

Καρατζάς Δημήτρης icsd13072
Λάζαρος Απόστολος icsd13096

1η ΟΜΑΔΙΚΗ ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ ΚΑΤΑΝΕΜΗΜΕΝΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΥΛΟΠΟΙΗΣΗ

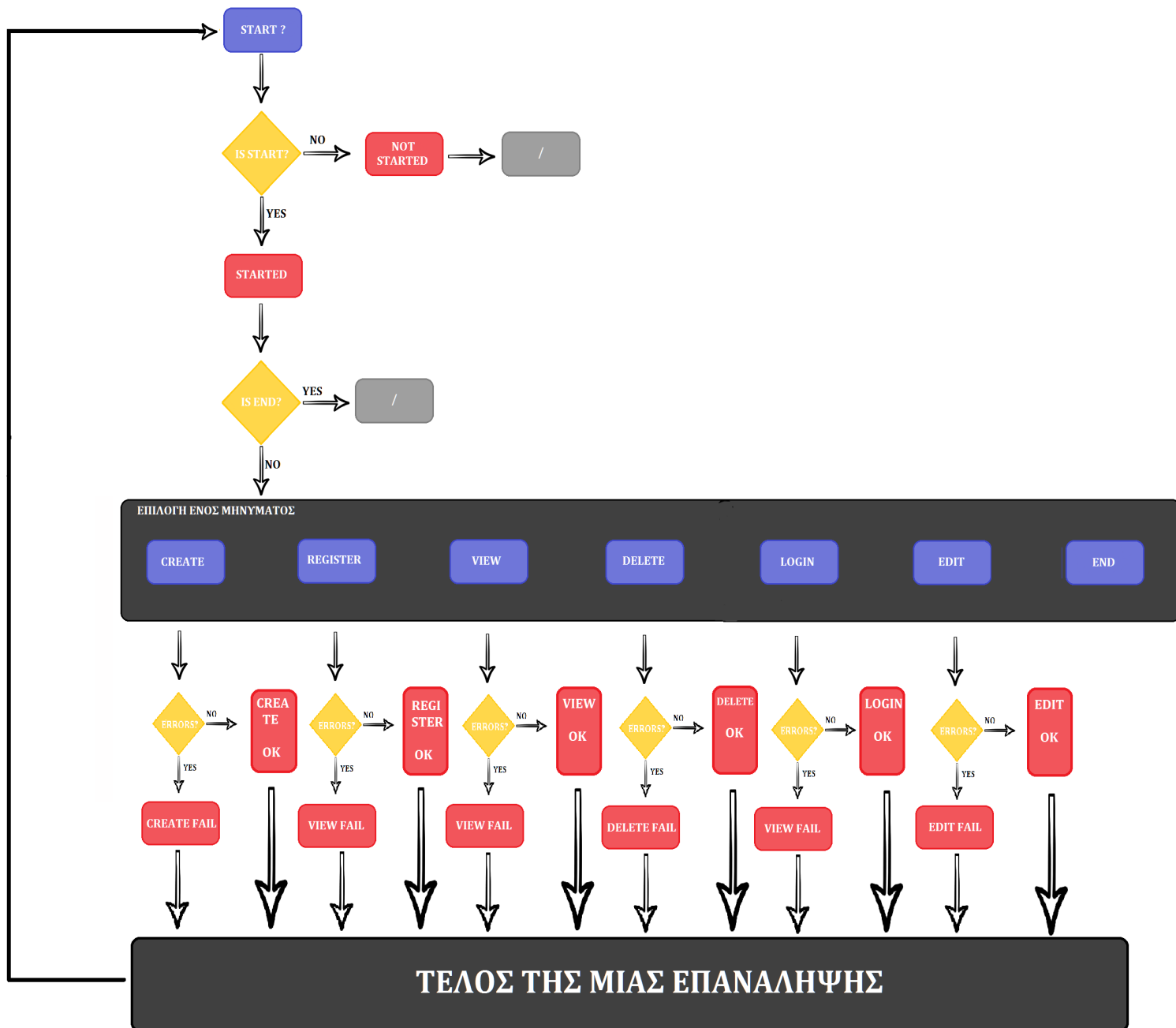
A) ΒΑΣΙΚΕΣ ΣΧΕΔΙΑΣΤΙΚΕΣ ΑΠΟΦΑΣΕΙΣ

Την εφαρμογή την υλοποιήσαμε χρησιμοποιώντας το πρόγραμμα NetBeans. Για την υλοποίηση της, έχουμε δημιουργήσει 2 projects, ένα για τον **client** και ένα για τον **server**. Επειδή συνδέονται πολλοί clients ταυτόχρονα στον server, χρησιμοποιήσαμε νήματα για να το επιτρέψουμε αυτό. Ο server και ο client επικοινωνούν μέσω Java sockets και τέλος ο server είναι αυτός που έχει τη λίστα με τους εγγεγραμμένους χρήστες.

