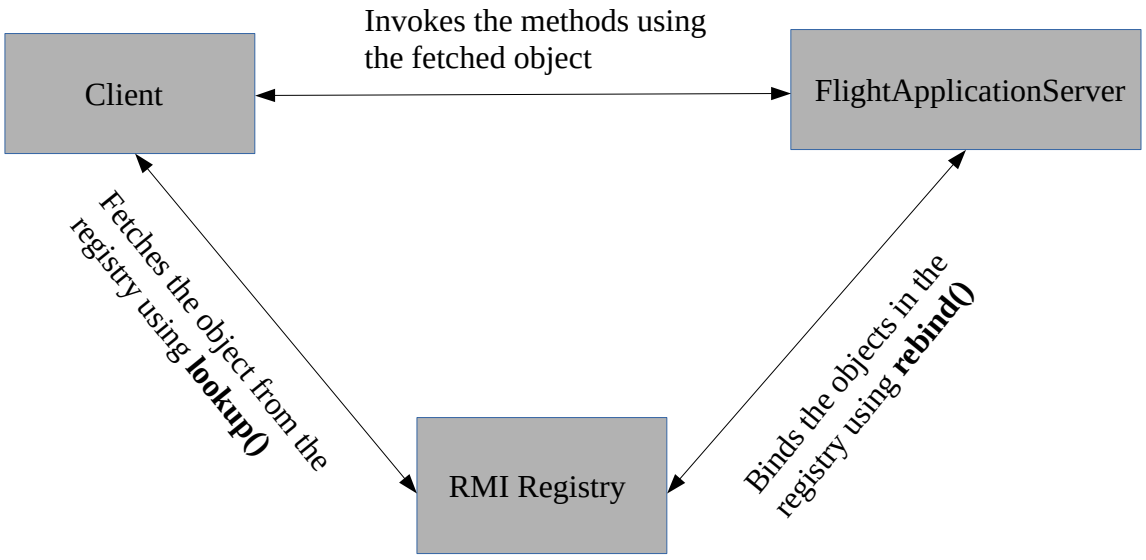
JAVA RMI - SOCKETS

1. **Communication Model Client – FlightApplicationServer**
2. **Communication Model FlightApplicationServer – DBServer**
3. **Application Protocol FlightApplicationServer ↔ DBServer**
4. **Review of FlightApplicationServer**
5. **MySQL Structure & commands**
6. **Review of DBServer**
7. **Race Conditions**
8. **Architecture/Technology Implementation**
9. **Security**
10. **Development Tools (included jars)**
11. **App Execution Guidelines**
12. **Runtime Screenshots**
13. **Links**

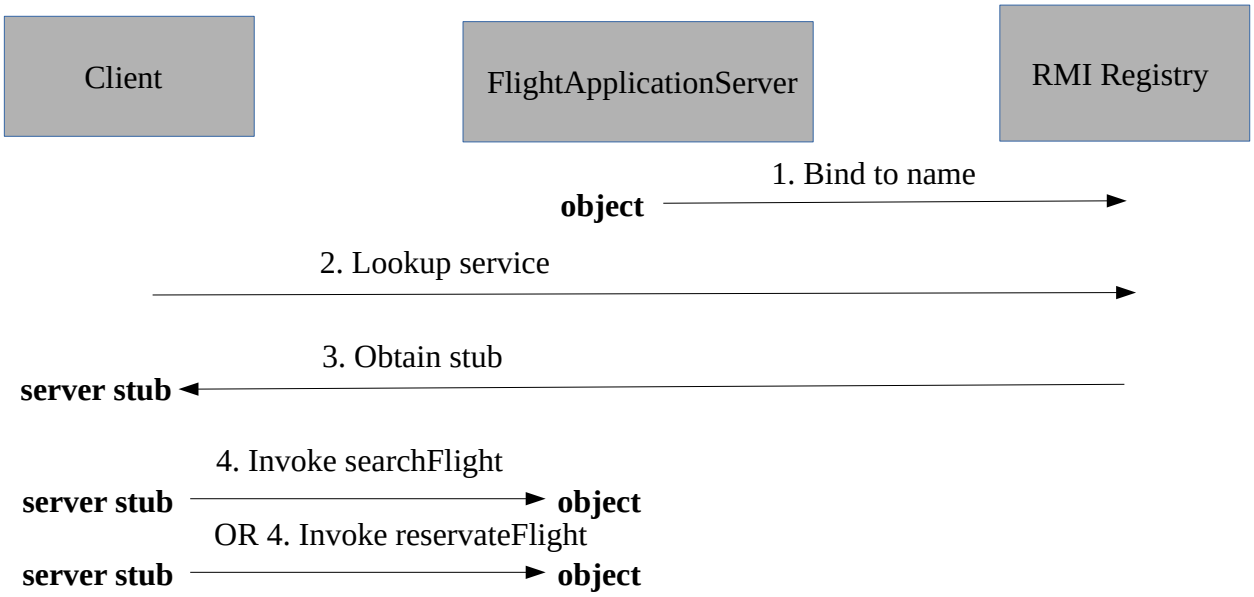
1st Server: FlightApplicationServer
2nd Server: DBServer

1. Communication Model Client – FlightApplicationServer

Μια απλή προσέγγιση στην rmi αρχιτεκτονική είναι το παρακάτω διάγραμμα. Το πρόγραμμα Client, χρειάζεται να αναζητήσει το δικτυακό όνομα του rmi server, προκειμένου να καλέσει τις απομακρυσμένες μεθόδους του. Ο server, από την πλευρά του, συσχετίζει το δικτυακό του όνομα κάνοντας χρήση της υηρεσίας RMI Registry, την ίδια υπηρεσία που χρησιμοποιεί το πρόγραμμα Client.

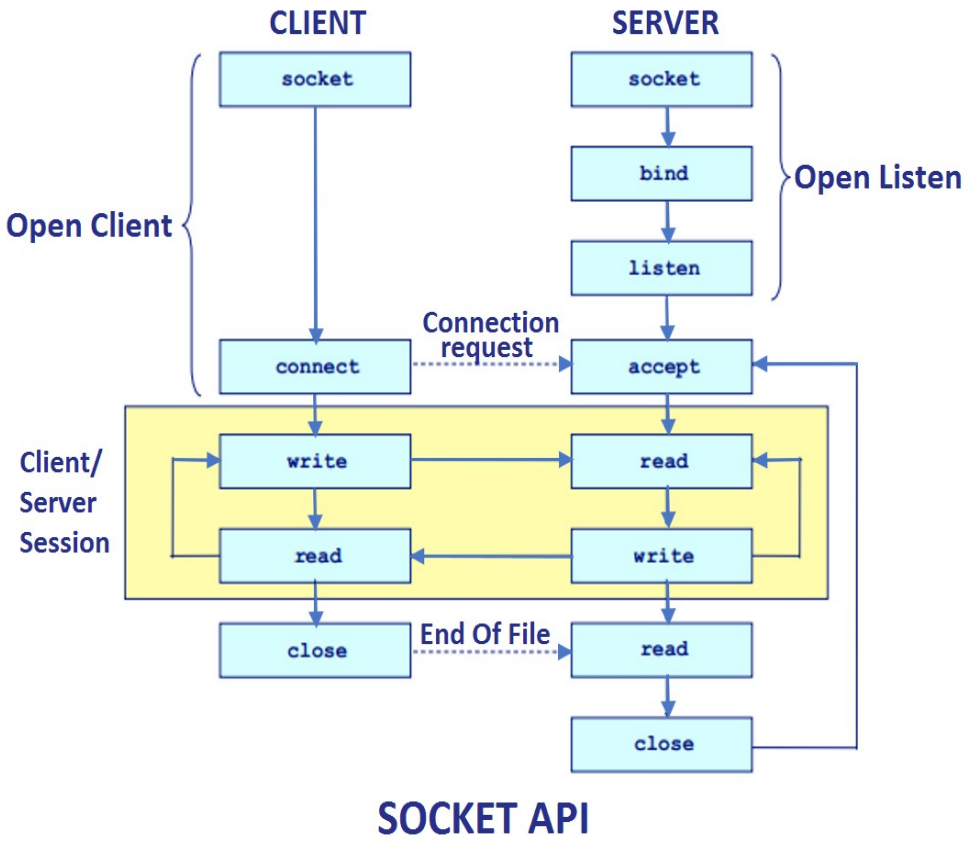


Από την πλευρά του πρωτοκόλλου, οι ακόλουθες ενέργειες συμβαίνουν. Ο RMI Server προσφέρει τις ***searchFlight*** και ***reservateFlight*** μεθόδους.



