ΕΠΕΚΤΑΣΗ ΠΡΩΤΟΚΟΛΛΟΥ SIP

Έντυπο Προδιαγραφής Σχεδίασης

Συγγραφείς Εγγράφου (Ομάδες 12 και 13)

Όνομα	**Αριθμός Μητρώου**
Ντάλλας Ιωάννης	03111418
Πενταράκης Μανώλης	03111048
Τσιτσεκλης Κωνσταντίνος	03111409
Χατζηκυριάκος Γιώργος	03111164

Περιεχόμενα

1 Εισαγωγή

Σκοπός της εργασίας αυτής είναι η επέκταση του προγράμματος πελάτη Sip Communicator και του προγράμματος εξυπηρετητή JAIN Sip Proxy. Και τα δύο χρησιμοποιούν το πρωτόκολλο *SIP (Session Initiation Protocol)* , όπως έχει οριστεί στο πρότυπο *RFC 3261*.

Το project στοχεύει στην υλοποίηση των λειτουργιών της φραγής κλήσεων από ορισμένους χρήστες, προώθησης κλήσεων και χρέωσης κλήσεων με βάση κάποια συγκεκριμένη πολιτική.

1.1 Αναφορές

- *RFC 3261* http://www.ietf.org/rfc/rfc3261.txt
- SRS.pdf

2 Κύριες Σχεδιαστικές Αποφάσεις

Για την αποθήκευση των πληροφοριών πού αφορούν τους χρήστες (στοιχεία εγγραφής, φραγή, προώθηση κλπ) χρησιμοποιήθηκε μία απλή βάση δεδομένων. Για το στήσιμο της βάσης αυτής χρησιμοποιήθηκε η *MySQL*, ένα ανοιχτού κώδικα σύστημα διαχείρισης σχεσιακών βάσεων δεδομένων και ο *Apache HTTP web server* για τοπική διαχείριση στο σύστημα μας.

Για την εγγραφή των χρηστών στο σύστημα γίνεται άμεση επικοινωνία με την τοπική βάση δεδομένων. Συγκεκριμένα, στη πρώτη εγγραφή στο σύστημα πρέπει ο χρήστης να περνάει τα στοιχεία του στο σύστημα και στην συνέχεια σε κάθε σύνδεση βάζει μόνο username και password. Επίσης, η διαδικασία της εγγραφής θα πρέπει να έχει προηγηθεί αλλιώς ο χρήστης δεν θα μπορεί να συνδεθεί στο σύστημα.

Αντίστοιχα, η επιλογή χρηστών για φραγή και προώθηση γίνεται από την πλευρά του χρήστη και η πληροφορίες αυτές αποθηκεύονται στην βάση δεδομένων μετά από άμεση επικοινωνία client-database.

Η διαδικασία της φραγής κλήσεων γίνεται από τον Proxy. Όταν ένα αίτημα για κλήση στέλνεται από ένα χρήστη Α προς ένα χρήστη Β, ο Proxy ελέγχει την βάση για να επιβεβαιώσει ότι ο χρήστης Β δεν μπλοκάρει τον Α. Αν υπάρχει φραγή στέλνεται στον χρήστη Α απάντηση *busy here.*

Στην λειτουργία προώθησης ο Proxy χρησιμοποιεί την βάση δεδομένων για να βρει τον τελικό παραλήπτη. Αν ο τελικός παραλήπτης δεν είναι συνδεδεμένος τότε ο χρήστης που ξεκίνησε την κλήση θα λάβει μήνυμα *temporarily unavailable.* Ακόμη, αν στην διαδικασία εύρεσης τελικού παραλήπτη παρατηρηθεί κύκλος τότε στέλνεται μήνυμα *loop detected.* Αν βρεθεί τελικός παραλήπτης (χωρίς προβλήματα) τότε στέλνεται σε αυτόν το αίτημα κλήσης.

Η χρέωση των κλήσεων χρησιμοποιεί τα μηνύματα *ACK*, για τον εντοπισμό έναρξης της κλήσης και των χρηστών που εμπλέκονται στην κλήση, και *BYE* , για τον εντοπισμό τερματισμού της κλήσης, τον υπολογισμό διάρκειας της κλήση και την αντίστοιχη χρέωση. Η στρατηγική χρέωσης επιλέγεται από τον Proxy αφού επικοινωνήσει με την βάση και επιλεχτεί το κατάλληλο πακέτο.