B) ΠΡΩΤΟΚΟΛΛΟ ΕΠΙΚΟΙΝΩΝΙΑΣ CLIENT ΚΑΙ SERVER

Το πρωτόκολλο είναι το εξής: Πρώτα ο client πρέπει να ξεκινήσει τη σύνδεση, έτσι στέλνει ένα μήνυμα **START**. Ο server αποκρίνεται με **STARTED**. Αν ο client δε στείλει **START** αρχικά, τότε ο server στέλνει μήνυμα **NOT STARTED** και διακόπτει τη σύνδεση. Αν ο client θέλει να εισάγει κάποια ανακοίνωση, διαγράψει ή τροποποιήσει (δικιά του μόνο) πρέπει να ταυτοποιηθεί οπότε πρέπει να κάνει login. Δίνεται η δυνατότητα να κάνει register αν δεν έχει ήδη εγγραφεί. Για αυτό στέλνει ένα μήνυμα **LOGIN** ή **REGISTER**. Ο server θα απαντήσει με **LOGIN OK** ή **LOGIN FAIL** στη περίπτωση του login και **REGISTER OK** ή **REGISTER FAIL** αντίστοιχα. Στη συνέχεια ο χρήστης έχει 4 επιλογές (διαγραφή/δημιουργία/τροποποίηση/ανάγνωση ανακοινώσεων, αν δεν έχει κάνει login μπορεί μόνο να διαβάσει) οπότε και θα στείλει τα μηνύματα **DELETE**, **CREATE**, **EDIT**, **VIEW** αντίστοιχα. Πάλι θα υπάρχουν αντίστοιχα **OK** και **FAIL** μηνύματα για τις παραπάνω 4 ενέργειες. Πέρα από αυτές τις ενέργειες, ο client μπορεί να στείλει και το μήνυμα **END** όπου ο server θα διακόψει τη σύνδεση (δε θα στείλει στον client μήνυμα μετά). Κάθε μήνυμα θα περιέχει το μήνυμα και κάποια πληροφορία που χρειάζεται να σταλεί για το συγκεκριμένο μήνυμα εκτός από τα **END**, **START**, **STARTED** και **NOT STARTED**, π.χ για το **LOGIN** θα σταλούν τα στοιχεία χρήστη, για το **VIEW** θα σταλούν οι 2 ημερομηνίες (στον σερβερ) που ο χρήστης επιθυμεί να δει ανακοινώσεις στο διάστημα αυτό, ενώ για το **VIEW OK** θα σταλούν οι ανακοινώσεις αυτές στον client κτλ. Η διαδικασία επαναλαμβάνεται μέχρις όπου ο Client να στείλει μήνυμα **END**. Το πρωτόκολλο φαίνεται στην παρακάτω εικόνα, όπου με **Μπλε** πλαίσιο είναι τα μηνύματα του Client, με **Κόκκινο** πλαίσιο του server και με **Γκρι** ο τερματισμός της σύνδεσης.

(Σημείωση: Το μήνυμα **STARTED** του server είναι το μήνυμα με το οποίο ο server περιμένει κάποιο μήνυμα από τον client, δηλαδή μετά από αυτό το μήνυμα ξεκινάει η ανταλλαγή χρησιμικών δεδομένων. Επίσης στην εικόνα το μήνυμα "**STARTED ?**" είναι το μήνυμα με το οποίο ο client θέλει να ξεκινήσει τη σύνδεση, και αυτό αναμένει ο server, όμως δεν ξέρει ο server ακόμα αν είναι **START** οπότε πρέπει να ελεγχθεί. Τέλος, η εικόνα φαίνεται μόνο με zoom επειδή είναι πολύ μεγάλη και δε χωρούσε αλλιώς, εκτός και αν είχαμε παραπάνω από μια εικόνα όπου δε θα φαινόταν όλο το πρωτόκολλο.)





Γ) ΣΥΝΘΗΚΕΣ ΑΝΤΑΓΩΝΙΣΜΟΥ

Συνθήκες ανταγωνισμού εντοπίζονται μόνο όταν χρησιμοποιούμε νήματα. Συνθήκη ανταγωνισμού είναι όταν δύο ή παραπάνω νήματα προσπαθούν να γράψουν στο ίδιο στοιχείο ταυτόχρονα και έτσι έχουμε απρόσμενα αποτελέσματα, διότι το λειτουργικό σύστημα θα αποφασίσει τον χρονοπρογραμματισμό, οπότε δεν είναι προβλέψιμο το αποτέλεσμα. Για να το αντιμετωπίσουμε αυτό πρέπει να συγχρονίσουμε τα νήματα. Στη Java ο συγχρονισμός γίνεται με τη λέξη κλειδί **synchronized**. Όταν συγχρονίζουμε ένα κομμάτι κώδικα, αυτό σημαίνει πως κανένα άλλο νήμα της κλάσης αυτής δε μπορεί να χρησιμοποιήσει το τμήμα αυτό, παρα μόνο το νήμα που τρέχει τώρα (δηλαδή δεν επιτρέπουμε παραλληλία, το αντίθετο από αυτό που πετυχαίνουμε με νήματα, οπότε πρέπει να το εφαρμόζουμε μόνο εκεί που πρέπει). Έτσι, πρέπει να δηλώσουμε τα τμήματα κώδικα ως **synchronized** που υπάρχει πιθανότητα δυο νήματα να κάνουν ταυτόχρονα write σε έναν κοινό πόρο, οπότε στην άσκηση αυτή συγχρονίσαμε τα τμήματα κώδικα που:

- Κάνουν **διαγραφή** κάποιας ανακοίνωσης.
- Κάνουν **εισαγωγή** κάποιας ανακοίνωσης.
- Κάνουν **εισαγωγή** κάποιου χρήστη (register).
- **Τροποποιούν** κάποια ανακοίνωση.

