Έγγραφο Προδιαγραφής Σχεδίασης

Version:	
Print Date:	21-Dec-16
Release Date:	
Release State:	
Approval State:	
Approved by:	
Prepared by:	
Reviewed by:	
Path Name:	
File Name:	SDD (2).odt
Document No:	

Document Sign-Off

Ονοματεπώνυμο	Α.Μ.	
Πασχάλης Ηλίας	031 12122	
Σκλήκας Νικόλαος	031 12503	
Σκουλός Ραφαήλ	031 12404	
Σταθόπουλος Αναστάσιος	031 12140	

Contents

1 Introduction

1.1 Overview

Στο παρόν έγγραφο περιγράφουμε τις προδιαγραφές σχεδίασης μιας εφαρμογής επικοινωνίας μέσω VoIP. Η εφαρμογή μας αναπτύσσεται με βάση το πρωτόκολλο SIP και χρησιμοποιεί SIP Communicator για το πρόγραμμα πελάτη και JAIN SIP Proxy για το πρόγραμμα εξυπηρέτησης. Εμείς καλούμαστε να υλοποιήσουμε τις εξής επεκτάσεις:

- **Εγγραφή χρήστη**: Κάθε χρήστης εγγράφεται στο σύστημα κατά την πρώτη επίσκεψή του, συμπληρώντας τα προσωπικά του στοιχεία στην κατάλληλη φόρμα που του εμφανίζεται, τα οποία στη συνέχεια αποθηκεύονται στη βάση δεδομένων. Στη συνέχεια ο χρήστης μπορεί να εισέλθει στο σύστημα.

- **Λειτουργία Blocking**: Επιτρέπεται στο χρήστη να δημιουργήσει μια λίστα με άλλους χρήστες, με τους οποίους δεν θέλει να επικοινωνήσει. Αυτή η λίστα αποθηκεύεται στη βάση δεδομένων, με αποτέλεσμα όταν κάποιος χρήστης προσπαθήσει να καλέσει κάποιον που τον έχει μπλοκάρει, απορρίπτεται η κλήση του και λαμβάνει κατάλληλο μήνυμα.

- **Λειτουργία Billing**: Ο χρήστης μπορεί να καλέσει κάποιον άλλο χρήστη, πληρώνοντας το κατάλληλο ποσό με βάση την τιμολογιακή πολιτική, που ορίζει πως δεν υπάρχει ελάχιστη χρέωση και ο καθένας πληρώνει ανάλογα με το χρόνο ομιλίας του.

- **Λειτουργία Forwarding**: Ο χρήστης έχει τη δυνατότητα να προωθεί όλες τις εισερχόμενες κλήσεις σε κάποιον άλλο χρήστη.

1.2 Resources - References

[1] RFC 3261 - SIP: Session Initiation Protocol: http://www.ietf.org/rfc/rfc3261.txt

[2] mycourses: https://mycourses.ntua.gr/

2 Major Design Decisions

Οι αποφάσεις που πήραμε σχετικά με τις επεκτάσεις που θα υλοποιήσουμε είναι οι εξής:

- Κατά την είσοδο του χρήστη στην εφαρμογή θα του εμφανίζεται η κατάλληλη φόρμα για να εισάγει τα στοιχεία του και η επιλογή για sign up ή log in. Την πρώτη φορά που ο χρήστης εισέρχεται στην εφαρμογή πρέπει να κάνει sign up. Για το sign up ζητούμε διάφορα στοιχεία του χρήστη τα οποία αποθηκεύονται στη βάση δεδομένων στην οποία έχει πρόσβαση ο SIP Proxy Server. Τις επόμενες φορές που ο χρήστης προσπαθεί να εισέλθει στην εφαρμογή του ζητούνται ορισμένα στοιχεία ταυτοποιήσης, από αυτά που έδωσε κατά το sign up, και έπειτα μπορεί να εισέλθει, αν αυτά είναι σωστά.

- Για την blocking λειτουργία ο χρήστης δίνει, μέσω του SIP Communicator, μία λίστα από ονόματα χρηστών που θέλει να φράξει. Αυτή η λίστα αποθηκεύεται στη βάση δεδομένων του συστήματος στην οποία έχει πρόσβαση μόνο ο SIP Proxy Server.

