Αυτό δεν έχει αρνητικές επιπτώσεις στο business επίπεδο, όπως θα δούμε και παρακάτω. Τα πεδία created_on και uploaded_on, αντιστοιχούν στο πότε δημιουργήθηκε (ώρα κινητού) και πότε ο εξυπηρετητής έλαβε τη μεταβολή του υπολοίπου. Το πεδίο balance_changes είναι τύπου JSONB και σε αυτό αποθηκεύονται οι λογιστικές αλλαγές των μελών (JSON πίνακας). Κάθε λογιστική αλλαγή αποτελεί JSON αντικείμενο με πεδία το αναγνωριστικό του χρήστη και την αρνητική ή θετική τιμή μεταβολής του βασικού ή/και του δευτερεύοντος υπολοίπου. Αν κάποιο από τα δύο υπόλοιπα είναι μηδενικό, τότε παραλείπεται η εισαγωγή του στο αντικείμενο. Αν και τα δύο υπόλοιπα είναι μηδενικά, παραλείπεται η εισαγωγή του αντικειμένου στον πίνακα. Τέλος, κάποιος θα μπορούσε να υποθέσει ότι η τιμή GROUP για τον τύπο balance_change_gen_for_enum δεν έχει κάποια βάση, καθώς δεν αποτελεί λογιστικό συμβάν. Αυτή η υπόθεση είναι σωστή και οποιαδήποτε ενέργεια αυτού του τύπου δεν μεταβάλλει τα υπόλοιπα των χρηστών. Ο λόγος που προστέθηκε περιγράφεται σε επόμενη ενότητα.

Τα λογιστικά συμβάντα που αναφέρθηκαν παραπάνω αποτελούν οντότητες του μοντέλου και διαθέτουν τους αντίστοιχους πίνακες. Επιπλέον, όσον αφορά τις συναλλαγές δημιουργήθηκαν δύο ακόμα οντότητες (πίνακες). Κάθε συναλλαγή

περιλαμβάνει ένα ή περισσότερα αντικείμενα. Ο πίνακας item περιέχει αυτά τα αντικείμενα και τις πληροφορίες αυτών, όπως το όνομα, η τιμή, ο τρόπος διαμοιρασμού (share_type) που έχει enumerated τύπο δεδομένων με τιμές ποσοστό (PERCENTAGE), ποσό (AMOUNT) και βάρος (WEIGHT). Επίσης διαθέτει, το ξένο κλείδι transaction_id που συσχετίζει την εγγραφή με κάποια συναλλαγή, το πεδίο έκδοσης και το πεδίο users, τύπου JSONB, με πεδία που αντιστοιχίζουν το μερίδιο, το ποσό πληρωμής και το μη πληρωτέο ποσό κάθε μέλους. Οι συνολικές πληροφορίες κάθε μέλους για κάθε συναλλαγή φυλάσσονται σε ξεχωριστό πίνακα (transactions_users). Τα πεδία του πίνακα περιέχουν τα ξένα κλειδιά της συναλλαγής και του χρήστη, το ποσό που πληρώθηκε, το ποσό που έπρεπε να πληρωθεί και το μη πληρωτέο ποσό. Από αυτούς τους δύο πίνακες μπορούν να εξαχθούν πολύτιμα συμπεράσματα και στατιστικά στοιχεία, όπως η εύρεση συναλλαγών με βάση τα αντικείμενα ή το πλήθος των συναλλαγών που έλαβε μέρος ο χρήστης, καθώς και το ύψος της δαπάνης του.

Τα ποσά σε όλους τους πίνακες περιγράφονται με τον τύπο δεδομένων bigint (Big Integer), δηλαδή ακέραιους αριθμούς, ενώ η μοναδιαία τιμή αφορά το ένα λεπτό του ευρώ. Παρόλο που η PostgreSQL διαθέτει τύπο, ειδικά σχεδιασμένο για χρήματα (τύπος money), επιλέχθηκε ο τύπος bigint, καθώς η εφαρμογή επεξεργάζεται τα ποσά ως ακεραίους για μεγαλύτερη ευελιξία. Αποθηκεύοντας τα δεδομένα ως ακεραίους εξ’ αρχής, επιτρέπεται η ανάκτηση τους δίχως την ανάγκη μετατροπής τους σε κάποια άλλη μορφή, όπως από money σε bigint. Σε αυτό το σημείο, θα πρέπει να αναφερθεί ότι τα πεδία ep του πίνακα transaction_users και ep_balance του πίνακα groups_users, είναι επίσης πεδία τύπου bigint. Ο διαχωρισμός των ποσών, επιτρέπει την διατήρηση μεγάλων τιμών. Στην περίπτωση των απλών ποσών, αυτό μεταφράζεται στη διατήρηση τιμών της τάξης των πεντάκις εκατομμυρίων, το οποίο θεωρήθηκε ικανοποιητικό για τις ανάγκες της εφαρμογής, ενώ στις περιπτώσεις των ep και ep_balance επιτρέπει μεγαλύτερη ακρίβεια στην αναπαράσταση ποσών που είναι αδύνατο να πληρωθούν. Το επίπεδο view του προγράμματος πελάτη είναι επιφορτισμένο για την ορθή αναπαράσταση των ποσών.



Εικόνα 3.1: Σχεδιάγραμμα του μοντέλου της βάσης δεδομένων: Οι πίνακες και οι μεταξύ τους σχέσεις (Παράχθηκε από την εφαρμογή dbdesigner.net)

Τα πεδία που αποτελούν συχνά κριτήριο αναζήτησης έχουν ευρετηριαστεί (index) για γρηγορότερη προσπέλαση των καταχωρήσεων. Αυτά είναι τα πεδία user_id του πίνακα groups_users, invite_code του πίνακα groups, group_id, gen_for και gen_for_action του πίνακα balance_change, transaction_id του πίνακα item, user_id του πίνακα transaction_users, καθώς και το πεδίο της ονομασίας τοποθεσίας των συναλλαγών και μεταφορών. Τα κύρια κλειδιά αποτελούν ήδη ευρετηριασμένα πεδία. Ακόμα και στα σύνθετα κύρια κλειδιά (composite primary keys), όπως ο σύνθετος συνδυασμός κύριου κλειδιού (group_id, user_id) του πίνακα groups_users, γίνεται αυτόματη ευρετηρίαση των στηλών τους, αλλά με βελτιστοποίηση υπέρ των συνδυασμών που βρίσκεται πιο αριστερά στο ευρετήριο. Για παράδειγμα, ο σχεδιαστής αιτήματος (Query planner) εκτελεί βελτιστοποιημένα αιτήματα προς τη βάση για τον πίνακα groups_users όταν εκτελείται αναζήτηση με κριτήριο την πρώτη στήλη της ομάδας (group_id), καθώς και για τον συνδυασμό της πρώτης με την επόμενη στήλη του χρήστη (user_id). Σημειώνεται ότι η στήλη του χρήστη παραμένει

μέρος του ευρετηρίου, αλλά οι δυνατότητες βελτιστοποίησης περιορίζονται. Αυτό αντιμετωπίζεται αν προστεθεί σε ευρετήριο ως μοναδική στήλη ή ως το πρώτο μέρος συνδυασμού στηλών. Επιπλέον, με χρήση index επιτυγχάνεται και η μοναδικότητα των διευθύνσεων ηλεκτρονικού ταχυδρομείου στον πίνακα των τοπικών χρηστών. Η PostgreSQL, συγκρίνει τα αλφαριθμητικά με ευαισθησία πεζών-κεφαλαίων. Για να αποφευχθούν οι διπλότυπες διευθύνσεις λόγω αλλαγών της κεφαλαιοποίησης των γραμμάτων, το index βασίζεται στην πεζή καταγραφή των διευθύνσεων. (PostgreSQL: Documentation 10: 11.3. Multicolumn Indexes, n.d.)

Όσον αφορά τον χειρισμό των χρονοσημάνσεων, θα πρέπει να ληφθούν υπόψιν οι διαφορετικές ζώνες ώρας. Κοινή πρακτική αποτελεί η χρήση της Συντονισμένης Παγκόσμιας Ώρας (UTC). Επιλέγοντας τον τύπο δεδομένων «timestamp with time zone», η εσωτερική τιμή της ημερομηνίας αποθηκεύεται σε UTC, αλλά διατηρείται και η ζώνη ώρας και χρησιμοποιείται για μορφοποίηση της εξόδου στην τοπική ώρα. Αφού οι τιμές παρέχουν τύπο UTC θα μπορούσε να χρησιμοποιηθεί και η ο τύπος «timestamp without time zone», όμως για λόγους ανάκτησης των δεδομένων επιλέχθηκε ο πρώτος. Όπως έχει αναφερθεί, το TypeORM, καθώς διαβάζει τα δεδομένα, τα μετατρέπει σε JavaScript αντικείμενα και η απουσία της ζώνης ώρας προκαλούσε τον κατασκευαστή Date της JavaScript να εφαρμόζει την ζώνη ώρας του συστήματος.

Τέλος, στη βάση δεδομένων μπορούν να προκύψουν συνθήκες συνθηκών συναγωνισμού (racing conditions). Αυτές συνήθως αντιμετωπίζονται με χρήση συναλλαγών της βάσης (database transactions) με αυστηρό επίπεδο απομόνωσης (isolation level) ή με κάποιο είδους έμμεσο ή άμεσο κλείδωμα.

3.1.2 Αφυπνιζόμενα προγράμματα και συναρτήσεις

Τα αφυπνιζόμενα προγράμματα ή triggers, είναι αντικείμενα της βάσης δεδομένων τα οποία ενεργοποιούνται όταν συμβαίνει κάποιο γεγονός. Ως γεγονότα εννοούνται οι ενέργειες INSERT, UPDATE και DELETE, οι οποίες είναι ικανές να μεταβάλλουν τα δεδομένα. Η ενεργοποίηση των triggers στην PostgreSQL επιφέρει την εκτέλεση μιας συνάρτησης σε ένα συγκεκριμένο πίνακα. (Σκουρλάς, 2019)

Στην εφαρμογή γίνεται χρήση triggers σε τρία σημεία. Τα δύο πρώτα αφορούν την ενημέρωση του πεδίου modified_on των πινάκων transaction και transfer, κατά την εκτέλεση της ενέργειας UPDATE σε κάποια καταχώρησή τους. Επιπλέον, γίνεται

χρήση trigger για τον έλεγχο του μέγιστου αριθμού μελών μιας ομάδας. Σε περίπτωση που το όριο ξεπεραστεί, παρουσιάζεται μια εξαίρεση, ακυρώνοντας τη διαδικασία.

3.1.3 Αρχικοποίηση βάσης δεδομένων

Η δημιουργία όλων των απαραίτητων αντικειμένων στη βάση γίνεται με χρήση ενός script που εμπεριέχει όλες τις κατάλληλες εντολές για αυτό το σκοπό. Το script εκτελείται μια φορά πριν την πρώτη εκκίνηση του εξυπηρετητή.

3.2 Εφαρμογή

3.2.1 Περιγραφή λειτουργίας εφαρμογής εξυπηρετητή

Ο εξυπηρετητής είναι μια σχετικά απλή εφαρμογή επιφορτισμένη με τον έλεγχο ορθότητας και τη διατήρηση των δεδομένων, καθώς και με το συντονισμό του συγχρονισμού τους με τις συσκευές των χρηστών. Αναπτύχθηκε ως εργαλείο με αποκλειστικό σκοπό τη συμβολή στον διαμοιρασμό των χρεών μιας παρέας που παράγονται από τις μεταξύ τους συναλλαγές. Δεν αποτελεί επίσημο τραπεζικό εργαλείο και δεν τα δεσμεύει τα μέλη της με οποιοδήποτε τρόπο.

Η εφαρμογή χρησιμοποιεί ένα νόμισμα, το ευρώ και δεν παρέχονται μετατροπές νομισμάτων. Επιλέχθηκε να αναπτυχθεί με βάση την αγγλική γλώσσα, οπότε τα μηνύματα του εξυπηρετητή είναι στα αγγλικά. Η αποστολή ή η λήψη δεδομένων χρόνου, καθώς και κάθε λειτουργία που χειρίζεται ανάλογα δεδομένα στηρίζεται στο πρότυπο χρόνου UTC.

Παρακάτω περιγράφονται καίρια κομμάτια της εφαρμογής.

3.2.2 Domain Models

Οι οντότητες είναι αντικείμενα που αναπαριστούν τα μοντέλα. Οι domain οντότητες αντιπροσωπεύουν αντικείμενα του domain model και περιέχουν συμπεριφορά και ιδιότητες, ενώ οι οντότητες του persistence model είναι αντικείμενα, η κατάσταση (state) των οποίων στη συνέχεια αποθηκεύεται σε μια σχεσιακή βάση δεδομένων. Οι τελευταίες οντότητες θυμίζουν τη δομή που έχει ο πίνακας που τους αντιστοιχεί. Οι οντότητες του εξυπηρετητή χρησιμοποιούνται τόσο από το domain model, όσο και από το persistence model, οπότε είναι κοινές, κυρίως επειδή η πλευρά του εξυπηρετητή επιφορτίζεται με την αποθήκευση των δεδομένων και δεν εκτελεί

επεξεργασία σε αυτά πέρα από τον αρχικό έλεγχό (validation) τους. Για κάθε οντότητα αντιστοιχεί ένα αμετάβλητο Repository με βασικά αιτήματα προς τη βάση δεδομένων και δημιουργείται αυτόματα από την βιβλιοθήκη TypeORM. Σε περίπτωση που χρειαστεί η ανάγκη συγγραφής πιο πολύπλοκων αιτημάτων, δίνεται η δυνατότητα δημιουργίας επιπλέον Repositories από τον προγραμματιστή, βελτιώνοντας την επαναχρησιμότητα. (Design a microservice domain model, 2020)

Επίσης, γίνεται χρήση DTOs, τα οποία όπως υποδεικνύει και το όνομά τους είναι αντικείμενα και επιφορτίζονται με τη μεταφορά δεδομένων μεταξύ διαδικασιών. Πρόκειται ξεκάθαρα για δοχεία δεδομένων και δεν περιέχουν λογική, πέρα από μεθόδους που αφορούν την αποθήκευση, ανάκτηση και μετατροπή (για τη μεταφορά) των δεδομένων τους. Χρησιμοποιούνται κυρίως για την συγκέντρωση δεδομένων από απομακρυσμένες διεπαφές ώστε μειωθεί το χρονικό κόστος που συνεπάγονται πολλαπλές κλήσεις. Στην εφαρμογή χρησιμοποιούνται για να περιορίσουν τα δεδομένα που αποστέλλει κάθε endpoint. Ο περιορισμός έχει συνήθως σκοπό είτε τη μείωση του μεγέθους της απάντησης, είτε την απόκρυψη πληροφορίων. (Data transfer object, 2021)

Αναφέρθηκε ότι οι domain οντότητες της εφαρμογής βασίζονται στο persistence model, οπότε και οι ιδιότητές τους είναι τα πεδία των πινάκων που τους αντιστοιχούν. Τα περισσότερα από αυτά περιέχουν και κάποια επιπλέον πεδία για τις μεταξύ τους σχέσεις, καθώς και ένα κατασκευαστή. Στις ιδιότητες αντιστοιχούν ανάλογοι decorators για την περιγραφή του mapping και αφορούν τη βάση δεδομένων. Αυτοί περιγράφουν τον τύπο, τους περιορισμούς, όπως το μέγιστο μήκος των αλφαριθμητικών, τις σχέσεις μεταξύ των οντοτήτων, τις πιθανές μετατροπές κατά την εγγραφή και ανάγνωση των πεδίων από τη βάση δεδομένων κ.α.. Οι decorators είναι λέξεις κλειδιά που έχουν ως πρόθεμα το σύμβολο «@» και η λέξη που ακολουθεί αποτελεί μια συνάρτηση με τις πληροφορίες τους, η κλήση της οποίας γίνεται κατά την εκτέλεση (runtime). Αποτελούν ένα ειδικό είδος δήλωσης και μπορούν να προσδεθούν σε επίπεδο κλάσης, μεθόδου, ιδιότητας ή παραμέτρου. (Decorators, n.d.)

Όσον αφορά, τις επιπλέον ιδιότητες των σχέσεων των οντοτήτων, αυτές υπάρχουν για να εκφράσουν το δεύτερο μέρος της σχέσης. Με βάση αυτά που περιεγράφηκαν ανωτέρω, οι domain οντότητες θα είναι:

- User: Γενικός χρήστης συστήματος και αποτελεί ξένο κλειδί από εξαρτώμενα μέλη

- FBUser: Λογαριασμός χρήστη για είσοδο με χρήση Facebook
- LocalUser: Τοπικός λογαριασμός χρήστη για είσοδο
- Group: Ομάδα
- GroupUser: Μέλος της ομάδας
- Transaction: Συναλλαγή
- TransactionUser: Χρήστης που έλαβε μέρος στη Συναλλαγή
- Item: Αντιπροσωπεύει κάποιο αντικείμενο της Συναλλαγής
- Transfer: Μεταφορά
- BalanceChange: Καταγράφει συμβάντα που κατά κύριο λόγο αφορούν τη μεταβολή των λογιστικών υπολοίπων των μελών

Επίσης χρησιμοποιούνται κάποια επιπλέον αντικείμενα που επιτρέπουν την επαναχρησιμότητα του κώδικα και τον καθιστούν ευανάγνωστο. Αυτά είναι το Location και το Content. Το πρώτο χρησιμοποιείται ως embedded κλάση σε κάποια οντότητα, ενθυλακώνοντας πλήθος ιδιοτήτων. Το δεύτερο αντικείμενο εκμεταλλεύεται τα πλεονεκτήματα της κληρονομικότητας και χρησιμοποιείται ως γονική κλάση, περιέχοντας τις κοινές ιδιότητες. Όλα τα παραπάνω γίνονται εμφανή στην κλάση Transaction που παρουσιάζεται παρακάτω.

```
@Entity({ name: "transaction" })
export class Transaction extends Content {
    @PrimaryGeneratedColumn({ type: 'bigint' })
    id: string;

    @ManyToOne(() => Group, group => group.transactions, { onDelete: 'CASCADE' })
    @JoinColumn({ name: 'group_id' })
    group: Group;

    @Column({ length: 30 })
    title: string;

    @Column({ length: 200, nullable: true })
    description: string;

    @Column({ name: 'total_price', type: 'bigint' })
    totalPrice: string;

    @VersionColumn({ default: 1 })
    version: number;


    @OneToMany(() => Item, items => items.transaction)
    items: Item[];

    @OneToMany(() => TransactionUser, user => user.transaction)
    users: TransactionUser[];

    public constructor(init?: Partial<Transaction>) {
        super();
        Object.assign(this, init);
    }
}
```

Εικόνα 3.2: Domain κλάση Transaction (Συναλλαγή)

Σημειώνεται ότι οι Συναλλαγές και οι Μεταφορές αποτελούν βασικά δεδομένα της εφαρμογής οπότε θα γίνεται αναφορά σε αυτές και με τον όρο Περιεχόμενο.

3.2.3 Υλοποίηση Αρχιτεκτονικής

Όπως αναφέρθηκε η οργάνωση του κώδικα βασίζεται στο δόγμα package by feature. Αυτό σημαίνει ότι κάθε χαρακτηριστικό φιλοξενείται σε αυτόνομο πακέτο μαζί με τα αρχεία του. Άλλα αρχεία γενικής φύσης, όπως είναι το αρχείο main, που περιέχουν τις παραμέτρους εκκίνησης του εξυπηρετητή, βρίσκονται στον κύριο (root)

φάκελο. Τα βασικά χαρακτηριστικά εφαρμογής, διαθέτουν τουλάχιστον ένα ή περισσότερα, Controller, Service, Repository ή Module, καθώς και άλλα προαιρετικά αρχεία που μπορεί να χρειαστούν αναλόγως των αναγκών και της πολυπλοκότητας του χαρακτηριστικού. Αυτά τα βασικά χαρακτηριστικά της, καθώς και οι συνοπτικές εργασίες που περιλαμβάνουν είναι:

- Χρήστης (πακέτο auth): Σύνδεση, εγγραφή, αποσύνδεση, διαχείριση των ενεργών συνεδριών, καθώς και ενημέρωση των πληροφοριών του συνδεδεμένου χρήστη
- Χρήστης (πακέτο user): Πληροφορίες γενικού χρήστη συστήματος
- Ομάδα (πακέτο group): Δεδομένα ομάδας και μελών, συγχρονισμός, συμμετοχή σε ή αποχώρηση από μια ομάδα, μεταβολή ρυθμίσεων ή πληροφοριών και αξιολόγηση των AutoEps που παρέχονται από τα κινητά των χρηστών. Τα τελευταία αποτελούν αυτοματοποιημένες εντολές μεταφοράς και συμβαίνουν όταν μέρος του δευτερεύοντος υπόλοιπου καθίσταται πληρωτέο και έτσι μεταφέρεται στο πρωτεύων
- Συναλλαγές (πακέτο transaction): Λήψη παλαιότερων Συναλλαγών, δημιουργία νέων, ανανέωση ή διαγραφή τους
- Μεταφορές (πακέτο transfer): Μοιράζεται παρόμοιες ανάγκες με τις Συναλλαγές, μόνο που αφορούν τις Μεταφορές

Όπως αναφέρθηκε τα δεδομένα παρέχονται με κλήσεις στους αντίστοιχους Controllers. Οι Controllers της εφαρμογής είναι οι AuthController, GroupController, TransactionController και TransferController. Παρακάτω παρουσιάζονται κάποια αποσπάσματα κώδικα για το χαρακτηριστικό του Χρήστη. Αρχικά, ακολουθούν δύο από τις μεθόδους του AuthController και συγκεκριμένα οι κλήσεις για την σύνδεση και τον έλεγχο διαθεσιμότητας της διεύθυνσης ηλεκτρονικού ταχυδρομείου.

```
@Controller()
export class AuthController {
  constructor(
    private authService: AuthService,
    private sessionSerializer: SessionSerializer
  ) { }

  /**
   * If credentials were wrong, Passport middleware will disrupt (throw) login
   * procedure and the method will not run. Runs only after successful login.
   */
  @Post('auth/login')
  @HttpCode(200)
  @UseGuards(LoginGuard)
  async login(@Request() req, @Session() sess) {
    this.authService.successfulLogin(sess.passport.user.id, req.sessionID);
    return req.user;
  }

  @Post('auth/checkEmail')
  @HttpCode(200)
  async checkEmail(
    @Request() req, @Session() sess, @Body('email', EmailPipe) email: string
  ): Promise<{ emailInUse: boolean }> {
    return {
      emailInUse: !await this.authService.isEmailAvailableForSignup(email)
    };
  }
}
```

Εικόνα 3.3: Απόσπασμα κώδικα από τον AuthController

Αρχικά εμφανίζεται ο decorator Controller σε επίπεδο κλάσης. Αυτός μπορεί να περιέχει επιλογές σχετικά με αυτόν, κυριότερη από τις οποίες είναι η διαδρομή (path). Αυτή ορίζει ότι όλες οι μέθοδοι του Controller βρίσκονται στην ίδια διαδρομή. Κάθε συνάρτηση εντός του Controller θα έχει ένα decorator σε επίπεδο μεθόδου με την HTTP μέθοδο και τη διαδρομή της ως παράμετρο. Το πλήρης διαδρομή της μεθόδου είναι ο συνδυασμός και των δύο. Η χρήση του decorator HttpCode είναι για τη δήλωση του επιστρεφόμενου HTTP κωδικού όταν το αίτημα εκτελεστεί επιτυχώς. Ο κωδικός που επιστρέφεται και στις δύο μεθόδους είναι ο 200, έναντι του προεπιλεγμένου κωδικού για μέθοδο Post που είναι το 201. Αυτό συμβαίνει επειδή η HTTP μέθοδος Post συνήθως δημιουργεί κάποιο πόρο και η επιτυχής εκτέλεση

τέτοιας ενέργειας δηλώνεται με τον κωδικό 201. Στις παραπάνω περιπτώσεις δεν συμβαίνει κάτι ανάλογο, οπότε και επιστρέφεται ο γενικός κωδικός επιτυχίας. Ο decorator UseGuards αφορά την άδεια πρόσβασης αυτής της διαδρομής (URI) και περιγράφεται παρακάτω. Οι παράμετροι της συνάρτησης επιλέγονται από τον προγραμματιστή και συμπληρώνονται (inject) από το Framework με χρήση των κατάλληλων decorators. Αυτές μπορεί να είναι το αντικείμενο Request που περιλαμβάνει λεπτομέρειες σχετικά με το αίτημα ή το αντικείμενο Session που περιλαμβάνει τις πληροφορίες της συνεδρίας. Μπορούν να χρησιμοποιηθούν και άλλοι παρόμοιοι decorators που παρέχουν query παραμέτρους ή το περιεχόμενο του body, ενώ ταυτόχρονα μπορούν να εκτελούν και επικύρωση των δεδομένων πριν την προώθηση του αιτήματος στο επίπεδο Service. Αυτό συμβαίνει στην δεύτερη μέθοδο, όπου το σώμα περιλαμβάνει ένα αντικείμενο που περιέχει το email και το επικυρώνεται με χρήση του EmailPipe. Η επικύρωση διαθέτει ξεχωριστό υποκεφάλαιο με περισσότερες λεπτομέρειες σχετικά με τη χρηστικότητα των Pipes.

Σκοπός του controller είναι να διαμορφώσει τις διαδρομές και να επικυρώσει τα δεδομένα. Στη συνέχεια καλεί τις κατάλληλες μεθόδους του Service, το οποίο και δηλώνεται στον κατασκευαστή του Controller μέσω Dependency Injection.

```
@Injectable()
export class AuthService {
    private readonly logger = new Logger(AuthService.name);
    constructor(
        private usersService: UsersService,
        private userRepository: UserRepository,
        private localUserRepository: LocalUserRepository,
        private mongoRepository: MongoRepository,
        private connection: Connection) { }

    async isEmailAvailableForSignup(email: string): Promise<boolean> {
        const localUser: LocalUser =
        await this.localUserRepository.findByLocalEmail(email);
        return localUser == null;
    }
}
```

Εικόνα 3.4: Απόσπασμα κώδικα από το AuthService

Τα Services περιέχουν τη λογική, ώστε να εκτελεστεί κάποια εργασία. Το παραπάνω Service που αφορά το παρών χαρακτηριστικό (AuthService), διαθέτει τον decorator Injectable σε επίπεδο κλάσης. Αυτό συμβαίνει για να μπορεί να δηλωθεί στο Module του χαρακτηριστικού ως Provider, ώστε να είναι δυνατή η χρήση του Service μέσω Dependency Injection. Στον κατασκευαστή του δηλώνονται με παρόμοιο τρόπο άλλα αντικείμενα, όπως άλλα Services ή Repositories. Στο απόσπασμα του AuthService, απεικονίζεται η υλοποίηση της μεθόδου που καλείται από το δεύτερη συνάρτηση του αποσπάσματος του AuthController. Τα Services της εφαρμογής είναι τα AuthService, FacebookService, GroupService, TransactionService και TransferService.

Για χρήση του Dependency Injection απαιτείται η δήλωση ενός Module που δίνει στο Framework τις πληροφορίες που χρειάζεται. Το Module για το παρών χαρακτηριστικό είναι το AuthModule και παρουσιάζεται παρακάτω.

```
@Module({
  imports: [
    UsersModule, PassportModule, FacebookModule,
    TypeOrmModule.forFeature([LocalUserRepository]), MongoModule
  ],
  providers: [AuthService, SessionSerializer, LocalStrategy, FacebookStrategy],
  controllers: [AuthController],
  exports: [AuthService]
})
export class AuthModule {}
```

Εικόνα 3.5: Ο κώδικας του AuthModule

Το πεδίο των imports, εισάγει άλλα Modules που εξάγουν Providers, οι οποίοι χρησιμοποιούνται από αυτό το Module. Σε αυτούς συμπεριλαμβάνονται και η σύνδεση με τις βάσεις δεδομένων. Οι providers αποτελούν αντικείμενα που αρχικοποιούνται (instantiated) από το Framework και μπορούν να χρησιμοποιηθούν μέσω Dependency Injection. Οι controllers περιλαμβάνουν τους controllers του Module. Τέλος, το πεδίο exports περιέχει τους Providers προς εξαγωγή. Η εφαρμογή διαθέτει τα Modules των βασικών χαρακτηριστικών, AuthModule, UserModule, FacebookModule, GroupModule, TransactionModule και TransferModule. Επιπλέον, διαθέτει τα Modules, AppModule, DBModule και MongoModule. Το πρώτο αφορά το κεντρικό Module της εφαρμογής, όπου εκεί συμπληρώνονται όλα τα Modules που

χρησιμοποιούνται. Τα υπόλοιπα αφορούν τα Modules των βάσεων PostgreSQL και MongoDB. Περιλαμβάνουν τη διαμόρφωσή (configuration) τους ή/και απαραίτητο κώδικα για την επίτευξη σύνδεσης.

3.2.4 Χρήστες, αυθεντικοποίηση και εξουσιοδότηση

Το σύνολο των services του εξυπηρετητή, πέρα από εκείνα της εγγραφής και της εισόδου, είναι προστατευμένα και η πρόσβαση σε αυτά απαιτεί αυθεντικοποίηση. Η αυθεντικοποίηση επιτυγχάνεται από τη σύνδεση των χρηστών, ενώ αυτή προϋποθέτει την εγγραφή τους στο σύστημα.

Η εγγραφή επιτυγχάνεται με την κλήση του αντίστοιχου service και παρέχοντάς του ένα DTO τύπου CreateUserDto. Αυτό το αντικείμενο περιέχει όλες τις απαραίτητες πληροφορίες για την εγγραφή. Αυτές είναι η διεύθυνση ηλεκτρονικού ταχυδρομείου, ο κωδικός πρόσβασης και ένα ονοματεπώνυμο. Επισημαίνεται ότι κάθε μία από αυτές μπορεί να αλλάξει σε μεταγενέστερο χρόνο καλώντας το αντίστοιχο service. Η μοναδικότητα της διεύθυνσης ηλεκτρονικού ταχυδρομείου ελέγχεται και από το επίπεδο εφαρμογής, αλλά καθώς αποτελεί σημείο συνθηκών ανταγωνισμού, η βάση δεδομένων είναι το επίπεδο που την εγγυάται. Ο κωδικός πρόσβασης κωδικοποιείται με τη μέθοδο bcrypt. Η κωδικοποίηση είναι μιας κατεύθυνσης και αυτό εξασφαλίζει ότι το μόνο άτομο που τον γνωρίζει είναι ο χρήστης. Η μέθοδος bcrypt επιλέχθηκε καθώς χρησιμοποιείται ευρέως, εξαιτίας των ιδιοτήτων της. Ενσωματώνει εκ προεπιλογής τη χρήση salt για προστασία ενάντια σε επιθέσεις τύπου rainbow table. Παράλληλα, εισάγει έναν μεταβλητό παράγοντα εργασίας (work factor) με σκοπό την προστασία από επιθέσεις τύπου brute force, καθώς εξισορροπεί την αυξανόμενη ισχύ των σύγχρονων υπολογιστών με το υπολογιστικό κόστος της συνάρτησης κωδικοποίησης. Η εγγραφή συνεπάγεται την δημιουργία ενός τοπικού λογαριασμού για σκοπούς αυθεντικοποίησης και ενός χρήστη συστήματος που χρησιμοποιείται για όλους τους υπόλοιπους.

Επιπλέον, δίνεται η δυνατότητα σύνδεσης μέσω λογαριασμού από υπηρεσία τρίτου, όπως είναι τα μέσα κοινωνικής δικτύωσης ή υπηρεσίες ηλεκτρονικής αλληλογραφίας. Η εφαρμογή υποστηρίζει σύνδεση μέσω λογαριασμού Facebook. Για να καταστεί δυνατή η αξιόπιστη επικοινωνία μεταξύ της εφαρμογής και του Facebook, μας παρέχεται ένα αναγνωριστικό εφαρμογής και ένας μυστικός κωδικός. Ο εξυπηρετητής της εφαρμογής διαθέτει συγκεκριμένο service για την υποστήριξη

αυτού του χαρακτηριστικού. Το service, πριν ολοκληρώσει την εκκίνησή του μαζί με τις υπόλοιπες υπηρεσίες του εξυπηρετητή, εκτελεί κλήση σε εξυπηρετητή του Facebook αποστέλλοντας το αναγνωριστικό και τον μυστικό κωδικό της εφαρμογής, ώστε να αποκτηθεί το app access token. Αυτό αποστέλλεται από τον εξυπηρετητή της εφαρμογής σε κλήσεις προς τον εξυπηρετητή του Facebook, ώστε να ληφθούν πληροφορίες σχετικά με τον χρήστη που προσπαθεί να συνδεθεί. Συγκεκριμένα, όταν ο χρήστης πραγματοποιήσει είσοδο στη συσκευή, λαμβάνει και στη συνέχεια αποστέλλει στον εξυπηρετητή ένα user access token, καθώς και το αναγνωριστικό του. Το αναγνωριστικό είναι μοναδικό για την εφαρμογή και ισχύει για κάθε σύνδεσή του, ακόμα και αν αφαιρεθούν τα δικαιώματα πρόσβασης και δοθούν ξανά, το αναγνωριστικό δεν θα αλλάξει. Για να επαληθεύσουμε την εγκυρότητά του access token του χρήστη, ο εξυπηρετητής επικοινωνεί με τις υπηρεσίες του Facebook, αποστέλλοντάς το μαζί με το app access token. Η απάντηση που λαμβάνει είναι είτε κάποια περιγραφή σφάλματος, οπότε και ακυρώνεται η διαδικασία σύνδεσης, είτε οι πλήρης πληροφορίες του token, όπως το πότε λήγει, τα δικαιώματα πρόσβασης στις πληροφορίες του χρήστη κτλ. Έπειτα από επιτυχή επαλήθευση, εκτελούμε αίτημα λήψης των πληροφοριών του χρήστη που περιλαμβάνουν τα πεδία που ζητήθηκαν. Αυτά είναι το αναγνωριστικό (εκ προεπιλογής) και το όνομα. Η διαχείριση λαθών στη δεύτερη κλήση είναι πιο πολύπλοκη, καθώς υπάρχουν διαφορετικοί επιστρεφόμενοι κωδικοί σφάλματος που πρέπει να αντιμετωπίζονται διαφορετικά, αλλά και να μεταφράζονται σε κατανοητά για τους χρήστες μηνύματα. Ένα μεγάλο ποσοστό επιλύεται με επανάληψη της σύνδεσης, ενώ για τους υπόλοιπους που χρήζουν περαιτέρω προσοχής, οι χρήστες λαμβάνουν μηνύματα σφάλματος του εξυπηρετητή. Αφού επαληθευτεί η ταυτότητα του χρήστη, παρομοίως δημιουργείται ένας λογαριασμός τύπου Facebook και ο χρήστης συστήματος. Σημειώνεται ότι με το πραγματοποιηθεί σύνδεση μέσω Facebook, η σύνδεση στο σύστημα στηρίζεται στον τοπικό λογαριασμό τύπου Facebook, οπότε η λήξη του user access token ή αφαίρεση της πρόσβασης της εφαρμογής στο λογαριασμό Facebook του χρήστη, έχει ως αποτελέσματα την αδυναμία λήψης των πιο πρόσφατων πληροφοριών του χρήστη από το Facebook. Δεν επηρεάζεται οτιδήποτε άλλο στην εφαρμογή, όπως η είσοδος.

Η αυθεντικοποίηση γίνεται με χρήση τρίτης βιβλιοθήκης ονόματι Passport, ενώ η προστασία των endpoints από μη εξουσιοδοτημένα άτομα γίνεται μέσω της συνεργασίας του Nest framework και της βιβλιοθήκης, χρησιμοποιώντας τους

decorators τύπου UseGuards στις μεθόδους των Controllers ή ακόμα και στο επίπεδο των Controllers. Αυτοί επιλέγουν αν ένα αίτημα θα εξυπηρετηθεί, ανάλογα με κάποιες συνθήκες. Στην εφαρμογή διατίθενται δύο τύποι UseGuards, ο LoginGuard και ο AuthenticatedGuard. Ο πρώτος είναι υπεύθυνος για την εκτέλεση της εισόδου, όπου δικαίωμα πρόσβασης έχουν όλοι οι χρήστες, ενώ ο δεύτερος για την επαλήθευση της σύνδεσης και του δικαιώματος πρόσβασης των χρηστών στους αιτούμενους πόρους.

Η επιτυχής εγγραφή ή σύνδεση, ανεξαρτήτως παρόχου, δημιουργεί μια συνεδρία (session) και η απάντηση περιέχει τις πληροφορίες του συνδεδεμένου χρήστη μαζί με ένα cookie, το οποίο οι χρήστες θα χρησιμοποιούν για να αποκτήσουν πρόσβαση στα προστατευμένα endpoints. Οι πληροφορίες περιλαμβάνονται σε ένα αντικείμενο, ένα instance της διεπαφής LoggedInUser, που περιέχει τα πεδία με το αναγνωριστικό χρήστη, το όνομα και τον πάροχο σύνδεσης, ενώ το πεδίο της διεύθυνσης ηλεκτρονικού ταχυδρομείου συμπληρώνεται στην περίπτωση χρήσης του τοπικού παρόχου. Ο μέγιστος αριθμός ενεργών συνεδριών είναι τρεις. Πραγματοποίηση σύνδεσης ξεπερνώντας τον μέγιστο αριθμό ενεργών συνεδριών προκαλεί την έξοδο από τη συνεδρία που λήγει πιο σύντομα.

Τα cookies τα διαχειρίζεται η βιβλιοθήκη express-session. Ο λόγος που επιλέχθηκε η λύση των cookies είναι ώστε να διατηρούνται όλες οι ενεργές συσκευές του χρήστη σε ένα σημείο με σκοπό την πιθανή απομακρυσμένη αποσύνδεσή τους. Οι εφαρμογές REST κάνουν χρήση stateless τεχνολογιών, όπως είναι τα JWTs, για να πραγματοποιήσουν ελέγχους ταυτότητας και δικαιωμάτων που όμως κάνουν αδύνατη την απομακρυσμένη αποσύνδεση. Οι συνεδρίες αποθηκεύονται στο χώρο αποθήκευσής των συνεδριών (session store) που υλοποιείται με μια NoSQL βάση δεδομένων, τη MongoDB. Η μη σχεσιακές βάσεις δεομένων αποτελούν δημοφιλή επιλογή για τέτοιου είδους εφαρμογές καθώς εκτελούν γρήγορη ανάγνωση δεδομένων και συνήθως δεν απαιτείται σχεσιακή σχεδίαση. Η γρήγορη ανάγνωση είναι καίριας σημασίας, καθώς από το cookie που στέλνεται με κάθε αίτημά του πελάτη διαβάζεται το session id, το οποίο και στη συνέχεια αναζητείται στη βάση δεδομένων. Το cookie αποστέλλεται και σε κάθε απάντηση του διακομιστή, καθώς μπορεί να ανανεωθεί ή να αλλάξει από εκείνον. Αν η συνεδρία έχει λήξει ή δεν υπάρχει, λόγω αποσύνδεσης, τότε η αυθεντικοποίηση και ακολούθως η εξουσιοδότηση αποτυγχάνει, καθιστώντας την πρόσβαση στα endpoints αδύνατη. Αντ’ αυτού οι χρήστες λαμβάνουν μήνυμα σφάλματος με HTTP κωδικό τύπου 401

(Unauthorized). Η εξουσιοδότηση επίσης αποτυγχάνει αν ο χρήστης δεν έχει δικαίωμα πρόσβασης, όπως είναι η προσπάθεια πρόσβασης σε πόρους άλλων χρηστών. Η συνεδρία περιέχει διάφορες πληροφορίες σχετικά με τη σύνδεση και το cookie. Οι πληροφορίες της σύνδεσης περιλαμβάνουν το αναγνωριστικό του χρήστη, τον πάροχο της σύνδεσης, ημερομηνία και ώρα σύνδεσης, πληροφορίες συσκευής (μοντέλο, έκδοση λειτουργικού συστήματος), καθώς και ένα τυχαία παραγόμενο αλφαριθμητικό που αποτελεί το αναγνωριστικό συσκευής. Αυτές οι πληροφορίες προωθούνται στο SessionSerializer του Passport, για να μετατραπούν σε JSON και στη συνέχεια να αποθηκευτούν μαζί με τις υπόλοιπες πληροφορίες της συνεδρίας. Τα αναγνωριστικά συσκευής αντικαθιστούν τα αναγνωριστικά συνεδρίας, ώστε να μην εκθέτονται σε κοινή θέα όταν οι χρήστες ζητούν τη λίστα ενεργών συνδέσεων. Όπως και στις σχεσιακές βάσεις, πεδία που αποτελούν κριτήριο αναζήτησης ευρετηριάζονται. Ακριβώς αυτό συμβαίνει και στα πεδία id και deviceId.

Η βιβλιοθήκη που παράγει τα cookies διαθέτει κάποιες ρυθμίσεις για αυτά, όσο και για το χώρο αποθήκευσής τους. Οι ιδιότητες session store είναι οι:

- store: Το instance του αποθηκευτικού χώρου των συνεδριών. Χρησιμοποιείται το πακέτο connect-mongo
- name: Το όνομα του αναγνωριστικού της συνεδρίας
- secret: Αλφαριθμητικό που χρησιμοποιείται για να υπογραφεί το cookie της συνεδρίας (κωδικοποίηση-αποκωδικοποίηση)
- resave: Με αρνητική τιμή αποτρέπεται να εκτελεστεί ξανά αποθήκευση της συνεδρίας στο session store. Ο εμπειρικός κανόνας ορίζει ότι αν το session store υποστηρίζει τη μέθοδο touch, τότε μπορεί να απενεργοποιηθεί (τιμή false). Το επιλεγμένο store την υποστηρίζει, οπότε και απενεργοποείται
- saveUninitialized: Δεν δημιουργείται κενό cookie από το πρώτο αίτημα, αλλά μόνο όταν πραγματοποιηθεί σύνδεση ώστε να περιέχει και τις πληροφορίες που την αφορούν
- rolling: Η αποστολή της συνεδρίας με κάθε αίτημα είναι ενεργοποιημένη, επαναθέτοντας την ημερομηνία λήξης του cookie
- proxy: Εμπιστεύεται τον αντίστροφο διακομιστή μεσολάβησης όταν θέτει ασφαλή cookies μέσω της επικεφαλίδας «X-Forwarded-Proto»

Ιδιότητες cookie:

- httpOnly: Θετική τιμή αποτρέπει την πρόσβαση της JavaScript στα cookies.

- secure: Ορίζει ότι το cookie στέλνεται στον εξυπηρετητή μόνο μέσω ασφαλούς, κρυπτογραφημένης σύνδεσης (HTTPS)
- sameSite: Με την τιμή true/strict ορίζεται ότι το cookie αποστέλλεται μόνο όταν το domain του είναι ίδιο με το domain του αιτήματος
- maxAge: Ο μέγιστος χρόνος ζωής του cookie σε χιλιοστά του δευτερολέπτου

Οι περισσότερες ιδιότητες για τα cookies αφορούν εφαρμογές που εκτελούνται σε προγράμματα περιήγησης ή τεχνολογίες που η εφαρμογή του πελάτη δεν χρησιμοποίει. Παρόλα αυτά οι ρυθμίσεις θέτονται στη μέγιστη δυνατή ασφάλεια. Σημειώνεται ότι επειδή η εφαρμογή δεν εκτελείται σε προγράμματα περιήγησης δεν απαιτείται και η λήψη μέτρων για κάποιους τύπους επιθέσεων, όπως το CSRF.

Εικόνα 3.6: Το αντικείμενο της συνεδρίας στη βάση

Η αποσύνδεση ή έξοδος πραγματοποιείται διαγράφοντας τη συνεδρία στον εξυπηρετητή και αφαιρώντας το cookie από την απάντηση που αποστέλλει ο εξυπηρετητής στο αίτημα του πελάτη. Η διαγραφή μιας συνεδρίας μπορεί να αφορά την παρούσα συνεδρία ή την συνεδρία κάποιας άλλης συσκευής. Η πρώτη περίπτωση αποτελεί ουσιαστικά την έξοδο του χρήστη, οπότε εκτελείται η ρουτίνα εξόδου. Στη δεύτερη περίπτωση αναζητάτε η συνεδρία με βάση το αναγνωριστικό της συσκευής στο χώρο αποθήκευσης των συνεδριών και εφόσον υπάρχει και ανήκει στον χρήστη που εκτέλεσε το αίτημα, τότε αυτή διαγράφεται. Με αυτό τον τρόπο επιτυγχάνεται η απομακρυσμένη αποσύνδεση συσκευής. Σημειώνεται ότι δεν

επιτρέπεται η διαγραφή της παρούσας συνεδρίας με χρήση του αναγνωριστικού συσκευής, επειδή όπως περιγράφηκε δεν απαιτεί απλά τη διαγραφή της συνεδρίας αλλά προβαίνει και σε επιπλέον ενέργειες.

Η διαγραφή του χρήστη αποτελεί περίπλοκη διαδικασία, καθώς ένας χρήστης πριν αποχωρήσει από την εφαρμογή, δηλαδή πριν διαγράψει τον λογαριασμού του, θα πρέπει να έχει τακτοποιήσει τα χρέη του μεταξύ των ομάδων που συμμετέχει. Το σύστημα όταν λάβει αίτημα διαγραφής λογαριασμού θα πρέπει να ελέγξει αν ο χρήστης διαθέτει υπόλοιπο διάφορο του μηδενός σε κάθε ομάδα που ανήκει. Αν ισχύει αυτή η συνθήκη στη συνέχεια διαγράφεται από μέλος κάθε ομάδας και εν τέλει διαγράφεται και ο λογαριασμός του. Παρόλα αυτά θα πρέπει να ληφθούν υπόψιν αρκετές περιπτώσεις που ευνοούν την ύπαρξη συνθηκών ανταγωνισμού. Για να αποφύγουμε την μεταβολή του υπολοίπου ανά ομάδα του χρήστη, γίνεται ανάγνωση τους σε συναλλαγή της βάσης δεδομένων με επίπεδο απομόνωσης τύπου Repeatable Read. Σε περίπτωση μεταβολής κατά τη διάρκεια της συναλλαγής, αυτή θα αποτύχει με serialization σφάλμα. Αν κατά τη διάρκεια της διαγραφής, ο χρήστης προστεθεί σε νέα ομάδα, τότε αν αυτή η αλλαγή έγινε αντιληπτή από τη συναλλαγή κατά την ανάγνωση και η ανωτέρω συνθήκη ικανοποιείται, τότε η διαγραφή λογαριασμού εκτελείται κανονικά. Σε διαφορετική περίπτωση, η συναλλαγή θα αποτύχει, καθώς η διαγραφή του χρήστη δεν είναι δυνατή λόγω εξάρτησης της οντότητας από τις ομάδες (σφάλμα ξένου κλειδιού). Η επιτυχής διαγραφή λογαριασμού συνεπάγεται και την αυτόματη αποσύνδεση του χρήστη από όλες τις ενεργές συνεδρίες, συμπεριλαμβανομένου και της παρούσης.

