Εθνικό Μετσόβιο Πολυτεχνείο

Σχολή Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών

Χειμερινό Εξάμηνο 2019-2020

Τελική εργασία στο μάθημα Τεχνολογία Πολυμέσων

Νικόλας Παπαδόπουλος

Περιεχόμενα

1. Εισαγωγή

Στην εργασία υλοποιείται μια εφαρμογή που διευκολύνει τη διαχείριση της στάθμευσης και εξυπηρέτησης αεροσκαφών σε κάποιο αεροδρόμιο, προκειμένου να τους παρέχονται οι αναγκαίες υπηρεσίες. Στην παρούσα αναφορά γίνεται αναλυτικά η περιγραφή της λογικής και της γραφικής διεπαφής (GUI) σύμφωνα με τις απαιτήσεις που δόθηκαν στην εκφώνηση και παράλληλα αναφέρονται οι επεκτάσεις, παραδοχές και περιορισμοί που τέθηκαν στα πλαίσια της υλοποίησης. Η ανάλυση γίνεται πάνω στη γραφική διεπαφή που βλέπει ο χρήστης και με βάση αυτή εξηγείται και η λογική των αλγορίθμων που τρέχουν από πίσω.

2. Εργαλεία και οργάνωση του project

Η υλοποίηση της εφαρμογής έγινε σε JavaFX, με τη χρήση Scene Builder και το IDE που χρησιμοποιήθηκε είναι το Intellij IDEA. Συγκεκριμένα, μέσω του Scene Builder δημιουργήθηκαν τα αρχεία *.fxml* που αφορούν στη διεπαφή και μετά υλοποιήθηκαν κατάλληλοι *controllers* σε JAVA για τον έλεγχο του GUI και την υλοποίηση των λειτουργιών της εφαρμογής.

Το κύριο παράθυρο του GUI περιγράφεται στο αρχείο ***homepage.fxml*** και ελέγχεται από τον κύριο controller ***HomepageController.java*** που περιέχει και το μεγαλύτερο μέρος των λειτουργιών της εφαρμογής. Για την "τρέξιμο" της εφαρμογής "τρέχουμε" την κλάση Main.java που βρίσκεται στο φάκελο src. Στη συνέχεια, για κάθε επιπλέον παράθυρο, δημιουργούμε το *.fxml* αρχείο και τον αντίστοιχο controller, όπως φαίνεται και από τα ονόματα των αρχείων στο project. Όλοι οι controllers βρίσκονται στον **φάκελο src/controllers** μαζί με τις κλάσεις των αντικειμένων που ορίσαμε, όλα τα *.fxml* αρχεία βρίσκονται στον **φάκελο src/resources** και υπάρχει και ο **φάκελος medialab** που έχει μέσα τα *.txt* αρχεία που περιγράφουν τα σενάρια του αεροδρομίου και των πτήσεων με την κατάλληλη δομή και ονομασία.

Στα πλαίσια της αντικειμενοστραφούς λογικής, ορίστηκαν οι 3 βασικές κλάσεις Airport, ParkingSeat και Flight. Η **κλάση Airport** περιγράφει τη δομή του αεροδρομίο και περιέχει τις θέσεις πάρκινγκ και τις πτήσεις που εξυπηρετούνται. Οι θέσεις πάρκινγκ περιγράφονται από ένα *HashMap<Integer, List<ParkingSeat>>* που παίρνει ως key την κατηγορία πάρκινγκ και επιστρέφει μια λίστα από ParkingSeats objects. Οι πτήσεις περιγράφονται από μια λίστα από Flight objects. Πέρα από αυτό στην κλάση Airport υπάρχουν και μέθοδοι απαραίτητες για λειτουργίες της εφαρμογής (όπως *countParkedFlights*, *countAvailableParkingSpaces*), τα ονόματα των οποίων είναι χαρακτηριστικά της λειτουργίας τους. Η **κλάση ParkingSeat** περιλαμβάνει τις ιδιότητες μιας θέσης πάρκινγκ όπως ορίζονται από την εκφώνηση. Επιπλέον, το status της κάθε πτήσης

μπορεί να είναι *parked* ή *free*. Η **κλάση Flight** περιλαμβάνει τις ιδιότητες μιας πτήσης όπως ορίζονται από την εκφώνηση. Μαζί περιέχεται ένας timer για να υπολογίζει την χρονική καθυστέρηση από τη στιγμή που το status μιας πτήση γίνεται από "*landing"* σε *"parked"*. Επιπλέον, το status της κάθε πτήσης μπορεί να πάρει τις παρακάτω τιμές:

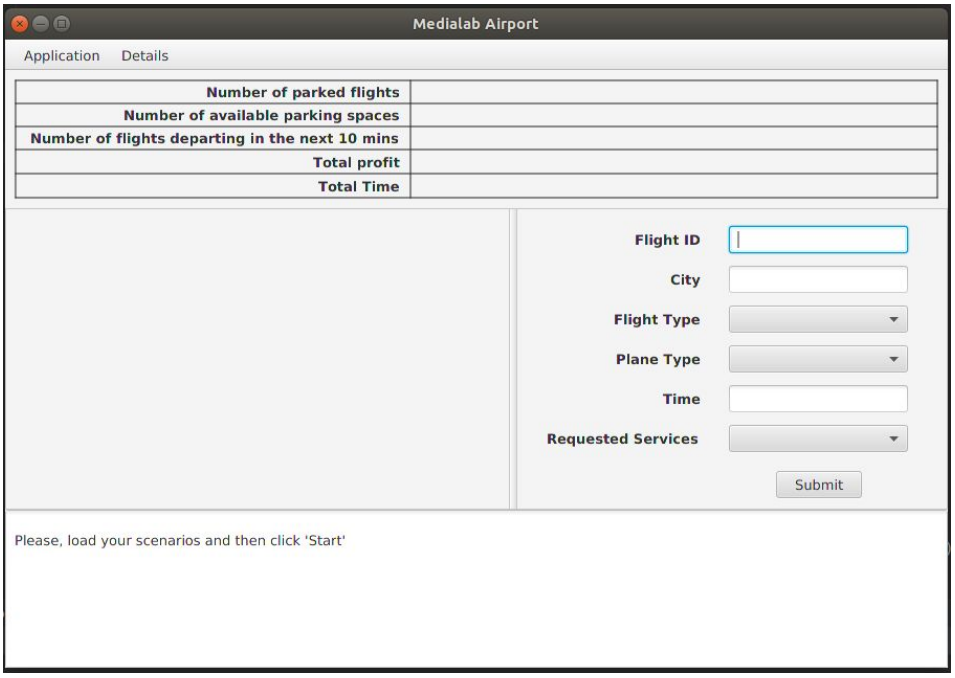
- *parked*
- *holding*
- *landing*
- *departed*
- *ignored,* στη περίπτωση που μια πτήση ανήκει στο αρχικό σενάριο και δεν υπάρχει διαθέσιμη θέση
- *rejected*, για τις περιπτώσεις που το αεροδρόμιο δεν έχει κατάλληλη κατηγορία πάρκινγκ για να φιλοξενήσει μια πτήση. Αυτό συμβαίνει επειδή, όπως θα αναφερθεί και παρακάτω, έχουμε θεωρήσει ότι ένα αεροδρόμιο μπορεί να μην καλύπτει όλες τις κατηγορίες θέσεων πάρκινγκ.

