Τεχνολογικό Εκπαιδευτικό Ίδρυμα Κρήτης
Σχολή Τεχνολογικών Εφαρμογών
Τμήμα Μηχανικών Πληροφορικής

Πτυχιακή Εργασία

Τίτλος:
Κατανεμημένο Σύστημα για Εξαγωγή Γνώσης σε Peer-to-Peer Δίκτυα πάνω από Android.

Αλέξανδρος Δριδάκης (3836)

Επιβλέπων εκπαιδευτικός : Νικόλαος Παπαδάκης

ΗΡΑΚΛΕΙΟ

2019

Σύνοψη

Η λογική στην επιστήμη των υπολογιστών περιγράφει θέματα όπου τα μοντέλα λογικής εφαρμόζονται στην εκπροσώπηση της γνώσης και της τεχνητής νοημοσύνης. Η χρήση κατανεμημένων συστημάτων για την επίλυση λογικών προβλημάτων είναι ένα ενδιαφέρον αλλά και προκλητικό θέμα και οι αλγόριθμοι των ερωτήσεων δικτύου κλάδεμα έχουν ως αποτέλεσμα τη βελτιστοποίηση της διαδικασίας επίλυσης. Η τρέχουσα πτυχιακή προτείνει την ανάπτυξη ενός εργαλείου συλλογιστικής υλοποιημένο σε Peer-To-Peer δίκτυο, όπου αξιολογούνται τοπικοί κανόνες συλλογιστικής με ελλιπή στοιχεία. Αυτό επιτυγχάνεται με την αποστολή ερωτημάτων για τις ανάγκες σε συνομηλίκους, χρησιμοποιώντας ένα πρωτόκολλο που διαγράφει από το δίκτυο τα περιττά ερωτήματα και ερωτήματα που καθυστερούν την απάντηση.

Abstract

Logic in computer science describes topics where logic models are applied to the representation of knowledge and artificial intelligence. The usage of distributed systems to solve logical problems is both an interesting and a challenging one, and algorithms of pruning network queries result in the optimization of the solving process. The current thesis proposes the development of a Peer-To-Peer Reasoning tool, which evaluates local rules of reasoning with incomplete data. This is achieved by sending queries for those needs to peers using a protocol that deletes unnecessary queries and queries that delay the response from the network.

Πίνακας περιεχομένων

Λίστα εικόνων

Λίστα πινάκων

1. Εισαγωγή

Στην σύγχρονη εποχή οι συσκευές smartphone αποτελούν ένα μεγάλο μέρος της καθημερινότητας των ανθρώπων. Αυτό οφείλεται στο γεγονός ότι τα smartphones παρέχουν ένα συνεχώς αυξανόμενο εύρος λειτουργειών, δίνοντας στους χρήστες νέες δυνατότητες και εμπειρίες. Καθώς η επεξεργαστική ισχύ των φορητών συσκευών αυξάνεται, ανοίγουν νέοι ορίζοντες στον τομέα της πληροφορικής και συγκεκριμένα της τεχνικής νοημοσύνης. Μια συσκευή smartphone έχει πλέον τη δυνατότητα να επεξεργάζεται τεράστιους όγκους δεδομένων και με βάση αυτά να παράγει νέα δεδομένα. Αυτή η ικανότητα θα μπορούσε να συγκριθεί με εκείνη των ανθρώπων να εξάγουν συμπεράσματα με βάση την προϋπάρχουσα γνώση τους.

Πολλές φορές η συνεργασία μεταξύ των ανθρώπων είναι απαραίτητή για την επίτευξη ενός στόχου. Η ιστορία της ανθρωπότητας έχει αποδείξει ότι η συλλογική προσπάθεια μπορεί να οδηγήσει σε καλύτερα αποτελέσματα σε σχέση με την ατομική, διότι κάθε μέλος μια ομάδας προσφέρει τη γνώση και τις δυνατότητες του. Αυτή η φιλοσοφία μπορεί να επεκταθεί και στις συσκευές smartphone. Οι σύγχρονες τεχνολογίες ραδιοκυμάτων επιτρέπουν στις συσκευές να σχηματίσουν ένα δίκτυο αποκεντρωτικό, ανεξάρτητο από τα δίκτυα κινητή τηλεφωνίας, όπου οι συσκευές διαμοιράζονται δεδομένα και συνεργάζονται με τέτοιο τρόπο ώστε να ανακαλύπτουν νέα γνώση. Η εργασία έχει ως στόχο τον σχεδιασμό και την υλοποίηση ενός συστήματος το οποίο ανακαλύπτει νέα γνώση στα πλαίσια ενός Peer-to-peer δικτύου. Η υλοποίηση του συστήματος βασίζεται στο λειτουργικό σύστημα Android το οποίο είναι ένα από τα δημοφιλέστερα συστήματα φορητών συσκευών.

1.1 Το Λειτουργικό Σύστημα Android

Το Android[1], είναι ένα λειτουργικό σύστημα, σχεδιασμένο για συσκευές κινητής τηλεφωνίας, το οποίο αρχικά αναπτύχθηκε από την εταιρεία Google[2]. Ο σχεδιασμός του είναι βασισμένος στον πυρήνα του λειτουργικού Linux[3]. Υποστηρίζεται από συσκευές με οθόνη αφής όπως τα έξυπνα τηλέφωνα (smartphones), τάμπλετ (tablets) καθώς και συσκευές με εξειδικευμένη διεπαφή χρήστη όπως έξυπνα ρολόγια (Android Wear), έξυπνες τηλεοράσεις (Android TV) κι κεντρικές μονάδες αυτοκινήτων (Android Auto)[1]. Το Android συνήθως περιλαμβάνει ένα πακέτο προ-εγκατεστημένων εφαρμογών, ανεπτυγμένων από την Google, το οποίο ονομάζεται Google Mobile Services[5]. Αυτή το πακέτο περιέχει εφαρμογές όπως το Google Chrome, Gmail, Google Maps, Google Drive, Google Play Store, YouTube κ.α. Επιπλέον η εφαρμογή Google Play Store (ή Google Play) διαθέτει ένα μεγάλο εύρος εφαρμογών οι οποίες επεκτείνουν την λειτουργικότητα των συσκευών, δίνοντας την δυνατότητα στον χρήστη να χρησιμοποιήσει την συσκευή ως μέσο επικοινωνίας, ενημέρωσης, πλοήγησης αποθήκευσης δεδομένων και ψυχαγωγίας.

1.2 Χαρακτηριστικά του Android

Το Android είναι ένα ισχυρό λειτουργικό σύστημα ανοιχτού κώδικα το οποίο διαθέτει πληθώρα χαρακτηριστικά. Τα χαρακτηριστικά αυτά είναι τα ακόλουθα:

- Είναι λογισμικό ανοιχτού κώδικα, οπότε μπορεί να τροποποιηθεί σύμφωνα με τις ανάγκες του χρήστη

- Υποστηρίζει σύγχρονες τεχνολογίες συνδεσιμότητας όπως GSM, CDMA, WIFI, NFC, Bluetooth κτλ., οι οποίες εξυπηρετούν την επικοινωνία και την μεταφορά δεδομένων. Επιτρέπει στον χρήστη να εκτελεί και να δέχεται κλήσεις, να στέλνει μηνύματα SMS ή MMS και να έχει πρόσβαση σε δίκτυα κινητής τηλεφωνίας
- Επιτρέπει την σύζευξη 2 συσκευών, μέσω της τεχνολογίας Wi-Fi
- Εμπεριέχει βιβλιοθήκες που υποστηρίζουν υπηρεσίες τοποθεσίας όπως το GPS
- Δίνει την δυνατότητα χρήσης της βάσης δεδομένων SQLite για αποθήκευση δεδομένων που σχετίζονται με τις δραστηριότητες (activity) μιας εφαρμογής
- Υποστηρίζει ένα μεγάλο εύρος τεχνολογιών όπως AVI, MKV, FLV, MPEG4 για αναπαραγωγή βίντεο, ήχου καθώς και διάφορες μορφές εικόνας όπως JPEG, PNG, GIF, BMP, MP3 κα.
- Περιλαμβάνει έναν περιηγητή ιστού ανοιχτού κώδικα ο οποίος υποστηρίζει HTML5 και CSS3
- Η πολυδιεργασία επιτρέπει στον χρήστη να εκτελεί πολλές εφαρμογές ταυτόχρονα
- Ο χρήστης έχει πρόσβαση σε εξαρτήματα του υλικού όπως η κάμερα, το GPS κτλ.
- Υποστηρίζει 2D και 3D γραφικά

1.3 Οι Εκδόσεις του Android

Η πρώτη έκδοση του κυκλοφόρησε στις 23 Σεπτεμβρίου 2008[4] και είχε την ονομασία Alpha. Η δεύτερη έκδοση ονομαζόταν Beta και κυκλοφόρησε τον Φεβρουάριο του 2009. Έκτοτε η Google κυκλοφορεί κάθε χρόνο μια καινούργια έκδοση του λειτουργικού, με κωδικές ονομασίες εμπνευσμένες από γλυκίσματα και επιδόρπια από το 2009 έως το 2018.

1.3.1 Android 1.5 Cupcake

Η τρίτη έκδοση του Android, το Android Cupcake κυκλοφόρησε στις 27 Απριλίου 2009 και τα κύρια χαρακτηριστικά της ήταν το εικονικό πληκτρολόγιο, υποστήριξη Bluetooth καθώς και βελτιώσεις που αφορούν την διεπαφή χρήστη. Πλέον δεν υποστηρίζεται από φορητές συσκευές[6].

1.3.2 Android 1.6 Donut

Η τέταρτη έκδοση Android, είχε την κωδική ονομασία Android Donut και κυκλοφόρησε στις 15 Σεπτεμβρίου 2009. Η αναβάθμιση αυτή συμπεριλάμβανε, μεταξύ άλλων νέων χαρακτηριστικών, επιπρόσθετα μεγέθη οθόνης, ένδειξη χρήσης της μπαταρίας της συσκευής και έναν μηχανισμό μετατροπής κειμένου σε ομιλία[7].

1.3.3 Android 2.0 Eclair

Η πέμπτη έκδοση του Android, το Android Eclair κυκλοφόρησε στις 26 Οκτώβριου 2009. Η ανάπτυξη του λειτουργικού στηρίχθηκε στις αλλαγές που έφερε η έκδοση που προηγήθηκε, το Android Donut, όπως για παράδειγμα η βελτίωση της λειτουργικότητας του Google Maps, υποστήριξη HTML5 στον περιηγητή ιστού κ.α.[8].

1.3.4 Android 2.2 Froyo

Το Android Froyo είναι η κωδική ονομασία της έκτης έκδοσης του λειτουργικού Android, η οποία παρουσιάστηκε στις 20 Μαΐου στο συνέδριο Google I/O 2010. Οι πιο διακεκριμένες προσθήκες, ήταν η τεχνικά USB tethering, η οποία επιτρέπει στην συσκευή να λειτουργεί ως δρομολογητής (router)[9] καθώς και η τεχνολογία Wi-Fi Hotspot[10]

1.3.5 Android 2.3 Gingerbread

Το Android Gingerbread κυκλοφόρησε στις 6 Δεκέμβριου 2010. Η έκδοση αυτή εισήγαγε την τεχνολογία *επικοινωνία κοντινού πεδίου (NFC) και την τεχνολογία VoIP*. Η διεπαφή του επανασχεδιάστηκε με τέτοιο τρόπο, ώστε η συσκευή να είναι πιο γρήγορη στην χρήση και περισσότερο αποδοτική σε ισχύ[11].

1.3.6 Android 3.2 Honeycomb

Η όγδοη έκδοση του, το Android Honeycomb σχεδιάστηκε για συσκευές με μεγαλύτερες σε μέγεθος οθόνες , όπως για παράδειγμα συσκευές τάμπλετ. Διέθετε ένα νέο για την εποχή μοντέλο διάδρασης το οποίο ήταν βασισμένο στα κύρια χαρακτηριστικά του λειτουργικού, όπως πολυδιεργασία (multitasking), ειδοποιήσεις (notifications) κ.α.[12]

1.3.7 Android 4.0 Ice Cream Sandwich

Το Android Ice Cream Sandwich κυκλοφόρησε στις 19 Οκτώβριου 2011 και έφερε σημαντικές αλλαγές όσον αφορά την λειτουργικότητα και τη διαδραστικότητα. Διέθετε έναν ανανεωμένο περιηγητή ιστού, ενώ παρείχε την δυνατότητα πρόσβασης στην κάμερα και την αναπαραγωγή μουσικής από την οθόνη κλειδώματος. Επιπλέον συμπεριλάμβανε αναγνώριση προσώπου για «ξεκλείδωμα» της συσκευής και παρακολούθηση των δεδομένων κινητής τηλεφωνίας[13].

1.3.8 Android 4.1-4.3 Jelly Bean

Η δέκατη έκδοση του λειτουργικού Android το Android Jelly Bean επεκτάθηκε σε τρείς εκδόσεις μεταξύ 4.1 και 4.3.1, οι οποίες κυκλοφόρησαν στις 13 Ιουλίου 2012 και 7 Οκτωβρίου 2013 αντίστοιχα. Η συγκεκριμένη αναβάθμιση επικεντρώθηκε στην βελτίωση της απόδοση των συσκευών, ενώ ο σχεδιασμός της είχε ως στόχο να προσδώσει στον χρήστη μια ομαλότερη και με γρηγορότερη απόκριση διαδραστικότητα[14].

1.3.9 Android 4.4 KitKat

Το Android KitKat είναι η κωδική ονομασία της ενδέκατης έκδοσης του Android, ενώ επεκτάθηκε σε τέσσερεις εκδόσεις μεταξύ των 4.4 και 4.4.4. Κυκλοφόρησε στις 31 Οκτωβρίου 2013 με επίκεντρο την βελτιστοποίηση της απόδοσης του λειτουργικού συστήματος σε

συσκευές με περιορισμένους πόρους. Σύμφωνα με επίσημα στατιστικά στοιχεία που εκδόθηκαν από την Google τον Μάϊο του 2019, το 6,9% των χρηστών που είναι εγγεγραμμένοι στην υπηρεσία Google Play έχουν εγκατεστημένο στις συσκευές τους το λειτουργικό Android KitKat[15][16].

1.3.10 Android 5.0-5.1 Lollipop

Η δωδέκατη έκδοση του Android, το Android Lollipop παρουσιάστηκε στο συνέδριο Google I/O 2014 στις 25 Ιουνίου 2014. Έπειτα έγινε διαθέσιμο σε επιλεγμένες συσκευές, κατασκευασμένες από την Google, στις 12 Νοέμβριου για συσκευές της Google. Μια από τις προφανέστερες αλλαγές που εισήγαγε αυτή η αναβάθμιση, ήταν η επανασχεδιασμένη διεπαφή χρήστη, η οποία ήταν βασισμένη στην σχεδιαστική γλώσσα Material Design[17].

1.3.11 Android 6.0 Marshmallow

Το Android Marshmallow κυκλοφόρησε επισήμως στις 5 Οκτωβρίου 2015. Οι αναβαθμίσεις επικεντρώθηκαν στην βελτίωση της διαδραστικότητας που εισήγαγε η προηγούμενη έκδοση, ενώ παράλληλα παρουσίασε ένα νέο σύστημα διαχείρισης πόρων του συστήματος, η οποία περιορίζει την εκτέλεση διεργασιών στο παρασκήνιο. Άλλες διαθέσιμες τεχνολογίες ήταν η αναγνώριση δακτυλικού αποτυπώματος για το «ξεκλείδωμα» της οθόνης και η συνδεσμολογία USB-C[18]

1.3.12 Android 7.0-7.1 Nougat

Το Android Nougat κυκλοφόρησε ως alpha test έκδοση στις 9 Μαρτίου 2016 και αργότερα κυκλοφόρησε επίσημα στις 22 Αυγούστου. Αρχικά η διαθεσιμότητα της αναβάθμισης ήταν περιορισμένη σε συσκευές της σειράς Nexus. Επιπροσθέτως η έκδοση αυτή, έφερε σημαντικές αλλαγές στο λειτουργικό σύστημα, όπως η προβολή πολλαπλών εφαρμογών στην οθόνη και ένα σύστημα εξοικονόμησης ενέργειας, το οποίο περιορίζει την λειτουργικότητα της συσκευής όταν παραμένει αδρανής για μεγάλο χρονικό διάστημα[19].

1.3.13 Android 8.0-8.1 Oreo

Το Android Oreo, κυκλοφόρησε στις 21 Αυγούστου 2017. Η έκδοση αυτή εισήγαγε νέα χαρακτηρίστηκα όπως η ομαδοποίηση ειδοποιήσεων, υποστήριξη picture-in-picture για αναπαραγωγή βίντεο, Bluetooth 5, Wi-Fi Aware κ.α.[20].

1.3.14 Android 9 Pie

Το Android Pie είναι η δέκατη έκτη έκδοση του λειτουργικού Android, η οποία κυκλοφόρησε στις 6 Αυγούστου 2018[21].

1.3.15 Android 10

Η δέκατη έβδομη και πιο πρόσφατη έκδοση του λειτουργικού συστήματος Android έχει την κωδική ονομασία Android 10 και κυκλοφόρησε στις 3 Σεπτεμβρίου 2019[22]. Ο παρακάτω πίνακας περιγράφει συνοπτικά το ιστορικό των εκδόσεων του Android

Ημερομηνία Έκδοσης	Έκδοση	Επίπεδο API	Κωδική Ονομασία
23 Σεπτεμβρίου, 2008	1.0	1	Alpha
9 Φεβρουαρίου, 2009	1.1	2	Beta
30 Απριλίου, 2009	1.5	3	Cupcake
15 Σεπτεμβρίου, 2009	1.6	4	Donut
26 Οκτωβρίου, 2009	2.0	5	
3 Δεκεμβρίου, 2009	2.0.1	6	Eclair
12 Ιανουαρίου, 2009	2.1	7	
20 Μαΐου, 2010	2.2	8	
18 Ιανουαρίου, 2011	2.2.1	8	Froyo
22 Ιανουαρίου, 2011	2.2.2	8	
21 Νοεμβρίου, 2011	2.2.3	8	
6 Δεκεμβρίου, 2010	2.3	9	
9 Φεβρουαρίου, 2011	2.3.1	9	Gingerbread
25 Ιουλίου, 2011	2.3.3	10	
2 Σεπτεμβρίου, 2011	2.3.4	10	
22 Φεβρουαρίου, 2011	3.0	11	
10 Μαΐου, 2011	3.1	12	Honeycomb
15 Ιουλίου, 2011	3.2	13	
18 Οκτωβρίου,2011	4.0	14	
16 Δεκεμβρίου, 2011	4.0.3	15	Ice Cream Sandwich
4 Φεβρουαρίου, 2012	4.0.4	15	
9 Ιουλίου, 2012	4.1	16	
23 Ιουλίου, 2012	4.1.1	16	
9 Οκτωβρίου, 2012	4.1.2	16	Jelly Bean
13 Νοεμβρίου, 2012	4.2	17	
27 Νοεμβρίου, 2012	4.2.1	17	
11 Φεβρουαρίου, 2013	4.2.2	17	

24 Ιουλίου , 2013	4.3	18	
31 Οκτωβρίου, 2013	4.4	19	KitKat
17 Οκτωβρίου, 2014	5.0	21	Lollipop
09 Μαρτίου, 2015	5.1	22	
5 Οκτωβρίου, 2015	6.0	23	Marshmallow
7 Δεκεμβρίου, 2015	6.0.1	23	
22 Αυγούστου, 2016	7.0	24	Nougat
4 Οκτωβρίου, 2016	7.1	25	
21 Αυγούστου 2017	8.0	26	O
5 Δεκεμβρίου 2017	8.1.0	27	
6 Αυγούστου 2018	9	28	Pie
3 Σεπτεμβρίου 2018	10	29	Android 10

Πίνακας: Οι εκδόσεις του Android και οι ημερομηνίες κυκλοφορίας τους

1.4 Αρχιτεκτονική του Android

Το Android είναι ένα λογισμικό ή μια στοίβα λογισμικού ανοιχτού κώδικα, το οποίο είναι σχεδιασμένο με βάση τον πυρήνα του λειτουργικού συστήματος Linux. Επιπλέον η στοίβα χωρίζεται σε πέντε επίπεδα τα οποία αντιπροσωπεύουν τα βασικά στοιχεία του λειτουργικού συστήματος[25]. Τα στοιχεία αυτά αναλύονται παρακάτω.

1.4.1 Πυρήνας Linux (Linux Kernel)

Ο πυρήνας Linux αποτελεί το κατώτερο επίπεδο της πλατφόρμας του Android. Το επίπεδο αυτό διαχειρίζεται όλα τα πρόγραμμα οδήγησης (drivers) που αφορούν το υλικό της συσκευής, όπως για παράδειγμα την κάμερα, το Bluetooth, την μνήμη, τους πόρους και την διαχείριση της ενέργεια της μπαταρίας.

1.4.2 Επίπεδο αφαίρεσης υλικού (Hardware Abstraction Layer)

Το επίπεδο αφαίρεσης υλικού ορίζει ένα σύνολο διεπαφών (interfaces) οι οποίες διευκολύνουν την επικοινωνία του υλικού με το επίπεδο πλαισίου εφαρμογής (Application Framework). Περιέχει ένα σύνολο βιβλιοθηκών, κάθε μια από τις οποίες υλοποιούν μια διεπαφή για ένα συγκεκριμένο στοιχείο υλικού, δηλαδή βιβλιοθήκη για την κάμερα ή το μικρόφωνο κα. Όταν μια εφαρμογή ζητά πρόσβαση στο υλικό της συσκευής, τότε το λειτουργικό σύστημα φορτώνει την αντίστοιχη βιβλιοθήκη που σχετίζεται με αυτό το στοιχείο υλικού.

1.4.3 Android Runtime (ART)

Το επίπεδο Android Runtime ενσωματώθηκε στην αρχιτεκτονική του Android στην έκδοση 5.0. Κάθε εφαρμογή εκτελείται σε μια ξεχωριστή διεργασία, πάνω στο Android Runtime. Το Android Runtime είναι ένα περιβάλλον το τρέχει πολλές εικονικές μηχανές για συσκευές χαμηλής μνήμης, εκτελώντας αρχεία DEX. Τα αρχεία αυτά αποτελούν μια μορφή κώδικα (bytecode) ειδικά σχεδιασμένη για το Android. Με τη χρήση εξειδικευμένων εργαλείων, ο κώδικας γραμμένος σε Java μεταφράζεται σε κώδικα DEX (Dalvik Executable), ο οποίος στη συνέχεια εκτελείται από το λειτουργικό σύστημα. Κάποια από τα βασικά χαρακτηριστικά του περιβάλλοντος Android Runtime είναι η βελτιστοποιημένη συλλογή απορριμμάτων (garbage collection) , η υποστήριξη εκσφαλμάτωση κ.α.

Σε προηγούμενες εκδόσεις το επίπεδο αυτό αποτελούσε η εικονική μηχανή Dalvik. Οι εφαρμογές που τρέχουν αποτελεσματικά στο περιβάλλον Android Runtime, έχουν την ίδια απόδοση στην μηχανή Dalvik, όμως το αντίστροφο δεν ισχύει.

1.4.4 Βιβλιοθήκες C/C++

Τα κατώτερα επίπεδα της στοίβας του λογισμικού του Android , δηλαδή το επίπεδο Android Runtime και το επίπεδο αφαίρεσης υλικού (HAL) έχουν υλοποιηθεί με βάση της ενσωματωμένες βιβλιοθήκες C και C++. Το Android παρέχει βιβλιοθήκες υλοποιημένες στην γλώσσα προγραμματισμού Java, ώστε να επιτρέψει στις εφαρμογές να αξιοποιήσουν τις λειτουργίες αυτών των βιβλιοθηκών.

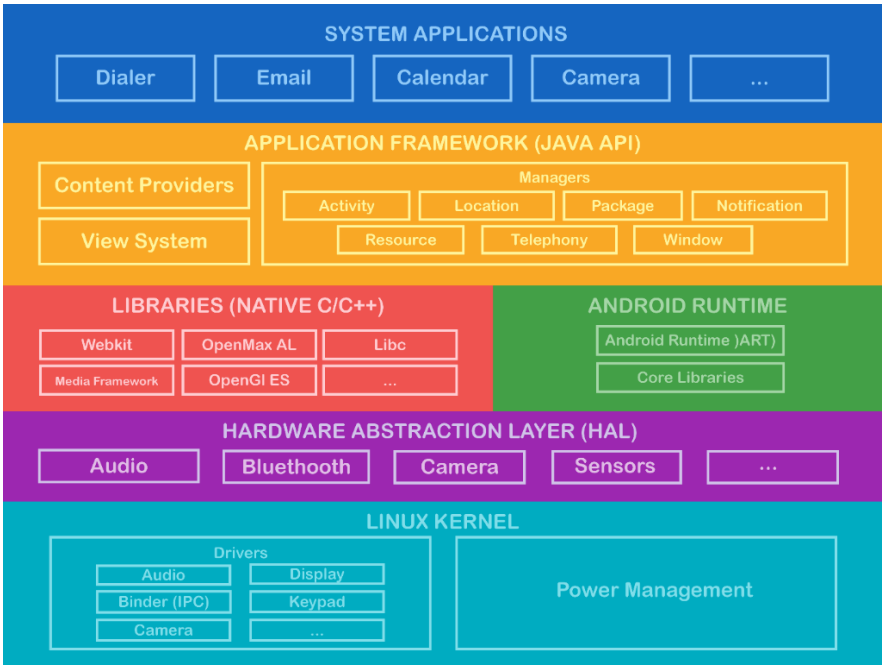
1.4.5 Java API Framework

Το Java API Framework είναι ένα σύνολο από βιβλιοθήκες γραμμένες σε Java, οι οποίες χρησιμοποιούνται για ανάπτυξή εφαρμογών καθώς και υλοποίηση χαρακτηριστικών του λειτουργικού συστήματος. Οι βιβλιοθήκες αυτές σχηματίζουν δομικά στοιχεία (building blocks) τα οποία είναι απαραίτητα για την ανάπτυξη εφαρμογών, απλοποιώντας την επαναχρησιμοποίηση στοιχείων του συστήματος ή υπηρεσιών όπως τα παρακάτω:

- Το View System το οποίο χρησιμοποιείται για τον σχεδιασμό διεπαφών χρήστη και περιλαμβάνει λίστες, πεδία κειμένου, κουμπιά κα.
- Ο διαχειριστής πόρων (Resource Manager), ο οποίος διαχειρίζεται πόρους που χρησιμοποιούνται από εφαρμογές, όπως για παράδειγμα πολυμέσα, γραφικά κτλ.
- Ο διαχειριστής ειδοποιήσεων (Notification Manager), ο οποίος επιτρέπει στις εφαρμογές να εμφανίζει ειδοποιήσεις στην γραμμή κατάστασης (Status Bar)
- Ο διαχειριστής δραστηριοτήτων (Activity Manager), ο οποίος αναλαμβάνει την διαχείριση του κύκλου ζωής των δραστηριοτήτων (Activity lifecycle) που απαρτίζουν μια εφαρμογή.
- Ο πάροχος περιεχομένου (Content Provider), ο οποίος δίνει την δυνατότητα σε μια εφαρμογή να έχει πρόσβαση σε δεδομένα άλλων εφαρμογών, η να διαμοιράζονται τα δεδομένα της.

1.4.6 Εφαρμογές συστήματος (System Apps)

Το λειτουργικό σύστημα Android εμπεριέχει ένα σύνολο από προ-εγκατεστημένες εφαρμογές που χρησιμοποιούνται για αποστολή μηνυμάτων, email, καθώς και ημερολόγιο περιηγητή διαδικτύου (internet browser), λίστα επαφών κα. Παρόλα αυτά ο χρήστης μπορεί να εγκαταστήσει στην συσκευή άλλες εφαρμογές που εκτελούν τις ίδιες λειτουργίες. Επιπλέον οι λειτουργίες των εφαρμογών συστήματος επιτρέπουν στους προγραμματιστές να χρησιμοποιήσουν της λειτουργίες αυτές στις δικές τους εφαρμογές. Για παράδειγμά, εάν κάποια εφαρμογή έχει δυνατότητα λήψης φωτογραφιών, δεν είναι απαραίτητο να υλοποιεί τη λειτουργία αυτή, διότι μπορεί να έχει πρόσβαση στην εφαρμογή κάμερας που είναι ήδη εγκατεστημένη στην συσκευή. Η εικόνα 1 απεικονίζει την στοίβα λογισμικού του Android.



Εικόνα 1: Η στοίβα του λογισμικού Android

1.5 Η εφαρμογή Group Reasoning

Η εφαρμογή Group Reasoning είναι μια εφαρμογή ανεπτυγμένη στην πλατφόρμα του Android και αποτελεί ένα σύστημα εξαγωγής συμπερασμάτων το οποίο είναι κατανεμημένο σε ένα δίκτυο Peer-to-peer. Κάθε κόμβος του δικτύου επιχειρεί να επιλύσει λογικά προβλήματα με βάση την τοπική γνώση του. Εάν η γνώση είναι ανεπαρκής τότε θέτει ερωτήματα στους γειτονικούς κόμβους οι οποίοι προσπαθούν να απαντήσουν αυτά τα ερωτήματα και στέλνουν πίσω τα αποτελέσματα.

1.6 Δομή εργασίας

Η εργασία αυτή αποτελείται από 5 κεφάλαια:

- Στο κεφάλαιο 1 γίνεται μια εισαγωγή για το λειτουργικό σύστημα Android, αναφέρονται οι προηγούμενες εκδόσεις του κι η περιγραφή της αρχιτεκτονικής του ενώ συμπεριλαμβάνεται και μια σύντομη περιγραφή της εφαρμογής Group Reasoning.
- Στο κεφάλαιο 2 περιγράφονται αναλυτικά τα θεμελιώδη χαρακτηριστικά μιας εφαρμογής ανεπτυγμένης στην πλατφόρμα Android.
- Στο κεφάλαιο 3 αναλύονται οι προδιαγραφές και η φιλοσοφία στην οποία βασίζεται η εφαρμογή Group Reasoning
- Στο κεφάλαιο 4 περιγράφεται η διαδικασία υλοποίησης της εφαρμογής Group Reasoning
- Στο κεφάλαιο 5 παρουσιάζονται τα αποτελέσματα και τα συμπεράσματα της ανάπτυξης της εφαρμογής

2 Θεμελιώδη χαρακτηριστικά μιας Android εφαρμογής

Μια εφαρμογή Android μπορεί να αναπτυχθεί με τη χρήση της γλώσσας προγραμματισμού Java, καθώς και Kotlin κι C++. Το Android SDK διαθέτει εργαλεία τα οποία αναλαμβάνουν την μεταγλώττιση του κώδικα και την μετατροπή του σε αρχείο της μορφής APK (Android Package). Επιπλέον το αρχείο APK περιλαμβάνει όλα τα δεδομένα και τους πόρους της εφαρμογής και χρησιμοποιείται για την εγκατάσταση της εφαρμογής σε μια συσκευή.

Το λειτουργικό σύστημα Android αντιμετωπίζει κάθε εφαρμογή σαν έναν διαφορετικό χρήστη, αναθέτοντας της ένα μοναδικό αναγνωριστικό χρήστη (user ID). Επίσης κάθε εφαρμογή εκτελείται σε διαφορετική διεργασία κάθε μια από τις οποίες έχει την δική της εικονική μηχανή[26].

Το Android ακολουθεί την αρχή του ελάχιστου προνομίου (Principle of Least Privilege[27]), κατά την οποία κάθε εφαρμογή έχει πρόσβαση μόνο στα συστατικά του λογισμικού που είναι απαραίτητα για την λειτουργία της. Με αυτόν το τρόπο το σύστημα απαγορεύει στις εφαρμογές την πρόσβαση σε πόρους του συστήματος για τους οποίους δεν έχουν άδεια (permission). Παρόλα αυτά μια εφαρμογή μπορεί να αιτηθεί άδεια πρόσβασης σε υπηρεσίες του συστήματος όπως η τοποθεσία, η χρήση κάμερας, ή διασύνδεση με άλλη συσκευή μέσω Bluetooth. Τέτοιου είδους άδειες θα πρέπει να εγκρίνονται από τον ίδιο τον χρήστη.

2.1 Δομικά στοιχεία μιας εφαρμογής

Τα δομικά στοιχεία μιας εφαρμογής Android χωρίζονται σε 4 βασικές κατηγορίες κάθε μια από τις οποίες εξυπηρετεί το δικό της σκοπό και έχει συγκεκριμένο κύκλο ζωής (lifecycle):

- Δραστηριότητες (Activities)
- Υπηρεσίες (Services)
- Δέκτες εκπομπής (Broadcast Receivers)
- Πάροχοι περιεχομένου (Content Providers)

Ο σχεδιασμός του συστήματος του Android επιτρέπει σε μια εφαρμογή να θέσει σε λειτουργία ένα δομικό στοιχείο μιας άλλης εφαρμογής. Παραδείγματος χάρη, εάν μια εφαρμογή δίνει στον χρήστη την επιλογή αποστολής ενός email, μπορεί να χρησιμοποιήσει την εφαρμογή ηλεκτρονικού ταχυδρομείου (π.χ. Gmail) για την εκτέλεση αυτής της εργασίας.

2.1.1 Δραστηριότητες (Activities)

Μια δραστηριότητα αποτελεί το σημείο εκκίνησης μιας εφαρμογής αλλά και μέσο με το οποίο ο χρήστης αλληλεπιδρά με μια εφαρμογή. Αντιπροσωπεύει μια οθόνη η οποία απεικονίζει την διεπαφή χρήστη. Λόγω του μικρού μεγέθους των συσκευών που χρησιμοποιούν το λειτουργικό σύστημα Android, μια εφαρμογή αποτελείται από περισσότερες από μια δραστηριότητες στις οποίες αντιστοιχεί ένα περιορισμένο σύνολο λειτουργειών. Για παράδειγμα η εφαρμογή των μηνυμάτων περιέχει μια δραστηριότητα η οποία απεικονίζει την λίστα με τις κατηγορίες μηνυμάτων (εισερχόμενα, απεσταλμένα κ.α.), μια διαφορετική δραστηριότητα με την οποία ο χρήστης μπορεί να συντάξει ένα νέο μήνυμα ή κάποια άλλη όπου θα μπορεί να διαβάζει εισερχόμενα μηνύματα. Κάθε δραστηριότητα μιας εφαρμογής είναι ανεξάρτητη από τις υπόλοιπες, ωστόσο όλες οι δραστηριότητες συνεργάζονται προκειμένου να διαμορφώσουν μια συνεκτική εμπειρία χρήσης[27]. Μια δραστηριότητα υλοποιείται ως υποκλάση (subclass) της κλάσης Activity[26][28].

2.1.2 Υπηρεσίες (Services)

Μια υπηρεσία είναι ένα συστατικό μιας εφαρμογής το οποίο χρησιμοποιείται για την εκτέλεση επαναλαμβανόμενων και χρονοβόρων διεργασιών στο παρασκήνιο και δεν διαθέτει διεπαφή χρήστη. Ενεργοποιείται από κάποια δραστηριότητα (Activity) και η εκτέλεση της μπορεί να συνεχιστεί ακόμα και εάν ο χρήστης μεταβεί σε κάποια άλλη εφαρμογή. Οι υπηρεσίες εκτελούνται στην ίδια διεργασία με το κύριο νήμα εκτέλεσης (Main thread) της εφαρμογής. Για το λόγο αυτό οι εργασίες που είναι απαιτητικές σε πόρους και υπολογιστική ισχύ θα πρέπει να εκτελούνται σε ένα νέες διεργασίες οι οποίες δημιουργούνται ασύγχρονα (AsyncTask[29]). Για παράδειγμα μια εφαρμογή αναπαραγωγής μουσικής μπορεί να χρησιμοποιήσει μια υπηρεσία για να αναπαράγει μουσική, ή να ανασύρει δεδομένα από το διαδίκτυο, όταν ο χρήστης δεν χρησιμοποιεί την εφαρμογή. Μια υπηρεσία υλοποιείται ως υποκλάση (subclass) της κλάσης Service[26][30].

2.1.3 Δέκτες εκπομπής (Broadcast Receivers)

Ένας δέκτης εκπομπής αποτελεί ένα συστατικό του Android, το οποίο δέχεται ειδοποιήσεις που εκπέμπει το σύστημα για γεγονότα του συστήματος όπως χαμηλή μπαταρία, εισερχόμενες κλήσης ή διαθεσιμότητα δικτύου κ.α. Για να λαμβάνει τέτοιου είδους ειδοποιήσεις η εφαρμογή θα πρέπει να έχει καταχωρημένο έναν δέκτη σε κάποιο από τα συστατικά της (Δραστηριότητες, Θραύσματα κτλ.). Ένα παράδειγμα δέκτη εκπομπής είναι εκείνος που είναι καταχωρημένος στην εφαρμογή του ξυπνητηριού. Ο δέκτης αυτός λαμβάνει ειδοποίηση από το σύστημα την χρονική στιγμή που έχει ορίσει ο χρήστης και στη συνέχεια ξεκινά η αναπαραγωγή του ήχου αφύπνισης. Επίσης μια εφαρμογή έχει τη δυνατότητα να εκπέμψει ειδοποιήσεις ή μηνύματα, τα οποία ονομάζονται Intents, είτε σε άλλες εφαρμογές είτε στο σύστημα, εμφανίζοντας μια ειδοποίηση στην γραμμή κατάστασης (status bar). Ένας δέκτης εκπομπής υλοποιείται ως υποκλάση (subclass) της κλάσης BroadcastReceiver[31], ενώ μια εκπομπή υλοποιείται ως υποκλάση της κλάσης Intent[32].

2.1.4 Πάροχοι περιεχομένου (Content providers)

Σκοπός του παρόχου περιεχομένου είναι η διαχείριση των δεδομένων μιας εφαρμογής είτε στον εσωτερικό χώρο αποθήκευσης ή σε μια βάση δεδομένων SQLite. Μέσω του παρόχου περιεχομένου μια εφαρμογή μπορεί να λάβει ή να τροποποιήσει δεδομένα μιας άλλης εφαρμογής, εάν ο πάροχος το εγκρίνει. Για παράδειγμα το λειτουργικό σύστημα διαθέτει έναν πάροχο ο οποίος διαχειρίζεται τις επαφές που ο χρήστης έχει αποθηκευμένες στην συσκευή του, Οπότε οποιαδήποτε εφαρμογή έχει τις απαραίτητες άδειες, έχει πρόσβαση στα δεδομένα των επαφών, προσθέτοντας νέες επαφές ή επεξεργάζονται τα στοιχεία των ήδη αποθηκευμένων επαφών. Ένας πάροχος περιεχομένου υλοποιείται ως υποκλάση (subclass) της κλάσης ContentProvider[33].

2.1.5 Το αρχείο Manifest

Κάθε εφαρμογή περιλαμβάνει ένα αρχείο όπου δηλώνονται όλα τα δομικά συστατικά της προκειμένου το σύστημα να μπορεί να τα ενεργοποιήσει όταν είναι απαραίτητο. Αυτό το αρχείο ονομάζεται manifest[34] και είναι αρχείο της μορφής xml. Εκτός από συστατικά στο αρχείο manifest δηλώνονται άδειες της εφαρμογής, την ελάχιστη προ-απαιτούμενη έκδοση του λειτουργικού που είναι εγκατεστημένη στην συσκευή και υπηρεσίες του συστήματος όπως Bluetooth ή GPS κ.α. Τα συστατικά μιας εφαρμογής δηλώνονται χρησιμοποιώντας τα εξής στοιχεία:

- Το στοιχείο <activity> δηλώνει δραστηριότητες.
- Το στοιχείο <service> δηλώνει υπηρεσίες.
- Το στοιχείο <receiver> δηλώνει δέκτες εκπομπής.
- Το στοιχείο <provider> δηλώνει παρόχους περιεχομένου.

Στην παρακάτω εικόνα απεικονίζεται ένα παράδειγμα ενός αρχείου manifest

```
<?xml version="1.0" encoding="utf-8"?>
<manifest ... >
    <application android:icon="@drawable/app_icon.png" ... >
        <activity android:name="com.example.project.ExampleActivity"
                  android:label="@string/example_label" ... >
        </activity>
         ...
        <service>...</service>
        <provider>...</service>
    </application>
    <uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
</manifest>
```

Εικόνα 2: Δήλωση συστατικών εφαρμογής στο αρχείο manifest

3 Απαιτήσεις ανάπτυξης της εφαρμογής Group Reasoning

Η εφαρμογή Group Reasoning αποτελεί ένα κατανεμημένο σύστημα το οποίο ανταλλάσσει πόρους στο πλαίσιο ενός peer-to-peer δικτύου. Είναι επίσης ένα σύστημα πολλαπλών πρακτόρων το οποίο διαθέτει γνώση βασισμένη στην προτασιακή λογική και έχει στόχο την επίλυση λογικών προβλημάτων. Κάθε κόμβος του δίκτυού είναι ένας νοήμων πράκτορας ο οποίος περιέχει ένα σύνολο λογικών προβλημάτων τα οποία χρειάζονται στοιχεία από γειτονικούς κόμβους ώστε να πραγματοποιηθεί η επίλυση τους. Αρχικά ο πράκτορας επιχειρεί την επίλυση των προβλημάτων με τη μέθοδο της προς τα εμπρός αλυσιδωτής επίλυσης ή forward chaining. Εάν η διαδικασία αυτή αποτύχει στην επίλυση ενός ή περισσότερων προβλημάτων, τότε ο κόμβος στέλνει ερωτήματα στους γειτονικούς κόμβους που ενδεχομένως διαθέτουν την απαραίτητη γνώση που θα οδηγήσει στην επίλυση των προβλημάτων αυτών.

Η εφαρμογή είναι αναπτυγμένη για συστήματα Android και υλοποιημένη σε γλώσσα java και xml. Επιπλέον χρησιμοποιεί την βιβλιοθήκη Google Nearby Connections API για την υλοποίηση του δικτύου.

3.1 Η γλώσσα προγραμματισμού Java

Η γλώσσα Java αναπτύχθηκε στις αρχές της δεκαετίας του ’90 από τον James Gosling, ο οποίος εργάζονταν εκείνη την εποχή στην εταιρεία Sun Microsystems. Η αρχική ονομασία της γλώσσας ήταν Oak, άλλα επειδή η ονομασία αυτή ανήκε σε άλλη γλώσσα, μετονομάστηκε σε Java. Είναι μια αντικειμενοστραφής, ασφαλής γλώσσα, ανεξάρτητη από πλατφόρμα και είναι σχεδιασμένη για ευκολότερη ανάπτυξη λογισμικού από άλλες αντικειμενοστραφείς γλώσσες όπως η C++.

3.1.1 Χαρακτηριστικά της Java

- **Απλότητα:** Η Java είναι μια γλώσσα με απλούστερη σύνταξη και με μεγαλύτερη ευκολία εκμάθησης από την C++.
- **Ασφάλεια:** Η Java διαθέτει ένα δυναμικό και ασφαλές σύστημα διαχείρισης μνήμης, διότι διαθέτει αυτόματη διαχείριση μνήμης με τη χρήστη του συλλέκτη απορριμμάτων (Garbage Collector), ενώ δεν χρησιμοποιεί δείκτες όπως η C++.
- **Ουδετερότητα πλατφόρμας**: Είναι η δυνατότητα ενός προγράμματος να εκτελείται σε διαφορετικά υπολογιστικά περιβάλλοντα χωρίς τροποποίηση. Ένα πρόγραμμα γραμμένο

σε Java μετασχηματίζεται σε μορφή bytecode η οποία μπορεί να εκτελεστεί σε οποιοδήποτε υπολογιστικό σύστημα που είναι εξοπλισμένο με την εικονική μηχανή της Java.

- **Αντικειμενοστρέφεια**: Η Java είναι γλώσσα αντικειμενοστραφούς προγραμματισμού, που σημαίνει ότι ένα πρόγραμμα γίνεται αντιληπτό ως μια ομάδα αντικειμένων που αλληλοεπιδρούν μεταξύ τους. Τα αντικείμενα δημιουργούνται σύμφωνα με πρότυπα που ονομάζονται κλάσεις, οι οποίες περιγράφουν τα χαρακτηριστικά και την συμπεριφορά των αντικειμένων.

3.2 XML

Η Επεκτάσιμη γλώσσα σήμανσης XML (Extensible Markup Language) αποτελεί μια ειδική μορφή κειμένου η οποία προέρχεται από τη γλώσσα SGML[36] (Standard Generalized Markup Language[37]). Είναι σχεδιασμένη με έμφαση στην απλότητα, τη γενικότητα και την χρησιμότητα στο διαδίκτυο. Υποστηρίζει το πρότυπο κωδικοποίησης Unicode[38] και χρησιμοποιείται για την αναπαράσταση αυθαίρετων δομών δεδομένων, που προκύπτουν από διαδικτυακά συστήματα[39]. Η XML συχνά είναι συμπληρωματική της γλώσσας σήμανσης HTML (Hypertext Markup Language)[40].

3.2.1 Δομή XML αρχείων

Η δομή ενός αρχείου XML είναι ένα δέντρο από στοιχεία (elements), οπού το πρώτο στοιχείο του αρχείου είναι το στοιχείο-ρίζας (root element). Κάθε στοιχείο ορίζεται από την ετικέτα αρχής (start tag) το περιεχόμενο και την ετικέτα τέλους (end tag). Μια ετικέτα μπορεί να περιλαμβάνει χαρακτηριστικά (attributes), ενώ το περιεχόμενο μπορεί να είναι κείμενο η αλλά στοιχεία (child elements).

3.2.2 Σύνταξη XML κείμενων

Ένα κείμενο γραμμένο σε XML μπορεί να ξεκινάει με την δήλωση πληροφορίων όπως η έκδοση και η κωδικοποίηση. Έπειτα δηλώνεται το στοιχείο-ρίζα με τα στοιχεία παιδία, καθένα από τα οποία μπορούν να έχουν χαρακτηριστικά, που δηλώνονται στην ετικέτα αρχής.

Η παρακάτω εικόνα αναπαριστά ένα παράδειγμα XML αρχείου, το οποίο αναπαριστά την οργάνωση των δεδομένων ενός βιβλιοπωλείου. Το στοιχείο-ρίζας περιγράφεται με την ετικέτα <bookstore> και τα στοιχεία-παιδία με την ετικέτα <book>. Όπως παρατηρούμε στην εικόνα κάθε βιβλίο έχει ως χαρακτηριστικό την κατηγορία στην οποία ανήκει, και περιέχει στοιχεία που περιγράφουν τον τίτλο, τον συγγραφέα το έτος έκδοσης και την τιμή του βιβλίου.

Εικόνα 3: Οργάνωση δεδομένων ενός βιβλιοπωλείου σε αρχείο XML

3.3 Android Studio

Το Android Studio είναι το επίσημο ολοκληρωμένο περιβάλλον ανάπτυξης (IDE) εφαρμογών το οποίο είναι ειδικά σχεδιασμένο για ανάπτυξη εφαρμογών Android. Είναι διαθέσιμο για συστήματα Windows, macOS και Linux. Η επίσημη ανακοίνωσή του έγινε στις 16 Μαΐου 2013 στα πλαίσια του συνέδριου Google I/O 2013 και η πρώτη σταθερή έκδοση του κυκλοφόρησε τον Δεκέμβριο του 2014[41].

Ο σχεδιασμός του Android Studio βασίστηκε στο IntelliJ IDEA, ένα ευέλικτο και δυναμικό περιβάλλον ανάπτυξης λογισμικού, με επιπρόσθετες λειτουργίες και χαρακτηριστικά οι οποίες είναι ιδιαίτερα χρήσιμες την ανάπτυξη εφαρμογών Android. Τα χαρακτηριστικά αυτά περιλαμβάνουν τον προσομοιωτή συσκευών Android (Android emulator), εφαρμογή αλλαγών κώδικα κατά τη διάρκεια εκτέλεσης χωρίς επανεκκίνηση της εφαρμογής υποστήριξη C++, δείγματα κώδικα αντλούμενα από την πλατφόρμα GitHub αλλά και σύγχρονα frameworks που σχετίζονται με την δοκιμή λοιμικού (π.χ. JUnit)[42].

3.3.1 Δομή μιας εργασίας

Μια εργασία στο περιβάλλον του Android Studio περιλαμβάνει μια ή περισσότερες ενότητες με έγγραφα πηγαίου κώδικα και έγγραφα πόρων. Οι ενότητες αυτές ανήκουν στις εξής κατηγορίες:

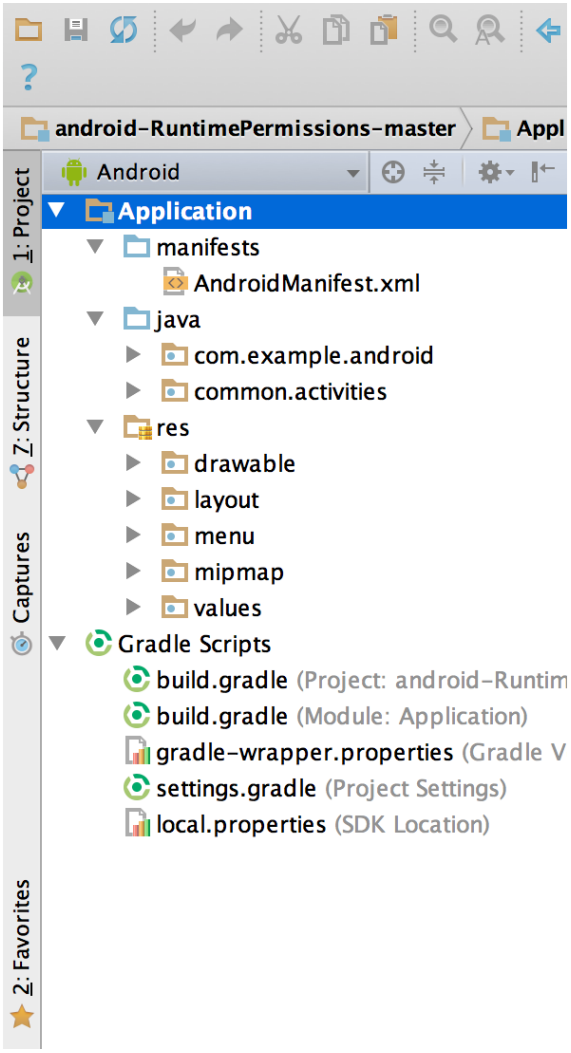
- ενότητες που αναφέρονται στην εφαρμογή (Android app modules)
- ενότητες που σχετίζονται με βιβλιοθήκες (Library module)
- ενότητες της μηχανής εφαρμογών της Google (Google app engine module)

Από προεπιλογή, το Android Studio προβάλει τα έγγραφα μια εργασίας στο αριστερό τμήμα της διεπαφής του περιβάλλοντος, το οποίο ονομάζεται Android project view. Τα έγγραφα είναι οργανωμένα σε ενότητες, ούτως ώστε η πρόσβαση σε αυτά να είναι ταχύτερη.

Τα αρχεία οικοδόμησης του λογισμικού ανήκουν στην ενότητα Gradle Scripts. Η ενότητα Application αποτελείται από τους παρακάτω φακέλους:

- Ο φάκελος manifest στον οποίο είναι αποθηκευμένο το αρχείο manifest
- Ο φάκελος java ο οποίος περιέχει τα αρχεία πηγαίου κώδικα σε γλώσσα Java καθώς και αρχεία JUnit.
- Ο φάκελος res ο οποίος περιέχει όλους τος πόρους μιας εφαρμογής όπως αρχεία XML, UI Strings και εικόνες bitmap.

Η παρακάτω εικόνα απεικονίζει την δομή μιας εργασίας στο Android Studio



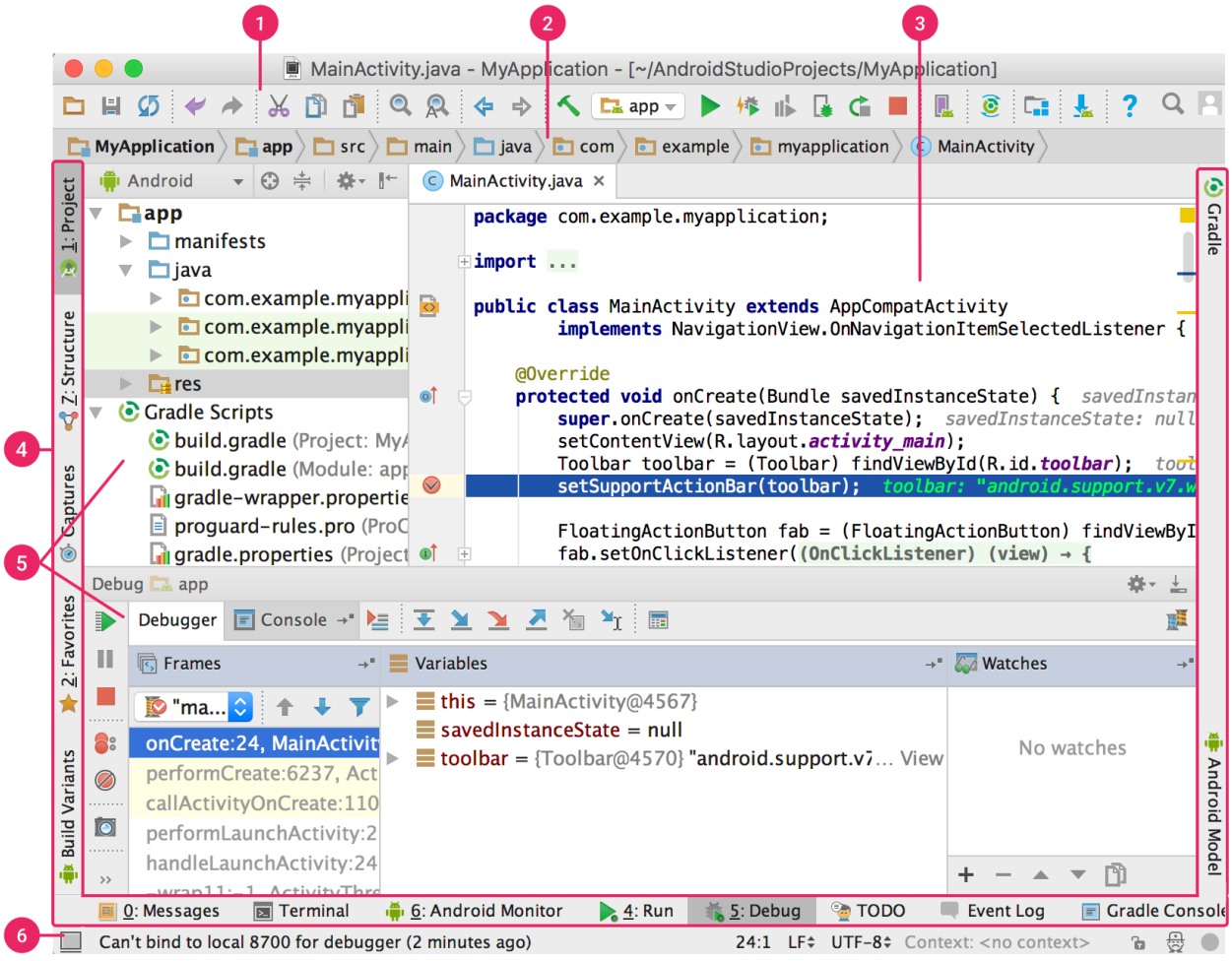
Εικόνα 4: Τα αρχεία μια εργασίας στο Android Studio[43]

3.3.2 Η διεπαφή χρήστη

Σύμφωνα με την εικόνα 5 η διεπαφή χρήστη του Android Studio διαθέτει τα εξής χαρακτηριστικά:

1. Η γραμμή εργαλείων (Toolbar) επιτρέπει στους προγραμματιστές να εκτελούν ένα ευρύ φάσμα εργασιών όπως η εκκίνηση μιας εφαρμογής, μορφοποίηση κειμένου κ.α.

