

<SIP++>

Έντυπο Σχεδίασης

MTVTEAM

Θάλεια-Δήμητρα Δούδαλη

Μαρία Καρούτα

Βασιλική Σάββα

Έκδοση:	1.0
Ημ/νια Εκτύπωσης:	5/20/2015 10:44:00 AM
Ημ/νια Έκδοσης:	
Κατάσταση Έκδοσης:	Βασική
Κατάσταση Έγκρισης:	Τελική
Έγκριθηκε από:	
Ετοιμάστηκε από:	MTV TEAM
Επιθεωρήθηκε από:	
Όνομα Αρχείου:	SIP++ SDD.pdf
Αριθμός Εντύπου:	1

Περιεχόμενα

1	ΕΙΣΑΓΩΓΗ.....	3
1.1	Σκοπός.....	3
1.2	Περίληψη.....	3
2	ΣΗΜΑΝΤΙΚΕΣ ΣΧΕΔΙΑΣΤΙΚΕΣ ΑΠΟΦΑΣΕΙΣ.....	4
3	ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΣΥΣΤΗΜΑΤΟΣ.....	5
3.1	Component Diagram.....	5
3.2	Deployment Diagram.....	6
3.3	Database Schema.....	7
4	ΑΝΑΛΥΤΙΚΑ ΔΙΑΓΡΑΜΜΑΤΑ ΚΛΑΣΕΩΝ.....	8
4.1	Client (SIP Communicator).....	8
4.1.1	net.java.sip.communicator.....	8
4.1.2	net.java.sip.communicator.gui.....	9
4.1.3	net.java.sip.communicator.sip.....	11
4.2	Method Details.....	12
4.2.1	govnist.sip.proxy.....	12
4.2.2	govnist.sip.proxy.gui.....	13
4.2.3	govnist.sip.proxy.registrar.....	13
5	STATE DIAGRAMS.....	14
6	ΑΝΟΙΧΤΑ ΖΗΤΗΜΑΤΑ.....	15
6.1	Κλήση όπου ο caller και ο callee είναι το ίδιο πρόσωπο.....	15
6.2	Επικύρωση δεδομένων.....	15
6.3	Κρυπτογραφημένη επικοινωνία πελάτη - διακομιστή.....	15
6.4	Εφαρμογή πιστωτικού συστήματος.....	15
6.5	Επέκταση του SIP-Proxy GUI.....	15
6.6	Πολλαπλές Προωθήσεις.....	15

1 Εισαγωγή

1.1 Σκοπός

Σε αυτή την άσκηση εργαστηρίου ο σκοπός είναι να σχεδιάσουμε και να υλοποιήσουμε τρεις νέες λειτουργίες στο πρόγραμμα πελάτη SIP Communicator και στο πρόγραμμα εξυπηρέτησης JAIN SIP Proxy. Στο παρόν έγγραφο περιλαμβάνονται διαγράμματα και προδιαγραφές σχετικές με την αρχιτεκτονική σχεδίαση του συστήματος και τη λεπτομερή σχεδίαση των προγραμμάτων επέκτασης που θα υλοποιήσουμε.

1.2 Περίληψη

Στο project αυτό επεκτείνουμε τις λειτουργίες μιας εφαρμογής διαδικτυακής τηλεφωνίας (VoIP) με δυνατότητες για user registration, billing, call forwarding και call blocking. Συγκεκριμένα, θα προσθέσουμε στα χαρακτηριστικά του VoIP software SIP-Communicator and JAIN SIP Proxy τα εξής:

- **Προώθηση κλήσεων - Call forwarding**

Ο χρήστης θα έχει τη δυνατότητα να προωθεί εισερχόμενες κλήσεις σε έναν άλλο χρήστη (client) του συστήματος.

- **Μπλοκάρισμα κλήσεων - Call blocking**

Οι χρήστες (clients) θα έχουν τη δυνατότητα να μπλοκάρουν κλήσεις από επιλεγμένους χρήστες. Επιπλέον, οι μπλοκαρισμένοι χρήστες δεν θα γνωρίζουν ότι ο καλών χρήστης τους έχει μπλοκάρει, αλλά θα τον βλέπουν ως μη διαθέσιμο.

- **Χρέωση - Billing**

Όταν μία κλήση τερματίσει, ο Proxy Server υπολογίζει το κόστος της κλήσης ανάλογα το τρέχον πρόγραμμα χρέωσης που αντιστοιχεί στον καλών χρήστη (caller) και ενημερώνει τον λογαριασμό του.

2 Σημαντικές Σχεδιαστικές Αποφάσεις

Προκειμένου να υλοποιήσουμε το σύστημα, πρέπει να πάρουμε κάποιες σχεδιαστικές αποφάσεις.

Οσον αφορά το σενάριο πρώτης εγγραφής του χρήστη στο σύστημα, η αποθήκευση των στοιχείων του γίνεται στη βάση δεδομένων. η οποία είναι προσβάσιμη από τον Proxy Server. Επίσης, η διαπροσωπεία που χρησιμοποιείται για την είσοδο εγγεγραμμένου χρήστη στο σύστημα (username, password) επιλέξαμε να είναι διαφορετική από εκείνη που χρησιμοποιείται για την πρώτη εγγραφή του.

Σχετικά με τον περιορισμό εισερχόμενων κλήσεων, όλες οι απαραίτητες πληροφορίες για το ποιος χρήστης μπλοκάρει ποιον αποθηκεύονται σε μια βάση δεδομένων η οποία είναι προσβάσιμη από τον Proxy Server.

Αντίστοιχα με τον περιορισμό κλήσεων, οι πληροφορίες σχετικά με την προώθηση κλήσεων (ποιος χρήστης προωθεί κλήση σε ποιον) αποθηκεύονται σε μια βάση δεδομένων που είναι προσβάσιμη από τον Proxy Server.

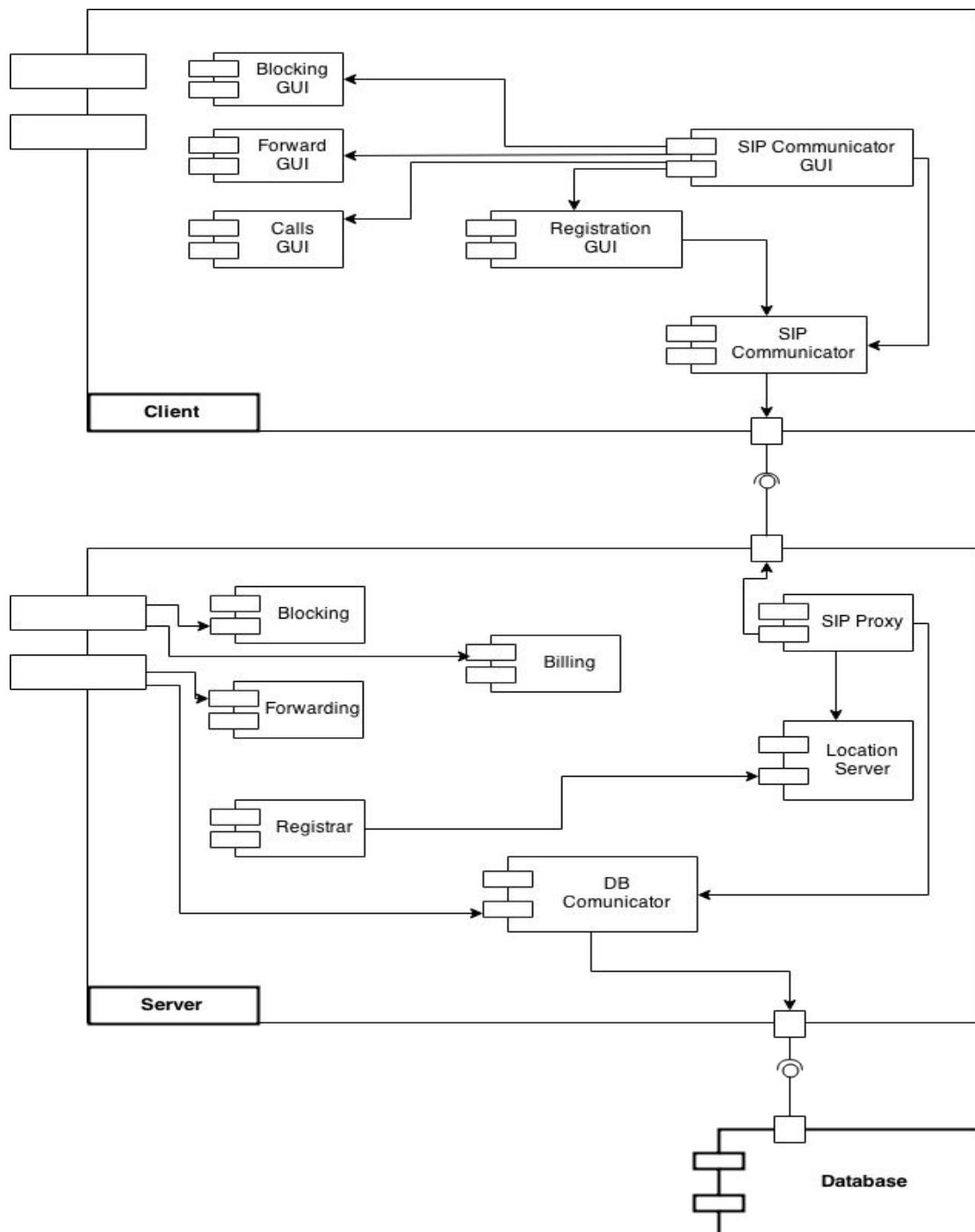
Η επίλυση των κυκλικών προωθήσεων γίνεται μόλις ο χρήστης καταχωρήσει στο σύστημα την επιλογή της προώθησης της κλήσης, και όχι όταν θα πραγματοποιήσει μια κλήση. Για να βελτιώσουμε την απόδοση της προώθησης κλήσεων επιλέξαμε να αποθηκεύουμε τον τελικό παραλήπτη μιας κλήσης στη βάση δεδομένων και όχι τον παραλήπτη του πρώτου βήματος, αποφεύγοντας έτσι την καθυστέρηση σε περίπτωση πολλαπλών προωθήσεων.