2. Communication Model FlightApplicationServer – DBServer

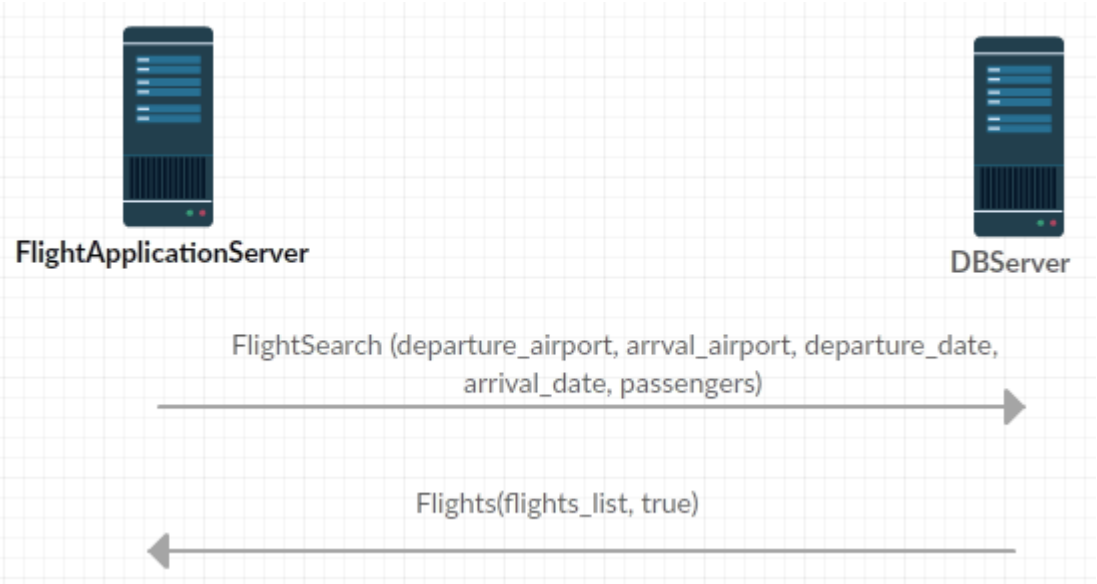
Σε αυτό το κεφάλαιο, εξηγείται η επικοινωνία μέσω socket, ανάμεσα στους δύο εξυπηρετητές. Ο FlightApplicationServer είναι ο πελάτης και ο DBServer, είναι ο εξυπηρετητής (ServerSocket). Αυτή η υλοποίηση προκύπτει από την ύπαρξη μιας Βάσης Δεδομένων που διατηρεί όλες τις πτήσεις. Έτσι, ο FlightApplicationServer πρέπει να επικοινωνήσει με τον DBServer, μέσω Java sockets. Η επικοινωνία με την Βάση Δεδομένων πρέπει να υλοποιηθεί με ένα χαμηλότερο επίπεδο από ότι το rmi. Επιπλέον, η τεχνολογία rmi προσφέρει ποικίλες μεθόδους σε ένα υψηλό και αυτοματοποιημένο επίπεδο. Αντίστοιχα, τα sockets, διαχειρίζονται την όλη επικοινωνία και σύνδεση σε πιο χαμηλό επίπεδο, κάτι που προσφέρει ταχύτητα, και λεπτομερή χειρισμό σε μια περίπτωση όπως η παραπάνω.



Socket Lifecycle

3. Application Protocol FlightApplicationServer ↔ DBServer

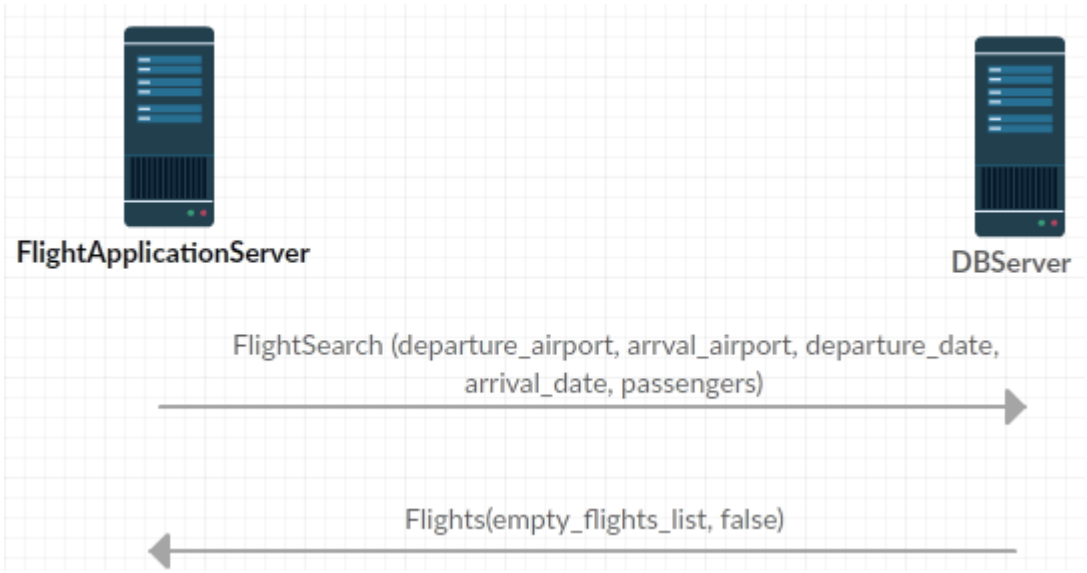
Το πρωτόκολλο απεικονίζει τα ανταλλασσόμενα μηνύματα, το μορφότυπο και τους request/reply κανόνες. Ένα σύνολο κλάσεων λοιπόν, έχουν υλοποιηθεί για αυτό το σκοπό, αναπαριστώντας τα διάφορα μηνύματα.



Πρωτόκολλο αναζήτησης πτήσεων με βάση – Επιτυχής Αναζήτηση: αεροδρόμιο αναχώρησης-προορισμού, ημερομηνία αναχώρησης-άφιξης, και επιβάτες πτήσης.

Το αντικείμενο (Κλάση) FlightSearch, αναπαριστά ακριβώς το αίτημα του πελάτη που το προωθεί στον RMI Εξυπηρετητή. Περιέχει όλες τις πληροφορίες για να μπορέσει ο DB Εξυπηρετητής να αναζητήσει πτήσεις στον πίνακα με τις πτήσεις, και να επιστρέψει τα αποτελέσματα εάν βρεθούν.

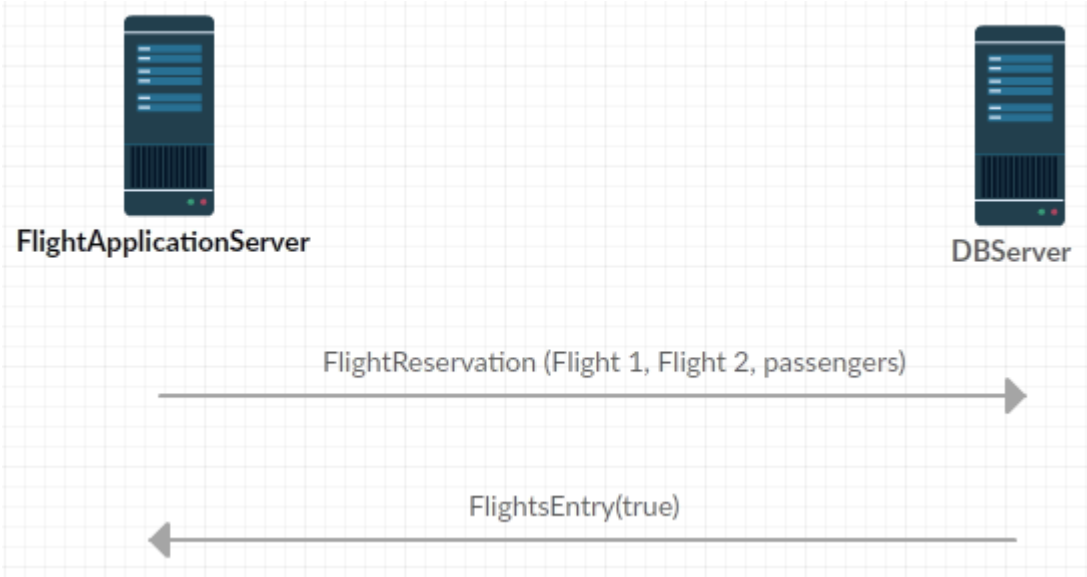
Η απάντηση έρχεται από τον Εξυπηρετητή ΒΔ, με το αντικείμενο Flights. Η πληροφορία που βρίσκεται σε αυτό το αντικείμενο είναι μια λίστα με τις πτήσεις που πληρούν τα κριτήρια αναζήτησης, και μια σημαία (boolean flag), η οποία αν η λίστα έχει πτήσεις θα είναι αληθή. Αν όμως η λίστα είναι άδεια, τότε δεν θα έχουν βρεθεί πτήσεις και επομένως η σημαία θα είναι ψευδή.



Πρωτόκολλο αναζήτησης πτήσεων με βάση – Ανεπιτυχής Αναζήτηση: αεροδρόμιο αναχώρησης-προορισμού, ημερομηνία αναχώρησης-άφιξης, και επιβάτες πτήσης.