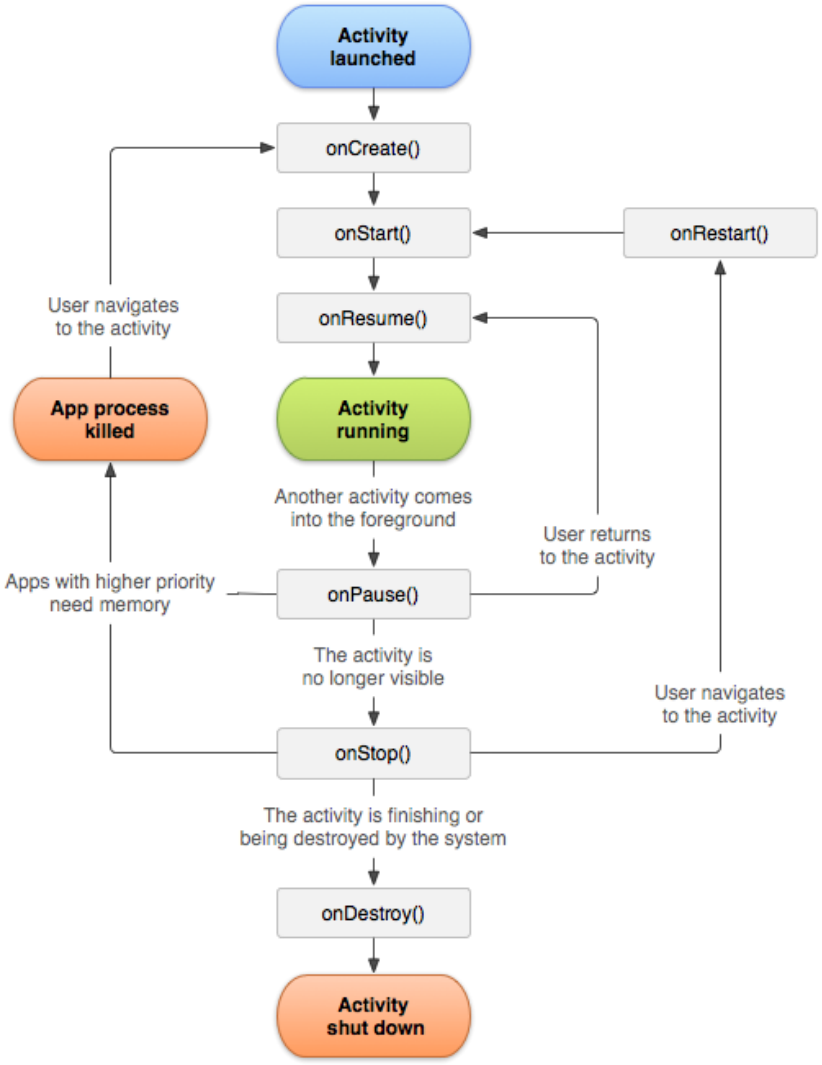
2. Η γραμμή πλοήγησης (Navigation Bar) διευκολύνει την πρόσβαση στα έγγραφα ενός project. Επίσης είναι μια συμπαγής αναπαράσταση της δομής του Project.
3. Το παράθυρο συντάκτη (Editor Window) αποτελεί το πλαίσιο όπου γίνεται η δημιουργία και η τροποποίηση του κώδικα. Τα χαρακτηριστικά και οι λειτουργιές του συντάκτη αλλάζουν ανάλογα με τον τύπο του εγγράφου που χρησιμοποιείται.
4. Η γραμμή παραθύρου εργαλείων (Tool Window Bar) είναι πλαίσιο που αφορά την χρήση παραθύρων εργαλείων του Android Studio.
5. Τα παράθυρα εργαλείων δίνουν στους προγραμματιστές πρόσβαση σε συγκεκριμένες εργασίες όπως η διαχείριση του project, αναζήτηση κτλ.
6. Η γραμμή κατάστασης αναπαριστά την κατάσταση του project και του ίδιου του περιβάλλοντος ανάπτυξης, εμφανίζοντας μηνύματα ή προειδοποιήσεις.



Εικόνα 5: Η διεπαφή χρήστη του Android Studio[44]

3.3.3 Δημιουργία Activities

Λόγω του περιορισμένου χώρου της οθόνης μια συσκευής smartphone κάθε λειτουργία ή ένα περιορισμένο σύνολο λειτουργιών μιας εφαρμογής θα πρέπει να αντιστοιχεί σε μια δραστηριότητα ή activity. Τα activities επικοινωνούν μεταξύ τους μέσω μηνυμάτων τα οποία υλοποιούνται από την κλάση Intent. Η διεπαφή ενός activity σχεδιάζεται σε κώδικα xml και αποτελείται από μια ιεραρχία στοιχείων που τα οποία ονομάζονται view groups ή views. Τα views είναι στοιχεία τα οποία αντιπροσωπεύουν δομικά συστατικά μιας διεπαφής όπως για παράδειγμα κείμενα εικόνες, κουμπιά, λίστες κ.α. Η λογική και λειτουργικότητα των συστατικών της διεπαφής υλοποιείται σε κώδικα java. Ο κώδικας αυτός υλοποιεί την λογική της κλάσης Activity η οποία περιέχει μεθόδους που σχετίζονται με τον κύκλο ζωής ενός activity. Ο κύκλος ζωής ενός activity απεικονίζεται στην παρακάτω εικόνα



Εικόνα 6: Ο κύκλος ζωής μιας δραστηριότητας [45]

Σύμφωνα με την εικόνα 6:

- όταν ο χρήστης ενεργοποιήσει μια εφαρμογή καλείται η μέθοδος onCreate() και το activity είναι ενεργό.

- όταν ο χρήστης μεταβεί σε μια άλλη εφαρμογή καλείται η μέθοδος onPause() και λειτουργία του τρέχοντος activity παύει προσωρινά.
- όταν ο χρήστης μεταβεί ξανά στην εφαρμογή καλείται η μέθοδος onResume() η οποία συνεχίζει την λειτουργία της εφαρμογή από το σημείο παύσης και
- όταν ο χρήστης αφαιρέσει την εφαρμογή από τη λίστα εργασιών του συστήματος τότε καλείται η μέθοδος onDestroy() και το στιγμιότυπο του activity καταστρέφεται.

3.3.3 Δημιουργία Fragments

Ένα θραύσμα ή fragment αποτελεί ένα τμήμα της διεπαφής χρήστη και έχει παρόμοια λειτουργικότητα με ένα activity. Ένα activity μπορεί να διαθέτει ένα ή περισσότερα fragments καθένα από τα οποία έχει δικό του κύκλο ζωής παρόμοιο με εκείνο ενός activity.

3.3.4 Δημιουργία προσαρμογέων (Adapters)

Ένας προσαρμογέας ή adapter αντλεί δεδομένα από μια πηγή όπως για παράδειγμα μια βάση δεδομένων sql ή από δομές δεδομένων (λίστες, ουρές, χάρτες). Ο adapter αναλαμβάνει να αναπαραστήσει αυτά τα δεδομένα σε κατάλληλα views τα οποία είναι λίστες δεδομένων παραδείγματος χάριν ListView ή RecyclerView.

3.4 Προτασιακή λογική

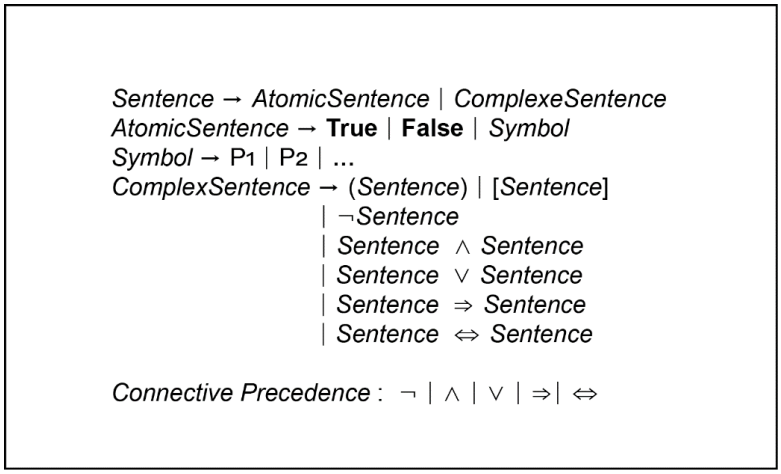
Η προτασιακή λογική (ή προτασιακός λογισμός) είναι ένας κλάδος της μαθηματικής λογικής, οποίος μελετά την αλήθεια των λογικών προτάσεων, δηλαδή εάν είναι αληθείς ή ψευδείς. Μια πρόταση μπορεί να είναι ατομική πρόταση (ή ατομικός τύπος) καθώς και σύνθετη πρόταση (ή σύνθετος τύπος) η οποία αποτελείται από ατομικές προτάσεις και λογικούς συνδέσμους.

3.4.1 Σύνταξη και γραμματική της προτασιακής λογικής

Τα σύμβολα που χρησιμοποιούνται στην προτασιακή λογική είναι τα εξής:

- Οι σταθερές True και False
- Προτασιακά σύμβολα (Proposition symbols)
- Οι λογικοί σύνδεσμοι (Logical Connectives)
- Παρενθέσεις (,)

Η γραμματική που απεικονίζεται στην εικόνα που ακολουθεί, είναι της μορφής BNF (Backus–Naur form[]) και ορίζει τις καλά ορισμένες προτάσεις (well-formed sentences) της προτασιακής λογικής.



Εικόνα 7: Η γραμματική BNF της προτασιακής λογικής [46]

Σύμφωνα με την παραπάνω γραμματική μια πρόταση μπορεί να είναι είτε ατομική είτε σύνθετη. Μια ατομική πρόταση μπορεί να είναι True όταν η πρόταση είναι πάντα αληθής και False όταν είναι πάντα ψευδής. Σε κάθε άλλη περίπτωση μια ατομική πρόταση είναι ένα σύμβολο (προτασιακό σύμβολο). Οι λογικοί σύνδεσμοι που χρησιμοποιούνται για την δημιουργία σύνθετων προτάσεων έχουν την εξής σειρά προτεραιότητας:

Προτεραιότητα	Σύνδεσμος	Σύμβολο	Έννοια στα Ελληνικά	Έννοια στα Αγγλικά
1	Άρνηση	¬	όχι	not, it is not the case that
2	Σύζευξή	∧	και	and
3	Διάζευξη	∨	ή	or
4	Συνεπαγωγή	→	εάν…τότε	if...then
5	Ισοδυναμία	↔	εάν και μόνο εάν	If and only if

Πίνακας 1: Προτεραιότητα και σημασιολογία των λογικών συνδέσμων

Το ακόλουθο παράδειγμα περιγράφει μια πρόταση σε προτασιακή λογική

P → Q (1)

Εάν αντικαταστήσουμε τις ατομικές προτάσεις P και Q με τις προτάσεις:

- P : Ο καιρός είναι βροχερός
- Q: Έχει συννεφιά

Τότε η πρόταση (1) μεταφράζεται σε

Εάν ο καιρός είναι βροχερός, τότε έχει συννεφιά

3.4.2 Πίνακες Αληθείας

Οι πίνακες αλήθειας εκφράζουν την αλήθεια των προτάσεων. Μια πρόταση, ατομική ή σύνθετη, μπορεί να είναι είτε αληθής (True) ή ψευδής (False). Ο παρακάτω πίνακας αληθείας εκφράζει την αλήθεια των προτάσεων που σχηματίζονται από τους λογικούς συνδέσμους.

P	Q	¬P	P ∧ Q	P ∨ Q	P →Q	P →Q
T	T	F	T	T	T	T
T	F	F	F	T	F	F
F	T	T	F	T	T	F
F	F	T	F	F	F	T

Πίνακας 2: Αληθείας των Προτασιακών Συνδέσμων

Κάθε τιμή αληθείας που καταχωρείται σε μια πρόταση είναι μια ερμηνεία αυτής της πρότασης. Εάν μια πρόταση είναι αληθής για κάθε πιθανή ερμηνεία της τότε θεωρείται ταυτολογία, ενώ εάν είναι ψευδής για όλες τις ερμηνείες της ονομάζεται αντίφαση.

3.4.3 Στοιχειώδεις τύποι (Literals)

Ένας στοιχειώδης τύπος (literal) είναι προτασιακό σύμβολο ή άρνηση του, δηλαδή μπορεί να είναι θετικός είτε αρνητικός στοιχειώδης τύπος. Για παράδειγμα ο στοιχειώδης τύπος P είναι θετικός ενώ ο στοιχειώδης τύπος $\neg Q$ είναι αρνητικός.

3.4.4 Απλοποίηση σύνθετων προτάσεων

Οι σύνθετες προτάσεις απλοποιούνται με την χρήση λογικών ισοδυναμιών (logical equivalences[]. Οι νόμοι της προτασιακής λογικής εφαρμόζονται όταν μια σύνθετη πρόταση ισοδυναμεί λογικά με μια απλούστερη πρόταση. Οι λογικές ισοδυναμίες και οι αντίστοιχοι νομοί που εφαρμόζονται σε αυτές, φαίνονται στον παρακάτω πίνακα.

Ισοδυναμία	Νόμος
¬¬P ⇔ P	Νόμος της διπλής άρνησης
P ∨ Q ⇔ Q ∨ P	Αντιμεταθετικός νομός (Διάζευξη)
P ∧ Q ⇔ Q ∧ P	Αντιμεταθετικός νομός (Σύζευξη)
P ∨ (Q ∨ R) ⇔ (P ∨ Q) ∨ R	Προσεταιριστικός νομός (Διάζευξη)
P ∧ (Q ∧ R) ⇔ (P ∧ Q) ∧ R	Προσεταιριστικός νομός (Σύζευξη)
P ∨ (Q ∧ R) ⇔ (P ∨ Q) ∧ (P ∨ R)	Επιμεριστικός νόμος (Διάζευξη)
P ∧ (Q ∨ R) ⇔ (P ∧ Q) ∨ (P ∧ R)	Επιμεριστικός νόμος (Σύζευξη)
P ∨ P ⇔ P	Νόμος απορρόφησης (Διάζευξη)
P ∧ P ⇔ P	Νόμος απορρόφησης (Σύζευξη)

$P \rightarrow Q \Leftrightarrow \neg P \vee Q$	Νόμος αντικατάστασης συνεπαγωγής
$P \leftrightarrow Q \Leftrightarrow (P \rightarrow Q) \wedge (Q \rightarrow P)$	Νόμος αντικατάστασης ισοδυναμίας
$P \vee \neg P \Leftrightarrow T$	Αποκλεισμός τρίτου
$P \rightarrow Q \Leftrightarrow \neg Q \rightarrow \neg P$	Νόμος αντιθετοαντιστροφής
$P \Leftrightarrow P$	Νόμος ταυτότητας
$P \wedge \neg P \Leftrightarrow F$	Νόμος αντίφασης
$P \wedge Q \rightarrow R \Leftrightarrow P \rightarrow (Q \rightarrow R)$	Νόμος εξαγωγής
$\neg(P \vee Q) \Leftrightarrow \neg P \wedge \neg Q$	Νόμος De Morgan (Διάζευξη)
$\neg(P \wedge Q) \Leftrightarrow \neg P \vee \neg Q$	Νόμος De Morgan (Σύζευξη)
$\neg(P \rightarrow Q) \Leftrightarrow P \wedge \neg Q$	Νόμος άρνησης συνεπαγωγής
$\neg(P \leftrightarrow Q) \Leftrightarrow P \leftrightarrow \neg Q$	Νόμος άρνησης ισοδυναμίας

Πίνακας 3: Λογικές ισοδυναμίες και νόμοι της προτασιακής λογικής

3.4.5 Διαζευκτική και συζευκτική κανονική μορφή

Μια πρόταση είναι σε διαζευκτική κανονική μορφή (disjunctive normal form), εάν είναι σύζευξη στοιχειωδών τύπων (literals) ή διάζευξη σύνθετων προτάσεων οι οποίες είναι σύζευξη στοιχειωδών τύπων. Για παράδειγμα οι παρακάτω προτάσεις είναι σε διαζευκτική κανονική μορφή:

- $P \wedge Q$
- $(P \wedge Q) \vee R$
- $(P \wedge \neg Q) \vee (R \wedge S \wedge \neg T)$

Μια πρόταση είναι σε συζευκτική κανονική μορφή (conjunctive normal form), εάν είναι διάζευξη στοιχειωδών τύπων (literals) ή σύζευξη σύνθετων προτάσεων οι οποίες είναι διάζευξη στοιχειωδών τύπων. Για παράδειγμα οι παρακάτω προτάσεις είναι σε συζευκτική κανονική μορφή:

- $P \vee Q$
- $(P \vee Q) \wedge R$
- $(P \vee \neg Q) \wedge (R \vee S \vee \neg T)$

3.4.6 Προτάσεις Horn

Εάν ένας τύπος είναι σε συζευκτική κανονική μορφή, δηλαδή αποτελείται από διαζεύξεις στοιχειωδών τύπων (literals) ή τις αρνήσεις τους, τότε το σύνολο των τύπων αυτών μπορεί να

ομαδοποιηθεί σε ένα σύνολο διαζεύξεων. Ένας τύπος αυτής της μορφής ονομάζεται πρόταση (clause), ενώ το σύνολο διαζεύξεων ονομάζεται προτασιακή μορφή του τύπου αυτού. Για παράδειγμα ο τύπος (clause)

$$P \vee Q \vee \neg R$$

μπορεί να μετατραπεί στην προτασιακή μορφή

$$\{P, Q, \neg R\}$$

Μια πρόταση Horn είναι μια διάζευξη στοιχειωδών τύπων η οποία έχει το πολύ έναν θετικό στοιχειώδη τύπο. Για παράδειγμα η πρόταση

$$P_1 \wedge P_2 \wedge P_3 \wedge \ldots \wedge P_n \rightarrow Q$$

ή ισοδύναμα

$$\neg P_1 \vee \neg P_2 \vee \neg P_3 \vee \ldots \vee \neg P_n \vee Q$$

είναι πρόταση Horn. Οπότε η πρόταση $\{\neg P_1 , \neg P_2 , \neg P_3, \ldots ,\neg P_n, Q \}$ περιέχει μόνο έναν θετικό στοιχειώδη τύπο, τον Q.

3.4.7 Εξαγωγή συμπερασμάτων

Η εξαγωγή συμπερασμάτων προκύπτει από προτάσεις που λέγονται συλλογισμοί. Η ορθότητα ενός συμπεράσματος αποδεικνύεται μέσω της λογικής συνέπειας, δηλαδή της σχέσης που ισχύει μεταξύ ενός συνόλου προτάσεων $P_1, P_2, P_3, \ldots, P_n$ και μιας πρότασης Q. Μια λογική συνέπεια έχει την μορφή

$$P_1, P_2, P_3, \ldots, P_n \vdash Q,$$

όπου $P_1, P_2, P_3, \ldots, P_n$ είναι οι υποθέσεις και Q είναι το συμπέρασμα.

Χρησιμοποιώντας πίνακες αληθείας μπορούμε να αποδείξουμε την ορθότητα μιας λογικής συνέπειας όπως η $P_1, P_2, P_3, \ldots, P_n \vdash Q$, δείχνοντας ότι η πρόταση $P_1 \wedge P_2 \wedge P_3 \wedge \ldots \wedge P_n \rightarrow Q$ είναι ταυτολογία. Ο παρακάτω πίνακας αποδεικνύει την ορθότητα του συμπερασματικού κανόνα απόσπασης, ο περιγράφεται με τη λογική συνέπεια $P, P \rightarrow Q \vdash Q$.

P	**Q**	**P → Q**	**P ∧ (P → Q)**	**(P ∧ (P → Q)) → Q**
T	T	T	T	T
T	F	F	F	T
F	T	F	F	T
F	T	T	F	T

***Πίνακας 4: Ορθότητα κανόνα απόσπαση*ς.**

Μια άλλη μέθοδος εξαγωγής συμπερασμάτων είναι η εφαρμογή συμπερασματικών κανόνων. Οι συμπερασματικοί κανόνες είναι προδιαγεγραμμένες λογικές συνέπειες και χρησιμοποιούνται

όταν ένα σύνολο προτάσεων ταιριάζουν με τις υποθέσεις ενός κανόνα με σκοπό την εξαγωγή συμπερασμάτων. Επιπλέον επιλύουν το πρόβλημα που προκύπτει από τους πίνακες αλήθειας όταν ο αριθμός των μεταβλητών αυξάνεται.

Κανόνας	Ονομασία
$P, P \rightarrow Q \vdash Q$	Κανόνας απόσπασης (modus ponens)
$\neg Q, P \rightarrow Q \vdash \neg Q$	Κανόνας modus tollens
$P, Q \vdash P \wedge Q$	Κανόνας εισαγωγής σύζευξης
$P \wedge Q \vdash P$	Κανόνας διαγραφής σύζευξης
$P \rightarrow Q, Q \rightarrow R \vdash P \rightarrow R$	Υποθετικός συλλογισμός
$P \vee Q, \neg P \vdash Q$	Διαζευκτικός συλλογισμός
$P \rightarrow Q, \neg P \rightarrow Q \vdash Q$	Νόμος των περιπτώσεων

Πίνακας 5: Συμπερασματικοί κανόνες

3.4.8 Συστήματα Βασισμένα σε Κανόνες (Rule-Based Systems)

Τα συστήματα κανόνων είναι μια κατηγορία συστημάτων βασισμένων στη γνώση (Knowledge-Based Systems), όπου η γνώση που αφορά την επίλυση ενός προβλήματος είναι εκφρασμένη υπό την μορφή έγκυρων συλλογισμών. Τα συστήματα τα οποία χρησιμοποιούν κανόνες για την αναπαράσταση της γνώσης ονομάζονται συστήματα βασισμένα σε κανόνες (Rule-Based Systems)[47][48].

Κάθε κανόνας αποτελείται από δύο μέρη

- Το IF μέρος το οποίο περιέχει τις υποθέσεις (premises) ή συνθήκες (conditions)
- Το THEN μέρος το οποίο περιέχει το συμπέρασμα.

Η βασική σύνταξη ενός κανόνα είναι

IF <premise>
THEN <conclusion>

Εάν υπάρχουν περισσότερες από μια υποθέσεις τότε χρησιμοποιούμε την λέξη AND η οποία εκφράζει την σύζευξη δύο υποθέσεων και την λέξη OR για διάζευξη. Παρότι είναι εφικτός ο συνδυασμός των δυο συνδέσμων, δεν θεωρείται καλή πρακτική, οπότε είναι προτιμότερο να αποφεύγεται. Ακολουθούν παραδείγματα:

IF <premise$_1$>
AND <premise$_2$>

.
.
AND <premise$_n$>
THEN <conclusion>

IF <premise$_1$>
OR <premise$_2$>
.
.
OR <premise$_n$>
THEN <conclusion>

3.4.9 Η προς τα εμπρός αλυσιδωτή εκτέλεση κανόνων (Forward Chaining)

Η προς τα εμπρός αλυσιδωτή εκτέλεση κανόνων (forward chaining) είναι μια παραγωγική διαδικασία εξαγωγής συμπερασμάτων από κανόνες σύμφωνα με το μοντέλο έγκυρων συλλογισμών modus ponens, μίας από τις δυο μορφές έγκυρων συλλογισμών του προτασιακού λογισμού που χρησιμοποιεί την παραγωγή (deduction) για την εξαγωγή συμπερασμάτων[49].

Ακολουθεί παράδειγμα συλλογισμού με modus ponens:

AN A TOTE B

Ισχύει το Α επομένως ισχύει το Β

Η δομή του παραπάνω κανόνα έχει όμοια δομή με μια πρόταση Horn, οπότε ο παραπάνω κανόνας μπορεί να εκφραστεί ως η πρόταση

$$A \rightarrow B$$

ή ισοδύναμα

$$\neg A \vee B$$

3.5 Κατανεμημένα συστήματα

Ένα κατανεμημένο σύστημα είναι ένα σύστημα το οποίο αποτελείται από ένα σύνολο υπολογιστών, οι οποίοι επικοινωνούν και συντονίζουν τις ενέργειές τους μεταφέροντας μηνύματα μεταξύ τους. Οι υπολογιστές αλληλοεπιδρούν μεταξύ τους προκειμένου να επιτευχθεί ένας κοινός στόχος όπως για παράδειγμα η επίλυση υπολογιστικών προβλημάτων. Παραδείγματα κατανεμημένων συστημάτων συναντώνται σε διαδικτυακά βιντεοπαιχνίδια ή εφαρμογές Peer-to-Peer δικτύων

3.6 Peer-to-Peer δίκτυα

Ένα δίκτυο peer-to-peer είναι μια αρχιτεκτονική κατανεμημένου συστήματος η οποία αποτελείται από δύο ή περισσότερους υπολογιστές ή κόμβους. Οι κόμβοι του δικτύου μοιράζονται μεταξύ τους πόρους όπως επεξεργαστική ισχύ, αποθηκευτικός χώρος και εύρος ζώνης, χωρίς να είναι απαραίτητη η παρουσία ενός διακομιστή (server). Είναι ταυτόχρονα και προμηθευτές και καταναλωτές πόρων σε αντίθεση με το συμβατικό μοντέλο πελάτη-διακομιστή (client-server model)

3.6.1 Η βιβλιοθήκη Google Nearby Connections API

Η υλοποίηση του peer-to-peer δικτύου το οποίο χρησιμοποιείται στην εφαρμογή Group Reasoning βασίζεται στην βιβλιοθήκη Google Nearby Connections η οποία αναπτύχθηκε από την Google. Η βιβλιοθήκη αυτή διευκολύνει την ανακάλυψη συσκευών την διασύνδεση και την μεταφορά δεδομένων μεταξύ συσκευών smartphone ανεξαρτήτως συνδεσιμότητας δικτύου ή έκδοσης του λογισμικού.

Επίσης η βιβλιοθήκη αυτή παρέχει τη δυνατότητα μεταφοράς κρυπτογραφημένων δεδομένων στα πλαίσια ενός υψηλού εύρους ζώνης και με μικρή καθυστέρηση χωρία να υπάρχει σύνδεση στο internet. Μπορεί να μεταφέρει ένα σύνολο από bytes, αρχεία και ροές δεδομένων.

Χρησιμοποιεί τις τεχνολογίες Wi-Fi, Bluetooth και Bluetooth Low Energy, ούτως ώστε να ανακαλύπτει και να δημιουργεί συνδέσεις με τις κοντινότερες συσκευές smartphone. Απλοποιεί την πολυπλοκότητα αυτών των τεχνολογιών αξιοποιώντας τις δυνάμεις της καθεμίας και παρακάμπτει τις αντίστοιχες αδυναμίες τους[50].

Όσον αφορά την τοπολογία του δικτύου η βιβλιοθήκη παρέχει τη δυνατότητα χρήσης δύο ειδών τοπολογίας την αστεροειδή τοπολογία (P2P_STAR) και την τοπολογία συστάδας (P2P_CLUSTER).

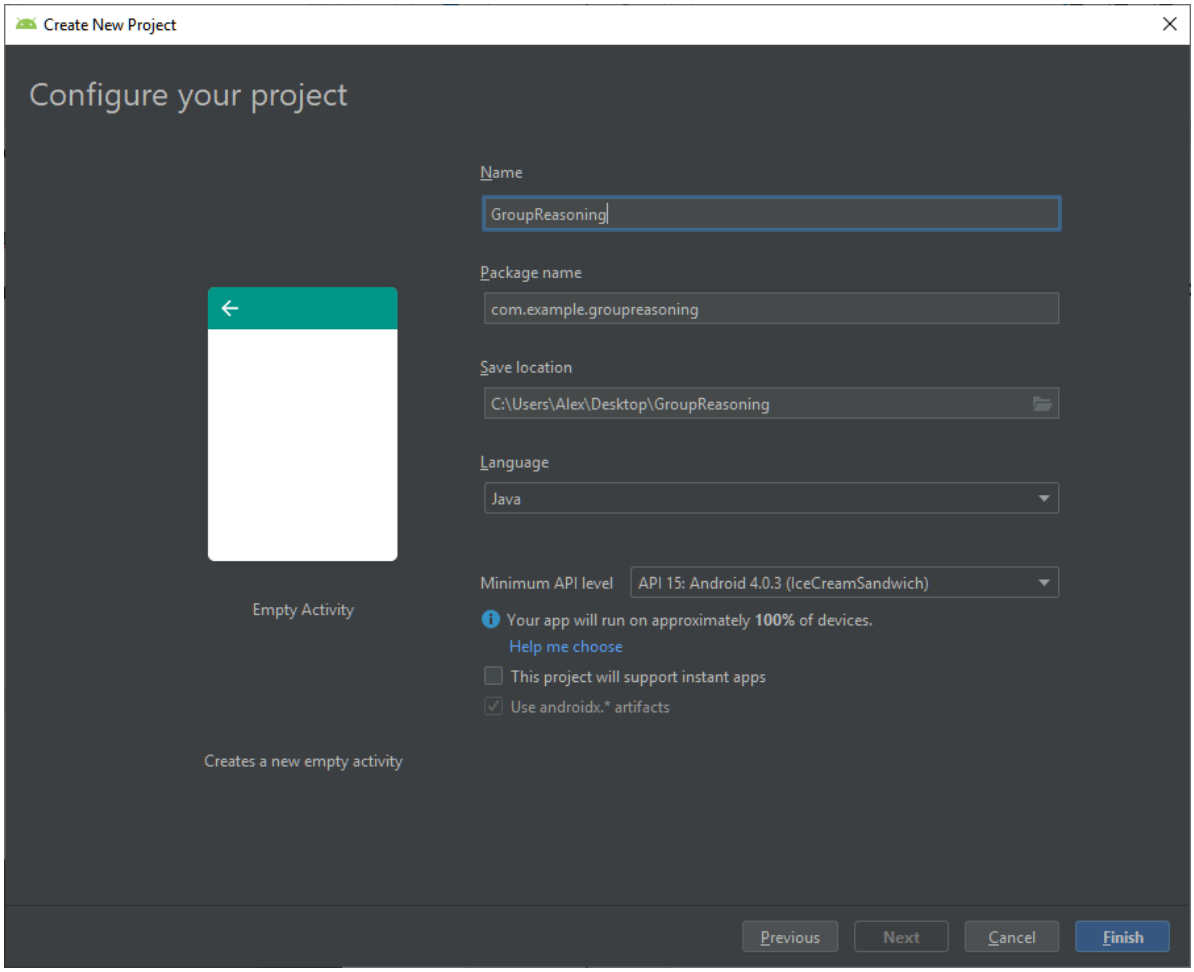
3.7 Συστήματα πολλαπλών πρακτόρων

Στον τομέα της τεχνικής νοημοσύνης ο όρος ευφυής πράκτορας αναφέρεται σε μια οντότητα η οποία στοχεύει στην πραγματοποίηση στόχων στο πλαίσιο ενός περιβάλλοντος με τη χρήση αισθητήρων ή τη χρήση εσωτερικής γνώσης. Ένα σύστημα πολλαπλών πρακτόρων αποτελείται από εάν σύνολο νοήμων πρακτόρων οι οποίοι συνεργάζονται για την επίλυση προβλημάτων τα οποία ένας μεμονωμένος πράκτορας αδυνατεί να επιλύσει.

4 Υλοποίηση της εφαρμογής Group Reasoning

Σε αυτό το κεφάλαιο θα αναλυθούν τα βήματα υλοποίησης της εφαρμογής Group Reasoning. Η εφαρμογή υλοποιήθηκε στο προγραμματιστικό περιβάλλον Android studio με τη χρήση Java και xml.

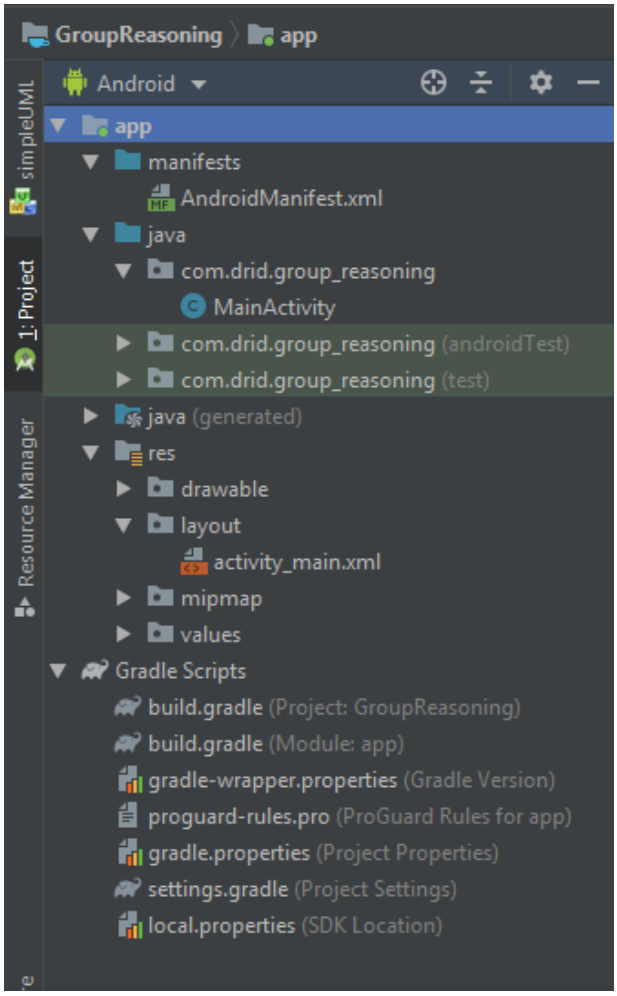
Η δημιουργία ενός νέου project στο Android Studio απεικονίζεται στην παρακάτω εικόνα. Στο παράθυρο δημιουργίας ενός project δηλώνουμε το όνομα της εφαρμογής, το πακέτο των αρχείων πόρων, την γλώσσα προγραμματισμού (Java ή Kotlin) και το κατώτερο επίπεδο του Android API που θα υποστηρίζει την εφαρμογή. Για την ολοκλήρωση της διαδικασίας πατάμε το κουμπί Finish.



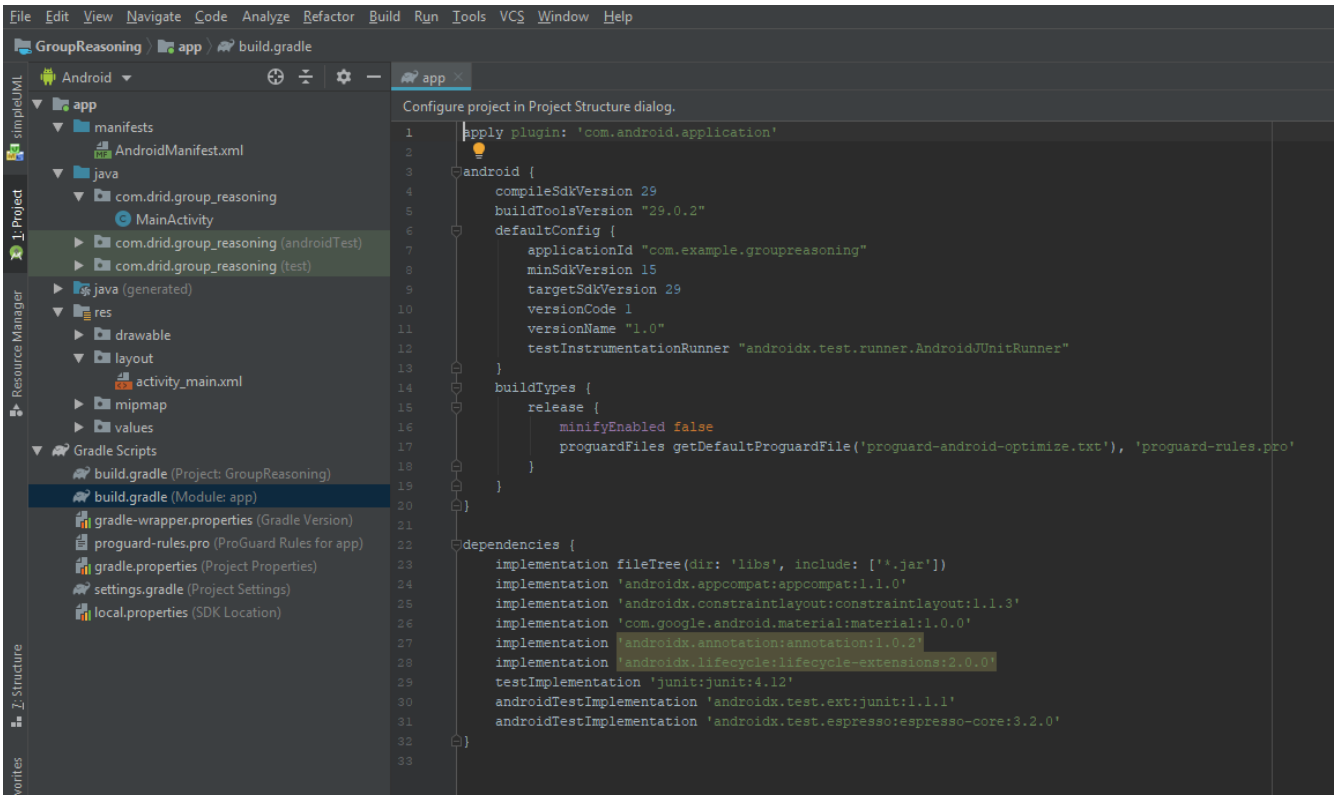
Εικόνα 8: Δημιουργία του Group Reasoning στο Android Studio

Με το πέρας της παραπάνω διαδικασίας το Android Studio μεταβαίνει στο κύριο παράθυρο του περιβάλλοντος ενώ ταυτόχρονα ξεκινά η διαδικασία δημιουργίας όλων των απαραίτητων αρχείων που θα χρειαστούν για την υλοποίηση της εφαρμογής. Τα αρχεία αυτά οργανώνονται στους εξής φακέλους

- **manifest** : Ο φάκελος αυτός περιλαμβάνει το αρχείο manifest.xml
- **java**: Ο φάκελος αυτός περιέχει 3 υποφακέλους. Στον πρώτο φάκελο βρίσκονται τα αρχεία Java τα οποία υλοποιούν την λογική της εφαρμογής ενώ στους άλλους 2 βρίσκονται τα αρχεία test που υλοποιεί το εργαλείων JUnit.
- **res:** Ο φάκελος αυτός περιλαμβάνει αρχεία πόρων όπως σχεδιασμό διεπαφής (layout), εικόνες (drawable), επιλογές του menu, animations κ.α. τα οποία είναι οργανωμένα σε αντίστοιχους υποφακέλους.
- **Gradle Scripts:** Ο φάκελος αυτός περιέχει τα αρχεία που σχετίζονται με τις ρυθμίσεις της εφαρμογής όπως οι εκδόσεις που την υποστηρίζουν, οι βιβλιοθήκες που χρησιμοποιούνται.



Εικόνα 9: Η δομή του project Group Reasoning



Εικόνα 10: Το αρχείο build.gradle

4.1 Ο μηχανισμός εξαγωγής συμπερασμάτων

Ο μηχανισμός αποτελείται από έναν συντακτικό αναλυτή προτάσεων εκφρασμένο σε προτασιακή λογική και από κλάσεις που υλοποιούν την μέθοδο της προς τα εμπρός αλυσιδωτής εκτέλεσης κανόνων (forward chaining). Η μέθοδος αυτή βασίζεται στον αλγόριθμο[51] που ακολουθεί:

function PL-FC-ENTAILS?(KB, q) **returns** true or false
inputs: KB, the knowledge base, a set of propositional definite clauses
q, the query, a proposition symbol
count ←a table, where count [c] is the number of symbols in c's premise
inferred ←a table,where inferred[s] is initially false for all symbols
agenda ←a queue of symbols, initially symbols known to be true in KB
while agenda is not empty **do**
p←POP(agenda)
if p = q **then return** true
if inferred[p] = false **then**
inferred[p]←true
for each clause c in KB where p is in c.PREMISE **do**
decrement count [c]
if count [c] = 0 **then** add c.CONCLUSION to agenda
return false

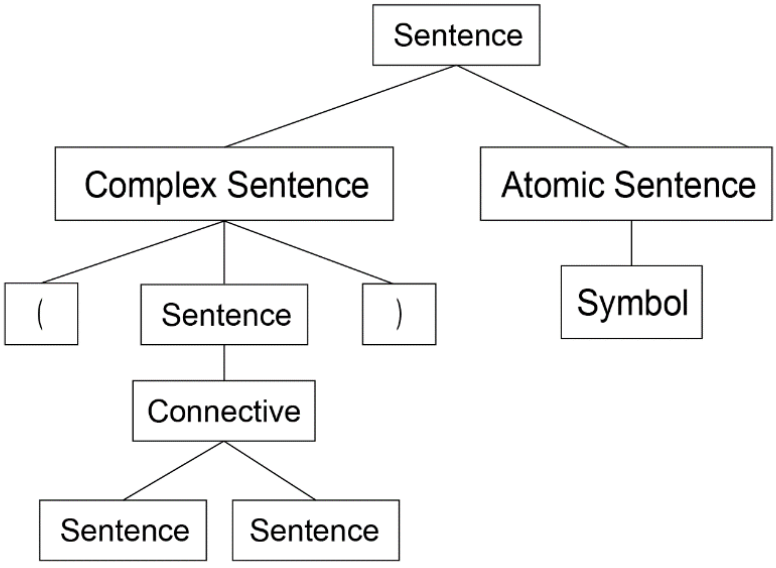
4.1.1 Ο συντακτικός αναλυτής λογικών προτάσεων

Η εφαρμογή Group Reasoning υλοποιεί έναν συντακτικό αναλυτή αναδρομικής κατάβασης (recursive descent parser) ο οποίος αρχικά μετατρέπει μια συμβολοσειρά που αντιπροσωπεύει μια πρόταση σε μια ακολουθία από λεκτικές μονάδες (tokens) και έπειτα παράγει ένα συντακτικό δέντρο, εάν οι λεκτικές μονάδες είναι σύμφωνες με μια ορισμένη γραμματική.

Η γραμματική την οποία χρησιμοποιεί ο συντακτικός αναλυτής που υλοποιείται στην εφαρμογή Group Reasoning, βασίζεται στην γραμματική που έχει οριστεί στην εικόνα 3.6 και αποτελείται από ένα σύνολο κανόνων που καθορίζει την δομή μιας πρότασης.

Ο συντακτικός αναλυτής αναγνωρίζει πια σύμβολα της πρότασης που αναλύει είναι τερματικά και ποια είναι μη τερματικά. Τερματικά σύμβολα είναι σύμβολα στα οποία δεν μπορεί να γίνει μετατροπή και δεν μπορεί να εφαρμοστεί ένας κανόνας της γραμματικής. Μη τερματικά σύμβολα είναι σύμβολα τα οποία χρειάζονται περαιτέρω ανάλυση και σε αυτά εφαρμόζονται κανόνες της γραμματικής. Κάθε κανόνας της γραμματικής εκφράζει τον τρόπο με τον οποίο ένα μη τερματικό σύμβολο μετατρέπεται σε μια ακολουθία από τερματικά και μη τερματικά σύμβολα. Σε κάθε μη τερματικό σύμβολο αντιστοιχεί ένας γραμματικός κανόνας ενώ κάθε λεκτική μονάδα αντιπροσωπεύει ένα τερματικό σύμβολο της γραμματικής.

Η συντακτική ανάλυση γίνεται από τα αριστερά προς τα δεξιά και για κάθε τερματικό σύμβολο παράγεται ένας κόμβος του συντακτικού δέντρου. Τερματικά σύμβολα είναι τα σύμβολα (,) , ¬,∧,∨,→,↔ καθώς και ακολουθίες αλφαριθμητικών χαρακτήρων. Για τις παρενθέσεις δεν δημιουργείται κόμβος στο δέντρο. Κάθε φορά που ο αναλυτής εντοπίζει ένα σύμβολο (, δηλαδή αριστερή παρένθεση, τότε έχει εντοπίσει μια σύνθετη πρόταση και με βάση τον λογικό σύνδεσμο αυτής της πρότασης εφαρμόζει τον ανάλογο κανόνα γραμματικής. Γενικότερα η κατασκευή ενός συντακτικού δέντρου βασίζεται σε ένα δέντρο αφηρημένης σύνταξης (abstract syntax tree) το οποίο σχεδιάστηκε με βάση την γραμματική που έχε οριστεί. Το δέντρο αυτό παρουσιάζεται στην παρακάτω εικόνα:

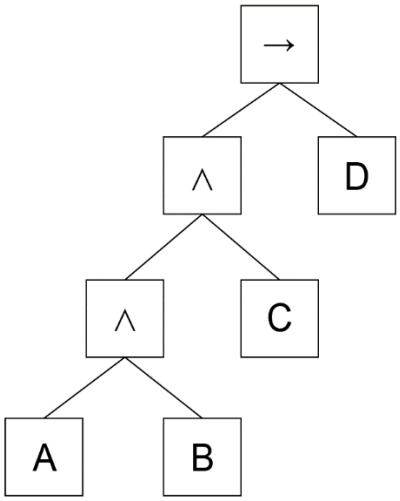


Εικόνα 11: Το δέντρο αφηρημένης σύνταξης που ορίζει την δομή μιας πρότασης στη γλώσσα της προτασιακής λογικής

Οπότε σύμφωνα με τα παραπάνω, όταν ο αναλυτής επεξεργάζεται μια πρόταση, παραδείγματος χάριν την πρόταση:

$$(A \wedge B \wedge C) \rightarrow D$$

Σχηματίζει το ακόλουθο δέντρο



Εικόνα 12: Συντακτικό δέντρο της πρότασης

Η διεπαφή SentenceNode αντιπροσωπεύει αφαιρετικά έναν κόμβο του συντακτικού δέντρου. Ορίζει την μέθοδο accept η οποία δέχεται έναν λογικό επισκέπτη. Ακολουθεί ο κώδικας του αρχείου SentenceNode.java.

```
1  package com.drid.group_reasoning.engine.parser.ast;
2
3  import com.drid.group_reasoning.engine.parser.visitors.LogicVisitor;
4
5  public interface SentenceNode {
6      <T> T accept(LogicVisitor<T> visitor);
7  }
8
```

Η αφηρημένη κλάση Sentence υλοποιεί την διεπαφή SentenceNode και κατ' επέκταση την μέθοδο accept ορίζει μεθόδους που αφορούν τη δομή μιας πρότασης. Αποτελείται από τις μεθόδους ακολουθούν παρακάτω.

Η μέθοδος isAtomicSentence() επιστέφει τιμή επιστροφής τύπου boolean και ελέγχει εάν μια πρόταση είναι ατομική, δηλαδή εάν δεν διαθέτει κάποιο λογικό σύνδεσμο.

Η μέθοδος isBiconditional() επιστρέφει τιμή επιστροφής τύπου boolean και ελέγχει εάν μια πρόταση έχει το λογικό σύνδεσμο της ισοδυναμίας.

Η μέθοδος isImplication() επιστρέφει τιμή επιστροφής τύπου boolean και ελέγχει εάν μια πρόταση έχει το λογικό σύνδεσμο της συνεπαγωγής.

Η μέθοδος isOrSentence() επιστρέφει τιμή επιστροφής τύπου boolean και ελέγχει εάν μια πρόταση έχει το λογικό σύνδεσμο της διάζευξης.

Η μέθοδος isAndSentence () επιστρέφει τιμή επιστροφής τύπου boolean και ελέγχει εάν μια πρόταση έχει το λογικό σύνδεσμο της σύζευξης.

Η μέθοδος isNotSentence () επιστρέφει τιμή επιστροφής τύπου boolean και ελέγχει εάν μια πρόταση έχει το λογικό σύνδεσμο της άρνησης.

Η μέθοδος isUnarySentence () επιστρέφει τιμή επιστροφής τύπου boolean και ελέγχει εάν μια πρόταση είναι μοναδιαία, δηλαδή αν είναι άρνηση μιας πρότασης, καλώντας την μέθοδο isNotSentence().

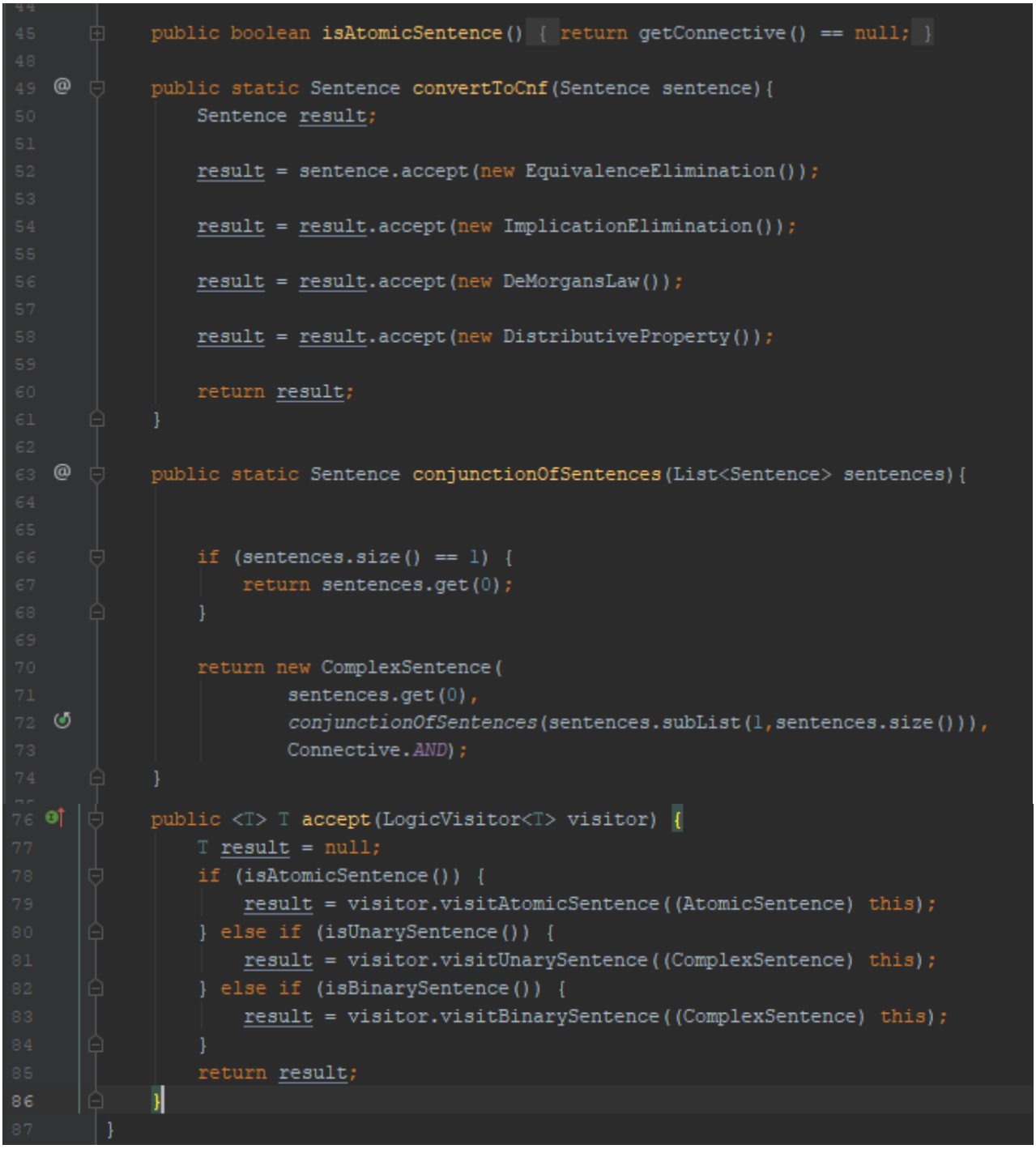
Η μέθοδος isBinarySentence () επιστρέφει τιμή επιστροφής τύπου boolean και ελέγχει εάν μια πρόταση είναι δυαδική, δηλαδή αν διαθέτει ένα λογικό σύνδεσμο εκτός εκείνον την άρνησης

Η μέθοδος converToCnf() μετατρέπει μια σύνθετη πρόταση σε κανονική συζευκτική μορφή. Ορίζει ένα στιγμιότυπο της κλάσης Sentence το οποίο δέχεται τους απαραίτητους επισκέπτες, καλώντας την μέθοδο accept()

Η μέθοδος conjunctionOfSentences() μετατρέπει μια λίστα προτάσεων σε μια σύνθετη πρόταση η οποία είναι σύζευξη των προτάσεων της λίστας αυτής.