Παραπάνω λεπτομέρεις φαίνονται και στα σχόλια του κώδικα του server.

Δ) ΠΑΡΑΔΟΧΕΣ

- Δεν έχουμε βάλει **Log out** button για αποσύνδεση, διότι υποθέσαμε πως ένας χρήστης όταν κλείσει την εφαρμογή (το JFrame δηλαδή) θα θέλει να αποσυνδεθεί, οπότε τη στιγμή που πατιέται το κουμπί  του παραθύρου ενεργοποιείται listener για αποσύνδεση του client και κλείσιμο socket και streams. Υπάρχει έλεγχος για το αν είναι Logged in, δηλαδή αν πατήσει το κουμπί  και δεν είναι logged in, η εφαρμογή απλώς κλείνει τη σύνδεση και δεν προσπαθεί να κάνει log out έναν χρήστη που δεν είναι logged in.
- Ο σερβερ κανονικά πρέπει να αποθηκεύει τα δεδομένα του σε κάποιο μέσο που διατηρεί τα δεδομένα ακόμα και μετά το κλείσιμο, π.χ κάποια Βάση Δεδομένων. Εμείς χρησιμοποιήσαμε Java ArrayLists που δεν το κάνουν αυτό (δηλαδή χάνονται τα δεδομένα μετά το κλείσιμο, αφού τα ArrayLists δουλεύουν στη RAM) που για έναν σερβερ δεν είναι καλή λύση, αλλά για απλότητα κάναμε αυτή τη παραδοχή.

Ε) ΟΔΗΓΙΕΣ ΓΙΑ ΤΗΝ ΕΚΤΕΛΕΣΗ ΤΗΣ ΕΦΑΡΜΟΓΗΣ

Server: Η εφαρμογή του server ξεκινάει και λειτουργεί αυτόματα, δεν χρειάζεται κάποια συγκεκριμένη διευκρίνιση.

Client: Ο Χρήστης της εφαρμογής πρέπει να δημιουργήσει λογαριασμό για να μπορεί να δημιουργήσει, τροποποιήσει και να διαγράψει κάποια (δικιά του μόνο) ανακοίνωση, εκτός από τους **administrators** οι οποίοι μπορούν να κάνουν τις παραπάνω ενέργειες σε οποιαδήποτε ανακοίνωση. Αν ο χρήστης επιθυμεί να αναζητήσει ανακοινώσεις, δε χρειάζεται να υπάρχει λογαριασμός.

Συγκεκριμένα:

- **Register:** Επιλογή της καρτέλας για register και εισαγωγή των στοιχείων χρήστη. Συγκεκριμένα username, password, όνομα και επίθετο. Αν το username υπάρχει ήδη στη βάση του σερβερ τότε δεν επιτρέπεται το register.
- **Login:** Επιλογή της καρτέλας και εισαγωγή username και password. Αν ο server ελέγξει τα στοιχεία και όντως είναι σωστά, τότε επιτρέπει το login.
- **Εισαγωγή ανακοίνωσης:** Η καρτέλα αυτή ενεργοποιείται μόνο όταν γίνει επιτυχημένα το login. Εδώ ο χρήστης γράφει την ανακοίνωση του, και πατώντας το κουμπί στο κάτω μέρος του παραθύρου στέλνει την ανακοίνωση στον σερβερ.
- **Διαγραφή/Εισαγωγή ανακοίνωσης:** Η καρτέλα είναι μια για εισαγωγή και διαγραφή. Πρώτα ο server στέλνει τη λίστα με τις ανακοινώσεις για τον χρήστη αυτόν. Έπειτα ο χρήστης μπορεί να επιλέξει το checkbox δίπλα από κάθε ανακοίνωση για να τις διαγράψει. Εφόσον σιγουρευτεί για τις επιλογές του, πατάει το κουμπί διαγραφής για να διαγραφούν οι ανακοινώσεις. Για να τροποποιήσει τις ανακοινώσεις, απλά επιλέγει όποια ανακοίνωση θέλει και γράφει σε αυτή το αλλαγμένο κείμενο. Πατώντας το αντίστοιχο κουμπί της τροποποίησης θα τροποποιηθούν οι ανακοινώσεις.

Ο χρήστης της εφαρμογής δε χρειάζεται να πατήσει κάποιο κουμπί για Logout. Κλείνοντας το παράθυρο της εφαρμογής, η εφαρμογή τον κάνει αυτόματα Logout ή απλά κλείνει την εφαρμογή αν ο χρήστης δεν έχει συνδεθεί ακόμα.