Για μια γενικότερη εικόνα της εφαρμογής, αξίζει να σημειωθεί ότι στον βασικό controller (HomepageController) ορίζεται ένα global object της κλάσης Airport το οποίο θα είναι και το αντικείμενο πάνω στο οποίο θα αποθηκεύονται και θα επεξεργάζονται τα στοιχεία της εφαρμογής και πάνω στο οποίο θα εκτελούνται όλες οι λειτουργίες που περιγράφονται παρακάτω. Επίσης, για όλες τις λειτουργίες που σχετίζονται με τον χρόνο υλοποιούνται οι κατάλληλοι timers, που περιγράφονται στην ενότητα 4.

3. Περιγραφή γραφικής διεπαφής και σχετικών λειτουργιών

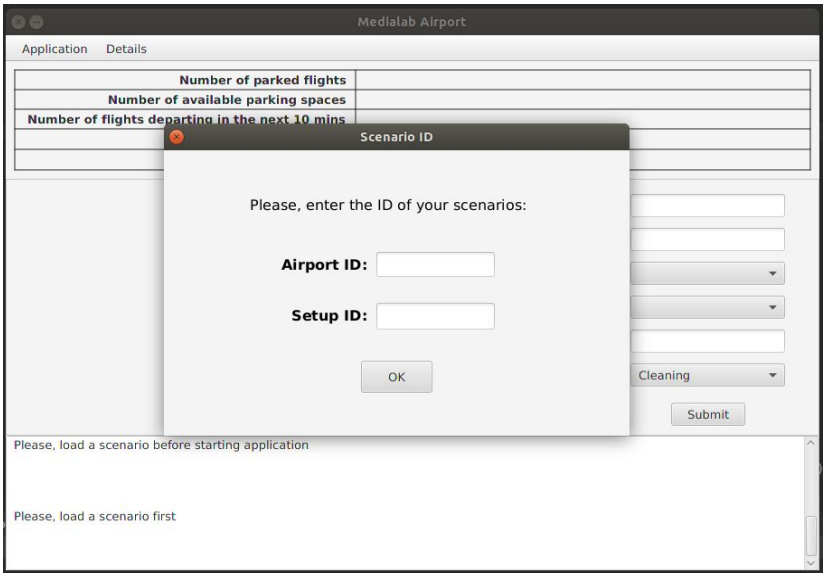
Η περιγραφή θα ακολουθήσει τη σειρά με την οποία ο χρήστης αλληλεπιδρά με την εφαρμογή.

a. Homepage



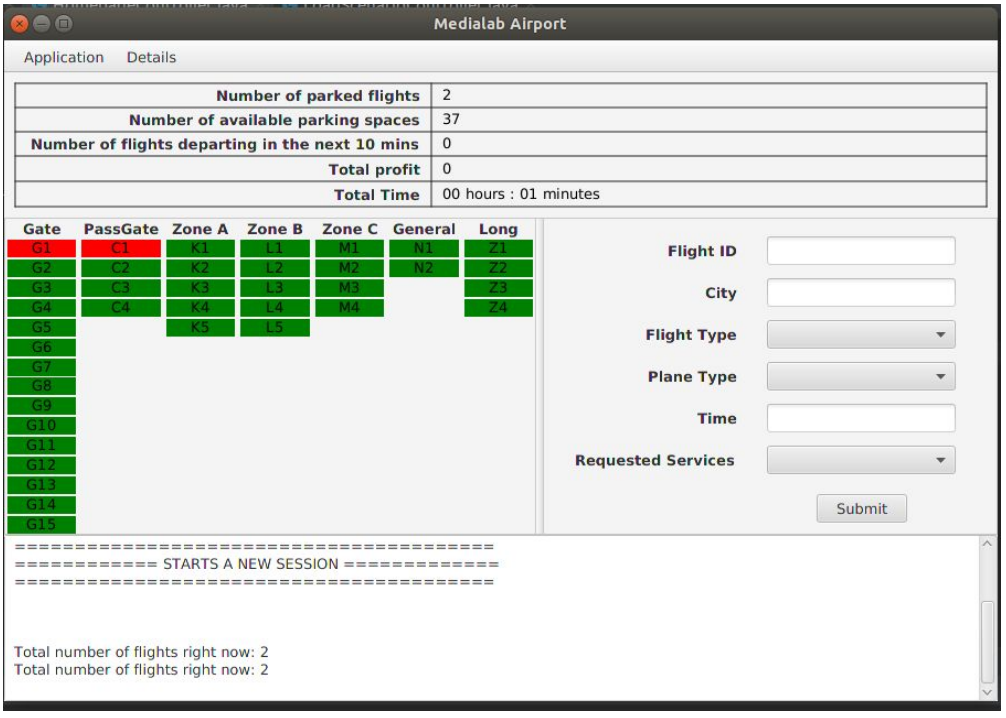
Το πρώτο παράθυρο που εμφανίζεται στον χρήστη είναι το homepage, όπως φαίνεται παραπάνω. Περιέχει όλα τα συστατικά που ορίζει η εκφώνηση της άσκησης και βρίσκεται στο κέντρο της οθόνης. Το αρχικό μήνυμα προτρέπει τον χρήστη να φορτώσει τα σενάρια που θέλει και μετά να πιέσει start. Σε κάθε περίπτωση η εφαρμογή δεν επιτρέπει στον χρήστη να ενεργοποιήσει τα κουμπιά με λάθος σειρά. Επίσης, ο χρήστης δεν μπορεί να φορτώσει κανένα από τους πίνακες στο Menu/Details πριν κάνει load ένα σενάριο και πατήσει start. Για αυτό τον σκοπό τα κουμπιά ενεργοποιούνται μόνο όταν τηρείται η κατάλληλη σειρά. Σε αντίθετη περίπτωση, εμφανίζονται κατάλληλα μηνύματα υπενθύμισης της σωστής σειράς.