Η μέθοδος accept έχει τιμή επιστροφής ένα γενικό τύπο (Generic Type) ο οποίος αλλάζει ανάλογα τον τύπο του επισκέπτη. Εντός της μεθόδου ελέγχεται η δομή της πρότασης και καλείται η κατάλληλη μέθοδος του επισκέπτη. Ακολουθεί ο κώδικας του αρχείου Sentence.java

```java
package com.drid.group_reasoning.engine.parser.ast;

import com.drid.group_reasoning.engine.parser.visitors.DeMorgansLaw;
import com.drid.group_reasoning.engine.parser.visitors.DistributiveProperty;
import com.drid.group_reasoning.engine.parser.visitors.EquivalenceElimination;
import com.drid.group_reasoning.engine.parser.visitors.ImplicationElimination;
import com.drid.group_reasoning.engine.parser.visitors.LogicVisitor;

import java.util.List;

public abstract class Sentence implements SentenceNode{

    public abstract String getProposition();

    public abstract Connective getConnective();

    public boolean isBiconditional() { return getConnective() == Connective.BICONDITIONAL; }

    public boolean isImplication() { return getConnective() == Connective.IMPLICATION; }

    public boolean isAndSentence() { return getConnective() == Connective.AND; }

    public boolean isOrSentence() { return getConnective() == Connective.OR; }

    public boolean isNotSentence() { return getConnective() == Connective.NOT; }

    public boolean isUnarySentence() { return isNotSentence(); }

    public boolean isBinarySentence() {
        return getConnective() != null && !(getConnective() == Connective.NOT);
    }

```



Η κλάση AtomicSentence αντιπροσωπεύει μια ατομική πρόταση. Είναι επέκταση της κλάσης Sentence και υλοποιεί τις διεπαφές Serializable και Parcelable και τις μεθόδους τους. Επίσης διαθέτει τα πεδία symbol και proposition τα οποία είναι συμβολοσειρές και αντιπροσωπεύουν το σύμβολο και την πρόταση εκφρασμένη σε φυσική γλώσσα, αντιστοίχως. Ακολουθεί ο κώδικας του αρχείου AtomicSentence.java:

```java
package com.drid.group_reasoning.engine.parser.ast;

import android.os.Parcel;
import android.os.Parcelable;

import java.io.Serializable;

public class AtomicSentence extends Sentence implements Serializable,Parcelable {

    private String symbol;
    private String proposition;

    public AtomicSentence(String symbol) {
        this.symbol = symbol;
    }

    protected AtomicSentence(Parcel in) {
        symbol = in.readString();
        proposition = in.readString();
    }

    public static final Creator<AtomicSentence> CREATOR = new Creator<AtomicSentence>() {
        @Override
        public AtomicSentence createFromParcel(Parcel in) { return new AtomicSentence(in); }

        @Override
        public AtomicSentence[] newArray(int size) { return new AtomicSentence[size]; }
    };

    public void setProposition(String proposition) { this.proposition = proposition; }

    public String getSymbol() { return symbol; }

    @Override
    public String getProposition() { return proposition; }

    @Override
    public int describeContents() { return 0; }

    @Override
    public void writeToParcel(Parcel dest, int flags) {
        dest.writeString(symbol);
        dest.writeString(proposition);
    }

    @Override
    public String toString() {
        return "{Atomic sentence: symbol=" + symbol + " , proposition=" +  proposition + "}";
    }


    @Override
    public Connective getConnective() { return null; }
68
```

```java
    @Override
    public boolean equals(Object o) {

        if (this == o) {
            return true;
        }
        if ((o == null) || (this.getClass() != o.getClass())) {
            return false;
        }
        AtomicSentence atom = (AtomicSentence) o;
        return symbol.equals(atom.symbol) && proposition.equals(atom.proposition);

    }

    @Override
    public int hashCode() { return symbol.hashCode(); }
}

```

Η απαρίθμηση (emun) Connective ξαπολείται από πέντε σταθερές τιμές που αντιστοιχούν σε σύμβολα λογικών συνδέσμων και την σειρά προτεραιότητάς τους. Ακολουθεί ο κώδικας του αρχείου Connective.java.

```java
package com.drid.group_reasoning.engine.parser.ast;

public enum Connective {
    NOT( symbol: "¬"),
    AND( symbol: "∧"),
    OR( symbol: "∨"),
    IMPLICATION( symbol: "→"),
    BICONDITIONAL( symbol: "↔");

    private String symbol;

    Connective(String symbol) { this.symbol = symbol; }

    public String getSymbol() { return symbol; }
}

```

Η κλάση ComplexSentence αντιπροσωπεύει μια σύνθετη πρόταση. Είναι επέκταση της κλάσης Sentence και διαθέτει τα πεδία a,b και connective. Τα πεδίο a και b αναφέρονται στις προτάσεις που συνθέτους την σύνθετη πρόταση ενώ το πεδίο connective αναφέρεται στον λογικό σύνδεσμο. Ακολουθεί ο κώδικας του αρχείου ComplexSentence.java:

```java
package com.drid.group_reasoning.engine.parser.ast;

public class ComplexSentence extends Sentence {

    private Sentence a;
    private Sentence b;

    private Connective connective;

    public ComplexSentence(Sentence a, Sentence b, Connective connective) {
        this.a = a;
        this.b = b;
        this.connective = connective;
    }

    public ComplexSentence(Sentence negated, Connective connective) {
        this.a = negated;
        this.connective = connective;
    }

    public Sentence getSentenceA() { return a; }

    public Sentence getSentenceB() { return b; }

    @Override
    public Connective getConnective() { return connective; }

    public String getProposition() {
        StringBuilder sb = new StringBuilder();

        if (isNotSentence()) {
            if (getSentenceA().getProposition() != null) {
                sb.append("it is not the case that")
                        .append(" ")
                        .append(getSentenceA().getProposition());
            }

        } else if (isAndSentence()) {
            if (getSentenceA().getProposition() != null && getSentenceB() != null) {
                sb.append(getSentenceA().getProposition())
                        .append(" ")
                        .append("and")
                        .append(" ")
                        .append(getSentenceB().getProposition());
            }
        } else if (isOrSentence()) {
            if (getSentenceA().getProposition() != null && getSentenceB() != null) {
```

```
                sb.append(getSentenceA().getProposition())
                        .append(" ")
                        .append("or")
                        .append(" ")
                        .append(getSentenceB().getProposition());
            }
        } else if (isImplication()) {
            if (getSentenceA().getProposition() != null && getSentenceB() != null) {
                sb.append("if")
                        .append(" ")
                        .append(getSentenceA().getProposition())
                        .append(", ")
                        .append("then")
                        .append(" ")
                        .append(getSentenceB().getProposition());
            }
        } else {
            if (getSentenceA().getProposition() != null && getSentenceB() != null) {
                sb.append(getSentenceA().getProposition())
                        .append(" ")
                        .append("if and only if")
                        .append(" ")
                        .append(getSentenceB().getProposition());
            }
        }

        return sb.toString();
    }
```

Η κλάση LogicToken περιέχει όλες τις σταθερές που περιγράφουν τις λεκτικές μονάδες (tokens) που χρησιμοποιούνται κατά τη συντακτική ανάλυση μιας λογικής πρότασης. Ακολουθεί ο κώδικας του αρχείου LogicToken.java

```
package com.drid.group_reasoning.engine.parser;

public class LogicToken {

    public static final int END_OF_INPUT = 0;
    public static final int SYMBOL = 1;
    public static final int OPEN_BRACKET = 2;
    public static final int CLOSE_BRACKET = 3;
    public static final int NOT = 4;
    public static final int AND = 5;
    public static final int OR = 6;
    public static final int IMPLICATION = 7;
    public static final int EQUIVALENCE = 8;

    public final int token;
    public final String sequence;
    public  int position;
```

```
    public LogicToken(int token, String sequence) {
        super();
        this.token = token;
        this.sequence = sequence;
    }

    public LogicToken(int token, String sequence, int position) {
        super();
        this.token = token;
        this.sequence = sequence;
        this.position = position;
    }

    public String getSequence() { return sequence; }
}
```

Η υλοποίηση της λεκτικής ανάλυσης μιας λογικής πρότασης γίνεται με τη χρήση της κλάσης LogicTokenizer. Η κλάση LogicTokenizer περιέχει τις μεθόδους οι οποίες μετατρέπουν μια συμβολοσειρά που περιγράφει μια πρόταση σε λεκτικές μονάδες. Η μετατροπή γίνεται με τη χρήση κανονικών εκφράσεων (Regular expressions) και των κλάσεων Matcher και Pattern. Η κλάση Pattern δημιουργεί με βάση τις κανονικές εκφράσεις, πρότυπα που περιγράφουν τερματικά σύμβολα της γραμματικής που έχει οριστεί. Η κλάση Matcher ελέγχει εάν τα πρότυπα ταιριάζουν με ακολουθίες χαρακτήρων της συμβολοσειράς. Ακολουθεί ο κώδικας του αρχείου LogicTokenizer.java.

```java
package com.drid.group_reasoning.engine.parser;

import com.drid.group_reasoning.engine.parser.exceptions.ParserException;

import java.util.LinkedList;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class LogicTokenizer {

    private LinkedList<TokenInfo> tokenInfoList;
    private LinkedList<LogicToken> tokens;

    public LogicTokenizer() {
        tokenInfoList = new LinkedList<>();
        tokens = new LinkedList<>();
    }

    public static LogicTokenizer getTokenizer() {
        LogicTokenizer tokenizer = new LogicTokenizer();
        tokenizer.add( regex: "[a-zA-Z][a-zA-Z0-9_]*",  token: 1); // symbol
        tokenizer.add( regex: "\\(",  token: 2); // open bracket
        tokenizer.add( regex: "\\)",  token: 3); // close bracket
        tokenizer.add( regex: "\\¬",  token: 4); // negation
        tokenizer.add( regex: "\\∧",  token: 5); // and
        tokenizer.add( regex: "\\∨",  token: 6); // or
        tokenizer.add( regex: "\\→",  token: 7); // implication
        tokenizer.add( regex: "\\↔",  token: 8); // equivalence

        return tokenizer;
    }

```

```
    public void add(String regex, int token) {
        tokenInfoList.add(
                new TokenInfo(Pattern.compile("^(" + regex + ")"), token));
    }

    public void tokenize(String str) {
        String s = str.trim();
        tokens.clear();

        while (!s.equals("")) {
            boolean match = false;

            for (TokenInfo info : tokenInfoList) {
                Matcher m = info.regex.matcher(s);

                if (m.find()) {
                    match = true;

                    String tok = m.group().trim();
                    s = m.replaceFirst( replacement: "").trim();
                    tokens.add(new LogicToken(info.token, tok));

                    break;
                }
            }
            if (!match) {
                throw new ParserException(
                        "Unexpected character in input: " + s);
            }
        }
    }
```

```
    public LinkedList<LogicToken> getTokens() { return tokens; }

    private class TokenInfo {

        public final Pattern regex;
        public final int token;

        public TokenInfo(Pattern regex, int token) {
            this.regex = regex;
            this.token = token;
        }
    }
}
```

Η κλάση LogicParser αποτελεί την υλοποίηση ενός συντακτικού αναλυτή που ειδικεύεται σε προτάσεις εκφρασμένες σε προτασιακή λογική. Περιέχει τις μεθόδους που υλοποιούν κανόνες της γραμματικής που εφαρμόζονται κατά τη συντακτική ανάλυση μιας πρότασης. Ακολουθεί ο κώδικας του αρχείου LogicParser.java

```java
package com.drid.group_reasoning.engine.parser;

import com.drid.group_reasoning.engine.parser.ast.AtomicSentence;
import com.drid.group_reasoning.engine.parser.ast.ComplexSentence;
import com.drid.group_reasoning.engine.parser.ast.Connective;
import com.drid.group_reasoning.engine.parser.ast.Sentence;
import com.drid.group_reasoning.engine.parser.exceptions.ParserException;

import java.util.LinkedList;

public class LogicParser {

    private LinkedList<LogicToken> tokens;
    private LogicToken lookahead;

    public Sentence parse(String expression) {
        LogicTokenizer tokenizer = LogicTokenizer.getTokenizer();
        tokenizer.tokenize(expression);
        LinkedList<LogicToken> tokens = tokenizer.getTokens();
        return this.parse(tokens);
    }

    private Sentence parse(LinkedList<LogicToken> tokens) {
        this.tokens = new LinkedList<>(tokens);
        lookahead = this.tokens.getFirst();
        Sentence node = sentence();

        if (lookahead.token != LogicToken.END_OF_INPUT) {
            throw new ParserException(String.format(
                    "Unexpected symbol %s found", lookahead.getSequence()));
        }
        return node;
    }

    private void nextToken() {
        tokens.pop();
        // at the end of input we return an epsilon token
        if (tokens.isEmpty()) {
            lookahead = new LogicToken(LogicToken.END_OF_INPUT, sequence: "", position: -1);
        } else {
            lookahead = tokens.getFirst();
        }
    }

    private Sentence sentence() {
        Sentence node = atomicSentence();
        return complexSentence(node);
    }

    private Sentence atomicSentence() {
        Sentence sentenceNode;
        if (lookahead.token == LogicToken.SYMBOL) {
            sentenceNode = new AtomicSentence(lookahead.getSequence());
            nextToken();
            return sentenceNode;
        }
        return null;
    }

```

```
60      private Sentence complexSentence(Sentence node) {
61          if (lookahead.token == LogicToken.OPEN_BRACKET) {
62              nextToken();
63              Sentence expr = sentence();
64              if (lookahead.token != LogicToken.CLOSE_BRACKET) {
65                  throw new ParserException("Closing brackets expected", lookahead);
66              }
67              nextToken();
68              expr = complexSentence(expr);
69              return expr;
70          } else if (lookahead.token == LogicToken.NOT) {
71              nextToken();
72              return new ComplexSentence(sentence(), Connective.NOT);
73          } else if (lookahead.token == LogicToken.AND) {
74              nextToken();
75              Sentence rightSentence = sentence();
76              return new ComplexSentence(node, rightSentence, Connective.AND);
77          } else if (lookahead.token == LogicToken.OR) {
78              nextToken();
79              Sentence rightSentence = sentence();
80              return new ComplexSentence(node, rightSentence, Connective.OR);
81          } else if (lookahead.token == LogicToken.IMPLICATION) {
82              nextToken();
83              Sentence rightSentence = sentence();
84              return new ComplexSentence(node, rightSentence, Connective.IMPLICATION);
85          } else if (lookahead.token == LogicToken.EQUIVALENCE) {
86              nextToken();
87              Sentence rightSentence = sentence();
88              return new ComplexSentence(node, rightSentence, Connective.BICONDITIONAL);
89          }
90          return node;
91      }
```

Το πρότυπο σχεδίασης «Επισκέπτης» (Visitor Design Pattern), επιτρέπει τον ορισμό μεθόδων οι οποίες αλλάζουν την συμπεριφορά των στιγμιότυπων μιας κλάσης χωρίς να τροποποιούν τον κώδικα της κλάσης. Η εφαρμογή Group Reasoning υλοποιεί «λογικούς επισκέπτες» οι οποίοι χρησιμοποιούνται για τη μετατροπή μιας σύνθετης πρότασης σε κανονική συζευκτική μορφή. Για κάθε συμπερασματικό κανόνα που πρέπει να εφαρμοστεί για τη μετατροπή έχει δημιουργηθεί ένας επισκέπτης. Επίσης υλοποιούνται επισκέπτες οι οποίοι λειτουργούν ως συλλέκτες των φύλλων ενός συντακτικού δέντρου όπως για παράδειγμα συλλέκτης ατομικών προτάσεων ή στοιχειωδών τύπων.

Η διεπαφή LogicVisitor η οποία δέχεται έναν παραμετρικό τύπο (Generic type) και ορίζει τις γενικεύσεις των παρακάτω μεθόδων:

- visitAtomicSentence(), η οποία επισκέπτεται μια ατομική πρόταση
- visitUnarySentence(), η οποία επισκέπτεται μια μοναδιαία πρόταση, δηλαδή άρνηση μιας πρότασης
- visitBInraySentence() η οποία επισκέπτεται μια δυαδική πρόταση, δηλαδή μια σύνθετη πρόταση που διαθέτει σύνδεσμο σύζευξης, διάζευξης, συνεπαγωγής ή ισοδυναμίας.

Η διεπαφή LogicVisitor υλοποιείται από κλάσεις οι οποίες είτε ομαδοποιούν κόμβος του συντακτικού δέντρου ή προσδίδουν νέα χαρακτηριστικά σε αυτούς. Οι κλάσεις που υλοποιούν τους λογικούς επισκέπτες αναλύονται παρακάτω.

Η κλάση AtomCollector αποτελεί έναν επισκέπτη, ο οποίος προσθέτει τα φύλλα ενός συντακτικού δέντρου, δηλαδή τις ατομικές προτάσεις, σε μια λίστα. Υλοποιεί την διεπαφή LogicVisitor με παραμετρικό τύπο μια λίστα με στοιχεία τα οποία είναι στιγμιότυπα της κλάσης AtomicSentence.

Η κλάση LiteralCollector είναι ένας επισκέπτης, ο οποίος διασχίζει το συντακτικό δέντρο και δημιουργεί μια λίστα με στοιχειώδους τύπους (list of literals). Υλοποιεί την διεπαφή LogicVisitor με παραμετρικό τύπο μια λίστα με στοιχεία τα οποία είναι στιγμιότυπα της κλάσης Literal.

Η κλάση ClauseCollector είναι ένας επισκέπτης, ο οποίος διασχίζει το συντακτικό δέντρο και δημιουργεί μια λίστα με προτάσεις που είναι διαζεύξεις στοιχειωδών τύπων (disjunction of literals). Υλοποιεί την διεπαφή LogicVisitor με παραμετρικό τύπο μια λίστα με στοιχεία τα οποία είναι στιγμιότυπα της κλάσης Clause.

Η αφηρημένη κλάση SimplificationVisitor υλοποιεί τη διεπαφή LogicVisitor με παραμετρικό τύπο την κλάση Sentence. Ο επισκέπτης SimplifiactionVisitor μια και αποτελεί ένα πρότυπο στο οποίο βασίζονται οι επισκέπτες που εφαρμόζουν κανόνες απλοποίησης σε μια πρόταση που χρειάζονται για τη μετατροπή της σε κανονική συζευκτική μορφή.

Η κλάση DeMorgnasLaw είναι επέκταση της κλάσης SimplificationVisitor και εφαρμόζει τον νόμο απλοποίησης De Morgan.

Η κλάση ImplicationElimination είναι επέκταση της κλάσης SimplificationVisitor εφαρμόζει τον κανόνα της απαλοιφής της συνεπαγωγής

Η κλάση EquivalenceElimination είναι επέκταση της κλάσης SimplificationVisitor εφαρμόζει τον κανόνα της απαλοιφής της ισοδυναμίας

Η κλάση DistributiveProperty είναι επέκταση της κλάσης SimplificationVisitor εφαρμόζει την επιμεριστική ιδιότητα ως προς τη σύζευξη

Η κλάση SetProposition είναι ένας επισκέπτης ο οποίος ορίζει μια έκφραση σε φυσική γλώσσα σε μια ατομική πρόταση. Για παράδειγμα ο επισκέπτης θα ορίσει στην ατομική πρόταση P την έκφραση “Earth is a planet”.

Ακολουθούν οι κώδικες των «λογικών επισκεπτών»:

```java
package com.drid.group_reasoning.engine.parser.visitors;

import com.drid.group_reasoning.engine.parser.ast.AtomicSentence;
import com.drid.group_reasoning.engine.parser.ast.ComplexSentence;


public interface LogicVisitor<T> {
    T visitAtomicSentence(AtomicSentence atomicSentence);
    T visitUnarySentence(ComplexSentence unarySentence);
    T visitBinarySentence(ComplexSentence binarySentence);
}

```

Ο κώδικας του αρχείου LogicVisitor.java

```java
package com.drid.group_reasoning.engine.parser.visitors;

import com.drid.group_reasoning.engine.parser.ast.AtomicSentence;
import com.drid.group_reasoning.engine.parser.ast.ComplexSentence;
import com.drid.group_reasoning.engine.parser.ast.Sentence;

import java.util.ArrayList;
import java.util.List;

public class AtomCollector implements LogicVisitor<List<AtomicSentence>> {

    public static List<AtomicSentence> getAtomicSentences(Sentence sentence){
        AtomCollector collector = new AtomCollector();
        return sentence.accept(collector);
    }

    @Override
    public List<AtomicSentence> visitAtomicSentence(AtomicSentence atomicSentence) {
        List<AtomicSentence> result = new ArrayList<>();
        result.add(atomicSentence);
        return result;
    }

    @Override
    public List<AtomicSentence> visitUnarySentence(ComplexSentence unarySentence) {
        List<AtomicSentence> result = new ArrayList<>();
        result.add((AtomicSentence) unarySentence.getSentenceA());
        return result;
    }

    @Override
    public List<AtomicSentence> visitBinarySentence(ComplexSentence binarySentence) {
        List<AtomicSentence> result = new ArrayList<>();
        result.addAll(binarySentence.getSentenceA().accept( visitor: this));
        result.addAll(binarySentence.getSentenceB().accept( visitor: this));
        return result;
    }

}
```

Ο κώδικας του αρχείου AtomCollector.java

```java
package com.drid.group_reasoning.engine.parser.visitors;

import com.drid.group_reasoning.engine.knowledgebase.Literal;
import com.drid.group_reasoning.engine.parser.ast.AtomicSentence;
import com.drid.group_reasoning.engine.parser.ast.ComplexSentence;
import com.drid.group_reasoning.engine.parser.ast.Sentence;

import java.util.ArrayList;
import java.util.List;

public class LiteralCollector implements LogicVisitor<List<Literal>> {

    public static List<Literal> getLiterals(Sentence sentence){
        LiteralCollector collector = new LiteralCollector();
        List<Literal> literals = sentence.accept(collector);
        return literals;
    }
```

```
    @Override
    public List<Literal> visitAtomicSentence(AtomicSentence atomicSentence) {
        List<Literal> result = new ArrayList<>();
        result.add(new Literal(atomicSentence));
        return result;
    }

    @Override
    public List<Literal> visitUnarySentence(ComplexSentence negated) {
        List<Literal> result = new ArrayList<>();
        result.add(new Literal((AtomicSentence) negated.getSentenceA(), isPositive: false));
        return result;
    }

    @Override
    public List<Literal> visitBinarySentence(ComplexSentence complexSentence) {
        List<Literal> result = new ArrayList<>();
        result.addAll(
                complexSentence.getSentenceA().accept( visitor: this));
        result.addAll(
                complexSentence.getSentenceB().accept( visitor: this));

        return result;
    }
}
```

Ο κώδικας του αρχείου LiteralCollector.java

```
package com.drid.group_reasoning.engine.parser.visitors;

import com.drid.group_reasoning.engine.knowledgebase.Clause;
import com.drid.group_reasoning.engine.knowledgebase.Literal;
import com.drid.group_reasoning.engine.parser.ast.AtomicSentence;
import com.drid.group_reasoning.engine.parser.ast.ComplexSentence;
import com.drid.group_reasoning.engine.parser.ast.Sentence;

import java.util.ArrayList;
import java.util.List;


public class ClauseCollector implements LogicVisitor<List<Clause>> {

    public static List<Clause> getClauses(Sentence sentence){
        ClauseCollector collector = new ClauseCollector();
        List<Clause> clauses = sentence.accept(collector);
        return clauses;
    }

    @Override
    public List<Clause> visitAtomicSentence(AtomicSentence atomicSentence) {
        List<Clause> result = new ArrayList<>();
        Literal pLiteral = new Literal(atomicSentence);
        result.add(new Clause(pLiteral));
        return result;
    }

    @Override
    public List<Clause> visitUnarySentence(ComplexSentence negated) {
        List<Clause> result = new ArrayList<>();
        Literal nLiteral = new Literal((AtomicSentence) negated.getSentenceA(), isPositive: false);
        result.add(new Clause(nLiteral));
```

```
        return result;
    }

    @Override
    public List<Clause> visitBinarySentence(ComplexSentence complexSentence) {
        List<Clause> result = new ArrayList<>();
        if(complexSentence.isAndSentence()){
            result.addAll(
                    complexSentence.getSentenceA().accept( visitor: this));
            result.addAll(
                    complexSentence.getSentenceB().accept( visitor: this));
        }else if (complexSentence.isOrSentence()){
            List<Literal> literals = new ArrayList<>(LiteralCollector.getLiterals(complexSentence))
            result.add(new Clause(literals));
        }

        return result;
    }
}

```

Ο κώδικας του αρχείου ClauseCollector.java

```
package com.drid.group_reasoning.engine.parser.visitors;


import com.drid.group_reasoning.engine.parser.ast.AtomicSentence;
import com.drid.group_reasoning.engine.parser.ast.ComplexSentence;
import com.drid.group_reasoning.engine.parser.ast.Sentence;


public abstract class SimplificationVisitor
        implements LogicVisitor<Sentence> {

    @Override
    public Sentence visitAtomicSentence(AtomicSentence node) { return node; }

    @Override
    public Sentence visitUnarySentence(ComplexSentence node){
        return new ComplexSentence(
                node.getSentenceA().accept( visitor: this),
                node.getConnective());
    }

    @Override
    public Sentence visitBinarySentence(ComplexSentence node){
        return new ComplexSentence(
                node.getSentenceA().accept( visitor: this),
                node.getSentenceB().accept( visitor: this),
                node.getConnective());
    }
}
```

Ο κώδικας του αρχείου SimplificationVisitor.java

```java
package com.drid.group_reasoning.engine.parser.visitors;

import com.drid.group_reasoning.engine.parser.ast.ComplexSentence;
import com.drid.group_reasoning.engine.parser.ast.Connective;
import com.drid.group_reasoning.engine.parser.ast.Sentence;

public class DeMorgansLaw extends SimplificationVisitor {

    @Override
    public Sentence visitUnarySentence(ComplexSentence node) {
        Sentence result = null;

        Sentence negated = node.getSentenceA();

        if (negated.isAtomicSentence()) {
            result = node;
        } else if (negated.isNotSentence()) {
            Sentence a = ((ComplexSentence) negated).getSentenceA();
            result = a.accept( visitor: this);
        } else if (negated.isAndSentence()
                || negated.isOrSentence()) {
            Sentence alpha = ((ComplexSentence) negated).getSentenceA();
            Sentence beta = ((ComplexSentence) negated).getSentenceB();

            Sentence notAlpha
                    = (new ComplexSentence(alpha, Connective.NOT)).accept( visitor: this);
            Sentence notBeta
                    = (new ComplexSentence(beta, Connective.NOT)).accept( visitor: this);
            if (negated.isAndSentence()) {
                result = new ComplexSentence(notAlpha, notBeta, Connective.OR);
            } else {
                result = new ComplexSentence(notAlpha, notBeta, Connective.AND);
            }
        }
        return result;
    }
}
```

Ο κώδικας του αρχείου DeMorgansLaw.java

```java
package com.drid.group_reasoning.engine.parser.visitors;

import com.drid.group_reasoning.engine.parser.ast.ComplexSentence;
import com.drid.group_reasoning.engine.parser.ast.Connective;
import com.drid.group_reasoning.engine.parser.ast.Sentence;

public class DistributiveProperty extends SimplificationVisitor {

    @Override
    public Sentence visitBinarySentence(ComplexSentence node) {
        Sentence result;

        if (node.isOrSentence()) {
            Sentence s1 = node.getSentenceA().accept( visitor: this);
            Sentence s2 = node.getSentenceB().accept( visitor: this);
```

```
            if (s1.isAndSentence() || s2.isAndSentence()) {
                Sentence alpha, betaAndGamma;
                if (s2.isAndSentence()) {
                    alpha = s1;
                    betaAndGamma = s2;
                } else {
                    alpha = s2;
                    betaAndGamma = s1;
                }
                Sentence beta = ((ComplexSentence) betaAndGamma).getSentenceA();
                Sentence gamma = ((ComplexSentence) betaAndGamma).getSentenceB();

                if (s2.isAndSentence()) {
                    Sentence alphaOrBeta = new ComplexSentence(
                            alpha, beta, Connective.OR)
                            .accept( visitor: this);

                    Sentence alphaOrGamma = new ComplexSentence(
                            alpha, gamma, Connective.OR)
                            .accept( visitor: this);

                    result = new ComplexSentence(alphaOrBeta, alphaOrGamma, Connective.AND);
                } else {
                    ComplexSentence bOrA
                            = (ComplexSentence) (new ComplexSentence(
                                    beta,
                                    alpha,
                                    Connective.OR))
                                    .accept( visitor: this);
                    ComplexSentence cOrA
                            = (ComplexSentence) (new ComplexSentence(
                                    gamma,
                                    alpha,
                                    Connective.OR))
                                    .accept( visitor: this);

                    result = new ComplexSentence(bOrA, cOrA, Connective.AND);
                }
            } else {
                result = new ComplexSentence(s1, s2, Connective.OR);
            }
        } else {
            result = super.visitBinarySentence(node);
        }

        return result;
    }
}
```

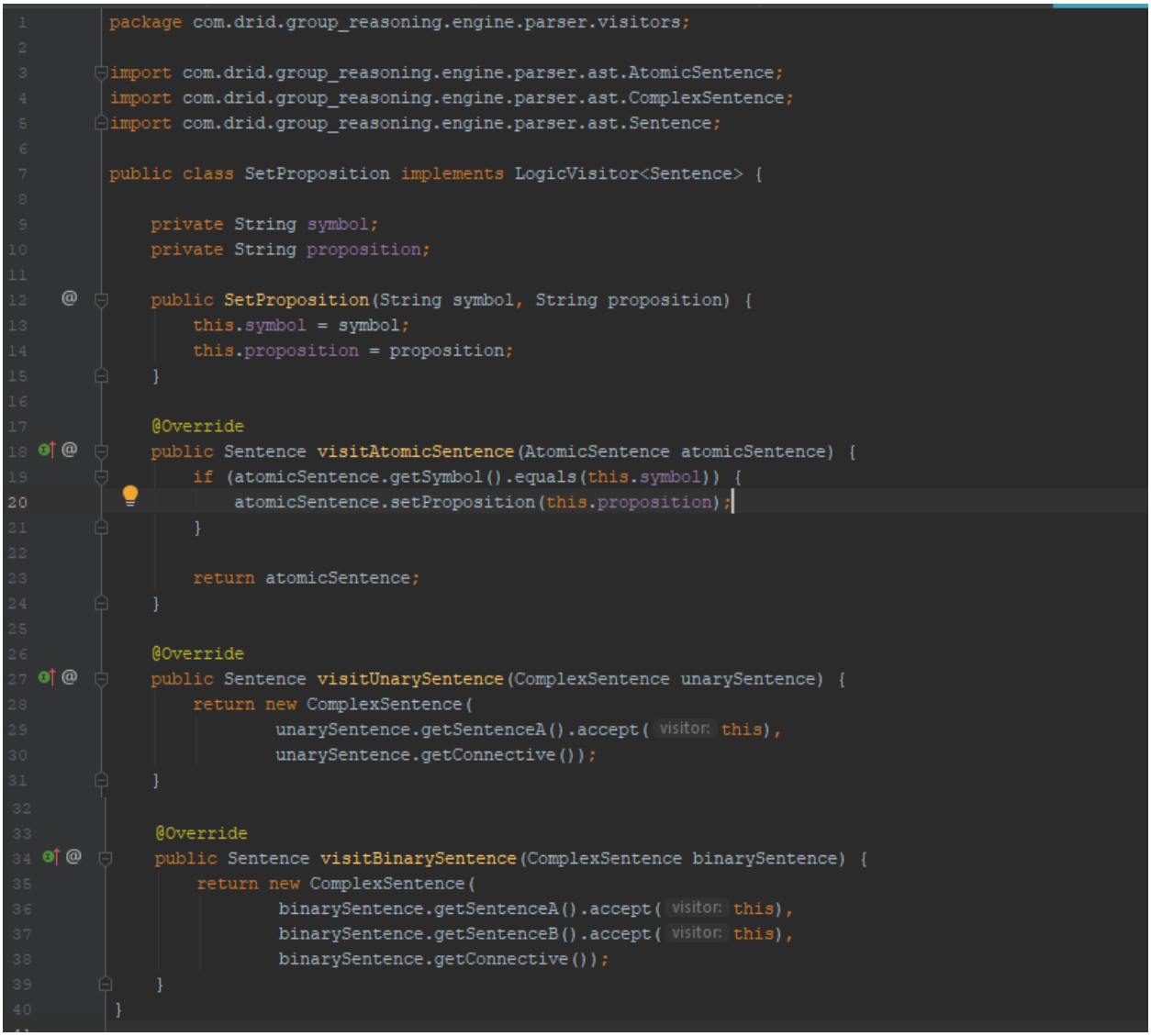
Ο κώδικας του αρχείου DistributiveProperty.java

```java
package com.drid.group_reasoning.engine.parser.visitors;

import com.drid.group_reasoning.engine.parser.ast.ComplexSentence;
import com.drid.group_reasoning.engine.parser.ast.Connective;
import com.drid.group_reasoning.engine.parser.ast.Sentence;

public class ImplicationElimination extends SimplificationVisitor {

    @Override
    public Sentence visitBinarySentence(ComplexSentence node) {
        Sentence result;
        if (node.isImplication()) {
            Sentence a = node.getSentenceA().accept( visitor: this);
            Sentence b = node.getSentenceB().accept( visitor: this);
            Sentence notA = new ComplexSentence(a, Connective.NOT);
            result = new ComplexSentence(notA, b, Connective.OR);

        } else {
            result = super.visitBinarySentence(node);

        }
        return result;
    }
}
```

Ο κώδικας του αρχείου ImplicationElimination.java

```java
package com.drid.group_reasoning.engine.parser.visitors;

import com.drid.group_reasoning.engine.parser.ast.ComplexSentence;
import com.drid.group_reasoning.engine.parser.ast.Connective;
import com.drid.group_reasoning.engine.parser.ast.Sentence;

public class EquivalenceElimination extends SimplificationVisitor {

    @Override
    public Sentence visitBinarySentence(ComplexSentence sentence) {
        Sentence result;

        if (sentence.isBiconditional()) {
            Sentence a = sentence.getSentenceA().accept( visitor: this);
            Sentence b = sentence.getSentenceB().accept( visitor: this);

            Sentence aImpliesB
                    = new ComplexSentence(a, b, Connective.IMPLICATION);

            Sentence bImpliesA
                    = new ComplexSentence(b, a, Connective.IMPLICATION);

            result = new ComplexSentence(aImpliesB, bImpliesA, Connective.AND);
        }else{
            result = super.visitBinarySentence(sentence);
        }

        return result;
    }
}
```

Ο κώδικας του αρχείου EquivalenceElimination.java



```java
package com.drid.group_reasoning.engine.parser.visitors;

import com.drid.group_reasoning.engine.parser.ast.AtomicSentence;
import com.drid.group_reasoning.engine.parser.ast.ComplexSentence;
import com.drid.group_reasoning.engine.parser.ast.Sentence;

public class SetProposition implements LogicVisitor<Sentence> {

    private String symbol;
    private String proposition;

    public SetProposition(String symbol, String proposition) {
        this.symbol = symbol;
        this.proposition = proposition;
    }

    @Override
    public Sentence visitAtomicSentence(AtomicSentence atomicSentence) {
        if (atomicSentence.getSymbol().equals(this.symbol)) {
            atomicSentence.setProposition(this.proposition);
        }

        return atomicSentence;
    }

    @Override
    public Sentence visitUnarySentence(ComplexSentence unarySentence) {
        return new ComplexSentence(
                unarySentence.getSentenceA().accept( visitor: this),
                unarySentence.getConnective());
    }

    @Override
    public Sentence visitBinarySentence(ComplexSentence binarySentence) {
        return new ComplexSentence(
                binarySentence.getSentenceA().accept( visitor: this),
                binarySentence.getSentenceB().accept( visitor: this),
                binarySentence.getConnective());
    }
}
```

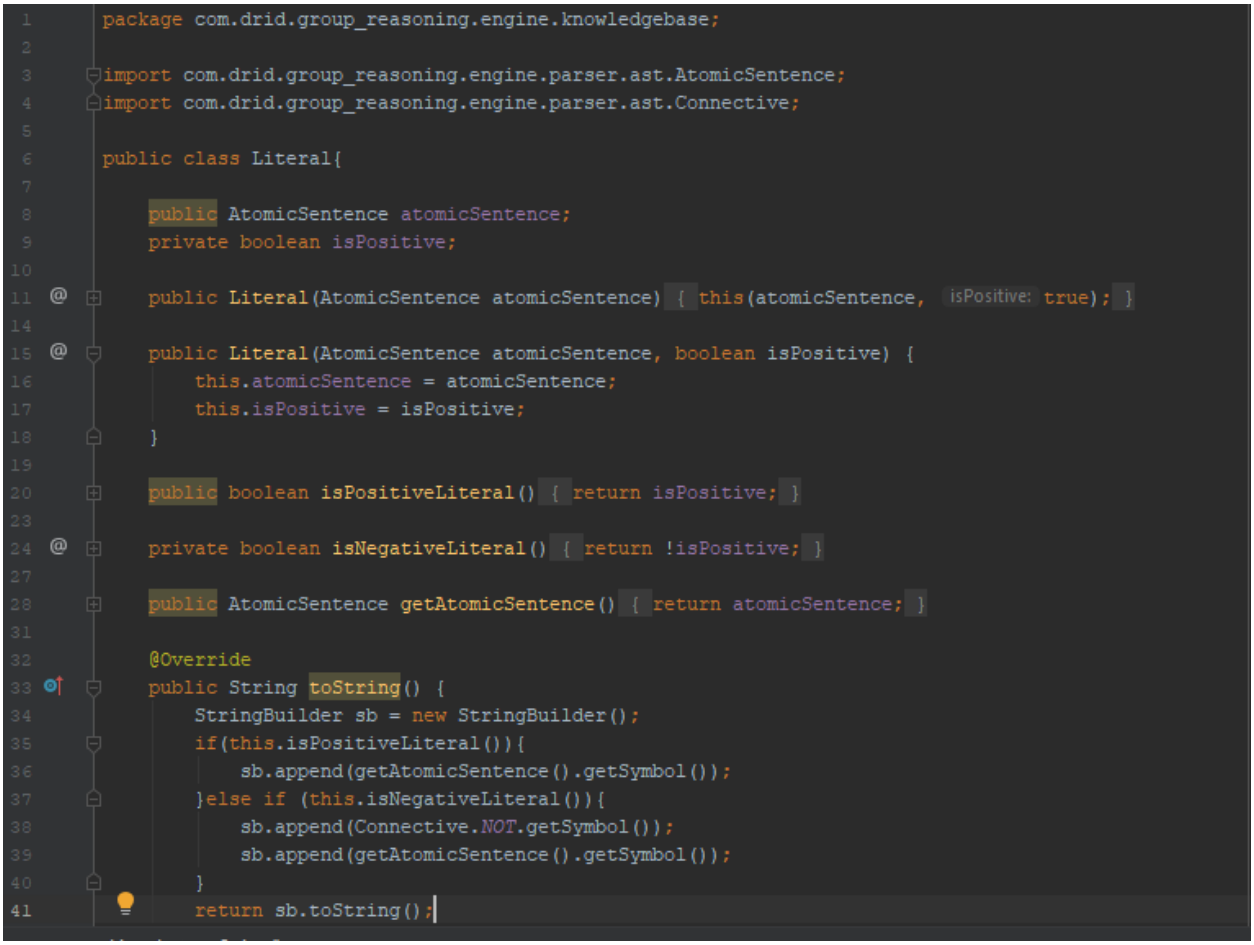
Ο κώδικας του αρχείου SetProposition.java

4.1.2 Υλοποίηση της βάσης γνώσης

Η υλοποίηση της βάσης γνώσης πραγματοποιείται με τη χρήση των παρακάτω κλάσεων Literal, Clause και KnowledgeBase.

Η κλάση Literal διαθέτει τα πεδία atomicSentence και isPositive. Το πεδίο atomicSentence είναι ένα στιγμιότυπο της κλάσης AtomicSentence και το πεδίο isPositive μια μεταβλητή τύπου boolean.

Επίσης η κλάση Literal διαθέτει τις μεθόδους isPositiveLiteral και isNegativeLiteral.Η μέθοδος isPositiveLiteral επιστρέφει την τιμή της μεταβλητής isPositive ενώ η μέθοδος isNegativeLiteral επιστρέφει την άρνηση της τιμής της μεταβλητής isPositive. Ακολουθεί ο κώδικας της κλάσης Literal.



```java
package com.drid.group_reasoning.engine.knowledgebase;

import com.drid.group_reasoning.engine.parser.ast.AtomicSentence;
import com.drid.group_reasoning.engine.parser.ast.Connective;

public class Literal{

    public AtomicSentence atomicSentence;
    private boolean isPositive;

    public Literal(AtomicSentence atomicSentence) { this(atomicSentence, isPositive: true); }

    public Literal(AtomicSentence atomicSentence, boolean isPositive) {
        this.atomicSentence = atomicSentence;
        this.isPositive = isPositive;
    }

    public boolean isPositiveLiteral() { return isPositive; }

    private boolean isNegativeLiteral() { return !isPositive; }

    public AtomicSentence getAtomicSentence() { return atomicSentence; }

    @Override
    public String toString() {
        StringBuilder sb = new StringBuilder();
        if(this.isPositiveLiteral()){
            sb.append(getAtomicSentence().getSymbol());
        }else if (this.isNegativeLiteral()){
            sb.append(Connective.NOT.getSymbol());
            sb.append(getAtomicSentence().getSymbol());
        }
        return sb.toString();
```

Η κλάση Clause διαθέτει τα πεδία literals, positiveLIterals και negatveLiterals. Το πεδίο literals είναι μια λίστα στην οποία εισάγονται οι στοιχειώδεις τύποι μιας πρότασης. Το πεδίο positiveLIterals είναι μια λίστα η οποία περιέχει θετικούς στοιχειώδεις τύπους μιας πρότασης. Επίσης το πεδίο negativeLitearals είναι μια λίστα η οποία περιέχει αρνητικούς στοιχειώδεις τύπους μιας πρότασης.

Οι μέθοδοι της κλάσης Clause είναι οι addLiterals(), isDefiniteClause() και μέθοδος totalPositiveLiterals().

Η μέθοδος addLiterals() δέχεται ως παράμετρο μια λίστα από στιγμιότυπα της κλάσης Literal. Κάθε στοιχείο ελέγχεται εάν είναι θετικός ή αρνητικός στοιχειώδης τύπος και έπειτα προστίθεται στην ανάλογη λίστα.

Η μέθοδος isDefiniteClause() ελέγχει εάν η λίστα positiveLiterals διαθέτει ένα και μόνο στοιχείο. Ουσιαστικά ελέγχει εάν μια πρόταση έχει το πολύ ένα θετικό στοιχειώδη τύπο.

Η μέθοδος totalPositiveLiterals() επιστρέφει μια τιμή τύπου int με τιμή έναν ακέραιο αριθμό που είναι ίσος με τον μέγιστο αριθμό στοιχείων της λίστας positiveLiterals. Ομοίως η μέθοδος totalNegativeLiterals() επιστρέφει τον μέγιστο αριθμό στοιχείων της λίστας negativeLiterals. Ακολουθεί ο κώδικας του αρχείου Clause.java

```java
package com.drid.group_reasoning.engine.knowledgebase;

import com.drid.group_reasoning.engine.parser.ast.AtomicSentence;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;


public class Clause {

    private List<Literal> literals;
    private List<AtomicSentence> positiveLiterals;
    private List<AtomicSentence> negativeLiterals;

    public Clause(Literal... literals) {
        this(Arrays.asList(literals));

    }

    public Clause(List<Literal> literals){
        this.literals = literals;
        this.positiveLiterals = new ArrayList<>();
        this.negativeLiterals = new ArrayList<>();

        addLiterals(literals);
    }

    private void addLiterals(List<Literal> literals) {
        for (Literal l : literals) {
            // Only add to caches if not already added
            if (l.isPositiveLiteral()) {
                this.positiveLiterals.add(l.getAtomicSentence());
```

```
            } else {
                this.negativeLiterals.add(l.getAtomicSentence());
            }

        }
    }

    public boolean isDefiniteClause() { return positiveLiterals.size() == 1; }

    public int totalNegativeLiterals() { return negativeLiterals.size(); }

    public List<AtomicSentence> getPositiveLiterals() { return positiveLiterals; }

    public List<AtomicSentence> getNegativeLiterals() { return negativeLiterals; }

    @Override
    public String toString() {
        StringBuilder sb = new StringBuilder();
        sb.append("{");
        for (Literal literal : literals) {
            sb.append(literal);
            if(!literal.equals(literals.get(literals.size() - 1))){
                sb.append(",");
            }
        }
        sb.append("}");
        return sb.toString();
    }

}
```

Η κλάση KnowledgeBase διαθέτει τα πεδία sentences, symbols, clauses και parser. Το πεδίο sentences είναι μια λίστα η οποία περιέχει τις προτάσεις της βάσης γνώσης. Το πεδίο symbols είναι ένα σύνολο τύπου HashSet το οποίο περιέχει σύμβολα των προτάσεων της βάσης γνώσης, ωστόσο ένα σύμβολο εισάγεται στο σύνολο μόνο μια φορά. Το πεδίο clauses είναι μια λίστα όπου εισάγονται προτάσεις που είναι σε μορφή διάζευξης στοιχειωδών τύπων. Επιπροσθέτως το πεδίο parser είναι ένα στιγμιότυπο της κλάσης LogicParser και αντιπροσωπεύει τον συντακτικό αναλυτή των προτάσεων της βάσης γνώσης.

Η μέθοδος tell() δέχεται ως παράμετρο μια συμβολοσειρά η αναλύεται συντακτικά και μετατρέπεται σε ένα αντικείμενο της κλάσης Sentence με μεταβλητή την sentence. Έπειτα καλείται η μέθοδος addSentence με όρισμα αυτή την sentence

Η μέθοδος addSentence δέχεται ως όρισμα ένα στιγμιότυπο της κλάσης Sentence, δηλαδή μια λογική πρόταση η οποία προστίθεται στη λίστα sentences. Στη συνέχεια μετατρέπει την πρόταση αυτή σε κανονική συζευκτική μορφή, καλώντας τη μέθοδο convertToCnf(), συλλέγει όλους τους στοιχειώδεις τύπους της και δημιουργεί ένα αντικείμενο της κλάσης Clause το οποίο προστίθεται στην λίστα clauses.

Η μέθοδος getConcuntionOfClauses() αρχικά μετατρέπει όλες τις προτάσεις της βάσης γνώσης σε μια πρόταση που είναι σύζευξη προτάσεων καλώντας τη μέθοδο conjunctionOfsenteneces(). Ακολούθως μετατρέπει αυτή τη πρόταση σε κανονική διαζευκτική μορφή καλώντας τη μέθοδο convertToCnf() και έπειτα καλεί τη μέθοδο getClauses(), η οποία επιστρέφει μια λίστα με αντικείμενα της κλάσης Clause. Η λίστα αυτή επιστρέφεται κατά την ολοκλήρωση της εκτέλεσης της μεθόδου. Ακολουθεί ο κώδικας της κλάσης KnowledgeBase.

```java
package com.drid.group_reasoning.engine.knowledgebase;

import com.drid.group_reasoning.engine.parser.LogicParser;
import com.drid.group_reasoning.engine.parser.ast.AtomicSentence;
import com.drid.group_reasoning.engine.parser.ast.Sentence;
import com.drid.group_reasoning.engine.parser.visitors.AtomCollector;
import com.drid.group_reasoning.engine.parser.visitors.ClauseCollector;
import com.drid.group_reasoning.engine.parser.visitors.LiteralCollector;

import java.util.ArrayList;
import java.util.HashSet;
import java.util.List;
import java.util.Set;


public class KnowledgeBase {

    private List<Sentence> sentences;
    private Set<AtomicSentence> symbols;
    private List<Clause> clauses;
    private LogicParser parser;

    public KnowledgeBase() {
        this.clauses = new ArrayList<>();
        this.sentences = new ArrayList<>();
        this.symbols = new HashSet<>();
        this.parser = new LogicParser();
    }

    public void tell(Sentence sentence) {
        addSentence(sentence);
    }

    private void addSentence(Sentence sentence) {
        this.sentences.add(sentence);
        Sentence cnfSentence = Sentence.convertToCnf(sentence);
        List<Literal> literals = LiteralCollector.getLiterals(cnfSentence);
        this.clauses.add(new Clause(literals));
    }

    public List<Clause> getClauses() { return clauses; }

    public List<Clause> getConjunctionOfClauses() {
        List<Clause> result = new ArrayList<>();
        if(!sentences.isEmpty()){
            Sentence kbSentence = Sentence.conjunctionOfSentences(sentences);
            Sentence kbCnfSentence = Sentence.convertToCnf(kbSentence);
            result.addAll(ClauseCollector.getClauses(kbCnfSentence));
        }

        return result;
    }

    public List<Sentence> getSentences() { return sentences; }

    public Set<AtomicSentence> getSymbols() {
        for (Sentence sentence : sentences) {
            this.symbols.addAll(AtomCollector.getAtomicSentences(sentence));
        }
        return symbols;
    }
}
```

4.1.3 Η κλάση ForwardChainging

Η υλοποίηση της μεθόδου εξαγωγής συμπερασμάτων προς τα εμπρός αλυσιδωτή εκτέλεση κανόνων πραγματοποιείται με τη χρήση των κλάσεων ForwardChaining.

Η κλάση ForwardChaining περιέχει όλες τις μεθόδους που υλοποιούν την διαδικασία εξαγωγής συμπερασμάτων μέσω της τεχνικής της προς τα εμπρός αλυσιδωτής εκτέλεσης (Forward Chaining). Περιλαμβάνει τις ακόλουθες μεθόδους.

Η μέθοδος initCount() δημιουργεί μια δομή τύπου HashMap με μεταβλητή που ονομάζεται count. Ο χάρτης count δέχεται ως κλειδιά στιγμιότυπα της κλάσης Clause και ως τιμές στιγμιότυπα της κλάσης Integer. Σε αυτή τη δομή αποθηκεύονται οι οριστικές προτάσεις (definite clauses) που διαθέτει η βάση γνώσης καθώς και ο αριθμός των αρνητικών στοιχειωδών τύπων (literals). Με τη χρήση αυτού του χάρτη γίνεται διαχωρισμός των κανόνων από τα γεγονότα. Για παράδειγμα η οριστική πρόταση { ¬A, ¬B, C} έχει δυο αρνητικούς στοιχειώδεις τύπους οπότε είναι κανόνας, ενώ η πρόταση {A} είναι ένα γεγονός. Η μέθοδος initCount() επιστρέφει τον χάρτη count.

Η μέθοδος initInferred() δημιουργεί μια δομή τύπου HashMap με μεταβλητή που ονομάζεται inferred. Ο χάρτης inferred δέχεται ως κλειδιά στιγμιότυπα της κλάσης Sentence και ως τιμές στιγμιότυπα της κλάσης Boolean. Έπειτα εισάγονται ως κλείδα όλα τα προτασιακά σύμβολα που υπάρχουν στις προτάσεις της βάσης γνώσης, μια φορά το καθένα και όλα έχουν την τιμή false. Αυτός ο χάρτης χρησιμοποιείται ούτως ώστε να έχουμε επίγνωση των προτάσεων ή ερωτημάτων που έχουν επιλυθεί. Η μέθοδος initInferred() επιστρέφει τον χάρτη inferred.

Η μέθοδος initAgenda() δημιουργεί μια δομή ουράς που δέχεται στιγμιότυπα της κλάσης Sentence. Σε αυτή την δομή εισάγονται προτάσεις (clauses) που δεν έχουν αρνητικούς στοιχειώδεις τύπους, δηλαδή τα γεγονότα (facts). Η μέθοδος initAgenda() επιστρέφει την δομή agenda.

Η μέθοδος conclusion() δέχεται ως παράμετρο ένα αντικείμενο της κλάσης Clause και επιστρέφει τους θετικούς στοιχειώδης τύπους. Στην προκειμένη περίπτωση επιστρέφει ένα στοιχειώδη τύπο ο οποίος αντιπροσωπεύει το συμπέρασμα ενός κανόνα ο οποίος είναι οριστική πρόταση και διάζευξή στοιχειωδών τύπων.

Η μέθοδος initAtomOccurence() δημιουργεί μια δομή τύπου HashMap με μεταβλητή που ονομάζεται atomOccurrence. Ο χάρτης inferred δέχεται ως κλειδιά στιγμιότυπα της κλάσης Sentence και ως τιμές στιγμιότυπα της κλάσης HashSet το οποίο περιέχει στοιχεία που είναι στιγμιότυπα της κλάσης Clause. Σε αυτή τη δομή εισάγονται προτάσεις οι οποίες είναι κλειδιά του χάρτη inferred και συγχρόνως ανήκουν στους αρνητικούς στοιχειώδεις τύπους των προτάσεων (clauses) που είναι κλειδιά του χάρτη count. Με τη χρήση αυτού του χάρτη γίνεται αντιστοίχιση των προτασιακών συμβόλων της βάσης γνώσης και των κανόνων στους οποίος τα σύμβολα αυτά είναι ορισμένα ως υποθέσεις. Κατά την ολοκλήρωση της η μέθοδος initAgenda() επιστρέφει την δομή atomOccurence.

Η μέθοδος initPremises() δημιουργεί μια δομή τύπου HashMap με μεταβλητή που ονομάζεται premises. Ο χάρτης premises δέχεται ως κλειδιά στιγμιότυπα της κλάσης Sentence και ως τιμές στιγμιότυπα της κλάσης HashSet το οποίο περιέχει στοιχεία που είναι στιγμιότυπα της κλάσης Sentence. Στον χάρτη εισάγονται ως κλειδιά, προτάσεις οι οποίες είναι θετικοί στοιχειώδεις τύποι μιας πρότασης και ως τιμές το σύνολο των αρνητικών στοιχειωδών της ίδιας πρότασης. Ουσιαστικά ο χάρτης αυτός περιγράφει την αντιστοιχία του συμπεράσματος ενός κανόνα με τις υποθέσεις του. Η μέθοδος επιστρέφει την δομή premises.

Η μέθοδος fcEntails() επιστρέφει μια τιμή boolean και ελέγχει εάν μια πρόταση- ερώτημα καλύπτεται λογικά από τη βάση γνώσης. Όσο η δομή agenda δεν είναι κενή αφαιρούμε το πρώτο στοιχείο της το οποίο είναι μια ατομική πρόταση και το αναθέτουμε σε ένα νέο στιγμιότυπο της

κλάσης AtomicSentence με τη μεταβλητή p. Ακολουθεί η σύγκριση της πρότασης p με το ερώτημα. Εάν οι προτάσεις είναι ισότιμες τότε το ερώτημα έχει επιλυθεί και η επιστρέφεται η τιμή true. Διαφορετικά εάν η πρόταση p είναι κλειδί της δομή inferred και έχει τιμή false τότε η p εισάγεται εκ νέου στην δομή με τιμή true. Έπειτα κάθε κανόνας στον οποίο η πρόταση p είναι ορισμένη ως υπόθεση, εισάγεται ως κλειδί στην δομή count και ως κλειδί εισάγεται ο αριθμός των αρνητικών τύπων μειωμένος κατά ένα. Εάν ο αριθμός των αρνητικών τύπων μιας πρότασης αυτός γίνει ίσος με 0 τότε καλείται η μέθοδος conclusion με παράμετρο αυτόν τον κανόνα. Ουσιαστικά η μέθοδος αυτή ελέγχει αν οι προτάσεις της βάσης γνώσης είναι αληθείς μέχρι να αποδειχθεί ότι το ερώτημα που τέθηκε είναι αληθές.