Τα πακέτα που υπάρχουν είναι τρία:

- *Premium Package* : Ο χρήστης πληρώνει 15$ και έχει απεριόριστες κλήσεις για ένα μήνα.

- *Friends Package* : Ο χρήστης πληρώνει 2$ και έχει έκπτωση στις κλήσεις με χρήστες που βρίσκονται στην λίστα των φίλων του για ένα μήνα. Συγκεκριμένα για τα πρώτα 30” η χρέωση κλήσης είναι 0.2$ μετά για 31”-120” γίνεται χρέωση 0.005$/sec και για περισσότερα από 120” η χρέωση είναι 0.0025$/sec

- *Standard Package* : Ο χρήστης δεν πληρώνει κάποια αρχική τιμή. Συγκεκριμένα για τα πρώτα 30” η χρέωση κλήσης είναι 0.25$ μετά για 31”-120” γίνεται χρέωση 0.01$/sec και για περισσότερα από 120” η χρέωση είναι 0.005$/sec.

Τα πακέτα αυτά έχουν και κάποια ιεραρχία για να επιλεχθεί η σωστή πολιτική χρέωσης. Αρχικά, ελέγχεται αν ο χρήστης έχει *Premium Package* και μετά αν έχει *Friends Package*. Αν δεν έχει κανένα από τα δύο τότε επιλέγεται το *Standard Package*.

Οι υλοποιημένες επεκτάσεις του Proxy έχουν κι αυτές κάποια σειρά/ιεραρχία. Όταν λαμβάνει μήνυμα για την έναρξη μιας κλήσης ελέγχει αρχικά αν υπάρχει φραγή μεταξύ των δύο χρηστών ή αν οι προωθήσεις που γίνονται οδηγούν σε κύκλο. Αν δεν συμβαίνει τίποτα από αυτά, τότε ο Proxy βρίσκει τον τελικό παραλήπτη (επικοινωνώντας με την βάση για να ελέγξουμε τις προωθήσεις) και η χρέωση γίνεται μεταξύ του χρήστη που ξεκίνησε την κλήση και του τελικού παραλήπτη. Η πίστωση του κόστους της κλήσης γίνεται ανάλογα με αυτούς τους δύο χρήστες.

Ακόμα, έχει υλοποιηθεί και μέριμνα για λάθη καθώς δεν επιτρέπεται να κάποιος να μπλοκάρει, να κάνει προώθηση και να προσθέσει στην λίστα φίλων να τον εαυτό του. Επίσης, δεν είναι εφικτό κάποιος χρήστης να προωθεί τις κλήσεις σε πολλά άτομα ή να αγοράσει πολλαπλές φορές το ίδιο πακέτο.

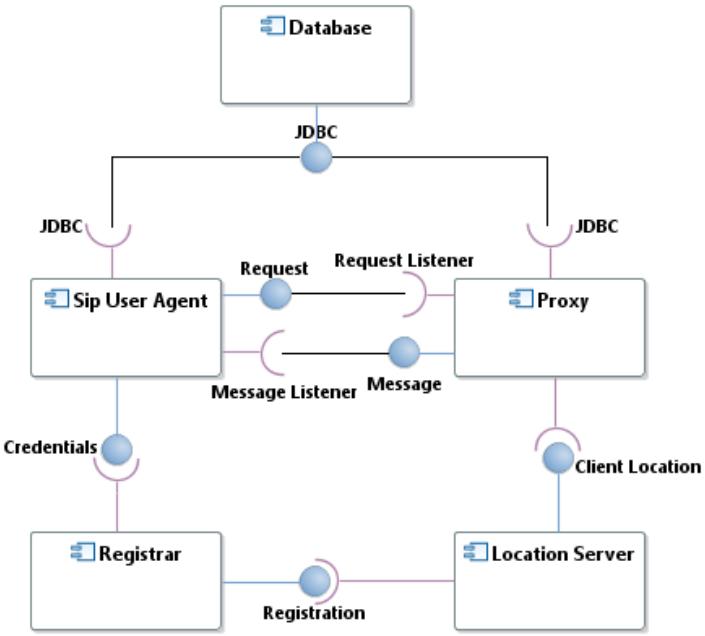
Σημειώνεται επίσης, πως για επιπλέον ασφάλεια οι κωδικοί των χρηστών περνιούνται στην βάση δεδομένων αφού έχει εφαρμοστεί πάνω τους ο αλγόριθμος hashing *SHA1* και με βάση αυτό γίνεται και ο έλεγχος των στοιχείων του χρήστη κατά στην διαδικασία σύνδεσης.