Τέλος, άλλες σχεδιαστικές αποφάσεις αφορούν τα ξεχωριστά παράθυρα διαλόγου που υλοποιούν συγκεκριμένες υπηρεσίες, όπως:

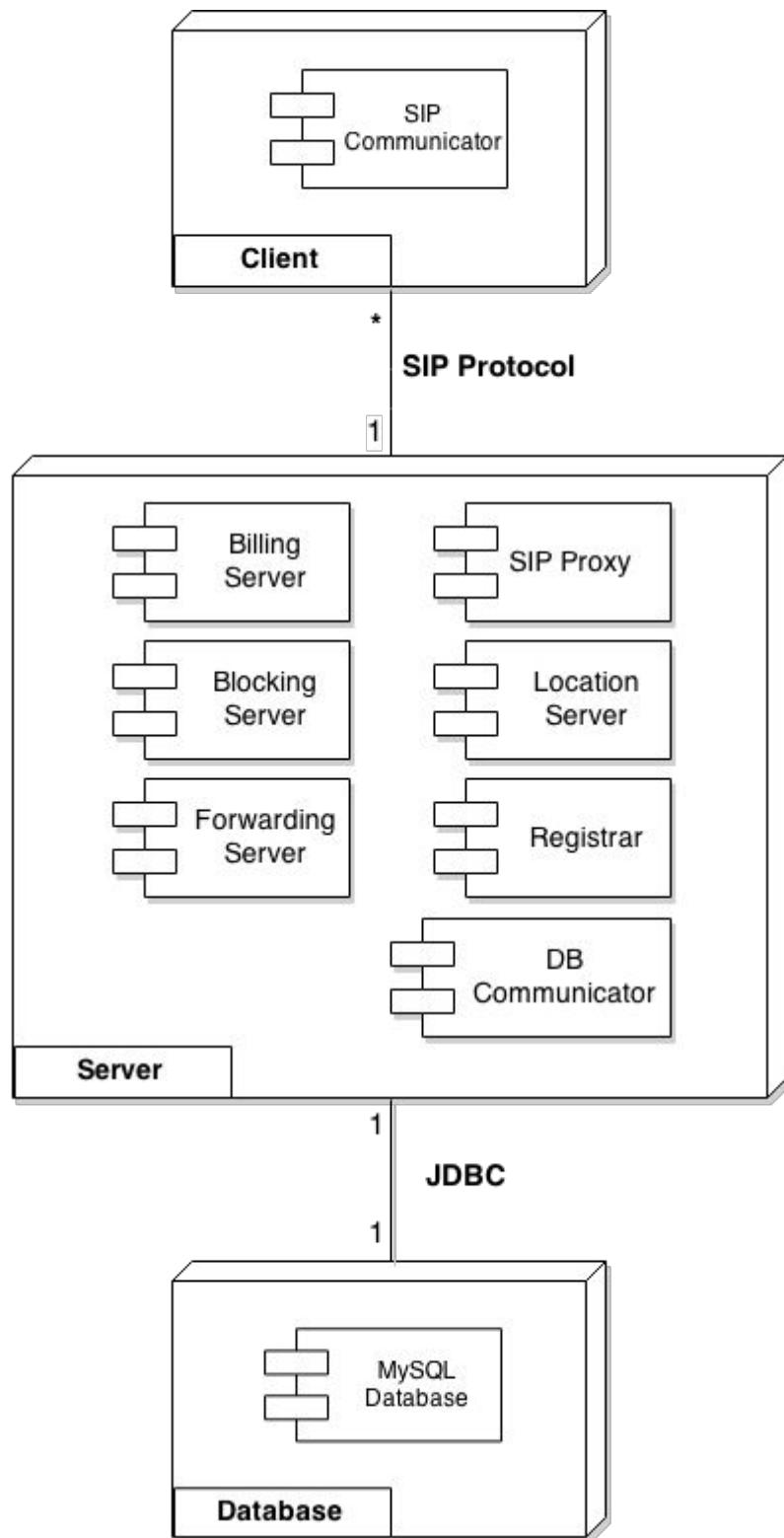
1. Registration, όταν ο χρήστης εγγράφεται στο σύστημα 1η φορά.
2. Blocking, όταν ο χρήστης θέλει να μπλοκάρει κλήσεις προερχόμενες από άλλον χρήστη.
3. Forwarding, όταν ο χρήστης επιθυμεί να προωθήσει τις εισερχόμενες κλήσεις του σε άλλον χρήστη.
4. Call's Information, όπου ο χρήστης μπορεί να δει τη διάρκεια μιας κλήσης, το κόστος της και εάν είναι πληρωμένη.