Όσον αφορά την εξουσιοδότηση, ο HTTP κωδικός για την ανεπάρκεια αυτής είναι ο 403 (Forbidden), ενώ για πόρους που δεν υπάρχουν αντιστοιχίζεται ο κωδικός τύπου 404 (Not Found). Επειδή τα αναγνωριστικά πεδία (id fields) των οντοτήτων είναι συνήθως σειριακά, σημαίνει ότι μπορούν εύκολα να αποτελέσουν προϊόν εικασίας από επιτιθέμενους. Παρόλο που δεν μπορούν να αποκτήσουν πρόσβαση στους πόρους, πληροφορούνται για την ύπαρξή τους. Για να αποφευχθεί το σενάριο οι χρήστες να είναι σε θέση να επαληθεύουν την ύπαρξη πόρων βάσει τυχαίων αναγνωριστικών, επιλέχθηκε η σύνθεση των δύο παραπάνω σφαλμάτων σε ένα, μέσω του γενικού HTTP κωδικού τύπου 400 (Bad Request) και ένα γενικό μήνυμα σφάλματος. Έτσι το σφάλμα οριοθετείτε ανάμεσα στη μη εύρεσή του πόρου και στην έλλειψη δικαιώματος πρόσβασης σε αυτόν, χωρίς όμως να αποσαφηνίζεται η ακριβής αιτία.

3.2.5 Ομάδες και τα μέλη τους

Οι χρήστες της ομάδας, δηλαδή τα μέλη της, έχουν πλήρη πρόσβαση στις πληροφορίες της ομάδας και των μελών της. Επιπλέον έχουν δικαίωμα να μεταβάλλουν τις πληροφορίες της ομάδας, καθώς μπορούν να αλλάξουν το όνομά της, να ενεργοποιήσουν τη δυνατότητα διαμοιρασμού της ομάδας για να προστεθούν νέα μέλη και να επιλέξουν αν είναι ενεργοποιημένη η δυνατότητα της αντικατάστασης (overwrite).

Ο διαμοιρασμός της ομάδας επιτυγχάνεται μέσω ενός δεκαψήφιου, τυχαίου αλφαριθμητικού. Αυτός παράγεται μόλις ενεργοποιηθεί η δυνατότητα διαμοιρασμού και οι χρήστες που θέλουν να λάβουν μέρος στην ομάδα τον εισάγουν στην εφαρμογή του πελάτη. Μόλις ο εξυπηρετητής λάβει αίτημα συμμετοχής, ελέγχει αν ο κωδικός αντιστοιχεί σε κάποια ομάδα και προσθέτει τον χρήστη. Η απενεργοποίηση του χαρακτηριστικού συνεπάγεται και αφαίρεση του κωδικού από την ομάδα, καθιστώντας τον άχρηστο. Όπως αναφέρθηκε ο κωδικός είναι τυχαίος, αλλά θα πρέπει να είναι μοναδικός στον πίνακα. Για το λόγο αυτό η βάση δεδομένων εμφανίζει εξαίρεση όταν υπάρχει ήδη ίδιος κωδικός και η διαδικασία επανεκκινείται μέχρι τρεις φορές.

Η ρύθμιση της αντικατάστασης αποβλέπει στο να ξεκαθαρίζει την απόφαση που θα πάρει το σύστημα όταν η ανανέωση της κατάστασης ενός πόρου αντικαθιστά μια νεότερη κατάστασή του. Δηλαδή, όταν η νέα κατάσταση του πόρου προς αντικατάσταση προήρθε από παλαιότερη κατάσταση από αυτή που έχει αποθηκευμένη ο εξυπηρετητής και έτσι ο χρήστης προσπαθεί να αλλάξει ένα πόρο για τον οποίο δεν γνωρίζει την τωρινή του κατάσταση. Η κατάσταση περιγράφεται αριθμητικά (ακέραιος) μέσω του πεδίου της έκδοσης. Επομένως, αν η νέα κατάσταση του πόρου διαθέτει έκδοση μικρότερη της αποθηκευμένης από τον εξυπηρετητή, τότε εκείνος βάσει της ρύθμισης overwrite θα αποφασίσει ποια κατάσταση του πόρου θα διατηρήσει. Στην περίπτωση που βρεθεί ασυμβατότητα εκδόσεων, ο πόρος γίνεται αποδεκτός μόνο αν η ρύθμιση είναι ενεργή.

Η δημιουργία μιας ομάδας είναι απλή διαδικασία και οι απαιτούμενες πληροφορίες είναι μόνο το όνομά της. Ο δημιουργός μιας ομάδας είναι και το πρώτο μέλος της. Η πρόσθεση επιπλέον χρηστών απαιτεί την ενεργοποίηση και τη χρήση του κωδικού

διαμοιρασμού. Οι ομάδες έχουν ως μέγιστο αριθμό μελών τα δεκαπέντε άτομα. Ο περιορισμός του μέγιστου ορίου μελών τέθηκε για τους παρακάτω λόγους:

1. Ο αλγόριθμος για την παραγωγή των διαδρομών χρέους

   Η επιλογή του αλγορίθμου για τη παραγωγή των βέλτιστων διαδρομών χρέους είναι μια απαιτητική διαδικασία, η πολυπλοκότητα της οποίας εξαρτάται από το πλήθος των μελών της ομάδας.
2. Εύκολη επεξεργασία Περιεχομένου

   Ο χρήστης που επεξεργάζεται Περιεχόμενο θα βλέπει μια λίστα με όλα τα μέλη της ομάδας. Δεν είναι επιθυμητό αυτή η λίστα να είναι πολύ μεγάλη, επειδή θα σπαταλάει χρόνο να αναζητά τα μέλη. Για παράδειγμα, μια Συναλλαγή μπορεί να έχει πολλαπλά αντικείμενα, τα οποία μπορεί να ανήκουν σε πολλούς χρήστες, οπότε η αναδρομή σε μέλη θα είναι αρκετά συχνή.
3. Η διαχείριση της ομάδας θα είναι δυσκολότερη

Ο έλεγχος για τον περιορισμό των μελών πραγματοποιείται κατά τη διαδικασία προσθήκης του χρήστη, αμέσως μετά αφού επαληθευτεί ότι ο κωδικός διαμοιρασμού ανήκει σε κάποια ομάδα. Αυτός ο έλεγχος είναι επιρρεπής σε φαινόμενα συνθηκών ανταγωνισμού με αποτέλεσμα αν αυτά δεν ληφθούν υπόψη, τότε το όριο να ξεπεραστεί. Μία προσέγγιση είναι η χρήση του αυστηρότερου επιπέδου απομόνωσης, Serializable. Με αυτό το επίπεδο, κάθε συναλλαγή της βάσης εκτελείται σειριακά, επομένως εξασφαλίζεται ότι από τη στιγμή που διαβάζεται το πλήθος των μελών της ομάδας μέχρι και την ολοκλήρωση της διαδικασίας, τα μέλη της ομάδας παραμένουν αμετάβλητα. Το μειονέκτημα είναι ότι ένα τόσο αυστηρό επίπεδο απομόνωσης είναι εις βάρος της απόδοσης, καθώς στην περίπτωση που έχουμε δύο αιτήσεις για συμμετοχή σε ομάδα, ακόμα και αν αυτές δεν είναι ίδιες, τότε η κάθε μία συναλλαγή συμμετοχής θα εκτελεστεί ξεχωριστά, απαιτώντας περισσότερο χρόνο. Η προσέγγιση που τελικά υλοποιήθηκε είναι να βασιστούμε σε κλείδωμα της ομάδας, κλειδώνοντας την αντίστοιχη εγγραφή στον πίνακα των ομάδων (table row lock). Έτσι επιτρέπεται σε διαφορετικές ομάδες να προσθέτουν μέλη ασύγχρονα, αλλά για την ίδια ομάδα, η διαδικασία είναι σειριακή επιτρέποντας τον ορθό έλεγχο του ορίου των μελών. Η χρήση του trigger αποτελεί την εγγύηση της μη υπέρβασης του ορίου, αλλά η σειριακή εκτέλεση εγγυάται την ελάχιστη όχληση για τον χρήστη, καθώς εκμηδενίζει τις πιθανότητες να προκύψει τέτοιο σφάλμα.

Η αποχώρηση από την ομάδα αποτελεί προσωπική επιλογή του κάθε μέλους, δηλαδή δεν μπορεί ένας χρήστης να αφαιρέσει κάποιον άλλο αυτοβούλως, ενώ αυτή γίνεται αποδεκτή μόνο αν το μέλος προς αποχώρηση έχει μηδενικό υπόλοιπο. Ο έλεγχος του υπολοίπου γίνεται εντός συναλλαγής της βάσης, με αυστηρό επίπεδο απομόνωσης, τύπου Repeatable Read, ώστε να μην υπάρξουν αλλαγές σε αυτό κατά τη διάρκεια του ελέγχου. Επίσης, πραγματοποιείται ακόμα ένας έλεγχος, ώστε να βρεθεί αν η ομάδα διαθέτει ακόμα μέλη. Αν η συνθήκη είναι αρνητική, τότε η ομάδα διαγράφεται συμπεριλαμβανομένου του Περιεχομένου της. Αυτός ο έλεγχος αποτελεί τον μοναδικό τρόπο διαγραφής μιας ομάδας από το σύστημα. Σε περίπτωση αποχώρησης μέλους από την ομάδα, τότε οι πληροφορίες του χρήστη αποκρύπτονται. Για παράδειγμα, κατά την προβολή/επεξεργασία Περιεχομένου που περιλαμβάνει διαγραμμένα μέλη, τότε η θέση του ονόματος τους αντικαθιστάτε από τη φράση «Deleted member». Δεν επιτρέπεται η μεταβολή των δεδομένων του Περιεχομένου που αφορά διαγραμμένα μέλη.

Σημειώνεται ότι η προσθαφαίρεση χρηστών αποτελεί μια Αλλαγή για το σύστημα. Η σημασία αυτού του όρου, καθώς και ο λόγος που τελικά επιλέχθηκε αυτή η προσέγγιση επεξηγούνται αναλυτικά στην περιγραφή του χαρακτηριστικού του συγχρονισμού (λήψη δεδομένων).

3.2.6 Αποστολή και λήψη δεδομένων

Ο εξυπηρετητής δρα ως η μοναδική πηγή αλήθειας (SSOT) του συστήματος αφού αποτελεί τον συνδετικό κρίκο ανάμεσα σε όλες τις συσκευές παρέχοντας αξιόπιστα δεδομένα σε αυτές. Ο εξυπηρετητής κατά την παραλαβή των δεδομένων πάντα φροντίζει να τα αξιολογεί ελέγχοντας την ορθότητά τους. Μη έγκυρα δεδομένα δεν γίνονται αποδεκτά, ενώ αυτά που γίνονται αποδεκτά αποστέλλονται στους υπόλοιπους χρήστες είτε μέσω της διαδικασίας του συγχρονισμού, είτε καλώντας συγκεκριμένες υπηρεσίες.

Ο συγχρονισμός προσφέρει όλα τα απαραίτητα δεδομένα για τη λειτουργία της εφαρμογής του πελάτη, όπως:

- Οι προσωπικές πληροφορίες του χρήστη
- Οι ομάδες και οι ρυθμίσεις τους
- Τα μέλη των ομάδων και τα δεδομένα τους (πχ. όνομα, υπόλοιπο κτλ.)

- Αλλαγές, Συναλλαγές και Μεταφορές των ομάδων (αφορά μόνο εκείνες που δημιουργήθηκαν μετά από τον τελευταίο συγχρονισμό και όχι παλαιότερες)

Κατά τη διάρκεια της σχεδίασης αναδύθηκε το πρόβλημα του πως είναι δυνατό να παρακολουθούμε τις αλλαγές των υπολοίπων (balances) που προκαλούνται από την ανανέωση Περιεχομένου. Αρχικά, μια επιλογή θα ήταν η τοπική σύγκριση (στην εφαρμογή του πελάτη) της παρούσας και της νέας κατάστασης του πόρου για την παραγωγή των αλλαγών στα υπόλοιπα των χρηστών. Το μειονέκτημα αυτής της μεθόδου είναι η ανάγκη ύπαρξης της υπάρχουσας κατάστασης του πόρου στην τοπική μνήμη. Αυτό σημαίνει ότι όλες οι Συναλλαγές και οι Μεταφορές θα έπρεπε να βρίσκονται αποθηκευμένες εξαρχής στο κινητό των χρηστών και όχι κατά βούληση τους. Γρήγορα απορρίφθηκε αυτή η προσέγγιση λόγω του όγκου των δεδομένων που θα προκαλούσαν δριμεία αρνητική εμπειρία, ακόμα και αν πραγματοποιούταν μόνο κατά την πρώτη εκτέλεση. Μερικά από τα προβλήματα που θα προέκυπταν θα ήταν η ιδιαίτερα χρονοβόρα λήψη, η ανάγκη απαιτητικής σύνδεσης στο διαδίκτυο (υψηλή ταχύτητα μεταφοράς δεδομένων και περιορισμός ανώτατου ορίου χρήσης) και η δέσμευση πολύτιμου αποθηκευτικού χώρου στη συσκευή των χρηστών. Τέλος, πρόκληση θα αποτελούσε και η εύρεση του Περιεχομένου που διαθέτει νέα κατάσταση, δηλαδή έχει ενημερωθεί. Κάποιες επιλογές περιλαμβάνουν τη συγκέντρωση του μεταβληθέντος Περιεχομένου αποκλειστικά από τον εξυπηρετητή με παρακολούθηση των αλλαγών του ή τη συνεργασία εξυπηρετητή-πελάτη για σύγκριση των καταστάσεων του Περιεχομένου. Όμως καμία από αυτές τις επιλογές δεν είναι βέλτιστη, ενώ οι απαιτήσεις τους για πόρους είναι μεγάλες. Άλλη επιλογή θα ήταν η εκμετάλλευση του πεδίου χρονικής μεταβολής του πόρου (modified_on). Ανανεωμένο Περιεχόμενο θα αποτελούσε εκείνο με τιμή πεδίου μεταγενέστερη της τελευταίας χρονοσφραγίδας συγχρονισμού που παρέχεται από την εφαρμογή του πελάτη. Πρακτικά, αυτό δεν ισχύει, καθώς ο χρόνος είναι σχετικός και αναξιόπιστος για αυτόν τον σκοπό. Ιδιαίτερα αν ληφθεί υπόψιν ότι δεν υπάρχει καμία απολύτως εγγύηση για τον συγχρονισμό της ώρας μεταξύ εξυπηρετητή και πελάτη.

Η λύση που βρέθηκε είναι ο πελάτης να λαμβάνει μόνο τα υπόλοιπα των μελών κατά τον πρώτο συγχρονισμό και σε κάθε επόμενο συγχρονισμό να λαμβάνει μόνο τις μεταβολές τους. Για το λόγο αυτό, δημιουργήθηκε η οντότητα BalanceChange (στο εξής Αλλαγή). Συγκεκριμένα, όταν οι χρήστες προσθέτουν, διαγράφουν ή αλλάζουν κάποια Συναλλαγή ή Μεταφορά ή γενικά όταν υπάρξει μεταβολή

υπολοίπων, ο εξυπηρετητής, αφού ελέγξει την ορθότητά των εισαγόμενων δεδομένων, υπολογίζει τις λογιστικές αλλαγές, τις οποίες αποθηκεύει και στη συνέχεια αποστέλλει στον αποστολέα για να τις εφαρμόσει στα τοπικά υπόλοιπα. Ακολούθως, θα προωθηθούν στους χρήστες της ομάδας την επόμενη φορά που θα εκτελέσουν συγχρονισμό. Οι Αλλαγές παραπέμπουν στις Συναλλαγές/Μεταφορές μέσω της ιδιότητας genForId, οπότε παράλληλα διευκολύνουν τη λήψη καινούργιου ή αλλαγμένου Περιεχομένου, ενώ βοηθούν και στην αφαίρεση του διαγραμμένου.

Ο συγχρονισμός αποτελείται από δύο στάδια:

1. Την λήψη των προσωπικών πληροφοριών του χρήστη καθώς και όλων των ομάδων και των μελών τους
2. Τη παραλαβή των νέων Αλλαγών ή εκτεταμένων πληροφοριών για την κάθε ομάδα

Στο πρώτο στάδιο, οι πελάτες εκτελούν λήψη όλων των ομάδων στις οποίες λαμβάνουν μέρος. Οι ομάδες συνοδεύονται από τις βασικές πληροφορίες τους, δηλαδή όλα τα πεδία της οντότητας Group, καθώς και τα αναγνωριστικά των μελών τους. Επιπλέον, λαμβάνουν και βασικές πληροφορίες για τα μέλη των ομάδων, δηλαδή το όνομα κάθε μέλους. Τέλος, περιλαμβάνονται και προσωπικές πληροφορίες του λογαριασμού (όνομα και email). Όλα τα δεδομένα του συγκεντρώνονται και αποστέλλονται σε μία απάντηση.

Το δεύτερο στάδιο αφορά κάθε ομάδα. Αν μια ομάδα δεν έχει συγχρονιστεί ποτέ με τη συσκευή, τότε εκτελείται εκτενής λήψη των πληροφοριών της, συμπεριλαμβανομένου και των υπολοίπων των μελών (αρχικοποίηση υπολοίπων). Σε διαφορετική περίπτωση, αποστέλλονται τρεις πίνακες δεδομένων, οι οποίοι είναι οι νέες Αλλαγές που προέκυψαν από τον τελευταίο συγχρονισμό, καθώς και οι Συναλλαγές και οι Μεταφορές που τις δημιούργησαν. Ο πελάτης αποστέλλει δύο παραμέτρους στον εξυπηρετητή. Αυτές είναι query παράμετρος lastChangeId που υποδεικνύει το αναγνωριστικό της τελευταίας συγχρονισμένης Αλλαγής. Αφού οι Αλλαγές διαθέτουν σταδιακά αυξανόμενο αναγνωριστικό (serial/auto incremented), οι νέες Αλλαγές είναι αυτές που το αναγνωριστικό τους είναι μεγαλύτερο του lastChangeId. Ο πελάτης συμπληρώνει το πεδίο excludeChangeIds του σώματος (body) του αιτήματος, με ένα πίνακα που περιέχει τα αναγνωριστικά των ήδη συγχρονισμένων Αλλαγών, ώστε αυτές να εξαιρεθούν της αποστολής. Αυτές παράχθηκαν από την ίδια συσκευή και συνεπώς έχουν ήδη εφαρμοστεί. Σε

διαφορετική περίπτωση, η απάντηση του εξυπηρετητή θα εμπεριείχε όλες τις Αλλαγές, με αποτέλεσμα την διπλή εφαρμογή κάποιων εξ' αυτών, αλλοιώνοντας τα τοπικά υπόλοιπα των μελών.

Η ανάγνωση των δεδομένων γίνεται εντός συναλλαγής της βάσης δεδομένων με επίπεδο απομόνωσης Repeatable Read Read Only (βελτιστοποίηση για συναλλαγές που εκτελούν μόνο ανάγνωση δεδομένων). Ο λόγος που επιλέχθηκε το συγκεκριμένο επίπεδο απομόνωσης είναι για να εξασφαλίζει συνοχή στα δεδομένα, χωρίς να περιλαμβάνονται πιθανές αλλαγές που συνέβησαν κατά την εκτέλεση της ανάγνωσής τους. Αυτές οι νέες Αλλαγές μπορούν να παραληφθούν από τον επόμενο συγχρονισμό.

Ο όγκος των δεδομένων του δεύτερου σταδίου για τις ομάδες που έχει εκτελεστεί ο συγχρονισμός τουλάχιστον μια φορά, θα μπορούσε να είναι αρκετά μεγάλος (μεγάλο πλήθος Αλλαγών). Γι' αυτό επιλέχθηκε να εκτελείται συγχρονισμός των δεδομένων ανά ομάδα και να μην γίνεται ενιαία λήψη των δεδομένων όλων των ομάδων. Παρόλα αυτά, ακόμα και με τον κατακερματισμό των πληροφοριών ανά ομάδα, το μέγεθος των δεδομένων θα μπορούσε να αποτελέσει τροχοπέδη. Για το λόγο αυτό, επιλέχθηκε ο εξυπηρετητής να αποστέλλει σταδιακά τα δεδομένα μέσω μετάδοσης συνεχούς ροής (streaming). Αρχικά ο εξυπηρετητής κτίζει την απάντηση του γράφοντας σε ένα κανάλι εξόδου (HTTP Response), ενώ το κύριο μέρος του περιεχομένου της απάντησης επίσης τροφοδοτείται από τη βάση μέσω streaming. Με αυτόν τον τρόπο, τα δεδομένα διαβάζονται τμηματικά και η επεξεργασία τους γίνεται σχεδόν παράλληλα με την ανάκτησή τους, βελτιστοποιώντας τη διαδικασία και κάνοντας λελογισμένη χρήση πόρων και από τα δύο άκρα (εξυπηρετητής και πελάτης).

Προαναφέρθηκε ότι σαν Αλλαγή θεωρούνται και οι ενέργειες της προθήκης και της αφαίρεσης ενός χρήστη από μια ομάδα. Ο λόγος που οδήγησε σε αυτή την απόφαση είναι ότι από τη στιγμή που λαμβάνονται οι πληροφορίες μιας ομάδας, μαζί με τους χρήστες, στο πρώτο στάδιο του συγχρονισμού, μέχρι και τη στιγμή που εκκινεί το δεύτερο στάδιο, είναι πιθανό να έχουν αλλάξει τα μέλη της. Αυτό θα είχε ως αποτέλεσμα την πιθανή λήψη Αλλαγών σχετικές με την ενημέρωση υπολοίπων ανύπαρκτων χρηστών. Η διαδικασία θα αποτύγχανε και θα έπρεπε να επανεκκινηθεί. Θέτοντας όμως τις παραπάνω ενέργειες ως Αλλαγές, σημαίνει ότι για να ληφθεί μια Αλλαγή σχετικά με το υπόλοιπο κάποιου χρήστη, τότε έχει ήδη ληφθεί

η Αλλαγή που σηματοδοτεί τη συμμετοχή του στην ομάδα και άρα έχει αρχικοποιηθεί και το υπόλοιπό του στην εφαρμογή του πελάτη.

3.2.7 Διαχείριση υπολοίπων

Όπως αναφέρθηκε κάθε μέλος διαθέτει το πρωτεύων υπόλοιπο, το οποίο εκφράζει την λογιστική του κατάσταση. Θετικό υπόλοιπο σημαίνει ότι το μέλος είναι δανειστής ή πιστωτής και αρνητικό ότι είναι οφειλέτης ή χρεώστης. Αρχή για την ορθή λειτουργία της εφαρμογής αποτελεί ότι το άθροισμα των υπολοίπων όλων των μελών μιας ομάδας είναι ίσο με μηδέν, καθώς τα αρνητικά υπόλοιπα (οφειλές), συμπληρώνονται από τα θετικά υπόλοιπα (πιστώσεις).

Ανάλογα τον τύπο του Περιεχομένου, η μεταβολή του υπολοίπου υπολογίζεται διαφορετικά. Για τις συναλλαγές ισχύει ότι:

$$μεταβολή\ υπολοίπου_{χρήστη} = πίστωση_{χρήστη|συναλλαγής} - χρέωση_{χρήστη|συναλλαγής}$$

Ένα παράδειγμα εφαρμογής του ανωτέρου τύπου αποτελεί μια συναλλαγή αξίας δέκα ευρώ με ισότιμη κατανομή μεταξύ δύο χρηστών Α και Β. Αυτό σημαίνει ότι και τα δύο μέλη χρεώνονται με πέντε ευρώ. Αν για τη συναλλαγή πληρώσει μόνο το μέλος Α, τότε τα υπόλοιπα διαμορφώνονται:

$$\text{Α} = 10 - 5 = 5\ (\text{πιστωτής}) \text{ και } \text{Β} = 0 - 5 = -5 (\text{χρέωστης})$$

Για τις μεταφορές ισχύουν οι τύποι:

$$μεταβολή\ υπολοίπου_{εντολέα} = ποσό_{μεταφοράς}$$

$$μεταβολή\ υπολοίπου_{δικαιούχου} = -ποσό_{μεταφοράς}$$

Όπως φαίνεται και από τους παραπάνω τύπους, η μεταβολή του υπολοίπου του εντολέα είναι θετική. Για το λόγο αυτό γίνεται αναφορά σε αυτόν και ως πιστωτή. Το αντίθετο συμβαίνει στον δικαιούχο και ως εκ τούτου ταυτίζεται με τον χρεώστη.

Η ενημέρωση των υπολοίπων γίνεται βάσει των παραγόμενων Αλλαγών, προσθέτοντας τα ποσά (μεταβολές υπολοίπων) που αυτά περιέχουν στο συνολικό υπόλοιπο των μελών. Αποτελεί ιδιαίτερη διαδικασία, καθώς πρέπει να εγγυάται ότι τα ποσά μεταβάλλονται με όρθρο τρόπο ώστε να προληφθούν φαινόμενα αλλοίωσής

τους. Για το λόγο αυτό πρέπει να αποφευχθούν φαινόμενα συνθηκών ανταγωνισμού ή θα πρέπει να λαμβάνονται υπόψιν και να τα χειριζόμαστε κατάλληλα. Μια πρώιμη προσέγγιση είναι η επεξεργασία των υπολοίπων να γίνεται σειριακά. Παρόλα αυτά, θα ήταν επιθυμητό η εφαρμογή να εξυπηρετεί όσους περισσότερους χρήστες γίνεται και η σειριακή προσέγγιση δεν εξυπηρετεί αυτόν τον σκοπό.

Τα φαινόμενα συνθηκών ανταγωνισμού γίνονται εμφανή κατά τον κύκλο ανάγνωσης-επεξεργασίας-εγγραφής στη βάση δεδομένων. Αυτός ο κύκλος περιγράφει την ανάγνωση δεδομένων, την επεξεργασία αυτών ώστε να φτάσουν στην τελική τους μορφή και εν τέλει την αποθήκευσή τους. Κατά τη φάση της εκτέλεσης του είναι πιθανή η αλλαγή των δεδομένων στη βάση από κάποια άλλη διαδικασία, καθιστώντας τα αναγνωσμένα δεδομένα ξεπερασμένα και ακατάλληλα για επεξεργασία. Η αποθήκευση αυτών των επεξεργασμένων δεδομένων αποτελεί αιτία αλλοίωσης της βάσης. Αυτή η πιθανότητα είναι αντιμετωπίσιμη είτε με πιο αυστηρό επίπεδο απομόνωσης (και λογική για επανάληψη προσπάθειας σε περίπτωση αποτυχίας) ή κλειδώματος, είτε με αντικατάσταση αυτού του κύκλου.

Τελικώς η υλοποίηση βασίστηκε στην αποφυγή του κύκλου, ο οποίος και ορίζει ότι η ενημέρωση των δεδομένων γίνεται παραλείποντας το βήμα της ανάγνωσης και συνδυάζοντας τα δύο εναπομείναντα βήματα με ένα αίτημα στη βάση. Αυτή η τεχνική, σε συνδυασμό με το προεπιλεγμένο επίπεδο απομόνωσης (Read Committed) της βάσης, εξασφαλίζει την ακεραιότητα των δεδομένων. Σημειώνεται ότι επιλέχθηκε ο τύπος δεδομένων των υπολοίπων να είναι Big Integer ώστε να είναι υλοποιήσιμη αυτή η τεχνική, καθώς η ενημέρωση κάθε υπολοίπου στη βάση είναι ουσιαστικά το αποτέλεσμα μιας μαθηματικής πράξης από ένα αίτημα (query).

```
UPDATE groups_users SET balance = balance + 1000 WHERE user_id = 32 AND group_id = 8;
UPDATE groups_users SET balance = balance + -1000 WHERE user_id = 31 AND group_id = 8;
```

Εικόνα 3.7: Παράδειγμα παραγόμενων αιτημάτων που ενημερώνουν το υπόλοιπο των μελών στη βάση όταν δημιουργείται μια μεταφορά δέκα ευρώ μεταξύ δύο χρηστών

Σημειώνεται ότι παρόμοια προσέγγιση ακολουθείται και για τα δευτερεύοντα υπόλοιπα, με την διαφορά ότι δεν αποτελεί ανάγκη το άθροισμα των υπόλοιπων να είναι μηδενικό. Το συγκεκριμένο υπόλοιπο αναλύεται στην επόμενη υποενότητα.

3.2.8 Διαχείριση ποσών μικρότερων της μονάδας του νομίσματος

Κατά την ανάπτυξη εμφανίστηκε το ζήτημα της παρακολούθησης ποσών μικρότερων της ελάχιστης τιμής του νομίσματος. Ανάλογο παράδειγμα, όπως προαναφέρθηκε, αποτελεί ο διαμοιρασμός ενός λεπτού του ευρώ σε πάνω από ένα άτομο. Αυτό είναι απαραίτητο, καθώς όπως αναγράφεται στις απαιτήσεις της εφαρμογής, είναι καίριας σημασίας να γίνει όσο πιο δίκαιη γίνεται. Σκοπός αυτού του χαρακτηριστικού είναι, όταν εντοπίζονται παρόμοιες περιπτώσεις, να πληρώνουν κάποια μέλη το απαιτούμενο πόσο, δηλαδή το ένα λεπτό, και στις επόμενες συναλλαγές (όταν χρειαστεί) να πληρώνεται από τα υπόλοιπα μέλη. Σημειώνεται ότι ανά συναλλαγή, αν κάποιο μέλος κληθεί να συμπληρώσει το εν λόγω ποσό, η τιμή αυτού δεν πρόκειται να ξεπερνάει το ένα λεπτό.

Όπως είναι φυσικό, οι μεταφορές δεν παρουσιάζουν αυτό το πρόβλημα, επειδή δεν υποστηρίζουν κάποιας μορφής διαμοιρασμό του ποσού. Το ποσό της μεταφοράς στέλνεται ακριβώς όπως είναι στον παραλήπτη. Έγινε αντιληπτό ότι θα έπρεπε να προστεθεί ένα ακόμα πεδίο υπολοίπου σε κάθε μέλος, η τιμή των οποίων θα καθόριζε ποια μέλη καλούνται να πληρώσουν σε περίπτωση τέτοιων συμβάντων. Αυτό το πεδίο είναι το δευτερεύων υπόλοιπο που αναφέρθηκε στο προηγούμενο κεφάλαιο και σε αντίθεση με το πρωτεύων υπόλοιπο δεν καθίσταται άμεσα πληρωτέο. Παρόλα αυτά, πρέπει να αποφασιστεί ο κατάλληλος τύπος δεδομένων που θα διατηρεί το πεδίο.

Η πρώτη σκέψη ήταν η διατήρηση αυτών των ποσών σε κλάσματα, πετυχαίνοντας τη μέγιστη δυνατή ακρίβεια. Παρόλα αυτά θα αναδυόντουσαν κάποια προβλήματα. Το πρώτο πηγάζει στο γεγονός ότι είναι μια διαδικτυακή εφαρμογή, η οποία αποτελεί ένα σύστημα και έτσι ολοκληρώνεται από πολλά κομμάτια. Για παράδειγμα, η πιθανή μελλοντική ανάγκη υποστήριξης πολλαπλών πλατφορμών δημιουργεί την ανάγκη ανάπτυξης του λογισμικού-πελάτη για πολλά διαφορετικά περιβάλλοντα. Κάθε πλατφόρμα υποστηρίζει τα δικά της εργαλεία ανάπτυξης και υπακούει σε κάποιους περιορισμούς, ανάλογα με το υλικολογισμικό της. Το κύριο πρόβλημα λοιπόν, είναι ότι θα υπήρχαν ασυμβατότητες των αποτελεσμάτων από την επεξεργασία των κλασμάτων. Συγκεκριμένα, τόσο ο εξυπηρετητής όταν υπολογίσει αυτό το ποσό, όσο και ο πελάτης όταν θα το λάβει, θα πρέπει να το προσθέσουν στην υπάρχουσα τιμή του (δευτερεύοντος) υπολοίπου που διατηρούν τοπικά. Από τις δοκιμές ήταν ξεκάθαρο το χάσμα ανάμεσα στις βιβλιοθήκες (των διαφορετικών περιβαλλόντων) που χρησιμοποιήθηκαν ώστε να γίνει η επεξεργασία των κλασμάτων. Παρόλο που

δεν είναι απαραίτητη η προϋπόθεση το άθροισμα των δευτερευόντων υπολοίπων (ep) να είναι μηδέν, όπως στα πρωτεύοντα υπόλοιπα, θα ήταν επιθυμητό κάθε υπολογισμός να είχε ίδιο αποτέλεσμα.

Λύση στο παραπάνω πρόβλημα θα μπορούσε να είναι ο υπολογισμός των κλασμάτων σε ένα κεντρικό μέρος, δηλαδή τον εξυπηρετητή, αλλά μεταφέρει επιπλέον φόρτο σε αυτόν, αυξάνοντας το κόστος συντήρησης και επηρεάζοντας την απόδοσή του. Επιπλέον, απορρίπτει οποιαδήποτε μελλοντική βλέψη για υποστήριξη λειτουργικότητας της εφαρμογής του πελάτη δίχως διαδίκτυο.

Άλλη εναλλακτική θα ήταν να βασιστούμε σε μια μόνο πλατφόρμα, τόσο για τις εφαρμογές του πελάτη, όσο και του εξυπηρετητή. Μια καλή επιλογή αποτελεί η Java, αλλά δεν είναι βιώσιμη λύση, καθώς περιορίζει τη μελλοντική υποστήριξη άλλων πλατφορμών και των κοινών (χρηστών) τους. Άλλη επιλογή θα ήταν η χρήση επιπλέον προγραμμάτων που γεφυρώνουν το NodeJs με τη Java (η εφαρμογή πελάτη θα πρέπει να είναι σε Java) ή η χρήση κάποιου microservice για αυτό το σκοπό. Αυτές οι επιλογές, επίσης περιορίζουν την υποστήριξη άλλων πλατφορμών για την εφαρμογή του πελάτη, ενώ παράλληλα δημιουργούν αχρείαστη πολυπλοκότητα και δυνητικά προβλήματα απόδοσης, όπως είναι η πιο αργή εκτέλεση.

Το πιο σημαντικό πρόβλημα όμως που δημιουργούν είναι ότι δεν θα ήταν δυνατό να αποφευχθεί ο κύκλος ανάγνωσης-επεξεργασίας-εγγραφής. Οι βάσεις δεδομένων δεν διαθέτουν δυνατότητες επεξεργασίας κλασμάτων, οπότε θα ήταν ευθύνη της εφαρμογής ο υπολογισμός των αποτελεσμάτων, η μετατροπή τους σε αλφαριθμητικά και στη συνέχεια η βάση απλά θα αποθήκευε τα δεδομένα. Η χρήση του κύκλου έχει μειονεκτήματα όπως αυτά παρουσιάστηκαν προηγουμένως. Ανησυχητικό ήταν επίσης το γεγονός ότι αν ανάμεσα στις συναλλαγές δεν ήταν δυνατή κάποια απλοποίηση, τότε οι όροι του κλάσματος θα είχαν αυξητικές τάσεις. Αυτό μεταφράζεται σε μεγαλύτερες απαιτήσεις αποθήκευσης, ενώ και η διαχείρισή τους θα ήταν πιο πολύπλοκη.

Για τους παραπάνω λόγους, τελικά επιλέχθηκε η χρήση ακέραιου αριθμού, τύπου Big Integer, ο οποίος θα περιγράφει το δευτερεύων υπόλοιπο. Παράγεται από τη διαίρεση του κλάσματος του μεριδίου κάθε χρήστη για το αντικείμενο. Επομένως, κάθε αντικείμενο, πέρα από το κλάσμα, θα έχει και το αποτέλεσμά αυτού ως ακέραιο αριθμό. Είναι σίγουρο ότι το ίδιο αποτέλεσμα παράγεται από τον εξυπηρετητή και τον πελάτη, καθώς χρησιμοποιούνται ίδιες ρυθμίσεις στρογγυλοποίησης των

αριθμών. Αυτές είναι η στρογγυλοποίηση προς τα πάνω και ο ορισμός μέγιστου αριθμού δεκαδικών ψηφίων. Από τα δεκαδικά ψηφία εξαρτάται και η ακρίβεια του αποτελέσματος. Ο τύπος Big Integer υποστηρίζει αριθμούς μέχρι δεκαεννιά ψηφία. Ορίστηκε ότι τα δύο πρώτα ψηφία αφορούν τα λεπτά και τα υπόλοιπα δεκαεπτά αφορούν ποσά που δεν αποτελούν πληρωτέο ποσό και υπολογίστηκαν από τη διαίρεση του κλάσματος. Τα λεπτά υπάρχουν, καθώς αναγκαστικά θα προκύψουν από τη σταδιακή πρόσθεση των δευτερευόντων υπολοίπων, ενώ τα περισσότερα ψηφία δόθηκαν για το αποτέλεσμα του κλάσματος, ώστε να διατηρηθεί όσο περισσότερη ακρίβεια γίνεται, αξιοποιώντας το μέγιστο όριο του τύπου δεδομένων.

Όμως τα λεπτά που αναφέρθηκαν παραπάνω αποτελούν πληρωτέο ποσό, οπότε και θα πρέπει να μεταφερθούν στο πρωτεύων υπόλοιπο. Αυτή τη δουλειά εκτελεί ο πελάτης ψάχνοντας αν προκύπτει (ίδιας απόλυτης τιμής) πληρωτέο ποσό από αντίθετα δευτερεύοντα υπόλοιπα μεταξύ δύο μελών. Αναζητούνται δύο μέλη, επειδή ωσάν μεταφορά, πρέπει να αποτελείται από ένα χρεώστη (αρνητικό δευτερεύων υπόλοιπο) και ένα πιστωτή (θετικό δευτερεύων υπόλοιπο). Αν η αναζήτηση έχει αποτέλεσμα, τότε δημιουργούνται οι αυτοματοποιημένες εντολές (AutoEps). Για το μέλος με αρνητικό δευτερεύων υπόλοιπο (χρεώστη), το πληρωτέο ποσό προστίθεται στο δευτερεύων υπόλοιπό του και αφαιρείται από το πρωτεύων. Το αντίθετο ισχύει για τον πιστωτή. Τα παραπάνω σημαίνουν ότι διατηρείται το μηδενικό άθροισμα των πρωτευόντων υπολοίπων των μελών.

Πριν την εκτέλεση του συγχρονισμού, ο εξυπηρετητής ειδοποιείται από τον πελάτη, με αντίστοιχη κλήση σε endpoint, για την εύρεση της αυτοματοποιημένης εντολής. Αφού ο εξυπηρετητής επαληθεύσει την ύπαρξή της με έλεγχο των δευτερευόντων υπολοίπων, την αποδέχεται. Η αποδοχή δημιουργεί τις νέες Αλλαγές και ενημερώνει τα υπόλοιπα των μελών. Κατά τον έλεγχο των δευτερευόντων υπολοίπων είναι πιθανό να εμφανιστούν συνθήκες ανταγωνισμού. Αυτό σημαίνει ότι τα αναγνωσμένα υπόλοιπα έχουν αλλάξει και οι συνθήκες αποδοχής δεν έχουν πια ισχύ. Αυτή η περίπτωση δεν αλλοιώνει τα υπόλοιπα, καθώς η αρχή ότι το άθροισμα των υπολοίπων ισούται με μηδέν δεν παραβιάζεται, ενώ τα μεταφερόμενα ποσά δεν χάνονται καθώς θα υπάρχουν στα δευτερεύοντα υπόλοιπα. Αυτοί οι λόγοι αιτιολογούν το γεγονός ότι δεν πάρθηκαν αυστηρά μέτρα αποφυγής των συνθηκών ανταγωνισμού, καθώς θα ήταν εις βάρος της δυνατότητας εξυπηρέτησης πολλαπλών μελών από τα services. Όμως ελήφθησαν μέτρα για τον κατευνασμό τέτοιων περιπτώσεων. Συγκεκριμένα, εκτελείται κλείδωμα της εγγραφής της ομάδας,

ώστε η πρόσβαση στο συγκεκριμένο endpoint να γίνεται σειριακά και να μην προκαλούνται έντονα τέτοια συμβάντα.

3.2.9 Ενημέρωση και διαγραφή δεδομένων

Όπως ειπώθηκε, τόσο η ενημέρωση, όσο και η διαγραφή Περιεχομένου προκαλούν τη δημιουργία Αλλαγών, καθώς μεταβάλλουν τα υπόλοιπα. Κατά τη διαδικασία της ενημέρωσης, γίνεται αντικατάστασή των δεδομένων στον αρχικό πόρο. Εν συνεχεία, υπολογίζονται οι λογιστικές αλλαγές μέσω της σύγκρισης παλαιών και νέων ποσών των μελών. Η διαγραφή αντιστρέφει το πόσο των μελών που επηρεάζονται και το επιστρέφει στο συνολικό τους υπόλοιπο.

Κάθε ενημέρωση Περιεχομένου υπόκειται στις επιπτώσεις του κύκλου ανάγνωσης-επεξεργασίας-εγγραφής. Αυτό σημαίνει ότι καθώς ένας χρήστης ενημερώνει τα δεδομένα ενός πόρου, μεταφοράς ή συναλλαγής, τότε τα αναγνωσμένα δεδομένα μπορεί να είναι ήδη ξεπερασμένα. Η σύγκρισή τους με τα νέα, εισαγόμενα από το χρήστη δεδομένα, θα μπορούσε να οδηγήσει σε λανθασμένο υπολογισμό διαφορών και εν τέλει σε αλλοίωση. Η λύση σε αυτή την περίπτωση, περιλαμβάνει τη συνδρομή της βάσης δεδομένων, εκτελώντας κλείδωμα του πόρου (table row lock). Αυτό σημαίνει ότι κάθε στιγμή, η πρόσβαση σε κάθε πόρο περιορίζεται στην εκτέλεση μιας και μόνο ενέργειας. Το κλείδωμα του πόρου συμβαίνει και κατά τη διαγραφή, συμβάλλοντας στην αποφυγή ενημέρωσής του ενόσω αυτός διαγράφεται. Η δημιουργία του πόρου δεν περιλαμβάνει κάποια ανάλογη ενέργεια, αφού αυτός δεν υπάρχει και δεν τίθεται θέμα πρόσβασής του από άλλο χρήστη. Επιπροσθέτως, κάθε ενημέρωση αυξάνει και την τιμή του πεδίου της έκδοσης. Σε περίπτωση που το πεδίο της έκδοσης της οντότητας που στέλνεται προς ενημέρωση από τον πελάτη, είναι μικρότερο της τιμής του αντίστοιχου πεδίου του πόρου που διαθέτει ο εξυπηρετητής, τότε η αποδοχή της εξαρτάται από την τιμή της ιδιότητας overwrite στις ρυθμίσεις της ομάδας.

Σημειώνεται ότι δεν επιτρέπεται η αλλαγή του υπολοίπου των μελών τα οποία έχουν διαγραφεί, δηλαδή έχουν αποχωρήσει από την ομάδα. Όταν κατά τον έλεγχο ανιχνευθεί ανάλογο συμβάν, τότε η διαδικασία ακυρώνεται, εμφανίζοντας κατάλληλο μήνυμα σφάλματος. Αρχικά, αυτό γίνεται επειδή τα αποχωρήσαντα μέλη δεν γίνεται να έχουν γνώση για κάτι που θα τους επηρέαζε άμεσα. Επίσης θα πραγματοποιούταν αλλοίωση των υπολοίπων, καθώς οι μεταβολές υπολοίπων που

επηρεάζουν ενεργά και διαγραμμένα μέλη, εφαρμόζονται μόνο στα ενεργά, διαταράσσοντας το μηδενικό άθροισμα των υπολοίπων της ομάδας.

3.2.10 Ανάκτηση παλαιότερου Περιεχομένου

Για τις Αλλαγές έχουμε ήδη αναφέρει ότι κυρίως αντικατοπτρίζουν μεταβολές στα υπόλοιπα. Μολαταύτα, έχουν και άλλες πρακτικές χρήσεις. Η ανανέωση των πόρων, ότι και αν αυτή περιλαμβάνει (ακόμα και μηδενική μεταβολή υπολοίπων), αποτυπώνεται ως Αλλαγή. Οι Αλλαγές λοιπόν, υποδεικνύουν ένα χρονοδιάγραμμα μεταβολής του Περιεχομένου, καθώς λαμβάνοντας κατά τον συγχρονισμό τις πιο πρόσφατες Αλλαγές, λαμβάνεται και το πιο πρόσφατο Περιεχόμενο. Το χρονοδιάγραμμα δεν στηρίζεται σε χρονοσήμανση, αλλά στις διακριτές Αλλαγές και τη σειριακή αύξηση του αναγνωριστικού τους.

