

ΣΥΣΤΗΜΑΤΑ ΒΑΣΕΩΝ ΔΕΔΟΜΕΝΩΝ II

PROJECT 2023-24

ΕΦΑΡΜΟΓΗ ΚΡΑΤΗΣΕΩΝ ΑΕΡΟΠΟΡΙΚΗΣ ΕΤΑΙΡΕΙΑΣ



Ον/μο και ΑΕΜ:

**ΑΝΑΓΝΩΣΤΟΠΟΥΛΟΣ ΒΑΣΙΛΕΙΟΣ
03153**

ΓΕΩΡΓΙΤΖΙΚΗ ΓΑΡΥΦΑΛΙΑ 03218

Πίνακας περιεχομένων

1. ΘΕΜΑ ΕΡΓΑΣΙΑΣ ΚΑΙ ΛΕΙΤΟΥΡΓΙΚΟΤΗΤΑ ΕΦΑΡΜΟΓΗΣ	2
2. ΠΕΡΙΓΡΑΦΗ ΒΑΣΗΣ ΔΕΔΟΜΕΝΩΝ ΤΗΣ ΕΦΑΡΜΟΓΗΣ.....	3
3. ΔΙΑΓΡΑΜΜΑ ΟΝΤΟΤΗΤΩΝ ΣΥΣΧΕΤΙΣΕΩΝ (ΔΟΣ) ΜΕ ΤΟ ER2SQL	5
4. ΜΕΤΑΤΡΟΠΗ ΔΟΣ ΣΕ ΣΧΕΣΙΑΚΟ ΣΧΗΜΑ (ΣΣ)	6
 4A. ΣΧΕΣΙΑΚΟ ΣΧΗΜΑ (ΣΣ)	6
 4B. ΕΠΕΞΗΓΗΣΗ ΣΧΕΣΙΑΚΟΥ ΣΧΗΜΑΤΟΣ (ΣΣ).....	7
5. ΚΑΤΑΓΡΑΦΗ ΣΥΝΑΡΤΗΣΙΑΚΩΝ ΕΞΑΡΤΗΣΕΩΝ ΤΟΥ ΣΧΕΣΙΑΚΟΥ ΣΧΗΜΑΤΟΣ ΚΑΙ ΚΑΝΙΚΟΠΟΙΗΣΗ ΤΟΥ ΣΣ ΣΕ BCNF ή 3NF.....	9
 5A. ΣΥΝΑΡΤΗΣΙΑΚΕΣ ΕΞΑΡΤΗΣΕΙΣ ΤΟΥ ΣΣ.....	9
 5B. ΒΗΜΑ-ΒΗΜΑ ΚΑΝΟΝΙΚΟΠΟΙΗΣΗ ΣΕ 3NF	10
6. ΥΛΟΠΟΙΗΣΗ ΤΟΥ ΣΣ ΣΤΗΝ POSTGRESQL ΚΑΙ ΕΙΣΑΓΩΓΗ ΕΓΓΡΑΦΩΝ ΣΤΟΥΣ ΠΙΝΑΚΕΣ.....	13
 6A. ΥΛΟΠΟΙΗΣΗ ΤΟΥ ΣΣ ΣΤΗΝ POSTGRESQL	13
 6B. ΕΙΣΑΓΩΓΗ ΕΓΓΡΑΦΩΝ ΣΤΟΥΣ ΠΙΝΑΚΕΣ	20
7. ΟΔΗΓΙΕΣ ΕΓΚΑΤΑΣΤΑΣΗΣ ΤΗΣ ΕΦΑΡΜΟΓΗΣ	27
8. ΕΞΗΓΗΣΗ ΤΟΥ BACKEND ΠΑΝΩ ΣΤΗΝ ΕΦΑΡΜΟΓΗ ΚΑΙ ΟΔΗΓΙΕΣ ΧΡΗΣΗΣ	38
9. ΒΙΒΛΙΟΓΡΑΦΙΑ	114

1. ΘΕΜΑ ΕΡΓΑΣΙΑΣ ΚΑΙ ΛΕΙΤΟΥΡΓΙΚΟΤΗΤΑ ΕΦΑΡΜΟΓΗΣ

Το θέμα της εργασίας είναι μία εφαρμογή Android της αεροπορικής εταιρείας «FlyNow», στηριζόμενη σε βάση δεδομένων PostgreSQL, που αφορά κράτηση αεροπορικών πτήσεων και αυτοκινήτων. Η βάση δεδομένων της εφαρμογής, μπορεί να αποθηκεύσει πληροφορία και λεπτομέρειες για τους επιβάτες, τις κρατήσεις, τις πτήσεις, τα αεροδρόμια, τα αεροπλάνα, τα μοντέλα αεροπλάνων, τις θέσεις, τις αποσκευές, τα κατοικίδια και τα αυτοκίνητα. Συγκεκριμένα, η εφαρμογή αναφέρεται σε χρήστες οι οποίοι θέλουν να αναζητήσουν πτήσεις και να πραγματοποιήσουν κρατήσεις πτήσεων (και αυτοκινήτου) με την συγκεκριμένη αεροπορική εταιρεία. Επίσης, οι χρήστες μπορούν να επεξεργαστούν τα δεδομένα κάποιας κράτησής τους, να ακυρώσουν την κράτηση ή να κάνουν check-in στην πτήση τους.



Logo Αεροπορικής Εταιρείας “FlyNow”

2. ΠΕΡΙΓΡΑΦΗ ΒΑΣΗΣ ΔΕΔΟΜΕΝΩΝ ΤΗΣ ΕΦΑΡΜΟΓΗΣ

Σύμφωνα με τη Βάση Δεδομένων της εφαρμογής(αεροπορικής εταιρείας) «FlyNow» οι πληροφορίες και λεπτομέρειες(δεδομένα) που πρέπει να αποθηκεύονται παρουσιάζονται παρακάτω.

Για κάθε **επιβάτη** αποθηκεύεται ο κωδικός επιβάτη, ο οποίος είναι μοναδικός, το όνομά του, το επώνυμό του, η ημερομηνία γέννησης του, το φύλο του, το ηλεκτρονικό ταχυδρομείο του και ο αριθμός τηλεφώνου του.

Στην συνέχεια, κάθε **κράτηση** προσδιορίζεται από τον κωδικό κράτησης, το Wifi On Board, εάν έχει επιλεχθεί για την κράτηση, τον τύπο κατοικίδιου (Small, Medium, Large) av υπάρχει και την τιμή της κράτησης.

Για κάθε **πτήση**, καταχωρείται ο κωδικός πτήσης, το κόστος της (EconomyPrice, FlexPrice, BusinessPrice), η ώρα αναχώρησης, η ώρα άφιξης, η διάρκεια του ταξιδιού και η ημερομηνία της πτήσης. Ένας επιβάτης σε μία πτήση θα έχει μία ακριβώς κράτηση. Ένας επιβάτης σε μία κράτηση θα έχει τουλάχιστον μία πτήση (πτήσεις με ανταπόκριση). Μία κράτηση σε μία πτήση αφορά τουλάχιστον έναν επιβάτη. Επίσης ο συνδυασμός επιβάτης, κράτηση και πτήση έχουν τον τύπο κλάσης (πχ. Economy Class, Flex Class, Business Class), τις αποσκευές των επιβατών μαζί με τον αριθμό τους για κάθε τύπο αποσκευής(πχ. baggage8kg(δικαιούνται όλοι από μία(δωρεάν) δεν αποθηκεύεται στη βάση), baggage23kg, baggage32kg, την θέση που είναι κρατημένη από έναν επιβάτη και το check-in της κράτησης για κάθε πτήση.

Έπειτα, για κάθε **αεροδρόμιο**, αποθηκεύεται το όνομα του αεροδρομίου(πχ. για το αεροδρόμιο Ελευθέριος Βενιζέλος το όνομα είναι ATH) και η χώρα στην οποία βρίσκεται. Κάθε

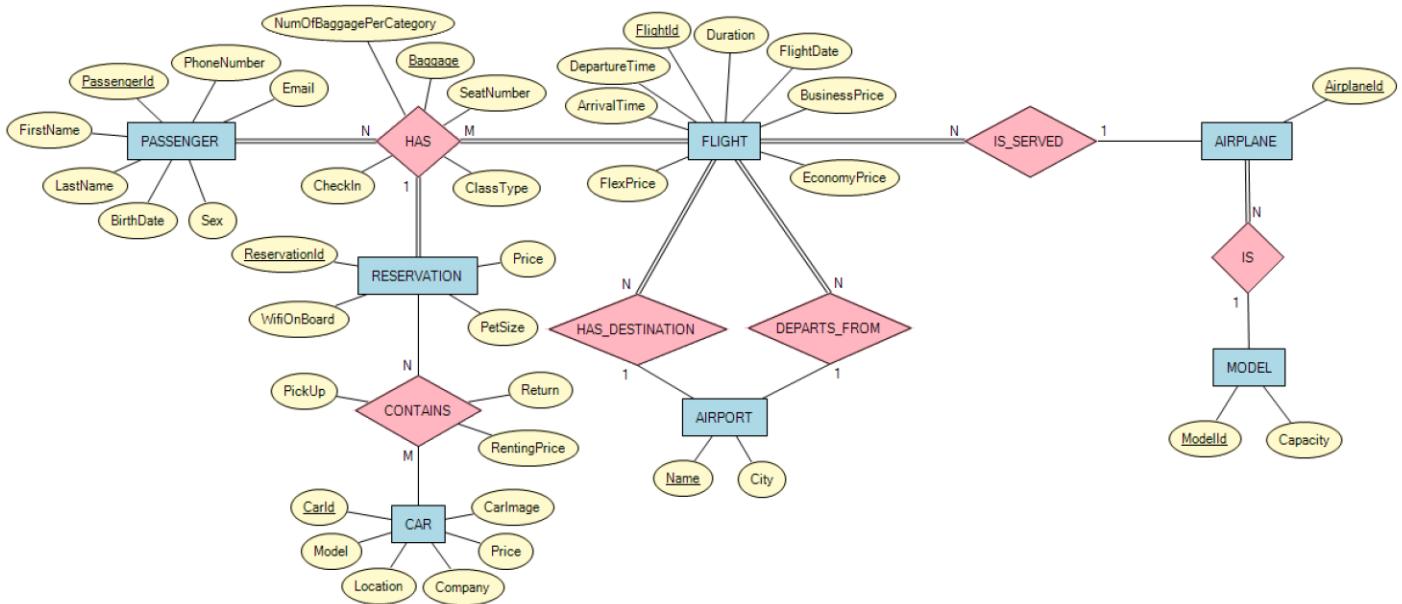
πτήση έχει προορισμό ένα (και μόνο ένα) αεροδρόμιο και αναχωρεί από ένα (και μόνο ένα) αεροδρόμιο.