Στη περίπτωση που δεν βρέθηκαν πτήσεις στον πίνακα, τότε το αντικείμενο Flights, θα ενσωματώσει μια άδεια λίστα, και θα θέσει την σημαία ως ψευδή.

Τώρα στη περίπτωση της κράτησης πτήσεων, ο χρήστης μπορεί να προχωρήσει στην κράτηση 2 πτήσεων, μια πτήση για $A \rightarrow B$, και την δεύτερη πτήση για $B \rightarrow A$. Με Α, ως αεροδρόμιο αναχώρησης και Β άφιξης. Επομένως, το πρωτόκολλο κράτησης:



Πρωτόκολλο κράτησης πτήσεων με βάση – Επιτυχής Κράτηση: Πτήση 1, πτήση 2 και αριθμός επιβατών για κάθε πτήση.

Το αντικείμενο FlightReservation, αντιστοίχως, συγκεντρώνει τις δύο αυτές πτήσεις και τον αριθμό επιβατών που θα πρέπει κάθε πτήση να ικανοποιεί. Έπειτα, αν ο εξυπηρετητής ΒΔ καταφέρει και ενημερώσει τις αντίστοιχες θέσεις στον πίνακα πτήσεων, τότε επιστρέφει ένα μήνυμα ως FlightsEntry. Χρησιμοποιώντας μια σημαία ως αληθή, γεγονός που δείχνει ότι πράγματι ενημέρωσε τις θέσεις.

Πρωτόκολλο κράτησης πτήσεων με βάση – Ανεπιτυχής Κράτηση: Πτήση 1, πτήση 2 και αριθμός επιβατών για κάθε πτήση.

4. Review of FlightApplicationServer

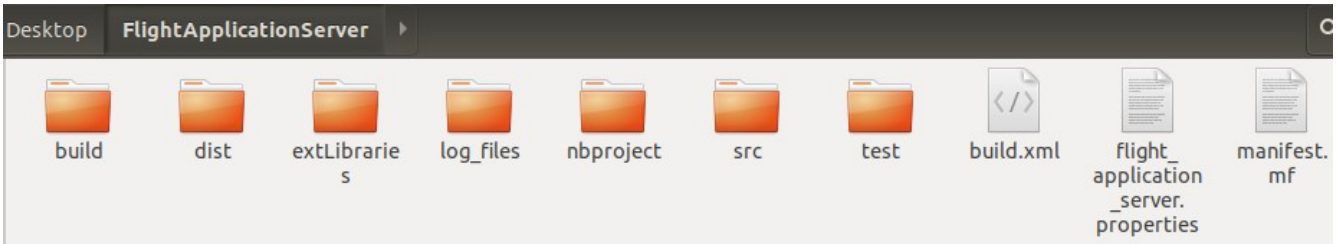
Σε αυτό το κεφάλαιο επιχειρείται, μια τεχνική ανάλυση στο project αυτό, στις λειτουργίες του, στις παραδοχές όπου βασίζεται αυτή.

- **Logging**. Η τεχνική αυτή επιτρέπει στο πρόγραμμα να τυπώνει και να αποθηκεύει σε αρχείο τα γεγονότα και τα μηνύματα που ορίζει ο προγραμματιστής ως σημαντικά. Έτσι ακόμη και αν το πρόγραμμα κλείσει βίαια, όλα τα μηνύματα που έγιναν logging, θα εξακολουθούν να υπάρχουν στο συγκεκριμένο αρχείο. Η τεχνική αυτή χρησιμοποείται και στα 3 projects. Στην ενότητα <4> αναφέρεται η βιβλιοθήκη που χρησιμοποιείται για το σκοπό αυτό. Στη δομή του netbeans project, υπάρχει ένας φάκελος **log_files**. Εκεί μέσα ο LOGGER, δημιουργεί ένα αρχείο **$.log**, και γράφει σε αυτό.

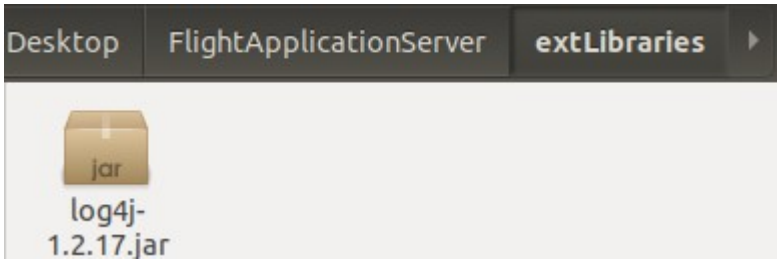
- Επίσης, υπάρχει ένα αρχείο **log4j.properties**, στο οποίο ρυθμίζεται το όνομα του log αρχείου, το κάθε πότε θα γίνεται rotate, η μορφή του μηνύματος που θα γράφεται και άλλα. Αυτό το αρχείο πρέπει να είναι τοποθετημένο μέσα στο φάκελο **.src** .

- **Properties file**. Το αρχείο ιδιοτήτων (properties or configuration file), επιτρέπει στον χρήστη που τρέχει το πρόγραμμα, να ορίσει τιμές ρητά, σε ορισμένες μεταβλητές του προγράμματος. Τέτοιο αρχείο, χρησιμοποείται στο παρόν αλλά και στα 3 projects. Στη δομή του netbeans project, βρίσκεται και έχει όνομα **$.properties**.

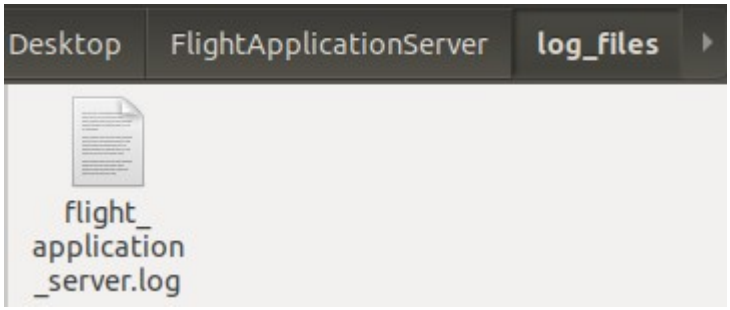
- **External Libraries**. Στη δομή του netbeans project, υπάρχει ένας φάκελος με όνομα **extLibraries**. Εκεί είναι τοποθετημένες όλες οι εξωτερικές βιβλβιοθήκες που χρησιμοποιούνται.



Δομή FlightApplicationServer netbeans project. Βρίσκεται το flight_application_server.properties.



*Φάκελος **extLibraries**. Τοποθετημένες οι εξωτερικές βιβλιοθήκες.*



*Φάκελος **log_files**. Τοποθετημένο το αρχείο flight_application_server.log*

log4j.properties

```
1  # Root logger option
2  log4j.rootLogger=INFO, file
3
4  # configuration to print into file
5  log4j.appender.file=org.apache.log4j.RollingFileAppender
6  log4j.appender.file.File=./log_files/dbserver.log
7  log4j.appender.file.MaxFileSize=12MB
8  #log4j.appender.file.MaxBackupIndex=10
9  log4j.appender.file.layout=org.apache.log4j.PatternLayout
10 log4j.appender.file.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss} %-5p %c{1}:%L - %m%n
```

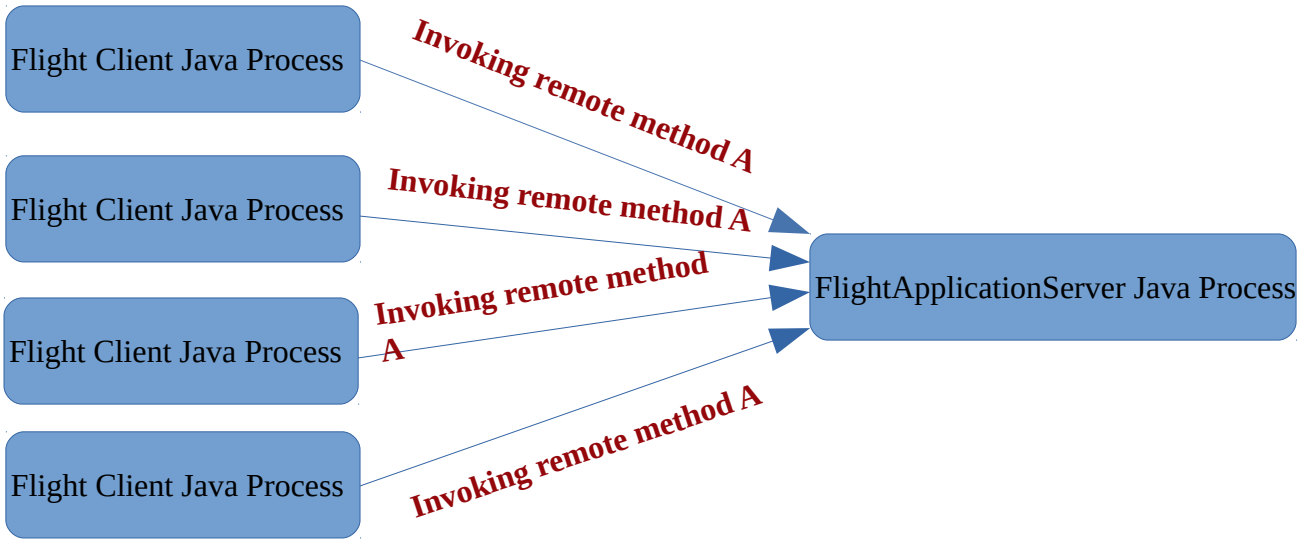
log4j.properties [FlightApplicationServer/src/log4j.properties]