Επιπλέον έχουν γίνει και μερικές αλλαγές στο γραφικό περιβάλλον για την προσθήκη τον επιπλέον λειτουργιών που αναπτύχθηκαν.

Τέλος, για την επέκταση του Proxy.java με τις λειτουργίες του blocking, forwarding και billing χρησιμοποιήθηκε το σχεδιαστικό μόρφημα *Decorator* και για την επιλογή της πολιτικής χρέωσης το σχεδιαστικό μόρφημα *Strategy.*

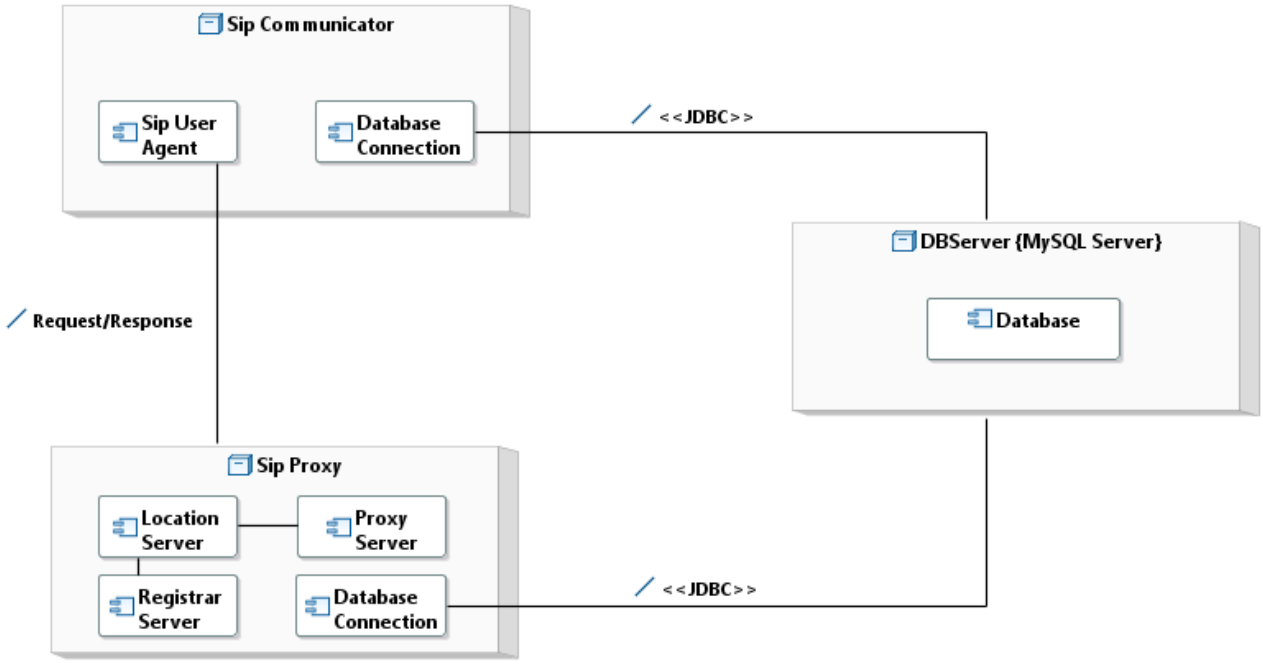
3 Αρχιτεκτονική

Component diagram



Οι *Sip User Agent* (χρήστης) και *Proxy* χρειάζονται το interface της database για να συνδεθούν στην βάση δεδομένων. Επίσης, ο χρήστης επικοινωνεί με τον *Registrar* και του στέλνει τα στοιχεία της εγγραφής του και στην συνέχεια εκείνος επικοινωνεί με τον *Location Server* για την αποθήκευση της διεύθυνσης του χρήστη. Ο *Proxy* από την άλλη πλευρά ζητάει από τον *Location Server* τα στοιχεία του παραλήπτη του μηνύματος που πρέπει να προωθήσει. Τέλος, όπως είναι λογικό ο *Sip User Agent* και ο *Proxy server* επικοινωνούν ανταλλάζοντας μηνύματα του πρωτοκόλλου *SIP*.