Η μέθοδος getMissingFacts() επιστρέφει το σύνολο των προτάσεων που ανήκουν στις υποθέσεις ενός κανόνα αλλά δεν υπάρχουν ως γεγονότα στη βάση γνώσης. Δέχεται ως παράμετρο το συμπέρασμα ενός κανόνα και στη συνέχεια ελέγχει εάν οι υποθέσεις αυτού του κανόνα ανήκουν στην δομή inferred και έχουν τιμή false. Ακολουθεί ο κώδικας της κλάσης ForwardChaining σε γλώσσα java.

```java
package com.drid.group_reasoning.engine.inference;

import android.util.Log;

import com.drid.group_reasoning.engine.knowledgebase.Clause;
import com.drid.group_reasoning.engine.knowledgebase.KnowledgeBase;
import com.drid.group_reasoning.engine.parser.ast.AtomicSentence;
import com.drid.group_reasoning.engine.parser.ast.Sentence;

import java.util.HashMap;
import java.util.HashSet;
import java.util.LinkedList;
import java.util.List;
import java.util.Map;
import java.util.Queue;
import java.util.Set;

public class ForwardChaining {

    private Map<Sentence, Boolean> inferred;
    private Map<Sentence, Set<Sentence>> premises;

    private static String TAG = ForwardChaining.class.getName();

    private Map<Clause, Integer> initCount(KnowledgeBase kb) {
        Map<Clause, Integer> count = new HashMap<>();

        Set<Clause> clauses = new HashSet<>(kb.getConjunctionOfClauses());

        Log.i(ForwardChaining.class.getSimpleName(), msg: "initCount: clauses = " + clauses);
        for (Clause c : clauses) {
            if (!c.isDefiniteClause()) {
                throw new IllegalArgumentException(
```

```
                    "Knowledge Base contains non-horn clauses:" + c);
            }
            count.put(c, c.totalNegativeLiterals());
        }
        return count;
    }

    private Map<Sentence, Boolean> initInferred(KnowledgeBase kb) {
        Map<Sentence, Boolean> inferred = new HashMap<>();
        for (Sentence p : kb.getSymbols()) {
            inferred.put(p, false);
        }
        return inferred;
    }

    private Queue<Sentence> initAgenda(Map<Clause, Integer> count) {
        Queue<Sentence> agenda = new LinkedList<>();
        for (Clause c : count.keySet()) {
            if (c.totalNegativeLiterals() == 0) {
                agenda.add(conclusion(c));
            }
        }

        return agenda;
    }

    private Sentence conclusion(Clause c) { return c.getPositiveLiterals().iterator().next(); }

    private Map<Sentence, Set<Clause>> initAtomOccurrence(
            Map<Clause, Integer> count,
            Map<Sentence, Boolean> inferred) {...}

        Map<Sentence, Set<Clause>> atomOccurrence = new HashMap<>();
        for (Sentence p : inferred.keySet()) {
            Set<Clause> clausesWithPInPremise = new HashSet<>();
            for (Clause c : count.keySet()) {
                // Note: The negative symbols comprise the premise
                if (c.getNegativeLiterals().contains(p)) {
                    clausesWithPInPremise.add(c);
                }
            }
            atomOccurrence.put(p, clausesWithPInPremise);
        }

        return atomOccurrence;
    }

    private Map<Sentence, Set<Sentence>> initPremises(KnowledgeBase kb) {
        Map<Sentence, Set<Sentence>> premises = new HashMap<>();
        List<Clause> conjunctionOfClauses = kb.getConjunctionOfClauses();

        for (Clause c : conjunctionOfClauses) {

            if (!c.getNegativeLiterals().isEmpty()) {
                Set<Sentence> negativeLiterals = new HashSet<>(c.getNegativeLiterals());
                Sentence goal = c.getPositiveLiterals().get(0);
                premises.put(goal, negativeLiterals);
            }

        }
        return premises;
    }
```

```
    public boolean fcEntails(KnowledgeBase kb, Sentence q) {

        Map<Clause, Integer> count = initCount(kb);
        inferred = initInferred(kb);
        Queue<Sentence> agenda = initAgenda(count);
        Map<Sentence, Set<Clause>> atomOccurrenceInPremise = initAtomOccurrence(count, inferred);
        premises = initPremises(kb);


        Log.i(TAG,  msg: "fcEntails: symbols " + kb.getSymbols());
        while (!agenda.isEmpty()) {
            AtomicSentence p = (AtomicSentence) agenda.remove();
            if (p.equals(q)) {
                return true;
            }

            if (inferred.get(p).equals(Boolean.FALSE)) {

                inferred.put(p, true);
                for (Clause c : atomOccurrenceInPremise.get(p)) {
                    count.put(c, count.get(c) - 1);
                    if (count.get(c) == 0) {
                        agenda.add(conclusion(c));
                    }
                }
            }
        }

        return false;
    }


    public Set<AtomicSentence> getMissingFacts(Sentence sentence) {
        Set<AtomicSentence> result = new HashSet<>();
        Log.i(ForwardChaining.class.getSimpleName(),
                msg: "getMissingFacts:  getting Missing Facts of" + sentence);
        Log.i(ForwardChaining.class.getSimpleName(),
                msg: "getMissingFacts: premises " + premises);

        if (premises.containsKey(sentence)) {
            for (Sentence premise : premises.get(sentence)) {
                if (inferred.get(premise).equals(Boolean.FALSE)) {
                    result.add((AtomicSentence) premise);
                }
            }
        }
        return result;
    }
}
```

4.1.4 Η κλάση InferenceEngine

Η κλάση InferenceEngine περιλαμβάνει τις μεθόδους που υλοποιούν ένα μηχανισμό εξαγωγής συμπερασμάτων. Περιλαμβάνει τις παρακάτω μεθόδους:

Η μέθοδος initKnowledgeBase δημιουργεί ένα στιγμιότυπο της κλάσης KnowledgeBase το οποίο αντιπροσωπεύει τη βάση γνώσης. Έπειτα καλούνται η μέθοδος getRulesFromDb() η οποία επιστέφει λίστα που περιέχει κανόνες και η getFactsFromDb() οποία επιστέφει λίστα που

περιέχει γεγονότα. Όλοι οι κανόνες και τα γεγονότα εισάγονται στη βάση γνώσης με την κλήση της μεθόδου tell().

Η μέθοδος getRulesFromDb() λαμβάνει όλους τους κανόνες που είναι αποθηκευμένοι στη βάση δεδομένων rules και τους προσθέτει σε μια λίστα την οποία επιστρέφει κατά την ολοκλήρωση της εκτέλεσής της.

Η μέθοδος getFactsFromDb() λαμβάνει τα γεγονότα που είναι αποθηκευμένα στη βάση δεδομένων facts και τα προσθέτει σε μια λίστα. Η μέθοδος επιστρέφει τη λίστα αυτή κατά την ολοκλήρωση της εκτέλεσής της.

Η μέθοδος conclusionOfRule() αρχικά αναλύει συντακτικά όλους τους κανόνες της βάσης γνώσης, τους μετατρέπει σε ένα σύνολο από προτάσεις που είναι διαζεύξεις στοιχειωδών τύπων, δηλαδή Clauses και έπειτα ελέγχει εάν οι προτάσεις αυτές είναι οριστικές. Εάν μια πρόταση είναι οριστική τότε ο θετικός στοιχειώδης τύπος δηλαδή το συμπέρασμα προστίθεται σε μια λίστα την queries.

Η μέθοδος getPropositionFromDb() λαμβάνει την πρόταση φυσικής γλώσσας κανόνες που είναι αποθηκευμένη στον πίνακα symbols της βάσης δεδομένων rules και την καταχωρεί σε μια συμβολοσειρά την proposition την οποία και επιστρέφει.

Η μέθοδος solve() έχει δυο θεοποιήσεις. Στην πρώτη υλοποίηση δημιουργεί ένα στιγμιότυπο της κλάσης ForwardChaining και καλεί την μέθοδο fcEntails() για κάθε κανόνα της βάσης γνώσης. Ουσιαστικά η μέθοδος αυτή ελέγχει εάν το συμπέρασμα ενός κανόνα μπορεί να καλυφθεί λογικά από τη βάση γνώσης. Η δεύτερη υλοποίηση έχει παρόμοια λογική με τη διαφορά ότι καλείται όταν μια συσκευή λαμβάνει μια λίστα ερωτήματα (τα οποία είναι στιγμιότυπα της κλάσης AtomicSentence) από το δίκτυο και έχει παραμέτρους τη λίστα αυτή.
η μέθοδος addNewFactsToKb() έχει ως παράμετρο τη λίστα με τα γεγονότα που βρέθηκαν στο δίκτυο και τα προσθέτει στη βάση γνώσης. Ακολουθεί ο κώδικας της κλάσης InferenceEngine

```
1   package com.drid.group_reasoning.engine.inference;
2
3   import android.content.Context;
4   import android.database.Cursor;
5   import android.net.Uri;
6   import android.util.Log;
7
8   import com.drid.group_reasoning.data.contracts.FactContract;
9   import com.drid.group_reasoning.data.contracts.RuleContract;
10  import com.drid.group_reasoning.engine.knowledgebase.Clause;
11  import com.drid.group_reasoning.engine.knowledgebase.KnowledgeBase;
12  import com.drid.group_reasoning.engine.parser.LogicParser;
13  import com.drid.group_reasoning.engine.parser.ast.AtomicSentence;
14  import com.drid.group_reasoning.engine.parser.ast.Sentence;
15  import com.drid.group_reasoning.engine.parser.visitors.AtomCollector;
16  import com.drid.group_reasoning.engine.parser.visitors.ClauseCollector;
17  import com.drid.group_reasoning.engine.parser.visitors.SetProposition;
18
19  import java.util.ArrayList;
20  import java.util.HashSet;
21  import java.util.List;
22  import java.util.Set;
```

```java
import static com.drid.group_reasoning.data.contracts.RuleContract.PATH_SYMBOL;

public class InferenceEngine {

    private static final String TAG = "InferenceEngine";
    private Context context;

    private KnowledgeBase kb;
    private List<Sentence> rules = new ArrayList<>();
    private List<Sentence> facts = new ArrayList<>();

    private LogicParser parser = new LogicParser();
    private List<AtomicSentence> queries = new ArrayList<>();
    private List<AtomicSentence> resolvedQueries = new ArrayList<>();
    private List<AtomicSentence> unresolvedQueries = new ArrayList<>();

    private Set<AtomicSentence> missingFacts = new HashSet<>();

    public InferenceEngine(Context context) { this.context = context; }

    public void initKnowledgeBase() {

        kb = new KnowledgeBase();

        rules = getRulesFromDatabase();
        facts = getFactsFromDatabase();

        for (Sentence rule : rules) {
            kb.tell(rule);
        }

        for (Sentence fact : facts) {
            kb.tell(fact);
        }
    }

    private List<Sentence> getRulesFromDatabase(){
        List<Sentence> rulesList = new ArrayList<>();

        Cursor ruleCursor = context.getContentResolver().query(
                RuleContract.RuleEntry.RULES_URI,
                new String[]{RuleContract.RuleEntry.COLUMN_RULE_PL},
                 selection: null,
                 selectionArgs: null,
                 sortOrder: null);
        if (ruleCursor.moveToFirst()) {
            do {
                String rulePl = ruleCursor.getString(
                        ruleCursor.getColumnIndex(RuleContract.RuleEntry.COLUMN_RULE_PL));

                Sentence sentence = parser.parse(rulePl);

                List<AtomicSentence> symbols = AtomCollector.getAtomicSentences(sentence);
                for(AtomicSentence atom : symbols){
                    String proposition = getPropositionFromDatabase(atom);
                    atom.accept(new SetProposition(atom.getSymbol(),proposition));
                }

                rulesList.add(sentence);
```

```
85          } while (ruleCursor.moveToNext());
86      }
87      return rulesList;
88  }
89
90  private List<Sentence> getFactsFromDatabase() {
91      List<Sentence> factList = new ArrayList<>();
92
93      Cursor factCursor = context.getContentResolver().query(
94              FactContract.FactEntry.FACT_URI,
95               projection: null,
96               selection: null,
97               selectionArgs: null,
98               sortOrder: null);
99
100     if (factCursor.moveToFirst()) {
101         do {
102             String symbol = factCursor.getString(
103                     factCursor.getColumnIndex(FactContract.FactEntry.COLUMN_FACT_PROPOSITION));
104             String proposition = factCursor.getString(
105                     factCursor.getColumnIndex(FactContract.FactEntry.COLUMN_FACT_SYMBOL));
106
107             Sentence sentence = parser.parse(symbol);
108
109             sentence.accept(new SetProposition(symbol, proposition));
110
111             factList.add(sentence);
112         } while (factCursor.moveToNext());
```

```
114
115         return factList;
116     }
117
118 @   private String getPropositionFromDatabase(AtomicSentence sentence) {
119         String proposition = "";
120         Uri queryVarUri = RuleContract.SymbolEntry.SYMBOLS_URI.buildUpon()
121                 .appendPath(PATH_SYMBOL).build();
122
123         Cursor symbolCursor = context.getContentResolver().query(
124                 queryVarUri,
125                 new String[]{RuleContract.SymbolEntry.COLUMN_PROPOSITION},
126                  selection: RuleContract.SymbolEntry.COLUMN_SYMBOL + "=?",
127                 new String[]{sentence.getSymbol()},
128                  sortOrder: null
129         );
130         if (symbolCursor.moveToFirst()) {
131             proposition = symbolCursor
132                     .getString(symbolCursor.getColumnIndex(
133                             RuleContract.SymbolEntry.COLUMN_PROPOSITION));
134         }
135         return proposition;
136     }
137
138     private void conclusionOfRule(Sentence rule) {
139         Sentence cnf = Sentence.convertToCnf(rule);
140         List<Clause> clauses = ClauseCollector.getClauses(cnf);
141
142         for(Clause clause : clauses){
143             if(clause.isDefiniteClause()){
144                 List<AtomicSentence> positiveLiterals = clause.getPositiveLiterals();
145                 for(AtomicSentence positiveLiteral : positiveLiterals){
```

```
                    String proposition = getPropositionFromDatabase(positiveLiteral);

                    if (proposition != null) {
                        positiveLiteral.accept(new SetProposition(positiveLiteral.getSymbol(),
                                proposition));
                    }
                }
                queries.addAll(positiveLiterals);
            }else{
                Log.i(TAG, msg: "conclusionOfRule: Not a definite clause " + clause);
            }
        }
    }

    public void solve() {
        resolvedQueries.clear();
        unresolvedQueries.clear();
        missingFacts.clear();
        queries.clear();
        Log.i(TAG, msg: "solve: Knowledge Base " + kb.getClauses());
        ForwardChaining fc = new ForwardChaining();


        for(Sentence rule : rules){
            conclusionOfRule(rule);
        }

        for(AtomicSentence query : queries){
            boolean isProved = fc.fcEntails(kb, query);

            if (isProved) {
                Log.i(TAG, msg: "solve: Query " + query + " is resolved");
                resolvedQueries.add(query);
            } else {

                Log.i(TAG, msg: "solve: Query " + query + " is not resolved");

                unresolvedQueries.add(query);
                missingFacts.addAll(fc.getMissingFacts(query));
            }
        }

        if (!resolvedQueries.isEmpty()) {
            Log.i(TAG, msg: "solve: Resolved queries:\n " + resolvedQueries);
        }

        if (!unresolvedQueries.isEmpty()) {
            Log.i(TAG, msg: "solve: Unresolved queries:\n " + unresolvedQueries);
            Log.i(TAG, msg: "solve: Missing facts: " + missingFacts);
        }
    }

    public void solve(List<AtomicSentence> queries) {
        Log.i(TAG, msg: "solve' : Knowledge Base " + kb.getClauses());
        Log.i(TAG, msg: "solve' : Queries  " + queries);
```

```
        ForwardChaining fc = new ForwardChaining();

        for (AtomicSentence query : queries) {
            Log.i(TAG,  msg: "solve: Does " + query.getSymbol() + " entail the knowledge base?");
            boolean isProved = fc.fcEntails(kb, query);

            if (isProved) {
                resolvedQueries.add(query);
            } else {
                unresolvedQueries.add(query);
                Set<AtomicSentence> notInferred = fc.getMissingFacts(query);

                if (!notInferred.isEmpty()) {
                    missingFacts.addAll(notInferred);
                }
            }

        }

        Log.i(TAG,  msg: "solve' : Resolved queries: " + resolvedQueries);
        Log.i(TAG,  msg: "solve' : Unresolved queries: " + unresolvedQueries);
        Log.i(TAG,  msg: "solve' : Missing facts: " + missingFacts);

    }

    public List<AtomicSentence> getResolvedQueries() { return resolvedQueries; }

    public List<AtomicSentence> getUnresolvedQueries() { return unresolvedQueries; }
```

```
    public Set<AtomicSentence> getMissingFacts() {
        return missingFacts;
    }

@   public void addNewFactsToKb(List<AtomicSentence> facts) {

        for (AtomicSentence fact : facts) {

            if (!kb.getSentences().contains(fact)) {
                kb.tell(fact);
                Log.i(TAG,  msg: "addNewFactsToKb: Knowledge Base " + kb.getClauses());
            } else {
                Log.i(TAG,  msg: "addNewFactsToKb: " + fact.toString() + " already exists");
                Log.i(TAG,  msg: "addNewFactsToKb: Knowledge Base " + kb.getClauses());
            }
        }
    }

    public KnowledgeBase getKnowledgeBase() { return kb; }
}
```

4.2 Υλοποίηση του Peer-To-Peer δικτύου

Σε αυτή την ενότητα αναλύεται ο σχεδιασμός ενός μοντέλου στο οποίο βασίζεται η υλοποίηση του Peer-to-peer δικτύου. Το μοντέλο αυτό αποτελείται από οντότητες που αντιπροσωπεύουν ένα κόμβο του δίκτυο ή ένα μήνυμα που στέλνεται μεταξύ των κόμβων. Επίσης γίνεται ορισμός μιας υπηρεσίας η οποία αναλαμβάνει τις απαραίτητες διεργασίες όπως σύνδεση μεταξύ των κόμβων ή ανταλλαγή δεδομένων.

4.2.1 Η κλάση Message

Η κλάση Message είναι μια αφηρημένη αναπαράσταση ενός μηνύματος που στέλνεται στο δίκτυο. Υλοποιεί την διεπαφή Serializable ούτως ώστε να είναι εφικτή η μετατροπή ενός αντικειμένου σε πίνακα bytes, ενώ δεν υλοποιεί καμία μέθοδο.

```java
package com.drid.group_reasoning.network.model;

import java.io.Serializable;

public abstract class Message implements Serializable {}
```

4.2.2 Η κλάση QueryMessage

Η κλάση QueryMessage είναι επέκταση της κλάσης Message και αντιπροσωπεύει ένα μήνυμα το οποίο είναι ερώτημα για γεγονότα. Περιλαμβάνει τις μεταβλητές sender η οποία είναι τύπου String και αντιπροσωπεύει τον αποστολέα και την requestedFacts η οποία είναι μια λίστα από ατομικές προτάσεις και αντιπροσωπεύει την τα γεγονότα που ζητούνται. Ακολουθεί ο κώδικας του αρχείου QueryMessage.java:

```java
package com.drid.group_reasoning.network.model;

import com.drid.group_reasoning.engine.parser.ast.AtomicSentence;

import java.util.List;

public class QueryMessage extends Message {

    private String sender;
    private List<AtomicSentence> requestedFacts;

    public QueryMessage(String sender, List<AtomicSentence> requestedFacts) {
        this.sender = sender;
        this.requestedFacts = requestedFacts;
    }
    public String getSender() { return sender; }

    public List<AtomicSentence> getRequestedFacts() { return requestedFacts; }

}
```

4.2.3 Η κλάση ResponseMessage

Η κλάση ResponseMessage είναι επέκταση της κλάσης Message και αντιπροσωπεύει ένα μήνυμα το οποίο είναι απάντηση με τα γεγονότα που έχουν βρεθεί στο δίκτυο. Περιλαμβάνει 2 μεταβλητές στιγμιότυπου την sender η οποία είναι τύπου String και αντιπροσωπεύει τον αποστολέα και την foundFacts η οποία είναι μια λίστα από ατομικές προτάσεις και αντιπροσωπεύει την τα γεγονότα που έχουν βρεθεί. Ακολουθεί ο κώδικας του αρχείου ResponseMessage.java:

```java
package com.drid.group_reasoning.network.model;

import com.drid.group_reasoning.engine.parser.ast.AtomicSentence;

import java.util.List;

public class ResponseMessage extends Message {

    private String sender;
    private List<AtomicSentence> foundFacts;

    public ResponseMessage(String sender, List<AtomicSentence> foundFacts) {
        this.sender = sender;
        this.foundFacts = foundFacts;
    }

    public String getSender() { return sender; }

    public List<AtomicSentence> getFoundFacts() { return foundFacts; }

}

```

4.2.4 Η κλάση QueryHistoryMessage

Η κλάση QueryHistoryMessage είναι επέκταση της κλάσης Message και αντιπροσωπεύει ένα μήνυμα το οποίο είναι ερώτημα για ιστορικό ερωτημάτων. Διαθέτει μια μεταβλητή στιγμιότυπου, την sender η οποία αντιπροσωπεύει τον αποστολέα του μηνύματος. Ακολουθεί ο κώδικας του αρχείου QueryHistoryMessage.java:

```java
package com.drid.group_reasoning.network.model;

public class QueryHistoryMessage extends Message {
    private String sender;

    public QueryHistoryMessage(String sender) { this.sender = sender; }

    public String getSender() { return sender; }
}
```

4.2.5 Η κλάση ResponseHistoryMessage

Η κλάση ResponseHistoryMessage είναι επέκταση της κλάσης Message και αντιπροσωπεύει ένα μήνυμα το οποίο είναι απάντηση που περιέχει το ιστορικό ερωτημάτων. Διαθέτει 2 μεταβλητές στιγμιότυπου:

- την sender η οποία είναι τύπου String και αντιπροσωπεύει τον αποστολέα του μηνύματος
- την queryHistory η οποία είναι τύπου HashMap με αντιπροσωπεύει το ιστορικό των ερωτημάτων

Ακολουθεί ο κώδικας του αρχείου ResponseHistoryMessage.java:

```java
package com.drid.group_reasoning.network.model;

import com.drid.group_reasoning.engine.parser.ast.AtomicSentence;

import java.util.List;
import java.util.Map;

public class ResponseHistoryMessage extends Message {
    private String sender;
    private Map<AtomicSentence, List<String>> queryHistory;

    public ResponseHistoryMessage(String sender, Map<AtomicSentence, List<String>> queryHistory) {
        this.sender = sender;
        this.queryHistory = queryHistory;
    }

    public String getSender() { return sender; }

    public Map<AtomicSentence, List<String>> getQueryHistory() { return queryHistory; }
}
```

4.2.6 Η κλάση Peer

Η κλάση Peer περιγράφει έναν κόμβο του δικτύου. Αποτελείται από τα πεδία peer_id, username και status. Το πεδίο peer_id είναι μια συμβολοσειρά η οποία είναι ένα μοναδικό αναγνωριστικό με το οποίο ένας κόμβος του δίκτυο δημοσιεύεται προκειμένου να είναι ορατός από άλλους κόμβους. Το πεδίο username είναι μια συμβολοσειρά η οποία αντιπροσωπεύει το όνομα ενός κόμβου. Το πεδίο status είναι μια συμβολοσειρά η οποία περιγράφει την κατάσταση ενός κόμβου οπός για παράδειγμα εάν είναι διαθέσιμος για νέα σύνδεση. Ακουσθεί ο κώδικας της κλάσης Peer

```java
package com.drid.group_reasoning.network.model;

import android.os.Parcel;
import android.os.Parcelable;
import android.support.annotation.NonNull;

public class Peer implements Parcelable {

    private String peerId;
    private String name;
    private String status;

    public static final String AVAILABLE = "Available";
    public static final String CONNECTING = "Connecting";
    public static final String CONNECTED = "Connected";

    public Peer(){ }

    public Peer(String peerId, String name, String status) {
        this.peerId = peerId;
        this.name = name;
        this.status = status;
    }
```

```
    protected Peer(Parcel in) {
        peerId = in.readString();
        name = in.readString();
        status = in.readString();
    }

    public static final Creator<Peer> CREATOR = new Creator<Peer>() {
        @Override
        public Peer createFromParcel(Parcel in) { return new Peer(in); }

        @Override
        public Peer[] newArray(int size) { return new Peer[size]; }
    };


    @Override
    public int describeContents() { return 0; }

    @Override
    public void writeToParcel(Parcel dest, int flags) {
        dest.writeString(peerId);
        dest.writeString(name);
        dest.writeString(status);
    }


    public void setPeerId(String peerId) { this.peerId = peerId; }

    public String getPeerId() { return peerId; }

    public void setName(String name) { this.name = name; }

    public String getName() { return name; }

    public void setStatus(String status) { this.status = status; }

    public String getStatus() { return status; }

    @NonNull
    @Override
    public String toString() {
        return "Peer {id: " + peerId + ", name: " + name + " status " + status + "}";
    }
}
```

4.2.7 Η υπηρεσία NearbyService

Η υπηρεσία NearbyService υλοποιείται από τη κλάση NearbyService. Η κλάση αυτή επεκτείνει την κλάση Service, περιέχει όλες τις απαραίτητες μεθόδους για την αναζήτηση διαθέσιμων συσκευών, σύνδεση με αυτές και αποστολή δεδομένων. Επίσης έχει μια εσωτερική κλάση την NearbyBinder η οποία επεκτείνει την κλάση Binder η οποία επιτρέπει την σύνδεση της υπηρεσίας με το MainActivity.

Η κλάση NearbyService περιλαμβάνει υλοποιεί τις μεθόδους των κλάσεων EndpointDiscoveryCallback, ConnectionLifecycleCallback και PayloadCallback οι οποίες ανήκουν στην βιβλιοθήκη Google Nearby Connections.

Η αφηρημένη κλάση EndpointDiscoveryCallback αντιπροσωπεύει την διαδικασία ανακάλυψης νέων κόμβων. Διαθέτει τις μέθοδους onEndpointFound() η καλείται όταν ένας

κόμβος ανακαλύπτει έναν γειτονικό κόμβο, και την μέθοδο onEndpointLost η οποία καλείται όταν ένας γειτονικός κόμβος δεν είναι πλέον διαθέσιμος για συνδέση.

Η αφηρημένη κλάση ConnectionLifecycleCallback αντιπροσωπεύει τον κύκλο ζωής της διασύνδεσης ενός κόμβου με έναν ή περισσότερους κόμβους. Διαθέτει τις μέθοδους onConnectionInitiated() η καλείται όταν μια συσκευή λαμβάνει ένα αίτημα σύνδεσης, την μέθοδο onConnectionResult() η οποία καλείται όταν γίνεται αποδοχή ή απόρριψη μια σύνδεσης και την μεθοδο onDisconnected() που καλειται όταν μια συσκευη αποσυνδεεται.

Η αφηρημένη κλάσης PayloadCallback αντιπροσωπεύει την διαδικασία μεταφοράς δεδομένων από κόμβο σε κόμβο. Περιλαμβάνει τις μεθόδους onPayloadReceived() η οποία καλείται όταν ένας κόμβος λαμβάνει δεδομένα και τη μέθοδο onPayloadTransferUpdate() η δείχνει την εξέλιξη της μεταφοράς δεδομένων όπως για παράδειγμα η μεταφορά ενός αρχείου.

Επιπροσθέτως η κλάση NearbyService περιέχει μεθόδους οι οποίες αναλαμβάνουν εργασίες όπως η ανακάλυψη νέων κόμβων, αναζήτηση γεγονότων που ζητούνται από άλλους κόμβους και λοιπές μεθόδους που αναλύονται παρακάτω.

Η μέθοδος startAdvertising() δημιουργεί ένα Thread στο οποίο καλείται η μέθοδος startAdvertising() της κλάσης ConnectionsClient. Τα ορίσματα που δέχεται είναι μια ονομασία με την οποία θα είναι ορατή η συσκευή, το id της υπηρεσίας το στιγμιότυπο της κλάσης ConnectionLifecycleCallback, η στρατηγική στην οποία βασίζεται η δομή του δικτύου, Με αυτό τον τρόπο η συσκευή είναι ορατή από άλλες συσκευές και μπορεί να δέχεται αιτήματα σύνδεσης από αυτές.

Η μέθοδος startDiscovering() δημιουργεί ένα Thread στο οποίο αρχικά καλείται η καλείται η μέθοδος startDiscovering() της κλάσης ConnectionsClient και ξεκινά η διαδικασία αναζήτησης συσκευών. Τα ορίσματα που δέχεται είναι το id της υπηρεσίας το στιγμιότυπο της κλάσης EndpointDiscoveryCallback, η στρατηγική στην οποία βασίζεται η δομή του δικτύου. Επίσης καλείται η μέθοδος deleteAvailbalePeers().

Η μέθοδος stopDiscovery() διακόπτει την διαδικασία αναζήτησης συσκευών καλόντας την μέθοδο stopDiscovery() της κλάσης ConnectionsClient.

Η μέθοδος acceptConnection() δημιουργεί ένα νέο Thread στο οποίο καλεί την αντίστοιχη μέθοδο acceptConnection() του ConnectionsClient και δέχεται ένα εισερχόμενο αίτημα σύνδεσης με μια συσκευή.

Η μέθοδος rejectConnection() δημιουργεί ένα νέο Thread στο οποίο καλεί την αντίστοιχη μέθοδο rejectConnection () του ConnectionsClient και απορρίπτει ένα αίτημα σύνδεσης.

Η μέθοδος onPayloadReceived() καλείται όταν η συσκευή δέχεται δεδομένα από μια άλλη. Δημιουργεί ένα thread στο οποίο τα δεδομένα που έχουν ληφθεί, δηλαδή ένα αντικείμενο της κλάσης Payload μετατρέπονται από πίνακα bytes σε αντικείμενο της κλάσης Message. Στη συνέχεια ελέγχεται σε ποια κατηγορία μηνυμάτων ανήκει το μήνυμα. Εάν είναι μήνυμα το οποία είναι ερώτημα γεγονότων, δηλαδή το αντικείμενο είναι στιγμιότυπο της κλάσης QueryMessage. τότε στέλνεται ένα intent στο LogicSolverFragment ώστε να εμφανιστεί ένα μήνυμα στο πεδίο log και έπειτα καλείται η μέθοδος searchFacts() για ξεκινήσει η διαδικασία αναζήτησης των γεγονότων. Εάν είναι μήνυμα το οποία είναι ερώτημα για ιστορικό γεγονότων, δηλαδή το αντικείμενο είναι στιγμιότυπο της κλάσης QueryHistoryMessage, καλείται η μέθοδος replyQueryHistory(). Εάν είναι μήνυμα το οποία είναι απάντηση που περιέχει γεγονότα που βρέθηκαν, δηλαδή το αντικείμενο είναι στιγμιότυπο της κλάσης ResponseMessage, τότε τα γεγονότα προθέτονται στη λίστα tempFoundFacts. Εάν είναι μήνυμα το οποία είναι απάντηση που περιέχει το ιστορικό ερωτημάτων, δηλαδή το αντικείμενο είναι στιγμιότυπο της κλάσης ResponseHistoryMessage, τότε καλείται η μέθοδος setHistoryUpdated με όρισμα την τιμή true.

Επίσης τα δεδομένα που αφορούν το ιστορικό ερωτημάτων προσθετονται στον χάρτη queryHistory.

Η μέθοδος connectToPeer() δημιουργεί ένα Thread στο οποίο καλείται η μέθοδος stopDiscovery() διότι η διαδικασία αναζήτησης μέσω Bluetooth μειώνει σημαντικά την ενέργεια της μπαταρίας του τηλεφώνου. Έπειτα καλείται η μέθοδος requestConnection() η οποία ανήκει στην κλάση ConnectionsClient και στέλνει αίτημα σύνδεσης σε μια συσκευή με παραμέτρους την ονομασία της συσκευής του χρήστη, το id της συσκευής στην οποία στέλνεται το αίτημα το στιγμιότυπο της κλάσης ConnectionLifecycleCallback.

Η μέθοδος disconnectFromPeer() αποσυνδέει μια συσκευή από τη συσκευή του χρήστη.
Η μέθοδος disconnectFromAllPeers() αποσυνδέει όλες τις συνδεδεμένες συσκευές και καλεί την μέθοδο deleteAllPeers().

Η μέθοδος askPeers() ορίζεται και υλοποιείται με 2 τρόπους. Στην πρώτη υλοποίηση της, η μέθοδος καλείται όταν οι κανόνες δεν μπορούν να επιλυθούν με βάση την βάση γνώσης της συσκευής, οπότε πρέπει να αναζητήσει στο δίκτυο, τη γνώση η οποία δεν είναι διαθέσιμη τοπικά. Η μέθοδος δέχεται ως ορίσματα την λίστα με τα γεγονότα που δεν είναι διαθέσιμα στη βάση γνώσης. Δημιουργεί ένα Thread στο οποίο γίνεται έλεγχος εάν υπάρχουν συνδεδεμένες συσκευές. Εάν δεν υπάρχουν τότε ειδοποιεί me ένα intent το LogicSolverFragmet για την απουσία συσκευών το οποίο θα εμφανίσει αντίστοιχο μήνυμα στην οθόνη. Διαφορετικά δημιουργεί ένα αντικείμενο της κλάσης Payload με τιμή null και ένα αντικείμενο της κλάσης QueryMessage, το οποίο περιέχει το όνομα της συσκευής δηλαδή του αποστολέα, και τη λίστα με τα γεγονότα. Αυτό το αντικείμενο μετατρέπεται σε πίνακα από bytes μέσω της μεθόδου toByteArray() και το αντικείμενο payload δέχεται σαν όρισμα τον πίνακα αυτό. Εν τέλει καλείται η μέθοδος sendPayload της κλάσης ConnectionsClient με ορίσματα την λίστα τον συνδεδεμένων συσκευών και το αντικείμενο payload και ακολουθεί η κλήση της μεθόδου waitForResponses.

Στην δεύτερη υλοποίηση της μεθόδου askPeers() καλείται όταν μια συσκευή έχει λάβει αίτημα για αναζήτηση γνώσης από μια άλλη συσκευή, αλλά η γνώση της είναι ανεπαρκής. Οπότε στέλνει το αίτημα σε άλλες συσκευές του δικτύου. Η υλοποίηση της είναι παρόμοια με την πρώτη υλοποίηση με τη διαφορά ότι δέχεται ως παραμέτρους το id συσκευής που έστειλε το αίτημα**,** την λίστα με τα γεγονότα που ζητήθηκαν, μια λίστα με τα γεγονότα που δεν έχουν βρεθεί και μια λίστα που περιέχει τις συσκευές που θα λάβουν το μήνυμα.

Η μέθοδος searchFacts() έχει ως σκοπό την αναζήτηση των γεγονότων που ζητούνται από μια συσκευή του δικτύου στην τοπική βάση γνώσης. Αρχικά γίνεται έλεγχος εάν τα ζητούμενα γεγονότα έχουν ήδη βρεθεί, δηλαδή εάν υπάρχουν στην λίστα tempFoundFacts. Ένα γεγονός το οποίο έχει βρεθεί δεν αναζητείται ξανά. Έπειτα δημιουργείται ένα αντικείμενο της κλάσης InferenceEngine και εξετάζεται εάν τα αιτούμενα γεγονότα είναι διαθέσιμα στην βάση γνώσης είτε με τη μορφή γεγονότων είτε με τη μορφή συμπεράσματος ενός κανόνα. Εάν όλα τα γεγονότα εντοπιστούν τότε καλείται η μέθοδος sendResponse().Εάν τα γεγονότα δεν μπορούν να βρεθούν από τη βάση γνώσης, τότε η συσκευή στέλνει αίτημα για ιστορικό αιτημάτων στον αποστολέα, με τη μέθοδο requestQueryHistory().

Η μέθοδος waitForResponses() έχει 2 υλοποιήσεις Στην πρώτη υλοποίηση δημιουργείται ένα Runnable αντικείμενο, το waitRunnable, το οποίο ελέγχει εάν η συσκευή περιμένει απαντήσεις σε ερωτήματα που έχει αποστείλει στο δίκτυο. Ο έλεγχος πραγματοποιείται σε μια δομή επανάληψης while της οποίας η συνθήκη είναι εάν η μεταβλητή isWaiting έχει τιμή true. Εάν η συσκευή έχει λάβει όλες τις απαντήσεις τότε η μεταβλητή isWaiting παίρνει την τιμή false, καλείται η μέθοδος addNewFactsToKb() και η δομή επανάληψης τερματίζεται. Ο μέγιστος

χρόνος αναμονής για απαντήσεις είναι 2 δευτερόλεπτα οπότε δημιουργείται ένα αντικείμενο της διεπαφής ExecutorService το οποίο εκτελεί το αντικείμενο waitRunnable και τερματίζει την εκτέλεση του εάν περάσουν 2 δευτερόλεπτα. Σε περίπτωση τερματισμού καλείται η μέθοδος addNewFactsToKb() η οποία θα προσθέσει στη βάση γνώσης όσα γεγονότα κατάφερε να συλλέξει η συσκευή από το δίκτυο.

Η δεύτερη υλοποίηση της μεθόδου waitForResponses() είναι παρόμοια με την πρώτη υλοποίηση με τη διαφορά ότι η αναμονή για απαντήσεις είναι ένα δευτερόλεπτο. Σε περίπτωση που βρεθούν όλα τα γεγονότα από το δίκτυο αλλά και εάν περάσει ένα δευτερόλεπτο καλείται η μέθοδος sendResponse().

Η μέθοδος waitForHistory() καλείται όταν η συσκευή έχει υποβάλλει ερώτημα για ιστορικό ερωτημάτων και περιμένει απάντηση. Ο μέσος χρόνος αναμονής είναι ένα δευτερόλεπτο. Εάν το ιστορικό έχει ληφθεί τότε επιλέγονται οι συσκευές στις οποίες θα σταλθεί ένα ερώτημα και καλείται η δεύτερη εκδοχή της μέθοδού askPeers().

Η μέθοδος sendResponse() καλείται όταν όλα τα γεγονότα που έχουν βρεθεί ή είναι δυνατό να βρεθούν στέλνονται ως απάντηση στη συσκευή που έστειλε ένα αίτημα. Δημιουργεί ένα αντικείμενο της κλάσης ResponseMessage το οποίο περιέχει τον αποστολέα και μια λίστα με τα γεγονότα που βρέθηκαν και καλεί την μέθοδο sendPayload() της κλάσης ConnectionsClient.

Η μέθοδος requestQueryHistory() στέλνει αίτημα για το ιστορικό των ερωτημάτων στη συσκευή που έστειλε το αίτημα για γεγονότα. Δημιουργεί ένα αντικείμενο της κλάσης QueryHistoryMessage και στην συνέχεια καλείται η μέθοδος sendPayload().

Η μέθοδος replyQueryHistory() στέλνει το ιστορικό των ερωτημάτων στην συσκευή που έστειλε το αντίστοιχο αίτημα. Δημιουργεί ένα αντικείμενο της κλάσης ResponseHistoryMessage και στην συνέχεια καλείται η μέθοδος sendPayload().

Η μέθοδος addNewFactsToKb στέλνει ένα intent που περιέχει τα γεγονότα που βρέθηκαν στο δίκτυο στο LogicSolverFragment. Στη συνέχεια διαγράφει όλα τα γεγονότα από τη λίστα tempFoundFacts.

Η μέθοδος toByeArray() μετατρέπει ένα αντικείμενο της κλάσης Message σε πίνακα bytes, δηλαδή υλοποιεί την διαδικασία Serialization ενώ η μέθοδος toObject() μετατρέπει ένα πίνακα bytes σε αντικείμενο της κλάσης Message δηλαδή υλοποιεί την διαδικασία Deserialization. Ακολουθεί ο κώδικας του αρχείου NearbyService.java:

```
1   package com.drid.group_reasoning.network;
2
3   import android.app.Service;
4   import android.content.ContentValues;
5   import android.content.Intent;
6   import android.net.Uri;
7   import android.os.Binder;
8   import android.os.IBinder;
9   import android.preference.PreferenceManager;
10  import android.support.annotation.NonNull;
11  import android.util.Log;
12  import android.widget.Toast;
13
```

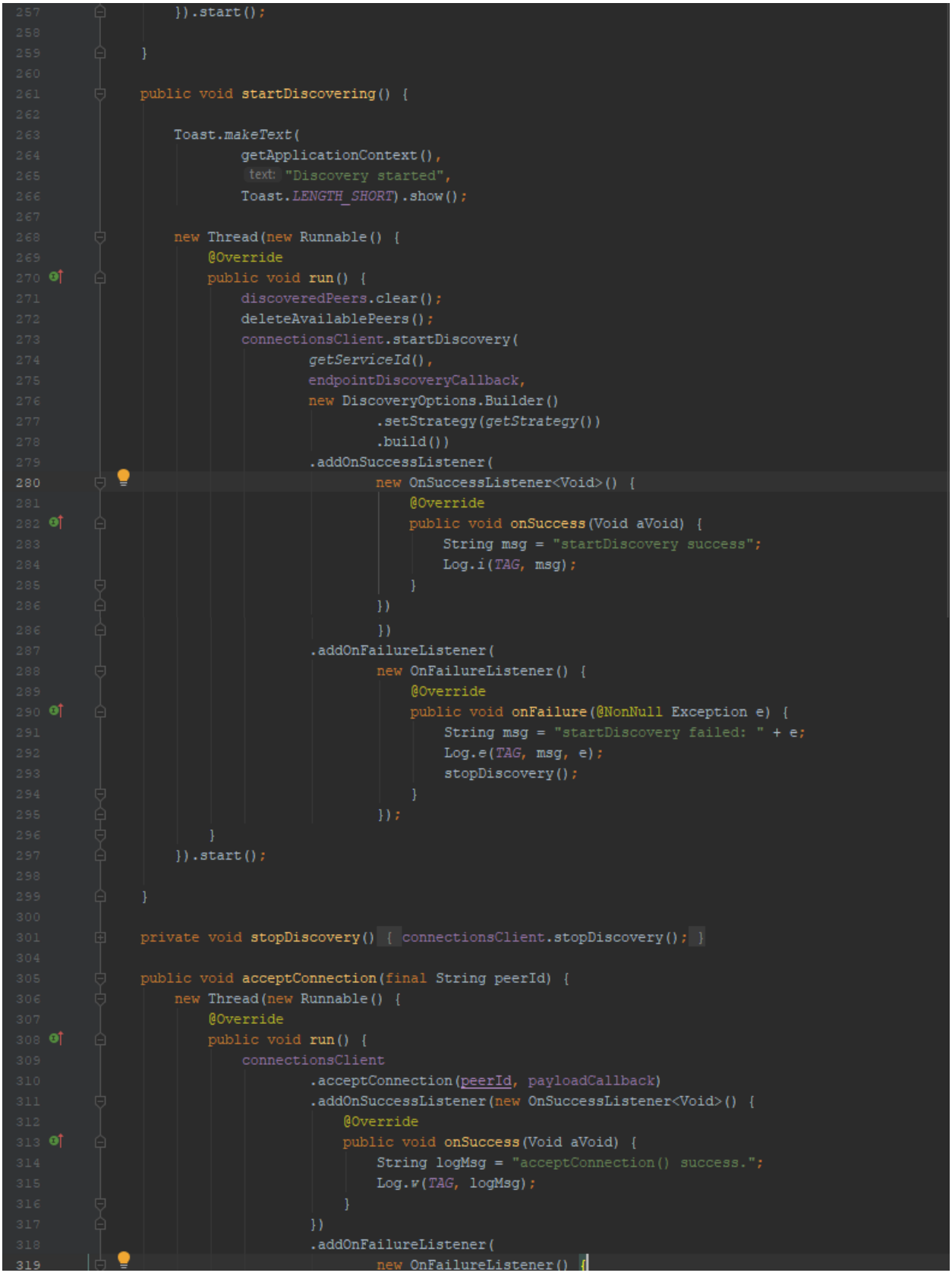
```
14  import com.drid.group_reasoning.data.contracts.PeerContract;
15  import com.drid.group_reasoning.engine.inference.InferenceEngine;
16  import com.drid.group_reasoning.engine.parser.ast.AtomicSentence;
17  import com.drid.group_reasoning.network.model.Message;
18  import com.drid.group_reasoning.network.model.Peer;
19  import com.drid.group_reasoning.network.model.QueryHistoryMessage;
20  import com.drid.group_reasoning.network.model.QueryMessage;
21  import com.drid.group_reasoning.network.model.ResponseHistoryMessage;
22  import com.drid.group_reasoning.network.model.ResponseMessage;
23  import com.google.android.gms.nearby.Nearby;
24  import com.google.android.gms.nearby.connection.AdvertisingOptions;
25  import com.google.android.gms.nearby.connection.ConnectionInfo;
26  import com.google.android.gms.nearby.connection.ConnectionLifecycleCallback;
27  import com.google.android.gms.nearby.connection.ConnectionResolution;
28  import com.google.android.gms.nearby.connection.ConnectionsClient;
29  import com.google.android.gms.nearby.connection.DiscoveredEndpointInfo;
30  import com.google.android.gms.nearby.connection.DiscoveryOptions;
31  import com.google.android.gms.nearby.connection.EndpointDiscoveryCallback;
32  import com.google.android.gms.nearby.connection.Payload;
33  import com.google.android.gms.nearby.connection.PayloadCallback;
34  import com.google.android.gms.nearby.connection.PayloadTransferUpdate;
35  import com.google.android.gms.nearby.connection.Strategy;
36  import com.google.android.gms.tasks.OnFailureListener;
37  import com.google.android.gms.tasks.OnSuccessListener;
38
39  import java.io.ByteArrayInputStream;
40  import java.io.ByteArrayOutputStream;
41  import java.io.IOException;
42  import java.io.ObjectInputStream;
43  import java.io.ObjectOutputStream;
44  import java.util.ArrayList;
45  import java.util.Collections;
46  import java.util.HashMap;
47  import java.util.List;
48  import java.util.Map;
49  import java.util.Set;
50  import java.util.concurrent.ExecutorService;
51  import java.util.concurrent.Executors;
52  import java.util.concurrent.TimeUnit;
53
54  import static com.drid.group_reasoning.network.model.Peer.AVAILABLE;
55  import static com.drid.group_reasoning.network.model.Peer.CONNECTED;
56  import static com.drid.group_reasoning.network.model.Peer.CONNECTING;
57
58  public class NearbyService extends Service {
59
60      private static final String TAG = NearbyService.class.getSimpleName();
61      private String username;
62      private static final String SERVICE_ID = "nearby_service";
63
64      private IBinder binder = new NearbyBinder();
65
66      private ConnectionsClient connectionsClient;
67
68      private static final Strategy STRATEGY = Strategy.P2P_CLUSTER;
69
70      private final Map<String, Peer> discoveredPeers = new HashMap<>();
71      private final Map<String, Peer> pendingConnections = new HashMap<>();
72      private final Map<String, Peer> connectedPeers = new HashMap<>();
73
74      Map<AtomicSentence, List<String>> queryHistory = new HashMap<>();
```

```
75
76          List<AtomicSentence> tempFoundFacts =
77                  Collections.synchronizedList(new ArrayList<AtomicSentence>());
78
79
80          private boolean isWaiting = false;
81
82          public boolean isHistoryUpdated = false;
83
84          private EndpointDiscoveryCallback endpointDiscoveryCallback =
85                  new EndpointDiscoveryCallback() {
86                      @Override
87                      public void onEndpointFound(
88                              @NonNull final String endpointId,
89                              @NonNull final DiscoveredEndpointInfo info) {
90
91                          new Thread(new Runnable() {
92                              @Override
93                              public void run() {
94                                  if (getServiceId().equals(info.getServiceId())) {
95
96                                      if (!discoveredPeers.containsKey(endpointId)) {
97                                          final Peer peer =
98                                                  new Peer(endpointId, info.getEndpointName(), AVAILABLE);
99
100                                         discoveredPeers.put(endpointId, peer);
101
102
103                                         if (!connectedPeers.containsKey(endpointId)) {
104                                             insertPeer(peer);
105                                         }
106                                     }
107                                 }
108                             }
109                         }).start();
110
111                 }
112
113                 @Override
114                 public void onEndpointLost(@NonNull String s) {
115                     Peer peer = discoveredPeers.remove(s);
116                     deletePeer(peer);
117                 }
118         };
119
120         private ConnectionLifecycleCallback connectionLifecycleCallback =
121                 new ConnectionLifecycleCallback() {
122                     @Override
123                     public void onConnectionInitiated(
124                             @NonNull final String peerId, @NonNull final ConnectionInfo info) {
125
126                         new Thread((Runnable) () → {
129                                 Log.i(TAG,  msg: "onConnectionInitiated: connecting to "
130                                         + info.getEndpointName());
131
132
133                                 final Peer peer = new Peer(peerId, info.getEndpointName(), CONNECTING);
134                                 pendingConnections.put(peerId, peer);
```

```
135
136                         if (info.isIncomingConnection()) {
137                             Intent intent = new Intent();
138                             intent.setAction("incoming_connection");
139                             intent.putExtra( name: "incoming_peer", peer);
140                             sendBroadcast(intent);
141
142                             Log.i(TAG,  msg: "Incoming connection from "
143                                     + info.getEndpointName());
144
145                         } else {
146                             acceptConnection(peer.getPeerId());
147                         }
148                     }).start();
150                 }
151
152                 @Override
153                 public void onConnectionResult(
154                         @NonNull final String peerId,
155                         @NonNull final ConnectionResolution resolution) {
156                     new Thread(new Runnable() {
157                         @Override
158                         public void run() {
159                             if (!resolution.getStatus().isSuccess()) {
160                                 String errorMsg = String.format(
161                                         "Connection failed. Received status %s.",
162                                         resolution.getStatus());
163                                 Log.w(TAG, errorMsg);
```

```
                                Peer peer = pendingConnections.remove(peerId);
                                deletePeer(peer);
                            } else {
                                Peer connectedPeer = pendingConnections.remove(peerId);
                                connectedPeer.setStatus(CONNECTED);
                                connectedPeers.put(peerId, connectedPeer);
                                updatePeer(connectedPeer);

                                String msg = String.format("Connected to %s",
                                        connectedPeer.getName());
                                Log.i(TAG, msg);
                            }
                        }
                    }).start();
                }

                @Override
                public void onDisconnected(@NonNull String peerId) {
                    Peer peer = connectedPeers.remove(peerId);
                    deletePeer(peer);
                }
            };

    private PayloadCallback payloadCallback = new PayloadCallback() {
        @Override
        public void onPayloadReceived(@NonNull String senderId, @NonNull final Payload payload) {
            Log.i(TAG,  msg: "onPayloadReceived: Payload received thread " + Thread.currentThread());
            NearbyService.this.onPayloadReceived(senderId, payload);
        }
                                deletePeer(peer);
                            } else {
                                Peer connectedPeer = pendingConnections.remove(peerId);
                                connectedPeer.setStatus(CONNECTED);
                                connectedPeers.put(peerId, connectedPeer);
                                updatePeer(connectedPeer);

                                String msg = String.format("Connected to %s",
                                        connectedPeer.getName());
                                Log.i(TAG, msg);
                            }
                        }
                    }).start();
                }

                @Override
                public void onDisconnected(@NonNull String peerId) {
                    Peer peer = connectedPeers.remove(peerId);
                    deletePeer(peer);
                }
            };

    private PayloadCallback payloadCallback = new PayloadCallback() {
        @Override
        public void onPayloadReceived(@NonNull String senderId, @NonNull final Payload payload) {
            Log.i(TAG,  msg: "onPayloadReceived: Payload received thread " + Thread.currentThread());
            NearbyService.this.onPayloadReceived(senderId, payload);
        }
```