Για κάθε **αεροπλάνο** αποθηκεύεται ο κωδικός αεροπλάνου, ο οποίος το διαχωρίζει από άλλα αεροπλάνα ίδιου μοντέλου. Κάθε πτήση εξυπηρετείται σίγουρα από ένα αεροπλάνο. Ένα αεροπλάνο μπορεί να εξυπηρετεί πολλές πτήσεις.

Για κάθε **αυτοκίνητο** αποθηκεύεται ο κωδικός του αυτοκινήτου, το μοντέλο, η τοποθεσία παραλαβής/επιστροφής και η εταιρεία ενοικίασης αυτοκινήτων. Ένα αυτοκίνητο μπορεί να ανήκει σε πολλές κρατήσεις, ενώ η κράτηση μπορεί να περιέχει πολλά ενοικιαζόμενα αυτοκίνητα αποθηκεύοντας επίσης την ημερομηνία παραλαβής, την ημερομηνία επιστροφής του αυτοκινήτου καθώς και η συνολική τιμή της ενοικίασης του αυτοκινήτου.

Τέλος, το **μοντέλο** κάθε αεροπλάνου αναγνωρίζεται από τον κωδικό μοντέλου(π.χ. Airbus A320-200) και την χωρητικότητα του, ενώ κάθε αεροπλάνο είναι ένα συγκεκριμένο μοντέλο.

3. ΔΙΑΓΡΑΜΜΑ ΟΝΤΟΤΗΩΝ ΣΥΣΧΕΤΙΣΕΩΝ (ΔΟΣ) ΜΕ ΤΟ ER2SQL



4. ΜΕΤΑΤΡΟΠΗ ΔΟΣ ΣΕ ΣΧΕΣΙΑΚΟ ΣΧΗΜΑ (ΣΣ)

4A. ΣΧΕΣΙΑΚΟ ΣΧΗΜΑ (ΣΣ)

PASSENGER(PassengerId, FirstName, LastName, BirthDate, Sex, Email, PhoneNumber)

RESERVATION(ReservationId, PetSize, WifiOnBoard, Price)

CAR(CarId, Model, Location, Company, Price, CarImage)

CONTAINS(CarId, ReservationId, PickUp, Return, RentingPrice)

HAS(PassengerId, ReservationId, FlightId, Baggage, SeatNumber, NumOfBaggagePerCategory, ClassType, CheckIn)

FLIGHT(FlightId, FlightDate, Duration, ArrivalTime, DepartureTime, EconomyPrice, FlexPrice, BusinessPrice, ArrivalAirport, DepartureAirport, Airplane)

AIRPORT(Name, City)

AIRPLANE(AirplaneId, Model)

MODEL(ModelId, Capacity)

4B. ΕΠΕΞΗΓΗΣΗ ΣΧΕΣΙΑΚΟΥ ΣΧΗΜΑΤΟΣ (ΣΣ)

Αναλύοντας τις οντότητες του ER σε σχέσεις τοποθετούμε τα γνωρίσματά τους όπως φαίνεται παραπάνω, το primary key της σχέσης υπογραμμίζεται και οι τιμές τους είναι όλες διαφορετικές μεταξύ τους σε όλες τις πλειάδες. Επίσης τα primary keys σε μία σχέση έχουν περιορισμό στήλης “**NOT NULL**”, δηλαδή να μην πάρει μία στήλη(που αφορά primary key) **NULL** τιμές.

Ξεκινώντας, για την σχέση **PASSENGER** το **PassengerId** είναι το primary key καθώς και το **PhoneNumber** και το **Email** έχουν unique τιμές σε κάθε μία από τις πλειάδες.

Για την σχέση **RESERVATION** το **ReservationId** είναι το primary key. Το **PetSize** μπορεί να πάρει και **NULL** τιμές καθώς κάποιος μπορεί να μην θέλει να ταξιδέψει με κατοικίδιο.

Για την σχέση **FLIGHT** το **FlightId** είναι το primary key, περιλαμβάνει 2 foreign keys το **ArrivalAirport** και το **DepartureAirport** τα οποία δείχνουν στην σχέση **AIRPORT** και συγκεκριμένα στο primary key **Name**. Επίσης περιλαμβάνει ακόμη 1 foreign key το **Airplane** το οποίο δείχνει στην σχέση **AIRPLANE** και συγκεκριμένα στο primary key **Name**.

Για την σχέση **AIRPORT** το **Name** είναι το primary key.

Για την σχέση **AIRPLANE** το **AirplaneId** είναι το primary key. Επίσης περιλαμβάνει 1 foreign key, όπου το **Model** δείχνει στην σχέση **MODEL** και συγκεκριμένα στο primary key **ModelId**.

Για την σχέση **MODEL** το **ModelId** είναι το primary key.

Για την σχέση **CAR** το **CarId** είναι το primary key.

Σχετικά με την ανάλυση των δυαδικών συσχετίσεων του ER σε σχέσεις, χρησιμοποιείται χαρτογραφημένη απεικόνιση και

αναλύεται παρακάτω το τι συμβαίνει για διαφορετικούς λόγους πληθικότητας:

- Δημιουργούμε καινούρια σχέση αν η συσχέτιση είναι πολλά προς πολλά(N:M) τοποθετώντας τα primary keys των οντοτήτων ως foreign keys. Ο συνδυασμός τους αποτελεί ένα primary key και επίσης τοποθετούνται όσα γνωρίσματα έχει η συσχέτιση. Στην προκειμένη περίπτωση αυτό συμβαίνει, σύμφωνα με το ER που φαίνεται παραπάνω στην συσχέτιση: **CONTAINS**
- Αν είναι 1:N ή N:1 τότε δεν δημιουργείται καινούρια σχέση και μετακινείται το primary key από την οντότητα που συνδέεται με 1 στην συσχέτιση προς την οντότητα που συνδέεται με N. Στην σχέση αυτή το foreign key δεν υπογραμμίζεται και δείχνει στο primary key της άλλης σχέσης που συνδέεται με την συσχέτιση. Επίσης με το primary key μετακινούνται προς την ίδια σχέση και όσα γνωρίσματα έχει η συσχέτιση. Στην προκειμένη περίπτωση αυτό συμβαίνει, σύμφωνα με το ER που φαίνεται παραπάνω, με τις 5 συσχετίσεις που ακολουθούν: **HAS_DESTINATION, DEPARTS_FROM, IS_SERVED, IS.**

Για την τριαδική συσχέτιση **HAS** δημιουργείται καινούρια σχέση έχοντας τα τρία κλειδιά των οντοτήτων και τα γνωρίσματα της, δηλαδή **PassengerId, ReservationId, FlightId** ως foreign keys στις αντίστοιχες οντότητες που συνδέει.

5. ΚΑΤΑΓΡΑΦΗ ΣΥΝΑΡΤΗΣΙΑΚΩΝ ΕΞΑΡΤΗΣΕΩΝ ΤΟΥ ΣΣ ΚΑΙ ΚΑΝΟΝΙΚΟΠΟΙΗΣΗ ΤΟΥ ΣΣ ΣΕ BCNF Ή 3NF

5A. ΣΥΝΑΡΤΗΣΙΑΚΕΣ ΕΞΑΡΤΗΣΕΙΣ ΤΟΥ ΣΣ

Η σχέση **PASSENGER** έχει ως πρωτεύον κλειδί το **PassengerId** το οποίο καθορίζει όλα τα υπόλοιπα γνωρίσματα (**PassengerId**→**FirstName**, **LastName**, **BirthDate**, **Sex**, **Email**, **PhoneNumber**). Δεν υπάρχει κάποια άλλη συναρτησιακή εξάρτηση μεταξύ των άλλων γνωρισμάτων.

Η σχέση **RESERVATION** έχει ως πρωτεύον κλειδί το **ReservationId** το οποίο καθορίζει όλα τα υπόλοιπα γνωρίσματα (**ReservationId**→**WifiOnBoard**, **PetSize**, **Price**) και δεν υπάρχουν επιπλέον συναρτησιακές εξαρτήσεις μέσα σε αυτή τη σχέση.

Η σχέση **CAR** έχει ως πρωτεύον κλειδί το **CarId** το οποίο καθορίζει όλα τα υπόλοιπα γνωρίσματα (**CarId**→**Model**, **Location**, **Company**, **Price**, **CarImage**) και δεν υπάρχουν επιπλέον συναρτησιακές εξαρτήσεις μέσα σε αυτή τη σχέση.

Η σχέση **FLIGHT** έχει ως πρωτεύον κλειδί το **FlightId** το οποίο καθορίζει όλα τα υπόλοιπα γνωρίσματα (**FlightId**→**FlightDate**, **Duration**, **ArrivalTime**, **DepartureTime**, **EconomyPrice**, **FlexPrice**, **BusinessPrice**, **ArrivalAirport**, **DepartureAirport**, **Airplane**) και υπάρχει άλλη μία συναρτησιακή εξάρτηση όπου ο συνδυασμός των **ArrivalTime**, **DepartureTime** καθορίζουν το **Duration**. Δεν υπάρχει κάποια άλλη συναρτησιακή εξάρτηση μεταξύ των άλλων γνωρισμάτων.

Η σχέση **CONTAINS** έχει ως πρωτεύον κλειδί τον συνδυασμό των γνωρισμάτων **CarId**, **ReservationId** όπου ο συνδυασμός τους καθορίζουν τα γνωρίσματα **PickUp**, **Return** και **RentingPrice**.