ΣΤ) ΟΘΟΝΕΣ ΕΚΤΕΛΕΣΗΣ

The screenshot shows a web application window titled "Reservations". It has a navigation bar with five tabs: "Register", "Login", "View Announcements", "Insert announcement", and "Edit/Delete Your Announcements". The "Register" tab is currently selected. Below the tabs, there is a registration form with the following fields: "Username:", "Password:", "Name:", and "Last Name:". Each field has a corresponding text input box. At the bottom of the form, there is a "Register" button. The window also features standard window controls (minimize, maximize, close) in the top right corner.

Reservations

Register

Login

View Announcements

Insert announcement

Edit/Delete Your Announcements

Username:

Password:

Login

Reservations

Register

Login

View Announcements

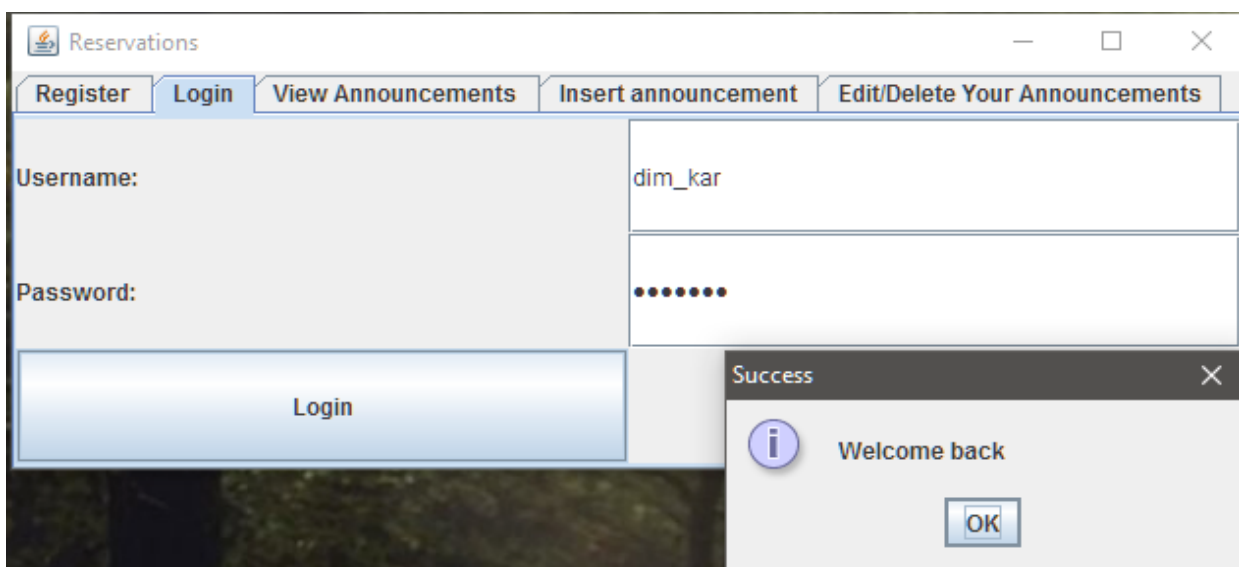
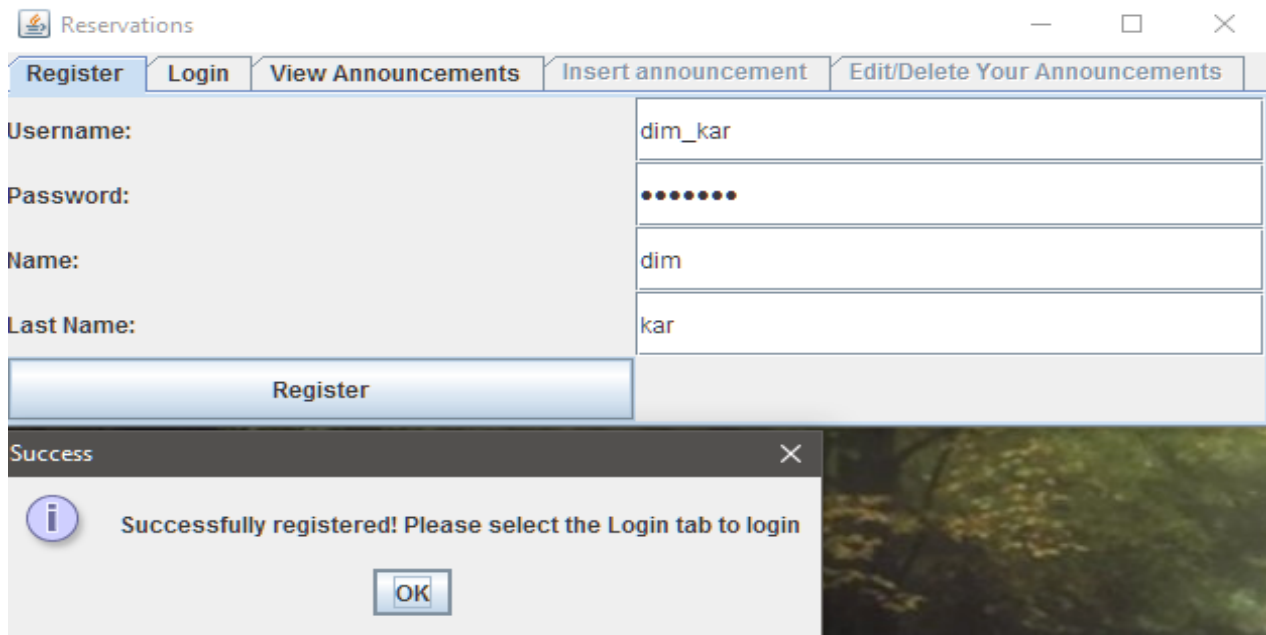
Insert announcement