b. Load button



Με το που ο χρήστης πατήσει το κουμπί Load από το Menu/Application, εμφανίζεται ένα popup παράθυρο και ο χρήστης εισάγει τα ID των σεναρίων. Οι λειτουργίες και η εμφάνιση του παραθύρου περιγράφεται στα LoadScenarioController.java και loadScenario.fxml αντίστοιχα. Η εγκυρότητα των τιμών που εισάγει ο χρήστης ελέγχεται με το που πατήσει το κουμπί start και αν δεν είναι έγκυρες του ζητείται να φορτώσει έγκυρα σενάρια.

c. Start button



Αφού έχει φορτώσει έγκυρα σενάρια, ο χρήστης καλείται να πατήσει το κουμπί Start από το Menu/Application. Όπως φαίνεται στην παρακάτω εικόνα, με το που πατηθεί το Start, εμφανίζεται το κατάλληλο γραφικό στη μέση αριστερά με την περιγραφή των θέσεων πάρκινγκ του αεροδρομίου και αρχικοποιούνται οι γενικές πληροφορίες στο πάνω μέρος της οθόνης. Αναλυτικότερα, η μέθοδος startButtonClicked() αρχικοποιεί όλες τις μεταβλητές της εφαρμογής και τους timers και ορίζει κάθε φορά ένα νέο αντικείμενο της κλάσης Airport. Στη συνέχεια, ελέγχει την εγκυρότητα των τιμών ID που δόθηκαν κατά το load, διαβάζει τα αρχεία σεναρίων και αρχικοποιεί τον HashMap για τις θεσεις παρκινγκ και τη λίστα με τις αρχικές πτήσεις. Για τις πτήσεις συγκεκριμένα, καλείται η μέθοδος checkParkingAvailabilityInitial(), η οποία αρχικοποιεί το status των πτήσεων σύμφωνα με τον Πίνακα που δόθηκε στη δεύτερη σελίδα της εκφώνησης. Τέλος, με το πάτημα του Start, κατασκευάζεται η γραφική παρουσίαση των θέσεων πάρκινγκ, η οποία αναλύεται παρακάτω.

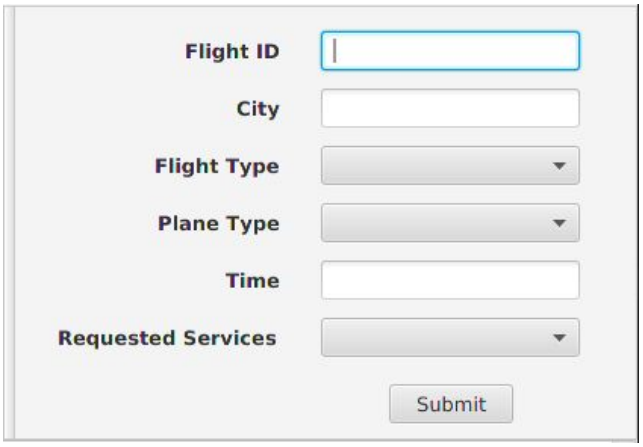
d. Πάνω μέρος (General information)

Number of parked flights	4
Number of available parking spaces	35
Number of flights departing in the next 10 mins	2
Total profit	23700.0
Total Time	01 hours : 06 minutes

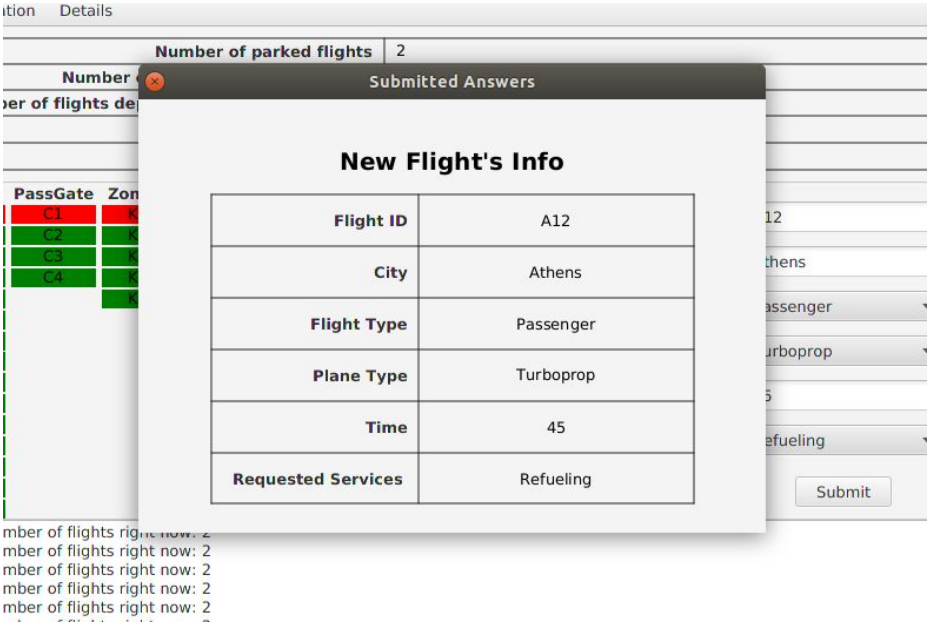
Στο πάνω μέρος της οθόνης, κάτω από το Menu bar, βρίσκεται το πλαίσιο με τις γενικές πληροφορίες που ζητείται στην εκφώνηση, οι οποίες ανανεώνονται μέσω των timers. Στην εικόνα φαίνεται ένα στιγμιότυπο του πλαισίου.

e. Μεσαίο δεξιό τμήμα (Submit form)