Το παλαιότερο Περιεχόμενο, ταξινομημένο με βάση τα φθίνοντα αναγνωριστικά των Αλλαγών, είναι προσβάσιμο μέσω άλλων endpoints. Επειδή όμως ο όγκος αυτών των δεδομένων είναι δυνητικά μεγάλος, απαιτείται ιδιαίτερη διαχείριση, καθώς θα μπορούσαν να φτάσουνε σε σημείο που θα κατέκλυζαν τόσο τον εξυπηρετητή, όσο και τον πελάτη, οδηγώντας συχνά σε αστοχίες. Επίσης, δεν είναι αναγκαία η λήψη όλου του Περιεχομένου, αλλά μόνο αυτού που επιθυμεί ο χρήστης. Γι’ αυτό επιλέχθηκε η σταδιακή λήψη αυτού του Περιεχομένου, κατά βούληση του χρήστη, μέσω σελιδοποίησης. Η τεχνική αυτή τεμαχίζει τα δεδομένα σε μικρότερα και πιο διαχειρίσιμα κομμάτια, ενώ κάθε κομμάτι λαμβάνεται όταν αυτό απαιτηθεί. Το κομμάτι που ανακτάται εξαρτάται από την query παράμετρο minTransactionChangeId/minTransferChangeId και υποδεικνύει το μικρότερο αναγνωριστικό της τελευταίας ληφθείσας Αλλαγής που αφορούσε Συναλλαγές ή Μεταφορές. Ανάλογα με ποιο τύπο Περιεχομένου αποζητά ο χρήστης, εκτελείται αίτηση στο αντίστοιχο service. Το service θα απαντήσει με τις Συναλλαγές/Μεταφορές που περιέχονται στις Αλλαγές με αναγνωριστικό μικρότερο του παρεχόμενου, δηλαδή όλες τις προηγούμενες. Για να περιορίσουμε το μέγεθος της απάντησης γίνεται χρήση των παραμέτρων σελιδοποίησης που προαιρετικά παρέχονται από τον πελάτη. Η απουσία τους σημαίνει ότι θα χρησιμοποιηθούν οι προεπιλεγμένες από τον εξυπηρετητή τιμές. Αυτές οι (query) παράμετροι είναι η σελίδα (page) και το όριο (limit). Η σελίδα δείχνει το κομμάτι προς ανάκτηση, είναι θετικός ακέραιος και η προεπιλεγμένη τιμή του είναι το ένα. Συνήθως αυτή η

παράμετρος παραμένει κενή, καθώς το κάθε κομμάτι στηρίζεται στις παραμέτρους minTransactionChangeId/minTransferChangeId, και όχι στον αριθμό της σελίδας. Το όριο περιορίζει το μέγιστο πλήθος των Αλλαγών που θα παραληφθούν. Αποτελεί θετικό ακέραιο με προεπιλεγμένη τιμή το πέντε και μέγιστη το εκατό. Σε αυτή την παράμετρο υπάρχει μέγιστη τιμή, καθώς σκοπός είναι ο περιορισμός των δεδομένων.

3.2.11 Ασφάλεια και έλεγχος δεδομένων

Η ασφάλεια της εφαρμογής είναι υψίστης σημασίας, καθώς προστατεύει τα δεδομένα των χρηστών και υποστηρίζει την ανεμπόδιστη εκτέλεση της εφαρμογής. Προηγουμένως παρουσιάστηκε η προστασία των endpoints από μη αυθεντικοποιημένους ή/και μη εξουσιοδοτημένους χρήστες. Παρόλα αυτά υπάρχουν και άλλες προκλήσεις και απειλές.

Τα δεδομένα εισόδου θα μπορούσαν να αποτελέσουν πηγή εκμετάλλευσης ευπαθειών με σκοπό κακόβουλες ενέργειες. Για αυτό, τα δεδομένα πρέπει να ελέγχονται και αν χρειαστεί να τροποποιούνται ώστε να εξασφαλιστεί ότι το περιεχόμενό τους είναι αποδεκτό και ασφαλές για αποθήκευση και χρήση από την εφαρμογή. Αρχικά θα πρέπει τα δεδομένα να μην ξεπερνάνε ένα συγκεκριμένο μέγεθος, καθώς αυτό θα σήμαινε ότι ο εξυπηρετητής θα πρέπει να αφιερώσει μεγάλο αριθμό πόρων για τη διαχείριση εξίσου μεγάλου μεγέθους δεδομένων. Αυτή η επίθεση ονομάζεται Large Payload Post DDoS και θα μπορούσε να προκαλέσει κατάρρευση του συστήματος, καθώς αυτό ασφυκτιά από την έλλειψη διαθέσιμων πόρων. Για το λόγο αυτό έχουν οριστεί τα πενήντα megabytes ως μέγιστο όριο μεγέθους σε όλα τα αιτήματα.

Στη συνέχεια θα πρέπει να ελέγχεται το περιεχόμενο των αιτημάτων. Αυτό επιτυγχάνεται αρχικά με την «απολύμανσή» (sanitization) και στη συνέχεια με την επικύρωση (validation). Τα μέτρα που λαμβάνονται κατά την πρώτη διαδικασία συνήθως περιέχουν την διαφυγή χαρακτήρων (escape characters) με ιδιαίτερη σημασία που εν τέλει μπορεί να έχουν αρνητική επίδραση. Σε αυτή την τεχνική ο χρήστης δεν βλέπει κάποια διαφορά, παρόλο που τα δεδομένα έχουν αποθηκευτεί με διαφορετική μορφή από αυτή που εισήγαγε. Επίσης αποδεκτή τεχνική αποτελεί και η μερική ή/και η ολοκληρωτική αφαίρεση (strip) των επιβλαβών λέξεων. Τέτοιες τεχνικές είναι χρήσιμες κυρίως για διαδικτυακές εφαρμογές που εκτελούνται σε

περιηγητές, καθώς διευκολύνουν επιθέσεις τύπου XSS. Σε αντίθεση με αυτές, η παρούσα εφαρμογή δεν κάνει χρήση τέτοιων τεχνολογιών (HTML, JavaScript) στην εφαρμογή του πελάτη, οπότε δεν περιλαμβάνει παρόμοιους ελέγχους. Το sanitization της εφαρμογής περιλαμβάνει μόνο την αφαίρεση των μη αναγκαίων κενών που μπορεί να βρίσκονται άκρες της λέξης (trim).

Ο έλεγχος από τη δεύτερη διαδικασία εγγυάται ότι τα δεδομένα είναι στη μορφή (τύπος δεδομένων, δομή κτλ.) που απαιτείται. Η επικύρωση πραγματοποιείται με διάφορους τρόπους. Στις διαθέσιμες επιλογές συμπεριλαμβάνεται η χρήση pipes που προσφέρει το Nest framework. Πρόκειται για κλάσεις που υλοποιούν ένα πρότυπο διεπαφής (interface), την PipeTransform, η οποία περιλαμβάνει μια και μόνο μέθοδο, τη transform. Κάθε κλάση περιέχει μια υλοποίηση της μεθόδου που αντιστοιχίζεται στις εκάστοτε ανάγκες επικύρωσης της εισόδου. Άλλος τρόπος είναι η δημιουργία κλάσης που εκτελεί χρέη DTO, οπότε και περιέχει αποκλειστικά πεδία που περιγράφουν τις ιδιότητές της. Κάθε πεδίο μπορεί να περιέχει πλήθος decorators που ορίζουν τις συνθήκες για επιτυχή επικύρωση. Η χρήση αυτής της μεθόδου απαιτεί την βιβλιοθήκη class-validator, η οποία και απολαύει πλήρης υποστήριξης του Nest framework. Για πιο περίπλοκα αντικείμενα, όπου οι συνθήκες επικύρωσης θα ήταν προτιμότερο να δηλωθούν ως σχηματικό (schema), το framework υποστηρίζει και τη βιβλιοθήκη Joi. Η εφαρμογή αξιοποιεί όλο το φάσμα των διαθέσιμων επιλογών επικύρωσης, ανάλογα με τις ανάγκες που παρουσιαστήκαν.

Σε σημεία που απαιτείται επικύρωση σε μεμονωμένα δεδομένα, όπως για παράδειγμα ο έλεγχος ενός αναγνωριστικού, όπου απαιτείται να είναι θετικός, πεπερασμένος ακέραιος, τότε απλά δημιουργείται μια κλάση. Αυτή υλοποιεί το πρότυπο διεπαφής PipeTransform και εντός της μεθόδου transform περιγράφεται με κώδικα ο έλεγχος του περιεχομένου ώστε να επαληθευτεί ότι περιλαμβάνει μόνο αριθμητικούς χαρακτήρες και ότι ο αριθμός που αναπαριστά είναι εντός των απαιτούμενων ορίων.

```
@Injectable()
export class BigIntPipe implements PipeTransform<string> {
  static max: bigint = BigInt("9223372036854775807");
  static min: bigint = BigInt("-9223372036854775808");

  protected positiveOnly: boolean;

  constructor(@Optional() options?: { positiveOnly?: boolean }) {
    options = options || {};
    const { positiveOnly } = options;

    this.positiveOnly = positiveOnly || false;
  }

  transform(value: string, metadata: ArgumentMetadata): string {
    if (!BigIntPipe.check(value, !this.positiveOnly)) {
      this.error(value);
    }

    return value;
  }

  static check(value: string, checkNeg: boolean = false): boolean {
    const regex = (checkNeg) ? bigIntRegexAndNeg : bigIntRegex;
    const containsNumbersOnly = regex.test(value);

    if (!containsNumbersOnly) return false;

    if (BigIntPipe.max != null) {
      if (BigInt(value) > BigIntPipe.max) return false;
    }

    if (checkNeg) {
      if (BigInt(value) < BigIntPipe.min) return false;
    }

    return true;
  }

  error(value: string) {
    console.error(`BigIntPipe validation failed for value ${value}`)
    throw new BadRequestException('Validation failed');
  }
}
```

Εικόνα 3.8: Κομμάτι κώδικα από το αρχείο bigint.pipe.ts

Παρόμοια μέθοδος χρησιμοποιείται και όταν είναι επιθυμητή η επαλήθευση των δεδομένων συνδυαστικά (και όχι απλά η επαλήθευση των πεδίων), όπως για

παράδειγμα στον έλεγχο ότι ο συνδυασμός των δεδομένων μιας Συναλλαγής την καθιστά αποδεκτή. Εκεί η εκτέλεση μαθηματικών πράξεων είναι αναγκαία, ώστε να γίνει επιβεβαίωση των αποτελεσμάτων που παρήγαγε η εφαρμογή του πελάτη και περιέχονται στο DTO της Συναλλαγής. Αυτά είναι η λίστα των αντικειμένων, με πληροφορίες σχετικές με αυτά και τα μερίδια τους, η λίστα των συμμετεχόντων μελών με το σύνολο των πληροφοριών τους για τη συναλλαγή (πίστωση, θετικό ή αρνητικό πρωτεύων και δευτερεύων υπόλοιπο), καθώς και γενικές πληροφορίες, όπως τίτλος, περιγραφή κτλ..

Αρχικά γίνεται επαλήθευση κάθε αντικειμένου. Αυτή περιλαμβάνει τον υπολογισμό του ποσού που αντιστοιχεί σε κάθε συμμετέχων μέλος βάσει του κλάσματος του μεριδίου του. Τα κλάσματα ανακτώνται από τη λίστα των αντικειμένων του DTO. Επιπλέον ευρίσκονται τα μέλη που κλήθηκαν να συμπληρώσουν τα μη πληρωτέα ποσά και προστίθεται ένα λεπτό στην πίστωσή τους (για το αντικείμενο). Εν συνεχεία, συγκρίνονται τα υπολογισμένα από τον εξυπηρετητή δεδομένα με τα ληφθέντα/υπολογισμένα από την εφαρμογή του πελάτη. Αν διαφέρουν, εμφανίζεται σφάλμα, τερματίζοντας ολοκληρωτικά τη διαδικασία. Σημειώνεται ότι η εφαρμογή του εξυπηρετητή δεν μπορεί να ελέγξει αν το μη πληρωτέο ποσό έχει ανατεθεί στα μέλη που θα έπρεπε, καθώς η κατάσταση των υπολοίπων των μελών της εφαρμογής-πελάτη τη στιγμή που υπολόγιζε τη συναλλαγή, μπορεί να διαφέρει από την κατάσταση των υπολοίπων που διατηρεί ο εξυπηρετητής τη στιγμή που έλαβε το αίτημα.

Ταυτόχρονα για κάθε μέλος αθροίζεται το ποσό που οφείλει ανά αντικείμενο και στο τέλος γίνεται έλεγχος αν οι τιμές συμπίπτουν με αυτές που περιέχει η λίστα των συμμετεχόντων μελών. Στη συνέχεια, αυτά προστίθενται μεταξύ τους ώστε να βρεθεί η συνολική αξία της συναλλαγής. Αυτό το άθροισμα συγκρίνεται με το πεδίο της αξίας της συναλλαγής, ώστε να επαληθευτεί η μεταξύ τους ισότητα. Επίσης παρόμοια τιμή πρέπει να έχει το άθροισμα των ποσών που πλήρωσε κάθε μέλος για τη συναλλαγή. Η παραπάνω διαδικασία έχει αρκετά κοινά με τη διαδικασία που περιγράφεται παρακάτω στην υποενότητα «Αλγόριθμος υπολογισμού ποσών βάσει κατανομής».

Για επαλήθευση των ιδιοτήτων των αντικείμενων γίνεται χρήση των decorators σε DTOs σε συνδυασμό με ValidationPipes. Για παράδειγμα, η εγγραφή του χρήστη απαιτεί τρία πεδία, το όνομα, την διεύθυνση ηλεκτρονικού ταχυδρομείου και ένα κωδικό πρόσβασης. Για τις ανάγκες επικύρωσης αυτών των δεδομένων,

δημιουργείται μια κλάση (DTO), της οποίας οι ιδιότητες είναι τα ανωτέρω πεδία. Κάθε ιδιότητα θα έχει τους αντίστοιχους decorators. Συγκεκριμένα είναι δυνατόν να χρησιμοποιηθούν decorators που παρέχονται άμεσα από την βιβλιοθήκη, όπως ο decorator IsEmail που περιλαμβάνει τους κανόνες για τη σωστή μορφή που θα πρέπει να έχει ένα αλφαριθμητικό, ώστε αυτό να αποτελεί έγκυρη διεύθυνση ηλεκτρονικού ταχυδρομείου. Για πιο ειδικές εφαρμογές υπάρχει η δυνατότητα να δημιουργηθούν decorators από τους προγραμματιστές. Η εφαρμογή εκμεταλλεύεται αυτή τη δυνατότητα έχοντας ξεχωριστούς decorators για τον έλεγχο του ονόματος χρήστη, καθώς και του κωδικού πρόσβασης. Στην πρώτη περίπτωση είναι επιθυμητό το όνομα να είναι ένα μη κενό αλφαριθμητικό, μήκους δύο έως σαράντα χαρακτήρων. Ο κωδικός πρόσβασης θα πρέπει να είναι ένα μη κενό αλφαριθμητικό με τουλάχιστον οκτώ χαρακτήρες και να περιέχει τουλάχιστον ένα γράμμα, αριθμό και ειδικό χαρακτήρα. Η χρήση νέων decorators αντικαθιστά την χρήση πολλαπλών decorators με ένα ενιαίο και έτσι επιτρέπει την εύκολη επαναχρησιμοποίησή τους. Η διαδικασία της επικύρωσης απαιτεί τη δήλωση ενός ValidationPipe, σε επίπεδο μεθόδου, κλάσης ή εφαρμογής, ώστε να γίνει εμφανής η ανάγκη ελέγχου των εισαγόμενων δεδομένων. Στην περίπτωση της παρούσας διπλωματικής, γίνεται εφαρμογή σε επίπεδο εφαρμογής, εξασφαλίζοντας ότι τα δεδομένων όλων των εισόδων είναι ασφαλή. Η ανάγκη του sanitization στα δεδομένα παραμένει και εκτελείται παρέχοντας τον decorator Transform. Αυτός είναι υπεύθυνος για τη μετατροπή των δεδομένων της ιδιότητας, βάσει του κώδικα που δόθηκε στην παράμετρο transformFn, που αποτελεί την συνάρτηση μετατροπής.

```
export class CreateUserDto {
  @IsEmail({}, { message: emailErrorMessage })
  @Transform(({ value }: TransformFnParams) => trim(value))      // Santitize
  email: string;

  @IsPassword()
  password: string;

  @IsUsername()
  name: string;
}
```

Εικόνα 3.9: Αρχείο create.user.dto.ts

Για περίπλοκα αντικείμενα, όπως οι Συναλλαγές, η επαλήθευση της ορθότητας των ιδιοτήτων τους γίνεται με τη βιβλιοθήκη Joi σε συνδυασμό με ένα ValidationPipe με παράμετρο το αντίστοιχο σχηματικό. Η δομή κάποιων αντικειμένων διαφέρει ελαφρώς, όπως μια νέα και μια ανανεωμένη συναλλαγή. Για αποφυγή διπλότυπων σχηματικών είναι δυνατή η χρήση υπαρχόντων με αντικατάσταση κάποιων κανόνων.

```
export const transactionSchema = Joi.object({
  items: Joi.array().items(itemSchema),
  shares: transactionUsersSchema,
  id: Joi.string().pattern(bigIntRegex),
  groupId: Joi.string().pattern(bigIntRegex)
                .messages(bigIntErrorMessage('group id')).required(),
  title: Joi.string().trim().max(40).required(),
  description: Joi.string().trim().max(200).allow(null, ''),
  dateTime: Joi.date().required(),
  createdOn: Joi.date().required(),
  location: Joi.object({
    name: Joi.string().trim().max(60).allow(null),
    coordinates: Joi.object({
      lat: Joi.number().messages(pointErrorMessage()).required(),
      lon: Joi.number().messages(pointErrorMessage()).required()
    }).allow(null)
  }).allow(null),
  version: Joi.number().integer().min(0).optional()
});
```

Εικόνα 3.10: Κομμάτι κώδικα από το αρχείο joi.validation.pipe.ts με το σχηματικό επικύρωσης για τη δημιουργίας μιας νέας Συναλλαγής

Μία από τις πιο γνωστές επιθέσεις που μπορεί να συμβεί λόγω αφιλτράριστων δεδομένων εισόδου αποτελεί η έγχυση (injection) SQL. Η επίθεση αυτή έχει σκοπό την εισαγωγή κακόβουλου SQL κώδικα μέσω κάποιας εισόδου με τελικό σκοπό την μη εξουσιοδοτημένη πρόσβαση στη βάση δεδομένων. Ο πιο διαδεδομένος τρόπος να εμποδιστεί η έγχυση SQL είναι η χρήση παραμετροποιημένων αιτημάτων (parameterized queries). Αυτό δεν αλλάζει το γεγονός ότι τα εισαγόμενα δεδομένα πρέπει να περάσουν από προκαταρκτικό έλεγχο ορθότητας. Αυτή η τεχνική υπαγορεύει ότι πρώτο μέλημα αποτελεί ο ορισμός του SQL κώδικα και μετέπειτα ακολουθεί η παροχή των παραμέτρων, κάθε μία από τις οποίες περιέχει κάποια τιμή από τα δεδομένα εισόδου των χρηστών. Η σαφής διάκριση του κώδικα του αιτήματος και των δεδομένων που αποτελούν τα κριτήρια αναζήτησης, αποτρέπει

την εισαγωγή εντολών από τους χρήστες, καθώς τέτοιου είδους εντολές θα θεωρούνταν κριτήριο αναζήτησης και θα αντιμετωπιζόταν ανάλογα. Το API που προσφέρει η βιβλιοθήκη TypeORM εκτελεί τα αιτήματα με τον τρόπο που περιγράφηκε παραπάνω. Σε περιπτώσεις όπου δεν χρησιμοποιείται το API, τότε λαμβάνεται μέριμνα να γίνεται επίσης χρήση παραμετροποιημένων αιτημάτων. (SQL Injection Prevention, n.d.) (Έγχυση SQL, 2021)

Η χρήση ασφαλούς σύνδεσης (HTTPS) προσφέρει κρυπτογράφηση δεδομένων από το πρωτόκολλο SSL, διασφαλίζοντας την προστασία τους από υποκλοπή μέσω επιθέσεων MITM.

3.2.12 Διαχείριση σφαλμάτων

Όπως και σε κάθε εφαρμογή, σε περίπτωση σφάλματος ή προβλήματος, παρουσιάζεται κάποια εξαίρεση και μέσω εκείνης, οι χρήστες ειδοποιούνται για το λόγο αποτυχίας του αιτήματός τους. Το framework δίνει εργαλεία και οδηγίες για τον χειρισμό των σφαλμάτων. Επειδή πρόκειται για διαδικτυακή εφαρμογή τα σφάλματα θα πρέπει να μεταφράζονται σε μια συγκεκριμένη δομή που θα επιστρέφεται ως απάντηση και θα περιέχει τόσο το μήνυμα περιγραφής, όσο και τον κατάλληλο κωδικό κατάστασης, ενώ επίσης περιλαμβάνουν και ένα πεδίο χρονικής σήμανσης (timestamp). Οι τριψήφιοι κωδικοί κατάστασης που αφορούν σφάλματα του χρήστη ξεκινάνε με τέσσερα, ενώ εκείνα που ευθύνεται ο εξυπηρετητής ξεκινούν με πέντε. Το framework είναι αρκετά ευέλικτο όσον αφορά τη διαχείριση των εξαιρέσεων. Διαθέτει τη κλάση HttpException, η οποία μπορεί να χρησιμοποιηθεί ως έχει σε συνδυασμό με την παροχή κάποιων παραμέτρων, όπως το μήνυμα σφάλματος και ο κωδικός. Επίσης μπορούν να υπάρχουν παράγωγες κλάσεις αυτής, που θα διαθέτουν εκ προεπιλογής το κατάλληλο μήνυμα και κωδικό, αφού αυτά δηλώνονται στους κατασκευαστές τους. Όταν χρειαστεί να παρουσιαστεί κάποια εξαίρεση τότε θα πρέπει να δημιουργηθεί ένα instance της αντίστοιχης κλάσης. Είναι δυνατό να χρησιμοποιηθεί και το αντικείμενο Error που διαθέτει η JavaScript, καθώς και οι παράγωγες κλάσεις της. Το framework φροντίζει ώστε ακόμα και αυτό το γενικό σφάλμα να μετατραπεί σε ορθή απάντηση πριν αποσταλεί στους χρήστες. Για τη συγκεκριμένη περίπτωση ως απάντηση θα σταλεί μια δομή JSON με το μήνυμα σφάλματος και τον κωδικό πεντακόσια.

Αυτή η μετατροπή είναι αποτέλεσμα του ενσωματωμένου φίλτρου σφαλμάτων (exception filter). Τα φίλτρα αναλαμβάνουν δράση όταν παρουσιάζονται συγκεκριμένα σφάλματα και εκτελούν κώδικα σχετικά με τη διαχείρισή τους, καθώς και τη μορφοποίηση της απάντησης που θα σταλεί στους χρήστες. Δίνεται η δυνατότητα χρήσης πολλών τέτοιων φίλτρων για τη διαχείριση διαφορετικών τύπων σφαλμάτων. Για παράδειγμα, ένα τέτοιο φίλτρο χρησιμοποιείται για να υλοποιηθεί η τεχνική που περιγράφηκε στην υποενότητα των χρηστών για να εμποδιστεί η επαλήθευση τυχαίων αναγνωριστικών. Φιλτράρει σφάλματα σχετικά με τη πρόσβαση σε μη υπάρχοντες ή μη εξουσιοδοτημένους πόρους και κατασκευάζει ένα πιο γενικό μήνυμα σφάλματος. Τέλος, στις περιπτώσεις που μπορεί να υπάρχουν πολλαπλά μηνύματα σφάλματος, όπως για παράδειγμα όταν γίνεται επικύρωση κάποιας φόρμας, τότε προστίθεται στη απάντηση ακόμα ένα πεδίο, το errors, το οποίο περιέχει ένα πίνακα αντικειμένων. Αυτά τα αντικείμενα περιέχουν δύο ιδιότητες, το property που υποδεικνύει ποιο πεδίο έχει λανθασμένο περιεχόμενο και την ιδιότητα constraints, τιμή της οποίας είναι ένας πίνακας αλφαριθμητικών με τους κανόνες (περιορισμούς) που παραβιάστηκαν.

3.2.13 Δοκιμές (Testing)

Για τον έλεγχο λειτουργίας της εφαρμογής του εξυπηρετητή γράφτηκαν end-to-end (e2e) tests που ελέγχουν τη λειτουργία των services εκτελώντας κλήσεις σε endpoints των controllers. Το framework περιέχει εκ προεπιλογής τα απαραίτητα εργαλεία ελέγχου, τα οποία είναι οι βιβλιοθήκες supertest και jest. Δημιουργήθηκαν tests για συγκεκριμένους controllers, όπου γίνονται κλήσεις σε endpoints τους και ελέγχεται το αποτέλεσμα που επιστρέφεται. Ο έλεγχος του αποτελέσματος είναι η σύγκρισή της απάντησης με τα αναμενόμενα αποτελέσματα που έχουν υπολογιστεί προηγουμένως βάσει των εισαγόμενων δεδομένων. Οι κλήσεις εξυπηρετούνται από έναν εξυπηρετητή που εκτελείται σε δοκιμαστικό περιβάλλον. Κάθε test περιλαμβάνει βήματα πριν και μετά την εκτέλεση. Το αρχικό βήμα συνήθως περιλαμβάνει τον αφαίρεση δεδομένων από την βάση δεδομένων του δοκιμαστικού περιβάλλοντος, καθώς είναι πιθανό να δημιουργήθηκαν από κάποια προηγούμενη εκτέλεση του test. Επιπλέον, εισάγονται τα απαραίτητα δεδομένα σε αυτή, βάσει των οποίων και σε συνδυασμό με την εκτέλεση της δοκιμής αναμένονται τα επιθυμητά αποτελέσματα. Το τελικό βήμα αφορά τον τερματισμό των εργαλείων που απαιτούνταν για το πρώτο

βήμα, όπως για παράδειγμα είναι ο τερματισμός της σύνδεσης με τη βάση δεδομένων.

3.2.14 Εγκατάσταση

Η εγκατάσταση της εφαρμογής γίνεται σε περιβάλλον Ubuntu Server 20.04 με αυτό-φιλοξενία (self-hosted).

Εγκατάσταση βάσης δεδομένων PostgreSQL

Το PostgreSQL Apt Repository παρέχει την τελευταία έκδοση της βάσης. Παρακάτω προστίθεται το αντίστοιχο repository, εισάγεται το δημόσιο κλειδί και γίνεται ανανέωση της λίστας των πακέτων.

```
sudo sh -c 'echo "deb http://apt.postgresql.org/pub/repos/apt $(lsb_release -cs)-pgdg main" > /etc/apt/sources.list.d/pgdg.list'

wget --quiet -O - https://www.postgresql.org/media/keys/ACCC4CF8.asc | sudo apt-key add -

sudo apt-get update
```

Η PostgreSQL (και εργαλεία της) εγκαθίσταται με τα κάτωθι πακέτα:

```
sudo apt install postgresql postgresql-contrib
```

Πρόσβαση στο shell της βάσης με χρήση του superuser postgres:

```
sudo -u postgres psql
```

Δημιουργούμε τη βάση δεδομένων για χρήση από την εφαρμογή:

```
create database kchap;
```

Μέσω του shell θα δημιουργηθεί ένας νέος ρόλος (χρήστης) στη βάση για χρήση μόνο από την εφαρμογή:

```
create user kchap with encrypted password '<SECRET_PASSWORD>';
```

Για λόγους ασφαλείας, η πρόσβαση στη βάση περιορίζεται στους superusers και τον χρήστη kchap που μόλις δημιουργήθηκε.

```
revoke connect on database kchap from public;

grant connect on database kchap to kchap;
```

Σύνδεση στη βάση:

```
\c kchap
```

Σε αυτό το σημείο είναι δυνατός ο περιορισμός των δικαιωμάτων του νέου χρήστη στα απολύτως απαραίτητα και περιλαμβάνουν μόνο εντολές τύπου DML. Η δεύτερη εντολή αφορά την πρόσβαση και τη χρήση των αλληλουχιών (sequences), τα οποία είναι αντικείμενα που χρησιμοποιούνται για την παραγωγή μοναδικών κλειδιών. Παράδειγμα αποτελούν κάποια από τα κύρια κλειδιά της εφαρμογής που είναι τύπου serial και αυξάνονται σειριακά. Οι παρακάτω εντολές δεν αφορούν υπάρχοντα αντικείμενα (πχ. πίνακες), αλλά νέα. Ο χρήστης postgres είναι εκείνος που θα δημιουργήσει τα νέα αντικείμενα, γι' αυτό και δηλώνεται στο αίτημα.

```
alter default privileges for role postgres in schema public grant select, insert,
update, delete on tables to kchap;

alter default privileges for role postgres in schema public grant usage, select on
sequences to kchap;
```

Εν τέλει, εκτελούμε το script της αρχικοποίησης της βάσης για να δημιουργηθούν όλα τα απαραίτητα αντικείμενα.

Εγκατάσταση βάσης δεδομένων MongoDB

Εισαγωγή του δημόσιου κλειδιού και του repository, καθώς και ανανέωση της λίστας πακέτων:

```
wget --quiet -O - https://www.mongodb.org/static/pgp/server-5.0.asc | sudo apt-key
add -

echo "deb [ arch=amd64,arm64 ] https://repo.mongodb.org/apt/ubuntu focal/mongodb-
org/5.0 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-5.0.list

sudo apt-get update
```

Η MongoDB εγκαθίσταται με την εντολή:

```
sudo apt-get install -y mongodb-org
```

Εκκίνηση της υπηρεσίας (service) και προγραμματισμός εκτέλεσής του σε κάθε εκκίνηση του συστήματος:

```
sudo systemctl start mongod
sudo systemctl enable mongod
```

Πρόσβαση στο shell της βάσης και σύνδεση στη βάση admin:

```
mongosh
use admin
```

Ενεργοποίηση του Ελέγχου πρόσβασης, με δημιουργία του χρήστη-διαχειριστή με τη μέθοδο SCRAM (χρήση κωδικού πρόσβασης), ώστε να επιβληθεί η ανάγκη αυθεντικοποίησης και εξουσιοδότησης (δικαιώματα) των χρηστών της βάσης για αυξημένα επίπεδα ασφάλειας.

```
db.createUser(
  {
    user: "userAdmin",
    pwd: "<SECRET_PASSWORD>",
    roles: [
      { role: "userAdminAnyDatabase", db: "admin" },
      { role: "readWriteAnyDatabase", db: "admin" }
    ]
  }
)
```

Δημιουργία και σύνδεση στη βάση της εφαρμογής:

```
use kchap
```

Δημιουργία συλλογής (collection):

```
db.createCollection('sessions')
```

Δημιουργία του χρήστη για τη βάση της εφαρμογής:

```
db.createUser( { user: "kchap",
                 pwd: "<SECRET_PASSWORD>",
                 roles: [ "readWrite" ] } )
```

Έξοδος από το shell:

```
.exit
```

Άνοιγμα του αρχείου διαμόρφωσης της βάσης και ενεργοποίηση της αυθεντικοποίησης:

```
sudo nano /etc/mongod.conf
```

```
security:
 authorization: enabled
```

Επανεκκίνηση της υπηρεσίας για εφαρμογή των αλλαγών του αρχείου διαμόρφωσης:

```
sudo systemctl restart mongod
```

Εγκατάσταση Node.js

Το Node.js (μαζί με το npm) εγκαθίσταται με την παρακάτω εντολή. Το Node.js είναι το περιβάλλον εκτέλεσης JavaScript, δηλαδή εκτελεί την εφαρμογή, ενώ το npm είναι ο προεπιλεγμένος διαχειριστής πακέτων για το Node.js, δηλαδή είναι υπεύθυνο για τις βιβλιοθήκες.

```
curl -fsSL https://deb.nodesource.com/setup_current.x | sudo -E bash -
sudo apt-get install -y nodejs
```

Προετοιμασία εγκατάστασης εφαρμογής

Η λήψη της εφαρμογής θα γίνει μέσω μιας υπηρεσίας φιλοξενίας αποθετηρίου πηγαίου κώδικα τύπου Git.

```
git clone https://KleonChirakis@bitbucket.org/kchappTeam/kchap-server.git
```

Θα πρέπει να εγκατασταθούν τα απαραίτητα εργαλεία για την κατασκευή της εφαρμογής. Αυτά είναι το εργαλείο διεπαφής γραμμής εντολών (command-line interface tool) του Nest Framework, καθώς και ο διαχειριστής πακέτων Yarn που χρησιμοποιείται για τη λήψη πακέτων. Το module “copyfiles” χρησιμοποιείται κατά την κατασκευή της εφαρμογής για την αντιγραφή στατικών αρχείων στο φάκελο public, ώστε να είναι δημόσια.

```
npm i -g @nestjs/cli yarn copyfiles
```

Στη συνέχεια πραγματοποιείται πλοήγηση στον κατάλογο που φιλοξενεί την εφαρμογή, ώστε να γίνει εγκατάσταση των πακέτων (βιβλιοθηκών) της. Η σημαία «production», δηλώνει ότι δεν θα πρέπει να γίνει λήψη των πακέτων που δηλώνονται στη λίστα «devDependencies» του αρχείου «package.json». Αυτό περιέχει πληροφορίες σχετικά με την εφαρμογή, καθώς και τα πακέτα που απαιτεί για τη λειτουργία της. Η συγκεκριμένη λίστα αφορά πακέτα που απαιτούνται κατά την ανάπτυξη.

```
yarn install --production
```

Πριν την κατασκευή της εφαρμογής θα πρέπει να μεταβληθεί το αρχείο “config.json”, που περιέχει πληροφορίες για τη διαμόρφωση της εφαρμογής. Κάποιες από αυτές, αφορούν τους κωδικούς πρόσβασης ή τα URL σύνδεσης των βάσεων δεδομένων. Η κατασκευή της εφαρμογής γίνεται με την παρακάτω εντολή:

```
nest build
```

Εντός του ριζικού καταλόγου της εφαρμογής θα δημιουργηθεί ένας νέος κατάλογος “dist” που περιέχει τον μεταφρασμένο σε JavaScript κώδικα, ώστε να εκτελεστεί από το Node.js.

Εγκατάσταση PM2 και εφαρμογής

Το PM2 είναι μια υπηρεσία διαχείρισης εφαρμογών για το Node.js. Έχει πολλά πλεονεκτήματα, όπως η αυτόματη εκκίνηση της εφαρμογής σε περίπτωση απρόσμενου τερματισμού, ώστε να είναι πάντα online. Επιπλέον προσφέρει εργαλεία παρακολούθησης, θέσπιση ορίων κατανάλωσης μνήμης, ικανότητες καταγραφής συμβάντων κ.α.. Εγκαθίσταται μέσω του npm με την εντολή:

```
npm install -g pm2
```

Η πρόσθεση της εφαρμογής στο PM2 γίνεται με την παρακάτω εντολή. Η μεταβλητή περιβάλλοντος NODE_ENV θέτεται στην τιμή “production” για να γνωστοποιηθεί στην εφαρμογή, ότι η εκτέλεση γίνεται σε περιβάλλον παραγωγής. Σημειώνεται ότι κάποια κομμάτια εντός της εφαρμογής συμπεριφέρονται διαφορετικά σε περιβάλλον δοκιμών (test), ώστε να γίνεται ευκολότερη η ανάπτυξη.

```
NODE_ENV=production pm2 start apps/kchap-server/dist/apps/http/src/main.js
```

Εκκίνηση του PM2 κατά την εκκίνηση του (λειτουργικού) συστήματος. Με την εισαγωγή της παρακάτω εντολής, θα εμφανιστεί νέα κατάλληλη εντολή που θα πρέπει να εκτελεστεί και αφορά στο λειτουργικό σύστημα που εκτελεί το PM2.

```
pm2 startup
```

Ενεργοποίηση του Firewall

Πριν διαμορφωθεί ο αντίστροφος διακομιστής μεσολάβησης, θα πρέπει να ενεργοποιηθούν οι θύρες του τείχους προστασίας (firewall) του λειτουργικού ώστε να επιτρέπεται η επικοινωνία με αυτές. Πριν από αυτό, θα πρέπει να ενεργοποιηθεί το προεπιλεγμένο εργαλείο διαχείρισης του τείχους προστασίας στα Ubuntu. Μετέπειτα θα πρέπει να ενεργοποιηθούν οι θύρες 80 (http) και 443 (https), που χρησιμοποιούνται για την απλή και κρυπτογραφημένη επικοινωνία με τον διακομιστή.

```
ufw enable
ufw allow https
ufw allow http
```

Εγκατάσταση NGINX

Το NGINX είναι ένας διακομιστής ιστού με πληθώρα δυνατοτήτων για πολλές εφαρμογές, όπως για εξισορρόπηση φορτίου ή διαμεσολάβηση αλληλογραφίας. Στην παρούσα διπλωματική χρησιμοποιείται ως αντίστροφος διακομιστής μεσολάβησης (reverse proxy server). Ένας τέτοιος διακομιστής τοποθετείται σε ένα

ιδιωτικό δίκτυο με σκοπό την προώθηση αιτημάτων στον κατάλληλο backend εξυπηρετητή. Η εγκατάσταση πραγματοποιείται με την εντολή:

```
sudo apt install nginx
```

Θα πρέπει να μεταβληθεί το κατάλληλο αρχείο διαμόρφωσης, ώστε το NGINX να προωθεί τα αιτήματα στον διακομιστή που εκτελεί το PM2. Τα αιτήματα των χρηστών εξυπηρετούνται από τη διεύθυνση «kchap.ddns.net», στην προεπιλεγμένη θύρα 80. Ο αντίστροφος διακομιστής μεσολάβησης θα αποστέλλει αυτά τα αιτήματα στην θύρα 3000 που εκτελείται ο εξυπηρετητής της εφαρμογής.

```
sudo nano /etc/nginx/sites-available/default
```

```
server {
	listen		80;
	server_name	kchap.ddns.net;


	location / {
		proxy_set_header X-Forwarded-Proto $scheme;
		proxy_pass http://localhost:3000;
	}
```

Μετά την αποθήκευση του αρχείου διαμόρφωσης, θα πρέπει να ελέγχει η εγκυρότητά του και να επανεκκινηθεί η υπηρεσία του NIGINX, ώστε να εφαρμοστούν οι αλλαγές:

```
sudo nginx -t
sudo systemctl restart nginx
```

Εγκατάσταση κρυπτογράφησης SSL/TLS

Τέλος εγκαθίσταται το πιστοποιητικό που ενεργοποιεί την κρυπτογράφηση των δεδομένων παρέχοντας την απαιτούμενη ασφάλεια στην επικοινωνία μεταξύ του εξυπηρετητή και των πελατών. Η απόκτηση του πιστοποιητικού γίνεται μέσω της δωρεάν υπηρεσίας Let’s encrypt και τοποθετείται στο NGINX. Τα παραπάνω εκτελούν οι εντολές:

```
sudo snap install core; sudo snap refresh core
sudo snap install --classic certbot
sudo ln -s /snap/bin/certbot /usr/bin/certbot
sudo certbot --nginx
```

Το πρόγραμμα certbot θα εμφανίσει κάποιες ερωτήσεις που θα κληθούμε να απαντήσουμε και εν συνεχεία θα λάβει και θα εγκαταστήσει το πιστοποιητικό.

ΚΕΦΑΛΑΙΟ 4

FRONT END

4.1 Βάση δεδομένων

Ο πελάτης έχει την ικανότητα να αποθηκεύει τοπικά τα δεδομένα, ώστε οι χρήστες να μπορούν να ανατρέχουν σε αυτά. Αυτό επιτρέπει την ανάγνωση τους, ακόμα και χωρίς σύνδεση στο διαδίκτυο. Η αποθήκευση των δεδομένων γίνεται με χρήση μιας βάσης δεδομένων, της SQLite, που αποτελεί την προεπιλεγμένη βάση του Android.

Η τοπική βάση του πελάτη είναι παρόμοια με εκείνη του εξυπηρετητή, καθώς και οι δύο διαθέτουν αρκετά κοινά χαρακτηριστικά και έτσι συναντιόνται οι ίδιοι πίνακες. Παρ' όλα αυτά παρατηρούνται κάποιες διαφορές, κυρίως όσον αφορά τις στήλες, καθώς αποστέλλεται ένα μέρος των δεδομένων που διατηρεί ο εξυπηρετητής, ενώ προστίθενται νέες που εξυπηρετούν τους σκοπούς της εφαρμογής-πελάτη. Συγκεκριμένα:

- Προστέθηκε ο πίνακας auto_ep, ο οποίος αποτελεί τον προσωρινό χώρο αποθήκευσης των AutoEp. Τα πεδία του περιλαμβάνουν το αυτόματα αυξανόμενο κύριο κλειδί και αναγνωριστικό (id), την ομάδα, το πότε δημιουργήθηκε η καταχώρηση και το remote_id. Το τελευταίο αποτελεί το αναγνωριστικό της Αλλαγής που δημιουργήθηκε όταν έγινε αποδεκτό το AutoEp και προκάλεσε μεταβολές στα υπόλοιπα των μελών. Επειδή το AutoEp αποτελεί μια μορφή μεταφοράς, επιπλέον διαθέτει τα πεδία του ποσού, καθώς και τα πεδία from_user_id, to_user_id και amount. Περισσότερα σχετικά με αυτή την οντότητα θα δοθούν στο επόμενο κεφάλαιο.
- Προστέθηκε ο πίνακας groups_users_iou, ο οποίος περιγράφει τα ποσά που χρωστάει ένας χρήστης σε κάποιον άλλο. Αποτελείται από τέσσερα πεδία: πιστωτή (creditor_id), χρεώστη (debtor_id), ομάδα (group_id) και ποσό (amount). Τα τρία πρώτα πεδία αποτελούν και το κύριο κλειδί του πίνακα. Παρόμοια με την PostgreSQL, και στην SQLite, λόγω του σύνθετου κύριου κλειδιού γίνεται αυτόματη ευρετηρίαση των στηλών του, αλλά με βάση την στήλη που βρίσκεται πιο αριστερά. Αυτό σημαίνει ότι αρχικά ευρετηριάζει την πρώτη στήλη του πιστωτή, μετά τον συνδυασμό της πρώτης με τη στήλη του

χρεώστη και τέλος τον συνδυασμό όλων των στηλών που αποτελούν το κύριο κλειδί. Από τα παραπάνω προκύπτει ότι η στήλη του πιστωτή έχει ευρετηριαστεί. Οι υπόλοιπες δύο ευρετηριάζονται με τη δημιουργία ατομικών κανόνων (indexes), καθώς αποτελούν φίλτρα για κάποια αιτήματα προς τη βάση. (The SQLite Query Optimizer Overview, n.d.)

- Προηγουμένως, έγινε αναφορά στο ότι ο πελάτης οφείλει να αποστείλει τα αναγνωριστικά των ήδη εφαρμοσμένων Αλλαγών. Αυτά ανακτώνται από τον πίνακα change. Κάθε αποστολή AutoEp ή Περιεχομένου θα δημιουργήσει μια καταχώρηση στον πίνακα, εισάγοντας το αναγνωριστικό της Αλλαγής, όπως αυτό δόθηκε από τον εξυπηρετητή, καθώς και το αναγνωριστικό της ομάδας. Θα χρησιμοποιηθούν στον επόμενο συγχρονισμό, ώστε να παραλειφθούν οι ήδη εφαρμοσμένες Αλλαγές και έπειτα θα διαγραφούν.
- Στον πίνακα groups συναντιούνται όλες οι στήλες που υπάρχουν και στον εξυπηρετητή, καθώς και νέες. Συγκεκριμένα προστίθενται οι στήλες change_id, min_transaction_change_id και min_transfer_change_id. Η πρώτη σχετίζεται με το αναγνωριστικό της τελευταίας συγχρονισμένης Αλλαγής, ενώ οι υπόλοιπες διατηρούν το αναγνωριστικό της παλαιότερης Αλλαγής όσον αφορά τις Συναλλαγές και τις Μεταφορές.
- Στον πίνακα groups_users έχει αφαιρεθεί η στήλη joined_on, καθώς δεν εμφανίζεται ως πληροφορία στον χρήστη.
- Ο πίνακας users περιέχει μόνο τα πεδία του αναγνωριστικού και του ονόματος, καθώς τα πεδία email και isDeleted προορίζονται για χρήση μόνο από τον εξυπηρετητή.
- Οι πίνακες που αφορούν Περιεχόμενο, δηλαδή οι πίνακες transaction και transfer, δεν περιέχουν τη στήλη uploaded_on.
- Προστέθηκε ο πίνακας logged_in_user που διατηρεί τις πληροφορίες του συνδεδεμένου χρήστη με πεδία το αναγνωριστικό, το όνομα, τη διεύθυνση ηλεκτρονικού ταχυδρομείου, τον πάροχο σύνδεσης και το αναγνωριστικό της συσκευής. Επιλέχθηκε κυρίως για την εκμετάλλευση της δυνατότητας της ασύγχρονης παρατήρησης για μεταβολές των δεδομένων.