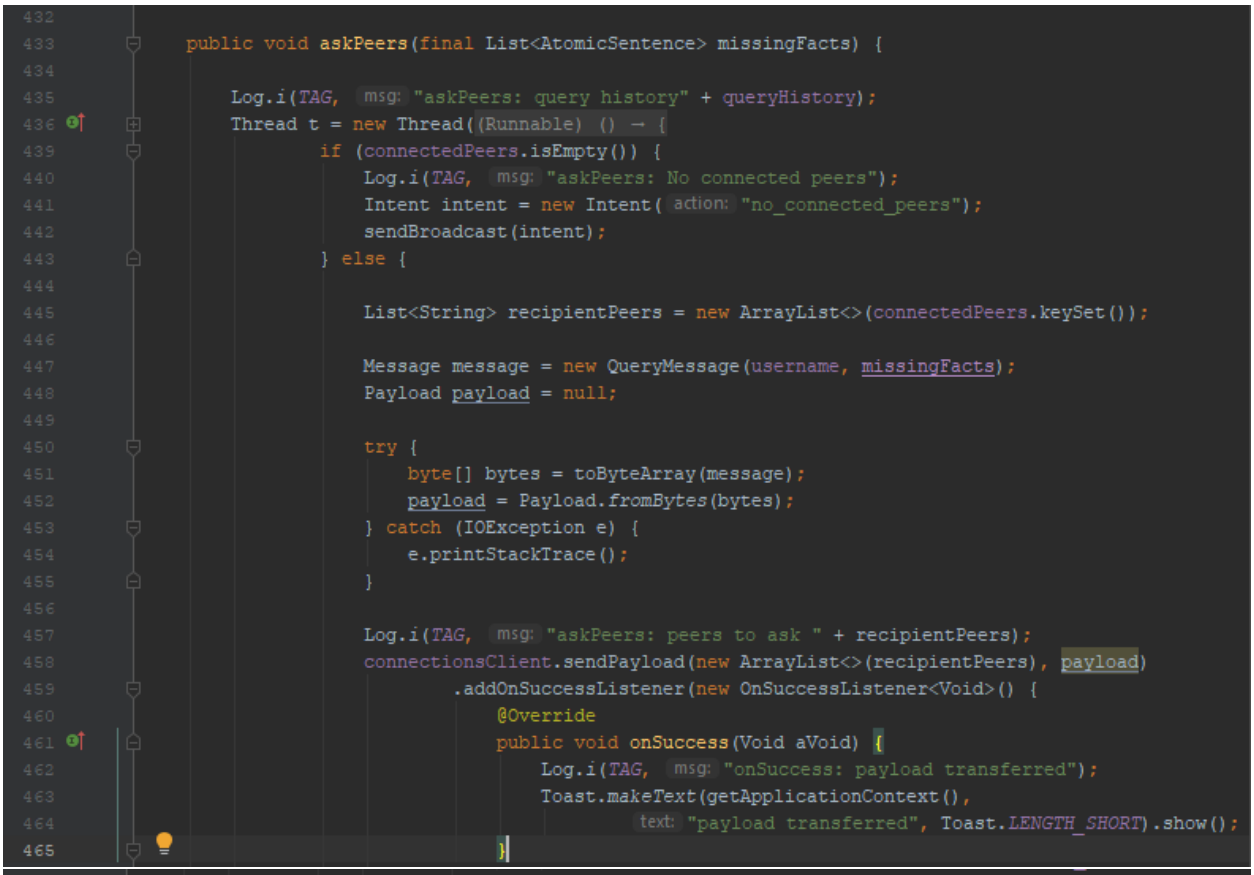
```
192         }
193
194         @Override
195         public void onPayloadTransferUpdate(
196                 @NonNull String s, @NonNull PayloadTransferUpdate payloadTransferUpdate) {
197         }
198     };
199
200     @Override
201     public void onCreate() {
202         super.onCreate();
203         connectionsClient = Nearby.getConnectionsClient( context: this);
204
205         username = PreferenceManager.getDefaultSharedPreferences(
206                 getApplicationContext()).getString( key: "username",  defValue: "");
207         startAdvertising();
208     }
209
210     @Override
211     public IBinder onBind(Intent intent) { return binder; }
214
215     @Override
216     public int onStartCommand(Intent intent, int flags, int startId) { return START_STICKY; }
219
220     @Override
221     public void onTaskRemoved(Intent rootIntent) {
222         super.onTaskRemoved(rootIntent);
223         disconnectFromAllPeers();
224         stopSelf();
225     }
226
227     private void startAdvertising() {
228         new Thread(new Runnable() {
229             @Override
230             public void run() {
231                 final String localName = getName();
232                 connectionsClient.startAdvertising(
233                         localName,
234                         getServiceId(),
235                         connectionLifecycleCallback,
236                         new AdvertisingOptions.Builder()
237                                 .setStrategy(getStrategy())
238                                 .build())
239                         .addOnSuccessListener(
240                                 new OnSuccessListener<Void>() {
241                                     @Override
242                                     public void onSuccess(Void unusedResult) {
243                                         String msg = "Now advertising endpoint " + localName;
244                                         Log.d(TAG, msg);
245                                     }
246                                 })
247                         .addOnFailureListener(
248                                 new OnFailureListener() {
249                                     @Override
250                                     public void onFailure(@NonNull Exception e) {
251                                         String msg = "startAdvertising() failed.";
252                                         Log.w(TAG, msg, e);
253                                     }
254                                 });
255
256             }
```



```
        }).start();

    }

    public void startDiscovering() {

        Toast.makeText(
                getApplicationContext(),
                text: "Discovery started",
                Toast.LENGTH_SHORT).show();

        new Thread(new Runnable() {
            @Override
            public void run() {
                discoveredPeers.clear();
                deleteAvailablePeers();
                connectionsClient.startDiscovery(
                        getServiceId(),
                        endpointDiscoveryCallback,
                        new DiscoveryOptions.Builder()
                                .setStrategy(getStrategy())
                                .build())
                        .addOnSuccessListener(
                                new OnSuccessListener<Void>() {
                                    @Override
                                    public void onSuccess(Void aVoid) {
                                        String msg = "startDiscovery success";
                                        Log.i(TAG, msg);
                                    }
                                })
                                })
                        .addOnFailureListener(
                                new OnFailureListener() {
                                    @Override
                                    public void onFailure(@NonNull Exception e) {
                                        String msg = "startDiscovery failed: " + e;
                                        Log.e(TAG, msg, e);
                                        stopDiscovery();
                                    }
                                });
            }
        }).start();

    }

    private void stopDiscovery() { connectionsClient.stopDiscovery(); }

    public void acceptConnection(final String peerId) {
        new Thread(new Runnable() {
            @Override
            public void run() {
                connectionsClient
                        .acceptConnection(peerId, payloadCallback)
                        .addOnSuccessListener(new OnSuccessListener<Void>() {
                            @Override
                            public void onSuccess(Void aVoid) {
                                String logMsg = "acceptConnection() success.";
                                Log.w(TAG, logMsg);
                            }
                        })
                        .addOnFailureListener(
                                new OnFailureListener() {
```

```
                                @Override
                                public void onFailure(@NonNull Exception e) {
                                    String errorMsg = "acceptConnection() failed.";
                                    Log.w(TAG, errorMsg, e);
                                }
                            });
                }
            }).start();
        }

        public void rejectConnection(final String peerId) {
            new Thread(new Runnable() {
                @Override
                public void run() {
                    connectionsClient.rejectConnection(peerId)
                            .addOnSuccessListener(new OnSuccessListener<Void>() {
                                @Override
                                public void onSuccess(Void aVoid) {
                                    String logMsg = "rejectConnection() success.";
                                    Log.w(TAG, logMsg);
                                }
                            });
                }
            }).start();

        }

        public void connectToPeer(final Peer peer) {
            String message = String.format("Attempting to connect to %s", peer.getName());
            Toast.makeText(getApplicationContext(), message, Toast.LENGTH_SHORT).show();
            new Thread(new Runnable() {
                @Override
                public void run() {
                    stopDiscovery();
                    connectionsClient.requestConnection(
                            getName(),
                            peer.getPeerId(),
                            connectionLifecycleCallback)
                            .addOnSuccessListener(new OnSuccessListener<Void>() {
                                @Override
                                public void onSuccess(Void aVoid) {
                                    Log.i(TAG,  msg: "requestConnection success");
                                }
                            })
                            .addOnFailureListener(new OnFailureListener() {
                                @Override
                                public void onFailure(@NonNull Exception e) {
                                    Log.e(TAG,  msg: "requestConnection failed " + e);
                                }
                            });
                }
            }).start();

        }

        public void disconnectFromPeer(Peer peer) {
            connectionsClient.disconnectFromEndpoint(peer.getPeerId());
            connectedPeers.remove(peer.getPeerId());
        }
```

```
    }

    public void disconnectFromAllPeers() {
        for (String peerId : connectedPeers.keySet()) {
            connectionsClient.disconnectFromEndpoint(peerId);
        }
        discoveredPeers.clear();
        connectedPeers.clear();
        deleteAllPeers();

    }

    private void insertPeer(Peer peer) {
        ContentValues values = new ContentValues();
        values.put(PeerContract.PeerEntry.COLUMN_PEER_ID, peer.getPeerId());
        values.put(PeerContract.PeerEntry.COLUMN_PEER_NAME, peer.getName());
        values.put(PeerContract.PeerEntry.COLUMN_PEER_STATUS, peer.getStatus());
        getContentResolver().insert(PeerContract.PeerEntry.PEER_URI, values);
    }

    private void updatePeer(Peer peer) {
        ContentValues values = new ContentValues();
        values.put(PeerContract.PeerEntry.COLUMN_PEER_STATUS, peer.getStatus());

        Uri peerUri = PeerContract.PeerEntry.PEER_URI.buildUpon()
                .appendPath(PeerContract.PATH_PEER_ID).build();

        getContentResolver().update(
                peerUri,
                values,
                where: PeerContract.PeerEntry.COLUMN_PEER_ID + "=?",
                new String[]{peer.getPeerId()});
    }

    private void deletePeer(Peer peer) {
        getContentResolver().delete(
                PeerContract.PeerEntry.PEER_URI,
                where: PeerContract.PeerEntry.COLUMN_PEER_ID + "=?",
                new String[]{peer.getPeerId()});
    }

    private void deleteAllPeers() {
        getContentResolver().delete(
                PeerContract.PeerEntry.PEER_URI,
                where: null,
                selectionArgs: null);
    }

    private void deleteAvailablePeers() {
        getContentResolver().delete(
                PeerContract.PeerEntry.PEER_URI,
                where: PeerContract.PeerEntry.COLUMN_PEER_STATUS + "=?",
                new String[]{Peer.AVAILABLE});
    }
```



```
public void askPeers(final List<AtomicSentence> missingFacts) {

    Log.i(TAG, msg: "askPeers: query history" + queryHistory);
    Thread t = new Thread((Runnable) () → {
            if (connectedPeers.isEmpty()) {
                Log.i(TAG, msg: "askPeers: No connected peers");
                Intent intent = new Intent( action: "no_connected_peers");
                sendBroadcast(intent);
            } else {

                List<String> recipientPeers = new ArrayList<>(connectedPeers.keySet());

                Message message = new QueryMessage(username, missingFacts);
                Payload payload = null;

                try {
                    byte[] bytes = toByteArray(message);
                    payload = Payload.fromBytes(bytes);
                } catch (IOException e) {
                    e.printStackTrace();
                }

                Log.i(TAG, msg: "askPeers: peers to ask " + recipientPeers);
                connectionsClient.sendPayload(new ArrayList<>(recipientPeers), payload)
                        .addOnSuccessListener(new OnSuccessListener<Void>() {
                            @Override
                            public void onSuccess(Void aVoid) {
                                Log.i(TAG, msg: "onSuccess: payload transferred");
                                Toast.makeText(getApplicationContext(),
                                        text: "payload transferred", Toast.LENGTH_SHORT).show();
                            }
                        })
                        .addOnFailureListener((e) → {
                                Log.i(TAG, msg: "onFailure: payload not transferred");
                                Toast.makeText(getApplicationContext(),
                                        text: "payload not transferred",
                                        Toast.LENGTH_SHORT).show();
                        });


                for (AtomicSentence fact : missingFacts) {
                    queryHistory.put(fact, recipientPeers);
                }
                Log.i(TAG, msg: "askPeers: query history" + queryHistory);

                waitForResponses(missingFacts);
            }
        });

    t.start();
}

private void askPeers(
        final String sender,
        final String senderId,
        final List<AtomicSentence> requestedFacts,
        final List<AtomicSentence> missingFacts,
        final List<String> peersToAsk) {
```

```
        new Thread(new Runnable() {
            @Override
            public void run() {
                Message query = new QueryMessage(username, missingFacts);

                Payload payload = null;

                try {
                    byte[] bytes = toByteArray(query);
                    payload = Payload.fromBytes(bytes);
                } catch (IOException e) {
                    e.printStackTrace();
                }


                Log.i(TAG,  msg: "askPeers': peers to ask" + peersToAsk);
                if (peersToAsk.isEmpty()) {
                    Log.i(TAG,  msg: "No peers to ask " + missingFacts);
                    Log.i(TAG,  msg: "found facts so far " + tempFoundFacts);
                    Intent intent = new Intent( action: "log_message");
                    intent.putExtra( name: "message",  value: "No peers to ask ");
                    sendBroadcast(intent);
                    sendResponse(sender, senderId, tempFoundFacts);
                } else {
                    Log.i(TAG,  msg: "Asking " + peersToAsk);
                    connectionsClient.sendPayload(peersToAsk, payload)
                            .addOnSuccessListener(new OnSuccessListener<Void>() {
                                @Override
                                public void onSuccess(Void aVoid) {
                                    Log.i(TAG,  msg: "onSuccess: payload transferred");
                                }
                            })
                            .addOnFailureListener((e) → {
                                    Log.i(TAG,  msg: "onFailure: payload not transferred");
                            });

                    Intent intent = new Intent( action: "log_message");
                    intent.putExtra( name: "message",  value: "Asking peers");
                    sendBroadcast(intent);
                    waitForResponses(sender, senderId, requestedFacts);
                }
            }
        }).start();

    }

    private void searchFacts(
            String sender, final String senderId, final List<AtomicSentence> requestedFacts) {

        Log.i(TAG,  msg: "searchFacts: Searching for missing facts " + requestedFacts);


        for (AtomicSentence fact : tempFoundFacts) {
            if (requestedFacts.contains(fact)) {
                requestedFacts.remove(fact);
            }
        }
```

```
558
559         if (requestedFacts.isEmpty()) {
560             Log.i(TAG,  msg: "searchFacts: All requested facts are already found ");
561         } else {
562             InferenceEngine engine = new InferenceEngine(getApplicationContext());
563             engine.initKnowledgeBase();
564
565             engine.solve(requestedFacts);
566
567             List<AtomicSentence> resolvedQueries = engine.getResolvedQueries();
568             List<AtomicSentence> unresolvedQueries = engine.getUnresolvedQueries();
569             Set<AtomicSentence> notInferred = engine.getMissingFacts();
570             StringBuilder builder = new StringBuilder();
571             if (unresolvedQueries.isEmpty()) {
572                 Log.i(TAG,  msg: "searchFacts: All requested facts found: " + resolvedQueries);
573                 for (AtomicSentence fact : resolvedQueries) {
574                     if (!requestedFacts.contains(fact)) {
575                         resolvedQueries.remove(fact);
576                     }
577                 }
578                 builder.append("Resolved queries:\n");
579
580                 for (AtomicSentence sentence : resolvedQueries) {
581                     builder.append("- " + sentence.getProposition() + "\n");
582                 }
583                 String log_message = builder.toString();
584                 Intent intent = new Intent( action: "log_message");
585                 intent.putExtra( name: "message", log_message);
586                 sendBroadcast(intent);
587
588                 sendResponse(sender, senderId, resolvedQueries);
589             } else {
590                 Log.i(TAG,  msg: "searchFacts: Resolved queries : " + resolvedQueries);
591                 Log.i(TAG,  msg: "searchFacts: Unresolved queries: " + unresolvedQueries);
592                 Log.i(TAG,  msg: "searchFacts: Missing facts: " + notInferred);
593
594                 builder.append("Resolved queries:\n");
595
596                 for (AtomicSentence sentence : resolvedQueries) {
597                     builder.append("- " + sentence.getProposition() + "\n");
598                 }
599                 builder.append("Unresolved queries:\n");
600
601                 for (AtomicSentence sentence : unresolvedQueries) {
602                     builder.append("- " + sentence.getProposition() + "\n");
603                 }
604                 builder.append("Missing facts:\n");
605
606                 for (AtomicSentence sentence : notInferred) {
607                     builder.append("- " + sentence.getProposition() + "\n");
608                 }
609                 String log_message = builder.toString();
610
611                 Intent intent = new Intent( action: "log_message");
612                 intent.putExtra( name: "message", log_message);
613                 sendBroadcast(intent);
614
615                 List<AtomicSentence> factsToAsk = new ArrayList<>();
616
617                 for (AtomicSentence requested : requestedFacts) {
618                     for (AtomicSentence resolved : resolvedQueries) {
619                         if (resolved.getProposition().equals(requested.getProposition())) {
```

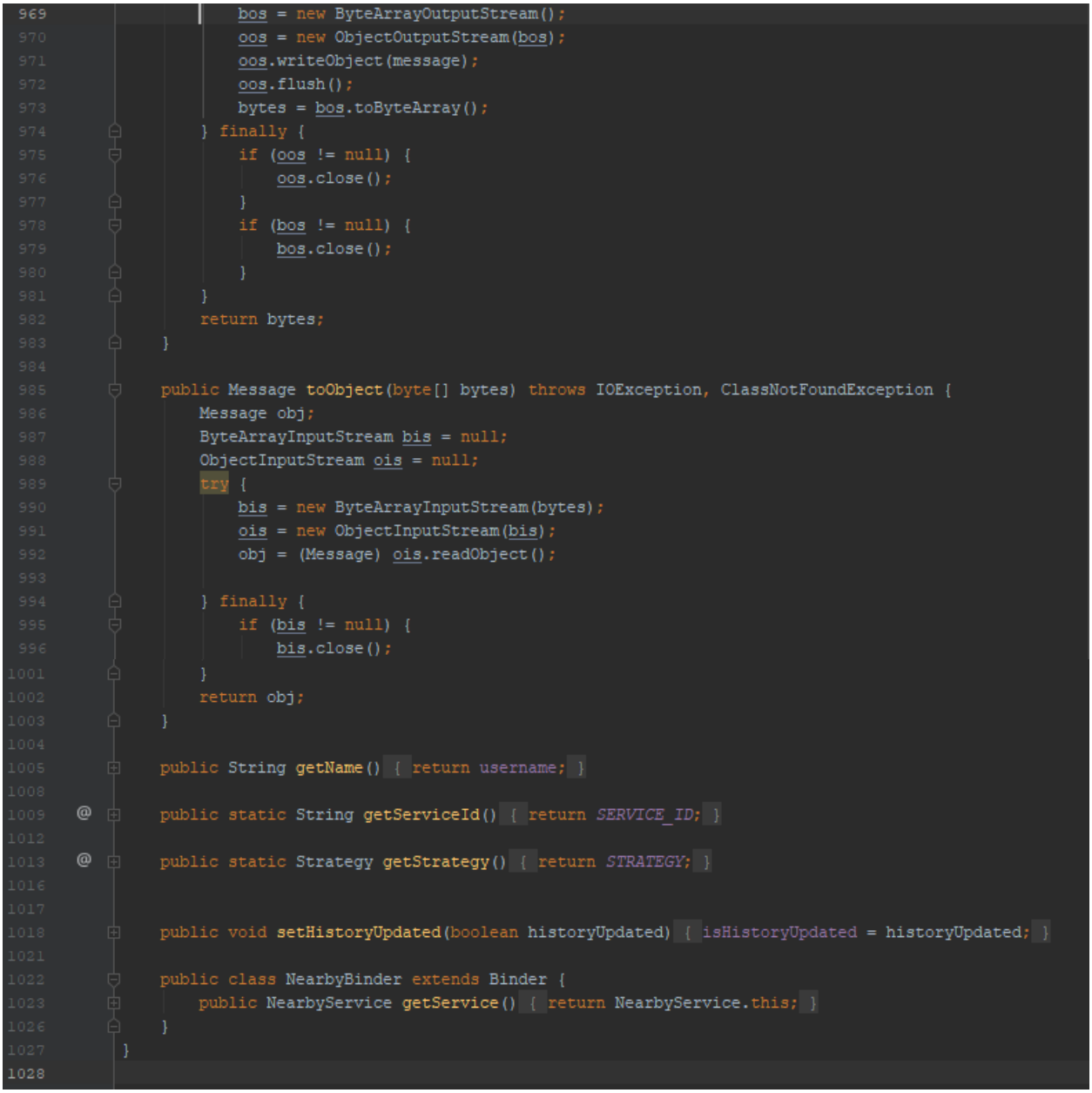
```
620                         tempFoundFacts.add(resolved);
621                     }
622                 }
623             }
624
625
626             for (AtomicSentence query : unresolvedQueries) {
627                 if (requestedFacts.contains(query)) {
628                     factsToAsk.add(query);
629                 }
630             }
631
632             requestQueryHistory(sender, senderId, requestedFacts, factsToAsk);
633         }
634     }
635 }
636
637 private void waitForResponses(final List<AtomicSentence> missingFacts) {
638     isWaiting = true;
639     Log.i(TAG,  msg: "waitForResponses: waiting for 2 seconds");
640     Runnable waitRunnable = () → {
643             while (isWaiting) {
644                 if (missingFacts.size() == tempFoundFacts.size()) {
645                     Log.i(TAG,  msg: "waitForResponses: All requested facts found");
646                     isWaiting = false;
647                     addNewFactsToKb();
648                 }
649             }
650     };
652
653     ExecutorService executor = Executors.newSingleThreadExecutor();
654     executor.submit(waitRunnable);
656     try {
657         if (!executor.awaitTermination( timeout: 2000, TimeUnit.MILLISECONDS)) {
658             executor.shutdownNow();
659             Log.i(TAG,  msg: "waitForResponses: i can't wait any longer");
660             Log.i(TAG,  msg: "waitForResponses: stopping " + tempFoundFacts);
661             Intent intent = new Intent();
662             intent.setAction("new_facts");
663             intent.putParcelableArrayListExtra(
664                     name: "facts",
665                     new ArrayList<>(tempFoundFacts));
666             sendBroadcast(intent);
667
668         }
669     } catch (InterruptedException e) {
670         executor.shutdownNow();
671     }
672 }
673
674 private void waitForResponses(
675         final String sender,
676         final String senderId,
677         final List<AtomicSentence> missingFacts) {
678     isWaiting = true;
679     Log.i(TAG,  msg: "waitForResponses: waiting for 2 seconds");
680     Runnable waitRunnable = () → {
683             while (isWaiting) {
684                 if (missingFacts.size() == tempFoundFacts.size()) {
685                     Log.i(TAG,  msg: "waitForResponses: All requested facts found");
686                     isWaiting = false;
687
688                     sendResponse(sender, senderId, tempFoundFacts);
```

```
689                     }
690                 }
691             };
693
694             ExecutorService executor = Executors.newSingleThreadExecutor();
695             executor.submit(waitRunnable);
696             executor.shutdown();
697             try {
698                 if (!executor.awaitTermination( timeout: 1000, TimeUnit.MILLISECONDS)) {
699                     executor.shutdownNow();
700                     Log.i(TAG,  msg: "waitForResponses: i can't wait any longer");
701                     sendResponse(sender, senderId, tempFoundFacts);
702                 }
703             } catch (InterruptedException e) {
704                 executor.shutdownNow();
705             }
706         }
707
708         private void onPayloadReceived(final String senderId, final Payload payload) {
709             Thread thread = new Thread((Runnable) () → {
712
713                 Log.i(TAG,  msg: "onPayloadReceived: New Message Received");
714
715                 Message message = null;
716                 try {
717                     message = toObject(payload.asBytes());
718                 } catch (IOException e) {
719                     e.printStackTrace();
720                 } catch (ClassNotFoundException e) {
721                     e.printStackTrace();
722                 }
723
724                 if (message instanceof QueryMessage) {
725                     String sender = ((QueryMessage) message).getSender();
726                     List<AtomicSentence> requestedFacts = ((QueryMessage) message).getRequestedFacts();
727                     Log.i(TAG,  msg: "Query = {Sender:" + sender
728                             + ", requesting facts: " + requestedFacts + "}");
729
730
731                     StringBuilder builder = new StringBuilder();
732                     builder.append(sender).append(" is asking for these facts:");
733                     builder.append("\n");
734                     for (AtomicSentence s : requestedFacts) {
735                         builder.append("- " + s.getProposition() + "\n");
736                     }
737
738                     Intent log_message_intent = new Intent( action: "log_message");
739                     String queryMessage = builder.toString();
740                     log_message_intent.putExtra( name: "message", queryMessage);
741                     sendBroadcast(log_message_intent);
742
743
744                     Intent notification_intent = new Intent( action: "new_query");
745                     String notificationMessage = sender + " is asking for facts";
746                     notification_intent.putExtra( name: "notification_message", notificationMessage);
747                     sendBroadcast(notification_intent);
748
749
750                     searchFacts(sender, senderId, requestedFacts);
751                 } else if (message instanceof QueryHistoryMessage) {
752                     Log.i(TAG,  msg: ((QueryHistoryMessage) message).getSender() + " is requesting history");
753                     replyQueryHistory(senderId);
754                 } else if (message instanceof ResponseMessage) {
755
```

```
                    List<AtomicSentence> receivedFacts = ((ResponseMessage) message).getFoundFacts()

                    Log.i(TAG,  msg: "Response = {Sender:"
                            + ((ResponseMessage) message).getSender()
                            + " Received facts: "
                            + ((ResponseMessage) message).getFoundFacts() + "}");

                    tempFoundFacts.addAll(receivedFacts);
                    Log.i(TAG,  msg: "onPayloadReceived: tempFoundFacts " + tempFoundFacts);


                } else if (message instanceof ResponseHistoryMessage) {
                    setHistoryUpdated(true);
                    Log.i(TAG,  msg: ((ResponseHistoryMessage) message).getSender()
                            + " has sent  history");
                    queryHistory.putAll(((ResponseHistoryMessage) message).getQueryHistory());
                }

            });

            thread.start();
        }

        private void waitForHistory(
                final String sender,
                final String senderId,
                final List<AtomicSentence> requestedFacts,
                final List<AtomicSentence> factsToAsk) {
            isWaiting = true;
            Log.i(TAG,  msg: "waitForHistory: waiting for history for 2 seconds");
            Runnable waitRunnable = () -> {
                    while (isWaiting) {
                        if (isHistoryUpdated) {
                            Log.i(TAG,  msg: "waitForHistory: query history is updated");
                            isWaiting = false;
                            List<String> peersToAsk = new ArrayList<>(connectedPeers.keySet());

                            peersToAsk.remove(senderId);

                            for (Map.Entry<AtomicSentence, List<String>> entry : queryHistory.entrySet()) {
                                AtomicSentence factAsked = entry.getKey();
                                List<String> peersAsked = entry.getValue();

                                if (factsToAsk.contains(factAsked)) {
                                    for (String peerId : peersToAsk) {
                                        if (peersAsked.contains(peerId)) {
                                            peersToAsk.remove(peerId);
                                        }
                                    }
                                }

                            }
                            askPeers(sender, senderId, requestedFacts, factsToAsk, peersToAsk);
                        }
                    }
            };

            ExecutorService executor = Executors.newSingleThreadExecutor();
            executor.submit(waitRunnable);
            executor.shutdown();
            try {
                if (!executor.awaitTermination( timeout: 1000, TimeUnit.MILLISECONDS)) {
                    executor.shutdownNow();
```

```
823                 Log.i(TAG,  msg: "waitForHistory: i can't wait any longer for history");
824             }
825         } catch (InterruptedException e) {
826             executor.shutdownNow();
827         }
828     }
829
830 @   private void sendResponse(String sender, String senderId, final List<AtomicSentence> facts) {
831
832         Log.i(TAG,  msg: "sendResponse: sending " + facts + " to " + senderId);
833
834         StringBuilder builder = new StringBuilder();
835
836         builder.append("Sending to " + sender + " these facts\n");
837
838         for (AtomicSentence sentence : facts) {
839             builder.append("- " + sentence.getProposition() + "\n");
840         }
841
842         String log_message = builder.toString();
843
844         Intent intent = new Intent( action: "log_message");
845         intent.putExtra( name: "message", log_message);
846
847         sendBroadcast(intent);
848
849         Message message = new ResponseMessage(username, facts);
850         Payload payload = null;
851
852         try {
853             byte[] bytes = toByteArray(message);
854             payload = Payload.fromBytes(bytes);
855         } catch (IOException e) {
856             e.printStackTrace();
857         }
858
859
860         connectionsClient.sendPayload(senderId, payload)
861 O↑              .addOnSuccessListener((OnSuccessListener) (aVoid) → {
864                     Log.i(TAG,  msg: "onSuccess: payload transferred");
865 O↑              }).addOnFailureListener((e) → {
869                 Log.i(TAG,  msg: "onFailure: payload not transferred");
870         });
872
873         tempFoundFacts.removeAll(facts);
874     }
875
876     private void requestQueryHistory(
877             final String sender,
878             final String senderId,
879             List<AtomicSentence> requestedFacts,
880             List<AtomicSentence> factsToAsk) {
881 O↑      new Thread((Runnable) () → {
884
885             Log.i(TAG,  msg: "requestQueryHistory: requesting query history from " + senderId);
886
887             Message message = new QueryHistoryMessage(username);
888
889             Payload payload = null;
```

```
890
891             try {
892                 byte[] bytes = toByteArray(message);
893                 payload = Payload.fromBytes(bytes);
894             } catch (IOException e) {
895                 e.printStackTrace();
896             }
897
898             connectionsClient.sendPayload(senderId, payload)
899                     .addOnSuccessListener((OnSuccessListener) (aVoid) → {
902                             Log.i(TAG,  msg: "onSuccess: payload transferred");
903                     }).addOnFailureListener((e) → {
907                     Log.i(TAG,  msg: "onFailure: payload not transferred");
908             });
910     }).start();
913
914     waitForHistory(sender, senderId, requestedFacts, factsToAsk);
915 }
916
917 private void replyQueryHistory(final String senderId) {
918     new Thread((Runnable) () → {
921
922             Log.i(TAG,  msg: "replyQueryHistory : sending query history to " + senderId);
923
924             Message message = new ResponseHistoryMessage(username, queryHistory);
925
926             Payload payload = null;
927
928             try {
930                 payload = Payload.fromBytes(bytes);
931             } catch (IOException e) {
932                 e.printStackTrace();
933             }
934
935             connectionsClient.sendPayload(senderId, payload)
936                     .addOnSuccessListener((OnSuccessListener) (aVoid) → {
939                             Log.i(TAG,  msg: "onSuccess: payload transferred");
940                     }).addOnFailureListener((e) → {
944                     Log.i(TAG,  msg: "onFailure: payload not transferred");
945             });
947
948
949     }).start();
951 }
952
953 private void addNewFactsToKb() {
954     Log.i(TAG,  msg: "addNewFactsToKb: temp facts " + tempFoundFacts);
955     Intent intent = new Intent();
956     intent.setAction("new_facts");
957     intent.putParcelableArrayListExtra(
958             name: "facts",
959             new ArrayList<>(tempFoundFacts));
960     sendBroadcast(intent);
961     tempFoundFacts.clear();
962 }
963
964 public byte[] toByteArray(Message message) throws IOException {
965     byte[] bytes;
966     ByteArrayOutputStream bos = null;
967     ObjectOutputStream oos = null;
968     try {
```

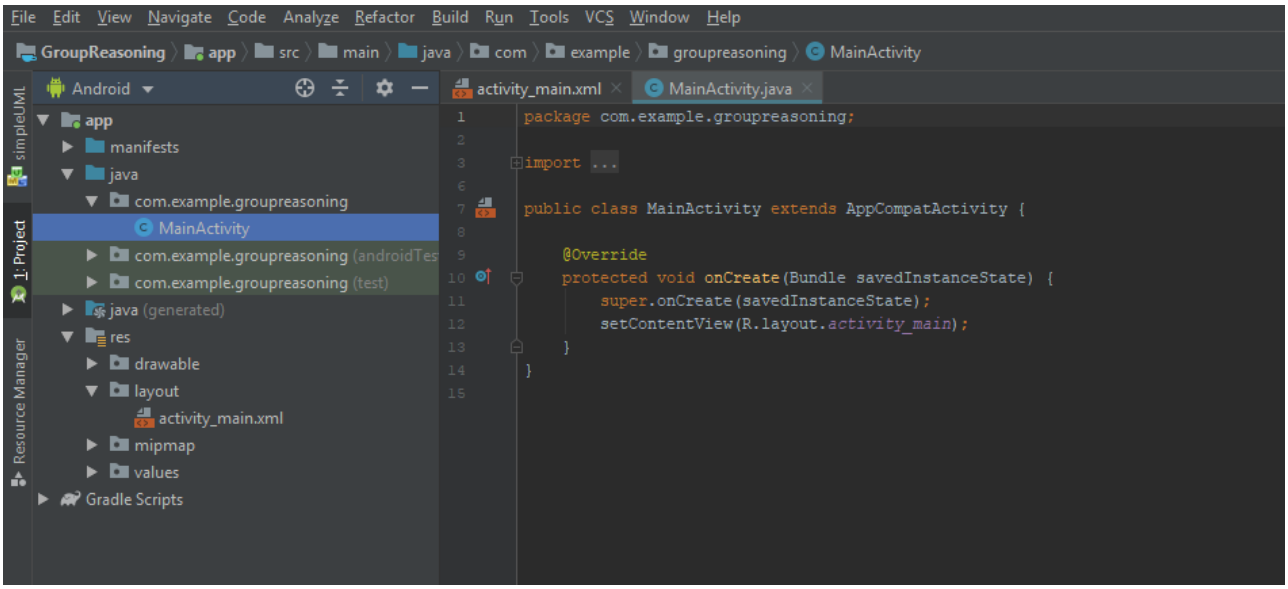


```
969                 bos = new ByteArrayOutputStream();
970                 oos = new ObjectOutputStream(bos);
971                 oos.writeObject(message);
972                 oos.flush();
973                 bytes = bos.toByteArray();
974             } finally {
975                 if (oos != null) {
976                     oos.close();
977                 }
978                 if (bos != null) {
979                     bos.close();
980                 }
981             }
982             return bytes;
983         }
984
985         public Message toObject(byte[] bytes) throws IOException, ClassNotFoundException {
986             Message obj;
987             ByteArrayInputStream bis = null;
988             ObjectInputStream ois = null;
989             try {
990                 bis = new ByteArrayInputStream(bytes);
991                 ois = new ObjectInputStream(bis);
992                 obj = (Message) ois.readObject();
993
994             } finally {
995                 if (bis != null) {
996                     bis.close();
1001             }
1002             return obj;
1003         }
1004
1005         public String getName() { return username; }
1008
1009 @       public static String getServiceId() { return SERVICE_ID; }
1012
1013 @       public static Strategy getStrategy() { return STRATEGY; }
1016
1017
1018         public void setHistoryUpdated(boolean historyUpdated) { isHistoryUpdated = historyUpdated; }
1021
1022         public class NearbyBinder extends Binder {
1023             public NearbyService getService() { return NearbyService.this; }
1026         }
1027     }
1028
```

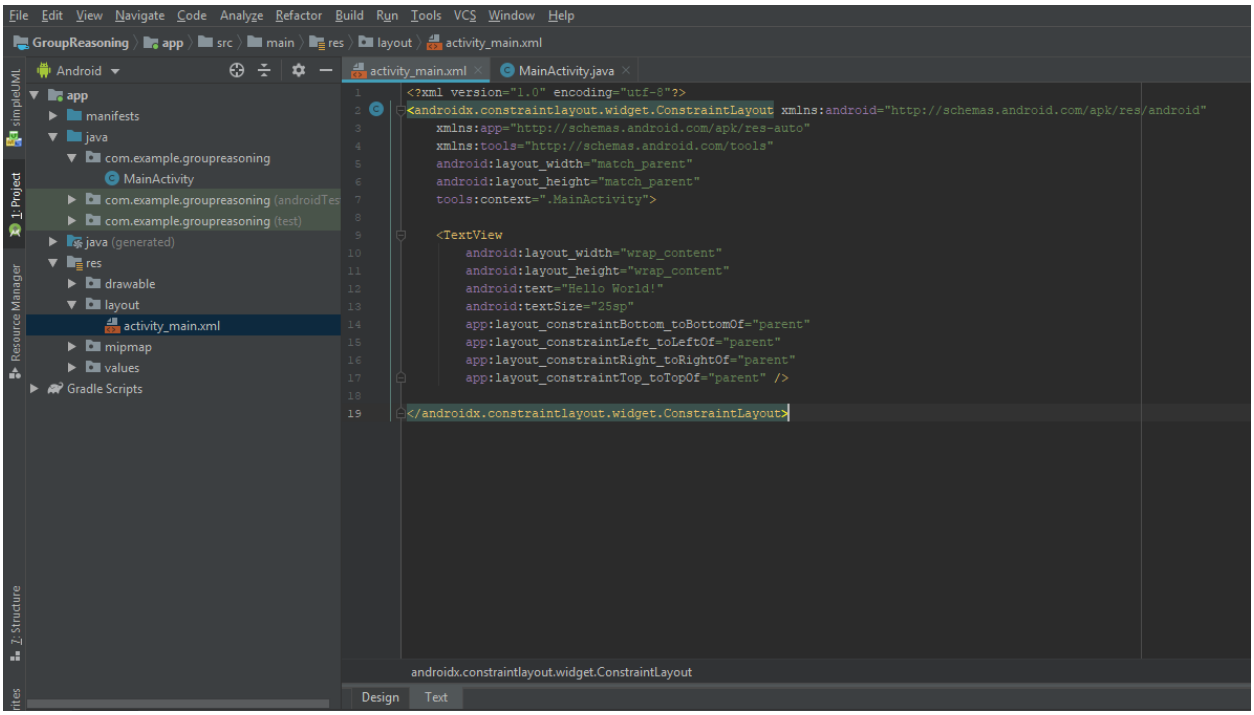
4.3 Το LoginActivity

Το Android Studio, όποτε δημιουργείται ένα νέο project δημιουργεί αυτόματα 2 αρχεία που αφορούν την κύρια δραστηριότητα. Τα αρχεία αυτά είναι το MainActivity.java στο οποίο υλοποιείται η λογική της δραστηριότητας στη γλώσσα Java καθώς και το αρχείο activity_main.xml όπου σχεδιάζεται η διεπαφή χρήστη σε γλώσσα xml. Ο προγραμματιστής έχει την δυνατότητα να τροποποιήσει το αρχείο xml στον text editor ή να χρησιμοποιήσει τον layout editor ο οποίος αναλαμβάνει την τροποποίηση του αρχείου.

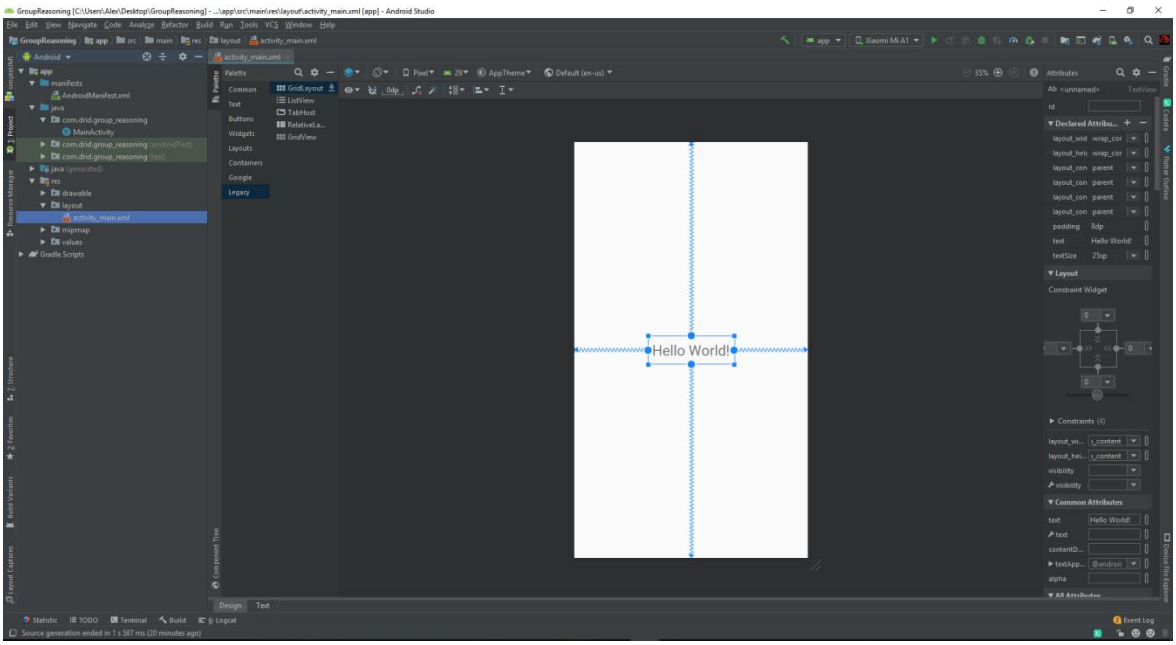
Η MainActivity είναι δηλωμένη, όπως και όλες οι δραστηριότητες στο αρχείο manifest με την διαφορά ότι επισημαίναν ότι είναι η δραστηριότητα η οποία θα εμφανίζεται όταν ξεκινά η εφαρμογή



Εικόνα 13: Το αρχείο MainActivity.java



Εικόνα 14: Το αρχείο activity_main.xml στον text editor



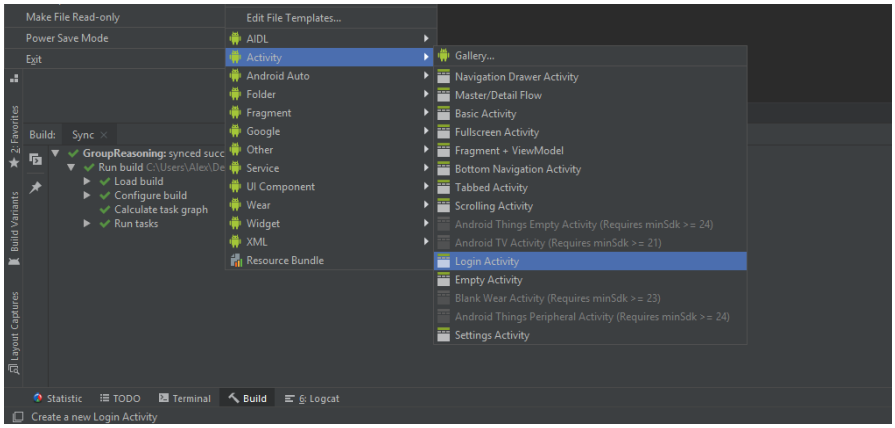
Εικόνα 15: Το αρχείο activity_main.xml στον layout editor

Στην εφαρμογή Group Reasoning η δραστηριότητα LoginActivity αντικαθιστά την MainActivity ως οθόνη εκκίνησης της εφαρμογής. Περιλαμβάνει τον τίτλο της εφαρμογής ένα πεδίο εισαγωγής κειμένου, ένα κουμπί που έχει την ένδειξη Log in και ένα TextView το οποίο προτρέπει τον χρήστη να συμπληρώσει το όνομα του στο πεδίο. Ο χρήστης αφού τυπώσει το όνομα του πατάει το κουμπί και τότε μεταβαίνει στην δραστηριότητα MainActivity η οποία αποτελεί την κυριά διεπαφή χρήστη της εφαρμογής.

Όπως φαίνεται στην εικόνα για την δημιουργία της δραστηριότητας LoginActivity ακολουθούμε τα εξής βήματα:

File -> New -> Activity -> Login Activity.

Ακολουθώντας τις οδηγίες ολοκληρώνουμε την διαδικασία δημιουργίας της δραστηριότητας και στη συνέχεια δηλώνουμε στο αρχείο manifest ότι το LoginActivity θα δραστηριότητα εκκίνησης προσθέτοντας το στοιχείο <intent-filter>.



Εικόνα 16: Δημιουργία νέου Activity

4.3.1 Το αρχείο activity_main.xml

Ένα αρχείο xml ενός activity είναι τοποθετημένο στο φάκελο layout και περιγράφει την διεπαφή χρήστη. Έχει ως στοιχείο ρίζας ένα view group όπως LinearLayout, RelavtiveLayout, FrameLayout, ConstraintLayout, CoordinatorLayout κ.α. το οποίο λέγεται και root group. Ανάλογα με τις απαιτήσεις της διεπαφής επιλέγεται το κατάλληλο view group. Στο root group μπορούν να προστεθούν components τα οποία λέγονται views ή και άλλα view groups. Όσο περισσότερα στοιχεία περιλαμβάνει μια διεπαφή, τόσο αυξάνεται η περιπλοκότητα της λειτουργίας της, ενώ υπάρχει το ενδεχόμενο ένα να επικαλύπτεται από άλλα, με αποτέλεσμα η διεπαφή να δημιουργεί σύγχυση στον χρήστη. Επομένως μια διεπαφή θα πρέπει να είναι όσο το δυνατόν πιο περιεκτική.

Σε όλα τα views ή view groups είναι απαραίτητο ορίζονται οι ιδιότητες *android:layout_width* και *android:layout_height* οι περιγράφουν αντίστοιχα το μήκος και το ύψος ενός component και έχουν τις τιμές ***match_parent*** και *wrap_content*. Η τιμή *match_parent* δηλώνει πως είτε το ύψος είτε το μήκος ενός view ή view group θα είναι ίσο με το ύψος η το μήκος του view group στο οποίο είναι εμφωλευμένο. Η τιμή *wrap_content* δηλώνει πως το μέγεθος ενός view ή view group είναι ανάλογο με το μέγεθος του περιεχομένου του. Το root group έχει πάντα τις τιμές match_parent στις ιδιότητες android:layout_width και android:layout_height διότι επεκτείνεται σε ολόκληρη την οθόνη της συσκευής. Το μέγεθος ενός view μετριέται με την μονάδα density independent pixel (dp).

Τα views στα οποία πρόκειται να γίνει επεξεργασία του περιεχόμενου τους από το αρχείο Java, όπως για παράδειγμα αλλαγή κειμένου σε ένα TextView, θα πρέπει να καταχωρείται ένα αναγνωριστικό χαρακτηριστικό το οποίο είναι μοναδικό και δηλώνεται με την ιδιότητα *android:id.* Το id αυτό καθιστά ευκολότερο τον εντοπισμό από τον κώδικα Java.

Άλλες συνήθεις ιδιότητες που χρησιμοποιούνται είναι το *android:margin* το οποίο ορίζει την απόσταση ενός view από κάποιο άλλο και η ιδιότητα *android:padding* η οποίο τοποθετεί κενά γύρο από ένα group.

Επίσης το αρχείο μπορεί να κάνει αναφορές σε πόρους, όπως κείμενα(strings), χρώματα, διαστάσεις κ.α. οι οποίοι είναι ορισμένοι σε αρχεία που βρίσκονται στον φάκελο values. Για παράδειγμα για να ορίσουμε ένα κείμενο σε ένα TextView θα πρέπει αρχικά να ορίσουμε το κείμενο στο αρχείο *strings.xml* και να κάνουμε αναφορά σε αυτό από ένα TextView δίνοντας την τιμή *@string/some_text* στην ιδιότητα του TextView android:text. Η δομή των αρχείου activity_login.xml φαίνεται στον παρακάτω κώδικα

```xml
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    tools:context=".ui.activities.LoginActivity">


    <LinearLayout
        android:id="@+id/layout"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:padding="64dp"
        android:orientation="vertical">


        <TextView
            android:id="@+id/title"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:gravity="center"
            android:padding="16dp"
            android:text="Group Reasoning"
            android:textColor="@android:color/black"
            android:textSize="25sp"
            app:fontFamily="@font/boogaloo" />

        <android.support.design.widget.TextInputLayout
            android:id="@+id/name_text_input"
            style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox.Dense"
            app:boxStrokeColor="@color/colorPrimaryLight"
            app:boxStrokeWidth="8dp"
            android:layout_width="match_parent"
            android:layout_height="wrap_content">

            <android.support.design.widget.TextInputEditText
                android:id="@+id/username_edit"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:hint="Your name"
                android:inputType="textAutoComplete"
                android:textSize="18sp" />

        </android.support.design.widget.TextInputLayout>


        <android.support.design.button.MaterialButton
            android:id="@+id/save_btn"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="8dp"
            android:text="LOG IN"
            android:textSize="18sp" />
    </LinearLayout>
```

```
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@+id/layout"
        android:gravity="center"
        android:padding="12dp"
        android:text="Type your name and log in"
        />

</RelativeLayout>
```

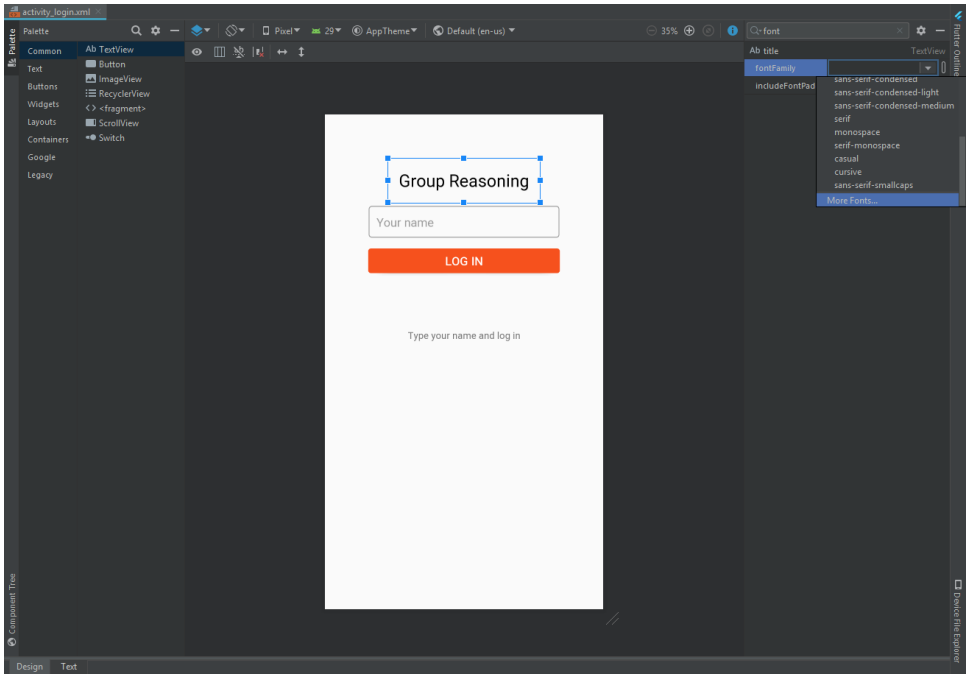
Η διεπαφή ενός του Login activity αποτελείται από τα εξής στοιχεία:

Ένα root group της μορφής RelativeLayout το οποίο έχει εμφωλευμένα τα στοιχεία

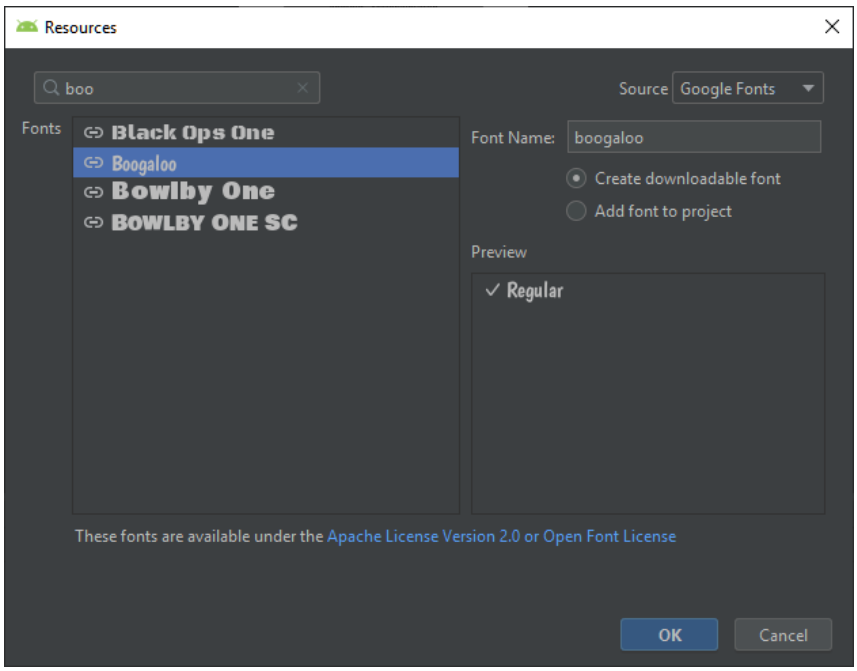
- ένα view group της μορφής LinearLayout το οποίο περιέχει
    - ένα πεδίο εισαγωγής κειμένου στο οποίο ο χρήστης εισάγει ένα όνομα
    - ένα κουμπί το οποίο όταν πατηθεί γίνεται μετάβαση στο MainActivity

- ένα view της μορφής TextView το οποίο προτρέπει τον χρήστη να εισάγει ένα όνομα συνδεθεί στην εφαρμογή. Επισημαίνεται ότι δεν χρειάζεται κωδικός πρόσβασης.

Το TextView στο οποίο εμφανίζεται ο τίτλος της εφαρμογής χρησιμοποεί τη γραμματοσειρά Boogaloo η οποία δεν ήταν προ-εγκατεστημένη με την πλατφόρμα του Android Studio και προστέθηκε χρησιμοποιώντας τον layout editor ο οποίος πραγματοποιεί την λήψη της από το διαδίκτυο. Μετά τη λήψη η γραμματοσειρά αποθηκεύεται στο φάκελο font

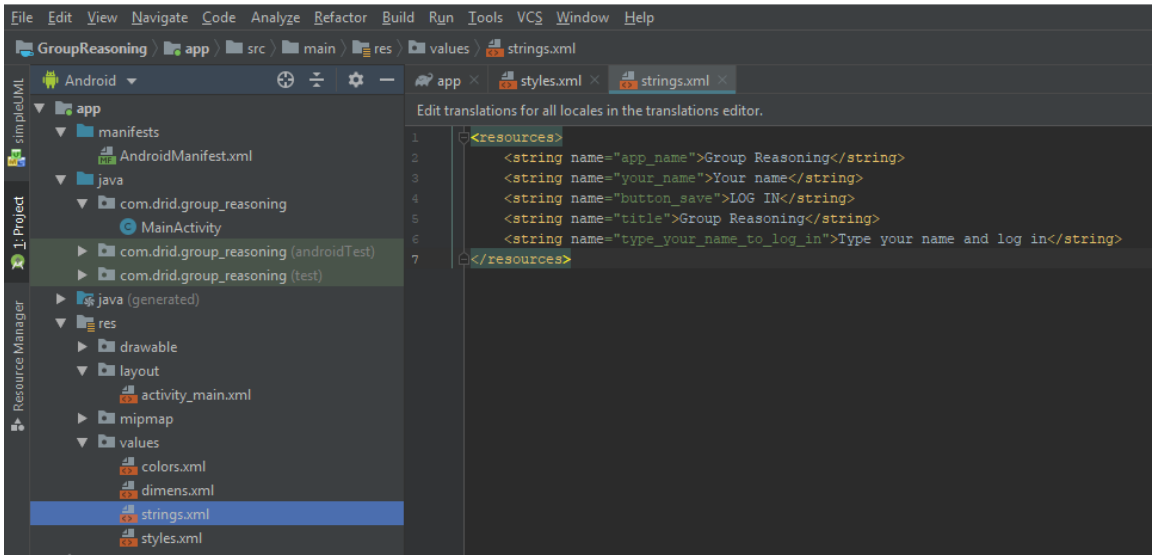


Εικόνα 17: Προσθήκη γραμματοσειράς από τον Layout Editor

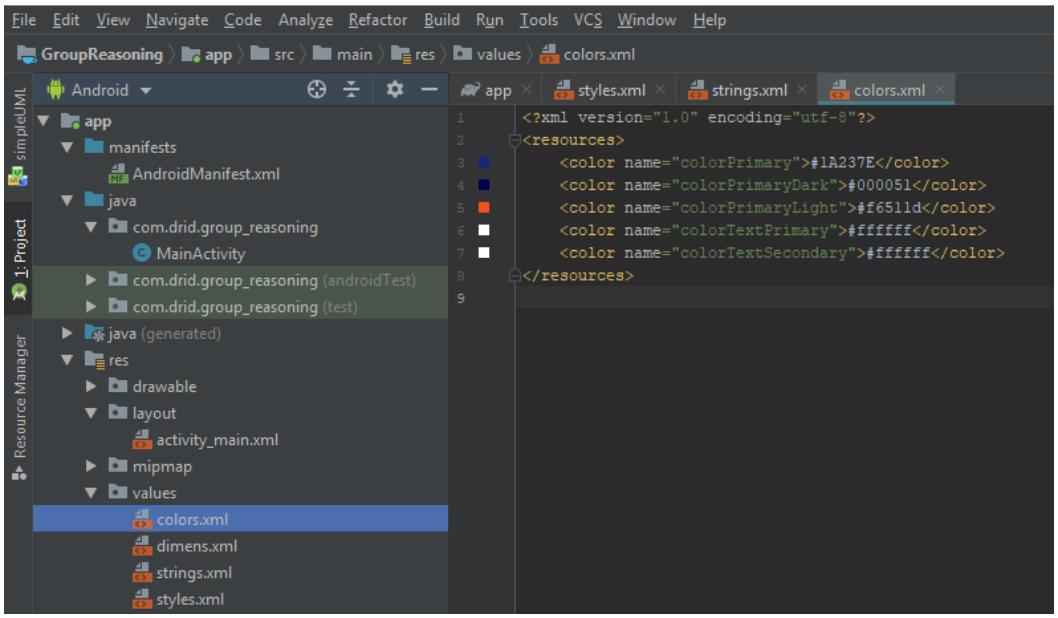


Εικόνα 18: Λήψη της γραμματοσειράς Boogaloo

Τα κειμενα και χρωματα που χρησιμοποιούνται αρχείο στο activity_login είναι δηλωμένα στα αρχεία strings και colors.xml

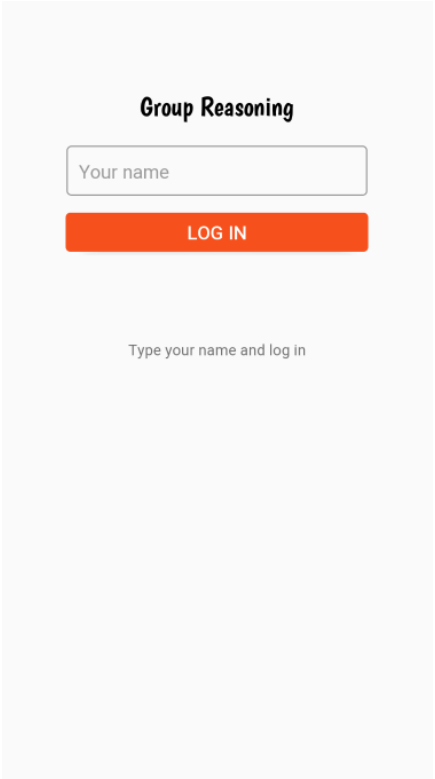


Εικόνα 19: Δήλωση κειμένων που χρησιμοποιούνται αρχείο στο activity_login.xml



Εικόνα 20: Δήλωση χρωμάτων στο αρχείο στο colors.xmlT

Η τελική μορφή της διεπαφής του LoginActivity φαίνεται στην παρακάτω εικόνα



Εικόνα 21: Η διεπαφή χρήστη του Login Activity

4.3.2 Το αρχείο LoginActivity.java

Το αρχείο LoginActivity.java υλοποιεί τη λογική της διεπαφής που είναι σχεδιασμένη στο αρχείο activity_login.xml. Η κλάση LoginActivity είναι ουσιαστικά μια υποκλάση της AppCompatActivity. Έχει 2 μεταβλητές στιγμιότυπου (instance variables) userEdit και saveBtn οι οποίες είναι μεταβλητές τύπου EditText και Button αντίστοιχα. Στην μέθοδο *onCreate()* αρχικά καλείται η μέθοδος *onCreate()* της κλάσης AppCompatActivity και έπειτα με τη μέθοδο *setContentView(),* αντλούνται οι πληροφορίες που σχετίζονται με την διεπαφή. Η μέθοδος *setContentView()* δέχεται την παράμετρο *R.layout.activity_main* η οποία αναφέρεται στο αρχείο activity_main.xml το οποίο είναι αποθηκευμένο στον φάκελο layout.