Στο μεσαίο δεξιά τμήμα του homepage υπάρχει η φόρμα για να μπορεί ο χρήστης να καταχωρεί νέες πτήσεις. Η φόρμα φαίνεται παρακάτω:



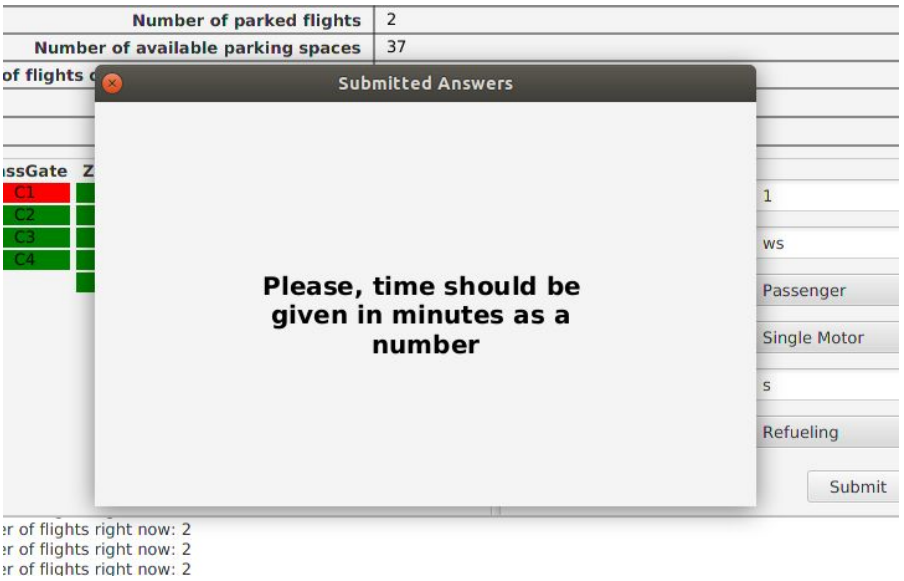
Με την υποβολή έγκυρων στοιχείων, εμφανίζεται στον χρήστη το παρακάτω παράθυρο που περιλαμβάνει όλες τις πληροφορίες που καταχωρήθηκαν για την νέα πτήση:

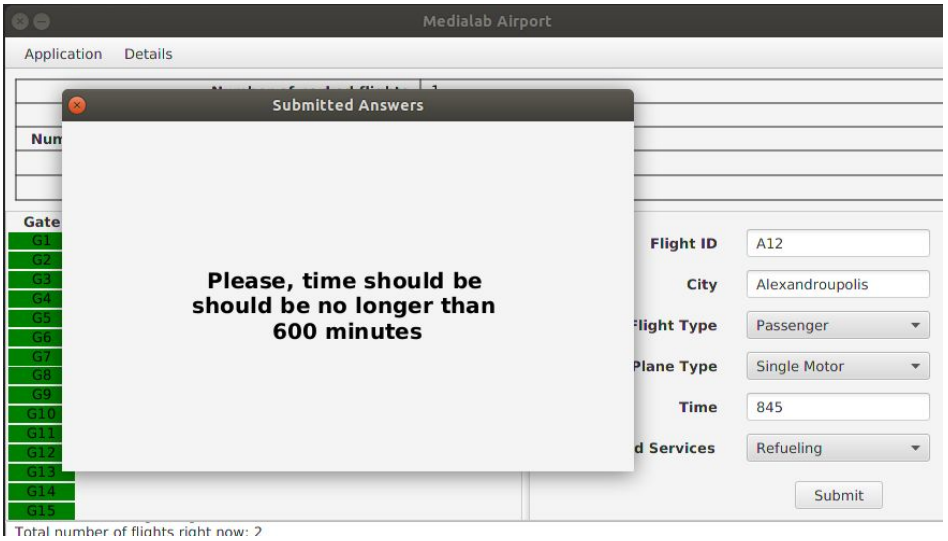


Κατά την υποβολή μιας νέας πτήσης, γίνονται οι παρακάτω έλεγχοι και εμφανίζονται κατάλληλα παράθυρα σε περίπτωση μη έγκυρων υποβολών. Τα νέα παράθυρα περιγράφονται από τα αρχεία με όνομα SubmittedAnswers και SubmittedWrongAnswers και τους αντίστοιχους controllers.

- Έλεγχος για καταχώρηση όλων των τιμών που απαιτούνται:

- Έλεγχος για καταχώρηση έγκυρων τιμών της ώρας, δηλαδή πρέπει να είναι integer και να είναι μικρότερη από 600 λεπτά = 10 ώρες:



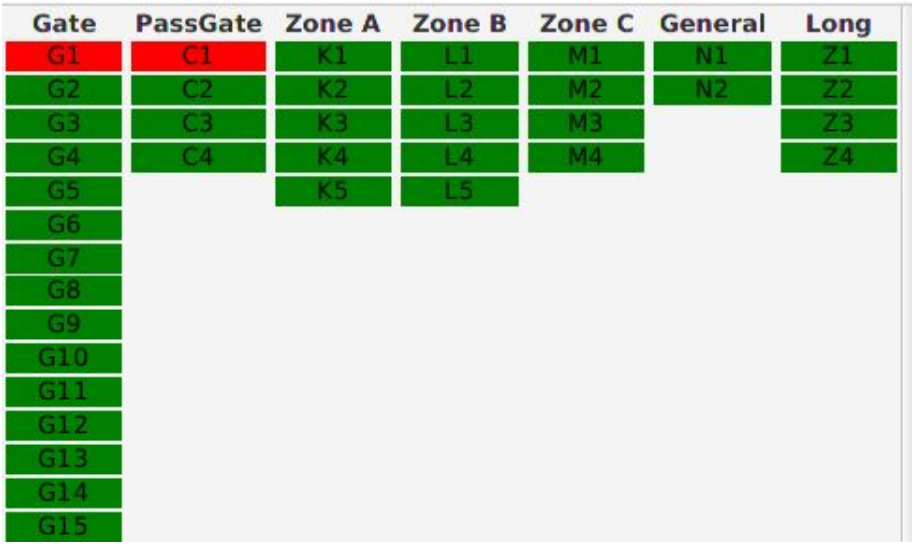


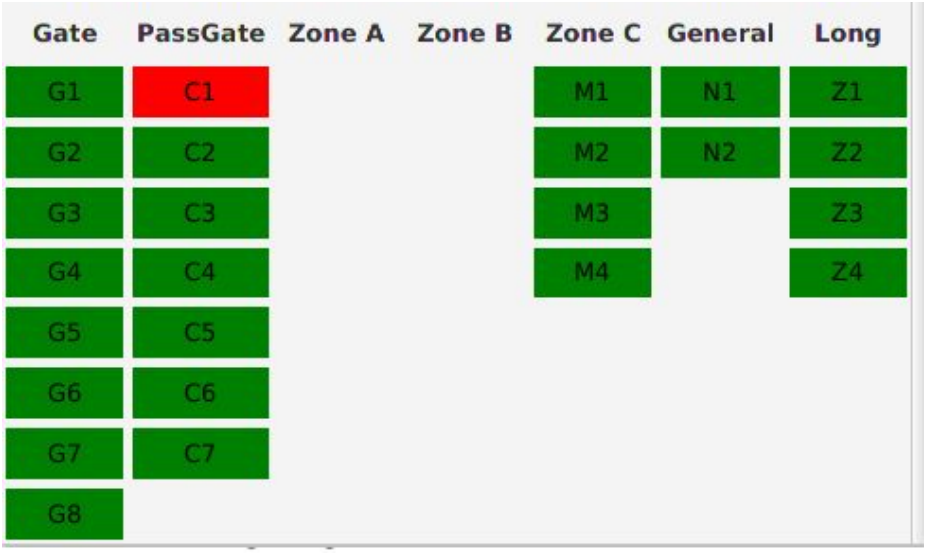
Μετά τους ελέγχους, ακολουθείται παρόμοια λογική με αυτή του Start button και καλείται η συνάρτηση checkParkingAvailabilityAfter(), οποία αρχικοποιεί το status των πτήσεων σύμφωνα με τον Πίνακα και αναθέτει τιμές *landing, holding* ή *rejected* στο status κάθε πτήσης.

f. Μεσαίο αριστερό τμήμα (Graphic GUI)

Με το πάτημα του Start button και αφού διαβαστούν τα σενάρια και γίνουν οι απαραίτητες αρχικοποιήσεις, κατασκευάζεται μια γραφική παρουσίαση των διαθέσιμων χώρων στάθμευσης που υπάρχουν σε κάθε αεροδρόμιο. Η γραφική απεικόνιση αποτελείται

από ένα AnchorPane που ορίζεται μέσα στο FXML και ένα GridPane που κατασκευάζεται κάθε φορά με τα υπάρχοντα δεδομένα. Στη πρώτη γραμμή εμφανίζονται τα ονόματα των κατηγοριών και από κάτω υπάρχουν όλες οι θέσεις με το ID τους και έχουν πράσινο ή κόκκινο χρώμα, ανάλογα με τη διαθεσιμότητά τους. Οι στήλες του GridPane είναι πάντοτε 7, όσες και οι κατηγορίες πάρκινγκ. Στη περίπτωση που ένα αεροδρόμιο δεν έχει κάποια κατηγορία παρκινγκ, τότε το GridPane έχει κενή την αντίστοιχη στήλη. Αυτό ορίζεται έτσι, ώστε να είναι εύκολα εμφανές ποιες κατηγορίες διαθέτει το εκάστοτε αεροδρόμιο. Όσον αφορά τις γραμμές, είναι όσες και οι διαθέσιμες θέσεις από κάθε κατηγορία και το μέγεθος του κάθε κουτιού καθορίζεται με βάση την κατηγορία με τη μέγιστη τιμή θέσεων. Όταν φορτώνεται ένα νέο σενάριο, το GridPane προσαρμόζεται ανάλογα. Το ίδιο συμβαίνει, με την βοήθεια ενός timer, όταν εισέρχεται μια νέα πτήση ή αλλάζει το parking status των υπαρχόντων πτήσεων. Όλα αυτά φαίνονται στις παρακάτω εικόνες:





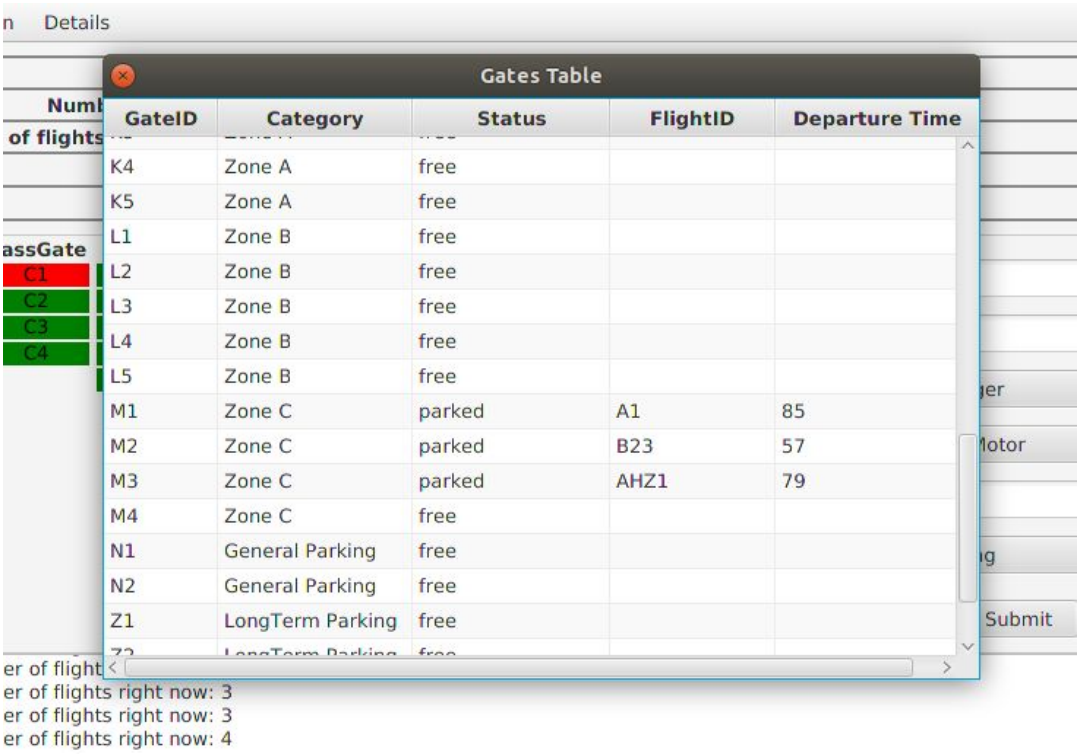
Gate	PassGate	Zone A	Zone B	Zone C	General	Long
G1	C1	K1	L1	M1	N1	Z1
G2	C2	K2	L2	M2	N2	Z2
G3	C3	K3	L3	M3	N3	Z3
G4	C4	K4	L4	M4	N4	Z4
G5	C5	K5	L5	M5	N5	Z5
G6	C6	K6	L6	M6	N6	Z6
G7	C7	K7	L7	M7	N7	Z7
G8	C8	K8	L8	M8	N8	Z8
G9	C9	K9	L9	M9	N9	Z9
G10	C10	K10	L10	M10	N10	Z10