3 Αρχιτεκτονική Συστήματος

3.1 Component Diagram



3.2 Deployment Diagram



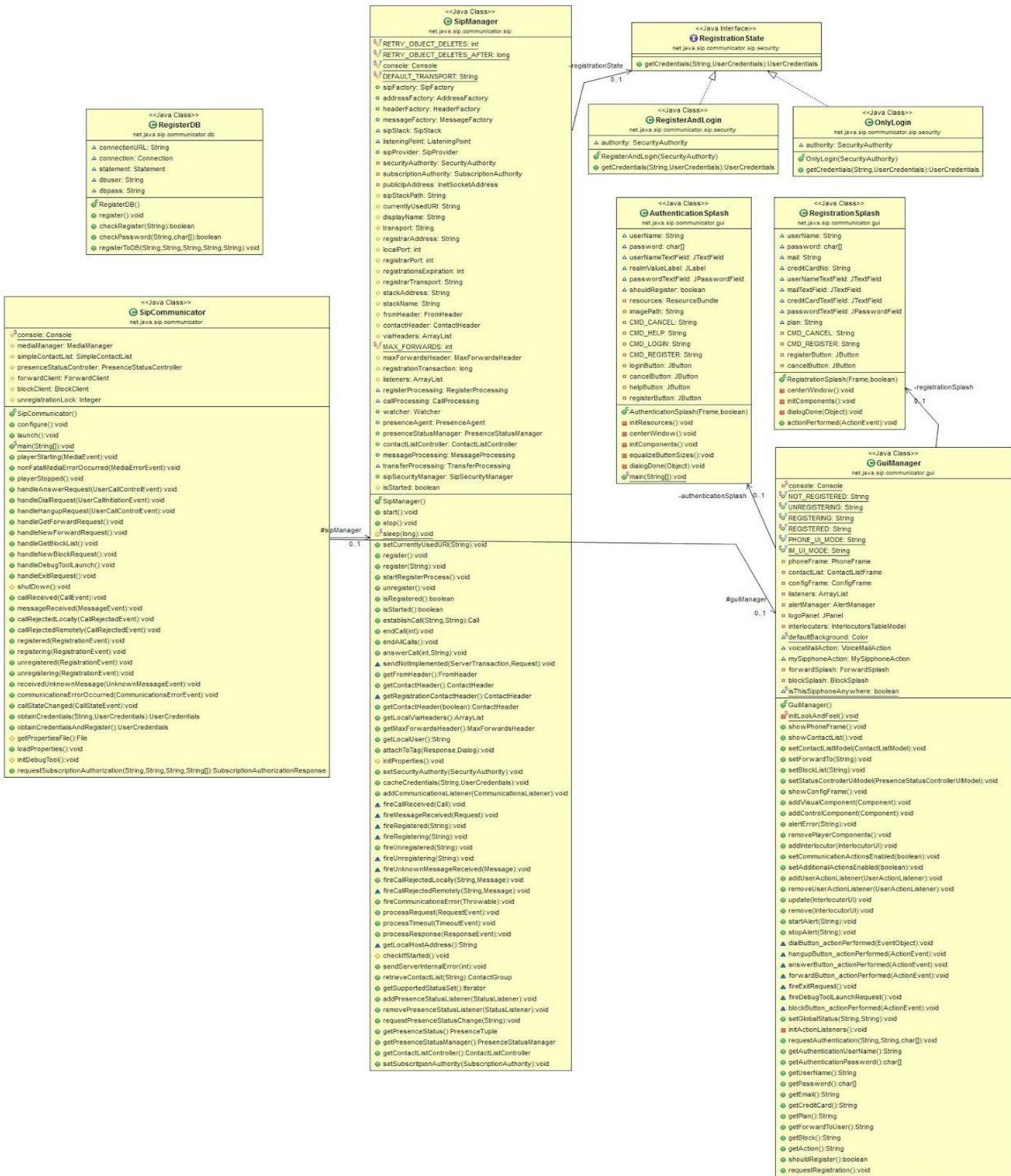
3.3 Database Schema



4 Αναλυτικά Διαγράμματα Κλάσεων

4.1 Client (Sip Communicator)

4.1.1 Registration



Κατά την εκκίνηση του Sip Communicator, τρέχει η συνάρτηση launch () της κλάσης *SipCommunicator* μέσα από τη main. Εκείνη καλεί τη συνάρτηση startRegisterProcess της κλάσης *sipManager*. Εδώ, αναλόγως με την μεταβλητή registrationState, θα κληθεί η κατάλληλη getCredentials. Χρησιμοποιείται λοιπόν εδώ το state design pattern, όπου η πρώτη πιθανή κατάσταση είναι να θέλει να κάνει register με νέο λογαριασμό και μαζί login ο χρήστης, ενώ η δεύτερη να θέλει να κάνει απλά login. Η διαπροσωπεία για αυτό το state design pattern είναι η *RegistrationState*, ενώ οι κλάσεις που την υλοποιούν είναι οι *RegisterAndLogin* και *OnlyLogin* αντίστοιχα.

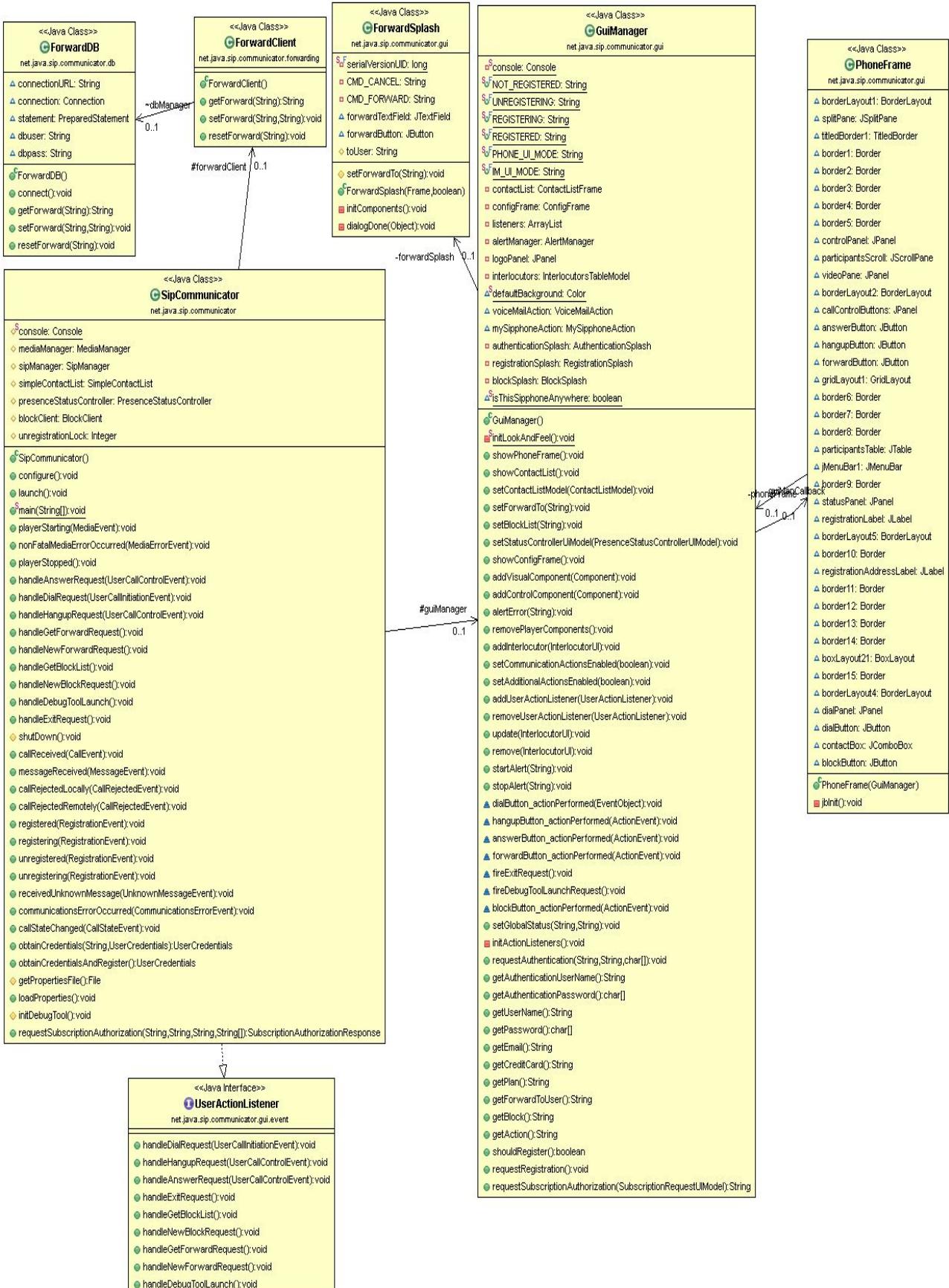
Η *RegisterAndLogin*, όταν κληθεί η getCredentials, καλεί με τη σειρά της τη συνάρτηση obtainCredentialsAndRegister πάνω στη μεταβλητή authority της κλάσης SecurityAuthority. Η *OnlyLogin*, όταν κληθεί η getCredentials, καλεί την obtainCredentials πάνω στη μεταβλητή authority της κλάσης SecurityAuthority. Και οι 2 συναρτήσεις αυτές υλοποιούνται από την κλάση SipCommunicator.

Η συνάρτηση obtainCredentialsAndRegister πραγματοποιεί το registration. Πρώτα, η requestRegistration του *GuiManager* παρουσιάζει το registration window στο χρήστη, το οποίο υλοποιείται από την κλάση RegistrationSplash. Κατόπιν, προσθέτει στη βάση RegisterDB τα δεδομένα του χρήστη, αφού πρώτα ελέγχει αν έχει ήδη πάρει άλλος το username που δήλωσε ο χρήστης, ειδοποιώντας τον σε αυτήν την περίπτωση.

Η obtainCredentials πραγματοποιεί το login του χρήστη, αφού πρώτα ελέγχει αν πάτησε το register button. Εμφανίζεται αρχικά το παράθυρο AuthenticationSplash. Αν ο χρήστης πατήσει το κουμπί register, τότε η συνάρτηση guiManager.shouldRegister επιστρέφει true, οπότε θα κληθεί τελικά η obtainCredentialsAndRegister.

Το τι ακριβώς θα συμβεί όμως από τα 2 σενάρια εξαρτάται από την τιμή της μεταβλητής registrationState. Κατά την εκκίνηση, στην κλήση της συνάρτησης launch του *SipCommunicator*, καλείται η συνάρτηση setSecurityAuthority της κλάσης *sipManager*. Εκεί ορίζεται η τιμή της ως OnlyLogin, και μπορεί να αλλάξει σε RegisterAndLogin μόνο μέσω του proxy.