- **Λειτουργίες.** Ο FlightApplicationServer, βασίζεται στην τεχνολογία Java Remote Method Invocation. Δηλαδή, δημιουργεί ένα αντικείμενο το οποίο το εξάγει σε μια συγκεκριμένη θύρα, και το συσχετίζει με ένα συγκεκριμένο όνομα. Αυτή η διαδικασία, γίνεται για να μπορεί ο πελάτης (αφού αποκτήσει μια αναφορά στο απομακρυσμένο αντικείμενο), να καλέσει τις μεθόδους που προσφέρει ο εξυπηρετητής. Έτσι, ο εξυπηρετητής, ορίζει τις μεθόδους – υπηρεσίες που θέλει να προσφέρει. Ο εν λόγω εξυπηρετητής, ορίζει δύο μεθόδους. Την searchFlight, και την reservateFlight.

  List<Flight> searchFlight(FlightSearch fs)
  boolean reservateFlight(FlightReservation fs)

JAVA RMI - SOCKETS

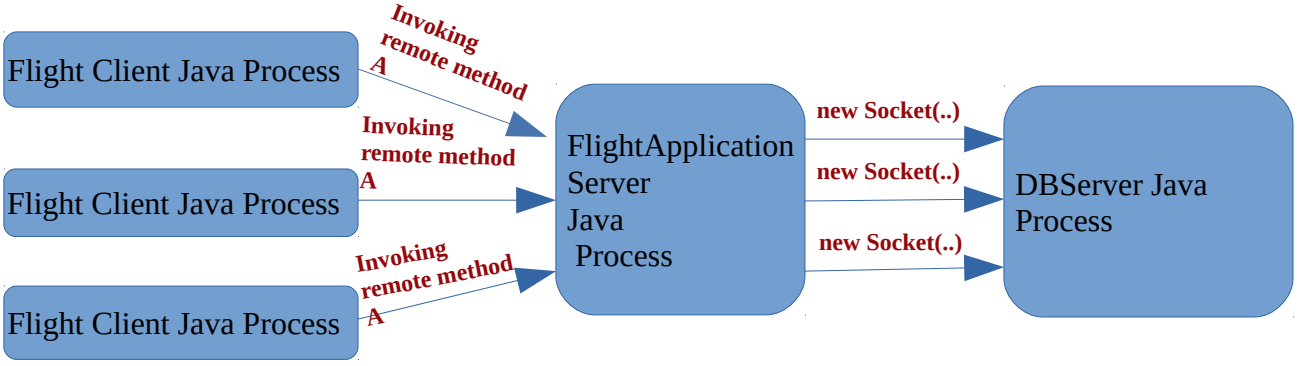
Η τεχνολογία rmi, προσφέρει τη δυνατότητα της ταυτόχρονης εξυπηρέτησης πελατών, χωρίς να γίνει ρητή χρήση κώδικα για δημιουργία νέου νήματος. Έτσι, παρέχεται η δυνατότητα εξυπηρέτησης πολλών πελατών την ίδια στιγμή. Όμως ο συγχρονισμός νημάτων, πρέπει να ληφθεί υπόψην, και συνεπώς να αντιμετωπιστεί ξεχωριστά, μιας και είναι μια κατάσταση που προκύπτει σε πολυνηματικό περιβάλλον.



4 διαφορικές διεργασίες Client, επικοινωνούν με τον RMI Server, την ίδια στιγμή, καλώντας την ίδια απομακρυσμένη μέθοδο.

Όσον αφορά, τη μετέπειτα συνέχεια του πρωτοκόλλου, ο εξυπηρετητής, αφού λάβει αίτημα για αναζήτηση πτήσεων (remote method **searchFlight**), θα αναλάβει να ξεκινήσει επικοινωνία (μέσω socket), με τον Εξυπηρετητή ΒΔ, ώστε να γυρίσει μια απάντηση στον πελάτη. Η διαδικασία περιγράφεται στο κεφάλειο <3>. Αντιστοίχως, για την μέθοδο **reservateFlight**.

Ωστόσο, γίνεται σαφή, ότι εφόσον ο εξυπηρετητής, ταυτόχρονα επεξεργάζεται πολλαπλά αιτήματα, τότε θα πρέπει να ξεκινάει και ταυτόχρονα πολλές συνόδους επικοινωνίας με τον Εξυπηρετητή ΒΔ.



Και άρα, ο εξυπηρετητής ΒΔ, θα πρέπει να είναι σε θέση να δέχεται και να εξυπηρετεί ταυτόχρονα πολλούς πελάτες (Δηλαδή, πολλές συνδέσεις από τον RMI Server).

5. MySQL Structure & commands

Ο Εξυπηρετητής ΒΔ, χρησιμοποιεί μια Βάση Δεδομένων για αποθήκευση των πτήσεων. Συγκριτικά με το αρχείο ή την δυναμική δομή δεδομένων όπως την λίστα, είναι η ρεαλιστικότερη προσέγγιση σε ένα πραγματικό σύστημα. Η Βάση Δεδομένων που υπάρχει ακολουθεί το πρότυπο της MySQL. Κατόπιν, ένας πίνακας δημιουργείται έχοντας 7 κολώνες αντιπροσωπεύοντας την πτήση.

Υπό την εξής παραδοχή, υπάρχει ένα **shell script [DBServer/setup_mysqldb.sh]**, το οποίο έχει ως ρόλο, τη λήψη της MySQL, την εγκατάσταση της, την δημιουργία χρήστη, βάσης δεδομένων, πίνακα και εισαγωγής δεδομένων στον πίνακα, ώστε να μπορεί το project DBServer να τρέξει επιτυχώς. Έτσι, υπό την παραδοχή, ότι ο υπεύθυνος για την εκτέλεση του συστήματος, τρέχει πρώτα αυτό το σενάριο κελύφους, έπειτα μπορεί να εκκινήσει τον DBServer ως Java Process.

Αναλυτικά, μετά την εκτέλεση του σενάριου κελύφους:

a) Εγκαθίσταται η MySQL στο linux os.
b) Δημιουργείται MySQL user με user: **fuser**, και κωδικό: **flight**
c) Δημιουργείται βάση δεδομένων: **flight_sys**
d) Δημιουργείται πίνακας: **Flights**
e) Γίνεται εισαγωγή δεδομένων στον πίνακα [36 rows]

Ενδεικτικά, για σύνδεση στη βάση τοπικά **$ mysql -ufuser -pflight.**

```
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 5
Server version: 5.7.26-0ubuntu0.18.04.1 (Ubuntu)

Copyright (c) 2000, 2019, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| flight_sys         |
+--------------------+
2 rows in set (0.00 sec)

mysql> use flight_sys;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;
+----------------------+
| Tables_in_flight_sys |
+----------------------+
| Flights              |
+----------------------+
1 row in set (0.00 sec)

mysql>
```

Ακολούθως, ενδιαφέρον έχει η δομή του πίνακα Flights, μιας και εκεί είναι ο χώρος αποθήκευσης των πτήσεων.

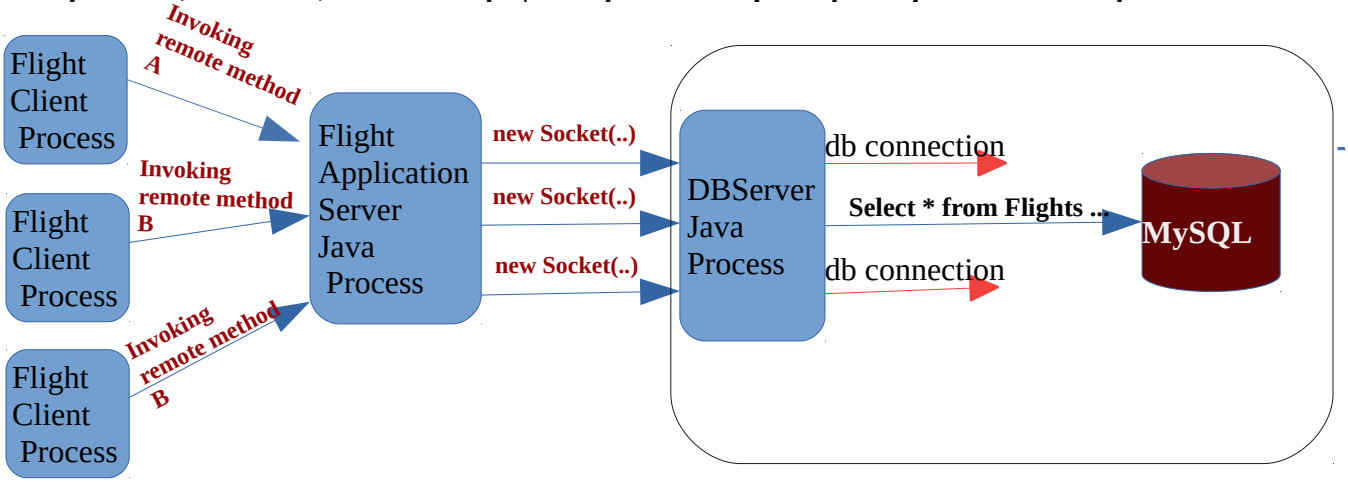
Flights	`fdate` date	‘ftime` time	`departure_airport` varchar(35)	`arrival_airport` varchar(35)	`flight_code` varchar(7)	`flight_code` varchar(7)	`seats` int(11)

PRIMARY KEY (`flight_code`). Ως μοναδικό πεδίο, θεωρείται ο κωδικός πτήσης.