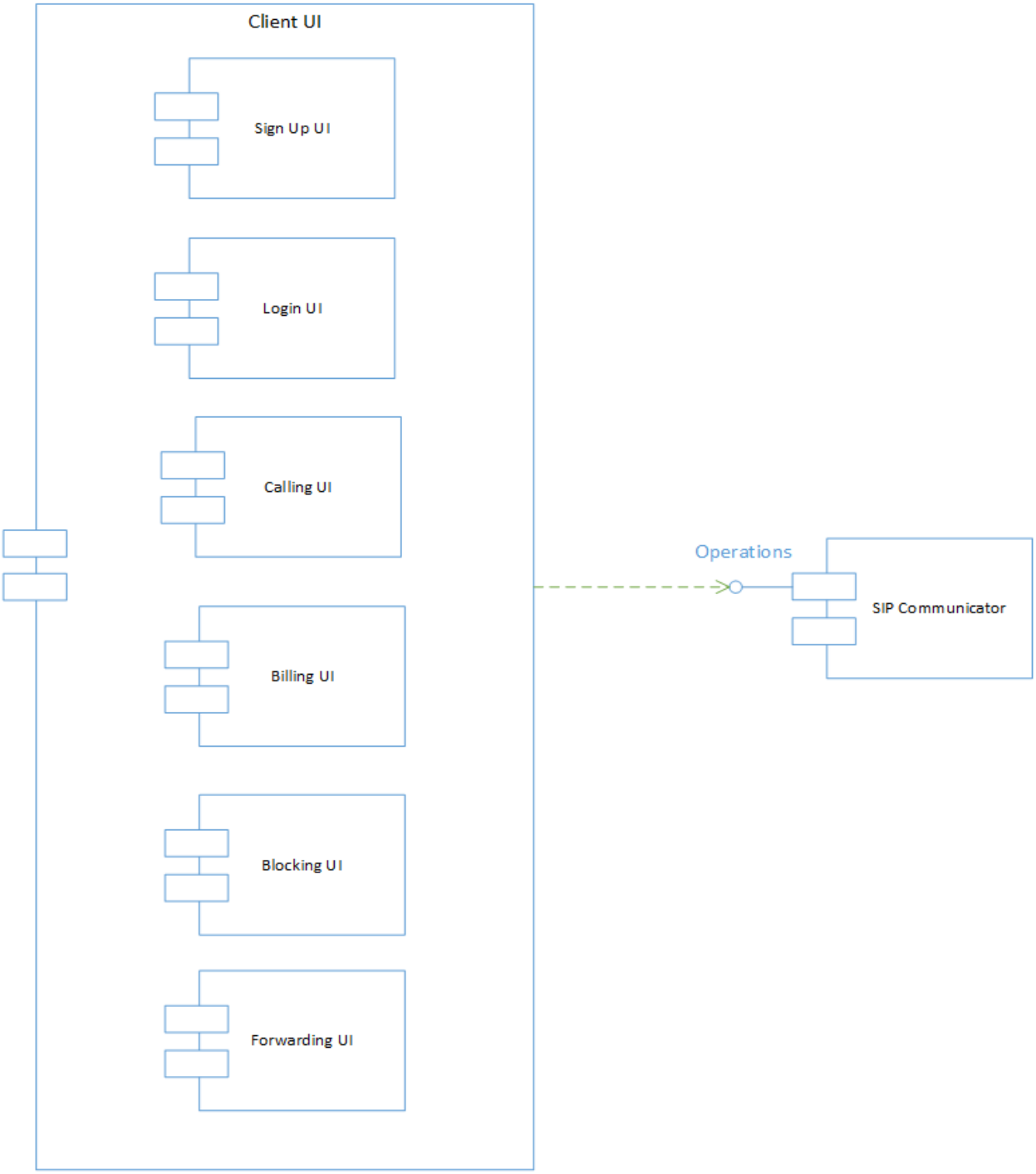
- Για τη λειτουργία billing, ο χρήστης έχει τη δυνατότητα να επιλέξει την τιμολογιακή πολιτική μέσω του SIP Communicator και η πολιτική αποθηκεύεται στη βάση δεδομένων. Η χρέωση της κλήσης γίνεται αμέσως μετά την ολοκλήρωση της, όταν ολοκληρωθεί η κλήση ο SIP Proxy Server ενημερώνεται μέσω των user agents και η χρέωση αποθηκεύεται στη βάση δεδομένων. Αφού η χρέωση γίνεται αμέσως μετά την κλήση, η αλλαγή της τιμολογιακής πολιτικής δεν επιρεάζει τις προηγούμενες χρεώσεις.

- Για την λειτουργία forwarding ο χρήστης δίνει, μέσω του SIP Communicator, τον χρήστη στον οποίο θέλει να προωθήσει τις κλήσεις του και αυτό αποθηκεύεται στη βάση δεδομένων, όπου θα έχει πρόσβαση μόνο ο SIP Proxy Server.