4.1.2 Forwarding



Στην κλάση *PhoneFrame* τοποθετήσαμε το forward button.

Στην κλάση *GuiManager* προσθέσαμε τη συνάρτηση *forwardButtonActionPerformed* η οποία χρησιμοποιείται για να εμφανίσει το *ForwardSplash*.

Στη διαπροσωπεία *UserActionListener* ορίζονται οι συναρτήσεις *handleGetForwardRequest* και *handleNewForwardRequest* οι οποίες υλοποιούνται στον *SipCommunicator*.

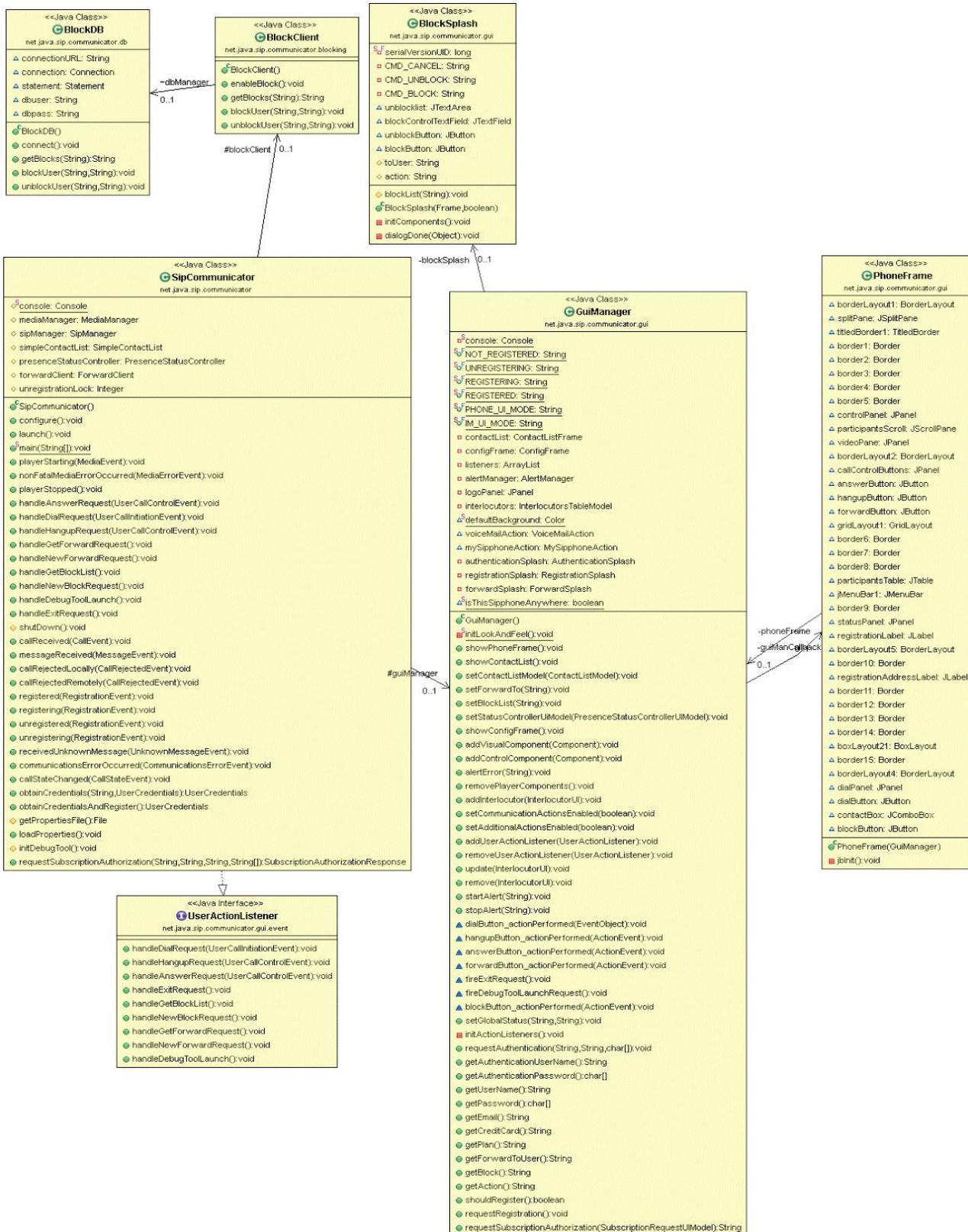
- Η συνάρτηση *handleGetForwardRequest* καλεί την *setForwardTo* πάνω στην κλάση *GuiManager*, ενώ
- η *handleNewForwardRequest* καλεί τη *setForward* πάνω στην κλάση *ForwardClient*, όπου ελέγχεται αν το όνομα που πληκτρολόγησε ο χρήστης είναι έγκυρο ή αν δημιουργείται κύκλος κατά την προώθηση.

Η κλάση *ForwardSplash* ορίζει το *toUser* string που πληκτρολόγησε ο χρήστης στο παράθυρο προώθησης.

Η κλάση *ForwardClient* είναι υπεύθυνη για την κλήση των συναρτήσεων της κλάσης *ForwardDB*.

Στην κλάση *ForwardDB* έχουμε τις εξής συναρτήσεις: *resetForward*, που διαγράφει από τη βάση την καταχώρηση του χρήστη για προώθηση των κλήσεών του, *getForward*, που επιστρέφει το όνομα του χρήστη προς τον οποίο προωθεί ο συγκεκριμένος χρήστης τις κλήσεις του, *setForward*, με την οποία γίνεται έλεγχος εάν το όνομα του χρήστη στον οποίο θέλουμε να προωθήσουμε τις κλήσεις υπάρχει και εάν δημιουργείται κύκλος στην προώθηση, κατόπιν διαγράφουμε την παλιά καταχώρηση στη βάση (αν υπάρχει) και εισάγουμε τη νέα.

4.1.3 Blocking



Στην κλάση *PhoneFrame* τοποθετούμε το blocking button πάνω στο phone frame.

Στην κλάση *GuiManager* προσθέσαμε τη συνάρτηση *blockButtonActionPerformed* η οποία χρησιμοποιείται για να εμφανίσει το *BlockSplash*. (handles)

Στη διαπροσωπεία *UserActionListener* ορίζονται οι συναρτήσεις *handleGetBlockList* και *handleNewBlockList* που υλοποιούνται από την κλάση *SipCommunicator*.

- Η συνάρτηση *handleGetBlockList* καλεί πάνω στην κλάση *BlockDB* τη συνάρτηση *getBlocks* ώστε να εμφανίσει στο χρήστη τη λίστα των χρηστών που εκείνος έχει μπλοκάρει.
- Η συνάρτηση *handleNewBlockList* χρησιμοποιείται για το μπλοκάρισμα και ξεμπλοκάρισμα ενός χρήστη μέσω των συναρτήσεων *blockUser* και *unblockUser* πάνω στην κλάση *BlockClient*.

Η κλάση *BlockSplash* θέτει το *toUser* string που πληκτρολόγησε ο χρήστης στο παράθυρο.

Η κλάση *BlockClient* είναι υπεύθυνη για την κλήση των συναρτήσεων της κλάσης *BlockDB*.

Η κλάση *BlockDB* υλοποιεί τη συνάρτηση *getBlocks*, που επιστρέφει τους μπλοκαρισμένους χρήστες, την *unblockUser*, για τη διαγραφή μιας καταχώρησης από τη βάση, και τη *blockUser*, για την εισαγωγή νέας καταχώρησης για μπλοκάρισμα στη βάση.