Η σχέση **HAS** έχει ως πρωτεύον κλειδί τον συνδυασμό των γνωρισμάτων **PassengerId**, **ReservationId**, **FlightId** και **Baggage** όπου ο συνδυασμός τους καθορίζουν τα γνωρίσματα **SeatNumber**, **NumOfBaggagePerCategory**, **ClassType** και **CheckIn**.

Η σχέση **AIRPORT** έχει ως πρωτεύον κλειδί το **Name** το οποίο καθορίζει το γνώρισμα **Country**(**Name** → **City**) και δεν υπάρχει κάποια άλλη συναρτησιακή εξάρτηση.

Η σχέση **AIRPLANE** έχει ως πρωτεύον κλειδί το **AirplaneId** το οποίο καθορίζει το γνώρισμα (**AirplaneId** → **Model**) και δεν υπάρχουν επιπλέον συναρτησιακές εξαρτήσεις μέσα σε αυτή τη σχέση.

Η σχέση **MODEL** έχει ως πρωτεύον κλειδί το **ModelId** το οποίο καθορίζει το γνώρισμα **Capacity**(**ModelId** → **Capacity**) και δεν υπάρχει κάποια άλλη συναρτησιακή εξάρτηση.

5B. ΒΗΜΑ-ΒΗΜΑ ΚΑΝΟΝΙΚΟΠΟΙΗΣΗ ΣΕ 3NF

➤ Κανονικοποίηση σε Πρώτη Κανονική Μορφή (1NF)

Στις σχέσεις του Σχεσιακού Σχήματος δεν υπάρχουν πλειότιμα και σύνθετα γνωρίσματα. Στο ER(ΔΟΣ) δεν υπάρχουν πλειότιμα γνωρίσματα οπότε όλες οι σχέσεις του σχεσιακού μοντέλου βρίσκονται στην Πρώτη Κανονική Μορφή (1NF).

➤ Κανονικοποίηση σε Δεύτερη Κανονική Μορφή (2NF)

Για να βρίσκεται μία σχέση στη Δεύτερη Κανονική Μορφή (2NF) πρέπει να βρίσκεται στην 1NF και κάθε μη πρωτεύον γνώρισμα του σχεσιακού σχήματος να είναι πλήρως συναρτησιακά εξαρτημένο από κάθε υποψήφιο κλειδί της σχέσης. Επομένως σε όλες τις σχέσεις του

σχεσιακού σχήματος τα μη πρωτεύοντα γνωρίσματα, αν υπάρχουν, εξαρτώνται πλήρως από το πρωτεύον κλειδί κάθε μίας σχέσης.

➤ Κανονικοποίηση σε Τρίτη Κανονική Μορφή (3NF)

Για να βρίσκεται μία σχέση στη Τρίτη Κανονική Μορφή (3NF) πρέπει να βρίσκεται στην 2NF και κανένα μη πρωτεύον γνώρισμα της σχέσης να μην είναι μεταβατικά εξαρτημένο από κάθε υποψήφιο κλειδί της σχέσης. Στις σχέσεις **PASSENGER**, **RESERVATION**, **CAR**, **CONTAINS**, **HAS**, **AIRPORT**, **AIRPLANE** και **MODEL** δεν υπάρχει καμία μεταβατική εξάρτηση, συνεπώς βρίσκονται όλες στην 3NF. Αναφορικά με την σχέση **FLIGHT** υπάρχει μεταβατική εξάρτηση που παραβιάζει την 3NF.

Για την σχέση **FLIGHT** υπάρχει η μεταβατική εξάρτηση όπου το **FlightId** ως πρωτεύον κλειδί καθορίζει τα **Duration**, **ArrivalTime** και **DepartureTime** και ο συνδυασμός των **ArrivalTime** και **DepartureTime** με την σειρά του καθορίζει το **Duration**. Έχοντας τον συνδυασμό **ArrivalTime** και **DepartureTime**, τα οποία είναι **times** με **time zone** μπορεί να βρεθεί το **Duration** μίας πτήσης. Επίσης δύο πλειάδες αν έχουν ίδιο συνδυασμό **ArrivalTime** και **DepartureTime** τότε το **Duration** θα είναι ίδιο και στις δύο πλειάδες (**ArrivalTime**, **DepartureTime**→**Duration**). Οπότε για να βρίσκεται η σχέση αυτή στην 3NF θα πρέπει να σπάσει σε δύο σχέσεις:

FLIGHT(FlightId, FlightDate, ArrivalTime,
DepartureTime, EconomyPrice, FlexPrice,
BusinessPrice, ArrivalAirport, DepartureAirport,
Airplane)

KAI

FLIGHT_DURATION(DepartureTime, ArrivalTime,
Duration)

Επομένως το σχεσιακό σχήμα που αφορά αυτές τις
σχέσεις θα αλλάξει σε:

FLIGHT_DURATION(DepartureTime, ArrivalTime,
Duration)

FLIGHT(FlightId, FlightDate, ArrivalTime,
DepartureTime, Price, ArrivalAirport,
DepartureAirport, Airplane)

AIRPORT(Name, City)

AIRPLANE(AirplaneId, Model)

Οι καινούριες σχέσεις που προκύπτουν **FLIGHT** και
FLIGHT_DURATION δεν παραβιάζουν τώρα την 3NF αφού
δεν υπάρχει κάποια μεταβατική εξάρτηση, οπότε μετά την
κανονικοποίηση βρίσκονται στην 3NF. Παρατηρείται ότι
έτσι όπως έγιναν οι σχέσεις κατά την κανονικοποίηση
τους σε 3NF το αριστερό μέλος κάθε συναρτησιακής
εξάρτησης είναι υπερκλειδί, οπότε οι σχέσεις του
σχεσιακού σχήματος βρίσκονται και στην BCNF μορφή.

6. ΥΛΟΠΟΙΗΣΗ ΤΟΥ ΣΣ ΣΤΗΝ POSTGRESQL ΚΑΙ ΕΙΣΑΓΩΓΗ ΕΓΓΡΑΦΩΝ ΣΤΟΥΣ ΠΙΝΑΚΕΣ

6A. ΥΛΟΠΟΙΗΣΗ ΤΟΥ ΣΣ ΣΤΗΝ POSTGRESQL

Οι πίνακες που θα χρειαστούν για αυτή την βάση δεδομένων δημιουργήθηκαν στην PostgreSQL. Παρακάτω παρατίθενται οι εντολές που έγιναν run για την δημιουργία των πινάκων με python script("creationOfDatabase.py") που συνδέθηκε στην PostgreSQL με βάση τις σχέσεις από το σχεσιακό μοντέλο. Επίσης κάθε φορά που τρέχει το script διαγράφονται οι πίνακες με τις εγγραφές τους και ξαναδημιουργούνται.

Τύποι δεδομένων που χρησιμοποιούνται παρακάτω είναι:

integer	ακέραιος
varchar(n)	μεταβλητού μήκους συμβολοσειρά μέγιστου μήκους n
date	ημερομηνία πχ. 10/4/1998
time with time zone	ώρα με ζώνη ώρας(UTC='Universal Time Coordinated') πχ. 10:45:00+01
numeric(precision, scale)	ένας ακριβής αριθμός με σταθερή ακρίβεια και κλίμακα. Το precision είναι ο μέγιστος συνολικός αριθμός δεκαδικών ψηφίων που πρέπει να αποθηκευτούν. Το scale είναι ο αριθμός των δεκαδικών ψηφίων που αποθηκεύονται στα δεξιά της υποδιαστολής.
timestamp	αποτελείται από τη συνένωση μιας ημερομηνίας και μιας ώρας πχ. 1999-01-08 04:05:06

bool	είναι μια τιμή που παίρνει true ή false.
serial4	είναι integer όπου είναι auto-increment και αυξάνεται κάθε φορά που εισάγεται μία νέα εγγραφή στον πίνακα ξεκινώντας από το 1.
bytea	είναι ένα αρχείο στην μορφή binary δηλαδή κωδικοποιημένο σε bytes, στην περίπτωση της εφαρμογής είναι οι εικόνες των αυτοκινήτων.

Περιορισμοί στήλης που εμφανίζονται:

primary key	δηλώνεται το πρωτεύον κλειδί
not null	δεν επιτρέπει σε κάποιο πεδίο να πάρει null τιμές
unique	οι διαφορετικές πλειάδες σε αυτό το πεδίο παίρνουν μοναδικές τιμές

Περιορισμός αναφορικής ακεραιότητας που εμφανίζεται:

foreign key	είναι το ξένο κλειδί που συνδέει δύο πίνακες/σχέσεις
--------------------	--

Πρώτα δημιουργούνται οι πίνακες στην PostgreSQL με τις παρακάτω εντολές χωρίς την ενσωμάτωση των foreign keys. Τα foreign keys τοποθετούνται με τις εντολές alter table μετά την δημιουργία των πινάκων. Αυτό γίνεται διότι μπορεί να μην έχουν δημιουργηθεί ακόμη κάποιοι πίνακες οπότε δεν θα είναι εφικτό να τεθεί foreign key μέσα στις create table. Οπότε πρέπει να δημιουργηθούν οι πίνακες χωρίς τα foreign keys και

στο τέλος να γίνει προσθήκη τους σε κάθε πίνακα ξεχωριστά όπως φαίνεται παρακάτω:

Για την σχέση **PASSENGER** ο πίνακας δημιουργήθηκε με βάση την παρακάτω εντολή:

```
#creation of table "passenger"
cur.execute("""DROP TABLE IF EXISTS passenger CASCADE""")
cur.execute("""
    CREATE TABLE passenger(
        passengerid SERIAL4 NOT NULL PRIMARY KEY,
        firstname VARCHAR (50) NOT NULL,
        lastname VARCHAR (50) NOT NULL,
        email VARCHAR (100) UNIQUE NOT NULL,
        birthdate DATE NOT NULL,
        sex VARCHAR(10) NOT NULL,
        phonenummer VARCHAR(20) UNIQUE NOT NULL
    );
""")
```