Ακολουθεί ο έλεγχος για το εάν το όνομα που έχει εισάγει ο χρήστης είναι αποθηκευμένο στο αρχείο preferences.xml. Εάν δεν είναι αποθηκευμένο τότε γίνεται η αρχικοποίηση των μεταβλητών *userEdit* και *saveBtn* με τη χρήση της μεθόδου *findViewById(),* η οποία παίρνει την παράμετρο το id των view αυτών όπως έχουν δηλωθεί στο xml αρχείο, π.χ. το *R.id.username_edit* αναφέρεται στο *TextInputEditText* (υποκλάση της *EditText*) το οποίο έχει δηλωθεί στο αρχείο activity_login.xml με id το *username_edit.* Στη συνέχεια καταχωρείται ένας listener στο κουμπί *saveBtn* ο οποίος καλεί την μέθοδο *saveData()* όταν το κουμπί πατηθεί μέθοδος *saveData()* αποθηκεύει το κείμενο που έχει εισάγει ο χρήστης στο πεδίο *userEdit* και αυτό αποθηκεύεται στο αρχείο preferences.xml. Αφού ολοκληρωθεί η απόθεσή τότε δημιουργείται ένα Intent το οποίο δηλώνει στο σύστημα «πρόθεση» για μετάβαση από το LoginActivity στο MainActivity και καλείται η μέθοδος finish η οποία τερματίζει το LoginActivity.

Η παραπάνω διαδικασία εκτελείται μόνο την πρώτη φορά που χρησιμοποιείται η εφαρμογή, διότι εάν το όνομα είναι ήδη αποθηκευμένο, γίνεται άμεσα η μετάβαση στο MainActivity, με αποτέλεσμα ο χρήστης να μην αλληλεπιδρά με το LoginActivity. Η LoginActivity υλοποιείται με τον παρακάτω κώδικα στη γλώσσα Java

```java
package com.drid.group_reasoning.ui.activities;

import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.preference.PreferenceManager;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;

import com.drid.group_reasoning.R;

public class LoginActivity extends AppCompatActivity {

    private EditText userEdit;
    private Button saveBtn;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);

        SharedPreferences pref = PreferenceManager.getDefaultSharedPreferences( context: this);

        if (pref.getBoolean( key: "activity_executed",  defValue: false)) {
            Intent intent = new Intent( packageContext: this, MainActivity.class);
            startActivity(intent);
            finish();
        }

        userEdit = findViewById(R.id.username_edit);
        saveBtn = findViewById(R.id.save_btn);
        saveBtn.setOnClickListener((v) -> { saveData(); });

    }

    private void saveData(){
        SharedPreferences pref = PreferenceManager.getDefaultSharedPreferences( context: this);
        SharedPreferences.Editor editor = pref.edit();
        editor.putBoolean("activity_executed", true);

        String username = userEdit.getText().toString();
        editor.putString("username", username);
        editor.apply();
        Intent intent = new Intent(getApplicationContext(), MainActivity.class);
        startActivity(intent);
        finish();
    }
}

```

4.4 To MainActivity

Η δραστηριότητα Main activity αποτελεί την κύρια διεπαφή χρήστη της εφαρμογής Group Reasoning. Περιλαμβάνει ένα πλαίσιο ή container στο οποίο σχεδιάζεται η διεπαφή και

ένα μενού πλοήγησης (BottomNavigationView) το οποίο τοποθετείται στο κάτω μέρος της οθόνης και διαθέτει 3 επιλογές Peers, Knowledge και Logic Solver. Σε κάθε μια από τις επιλογές του μενού αντιστοιχεί ένα Fragment καθένα από τα οποία δημιουργείται και εμφανίζεται εντός του container ανάλογα με την επιλογή του χρήστη. Κατά την εκκίνηση του Main activity, η επιλογή Peers είναι προκαθορισμένη, οπότε ο container εμφανίζει την διεπαφή του Peer list fragment. Έπειτα γίνεται έλεγχος για το αν έχουν εγκριθεί οι απαραίτητες άδειες χρήστης υπηρεσιών του συστήματος που είναι απαραίτητες για την λειτουργία της εφαρμογής να έχει η εφαρμογή όπως Wi-Fi, Bluetooth και τοποθεσίας. Όλες οι άδειες χρήσεις έχουν δηλωθεί στο αρχείο manifest, αλλά η άδεια χρήσης της τοποθεσίας πρέπει να δοθεί ρητά από τον χρήστη. Για το λόγο αυτό εμφανίζεται ένα προειδοποιητικό μήνυμα στην οθόνη, σε περίπτωση που ο χρήστης δεν έχει εγκρίνει την συγκεκριμένη άδεια.

Όταν ολοκληρώνεται η διαδικασία δημιουργίας του MainActivity, τότε γίνεται σύνδεση ή binding με την υπηρεσία NearbyService,η οποία θα αναλυθεί σε επόμενο κεφάλαιο και στη συνέχεια γίνεται καταχώρηση ενός BroadcastReceiver ο οποίος λαμβάνει δεδομένα από την προαναφερθείσα υπηρεσία. Επίσης το MainActivity αποτελεί κατά κάποιο τρόπο τον δίαυλο επικοινωνίας των Fragments Peer list fragment και Logic solver fragment με την υπηρεσία NearbyService, διότι τα Fragments αυτά δεν έχουν πρόσβαση στην υπηρεσία.

4.4.1 Το αρχείο activity_main.xml

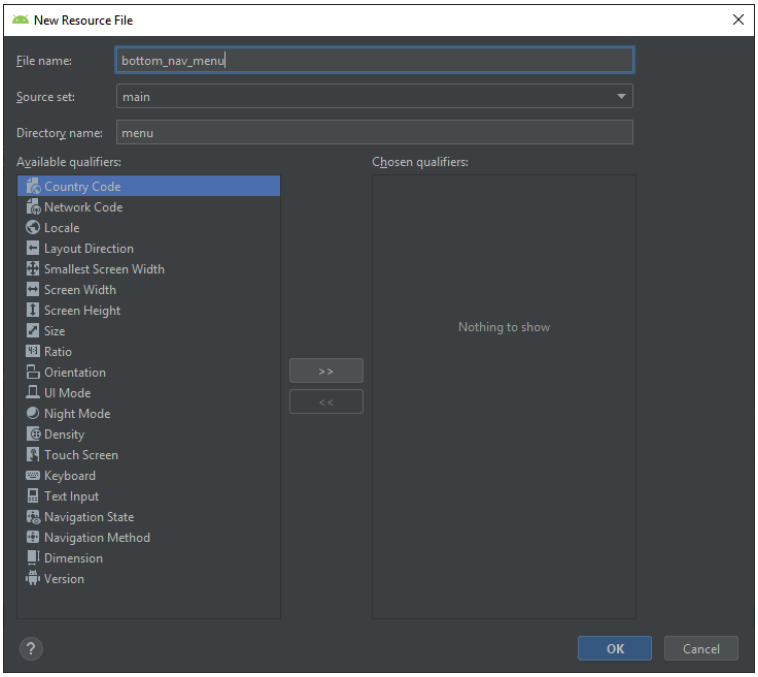
Το αρχείο activity_main.xml περιγράφει το σχεδιασμό της διεπαφής του MainActivity. Η διεπαφή του MainActivity περιλαμβάνει:

- ένα root view τύπου RelativeLayout
- μια μπάρα εργαλείων (Toolbar) στην οποία εμφανίζεται ο τίτλος της εφαρμογής
- ένα view group τύπου FrameLayout εντός του οποίου σχεδιάζεται ένα Fragment
- ένα view τύπου BottomNavigationView από το οποίο γίνεται η πλοήγηση ανάμεσα στα Fragments.
- ένα view το οποίο διαχωρίζει το FrameLayout από το BottomNavigationView.

Για το BottomNavigationView δημιουργήθηκε ένα αρχείο το οποίο ορίζει τα αντικείμενα του μενού με ονομασία bottom_navi_menu.xml καθώς και το drawble αρχείο με ονομασία bottom_navigation_colors.xml το οποίο ορίζει έναν selector ο οποίος επιλέγει τι χρώμα θα έχει ένα αντικείμενο του BottomNavigationView όταν είναι επιλεγμένο και όταν δεν είναι.

Επιπλέον για το διαχωριστικό view έχει δημιουργηθεί ένα αρχείο με όνομα shadow.xml, το σχεδιάζει ένα εφέ σκιάς.

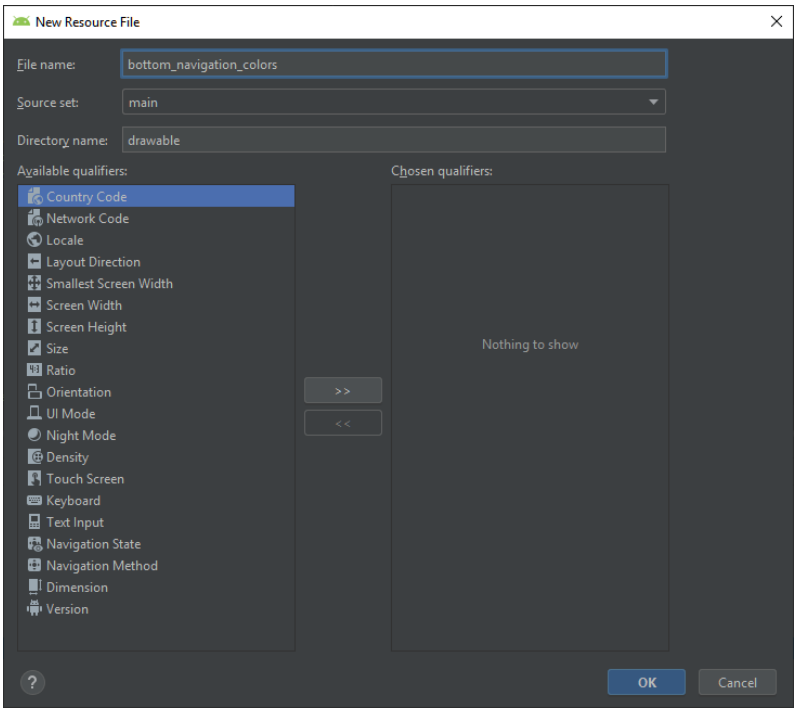
Η δημιουργία των αρχείου bottom_navi_menu.xml, bottom_navigation_colors.xml και shadow.xml όπως και οι αντίστοιχοι κώδικες σε xml φαίνονται παρακάτω:



Εικόνα 22: Δημιουργία του αρχείου bottom_navi_menu.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item
        android:id="@+id/nav_nearby_devices"
        android:icon="@drawable/ic_peers_white_24dp"
        android:title="Peers" />
    <item
        android:id="@+id/nav_knowledge"
        android:icon="@drawable/ic_knowledge_base"
        android:title="Knowledge" />
    <item
        android:id="@+id/nav_logic_resolver"
        android:icon="@drawable/ic_logic_solver"
        android:title="Logic Solver" />
</menu>
```

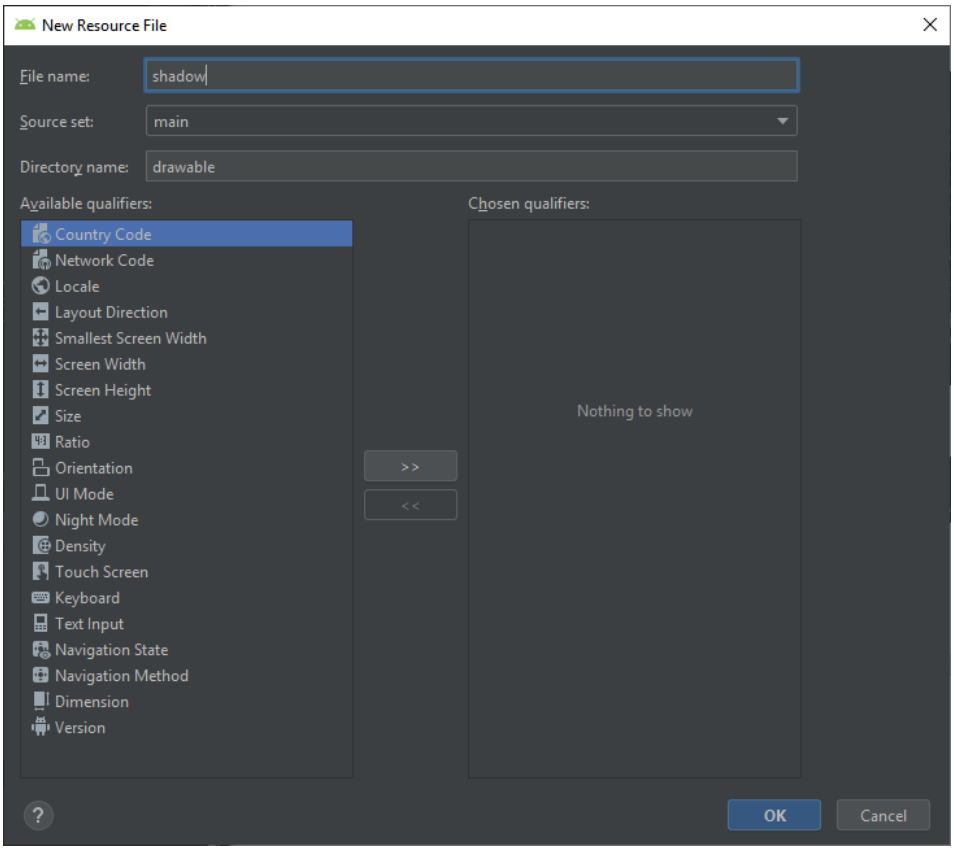
Ο κώδικας του αρχείου bottom_nav_menu



Εικόνα 23: Δημιουργία του drawable αρχείου bottom_navigation_colors.xml

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">

    <item
        android:state_checked="true"
        android:color="@color/colorPrimary" />
    <item
        android:state_checked="false"
        android:color="#919496" />

</selector>
```

Ο κώδικας του αρχείου bottom_navigation_colors



Εικόνα 24: Δημιουργία του drawable αρχείου shadow.xml

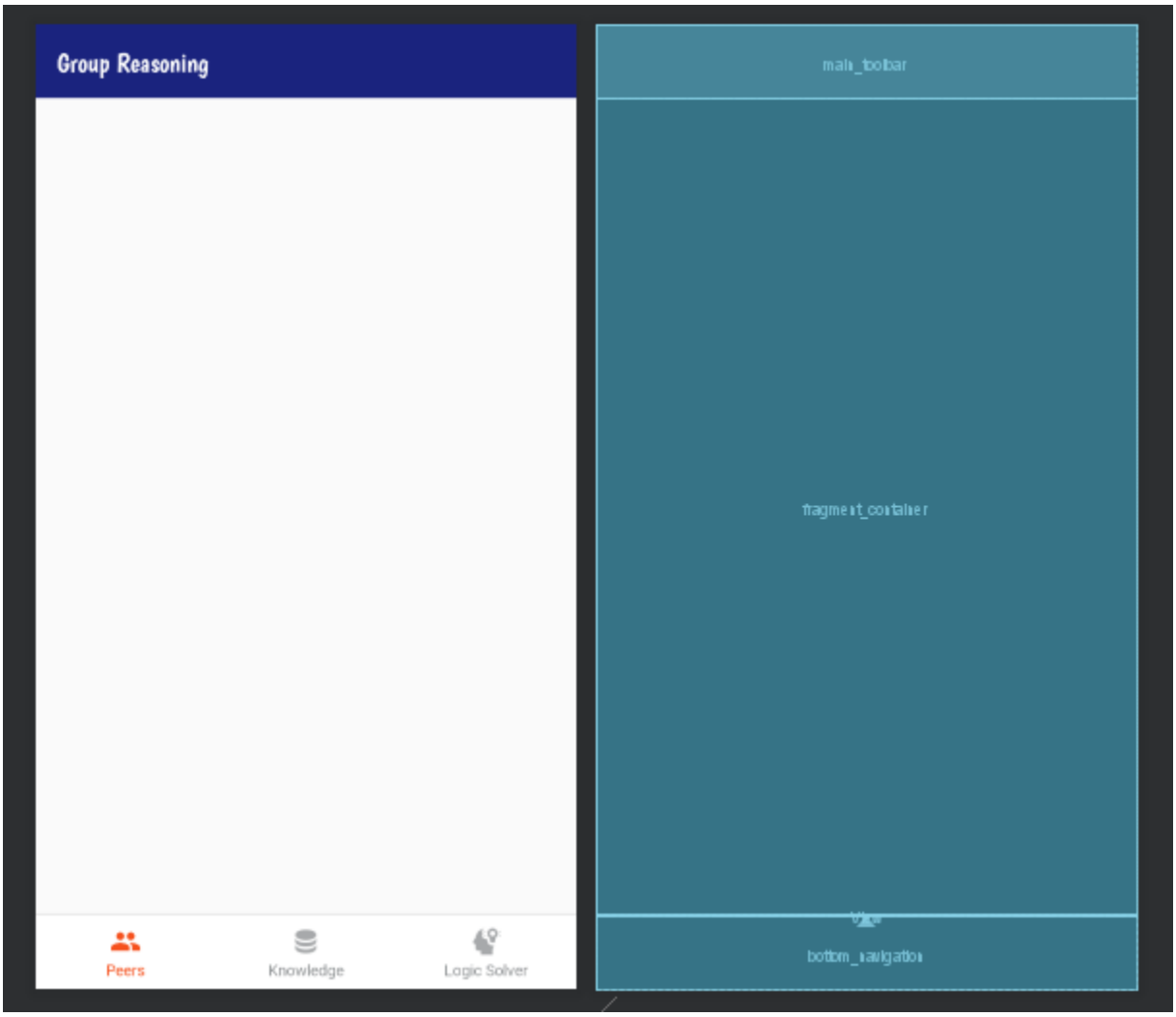
```xml
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android">
    <gradient
        android:startColor="#1F000000"
        android:endColor="@android:color/transparent"
        android:angle="90" />
</shape>
```

Ο κώδικας του αρχείου shadow.xml

Ακολουθούν ο τελικός κώδικας του αρχείου activity_main.xml και η τελική μορφή της διεπαφής του MainActivity

```xml
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".ui.activities.MainActivity"
    tools:menu="@menu/bottom_nav_menu">

    <android.support.design.widget.AppBarLayout
        android:id="@+id/app_bar"
        android:layout_width="match_parent"
        android:layout_height="wrap_content">

        <android.support.v7.widget.Toolbar
            android:id="@+id/main_toolbar"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_alignParentTop="true"
            android:background="@color/colorPrimary"
            app:title="Group Reasoning"
            android:minHeight="?attr/actionBarSize"
            android:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar"
            app:popupTheme="@style/ThemeOverlay.AppCompat.Light"
            app:titleTextAppearance="@style/TitleAppearance">

        </android.support.v7.widget.Toolbar>

    </android.support.design.widget.AppBarLayout>
```

```xml
    <FrameLayout
        android:layout_below="@+id/app_bar"
        android:id="@+id/fragment_container"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_above="@id/bottom_navigation"
        app:layout_behavior="android.support.design.widget.AppBarLayout$ScrollingVie..."
        tools:context=".ui.activities.MainActivity"/>

    <View
        android:layout_width="match_parent"
        android:layout_height="2dp"
        android:layout_above="@id/bottom_navigation"
        android:background="@drawable/shadow" />

    <android.support.design.widget.BottomNavigationView
        android:id="@+id/bottom_navigation"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:background="@color/bottom_nav_color"
        app:itemIconTint="@drawable/bottom_navigation_colors"
        app:itemTextColor="@drawable/bottom_navigation_colors"
        app:labelVisibilityMode="auto"
        app:menu="@menu/bottom_nav_menu" />

</RelativeLayout>
```



Εικόνα 25: Η διεπαφή χρήστη του MainActivity

4.4.2 Το αρχείο MainActivity.java

Η λογική του MainActivity υλοποιείται με τη χρήση της κλάσης MainActivity. Η κλάση MainActivity υλοποιεί 2 interfaces που ανήκουν στις κλάσεις PeerListFragment και LogicSolverFragment. Με αυτόν τον τρόπο τα 2 αυτά Fragments μπορούν να επικοινωνούν με την υπηρεσία NearbyService. Οι μέθοδοι που υλοποιούνται θα αναλυθούν σε επόμενο κεφάλαιο.

Στην μέθοδο onCreate() αρχικοποιούνται όλα τα απαραίτητα views του activity, καθώς και τα fragments που θα εμφανίζονται στον fragment container. Αυτά τα fragments προστίθενται σε έναν Fragment manager ο οποίος χρησιμοποιείται για την διαχείριση αυτών των fragment. Έπειτα γίνεται καταχώρηση ενός click listener στο bottom navigation view όπου ανάλογα με το αντικείμενο του μενού επιλέγεται ο fragment manager θα κρύβει από την οθόνη το εμφανιζόμενο fragment και θα το αντικαθιστά με το fragment που αντιστοιχεί με την επιλογή

του χρήστη. Για παράδειγμα εάν ο χρήστης βλέπει στην οθόνη του το Knowledge fragment, τότε όταν επιλέξει το αντικείμενο Logic solver από το μενού τότε ο fragment manager κρύβει το Knowledge fragment από τον fragment container και εμφανίζει το Logic solver fragment. Επιπλέον γίνεται καταχώρηση ενός broadcast receiver ο οποίος λαμβάνει μηνύματα ή intents από την υπηρεσία NearbyService και ένα intent filter το οποίο φιλτράρει αυτά τα intents

Όταν ολοκληρωθεί η εκτέλεση της onCreate() τότε καλείται η μέθοδος onStart() στην οποία γίνεται έλεγχος αν έχουν εγκριθεί οι άδειες χρήσης υπηρεσιών του συστήματος. και γίνεται εκκίνηση της υπηρεσίας NearbyService.

Η μεταβλητή στιγμιότυπου serviceConnection υλοποιεί την σύνδεση ή binding του MainActivity με την NearbyService, ούτως ώστε το MainActivity να μπορεί να καλεί της μεθόδους της υπηρεσίας αυτής. Με την διασύνδεση εξασφαλίζεται ότι η υπηρεσία NearbyService θα έχει τον ίδιο κύκλο ζωής με το MainActivity.

Επίσης η μεταβλητή στιγμιότυπου receiver υλοποιεί την λογική του broadcast receiver που έχει καταχωρηθεί στην μέθοδο onCreate(). Ειδικότερα ελέγχει την ιδιότητα ή δράση (action) του intent και εκτελεί το αντίστοιχο κομμάτι κώδικα. Για παράδειγμα στην μέθοδο onReceive() ο broadcast receiver δέχεται μόνο intents που δηλώνουν μια εισερχόμενη αίτηση διασύνδεσης με μια άλλη συσκευή. Όταν λαμβάνεται αυτό το intent, τότε εμφανίζεται ένα παράθυρο ή alert dialog το οποίο προτρέπει τον χρήστη να δεχτεί ή να απορρίψει την διασύνδεση με την συσκευή που έστειλε το αίτημα. Εάν ο χρήστης εγκρίνει την σύνδεση τότε καλείται η μέθοδος insertPeer() η οποία αποθηκεύει τα στοιχεία της συνδεδεμένης συσκευής στη βάση δεδομένων peers.db

Τέλος η μέθοδος onDestroy() καταστρέφει το MainActivity και τερματίζει την υπηρεσία NearbyService και στη συνέχει την εφαρμογή. Ακολουθεί ο κώδικας της κλάσης MainActivity

```java
package com.drid.group_reasoning.ui.activities;

import android.Manifest;
import android.app.AlertDialog;
import android.app.NotificationChannel;
import android.app.NotificationManager;
import android.content.BroadcastReceiver;
import android.content.ComponentName;
import android.content.ContentValues;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.IntentFilter;
import android.content.ServiceConnection;
import android.content.pm.PackageManager;
import android.os.Build;
import android.os.Bundle;
import android.os.IBinder;
import android.support.annotation.NonNull;
import android.support.design.internal.BottomNavigationItemView;
import android.support.design.internal.BottomNavigationMenuView;
import android.support.design.widget.BottomNavigationView;
import android.support.v4.app.ActivityCompat;
import android.support.v4.app.Fragment;
import android.support.v4.app.FragmentManager;
import android.support.v4.app.NotificationCompat;
import android.support.v4.app.NotificationManagerCompat;
import android.support.v4.content.ContextCompat;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.util.Log;
import android.view.ContextThemeWrapper;
import android.view.LayoutInflater;
import android.view.MenuItem;
import android.view.View;
import android.widget.TextView;
import android.widget.Toast;

import com.drid.group_reasoning.R;
import com.drid.group_reasoning.data.contracts.PeerContract;
import com.drid.group_reasoning.ui.fragments.KnowledgeFragment;
import com.drid.group_reasoning.ui.fragments.LogicSolverFragment;
import com.drid.group_reasoning.ui.fragments.PeerListFragment;
import com.drid.group_reasoning.engine.parser.ast.AtomicSentence;
import com.drid.group_reasoning.network.NearbyService;
import com.drid.group_reasoning.network.model.Peer;

import java.util.List;

public class MainActivity extends AppCompatActivity implements
        PeerListFragment.OnFragmentInteractionListener,
        LogicSolverFragment.OnFragmentInteractionListener {

    private static final String TAG = "MainActivity";


    private static final String[] REQUIRED_PERMISSIONS =
            new String[]{
                    Manifest.permission.BLUETOOTH,
                    Manifest.permission.BLUETOOTH_ADMIN,
                    Manifest.permission.ACCESS_WIFI_STATE,
                    Manifest.permission.CHANGE_WIFI_STATE,
                    Manifest.permission.ACCESS_COARSE_LOCATION,
                    Manifest.permission.ACCESS_FINE_LOCATION
            };
```

```
66
67      private static final int REQUEST_CODE_REQUIRED_PERMISSIONS = 1;
68      private static final String CHANNEL_ID = "group_reasoning_notification_channel";
69
70      public NearbyService service;
71
72      private ServiceConnection serviceConnection = new ServiceConnection() {
73          @Override
74          public void onServiceConnected(ComponentName name, IBinder iBinder) {
75              NearbyService.NearbyBinder nearbyBinder = (NearbyService.NearbyBinder) iBinder;
76              service = nearbyBinder.getService();
77              Log.i(TAG,  msg: "onServiceConnected: Service connected");
78          }
79
80          @Override
81          public void onServiceDisconnected(ComponentName name) {
82              service = null;
83              Log.i(TAG,  msg: "onServiceConnected: Service disconnected");
84          }
85      };
86
87      private BroadcastReceiver receiver = (context, intent) -> {
90              String action = intent.getAction();
91              assert action != null;
92              if (action.equals("incoming_connection")) {
93                  Peer peer = intent.getExtras().getParcelable( key: "incoming_peer");
94                  assert peer != null;
95                  displayConnectToPeerDialog(peer);
96              }else if (action.equals("new_query")){
97                  if(!(active instanceof LogicSolverFragment)){
98                      addNotificationBadge();
99                  }
100                 createNotification(intent.getStringExtra( name: "notification_message"));
101             }
102
103     };
105
106     private int notificationCounter = 0;
107
108     FragmentManager fragmentManager = getSupportFragmentManager();
109     Fragment peerListFragment = new PeerListFragment();
110     Fragment knowledgeFragment = new KnowledgeFragment();
111     Fragment logicSolverFragment = new LogicSolverFragment();
112     Fragment active = peerListFragment;
113
114     private BottomNavigationView bottomNavigationView;
115
116     private BottomNavigationView.OnNavigationItemSelectedListener bottomNavigationListener =
117             (menuItem) -> {
120                 switch (menuItem.getItemId()) {
121                     case R.id.nav_nearby_devices:
122                         fragmentManager.beginTransaction().hide(active)
123                                 .show(peerListFragment)
124                                 .commit();
125                         active = peerListFragment;
126                         break;
```

4.5 Το PeerListFragment

Το PeerListFragment περιλαμβάνει ένα κουμπί τύπου Floating Action Button το οποίο όταν πατηθεί ξεκινά η διαδικασία αναζήτησης συσκευών. Όταν η συσκευή εντοπίζει μια άλλη τότε αυτή προστίθεται σε μια λίστα με το όνομα της και την ένδειξη Available. Για να γίνει σύνδεση με αυτή τη συσκευή, θα πρέπει ο χρήστης να πιέσει το αντικείμενο της λίστας που περιέχει το όνομα της και να αναμένει έως ότου η συσκευή να δεχτεί το αίτημα σύνδεσης. Όταν πραγματοποιηθεί η σύνδεση τότε η ένδειξη Available αντικαθίσταται με την ένδειξη Connected. Αντίστοιχα όταν ο χτίστης θέλει να αποσυνδεθεί από μια συσκευή τότε απλά πιέζει το αντίστοιχο αντικείμενο από την λίστα συσκευών, ενώ έχει τη δυνατότητα να αποσυνδεθεί από όλες τις συσκευές, χρησιμοποιώντας το μενού που βρίσκεται στη γραμμή εργαλείων (toolbar). Προτού γίνει η αποσύνδεση εμφανίζεται ένα προειδοποιητικό παράθυρο (alert dialog) στην οθόνη από το οποίο ο χρήστης μπορεί να ολοκληρώσει την διαδικασία ή να την διακόψει.

4.5.1 Το fragment_peer_list.xml

Το αρχείο fragment_peer_list.xml περιγράφει την διεπαφή του PeerListFragment η οποία περιλαμβάνει:

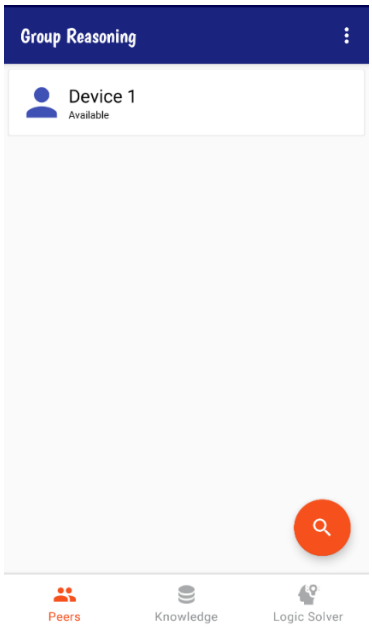
- ένα root view τύπου RelativeLayout
- ένα RecyclerView το οποίο είναι η λίστα με τις συσκευές
- ένα κουμπί τύπου Floating Action Button

Το recycler view με id peers_list είναι μια λίστα με συσκευές που είτε είναι διαθέσιμες για σύνδεση είτε είναι συνδεμένες με τη συσκευή μας. Η διεπαφή ενός αντικειμένου λίστας (list item) φαίνεται στον παρακάτω κώδικα

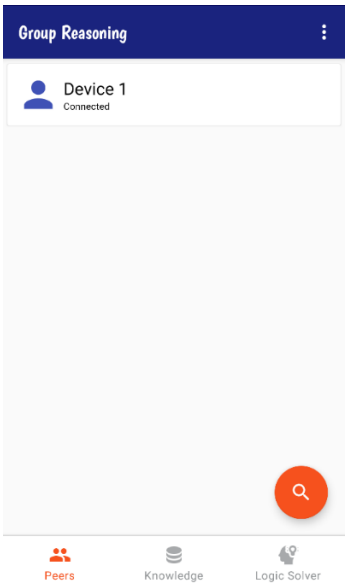
```
<?xml version="1.0" encoding="utf-8"?>
<android.support.v7.widget.CardView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:card_view="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="?android:attr/selectableItemBackground"
    android:clickable="true"
    android:focusable="true"
    android:foreground="?android:attr/selectableItemBackground"
    android:orientation="vertical"
    card_view:cardElevation="1dp"
    card_view:cardUseCompatPadding="true">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:padding="10dp">

        <ImageView
            android:layout_width="45dp"
            android:layout_height="45dp"
            android:layout_gravity="center"
            android:layout_weight="0"
            android:src="@drawable/ic_peer_black_24dp"
            android:tint="@color/design_default_color_primary" />
```

```
27
28          <LinearLayout
29              android:layout_width="0dp"
30              android:layout_height="wrap_content"
31              android:layout_marginLeft="4dp"
32              android:layout_weight="1"
33              android:layout_gravity="center"
34              android:orientation="vertical">
35
36              <TextView
37                  android:id="@+id/device_name"
38                  android:layout_width="match_parent"
39                  android:layout_height="wrap_content"
40                  android:layout_gravity="center"
41                  android:text="Device name"
42                  android:textColor="@android:color/black"
43                  android:textSize="18sp" />
44
45              <TextView
46                  android:id="@+id/status"
47                  android:layout_width="match_parent"
48                  android:layout_height="wrap_content"
49                  android:layout_gravity="center"
50                  android:text="Status"
51                  android:textColor="@android:color/black"
52                  android:textSize="10sp" />
53          </LinearLayout>
54      </LinearLayout>
55  </android.support.v7.widget.CardView>
```

Η τελική μορφή της διεπαφής του Peer list fragment όταν μια συσκευή είναι διαθέσιμη για σύνδεση αλλά και όταν είναι συνδεμένη με τη συσκευή μας φαίνονται στις παρακάτω εικόνες:



Εικόνα 26: Η διεπαφή του Peer list fragment όταν μια συσκευή είναι διαθέσιμη για σύνδεση



Εικόνα 27: Η διεπαφή του Peer list fragment όταν μια συσκευή είναι συνδεμένη

4.5.2 Το αρχείο PeerListFragment.java

Το αρχείο PeerListFragment.java περιγράφει την λογική της κλάσης PeerListFragment και ορίζει μια διεπαφή, την OnFragmentInteractionListener, οι μέθοδοι της οποίας υλοποιούνται από το MainActivity. Η κλάση PeerListFragment επεκτείνει την κλάση Fragment και υλοποιεί τις μεθόδους της διεπαφής LoaderManager.LoaderCallbacks<Cursor>, δηλαδή λαμβάνει δεδομένα από μια βάση δεδομένων, συγκεκριμένα τη peers.db, και εμφανίζει τα δεδομένα αυτά στην λίστα συσκευών. Η μέθοδοι που υλοποιεί είναι :

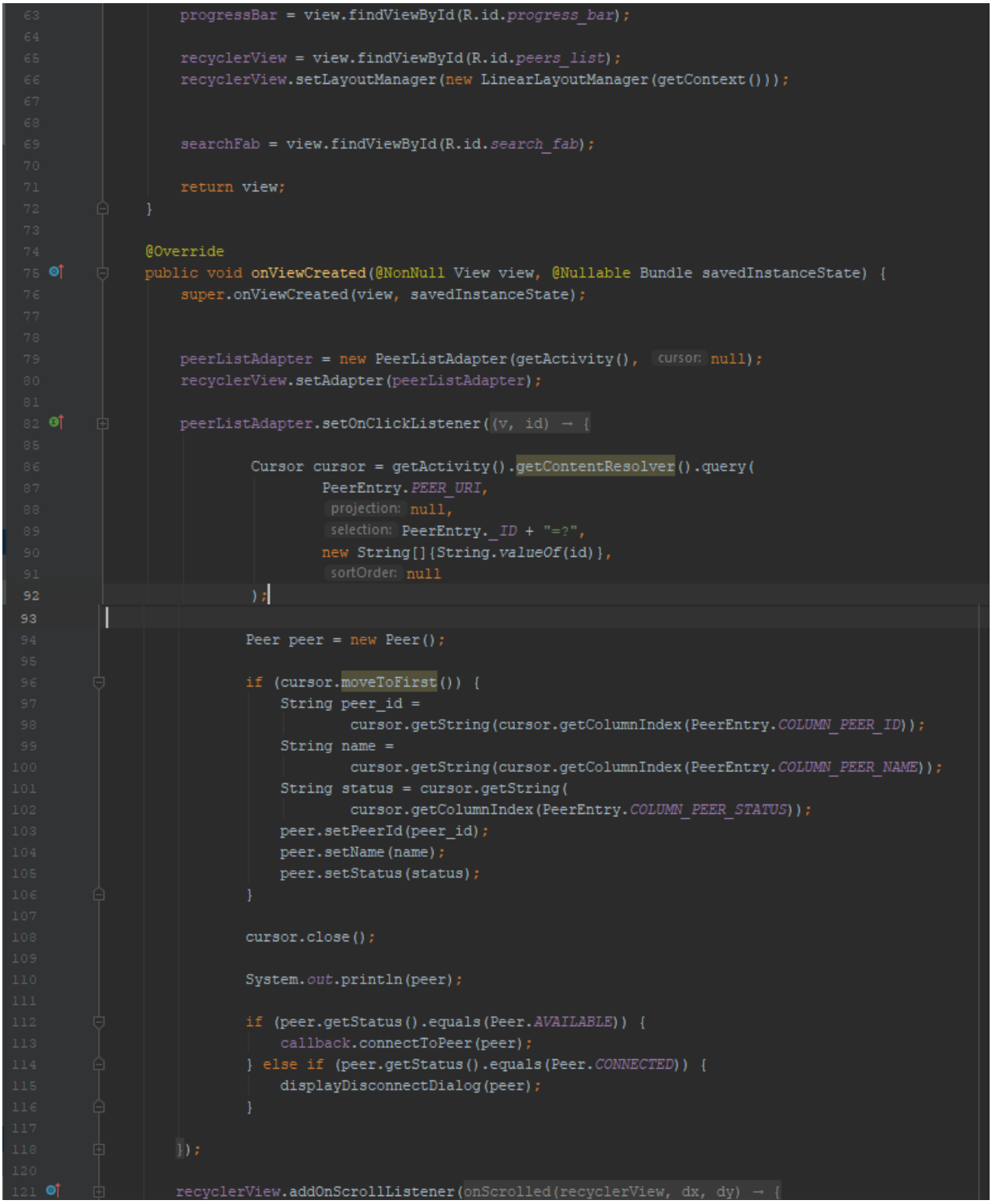
- η μέθοδος onCreateLoader(): όπου δημιουργείται ένα αντικείμενο της κλάσης CursorLoader με ορίσματα τα πεδία (columns) του πίνακα peers, όπως id, peer_id,peer_name και peer_status
- η μέθοδος onLoaderFinished() η οποία καλείται όταν ο CursorLoader έχει ανακτήσει όλα τα δεδομένα από τη βάση. Η μέθοδος καλεί την μέθοδο του swapCursor η οποία η οποία αντικαθιστά τον τρέχοντα Cursor με έναν καινούργιο.
- η μέθοδος onLoaderReset() η οποία μέθοδος καλεί την μέθοδο του swapCursor με όρισμα null.

Η διεπαφή OnFragmentInteractionListener ορίζει τις μεθόδους discoverPeers, connectToPeer, disconnectFromAllPeers και disconnectFromPeer οι οποίες υλοποιούνται στην MainActivity. Όταν μια από αυτές τις μεθόδους καλείται από το PeerListFrament, τότε το MainActivity καλεί την αντίστοιχη μέθοδο της υπηρεσίας NearbyService.

Όσον αφορά τις μεθόδους του PeerListFragment, η μέθοδος onCreate() δημιουργεί την λίστα με τις συσκευές και το κουμπί αναζήτησης συσκευών. Στη μέθοδο onViewCreated() ορίζεται ο προσαρμογέας (adapter) της λίστας των συσκευών ο οποίος είναι αντικείμενο της κλάσης PeerAdapter και ύστερα ορίζεται ένας listener για το πάτημα ενός αντικείμενου της λίστα. Εάν η συσκευή είναι διαθέσιμη τότε καλείται η μέθοδος connectToPeer η οποία στέλνει αίτημα σύνδεσης στην συσκευή με την οποία θέλει να συνδεθεί ο χρήστης. Αντίθετα αν η συσκευή είναι συνδεμένη, τότε καλείται η μέθοδος displayDisconnectDialog, όπου δημιουργείται και εμφανίζεται στην οθόνη ένα προειδοποιητικό παράθυρο ρωτώντας τον χρήστη αν επιθυμεί να αποσυνδεθεί ή όχι. Εάν ο χρήστης πιέσει την επιλογή Yes τότε καλείται η μέθοδος disconnectFromPeer ή οποία αποσυνδέει την συσκευή και ακολουθεί η

μέθοδος deletePeer η οποία διαγράφει τη συσκευή από τη βάση δεδομένων . Τέλος στη μέθοδο onResume ενεργοποιείται ο CursorLoader. Ακολουθεί ο κώδικας του PeerListFragment σε γλώσσα java

```java
package com.drid.group_reasoning.ui.fragments;

import android.app.AlertDialog;
import android.content.DialogInterface;
import android.database.Cursor;
import android.os.Bundle;
import android.support.annotation.NonNull;
import android.support.annotation.Nullable;
import android.support.design.widget.FloatingActionButton;
import android.support.v4.app.Fragment;
import android.support.v4.app.LoaderManager;
import android.support.v4.content.CursorLoader;
import android.support.v4.content.Loader;
import android.support.v7.widget.LinearLayoutManager;
import android.support.v7.widget.RecyclerView;
import android.view.ContextThemeWrapper;
import android.view.LayoutInflater;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.view.ViewGroup;
import android.widget.LinearLayout;
import android.widget.ProgressBar;

import com.drid.group_reasoning.R;
import com.drid.group_reasoning.ui.adapters.PeerListAdapter;
import com.drid.group_reasoning.data.contracts.PeerContract.PeerEntry;
import com.drid.group_reasoning.network.model.Peer;


public class PeerListFragment extends Fragment implements LoaderManager.LoaderCallbacks<Cursor> {

    public static final String TAG = PeerListFragment.class.getSimpleName();

    private OnFragmentInteractionListener callback;

    private static final int PEER_LOADER = 0;

    private PeerListAdapter peerListAdapter;

    private FloatingActionButton searchFab;
    private RecyclerView recyclerView;

    private LinearLayout emptyList;
    private ProgressBar progressBar;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setHasOptionsMenu(true);
    }


    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {

        View view = inflater.inflate(R.layout.fragment_peer_list, container,  attachToRoot: false);

        emptyList = view.findViewById(R.id.empty_peer_list);

```



```
        progressBar = view.findViewById(R.id.progress_bar);

        recyclerView = view.findViewById(R.id.peers_list);
        recyclerView.setLayoutManager(new LinearLayoutManager(getContext()));


        searchFab = view.findViewById(R.id.search_fab);

        return view;
    }

    @Override
    public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {
        super.onViewCreated(view, savedInstanceState);


        peerListAdapter = new PeerListAdapter(getActivity(),  cursor: null);
        recyclerView.setAdapter(peerListAdapter);

        peerListAdapter.setOnClickListener((v, id) -> {

                Cursor cursor = getActivity().getContentResolver().query(
                        PeerEntry.PEER_URI,
                         projection: null,
                         selection: PeerEntry._ID + "=?",
                        new String[]{String.valueOf(id)},
                         sortOrder: null
                );

                Peer peer = new Peer();

                if (cursor.moveToFirst()) {
                    String peer_id =
                            cursor.getString(cursor.getColumnIndex(PeerEntry.COLUMN_PEER_ID));
                    String name =
                            cursor.getString(cursor.getColumnIndex(PeerEntry.COLUMN_PEER_NAME));
                    String status = cursor.getString(
                            cursor.getColumnIndex(PeerEntry.COLUMN_PEER_STATUS));
                    peer.setPeerId(peer_id);
                    peer.setName(name);
                    peer.setStatus(status);
                }

                cursor.close();

                System.out.println(peer);

                if (peer.getStatus().equals(Peer.AVAILABLE)) {
                    callback.connectToPeer(peer);
                } else if (peer.getStatus().equals(Peer.CONNECTED)) {
                    displayDisconnectDialog(peer);
                }

        });

        recyclerView.addOnScrollListener(onScrolled(recyclerView, dx, dy) -> {
```

```
122                 @Override
123                 public void onScrolled(@NonNull RecyclerView recyclerView, int dx, int dy) {
124                     if (dy > 0) {
125                         searchFab.hide();
126                     } else {
127                         searchFab.show();
128                     }
129                     super.onScrolled(recyclerView, dx, dy);
130
131                 }
132             });
133
134
135             searchFab.setOnClickListener((v) -> {
138                     callback.discoverPeers();
139                     emptyList.setVisibility(View.GONE);
140                     progressBar.setVisibility(View.VISIBLE);
141             });
143         }
144
145         @Override
146         public void onActivityCreated(@Nullable Bundle savedInstanceState) {
147             super.onActivityCreated(savedInstanceState);
148
149         }
150
151         @Override
152         public void onPause() { super.onPause(); }
155
156         @Override
157         public void onResume() {
158             super.onResume();
159             getLoaderManager().initLoader(PEER_LOADER,  bundle: null,  loaderCallbacks: this);
160         }
161
162         @Override
163         public void onCreateOptionsMenu(Menu menu, MenuInflater inflater) {
164             inflater.inflate(R.menu.peer_options_menu, menu);
165             return;
166         }
167
168         @Override
169         public boolean onOptionsItemSelected(MenuItem item) {
170             switch (item.getItemId()) {
171                 case R.id.delete_all_peers:
172                     displayDisconnectFromAllPeersDialog();
173                     return true;
174             }
175
176             return super.onOptionsItemSelected(item);
177         }
178
```

```
    @NonNull
    @Override
    public Loader<Cursor> onCreateLoader(int i, @Nullable Bundle bundle) {
        String[] projection = {
                PeerEntry._ID,
                PeerEntry.COLUMN_PEER_ID,
                PeerEntry.COLUMN_PEER_NAME,
                PeerEntry.COLUMN_PEER_STATUS};

        return new CursorLoader(
                getContext(),
                PeerEntry.PEER_URI,
                projection,
                 selection: null,
                 selectionArgs: null,
                 sortOrder: null);
    }

    @Override
    public void onLoadFinished(@NonNull Loader<Cursor> loader, Cursor cursor) {
        peerListAdapter.swapCursor(cursor);

        if(cursor.getCount() == 0){
            emptyList.setVisibility(View.VISIBLE);
        }else{
            progressBar.setVisibility(View.GONE);
            emptyList.setVisibility(View.GONE);
        }
    }

    @Override
    public void onLoaderReset(@NonNull Loader<Cursor> loader) { peerListAdapter.swapCursor( newCursor: null); }

    private void deletePeer(String peer_id) {
        getActivity().getContentResolver().delete(
                PeerEntry.PEER_URI,
                 where: PeerEntry.COLUMN_PEER_ID + "=?",
                new String[]{peer_id});

    }

    private void deleteAllPeers() {
        getActivity().getContentResolver()
                .delete(PeerEntry.PEER_URI,  where: null,  selectionArgs: null);

    }

    private void disconnectFromAllPeers() { callback.disconnectFromAllPeers(); }

    private void disconnectFromPeer(Peer peer) { callback.disconnectFromPeer(peer); }

    private void displayDisconnectDialog(final Peer peer) {
        AlertDialog dialog = new AlertDialog.Builder(
                new ContextThemeWrapper(getContext(), R.style.AlertDialog))
                .setTitle("Are you sure you want to disconnect from " + peer.getName() + "?")
                .setPositiveButton( text: "Yes", (dialog, which) → {
                        deletePeer(peer.getPeerId());
```

```
                        deletePeer(peer.getPeerId());
                        disconnectFromPeer(peer);
                    }).setNegativeButton( text: "No", (dialog, which) → {
                        dialog.dismiss();
                    }).create();
            dialog.show();
            dialog.setCancelable(false);
            dialog.setCanceledOnTouchOutside(false);
        }

        private void displayDisconnectFromAllPeersDialog() {
            AlertDialog dialog = new AlertDialog.Builder(
                    new ContextThemeWrapper(getContext(), R.style.AlertDialog))
                    .setTitle("Are you sure you want to disconnect from all peers ? ")
                    .setPositiveButton( text: "Yes", (dialog, which) → {
                        deleteAllPeers();
                        disconnectFromAllPeers();
                    }).setNegativeButton( text: "No", (dialog, which) → {
                        dialog.dismiss();
                    }).create();
            dialog.show();
            dialog.setCancelable(false);
            dialog.setCanceledOnTouchOutside(false);
        }


        public void setOnFragmentCreatedListener(OnFragmentInteractionListener callback) {
            this.callback = callback;
        }
        public interface OnFragmentInteractionListener {
            void discoverPeers();
            void connectToPeer(Peer peer);
            void disconnectFromAllPeers();
            void disconnectFromPeer(Peer selectedPeer);
        }
    }

```

4.5.3 Το αρχείο PeerDbHelper.java

Η κλάση PeerDbHelper επεκτείνει την κλάση SQLiteOpenHelper και δημιουργεί την βάση δεδομένων peers. Κληρονομεί και υπερβαίνει την μέθοδο onCreate στην οποία ορίζεται μια συμβολοσειρά η οποία περιγράφει μια εντολή σε γλώσσα Sql για δημιουργία του πίνακα peers. Ο πίνακας peers αποτελείται από τα εξής πεδία. Ακολουθούν παρακάτω ένα στιγμιότυπο του πίνακα peers και ο κώδικας του αρχείου PeerDbHelper.java

Id	peer_id	peer_name	Peer_status
1	yP28	Alex	Connected

Πίνακας 6: Στιγμιότυπο του πίνακα peers

```java
package com.drid.group_reasoning.data.database_helpers;

import ...

public class PeerDbHelper extends SQLiteOpenHelper {

    public static final String TAG = PeerDbHelper.class.getSimpleName();
    public static final String DATABASE = "peers.db";
    public static final int DATABASE_VERSION = 1;

    public PeerDbHelper(Context context) { super(context, DATABASE, factory: null, DATABASE_VERSION); }

    @Override
    public void onCreate(SQLiteDatabase db) {
        // Create a String that contains the SQL statement to create the peers table
        String SQL_CREATE_PEERS_TABLE =  "CREATE TABLE " + PeerEntry.TABLE_NAME + " ("
                + PeerEntry._ID + " INTEGER PRIMARY KEY AUTOINCREMENT, "
                + PeerEntry.COLUMN_PEER_ID + " TEXT NOT NULL, "
                + PeerEntry.COLUMN_PEER_NAME + " TEXT NOT NULL, "
                + PeerEntry.COLUMN_PEER_STATUS + " TEXT NOT NULL);";

        db.execSQL(SQL_CREATE_PEERS_TABLE);
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {

    }
}
```

4.5.4 Το αρχείο PeerListAdapter.java

Στο αρχείο PeerListAdapter.java ορίζεται ένας προσαρμογέας ο οποίος ορίζει τον τρόπο με τον οποία εμφανίζονται τα δεδομένα στο RecyclerView της διεπαφής του PeerListFragment. Τα δεδομένα τα αντλεί από τη βάση δεδομένων peers.db.

```java
package com.drid.group_reasoning.ui.adapters;

import ...

public class PeerListAdapter extends RecyclerView.Adapter<PeerListAdapter.PeerViewHolder> {

    private Context context;
    private Cursor cursor;


    private ClickListener listener;

    public PeerListAdapter(Context context, Cursor cursor) {
        this.context = context;
        this.cursor = cursor;
    }

    @NonNull
    @Override
    public PeerViewHolder onCreateViewHolder(@NonNull ViewGroup viewGroup, int i) {
        LayoutInflater inflater = LayoutInflater.from(context);
        View view = inflater.inflate(R.layout.list_item_peer, viewGroup, attachToRoot: false);
        return new PeerViewHolder(view);
    }

    @Override
    public void onBindViewHolder(@NonNull PeerViewHolder peerViewHolder, int i) {
        if (!cursor.moveToPosition(i)) {
            return;
        }

        int id = cursor.getInt(cursor.getColumnIndex(PeerEntry._ID));
        String name = cursor.getString(cursor.getColumnIndex(PeerEntry.COLUMN_PEER_NAME));
```

4.6 Το KnowledgeFragment

Το KnowledgeFragment διαθέτει ένα View Pager οποίος διαχειρίζεται τα fragments RuleFragment και FactFragment. Επίσης διαθέτει ένα κουμπί του οποίου η λειτουργία αλλάζει δυναμικά ανάλογα με ποιο fragment είναι ορατό στην οθόνη. Όταν είναι ορατό το RuleFragment τότε αυτό με το πάτημα του κουμπιού ένα νέο Fragment στο οποίο ο χρήστης προσθέτει ένα καινούργιο κανόνα. Αντιστοίχως όταν είναι ορατό FactFragment, τότε ο χρήστης προσθέτει ένα νέο γεγονός.