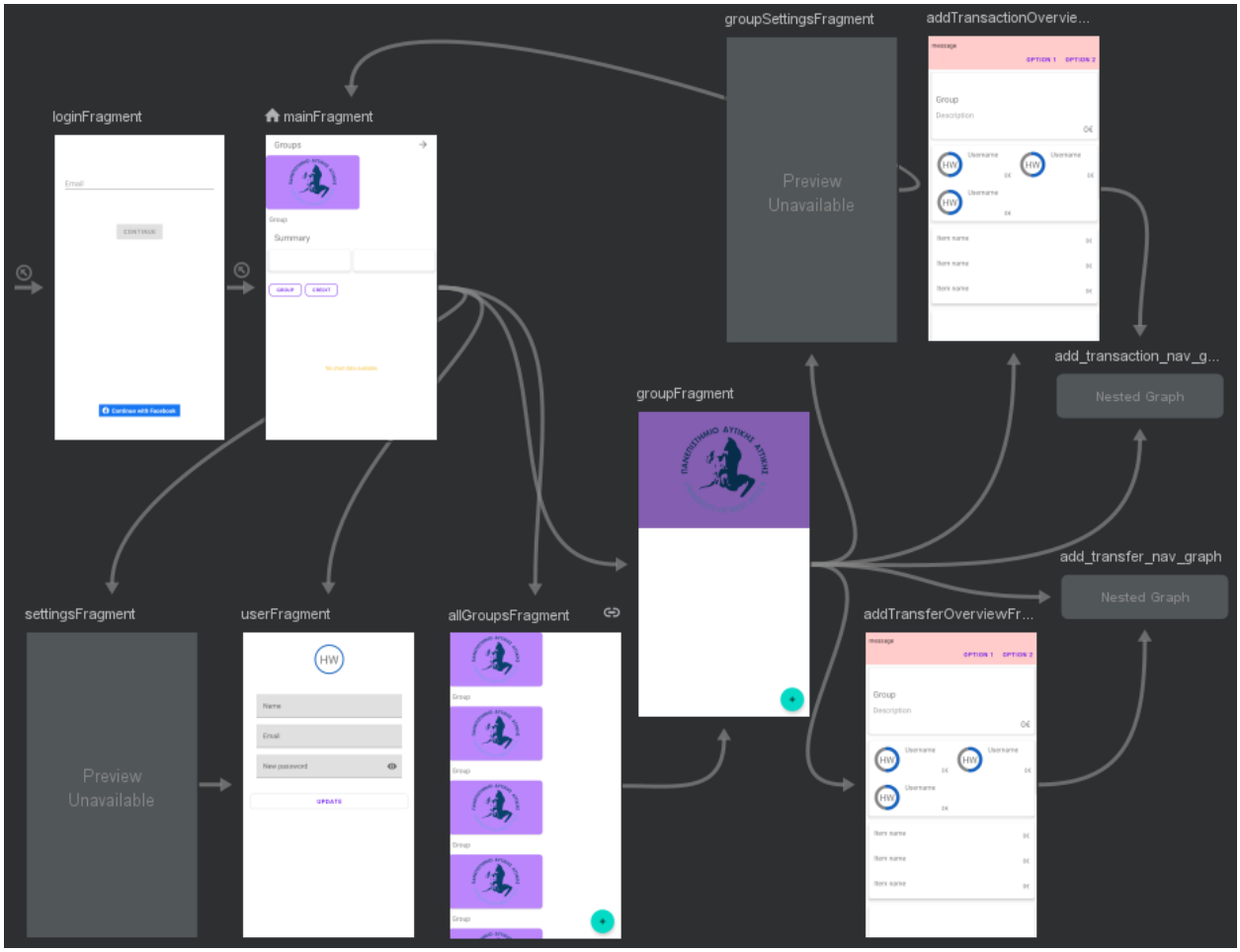
4.2 Εφαρμογή

4.2.1 Περιγραφή λειτουργίας εφαρμογής πελάτη

Η εφαρμογή του πελάτη επιφορτίζεται με την διαχείριση των δεδομένων του συστήματος, καθώς και με τον υπολογισμό των ροών χρέους των μελών. Ως διαχείριση νοείται η ανάγνωση, εγγραφή ή επεξεργασία των δεδομένων. Δεδομένα θεωρούνται τόσο οι ομάδες και τα μέλη τους, όσο και το Περιεχόμενό τους. Ροή χρέους αποτελεί η εντολή καταβολής ενός ποσού από κάποιον χρεώστη ως προς κάποιον πιστωτή.

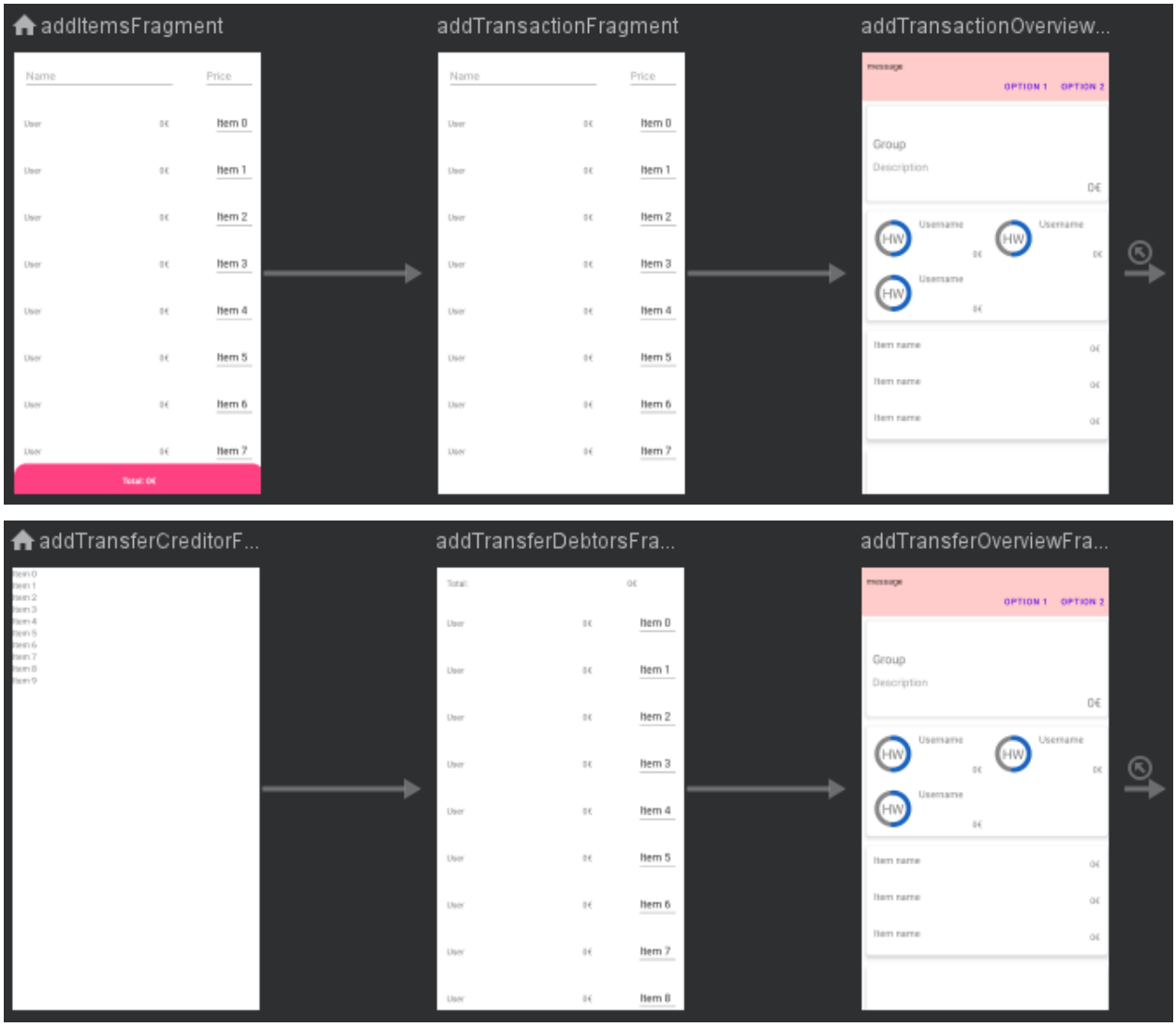
Παρόμοια με τον εξυπηρετητή, το νόμισμα και η γλώσσα παραμένουν κοινές. Όσον αφορά τα δεδομένα χρόνου, αυτά λαμβάνονται, στέλνονται και αποθηκεύονται στο πρότυπο χρόνου UTC. Στις περιπτώσεις που απαιτείται η προβολή ή η επεξεργασία τους, τότε μεμονωμένα μετατρέπονται σε τοπική ώρα και εν συνεχεία αποθηκεύονται ή/και αποστέλλονται σε UTC.

Η περιήγηση στην εφαρμογή γίνεται με χρήση της βιβλιοθήκης Navigation του Android Jetpack. Η σχεδίαση των διαδρομών περιλαμβάνει την πρόσθεση οθονών (Fragments) και τη σύνδεση μεταξύ τους με τη μορφή ενεργειών. Το τελικό αποτέλεσμα αποτελεί ένα γράφημα διαδρομών.



Εικόνα 4.1: Κεντρικό γράφημα διαδρομών πλοήγησης

Κάποιες διαδρομές μπορούν να οδηγούν και σε υπογραφήματα, όπως αυτά της πρόσθεσης/επεξεργασίας συναλλαγής και μεταφοράς. Αυτό συνεισφέρει στην επαναληψιμότητα και την οργάνωση του κώδικα, ενώ σε συνεργασία με τη βιβλιοθήκη έκχυσης εξαρτήσεων Hilt, είναι δυνατό να τεθεί η εμβέλεια του ViewModel στο (ύπο)γράφημα.



Εικόνα 4.2: Υπογράφηκα συναλλαγών (Πάνω) και μεταφορών (Κάτω)

Από τις παραπάνω εικόνες, τονίζεται η χρηστικότητα της επαναχρησιμότητας, καθώς είναι εμφανές ότι η τελευταία οθόνη των δύο υπογραφημάτων είναι κοινή, ενώ χρησιμοποιείται ακόμα δύο φορές στο κεντρικό γράφημα.

4.2.2 Views

Στο Android τα Views αποτελούν το βασικό δομικό στοιχείο για την κατασκευή εξαρτημάτων που συνθέτουν το περιβάλλον χρήστη. Απλά εξαρτήματα κληρονομούν την κλάση View, ενώ πιο σύνθετα υλοποιούνται από παράγωγες κλάσεις της κλάσης ViewGroup, η οποία με τη σειρά της κληρονομεί τη View. Η διαφορά έγκειται στο γεγονός ότι η πρώτη, αποτελεί μια κλάση που περιλαμβάνει μια τετράγωνη περιοχή για τη σχεδίαση ενός εξαρτήματος, ενώ στη δεύτερη έχει την ιδιότητα να φιλοξενεί

πολλαπλά Views. Για τις ανάγκες της εφαρμογής έχουν κατασκευαστεί δύο Views, βασισμένα στο FrameLayout, το οποίο αποτελεί υποκλάση της ViewGroup. (View, n.d.) (ViewGroup, n.d.)

Το CircleProgressBar, χρησιμεύει στην αναπαράσταση των αρχικών γραμμάτων του ονόματος ενός μέλους μέσα σε ένα κύκλο προόδου που χρησιμοποιείται για στατιστικούς λόγους. Η επιλογή της βασικής κλάσης (FrameLayout) ορίζεται στο ότι συγχωνεύει το View της βιβλιοθήκης CircularProgressbar του Ankit Kumar, η οποία παρέχει τον κύκλο προόδου, μαζί με ένα TextView για την αναπαράσταση των αρχικών. Η βιβλιοθήκη έχει τροποποιηθεί ώστε να μπορεί να εμφανίζει ακόμα και δύο κύκλους προόδου. Στη συνέχεια, βάσει των διαστάσεων του στοιχείου (ύψος, πλάτος) και του μεγέθους του κειμένου, γίνονται υπολογισμοί για την καλύτερη δυνατή αναπαράσταση.

Εικόνα 4.3: Διαφορετικές καταστάσεις εμφάνισης του CircleProgressBar

Επιπλέον, δημιουργήθηκε το View ItemUsersStatsView, σκοπός του οποίου είναι η εμφάνιση πολλαπλών Views τύπου CircleProgressBar εντός ενός πλαισίου (FrameLayout). Παρομοίως, το View διαχειρίζεται το τρόπο που εμφανίζεται με βάση τον διαθέσιμο χώρο εντός του πλαισίου που ορίζεται από τις διαστάσεις του View. Ο μέγιστος αριθμός CircleProgressBar Views που μπορούν να εμφανιστούν μέσα στο πλαίσιο καθορίζεται από τον διαθέσιμο χώρο ή/και ορισμό της αντίστοιχης παραμέτρου από τον χρήστη. Αν τα μέλη είναι περισσότερα από τα Views που μπορεί να φιλοξενήσει το πλαίσιο, τότε στο τέλος εμφανίζεται ένα TextView που απαριθμεί τα μέλη που απομένουν. Συνοπτικά, για να επιτευχθεί αυτό, εμφανίζει το πρώτο CircleProgressBar, ώστε να υπολογιστούν οι διαστάσεις τόσο του ίδιου του View, όσο και του FrameLayout και εν συνεχεία εμφανίζει τα υπόλοιπα.

Εικόνα 4.4: Το ItemUsersStatsView

4.2.3 Domain Models

Για τα domain models εφαρμογής του πελάτη ισχύει ότι έχει ειπωθεί και στο αντίστοιχο κεφάλαιο του εξυπηρετητή. Οι domain οντότητες είναι:

- GroupUserEntity: Μέλος της ομάδας
- GroupEntity: Ομάδα
- UserEntity: Χρήστης συστήματος
- GroupUserIOUEntity: Ροή χρέους μεταξύ δύο μελών μιας ομάδας
- ChangeEntity: Διατηρεί το αναγνωριστικό των Αλλαγών για τις οποίες ευθύνεται η παρούσα συσκευή, ώστε να εξαιρεθούν της λήψης κατά τον συγχρονισμό
- LoggedInUser: Συνδεδεμένος χρήστης
- TransactionEntity: Συναλλαγή
- TransferEntity: Μεταφορά
- AutoEpEntity: Οντότητα που διατηρεί την αυτοματοποιημένη εντολή, αναμένοντας έγκριση από τον εξυπηρετητή

Σημειώνεται ότι χρήση DTOs και mappers και σε αυτή την περίπτωση κρίνεται απαραίτητη, καθώς τα δεδομένα που εισάγονται και εξάγονται διαφέρουν. Όπως είναι φυσικό σε τέτοιες περιπτώσεις, δεδομένα που μοιράζονται κοινές ιδιότητες, κάνουν χρήση της κληρονομικότητας.

4.2.4 Υλοποίηση αρχιτεκτονικής

Σε αντίθεση με τον εξυπηρετητή, η εφαρμογή του πελάτη οργανώνει τα πακέτα βάσει της μεθόδου package by layer. Αυτά είναι:

- data: Αφορά τα δεδομένα της εφαρμογής. Συγκριμένα περιέχει τα ακόλουθα πακέτα:
    - adapter: Περιέχει αντικείμενα που σχετίζονται με τους Adapters. Αυτοί αποτελούν αντικείμενα γέφυρες ανάμεσα στα δεδομένα και την εμφάνισή τους. Διαθέτοντας πρόσβαση στα δεδομένα, κατασκευάζει τα Views που τα αναπαριστούν (Adapter, n.d.)
    - mapper: Περιλαμβάνει τους κατάλληλους Mappers για τη μετατροπή των δεδομένων
    - model: Διατηρεί όλες τις οντότητες

    - repository: Περιέχει τις πηγές δεδομένων, καθώς και τα Repositories που τις διαχειρίζονται
- di: Περικλείει τα Modules της βιβλιοθήκης Hilt
- exception: Φιλοξενεί τις κλάσεις τύπου Throwable, συμπεριλαμβανομένου και του τύπου Exception, καθώς και τη διαχείριση των σφαλμάτων
- service: Περιλαμβάνει κλάσεις που επιτελούν εργασίες για συγκεκριμένους σκοπούς, όπως είναι εκείνες που υλοποιούν αλγορίθμους για τον διαμοιρασμό των χρεών ή για την εύρεση των ροών χρέους
- ui: Φιλοξενεί τα αρχεία που έχουν κάποια συσχέτιση με το περιβάλλον χρήσης (UI). Αυτά είναι οι οθόνες του επιπέδου View (Fragments) και τα ViewModels
- util: Βοηθητικές κλάσεις και συναρτήσεις
- view: Αποτελείται από τα κατασκευασμένα από τον προγραμματιστή Views της εφαρμογής. Επομένως, σε αυτό περιέχονται τα CircleProgressBar και ItemUsersStatsView
- web: Μέρος του αποτελούν κλάσεις που χρησιμοποιούνται από τη βιβλιοθήκη OkHttp με την οποία εκτελούνται τα αιτήματα προς τον εξυπηρετητή. Ένα παράδειγμα είναι η τοποθέτηση του cookie σε κάθε αίτημα, ώστε να μπορεί να αυθεντικοποιηθεί. Επιπλέον περιέχει συναρτήσεις χειρισμού των cookies και της κατάστασης σύνδεσης, καθώς και το πρότυπο διεπαφής με όλες τις δυνατές κλήσεις προς τον εξυπηρετητή, το οποίο και υλοποιείται (αυτόματα) από τη βιβλιοθήκη Retrofit.

```
@POST("auth/login")
suspend fun login(
    @Body credentials: LocalLoginCredentials
): LoggedInUserEntity
```

Εικόνα 4.5: Παράδειγμα κλήσης endpoint

Σημειώνεται ότι το γνώριμο σύμβολο «@», το οποίο έχουν ως πρόθεμα οι decorators της TypeScript/JavaScript, χρησιμοποιείται και στην Java/Kotlin με την ονομασία annotations. Τα annotations είναι πιο πολύπλοκα σε σχέση με τη JavaScript. Παρέχουν δεδομένα στα στοιχεία που δηλώνονται, όπως σε κλάση, πρότυπο διεπαφής, κατασκευαστή, μέθοδο ή μεταβλητή. Χρησιμοποιούνται για παροχή πληροφοριών στον μεταγλωττιστή (compiler), την παραγωγή

αυτοματοποιημένου κώδικα κατά τη μεταγλώττιση ή την επεξεργασία κατά την εκτέλεση (runtime). (Lesson: Annotations, n.d.)

Κάθε οθόνη αποτελεί μια υποκλάση της Fragment (επίπεδο View) και μια (ή περισσότερες) κλάση ViewModel. Η πρώτη φιλοξενεί τα views της οθόνης, δηλαδή συνθέτει το UI. Το ViewModel διατηρεί και διαχειρίζεται τα δεδομένα σχετικά με το περιβάλλον χρήσης, λειτουργώντας αρμονικά με τον κύκλο ζωής (lifecycle) άλλων αντικειμένων, όπως είναι τα Activities και τα Fragments. Για παράδειγμα, η αλλαγή του προσανατολισμού της οθόνης προκαλεί την επαναδημιουργία των Activities, καταστρέφοντας τυχόν δεδομένα αποθηκευμένα εντός εκείνων. Η χρήση των ViewModels παραμένει ανεπηρέαστη από τέτοια γεγονότα, διατηρώντας τα δεδομένα και συνεχίζοντας απρόσκοπτα την εκτέλεση εργασιών. Επιπλέον, αποτελώντας τη γέφυρα μεταξύ των δεδομένων και του περιβάλλοντος χρήσης, αποκτά πρόσβαση στα δεδομένα, μπορεί να τα παρακολουθεί για τυχόν αλλαγές, προαιρετικά μπορεί να τα επεξεργαστεί, και στη συνέχεια να τα προωθήσει προς το επίπεδο View για εμφάνιση. Τα ViewModels μπορούν να είναι κοινά, ώστε να μοιράζονται τα δεδομένα τους ανάμεσα σε πολλές οθόνες, αλλά η άμεση επικοινωνία μεταξύ των ίδιων δεν είναι δυνατή.

Η εφαρμογή κάνει χρήση ασύγχρονων τεχνολογιών, παρακολουθώντας τις αλλαγές στα δεδομένα της βάσης, οι οποίες αποτελούν έναυσμα για την ενημέρωση του περιβάλλοντος χρήστη (UI). Υλοποιούνται από τα LiveData και Flow, ενώ σε συνδυασμό με το Data Binding εδραιώνεται απευθείας ενημέρωση του UI. Το πρώτο αποτελεί εξάρτημα με γνώση του κύκλου ζωής (lifecycle-aware component), οπότε και χρησιμοποιείται από τα VieModels/Fragments, ενώ το δεύτερο χρησιμοποιείται κυρίως από τα χαμηλότερα επίπεδα, όπως το επίπεδο δεδομένων.

Η υλοποίηση του Dependency Injection στηρίζεται στις δυνατότητες της βιβλιοθήκης Hilt. Αυτό επιτυγχάνεται είτε με αυτόματο τρόπο, είτε παρέχοντας τα κατάλληλα Modules που ενημερώνουν την βιβλιοθήκη πως να παραδώσει αντικείμενα των ζητούμενων τύπων.

Όλα τα παραπάνω γίνονται πιο ξεκάθαρα με την παρακάτω εικόνα που απεικονίζει ένα απόσπασμα του ViewModel για το χαρακτηριστικό των μεταφορών, το TransferViewModel. Αυτό περιλαμβάνει τις τοπικές μεταβλητές, κύριο μέρος εκ των οποίων διατηρεί την κατάσταση της οθόνης. Αυτές μπορούν να διακριθούν από τον τύπο δεδομένων LiveData.

```
/**
 * Shared ViewModel in transfer nested graph
 */
@HiltViewModel
class TransferViewModel
@Inject
constructor(
    private val groupDao: GroupDao,
    private val groupUserDao: GroupUserDao,
    private val transferRepository: TransferRepository,
    @ApplicationContext private val appContext: Context,
    savedStateHandle: SavedStateHandle
) : ContentViewModel<TransferFormState>() {

    private val transferId: Long =
        savedStateHandle.get<Long>("transferId")!!
    override var contentId: Long = transferId
    override val groupId: Long = (savedStateHandle.get<Long>("groupId")!!)

    private val _transferDebtors = MutableLiveData<List<Share>>()
    val transferDebtors: LiveData<List<Share>> = _transferDebtors

    private val _transferForm = MutableLiveData<TransferFormState>()
    override val stateForm: LiveData<TransferFormState> = _transferForm
    val transferForm = stateForm

    private val _isEditable = MutableLiveData<Boolean>()
    val isEditable: LiveData<Boolean> = _isEditable

    private val _uploadResult: MediatorLiveData<Event<Map<TransferApiOut-
bound, Result<Boolean>>>> = MediatorLiveData()
    val uploadResult: LiveData<Event<Map<TransferApiOutbound, Result<Bool-
ean>>>> = _uploadResult

    private var transferToUpdate: TransferEntity? = null
}
```

Εικόνα 4.6: Απόσπασμα του TransferViewModel

Το πρώτο annotation που συναντάται εδώ, το HiltViewModel, σε συνδυασμό με το δηλωμένο στον κατασκευαστή annotation Inject, ενημερώνουν τη βιβλιοθήκη Hilt ότι θα πρέπει να παραχθεί κώδικας που θα επιτρέπει το Dependency Injection του TransferViewModel σε άλλα αντικείμενα. Στον κατασκευαστή λαμβάνονται, μέσω της ίδιας μεθόδου, αντικείμενα που χρειάζεται η κλάση. Τα τρία πρώτα έχουν πρόσβαση σε πηγές δεδομένων για την άντληση, την αποθήκευση ή την αποστολή δεδομένων. Επιπλέον, μέσω του annotation ApplicationContext ενημερώνεται η βιβλιοθήκη ότι ζητείται πρόσβαση στο Context της εφαρμογής, χωρίς να χρειάζεται κάποια άλλη ενέργεια από τον προγραμματιστή. Το τελευταίο αντικείμενο διατηρεί την κατάσταση του UI στο ViewModel, ενώ μπορεί να παράσχει και τις παραμέτρους που δίνονται για την πλοήγηση σε μια οθόνη. Αυτό ακριβώς γίνεται στις πρώτες γραμμές του

σώματος της κλάσης, όπου ανακτώνται οι παράμετροι transferId και groupId. Προηγουμένως όμως, πρέπει να σημειωθεί ότι η κλάση TransferViewModel κληρονομεί την ContentViewModel που με τη σειρά αποτελεί υποκλάση της ViewModel.

Τα ViewModel αποτελούν αποθήκη κατάστασης και για το λόγο αυτό διατηρούν δομές δεδομένων. Στη συνέχεια του σώματος, δηλώνονται τα αντικείμενα _transferDebtors και transferDebtors. Αυτό αποτελεί κοινή πρακτική, καθώς το πρώτο αντικείμενο πρέπει είναι μεταβλητό ώστε η τιμή του να μπορεί να αλλάζει μόνο από την ίδια την κλάση, γι' αυτό και η ορατότητα της είναι ιδιωτική. Συγκεκριμένα, αυτό το αντικείμενο διατηρεί τους δικαιούχους της Μεταφοράς, οι οποίοι επιλέγονται από τον χρήστη και αποθηκεύονται στο ViewModel. Η μεταβλητή είναι τύπου LiveData, οπότε μόλις η τιμή της αλλάξει, τότε τα αντικείμενα του View που την παρακολουθούν, παραλαμβάνουν τα νέα δεδομένα και τα εμφανίζουν. Για το λόγο αυτό πρέπει να υπάρχει και ένα δημόσιο αντικείμενο. Αυτό το ρόλο έχει το αντικείμενο transferDebtors, ώστε να υπάρχει πρόσβαση σε αυτό, χωρίς όμως την ικανότητα μεταβολής της τιμής του. Παρόμοια λογική ακολουθείται και για τις υπόλοιπες μεταβλητές τύπου LiveData.

Επιπλέον, όπως φαίνεται και στον κώδικα, γίνεται override η μεταβλητή stateForm αναθέτοντάς της την τιμή της μεταβλητής _transferForm για χρήση από το ContentViewModel. Αυτή η μεταβλητή διατηρεί τις υπόλοιπες ιδιότητες μιας Μεταφοράς, όπως τον εντολέα ή την περιγραφή. Αυτή η κλάση αποτελεί υποκλάση της ContentFormState, καθώς Συναλλαγές και Μεταφορές μοιράζονται μεγάλο μέρος των ιδιοτήτων τους. Τέλος η μεταβλητή transferToUpdate προορίζεται για αποκλειστική εσωτερική χρήση και διατηρεί το αρχικό περιεχόμενο μιας Μεταφοράς που ανανεώνεται.

Οι οθόνες αποτελούνται από Views, τα οποία δηλώνονται σε αρχεία διάταξης (layout files) με χρήση της γλώσσας XML. Σε αυτά είναι δυνατό να προστεθεί αναφορά στα ViewModels, ώστε να παρέχουν τα δεδομένα απευθείας στο UI με χρήση Data Binding. Αυτή η μέθοδος, καθώς και η View Binding, διευκολύνουν τη συγγραφή κώδικα που αλληλοεπιδρά με το UI. Η δεύτερη μέθοδος είναι πιο απλή και για κάθε XML layout αρχείο δημιουργεί μια binding κλάση, η οποία περιέχει αναφορές σε κάθε View που διαθέτει αναγνωριστικό (id). Έπειτα οι αναφορές μπορούν να χρησιμοποιηθούν από άλλα αντικείμενα, ώστε να αποκτήσουν πρόσβαση στις ιδιότητες και τις μεθόδους των Views. Αυτή η εργασία εκτελείται και

από το Data Binding, όμως αυτό επιτρέπει ακόμα περισσότερες λειτουργίες. Αυτές είναι:

- Η συνεργασία με observable αντικείμενα, ώστε να ενημερώνει κάποιο View τη στιγμή που αλλάζει η τιμή του αντικειμένου που παρακολουθείται με νέα δεδομένα
- Η σύνδεση των Views με Στοιχεία Αρχιτεκτονικής (Architecture Components). Τέτοια στοιχεία αποτελούν τα LiveData και τα ViewModels
- Η χρήση Binding Adapters, μέσω των οποίων κάθε έκφραση διάταξης (layout expression), μπορεί να εκτελέσει τις απαραίτητες κλήσεις, ώστε να τεθούν οι αντίστοιχες ιδιότητες ή listeners
- Το αμφίδρομο Data Binding που πέρα από την λήψη των αλλαγών μιας ιδιότητας, είναι δυνατή και η παρακολούθηση για αλλαγές που προκαλούνται από τον χρήστη. Για παράδειγμα, σε ένα πεδίο κειμένου που παρακολουθεί μια μεταβλητή, είναι δυνατή τόσο η εμφάνιση της νέας τιμής της μεταβλητής στο πεδίο κειμένου, όσο και η αντικατάσταση της τιμής της με το νέο περιεχόμενο που εισάγεται από τον χρήστη

(Data Binding Library , n.d.) (View Binding, n.d.)

Για το παρών παράδειγμα, θα αναλυθεί το Data Binding για την οθόνη της Επισκόπησης Μεταφοράς, η οποία διαθέτει αναφορά στο TransferViewModel ή καλύτερα στην υπερκλάση της, την ContentViewModel. Παρακάτω παρουσιάζεται η κλάση TransferFormState και ένα απόσπασμα της ContentFormState στις οποίες έγινε αναφορά προηγουμένως.

```
class TransferFormState : ContentFormState {
    var creditor: UserIdAndName? = null

    constructor(): super()

    constructor(
        creditor: UserIdAndName?,
        totalValue: Long,
        location: Location?,
        version: Int,
        error: String? = null,
        description: String?,
        dateTime: LocalDateTime
    ): super(totalValue, location, version, error, description, dateTime) {
        this.creditor = creditor
    }
}
```

Εικόνα 4.7: Η κλάση TransferFormState

```
open class ContentFormState : BaseObservable {
    /* rest of the properties and constructors omitted */

    @get:Bindable
    var description : String? = null
        set(value) {
            if (description != value) {
                field = value

                notifyPropertyChanged(BR.description)
            }
        }
}
```

Εικόνα 4.8: Απόσπασμα της κλάσης ContentFormSate

Από το απόσπασμα της κλάσης ContentFormSate, αποκρύπτονται οι υπόλοιπες ιδιότητές και οι κατασκευαστές της, ώστε να φαίνεται καλύτερα πως γίνεται η διαχείριση μιας ιδιότητας που υποστηρίζει αμφίδρομο Data Binding. Στην οθόνη Επισκόπηση Μεταφοράς, το πεδίο της περιγραφής υποστηρίζει αυτή τη μέθοδο, ώστε όταν αλλάζει η τιμή του πεδίου στο UI από τον χρήστη, τότε ενημερώνεται και το περιεχόμενο της μεταβλητής description στη μνήμη, καθώς και το αντίστροφο. Όμως αυτό δημιουργεί ένα πρόβλημα, όταν ο χρήστης αλλάζει την τιμή, τότε δημιουργείται ειδοποίηση αλλαγής, μεταβάλλεται η τιμή της μεταβλητής στη μνήμη, η οποία με τη σειρά της δημιουργεί νέα ειδοποίηση αλλαγής που αλλάζει εκ νέου το περιεχόμενο του πεδίου στο UI. Έτσι δημιουργείται ένας ατέρμων βρόχος

ειδοποιήσεων αλλαγής τιμής. Για το λόγο αυτό οι ιδιότητες αμφίδρομου Data Binding διαθέτουν ένα έλεγχο σύγκρισης της παλαιάς και της νέας τιμής, ώστε να τον διακόψουν. Οι υπόλοιπες ιδιότητες της κλάσης, δηλώνονται ακριβώς όπως σε κάθε κοινή κλάση.

Στα αρχεία διάταξης, όταν μια ιδιότητα χαρακτηρίζεται ως αμφίδρομη, τότε της αναθέτεται τιμή της μορφής «@={όνομα_μεταβλητής}». Η μεταβλητή που παρακολουθείται προέρχεται από κάποιο ViewModel στην συγκεκριμένη περίπτωση το ContentViewModel. Σημειώνεται ότι σε αμφίδρομο Data Binding συνήθως επιλέγονται αντικείμενα τύπου LiveData ή Observable. Σε απλές υλοποιήσεις, όπως σε περιπτώσεις μονόδρομου Data Binding, μπορούν να χρησιμοποιηθούν, πέρα από τα παραπάνω, και αντικείμενα κοινών (απλών) κλάσεων. Στην παρακάτω περίπτωση επιλέχθηκε μια μίξη των δύο (LiveData παρακολουθεί Observable υποκλάση), ώστε να καθίσταται δυνατή η ειδοποίηση των LiveData όταν κάποιο πεδίο του σύνθετου αντικειμένου τους αλλάξει τιμή. Αυτό συμβαίνει, επειδή τα LiveData στέλνουν ειδοποίηση μόνο όταν αλλάξει η τιμή τους, δηλαδή όλο το αντικείμενο και όχι ένα μέρος του. Τα Observable αντικείμενα είναι ικανά να αντιλαμβάνονται τις αλλαγές (των ιδιοτήτων) και μέσω συγγραφής κατάλληλου κώδικα είναι δυνατή η προώθηση αυτού του γεγονότος στα LiveData.

```
<layout xmlns:tools="http://schemas.android.com/tools"
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto">
    <data>
        <variable name="viewModel"
            type="com.chirakis.kleon.kchap.ui.content.transaction.Transac-
tionViewModel" />
        <import type="com.example.kchap.ui.content.ContentFormState"/>
        <variable name="genericViewModel"
            type="com.chirakis.kleon.kchap.ui.content.ContentViewModel<Con-
tentFormState>" />
    </data>
</layout>
```

Εικόνα 4.9: Απόσπασμα των μεταβλητών του layout της πρώτης κάρτας της οθόνης Επισκόπησης (Μεταφοράς) που περιέχει την αναφορά στο ViewModel (ContentViewModel), αρχείο card_content_overview_main.xml

```
<EditText
    android:id="@+id/etDescription"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="12dp"
    android:background="@null"
    android:hint="@string/description"
    android:lines="2"
    android:maxLength="200"
    android:inputType="text"
    android:importantForAutofill="no"
    android:overScrollMode="always"
    android:scrollbarStyle="insideInset"
    android:scrollbars="vertical"
    android:text="@={genericViewModel.stateForm.description}"
    android:textColor="#bababa"
    app:layout_constraintTop_toBottomOf="@id/tvGroupName"
    tools:layout_editor_absoluteX="10dp" />
```

Εικόνα 4.10: Αμφίδρομο Data Binding στο πεδίο της περιγραφής της οθόνης Επισκόπησης (Μεταφοράς) χρησιμοποιώντας την αναφορά του ViewModel, αρχείο card_content_overview_main.xml

Όπως αναφέρθηκε η οθόνη της Επισκόπησης Μεταφοράς διαθέτει μια αναφορά στο TransferViewModel. Στην παρακάτω εικόνα, παρακολουθείται για αλλαγές η μεταβλητή που διατηρεί την κατάσταση της φόρμας. Τυχόν αλλαγή της τιμής της προκαλεί την ανανέωση των Views με χρήση των αναφορών τους από την binding κλάση.

```
transferViewModel.transferForm.observe(viewLifecycleOwner) {
    binding.lcContentOverviewMain.tvContentPrice.text =
        Utils.formatMoney(it.value, requireContext())

    binding.lcContentOverviewMain.lUserView.apply {
        cpb.text =
            Utils.Companion.Adapter.getNameInitialsIfNull(it.credi-
tor?.name)
        tvUserName.text = it.creditor?.name ?:
            requireContext().getString(R.string.deleted_member)
    }
}
```

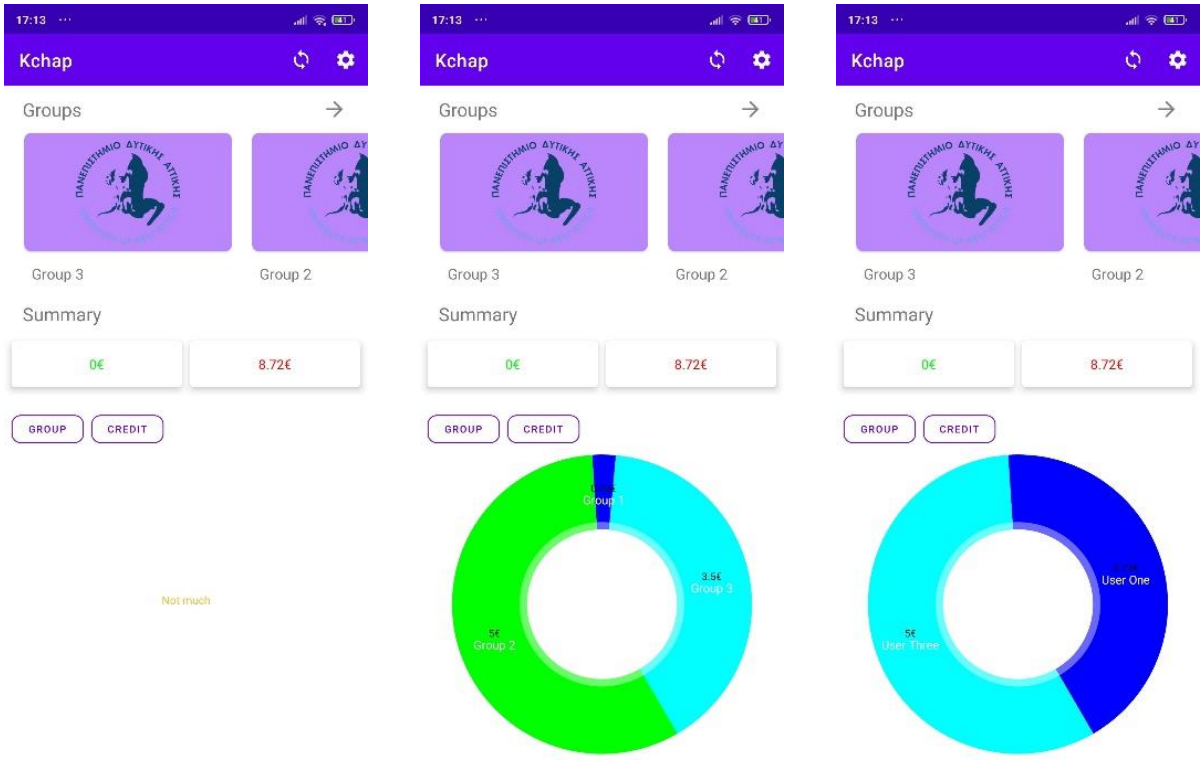
Εικόνα 4.11: Απόσπασμα από TransferOverviewFragment

4.2.5 Αρχική οθόνη

Η αρχική οθόνη εξυπηρετεί δύο κύριους σκοπούς, την γρήγορη πρόσβαση σε ενέργειες και καίρια σημεία της εφαρμογής, καθώς και την προβολή χρήσιμων πληροφοριών και στατιστικών. Χωρίζεται σε δύο μέρη, τις ομάδες και τη σύνοψη.

Στις ομάδες, προβάλλονται οι δέκα πρώτες, ταξινομημένες με βάση τις πιο πρόσφατες Αλλαγές. Επομένως, ομάδες που περιέχουν προσφάτως δημιουργημένες ή τροποποιημένες συναλλαγές ή μεταφορές, εμφανίζονται πρώτες. Από εκεί μπορεί να πραγματοποιηθεί απευθείας πλοήγηση στην επιλεγόμενη ομάδα. Σε περίπτωση που η ομάδα που αναζητείται δεν βρίσκεται εκεί, τότε επιλέγοντας την επικεφαλίδα πραγματοποιείται πλοήγηση στην οθόνη όλων των ομάδων.

Η σύνοψη χωρίζεται σε δύο υποκατηγορίες. Η πρώτη αφορά τη συνολική εικόνα των πιστώσεων και χρεώσεων του συνδεδεμένου χρήστη, τα οποία είναι χρωματισμένα με πράσινο και κόκκινο αντίστοιχα, ώστε να γίνεται αντιληπτός ο ρόλος τους. Η άλλη υποκατηγορία περιλαμβάνει ένα διάγραμμα τύπου πίτας για την γρήγορη πληροφόρηση σχετικά με τις κύριες πιστώσεις και χρεώσεις, είτε από ομάδες, είτε από άτομα. Αυτές οι παράμετροι μεταβάλλονται με τη χρήση δύο φίλτρων στο πάνω αριστερό μέρος της πίτας. Οι πληροφορίες που προβάλλονται από τα κομμάτια της πίτας είναι το όνομα της ομάδας ή του μέλους, καθώς και το ποσό. Όπως είναι φυσικό το πλήθος των δεδομένων μπορεί να καταστεί αρκετά πλούσιο, δυσκολεύοντας την αναπαράσταση τους στην πίτα. Εξού και η αναφορά για κύριες πιστώσεις ή χρεώσεις, ώστε οι σημαντικές αυτές πληροφορίες να είναι ευανάγνωστες μέσω της πίτας. Συγκεκριμένα, για κάθε κατηγορία γίνεται άντληση πληροφοριών των δέκα πρώτων εγγραφών, οι οποίες έχουν ταξινομηθεί με βάση το ποσό. Σε περίπτωση απουσίας δεδομένων για την παραγωγή στατιστικών, εμφανίζεται ένα μήνυμα ενημέρωσης αντί της πίτας. Επιπλέον από την αρχική είναι δυνατή η πλοήγηση στην οθόνη των ρυθμίσεων της εφαρμογής, καθώς και η εκκίνηση του συγχρονισμού κατά απαίτηση.



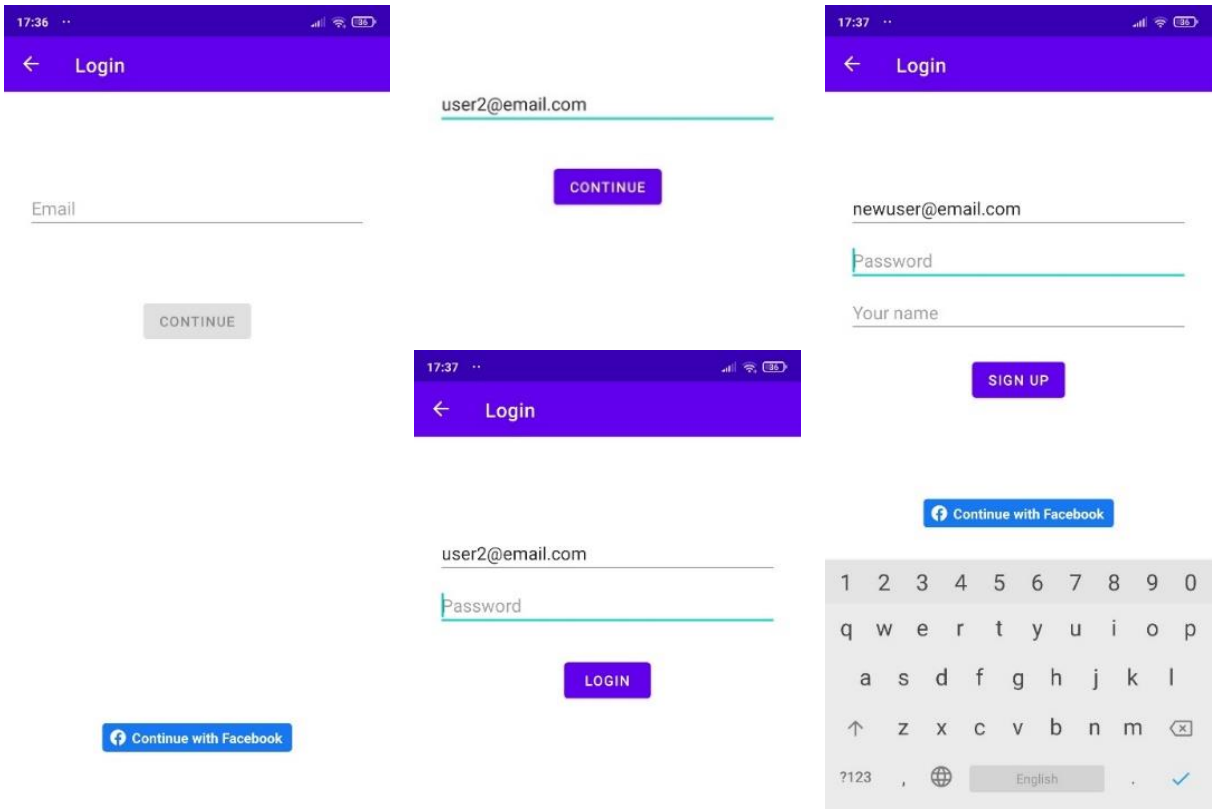
Εικόνα 4.12: Παράδειγμα στατιστικών πίτας: Απουσία πίστωσης (Αριστερά), Χρέος ανά ομάδα (Μέση), Χρέος ανά χρήστη (Δεξιά)

Στην παραπάνω εικόνα δίνεται ένα παράδειγμα χρήστη που δεν διαθέτει πίστωση, οπότε και δεν δημιουργούνται στατιστικά (στην πίτα) για αυτό το φίλτρο. Οι δύο επόμενες εικόνες αφορούν τις χρεώσεις ανά ομάδα και ανά χρήστη. Γίνεται φανερό ότι ο χρήστης κατέχει χρεώσεις σε τρεις ομάδες και δύο άτομα. Λόγω του μικρού μεγέθους των δεδομένων, το άθροισμα των χρεώσεων της πίτας, ανεξαρτήτως αν στο φίλτρο έχει επιλεχθεί ομάδα ή χρήστης, συμφωνεί με τις συνολικές υποχρεώσεις που παρουσιάζονται στη σύνοψη.

4.2.6 Αυθεντικοποίηση και στοιχεία χρήστη

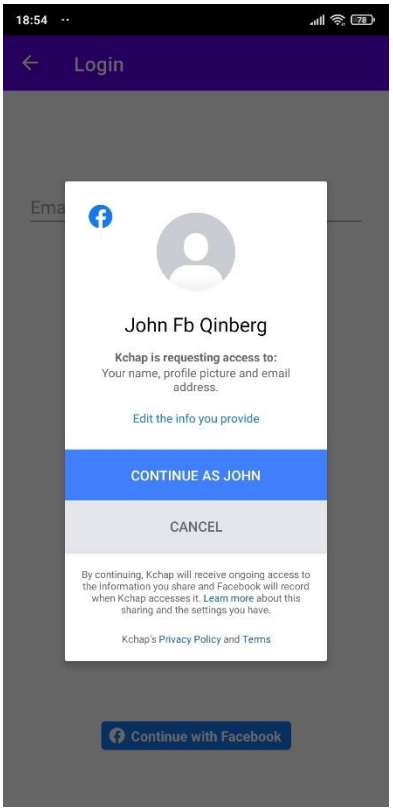
Με την πρώτη εκκίνηση της εφαρμογής ζητείται η σύνδεση του χρήστη στο σύστημα με αυτόματη πλοήγηση στην οθόνη σύνδεσης. Πρώτα ο χρήστης πρέπει να συμπληρώσει την διεύθυνση ηλεκτρονικού ταχυδρομείου (email) του. Το επόμενο βήμα είναι ο έλεγχος διαθεσιμότητας για αυτό το email. Αν δεν είναι διαθέσιμο, τότε υπάρχει λογαριασμός που αντιστοιχεί σε αυτό και εμφανίζεται το πεδίο του κωδικού πρόσβασης για την ολοκλήρωση της σύνδεσης. Στην αντίθετη περίπτωση, εμφανίζονται τα πεδία του κωδικού πρόσβασης και του ονόματος χρήστη, ώστε να συμπληρωθούν και να ολοκληρωθεί η διαδικασία της εγγραφής. Με την επιτυχή

εγγραφή, πραγματοποιείται σύνδεση και προώθηση στην κεντρική οθόνη. Σημειώνεται ότι κάθε μεταβολή της τιμής του πεδίου email, προκαλεί την επαναφορά της φόρμας σε κατάσταση ελέγχου διαθεσιμότητας της διεύθυνσης. Παρόλα αυτά τα συμπληρωμένα πεδία δεν χάνουν το περιεχόμενό τους, ώστε να μην ζητείται από τον χρήστη να επαναλάβει την εισαγωγή των στοιχείων του.