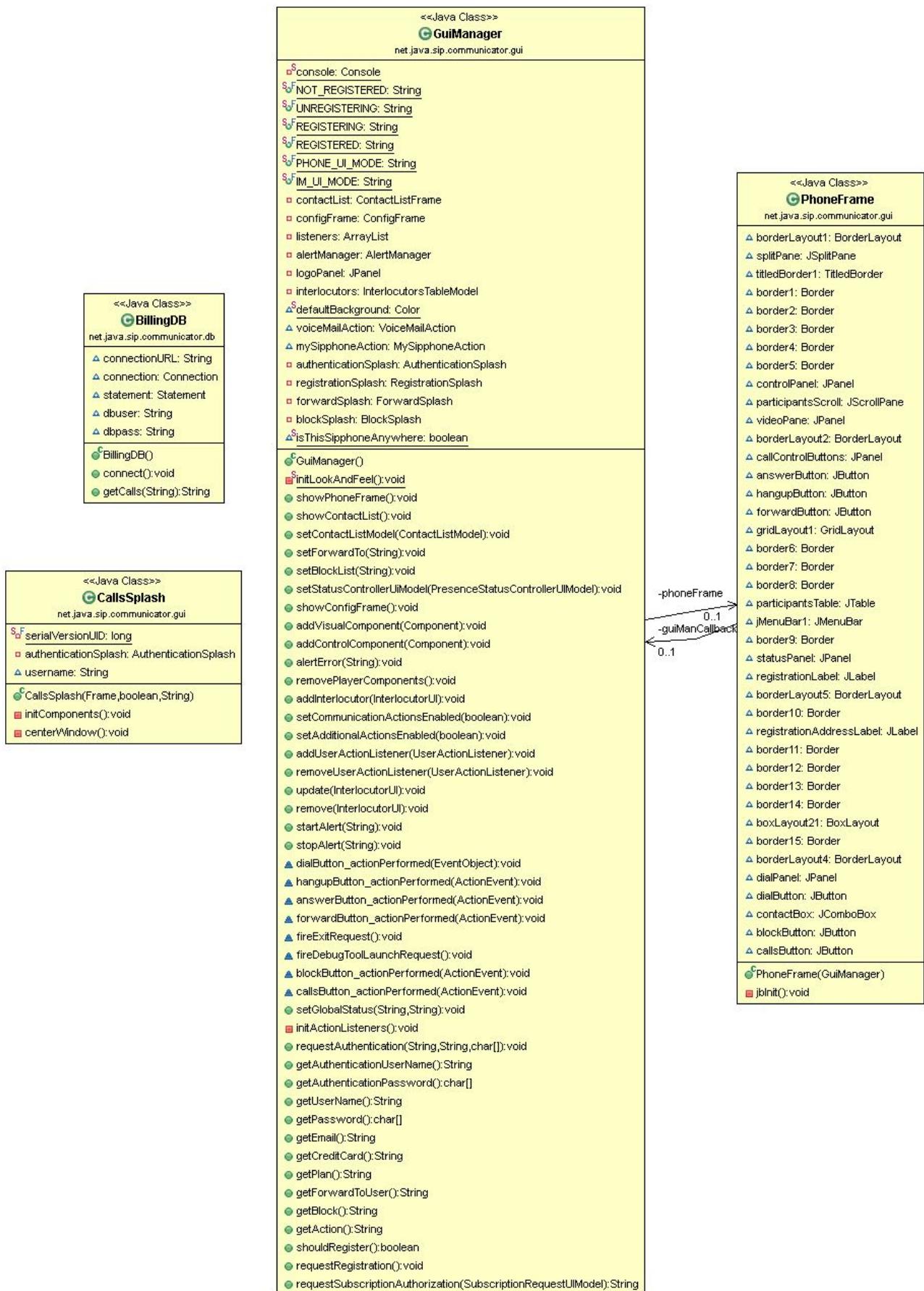
4.1.4 Call's Information

***net.java.sip.communicator.gui.PhoneFrame*:** Προσθέτει το button “Calls” στο παράθυρο της εφαρμογής του Sip Communicator.

***net.java.sip.communicator.gui.GuiManager*:** Όταν ο χρήστης πατήσει το button “Calls”, εκτελείται η μέθοδος *callsButtonActionPerformed*, η οποία με τη σειρά της δημιουργεί ένα αντικείμενο της κλάσης *CallsSplash*.

***net.java.sip.communicator.gui.CallsSplash*:** Είναι το παράθυρο που εμφανίζεται όταν ο χρήστης πατήσει το button “Calls”. Καλεί τη μέθοδο *getCalls(username)*, ώστε να εμφανιστούν όλες οι κλήσεις που έχει πραγματοποιήσει ο συγκεκριμένος χρήστης στο παρελθόν, και έχει χρεωθεί για αυτές.

***net.java.sip.communicator.db.BillingDB*:** Υλοποιεί τη μέθοδο *getCalls*, φροντίζοντας για την επικοινωνία με τη βάση δεδομένων.

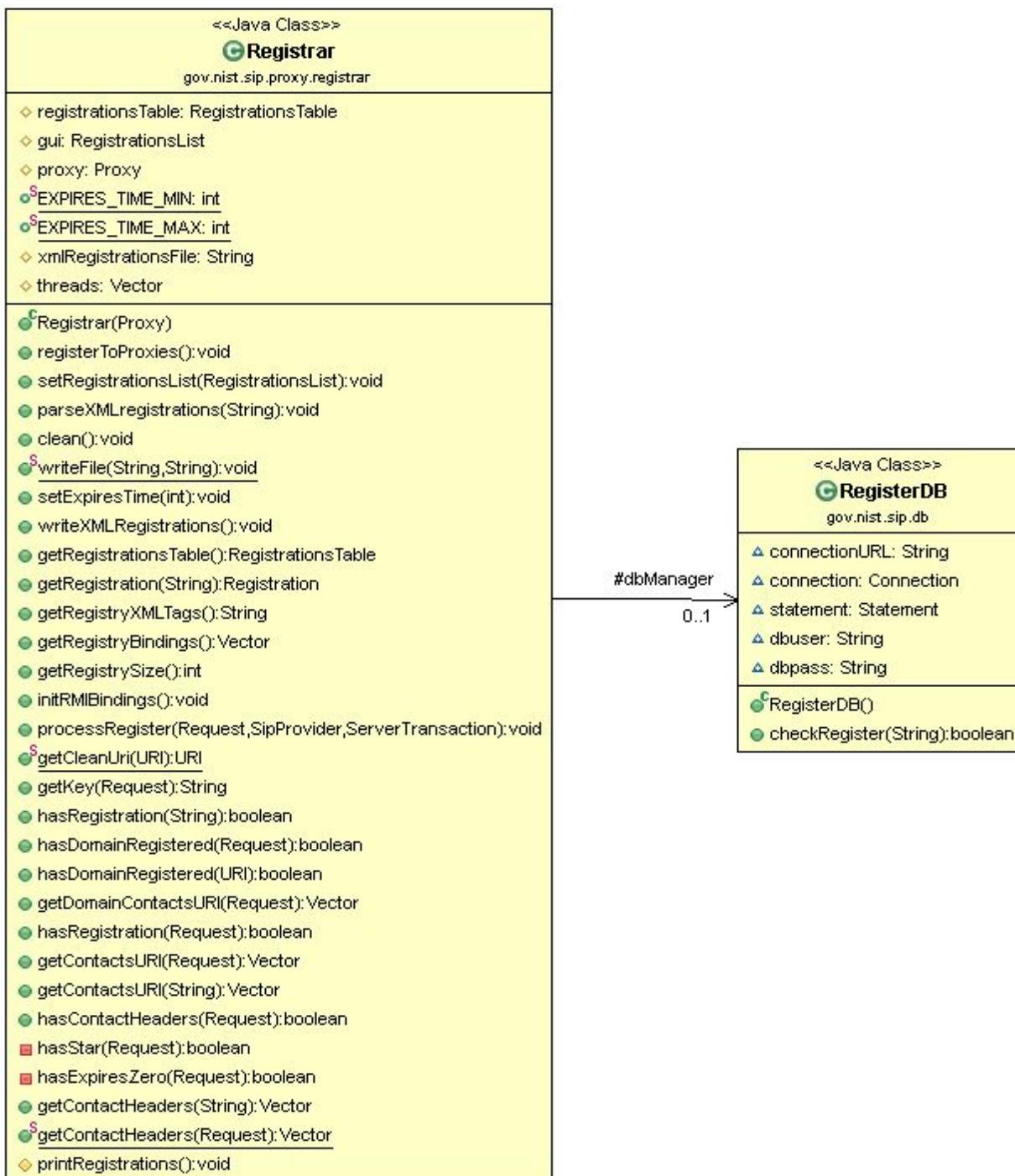


4.2 Server (Sip Proxy)

4.2.1 Registration

gov.nist.proxy.registrar.Registrar: Όταν ένας client θέλει να εγγραφεί στην υπηρεσία, γίνεται έλεγχος αν το username που έδωσε ο client υπάρχει ήδη στη βάση. Στην περίπτωση αυτή, ο proxy server στέλνει response not_found, στον αντίστοιχο sip communicator.

gov.nist.sip.db.RegisterDB: Υλοποιεί την επικοινωνία του server με τον πίνακα users της βάσης, ώστε να πραγματοποιηθεί ο παραπάνω έλεγχος.