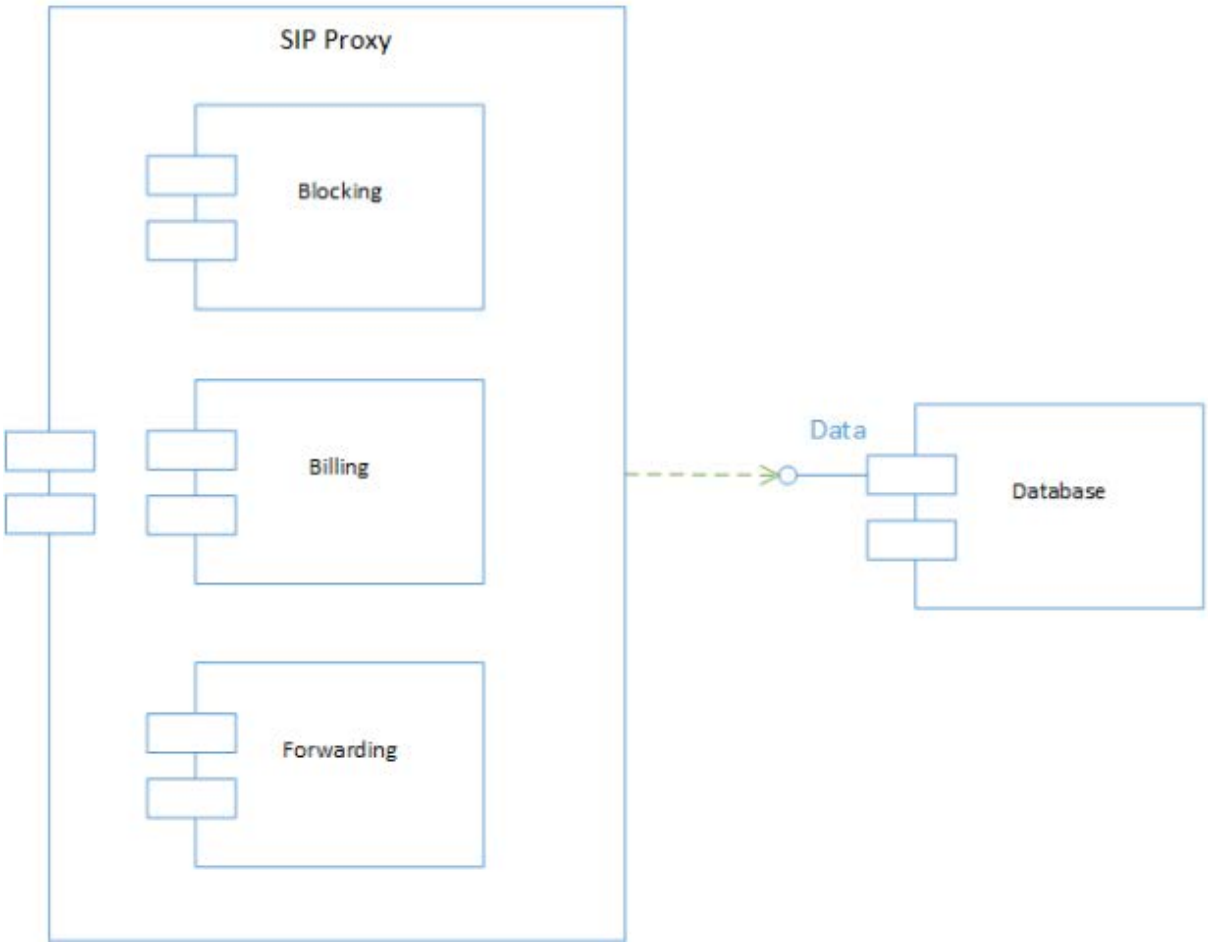
3 Architecture

3.1 Ψηφιδικά διαγράμματα

Ψηφιδικό διάγραμμα πελάτη – Sip Communicator:

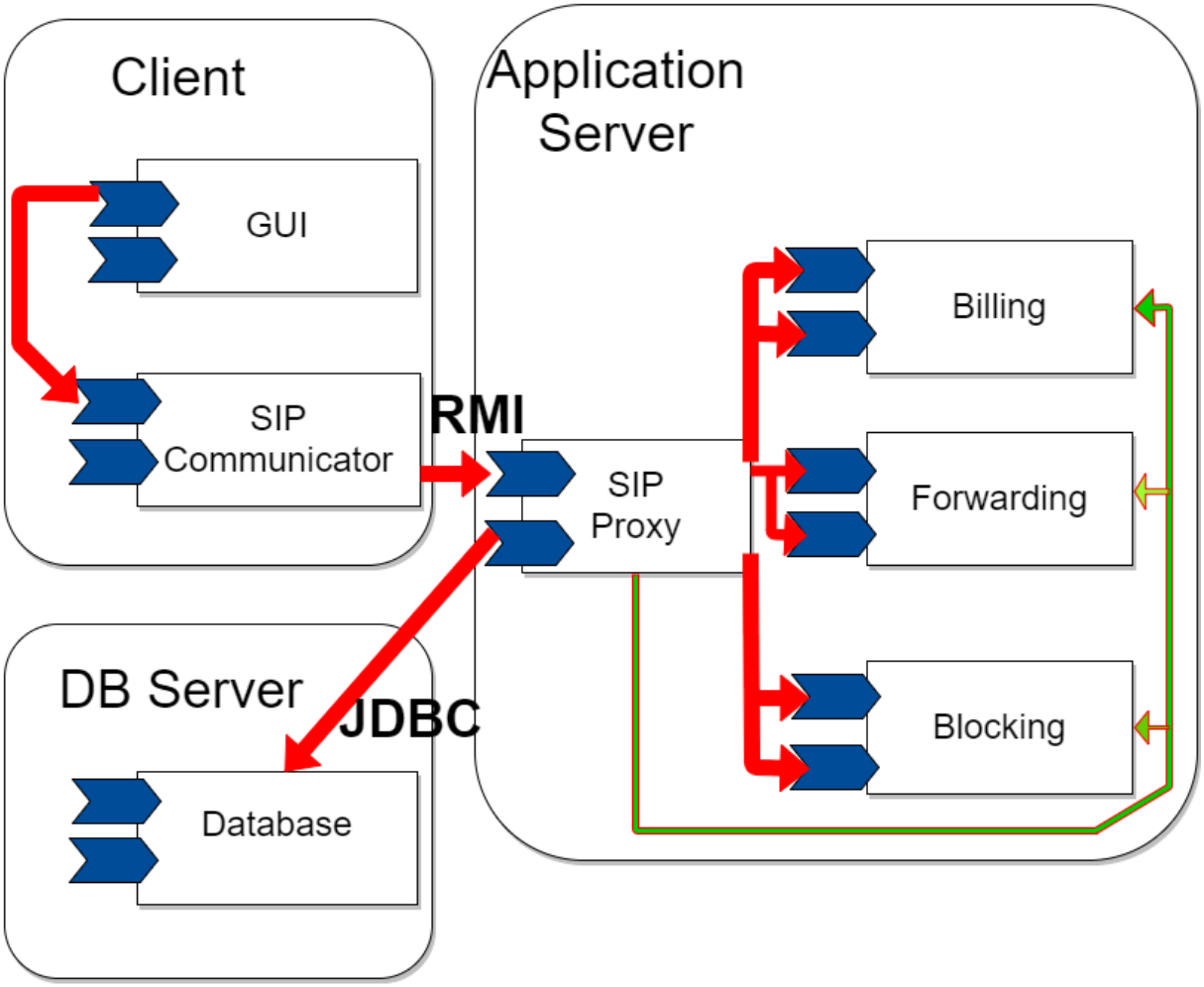


Ψηφιδικό διάγραμμα Server – SIP proxy



3.2 Παραταξιακό διάγραμμα

Το παραταξιακό διάγραμμα αποτελεί μία εικόνα του συστήματος κατά την εκτέλεση.