6. Review of DBServer

Το project αυτό, μοντελοποιεί τον Εξυπηρετητή Βάσης Δεδομένων. Δέχεται αιτήματα από τον FlightApplicationServer, και αλληλεπιδρά με την MySQL βάση, προκειμένου να εξάγει αποτελέσματα, τα οποία τα επιστρέφει σε αυτόν.

Το πρόγραμμα ξεκινάει αρχικοποιώντας ένα ServerSocket, το οποίο ακούει για αιτήματα σε μια θύρα ορισμένη στο properties file. Επίσης, παρέχεται η δυνατότητα εκκίνησης νέου νήματος για κάθε νέο αίτημα, με τη χρήση ThreadPool (java Executors). Το πιο κρίσιμα ζήτημα όμως, είναι η διαχείριση του συχγρονισμού των νημάτων αυτών, επειδή τα νήματα αυτά προσπαθούν να αποκτήσουν πρόσβαση στον πίνακα που αποτελεί και τον κοινό πόρο στο σύστημα αυτό. ***Σε αυτό το σημείο, εμφανίζονται δύο προσεγγίσεις. Η πρώτη υλοποιεί τον συγρονισμό νημάτων στον FlightApplicationServer, ο οποίος κλειδώνει τις υπόλοιπες συνδέσεις και επιτρέπει σε μια μόνο σύνδεση, να ξεκινήσει επικοινωνία με τον DBServer. Έτσι, στη πλευρά του DBServer, δε χρειάζεται να κλειδωθούν οι συνδέσεις. Η δεύτερη προσέγγιση η οποία και υλοποιείται στο σύστημα αυτό, υλοποιεί τον συχγρονισμό στον DBServer. Έτσι, ο rmi server, δε κλειδώνει καμιά σύνδεση πελάτη, αλλά ανοίγει ταυτόχρονα συνδέσεις προς τον Εξυπηρετητή ΒΔ. Και ο τελευταίος, έχει την αποκλειστική ευθύνη, να παγώσει τις συνδέσεις, και να επιτρέψει σε μια να αλληλεπιδράσει με τον κοινό πόρο.***



Έτσι, ο DBServer, αναλαμβάνει τον συχγρονισμό των συνδέσεων και μπλοκάρει τις συνδέσεις, όταν:

	SELECT (R)	UPDATE (W)
SELECT (R)	**OK**	**SYNCHRONIZE**
UPDATE (W)	**SYNCHRONIZE**	**SYNCHRONIZE**

Μια ακόμη σημαντική λειτουργία, είναι ο έλεγχος της IP διεύθυνσης του FlightApplicationServer. Στο properties αρχείο του DBServer, ορίζεται η IP του FlightApplicationServer. Έτσι, μπορεί να γίνεται δυναμικά έλεγχος, για το αν ήρθε σύνδεση πράγματι από τον rmi server, ή απλά έγινε προσπάθεια σύνδεσης από άλλη διεύθυνση.

Τέλος, κάθε νήμα για να εκτελέσει τη λειτουργία που πρέπει στον πίνακα, πρέπει να αποκτήσει σύνδεση στη βάση δεδομένων. Με βάση τις βιβλιοθήκες apache commons-dbcp2-2.6.0 (data base connection pool), και commons-pool2-2.6.2, δημιουργείται μια πισίνα από συνδέσεις που δίνονται κατά απαίτηση, για σύνδεση στη βάση.

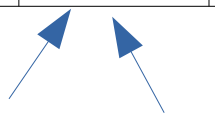
7. Race Conditions

Ο κοινός πόρος, μεταξύ των νημάτων είναι ο πίνακας Flights. Όμως, πιο συγκεκριμένα είναι η κολώνα (column) seats. Διότι εκεί γίνεται η αλλαγή της τιμής, μετά την κράτηση των πτήσεων. Έτσι, ανταγωνίζονται οι πιθανές συνδέσεις, για το ποια θα πραγματοποιήσει ένα UPDATE, στον πίνακα και στη στήλη αυτή.

Flights

2019-01-12	12:30:00	ATHENS	BERLIN	ATBR152	46	100
2019-09-19	15:00:00	SAMOS	ATHENS	SMAT098	34	234

Στιγμιότυπο δύο πτήσεων, στον πίνακα Flights.



Έτσι, τι θα συμβεί αν την ίδια στιγμή δύο νήματα, προσπαθήσουν να ενημερώσουν την στήλη ΘΕΣΕΙΣ, της ίδιας γραμμής; Η λύση είναι η δήλωση των μεθόδων search & book flights ως **synchronized**. Όμως, πρέπει να ληφθεί υπόψην και το deadlock. Δηλαδή, θα πρέπει να υπάρχουν δύο σημαίες, οι οποίες σηματοδοτούν, πότε ο πίνακας διαβάζεται και πότε ενημερώνεται. Ώστε το νήμα που πρόκειται να διαβάσει, και προλάβει να εκτελεσθεί, να θέσει FLAG A = true. Έτσι, κλειδώνεται η περιοχή για τα νήματα με ενημέρωση. Επιτρέπονται όμως, τα νήματα που διαβάζουν.

Αντίστοιχα, αν προλάβει την περιοχή ένα νήμα που πρόκειται να ενημερώσει, τότε θα πρέπει να θέσει FLAG A = true, ώστε να μην επιτρέπεται ΚΑΝΕΝΑ ΑΛΛΟ νήμα που θα ενημερώσει, αλλά και FLAG B = false, ώστε να μην επιτρέπει ΟΥΤΕ ΣΤΑ ΝΗΜΑΤΑ που πρόκειται να διαβάσουν.

8. Architecture/Technology Implementation

Πλεονεκτήματα μοντέλου Client – Server – Server
Στο μοντέλο 2 επιπέδων, ο εξυπηρετητής αναλαμβάνει γενικό ρόλο, εξυπηρετώντας τις εσειρχόμενες συνδέσεις, και αλληλεπιδρόντας με την βάση δεδομένων, υλοποιώντας και το data access layer. Υπάρχει μεγαλύτερη πολυπλοκότητα. Αντιθέτως, στο μοντέλο 3 επιιπέδων, οι εξυπηρετητές είναι υπεύθυνοι για συγκεκριμένη υπηρεσία (Server 1 → logic service, Server 2 → data access service).

Επιπλέον, η ασφάλεια επεκτείνεται και υλοποιείται για κάθε υπηρεσία στο server side μέρος, και όχι συγκεντρωτικά πάνω σε έναν server.

Επίσης, ο client, δεν έχει άμμεση πρόσβαση στη βάση δεδομένων, αλλά έμμεσα.

Η αναβάθμιση ή τροποποίηση μιας υπηρεσίας δεν επηρεάζει ολόκληρο το σύστημα.

Πλεονεκτήματα RMI τεχνολογίας έναντι Sockets
Η τεχνολογία Java RMI, είναι τεχνολογία που επιτρέπει σε ένα Java Client πρόγραμμα να επικοινωνήσει με έναν απομακρυσμένο server, καλώντας απομακρυσμένες μεθόδους. Οι οποίες εκτελούν την υπηρεσία. Σε αυτό το σημείο, η τεχνολογία αυτή διαχειρίζεται αυτόματα και αποκρύπτει τις λεπτομέρειες δικτύου, έναρξης, τερματισμού συνόδου. Αλλά ένα εξίσου σημαντικό στοιχείο που παρέχει είναι η παρουσία ασφάλειας μεταξύ των οντοτήτων (Policies), που επιτρέπει ή και απαγορεύει ορισμένες ενέργειες. Οπότε και προτιμάται στις εφαρμογές μεταξύ client – server.