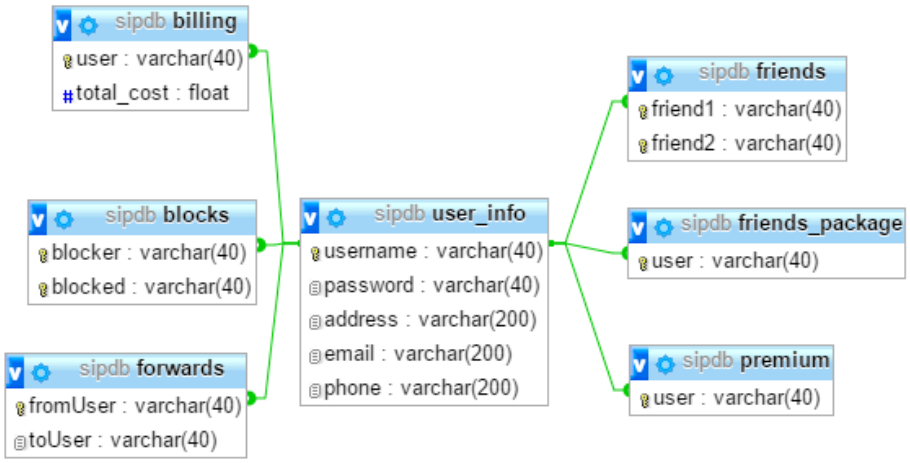
Deployment diagram



Ο *Sip Communicator* έχει το UI που χρησιμοποιεί ο χρήστης για να επικοινωνεί με την βάση και τον *Sip Proxy* και ένα component που χρησιμοποιείται για να γίνει η σύνδεση με την βάση.

Αντίστοιχα, ο *Sip Proxy* περιέχει τους τρεις διαφορετικούς server που φαίνονται στο σχήμα και ένα ίδιο component για την σύνδεση με την βάση. Επίσης, οι servers που περιλαμβάνονται στο *Sip Proxy* έχουν συνδέσεις μεταξύ τους (η επικοινωνία μεταξύ τους περιγράφτηκε καλύτερα από το *Component diagram*).