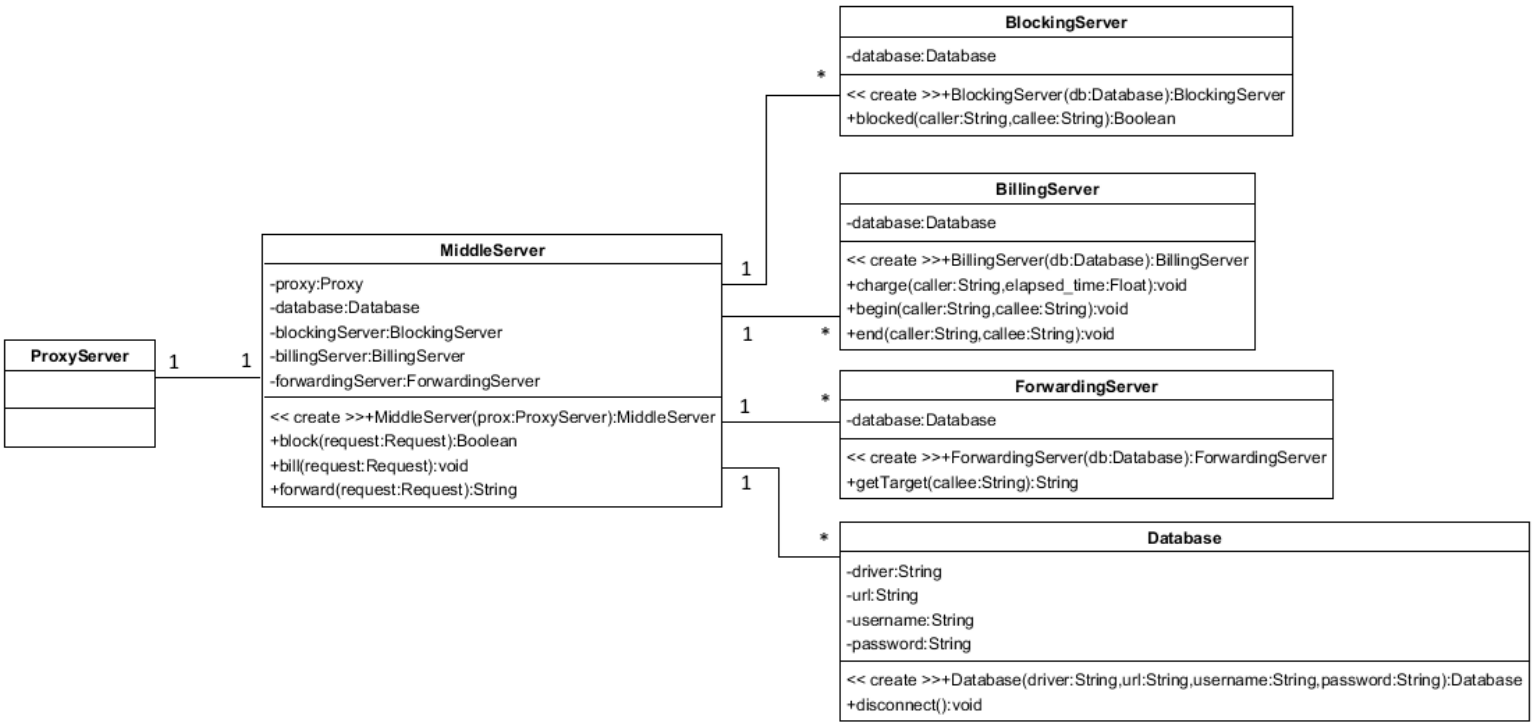


4 Detailed Class Diagrams

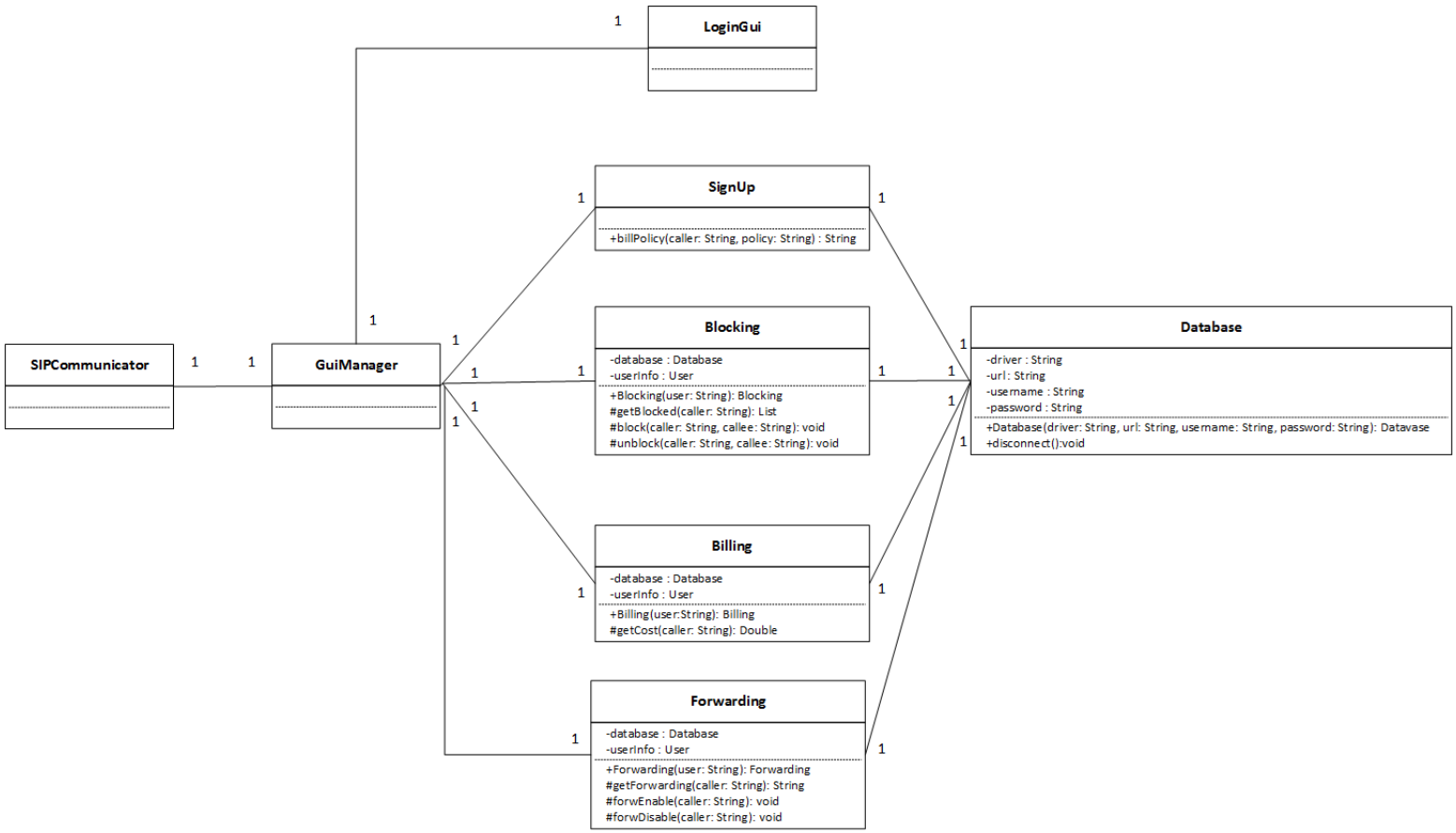
4.1 UML Class Diagrams

Στη συνέχεια παρουσιάζονται δυο διαγράμματα κλάσεων UML. Το πρώτο αφορά τον Εξυπηρετητή – SIP Proxy Server και το δεύτερο τον Πελάτη - SIP Communicator.

Εξυπηρετητής – SIP Proxy Server



Η υλοποιήση των πρόσθετων λειτουργιών γίνεται με βάση τη σχεδιαστική αρχιτεκτονική 3-Tier. Πιο συγκεκριμένα, δημιουργούμε μια ενδιάμεση κλάση (MiddleProxy) η οποία αποτελεί την καρδιά της υλοποίησης μας και έχει ως στόχο το συντονισμό και τη σωστή λειτουργία όλων των επιμέρων κλάσεων (BlockingServer, BillingServer, ForwardingServer, Database). Αυτός ο τρόπος σχεδιάσης είναι επιθυμητός, καθώς κάποια αλλάγη που ίσως χρειαστεί να γίνει σχετικά με κάποια πρόσθετη λειτουργία, θα πραγματοποιείται χωρίς να επηρεάζονται τα υπόλοιπα μέρη της εφαρμογής.



Οι κλάσεις SIPCommunicator, GuiManager, LoginGui χρησιμοποιούνται όπως μας δίνονται γι' αυτό και δεν έχουμε συμπληρώσει attibutes και operations. Η κλάση SignUp είναι επίσης ίδια με αυτή που δίνεται με την επιπρόσθετη μέθοδο billPolicy που περιγράφουμε παρακάτω.

4.2 Method Details

Σε αυτή την ενότητα παρέχουμε κάποια ανάλυση για τις κλάσεις που αναφέρονται στα παραπάνω διαγράμματα, καθώς και ψευδοκώδια υψηλού επιπέδου για τις μεθόδους τους.

Εξυπηρετητής – SIP Proxy Server

Κλάση Database:

Η δουλειά αυτής της κλάσης είναι να συνδεθεί με τη βάση δεδομένων, έτσι ώστε να μπορούν οι υπόλοιπες κλάσεις να χρησιμοποιήσουν τη σύνδεση αυτή με σκοπό να επικοινωνούν με τη βάση δεδομένων, γεγονός απαραίτητο για την πλήρωση των λειτουργιών τους. Η σύνδεση με τη βάση δεδομένων γίνεται από τον Constructor της κλάσης αυτής.