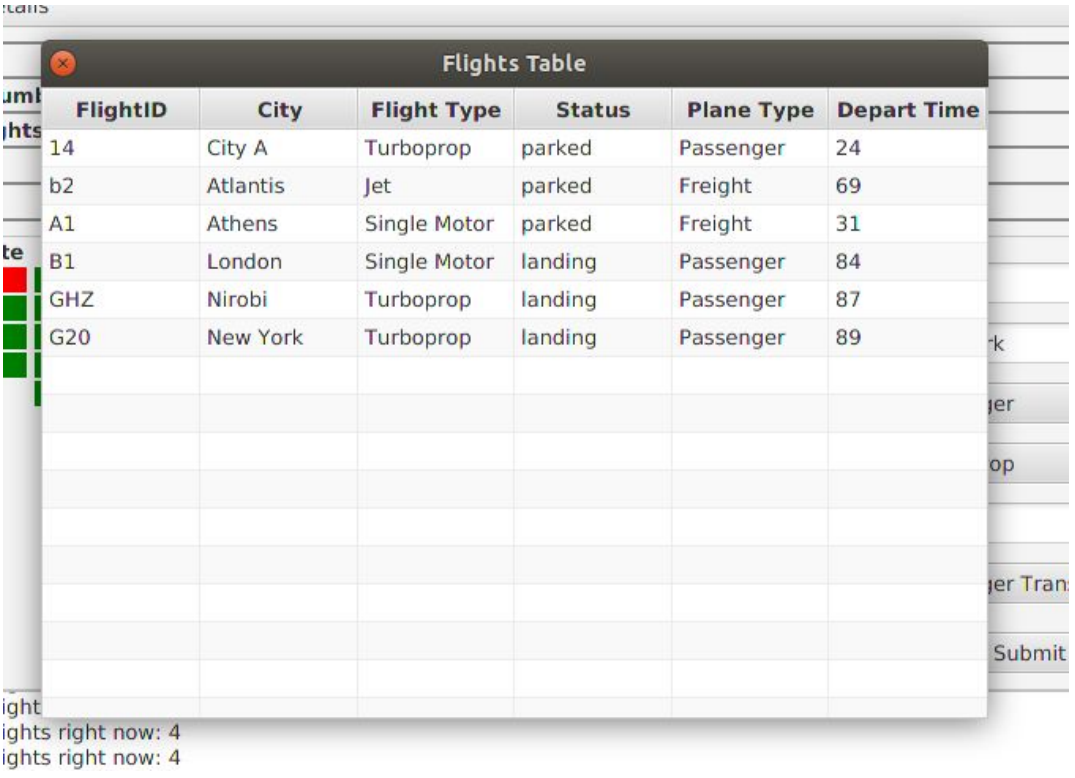
g. Κάτω μέρος (Log screen)

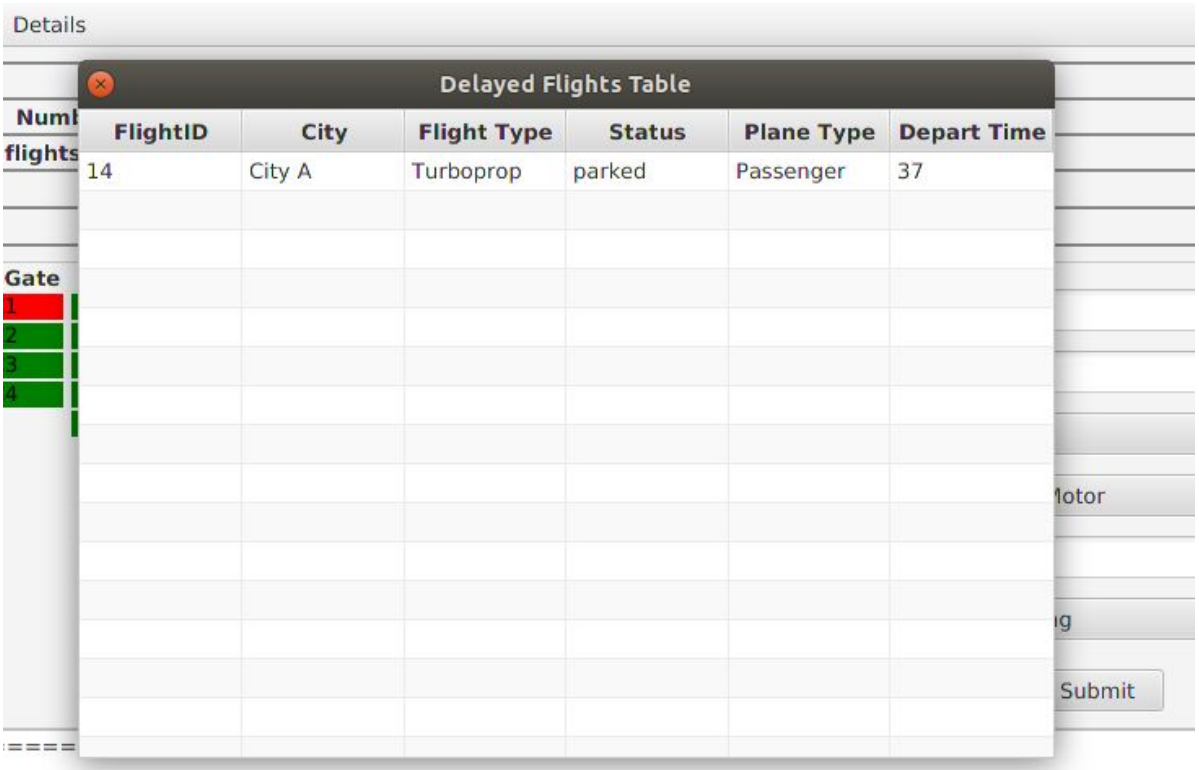
Στο κάτω μέρος της οθόνης υπάρχει το Log screen, ένα TextArea (read-only) όπου εμφανίζονται τα απαραίτητα μηνύματα για επικοινωνία με τον χρήστη. Το Log screen συνδέθηκε με το output του προγράμματος με αποτέλεσμα να εμφανίζονται εκεί όλες οι εντολές της μορφής System.out.println().