Από την άλλη πλευρά, τα Java Sockets, περιγράφουν την επικοινωνία σε χαμηλό επίπεδο, αναγκάζοντας τον προγραμματιστή, να ασχοληθεί με την δημιουργία, τερματισμό της συνόδου, αλλά και το σχεδιασμό του πρωτοκόλλου. Επίσης, δε παρέχεται ασφάλεια, αλλά πρέπει να δημιουργηθεί κατά απαίτηση κάποιο λογισμικό, που να παρέχει βασικές λειτουργίες ασφάλειας.

9. Security

Τεχνολογίες ασφάλειας στο σύστημα που μπορούν να εφαρμοσθούν:

- **Firewall**.
  Εφόσο είναι γνωστή η διαθέσιμη πόρτα (1099), και το πρωτόκολλο σύνδεσης (TCP), μπορεί να δημιουργηθεί κανόνας η σύνολο κανόνων, που να επιτρέπουν **μόνο** αυτή τη δικτυακή κίνηση.

- **Υποδομή Δημοσίου Κλειδιού / TLS Encryption.**
  Το πρωτόκολλο εφαρμογής TLS, προσφέρει αυθεντικοποίηση μεταξύ των μερών, και κρυπτογράφηση των δεδομένων, ώστε η όλη επικοινωνία να παρέχεται με ασφαλή τρόπο. Πρόκειται για ένα ιδανικό πρωτόκολλο, καθώς είναι ιδανικό για προγράμματα στο application layer. Έτσι, εμποδίζεται και η επίθεση man-in-the-middle, εφόσον τα πιστοποιητικά των δύο πλευρών μπορούν να οδηγήσουν σε αυθεντικοποίηση.

- **Ipsec ESP mode &TLS**
  Αυτή η περίπτωση κυρίως ταιριάζει για την επικοινωνία μεταξύ των δύο εξυπηρετητών. Το Ipsec esp, είναι ικανό να κρυπτογραφήσει όλα τα IP packets, και να τα αυθεντικοποιήσει. Έτσι, δεν επηρεάζει καθόλου το σύστημα που τα Java Sockets, δε παρέχουν ασφάλεια, επειδή το πρωτόκολλο Ipsec, ασφαλίζει ολόκληρο το επίπεδο δικτύου, οπότε καλύπτεται και το επίπεδο εφαρμογής.

10. Development Tools (included jars)

Operating System: Distributor ID: Ubuntu
Description: Ubuntu 18.04.2 LTS
Release: 18.04
Codename: bionic

IDE: Apache NetBeans IDE 11.0 (Build incubator-netbeans-release-404-on-20190319)

Java: java version "11.0.2" 2019-01-15 LTS
Java(TM) SE Runtime Environment 18.9 (build 11.0.2+9-LTS)
Java HotSpot(TM) 64-Bit Server VM 18.9 (build 11.0.2+9-LTS, mixed mode)

MySQL: mysql Ver 14.14 Distrib 5.7.26, for Linux (x86_64) using EditLine wrapper

Jars:

commons-dbcp2-2.6.0	2.6.0	https://commons.apache.org/proper/commons-dbcp/download_dbcp.cgi
commons-logging-1.2	1.2	http://commons.apache.org/proper/commons-logging/download_logging.cgi
commons-pool2-2.6.2	2.6.2	http://commons.apache.org/proper/commons-pool/download_pool.cgi
log4j-1.2.17	1.2.17	https://logging.apache.org/log4j/1.2/download.html
mysql-connector-java-5.1.47	5.1.47	https://dev.mysql.com/downloads/connector/j/

11. System (Apps) Execution Guidelines

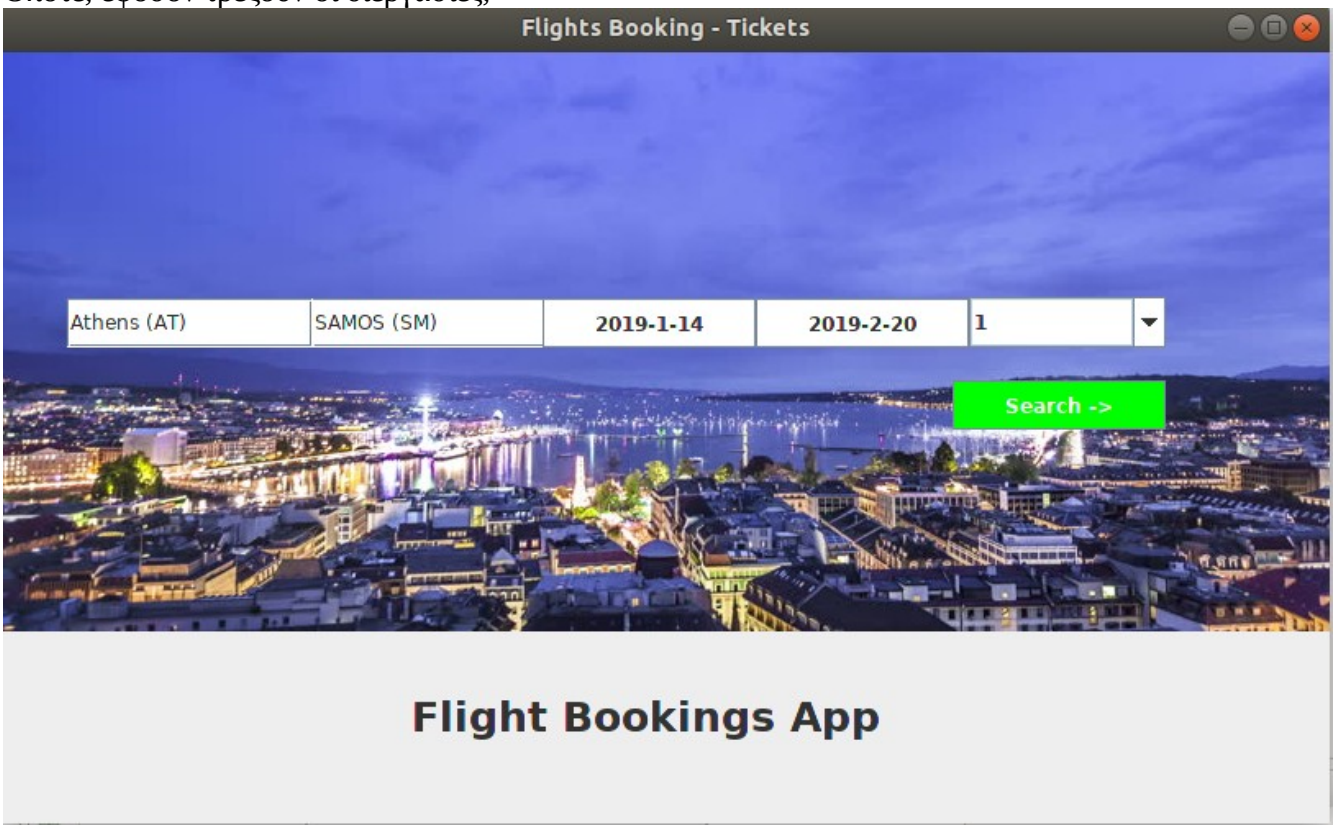
Πρόκειται να τρέξουν τρία projects. Αρχικά εκκινείται το project DBServer. Πρώτα, πρέπει να οριστεί η πόρτα στο properties file. Έπειτα , μέσω του NetBeans IDE, εκκινείται η διεργασία.
Στη συνέχεια, θα τρέξει το project FlightApplicationServer. Ορίζεται, η πόρτα και το όνομα που θα ακούει για κλήσεις μεθόδων, στο properties file και στη συνέχεια μέσω του IDE, εκκινείται η διεργασία.
Τέλος, για να τρέξει το πρόγραμμα πελάτη, απλά εκκινείται η διεργασία μέσω IDE.