Για την σχέση **FLIGHT** ο πίνακας δημιουργήθηκε με βάση την παρακάτω εντολή:

```
#creation of table "flight"
cur.execute("""DROP TABLE IF EXISTS flight CASCADE""")
cur.execute("""
    CREATE TABLE flight(
        flightid VARCHAR(20) NOT NULL PRIMARY KEY,
        flightdate DATE NOT NULL,
        departuretime TIME WITH TIME ZONE NOT NULL,
        arrivaltime TIME WITH TIME ZONE NOT NULL,
        economyprice NUMERIC(6,2) NOT NULL,
        flexprice NUMERIC(6,2) NOT NULL,
        businessprice NUMERIC(6,2) NOT NULL,
        departureairport VARCHAR(50) NOT NULL,
        arrivalairport VARCHAR(50) NOT NULL,
        airplane VARCHAR(20) NOT NULL
    );
""")
```

Για την σχέση **RESERVATION** ο πίνακας δημιουργήθηκε με βάση την παρακάτω εντολή:

```
#creation of table "reservation"
cur.execute("""DROP TABLE IF EXISTS reservation CASCADE""")
cur.execute("""
    CREATE TABLE reservation(
        reservationid VARCHAR(20) NOT NULL PRIMARY KEY,
        petsize VARCHAR(20),
        wifionboard INTEGER NOT NULL,
        price NUMERIC(6,2) NOT NULL
    );
""")
```

Για την σχέση **CONTAINS** ο πίνακας δημιουργήθηκε με βάση την παρακάτω εντολή:

```
#creation of table "contains"
cur.execute("""DROP TABLE IF EXISTS contains CASCADE""")
cur.execute("""
    CREATE TABLE contains(
        carid INTEGER NOT NULL,
        reservationid VARCHAR(20) NOT NULL,
        pickup TIMESTAMP NOT NULL,
        return TIMESTAMP NOT NULL,
        rentingprice NUMERIC(6,2) NOT NULL,
        PRIMARY KEY(carid,reservationid)
    );
""")
```

Για την σχέση **HAS** ο πίνακας δημιουργήθηκε με βάση την παρακάτω εντολή:

```
#creation of table "has"
cur.execute("""DROP TABLE IF EXISTS has CASCADE""")
cur.execute("""
    CREATE TABLE has(
        passengerid INTEGER NOT NULL,
        flightid VARCHAR(20) NOT NULL,
        reservationid VARCHAR(20) NOT NULL,
        seatnumber VARCHAR(10) NOT NULL,
        classtype VARCHAR(20) NOT NULL,
        baggage VARCHAR(20) NOT NULL,
        numofbaggagepercategory INTEGER NOT NULL,
        checkin BOOL NOT NULL,
        PRIMARY KEY(passengerid,flightid,reservationid,baggage)
    );
""")
```

Για την σχέση **CAR** ο πίνακας δημιουργήθηκε με βάση την παρακάτω εντολή:

```
#creation of table "car"
cur.execute("""DROP TABLE IF EXISTS car CASCADE""")
cur.execute("""
    CREATE TABLE car(
        carid SERIAL4 NOT NULL PRIMARY KEY,
        model VARCHAR(20) NOT NULL,
        location VARCHAR(50) NOT NULL,
        company VARCHAR(20) NOT NULL,
        price NUMERIC(6,2) NOT NULL,
        carimage BYTEA NOT NULL
    );
""")
```

Για την σχέση **AIRPORT** ο πίνακας δημιουργήθηκε με βάση την παρακάτω εντολή:

```
#creation of table "airport"
cur.execute("""DROP TABLE IF EXISTS airport CASCADE""")
cur.execute("""
    CREATE TABLE airport(
        name VARCHAR(50) NOT NULL,
        city VARCHAR(50) NOT NULL,
        PRIMARY KEY(name)
    );
""")
```

Για την σχέση **AIRPLANE** ο πίνακας δημιουργήθηκε με βάση την παρακάτω εντολή:

```
#creation of table "airplane"
cur.execute("""DROP TABLE IF EXISTS airplane CASCADE""")
cur.execute("""
    CREATE TABLE airplane(
        airplaneid VARCHAR(20) NOT NULL,
        model VARCHAR(20) NOT NULL,
        PRIMARY KEY(airplaneid)
    );
""")
```

Για την σχέση **MODEL** ο πίνακας δημιουργήθηκε με βάση την παρακάτω εντολή:

```
#creation of table "model"
cur.execute("""DROP TABLE IF EXISTS model CASCADE""")
cur.execute("""
    CREATE TABLE model(
        modelid VARCHAR(20) NOT NULL,
        capacity INTEGER NOT NULL,
        PRIMARY KEY(modelid)
    );
""")
```

Για την σχέση **FLIGHT_DURATION** ο πίνακας δημιουργήθηκε με βάση την παρακάτω εντολή:

```
#creation of table "flight_duration"
cur.execute("""DROP TABLE IF EXISTS flight_duration CASCADE""")
cur.execute("""
    CREATE TABLE flight_duration(
        departuretime TIME WITH TIME ZONE NOT NULL,
        arrivaltime TIME WITH TIME ZONE NOT NULL,
        duration VARCHAR(20) NOT NULL,
        PRIMARY KEY(departuretime,arrivaltime)
    );
""")
```

Κάποιες επιπλέον προσθήκες στο τέλος κάθε ορισμού ενός foreign key στις alter table σημαίνουν:

on update cascade	υποδηλώνει ότι πρέπει να αλλάξει η τιμή του ξένου κλειδιού στην τροποποιημένη τιμή του πρωτεύοντος κλειδιού, σε όλες τις πλειάδες που αναφέρονται στην τροποποιημένη πλειάδα.
on delete cascade	υποδηλώνει ότι πρέπει να διαγραφούν όλες οι πλειάδες που αναφέρονται στη διαγεγραμμένη πλειάδα.

Η ενσωμάτωση των foreign keys στους πίνακες γίνεται παρακάτω:

```
#creation of foreign keys
#@arrivalairport(flight) -> name(airport)
cur.execute("""
    ALTER TABLE flight
    ADD FOREIGN KEY(arrivalairport)
    REFERENCES airport(name)
    ON DELETE CASCADE ON UPDATE CASCADE;
""")

#departureairport(flight) -> name(airport)
cur.execute("""
    ALTER TABLE flight
    ADD FOREIGN KEY(departureairport)
    REFERENCES airport(name)
    ON DELETE CASCADE ON UPDATE CASCADE;
""")

#arrivaltime,departuretime(flight) -> arrivaltime,departuretime(flight_duration)
cur.execute("""
    ALTER TABLE flight
    ADD FOREIGN KEY(arrivaltime,departuretime)
    REFERENCES flight_duration(arrivaltime,departuretime)
    ON DELETE CASCADE ON UPDATE CASCADE;
""")

#flightid(has) -> flightid(flight)
cur.execute("""
    ALTER TABLE has
    ADD FOREIGN KEY(flightid)
    REFERENCES flight(flightid)
    ON DELETE CASCADE ON UPDATE CASCADE;
""")

#model(airplane) -> modelid(model)
cur.execute("""
    ALTER TABLE airplane
    ADD FOREIGN KEY(model)
    REFERENCES model(modelid)
    ON DELETE CASCADE ON UPDATE CASCADE;
""")

#carid(contains) -> carid(car)
cur.execute("""
    ALTER TABLE contains
    ADD FOREIGN KEY(carid)
    REFERENCES car(carid)
    ON DELETE CASCADE ON UPDATE CASCADE;
""")

#airplane(flight) -> airplaneid(airplane)
cur.execute("""
    ALTER TABLE flight
    ADD FOREIGN KEY(airplane)
    REFERENCES airplane(airplaneid)
    ON DELETE CASCADE ON UPDATE CASCADE;
""")

#passengerid(has) -> passengerid(passenger)
cur.execute("""
    ALTER TABLE has
    ADD FOREIGN KEY(passengerid)
    REFERENCES passenger(passengerid)
    ON DELETE CASCADE ON UPDATE CASCADE;
""")

#reservationid(has) -> reservationid(reservation)
cur.execute("""
    ALTER TABLE has
    ADD FOREIGN KEY(reservationid)
    REFERENCES reservation(reservationid)
    ON DELETE CASCADE ON UPDATE CASCADE;
""")

#reservationid(contains) -> reservationid(reservation)
cur.execute("""
    ALTER TABLE contains
    ADD FOREIGN KEY(reservationid)
    REFERENCES reservation(reservationid)
    ON DELETE CASCADE ON UPDATE CASCADE;
""")
```

6B. ΕΙΣΑΓΩΓΗ ΕΓΓΡΑΦΩΝ ΣΤΟΥΣ ΠΙΝΑΚΕΣ

Οι εισαγωγές στους πίνακες(εγγραφές) έγιναν με insert εντολές από το python script("creationOfDatabase.py") που είναι συνδεδεμένο στην PostgreSQL.