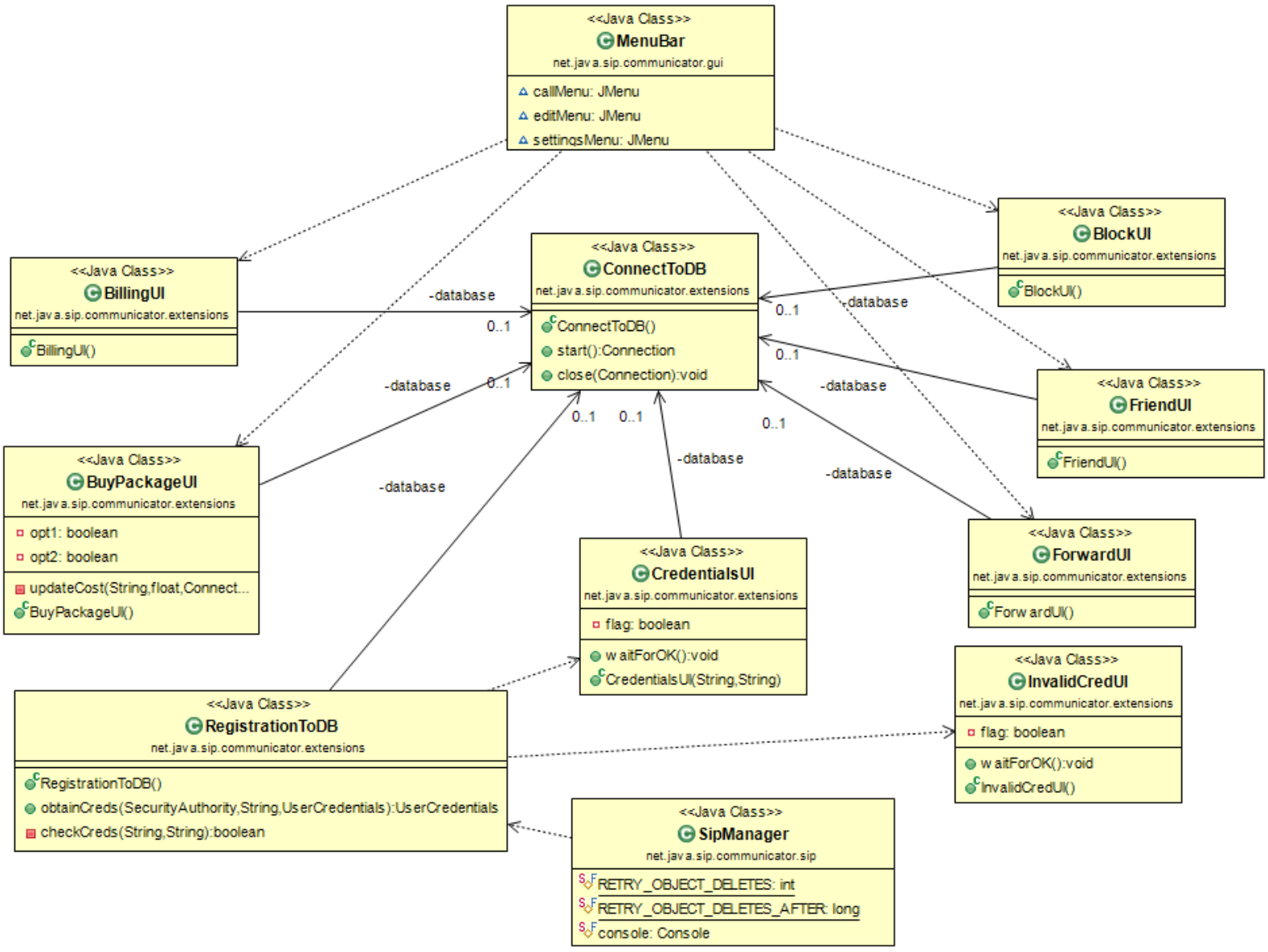
Βάση Δεδομένων



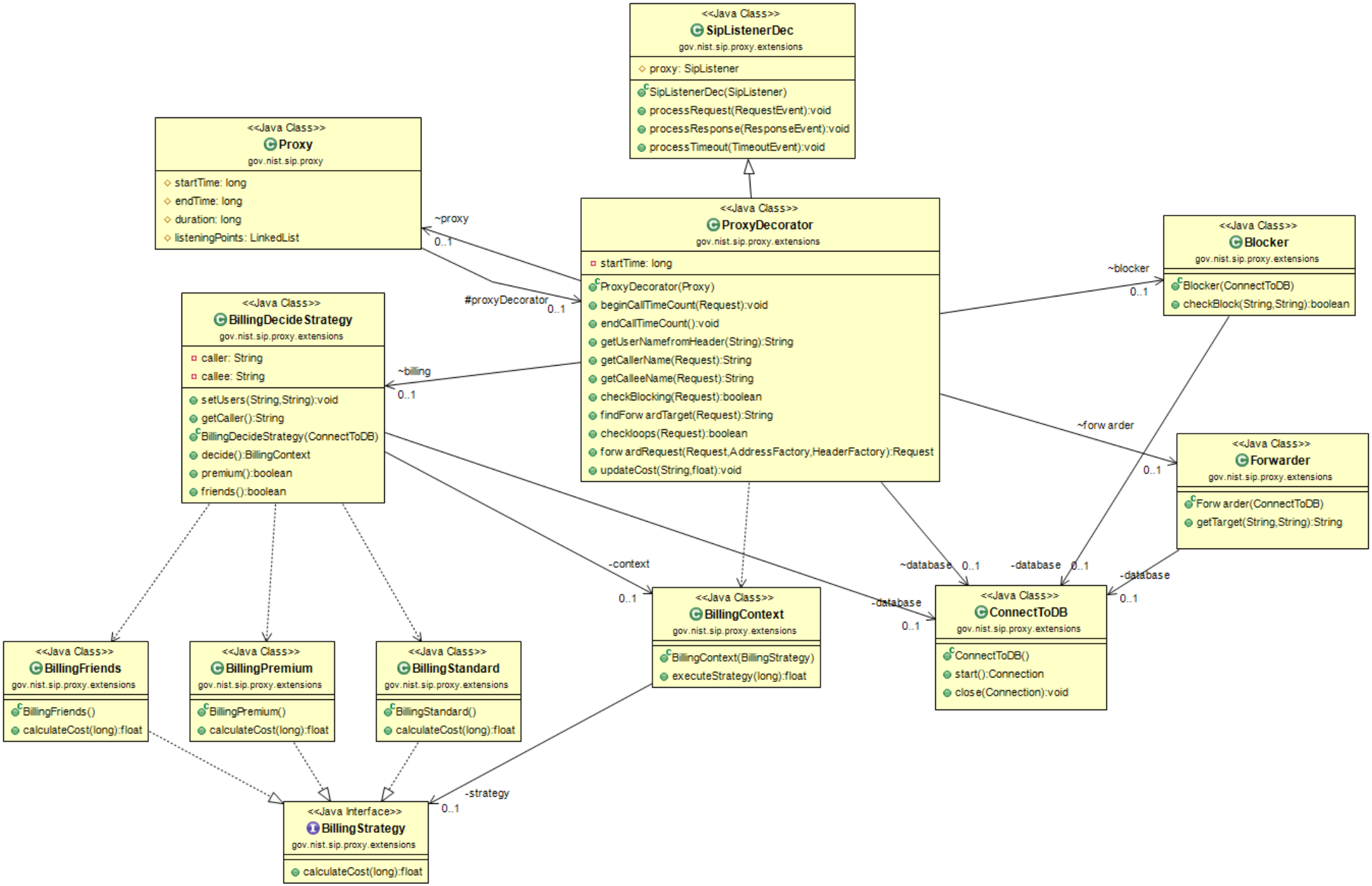
4 Λεπτομερή Διαγράμματα κλάσεων

4.1 UML Διαγράμματα Κλάσεων

Sip Communicator



Sip Proxy


<<Java Class>>
SipListenerDec
gov.nist.sip.proxy.extensions
proxy: SipListener
SipListenerDec(SipListener)
processRequest(RequestEvent):void
processResponse(ResponseEvent):void
processTimeout(TimeoutEvent):void
<<Java Class>>
Proxy
gov.nist.sip.proxy
startTime: long
endTime: long
duration: long
listeningPoints: LinkedList
~proxy
0..1
#proxyDecorator
0..1
<<Java Class>>
ProxyDecorator
gov.nist.sip.proxy.extensions
startTime: long
ProxyDecorator(Proxy)
beginCallTimeCount(Request):void
endCallTimeCount():void
getUserNamefromHeader(String):String
getCallerName(Request):String
getCalleeName(Request):String
checkBlocking(Request):boolean
findForwardTarget(Request):String
checkloops(Request):boolean
forwardRequest(Request,AddressFactory,HeaderFactory):Request
updateCost(String,float):void
<<Java Class>>
Blocker
gov.nist.sip.proxy.extensions
Blocker(ConnectToDB)
checkBlock(String,String):boolean
~blocker
0..1
<<Java Class>>
BillingDecideStrategy
gov.nist.sip.proxy.extensions
caller: String
callee: String
setUsers(String,String):void
getCaller():String