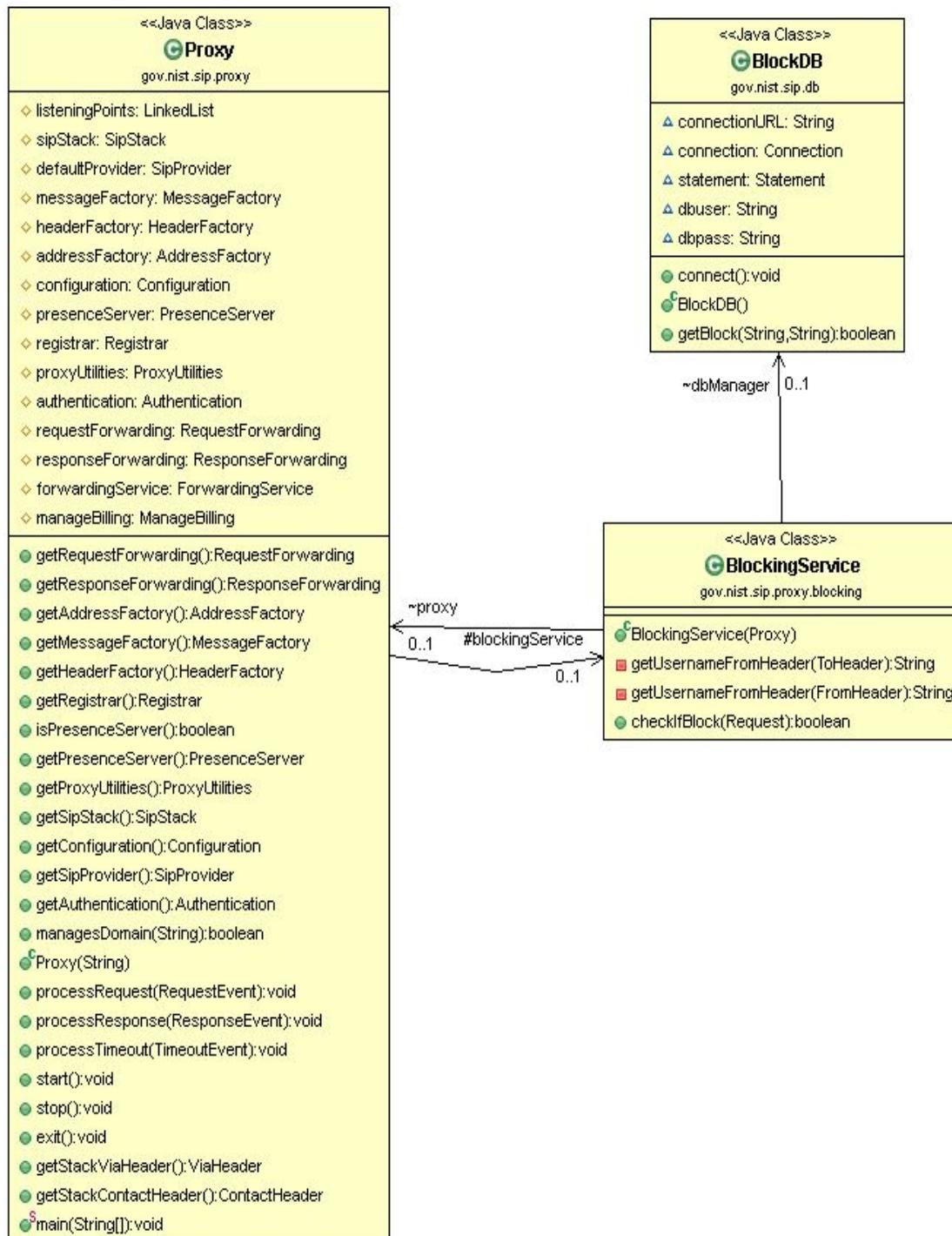


4.2.2 Blocking

gov.nist.sip.proxy.Proxy: Κάθε φορά που λαμβάνεται από τον proxy request invite, δηλαδή πρόσκληση σε κλήση, ελέγχεται αν ο caller είναι blocked από τον callee. Στην περίπτωση αυτή, ο proxy στέλνει busy_here response στον caller.

gov.nist.sip.proxy.blocking.BlockingService: Πραγματοποιεί τον παραπάνω έλεγχο με τη μέθοδο checkIfBlock(request).

gov.nist.sip.db.BlockDB: Υλοποιεί την επικοινωνία του server με τον πίνακα blocking της βάσης.

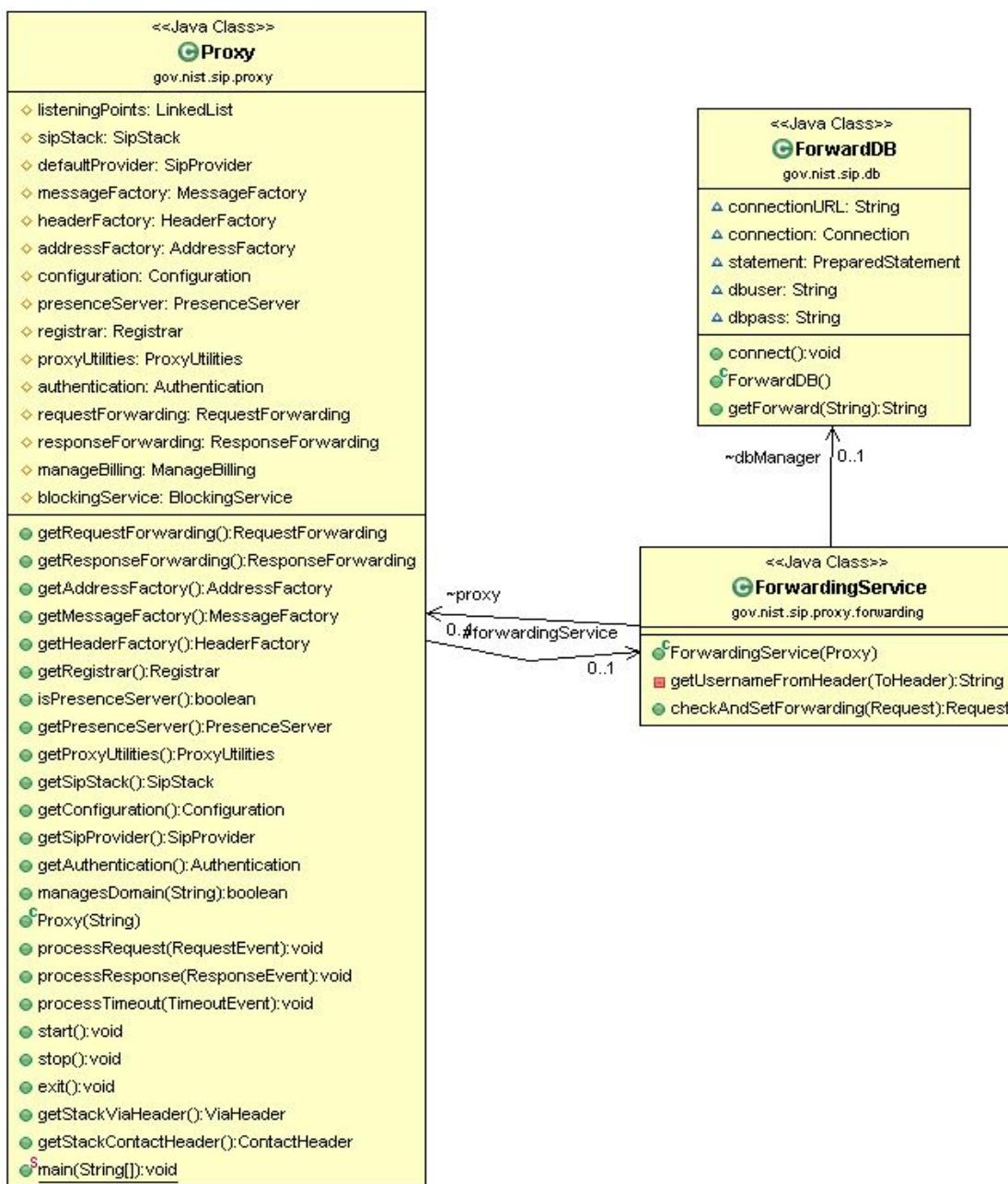


4.2.3 Forwarding

gov.nist.sip.proxy.Proxy: Κάθε φορά που λαμβάνεται από τον proxy request invite, δηλαδή πρόσκληση σε κλήση, ελέγχεται αν ο caller έχει θέσει κάποιον χρήστη στον οποίο θέλει να προωθηθεί η κλήση. Επίσης, γίνεται έλεγχος αν ο χρήστης στον οποίο πιθανώς θα προωθηθεί η κλήση, έχει μπλοκάρει τον callee.

gov.nist.sip.proxy.forwarding.ForwardingService: Πραγματοποιεί τον παραπάνω έλεγχο προώθησης με τη μέθοδο checkAndSetForwarding(request).

gov.nist.sip.db.ForwardDB: Υλοποιεί την επικοινωνία του server με τον πίνακα forwarding της βάσης.