Εικόνα 4.13: Η οθόνη σύνδεσης/εγγραφής (Αριστερά), ενεργοποίηση κουμπιού με ανίχνευση έγκυρης μορφής email (Μέση-Άνω), μορφοποίηση οθόνης για σύνδεση (Μέση-Κάτω) και μορφοποίηση οθόνης για εγγραφή (Δεξιά)

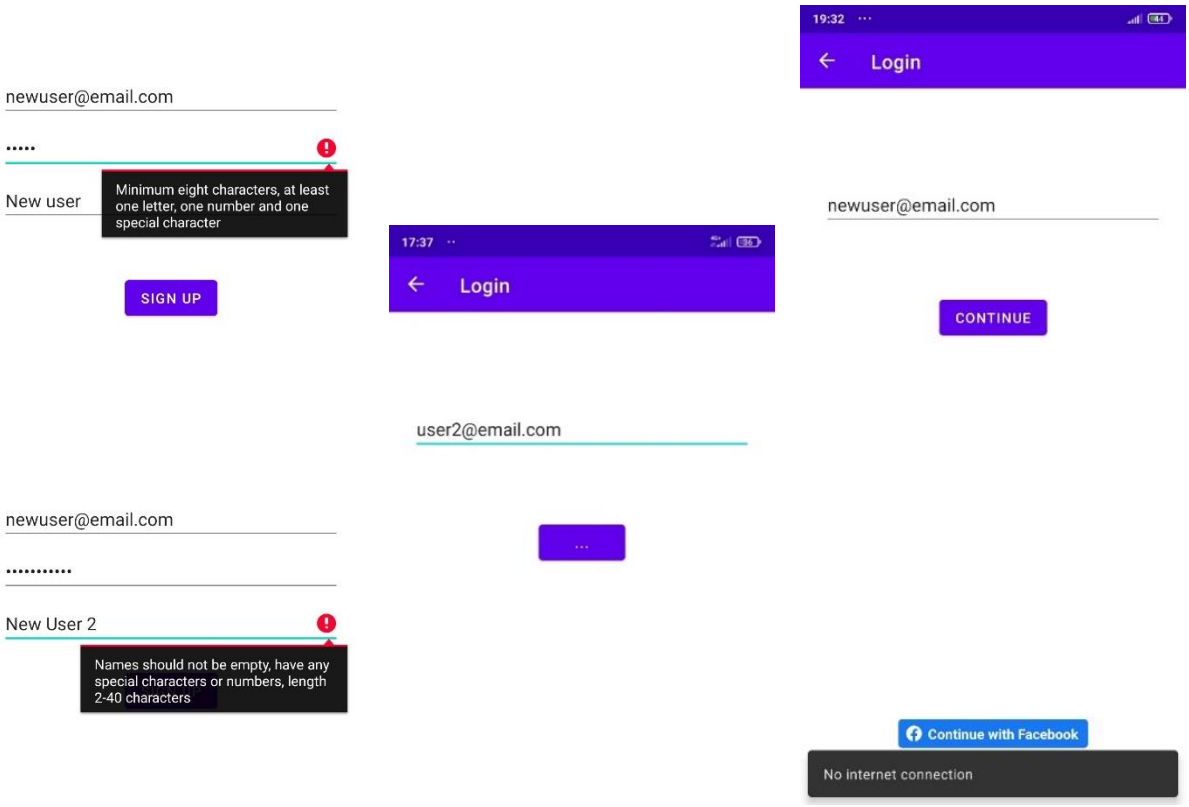
Όπως προαναφέρθηκε, πέρα από τον τοπικό λογαριασμό, κάποιος χρήστης μπορεί να συνδεθεί στην εφαρμογή με χρήση λογαριασμού Facebook πατώντας το ανάλογο κουμπί. Στη συνέχεια, ο χρήστης συνδέεται και αποδέχεται την πρόσβαση της εφαρμογής στις βασικές πληροφορίες που διατηρεί στο Facebook λογαριασμό του. Το χαρακτηριστικό βασίζεται στις δυνατότητες που παρέχει το Facebook SDK.



Εικόνα 4.15: Σύνδεση με λογαριασμό Facebook

Τόσο στην τοπική σύνδεση, όσο και στην εγγραφή γίνεται έλεγχος ορθότητας των πεδίων με εφαρμογή κανόνων ή περιορισμών. Απλοί και σταθεροί κανόνες, όπως επιβεβαίωση ότι τα πεδία είναι συμπληρωμένα ή η εγκυρότητα της μορφής της διεύθυνσης ηλεκτρονικού ταχυδρομείου, γίνονται τοπικά στη συσκευή. Πιο περίπλοκοι κανόνες, όπως η μορφή του ονόματος του χρήστη (μη κενή στοιχειοσειρά άνευ αριθμών) ή οι πολιτικές κωδικού πρόσβασης, καθώς και ο επανέλεγχος των απλών, πραγματοποιούνται από τον εξυπηρετητή. Επιλέχθηκε αυτή η προσέγγιση, καθώς διαφορετικά, τυχόν ανανέωση των κανόνων θα απαιτούσε και την ενημέρωση της εφαρμογής του πελάτη. Για τον κωδικό πρόσβασης εφαρμόζονται πολιτικές που έχουν ως σκοπό την ενίσχυση της ασφάλειας με τη δημιουργία ισχυρών κωδικών. Οι περιορισμοί που εφαρμόζονται είναι το ελάχιστο μήκος οκτώ χαρακτήρων με τουλάχιστον ένα γράμμα, έναν αριθμό και έναν ειδικό χαρακτήρα. Σε περίπτωση που κάποια πεδία παραβιάζουν τους κανόνες, τότε το όνομα του πεδίου, καθώς και τα σφάλματα που το συνοδεύουν, περιλαμβάνονται στη λίστα παραβιάσεων με την οποία απαντά ο εξυπηρετητής. Στη

συνέχεια, η εφαρμογή του πελάτη αντιστοιχεί τα σφάλματα στα κατάλληλα πεδία και τα εμφανίζει.

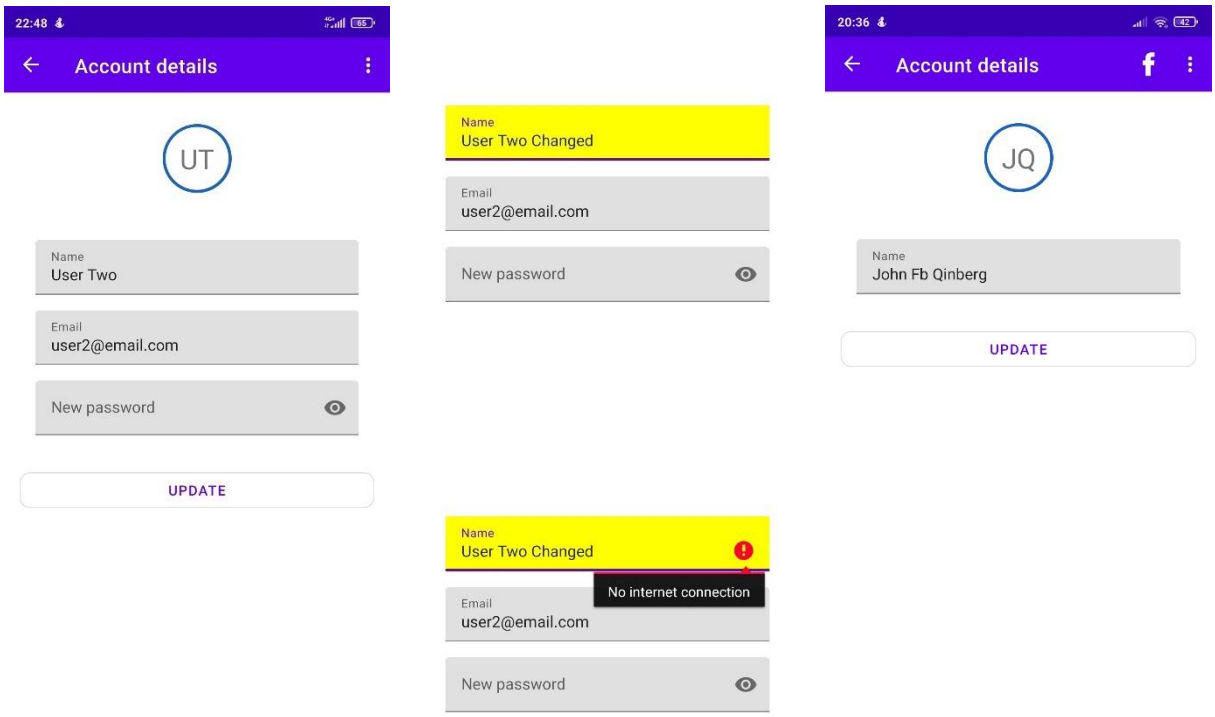


Εικόνα 4.14: Επισήμανση πεδίων με αποτυχημένη επικύρωση (Αριστερά), αλλαγή εμφάνισης κουμπιού κατά την εκτέλεση ενέργειας (Μέση) και ειδοποίηση για αποτυχία εκτέλεσης ενέργειας και επαναφορά φόρμας για επανέλεγχο email (Δεξιά)

Αφού ο χρήστης επιτύχει σύνδεση, τότε ο εξυπηρετητής επιστρέφει στην απάντηση ένα cookie. Αυτό αποθηκεύεται στα SharedPreferences του Android, αλλά κρυπτογραφημένο (AES256) μέσω της βιβλιοθήκης security-crypto, ώστε να μειωθούν οι πιθανότητες έκθεσής του. Το cookie περιλαμβάνεται σε κάθε απάντηση του διακομιστή και σε κάθε αίτημα του πελάτη. Για αυτό το λόγο, το αναγνωριστικό του cookie αποθηκεύεται στη μνήμη μόλις παραληφθεί ή αλλάξει τιμή. Διαφορετικά δεν θα ήταν βέλτιστη η διαρκής ανάκτηση και αποκρυπτογράφησή του. Σε περίπτωση που κάποιο αίτημα απαιτεί αυθεντικοποίηση, αλλά αποτύχει με HTTP κωδικό 401, και δεν αποτελεί αποτυχημένη προσπάθεια σύνδεσης, τότε εμφανίζεται η οθόνη σύνδεσης, καθώς ο χρήστης δεν είναι αυθεντικοποιημένος.

Η ανανέωση των πληροφοριών του χρήστη πραγματοποιείται από την οθόνη λεπτομερειών λογαριασμού και καθίσταται δυνατή η αλλαγή όλων των πληροφοριών

που συμπληρώθηκαν κατά την εγγραφή. Κάθε πεδίο που μεταβάλλεται η τιμή του, επισημαίνεται χρωματίζοντας το παρασκήνιό του. Για κάθε αλλαγμένο πεδίο εκτελείται κλήση της ανάλογης υπηρεσίας για ανανέωση του περιεχομένου του. Σημειώνεται ότι ανάλογα τον πάροχο σύνδεσης, εμφανίζονται και τα αντίστοιχα πεδία, ενώ στην μπάρα ενεργειών θα εμφανιστεί και το λογότυπο του τρίτου παρόχου. Τα πεδία που αφορούν ένα χρήστη που συνδέθηκε μέσω Facebook, είναι μόνο το πεδίο του ονόματος.

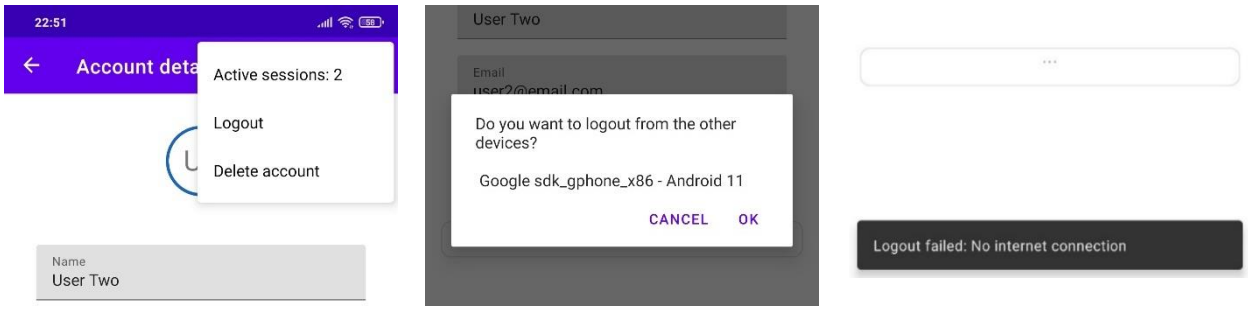


Εικόνα 4.16: Η οθόνη πληροφοριών λογαριασμού τοπικού λογαριασμού (Αριστερά), επισήμανση αλλαγμένης τιμής πεδίου (Μέση-Άνω) και ειδοποίηση για αποτυχία εφαρμογής αλλαγμένης τιμής (Μέση-Κάτω), οθόνη πληροφοριών λογαριασμού με πάροχο σύνδεσης το Facebook (Δεξιά)

Στο μενού επιλογών μπορούν να βρεθούν ο αριθμός των ενεργών συνεδριών, η έξοδος και η διαγραφή του λογαριασμού. Οι ενεργές συνεδρίες λαμβάνονται κάθε φορά που εμφανίζεται το μενού και δείχνουν το πλήθος των συσκευών που έχει πραγματοποιήσει είσοδο ο χρήστης και έχουν πρόσβαση στα δεδομένα του. Πατώντας πάνω στις συνεδρίες ο χρήστης ερωτάτε για πραγματοποίηση εξόδου από όλες τις συσκευές, εκτός της παρούσας.

Η αποσύνδεση χρήστη διαγράφει όλα τα δεδομένα από τη συσκευή (πίνακες της βάσης δεδομένων, cookie, τυχόν access token τρίτου παρόχου) και εμφανίζει την οθόνη σύνδεσης. Η διαγραφή έχει παρόμοια συμπεριφορά με την αποσύνδεση, αλλά διαγράφει και τα δεδομένα του χρήστη που διατηρούνται στον διακομιστή.

Κάθε ενέργεια, καθώς εκτελείται αλλάζει προσωρινά τη μορφή του κεντρικού κουμπιού ως ένδειξη αναμονής. Σε περίπτωση που υπάρξει σφάλμα, η διαδικασία ακυρώνεται και εμφανίζεται ανάλογο μήνυμα.



Εικόνα 4.17: Μενού επιλογών οθόνης λεπτομερειών λογαριασμού (Αριστερά), Ερώτηση στον χρήστη για αποσύνδεση από άλλες συσκευές (Μέση), αλλαγή εμφάνισης του κεντρικού κουμπιού κατά τη διάρκεια αιτήματος (Δεξιά-Άνω) και ειδοποίηση για αποτυχία εκτέλεσης αιτήματος (Δεξιά-Κάτω)

4.2.7 Συγχρονισμός

Ο συγχρονισμός πραγματοποιείται αυτόματα με την εκκίνηση της εφαρμογής, αλλά μπορεί να πραγματοποιηθεί και κατόπιν αιτήματος του χρήστη, πατώντας το κουμπί συγχρονισμού που βρίσκεται στην αρχική οθόνη. Ο συγχρονισμός συντονίζεται από την κλάση SyncManager και εκτελείται σε δύο φάσεις. Η πρώτη περιλαμβάνει την αποστολή των αυτοματοποιημένων εντολών που δημιουργήθηκαν από το σύστημα, καθώς τα δευτερεύοντα υπόλοιπα (ep) πλέον αντιστοιχούν σε πραγματικά χρήματα. Η δεύτερη φάση αφορά τη λήψη και ενημέρωση όλων των υπόλοιπων δεδομένων, όπως ομάδες, μέλη, υπόλοιπα και Περιεχόμενο.

Η εύρεση πραγματικών χρηματικών ποσών από τα δευτερεύοντα υπόλοιπα έχει ως αποτέλεσμα την μεταφορά του ποσού στο πραγματικό υπόλοιπο των χρηστών, ενημερώνοντας την τοπική βάση δεδομένων. Τυχόν αποτυχία αποστολής αυτής της ενέργειας στον εξυπηρετητή, έχει ως αποτέλεσμα την αναστροφή της ανωτέρω διαδικασίας. Σε αυτή την περίπτωση, θα διαγραφούν οι εντολές μεταφοράς, ενώ

παράλληλα θα αφαιρεθούν τα ποσά από τα πρωτεύοντα υπόλοιπα των μελών και θα επιστρέψουν στα δευτερεύοντα υπόλοιπα. Με αυτή την αλλαγή των υπολοίπων, εκκινεί και η διαδικασία υπολογισμού των ροών χρέους. Αυτή περιλαμβάνει τόσο την αναζήτηση για εύρεση πραγματικών χρηματικών ποσών στα δευτερεύοντα υπόλοιπα, όσο και τον υπολογισμό των ροών χρέους μεταξύ των μελών βάσει του πρωτεύοντος υπολοίπου τους. Παρόλα αυτά και μόνο για αυτή την περίπτωση, παραλείπεται το πρώτο βήμα, καθώς η εκτέλεση του πιθανότατα θα εύρισκε την ίδια αυτοματοποιημένη εντολή που ο εξυπηρετητής μόλις απέρριψε.

Ο συγχρονισμός των υπόλοιπων δεδομένων ξεκινά με την λήψη των βασικών πληροφοριών όλων των ομάδων και όλων των διακριτών μελών που συμμετέχουν σε αυτές. Με αυτό τον τρόπο γίνεται εμφανές ποιες ομάδες είναι καινούργιες και απαιτούν συγχρονισμό για πρώτη φορά και ποιες απαιτούν ανανέωση των δεδομένων τους. Επίσης γίνεται φανερό ποιες ομάδες δεν υπάρχουν πια, ώστε να διαγραφούν από τη βάση δεδομένων. Όταν μια ομάδα λαμβάνεται για πρώτη φορά, τότε αντί συγχρονισμού, εκτελείται κλήση σε κατάλληλη υπηρεσία για λήψη των αναλυτικών δεδομένων της, περιλαμβάνοντας πέρα από τις βασικές της ιδιότητες και λεπτομερείς πληροφορίες για τα μέλη της, όπως τα υπόλοιπά τους. Ακολούθως δημιουργούνται οι αντίστοιχες εγγραφές στη βάση δεδομένων και ολοκληρώνεται η διαδικασία για αυτή την ομάδα.

Στην περίπτωση που μια ομάδα υπάρχει ήδη, τότε εκτελείται συγχρονισμός των δεδομένων της, λαμβάνοντας τις νέες Αλλαγές και το Περιεχόμενο που τις συνοδεύει. Προηγουμένως λήφθηκαν οι βασικές της πληροφορίες και δεδομένου ότι τα πεδία έκδοσης του αποθηκευμένου και του ληφθέντα πόρου διαφέρουν, τότε πραγματοποιείται ενημέρωση των δεδομένων. Η διαφορά υπονοεί την ύπαρξη ενημερωμένων δεδομένων που με τη σειρά της δικαιολογεί και την ανάγκη εκτέλεσης αντίστοιχου αιτήματος (αποθήκευσης) στη βάση δεδομένων. Στη συνέχεια εξετάζεται αν υπάρχουν νέες Αλλαγές προς εφαρμογή, συγκρίνοντας το αναγνωριστικό της τελευταίας συγχρονισμένης Αλλαγής με το πεδίο του αναγνωριστικού της τελευταίας Αλλαγής που λήφθηκε από τις πληροφορίες της ομάδας. Αν το πεδίο διαθέτει μεγαλύτερη τιμή, τότε σημαίνει ότι υπάρχουν νέες Αλλαγές και εκτελείται νέο αίτημα στον εξυπηρετητή για την παραλαβή τους. Όπως αναφέρθηκε, η παραλαβή γίνεται μέσω streaming, όποτε και έχει κατασκευαστεί αρμόδια μέθοδος που διαβάζει σειριακά την απάντηση, μορφής JSON, και συγκεντρώνει όλες τις πληροφορίες, δηλαδή τις Αλλαγές και το Περιεχόμενο. Οι νέες πληροφορίες αποθηκεύονται στη

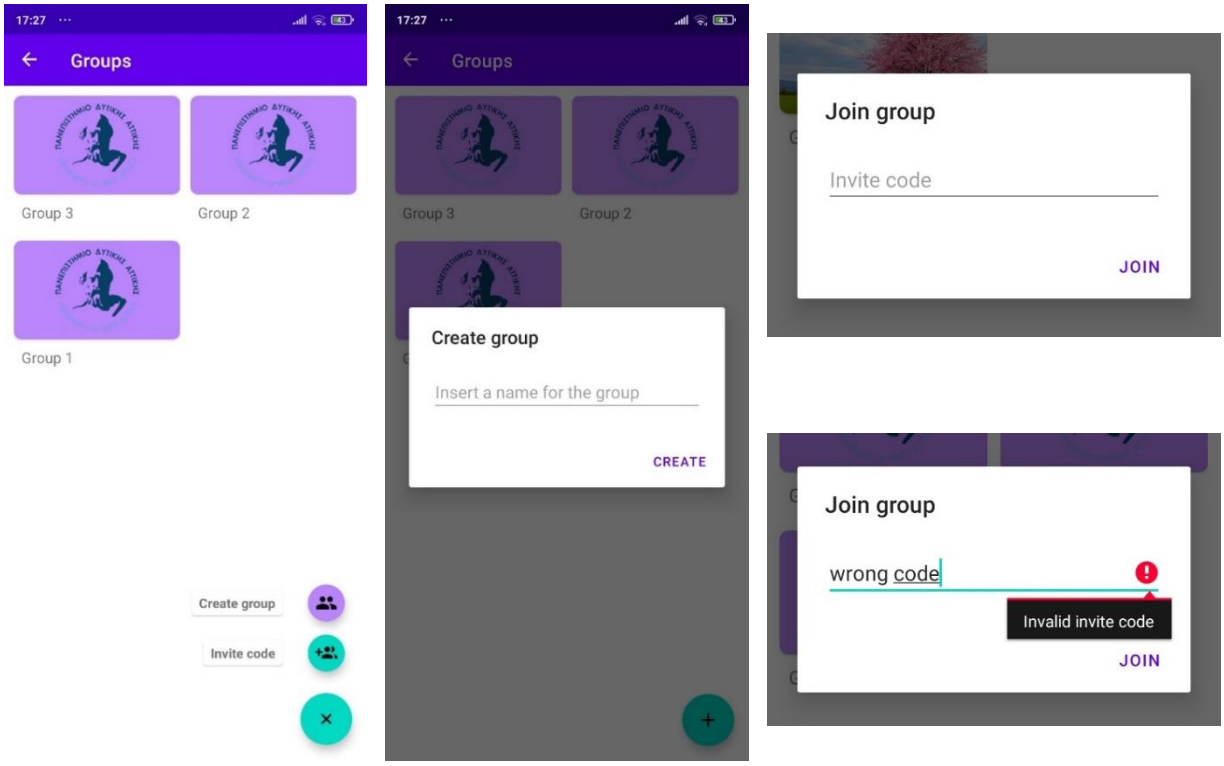
βάση και ενημερώνονται τα αντίστοιχα πεδία. Τυχόν Αλλαγές που δημιουργήθηκαν από το κινητό θα βρίσκονται στην τοπική μνήμη του, ώστε να εξαιρεθούν από τη λήψη. Διαφορετικά, θα εφαρμόζονταν ξανά και θα κατέστρεφαν τα δεδομένα. Με την ολοκλήρωση της διαδικασίας, αυτές θα αφαιρεθούν, καθώς ο επόμενος συγχρονισμός δεν θα τις περιλαμβάνει. Με την αποθήκευση των δεδομένων, ελέγχεται αν υπήρξαν αλλαγές στα υπόλοιπα των μελών. Αν αυτό ισχύει, τότε εκτελείται υπολογισμός των νέων ροών χρέους. Αν η διαδικασία αποτύχει, τότε δεν εφαρμόζεται καμία Αλλαγή για την συγκεκριμένη ομάδα.

Στο τέλος αποθηκεύονται οι πληροφορίες όλων των μελών που λήφθηκαν από το αίτημα που εκτελέστηκε με την εκκίνηση του συγχρονισμού. Η διαδικασία του συγχρονισμού είναι απαιτητική, καθώς εκτελεί πολλαπλά αιτήματα. Για το λόγο αυτό, εκτελείται ασύγχρονα και για πολλές ομάδες ταυτόχρονα. Τέλος, σημειώνεται ότι η κλάση διαθέτει Listeners που παρακολουθούν την εξέλιξη του συγχρονισμού, ώστε να ενημερώνουν το περιβάλλον χρήστη, όπως για παράδειγμα είναι η εμφάνιση μηνύματος αποτυχίας εκτέλεσης.

4.2.8 Ομάδες

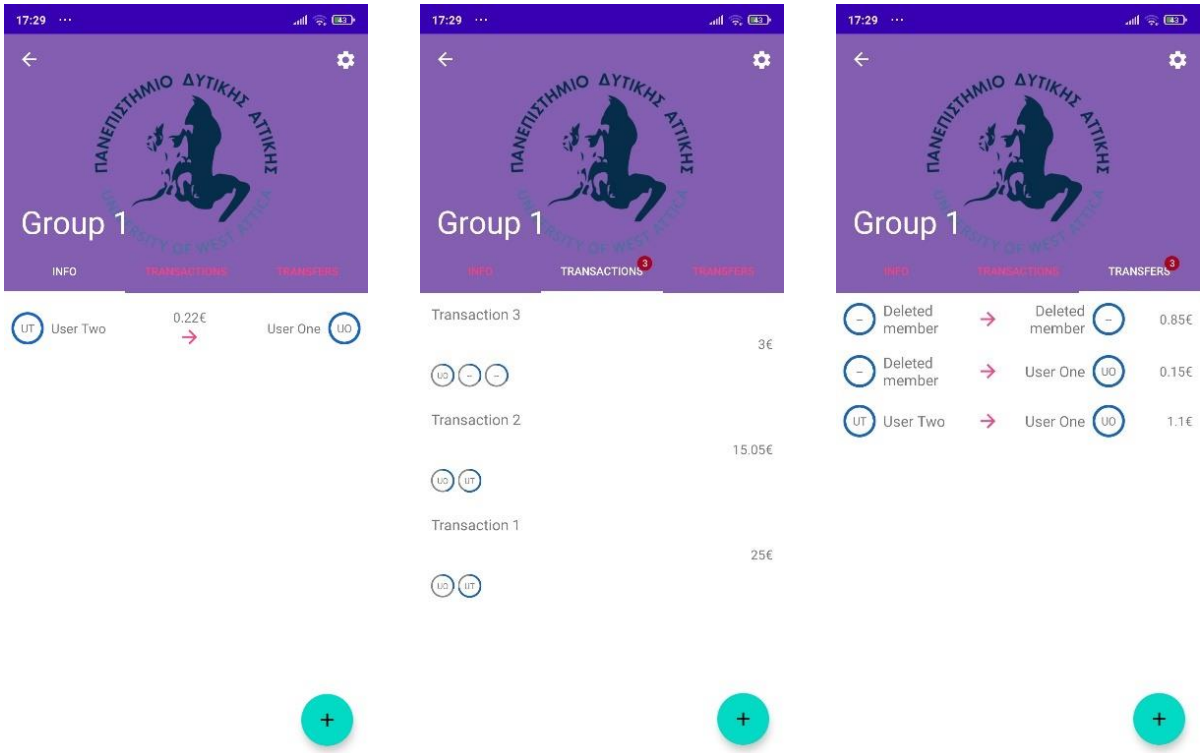
Η πλοήγηση για προβολή όλων των ομάδων πραγματοποιείται από την αρχική. Παρομοίως, οι ομάδες ακολουθούν την ίδια ταξινόμηση με αυτή της αρχικής σελίδας. Από την ίδια οθόνη είναι δυνατή η δημιουργία νέας ομάδας ή η συμμετοχή σε αυτή πατώντας στο κουμπί της προσθήκης. Η συμμετοχή σε κάποια ομάδα γίνεται με χρήση του κωδικού διαμοιρασμού της από τη συσκευή του ενδιαφερόμενου μέλους. Η εικόνα της ομάδας εξυπηρετεί σκοπούς επίδειξης και για το λόγο αυτό είναι κοινή σε όλες τις ομάδες.

Η δημιουργία μιας ομάδας απλά απαιτεί την εισαγωγή του ονόματός της και ο χρήστης γίνεται αυτόματα το πρώτο μέλος της. Η συμμετοχή σε ομάδα απαιτεί την εισαγωγή του κωδικού διαμοιρασμού. Σε περίπτωση που ο κωδικός δεν υπάρχει, τότε εμφανίζεται κατάλληλο μήνυμα σφάλματος.



Εικόνα 4.18: Λίστα όλων των ομάδων με εμφάνιση του μενού επιλογών (Αριστερά), παράθυρο δημιουργίας νέας ομάδας (Μέση), παράθυρο συμμετοχής σε υπάρχουσα ομάδα (Πάνω δεξιά) και εισαγωγή ανύπαρκτου κωδικού διαμοιρασμού (Κάτω δεξιά)

Επιλέγοντας κάποια ομάδα γίνεται μεταφορά του χρήστη στην οθόνη της ομάδας, η οποία αποτελείται από τρεις καρτέλες, τις Info, Transactions και Transfers. Η περιήγηση στις διάφορες καρτέλες πραγματοποιείται είτε με το πάτημα της αντίστοιχης καρτέλας είτε με χειρονομίες συρσίματος (σειριακή περιήγηση).



Εικόνα 4.19: Παραδείγματα εγγραφών Ροής χρέους (Αριστερά), Συναλλαγών (Μέση), Μεταφορών (Δεξιά), στο Περιεχόμενο συμπεριλαμβάνονται περιπτώσεις που τα μέλη έχουν αποχωρήσει

Η πρώτη καρτέλα περιέχει τις ροές χρέους. Μια ροή χρέους απεικονίζεται ως μια οριζόντια εγγραφή. Το αριστερό μέρος καταλαμβάνει ο οφειλέτης, στη μέση διακρίνεται το οφειλόμενο ποσό και στο δεξιό μέρος βρίσκεται ο πιστωτής. Ο κύκλος προόδου του οφειλέτη δείχνει το ποσοστό της συγκεκριμένης οφειλής ως προς τα συνολικά του χρέη, ενώ παρόμοια συμπεριφορά αντικατοπτρίζει και ο κύκλος προόδου του πιστωτή, όμως ως προς τη συνολική πίστωση που έχει διαθέσει.

Η καρτέλα των συναλλαγών περιέχει τις συναλλαγές της ομάδας. Κάθε συναλλαγή αποτελεί μια εγγραφή που περιέχει το όνομα, το ποσό και τα μέλη που αφορά. Η τιμή πλήρωσης του κύκλου προόδου των μελών εκφράζει το ποσοστό που καλούνται να πληρώσουν επί του συνολικού ποσού της συναλλαγής.

Η τρίτη καρτέλα περιλαμβάνει τις μεταφορές. Παρόμοια με παραπάνω αποτελεί εγγραφή που εμπεριέχει τον εντολέα και τον δικαιούχο, καθώς και το ποσό που μεταφέρθηκε. Σε αυτή την περίπτωση το ποσοστό του κύκλου προόδου δεν έχει κάποια ενημερωτική ιδιότητα, οπότε και είναι πλήρως συμπληρωμένο σε όλες τις εγγραφές.

Όσον αφορά τις καρτέλες που σχετίζονται με το Περιεχόμενο υπάρχουν κάποιες συμπεριφορικές ομοιότητες. Η είσοδος στην εκάστοτε καρτέλα ανακτά και εμφανίζει το τοπικά αποθηκευμένο Περιεχόμενο. Η εξάντληση των εγγραφών της λίστας, είτε λόγω αρχικής έλλειψής τους, είτε λόγω κύλισης προς τα κάτω εκκινεί τη διαδικασία λήψης παλαιότερου Περιεχομένου. Η διαδικασία ανάκτησης, τοπικής ή απομακρυσμένης, είναι πιθανό να εκκινήσει και χωρίς τη προβολή κάποιας σελίδας Περιεχομένου. Αυτό συμβαίνει λόγω της ιδιότητας setOffscreenPageLimit του αντικειμένου προβολής ViewPager. Αυτή η ιδιότητα ορίζει ότι μπορεί να πραγματοποιηθεί προφόρτωση των καρτελών που βρίσκονται δεξιά και αριστερά της καρτέλας που προβάλλεται. Στην εφαρμογή χρησιμοποιείται η προεπιλεγμένη τιμή ένα που σημαίνει ότι αυτό μπορεί να συμβεί μόνο σε μία καρτέλα αριστερά και σε άλλη μία δεξιά. Όταν κάποια καρτέλα Περιεχομένου είναι επιλεγμένη τότε πάνω σε αυτή εμφανίζεται ένα badge που υποδεικνύει το συνολικό αριθμό εγγραφών. Όταν επιλέγεται κάποια εγγραφή Περιεχομένου, τότε είναι δυνατή η προβολή και η ενημέρωσή του.

Η λήψη του παλαιότερου Περιεχομένου, καθώς αυτό δεν λαμβάνεται με τη διαδικασία του συγχρονισμού, εκτελείται με κλήσεις κατάλληλων υπηρεσιών συνοδεία παραμέτρων (minTransactionChangeId, minTransferChangeId), οι τιμές των οποίων αντλούνται από την τοπική βάση δεδομένων και αντιστοιχούν σε κάθε ομάδα. Τα δεδομένα λαμβάνονται κατά τμήματα ή σελίδες, ενώ αποθηκεύονται και στην τοπική βάση, ώστε να είναι διαθέσιμα για ανάγνωση χωρίς την ανάγκη σύνδεσης στο διαδίκτυο. Εν συνεχεία, αυτά ενσωματώνονται με τα υπόλοιπα αποτελέσματα της λίστας, καθώς το UI ανανεώνεται εξαιτίας της μεταβολής των δεδομένων της βάσης. Έτσι στη λίστα εμφανίζονται πρώτα τα τοπικά δεδομένα και σε περίπτωση που χρειαστεί, τότε εκτελείται λήψη επιπλέον δεδομένων από τον εξυπηρετητή. Σε περίπτωση που κατά τη διάρκεια λήψης προκύψει κάποιο σφάλμα δικτύου, τότε εμφανίζεται κατάλληλο κουμπί, ώστε να δοθεί η επιλογή στον χρήστη να ξαναπροσπαθήσει.

Εικόνα 4.20: Λήψη μεταφορών για πρώτη φορά (Αριστερά) και ενημέρωση για αδυναμία λήψης δεδομένων με κουμπί επαναπροσπάθειας (Δεξιά)

Η σελιδοποίηση υλοποιείται μέσω της κλάσης RemoteMediator που παρέχεται από την βιβλιοθήκη Paging 3. Σημειώνεται ότι η σελιδοποίηση εφαρμόζεται τόσο στα τοπικά όσο και στα απομακρυσμένα δεδομένα. Η χρήση στα τοπικά δεδομένα γίνεται για μείωση του φόρτου της συσκευής, καθώς κατά κύριο λόγο δεν απαιτείται η εμφάνιση όλων των εγγραφών. Οι λόγοι σελιδοποίησης της απάντησης του εξυπηρετητή έχει ειπωθεί σε προηγούμενο κεφάλαιο. Η προεπιλεγμένη τιμή του πλήθους των αποτελεσμάτων που εμφανίζονται στη λίστα είναι πέντε.

Στη βιβλιοθήκη διατίθενται τρεις μέθοδοι φόρτωσης. Η ανανέωση (refresh) που δεν χρησιμοποιείται, καθώς τα δεδομένα έχουν ήδη ληφθεί και αποθηκευτεί τοπικά, ενώ τυχόν ενημερώσεις αποκτώνται από τον συγχρονισμό. Η μέθοδος prepend που αφορά προηγούμενα τμήματα δεδομένων, τα οποία επίσης δεν χρειάζονται να ληφθούν, καθώς η σειριακή απόκτηση του (νεότερου) Περιεχομένου σημαίνει ότι αυτό έχει ήδη συμβεί. Για αυτές τις μεθόδους, η διαδικασία τερματίζεται αναφέροντας εξάντληση όλων των διαθέσιμων σελίδων. Τέλος, υπάρχει η μέθοδος append, στην οποία και βασίζεται η όλη υλοποίηση, καθώς σκοπός είναι η λήψη και η προσθήκη των επόμενων τμημάτων των (παλαιότερων) δεδομένων. Το service, πέρα από το παλαιότερο Περιεχόμενο, επιστρέφει το αναγνωριστικό που θα πρέπει να αποσταλεί την επόμενη φορά, ώστε να ληφθεί το επόμενο κομμάτι δεδομένων. Η τιμή του αναγνωριστικού αποθηκεύεται στην τοπική βάση δεδομένων. Σε περίπτωση που η τιμή ισούται με το μηδέν, τότε γίνεται αντιληπτό ότι έχουν ληφθεί όλοι οι πόροι που αφορούν το συγκεκριμένο Περιεχόμενο και δεν γίνεται πότε ξανά προσπάθεια λήψης.

Καθ’ όλη τη διάρκεια περιήγησης στη σελίδα της ομάδας είναι δυνατή η προσθήκη Περιεχομένου με χρήση του κουμπιού της πρόσθεσης. Το πάτημα σε αυτό εκκινεί τη

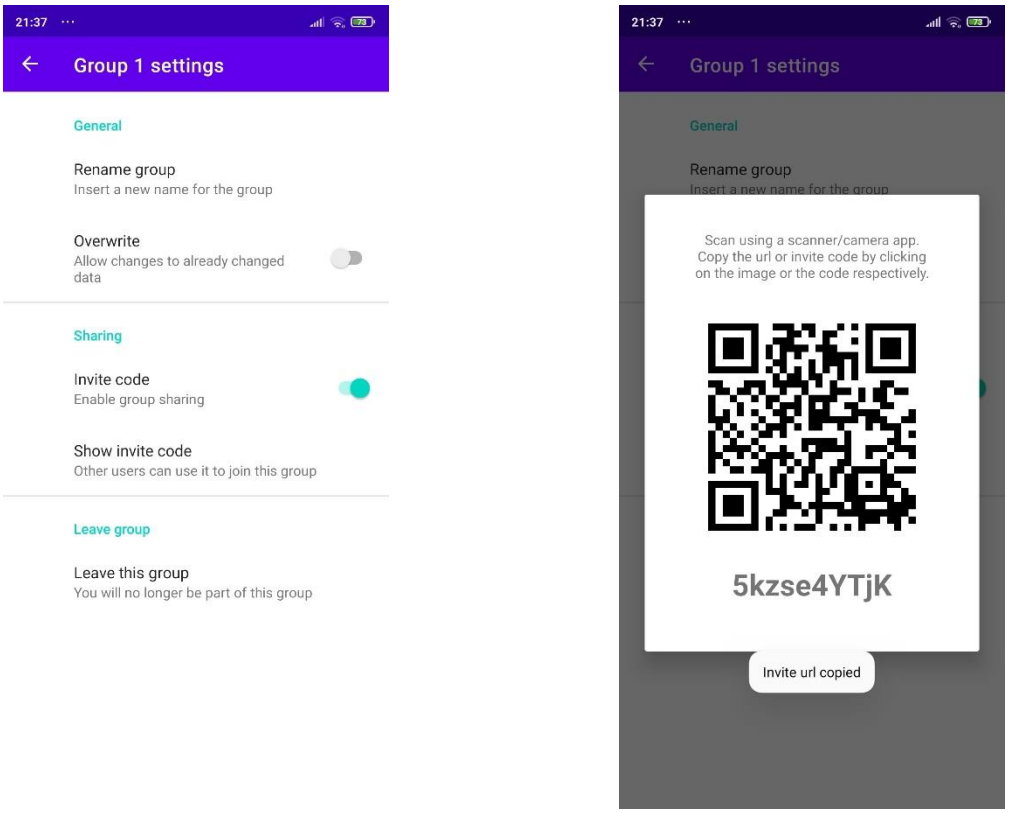
διαδικασία δημιουργίας νέας συναλλαγής. Η πρόσθεση μεταφοράς είναι δυνατή με το παρατεταμένο πάτημα του κουμπιού, ώστε να εμφανιστούν οι επιπλέον επιλογές του. Μπορεί να γίνει απόκρυψη της λίστας με πάτημα του κουμπιού.

Επιπλέον από την παρούσα οθόνη είναι δυνατή η πρόσβαση στις ρυθμίσεις της ομάδας. Οι ρυθμίσεις χωρίζονται σε 3 κατηγορίες και αφορούν τις γενικές ρυθμίσεις ομάδας, τις επιλογές διαμοιρασμού και τη διαχείριση συμμετοχής. Η πρώτη κατηγορία περιλαμβάνει τον έλεγχο του χαρακτηριστικού της αντικατάστασης Περιεχομένου (overwrite). Οι επιλογές διαμοιρασμού επιτρέπουν την ενεργοποίηση ή απενεργοποίηση της δυνατότητας διαμοιρασμού, καθώς και την προβολή του κωδικού. Η διαχείριση συμμετοχής αφορά μονάχα την αποχώρηση από την ομάδα. Αυτή επιτυγχάνεται μόνο αν εκπληρώνονται οι προαναφερθείσες προϋποθέσεις, διαφορετικά εμφανίζεται μήνυμα σφάλματος.

Οι ρυθμίσεις της ομάδας αποθηκεύονται στη βάση δεδομένων στον πίνακα των ομάδων. Όμως η διαχείριση των ρυθμίσεων (προτιμήσεις) πραγματοποιείται με τον επίσημο τρόπο διαχείρισης, ο οποίος προϋποθέτει την χρήση του προεπιλεγμένου API SharedPreferences. Αυτό διευκολύνει την ανάπτυξη της οθόνης που περιλαμβάνει τις προτιμήσεις, καθώς συνεργάζεται απρόσκοπτα με το προαναφερθέν API. Η εφαρμογή συνδυάζει και τους δύο τρόπους. Συγκεκριμένα η ανάκτηση και η αποθήκευση των προτιμήσεων γίνεται μέσω της βάσης δεδομένων. Έπειτα της ανάκτησης, ακολουθεί η ανάθεση των τιμών στις προτιμήσεις με χρήση του API. Παράλληλα παρακολουθείται η βάση για αλλαγές στα δεδομένα της. Έτσι είναι δυνατή η άμεση προβολή των αλλαγών στο περιβάλλον διεπαφής χρήστη (UI). Η πυροδότηση αλλαγών στις προτιμήσεις (από το χρήστη μέσω του UI), εκκινεί αίτημα προς τον εξυπηρετητή, χωρίς όμως να γίνεται αποθήκευσή τους τοπικά. Για το λόγο αυτό, η απουσία σύνδεσης στο διαδίκτυο ή τυχόν σφάλμα στον εξυπηρετητή, έπεται και ακύρωση της διαδικασίας. Η επιτυχής ολοκλήρωση του αιτήματος ενημερώνει τη βάση δεδομένων, ενώ η παρακολούθηση αυτής προκαλεί την ανανέωση των προτιμήσεων και του περιβάλλοντος χρήστη. Επιπροσθέτως, κατά τον συγχρονισμό λαμβάνονται και ενημερώνονται οι ρυθμίσεις κάθε ομάδας.

Όσον αφορά τον διαμοιρασμό της ομάδας, αφού ενεργοποιηθεί η προτίμηση, τότε καθίσταται δυνατή η προβολή του κωδικού πρόσκλησης. Επιπλέον παρέχεται και ένας κωδικός QR για σάρωση μέσω κατάλληλων εφαρμογών. Ο κωδικός QR κωδικοποιεί το σύνδεσμο που καλεί την ανάλογη υπηρεσία στον εξυπηρετητή που πραγματοποιεί την πρόσθεση ενός χρήστη σε μια ομάδα. Αποτελεί ένα deep link

προς την εφαρμογή, οπότε όταν ο χρήστης σαρώσει τον QR κωδικό και προσπαθήσει να ανοίξει τον σύνδεσμο, τότε θα του δοθεί η επιλογή να εκτελέσει αυτή την ενέργεια μέσω της εφαρμογής. Επιλέγοντας αυτή, θα προωθηθεί στην οθόνη των ομάδων με ενεργό το παράθυρο συμμετοχής και συμπληρωμένο το πεδίο του κωδικού πρόσκλησης, αναμένοντας την έγκριση του χρήστη. Πατώντας πάνω στο QR αντιγράφεται ο σύνδεσμος στο πρόχειρο (clipboard), ενώ πατώντας πάνω στον κωδικό έχει ως απόρροια την αντιγραφή αυτού.



Εικόνα 4.21: Εμφάνιση μηνύματος αποτυχημένης προσπάθειας αποχώρησης από την ομάδα λόγω ενεργών ροών χρέους (Αριστερά) και παράθυρο διαμοιρασμού ομάδας και εμφάνιση ειδοποίησης από το πάτημα του κωδικού QR (Δεξιά)

4.2.9 Διαχείριση Περιεχομένου

Η πρόσθεση και μεταβολή του Περιεχομένου γίνεται σε μια σειρά από τις τρεις οθόνες. Οι δύο πρώτες επικεντρώνονται στην εισαγωγή των κύριων δεδομένων, ενώ η τρίτη εμφανίζει συνοπτικά αυτά τα στοιχεία, ενώ παράλληλα επιτρέπει την εισαγωγή προαιρετικών δεδομένων. Σημειώνεται ότι αυτή η τελευταία οθόνη χρησιμοποιείται και για την προβολή του Περιεχομένου, με κάποιες διαφορές. Με το που ο χρήστης πλοηγηθεί σε ένα υπογράφημα Περιεχομένου, εκκινεί το κοινό

ViewModel που διατηρεί τα δεδομένα που είναι κοινόχρηστα μεταξύ των οθονών. Όσον αφορά τις συναλλαγές δημιουργήθηκε το (κοινό) TransactionViewModel, ενώ στις μεταφορές εξετάστηκε η δημιουργία ενός ViewModel ανά οθόνη, καθώς και ενός κοινού, του TransferViewModel.