Για τον πίνακα **passenger** δημιουργήθηκαν τυχαία δεδομένα:

```
female_names = [
    'Maria', 'Dimitra', 'Sofia', 'Georgia', 'Ioanna', 'Ellisabet', 'Helen', 'Rachel',
    'Amanda', 'Anna', 'Alexia', 'Angelina', 'Kate', 'Penelope', 'Sandra', 'Scarlett'
]
male_names = [
    'Vasilis', 'George', 'John', 'Michael', 'Victor', 'Panagiotis', 'Aggelos', 'Stavros',
    'Russel', 'Brad', 'Tom', 'Dimitris', 'Konstantinos', 'Nick', 'Iasonas'
]
female_lastnames = [
    'Amalidou', 'Nomikou', 'Papadopoulou', 'Papadimitriou', 'Karavasili', 'Kavoura',
    'Maniati', 'Weber', 'Beckett', 'Jones', 'Perry', 'Aniston'
]
male_lastnames = [
    'House', 'Pitt', 'Johanson', 'Wilson', 'Taylor', 'Antoniou', 'Papadopoulos', 'Ioannou',
    'Stergiou', 'Georgiou', 'Venetis', 'Grigoriou', 'Nikolopoulos', 'Papazoglou', 'Diamantidis',
    'Spanoulis'
]
greek_phone = "+3069"
num = 1
#insert records into passenger table
for i in range(1,21):
    #random combinations from above lists and makes 10 female persons and 10 male persons
    if i%2 == 1:
        firstname = female_names[r.randint(0,15)]
        lastname = female_lastnames[r.randint(0,11)]
        sex = 'Female'
    else :
        firstname = male_names[r.randint(0,14)]
        lastname = male_lastnames[r.randint(0,15)]
        sex = 'Male'

    #random birthdate from 18 to 65 ages in format for example 01/02/1987
    birthdate = fake.date_of_birth(minimum_age=18, maximum_age=65).strftime('%d/%m/%Y')

    # Create a simple email based on the name
    if int(birthdate.split('/')[2])>2000 :
        #puts the whole birthdate year
        email = f'{firstname.lower()}.{lastname.lower()}{"birthdate.split('/')[2]"}{num}@gmail.com"
    else :
        #puts only the two last digits from birthdate year, like from 1985, the digits 85
        email = f'{firstname.lower()}.{lastname.lower()}{"birthdate.split('/')[2][2:]"}{num}@gmail.com"

    num = num + 1

    #phones that has the type of greek to be more simple
    end_phone = r.randint(12345678,98765432)
    phone = greek_phone + str(end_phone)

    #insert each record into the passenger table with random values in the fields
    cur.execute("""INSERT INTO passenger(firstname,lastname,email,birthdate,sex,phonenumber)
    VALUES(%s,%s,%s,%s,%s,%s)""",
    [firstname,lastname,email,datetime.strptime(birthdate, '%d/%m/%Y'),sex,phone])
}
```

Για τον πίνακα **model**:

```
#insert records into model table
cur.execute("""INSERT INTO model
VALUES
('Airbus A320neo',180),
('Airbus A321-200',180),
('Airbus A320-200',180),
('Boeing 737-800',180),
('Airbus A220-100',120);
""")
```

Για τον πίνακα **airplane**:

```
#insert records into airplane table
cur.execute("""INSERT INTO airplane
VALUES
('FN1','Airbus A320neo'),
('FN2','Airbus A321-200'),
('FN3','Airbus A320-200'),
('FN4','Boeing 737-800'),
('FN5','Airbus A320-200'),
('FN6','Airbus A320neo'),
('FN7','Boeing 737-800'),
('FN8','Airbus A320neo'),
('FN9','Airbus A321-200'),
('FN10','Airbus A320neo'),
('FN11','Airbus A220-100'),
('FN12','Airbus A220-100');
""")
```

Για τον πίνακα **airport**:

```
#insert records into airport table
cur.execute("""INSERT INTO airport
VALUES
('SKG','Thessaloniki'),
('MAD','Madrid'),
('BER','Berlin'),
('AMS','Amsterdam'),
('CDG','Paris'),
('FCO','Rome'),
('ATH','Athens'),
('MJT','Mytilene'),
('VIE','Vienna'),
('BUD','Budapest'),
('HEL','Helsinki'),
('ZRH','Zurich'),
('LON','London'),
('PRG','Prague'),
('LIS','Lisbon');
""")
```

Για τον πίνακα **car**, κωδικοποιούμε τις εικόνες που βρίσκονται στον φάκελο **images** σε bytes και τις περνάμε στην βάση μαζί με τα υπόλοιπα στοιχεία με τα **insert**:

```
images = []
with open('./images/ToyotaAygo.png', 'rb') as image_file:
    # Read the content of the image as bytes
    image_data = image_file.read()
    images.append(psycopg2.Binary(image_data))
with open('./images/Fiat500Bev.png', 'rb') as image_file:
    image_data = image_file.read()
    images.append(psycopg2.Binary(image_data))
with open('./images/OpelCorsa.png', 'rb') as image_file:
    image_data = image_file.read()
    images.append(psycopg2.Binary(image_data))
with open('./images/ToyotaCorolla.png', 'rb') as image_file:
    image_data = image_file.read()
    images.append(psycopg2.Binary(image_data))
with open('./images/SeatIbiza.png', 'rb') as image_file:
    image_data = image_file.read()
    images.append(psycopg2.Binary(image_data))
with open('./images/Hyundaii10.png', 'rb') as image_file:
    image_data = image_file.read()
    images.append(psycopg2.Binary(image_data))
with open('./images/MercedesE220.png', 'rb') as image_file:
    image_data = image_file.read()
    images.append(psycopg2.Binary(image_data))
with open('./images/BMWSeries1.png', 'rb') as image_file:
    image_data = image_file.read()
    images.append(psycopg2.Binary(image_data))
with open('./images/NissanQashqai.png', 'rb') as image_file:
    image_data = image_file.read()
    images.append(psycopg2.Binary(image_data))
with open('./images/TeslaModelY.png', 'rb') as image_file:
    image_data = image_file.read()
    images.append(psycopg2.Binary(image_data))
```

```
#insert records into car table
cur.execute("""INSERT INTO car(model, location, company, price, carimage)
VALUES
('Toyota Aygo', 'ATH', 'Avis', '55.00,%s'),
('Fiat 500 Bev', 'SKG', 'Avis', '75.00,%s'),
('Opel Corsa', 'CDG', 'Thrifty', '80.00,%s'),
('Toyota Corolla', 'FCO', 'Firefly', '100.00,%s'),
('Seat Ibiza', 'LIS', 'Firefly', '75.00,%s'),
('Hyundai i10', 'PRG', 'Hertz', '65.00,%s'),
('Mercedes E220', 'AMS', 'Hertz', '82.00,%s'),
('BMW Series 1', 'BER', 'Hertz', '98.00,%s'),
('Nissan Qashqai', 'MAD', 'Hertz', '110.00,%s'),
('Toyota Yaris', 'MJT', 'Avis', '69.00,%s'),
('Tesla Model Y', 'VIE', 'Thrifty', '134.00,%s'),
('Fiat 500 Bev', 'BUD', 'Hertz', '83.00,%s'),
('Opel Corsa', 'HEL', 'Firefly', '85.00,%s'),
('Hyundai i10', 'ZRH', 'Hertz', '65.00,%s'),
('Toyota Corolla', 'LON', 'Thrifty', '90.00,%s'),
('Seat Ibiza', 'PRG', 'Hertz', '70.00,%s'),
('Toyota Corolla', 'HEL', 'Hertz', '106.00,%s'),
('Mercedes E220', 'SKG', 'Avis', '91.00,%s'),
('Seat Ibiza', 'CDG', 'Thrifty', '77.00,%s'),
('Opel Corsa', 'FCO', 'Hertz', '80.00,%s'),
('Hyundai i10', 'LIS', 'Thrifty', '73.00,%s'),
('Hyundai i10', 'PRG', 'Firefly', '80.00,%s'),
('BMW Series 1', 'AMS', 'Thrifty', '84.00,%s'),
('BMW Series 1', 'BER', 'Firefly', '100.00,%s'),
('Toyota Yaris', 'MAD', 'Hertz', '70.00,%s'),
('Nissan Qashqai', 'MJT', 'Avis', '105.00,%s'),
('Fiat 500 Bev', 'VIE', 'Firefly', '70.00,%s'),
('Fiat 500 Bev', 'BUD', 'Thrifty', '78.00,%s'),
('Tesla Model Y', 'HEL', 'Hertz', '127.00,%s'),
('Hyundai i10', 'ZRH', 'Avis', '75.00,%s'),
('Seat Ibiza', 'LON', 'Avis', '72.00,%s'),
('Toyota Corolla', 'PRG', 'Avis', '88.00,%s'),
('Hyundai i10', 'HEL', 'Thrifty', '73.00,%s'),
('Tesla Model Y', 'ATH', 'Firefly', '120.00,%s'),
('Mercedes E220', 'SKG', 'Thrifty', '94.00,%s'),
('Toyota Aygo', 'SKG', 'Firefly', '60.00,%s'),
('Toyota Aygo', 'HEL', 'Thrifty', '63.00,%s'),
('Nissan Qashqai', 'HEL', 'Firefly', '110.00,%s'),
('Toyota Yaris', 'HEL', 'Avis', '67.00,%s');

"""(images[0],images[1],images[2],images[3],images[4],images[5],
      images[6],images[7],images[8],images[9],images[10],images[1],
      images[2],images[5],images[3],images[4],images[3],images[6],
      images[4],images[2],images[5],images[5],images[7],images[7],
      images[9],images[8],images[1],images[1],images[10],images[5],
      images[4],images[3],images[5],images[10],images[6],images[0],
      images[0],images[8],images[9]))
```