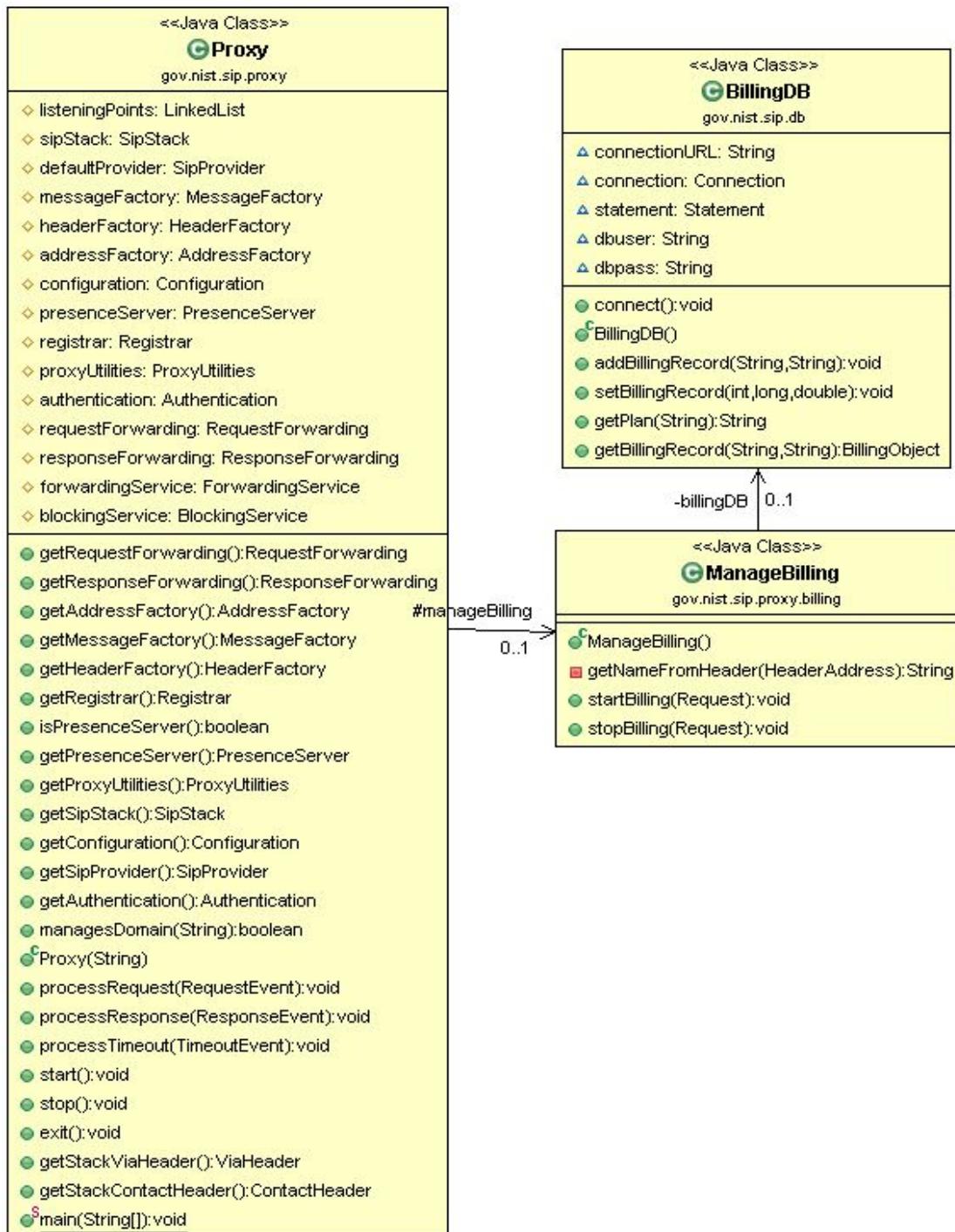
4.2.4 Billing

gov.nist.sip.proxy.Proxy: Κάθε φορά που λαμβάνεται από τον proxy request ACK, δηλαδή επιβεβαίωση έναρξης κλήσης, αρχίζει το billing. Επίσης, κάθε φορά που

λαμβάνεται από τον proxy request BYE, δηλαδή τερματισμός κλήσης, σταματά το billing.

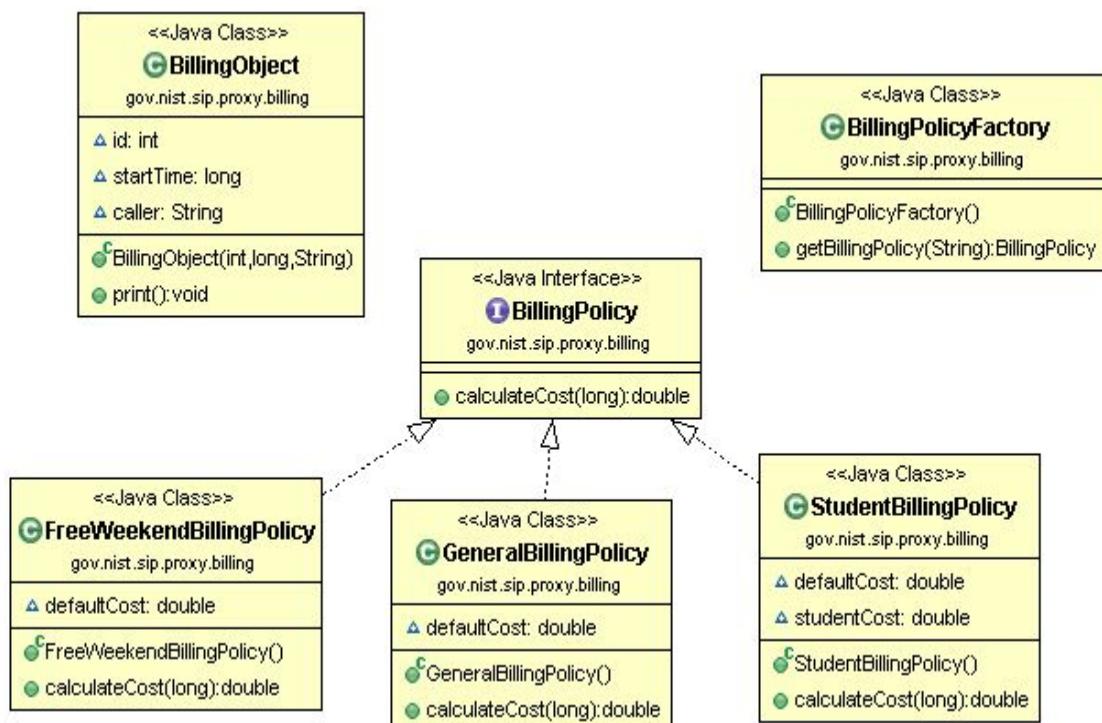
gov.nist.sip.proxy.billing.ManageBilling: Υλοποιεί τις μεθόδους startBilling και stopBilling, οι οποίες κάνουν τις απαραίτητες ενέργειες για να υπολογιστεί σωστά η διάρκεια και χρέωση της κλήσης.

gov.nist.sip.db.BillingDB: Υλοποιεί την επικοινωνία του server με τον πίνακα billing της βάσης.