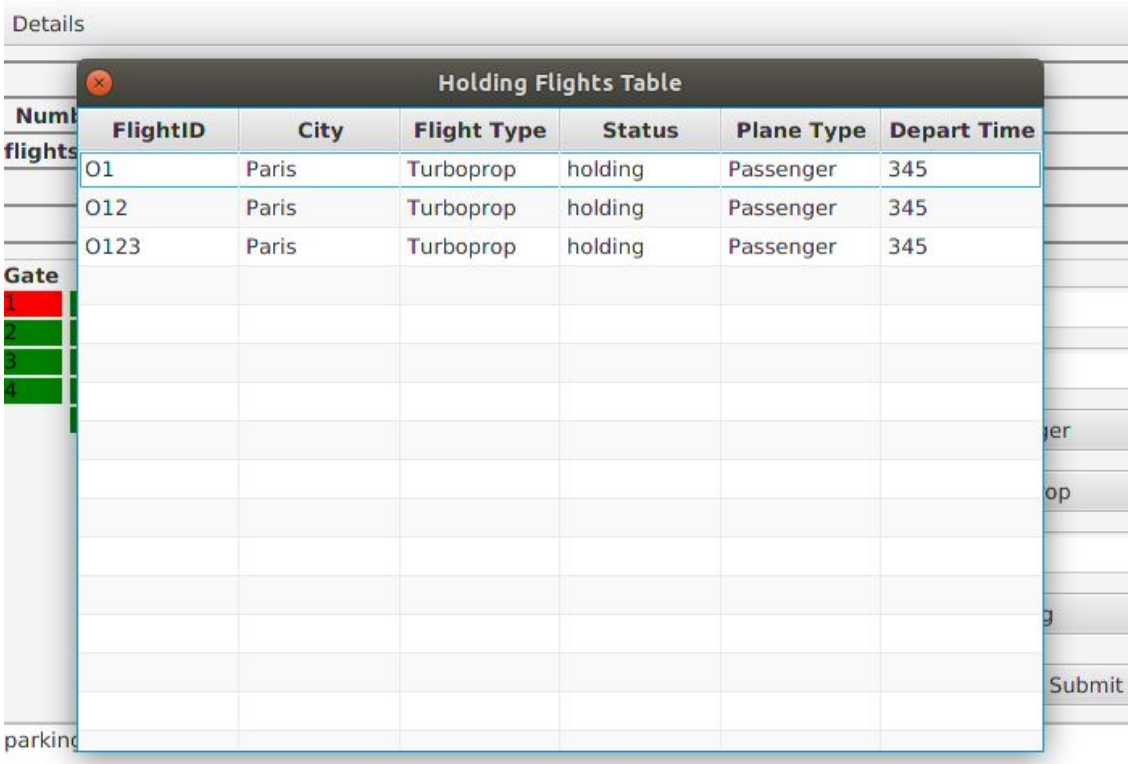
h. Details buttons

Στο Menu bar στο section Details υπάρχουν τα κουμπιά Gates, Flights, Delayed, Holding, Next Departures, όπου πατώντας τα εμφανίζονται κατάλληλα popups παράθυρα όπως περιγράφονται από την εκφώνηση. Τα popups και οι λειτουργίες τους περιγράφονται από τα σχετικά .fxml αρχεία και τους αντίστοιχους controllers (πχ. gateView.fxml και GateViewController.java), οι οποίοι καλούνται μέσα από τον HomepageController. Παρακάτω, φαίνονται όλα σε εικόνες.