12. Runtime Screenshots

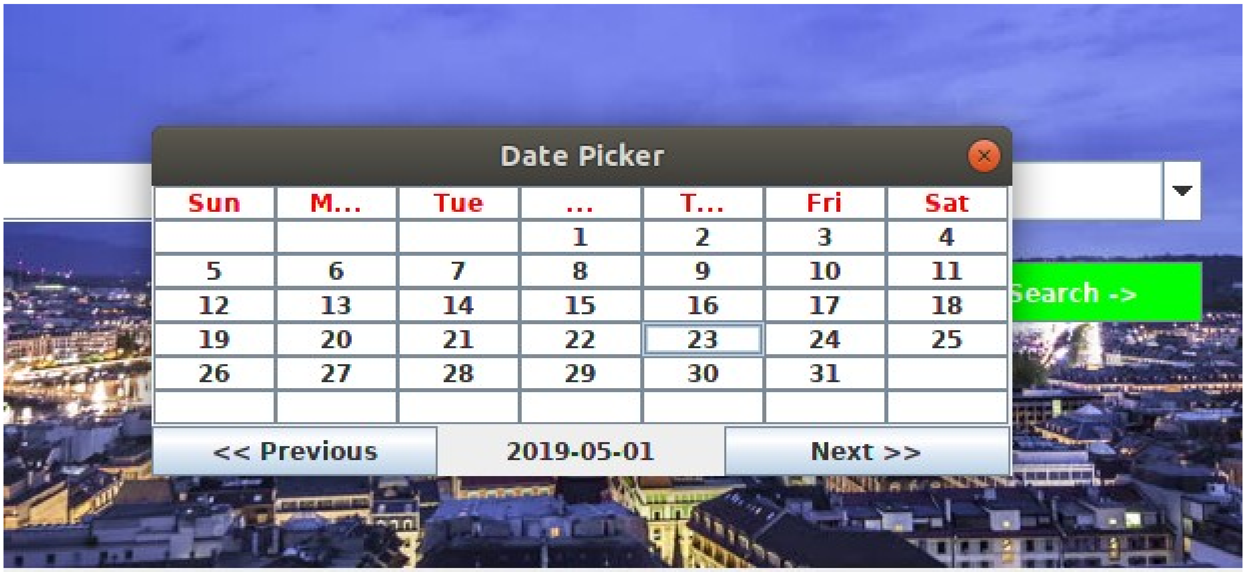
Για να παρατηρηθούν, οι ενέργειες που γίνονται από τους δύο εξυπηρετητές, αρκεί να διαβαστούν τα logs αρχεία που παράγονται. Ενώ για το πρόγραμμα πελάτη, υπάρχει γραφική ενημέρωση αλλά και log αρχείο.

***Αρκεί να αντιγραφεί το αρχείο Operations.java, δηλαδή η απομακρυσμένη διεπαφή στο Java project του πελάτη, ώστε να μπορεί να καλέσει τις μεθόδους αυτές ο client. Τα υπόλοιπα γίνονται αυτόματα από το IDE (rmic). ***

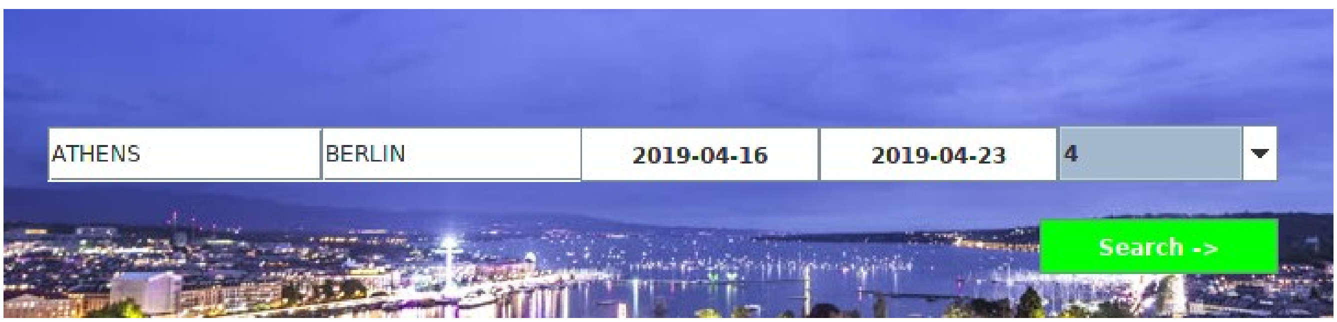
Οπότε, εφόσον τρέξουν οι διεργασίες,



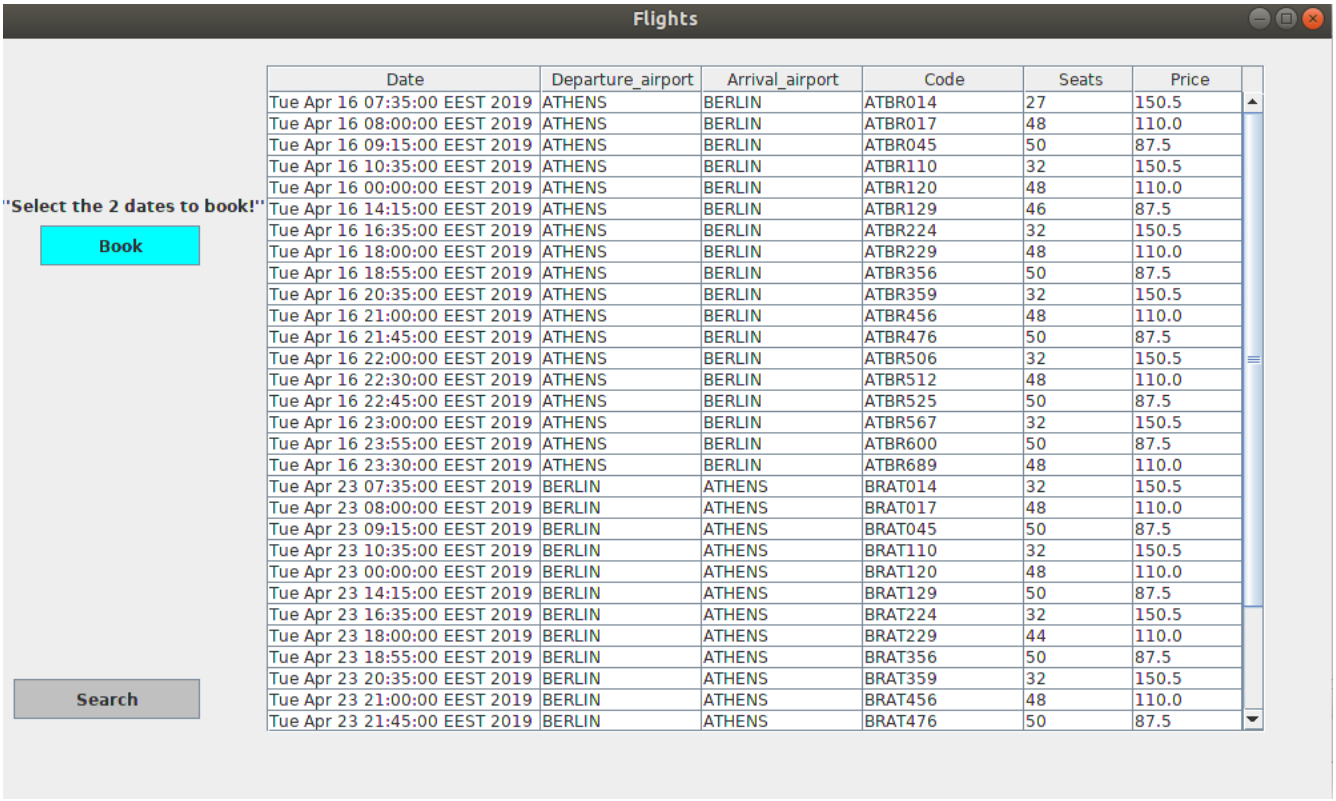
Το γραφικό παράθυρο της εφαρμογής πελάτη.



Η επιλογή των ημερομηνιών γίνεται με τη βοήθεια ειδικού γραφικού.

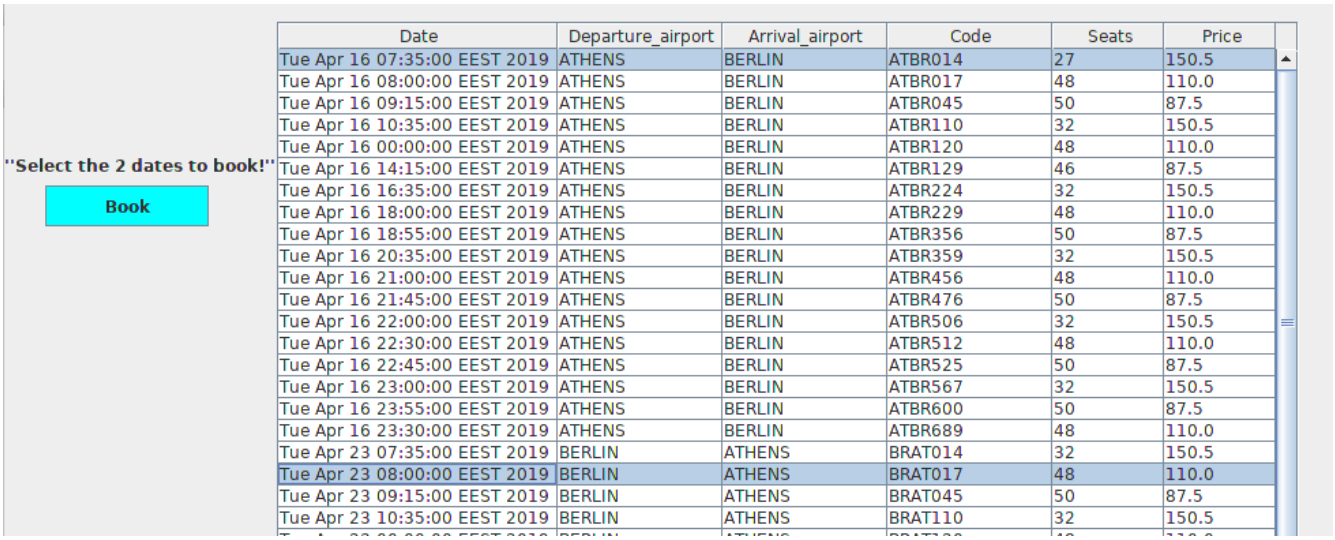


Αναζήτηση με αεροδρόμιο αναχώρησης ΑΘΗΝΑ, αεροδρόμιο άφιξης ΒΕΡΟΛΙΝΟ, ημερομηνία αναχώρησης: 2019-04-16, ημερομηνία άφιξης: 2019-04-23, και επιβάτες σε κάθε πτήση τέσσερις.