4.6.1 Το αρχείο fragment_knowledge.xml

Η διεπαφή του KnowledgeFragment αποτελείται από τα εξής views:

- ένα root view τύπου RelativeLayout
- ένα TabLayout το οποίο έχει 2 καρτέλες (tabs) με τίτλους RULES και FACTS αντίστοιχα
- έναν ViewPager στον οποίο εμφανίζονται τα fragments RuleFragment και FactFragment
- ένα κουμπί τύπου Floating Action Button

```xml
<?xml version="1.0" encoding="utf-8"?>

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".ui.fragments.KnowledgeFragment">

    <android.support.design.widget.TabLayout
        android:id="@+id/sliding_tabs"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="@color/colorPrimary"
        app:tabGravity="fill"
        app:tabIndicatorHeight="3dp"
        app:tabMaxWidth="0dp"
        app:tabMode="fixed"
        app:tabSelectedTextColor="@color/colorPrimaryLight"
        app:tabTextColor="@color/colorTextPrimary">

    </android.support.design.widget.TabLayout>


    <android.support.v4.view.ViewPager
        android:id="@+id/viewpager"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_marginTop="?android:attr/actionBarSize"
        app:layout_anchor="@id/sliding_tabs" />
```

```
    <android.support.design.widget.FloatingActionButton
        android:id="@+id/add_fab"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="16dp"
        android:src="@drawable/ic_add_white_24dp"
        android:layout_alignParentBottom="true"
        android:layout_alignParentRight="true"
        android:layout_alignParentEnd="true"/>

</RelativeLayout>
```

4.6.2 Το αρχείο list_item_knowledge.xml

Το αρχείο list_item_knowledge.xml περιέχει τον κώδικα που περιγράφει τον σχεδιασμό ενός αντικειμένου της λίστας των κανόνων και της λίστας των γεγονότων.

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.v7.widget.CardView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:card_view="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:clickable="true"
    android:focusable="true"
    android:foreground="?android:attr/selectableItemBackground"
    card_view:cardElevation="2dp"
    card_view:cardUseCompatPadding="true">


    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal">



        <LinearLayout
            android:layout_width="0dp"
            android:layout_weight="2"
            android:layout_height="wrap_content"
            android:orientation="vertical"
            android:padding="12dp">


            <TextView
                android:id="@+id/data_nl"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
```

```
            <TextView
                android:id="@+id/data_pl"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content" />
        </LinearLayout>

        <ImageView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center_vertical"
            android:layout_marginRight="12dp"
            android:alpha="0.8"
            android:scaleType="center"
            android:src="@drawable/ic_edit_white_24dp"
            android:tint="@color/colorPrimaryLight" />
    </LinearLayout>

</android.support.v7.widget.CardView>
```

Ένα αντικείμενο της λίστας κανόνων και γεγονότων, διαθέτει τα εξής:

- Ένα root view που ανήκει στην κατηγορία CardView
- Ένα view group τύπου LinearLayout το οποίο περιλαμβάνει ένα view group της κλάσης LinearLayout και ένα ImageView
- Το εμφωλευμένο view group περιέχει ένα TextView στο οποίο αναγράφεται ο κανόνας σε φυσική γλώσσα (αγγλικά) και ένα TextView όπου ο κανόνας είναι εκφρασμένος σε προτασιακή λογική.
- Το ImageView απεικονίζει ένα μολύβι το οποίο κατά τις αρχές του Material Design συμβολίζει τροποποίηση μιας πληροφορίας

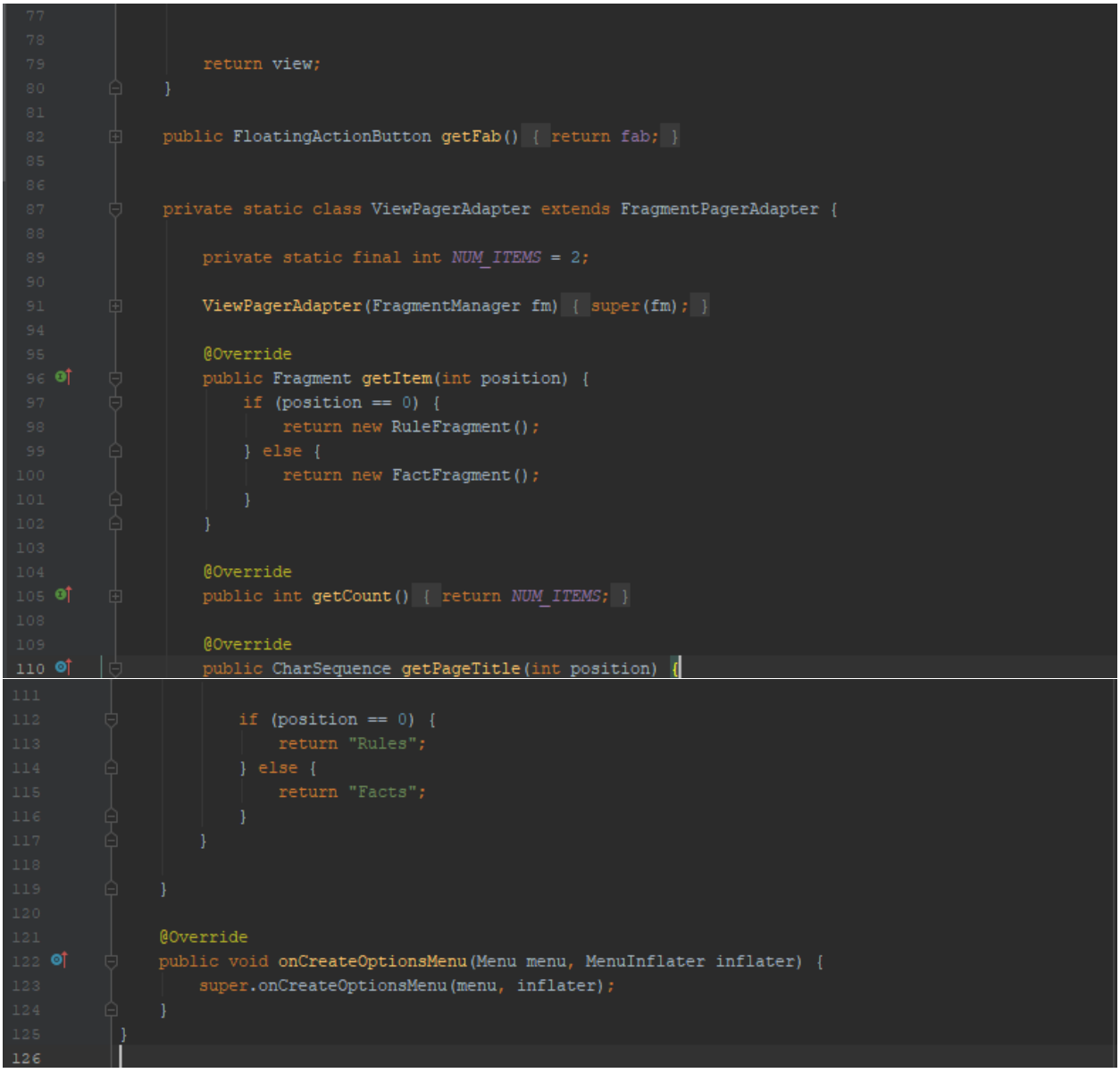
4.6.3 Το αρχείο KnowledgeFragment.java

Η κλάση KnowledgeFragment επεκτείνει την κλάση Fragment. Διαθέτει μια εμφωλιασμένη στατική κλάση την ViewPagerAdapter η οποία επεκτείνει την κλάση FragmentPagerAdapter και διαχειρίζεται ποιο fragment είναι ορατό στην οθόνη αλλά και τους τίτλους των καρτελών του TabLayout.

Στην μέθοδο onCreateView() δημιουργούνται τα components της διεπαφής του KnowledgeFragment όπως το TabLayout και το κουμπί προσθήκης. Επίσης ορίζουμε ως adapter του TabLayout ένα αντικείμενο της κλάσης ViewPagerAdapter και καταχωρούμε σε αυτόν τον adapter έναν listener ο οποίος εξαφανίζει για λίγα δευτερόλεπτα το κουμπί όταν ο χρήστης σύρει το δάκτυλο του (δηλαδή όταν κάνει swipe) προς στα δεξιά για να προηγηθεί μεταξύ της καρτέλας RULES και FACTS και αντίστροφα.

Η μέθοδος getFab() επιστέφει το αντικείμενο της κλάσης FloatingActionButton. Χρησιμοποιείται από το RuleFragment και FactFragement για τη προσθήκη νέων κανόνων και γεγονότων αντίστοιχα. Ο κώδικας που ακολουθεί υλοποιεί τη λογική του KnowledgeFragment

```java
package com.drid.group_reasoning.ui.fragments;

import ...

public class KnowledgeFragment extends Fragment {

    public static final String TAG = KnowledgeFragment.class.getSimpleName();

    private ViewPager viewPager;
    private ViewPagerAdapter viewPagerAdapter;

    private FloatingActionButton fab;

    @Override
    public void onCreate(Bundle savedInstanceState) { super.onCreate(savedInstanceState); }

    @Override
    public void onResume() {  super.onResume();  }

    @Override
    public View onCreateView(
            LayoutInflater inflater,
            ViewGroup container,
            Bundle savedInstanceState) {

        View view = inflater.inflate(R.layout.fragment_knowledge, container,  attachToRoot: false);

        fab = view.findViewById(R.id.add_fab);
        viewPager = view.findViewById(R.id.viewpager);
        viewPagerAdapter = new ViewPagerAdapter(getChildFragmentManager());
        viewPager.setAdapter(viewPagerAdapter);

        viewPager.addOnPageChangeListener(new ViewPager.OnPageChangeListener() {
            @Override
            public void onPageScrolled(int i, float v, int il) {

            }

            @Override
            public void onPageSelected(int position) {

            }

            @Override
            public void onPageScrollStateChanged(int state) {
                switch (state) {
                    case ViewPager.SCROLL_STATE_IDLE:
                        fab.show();
                        break;
                    case ViewPager.SCROLL_STATE_DRAGGING:
                    case ViewPager.SCROLL_STATE_SETTLING:
                        fab.hide();
                        break;
                }
            }
        });

        TabLayout tabLayout = view.findViewById(R.id.sliding_tabs);

        tabLayout.setupWithViewPager(viewPager);
```



```
        return view;
    }

    public FloatingActionButton getFab() { return fab; }


    private static class ViewPagerAdapter extends FragmentPagerAdapter {

        private static final int NUM_ITEMS = 2;

        ViewPagerAdapter(FragmentManager fm) { super(fm); }

        @Override
        public Fragment getItem(int position) {
            if (position == 0) {
                return new RuleFragment();
            } else {
                return new FactFragment();
            }
        }

        @Override
        public int getCount() { return NUM_ITEMS; }

        @Override
        public CharSequence getPageTitle(int position) {

            if (position == 0) {
                return "Rules";
            } else {
                return "Facts";
            }
        }

    }

    @Override
    public void onCreateOptionsMenu(Menu menu, MenuInflater inflater) {
        super.onCreateOptionsMenu(menu, inflater);
    }
}

```

4.7 Το RuleFragment

Το RuleFragment αποτελεί το τμήμα του Knowledge fragment που εμφανίζει στην οθόνη μια λίστα με κανόνες οι οποίο είναι εκφρασμένοι σε φυσική γλώσσα και προτασιακή λογική. Όταν το fragment αυτό είναι επιλεγμένο από τον view pager τότε η λειτουργία του κουμπιού προσθήκης είναι να δημιουργεί ένα fragment στο οποίο ο χρήστης εισάγει έναν νέο κανόνα.

4.7.1 Το αρχείο fragment_knowledge.xml

Η διεπαφή του RuleFragment περιλαμβάνει:

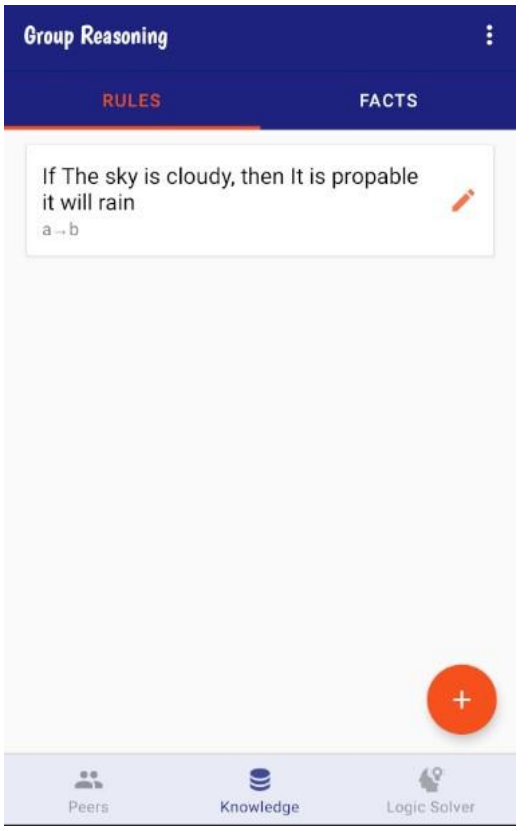
- Ένα root view τύπου RelativeLayout

- Ένα RecyclerView το οποίο είναι η λίστα με τους κανόνες

Ακολουθούν ο κώδικας του αρχείου fragment_rule.xml και η τελική μορφή της διεπαφής του Rule fragment

```xml
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center"
    android:orientation="vertical"
    tools:context=".ui.fragments.RuleFragment">


    <android.support.v7.widget.RecyclerView
        android:id="@+id/rule_list"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_marginLeft="16dp"
        android:layout_marginRight="16dp"
        tools:listitem="@layout/list_item_knowledge" />

    <LinearLayout
        android:id="@+id/empty_rule_list"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:layout_marginTop="120dp"
        android:layout_centerHorizontal="true"
        android:visibility="gone">

        <ImageView
            android:layout_width="128dp"
            android:layout_height="128dp"
            android:layout_gravity="center"
            android:src="@drawable/ic_rules"
            android:tint="@color/colorPrimaryAccent" />

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="No rules available"
            android:textColor="@color/colorPrimary"
            android:textSize="22sp" />

    </LinearLayout>
</RelativeLayout>
```

Ο κώδικας του αρχείου fragment_rule.xml



Εικόνα 28: Η διέπαφη του Rule fragment

4.7.2 Το αρχείο RuleFragment.java

Η κλάση RuleFragment επεκτείνει την κλάση Fragment και υλοποιεί τις μεθόδους της διεπαφής LoaderManager.LoaderCallbacks<Cursor>.

Στην μέθοδο onCreateView δημιουργούνται τα components της διεπαφής του RuleFragment. ενώ στην μέθοδο onViewCreated ορίζεται ο adapter του recycler view, και στη συνέχεια ορίζεται στο recycler view ένας listener ο οποίος εμφανίζει το κουμπί προσθήκης όταν ο χρήστης σύρει το δάκτυλο του προς τα πάνω (scroll up) ή το κρύβει αν το σύρει προς τα κάτω (scroll down).

Η μέθοδος onResume ενεργοποιεί τον loader ο οποίος ανασύρει τα δεδομένα από την βάση δεδομένων και ορίζει τη λειτουργικότητα του κουμπιού προσθήκης. Δηλαδή καταχωρεί έναν listener ο οποίος καλεί την μέθοδο openDialog

Η μέθοδος openDialog δημιουργεί ένα νέο παράθυρο ή dialog fragment της κλάσης EditRuleDialog. Η δημιουργία νέου παραθύρου γίνεται σε ένα ξεχωριστό νήμα εκτέλεσης από το κύριο (UI thread) για το λόγω ότι η δημιουργία νέου παραθύρου απαιτεί επεξεργαστική ισχύ,

οπότε με αυτόν τον τρόπο έχουμε ομαλότερη μετάβαση από το RuleFragment στο EditRuleDialog.
Ακολουθεί ο κώδικας της κλάσης RuleFragment

```
1   package com.drid.group_reasoning.ui.fragments;
2
3   import ...
33
34  public class RuleFragment extends Fragment implements LoaderManager.LoaderCallbacks<Cursor> {
35
36      public static final String TAG = RuleFragment.class.getSimpleName();
37
38      public static final String ARG_RULE_URI = "rule_uri";
39
40      private static final int RULE_LOADER = 0;
41
42      private LinearLayout emptyList;
43      private RecyclerView recyclerView;
44      private FloatingActionButton fab;
45
46      private RuleAdapter ruleAdapter;
47
48      private EditRuleDialog dialog;
49
50      @Override
51      public void onCreate(Bundle savedInstanceState) {
52          super.onCreate(savedInstanceState);
53          setHasOptionsMenu(true);
54      }
55
56      @Override
57      public View onCreateView(LayoutInflater inflater, ViewGroup container,
58                               Bundle savedInstanceState) {
59
60          View view = inflater.inflate(R.layout.fragment_rule, container,  attachToRoot: false);
61
62          emptyList = view.findViewById(R.id.empty_rule_list);
```

```
63
64          recyclerView = view.findViewById(R.id.rule_list);
65          recyclerView.setLayoutManager(new LinearLayoutManager(getContext()));
66
67
68          return view;
69      }
70
71      @Override
72      public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {
73          super.onViewCreated(view, savedInstanceState);
74
75
76          ruleAdapter = new RuleAdapter(getContext(),  cursor: null);
77          recyclerView.setAdapter(ruleAdapter);
78
79          ruleAdapter.setOnItemClickListener((v, id) -> {
82
83                  Uri selectedUri = ContentUris.withAppendedId(RuleEntry.RULES_URI, id);
84
85                  Bundle arguments = new Bundle();
86                  arguments.putString(ARG_RULE_URI, String.valueOf(selectedUri));
87
88                  openDialog(arguments);
89          });
91
92          recyclerView.addOnScrollListener(onScrolled(recyclerView, dx, dy) -> {
95                  if (dy > 0) {
96                      fab.hide();
97                  } else {
98                      fab.show();
99                  }
100                 super.onScrolled(recyclerView, dx, dy);
101
102         });
104     }
105
106     @Override
107     public void setUserVisibleHint(boolean isVisible) {
108         super.setUserVisibleHint(isVisible);
109
110         if (isVisible && isResumed()) {
111             onResume();
112         }
113     }
114
115     @Override
116     public void onResume() {
117         super.onResume();
118
119         if (!getUserVisibleHint()) {return;}
120
121         getLoaderManager().initLoader(RULE_LOADER,  bundle: null,  loaderCallbacks: this);
122
123         KnowledgeFragment knowledgeFragment = (KnowledgeFragment) getParentFragment();
```

```
124
125             fab = knowledgeFragment.getFab();
126             fab.setOnClickListener((v) → { openDialog( arguments: null); });
132         }
133
134         @Override
135 @       public void onCreateOptionsMenu(Menu menu, MenuInflater inflater) {
136             menu.clear();
137             inflater.inflate(R.menu.rule_options_menu, menu);
138             super.onCreateOptionsMenu(menu, inflater);
139         }
140
141         @Override
142 @       public boolean onOptionsItemSelected(MenuItem item) {
143             switch (item.getItemId()) {
144                 case R.id.delete_all_rules:
145                     displayDeleteAllRulesDialog();
146                     return true;
147             }
148
149             return super.onOptionsItemSelected(item);
150         }
151
152         private void displayDeleteAllRulesDialog() {
153             AlertDialog dialog = new AlertDialog.Builder(
154                     new ContextThemeWrapper(getContext(), R.style.AlertDialog))
155                     .setTitle("Are you sure you want to delete all rules ?")
156                     .setPositiveButton( text: "Yes", (dialog, which) → {
159                             deleteAllRules();
160                     }).setNegativeButton( text: "No", (dialog, which) → {
164                             dialog.dismiss();
165                     }).create();
167             dialog.show();
168             dialog.setCancelable(false);
169             dialog.setCanceledOnTouchOutside(false);
170         }
171
172         @NonNull
173         @Override
174         public Loader<Cursor> onCreateLoader(int i, @Nullable Bundle bundle) {
175             String[] projection = {
176                     RuleEntry._ID,
177                     RuleEntry.COLUMN_RULE_NL,
178                     RuleEntry.COLUMN_RULE_PL};
179
180             return new CursorLoader(
181                     getContext(),
182                     RuleEntry.RULES_URI,
183                     projection,
184                      selection: null,
185                      selectionArgs: null,
186                      sortOrder: null);
187         }
188
189         @Override
190         public void onLoadFinished(@NonNull Loader<Cursor> loader, Cursor cursor) {
191             ruleAdapter.swapCursor(cursor);
```

```
191             ruleAdapter.swapCursor(cursor);
192             emptyList.setVisibility(cursor.getCount() == 0 ? View.VISIBLE : View.GONE);
193         }
194
195         @Override
196         public void onLoaderReset(@NonNull Loader<Cursor> loader) { ruleAdapter.swapCursor( newCursor: null); }
199
200
201         private void deleteAllRules() {
202             getActivity().getContentResolver()
203                     .delete(RuleEntry.RULES_URI, where: null, selectionArgs: null);
204             getActivity().getContentResolver()
205                     .delete(SymbolEntry.SYMBOLS_URI, where: null, selectionArgs: null);
206             getActivity().getContentResolver()
207                     .delete(RuleSymbolEntry.RULE_SYMBOL_URI,
208                             where: null, selectionArgs: null);
209         }
210
211         private void openDialog(final Bundle arguments) {
212
213             Thread openDialogThread = new Thread((Runnable) () -> {
216                     dialog = new EditRuleDialog();
217                     dialog.show(getChildFragmentManager(), EditRuleDialog.TAG);
218                     dialog.setArguments(arguments);
219             });
221
222             openDialogThread.start();
223         }
224     }
```

4.7.3 Το αρχείο RuleDbHelper

Η κλάση RuleDbHelper επεκτείνει την κλάση SQLiteOpenHelper και δημιουργεί την βάση δεδομένων rules η οποία αποτελείται από τους πίνακες rules, symbols και rules_symbols. Ο πίνακας rules αποτελείται από τα εξής πεδία

id	rule_nl	rule_pl
1	If the sky is cloudy then it is possible it will rain	A → B

Πίνακας 7: Στιγμιότυπο του πίνακα rules

Ο πίνακας symbols αποτελείται από τα εξής πεδία

Id	variable	value
1	A	The sky is cloudy
2	B	It is possible it will rain

Πίνακας 8: Στιγμιότυπο του πίνακα symbols

Ο πίνακας rules_symbols αποτελείται από τα εξής πεδία

Id	rule_id	variable_id
1	1	1
2	1	2

Πίνακας 9: Στιγμιότυπο του πίνακα rules_symbols

Ο πίνακας rules και ο πίνακας symbols έχουν σχέση πολλά προς πολλά (M-το-N) η οποία περιγράφεται στον πίνακα rules_symbols. Τα πεδία rule_id και variable_id είναι ξένα κλειδιά που αναφέρονται στα πεδία id του πίνακα rules και symbols αντίστοιχα. Ακολουθεί ο κώδικας του αρχείου RuleDbHelper.java

```
package com.drid.group_reasoning.data.database_helpers;

import ...

public class RuleDbHelper extends SQLiteOpenHelper {

    public static final String TAG = RuleDbHelper.class.getSimpleName();
    public static final String DATABASE = "rules.db";
    public static final int DATABASE_VERSION = 1;

    public RuleDbHelper(Context context) { super(context, DATABASE,  factory: null, DATABASE_VERSION); }

    @Override
    public void onCreate(SQLiteDatabase db) {
        // A String that contains the SQL statement to create the rules table
        String CREATE_RULES_TABLE =  "CREATE TABLE " + RuleEntry.TABLE_RULES + " ("
                + RuleEntry._ID + " INTEGER PRIMARY KEY AUTOINCREMENT, "
                + RuleEntry.COLUMN_RULE_NL + " TEXT NOT NULL, "
                + RuleEntry.COLUMN_RULE_PL + " TEXT NOT NULL);";

        // A String that contains the SQL statement to create the symbols table
        String CREATE_SYMBOLS_TABLE =  "CREATE TABLE " + SymbolEntry.TABLE_SYMBOLS + " ("
                + SymbolEntry._ID + " INTEGER PRIMARY KEY AUTOINCREMENT, "
                + SymbolEntry.COLUMN_SYMBOL + " TEXT NOT NULL UNIQUE ON CONFLICT REPLACE, "
                + SymbolEntry.COLUMN_PROPOSITION + " TEXT NOT NULL UNIQUE ON CONFLICT REPLACE);";

        // A String that contains the SQL statement to create the symbols table
        String CREATE_RULES_SYMBOLS_TABLE =  "CREATE TABLE " + RuleSymbolEntry.TABLE_RULES_SYMBOLS + " ("
                + RuleSymbolEntry._ID + " INTEGER PRIMARY KEY AUTOINCREMENT, "
                + RuleSymbolEntry.KEY_RULE_ID + " INTEGER , "
                + RuleSymbolEntry.KEY_SYMBOL_ID + " INTEGER," +
                " FOREIGN KEY (" + RuleSymbolEntry.KEY_RULE_ID + ") " +
                " REFERENCES " + RuleEntry.TABLE_RULES +"( "+RuleEntry._ID +")" +
                    " ON UPDATE CASCADE ON DELETE CASCADE," +
                    " FOREIGN KEY (" + RuleSymbolEntry.KEY_SYMBOL_ID + ")"  +
                    " REFERENCES " + SymbolEntry.TABLE_SYMBOLS +"( "+ SymbolEntry._ID +")" +
                    " ON UPDATE CASCADE ON DELETE CASCADE);";



        db.execSQL(CREATE_RULES_TABLE);
        db.execSQL(CREATE_SYMBOLS_TABLE);
        db.execSQL(CREATE_RULES_SYMBOLS_TABLE);
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        db.execSQL("DROP TABLE IF EXISTS " + RuleEntry.TABLE_RULES);
        db.execSQL("DROP TABLE IF EXISTS " + SymbolEntry.TABLE_SYMBOLS);
        db.execSQL("DROP TABLE IF EXISTS " + RuleSymbolEntry.TABLE_RULES_SYMBOLS);
        onCreate(db);
    }
}
```

4.7.4 Το αρχείο RuleAdapter.java

Στο αρχείο RuleAdapter.java ορίζεται ένας προσαρμογέας ο οποίος ορίζει τον τρόπο με τον οποία εμφανίζονται τα δεδομένα στο RecyclerView της διεπαφής του RuleFragment. Τα δεδομένα τα αντλεί από τη βάση δεδομένων rules.db.

```
package com.drid.group_reasoning.ui.adapters;

import ...

public class RuleAdapter extends RecyclerView.Adapter<RuleAdapter.RuleViewHolder> {

    private Context context;
    private Cursor cursor;


    private static ClickListener listener;


    public RuleAdapter(Context context, Cursor cursor) {
        this.context = context;
        this.cursor = cursor;
    }

    @NonNull
    @Override
    public RuleViewHolder onCreateViewHolder(@NonNull ViewGroup viewGroup, int i) {
        View view =
                LayoutInflater
                        .from(context)
                        .inflate(R.layout.list_item_knowledge, viewGroup,  attachToRoot: false);
        return new RuleViewHolder(view);
    }

    @Override
    public void onBindViewHolder(@NonNull RuleViewHolder ruleViewHolder, int i) {
        if (!cursor.moveToPosition(i)) {
            return;
        }

        int id = cursor.getInt(cursor.getColumnIndex(RuleEntry._ID));
        String ruleNl = cursor.getString(cursor.getColumnIndex(RuleEntry.COLUMN_RULE_NL));
        String rulePL = cursor.getString(cursor.getColumnIndex(RuleEntry.COLUMN_RULE_PL));

        ruleViewHolder.id = id;
        ruleViewHolder.ruleNlTextView.setText(ruleNl);
        ruleViewHolder.rulePlTextView.setText(rulePL);

    }


    @Override
    public int getItemCount() { return (cursor == null) ? 0 : cursor.getCount(); }

    public void swapCursor(Cursor newCursor) {
        cursor = newCursor;

        if (newCursor != null) {
            notifyDataSetChanged();
        }

    }

    public void setOnItemClickListener(ClickListener listener) { RuleAdapter.listener = listener; }

    public class RuleViewHolder extends RecyclerView.ViewHolder {

```

4.7.5 To fragment_editor_rule.xml

Ο Rule editor είναι ένα παράθυρο στο οποίο ο χρήστης μπορεί να προσθέσει έναν νέο κανόνα στη βάση δεδομένων rules.db αλλά και να τροποποιήσει έναν κανόνα. Αρχικά ο χρήστης πληκτρολογεί έναν κανόνα εκφρασμένο σε προτασιακή λογική, έπειτα αναθέτει μια πρόταση εκφρασμένη σε φυσική γλώσσα σε καθένα από τα σύμβολα του κανόνα και τέλος μεταφράζει τον κανόνα σε φυσική γλώσσα. Όταν ο χρήστης τροποποιεί τον κανόνα τότε ο δεν μπορεί να αλλάξει την πρόταση που αντιστοιχεί σε ένα σύμβολο μπορεί όμως να διαγράψει το ζεύγος σύμβολο-πρόταση αλλά να προσθέσει ένα καινούργιο.

Το αρχείο fragment_editor_rule.xml περιγράφει τον σχεδιασμό της διεπαφής του παραθύρου Edit rule dialog, η οποία αποτελείται από τα εξής views:

- Το root view το οποίο ανήκει στην κατηγορία RelativeLayout
- Ένα toolbar στο οποίο αναγράφεται ο τίτλος “Add a Rule” όταν πρόκειται για προθήκη νέου κανόνα ή “Edit a Rule” όταν πρόκειται για τροποποίηση ενός υπάρχοντος κανόνα. Επίσης διαθέτει ένα κουμπί με την ένδειξη X το οποίο κλείνει το παράθυρο.
- Ένα view group τύπου RelativeLayout, το οποίο περιλαμβάνει:
    - Ένα group view τύπου SrollView το οποίο επιτρέπει στον χρήστη να κάνει scroll up ή scroll down. Αυτό το view group έχει εμφωλευμένα τα εξής views:
        - Ένα LinearLayout το οποίο περιέχει:
            - Ένα πεδίο εισαγωγής κείμενο (EditText)
            - Ένα LinearLayout το οποίο αρχικά δεν είναι ορατό στον χρήστη, αλλά αποτελεί το σημείο στο οποίο ο χρήστης, όταν πιέσει το κουμπί “Add propositions”, εισάγει ένα σύνολο από views τα οποία είναι ορισμένα στο αρχείο layout_add_ propositions.xml
            - Το κουμπί με την ένδειξη “Add propositions”, το οποίο ανήκει στην κατηγορία των MaterialButton
            - Το κουμπί με την ένδειξη “Translate” το οποίο ανήκει στην κατηγορία των MaterialButton

- Ένα view group της κλάσης LinearLayout το οποίο είναι τοποθετημένο στο κάτω μέρος του view group στο οποίο ανήκει, δηλαδή στο κάτω μέρος της οθόνης. Διαθέτει ένα οριζόντιο recycler view το οποίο έχει κατεύθυνση από τα αριστερά προς τα δεξιά και τα στοιχεία του είναι οι λογικοί σύνδεσμοι που χρειάζονται για την σύνταξη ενός κανόνα σε προτασιακή λογική

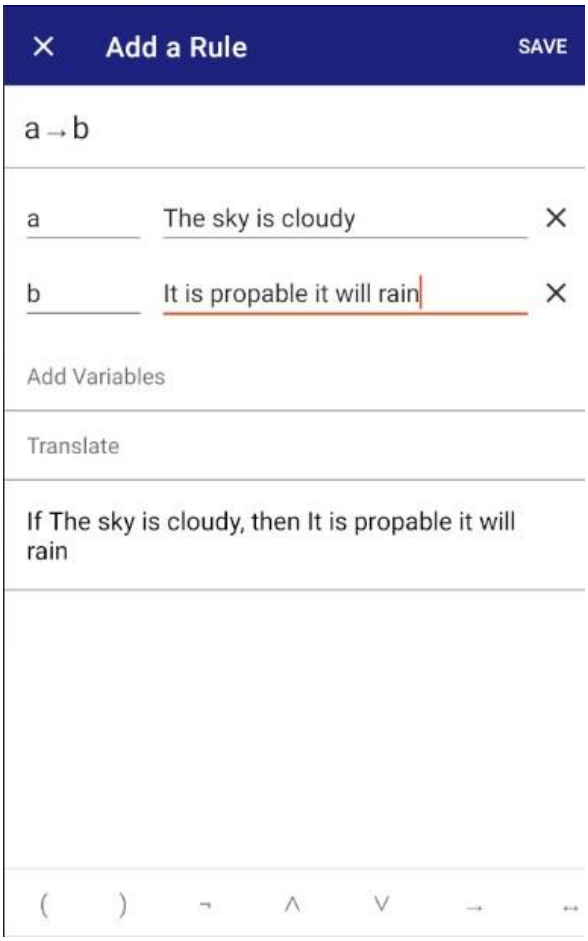
```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:scrollbars="vertical"
    tools:context=".ui.fragments.dialog_fragments.EditRuleDialog">

    <android.support.v7.widget.Toolbar
        android:id="@+id/toolbar"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:background="@color/colorPrimary"
        android:minHeight="?attr/actionBarSize"
        android:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar"
        app:navigationIcon="@drawable/ic_close_white_24dp"
        app:popupTheme="@style/ThemeOverlay.AppCompat.Light"
        app:title="Add a Rule"
        app:titleTextColor="@color/colorTextPrimary"/>

    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_below="@id/toolbar">

        <ScrollView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginBottom="48dp">

            <LinearLayout
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:orientation="vertical">

                <EditText
                    android:id="@+id/edit_text_rule"
                    android:layout_width="match_parent"
                    android:layout_height="match_parent"
                    android:padding="14dp"
                    android:background="@android:color/transparent"
                    android:hint="Rule"
                    android:focusable="true"
                    android:focusableInTouchMode="true"
                    android:inputType="text"
                    android:imeOptions="actionDone"
                    android:scrollbars="vertical"
                    android:textSize="22sp" />

                <View
                    android:layout_width="match_parent"
                    android:layout_height="1dp"
                    android:background="#a4a4a4" />

                <LinearLayout
                    android:id="@+id/insertion_point"
```

```
            android:id="@+id/insertion_point"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical"
            android:padding="8dp"
            android:visibility="gone"/>

        <android.support.design.button.MaterialButton
            android:id="@+id/add_propositions_button"
            style="@style/Widget.MaterialComponents.Button.TextButton"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:fontFamily="sans-serif"
            android:gravity="start|center_vertical"
            android:letterSpacing="0"
            android:paddingLeft="16dp"
            android:text="Add Propositions"
            android:textAllCaps="false"
            android:textColor="@color/common_google_signin_btn_text_light"
            android:textSize="16sp"
            app:iconPadding="16dp"
            app:iconTint="@color/common_google_signin_btn_text_light" />
```

```

        <View
            android:layout_width="match_parent"
            android:layout_height="1dp"
            android:background="#a4a4a4" />

        <android.support.design.button.MaterialButton
            android:id="@+id/translate"
            style="@style/Widget.MaterialComponents.Button.TextButton"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:fontFamily="sans-serif"
            android:gravity="start|center_vertical"
            android:letterSpacing="0"
            android:paddingLeft="16dp"
            android:text="Translate"
            android:textAllCaps="false"
            android:textColor="@color/common_google_signin_btn_text_light"
            android:textSize="16sp"
            app:iconPadding="16dp"
            app:iconTint="@color/common_google_signin_btn_text_light" />

        <View
            android:layout_width="match_parent"
            android:layout_height="1dp"
            android:background="#a4a4a4" />

```

```
107
108                 <LinearLayout
109                     android:id="@+id/result_layout"
110                     android:layout_width="match_parent"
111                     android:layout_height="wrap_content"
112                     android:orientation="vertical"
113                     android:visibility="gone" >
114
115                     <TextView
116                         android:id="@+id/result"
117                         android:layout_width="match_parent"
118                         android:layout_height="wrap_content"
119                         android:padding="16dp"
120                         android:text="Result"
121                         android:textAppearance="@android:color/black"
122                         android:textSize="18sp"
123                         />
124
125                     <View
126                         android:layout_width="match_parent"
127                         android:layout_height="1dp"
128                         android:background="#a4a4a4" />
129
130                 </LinearLayout>
131             </LinearLayout>
132         </ScrollView>
133
```

```
134         <LinearLayout
135             android:id="@+id/connectives_layout"
136             android:layout_width="match_parent"
137             android:layout_height="wrap_content"
138             android:layout_alignParentBottom="true"
139             android:orientation="vertical">
140
141             <View
142                 android:layout_width="match_parent"
143                 android:layout_height="2dp"
144                 android:layout_above="@id/bottom_navigation"
145                 android:background="@drawable/shadow" />
146
147             <android.support.v7.widget.RecyclerView
148                 android:id="@+id/connectives_recycler_view"
149                 android:layout_width="wrap_content"
150                 android:layout_height="wrap_content"
151                 android:layout_gravity="center"
152                 tools:listitem="@layout/list_item_connective" />
153         </LinearLayout>
154
155     </RelativeLayout>
156
157 </RelativeLayout>
```



Εικόνα 29: Η διεπαφη του edit rule fragment

4.7.6 Το αρχείο EditRuleDialog.java

Η κλάση EditRuleDialog επεκτείνει την κλάση DialogFragment. Στην μέθοδο onCreate() ορίζεται το οπτικό στυλ του παραθύρου το οποίο είναι σχεδιασμένο στο αρχείο styles.xml. Περιλαμβάνει τις παρακάτω μεθόδους.

Στην μέθοδο onStart() ορίζεται το μέγεθος του παραθύρου, και το animation το οποίο θα γίνεται η μετάβαση από το RuleFragment στο EditRuleDialog.
Στην μέθοδο onCreateView() γίνεται αρχικοποίηση όλων των views της διεπαφής του EditRuleDialog.

Στην μέθοδο onViewCreated() ορίζεται ο adapter του recycler view που περιέχει του λογικούς συνδέσμους, ο τίτλος και το μενού επιλογών του ToolBar. Όταν γίνεται προσθήκη ενός νέου κανόνα το toolbar έχει τον τίτλο “Add a Rule” και το μενού έχει την επιλογή SAVE, δηλαδή την αποθήκευση του κανόνα στην βάση δεδομένων rules.db. Όταν γίνεται τροποποίηση ενός κανόνα τότε το toolbar έχει τον τίτλο “Edit a Rule” και το μενού εκτός από την επιλογή SAVE διαθέτει και την επιλογή Delete, δηλαδή την διαγραφή του κανόνα. Όταν ο χρήστης

πιέσει την επιλογή save τότε γίνεται έλεγχος για την εγκυρότητα της σύνταξης του κανόνα ενώ όταν πιέσει την επιλογή Delete τότε εμφανίζεται προειδοποιητικό παράθυρο στην οθόνη. Στη συνέχεια στα κουμπιά Add Propostion και Translate καταχωρούνται listeners οι οποίοι, όταν πατηθούν τα κουμπιά, καλούν τις μεθόδους *addPropostion* και *translate* αντίστοιχα.

Η μέθοδος *addPropostion* εισάγει ένα σύνολο από views οριστεί στο αρχείο layout_add_propositions. Έπειτα γίνεται αρχικοποίηση των EditText με μεταβλητές symbolEdit κι propositionEdit καθώς επίσης και του ImageButton με μεταβλητή removePropostion. Στο symbolEdit όπου ο χρήστης εισάγει το σύμβολο της ατομικής πρότασης, έχει εγγραφεί ένας listener της κλάσης TextWatcher. Σκοπός αυτού του listener είναι να εντοπίζει αλλαγές που γίνονται στο κείμενο στο πεδίο symbolEdit. Συγκεκριμένα, εάν το πεδίο είναι κενό και ο χρήστης πληκτρολογήσει έναν σύμβολο τότε ο text watcher εντοπίζει την αλλαγή και ελέγχει αν το σύμβολο αυτό είναι αποθηκευμένο στην βάση δεδομένων rules.db. Ο έλεγχος αυτός γίνεται μέσω της μεθόδου searchProposition η οποία επιστέφει την πρόταση που αντιστοιχεί στο σύμβολο του symbolEdit και έπειτα η πρόταση αυτή συμπληρώνεται στο propositionEdit.

Στη μέθοδο *getSymbolProposition* τα σύμβολα ατομικών προτάσεων και οι αντίστοιχες προτάσεις τους, εισάγονται σε έναν χάρτη τύπου HashMap ο οποίος έχει οριστεί με την μεταβλητή varMap .

Η μέθοδος translate() αρχικά χρησιμοποιεί έναν logic parser ο οποίος κάνει συντακτική ανάλυση του κανόνα. Ο logic parser δημιουργεί ένα συντακτικό δέντρο (concrete syntax tree) της πρότασης όπου κάθε φύλλο του δέντρου είναι μια ατομική πρόταση. Στη συνέχεια, για κάθε εγγραφή του χάρτη varMap ορίζεται ως επισκέπτης (visitor) του δέντρου ένα αντικείμενο της κλάσης SetPropostion ο οποίος αναθέτει κάθε πρόταση σε φυσική γλώσσα στη κατάλληλη ατομική πρόταση, δηλαδή στο κατάλληλο φύλλο του δέντρου.

Η μέθοδος checkRuleValidity() έχει ως σκοπό τον έλεγχο της εγκυρότητας της σύνταξης του κανόνα. Συγκεκριμένα ελέγχει εάν η πρόταση που εισήγαγε ο χρήστης είναι πρόταση Horn.

Η μέθοδος saveRule() καλεί την μέθοδο insertRule() εάν ο κανόνας είναι καινούργιος, ενώ εάν είναι τροποποιημένος καλεί την μέθοδο updateRule()

Η μέθοδος insertRule() αποθηκεύει την πρόταση προτασιακής λογικής και την πρόταση φυσικής γλώσσας στον πίνακα rules της βάσης δεδομένων rules.db. Στη συνέχεια καλεί την μέθοδο *insertSymbols*.

Η μέθοδος *insertSymbols* περιλαμβάνει μια δομή επανάληψης for, η οποία σε κάθε επανάληψη αποθηκεύει μια εγγραφή του χάρτη symbol_Propositon στον πίνακα symbols της βάσης δεδομένων rules.db. Εάν κάποιο κλειδί του χάρτη, δηλαδή ένα σύμβολο μια ατομικής πρότασης είναι ήδη αποθηκευμένο, τότε γίνεται αναζήτηση του id με το οποίο το σύμβολο αυτό είναι εγγεγραμμένο στον πίνακα symbols. Έπειτα καλείται η μέθοδος *insertRuleSymbol.*

Η μέθοδος *insertRuleSymbol* αποθηκεύει το id του κανόνα και το id μιας ατομικής του πρότασης στον πίνακα rules_symbols της βάσης δεδομένων rules.db.

Η μέθοδος updateRule αποθηκεύει έναν τροποποιημένο κανόνα στον πίνακα rules. Η αντιστοιχία ατομικής πρότασης και πρότασης σε φυσική γλώσσα δεν τροποποιείται επειδή μπορεί να χρησιμοποιηθεί και από άλλους κανόνες. Οπότε καλείται ξανά η μέθοδος *insertSymbols* η οποία όπως προαναφέρθηκε ελέγχει εάν μια ατομική πρόταση είναι ήδη αποθηκευμένη στον πίνακα symbols.

Η μέθοδος deleteRule() αρχικά διαγράφει την εγγραφή του πίνακα rules_symbols που έχει ως τιμή στο πεδίο rule_id το id του κανόνα που πρόκειται να διαγραφεί, διότι το πεδίο rule_id είναι ένα ξένο κλειδί που αναφέρεται στο πεδίο id του πίνακα rules. Ύστερα γίνεται διαγραφή του κανόνα από τον πίνακα rules.

Στη μέθοδο searchProposition() γίνεται αναζήτηση της πρότασης εκφρασμένης σε φυσική γλώσσα που αντιστοιχεί σε μια ατομική πρόταση στο πίνακα symbols.

Η μέθοδος reloadData() καλείται όταν ένας κανόνας υπάρχει ήδη στην βάση δεδομένων και ανασύρει τα δεδομένα του κανόνα από τον πίνακα rules και την ένωση των πινάκων symbols και rules_symbols. Όλα τα δεδομένα του κανόνα συμπληρώνονται στα απαραίτητα views και ο χρήστης μπορεί να τα τροποποιήσει. Ακολουθεί ο κώδικας του αρχείου EditRuleDialog.java

```
1   package com.drid.group_reasoning.ui.fragments.dialog_fragments;
2
3   import ...
52
53
54  public class EditRuleDialog extends DialogFragment {
55
56      public static final String TAG = EditRuleDialog.class.getSimpleName();
57      private Toolbar toolbar;
58
59      private EditText ruleEditText;
60      private TextView resultTextView;
61
62      private View resultLayout;
63
64      private Button addPropositionButton;
65      private Button translateButton;
66
67      private ViewGroup viewGroup;
68
69      private LogicParser logicParser;
70
71      private Map<String, String> symbol_proposition;
72
73      private Bundle arguments;
74      private Uri currentUri;
75
76      private EditText symbolEdit;
77      private EditText propositionEdit;
78
79      private RecyclerView connectives;
80      private ConnectiveAdapter adapter;
81
```

```
82      private boolean ruleIsChanged = false;
83
84      private View.OnTouchListener onTouchListener = (v, event) → {
87              ruleIsChanged = true;
88              return false;
89      };
91
92
93      @Override
94      public void onCreate(@Nullable Bundle savedInstanceState) {
95          super.onCreate(savedInstanceState);
96          setStyle(DialogFragment.STYLE_NORMAL, R.style.FullscreenDialog);
97      }
98
99      @Override
100     public void onStart() {
101         super.onStart();
102         Dialog dialog = getDialog();
103         if (dialog != null) {
104             int width = ViewGroup.LayoutParams.MATCH_PARENT;
105             int height = ViewGroup.LayoutParams.MATCH_PARENT;
106             dialog.getWindow().setLayout(width, height);
107             dialog.getWindow().setWindowAnimations(R.style.AppTheme_Slide);
108             dialog.getWindow().setSoftInputMode(WindowManager.LayoutParams.SOFT_INPUT_ADJUST_RESIZE);
109         }
110     }
```

```
111
112
113     @Nullable
114     @Override
115     public View onCreateView(
116             @NonNull LayoutInflater inflater,
117             @Nullable ViewGroup container,
118             @Nullable Bundle savedInstanceState) {
119
120         View view = inflater.inflate(R.layout.fragment_editor_rule, container,  attachToRoot: false);
121
122         toolbar = view.findViewById(R.id.toolbar);
123
124         ruleEditText = view.findViewById(R.id.edit_text_rule);
125         resultLayout = view.findViewById(R.id.result_layout);
126         resultTextView = view.findViewById(R.id.result);
127
128         ruleEditText.setOnTouchListener(onTouchListener);
129
130         viewGroup = view.findViewById(R.id.insertion_point);
131
132
133         addPropositionButton = view.findViewById(R.id.add_propositions_button);
134
135         translateButton = view.findViewById(R.id.translate);
136
137         connectives = view.findViewById(R.id.connectives_recycler_view);
138
139         connectives.setLayoutManager(new LinearLayoutManager(
140                 getContext(), LinearLayoutManager.HORIZONTAL,  reverseLayout: false));
```

```
141
142         return view;
143     }
144
145
146     @Override
147     public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {
148         super.onViewCreated(view, savedInstanceState);
149
150         adapter = new ConnectiveAdapter(getContext());
151         connectives.setAdapter(adapter);
152
153         logicParser = new LogicParser();
154         symbol_proposition = new LinkedHashMap<>();
155
156         arguments = this.getArguments();
157
158         if (arguments != null) {
159             currentUri = Uri.parse(arguments.getString(ARG_RULE_URI));
160         }
161
162
163         if (currentUri == null) {
164             toolbar.setTitle("Add a Rule");
165         } else {
166             toolbar.setTitle("Edit a Rule");
167             reloadData();
168         }
169
170         toolbar.setNavigationOnClickListener((v) -> {
173             if (!ruleIsChanged) {
174                 dismiss();
175             } else {
176                 displayDiscardAlertDialog();
177             }
178         });
180
181         toolbar.inflateMenu(R.menu.editor_options_menu);
182
183         if (currentUri == null) {
184             MenuItem deleteMenuItem = toolbar.getMenu().findItem(R.id.delete);
185             deleteMenuItem.setVisible(false);
186         }
187
188         toolbar.setOnMenuItemClickListener((menuItem) -> {
191             switch (menuItem.getItemId()) {
192                 case R.id.save:
193                     checkRuleValidity();
194                     return true;
195                 case R.id.delete:
196                     displayDeleteAlertDialog();
197                     return true;
198             }
199             return false;
200         });
204
```

```
204
205         addPropositionButton.setOnClickListener((v) → {
208                 addProposition(viewGroup.getChildCount());
209         });
211
212
213         translateButton.setOnClickListener((v) → { translate(); });
219
220         adapter.setOnItemClickListener((v, position) → {
223                 int start = ruleEditText.getSelectionStart(); //this is to get the the cursor position
224                 String s = adapter.getSymbol(position);
225                 ruleEditText.getText().insert(start, s);
226         });
228
229     }
230
231     private void displayDiscardAlertDialog() {
232         AlertDialog dialog = new AlertDialog.Builder(
233                 new ContextThemeWrapper(getContext(), R.style.AlertDialog))
234                 .setTitle("Do you want to discard changes ?")
235                 .setPositiveButton( text: "Keep Editing", (dialog, which) → {
238                         dialog.dismiss();
239                 }).setNegativeButton( text: "Discard", (dialog, which) → {
243                         dismiss();
244                 }).create();
246
247         dialog.show();
248         dialog.setCancelable(false);
249         dialog.setCanceledOnTouchOutside(false);
250     }
251
252     private void displayDeleteAlertDialog() {
253         AlertDialog dialog = new AlertDialog.Builder(new ContextThemeWrapper(getContext(), R.style.AlertDialog))
254                 .setTitle("Are you sure you want to delete this rule ?")
255                 .setPositiveButton( text: "Yes", (dialog, which) → {
258                         deleteRule();
259                         dismiss();
260                 }).setNegativeButton( text: "No", (dialog, which) → {
264                         dialog.dismiss();
265                 }).create();
267
268         dialog.show();
269         dialog.setCancelable(false);
270         dialog.setCanceledOnTouchOutside(false);
271
272     }
273
274
275     private void addProposition(int position) {
276         viewGroup.setVisibility(View.VISIBLE);
277         LayoutInflater inflater = getLayoutInflater();
278         final View view = inflater.inflate(R.layout.layout_add_propositions,  root: null);
279         ImageButton removeProposition = view.findViewById(R.id.remove_proposition_button);
280         removeProposition.setOnClickListener((v) → {
283                 viewGroup.removeView(view);
284         });
286         viewGroup.addView(view, position);
```

```
        symbolEdit = getSymbolEditText(view);
        propositionEdit = getPropositionEditText(view);

        symbolEdit.setOnTouchListener(onTouchListener);
        propositionEdit.setOnTouchListener(onTouchListener);

        symbolEdit.addTextChangedListener(new TextWatcher() {
            @Override
            public void beforeTextChanged(CharSequence s, int start, int count, int after) {
            }

            @Override
            public void onTextChanged(CharSequence s, int start, int before, int count) {
            }

            @Override
            public void afterTextChanged(Editable s) {
                String symbol = symbolEdit.getText().toString();
                String proposition = searchProposition(symbol);
                if (!TextUtils.isEmpty(symbol) && !TextUtils.isEmpty(proposition)) {
                    propositionEdit.setText(proposition);
                } else {
                    propositionEdit.setText("");
                }
            }
        });
```

```
    }

    private void translate() {
        Sentence sentence = null;

        if (!TextUtils.isEmpty(ruleEditText.getText())) {
            sentence = logicParser.parse(ruleEditText.getText().toString());
        } else {
            Toast.makeText(getContext(),
                    text: "Please type in a rule",
                    Toast.LENGTH_SHORT)
                    .show();
        }

        for (Map.Entry<String, String> entry : getSymbolProposition().entrySet()) {
            sentence.accept(new SetProposition(entry.getKey(), entry.getValue()));
        }


        if (sentence != null && sentence.getProposition() != null) {
            String nlSentence = sentence.getProposition();
            if (!TextUtils.isEmpty(nlSentence)) {
                resultLayout.setVisibility(View.VISIBLE);
                resultTextView.setText(new StringBuilder()
                        .append(nlSentence.substring(0, 1).toUpperCase())
                        .append(nlSentence.substring(1))
                        .toString());
            }
        } else {
            Toast.makeText(getContext(),
                    text: "Something went wrong with translation",
```

```
                    Toast.LENGTH_SHORT)
                    .show();
        }
    }


    private void checkRuleValidity() {
        String rulePl = ruleEditText.getText().toString().trim();
        String ruleNl = resultTextView.getText().toString();

        Sentence sentence = logicParser.parse(rulePl);
        Log.i(TAG,  msg: "checkRuleValidity: sentence: " + sentence);

        Sentence cnf = Sentence.convertToCnf(sentence);

        Log.i(TAG,  msg: "checkRuleValidity: cnf sentence: " + cnf);

        List<Literal> literals = LiteralCollector.getLiterals(cnf);

        Log.i(TAG,  msg: "checkRuleValidity: literals: " + literals);

        Clause clause = new Clause(literals);

        Log.i(TAG,  msg: "checkRuleValidity: " + clause);


        saveRule(rulePl, ruleNl);
        dismiss();
    }


    private void saveRule(String rulePl, String ruleNl) {

        if (currentUri == null) {
            insertRule(rulePl, ruleNl, getSymbolProposition());
        } else {
            updateRule(currentUri, rulePl, ruleNl, getSymbolProposition());
        }

    }

    private void insertRule(String rulePl, String ruleNl, Map<String, String> symbolProposition) {
        ContentValues rules = new ContentValues();

        rules.put(RuleEntry.COLUMN_RULE_PL, rulePl);
        rules.put(RuleEntry.COLUMN_RULE_NL, ruleNl);

        Uri newRuleUri = getActivity().getContentResolver().insert(RuleEntry.RULES_URI, rules);

        insertSymbols(symbolProposition, newRuleUri);
    }

    private void insertSymbols(Map<String, String> symbolProposition, Uri newRuleUri) {

        for (Map.Entry<String, String> entry : symbolProposition.entrySet()) {
            ContentValues symbols = new ContentValues();
```

```
            symbols.put(SymbolEntry.COLUMN_SYMBOL, entry.getKey());
            symbols.put(SymbolEntry.COLUMN_PROPOSITION, entry.getValue());
            Uri newSymbolUri = getActivity().getContentResolver()
                    .insert(SymbolEntry.SYMBOLS_URI, symbols);
            if (newSymbolUri == null) {

                Uri symbolUri =
                        SymbolEntry.SYMBOLS_URI.buildUpon().appendPath(PATH_SYMBOL).build();


                Cursor symbolCursor = getActivity().getContentResolver().query(
                        symbolUri,
                        new String[]{SymbolEntry._ID},
                         selection: SymbolEntry.COLUMN_SYMBOL + "=?",
                        new String[]{symbols.getAsString(SymbolEntry.COLUMN_SYMBOL)},