BillingDecideStrategy(ConnectToDB)
decide():BillingContext
premium():boolean
friends():boolean
~billing
0..1
~forwarder
0..1
<<Java Class>>
Forwarder
gov.nist.sip.proxy.extensions
Forwarder(ConnectToDB)
getTarget(String,String):String
-context
0..1
~database
0..1
-database
0..1
-database
0..1
-database
0..1
<<Java Class>>
BillingContext
gov.nist.sip.proxy.extensions
BillingContext(BillingStrategy)
executeStrategy(long):float
<<Java Class>>
ConnectToDB
gov.nist.sip.proxy.extensions
ConnectToDB()
start():Connection
close(Connection):void
<<Java Class>>
BillingFriends
gov.nist.sip.proxy.extensions
BillingFriends()
calculateCost(long):float
<<Java Class>>
BillingPremium
gov.nist.sip.proxy.extensions
BillingPremium()
calculateCost(long):float
<<Java Class>>
BillingStandard
gov.nist.sip.proxy.extensions
BillingStandard()
calculateCost(long):float
-strategy
0..1
<<Java Interface>>
BillingStrategy
gov.nist.sip.proxy.extensions
calculateCost(long):float

4.2 Method Details

Sip Proxy

ConnectToDB

Η κλάση ConnectToDB χρησιμεύει για την σύνδεση των στοιχείων του προγράμματος με την Βάση Δεδομένων. Η κλάση παρέχει δυο μεθόδους, την start η οποία δημιουργεί μια σύνδεση (connection) με τη βάση και την close η οποία αναλαμβάνει να κλείσει μια υπάρχουσα σύνδεση.