Για τον πίνακα flight_duration:

```
#insert records into flight_duration table
cur.execute("""
    INSERT INTO flight_duration(departuretime, arrivaltime, duration)
VALUES
    ('10:00:00+02:00', '10:55:00+02:00', '55min'),
    ('12:05:00+02:00', '14:30:00+00:00', '4h 25min'),
    ('18:50:00+02:00', '19:45:00+02:00', '55min'),
    ('12:00:00+00:00', '18:00:00+02:00', '4h'),
    ('13:15:00+01:00', '17:45:00+02:00', '3h 30min'),
    ('09:15:00+02:00', '11:45:00+01:00', '3h 30min'),
    ('12:15:00+02:00', '14:45:00+01:00', '3h 30min'),
    ('12:55:00+01:00', '14:00:00+00:00', '2h 5min'),
    ('18:00:00+00:00', '21:05:00+01:00', '2h 5min'),
    ('14:50:00+01:00', '16:20:00+01:00', '1h 30min'),
    ('09:40:00+01:00', '11:10:00+01:00', '1h 30min'),
    ('09:00:00+01:00', '10:45:00+01:00', '1h 45min'),
    ('21:10:00+01:00', '22:55:00+01:00', '1h 45min'),
    ('08:30:00+01:00', '10:05:00+01:00', '1h 35min'),
    ('16:05:00+01:00', '17:40:00+01:00', '1h 35min'),
    ('18:35:00+02:00', '21:15:00+01:00', '3h 40min'),
    ('13:50:00+01:00', '18:20:00+02:00', '3h 30min'),
    ('08:35:00+02:00', '12:15:00+01:00', '3h 40min'),
    ('07:20:00+01:00', '11:40:00+02:00', '3h 20min'),
    ('07:30:00+02:00', '08:20:00+02:00', '50min'),
    ('10:30:00+02:00', '11:20:00+02:00', '50min'),
    ('14:50:00+02:00', '16:15:00+01:00', '2h 25min'),
    ('17:00:00+01:00', '20:25:00+02:00', '2h 25min'),
    ('10:15:00+01:00', '11:45:00+01:00', '1h 30min'),
    ('17:15:00+01:00', '18:45:00+01:00', '1h 30min'),
    ('22:00:00+01:00', '00:10:00+00:00', '3h 10min'),
    ('08:30:00+02:00', '10:20:00+01:00', '2h 50min'),
    ('16:30:00+01:00', '20:20:00+02:00', '2h 50min'),
    ('07:10:00+01:00', '08:55:00+00:00', '2h 45min'),
    ('21:50:00+00:00', '01:25:00+01:00', '2h 35min'),
    ('11:40:00+02:00', '13:25:00+01:00', '2h 45min'),
    ('17:30:00+01:00', '19:25:00+02:00', '55min'),
    ('09:30:00+02:00', '10:55:00+01:00', '2h 25min'),
    ('12:00:00+01:00', '15:05:00+02:00', '2h 5min'),
    ('16:50:00+02:00', '17:55:00+01:00', '2h 5min'),
    ('19:00:00+01:00', '22:45:00+02:00', '2h 45min'),
    ('18:00:00+02:00', '11:05:00+01:00', '2h 5min'),
    ('23:10:00+02:00', '00:15:00+01:00', '2h 5min'),
    ('07:00:00+01:00', '09:55:00+02:00', '1h 55min'),
    ('18:30:00+02:00', '19:20:00+02:00', '50min'),
    ('16:20:00+02:00', '17:10:00+02:00', '50min'),
    ('12:00:00+02:00', '12:50:00+02:00', '50min'),
    ('12:10:00+02:00', '13:15:00+01:00', '2h 5min'),
    ('18:00:00+01:00', '21:05:00+02:00', '2h 5min'),
    ('14:05:00+01:00', '18:10:00+02:00', '3h 5min'),
    ('10:40:00+02:00', '13:10:00+01:00', '3h 30min'),
    ('18:25:00+02:00', '20:05:00+01:00', '2h 40min'),
    ('20:50:00+01:00', '00:15:00+02:00', '2h 25min'),
    ('09:25:00+02:00', '11:05:00+01:00', '2h 40min'),
    ('11:50:00+01:00', '15:15:00+02:00', '2h 25min'),
    ('17:50:00+02:00', '18:45:00+02:00', '55min'),
    ('16:00:00+02:00', '16:50:00+01:00', '1h 50min'),
    ('20:25:00+01:00', '22:30:00+01:00', '2h 5min'),
    ('07:10:00+01:00', '09:05:00+01:00', '1h 55min'),
    ('11:40:00+01:00', '14:25:00+02:00', '1h 45min'),
    ('16:00:00+01:00', '16:25:00+00:00', '1h 25min'),
    ('16:45:00+00:00', '19:15:00+01:00', '1h 30min'),
    ('20:35:00+01:00', '01:10:00+02:00', '3h 35min'),
    ('21:25:00+02:00', '22:55:00+00:00', '3h 30min'),
    ('08:00:00+00:00', '13:10:00+02:00', '3h 10min'),
    ('08:00:00+02:00', '08:45:00+01:00', '1h 45min'),
    ('11:00:00+01:00', '14:25:00+02:00', '2h 25min'),
    ('17:05:00+02:00', '18:30:00+01:00', '2h 25min'),
    ('21:50:00+01:00', '00:25:00+02:00', '1h 35min'),
    ('10:50:00+02:00', '11:45:00+02:00', '55min'),
    ('09:00:00+02:00', '09:50:00+02:00', '50min'),
    ('18:50:00+02:00', '19:40:00+02:00', '50min'),
    ('17:00:00+02:00', '17:55:00+02:00', '55min');
    """)
```

Για τον πίνακα flight:

```

#insert records into flight table
cur.execute("""
    INSERT INTO flight(flightid, flightdate, departuretime, arrivalthime, economyprice, flexprice, businessprice, departureairport, arrivalairport, airplane)
VALUES
('SK123', '20/01/2024', '10:00:00+02:00', '10:55:00+02:00', 100.00, 115.00, 200.00, 'SKG', 'ATH', 'FN1'),
('AT567', '18/02/2024', '12:05:00+02:00', '14:30:00+00:00', 95.00, 110.00, 190.00, 'ATH', 'LIS', 'FN4'),
('AT124', '22/01/2024', '18:50:00+02:00', '19:45:00+02:00', 60.00, 75.00, 120.00, 'ATH', 'SKG', 'FN1'),
('MA456', '22/01/2024', '13:15:00+01:00', '17:45:00+02:00', 75.00, 90.00, 150.00, 'MAD', 'ATH', 'FN2'),
('AT457', '26/01/2024', '09:15:00+02:00', '11:45:00+01:00', 75.00, 90.00, 150.00, 'ATH', 'MAD', 'FN2'),
('BE789', '25/01/2024', '12:55:00+01:00', '14:00:00+00:00', 120.00, 135.00, 240.00, 'BER', 'LON', 'FN3'),
('LO790', '29/01/2024', '18:00:00+00:00', '21:05:00+01:00', 125.00, 140.00, 250.00, 'LON', 'BER', 'FN3'),
('AM234', '28/01/2024', '14:50:00+01:00', '16:20:00+01:00', 50.00, 65.00, 100.00, 'AMS', 'PRG', 'FN4'),
('PR235', '01/02/2024', '09:40:00+01:00', '11:10:00+01:00', 70.00, 85.00, 140.00, 'PRG', 'AMS', 'FN4'),
('CD567', '30/01/2024', '09:00:00+01:00', '10:45:00+01:00', 100.00, 115.00, 200.00, 'CDG', 'BER', 'FN5'),
('BE568', '06/02/2024', '21:10:00+01:00', '22:55:00+01:00', 85.00, 100.00, 170.00, 'BER', 'CDG', 'FN5'),
('FC890', '30/01/2024', '08:30:00+01:00', '10:05:00+01:00', 90.00, 105.00, 180.00, 'FCO', 'ZRH', 'FN6'),
('ZR891', '02/02/2024', '16:05:00+01:00', '17:40:00+01:00', 110.00, 125.00, 220.00, 'ZRH', 'FCO', 'FN6'),
('AT123', '02/02/2024', '18:35:00+02:00', '21:15:00+01:00', 65.00, 80.00, 130.00, 'ATH', 'AMS', 'FN1'),
('AM124', '07/02/2024', '13:50:00+01:00', '18:20:00+02:00', 85.00, 100.00, 170.00, 'AMS', 'ATH', 'FN7'),
('AT159', '02/02/2024', '08:35:00+02:00', '12:15:00+01:00', 105.00, 120.00, 210.00, 'ATH', 'AMS', 'FN5'),
('AM192', '05/02/2024', '07:20:00+02:00', '11:40:00+01:00', 90.00, 105.00, 180.00, 'AMS', 'ATH', 'FN4'),
('SK459', '03/02/2024', '07:30:00+02:00', '08:20:00+02:00', 95.00, 110.00, 190.00, 'SKG', 'MJT', 'FN3'),
('MJ458', '08/02/2024', '10:30:00+02:00', '11:20:00+02:00', 60.00, 75.00, 120.00, 'MJT', 'SKG', 'FN3'),
('AT789', '02/02/2024', '14:50:00+02:00', '16:15:00+01:00', 100.00, 115.00, 200.00, 'ATH', 'VIE', 'FN9'),
('VI800', '03/02/2024', '17:00:00+01:00', '20:25:00+02:00', 130.00, 145.00, 260.00, 'VIE', 'ATH', 'FN9'),
('BU234', '04/02/2024', '10:15:00+01:00', '11:45:00+01:00', 70.00, 85.00, 140.00, 'BUD', 'ZRH', 'FN10'),
('ZR235', '06/02/2024', '17:15:00+01:00', '18:45:00+01:00', 55.00, 70.00, 110.00, 'ZRH', 'BUD', 'FN10'),
('AM329', '02/02/2024', '22:00:00+00:00', '00:10:00+00:00', 75.00, 90.00, 150.00, 'AMS', 'LIS', 'FN2'),
('AT598', '03/02/2024', '08:30:00+02:00', '10:20:00+01:00', 95.00, 110.00, 190.00, 'ATH', 'ZRH', 'FN1'),
('ZR567', '05/02/2024', '16:30:00+01:00', '20:20:00+02:00', 120.00, 135.00, 240.00, 'ZRH', 'ATH', 'FN1'),
('FC899', '08/02/2024', '07:10:00+01:00', '08:55:00+00:00', 120.00, 135.00, 240.00, 'FCO', 'LON', 'FN6'),
('LO899', '10/02/2024', '21:50:00+00:00', '01:25:00+01:00', 150.00, 165.00, 300.00, 'LON', 'FCO', 'FN9'),
('AT122', '09/02/2024', '11:40:00+02:00', '13:25:00+01:00', 65.00, 80.00, 130.00, 'ATH', 'BER', 'FN1'),
('BE113', '09/02/2024', '17:30:00+01:00', '19:25:00+02:00', 95.00, 110.00, 190.00, 'BER', 'HEL', 'FN4'),
('HE113', '15/02/2024', '09:30:00+02:00', '10:55:00+01:00', 95.00, 110.00, 190.00, 'HEL', 'BUD', 'FN2'),
('BU213', '15/02/2024', '12:00:00+01:00', '15:05:00+02:00', 55.00, 70.00, 110.00, 'BUD', 'ATH', 'FN8'),
('HE176', '15/02/2024', '16:50:00+02:00', '17:55:00+01:00', 165.00, 180.00, 330.00, 'HEL', 'BER', 'FN1'),
('BE290', '15/02/2024', '19:00:00+01:00', '22:45:00+02:00', 100.00, 115.00, 200.00, 'BER', 'ATH', 'FN2'),
('AT223', '05/02/2024', '10:00:00+02:00', '11:05:00+01:00', 95.00, 110.00, 190.00, 'ATH', 'BUD', 'FN8'),
('AT801', '07/02/2024', '23:10:00+02:00', '00:15:00+01:00', 120.00, 135.00, 240.00, 'ATH', 'FCO', 'FN3'),
('FC893', '11/02/2024', '07:00:00+01:00', '09:55:00+02:00', 50.00, 65.00, 100.00, 'FCO', 'ATH', 'FN7'),
('AT102', '10/02/2024', '10:30:00+02:00', '11:20:00+02:00', 95.00, 110.00, 190.00, 'ATH', 'MJT', 'FN5'),
('MJ500', '15/02/2024', '18:30:00+02:00', '19:20:00+02:00', 75.00, 90.00, 150.00, 'MJT', 'ATH', 'FN5'),
('AT103', '10/02/2024', '16:20:00+02:00', '17:10:00+02:00', 60.00, 75.00, 120.00, 'ATH', 'MJT', 'FN3'),
('MJ501', '15/02/2024', '12:00:00+02:00', '12:50:00+02:00', 80.00, 95.00, 160.00, 'MJT', 'ATH', 'FN3'),
('AT802', '07/02/2024', '12:10:00+02:00', '13:15:00+01:00', 100.00, 115.00, 200.00, 'ATH', 'FCO', 'FN4'),
('FC894', '11/02/2024', '18:00:00+01:00', '21:05:00+02:00', 70.00, 85.00, 140.00, 'FCO', 'ATH', 'FN4'),
('CD543', '10/02/2024', '14:05:00+01:00', '18:10:00+02:00', 90.00, 105.00, 180.00, 'CDG', 'ATH', 'FN6'),
('AT521', '12/02/2024', '10:40:00+02:00', '13:10:00+01:00', 75.00, 90.00, 150.00, 'ATH', 'CDG', 'FN6'),
('AT220', '28/01/2024', '18:25:00+02:00', '20:05:00+01:00', 60.00, 75.00, 120.00, 'ATH', 'PRG', 'FN1'),
('PR773', '01/02/2024', '20:50:00+01:00', '00:15:00+02:00', 65.00, 80.00, 130.00, 'PRG', 'ATH', 'FN2'),
('AT222', '28/01/2024', '09:25:00+02:00', '11:05:00+01:00', 80.00, 95.00, 160.00, 'ATH', 'PRG', 'FN8'),
('PR775', '03/02/2024', '11:50:00+01:00', '15:15:00+02:00', 85.00, 100.00, 160.00, 'PRG', 'ATH', 'FN9'),
('SK150', '13/02/2024', '10:00:00+02:00', '10:55:00+02:00', 80.00, 95.00, 160.00, 'SKG', 'ATH', 'FN1'),
('AT151', '09/02/2024', '17:50:00+02:00', '18:45:00+02:00', 50.00, 65.00, 100.00, 'ATH', 'SKG', 'FN9'),
('SK527', '14/02/2024', '16:00:00+02:00', '16:50:00+01:00', 150.00, 165.00, 300.00, 'SKG', 'VIE', 'FN6'),
('VI1549', '14/02/2024', '20:25:00+01:00', '22:30:00+01:00', 135.00, 150.00, 270.00, 'VIE', 'CDG', 'FN7'),
('CD501', '19/02/2024', '07:10:00+01:00', '09:05:00+01:00', 70.00, 85.00, 140.00, 'CDG', 'VIE', 'FN2'),
('VI1502', '19/02/2024', '11:40:00+01:00', '14:25:00+02:00', 105.00, 120.00, 210.00, 'VIE', 'SKG', 'FN9'),
('AT466', '18/02/2024', '12:15:00+02:00', '14:45:00+01:00', 70.00, 85.00, 140.00, 'ATH', 'MAD', 'FN1'),
('MA429', '18/02/2024', '16:00:00+01:00', '16:25:00+00:00', 100.00, 115.00, 200.00, 'MAD', 'LIS', 'FN2'),
('LI122', '24/02/2024', '16:45:00+00:00', '19:15:00+01:00', 100.00, 115.00, 200.00, 'LIS', 'MAD', 'FN8'),
('MA448', '24/02/2024', '20:35:00+01:00', '01:10:00+02:00', 100.00, 115.00, 200.00, 'MAD', 'ATH', 'FN3'),
('SK167', '23/02/2024', '21:25:00+02:00', '22:55:00+00:00', 130.00, 145.00, 260.00, 'SKG', 'LON', 'FN5'),
('LO117', '28/02/2024', '08:00:00+00:00', '13:10:00+02:00', 110.00, 125.00, 220.00, 'LON', 'SKG', 'FN5'),
('SK522', '17/02/2024', '08:00:00+02:00', '08:45:00+01:00', 70.00, 85.00, 140.00, 'SKG', 'BUD', 'FN10'),
('BU502', '17/02/2024', '11:00:00+01:00', '14:25:00+02:00', 80.00, 95.00, 160.00, 'BUD', 'HEL', 'FN6'),
('HE112', '24/02/2024', '17:05:00+02:00', '18:30:00+01:00', 180.00, 195.00, 360.00, 'HEL', 'BUD', 'FN6'),
('BU812', '24/02/2024', '21:50:00+01:00', '00:25:00+02:00', 60.00, 75.00, 120.00, 'BUD', 'SKG', 'FN9'),
('AT154', '09/02/2024', '10:50:00+02:00', '11:45:00+02:00', 60.00, 75.00, 120.00, 'ATH', 'SKG', 'FN1'),
('AT158', '09/02/2024', '09:00:00+02:00', '09:50:00+02:00', 70.00, 85.00, 140.00, 'ATH', 'MJT', 'FN12'),
('MJ155', '15/02/2024', '18:50:00+02:00', '19:40:00+02:00', 55.00, 70.00, 110.00, 'MJT', 'ATH', 'FN12'),
('SK153', '15/02/2024', '17:00:00+02:00', '17:55:00+02:00', 75.00, 90.00, 150.00, 'SKG', 'ATH', 'FN11'),
('LI603', '24/02/2024', '12:00:00+00:00', '18:00:00+02:00', 120.00, 135.00, 240.00, 'LIS', 'ATH', 'FN1');
""")
"""