Παρόλο που η δημιουργία ή η ενημέρωση και των δύο τύπων Περιεχομένου διέπεται από μια σειρά τριών οθονών, αυτό δεν σημαίνει ότι όλες μοιάζουν μεταξύ τους. Κάθε μία είναι κατά τέτοιο τρόπο κατασκευασμένη ώστε να εξυπηρετεί όσο πιο αποδοτικά γίνεται τις ανάγκες του κάθε τύπου. Παρακάτω θα επεξηγηθούν οι οθόνες και οι λειτουργίες τους για κάθε τύπο Περιεχομένου.

Συναλλαγές

Πρώτη οθόνη: Διαχείριση αντικειμένων συναλλαγής

Η πρώτη οθόνη των συναλλαγών χωρίζεται σε τρία κύρια κομμάτια:

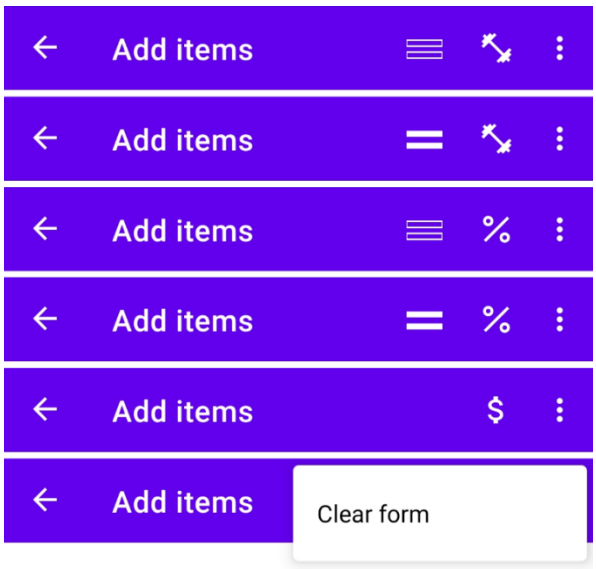
Μπάρα ενεργειών

Βελάκι πλοήγησης: Εμφανίζεται μόνο στην πρώτη από τις τρεις οθόνες. Παρόλο που θα μπορούσε να υπάρχει και στις υπόλοιπες, επιλέχθηκε να αφαιρεθεί από αυτές, καθώς ήδη υπάρχει αντίστοιχο κουμπί στην μπάρα πλοήγησης. Αυτή η πολιτική ακολουθείται και στην περίπτωση των μεταφορών.

Τρόπος κατανομής: Ανάλογα με τον τρόπο κατανομής γίνεται και υπολογισμός του πόσου που αντιστοιχεί σε κάθε χρήστη επί της συνολικής αξίας του αντικειμένου. Διατίθενται τρεις τρόποι κατανομής και κάθε πάτημα του κουμπιού τον αλλάζει σειριακά. Η προεπιλεγμένη μέθοδος είναι τα βάρη. Σε αυτή τη μέθοδο, ο χρήστης εισάγει ένα ακέραιο συντελεστή ή βάρος σε κάθε συμμετέχοντα. Με τη συμπλήρωση όλων των βαρών, αυτά αθροίζονται και παράγονται κλάσματα που με τη σειρά τους χρησιμοποιούνται ώστε να παραχθεί το οφειλόμενο ποσό κάθε συμμετέχοντα. Η επόμενη μέθοδος είναι αυτή των ποσοστών. Όπως μαρτυρά και το όνομά της, ο χρήστης εισάγει έναν ακέραιο από το ένα έως το εκατό για να αντιστοιχίσει το ποσοστό του κάθε συμμετέχοντα στο αντικείμενο. Τέλος, υπάρχει και η μέθοδος των χρημάτων με την απευθείας εισαγωγή του ποσού που καλείται να πληρώσει κάθε μέλος. Σε αυτή την περίπτωση η αξία του αντικειμένου είναι το αποτέλεσμα του αθροίσματος του ποσού κάθε μέλους. Σημειώνεται ότι η εισαγωγή του αριθμού μηδέν έχει την ίδια επίδραση με το κενό πεδίο και δηλώνει την απουσία συμμετοχής του μέλους από την πληρωμή του αντικειμένου.

Ισότητα: Αποτελεί ενέργεια, αλλά και ένδειξη. Πατώντας το κουμπί η αξία του αντικειμένου διανέμεται ισομερώς σε όλα τα μέλη, θέτοντας κοινό βάρος σε όλους. Παρόμοια συμπεριφορά εφαρμόζεται και στην περίπτωση των ποσοστών με την προϋπόθεση ότι το άθροισμα των τιμών ισούται με εκατό. Η ανίχνευση ισοκατανομής αλλάζει τη μορφή του κουμπιού, ώστε να ενημερωθεί ο χρήστης για αυτή. Η ισοκατανομή δεν απαιτεί τη συμμετοχή του συνόλου των μελών, καθώς και ένα μέρος τους αρκεί. Το κουμπί δεν εμφανίζεται όταν ο επιλεγμένος τρόπος κατανομής είναι τα χρήματα, καθώς ο χρήστης είναι υπεύθυνος να αντιστοιχήσει τα ποσά ανά μέλος.

Επιπλέον επιλογές (κρυμμένο μενού): Περιέχει την επιλογή για εκκαθάριση όλων των πεδίων της φόρμας.

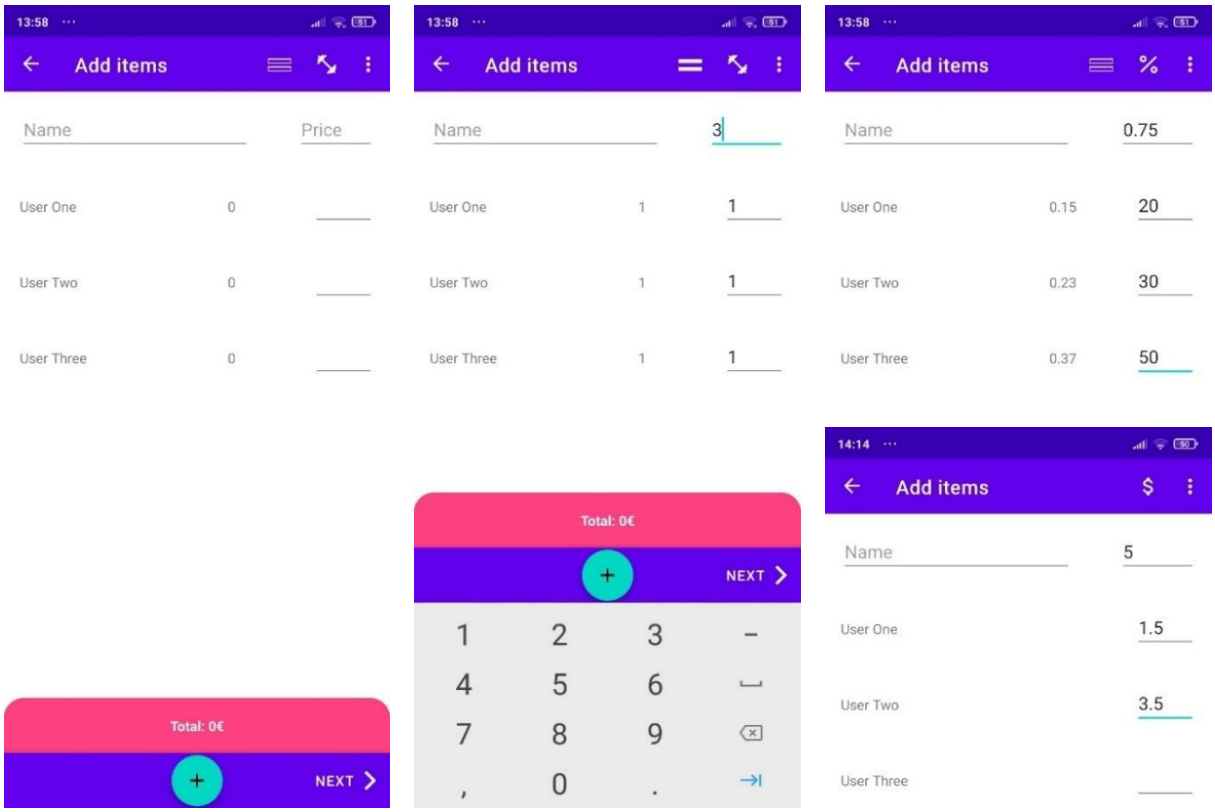


Εικόνα 4.22: Διαφορετικές απεικονίσεις της μπάρας ενεργειών στη δημιουργία/τροποποίηση συναλλαγής και εμφάνιση του μενού επιλογών

Φόρμα αντικειμένου

Η φόρμα πρόσθεσης/ενημέρωσης αντικειμένου αποτελείται από τρία μέρη, το πεδίο του ονόματός του, το πεδίο της αξίας του, καθώς και τη λίστα των μελών. Η λίστα αρχικοποιείται περιέχοντας όλα τα μέλη της ομάδας. Κάθε μέλος αποτελεί μια εγγραφή που περιλαμβάνει το όνομά του, το πεδίο της υπολογισμένης αξίας και το πεδίο του μεριδίου ή ποσού. Το δεύτερο πεδίο (υπολογισμένη αξία) αντιπροσωπεύει το ποσό που καλείται να πληρώσει ο χρήστης με βάση την αξία του αντικειμένου και

του μεριδίου που του αναλογεί και παρέχεται από το τρίτο πεδίο που προαναφέρθηκε.



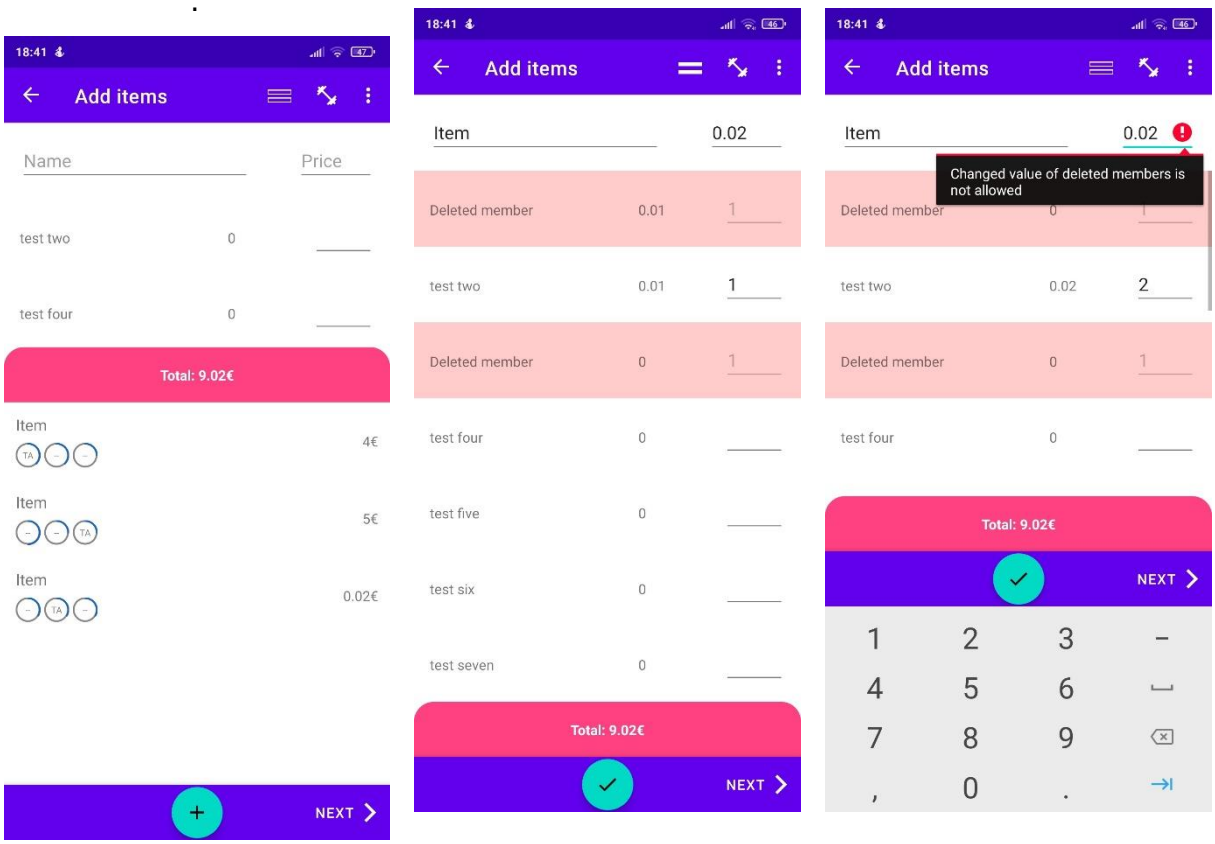
Εικόνα 4.23: Η οθόνη δημιουργίας/τροποποίησης συναλλαγής (Αριστερά), Προσθήκη πρώτου αντικειμένου με ίσα βάρη(Μέση), Προσθήκη δεύτερου αντικειμένου με διαφορετικά ποσοστά (Δεξιά-Πάνω) και προβολή φόρμας με τρόπο κατανομής τα χρήματα (Δεξιά-Κάτω)

Στους δύο πρώτους τρόπους κατανομής, έχουν παρουσία όλα τα πεδία, με το τελευταίο να γίνεται αναφορά σε αυτό ως πεδίο μεριδίου, καθώς εκφράζει συντελεστή βαρύτητας. Στον τρόπο κατανομής με τη μέθοδο των χρημάτων, το τελευταίο πεδίο αναφέρεται ως πεδίο ποσού και αποκρύπτεται το πεδίο της υπολογισμένης αξίας, καθώς δεν εκτελείται κάποιος υπολογισμός. Κάθε αλλαγή του πεδίου του μεριδίου ή της αξίας του αντικειμένου, προκαλεί τον επανυπολογισμό των ποσών που καλούνται να πληρώσουν οι χρήστες.

Η μπάρα ενεργειών έχει άμεση σχέση με τη φόρμα, καθώς κάθε ενέργειά της επιδρά αυτή. Η ενέργεια της ισότητας, αλλάζει τις τιμές των μεριδίων, ενώ ο τρόπος κατανομής αλλάζει τον τρόπο υπολογισμού της αξίας που αναλογεί στον κάθε χρήστη. Κάθε μία από τις ενέργειες της μπάρας, συμπεριλαμβανομένου και της εκκαθάρισης της φόρμας, μπορεί να προκαλέσει την ανανέωση των πεδίων της

λίστας με νέα δεδομένα. Τα νέα δεδομένα τροφοδοτούν την κλάση ContentProcessor, η οποία είναι υπεύθυνη για τους υπολογισμούς.

Υπάρχει περίπτωση να χρειαστεί η ανανέωση των δεδομένων ενός αντικειμένου, αλλά κάποιο μέλος που συμμετείχε σε αυτό να έχει αποχωρήσει από την ομάδα. Η επιλογή της εγγραφής από τη λίστα αντικειμένων θα μεταφέρει τα στοιχεία του στη φόρμα, όπως επίσης θα προσθέσει τυχόν εγγραφές απόντων μελών. Αυτές οι εγγραφές έχουν διαφορετική εμφάνιση και δεν επιτρέπεται η αλλαγή της τιμής του πεδίου μεριδίου/ποσού. Δεν υπάρχει κάποιος περιορισμός για τις υπόλοιπες εγγραφές των μελών. Σημειώνεται ότι αν για παράδειγμα ανανεωνόταν ένα αντικείμενο με μέθοδο κατανομής τα βάρη και μεταβληθεί η αξία του αντικειμένου, τότε είναι πιθανό να αλλάξει η υπολογισμένη αξία και του διαγραμμένου χρήστη και για το λόγο αυτό δεν θα επιτραπεί η εκτέλεση του αιτήματος.



Εικόνα 4.24: Παράδειγμα ανανέωσης συναλλαγής με χρήστες που έχουν αποχωρήσει: Αντικείμενα συναλλαγής (Αριστερά), Μεταφορά δεδομένων τρίτου αντικειμένου στη φόρμα για ανανέωση (Μέση), Μήνυμα σφάλματος για απόρριψη αλλαγών σε διαγραμμένα μέλη (Δεξιά)

Σε προηγούμενο κεφάλαιο είχαν αναφερθεί οι λόγοι που η εφαρμογή διαχειρίζεται τα χρήματα ως ακεραίους. Αυτό συμβαίνει και στο περιεχόμενο του τρίτου πεδίου, καθώς μετατρέπεται σε τύπο δεδομένων Long, του οποίου η μέγιστη τιμή συμπίπτει με εκείνη του τύπου Big Integer της PostgreSQL, ήτοι 9.223.372.036.854.775.807. Η μετατροπή του τύπου δεδομένων εκτελεί ελέγχους που ανιχνεύουν περιπτώσεις σφαλμάτων, όπως είναι η υπερχείλιση του αριθμού (υπέρβαση ορίου).

Ανεξαρτήτως με το πως διαχειρίζεται η εφαρμογή τα χρήματα, αυτό δεν αλλάζει κάτι στον τρόπο εισαγωγής τους από τον χρήστη. Οι μόνοι περιορισμοί που εφαρμόζονται είναι το εύρος τιμών του πεδίου, το οποίο μεταβάλλονται βάσει του επιλεγμένου τρόπου κατανομής του αντικειμένου. Σε όλες τις περιπτώσεις το εύρος εκκινεί από το μηδέν, που δηλώνει και απουσία συμμετοχής. Συγκεκριμένα, τα εύρη που μπορεί να εισάγει ο χρήστης ορίζονται ως εξής:

- Ποσοστά

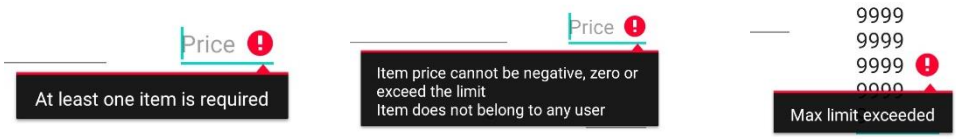
  Όπως είναι φυσικό το εύρος για ένα ποσοστό είναι από μηδέν έως εκατό και γίνεται αποδοχή μόνο ακέραιων τιμών.
- Χρήματα

  Τα πληρωτέα ποσά διαθέτουν μέχρι και 2 δεκαδικά ψηφία, οπότε και η μέγιστη τιμή σε ευρώ που μπορεί να αναπαρασταθεί είναι η 92.233.720.368.547.758,07.
- Βάρη

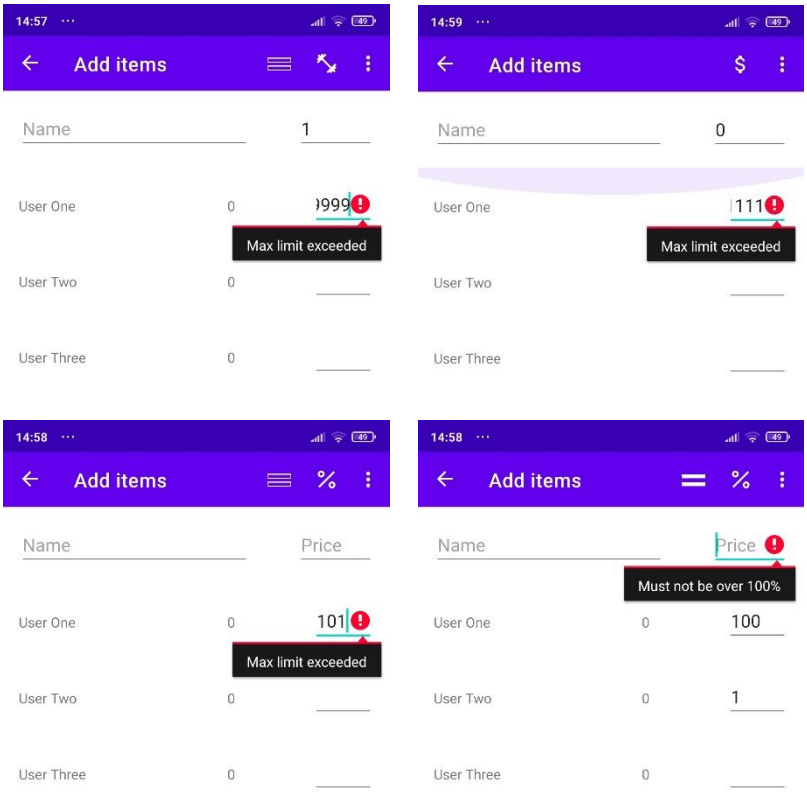
  Η μέγιστη τιμή του βάρους μπορεί να οριστεί αυθαίρετα, αρκεί να μην υπερβαίνει το όριο των τύπων δεδομένων Big Integer/Long. Παρόλα αυτά πριν θέσουμε το μέγιστο όριο, θα πρέπει να ληφθεί υπόψιν ότι για να πραγματοποιηθεί ο υπολογισμός των κλασμάτων θα πρέπει να βρεθεί το συνολικό βάρος, καθώς αυτό αποτελεί τον παρονομαστή τους. Υπολογίζεται αθροίζοντας το βάρος του κάθε χρήστη. Αν για παράδειγμα επιλεχθεί ένας μεγάλος αριθμός, τότε είναι πιο εύκολο το άθροισμα τους να προκαλεί υπερχείλιση του τύπου δεδομένων. Αυτό δεν είναι πρόβλημα για το πρόγραμμα, καθώς μπορεί να διαχειριστεί τέτοιες περιπτώσεις, αλλά θα μπορούσε να επηρεάσει την εμπειρία του χρήστη, ζητώντας του πολλές φορές να αλλάξει τις τιμές των βαρών σε μικρότερες. Επιπλέον ένα τεράστιο όριο δεν φαίνεται να προσδίδει ιδιαίτερα πλεονεκτήματα, καθώς οι περισσότερες περιπτώσεις χρήσης δεν απαιτούν κάτι τέτοιο, ενώ ταυτόχρονα

μπορεί να δυσκολέψουν τον χρήστη στην εισαγωγή και την παρακολούθηση των τιμών. Παρόλα αυτά μπορεί κάποια μερίδα των χρηστών να αναζητούν αυτή τη δυνατότητα. Για τους παραπάνω λόγους επιλέχθηκε ένα σχετικά μεγάλο νούμερο, αλλά μικρότερο του μεγίστου που συναντάμε στους τύπους δεδομένων. Το τελικό εύρος ορίζεται για ακέραιους αριθμούς από το μηδέν (0) έως το 999.999.999.999.999.

Οι έλεγχοι των τύπων και των ορίων περιλαμβάνονται στους γενικούς ελέγχους που γίνονται σε πραγματικό χρόνο. Ένας πρώτος έλεγχος αποτελεί η θέσπιση των κατάλληλων ρυθμίσεων σε κάθε πεδίο, ώστε να δέχεται συγκεκριμένες τιμές (πχ. αριθμοί) και έτσι να εμφανίζεται το πληκτρολόγιο με την ορθή διάταξη. Επίσης χρησιμοποιούνται φίλτρα (Text Filters) και ελεγκτές κειμένου (Text Watchers). Το πρώτο ελέγχει και τροποποιεί το περιεχόμενο του πεδίου, καθώς αυτό εισάγεται από τον χρήστη (πχ. αποδοχή υποδιαστολής ανάλογα του τύπου του αριθμού), ενώ το δεύτερο ελέγχει το κείμενο για σφάλματα, ώστε να εμφανιστούν τα ανάλογα μηνύματα (πχ. υπέρβαση μέγιστων ορίων). Πατώντας το κουμπί της πρόσθεσης αντικειμένου πραγματοποιείται ένας τελικός, συγκεντρωτικός έλεγχος ορθότητας των δεδομένων του αντικειμένου. Περιλαμβάνει και προηγούμενους ελέγχους (έλεγχος εύρους), καθώς και νέους, όπως η αξία του αντικειμένου να μην είναι μηδενική, αρνητική ή μεγαλύτερη του ορίου, το αντικείμενο να διαμοιράζεται σε τουλάχιστον ένα χρήστη και το άθροισμα του ποσού που καλούνται να πληρώσουν τα μέλη να αντιστοιχεί στη συνολική αξία του αντικειμένου. Σημειώνεται ότι δεν είναι αναγκαία η συμπλήρωση του πεδίου του ονόματος της φόρμας. Ο χρήστης μπορεί να το αφήσει κενό, ώστε να αποδοθεί αυτόματα η προεπιλεγμένη τιμή «Item». Στο πεδίο της αξίας αντικειμένου εμφανίζονται μηνύματα σφάλματος που την αφορούν, καθώς επίσης και μηνύματα που προέκυψαν από πιο γενικούς ελέγχους. Έχοντας ολοκληρώσει τις ενέργειες που αφορούν τα αντικείμενα, ο χρήστης μπορεί να προχωρήσει στην επόμενη οθόνη. Σε αυτό το σημείο θα εκτελεστεί ένας τελικός έλεγχος ορθότητας που περιλαμβάνει την ικανοποίηση της συνθήκης ότι η λίστα των αντικειμένων δεν είναι κενή, καθώς μια συναλλαγή απαιτεί τουλάχιστον ένα.



Εικόνα 4.25: Μηνύματα σφάλματος που αφορούν την τιμή, καθώς και γενικού τύπου

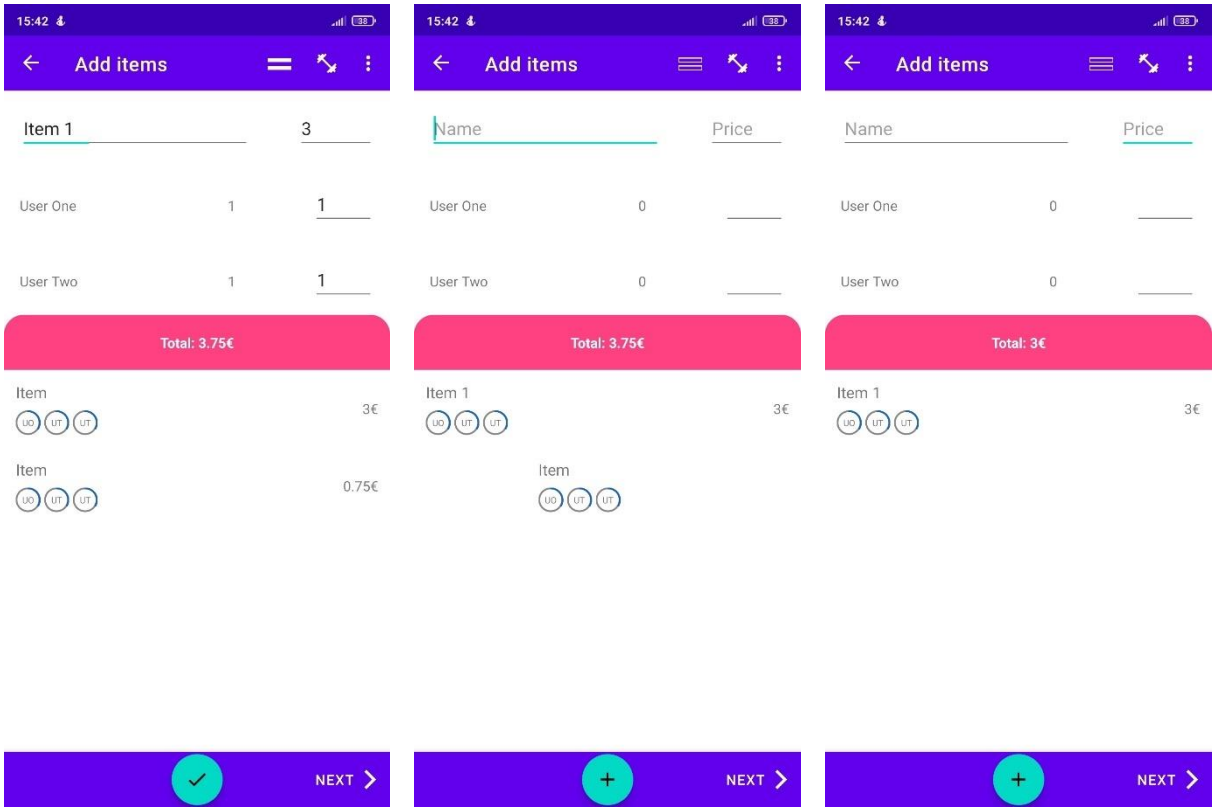


Εικόνα 4.26: Μηνύματα σφάλματος που αφορούν μερίδια

Λίστα αντικειμένων

Η πρόσθεση αντικειμένου έχει ως αποτέλεσμα την εκκαθάριση των πεδίων της φόρμας και την προσθήκη αυτού στη λίστα αντικειμένων. Η λίστα αυτή εμφανίζεται με χειρονομία συρσίματος προς τα πάνω, της μπάρας που εμφανίζει τη συνολική αξία της συναλλαγής. Η συνολική αξία προκύπτει από το άθροισμα της αξίας των αντικειμένων. Κάθε εγγραφή της λίστας παρουσιάζει συνοπτικά τις πληροφορίες ενός αντικειμένου, συμπεριλαμβάνοντας το όνομα, την τιμή, καθώς και των κύκλων προόδου των μελών. Τα μέλη είναι ταξινομημένα κατά αύξουσα σειρά βάσει της συμμετοχής τους, ενώ γίνεται αντιληπτός και ο συνολικός αριθμός των

συμμετεχόντων. Πατώντας σε κάθε εγγραφή, τα δεδομένα του αντικειμένου μεταφέρονται εκ νέου στη φόρμα, ώστε να είναι δυνατή η μεταβολή τους. Η διαγραφή του αντικειμένου επιτυγχάνεται με χειρονομία δεξιού συρσίματος της εγγραφής.



Εικόνα 4.27: Λίστα αντικειμένων με 2 αντικείμενα και επιλεγμένο το πρώτο για τροποποίηση (Αριστερά), Διαγραφή δεύτερου αντικειμένου με χειρονομία δεξιού συρσίματος (Μέση), Απουσία δεύτερου αντικειμένου και ανανεωμένη συνολική αξία (Δεξιά)

Προσθήκη αντικειμένου και πλοήγηση

Η μπάρα πλοήγησης είναι υπεύθυνη για την πλοήγηση μεταξύ των τριών οθονών, ενώ στην πρώτη οθόνη φιλοξενεί και το κουμπί της πρόσθεσης/ανανέωσης αντικειμένου.

Κάθε ενέργεια που αφορά τα αντικείμενα, όπως οι παραπάνω που αναφέρθηκαν (προσθήκη, ανανέωση, διαγραφή) ή ο υπολογισμός των ποσών που αντιστοιχεί σε κάθε μέλος, εκτελείται από την κλάση ContentProcessor, της οποίας ένα instance δημιουργείται με την εκκίνηση του ViewModel των Συναλλαγών. Ως όρισμα δέχεται την ομάδα με τα μέλη της και τα αποθηκεύει στη μνήμη, ώστε να αποκτήσει μια εικόνα της κατάστασης τους. Αυτή είναι τύπου GroupMember και αποτελεί υποκλάση

της ContentMember. Η τελευταία περιλαμβάνει το αναγνωριστικό, το όνομα και τα υπόλοιπα τους, τα οποία μπορούν να μεταβάλλονται σε πραγματικό χρόνο. Σε αυτό το σημείο ως υπόλοιπα νοούνται το πληρωτέο και μη πληρωτέο (ως ακέραιος και κλάσμα) ποσό κάθε μέλους που προκύπτει από τα αντικείμενα στα οποία συμμετέχει. Αντίστοιχα υπόλοιπα αφορούν και την τρέχουσα συναλλαγή, καθώς προκύπτουν από το σύνολο των αντικειμένων της. Στην κλάση GroupMember περιλαμβάνεται και μια μεταβλητή που διατηρεί το δευτερεύων υπόλοιπο του μέλους, όπως αυτό ανακτάται από τη βάση δεδομένων. Αυτή χρησιμοποιείται για να βρεθεί το μέλος με το μικρότερο δευτερεύων υπόλοιπο, το οποίο και θα επιλεγεί να καλύψει το μη πληρωτέο ποσό, αν αυτό προκύψει σε κάποιο αντικείμενο. Η διατήρηση των υπολοίπων στη μνήμη είναι σημαντική, καθώς φροντίζει κάθε νέος υπολογισμός σε κάποιο αντικείμενο να βασίζεται στα πιο πρόσφατα υπόλοιπα που περιέχουν την αξία των αντικειμένων της τρέχουσας συναλλαγής (αν και αυτή ακόμα δεν έχει αποδεχτή).

Τόσο η πρόσθεση, όσο και η διαγραφή αποτελούν απλές ενέργειες. Η πρώτη ενέργεια προσθέτει τα υπολογισμένα ποσά κάθε μέλους στα χρωστούμενα τους, ενώ η άλλη εκτελεί το αντίθετο, δηλαδή τα αφαιρεί. Η ενημέρωση ενός αντικειμένου είναι πιο πολύπλοκη διαδικασία, καθώς τα υπόλοιπα θα πρέπει να είναι στην κατάσταση που προηγούταν της πρόσθεσης του αντικειμένου. Διαφορετικά, οι νέοι υπολογισμοί θα βασίζονταν σε υπόλοιπα που περιέχουν την αξία του αντικειμένου στο οποίο γίνεται επεξεργασία. Για το λόγο αυτό σε κάθε ενημέρωση αντικειμένου, προσωρινά αφαιρούνται από τα υπόλοιπα των μελών, τα ποσά που είχαν υπολογιστεί και το αφορούν. Ο λόγος που γίνεται αυτό προσωρινά είναι επειδή ο χρήστης θα μπορούσε να αλλάξει τα δεδομένα του αντικειμένου, αλλά να ακυρώσει τη διαδικασία μέσω της εκκαθάρισης της φόρμας. Σε αυτή την περίπτωση η κατάσταση των υπολοίπων θα πρέπει να αποκατασταθεί. Παρακάτω περιγράφονται οι λειτουργίες της, οι μεταβλητές και οι μέθοδοί της:

- Μεταβλητή members: Η λίστα των μελών της τρέχουσας συναλλαγής. Διατηρεί ένα αντίγραφο της αρχικής κατάστασης τους, το οποίο και επεξεργάζεται κατά τη διάρκεια της δημιουργίας/τροποποίησης της συναλλαγής. Ο τύπος δεδομένων της λίστας είναι ο GroupMember. Αυτά τα δεδομένα μπορούν στη συνέχεια να εφαρμοστούν στα γενικά υπόλοιπα των μελών και να υπολογιστούν οι νέες ροές χρέους, αρκεί η συναλλαγή να σταλεί και να γίνει αποδεκτή από τον εξυπηρετητή.

- Μεταβλητή oldItem: Φιλοξενεί προσωρινά την αρχική κατάσταση του αντικειμένου προς ενημέρωση. Με αυτό τον τρόπο μπορεί γίνει σύγκριση με τη νέα κατάσταση και να ελεγχθεί τυχόν μεταβολή των δεδομένων για μέλη που έχουν αποχωρήσει. Η σύγκριση πραγματοποιείται μέσω κλήσης της συνάρτησης compareItemUserShares.
- compareItemUserShares: Παίρνει ως ορίσματα την αρχική και νέα κατάσταση του αντικειμένου. Επιβεβαιώνει ότι η νέα κατάσταση δεν παραβιάζει την αρχή ότι τα δεδομένα των μελών που έχουν αποχωρήσει παραμένουν αμετάβλητα. Σε περίπτωση παράβασης, απορρίπτεται η ανανέωση (του αντικειμένου) με τη δημιουργία αντίστοιχης εξαίρεσης.
- temp: Δέχεται ως όρισμα ένα αντικείμενο της συναλλαγής ή το κενό (null). Θέτει το όρισμα στη μεταβλητή που φιλοξενεί προσωρινά την αρχική κατάσταση του αντικειμένου (oldItem). Όμως, πριν από αυτό ελέγχει αν κάποιο αντικείμενο βρίσκεται ήδη σε διαδικασία επεξεργασίας. Αν αυτή η συνθήκη δεν ισχύει, και μόνο τότε, γίνεται έλεγχος για το αν η παράμετρος είναι κενή ή όχι. Για συμπληρωμένη παράμετρο, γίνεται αντιγραφή της κατάστασης των μελών της ομάδας και στο εξής κάθε μεταβολή του αντικειμένου αντικατοπτρίζεται σε αυτά τα δεδομένα. Η αντίθετη περίπτωση δείχνει ότι γίνεται ολοκλήρωση της διαδικασίας ανανέωσης, οπότε και η αρχική κατάσταση των μελών αντικαθίσταται με αυτήν που αντιγράφηκε προηγουμένως και περιέχει τις αλλαγές. Με αυτό τον τρόπο η ανανέωση των δεδομένων των μελών γίνεται σε δύο φάσεις και έτσι είναι πιο εύκολη η ακύρωση της διαδικασίας ενημέρωσης.
- cancelUpdate: Εκτελεί την ακύρωση της ανανέωσης του επιλεγμένου αντικειμένου, δίχως την αποθήκευση αλλαγών, επαναφέροντας την κατάσταση των μελών όπως ήταν πριν. Για να επιτευχθεί αυτό, ελέγχεται αν κάποιο αντικείμενο βρίσκεται σε διαδικασία ενημέρωσης. Αν η συνθήκη ισχύει, τότε τα δεδομένα των μελών επαναφέρονται εύκολα, καθώς η αρχική κατάσταση παραμένει στη μνήμη, έτοιμη για επαναφορά. Υπενθυμίζεται ότι τυχόν αλλαγές, έχουν εφαρμοστεί σε αντίγραφο της, το οποίο και καταστρέφεται. Τέλος, θέτεται η κενή τιμή στη μεταβλητή oldItem.
- endItemUpdate: Παίρνει ως παράμετρο ένα αντικείμενο. Καλείται ώστε να επιβεβαιώσει την ολοκλήρωση της ανανέωσης κάποιου αντικειμένου. Πρώτη

του μέριμνα είναι ο έλεγχος του αμετάβλητου των δεδομένων των απόντων μελών μέσω της μεθόδου compareItemUserShares. Στη συνέχεια καλεί τη συνάρτηση temp με όρισμα την κενή (null) παράμετρο, ώστε να ολοκληρωθεί η ενημέρωση με την οριστικοποίηση της νέας κατάστασης των μελών.

- addItem: Λαμβάνει ως παράμετρο ένα αντικείμενο της συναλλαγής. Σκοπός είναι η προσθήκη του αντικειμένου με την ανανέωση των υπόλοιπων των μελών (της συναλλαγής), καλώντας τη μέθοδο updateGroupMembers. Όμως, πριν από αυτό θα πρέπει να κληθεί η μέθοδος endItemUpdate, ώστε να ολοκληρωθεί η διαδικασία σε περίπτωση ανανέωσης του αντικειμένου έναντι της προσθήκης νέου.
- removeItem: Σκοπός είναι η αφαίρεση του αντικειμένου που παρέχεται στη μέθοδο ως παράμετρος. Αποτελεί αντίστροφη διαδικασία της πρόσθεσης αντικειμένου, καθώς αφαιρούνται τα υπολογισμένα ποσά από τα υπόλοιπά των μελών. Η ανανέωση των υπολοίπων επιτυγχάνεται με κλήση της συνάρτησης updateGroupMemberByShareMember.
- itemUpdate: Δέχεται ως παράμετρο το αντικείμενο προς ενημέρωση. Αρχικά ακυρώνεται πιθανή ενημέρωση άλλου αντικειμένου με κλήση της cancelUpdate. Στη συνέχεια καλείται η συνάρτηση temp παρέχοντας της το αντικείμενο, ώστε να γίνει η προεργασία της αντιγραφής της κατάστασης των μελών. Τέλος, γίνεται αφαίρεση του αντικειμένου από την καινούργια κατάσταση, ώστε οι νέοι υπολογισμοί που διενεργούνται από τη συνάρτηση calculateShares, να μη στηρίζονται σε υπόλοιπα που περιλαμβάνουν το αντικείμενο που τώρα ανανεώνεται.
- updateGroupMembers: Λαμβάνει ως όρισμα το αντικείμενο. Συντονίζει την ανανέωση των υπολοίπων των μελών. Αρχικά βρίσκονται τα μέλη που συμμετέχουν στην πληρωμή του αντικειμένου, καλώντας τη συνάρτηση findGroupMembersByShareMembers. Στη συνέχεια, για κάθε χρήστη, εκτελείται κλήση στη μέθοδο updateGroupMemberByShareMember για να ανανεωθούν τα υπόλοιπά του βάσει των ποσών που υπολόγισε η calculateShares.

  calculateShares: Δέχεται ως ορίσματα τον τρόπο κατανομής, την αξία του αντικειμένου, καθώς και τα μερίδια των χρηστών. Καλείται κάθε φορά που χρειάζεται να επανυπολογιστεί η αξία που πρέπει να πληρώσει κάθε

συμμετέχων. Αυτό συμβαίνει με τη μεταβολή οποιασδήποτε παραμέτρου της συνάρτησης. Ο τύπος δεδομένων του μεριδίου, δηλαδή η κλάση Share, περιλαμβάνει την τιμή του μεριδίου και μια ιδιότητα τύπου ContentMember. Ο τελευταίος τύπος συμπληρώνεται με δεδομένα που υπολογίζονται από την παρούσα συνάρτηση και αφορούν το συμμετέχων μέλος για το αντικείμενο. Τα δεδομένα περιλαμβάνουν το πληρωτέο και το δευτερεύων ποσό που του αντιστοιχεί. Το δεύτερο εκφράζεται τόσο ως τιμή (ακέραιος αριθμός), όσο και ως κλάσμα. Η συνάρτηση αποτελεί την υλοποίηση του αλγορίθμου υπολογισμού ποσών βάσει κατανομής και οι υπολογισμοί της προορίζονται για εφαρμογή στην κατάσταση των μελών. Προαιρετικά, επιστρέφει μια λίστα με τα αντικείμενα τύπου Share που περιέχουν σφάλματα, ώστε να εμφανιστούν στον χρήστη.

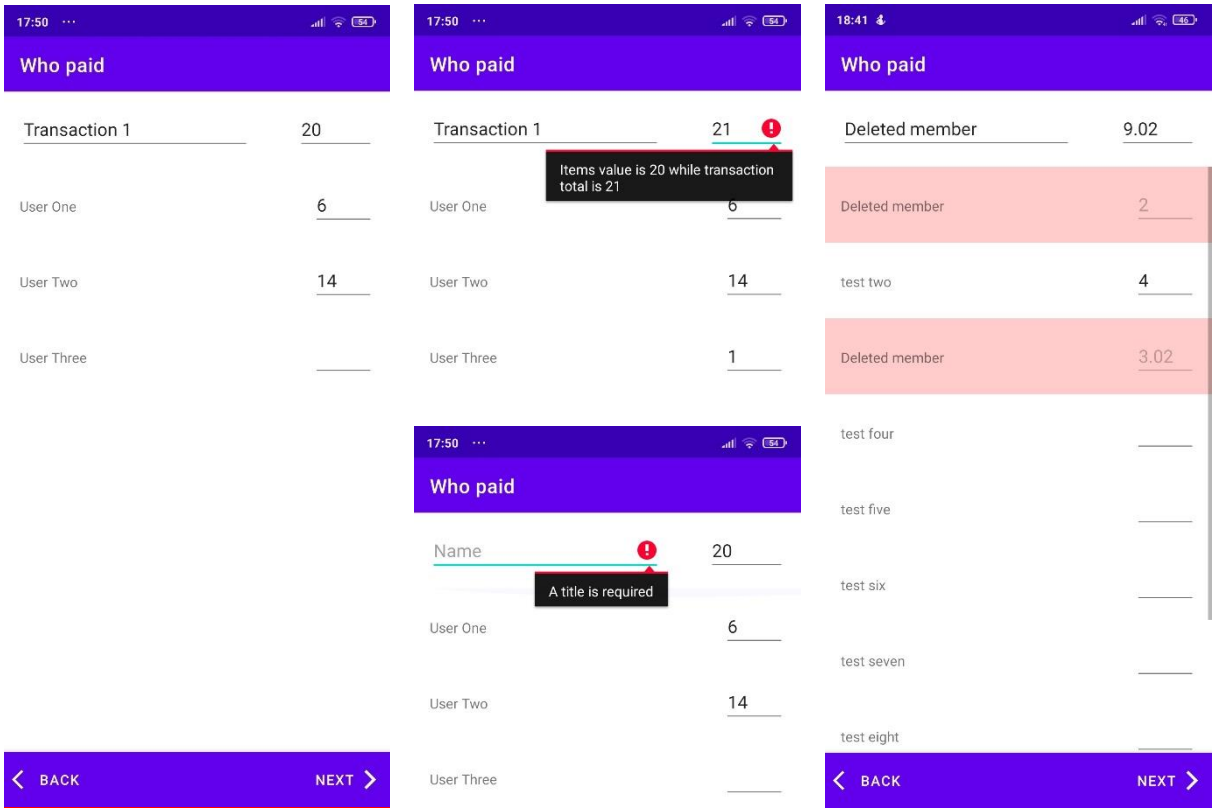
- sortGroupMembersByTotalEp: Δέχεται ως παράμετρο τη λίστα των μελών και σκοπό έχει να επιστρέψει αυτή τη λίστα ταξινομημένη βάσει των δευτερευόντων υπολοίπων. Έτσι κάθε φορά που προκύπτει μη πληρωτέο ποσό για το αντικείμενο, η ταξινομημένη κατά αύξουσα σειρά λίστα βοηθάει να βρεθούν τα μέλη που έχουν το χαμηλότερο δευτερεύων υπόλοιπο, ώστε να επιλεχθούν αυτόματα για να το πληρώσουν.
- findGroupMembersByShareMembers: Δέχεται ως όρισμα τα μέλη της ομάδας και τα μέλη που συμμετέχουν στην πληρωμή του αντικειμένου, δηλαδή κατέχουν μερίδια. Επιστρέφει τα μέλη της πρώτης λίστας που βρέθηκε από τη δεύτερη λίστα ότι κατέχουν μερίδια.
- updateGroupMemberByShareMember: Σκοπός είναι να ανανεώσει τα υπόλοιπα ενός μέλους για την τρέχουσα συναλλαγή. Δέχεται ως ορίσματα ένα μέλος της ομάδας, τα υπολογισμένα δεδομένα σύμφωνα με το μερίδιο του στο αντικείμενο, καθώς και αν η διαδικασία πρέπει να αντιστραφεί, κάτι που συμβαίνει όταν ένα αντικείμενο αφαιρείται. Όταν ένα αντικείμενο προστίθεται, τότε μειώνονται τα υπόλοιπα του μέλους, καθώς αθροίζονται σε αυτά τα πληρωτέα ή μη πληρωτέα ποσά που προκύπτουν από τον διαμοιρασμό της αξίας του αντικειμένου. Υπενθυμίζεται ότι τα παραπάνω ποσά αποτελούν χρέωση, οπότε διαθέτουν αρνητικό πρόσημο. Όπως είναι φυσικό αν υπάρξει μη πληρωτέο ποσό, τότε αυτό προστίθεται και στην αντίστοιχη μεταβλητή της

κλάσης GroupMember, ώστε να είναι εύκολη η εύρεση των ατόμων που είναι η σειρά τους να το πληρώσουν.