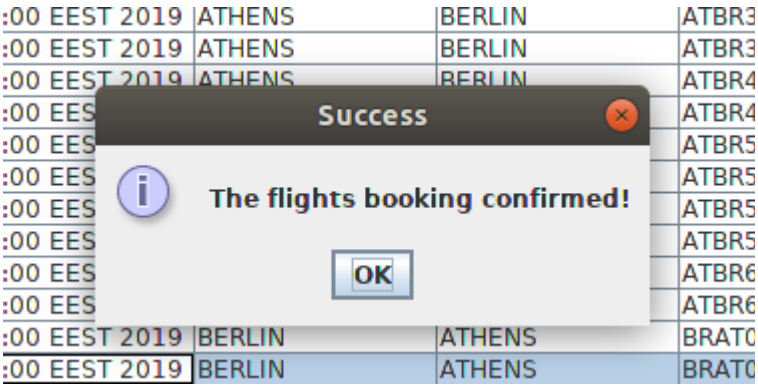
Date	Departure_airport	Arrival_airport	Code	Seats	Price
Tue Apr 16 07:35:00 EEST 2019	ATHENS	BERLIN	ATBR014	27	150.5
Tue Apr 16 08:00:00 EEST 2019	ATHENS	BERLIN	ATBR017	48	110.0
Tue Apr 16 09:15:00 EEST 2019	ATHENS	BERLIN	ATBR045	50	87.5
Tue Apr 16 10:35:00 EEST 2019	ATHENS	BERLIN	ATBR110	32	150.5
Tue Apr 16 00:00:00 EEST 2019	ATHENS	BERLIN	ATBR120	48	110.0
Tue Apr 16 14:15:00 EEST 2019	ATHENS	BERLIN	ATBR129	46	87.5
Tue Apr 16 16:35:00 EEST 2019	ATHENS	BERLIN	ATBR224	32	150.5
Tue Apr 16 18:00:00 EEST 2019	ATHENS	BERLIN	ATBR229	48	110.0
Tue Apr 16 18:55:00 EEST 2019	ATHENS	BERLIN	ATBR356	50	87.5
Tue Apr 16 20:35:00 EEST 2019	ATHENS	BERLIN	ATBR359	32	150.5
Tue Apr 16 21:00:00 EEST 2019	ATHENS	BERLIN	ATBR456	48	110.0
Tue Apr 16 21:45:00 EEST 2019	ATHENS	BERLIN	ATBR476	50	87.5
Tue Apr 16 22:00:00 EEST 2019	ATHENS	BERLIN	ATBR506	32	150.5
Tue Apr 16 22:30:00 EEST 2019	ATHENS	BERLIN	ATBR512	48	110.0
Tue Apr 16 22:45:00 EEST 2019	ATHENS	BERLIN	ATBR525	50	87.5
Tue Apr 16 23:00:00 EEST 2019	ATHENS	BERLIN	ATBR567	32	150.5
Tue Apr 16 23:55:00 EEST 2019	ATHENS	BERLIN	ATBR600	50	87.5
Tue Apr 16 23:30:00 EEST 2019	ATHENS	BERLIN	ATBR689	48	110.0
Tue Apr 23 07:35:00 EEST 2019	BERLIN	ATHENS	BRAT014	32	150.5
Tue Apr 23 08:00:00 EEST 2019	BERLIN	ATHENS	BRAT017	48	110.0
Tue Apr 23 09:15:00 EEST 2019	BERLIN	ATHENS	BRAT045	50	87.5
Tue Apr 23 10:35:00 EEST 2019	BERLIN	ATHENS	BRAT110	32	150.5
Tue Apr 23 00:00:00 EEST 2019	BERLIN	ATHENS	BRAT120	48	110.0
Tue Apr 23 14:15:00 EEST 2019	BERLIN	ATHENS	BRAT129	50	87.5
Tue Apr 23 16:35:00 EEST 2019	BERLIN	ATHENS	BRAT224	32	150.5
Tue Apr 23 18:00:00 EEST 2019	BERLIN	ATHENS	BRAT229	44	110.0
Tue Apr 23 18:55:00 EEST 2019	BERLIN	ATHENS	BRAT356	50	87.5
Tue Apr 23 20:35:00 EEST 2019	BERLIN	ATHENS	BRAT359	32	150.5
Tue Apr 23 21:00:00 EEST 2019	BERLIN	ATHENS	BRAT456	48	110.0
Tue Apr 23 21:45:00 EEST 2019	BERLIN	ATHENS	BRAT476	50	87.5

Πίνακας με όλες τις διαθέσιμες πτήσεις που πληρούν τα κριτήρια αναζήτησης. Αν δεν υπήρχαν πτήσεις, ένα JoptionPane θα εμφανιζόταν με αντίστοιχο μήνυμα.



Date	Departure_airport	Arrival_airport	Code	Seats	Price
Tue Apr 16 07:35:00 EEST 2019	ATHENS	BERLIN	ATBR014	27	150.5
Tue Apr 16 08:00:00 EEST 2019	ATHENS	BERLIN	ATBR017	48	110.0
Tue Apr 16 09:15:00 EEST 2019	ATHENS	BERLIN	ATBR045	50	87.5
Tue Apr 16 10:35:00 EEST 2019	ATHENS	BERLIN	ATBR110	32	150.5
Tue Apr 16 00:00:00 EEST 2019	ATHENS	BERLIN	ATBR120	48	110.0
Tue Apr 16 14:15:00 EEST 2019	ATHENS	BERLIN	ATBR129	46	87.5
Tue Apr 16 16:35:00 EEST 2019	ATHENS	BERLIN	ATBR224	32	150.5
Tue Apr 16 18:00:00 EEST 2019	ATHENS	BERLIN	ATBR229	48	110.0
Tue Apr 16 18:55:00 EEST 2019	ATHENS	BERLIN	ATBR356	50	87.5
Tue Apr 16 20:35:00 EEST 2019	ATHENS	BERLIN	ATBR359	32	150.5
Tue Apr 16 21:00:00 EEST 2019	ATHENS	BERLIN	ATBR456	48	110.0
Tue Apr 16 21:45:00 EEST 2019	ATHENS	BERLIN	ATBR476	50	87.5
Tue Apr 16 22:00:00 EEST 2019	ATHENS	BERLIN	ATBR506	32	150.5
Tue Apr 16 22:30:00 EEST 2019	ATHENS	BERLIN	ATBR512	48	110.0
Tue Apr 16 22:45:00 EEST 2019	ATHENS	BERLIN	ATBR525	50	87.5
Tue Apr 16 23:00:00 EEST 2019	ATHENS	BERLIN	ATBR567	32	150.5
Tue Apr 16 23:55:00 EEST 2019	ATHENS	BERLIN	ATBR600	50	87.5
Tue Apr 16 23:30:00 EEST 2019	ATHENS	BERLIN	ATBR689	48	110.0
Tue Apr 23 07:35:00 EEST 2019	BERLIN	ATHENS	BRAT014	32	150.5
Tue Apr 23 08:00:00 EEST 2019	BERLIN	ATHENS	BRAT017	48	110.0
Tue Apr 23 09:15:00 EEST 2019	BERLIN	ATHENS	BRAT045	50	87.5
Tue Apr 23 10:35:00 EEST 2019	BERLIN	ATHENS	BRAT110	32	150.5

Τώρα για να επιλέξει ο χρήστης τις 2 πτήσεις, αρκεί να κάνει κλικ στη πρώτη, και με control + click, την δεύτερη. Ο κώδικας 'πιάνει' αυτές τις δύο πτήσεις, και με το κουμπί Book, στέλνει το αίτημα κράτησης.



Μήνυμα επιτυχίας κράτησης.

Τώρα, τα logs, μπορούν να διαβαστούν και να παρατηρηθούν οι ενέργειες των διεργασιών.

12. Runtime Screenshots

[1]. https://www.techopedia.com/definition/23813/three-tier-clientserver
[2]. https://www.softwaretestingclass.com/what-is-difference-between-two-tier-and-three-tier-architecture/
[3]. https://security.stackexchange.com/questions/42990/which-is-better-for-server-to-server-communication-ipsec-or-tls
[4]. https://www.digitalocean.com/community/tutorials/7-security-measures-to-protect-your-servers