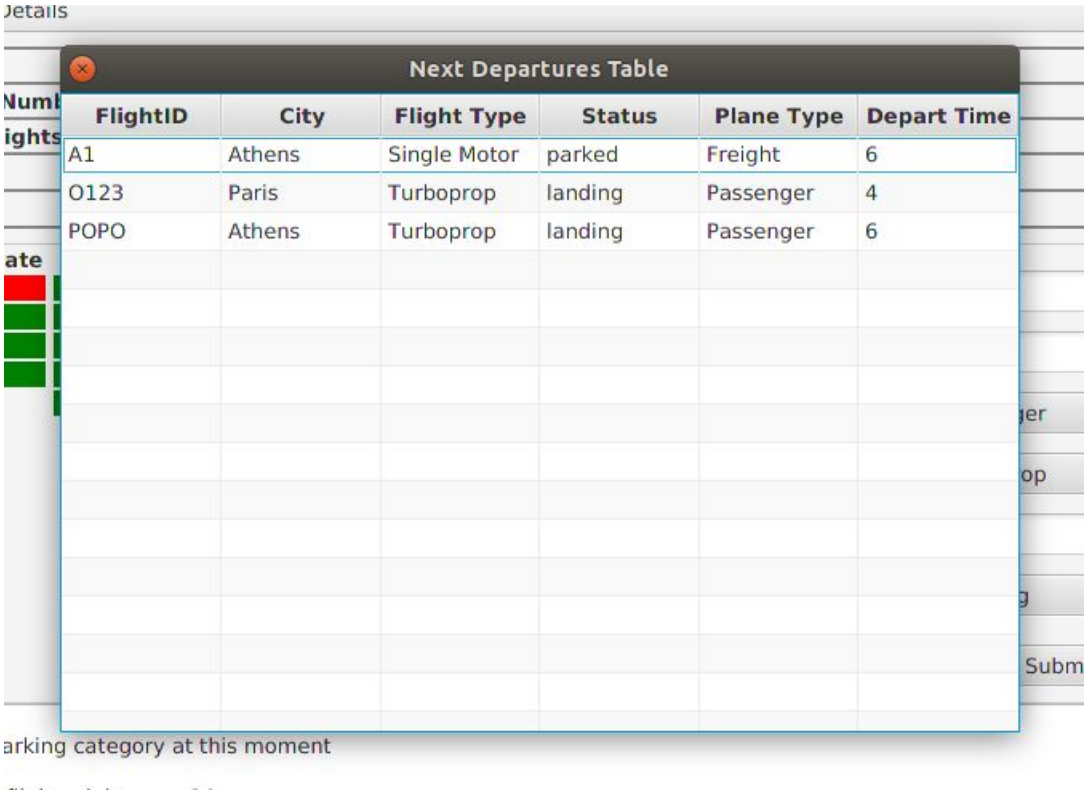

Details
Gates Table
GateID
Category
Status
FlightID
Departure Time
K4 Zone A free
K5 Zone A free
L1 Zone B free
L2 Zone B free
L3 Zone B free
L4 Zone B free
L5 Zone B free
M1 Zone C parked A1 85
M2 Zone C parked B23 57
M3 Zone C parked AHZ1 79
M4 Zone C free
N1 General Parking free
N2 General Parking free
Z1 LongTerm Parking free
Submit
er of flights right now: 3
er of flights right now: 3
er of flights right now: 4


Flights Table
FlightID
City
Flight Type
Status
Plane Type
Depart Time
14 City A Turboprop parked Passenger 24
b2 Atlantis Jet parked Freight 69
A1 Athens Single Motor parked Freight 31
B1 London Single Motor landing Passenger 84
GHZ Nirobi Turboprop landing Passenger 87
G20 New York Turboprop landing Passenger 89
Submit
ights right now: 4
ights right now: 4





parking category at this moment



i. Exit button

Με το Exit button τερματίζει η εφαρμογή.

4. Timers

Με τη χρήση timers καταφέρνουμε να προγραμματίσουμε όλες τις δυναμικές λειτουργίες της εφαρμογής, αυτές δηλαδή που μεταβάλλουν τα στοιχεία σε σχέση με τον χρόνο. Όλοι οι timers επαναλαμβάνουν τις λειτουργίες τους κάθε 5 seconds = 1 minute στην εφαρμογή σύμφωνα με τη σύμβαση της εκφώνησης. Ορίστηκαν 3 timers μέσα στον HomepageController. Ο **myTimer** της κλάσης AnimationTimer, ο οποίος είναι ο timer που μετράει και εμφανίζει ο χρόνο που δουλεύει η εφαρμογή από τη στιγμή που άνοιξε μέχρι και το exit(). Ο **ControlTimer** της κλάσης Timer, ο οποίος είναι και ο βασικότερος timer της εφαρμογής, αφού σχετίζεται με τον έλεγχο των πτήσεων. Συγκεκριμένα, ο ControlTimer υπολογίζει τις πτήσεις που υπάρχουν κάθε στιγμή στο αεροδρόμιο. Μέσω της reduceDepartureTime() της κλάσης Flight, υπολογίζει και μειώνει κάθε λεπτό το χρόνο αναχώρησης των παρκαρισμένων πτήσεων. Υπολογίζει τις διαθέσιμες θέσεις παρκινγκ. Υπολογίζει τις πτήσεις που αναχωρούν στα επόμενα 10 λεπτά. Υπολογίζει το κόστος της κάθε πτήσης πριν αναχωρήσει. Διαγράφει τις πτήσεις που έχουν αναχωρήσει και το κόστος έχει ήδη καταμετρηθεί. Και τέλος, τσεκάρει για πτήσεις που είναι σε κατάσταση “holding” αν μπορούν να μπουν σε κατάσταση “landing”. Ο τρίτος timer είναι ο gridGuiTimer της κλάσης Timeline, ο οποίος κάθε λεπτό (5 sec) ανανεώνει τη γραφική αναπαράσταση των θέσεων στάθμευσης του αεροδρομίου που βρίσκεται στο δεξιό μέρος στη μέση του homepage.

5. Λεπτομέρειες υλοποίησης

Στην ενότητα αυτή αναφέρονται κάποιες βασικές απαιτήσεις της εκφώνησης και πώς αυτές αντιμετωπίστηκαν στην εργασία.

- Για την επιλογή χώρου στάθμευσης επιλέγουμε τη πρώτη θέση που καλύπτει τις συνθήκες του Πίνακα που δίνεται στη δεύτερη σελίδα της εκφώνησης. Επίσης, με το που καλυφθεί μια θέση από πτήση, ακόμα και όσο αυτή είναι σε “landing” status, στην γραφική απεικόνιση η αντίστοιχη θέση στάθμευσης φαίνεται με κόκκινο χρώμα.

- Για να γίνει μια πτήση από “landing” σε “parked”, με το που γίνει “landing” καλείται ο timer της κλάσης Flight, ο οποίος μετράει αντίστροφα από το parkingTime μέχρι το μηδέν και τότε κάνει την πτήση “parked”. Όπου parkingTime, είναι ο χρόνος σύμφωνα με την εκφώνηση που κάνει κάθε τύπος αεροπλάνου να παρκάρει.

- Οι καθυστερήσεις των πτήσεων υλοποιούνται ως εξής. Με το που καταχωρείται μια πτήση στο αεροδρόμιο, ταυτόχρονα της ανατίθεται ένας τυχαίος αριθμός από το 0 έως το 3, όπου 0 = καθυστέρηση, 1 = αναχώρηση 10-20 λεπτά νωρίτερα, 2 = 25 λεπτά νωρίτερα, 4 = αναχωρεί στην προγραμματισμένη ώρα.

6. Σχόλια

Επιλέγουμε την κλάση **Flight** για να την τεκμηριώσουμε σύμφωνα με τις προδιαγραφές του εργαλείου **Javadoc**.

Τέλος, θα αναφέρουμε πιθανές αλλαγές που δεν υλοποιήθηκαν, αλλά θα βελτίωναν την εφαρμογή:

- Η εφαρμογή να δίνει τη δυνατότητα σε κάθε νέα πτήση να επιλέγει πάνω από ένα service. Τώρα, μπορεί να επιλέξει μόνο ένα.

- Να γίνει το homepage παράθυρο resizeable. Τώρα, αν πατηθεί το κουμπί για να μεγαλώσει το παράθυρο, τα μεγέθη δεν προσαρμόζονται κατάλληλα.