Edit/Delete Your Announcements

Search Dates from (dd-MM-yyyy only):

Search Dates to (dd-MM-yyy only):

Search



Παρατήρηση σε αυτήν την εικόνα: Εδώ βλέπουμε πως τα tabs **εισαγωγής** και **διαγραφής/τροποποίησης** ενεργοποιούνται αυτόματα όταν γίνει το Login.

Reservations

Register

Login

View Announcements

Insert announcement

Edit/Delete Your Announcements

Search Dates from (dd-MM-yyyy only):

15-04-2016

Search Dates to (dd-MM-yyy only):

18-04-2016

Search

Search results

Author: admin1
Announcement: First announcement!
Last Edit: Sun Apr 17 16:12:42 EEST 2016

Author: admin1
Announcement: Second announcement!
Last Edit: Sun Apr 17 16:12:48 EEST 2016

Author: admin1
Announcement: Third announcement!
Last Edit: Sun Apr 17 16:14:05 EEST 2016

Author: admin1
Announcement: Fourth announcement!
Last Edit: Sun Apr 17 16:14:10 EEST 2016

Author: admin1
Announcement: Fifth announcement!
Last Edit: Sun Apr 17 16:14:20 EEST 2016

Σε αυτή την εικόνα οι ανακοινώσεις που ψάχνει ο client εμφανίζονται σε ένα νέο παράθυρο, επίσης εκτός της ημερομηνίας εμφανίζεται και η ώρα υποβολής της ανακοίνωσης.

Reservations

Register

Login

View Announcements

Insert announcement

Edit/Delete Your Announcements

First announcement!

Success

i

Successfully added your announcement

OK

Create

Reservations

Register

Login

View Announcements

Insert announcement

Edit/Delete Your Announcements

Check to delete	Announcement	Last Edit Date
<input checked="" type="checkbox"/>	Second!	Sun Apr 17 22:08:44 EEST 2016
<input type="checkbox"/>	Fourth!	Sun Apr 17 22:12:18 EEST 2016
<input checked="" type="checkbox"/>	First!	Sun Apr 17 22:29:37 EEST 2016
<input type="checkbox"/>	Second!	Sun Apr 17 22:29:37 EEST 2016
<input checked="" type="checkbox"/>	Third!	Sun Apr 17 22:29:37 EEST 2016
<input type="checkbox"/>	Fourth!	Sun Apr 17 22:29:37 EEST 2016

Delete

Update

Reservations

Register

Login

View Announcements

Insert announcement

Edit/Delete Your Announcements

Check to delete	Announcement	Last Edit Date
<input type="checkbox"/>	Second!	Sun Apr 17 22:08:44 EEST 2016
<input checked="" type="checkbox"/>	CHANGED!	Sun Apr 17 22:12:18 EEST 2016
<input checked="" type="checkbox"/>	First!	Sun Apr 17 22:29:37 EEST 2016
<input checked="" type="checkbox"/>	Second!	Sun Apr 17 22:29:37 EEST 2016
<input type="checkbox"/>	Third!	Sun Apr 17 22:29:37 EEST 2016
<input type="checkbox"/>	Fourth!	Sun Apr 17 22:29:37 EEST 2016

Delete

Update

Success

i

Successfully updated your announcements

OK

307

308

Message msg =

//έλεγχος

Reservations

Register Login **View Announcements** Insert announcement Edit/Delete Your Announcements

Search Dates from (dd-MM-yyyy only): 10-10-2015

Search Dates to (dd-MM-yyyy only): 12-10-2015

Search

Nothing found

Could not find any announcements

OK

Reservations

Register Login **View Announcements** Insert announcement Edit/Delete Your Announcements