```

Για τον πίνακα reservation:

```
#insert records into reservation table
cur.execute("""
    INSERT INTO reservation(reservationid, petsize, wifionboard, price)
VALUES
    ('AHDSSD', NULL, 0, 300.00),
    ('DFAKEM', NULL, 0, 195.00),
    ('HAOSMC', NULL, 0, 85.00),
    ('POWTWU', NULL, 0, 1020.00),
    ('UIWERT', NULL, 0, 135.00),
    ('BCVZSD', NULL, 0, 135.00),
    ('LAKSJF', 'Large', 0, 525.00),
    ('JQUERF', 'Small', 0, 255.00),
    ('AKJDFN', NULL, 0, 95.00),
    ('SAKSDM', NULL, 0, 100.00),
    ('QEIEUN', NULL, 0, 100.00),
    ('SAHIBU', NULL, 0, 100.00),
    ('ADDBSA', NULL, 0, 100.00),
    ('IAKDKA', NULL, 0, 70.00),
    ('AKSDJS', NULL, 0, 70.00);
""")
""")
```

Για τον πίνακα has:

```
#insert records into has table
cur.execute("""
    INSERT INTO has(passengerid, flightid, reservationid, seatnumber, classtype, baggage, numofbaggagepercategory, checkin)
VALUES
    (1, 'SK527', 'AHDSSD', '1A', 'BUSINESS CLASS', 'baggage23kg', 1, false),
    (1, 'SK527', 'AHDSSD', '1A', 'BUSINESS CLASS', 'baggage32kg', 0, false),
    (2, 'AM192', 'DFAKEM', '1B', 'BUSINESS CLASS', 'baggage23kg', 1, false),
    (2, 'AM192', 'DFAKEM', '1B', 'BUSINESS CLASS', 'baggage32kg', 1, false),
    (3, 'FC894', 'HAOSMC', '1C', 'FLEX CLASS', 'baggage23kg', 1, false),
    (3, 'FC894', 'HAOSMC', '1C', 'FLEX CLASS', 'baggage32kg', 0, false),
    (4, 'HE176', 'POWTWU', '1D', 'FLEX CLASS', 'baggage32kg', 1, false),
    (5, 'HE176', 'POWTWU', '1E', 'FLEX CLASS', 'baggage32kg', 1, false),
    (6, 'HE176', 'POWTWU', '1F', 'FLEX CLASS', 'baggage32kg', 1, false),
    (4, 'BE290', 'POWTWU', '2D', 'FLEX CLASS', 'baggage32kg', 1, false),
    (5, 'BE290', 'POWTWU', '2E', 'FLEX CLASS', 'baggage32kg', 1, false),
    (6, 'BE290', 'POWTWU', '2F', 'FLEX CLASS', 'baggage32kg', 1, false),
    (4, 'HE176', 'POWTWU', '1D', 'FLEX CLASS', 'baggage23kg', 0, false),
    (5, 'HE176', 'POWTWU', '1E', 'FLEX CLASS', 'baggage23kg', 0, false),
    (6, 'HE176', 'POWTWU', '1F', 'FLEX CLASS', 'baggage23kg', 0, false),
    (4, 'BE290', 'POWTWU', '2D', 'FLEX CLASS', 'baggage23kg', 0, false),
    (5, 'BE290', 'POWTWU', '2E', 'FLEX CLASS', 'baggage23kg', 0, false),
    (6, 'BE290', 'POWTWU', '2F', 'FLEX CLASS', 'baggage23kg', 0, false),
    (7, 'LI603', 'UIWERT', '1E', 'FLEX CLASS', 'baggage23kg', 0, false),
    (7, 'LI603', 'UIWERT', '1E', 'FLEX CLASS', 'baggage32kg', 0, false),
    (8, 'LI603', 'BCVZSD', '2F', 'FLEX CLASS', 'baggage23kg', 0, false),
    (8, 'LI603', 'BCVZSD', '2F', 'FLEX CLASS', 'baggage32kg', 0, false),
    (9, 'ZR567', 'LAKSJF', '1E', 'ECONOMY CLASS', 'baggage32kg', 1, false),
    (10, 'ZR567', 'LAKSJF', '5D', 'ECONOMY CLASS', 'baggage32kg', 1, false),
    (11, 'ZR567', 'LAKSJF', '5E', 'ECONOMY CLASS', 'baggage32kg', 1, false),
    (9, 'ZR567', 'LAKSJF', '1E', 'ECONOMY CLASS', 'baggage23kg', 0, false),
    (10, 'ZR567', 'LAKSJF', '5D', 'ECONOMY CLASS', 'baggage23kg', 0, false),
    (11, 'ZR567', 'LAKSJF', '5E', 'ECONOMY CLASS', 'baggage23kg', 0, false),
    (12, 'AT567', 'JQUERF', '7E', 'ECONOMY CLASS', 'baggage23kg', 1, false),
    (13, 'AT567', 'JQUERF', '1A', 'ECONOMY CLASS', 'baggage23kg', 1, false),
    (12, 'AT567', 'JQUERF', '7E', 'ECONOMY CLASS', 'baggage32kg', 0, false),
    (13, 'AT567', 'JQUERF', '1A', 'ECONOMY CLASS', 'baggage32kg', 0, false),
    (14, 'AT567', 'AKJDFN', '2A', 'ECONOMY CLASS', 'baggage23kg', 0, false),
    (15, 'AT789', 'SAKSDM', '2D', 'ECONOMY CLASS', 'baggage23kg', 0, false),
    (16, 'AT789', 'QEIEUN', '3E', 'ECONOMY CLASS', 'baggage23kg', 0, false),
    (17, 'AT802', 'SAHIBU', '1F', 'ECONOMY CLASS', 'baggage23kg', 0, false),
    (18, 'AT802', 'ADDBSA', '2C', 'ECONOMY CLASS', 'baggage23kg', 0, false),
    (19, 'AT466', 'IAKDKA', '1B', 'ECONOMY CLASS', 'baggage23kg', 0, false),
    (20, 'AT466', 'AKSDJS', '4D', 'ECONOMY CLASS', 'baggage23kg', 0, false),
    (14, 'AT567', 'AKJDFN', '2A', 'ECONOMY CLASS', 'baggage32kg', 0, false),
    (15, 'AT789', 'SAKSDM', '2D', 'ECONOMY CLASS', 'baggage32kg', 0, false),
    (16, 'AT789', 'QEIEUN', '3E', 'ECONOMY CLASS', 'baggage32kg', 0, false),
    (17, 'AT802', 'SAHIBU', '1F', 'ECONOMY CLASS', 'baggage32kg', 0, false),
    (18, 'AT802', 'ADDBSA', '2C', 'ECONOMY CLASS', 'baggage32kg', 0, false),
    (19, 'AT466', 'IAKDKA', '1B', 'ECONOMY CLASS', 'baggage32kg', 0, false),
    (20, 'AT466', 'AKSDJS', '4D', 'ECONOMY CLASS', 'baggage32kg', 0, false);
""")
""")
```