                         sortOrder: null
                );

                if (symbolCursor.moveToFirst()) {
                    int symbol_id =
                            symbolCursor.getInt(symbolCursor.getColumnIndex(SymbolEntry._ID));
                    newSymbolUri =
                            ContentUris.withAppendedId(SymbolEntry.SYMBOLS_URI, symbol_id);
                }
            }
            insertRuleSymbol(newRuleUri, newSymbolUri);
        }
    }

    private void insertRuleSymbol(Uri newRuleUri, Uri newSymbolUri) {
        ContentValues rules_symbols = new ContentValues();

        long rule_id = ContentUris.parseId(newRuleUri);
        long symbol_id = ContentUris.parseId(newSymbolUri);


        rules_symbols.put(RuleSymbolEntry.KEY_RULE_ID, rule_id);
        rules_symbols.put(RuleSymbolEntry.KEY_SYMBOL_ID, symbol_id);

        getActivity().getContentResolver()
                .insert(RuleSymbolEntry.RULE_SYMBOL_URI, rules_symbols);

    }

    private void updateRule(
            Uri currentUri, String rulePl, String ruleNl, Map<String, String> symbol_proposition) {


        ContentValues rules = new ContentValues();

        rules.put(RuleEntry.COLUMN_RULE_PL, rulePl);
        rules.put(RuleEntry.COLUMN_RULE_NL, ruleNl);

        getActivity().getContentResolver().update(
                this.currentUri, rules,  where: null,  selectionArgs: null);
```

```
458
459         getActivity().getContentResolver().delete(
460                 RuleSymbolEntry.RULE_SYMBOL_URI,
461                  where: RuleSymbolEntry.KEY_RULE_ID + "=?",
462                 new String[]{String.valueOf(ContentUris.parseId(currentUri))}
463         );
464         insertSymbols(symbol_proposition, currentUri);
465     }
466
467     private void deleteRule() {
468
469         getActivity().getContentResolver().delete(
470                 RuleSymbolEntry.RULE_SYMBOL_URI,
471                  where: RuleSymbolEntry.KEY_RULE_ID + "=?",
472                 new String[]{String.valueOf(currentUri)}
473         );
474
475         getActivity().getContentResolver().delete(currentUri,  where: null,  selectionArgs: null);
476     }
477
478     public Map<String, String> getSymbolProposition() {
479
480         for (int i = 0; i < viewGroup.getChildCount(); i++) {
481             View child = viewGroup.getChildAt(i);
482
483             String symbol = getSymbolEditText(child).getText().toString();
484             String proposition = getPropositionEditText(child).getText().toString();
485             symbol_proposition.put(symbol, proposition);
486         }
487
488         return symbol_proposition;
489     }
490
491     private String searchProposition(String symbol) {
492
493         String result = null;
494         Uri symbolUri = SymbolEntry.SYMBOLS_URI.buildUpon().appendPath(PATH_SYMBOL).build();
495
496         Cursor symbolCursor = getActivity().getContentResolver().query(
497                 symbolUri,
498                  projection: null,
499                  selection: SymbolEntry.COLUMN_SYMBOL + "=?",
500                 new String[]{symbol},
501                  sortOrder: null
502         );
503
504         if (symbolCursor.moveToFirst()) {
505             result = symbolCursor.getString(symbolCursor.getColumnIndex(SymbolEntry.COLUMN_PROPOSITION));
506         }
507
508         return result;
509     }
510
511     private void reloadData() {
512
513         long id = ContentUris.parseId(currentUri);
```

```
514
515         Cursor ruleCursor = getActivity().getContentResolver().query(
516                 currentUri,
517                 new String[]{
518                         RuleEntry.COLUMN_RULE_PL, RuleEntry.COLUMN_RULE_NL
519                 },
520                  selection: RuleEntry._ID + "=?",
521                 new String[]{String.valueOf(id)},
522                  sortOrder: null
523         );
524
525         if (ruleCursor.moveToFirst()) {
526             ruleEditText.setText(
527                     ruleCursor.getString(ruleCursor.getColumnIndex(RuleEntry.COLUMN_RULE_PL)));
528             resultLayout.setVisibility(View.VISIBLE);
529             resultTextView.setText(
530                     ruleCursor.getString(ruleCursor.getColumnIndex(RuleEntry.COLUMN_RULE_NL)));
531         }
532
533         ruleCursor.close();
534
535
536         Cursor joinedTablesCursor = getActivity().getContentResolver().query(
537                 RuleSymbolEntry.RULE_SYMBOL_URI,
538                 new String[]{SymbolEntry.COLUMN_SYMBOL, SymbolEntry.COLUMN_PROPOSITION},
539                  selection: RuleSymbolEntry.KEY_RULE_ID + "=?",
540                 new String[]{String.valueOf(id)},
541                  sortOrder: null
542         );
543
544         try {
545             while (joinedTablesCursor.moveToNext()) {
546                 int index = joinedTablesCursor.getPosition();
547                 addProposition(viewGroup.getChildCount());
548                 symbolEdit = viewGroup.getChildAt(index).findViewById(R.id.symbol_edit);
549                 propositionEdit = viewGroup.getChildAt(index).findViewById(R.id.proposition_edit);
550
551                 symbolEdit.setText(
552                         joinedTablesCursor.getString(
553                                 joinedTablesCursor.getColumnIndex(SymbolEntry.COLUMN_SYMBOL)));
554                 propositionEdit.setText(
555                         joinedTablesCursor.getString(
556                                 joinedTablesCursor.getColumnIndex(SymbolEntry.COLUMN_PROPOSITION)));
557             }
558         } finally {
559             joinedTablesCursor.close();
560         }
561     }
562
563 @   private EditText getSymbolEditText(View view) { return view.findViewById(R.id.symbol_edit); }
566
567 @   private EditText getPropositionEditText(View view) {
568         return view.findViewById(R.id.proposition_edit);
569     }
570 }
```

4.8 Το FactFragment

Το FactFragment αποτελεί το τμήμα του KnowledgeFragment που εμφανίζει στην οθόνη μια λίστα με γεγονότα τα οποία είναι εκφρασμένα σε φυσική γλώσσα και προτασιακή λογική. Όταν το fragment αυτό είναι επιλεγμένο από τον view pager τότε η λειτουργία του κουμπιού προσθήκης είναι να δημιουργεί ένα fragment στο οποίο ο χρήστης εισάγει έναν νέο γεγονός.

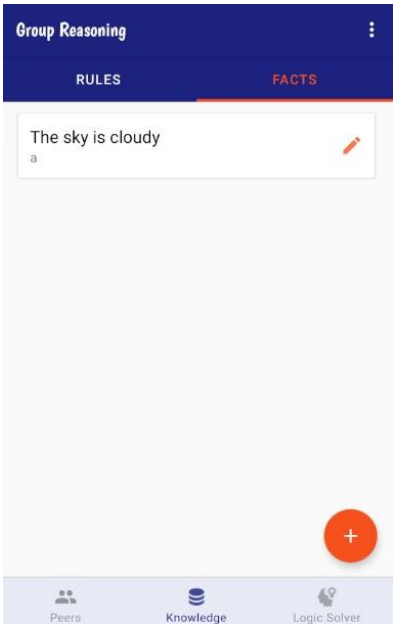
4.8.1 Το αρχείο fragment_fact.xml

Η διεπαφή του Fact fragment περιλαμβάνει ένα root view τύπου RelativeLayout και ένα RecyclerView το οποίο είναι η λίστα με τα γεγονότα. Ακολουθούν ο κώδικας του αρχείου fragment_fact.xml και η τελική μορφή της διεπαφής του Fact fragment:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    android:scrollbars="vertical"
    app:layout_behavior="android.support.design.widget.AppBarLayout$ScrollingVie..."
    tools:context=".ui.fragments.dialog_fragments.EditRuleDialog">

    <android.support.v7.widget.Toolbar
        android:id="@+id/toolbar"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:background="@color/colorPrimary"
        android:minHeight="?attr/actionBarSize"
        app:title="Add a Fact"
        android:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar"
        app:navigationIcon="@drawable/ic_close_white_24dp"
        app:popupTheme="@style/ThemeOverlay.AppCompat.Light"
        app:titleTextColor="@color/colorTextPrimary" />

    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:padding="16dp">

        <EditText
            android:id="@+id/edit_text_fact_symbol"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:background="@android:color/transparent"
            android:hint="Symbol"
            android:inputType="textMultiLine"
            android:padding="14dp"
            android:scrollbars="vertical"
            android:textSize="22sp" />
```

```
        <View
            android:layout_width="match_parent"
            android:layout_height="1dp"
            android:background="#a4a4a4" />

        <EditText
            android:id="@+id/edit_text_fact_proposition"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:background="@android:color/transparent"
            android:hint="Proposition"
            android:inputType="textMultiLine"
            android:padding="14dp"
            android:scrollbars="vertical"
            android:textSize="22sp" />

        <View
            android:layout_width="match_parent"
            android:layout_height="1dp"
            android:background="#a4a4a4" />
    </LinearLayout>

</LinearLayout>
```

Ο κώδικας του αρχείου framrnt_fact.xml



Εικόνα 30: Η διεπαφη του fact fragment

4.8.2 Το αρχείο FactFragment.java

Η λογική του FactFragment είναι παρόμοια με εκείνη του RuleFragment. Η κλάση FactFragment επεκτείνει την κλάση Fragment και υλοποιεί τις μεθόδους της διεπαφής LoaderManager.LoaderCallbacks<Cursor>.

Στην μέθοδο onCreateView δημιουργούνται τα components της διεπαφής του FactFragment. Στην μέθοδο onViewCreated ορίζεται ο adapter του recycler view, και στη συνέχεια ορίζεται στο recycler view ένας listener ο οποίος εμφανίζει το κουμπί προσθήκης όταν ο χρήστης σύρει το δάκτυλο του προς τα πάνω (scroll up) ή το κρύβει αν το σύρει προς τα κάτω (scroll down).

Η μέθοδος onResume ενεργοποιεί τον loader ο οποίος ανασύρει τα δεδομένα από την βάση δεδομένων και ορίζει τη λειτουργικότητα του κουμπιού προσθήκης. Δηλαδή καταχωρεί έναν listener ο οποίος καλεί την μέθοδο openDialog

Η μέθοδος openDialog δημιουργεί ένα νέο παράθυρο ή dialog fragment της κλάσης EditFactFragment. Ακολουθεί ο κώδικας σε Java του αρχείου FactFragment.java

```java
package com.drid.group_reasoning.ui.fragments;

import ...

public class FactFragment extends Fragment implements LoaderManager.LoaderCallbacks<Cursor> {

    public static final String TAG = FactFragment.class.getSimpleName();

    public static final String ARG_FACT_URI = "fact_uri";

    private static final int FACT_LOADER = 1;

    private LinearLayout emptyList;
    private FloatingActionButton fab;
    private RecyclerView recyclerView;

    private FactAdapter factAdapter;

    private EditFactDialog dialog;


    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setHasOptionsMenu(true);
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        // Inflate the layout for this fragment
        View view = inflater.inflate(R.layout.fragment_fact, container,  attachToRoot: false);
```

```
61
62          emptyList = view.findViewById(R.id.empty_facts_list);
63
64          recyclerView = view.findViewById(R.id.facts_list);
65          recyclerView.setLayoutManager(new LinearLayoutManager(getContext()));
66
67          factAdapter = new FactAdapter(getContext(),  cursor: null);
68          recyclerView.setAdapter(factAdapter);
69
70          KnowledgeFragment knowledgeFragment = (KnowledgeFragment) getParentFragment();
71
72          fab = knowledgeFragment.getFab();
73
74          return view;
75      }
76
77      @Override
78      public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {
79          super.onViewCreated(view, savedInstanceState);
80          factAdapter.setOnClickListener((v, position) → {
83                  Uri selectedUri = ContentUris.withAppendedId(FactEntry.FACT_URI, position);
84
85                  System.out.println(selectedUri.toString());
86                  Bundle arguments = new Bundle();
87                  arguments.putString(ARG_FACT_URI, String.valueOf(selectedUri));
88
89                  openDialog(arguments);
90          });
92
93          recyclerView.addOnScrollListener(onScrolled(recyclerView, dx, dy) → {
96                  if (dy > 0) {
97                      fab.hide();
98                  } else {
99                      fab.show();
100                 }
101                 super.onScrolled(recyclerView, dx, dy);
102         });
104     }
105
106     @Override
107     public void setUserVisibleHint(boolean isVisible) {
108         super.setUserVisibleHint(isVisible);
109
110         if (isVisible && isResumed()) {
111             onResume();
112         }
113     }
114
115     @Override
116     public void onResume() {
117         super.onResume();
118
119         if (!getUserVisibleHint()) {
120             return;
121         }
122         getLoaderManager().initLoader(FACT_LOADER,  bundle: null,  loaderCallbacks: this);
```

```
        KnowledgeFragment knowledgeFragment = (KnowledgeFragment) getParentFragment();

        fab = knowledgeFragment.getFab();
        fab.setOnClickListener((v) -> { openDialog( arguments: null); });

    }

    @NonNull
    @Override
    public Loader<Cursor> onCreateLoader(int i, @Nullable Bundle bundle) {
        String[] projection = {
                FactEntry._ID,
                FactEntry.COLUMN_FACT_SYMBOL,
                FactEntry.COLUMN_FACT_PROPOSITION};

        return new CursorLoader(
                getContext(),
                FactEntry.FACT_URI,
                projection,
                 selection: null,
                 selectionArgs: null,
                 sortOrder: null);
    }

    @Override
    public void onLoadFinished(@NonNull Loader<Cursor> loader, Cursor cursor) {
        factAdapter.swapCursor(cursor);
        emptyList.setVisibility(cursor.getCount() == 0 ? View.VISIBLE : View.GONE);
    }

    @Override
    public void onLoaderReset(@NonNull Loader<Cursor> loader) { factAdapter.swapCursor( newCursor: null); }


    @Override
    public void onCreateOptionsMenu(Menu menu, MenuInflater inflater) {
        menu.clear();
        inflater.inflate(R.menu.fact_options_menu, menu);
        super.onCreateOptionsMenu(menu, inflater);
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case R.id.delete_all_facts:
                displayDeleteAllFactsDialog();
                return true;
        }

        return super.onOptionsItemSelected(item);
    }

    private void displayDeleteAllFactsDialog() {
        AlertDialog dialog = new AlertDialog.Builder(
                new ContextThemeWrapper(getContext(), R.style.AlertDialog))
                .setTitle("Are you sure you want to delete all facts ?")
                .setPositiveButton( text: "Yes", (dialog, which) -> {
```

```
                        deleteAllFacts();
                }).setNegativeButton( text: "No", (dialog, which) → {
                        dialog.dismiss();
                }).create();
        dialog.show();
        dialog.setCancelable(false);
        dialog.setCanceledOnTouchOutside(false);
    }

    private void deleteAllFacts() {
        getActivity().getContentResolver().delete(
                FactEntry.FACT_URI,  where: null,  selectionArgs: null);
    }

    private void openDialog(Bundle arguments) {
        dialog = new EditFactDialog();
        dialog.show(getChildFragmentManager(), EditFactDialog.TAG);
        dialog.setArguments(arguments);
    }
}
```

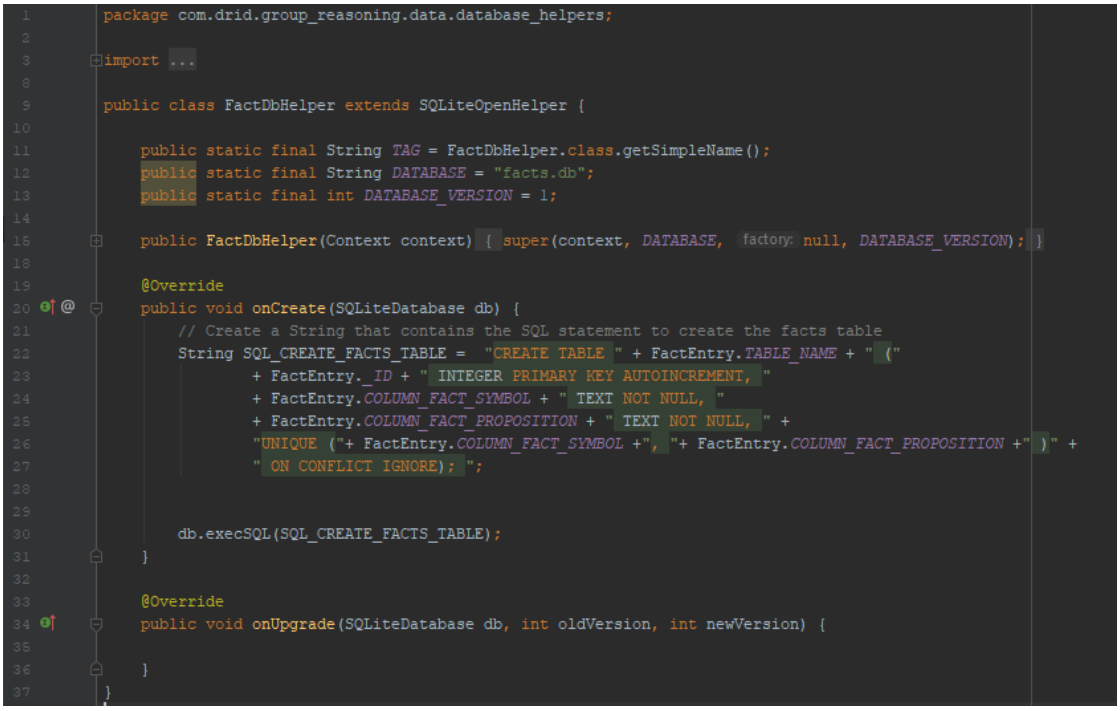
4.8.3 Το αρχείο FactDbHelper.java

Η κλάση FactDbHelper επεκτείνει την κλάση SQLiteOpenHelper και δημιουργεί την βάση δεδομένων facts. Κληρονομεί και υπερβαίνει την μέθοδο onCreate στην οποία ορίζεται μια συμβολοσειρά η οποία περιγράφει μια εντολή σε γλώσσα Sql για δημιουργία του πίνακα facts. Ο πίνακας peers αποτελείται από τα εξής πεδία. Ακολουθούν παρακάτω ένα στιγμιότυπό του πίνακα facts kai ο κώδικας του αρχείου FactDbHelper.java

Id	fact_nl	fact_pl
1	Alex is 25 years old	A

Πίνακας 10: Στιγμιότυπο του πίνακα facts

Τα πεδία fact_nl και fact_pl είναι μοναδικά δηλαδή δέχονται μια τιμή μόνο μια φορά και δεν επιτρέπεται να ξαναγίνει η εισαγωγή αυτής της τιμής σε αυτά.



```
package com.drid.group_reasoning.data.database_helpers;

import ...

public class FactDbHelper extends SQLiteOpenHelper {

    public static final String TAG = FactDbHelper.class.getSimpleName();
    public static final String DATABASE = "facts.db";
    public static final int DATABASE_VERSION = 1;

    public FactDbHelper(Context context) { super(context, DATABASE,  factory: null, DATABASE_VERSION); }

    @Override
    public void onCreate(SQLiteDatabase db) {
        // Create a String that contains the SQL statement to create the facts table
        String SQL_CREATE_FACTS_TABLE =  "CREATE TABLE " + FactEntry.TABLE_NAME + " ("
                + FactEntry._ID + " INTEGER PRIMARY KEY AUTOINCREMENT, "
                + FactEntry.COLUMN_FACT_SYMBOL + " TEXT NOT NULL, "
                + FactEntry.COLUMN_FACT_PROPOSITION + " TEXT NOT NULL, " +
                "UNIQUE ("+ FactEntry.COLUMN_FACT_SYMBOL +", "+ FactEntry.COLUMN_FACT_PROPOSITION +" )" +
                " ON CONFLICT IGNORE); ";


        db.execSQL(SQL_CREATE_FACTS_TABLE);
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {

    }
}
```

4.8.4 Το αρχείο FactAdapter.java

Στο αρχείο FactAdapter.java ορίζεται ένας προσαρμογέας ο οποίος ορίζει τον τρόπο με τον οποία εμφανίζονται τα δεδομένα στο RecyclerView της διεπαφής του FactFragment.

```java
package com.drid.group_reasoning.ui.adapters;

import ...

public class FactAdapter extends RecyclerView.Adapter<FactAdapter.FactViewHolder> {

    private Context context;
    private Cursor cursor;

    private ClickListener listener;

    public FactAdapter(Context context, Cursor cursor) {
        this.context = context;
        this.cursor = cursor;
    }

    @NonNull
    @Override
    public FactViewHolder onCreateViewHolder(@NonNull ViewGroup viewGroup, int i) {
        View view = LayoutInflater.from(context)
                .inflate(R.layout.list_item_knowledge, viewGroup,  attachToRoot: false);
        return new FactViewHolder(view);
    }

    @Override
    public void onBindViewHolder(@NonNull FactViewHolder factViewHolder, int i) {
        if (!cursor.moveToPosition(i)) {
            return;
        }

        int id = cursor.getInt(cursor.getColumnIndex(FactEntry._ID));
        String factNl = cursor.getString(cursor.getColumnIndex(FactEntry.COLUMN_FACT_SYMBOL));
        String factPl = cursor.getString(cursor.getColumnIndex(FactEntry.COLUMN_FACT_PROPOSITION));

        factViewHolder.id = id;
        factViewHolder.factNlTextView.setText(factNl);
        factViewHolder.factPlTextView.setText(factPl);
    }


    @Override
    public int getItemCount() { return (cursor == null) ? 0 : cursor.getCount(); }

    public void swapCursor(Cursor newCursor) {
        cursor = newCursor;

        if (newCursor != null) {
            notifyDataSetChanged();
        }
    }

    public void setOnClickListener(ClickListener listener) { this.listener = listener; }

    public class FactViewHolder extends RecyclerView.ViewHolder {

        private int id;
        private TextView factNlTextView;
        private TextView factPlTextView;

        public FactViewHolder(@NonNull final View itemView) {
            super(itemView);
```

```
            super(itemView);
            factNlTextView = itemView.findViewById(R.id.data_nl);
            factPlTextView = itemView.findViewById(R.id.data_pl);

            itemView.setOnClickListener((v) -> {
                    listener.onItemClick(v, id);
            });
        }

        public int getId() { return id; }
    }

    public interface ClickListener {
        void onItemClick(View v, int position);
    }
}
```

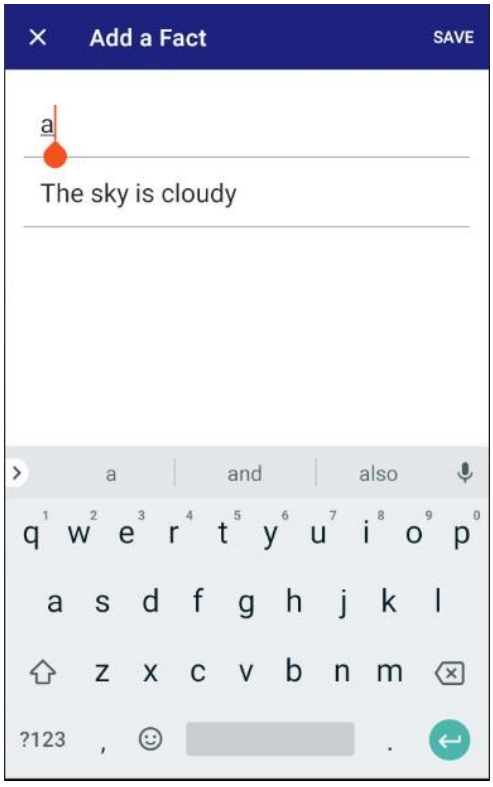
4.8.5 Το αρχείο fragment_editor_fact.xml

Ο Fact editor είναι ένα παράθυρο στο οποίο ο χρήστης μπορεί να προσθέσει έναν νέο γεγονός στη βάση δεδομένων facts.db ή να το τροποποιήσει. Ο χρήστης πληκτρολογεί έναν σύμβολο που αντιπροσωπεύει το γεγονός το οποίο είναι ουσιαστικά μια ατομική πρόταση, και έπειτα αναθέτει σε αυτή, μια πρόταση εκφρασμένη σε φυσική γλώσσα.

Το αρχείο fragment_editor_fact.xml περιγράφει τον σχεδιασμό της διεπαφής του παραθύρου Edit fact dialog, η οποία αποτελείται από τα εξής views:

- Το root view το οποίο ανήκει στην κατηγορία LinearLayout
- Ένα toolbar στο οποίο αναγράφεται ο τίτλος “Add a Fact” όταν πρόκειται για προθήκη νέου γεγονός ή “Edit a Fact ” όταν πρόκειται για τροποποίηση ενός υπάρχοντος γεγονότος. Επίσης διαθέτει ένα κουμπί με την ένδειξη X το οποίο κλείνει το παράθυρο.
- Ένα view group της κλάσης LinearLayout το οποίο περιλαμβάνει τα εξής εμφωλευμένα views:
    - Ένα EditText στο οποίο ο χρήστης πληκτρολογεί το σύμβολο της ατομικής πρότασης που αντιπροσωπεύει ένα γεγονός
    - Ένα EditText στο οποίο ο χρήστης πληκτρολογεί μια πρόταση εκφρασμένη σε φυσική γλώσσα η οποία αντιστοιχεί στο σύμβολο της ατομικής πρότασης

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    android:scrollbars="vertical"
    app:layout_behavior="android.support.design.widget.AppBarLayout$ScrollingVie..."
    tools:context=".ui.fragments.dialog_fragments.EditRuleDialog">

    <android.support.v7.widget.Toolbar
        android:id="@+id/toolbar"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:background="@color/colorPrimary"
        android:minHeight="?attr/actionBarSize"
        app:title="Add a Fact"
        android:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar"
        app:navigationIcon="@drawable/ic_close_white_24dp"
        app:popupTheme="@style/ThemeOverlay.AppCompat.Light"
        app:titleTextColor="@color/colorTextPrimary" />

    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:padding="16dp">

        <EditText
            android:id="@+id/edit_text_fact_symbol"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:background="@android:color/transparent"
            android:hint="Symbol"
            android:inputType="textMultiLine"
            android:padding="14dp"
            android:scrollbars="vertical"
            android:textSize="22sp" />
        <View
            android:layout_width="match_parent"
            android:layout_height="1dp"
            android:background="#a4a4a4" />

        <EditText
            android:id="@+id/edit_text_fact_proposition"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:background="@android:color/transparent"
            android:hint="Proposition"
            android:inputType="textMultiLine"
            android:padding="14dp"
            android:scrollbars="vertical"
            android:textSize="22sp" />

        <View
            android:layout_width="match_parent"
            android:layout_height="1dp"
            android:background="#a4a4a4" />
    </LinearLayout>
```



Εικόνα 31: Η διεπαφη του edit fact fragment

4.8.6 Το αρχείο EditFactDialog.java

Η κλάση EditFactDialog επεκτείνει την κλάση DialogFragment. Περιέχει τις εξής μεθόδους. Στην μέθοδο onCreate() ορίζεται το οπτικό στυλ του παραθύρου το οποίο είναι σχεδιασμένο στο αρχείο styles.xml.

Στην μέθοδο onStart() ορίζεται το μέγεθος του παραθύρου, και το animation το οποίο θα γίνεται η μετάβαση από το FactFragment στο EditFactDialog.

Στην μέθοδο onCreateView() γίνεται αρχικοποίηση όλων των views της διεπαφής του EditFactDialog.

Στην μέθοδο onViewCreated() ορίζεται ο τίτλος και το μενού επιλογών του ToolBar. Όταν γίνεται προσθήκη ενός νέου γεγονότος το toolbar έχει τον τίτλο “Add a Fact” και το μενού έχει την επιλογή SAVE, δηλαδή την αποθήκευση του γεγονότος στην βάση δεδομένων facts.db. Όταν γίνεται τροποποίηση ενός κανόνα τότε το γεγονότος έχει τον τίτλο “Edit a Fact” και το μενού εκτός από την επιλογή SAVE διαθέτει και την επιλογή Delete, δηλαδή την διαγραφή του γεγονότος. Όταν ο χρήστης πιέσει την επιλογή SAVE τότε γίνεται έλεγχος εάν το γεγονός είναι ήδη αποθηκευμένο στον πίνακα facts. Εάν το γεγονός δεν είναι αποθηκευμένο τότε καλείται η μέθοδος insertFact() ενώ αντίθετα καλεί την μέθοδο updateFact()

Η μέθοδος insertFact() αποθηκεύει το γεγονός εκφρασμένο σε προτασιακή λογική και σε φυσική γλώσσα στον πίνακα facts της βάσης δεδομένων facts.db.

Η μέθοδος updateRule() αποθηκεύει ένα τροποποιημένο γεγονός στον πίνακα facts. Η αντιστοιχία του συμβόλου μιας ατομικής πρότασης με μια πρόταση σε φυσική γλώσσα δεν τροποποιείται επειδή μπορεί να χρησιμοποιηθεί από κάποιους κανόνες . Οπότε όταν γίνεται τροποποίηση ενός γεγονότος τότε γίνεται αλλαγή τόσο στο σύμβολο όσο και στην πρόταση

Η μέθοδος deleteRule() αρχικά διαγράφει ένα γεγονός από τον πίνακα facts

Στη μέθοδο searchProposition() γίνεται αναζήτηση της πρότασης εκφρασμένης σε φυσική γλώσσα που αντιστοιχεί στο σύμβολο της ατομικής πρότασης στο πίνακα symbols.

Η μέθοδος reloadData() καλείται όταν ένα γεγονός υπάρχει ήδη στην βάση δεδομένων και ανασύρει τα δεδομένα του κανόνα από τον πίνακα facts. Τα δεδομένα αυτά συμπληρώνονται στα απαραίτητα views και ο χρήστης μπορεί να τα τροποποιήσει. Ακολουθεί ο κώδικας του αρχείου EditFactDialog.java

```
1   package com.drid.group_reasoning.ui.fragments.dialog_fragments;
2
3   import ...
32
33
34  public class EditFactDialog extends DialogFragment {
35
36      public static final String TAG = EditFactDialog.class.getSimpleName();
37
38      private Toolbar toolbar;
39
40      private EditText symbolEdit;
41      private EditText propositionEdit;
42
43
44      private Bundle arguments;
45      private Uri currentUri;
46
47      private boolean factIsChanged = false;
48
49      private View.OnTouchListener onTouchListener = (v, event) → {
52              factIsChanged = true;
53              return false;
54      };
56
57      @Override
58      public void onCreate(@Nullable Bundle savedInstanceState) {
59          super.onCreate(savedInstanceState);
60          setStyle(DialogFragment.STYLE_NORMAL, R.style.FullscreenDialog);
61      }
62
63      @Override
64      public void onStart() {
```

```
        super.onStart();
        Dialog dialog = getDialog();
        if (dialog != null) {
            int width = ViewGroup.LayoutParams.MATCH_PARENT;
            int height = ViewGroup.LayoutParams.MATCH_PARENT;
            dialog.getWindow().setLayout(width, height);
            dialog.getWindow().setWindowAnimations(R.style.AppTheme_Slide);
        }
    }


    @Nullable
    @Override
    public View onCreateView(
            @NonNull LayoutInflater inflater,
            @Nullable ViewGroup container,
            @Nullable Bundle savedInstanceState) {
        View view = inflater.inflate(R.layout.fragment_editor_fact, container,  attachToRoot: false);

        toolbar = view.findViewById(R.id.toolbar);

        symbolEdit = view.findViewById(R.id.edit_text_fact_symbol);
        propositionEdit = view.findViewById(R.id.edit_text_fact_proposition);

        symbolEdit.setOnTouchListener(onTouchListener);
        propositionEdit.setOnTouchListener(onTouchListener);
        return view;
    }

    @Override
    public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {
        super.onViewCreated(view, savedInstanceState);


        arguments = this.getArguments();

        if (arguments != null) {
            currentUri = Uri.parse(arguments.getString(ARG_FACT_URI));
        }


        if (currentUri == null) {
            toolbar.setTitle("Add a Fact");
        } else {
            toolbar.setTitle("Edit a Fact");
            reloadData();
        }

        toolbar.setNavigationOnClickListener((v) -> {
                if (!factIsChanged) {
                    dismiss();
                } else {
                    displayDiscardAlertDialog();
                }
        });
```

```
        });

        toolbar.inflateMenu(R.menu.editor_options_menu);

        if (currentUri == null) {
            MenuItem deleteMenuItem = toolbar.getMenu().findItem(R.id.delete);
            deleteMenuItem.setVisible(false);
        }

        toolbar.setOnMenuItemClickListener((menuItem) -> {
                switch (menuItem.getItemId()) {
                    case R.id.save:
                        saveFact();
                        dismiss();
                        return true;
                    case R.id.delete:
                        displayDeleteAlertDialog();
                        return true;
                }
                return false;
        });


        symbolEdit.addTextChangedListener(new TextWatcher() {
            @Override
            public void beforeTextChanged(CharSequence s, int start, int count, int after) { }

            @Override
            public void onTextChanged(CharSequence s, int start, int before, int count) {

            }

            @Override
            public void afterTextChanged(Editable s) {
                String symbol = symbolEdit.getText().toString();
                String proposition = searchProposition(symbol);
                if (!TextUtils.isEmpty(symbol) && !TextUtils.isEmpty(proposition)) {
                    propositionEdit.setText(proposition);
                } else {
                    propositionEdit.setText("");
                }
            }
        });
    }

    private void saveFact() {

        String factVariable = symbolEdit.getText().toString().trim();
        String factProposition = propositionEdit.getText().toString();


        if (currentUri == null) {
            insertFact(factVariable, factProposition);

        } else {
            updateFact(factVariable, factProposition);
        }

```

```
184
185     private void insertFact(String factVariable, String factProposition) {
186         ContentValues facts = new ContentValues();
187
188         facts.put(FactEntry.COLUMN_FACT_PROPOSITION, factVariable);
189         facts.put(FactEntry.COLUMN_FACT_SYMBOL, factProposition);
190
191         getActivity().getContentResolver().insert(FactEntry.FACT_URI, facts);
192     }
193
194     private void updateFact(String factPl, String factNl) {
195         ContentValues facts = new ContentValues();
196
197         facts.put(FactEntry.COLUMN_FACT_PROPOSITION, factPl);
198         facts.put(FactEntry.COLUMN_FACT_SYMBOL, factNl);
199
200         getActivity().getContentResolver().update(
201                 this.currentUri, facts,  where: null,  selectionArgs: null);
202     }
203
204     private void deleteFact() {
205         getActivity().getContentResolver().delete(currentUri,
206                 where: null,
207                 selectionArgs: null);
208     }
209
210     private void reloadData() {
210 private void reloadData() {
211         long id = ContentUris.parseId(currentUri);
212
213         Cursor factCursor = getActivity().getContentResolver().query(
214                 currentUri,
215                 new String[]{FactEntry.COLUMN_FACT_PROPOSITION, FactEntry.COLUMN_FACT_SYMBOL},
216                  selection: FactEntry._ID + "=?",
217                 new String[]{String.valueOf(id)},
218                  sortOrder: null
219         );
220
221         if (factCursor.moveToFirst()) {
222             symbolEdit.setText(
223                     factCursor.getString(factCursor.getColumnIndex(FactEntry.COLUMN_FACT_PROPOSITION)));
224             propositionEdit.setText(
225                     factCursor.getString(factCursor.getColumnIndex(FactEntry.COLUMN_FACT_SYMBOL)));
226         }
227
228         factCursor.close();
229
230     }
231
232     private String searchProposition(String symbol) {
233
234         String result = null;
235         Uri queryVarUri = RuleContract.SymbolEntry.SYMBOLS_URI.buildUpon().appendPath(PATH_SYMBOL).build();
236
237         Cursor symbolCursor = getActivity().getContentResolver().query(
238                 queryVarUri,
```

```
                projection: null,
                selection: RuleContract.SymbolEntry.COLUMN_SYMBOL + "=?",
                new String[]{symbol},

                sortOrder: null
        );

        if (symbolCursor.moveToFirst()) {
            result = symbolCursor.getString(symbolCursor.getColumnIndex(RuleContract.SymbolEntry.COLUMN_PROPOSITION));
        }

        return result;
    }


    private void displayDiscardAlertDialog() {
        AlertDialog dialog = new AlertDialog.Builder(
                new ContextThemeWrapper(getContext(), R.style.AlertDialog))
                .setTitle("Do you want to discard changes ?")
                .setPositiveButton( text: "Keep Editing", (dialog, which) -> {
                        dialog.dismiss();
                }).setNegativeButton( text: "Discard", (dialog, which) -> {
                        dismiss();
                }).create();

        dialog.show();
        dialog.setCancelable(false);
        dialog.setCanceledOnTouchOutside(false);
    }

    private void displayDeleteAlertDialog() {
        AlertDialog dialog = new AlertDialog.Builder(
                new ContextThemeWrapper(getContext(), R.style.AlertDialog))
                .setTitle("Are you sure you want to delete this fact ?")
                .setPositiveButton( text: "Yes", (dialog, which) -> {
                        deleteFact();
                        dismiss();
                }).setNegativeButton( text: "No", (dialog, which) -> {
                        dialog.dismiss();
                }).create();

        dialog.show();
        dialog.setCancelable(false);
        dialog.setCanceledOnTouchOutside(false);

    }

}
```

4.9 To LogicSolverFragment

Το Logic solver fragment αποτελεί την διεπαφή όπου γίνεται η επίλυση των προβλημάτων τα οποία είναι διατυπωμένα σε κανόνες. Όταν ο χρήστης πιέσει το κουμπί solve τότε εμφανίζονται τα αποτελέσματα της διαδικασίας επίλυσης των κανόνων σε ένα TextView. Αρχικά γίνεται επίλυση των προβλημάτων βάση τη γνώση, δηλαδή τους κανόνες και τα γεγονότα που είναι αποθηκευμένα στην συσκευή. Η επίλυση πραγματοποιείται με την μέθοδο της προς τα εμπρός αλυσιδωτής εκτέλεσης κανόνων (Forward Chaining). Για όσους κανόνες δεν μπορούν να επιλυθούν, η συσκευή στέλνει ερωτήματα στις συνδεμένες συσκευές, ζητώντας τα γεγονότα που είναι απαραίτητα για την επίλυση αυτών των κανόνων. Όταν η συσκευή λάβει

απαντήσεις επαναλαμβάνει την διαδικασία επίλυσης των κανόνων που δεν έχουν επιλυθεί προσθέτοντας προσωρινά στην ήδη υπάρχουσα γνώση, τα γεγονότα που έχει λάβει από το δίκτυο.

4.9.1 αρχείο fragment_logic_solver.xml

Η διεπαφή του Logic Fragment είναι σχεδιασμένη στο αρχείο fragment_logic_solver.xml. Αποτελείται από ένα TextView στο οποίο εμφανίζονται μηνύματα που σχετίζονται με την επίλυση ενός προβλήματος καθώς και τα ερωτήματα κι οι απαντήσεις που στέλνονται στο δίκτυο. Επίσης διαθέτει και ένα Button με την ένδειξη “Solve”, το οποίο ξεκινά την διαδικασία επίλυσης των λογικών προβλημάτων.Ακολουθεί ο κώδικας του αρχείου fragment_logic_solver.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="2dp"
    app:layout_behavior="android.support.design.widget.AppBarLayout$ScrollingVie..."
    tools:context=".ui.fragments.LogicSolverFragment">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_marginLeft="25dp"
        android:layout_marginTop="12dp"
        android:layout_marginRight="25dp"
        android:layout_marginBottom="12dp"
        android:orientation="vertical" >


        <TextView
            android:id="@+id/log"
            android:layout_width="match_parent"
            android:layout_height="0dp"
            android:layout_weight="1"
            android:background="@drawable/box"
            android:padding="12dp"
            android:scrollbars="vertical"/>

         <android.support.design.button.MaterialButton
             android:id="@+id/solve"
             android:layout_width="match_parent"
             android:layout_height="wrap_content"
             android:layout_marginTop="8dp"
             android:text="solve" />
    </LinearLayout>

</RelativeLayout>
```

Εικόνα 32: Η διεπαφή του logicSolverFragment

4.9.2 Το αρχείο LogicSolverFragment.java

Η κλάση LogicSolverFragment επεκτείνει την κλάση Fragment και υλοποιεί την λογική της διεπαφής του Logic Solver Fragment Ακολουθεί ο κώδικας της κλάσης LogicSolverFragment

```
1   package com.drid.group_reasoning.ui.fragments;
2
3   import ...
30
31  public class LogicSolverFragment extends Fragment {
32
33      public static final String TAG = LogicSolverFragment.class.getSimpleName();
34
35      private OnFragmentInteractionListener callback;
36
37      private TextView log;
38      private Button resolve;
39
40      private InferenceEngine engine;
41
42      private BroadcastReceiver receiver = (context, intent) → {
45              String action = intent.getAction();
46
47              assert action != null;
48              switch (action) {
49                  case "new_facts":
50                      List<AtomicSentence> facts = intent.getParcelableArrayListExtra( name: "facts");
```

```
51
52                  if (!facts.isEmpty()) {
53                      log.append("\nNew facts arrived\n");
54                      for (AtomicSentence fact : facts) {
55                          log.append("-" + fact.getProposition() + "\n");
56                      }
57                      addNewFacts(facts);
58                  } else {
59                      log.append("\nNo facts found in peers ");
60                  }
61                  break;
62              case "no_connected_peers":
63                  Log.i(TAG,  msg: "onReceive: No connected peers");
64                  log.append("\nNo connected peers");
65                  break;
66              case "connected_peers":
67                  Log.i(TAG,  msg: "onReceive: Connected peers");
68
69                  List<String> connectedPeers = intent.getStringArrayListExtra( name: "connected_peers_list");
70                  log.append("\nConnected peers: " + connectedPeers);
71                  break;
72              case "log_message":
73                  String log_message = intent.getStringExtra( name: "message");
74                  log.append(log_message + "\n");
75          }
```

```
76
77      };
79
80
81      @Override
82      public void onCreate(Bundle savedInstanceState) {
83          super.onCreate(savedInstanceState);
84          setHasOptionsMenu(true);
85      }
86
87
88      @Override
89  @   public View onCreateView(LayoutInflater inflater, ViewGroup container,
90                               Bundle savedInstanceState) {
91
92
93          View view = inflater.inflate(R.layout.fragment_logic_solver, container,  attachToRoot: false);
94
95
96          log = view.findViewById(R.id.log);
97          log.setMovementMethod(new ScrollingMovementMethod());
98
99          resolve = view.findViewById(R.id.solve);
100
101         return view;
102     }
```

```
    }

    @Override
    public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {
        super.onViewCreated(view, savedInstanceState);

        resolve.setOnClickListener((v) -> {
                log.setText("");
                new Handler().postDelayed(() -> { initEngine(); }, delayMillis: 200);
        });

    }

    @Override
    public void onResume() {
        super.onResume();

        IntentFilter intentFilter = new IntentFilter();
        intentFilter.addAction("new_facts");
        intentFilter.addAction("no_peer_left_ask");
        intentFilter.addAction("connected_peers");
        intentFilter.addAction("no_connected_peers");
        intentFilter.addAction("new_query");
        intentFilter.addAction("log_message");

        getActivity().registerReceiver(this.receiver, intentFilter);
    }

    @Override
    public void onDestroy() {
        super.onDestroy();
        getActivity().unregisterReceiver(receiver);
    }

    private void initEngine() {

        engine = new InferenceEngine(getContext());

        engine.initKnowledgeBase();
        engine.solve();

        log.append("Resolved queries:\n");
        for (AtomicSentence query : engine.getResolvedQueries()) {
            log.append(" - " + query.getProposition() + "\n");
            Log.i(TAG, msg: " " + query.getProposition());
            insertFact(query);
        }

        log.append("\nUnresolved queries:\n");
        for (AtomicSentence query : engine.getUnresolvedQueries()) {
            log.append(" - " + query.getProposition() + "\n");
        }
        log.append("\nMissing facts :\n");

        List<AtomicSentence> missingFacts = new ArrayList<>(engine.getMissingFacts());
        if (!missingFacts.isEmpty()) {

            for (AtomicSentence query : missingFacts) {
                log.append(" - " + query.getProposition() + "\n");
```

```
169             }
170             log.append("\nAsking peers for these facts:\n");
171             for (AtomicSentence a : missingFacts) {
172                 log.append("- " + a.getProposition() + "\n");
173             }
174             askPeers(missingFacts);
175         }
176
177     }
178
179     private void askPeers(List<AtomicSentence> missingFacts) {
180         this.callback.askPeers(missingFacts);
181     }
182
183 @   private void addNewFacts(List<AtomicSentence> facts) {
184
185         Log.i(TAG,  msg: "addNewFacts: Trying to solve again");
186
187
188         for (Iterator<AtomicSentence> iterator = facts.iterator(); iterator.hasNext(); ) {
189             AtomicSentence value = iterator.next();
190             if (engine.getKnowledgeBase().getSentences().contains(value)) {
191                 Log.i(TAG,  msg: "addNewFacts: " + value + " already exists");
192                 iterator.remove();
193             }
194         }
195
196         if (!facts.isEmpty()) {
```

```
197
198             engine.addNewFactsToKb(facts);
199             engine.solve();
200             List<AtomicSentence> resolvedQueries = engine.getResolvedQueries();
201             List<AtomicSentence> unResolvedQueries = engine.getUnresolvedQueries();
202             Set<AtomicSentence> missingFacts = engine.getMissingFacts();
203
204             Log.i(TAG,  msg: "addNewFacts: Resolving with new facts");
205             Log.i(TAG,  msg: "addNewFacts: Resolved - " + resolvedQueries);
206             log.append("\nResolving with new facts\n");
207             log.append("\nResolved queries:\n");
208
209             for (AtomicSentence query : resolvedQueries) {
210                 log.append("-" + query.getProposition() + "\n");
211                 insertFact(query);
212             }
213
214             log.append("\nUnresolved queries:\n ");
215             Log.i(TAG,  msg: "addNewFacts: Unresolved queries - " + unResolvedQueries);
216             for (AtomicSentence query : unResolvedQueries) {
217                 log.append("-" + query.getProposition() + "\n");
218             }
219             log.append("\nMissing facts:\n ");
220
221             if (!missingFacts.isEmpty()) {
222                 Log.i(TAG,  msg: "addNewFacts: missing facts - " + missingFacts);
223                 for (AtomicSentence query : missingFacts) {
224                     log.append("-" + query.getProposition() + "\n");
```

```
225                     }
226                 }
227             }
228         }
229
230 @       private void insertFact(AtomicSentence fact) {
231
232             ContentValues values = new ContentValues();
233             values.put(FactEntry.COLUMN_FACT_PROPOSITION, fact.getSymbol());
234             values.put(FactEntry.COLUMN_FACT_SYMBOL, fact.getProposition());
235             getActivity().getContentResolver().insert(FactEntry.FACT_URI, values);
236         }
237
238         public void setOnFragmentCreatedListener(OnFragmentInteractionListener callback) {
239             this.callback = callback;
240         }
241
242         public interface OnFragmentInteractionListener {
243             void askPeers(List<AtomicSentence> missingFacts);
244         }
245     }
246
```

5. Συμπεράσματα

Η επεξεργαστική των συσκευών Android σε συνδυασμό με τις τεχνολογίες ραδιοκυμάτων καθιστά την επίλυση λογικών προβλημάτων στα πλαίσια ενός συστήματος πολλαπλών πρακτόρων γρήγορη και αποτελεσματική. Παρόλα αυτά η αναπαράσταση της γνώσης σε προτασιακή λογική δεν είναι ιδανική διότι η προτασιακή γλώσσα δεν είναι αρκετά περιγραφική και είναι επιρρεπείς σε παρερμηνείες.

Για αυτό τον λόγο προτείνεται η χρήση της λογικής πρώτης τάξης για την αναπαράσταση της γνώσης ή χρήση βιβλιοθηκών όπως Drools κτλ. Επίσης το σύστημα μπορεί να υλοποιηθεί σε γλώσσα Kotlin η οποία παρέχει περισσότερες δυνατότητες σε σχέση με την γλώσσα Java.

Βιβλιογραφία

[1] "Android", Wikipedia, 2019 https://en.wikipedia.org/wiki/Android_(operating_system)

[2] "Google", Wikipedia, 2019 https://en.wikipedia.org/wiki/Google

[3] "Linux Kernel", Wikipedia, 2019 https://en.wikipedia.org/wiki/Linux_kernel

[4] "Announcing the Android 1.0 SDK, release 1", Android Developers Blog, 2008, https://android-developers.googleblog.com/2008/09/announcing-android-10-sdk-release-1.html

[5] "Google Mobile Services", Wikipedia, 2019 https://en.wikipedia.org/wiki/Google_mobile_services

[6] "Android Cupcake", Wikipedia, 2019 https://en.wikipedia.org/wiki/Android_Cupcake

[7] "Android Donut", Wikipedia, 2019 https://en.wikipedia.org/wiki/Android_Donut

[8] "Android Éclair", Wikipedia, 2019 https://en.wikipedia.org/wiki/Android_Eclair

[9] "Tethering", Wikipedia 2019 https://en.wikipedia.org/wiki/Tethering

[10] "Android Froyo", Wikipedia 2019 https://en.wikipedia.org/wiki/Android_Froyo

[11] "Android Gingerbread", Wikipedia 2019 https://en.wikipedia.org/wiki/Android_Gingerbread

[12] "Android Honeycomb", Wikipedia 2019 https://en.wikipedia.org/wiki/Android_Honeycomb

[13] "Android Ice Cream Sandwich", Wikipedia 2019 https://en.wikipedia.org/wiki/Android_Ice_Cream_Sandwich

[14] "Android Jelly Bean", Wikipedia 2019 https://en.wikipedia.org/wiki/Android_Jelly_Bean

[15] "Android KitKat", Wikipedia 2019 https://en.wikipedia.org/wiki/Android_KitKat#cite_note-1

[16] "Distribution dashboard", Android Developers 2019 https://developer.android.com/about/dashboards/index.html

[17] "Android Lollipop", Wikipedia 2019 https://en.wikipedia.org/wiki/Android_Lollipop

[18] "Android Marshmallow", Wikipedia 2019 https://en.wikipedia.org/wiki/Android_Marshmallow

[19] "Android Nougat", Wikipedia 2019 https://en.wikipedia.org/wiki/Android_Nougat

[20] "Android Oreo", Wikipedia 2019 https://en.wikipedia.org/wiki/Android_Oreo

[21] "Android Pie", Wikipedia 2019 https://en.wikipedia.org/wiki/Android_Pie

[22] "Android 10", Wikipedia 2019 https://en.wikipedia.org/wiki/Android_10

[23] "Android version history", Wikipedia 2019
https://en.wikipedia.org/wiki/Android_version_history

[24] "List of features in Android", Wikipedia 2019
https://en.wikipedia.org/wiki/List_of_features_in_Android

[25] "Platform Architecture", Android Developers 2019

https://developer.android.com/guide/platform

[26] "Android Fundamentals", Android Developers 2019
https://developer.android.com/guide/components/fundamentals

[27] "Principle of least privilege", Wikipedia 2019
https://en.wikipedia.org/wiki/Principle_of_least_privilege

[28] "Documentation - Activity", Android Developers 2019
https://developer.android.com/reference/android/app/Activity.html

[29] "Documentation - AsyncTask", Android Developers 2019
https://developer.android.com/reference/android/os/AsyncTask

[30] "Documentation - Service", Android Developers 2019
https://developer.android.com/reference/android/app/Service.html

[31] "Documentation - BroadcastReceiver", Android Developers 2019
https://developer.android.com/reference/android/content/BroadcastReceiver.html

[32] "Documentation – Intent", Android Developers 2019
https://developer.android.com/reference/android/content/Intent.html

[33] "Documentation – ContentProvider", Android Developers 2019
https://developer.android.com/reference/android/content/ContentProvider.html

[33] "Android Fundamentals, The manifest file", Android Developers 2019
https://developer.android.com/guide/components/fundamentals#Manifest

[34] "Garbage collection (computer science)", Wikipedia 2019
https://en.wikipedia.org/wiki/Garbage_collection_(computer_science)

[35] "The Java Language" Sams Teach Yourself Java in 21 Days (Covering Java 8), Seventh Edition https://www.oreilly.com/library/view/sams-teach-yourself/9780133755848/

[36] "Extensible Markup Language (XML), Introduction", W3C Information and Knowledge Domain 2019 https://www.w3.org/XML/

[37] "Standard Generalized Markup Language", Wikipedia 2019
https://en.wikipedia.org/wiki/Standard_Generalized_Markup_Language

[38] “Unicode”, Wikipedia 2019 https://en.wikipedia.org/wiki/Unicode

[39] “XML”, Wikipedia 2019 https://el.wikipedia.org/wiki/XML

[40] “HTML”, Wikipedia 2019 https://en.wikipedia.org/wiki/HTML

[41] “Android Studio”, Wikipedia 2019 https://en.wikipedia.org/wiki/Android_Studio

[42] “Meet Android Studio, The user interface”, Android Developers 2019 https://developer.android.com/studio/intro/

[43] “Meet Android Studio, Project structure, Figure 1”, Android Developers 2019 https://developer.android.com/studio/intro#project_structure

[44] “Meet Android Studio, The user interface, Figure 3”, Android Developers 2019 https://developer.android.com/studio/intro/

[45] “Understanding the activity lifecycle, Activity-lifecycle concepts, Figure 1” https://developer.android.com/guide/components/activities/activity-lifecycle#alc

[46] “Artificial Intelligence A Modern Approach, Stuart J. Russell and Peter Norvig, Figure 7.7, Page 244”, Internet archive 2019 https://archive.org/details/yvrcuddapah_gmail_AIMA/page/n263

[47] “Rule-based system”, Wikipedia 2019, https://en.wikipedia.org/wiki/Rule-based_system

[48] “Κεφάλαιο 3 – Συστήματα κανόνων”, Τεχνητή νοημοσύνη: Μια εισαγωγική προσέγγισή, Κατερίνα Γεωργούλη http://repfiles.kallipos.gr/html_books/93/03a-main.html#_idTextAnchor051

[49] “Κεφάλαιο 3 – Συστήματα κανόνων, Ενότητα 3.4.3 – Προς τα εμπρός αλυσιδωτή εκτέλεση κανόνων”, Τεχνητή νοημοσύνη: Μια εισαγωγική προσέγγισή, Κατερίνα Γεωργούλη http://repfiles.kallipos.gr/html_books/93/03a-main.html#_idTextAnchor051

[50] “Announcing Nearby Connections 2.0: fully offline, high bandwidth peer to peer device communication”, Android Developers 2019 https://android-developers.googleblog.com/2017/07/announcing-nearby-connections-20-fully.html

[51] “Artificial Intelligence A Modern Approach, Stuart J. Russell and Peter Norvig, Figure 7.15, Page 258”, Internet archive 2019 https://archive.org/details/yvrcuddapah_gmail_AIMA/page/n277

[52] “Java implementation of algorithms from Russell And Norvig's "Artificial Intelligence - A Modern Approach”, Github 2019 https://github.com/aimacode/aima-java#aima3e-java-jdk-8--