- findShareByBuyerId: Λαμβάνει ως ορίσματα το αναγνωριστικό χρήστη του μεριδίου, καθώς και τα μερίδια που υπολογίστηκαν. Βρίσκει ποιο μερίδιο αντιστοιχίζεται σε αυτό το αναγνωριστικό.

Δεύτερη οθόνη: Αξία συναλλαγής και πιστωτές

Μοιάζει με την φόρμα διαχείρισης αντικειμένων έχοντας επιλέξει για μέθοδο κατανομής τα χρήματα. Σε αυτή ο χρήστης καλείται να συμπληρώσει τα πεδία με το ποσό που πλήρωσε ο κάθε χρήστης για τη συναλλαγή. Αυτά τα άτομα ονομάζονται και πιστωτές. Ακόμα και μέλη που δεν έλαβαν μέρος στη συναλλαγή μέσω αγοράς αντικειμένων, μπορούν να συμμετάσχουν στην αποπληρωμή της. Τα ποσά αυτά αθροίζονται σε πραγματικό χρόνο, εμφανίζοντας το συνολικό ποσό που έχει συγκεντρωθεί. Για να προχωρήσει ο χρήστης στην επόμενη οθόνη πραγματοποιείται ο έλεγχος σύγκρισης της αθροιστικής αξίας των αντικειμένων με το σύνολο των ποσών που διέθεσαν οι χρήστες για πληρωμή. Αν δεν ταιριάζουν μεταξύ τους, τότε εμφανίζεται αντίστοιχο μήνυμα σφάλματος στο πεδίο της συνολικής αξίας της συναλλαγής. Κατά την ανανέωσης μιας συναλλαγής, οι εγγραφές των χρηστών που αποτελούν πιστωτές αλλά αποχώρησαν από την ομάδα, εμφανίζονται αλλαγμένες με τρόπο παρόμοιο με αυτόν που αναφέρθηκε προηγουμένως. Ομοίως, δεν επιτρέπεται καμία αλλαγή για τα συγκεκριμένα άτομα. Το πεδίο του ονόματος συναλλαγής δεν είναι προαιρετικό και απαιτείται η συμπλήρωσή του από τον χρήστη. Διαφορετικά, εμφανίζεται μήνυμα σφάλματος στο αντίστοιχο πεδίο.



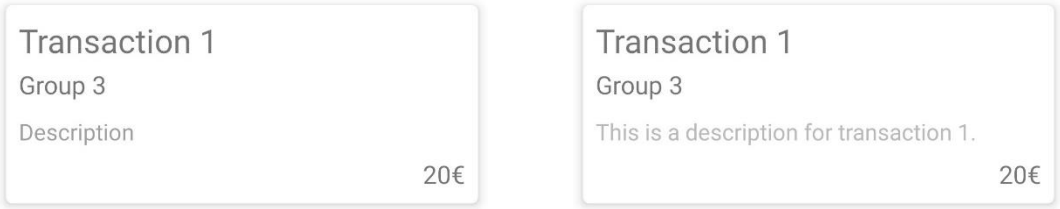
Εικόνα 4.28: Ορισμός του ποσού που πληρώνει (πίστωση) κάθε χρήστης για τη συναλλαγή (Αριστερά), Εμφάνιση μηνύματος σφάλματος σε περίπτωση λάθος πιστώσεων (Μέση-Πάνω) και απουσίας υποχρεωτικού πεδίου τίτλου (Μέση-Κάτω), Παράδειγμα εμφάνισης διαγραμμένων μελών στην οθόνη πιστωτών

Τρίτη οθόνη: Επισκόπηση συναλλαγής

Όπως αναφέρθηκε αποτελεί τη μόνη κοινή οθόνη μεταξύ των τύπων Περιεχομένου. Όμως, σε κάθε περίπτωση είναι προσαρμοσμένη για τον εκάστοτε τύπο. Αποτελείται από μια λίστα από κάρτες. Σκοπός κάθε κάρτας είναι η συνοπτική περιγραφή κάθε πτυχής του Περιεχομένου.

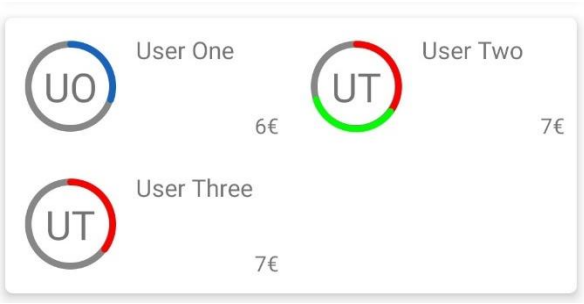
Όσον αφορά τις συναλλαγές συναντιόνται οι παρακάτω κάρτες:

- Κεντρική: Αυτή η κάρτα περιλαμβάνει το όνομα της συναλλαγής και της ομάδας, καθώς και τη συνολική της αξία. Επιπλέον, δίνεται η δυνατότητα στο χρήστη να προσθέσει μια προαιρετική περιγραφή.



Εικόνα 4.29: Κεντρική κάρτα στην αρχική της μορφή (Αριστερά) και με συμπληρωμένο το πεδίο της περιγραφής (Δεξιά)

- Συμμετέχοντες: Σε αυτή την κάρτα παρατίθενται οι χρήστες που συμμετέχουν στη συναλλαγή, ανεξαρτήτως αν είναι χρεώστες, πιστωτές ή και τα δύο. Κάθε εγγραφή αποτελείται από ένα κύκλο προόδου με τα αρχικά του ονόματος του χρήστη στο εσωτερικό του, το πλήρες όνομα χρήστη, καθώς και τη χρέωση, δηλαδή τι έπρεπε να πληρώσει για τη συναλλαγή. Ο κύκλος προόδου εμφανίζει το ποσοστό χρέωσης και πίστωσης του χρήστη επί της συνολικής αξίας της συναλλαγής. Η μορφή του αλλάζει ανάλογα με το τι καλείται να πληρώσει και τι πλήρωσε ο κάθε χρήστης. Αν η πίστωση και η χρέωση είναι ίδιες, τότε ο κύκλος είναι μονόχρωμος (μπλε). Διαφορετικά χρησιμοποιούνται δύο χρώματα, κόκκινο για χρέωση και πράσινο για πίστωση. Τα δύο χρώματα επικαλύπτουν το ένα το άλλο, οπότε εκείνο που έχει το μεγαλύτερο ποσοστό προόδου πληροφορεί για το αν ο χρήστης πλήρωσε παραπάνω από όσα χρεώθηκε ή το αντίθετο.

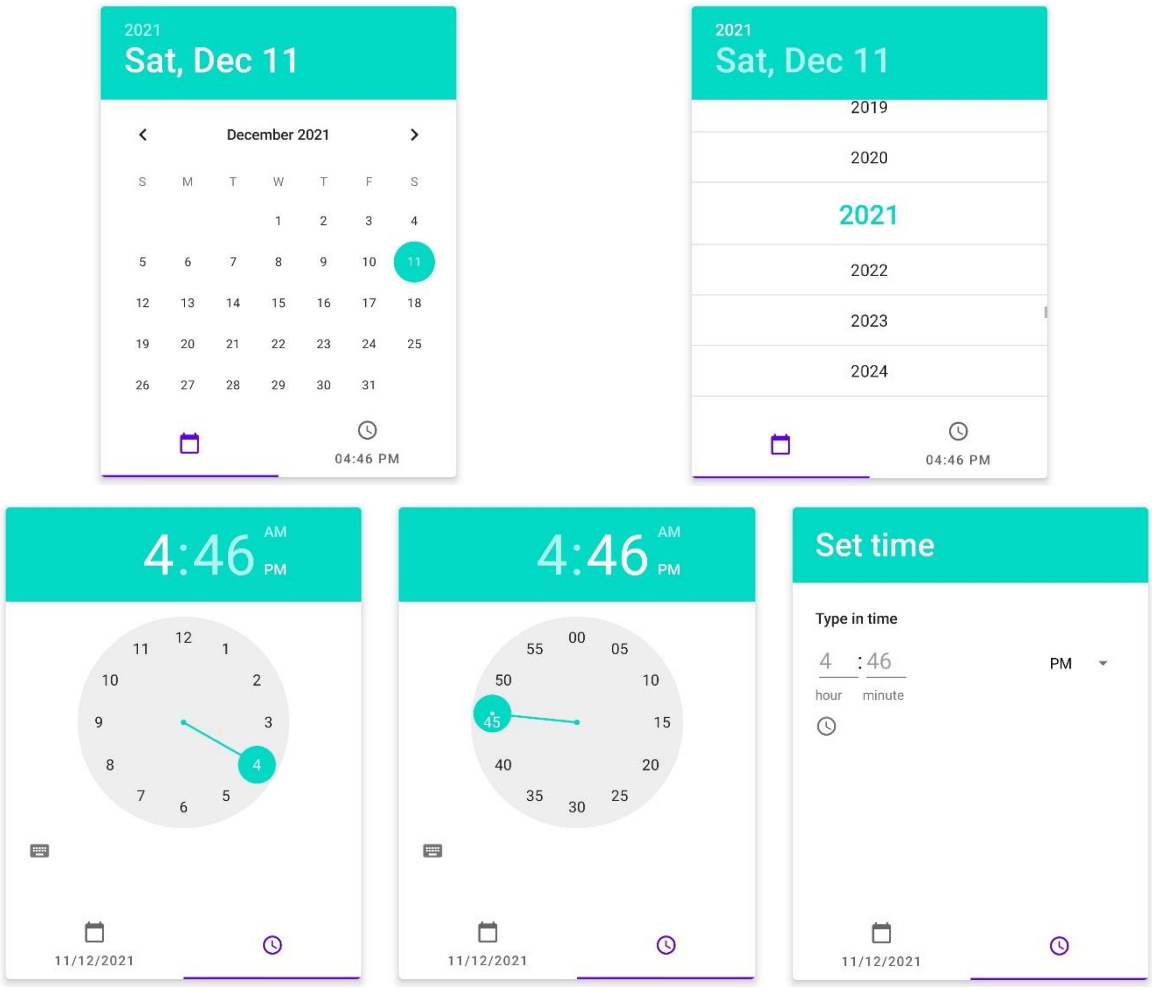


Εικόνα 4.30: Κάρτα συμμετεχόντων

- Αντικείμενα: Αυτή η κάρτα μοιάζει με τη λίστα αντικειμένων που παρουσιάστηκε στην πρώτη οθόνη. Πανομοίως, εμφανίζει συνοπτικά τα αντικείμενα και τους χρήστες με το μεγαλύτερο μερίδιο, ενώ μπορούν να βγουν και συμπεράσματα για το πλήθος των συμμετεχόντων τους. Για περισσότερες λεπτομέρειες σχετικά με τα αντικείμενα ο χρήστης θα πρέπει να πλοηγηθεί στην πρώτη οθόνη, ώστε να επιλέξει το αντικείμενο που επιθυμεί και να το επιθεωρήσει.

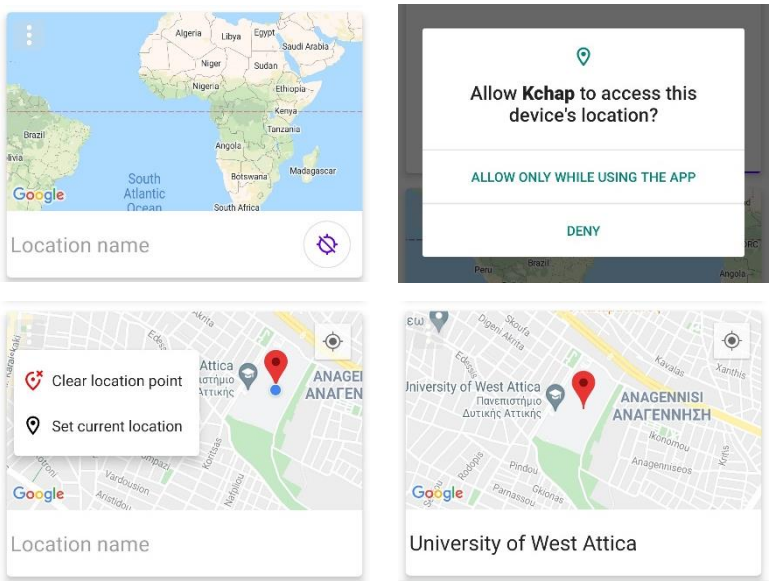
Εικόνα 4.31: Κάρτα αντικειμένων

- Ημερομηνία και ώρα: Ο χρήστης μπορεί να καθορίζει την ημερομηνία και την ώρα που πραγματοποιήθηκε η συναλλαγή. Η κάρτα περιλαμβάνει δύο οθόνες, η πρώτη είναι το ημερολόγιο και η δεύτερη το ρολόι, όπου μπορεί να επιλέξει τις τιμές που επιθυμεί. Η πλοήγηση στις οθόνες γίνεται είτε με χειρονομίες συρσίματος, είτε επιλέγοντας της πατώντας το αντίστοιχο κουμπί. Η προεπιλεγμένη τιμή είναι η ημερομηνία και ώρα που εκκίνησε η διαδικασία της δημιουργίας της συναλλαγής.



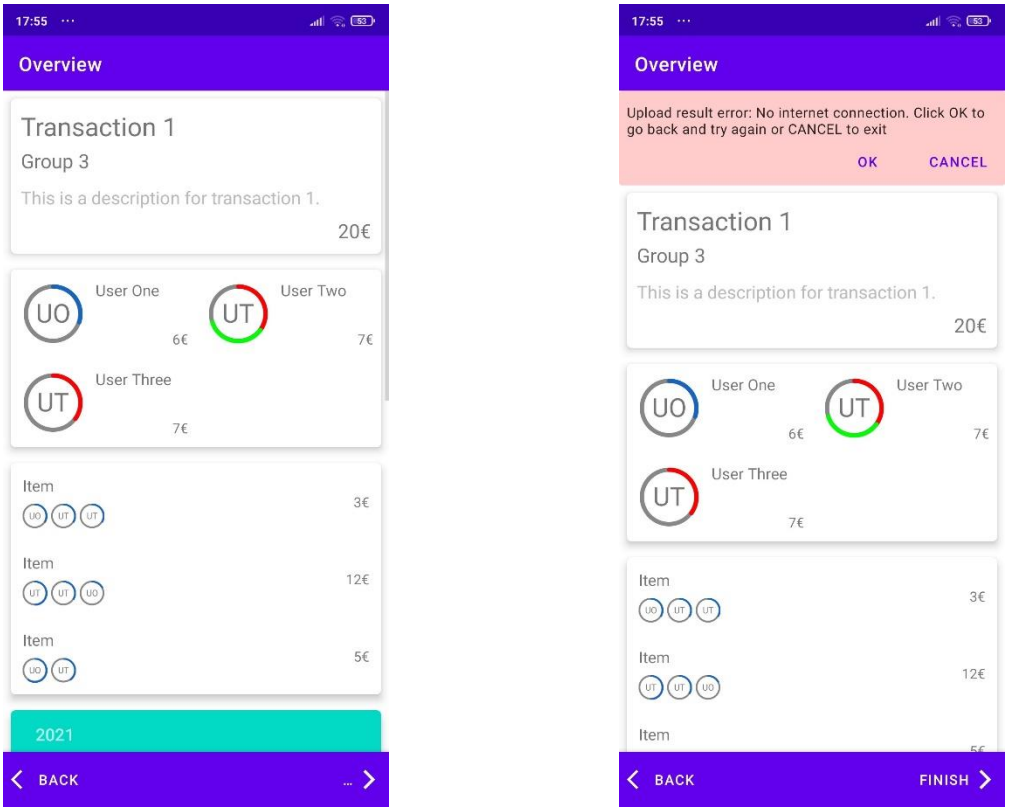
Εικόνα 4.32: Η κάρτα ημερομηνίας και ώρας με όλες τις δυνατές επιλογές εισαγωγής δεδομένων

- Τοποθεσία: Προαιρετικά, ο χρήστης μπορεί να προσθέσει την τοποθεσία που έλαβε μέρος η συναλλαγή. Η κάρτα τοποθεσίας, του δίνει τη δυνατότητα, πατώντας το αντίστοιχο κουμπί, να βρεθεί η θέση του στο χάρτη, αρκεί να δοθεί πρώτα το κατάλληλο δικαίωμα για πρόσβαση στην τοποθεσία της συσκευής. Στη συνέχεια από το μενού μπορεί να θέσει την τωρινή θέση ως την επιλεγμένη, αλλιώς μπορεί να επιλέξει οπουδήποτε στο χάρτη πατώντας παρατεταμένα στο σημείο που επιθυμεί. Αν θέλει να αφαιρέσει το σημείο, μπορεί να το πράξει μέσω του μενού. Η συμπλήρωση του ονόματος της τοποθεσίας/καταστήματος αποτελεί επιλογή του χρήστη.



Εικόνα 4.33: Κάρτα τοποθεσίας: εμφάνιση χάρτη χωρίς δικαίωμα τοποθεσίας (Αριστερά-Πάνω), αίτηση δικαιώματος (Δεξιά-Πάνω), επιλογή τωρινής τοποθεσίας και εμφάνιση μενού (Αριστερά-Κάτω) και εισαγωγή ονόματος τοποθεσίας από τον χρήστη (Δεξιά-Κάτω)

Αν ο χρήστης είναι ικανοποιημένος από τη σύνοψη, τότε μπορεί να προχωρήσει στην αποστολή της συναλλαγής στον εξυπηρετητή. Αν η διαδικασία είναι επιτυχής, τότε ο χρήστης πλοηγείται πίσω στην οθόνη απ' όπου εκκίνησε τη δημιουργία ή ενημέρωση της συναλλαγής. Παρόλα αυτά αν η διαδικασία απέτυχε, τότε εμφανίζεται μήνυμα σφάλματος στην κορυφή της οθόνης ενημερώνοντας τον για τον λόγο.



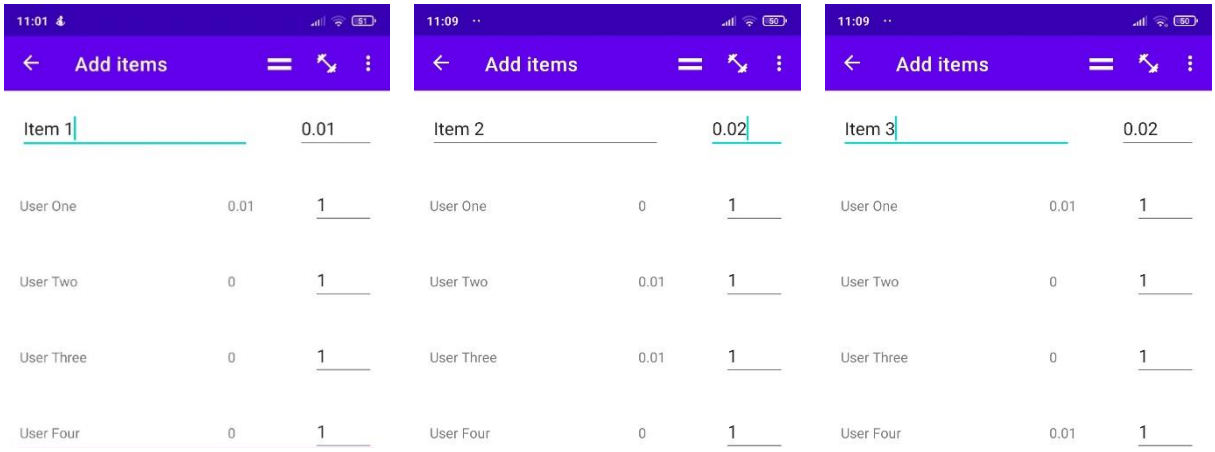
Εικόνα 4.34: Αλλαγή εμφάνισης του κουμπιού της ολοκλήρωσης ώστε να δηλώσει ενεργή αποστολή (Αριστερά), Εμφάνιση μηνύματος σφάλματος αποστολής (Δεξιά)

Παράδειγμα συναλλαγής με μη πληρωτέο ποσό

Έχει πλείστος αναφερθεί η πιθανότητα ο διαμοιρασμός χρέους να οδηγήσει σε μη πληρωτέα ποσά. Τέτοια θεωρούνται τα ποσά, τα δεκαδικά ψηφία των οποίων είναι πάνω από δύο ή άπειρα, όπως είναι τα επαναλαμβανόμενα δεκαδικά ψηφία. Σκοπός της εφαρμογής είναι να μπορεί να διαχειρίζεται αντίστοιχες καταστάσεις, πάντα με όσο το δυνατόν πιο δίκαιο τρόπο. Παρακάτω παρουσιάζεται η εισαγωγή αντικειμένων που εμπίπτουν σε αυτή την περίπτωση. Η ομάδα είναι νέα, δίχως προηγούμενο Περιεχόμενο και με συνολικά τέσσερα μέλη. Τα αντικείμενα που προθέτονται είναι:

Α/Α	Όνομα	Αξία	Τρόπος κατανομής
1	Item 1	0,01€	Βάρη, κατανεμημένα ισομερώς σε όλα τα μέλη
2	Item 2	0,02€	
3	Item 3	0,02€	

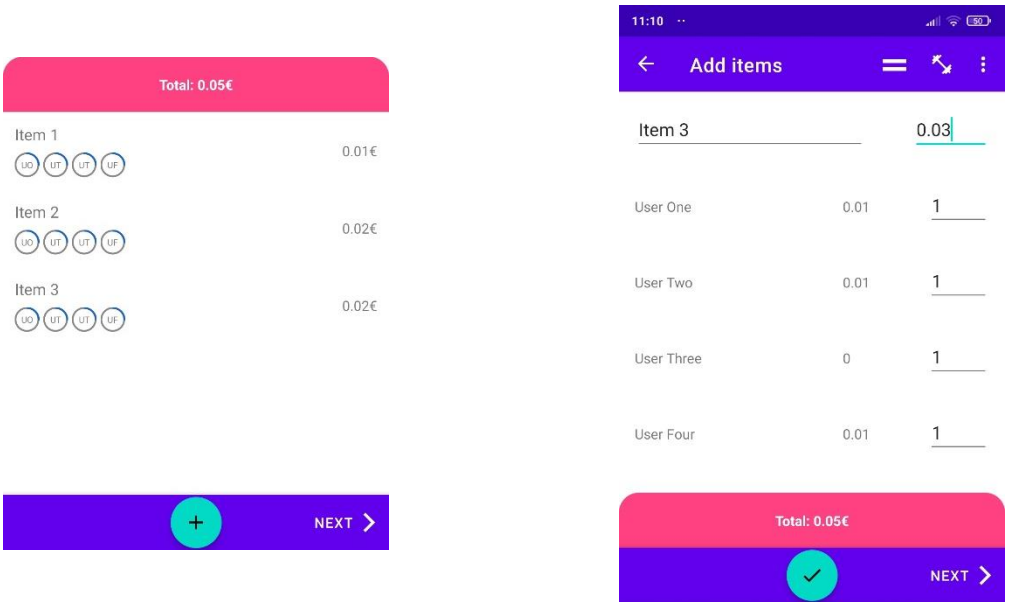
Πίνακας 4.1: Αντικείμενα παραδείγματος που παράγουν μη πληρωτέα ποσά



Εικόνα 4.35: Προσθήκη τριών αντικειμένων, διαχωρισμένων ισομερώς στα μέλη μιας ομάδας, δίχως όμως τη δυνατότητα πληρωμής από όλα τα μέλη (μη πληρωτέα μερίδια)

Από τον πίνακα 4.1 και τις παραπάνω εικόνες είναι εμφανές ότι η αξία κάθε αντικειμένου δεν είναι δυνατή να πληρωθεί από κάθε συμμετέχοντα. Για το λόγο αυτό, η εφαρμογή επιλέγει κάθε φορά όσα άτομα χρειάζονται ώστε να συμπληρωθεί η αξία του αντικειμένου. Εν συνεχεία ενημερώνει τα δευτερεύοντα υπόλοιπα και με βάση τα ανανεωμένα υπόλοιπα επιλέγονται οι χρήστες που θα κληθούν να επαναλάβουν την ίδια διαδικασία σε περίπτωση αντίστοιχου συμβάντος. Αυτό φαίνεται και στην προσθήκη των δύο επόμενων αντικειμένων. Στο δεύτερο αντικείμενο πληρώνουν μέλη που δεν είχαν επιλεχθεί προηγουμένως, ενώ στο τρίτο αντικείμενο επιλέγεται ξανά ο πρώτος χρήστης, καθώς τα υπόλοιπα μέλη είχαν επιλεχθεί και η τιμή του υπολοίπου τους δεν τους έθεσε ως έγκυρη επιλογή από τον αλγόριθμο. Σημειώνεται, ότι η επιλογή γίνεται μόνο βάσει των υπολοίπων, δεν υπάρχει κάποιος αλγόριθμος κυκλικής επιλογής, καθώς το ποσό που θα κληθούν να πληρώσουν οι χρήστες είναι μεταβλητό. Περισσότερα για τον μηχανισμό κατανομής της αξίας του αντικειμένου, ενημέρωσης των υπολοίπων και επιλογής χρηστών θα γίνει στο κεφάλαιο ανάλυσης των αλγορίθμων. Στην επόμενη εικόνα, μέσω των κύκλων προόδου, φαίνεται ότι παρόλο που για την αξία του αντικειμένου πληρώνουν μέρος των ατόμων που συμμετέχουν στην αγορά του, τα μερίδια είναι χωρισμένα ορθά με το ποσοστό που αντιστοιχεί στο κάθε άτομο. Επίσης, παρουσιάζεται η περίπτωση που γίνεται ανανέωση αντικειμένου και η νέα του κατάσταση εμπίπτει πάλι στην περίπτωση του μη πληρωτέου ποσού, αλλά απαιτώντας επιπλέον άτομα. Συγκεκριμένα, ανανεώνεται το τρίτο αντικείμενο με αυξημένη την τιμή του κατά ένα λεπτό, ώστε να χρειαστεί να επιλεχθούν περισσότερα μέλη για την αποπληρωμή

του. Παρομοίως, η κλάση ContentProcessor επιλέγει δίκαια τα μέλη που θα πληρώσουν βάσει των (δευτερευόντων) υπολοίπων τους.



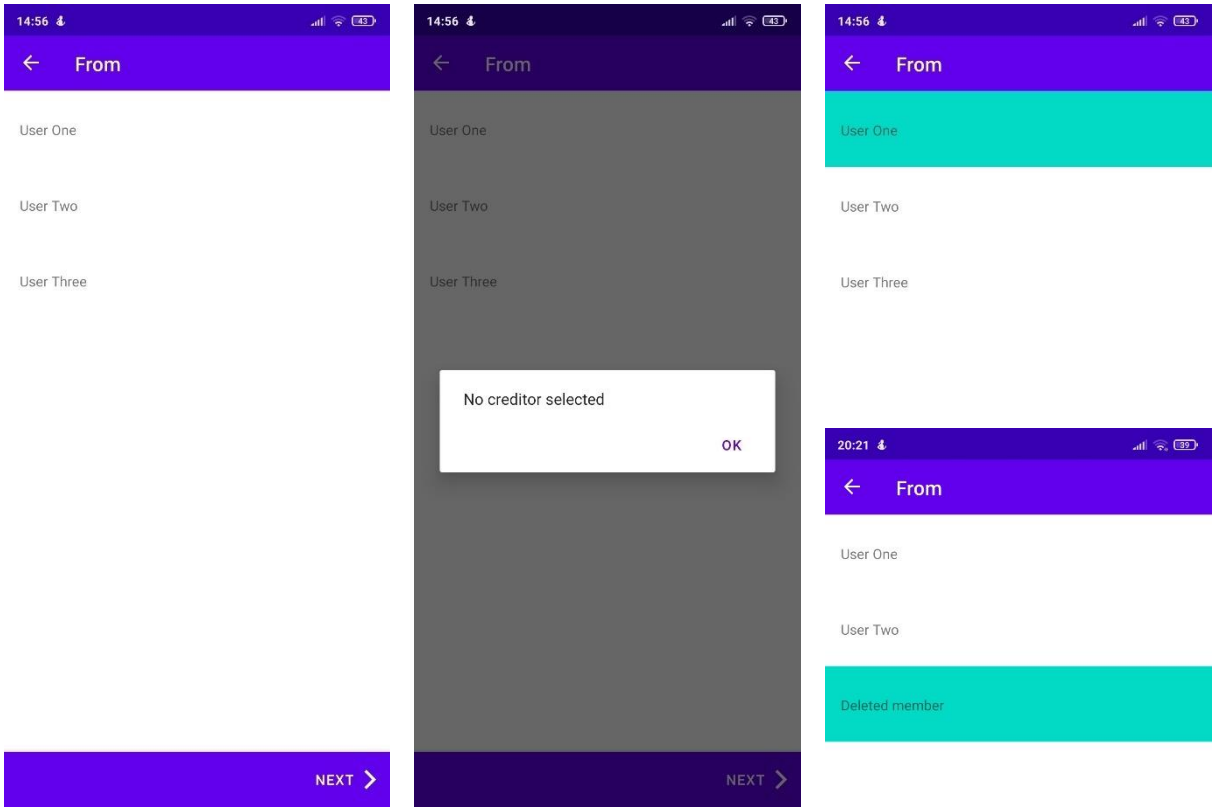
Εικόνα 4.36: Εμφάνιση των ισομερώς χωρισμένων αντικειμένων της συναλλαγής (Αριστερά) και ανανέωση αξίας αντικειμένου που παράγει μη πληρωτέα ποσά (Δεξιά)

Μεταφορές

Πρώτη οθόνη: Επιλογή εντολέα

Στην πρώτη οθόνη της δημιουργίας ή ενημέρωσης μεταφοράς επιλέγεται ο εντολέας, δηλαδή το άτομο που παραχωρεί τα χρήματα. Αρχικά εμφανίζονται όλα τα μέλη της ομάδας και η επιλογή κάποιου μέλους το θέτει ως εντολέα. Το επιλεγμένο μέλος διακρίνεται από την αλλαγμένη εμφάνιση της εγγραφής. Ο μόνος έλεγχος ορθότητας που εκτελείται για τη μετάβαση στην επόμενη οθόνη είναι η επαλήθευση ότι έχει επιλεχθεί κάποιο μέλος ως εντολέας.

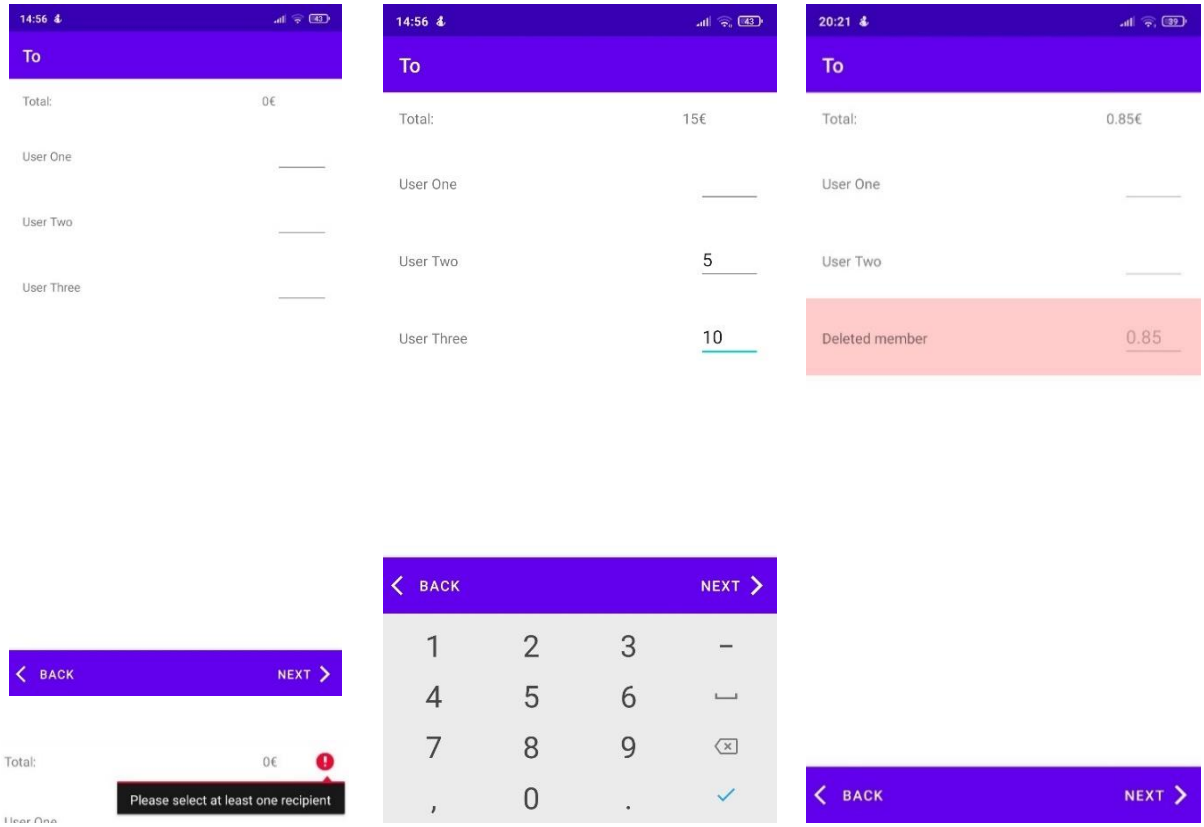
Σε περίπτωση ανανέωσης μεταφοράς στην οποία ο εντολέας είναι διαγραμμένο μέλος, τότε εμφανίζεται η λίστα των μελών με επιλεγμένο το διαγραμμένο μέλος-εντολέα, αλλά χωρίς την δυνατότητα μεταβολής της επιλογής. Η επιλογή παραμένει αμετάβλητη ακόμα και αν ο εντολέας είναι ενεργό μέλος, αλλά ο δικαιούχος είναι διαγραμμένο. Αυτό συμβαίνει, επειδή όπως ειπώθηκε, δεν επιτρέπεται η μεταβολή των υπολοίπων σε Περιεχόμενο που περιλαμβάνει διαγραμμένα μέλη.



Εικόνα 4.37: Επιλογή εντολέα μεταφοράς: Λίστα μελών ομάδας (Αριστερά), Μήνυμα σφάλματος μη επιλεγμένου εντολέα (Μέση), Επιλογή εντολέα και αλλαγή της μορφής της εγγραφής (Δεξιά-Πάνω), Προεπιλεγμένος διαγραμμένος εντολέας (Δεξιά-Κάτω)

Δεύτερη οθόνη: Επιλογή δικαιούχων

Η δεύτερη οθόνη μοιάζει αρκετά με τη δεύτερη οθόνη των συναλλαγών. Όπως και εκεί, εμφανίζεται μια λίστα με τα μέλη της ομάδας ως τις εγγραφές της, ενώ κάθε μέλος διαθέτει ένα πεδίο που συμπληρώνει ο χρήστης με το πόσο που μετέφερε ο εντολέας σε εκείνο. Κάθε εγγραφή χρήστη με τιμή (πεδίου) διάφορη του κενού ή του μηδέν, αποτελεί δικαιούχο. Η μόνη διαφορά που συναντάται σε σχέση με την οθόνη των συναλλαγών είναι η απουσία του πεδίου του ονόματος και ένα ελαφρώς (οπτικά) αλλαγμένο πεδίο συνολικής αξίας. Η παράλειψη του πεδίου ονόματος βασίζεται στο ότι η ουσία των μεταφορών βρίσκεται στα συμβαλλόμενα μέρη (εντολέας-δικαιούχος) και όχι σε κάποιο τίτλο. Ο έλεγχος ορθότητας που συναντάται εδώ είναι η αναγκαστική επιλογή τουλάχιστον ενός δικαιούχου. Όπως και πριν, αν ανανεώνεται μια μεταφορά στην οποία είτε ο δικαιούχος, είτε ο εντολέας είναι διαγραμμένο μέλος, τότε δεν επιτρέπεται η μεταβολή των τιμών της λίστας και απενεργοποιείται η εισαγωγή σε όλες τις εγγραφές.

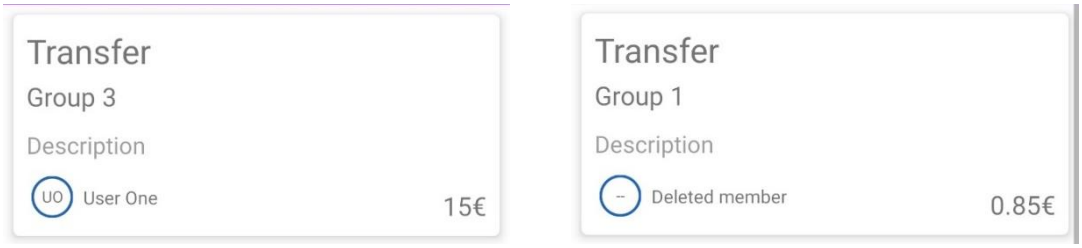


Εικόνα 4.38: Επιλογή δικαιούχων μεταφοράς: Λίστα μελών ομάδας (Αριστερά-Πάνω), Μήνυμα σφάλματος μη επιλεγμένου δικαιούχου (Αριστερά-Κάτω), Συμπλήρωση ποσών που μεταφέρονται στους δικαιούχους (Μέση), Απενεργοποιημένη εγγραφή λόγω διαγραφής (Δεξιά)

Τρίτη οθόνη: Επισκόπηση μεταφοράς

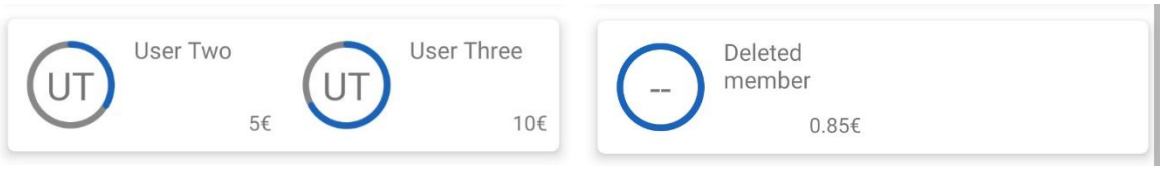
Όσον αφορά τις μεταφορές, η κοινή οθόνη της Επισκόπησης περιέχει τις κάρτες:

- Κεντρική: Οι μόνες διαφορές με την αντίστοιχη κάρτα των συναλλαγών είναι ότι το (αμετάβλητο) όνομα είναι ήδη συμπληρωμένο και εμφανίζεται και ο εντολέας.



Εικόνα 4.39: Κεντρική κάρτα με τον εντολέα: Ως ενεργό μέλος (Αριστερά), ως διαγραμμένο μέλος (Δεξιά)

- Δικαιούχοι: Περιλαμβάνει τους παραλήπτες της μεταφοράς. Κάθε εγγραφή περιλαμβάνει ένα κύκλο προόδου, όπου η τιμή της προόδου καθορίζεται από το ποσοστό του κάθε δικαιούχου ως προς το συνολικό ποσό της μεταφοράς.

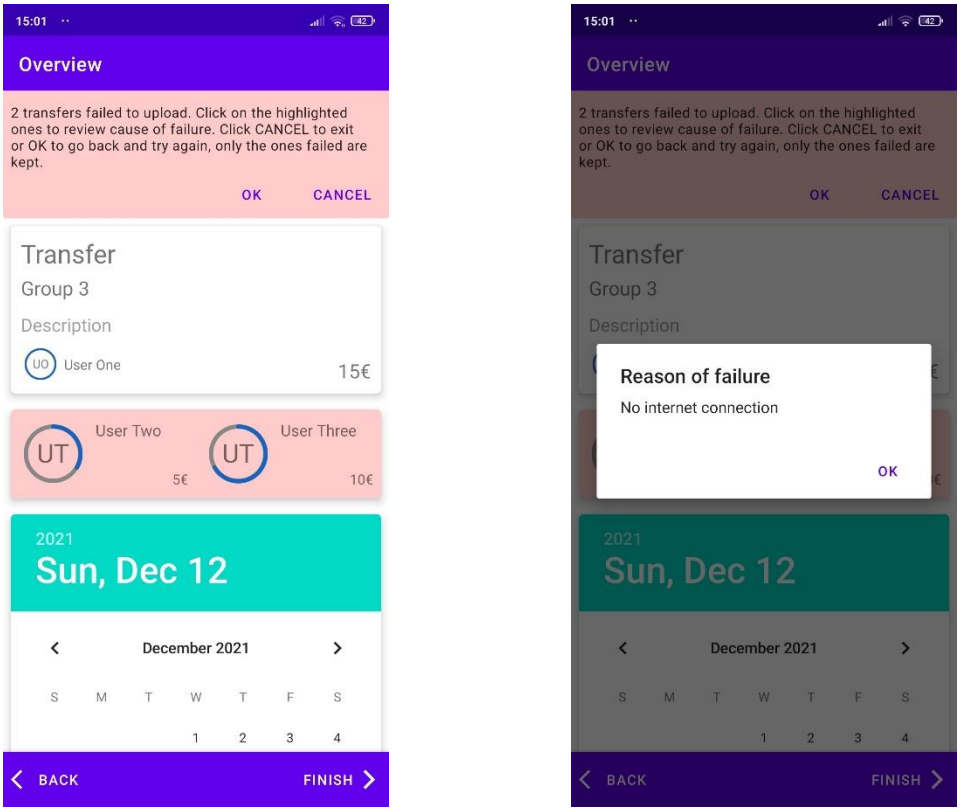


Εικόνα 4.40: Κάρτα δικαιούχων: Δημιουργία με πολλαπλά ενεργά μέλη (Αριστερά), Ανανέωση με διαγραμμένο μέλος (Δεξιά)

- Ημερομηνία/ώρα και τοποθεσία: Οι λειτουργίες τους παραμένουν ίδιες με την περίπτωση των συναλλαγών.

Σημειώνεται ότι μια μεταφορά έχει ένα εντολέα και ένα δικαιούχο. Κατά την ενημέρωση επιτρέπεται η αλλαγή τους αν δεν συναντώνται οι συνθήκες που αναφέρθηκαν ανωτέρω. Ανεξαρτήτως αν υπάρχουν ή δεν υπάρχουν διαγραμμένα μέλη, η ενημέρωση επιτρέπει τη μεταβολή των λοιπών πληροφοριών, δηλαδή τις περιγραφή, ημερομηνία, ώρα και τοποθεσία.

Παρόλα αυτά, από την παραπάνω διαδικασία είναι εμφανές ότι μπορούν να επιλεχθούν πολλοί δικαιούχοι. Αυτό μεταφράζεται σε αντίστοιχο αριθμό μεταφορών και επιτρέπεται μόνο κατά τη δημιουργία. Όταν από την οθόνη της σύνοψης, ο χρήστης πατήσει το κουμπί της αποστολής, θα σταλούν στον εξυπηρετητή τόσα αιτήματα, όσα είναι και ο αριθμός των δικαιούχων. Σε περίπτωση που αποτύχει κάποιο από αυτά, τότε η εγγραφή του παραλήπτη εμφανίζεται με αλλαγμένη μορφή σε συνδυασμό με εμφάνιση μηνύματος σφάλματος στην κορυφή της οθόνης. Πάτημα πάνω στην αποτυχημένη εγγραφή εμφανίζει το λόγο αποτυχίας. Δίνεται η ευκαιρία στον χρήστη να κάνει οποιαδήποτε αλλαγή και να ξαναπροσπαθήσει, ενώ εκτελώντας πλοήγηση προς τα πίσω θα δει (έχει ενημερωθεί προηγουμένως και από το μήνυμα σφάλματος) ότι έχουν μείνει μόνο οι αποτυχημένες μεταφορές.



Εικόνα 4.41: Εμφάνιση μηνύματος σφάλματος αποστολής μεταφορών (Αριστερά) και λόγος αποτυχίας της πρώτης μεταφοράς (Δεξιά)

Προβολή Περιεχομένου και διαγραφή

Πατώντας πάνω σε εγγραφές Περιεχομένου που βρίσκονται στις αντίστοιχες καρτέλες στην οθόνη της ομάδας, ο χρήστης μπορεί να δει τη σύνοψη. Χρησιμοποιείται η οθόνη της Επισκόπησης με τη διαφορά ότι επιτρέπεται μόνο η προβολή, απενεργοποιώντας την εισαγωγή δεδομένων από το χρήστη. Η μεταβολή των δεδομένων γίνεται πατώντας το κουμπί της επεξεργασίας και ακολουθώντας τα βήματα που αναφέρθηκαν προηγουμένως. Τέλος, με παρατεταμένο πάτημα στην εγγραφή εμφανίζεται το μενού επιλογών με την ενέργεια της διαγραφής.

Εικόνα 4.42: Μενού επιλογών Περιεχομένου (Συναλλαγής) με την ενέργεια της διαγραφής

ΚΕΦΑΛΑΙΟ 5

ΑΛΓΟΡΙΘΜΟΙ

Η εφαρμογή κάνει χρήση αλγορίθμων για τον υπολογισμό των ροών χρέους, της κατάτμησης ποσών, όπως ο διαμοιρασμός του κόστους ενός αντικειμένου, καθώς και για την εύρεση ποσών που αντιστοιχούν σε πραγματικά χρήματα και προκύπτουν από τα δευτερεύοντα υπόλοιπα των μελών.