```
public Database(driver, url, username, password){
    connection = connect_with_database(driver, url, username, password)
}
```

Επίσης, υλοποιείται μια μέθοδος που τερματίζει τη σύνδεση με τη βάση δεδομένων.

```
public disconnect(){
    terminate_connection(connection)
}
```

Στη συνέχεια ακολοθούν οι κλάσεις BlockingServer, BillingServer και ForwardingServer. Οι Consructors των κλάσεων αυτών θέτουν τη βάση δεδομένων για χρήση.

Κλάση BlockingServer

Αυτή η κλάση ελέγχει αν ο καλών είναι στη λίστα των επαφών που ο καλούμενος έχει μπλοκάρει.

```
public BlockingServer(db){
    database = db
}

public blocked(caller, callee){
    list = get_the_list_of_blocked_contacts_from_database(callee)
    if caller in list
        return true
    else
        return false
}
```

Κλάση BillingServer:

Αυτή η κλάση υπολογίζει τη χρέωση που πρέπει να λάβει ο καλών με βάση κάποια τιμολογιακή πολιτική . Πιο συγκεκριμένα καταγράφει την έναρξη και την λήξη της κλήσης και χρεώνει τον καλών το κατάλληλο ποσό.

```
public BillingServer(db){
        database = db
}

public void charge(caller, elapsed_time){
          toPay = policy(elapsed_time);
          creditsLeft = get_sql_database(caller);
          newCreditsLeft = creditsLeft - toPay;
          update_database(caller, newCreditsLeft);
}

public void begin(caller, callee){
        start_time = timestamp()
        insert_in_database(caller, callee)
}

public void end(caller, callee){
        stop_time = timestamp()
        elapsed_time = stop_time - start_time
        delete_from_database(caller, calle)
        charge(caller, elapsed_time)
}
```