Search Dates from (dd-MM-yyyy only): 14-04-2016

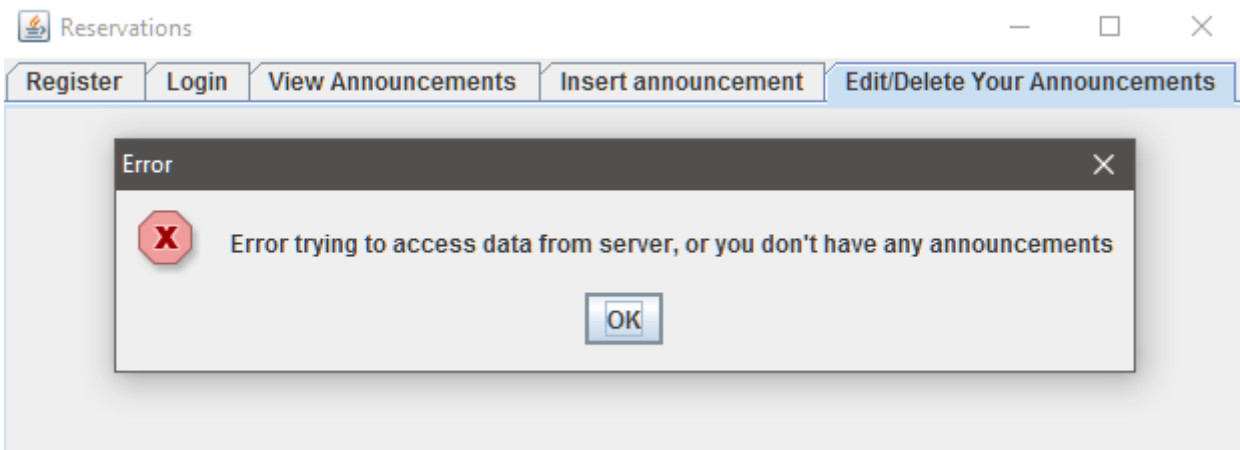
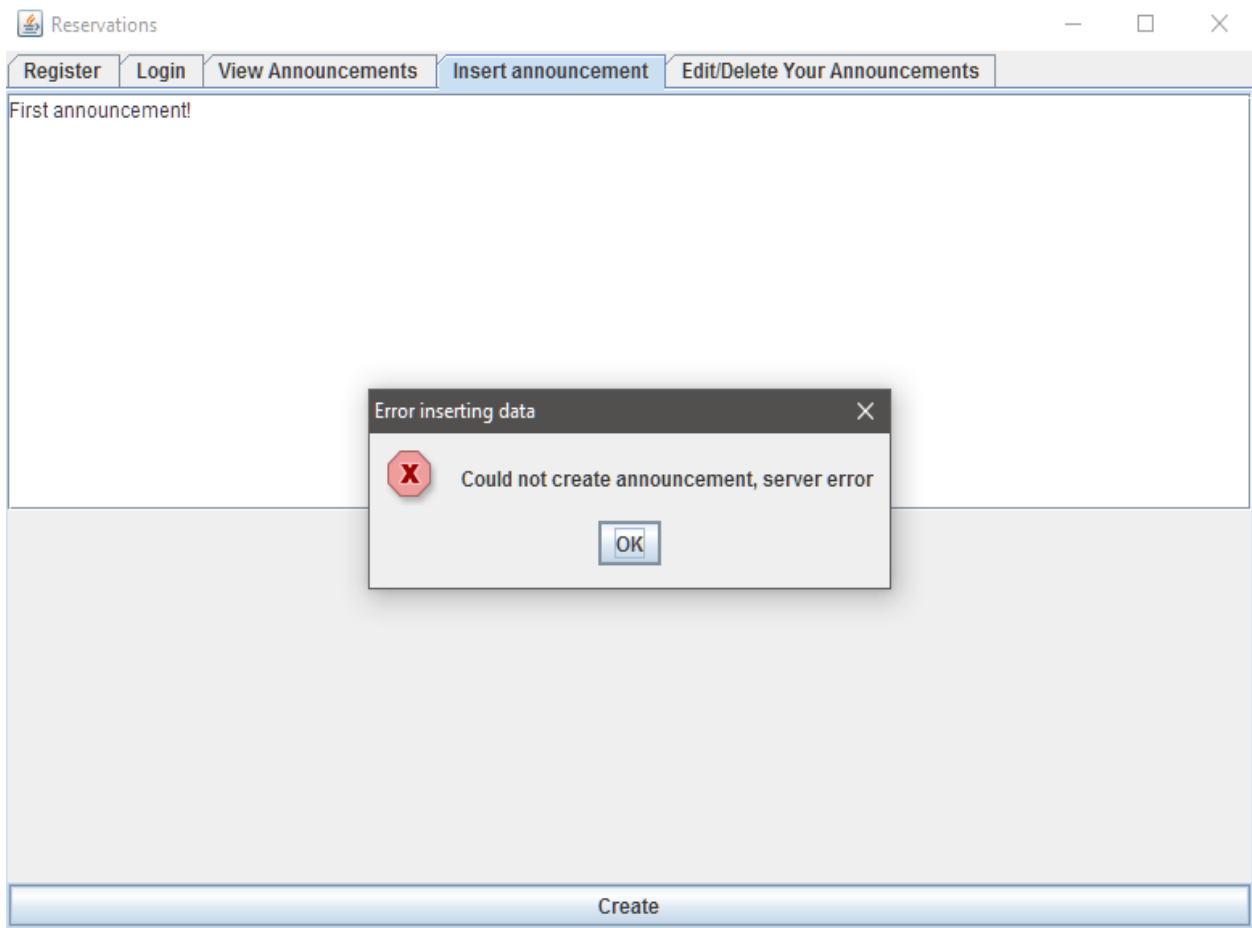
Search Dates to (dd-MM-yyyy only): 12-04-2016