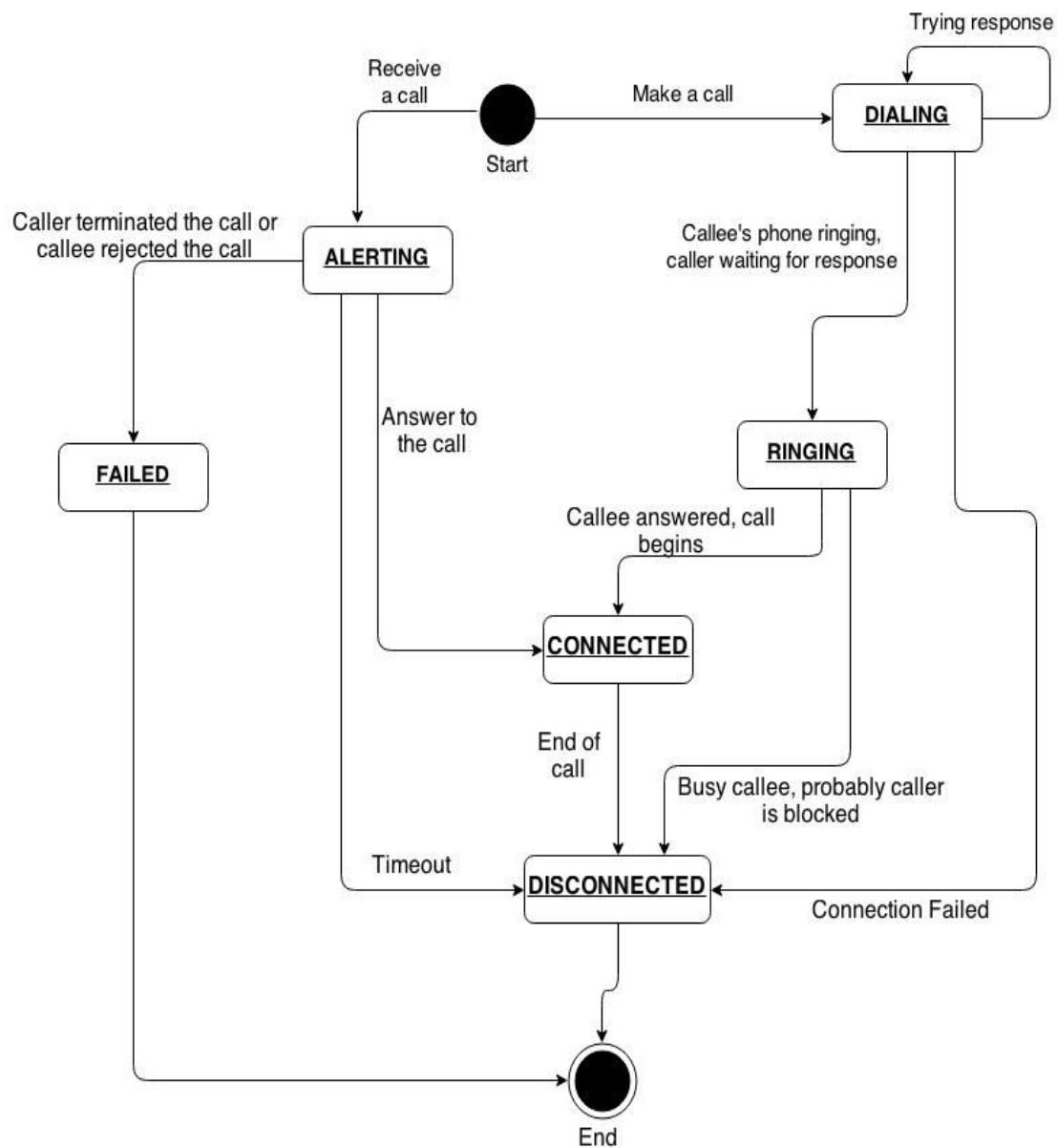


Factory Method Design Pattern:

Για την υλοποίηση των διαφόρων προγραμμάτων χρέωσης ακολουθούμε το factory method design pattern. Όταν σταματά η κλήση, γίνεται ο υπολογισμός της διάρκειας και χρέωσης. Δημιουργούμε ένα αντικείμενο *BillingObject* που κρατά τα στοιχεία της κλήσης, όπως πότε ξεκίνησε και ποιος είναι ο χρήστης που θα χρεωθεί, δηλαδή ο caller. Ο χρήστης κατά την εγγραφή του μπορούσε να διαλέξει ένα από τα τρία διαθέσιμα προγράμματα χρέωσης, General, Student, FreeWeekend. Ανάλογα το πρόγραμμα που έχει επιλέξει ο caller, γίνεται αντίστοιχα και η χρέωση. Αυτό πραγματοποιείται από την κλάση *BillingPolicyFactory*, η οποία επιστρέφει ένα από τα τρία αντικείμενα που υλοποιούν το *BillingPolicy* interface, ανάλογα το πρόγραμμα χρέωσης που είχε επιλέξει ο χρήστης. Με τον τρόπο αυτό, θα κληθεί η *calculateCost()* που αντιστοιχεί στο επιθυμητό billing policy.



5 State Diagram



6 Ανοιχτά Ζητήματα

6.1 Κλήση όπου ο caller και ο callee είναι το ίδιο πρόσωπο

Στην τρέχουσα εφαρμογή μας, ο χρήστης ήταν σε θέση να καλεί τον εαυτό του και να λαμβάνει το αίτημα της κλήσης. Αυτό μπορεί να προκαλέσει προβλήματα στο μέλλον και έχουμε την πρόθεση να το διορθώσουμε με την προσθήκη ενός διακομιστή (server) ή ενός ελέγχου από την πλευρά του πελάτη πριν πραγματοποιηθεί η κλήση. Ωστόσο, στα πλαίσια αυτού του project για την αποφυγή του παραπάνω σεναρίου, φροντίσαμε κατά την εγγραφή ενός χρήστη, να “μπλοκάρει” ο ίδιος τον εαυτό του (να γίνεται, δηλαδή, μία εγγραφή στον πίνακα blocking της βάσης), χωρίς όμως αυτός να βλέπει στο παράθυρο της εφαρμογής, στη λίστα με τους μπλοκαρισμένους χρήστες, τον εαυτό του.

6.2 Επικύρωση δεδομένων

Θα πρέπει να είμαστε σε θέση να ανιχνεύουμε και να ακυρώνουμε (απορρίπτουμε) τροποποιημένα (κακόβουλα ειδικά) δεδομένα που αποστέλλονται στο διακομιστή (server) από κάποιον πελάτη. Κυρίως κατά τη διάρκεια της κλήσης, η οποία προς το παρόν μετράται από την πλευρά του πελάτη, αυτό θα μπορούσε να αποτελέσει απειλή για την ασφάλεια του συστήματος. Οι χρήστες δεν θα πρέπει να έχουν πρόσβαση σε αυτές τις πληροφορίες.

6.3 Κρυπτογραφημένη επικοινωνία πελάτη - διακομιστή

Στην παρούσα εφαρμογή, ο διακομιστής και ο πελάτης επικοινωνούν μέσω μηνυμάτων απλού κειμένου. Αυτό δεν είναι μια ασφαλής πρακτική, ιδίως όταν μεταδίδονται ευαίσθητα προσωπικά δεδομένα (π.χ. κωδικοί πρόσβασης ή αριθμοί πιστωτικών καρτών). Ένα ασφαλές σύστημα κρυπτογράφησης θα πρέπει να προστεθεί, πιθανότατα χρησιμοποιώντας ασύμμετρη κρυπτογραφία - TLS.

6.4 Εφαρμογή πιστωτικού συστήματος

Αυτή τη στιγμή ο διακομιστής απλά χρεώνει τον αύξων αριθμό που δίνεται κατά την εγγραφή του χρήστη. Πρέπει να εφαρμόσουμε μια υπηρεσία η οποία να συνδέει αυτόν το σειριακό αριθμό σε πιστωτική / χρεωστική κάρτα ή άλλη μέθοδο πληρωμής (PayPal / Amazon Credit / PaySafe).

6.5 Επέκταση του SIP-Proxy GUI

Το πλάνο για επέκταση του SIP-Proxy GUI περιλαμβάνει το σχεδιασμό γραφικών UI για έναν Πίνακα Ελέγχου Διαχείρισης, ο οποίος θα εμφανίζει (μεταξύ άλλων) τις τελευταίες αλλαγές που έγιναν στα χαρακτηριστικά που εφαρμόστηκαν πιο πρόσφατα.

6.6 Πολλαπλές προωθήσεις

Στην παρούσα εφαρμογή ο χρήστης επιτρέπεται να ορίσει μόνο ένα άτομο στο οποίο θα προωθήσει τις κλήσεις του / της. Θα πρέπει να υπάρχει μια λίστα προτεραιότητας για τις προωθήσεις, έτσι ώστε σε περίπτωση που η πρώτη καθορισμένη προώθηση αποτύχει, η κλήση να μην αποτυγχάνει αμέσως, αλλά να γίνεται η προώθηση στους επόμενους κατά προτεραιότητα χρήστες στη λίστα.