Κλάση ForwardingServer:

Σε αυτή την κλάση ελέγχεται αν ο καλούμενος έχει επιλέξει κάποιο άλλο άτομο, έτσι ώστε οι κλήσεις που είναι για αυτόν, να προωθούνται στο άλλο άτομο. Η ύπαρξη κύκλων δεν ελέγχεται εδώ.

```
public ForwardingServer(db){
        database = db
}
```

```
public String getTarget(callee){
        ptr = find_callee_in_ForwardingTable
        if ptr == NULL
                return NULL
        else
                return forwards(ptr)
}
```

Κλάση MiddleServer:

Αυτή η κλάση αποτελεί τον συνδετικό κρίκο ανάμεσα στον Proxy Server, τη βάση δεδομένων και τους υπόλοιπους servers. Ουσιαστικά, προωθεί τα αιτήματα που δέχεται στην κατάλληλη κλάση για την επεξεργασία τους. Επιπλέον, ελέγχει και επιλύει την περίπτωση ύπαρξης κύκλου κατά τη λειτουργία forwarding.

```
public MiddleServer(prox){
        proxy = prox
}

public Boolean block(request) {
          caller = get_caller(request)
          callee = get_callee(request)
          blockServer = new BlockingServer(database)
          return blockServer.blocked(caller, callee)
}

public void bill(request)  {
          caller = get_caller(request)
          callee = get_callee(request)
          message = get_message_from_request(request)
          billingServer = new BillingServer(database)
          if  message == 'START'
                billingSever.begin(caller, callee)
          else if  message == 'BYE'
                billingSever.end(caller, callee)
}
```

```
public String forward(request) {
        caller = get_caller(request)
        callee = get_callee(request)
        forwardServer = new ForwardingServer(database)
        target = getForward(callee)
        if  target == NULL
            return NULL
        else {
            while (target != NULL ) {
                temp = find_if _already_exists(forward_list, target)
                if (temp == true)
                    return 'cycle'
                else {
                    add_to_list(forward_list, target)
                    newCallee = target
                    target = forwardServer.getTarget(newCallee)
                }
                return  newCallee
            }
        }
}
```

Πελάτης – SIP Communicator

Κλάση Database:

Η κλάση αυτή υλοποιείται με τον ίδιο τρόπο με παραπάνω.

Κλάση SignUp:

Η κλάση αυτή υλοποιεί το GUI για την πρώτη εγγραφή του χρήστη. Προσθέσαμε τη μέθοδο billPolicy με την οποία επιλέγεται η πολιτική χρέωσης από τον χρήστη.

```
public void billPolicy(caller, policy) {
    insert_in_database(caller, policy);
}
```

Κλάση Blocking:

Η κλάση αυτή υλοποιεί το GUI όπου φαίνονται οι μπλοκαρισμένοι χρήστες και υπάρχει η δυνατότητα να προστεθούν επιπλέον χρήστες στη λίστα των μπλοκαρισμένων ή να διαγραφούν από αυτή.

```
public Blocking(user) {
        database = new Database();
        userInfo = user;
}

protected List getBlocked(caller) {
        blocked_list = get_blocked(caller);
        return blocked_list;
}

protected void block(caller, callee) {
        insert_in_database (caller, callee);
}

protected void unblock (caller, callee) {
        delete_from_database(caller, callee);
}
```

Κλάση Billing:

Η κλάση αυτή υλοποιεί το GUI όπου ο χρήστης μπορεί να βλέπει το ποσό που οφείλει από τις κλήσεις που πραγματοποίησε.

```
public Billing(user) {
        database = new Database();
        userInfo = user;
}

protected Double getCost(caller) {
        cost = get_cost(caller);
```

```
        return cost;
}
```