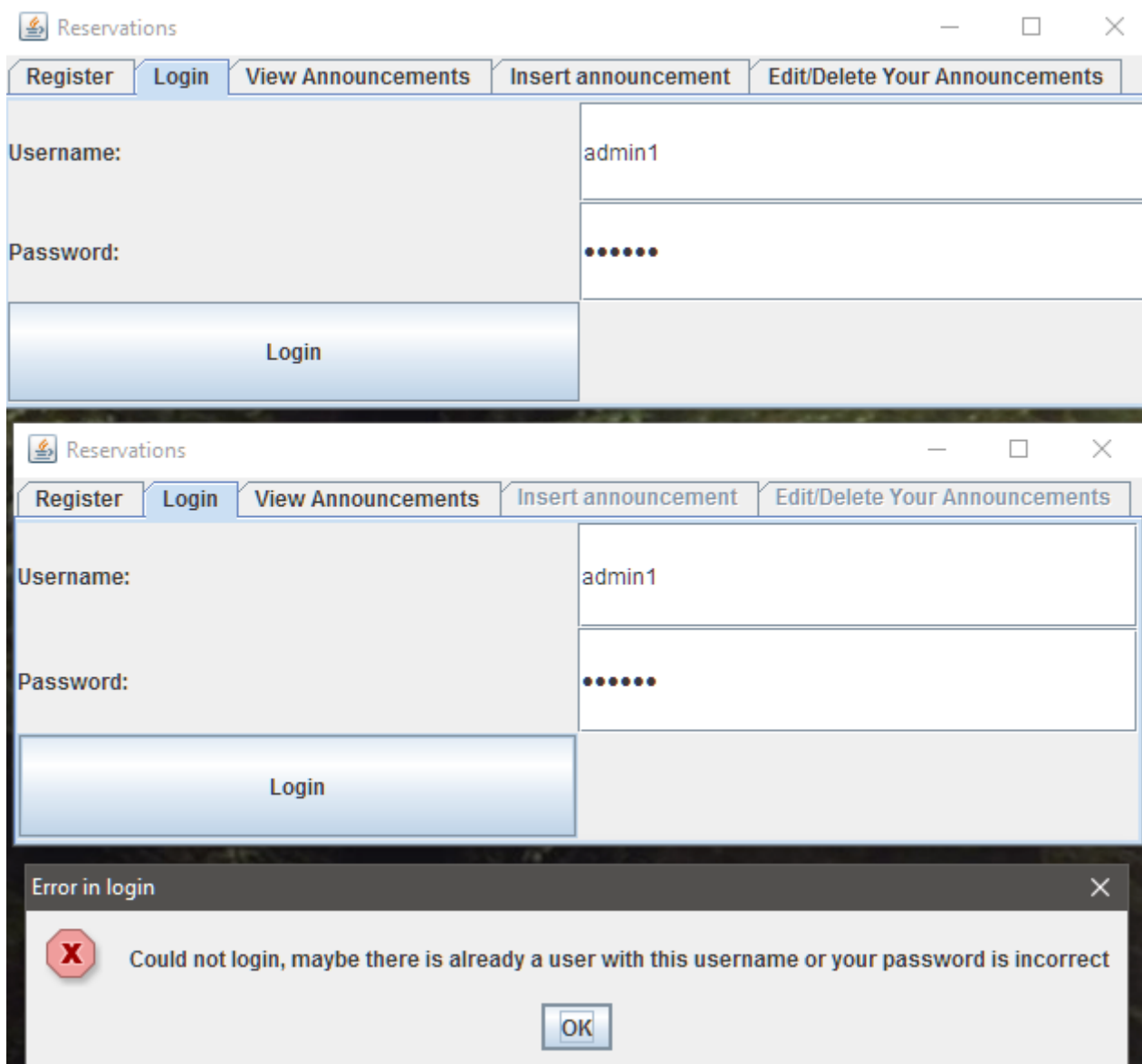
Search

Error searching data

Could not search announcements, maybe the first date is after the second?

OK





Από τις εικόνες, φαίνεται ότι το πρόγραμμα δουλεύει αρκετά καλά, για διάφορους συνδυασμούς, εντοπίζει λάθη και γενικά και ο Client και ο Server δουλεύουν αρκετά καλά, ο Client στέλνει τα δεδομένα που χρειάζονται καθώς επίσης ελέγχει αν οι τιμές είναι έγκυρες (όσον αφορά το κομμάτι των κανόνων, π.χ δε μπορεί ο κωδικός να είναι μικρότερος από 6 χαρακτήρες, τα πεδία να είναι κάποια κενά, η ημερομηνία έναρξης αναζήτησης να είναι μικρότερη της ημερομηνίας ορίου κτλ), ώστε να μη σταλούν άσκοπα στο δίκτυο αντικείμενα χωρίς λόγο και επίσης να μη φορτώνεται ο server και με άσκοπους ελέγχους που μπορεί να γίνουν στην πλευρά του πελάτη.