Proxy

Η κλάση Proxy παρέχει τις απαραίτητες μεθόδους για την σύνδεση των χρηστών σε κλήση καθώς και για την υλοποίηση των επιπλέον λειτουργιών που γράψαμε.

ProxyDecorator

Είναι ένας decorator που "προσαρμόζεται" στον Proxy παρέχοντας επιπλέον λειτουργίες που επιτρέπουν την υλοποίηση του blocking (checkblocking), του forwarding (findForwardTarget, checkloops,forwardRequest), του billing (beginCallTime,endCallTime,updateCost) καθώς και γενικότερες μεθόδους για την λήψη των ονομάτων του caller και callee (getUserName, getCallerName. GetCalleeName). Με την χρήση του decorator pattern προσθέτουμε τις λειτουργίες χωρίς να αλλάξουμε την δομή του Proxy.

BillingStrategy

Μια διαπροσωπεία (interface) η οποία επιτρέπει την υλοποίηση διαφορετικών στρατηγικών χρέωσης που κάνουν implement την διαπροσωπεία, δηλαδή τις: BillingStandard, BillingFriends, BillingPremium.

BillingDecideStrategy

Η κλάση αυτή έχει τις μεθόδους με τις οποίες αποφασίζουμε με ποια από τις υπάρχουσες στρατηγικές θα χρεώσουμε τον χρήστη.

BillingContext

Η κλάση BillingContext επιτρέπει να υπολογίζουμε το κόστος της κλήσης με την σωστή στρατηγική.

Blocker

Με τη μέθοδο checkBlock της κλάσης ελέγχουμε με query στη βάση αν ένας χρήστης έχει μπλοκάρει κάποιον άλλο.

Forwarder

Η κλάση αυτή παρέχει τη μέθοδο getTarget με την οποία βρίσκουμε τον τελικό αποδέκτη μιας κλήσης όταν υπάρχουν προωθήσεις.

Sip Communicator

ConnectToDB

Η κλάση ConnectToDB χρησιμεύει για την σύνδεση των στοιχείων του προγράμματος με την Βάση Δεδομένων. Η κλάση παρέχει δυο μεθόδους, την start η οποία δημιουργεί μια σύνδεση (connection) με τη βάση και την close η οποία αναλαμβάνει να κλείσει μια υπάρχουσα σύνδεση.

BillingUI, ForwardUI, BuyPackageUI, CredentialsUI, InvalidCredUI, ForwardUI, FriendUI, BlockUI

Κλάσεις οι οποίες είναι υπεύθυνες για την δημιουργία των παραθύρων διαλόγου που προσαρμόζονται στο πρόγραμμα πελάτη Sip Communicator και για την τροποποίηση των στοιχείων που πληκτρολογεί ο χρήστης στη βάση δεδομένων.