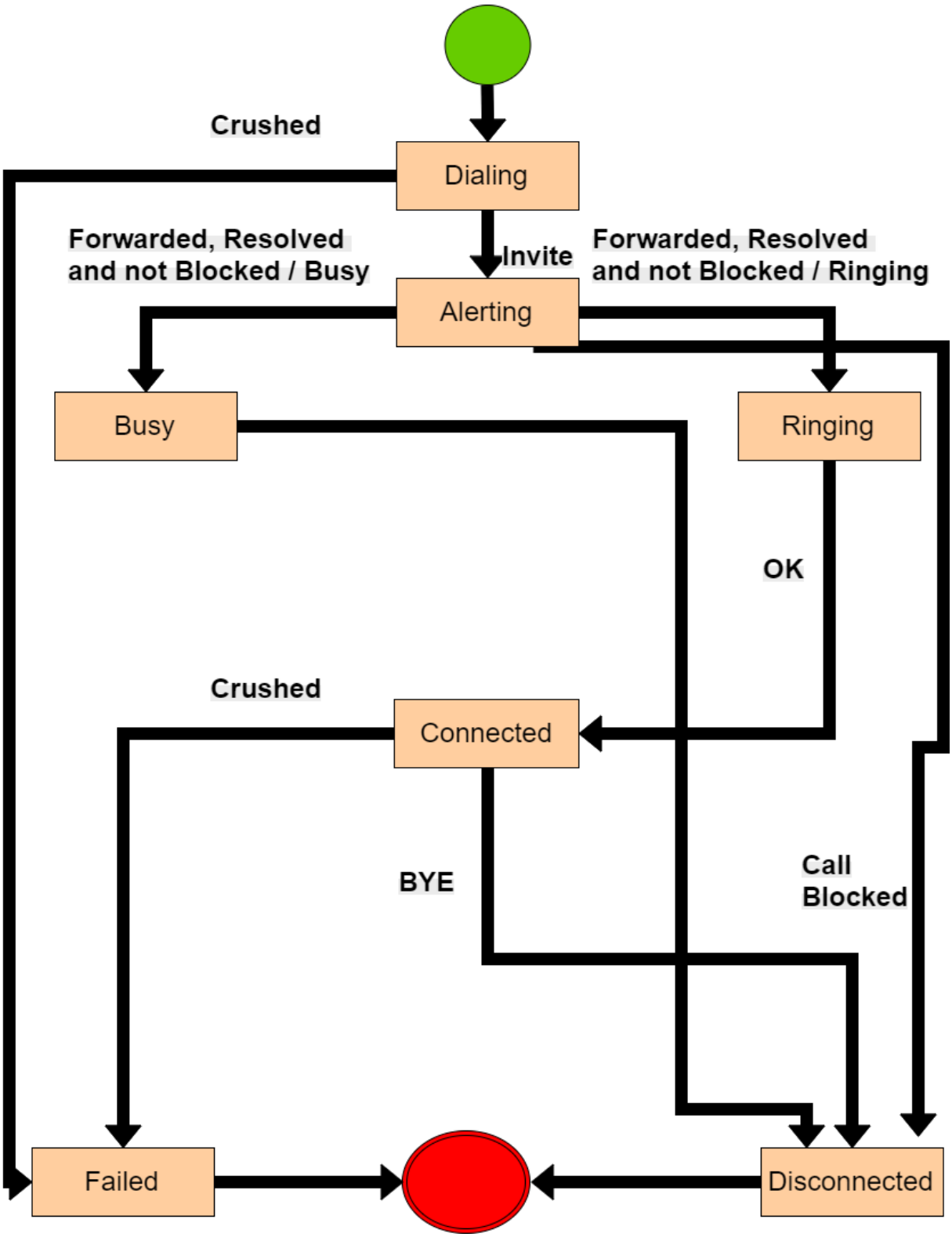
Κλάση Forwarding:

Η κλάση αυτή υλοποιεί το GUI όπου ο χρήστης μπορεί να ενεργοποιήσει και να απενεργοποιήσει την προώθηση κλήσεων, να δει σε ποιον χρήστη προωθούνται οι κλήσεις του και, αν επιθυμεί, να τον αλλάξει.

```
public Forwarding(user) {
        database = new Database();
        userInfo = user;
}

protected String getForwarding(caller) {
        target = get_target(caller);
        return target;
}

protected void forwEnable(caller, callee) {
        insert_in_database(caller, callee);
}

Protected void forwDisable(caller, callee) {
        delete_from_database(caller, calle);
}
```

5 State Diagrams

Το διάγραμμα κατάστασης για μία κλήση είναι το παρακάτω:



6 Open Issues

Μια επέκταση θα ήταν να έχει τη δυνατότητα ο καλών, σε περίπτωση που δεν απαντήσει ο καλούμενος, να του αφήσει κάποιο φωνητικό μήνυμα. Επιπλέον, ενδιαφέρον θα είχε η επέκταση κατά την οποία η επικοινωνία δεν περιορίζεται μεταξύ 2 ατόμων, αλλά περισσότεροι θα μπορούσαν να συμμετέχουν σε μια ομαδική κλήση. Αυτό βέβαια θα απαιτούσε να γίνουν αρκετές αλλαγές στην εφαρμογή μας.

7 Domain Dictionary

7.1 Terms and Abbreviations

Term	Definition
RMI	Remote Method Invocation
JDBC	Java Database Connectivity
GUI	Graphical User Interface