Ο Server ελέγχει αν τα (έγκυρα ως προς το συντακτικό τουλάχιστον κομμάτι) που στέλνει ο Client μπορούν να μπούν στις λίστες που κρατάει, ή απλώς ελέγχει αν υπάρχουν στις λίστες αυτές (ελέγχοι).

Τέλος η εφαρμογή δεν επιτρέπει login για τον ίδιο χρήστη, δηλαδή αν έχει κάνει ήδη login, και κάποια άλλη instance της εφαρμογής προσπαθήσει να κάνει login η εφαρμογή βγάζει σφάλμα, όπως στην παρακάτω εικόνα, αυτό φαίνεται γιατί το instance της εφαρμογής που προσπαθεί να κάνει login έχει απενεργοποιημένα τα tabs (δηλαδή δεν έχει κάνει login) αλλά εφόσον τρέχει ένα άλλο instance στο οποίο ο χρήστης αυτός είναι ήδη Logged in, η εφαρμογή βγάζει σφάλμα.

ΑΣΦΑΛΕΙΑ

Ένας επιτιθέμενος στο παραπάνω σύστημα client-server, μπορεί να εκμεταλλευτεί ευπάθειες του συστήματος ώστε να το παραβιάσει. Συγκεκριμένα τα δεδομένα που μεταφέρονται στο δίκτυο μεταξύ client και πελάτη (δηλαδή τα δεδομένα στα αντίστοιχα streams, οπότε στη συγκεκριμένη περίπτωση τα διάφορα αντικείμενα της κλάσης Message) δεν είναι **κρυπτογραφημένα**, δηλαδή μεταφέρονται αυτούσια στο δίκτυο και όχι η κρυπτογραφημένη μορφή τους η οποία είναι οπότε ένας επιτιθέμενος μπορεί να δει τα δεδομένα που μεταφέρονται στο μη ασφαλές κανάλι. Η λύση εδώ είναι η κρυπτογράφηση των δεδομένων, η Java προσφέρει τις δυνατότητες του πρωτοκόλλου **SSL** για αυτόν τον σκοπό με χρήση του JSSE (Java Secure Socket Extension), το οποίο πέρα από SSL παρέχει και **TLS**, τον διάδοχο του SSL.

Επίσης, οι κωδικοί αποθηκεύονται στον σερβερ όπως ακριβώς τους έβαλε ο χρήστης. Αν ένας επιτιθέμενος παραβιάσει τον σερβερ μόνο, τότε μπορεί να βρεί τους κωδικούς εύκολα οπότε το ασφαλές κανάλι SSL δε θα βοηθήσει. Για αυτό είναι σωστό να αποθηκεύονται οι κωδικοί ως συνάρτηση μιας τυχαίας συμβολοσειράς (salt) και μαζί με αυτή να γίνονται **hash**. Έτσι αν ο client δώσει τον κανονικό κωδικό του για Login, ο server θα ελέγξει το salt και τον κωδικό που έχει δώσει ο χρήστης, θα κάνει hash τον κωδικό και θα ελέγξει αν είναι ίδιος με τον κωδικό που έχει αποθηκευμένο.

Έτσι, ακόμα και αν παραβιαστεί ο σερβερ, ο επιτιθέμενος δε θα έχει στη κατοχή του τον αληθινό κωδικό του χρήστη αλλά το hash αυτουνού, οπότε θα χρειαστεί να τους σπάσει με κάποια μέθοδο (π.χ brute force) που γενικά είναι δύσκολο. Η Java προσφέρει τη κλάση **SecureRandom** για παραγωγή τυχαίων bytes οπότε το salt πρέπει να δημιουργείται με αυτή τη κλάση που είναι κρυπτογραφικά ασφαλές (π.χ σε σχέση με τη Math.random()).

Τέλος πέρα από αυτά, πρέπει να υπάρχει κάποιο πρωτόκολλο για **αυθεντικοποίηση** και **εξουσιοδότηση**, δηλαδή το σύστημα client-server να γνωρίζει ποιός είναι ποιός καθώς και τι ενέργειες μπορεί να κάνει κάποιος (π.χ οι admins στην άσκηση μπορούν να κάνουν παραπάνω ενέργειες από τους απλούς χρήστες). Η Java προσφέρει την υπηρεσία **JAAS** (Java Authentication and Authorization Service) για αυτόν τον σκοπό.

Ένα παράδειγμα σε αυτή την περίπτωση είναι πως ένας επιτιθέμενος έχει παραβιάσει το σύστημα ενός χρήστη και κάνει ενέργειες μη επιτρεπτές οπότε πρέπει να αποδειχθεί ότι όντως δεν τις έκανε ο χρήστης που παραβιάστηκε.

ΕΞΩΤΕΡΙΚΕΣ ΠΗΓΕΣ

Hashing κωδικών με salt

<https://crackstation.net/hashing-security.htm>

SSL (JSSE)

https://www.ibm.com/support/knowledgecenter/ssw_i5_54/rzaha/rzahajssemain.htm

JAAS

<https://docs.oracle.com/javase/8/docs/technotes/guides/security/jaas/JAASRefGuide.html>

Δημιουργία Multi-threaded server

http://www.tutorialspoint.com/javaexamples/net_multisoc.htm