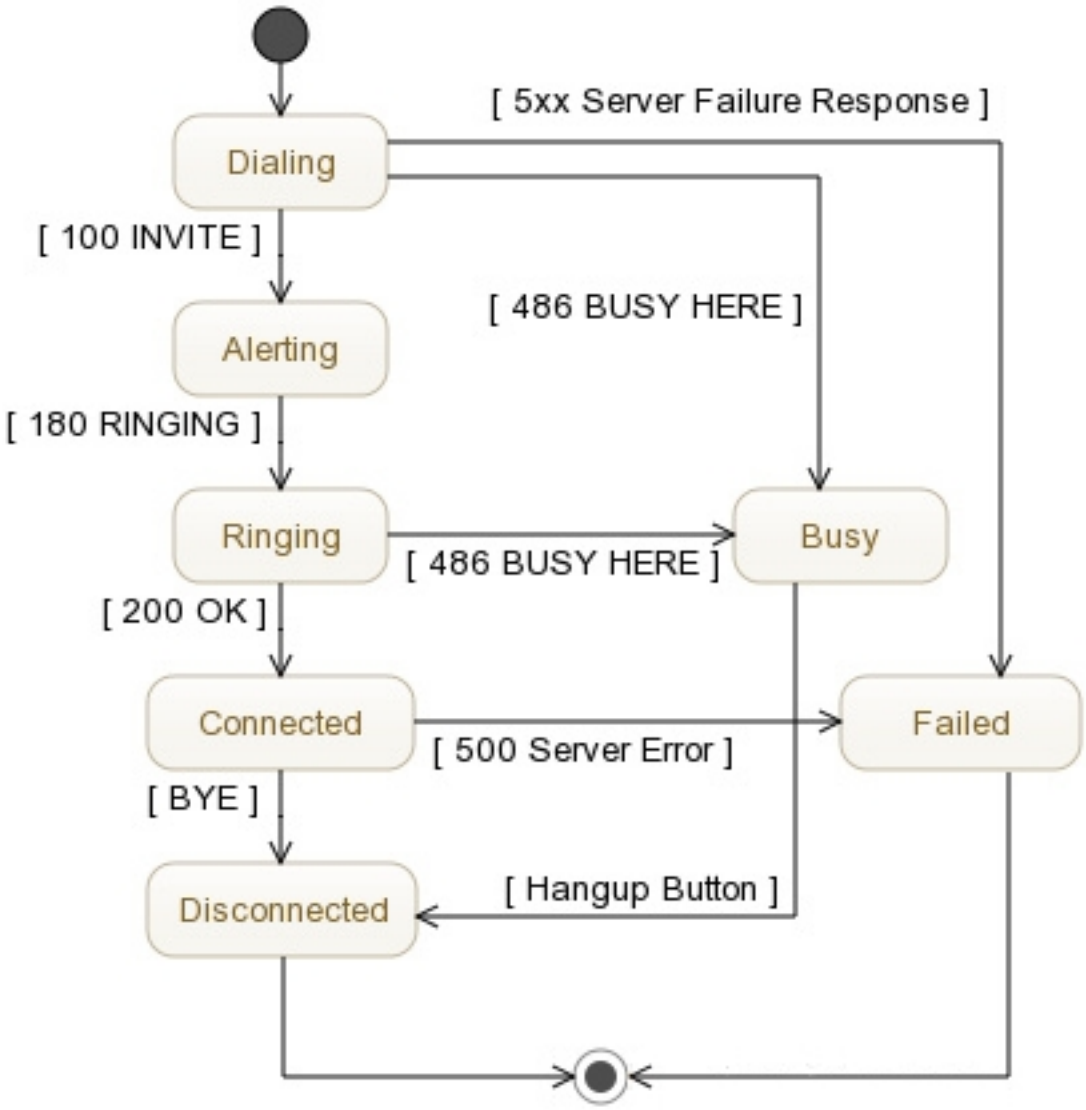
Menubar

Το αντικείμενο που δημιουργείται από αυτή την κλάση περιέχει και εμφανίζει τα κουμπιά και τα dropdown menus που βλέπει ο χρήστης στην εφαρμογή.

RegistrationToDB

Με τις μεθόδους της κλάσης φροντίζουμε για την σωστή εισαγωγή των στοιχείων του χρήστη στη βάση δεδομένων.

5 Διαγράμματα Καταστάσεων



6 Ανοιχτά Ζητήματα

- Αν ο τερματισμός της κλήσης δεν γίνει με φυσιολογικό τρόπο , δηλαδή με το κουμπί “Hangup”, οι κλήσεις μένουν ανοιχτές και οι χρεώσεις δεν γίνονται σωστά.
- Κατά τη διαδικασία testing της εφαρμογής παρατηρήθηκε ότι σε κάποιες σπάνιες περιπτώσεις, κάποιος client λάμβανε κλήση από κάποιο χρήστη που δεν ήταν εγγεγραμμένος στο σύστημα ούτε στην βάση δεδομένων. Δεν γνωρίζουμε αν αυτό οφείλεται σε bug του κώδικα η σε κάποιο θέμα του δικτύου.