5.1 Άπληστος αλγόριθμος

Ο πρώτος και πιο απλός αλγόριθμος που χρησιμοποιείται είναι ένας άπληστος αλγόριθμος (Greedy algorithm). Η υλοποίηση βασίστηκε στην αρχική εκδοχή του αλγορίθμου, όπως παρουσιάζεται στο άρθρο του Gaurav. (Gaurav, 2022)

Ο αλγόριθμος υπολογίζει τις ροές χρέους σε N-1 συναλλαγές, όπου N ο αριθμός των μελών μιας ομάδας, ενώ η πολυπλοκότητά του είναι $O(N^2)$. Ο αλγόριθμος χαρακτηρίζεται από την επαναλαμβανόμενη εκτέλεση ενός βήματος (αναδρομικότητα), το οποίο είναι η εύρεση του μέγιστου πιστωτή (μέγιστο θετικό υπόλοιπο) και χρεώστη (ελάχιστο αρνητικό υπόλοιπο). Μεταξύ των δύο χρηστών, αναζητείται το (πρωτεύων) υπόλοιπο με τη μικρότερη απόλυτη τιμή και ορίζεται ως το ποσό της ροής χρέους που θα δημιουργηθεί μεταξύ τους. Στην περίπτωση του πιστωτή, το επιλεγμένο ποσό αφαιρείται από το υπόλοιπό του, καθώς πρόκειται για επιστροφή χρημάτων, ενώ στον χρεώστη προστίθεται, αφού μειώνεται το χρέος του. Με ανανεωμένα τα υπόλοιπα, εκτελείται εκ νέου επανάληψη για την ρύθμιση όλων των χρεών μέχρις ότου όλα τα υπόλοιπα των μελών να εκμηδενιστούν. Στην αντίθετη περίπτωση, τα δεδομένα θεωρούνται εσφαλμένα και η εκτέλεση αποτυγχάνει.

Ο ίδιος αλγόριθμος χρησιμοποιείται και για την εύρεση πληρωτέων ποσών βάσει των δευτερευόντων υπολοίπων. Η μόνη διαφορά σε αυτή την περίπτωση είναι ότι δεν απαιτείται ο τερματισμός του αλγορίθμου να προκαλεί και τον μηδενισμό των (δευτερευόντων) υπολοίπων. Επομένως, η λειτουργία του βαίνει όπως αναφέρθηκε παραπάνω έως ότου τα δεδομένα να μην επιτρέπουν την περαιτέρω εκτέλεσή του.

Παρακάτω παρατίθεται ο ψευδοκώδικας του αλγορίθμου, ενώ η υλοποίησή του σε γλώσσα Java τοποθετείται στο παράρτημα Α.

```
1:  procedure calculateRec(members, zeroEquilibrium)
2:      solutions ← [ ]
3:      mxCredit ← getMax(members)
4:      mxDebit ← getMin(members)
5:      if (zeroEquilibrium and mxCredit.balance = 0 and mxDebit.balance = 0) or
        (not zeroEquilibrium and (mxDebit.balance ≥ 0 or mxCredit.balance ≤ 0))
        then
6:          return [ ]
7:      minAmount ← min(abs(mxDebit.balance), mxCredit.balance)
8:      mxCredit.balance ← mxCredit.balance - minAmount
9:      mxDebit.balance ← mxDebit.balance + minAmount
10:     solutions ← solutions + DebtFlow(mxDebit, minAmount, mxCredit)
11:     solutions ← solutions ∪ calculateRec(members, zeroEquilibrium)
12:     return solutions
```

Ψευδοκώδικας άπληστου αλγόριθμου

5.2 Εκθετικός αλγόριθμος

Παρόλα αυτά, η παραπάνω προσέγγιση δεν ικανοποιεί τη συνθήκη βέλτιστης λύσης που ζητήθηκε από τις προϋποθέσεις, καθώς εύκολα μπορούν να υπάρξουν περιπτώσεις που αυτός ο αλγόριθμος δημιουργεί περίσσιες ροές. Για παράδειγμα αν είχαμε τα παρακάτω μέλη και τα υπόλοιπά τους:

A	B	Γ	Δ	E	Z
-2	-3	5	-4	-6	10

Πίνακας 5.1: Υπόλοιπα μελών παραδείγματος

Ο άπληστος αλγόριθμος θα εξάγει τις ακόλουθες ροές:

$E \overset{6}{\rightarrow} Z$	$\Delta \overset{4}{\rightarrow} \Gamma$	$B \overset{3}{\rightarrow} Z$	$A \overset{1}{\rightarrow} \Gamma$	$A \overset{1}{\rightarrow} Z$

Πίνακας 5.2: Ροές χρέους που παράγονται από τον άπληστο αλγόριθμο

Εύκολα κάποιος μπορεί να συμπεράνει ότι το βέλτιστο αποτέλεσμα θα ήταν:

$A \overset{2}{\rightarrow} \Gamma$	$B \overset{3}{\rightarrow} \Gamma$	$\Delta \overset{4}{\rightarrow} Z$	$E \overset{6}{\rightarrow} Z$

Πίνακας 5.3: Βέλτιστες ροές χρέους

Στο παραπάνω παράδειγμα είναι εμφανές ότι υπάρχουν ασύνδετα υποσύνολα μηδενικού αθροίσματος, δηλαδή σύνολα, τα στοιχεία των οποίων αν αθροιστούν έχουν ως αποτέλεσμα το μηδέν. Η εκτέλεση του άπληστου αλγορίθμου σε αυτά τα υποσύνολα θα είχε ως αποτέλεσμα τα δεδομένα του πίνακα 5.3. Η παρατήρηση ότι εκείνα τα υποσύνολα αποτελούν τη βάση για βέλτιστα αποτελέσματα επαληθεύεται και από άλλες πηγές προσκείμενες σε αυτό το θέμα. Μια τέτοια πηγή αποτελεί το άρθρο των Csaba Păţcaş και Attila Bartha, όπου αναφέρεται ότι «Παρατηρούμε ότι η εύρεση του ελάχιστου γραφήματος συναλλαγών είναι ίσο με το διαχωρισμό του V στο μέγιστο αριθμό υποσυνόλων μηδενικού αθροίσματος», όπου V είναι οι κορυφές του γραφήματος, δηλαδή οι οντότητες των συναλλαγών, όπως είναι τα μέλη μιας ομάδας. Παρομοίως, στη διπλωματική εργασία του Yuanfan Yao, αποδεικνύεται η παρατήρηση ότι «σε όσα περισσότερα σύνολα μηδενικού αθροίσματος γίνεται να διαχωριστεί το σύνολο X, τόσο λιγότερες συναλλαγές χρειάζονται», όπου X είναι το σύνολο των υπολοίπων των οντοτήτων. Από τα παραπάνω είναι εμφανής η αλληλοσύνδεση μεταξύ της βέλτιστης εύρεσης ροών χρέους, δηλαδή της ελαχιστοποίησης αυτών, με την εύρεση του μέγιστου αριθμού ασύνδετων υποσυνόλων μηδενικού αθροίσματος. (Păţcaş & Bartha, 2019) (Yao, 2017)

Την παραπάνω λογική υλοποιεί ο αλγόριθμος εκθετικού χρόνου που περιγράφεται στην προαναφερθείσα διπλωματική εργασία. Αρχικά θα πρέπει να βρεθούν και να αφαιρεθούν από το αρχικό σύνολο, όλα τα υποσύνολα μηδενικού αθροίσματος με μέγεθος δύο. Εκείνα θα διαθέτουν δύο υπόλοιπα, των οποίων η απόλυτη τιμή τους είναι ίση, αλλά το πρόσημό τους είναι αντίθετο. Εν συνεχεία, τα εναπομείναντα υπόλοιπα m, θα αποτελέσουν την είσοδο του εκθετικού αλγορίθμου. Αυτός θα προσπαθήσει να βρει το σύνολο που εμπεριέχει τα περισσότερα υποσύνολα μηδενικού αθροίσματος που παράχθηκαν από τα εναπομείναντα υπόλοιπα. Αυτό θα γίνει βρίσκοντας κάθε υποσύνολο s, το μήκος του οποίου είναι μεταξύ τρία και του μισού του πλήθους m (m/2), και ελέγχοντας αν το άθροισμά του ισούται με μηδέν. Αν ισχύει αυτή η συνθήκη, τότε τα στοιχεία του υποσυνόλου αφαιρούνται από το αρχικό σύνολο και μέσω αναδρομής ελέγχονται τα στοιχεία που παρέμειναν. Αυτή η

διαδικασία εκτελείται διαρκώς έως ότου οι συνδυασμοί των στοιχείων για την παραγωγή των υποσυνόλων εξαντληθούν. Στο τέλος κάθε αναδρομής επιλέγεται το σύνολο με τα περισσότερα υποσύνολα (μεγαλύτερο μέγεθος). Η πολυπλοκότητα του αλγορίθμου είναι $O(3^N)$, όπου N το πλήθος των μελών.

Παρομοίως, παραθέτονται ο ψευδοκώδικας, όπως περιγράφηκε από τον συγγραφέα Yuanfan Yao. Η υλοποίηση σε γλώσσα Java μπορεί να βρεθεί στο παράρτημα Β.

```
1:  procedure BestPartition(subset, debts)
2:      subset ← subset of vertices I'm looking at
3:      debts ← array of debt values
4:      return ← the best partitioning of subset
5:  Base Case:
6:      if susbet is empty then return [ ]
7:  Recursive Case:
8:      solutions ← [ ]
9:      for s ⊆ subset and 3 ≤ |s| ≤ len(subset)/2 do
10:         if subset is valid (has sum 0) then
11:             solutions.append ← [s] + BestPartition(subset − s, debts)
12:     return element in solutions with the largest length.
```

Ψευδοκώδικας εκθετικού αλγόριθμου

Υπενθυμίζεται, όπως αναφέρθηκε και στο παράδειγμα, ότι τα μέγιστα υποσύνολα που θα βρει ο εκθετικός αλγόριθμος θα τροφοδοτήσουν τον άπληστο αλγόριθμο για την παράγωγή των βέλτιστων ροών χρέους. Σημειώνεται ότι υπάρχουν και άλλοι αλγόριθμοι που εξυπηρετούν τον ίδιο σκοπό και είναι πιο γρήγοροι, αλλά βασίζονται σε προσεγγιστικές μεθόδους ή/και ευρετικές παραμέτρους και έτσι η συμπεριφορά τους αλλάζει ανάλογα με τα δεδομένα της εισόδου.

Οι αλγόριθμοι εκτελούνται στην εφαρμογή του πελάτη, έτσι ώστε να μην επωμιστεί ο διακομιστής επιπλέον φόρτο που μάλιστα απαιτείται να εκτελεστεί με την παραμικρή αλλαγή σε κάποιο υπόλοιπο μέλους. Παρόλο που ο εκθετικός αλγόριθμος αποτελεί απαιτητική διαδικασία, η εκτέλεση ανά ομάδα (σε παραλληλία)

και ο περιορισμός των ατόμων μιας ομάδας οδηγούν σε ομαλή εκτέλεση και εμπειρία χρήσης.

5.3 Αλγόριθμος υπολογισμού ποσών βάσει κατανομής

Στο κεφάλαιο 4, στην υποενότητα των Συναλλαγών, έγινε αναφορά στη κλάση ContentProcessor και στον αλγόριθμο που εμπεριέχει ώστε να υπολογίζει δίκαια τα μερίδια ενός αντικειμένου μεταξύ των μελών, ακόμα και σε περιπτώσεις μη πληρωτέων ποσών. Όπως και προηγουμένως, οι υπολογισμοί αφορούν λεπτά του ευρώ.

Αρχικά, ελέγχεται ο επιλεγμένος τρόπος κατανομής. Όπως είναι φυσικό, αν είναι επιλεγμένη η μέθοδος των χρήματων, τότε ο χρήστης είναι υπεύθυνος για την απόδοση των ποσών στα μέλη και έτσι ο αλγόριθμος τερματίζει άμεσα. Όπως έχει ήδη αναφερθεί, τα μερίδια μετατρέπονται σε κλάσματα, οπότε επόμενη ενέργεια είναι ο ορισμός τους. Ως αριθμητής θέτεται η τιμή του μεριδίου, ενώ ως κοινός παρονομαστής ορίζεται το άθροισμα του συνόλου των μεριδίων.

$$κλάσμα\ μερίδιου\ χρήστη_{αντικειμένου} = \frac{μερίδιο\ χρήστη_{αντικειμένου}}{\sum μεριδίων_{αντικειμένου}} \qquad (1.1)$$

Αν το άθροισμα των μεριδίων είναι μηδενικό, τότε ως τιμή του παρονομαστή τίθεται η μονάδα. Κάποιος θα μπορούσε να παρατηρήσει ότι αφού το άθροισμα είναι μηδενικό, τότε ίσως θα ήταν πιο αποδοτικό ο αλγόριθμος να τερματίζει, αφού δεν υπάρχουν μερίδια για υπολογισμό. Κάτι τέτοιο δεν ισχύει και η αιτία περιγράφεται παρακάτω. Για κάθε αντικείμενο, υπολογίζεται το ποσό που πληρώνει ο κάθε χρήστης, πολλαπλασιάζοντας την συνολική αξία (σε λεπτά του ευρώ) του αντικειμένου με την τιμή του κλασματικού μεριδίου που του αναλογεί.

$$χρέωση\ χρήστη_{αντικειμένου} = κλάσμα\ μερίδιου\ χρήστη_{αντικειμένου} \times αξία_{\ αντικειμένου} \qquad (1.2)$$

Τέτοιες μαθηματικές πράξεις είναι ευάλωτες σε υπερχείλιση τιμής. Γι' αυτό λαμβάνεται μερίμνα, ώστε αν εντοπιστούν ανάλογες περιπτώσεις, τότε η τιμή του μεριδίου μηδενίζεται και δεν λαμβάνεται υπόψιν. Αυτά τα σφάλματα μπορούν να διακόψουν την κανονική ροή της εκτέλεσης.

Το μη πληρωτέο ποσό ($ep_{\text{αντικειμένου}}$) που ίσως προκύψει διατηρείται ως κλάσμα (ιδιότητα epFraction). Αυτό θα έχει αρνητικό πρόσημο, ώστε να παρουσιαστεί ως χρωστούμενο. Σε αυτό το σημείο της διαδικασίας, απλά αποφασίζεται ποιο μη πληρωτέο ποσό αναλογεί σε κάθε χρήστη (χρέωση) και δεν αναζητούνται τα μέλη που θα το καταβάλουν (πίστωση).

$$epFraction_{\text{αντικειμένου}} = \left(\left(\text{κλάσμα μερίδιου χρήστη}_{\text{αντικειμένου}} \times \text{αξία}_{\text{αντικειμένου}}\right) - \text{χρέωση χρήστη}_{\text{αντικειμένου}}\right) \times (-1) \quad (1.3)$$

Επίσης το μη πληρωτέο ποσό διατηρείται και ως ακέραια τιμή (ιδιότητα ep), στρογγυλοποιώντας το κλάσμα προς τα πάνω (half up) και θέτοντας τον αριθμό των δεκαδικών ψηφίων σε δεκαεπτά (17), ώστε να διατηρηθεί υψηλή ακρίβεια. Παρόλα αυτά, δεν μπορεί να συγκριθεί με την ακρίβεια του κλάσματος. Η προηγούμενη πράξη θα παράγει ένα δεκαδικό αριθμό. Υπενθυμίζεται ότι το ακέραιο μέρος του αριθμού εκφράζει λεπτά του ευρώ, ενώ το δεκαδικό υπομονάδες του λεπτού. Το δεκαδικό μέρος μορφοποιείται κατάλληλα, συμπληρώνοντας (trailing) μηδενικά στο τέλος, ώστε το μήκος του να αποτελείται από δεκαεπτά ψηφία. Εν συνεχεία, ακέραιος και τροποποιημένος δεκαδικός συνενώνονται και το παραγόμενο αλφαριθμητικό μετατρέπεται σε ακέραιο αριθμό. Η μετατροπή διατηρεί το πρόσημο του αρχικού αριθμού, ενώ θα αφαιρέσει και τυχόν αρχικά (leading) μηδενικά από το δεκαδικό κομμάτι όταν ο ακέραιος είναι μηδενικός.

Μετά τους αρχικούς υπολογισμούς, ελέγχεται αν όλα τα μερίδια είναι μηδενικά, τότε και πάλι διακόπτεται η εκτέλεσή του. Παρόμοιος έλεγχος για καθολικά μηδενικά μερίδια αναφέρθηκε και παραπάνω. Ο λόγος που ο έλεγχος δεν γίνεται εκ των προτέρων, είναι επειδή τα μερίδια θα μπορούσαν αρχικά να έχουν συμπληρωθεί και εν συνεχεία να μηδενιστούν όλα. Τα νέα μηδενικά μερίδια θα πρέπει να υπολογιστούν ξανά, ώστε να εμφανιστεί η μηδενική συμμετοχή των χρηστών, δηλαδή ότι δεν πληρώνουν κάποιο ποσό.

Ακολούθως, αθροίζονται τα ποσά που καλούνται να πληρώσουν οι χρήστες για το αντικείμενο. Αν η αξία του αντικειμένου ισούται με το άθροισμα, τότε η κατανομή είναι τέλεια ανάμεσα στα μέλη, δίχως την παρουσία μη πληρωτέου ποσού. Απεναντίας, θα πρέπει να υπολογιστεί το υπόλοιπο της αφαίρεσης του προαναφερθέντος

αθροίσματος από την αξία του αντικειμένου. Το αποτέλεσμα είναι σε λεπτά του ευρώ και υποδεικνύει πόσοι χρήστες απαιτούνται για να συμπληρωθεί, πληρώνοντας από ένα λεπτό ο καθένας. Τα μέλη της ομάδας θα ταξινομηθούν βάσει του ποιος έχει το μικρότερο δευτερεύων υπόλοιπο ($ep_{συναλλαγής}$) και θα επιλεχθούν σειριακά έως ότου συμπληρώσεως του ποσού. Τα επιλεγμένα μέλη θα έχουν αύξηση του ποσού που καλούνται να πληρώσουν κατά ένα λεπτό, ενώ θα ενημερωθεί και το τρέχων μη πληρωτέο ποσό τους για το αντικείμενο, τόσο σε ακέραια, όσο και σε κλασματική μορφή. Για το κλασματικό μη πληρωτέο ποσό ισχύει:

$$(1.3) \xrightarrow{+1\ \lambda\varepsilon\pi\tau\acute{o}} epFraction_{\alpha\nu\tau\iota\kappa\varepsilon\iota\mu\acute{\varepsilon}\nu o\upsilon|\chi\rho\acute{\eta}\sigma\tau\eta} = epFraction_{\alpha\nu\tau\iota\kappa\varepsilon\iota\mu\acute{\varepsilon}\nu o\upsilon|\chi\rho\acute{\eta}\sigma\tau\eta} + 1 \qquad (1.4)$$

Για την ακέραια μορφή του μη πληρωτέου ποσού, αρκεί η άμεση πρόσθεση ενός λεπτού του ευρώ στην αρχική του τιμή:

$$ep_{\alpha\nu\tau\iota\kappa\varepsilon\iota\mu\acute{\varepsilon}\nu o\upsilon|\chi\rho\acute{\eta}\sigma\tau\eta} = ep_{\alpha\nu\tau\iota\kappa\varepsilon\iota\mu\acute{\varepsilon}\nu o\upsilon|\chi\rho\acute{\eta}\sigma\tau\eta} + 10000000000000000(0{,}01€) \qquad (1.5)$$

Τα παραπάνω αποτελέσματα θα μεταφερθούν στις γενικές τιμές της συναλλαγής, μόλις το αντικείμενο προστεθεί σε αυτή. Έτσι σε περίπτωση επανεμφάνισης μη πληρωτέου ποσού στη συναλλαγή θα γίνει ορθή επιλογή μελών. Συγκεκριμένα:

$$ep_{συναλλαγής|\chi\rho\acute{\eta}\sigma\tau\eta} = ep_{συναλλαγής|\chi\rho\acute{\eta}\sigma\tau\eta} + ep_{\alpha\nu\tau\iota\kappa\varepsilon\iota\mu\acute{\varepsilon}\nu o\upsilon|\chi\rho\acute{\eta}\sigma\tau\eta} \qquad (1.6)$$

Υπενθυμίζεται ότι για να παγιωθούν τα υπόλοιπα που προκύπτουν συνολικά από τη συναλλαγή, θα πρέπει αυτή να σταλεί και να γίνει αποδεκτή από τον εξυπηρετητή. Η πολυπλοκότητα του αλγορίθμου είναι O(N*logN), όπου N είναι τα μέλη της συναλλαγής, δηλαδή όλα τα μέλη της ομάδας και πιθανώς κάποια διαγραμμένα. Αυτή προκύπτει τόσο από την ταξινόμηση των μελών (χειρότερη περίπτωση), όσο και από την αναζήτηση των μελών της συναλλαγής βάσει των μεριδίων των μελών που συμμετέχουν στο αντικείμενο. Σημειώνεται ότι ο προεπιλεγμένος αλγόριθμος ταξινόμησης για την Java 8 είναι ο Timsort. Η υλοποίηση του αλγορίθμου σε γλώσσα Kotlin παρατίθεται στο παράρτημα Γ. (Arrays (Java Platform SE 8), n.d.) (Nicolas Auger, Vincent Jugé, Cyril Nicaud, & Carine Pivoteau, 2018)

Παράδειγμα

Με βάση τα παραπάνω, είναι δυνατό να γίνει επιστροφή στο παράδειγμα της υποενότητας Συναλλαγές για να ολοκληρωθεί ο τρόπος υπολογισμού των δευτερευόντων υπολοίπων και της επιλογής των χρηστών.

Τα δευτερεύοντα υπόλοιπα των μελών ανακτώνται από τη βάση δεδομένων με την εκκίνηση της διαδικασίας δημιουργίας μιας συναλλαγής και αποθηκεύονται προσωρινά σε μία λίστα. Αφού δεν έχει προστεθεί κάποια προηγούμενη συναλλαγή, τα στοιχεία της λίστας με τα αρχικά δευτερεύοντα υπόλοιπα της συναλλαγής ($ep_{συναλλαγής}$) θα είναι επίσης μηδενικά:

	User One	User Two	User Three	User Four
Ep	0	0	0	0

Πίνακας 5.4: Αρχική κατάσταση των δευτερευόντων υπολοίπων της συναλλαγής

Το πρώτο αντικείμενο αξίας ενός λεπτού διαμοιράζεται ανάμεσα στα τέσσερα μέλη. Γι' αυτό σε κάθε μέλος αντιστοιχεί:

$$epFraction_{Item\ 1|all\ users} = \left(\left(\frac{1}{4} \times 1\right) - 0\right) \times (-1) = -\frac{1}{4}$$

και $ep_{Item\ 1|all\ users} = -25000000000000000$

Το κλάσμα που προκύπτει είναι σε λεπτά του ευρώ και εκφράζει ότι κάθε ένας χρήστης χρωστάει ένα κομμάτι αυτού του λεπτού (αξία αντικειμένου), ενώ εκφράζεται και με ακέραια δεκαεπταψήφια μορφή. Το ένα λεπτό του ευρώ μπορεί να συμπληρωθεί από ένα μόνο χρήστη, ο οποίος θα επωμιστεί και τις χρεώσεις των υπόλοιπων τριών μελών, οι οποίοι θα του χρωστάνε. Η λίστα ταξινομείται κατά αύξουσα σειρά των δευτερευόντων υπολοίπων. Αφού όλα τα μέλη έχουν μηδενικό δευτερεύων υπόλοιπο, επιλέγεται το πρώτο μέλος της λίστας, ο χρήστης User One. Εκείνος θα πληρώσει το ένα λεπτό και επομένως θα προσθέσει την αξία του (λεπτού) στα υπόλοιπά του, ανανεώνοντας το πληρωτέο και μη πληρωτέο (κλάσμα, ακέραιος) ποσό του για το αντικείμενο.

$$ep_{Item\ 1|User\ One} = (-25000000000000000) + 10000000000000000\ (0{,}01€) =$$
$$= 75000000000000000$$

Τα μη πληρωτέα ποσά των άλλων μελών (για το αντικείμενο) παραμένουν ως έχουν, δηλαδή αρνητικά καθώς δεν κλήθηκαν να πληρώσουν. Η πρόσθεση του αντικειμένου ενημερώνει τα υπόλοιπα της λίστας, μεταφέροντας σε αυτά τα υπολογισμένα μη πληρωτέα ποσά των μελών για το αντικείμενο. Η νέα κατάσταση των δευτερευόντων υπολοίπων των μελών για τη συναλλαγή θα είναι:

$$ep_{συναλλαγής|User\ One} = 0 + 75000000000000000 = 75000000000000000$$

$$ep_{συναλλαγής|all\ users \setminus User\ One} = 0 + (-25000000000000000) = -25000000000000000$$

	User One	User Two	User Three	User Four
Ep	75000000000000000	−25000000000000000	−25000000000000000	−25000000000000000

Πίνακας 5.5: Κατάσταση των δευτερευόντων υπολοίπων της συναλλαγής μετά την πρόσθεση του πρώτου αντικειμένου

Για το δεύτερο αντικείμενο, αξίας δύο λεπτών (0,02€) και διαμοιραζόμενο ανάμεσα σε όλα (4) μέλη, αντιστοιχούν οι τιμές:

$$epFraction_{Item\ 2|all\ users} = -\frac{1}{2} \qquad ep_{Item\ 2|all\ users} = -50000000000000000$$

Για να συμπληρωθεί το ποσό απαιτούνται δύο μέλη, οπότε και επιλέγονται οι δύο πρώτες εγγραφές της επαναταξινομημένης λίστας, δηλαδή οι χρήστες User Two και User Three. Με την πρόσθεση του δεύτερου αντικειμένου, η νέα κατάσταση των δευτερευόντων υπολοίπων των μελών υπολογίζεται ως εξής:

$$ep_{συναλλαγής|User\ Two\ \&\ Three} = -25000000000000000 + (-50000000000000000) + 10000000000000000\ (0{,}01€) = 25000000000000000$$

$$ep_{συναλλαγής|User\ One} = 75000000000000000 + (-50000000000000000) = 25000000000000000$$

$$ep_{συναλλαγής|User\ Four} = -25000000000000000 + (-50000000000000000) = -75000000000000000$$

	User One	User Two	User Three	User Four

Ep	25000000000000000	25000000000000000	25000000000000000	−75000000000000000

Πίνακας 5.6: Κατάσταση των δευτερευόντων υπολοίπων της συναλλαγής μετά την πρόσθεση του δεύτερου αντικειμένου

Παρόμοια διαδικασία εφαρμόζεται και για το τρίτο αντικείμενο, όπως και για κάθε αντικείμενο μιας συναλλαγής.

ΚΕΦΑΛΑΙΟ 6

ΒΕΛΤΙΩΣΗ - ΣΥΜΠΕΡΑΣΜΑΤΑ

6.1 Ιδέες για βελτίωση

Ταυτόχρονη επεξεργασία Περιεχομένου σε πραγματικό χρόνο από πολλαπλούς χρήστες

Η δημιουργία και η επεξεργασία του Περιεχομένου, συνεπώς και των συναλλαγών, είναι δυνατή μόνο από μία συσκευή, ο χρήστης της οποίας επωμίζεται όλο το βάρος αυτής της εργασίας. Κάποιες συναλλαγές μπορεί να περιέχουν πληθώρα αντικειμένων, κάνοντας τη διαδικασία πρόσθεσής τους αρκετά χρονοβόρα, κουραστική και μονότονη. Ένας τρόπος να καταπραϋνθούν αυτές οι συνέπειες είναι να αυξηθούν οι χρήστες που προσθέτουν τα αντικείμενα. Αυτό μπορεί να γίνει με την χρήση WebSockets, όπου κάθε χρήστης, σε πραγματικό χρόνο, προσθέτει αντικείμενα σε μια συναλλαγή, ενώ ταυτόχρονα λαμβάνει τα αντικείμενα που πρόσθεσαν άλλα μέλη της ομάδας. Απόρροια είναι η μείωση του χρόνου που απαιτείται για την ολοκλήρωση μιας συναλλαγής, αλλά και η ενεργή συλλογική συμμετοχή των μελών της ομάδας, χωρίς να επιφορτίζεται μόνο ένα μέλος. Αυτή η προσέγγιση προϋποθέτει την προσωρινή αποθήκευση και επεξεργασία των δεδομένων σε ένα κεντρικό μέρος, όπως στον εξυπηρετητή ή/και στη συσκευή του δημιουργού της Συναλλαγής.

Χρήση κάμερας για ανάγνωση αποδείξεων

Όπως αναφέρθηκε και στην προηγούμενη παράγραφο η ολοκλήρωση μιας συναλλαγής μπορεί να αποτελέσει σύνθετη και χρονοβόρα διαδικασία, λόγω της ποσότητας των αντικειμένων. Πέρα από την παράλληλη πρόσθεση αντικειμένων από πολλούς χρήστες, θα ήταν δυνατή η εκμετάλλευση της μηχανικής όρασης (Machine Vision) ή/και της τεχνητής νοημοσύνης (Artificial Intelligence) για την ανάγνωση των αντικειμένων που αναγράφονται στις αποδείξεις με χρήση της κάμερας του κινητού. Αντίστοιχες τεχνολογίες αποτελούν το Vision AI της Google, καθώς και η μηχανή οπτικής αναγνώρισης χαρακτήρων ανοικτού λογισμικού, Tesseract OCR. Εν συνεχεία ο χρήστης απλά απαιτείται να εισάγει τα μερίδια κάθε μέλους ανά αντικείμενο, καθώς και το ποσό που πλήρωσε για τη συναλλαγή.

Υποστήριξη λειτουργικότητας άνευ διαδικτύου

Η εφαρμογή επιτρέπει στους χρήστες την προβολή του Περιεχομένου χωρίς σύνδεση στο διαδικτύου. Παρόλα αυτά, θα ήταν ωφέλιμο να μπορούν να εκτελούν ακόμα περισσότερες λειτουργίες χωρίς το προαπαιτούμενο της σύνδεσης. Αυτές περιλαμβάνουν την προσθήκη και επεξεργασία του Περιεχομένου και τον μετέπειτα συγχρονισμό τους μόλις η σύνδεση αποκατασταθεί. Ο μηχανισμός του συγχρονισμού, βασιζόμενος στις Αλλαγές, διευκολύνει αρκετά αυτή την επέκταση λειτουργικότητας. Με την απαραίτητη υποδομή, τα νέα δεδομένα μπορούν να αποθηκευτούν τοπικά και όταν καταστεί δυνατό, να προωθηθούν, ώστε εν τέλει να αξιολογηθούν από τον διακομιστή.

Υποστήριξη χαρακτηριστικών τόπων

Με αυτό το χαρακτηριστικό είναι δυνατή η εξατομίκευση της εφαρμογής βάσει του τόπου του χρήστη. Αυτό επιτρέπει την υποστήριξη πολλαπλών νομισμάτων, έχοντας ως προεπιλεγμένο το τοπικό νόμισμα. Επιπλέον, η γλώσσα της εφαρμογής του πελάτη, καθώς και τα μηνύματα του εξυπηρετητή θα πρέπει να είναι μεταφρασμένα στην τοπική γλώσσα. Λόγω της υποστήριξης πολλαπλών νομισμάτων, είναι απαραίτητη η υποδομή μετατροπής τους σύμφωνα με τις ισχύουσες συναλλαγματικές αξίες.

Αναζήτηση

Η βάση του εξυπηρετητή έχει σχεδιαστεί με τρόπο κατάλληλο για να επιτραπούν αιτήματα σχετικά με την ανάκτηση δεδομένων και την παραγωγή στατιστικών με χρήση πολλαπλών φίλτρων. Παρόλα αυτά δεν γίνεται πλήρης εκμετάλλευση των δυνατοτήτων της σχεδίασης, καθώς δεν έχει υλοποιηθεί κάποια λειτουργία αναζήτησης. Εκείνη θα μπορούσε να λαμβάνει μια είσοδο από ένα πεδίο κειμένου, καθώς και κάποιες ετικέτες (φίλτρα) για τον περιορισμό των αποτελεσμάτων. Οι όροι αναζήτησης θα μπορούσαν να είναι γενικοί και να περιλαμβάνουν τον τίτλο ενός αντικειμένου, την περιγραφή μιας συναλλαγής, το ποσό μιας μεταφοράς κ.α.. Η αναζήτηση θα πρέπει να εκτελεστεί στον εξυπηρετητή, απαιτώντας σύνδεση στο διαδίκτυο, καθώς όπως έχει αναφερθεί, ο πελάτης μπορεί να μην διαθέτει όλα τα δεδομένα.

Βελτιωμένος αλγόριθμος με προτεινόμενη επιλογή μελών

Προϋπόθεση της παρούσας διπλωματικής είναι η εύρεση των βέλτιστων ροών χρέους. Παρόλα αυτά κάποια μέλη μπορεί να μην είναι δυνατόν να ξεπληρώσουν το χρέος τους σε κάποια άλλα λόγω αδυναμίας συνάντησής τους. Σε αυτή την περίπτωση θα μπορούσαν να στείλουν τα χρήματα μέσω κάποιου χρηματοπιστωτικού ιδρύματος και να δηλώσουν στην εφαρμογή την οφειλή ως εξοφλημένη. Διαφορετικά, η εφαρμογή θα μπορούσε να τους διευκολύνει θέτοντας κάποιον ενδιάμεσο. Μια σκέψη είναι τα μέλη να ορίζουν σε ποια άλλα μέλη προτιμούν να χρωστούν χρήματα και η εφαρμογή να αναλαμβάνει να διαμοιράζει τα χρέη προσπαθώντας να τηρήσει την ανωτέρω προτίμηση.

6.2 Συμπεράσματα

Εν συμπεράσματι, γίνεται αντιληπτό ότι η κατασκευή εφαρμογών, πόσο μάλλον ενός συστήματος που συντίθεται από περισσότερες (αρμονικά συνεργαζόμενες) εφαρμογές, αποτελεί μια πολυδιάστατη και απαιτητική διαδικασία. Η κατασκευή κάθε εφαρμογής απαιτεί πληθώρα βημάτων ώστε να μελετηθεί, να σχεδιαστεί και εν τέλει να υλοποιηθεί ορθά. Κάθε στάδιο και κάθε τεχνολογία ανάπτυξης (εργαλεία, στοίβα τεχνολογίας κτλ.) απαιτούν από τον προγραμματιστή πνευματική διαύγεια και επένδυση αμέτρητων ωρών εκμάθησης και δουλειάς για να ολοκληρώσει ένα έργο. Ταυτόχρονα η όλη διαδικασία χαρακτηρίζεται από την πολυπραγμοσύνη που αποζητά από εκείνον, καθώς εκτελεί την εργασία πολλαπλών ρόλων (σχεδίαση βάσης δεδομένων, ανάλυση λογισμικού, αρχιτεκτονική, υλοποίηση κ.α.).

Όσον αφορά το θέμα της παρούσας διπλωματικής αποτελεί ένα σύνθετο και πολύπλευρο πρόβλημα που αγγίζει αρκετούς τομείς, από τα μαθηματικά και τους αλγόριθμους μέχρι τα οικονομικά και την εμπειρία χρήσης. Παρόλο που υπάρχουν και άλλες εναλλακτικές επιλογές στην αγορά, η παρούσα προσέγγιση εστιάζει στην ακριβοδικαιοσύνη των συναλλαγών, τόσο με την λεπτομερή καταγραφή των αντικειμένων και των μεριδίων τους, όσο και με την καταμέριση των χρεών σε ειδικές περιπτώσεις, όπως η διαμέριση των μη πληρωτέων ποσών. Έτσι ικανοποιώντας τις αρχικές απαιτήσεις, εκπληρώνει τον σκοπό της και λύνει ένα καίριο πρόβλημα μέσα σε μια παρέα και κάνοντας ευκολότερη την καθημερινότητα των χρηστών της.

Κατά την ανάπτυξη ανέκυψαν διάφορα προβλήματα και δυσκολίες. Εξαιτίας αυτού είχα την ευκαιρία να εφαρμόσω τις γνώσεις μου, να μάθω και να υλοποιήσω

βέλτιστες τεχνικές, να ασχοληθώ με καινούριες τεχνολογίες και εργαλεία και γενικά να διευρύνω τον επαγγελματικό ορίζοντά μου.

ΠΑΡΑΡΤΗΜΑΤΑ

Παράρτημα Α

```
Set<DebtFlow> calculateRec(Set<Member> members, boolean zeroEquilibrium) {
    Set<DebtFlow> debtFlowSet = new HashSet<>();

    Member mxCredit = getMax(members), mxDebit = getMin(members);

    if (zeroEquilibrium && (mxCredit.balance == 0 && mxDebit.balance == 0))
        return new HashSet<>();
    else if (!zeroEquilibrium) {
        if (mxDebit.balance > 0 || mxCredit.balance < 0)
            return new HashSet<>();
        if (mxCredit.balance == 0 || mxDebit.balance == 0)
            return new HashSet<>();
    }

    long min = minOf2(Math.abs(mxDebit.balance), mxCredit.balance);
    mxCredit.balance = Math.subtractExact(mxCredit.balance, min);
    mxDebit.balance = Math.addExact(mxDebit.balance, min);

    debtFlowSet.add(new DebtFlow(mxDebit, min, mxCredit));

    System.out.println("Person " + mxDebit + " pays " + min
            + " to " + "Person " + mxCredit);

    debtFlowSet.addAll(calculateRec(members, zeroEquilibrium));

    return debtFlowSet;
}
```

Εικόνα 1: Υλοποίηση άπληστου αλγόριθμου σε γλώσσα Java

Παράρτημα Β

```
private Set<Set<Member>> bestPartition(Set<Member> members) {
    if (members.size() == 0) return new HashSet<>();
    List<Set<Set<Member>>> solutions = new ArrayList<>();

    for (Set<Member> set: calculateSubsets(members)) {
        if (set.size() >= 3 && set.size() <= members.size() / 2) {
            if (calculateSum(set) == 0) {
                Set<Set<Member>> temp = new HashSet<>();
                Set<Member> tempSubset = new HashSet<>(members);
                tempSubset.removeAll(set);
                temp.add(set);
                temp.addAll(bestPartition(tempSubset));
                solutions.add(temp);
            }
        }
    }

    int maxSize = -1; Set<Set<Member>> res = null;
    for (Set<Set<Member>> set : solutions) {
        if (set.size() > maxSize) {
            maxSize = set.size();
            res = set;
        }
    }
    return (res != null) ? res : new HashSet<>();
}
```

Εικόνα 2: Υλοποίηση εκθετικού αλγόριθμου σε γλώσσα Java

Παράρτημα Γ

```
override fun calculateShares(type: ShareType, valueInPennies: Long, shares:
Map<Long, Share>): MutableList<Share>? {
    val errorsInShares: MutableList<Share> = mutableListOf()
    if (type == ShareType.AMOUNT) return null
    val totalShares: Long = shares.values.sumOf { it.value }
    val denominator = if (totalShares == 0L) 1 else totalShares
    for (share in shares.values) {
        share.valueFraction = SimpleFraction(share.value, denominator)
        try { // Pay
            share.buyer.pay = Math.multiplyExact(valueInPennies,
share.valueFraction.numerator) / share.valueFraction.denominator
        } catch (e: ArithmeticException) {
            share.value = 0; share.valueFraction = SimpleFraction(0)
            share.buyer.pay = 0L; share.buyer.ep = 0L
            share.buyer.epFraction = BigFraction(0)
            errorsInShares.add(share); continue
        }

        // Remaining
        share.buyer.epFraction =
share.valueFraction.toBigFraction().multiply(valueInPennies.toBigInteger())
.subtract(share.buyer.pay.toBigInteger()).multiply(-1)
        share.buyer.ep =
epBigDecimalToLong(share.buyer.epFraction.bigDecimalValue(roundingScale,
roundingMode))

    if (errorsInShares.size > 0) return errorsInShares
    if (shares.count { it.value.value == 0L } == shares.size) return null
    var totalPay: Long = 0
    for (share in shares.values)
        totalPay = Math.addExact(totalPay, share.buyer.pay)

    if (totalPay == valueInPennies) { println("Perfect division") }
    else {
        val totalRemaining: Long = Math.subtractExact(valueInPennies,
totalPay)
        val noOfEpCreditors: Int = totalRemaining.toInt()
        val sortedGroupMembers: List<GroupMember> =
sortGroupMembersByTotalEp(findGroupMembersByShareMembers(members,
shares.filter { it.value.value > 0 }.map { shareEntry ->
shareEntry.value.buyer }))
        for (i in 0 until noOfEpCreditors) { // Creditors: paying EP
            val shareMember: ContentMember =
findShareByBuyerId(sortedGroupMembers[i].id, shares).buyer
            shareMember.ep = Math.addExact(shareMember.ep, ONE_CENT)
            shareMember.epFraction = shareMember.epFraction.add(1)
            shareMember.pay = Math.addExact(shareMember.pay, 1)
        } // Item EP and pay of Debtors have already been filled
    }
    return null
}
```

Εικόνα 3: Υλοποίηση αλγόριθμου υπολογισμού ποσών βάσει κατανομής σε γλώσσα Kotlin

ΒΙΒΛΙΟΓΡΑΦΙΑ

Adapter. (χ.χ.). Ανάκτηση από Android Developers: https://developer.android.com/reference/android/widget/Adapter

Arrays (Java Platform SE 8). (χ.χ.). Ανάκτηση από Oracle Help Center: https://docs.oracle.com/javase/8/docs/api/java/util/Arrays.html

Data access object. (2020, 12 25). Ανάκτηση από Wikipedia: https://en.wikipedia.org/w/index.php?title=Data_access_object&oldid=996193824

Data Binding Library . (χ.χ.). Ανάκτηση από Android Developers: https://developer.android.com/topic/libraries/data-binding

Data transfer object. (2021, 3 31). Ανάκτηση από Wikipedia: https://en.wikipedia.org/wiki/Data_transfer_object

Decorators. (χ.χ.). Ανάκτηση από TypeScript: https://www.typescriptlang.org/docs/handbook/decorators.html

Dependency injection. (2021, 8 23). Ανάκτηση από Wikipedia: https://en.wikipedia.org/w/index.php?title=Dependency_injection&oldid=1040179413

Design a microservice domain model. (2020, 1 30). Ανάκτηση από Microsoft Docs: https://docs.microsoft.com/en-us/dotnet/architecture/microservices/microservice-ddd-cqrs-patterns/microservice-domain-model

Design the infrastructure persistence layer. (2018, 8 10). Ανάκτηση από Microsoft Docs: https://docs.microsoft.com/en-us/dotnet/architecture/microservices/microservice-ddd-cqrs-patterns/infrastructure-persistence-layer-design

Documentation. (χ.χ.). Ανάκτηση από NestJS: https://docs.nestjs.com/

FAQ. (22, 1 11). Ανάκτηση από Kotlin Programming Language: https://kotlinlang.org/docs/faq.html#is-kotlin-an-object-oriented-language-or-a-functional-one

Gaurav. (2022, 2 4). *Minimize Cash Flow among a given set of friends who have borrowed money from each other*. Ανάκτηση από Geeks for Geeks: https://www.geeksforgeeks.org/minimize-cash-flow-among-given-set-friends-borrowed-money/

Khorikov, V. (2016, 4 5). *Having the domain model separated from the persistence model*. Ανάκτηση από Enterprise Craftsmanship: https://enterprisecraftsmanship.com/posts/having-the-domain-model-separate-from-the-persistence-model/

Lesson: Annotations. (χ.χ.). Ανάκτηση από The Java Tutorials: https://docs.oracle.com/javase/tutorial/java/annotations/

Model-view-controller. (2020, 4 15). Ανάκτηση από Wikipedia: https://el.wikipedia.org/wiki/Model-view-controller

Model–view–viewmodel. (2021, 6 15). Ανάκτηση από Wikipedia: https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93viewmodel

Nicolas Auger, Vincent Jugé, Cyril Nicaud, & Carine Pivoteau. (2018). On the Worst-Case Complexity of TimSort. Στο Yossi Azar, Hannah Bast, & Grzegorz Herman, *26th Annual European Symposium on Algorithms (ESA 2018)* (σσ. 4:1--4:13). Dagstuhl, Germany: Schloss Dagstuhl--Leibniz-Zentrum fuer Informatik. Ανάκτηση από https://drops.dagstuhl.de/opus/volltexte/2018/9467/

Object–relational mapping. (2021, 7 11). Ανάκτηση από Wikipedia: https://en.wikipedia.org/w/index.php?title=Object%E2%80%93relational_mapping&oldid=1033054602

Pătcaş, C., & Bartha, A. (2019). Evolutionary solving of the debts' clearing. *Acta Universitatis Sapientiae*.

PostgreSQL: Documentation 10: 11.3. Multicolumn Indexes. (χ.χ.). Ανάκτηση από PostgreSQL: https://www.postgresql.org/docs/10/indexes-multicolumn.html

SQL Injection Prevention. (χ.χ.). Ανάκτηση από OWASP Cheat Sheet Series: https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html

SQLite Home Page. (χ.χ.). Ανάκτηση από SQLite Home Page: https://www.sqlite.org/index.html

The SQLite Query Optimizer Overview. (χ.χ.). Ανάκτηση από SQLite: https://www.sqlite.org/optoverview.html

TypeScript. (2021, 7 2). Ανάκτηση από Wikipedia: https://en.wikipedia.org/w/index.php?title=TypeScript&oldid=1031588238

View. (χ.χ.). Ανάκτηση από Android Developers: https://developer.android.com/reference/android/view/View

View Binding. (χ.χ.). Ανάκτηση από Android Developers: https://developer.android.com/topic/libraries/view-binding

ViewGroup. (χ.χ.). Ανάκτηση από Android Developers: https://developer.android.com/reference/android/view/ViewGroup

What is Java? (χ.χ.). Ανάκτηση από IBM: https://www.ibm.com/cloud/learn/java-explained

Yao, Y. (2017). *Settling Debts Efficiently: Zero-Sum Set Packing.* MIT.

Έγχυση SQL. (2021, 7 3). Ανάκτηση από Wikipedia: https://el.wikipedia.org/wiki/%CE%88%CE%B3%CF%87%CF%85%CF%83%CE%B7_SQL

Ι. Ν. Έλληνας, Ν. Ι. (2014, 2017). *Εισαγωγή στον προγραμματισμό Android.* Θεσσαλονίκη: Τζιόλα.

Σκουρλάς, Χ. (2019). *Παρουσίαση «Εισαγωγή στον προγραμματισμό με χρήση triggers. Χρήση τεχνολογίας PL/SQL, χρήση MySQL».* Πανεπιστήμιο Δυτικής Αττικής, Τμήμα Μηχανικών Πληροφορικής και Υπολογιστών, Αθήνα.