Και τέλος η δημιουργία ενός trigger ώστε πριν από insert queries στον πίνακα contains να διαγράφονται expired κρατήσεις/ενοικιάσεις αυτοκινήτων:

```
#create the trigger function that deletes the expired car rentings
cur.execute("""DROP TRIGGER IF EXISTS before_insert_trigger_car ON contains""")
cur.execute("""DROP FUNCTION IF EXISTS delete_expired_car_rentings()""")
cur.execute("""
    CREATE OR REPLACE FUNCTION delete_expired_car_rentings()
    RETURNS TRIGGER AS $$%
    BEGIN
        DELETE FROM contains
        WHERE return < CURRENT_TIMESTAMP;

        RETURN NEW;
    END;
    $$ LANGUAGE plpgsql;
""")

#create the trigger that automatically is executed before every insert in the table contains
cur.execute("""
    CREATE TRIGGER before_insert_trigger_car
    BEFORE INSERT ON contains
    FOR EACH ROW EXECUTE FUNCTION delete_expired_car_rentings();
""")
```

7. ΟΔΗΓΙΕΣ ΕΓΚΑΤΑΣΤΑΣΗΣ ΤΗΣ ΕΦΑΡΜΟΓΗΣ

Η εγκατάσταση και χρήση της εφαρμογής “FlyNow” προϋποθέτει την εφαρμογή “**pgAdmin4**” για δημιουργία βάσης δεδομένων, την εφαρμογή “**Android Studio**” για την υλοποίηση της εφαρμογής και την εφαρμογή “**Tailscale**” για την σύνδεση του κινητού τηλεφώνου και του υπολογιστή με τον server. Περισσότερες πληροφορίες για την εφαρμογή “**pgAdmin4**” υπάρχουν στο link:

<https://www.pgadmin.org/download/>. Για την εγκατάσταση του “**Android Studio**” προτείνεται το link:

https://developer.android.com/studio?gclid=Cj0KCQiAnfmsBhDfARIsAM7MKi0tyN38_Yx0WC8QNL4HEW8G3uc189kPdVk2rzWsZdmFVYLyma8DabcaApohEALw_wcB&gclsrc=aw.ds

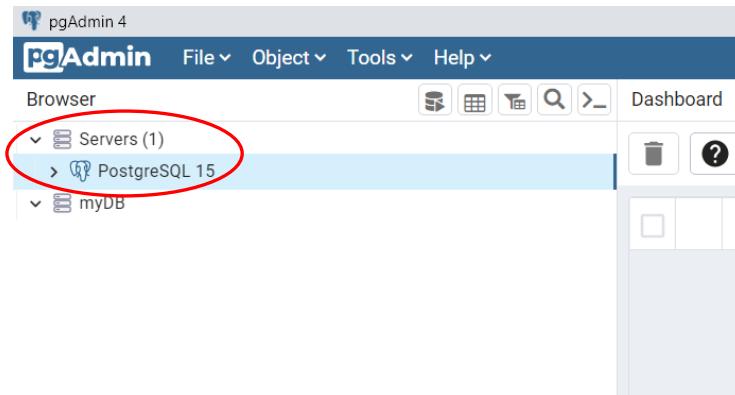
Για την εγκατάσταση του “**Tailscale**” στον υπολογιστή προτείνεται το link:

<https://tailscale.com/>

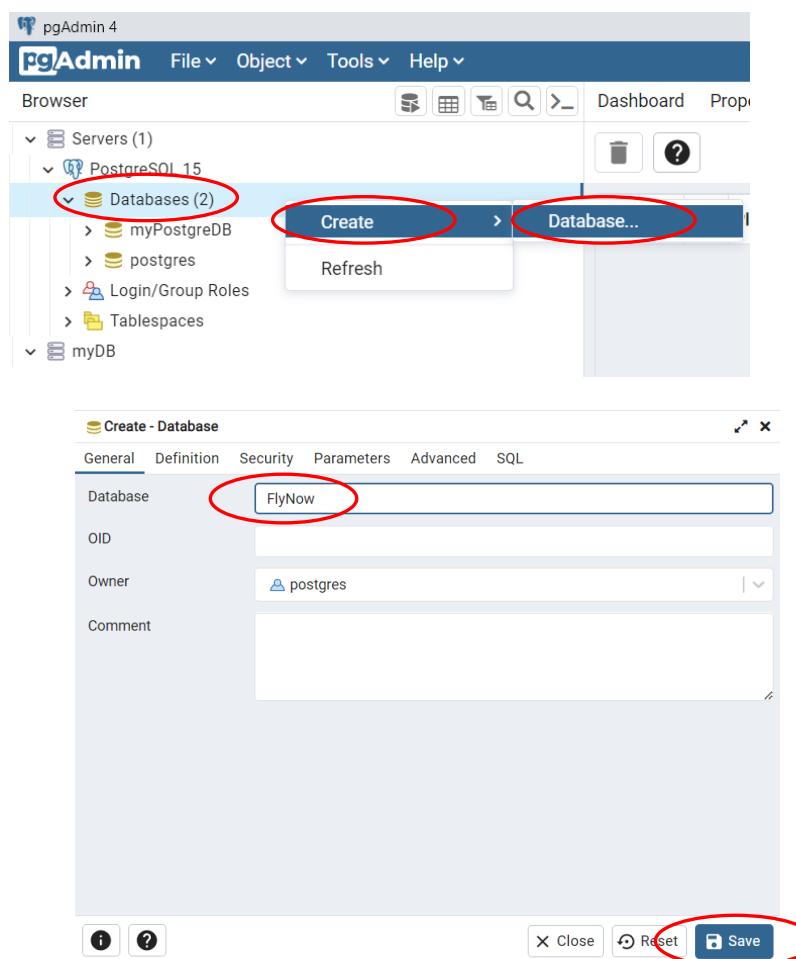
Η εφαρμογή θα πρέπει να εγκατασταθεί και στο κινητό τηλέφωνο.

Αφού είναι εγκατεστημένες οι δύο παραπάνω εφαρμογές και έχει γίνει είσοδος στην εφαρμογή “**pgAdmin4**” πληκτρολογώντας τον κωδικό που ζητάει ακολουθούν τα επόμενα βήματα:

**1. Επιλέγουμε τον server που χρησιμοποιείται από
“Servers”, εδώ ο server “PostgreSQL 15”.**



**2. Κάνουμε κλικ στο πεδίο “Databases” και δημιουργούμε
μία νέα βάση δεδομένων με όνομα “FlyNow”.**



The screenshot shows the pgAdmin 4 interface. In the main pane, under the 'PostgreSQL 15' server, the 'Databases' node is expanded, showing two existing databases: 'myPostgreDB' and 'postgres'. To the right of the database list, there is a toolbar with a 'Create' button and a 'Database...' button. A red circle highlights the 'Create' button. Below the main pane, a detailed 'Create - Database' dialog box is open. It has tabs for General, Definition, Security, Parameters, Advanced, and SQL. The General tab is selected, showing fields for 'Database' (containing 'FlyNow'), 'OID' (empty), 'Owner' (set to 'postgres'), and 'Comment' (empty). At the bottom of the dialog, there are 'Close', 'Reset', and 'Save' buttons. The 'Save' button is highlighted with a red circle.

3. Στο “Visual Studio” ή οποιονδήποτε άλλον editor ανοίγουμε το αρχείο “creationOfDatabase.py” και βάζουμε το password που θέσαμε στην postgres.

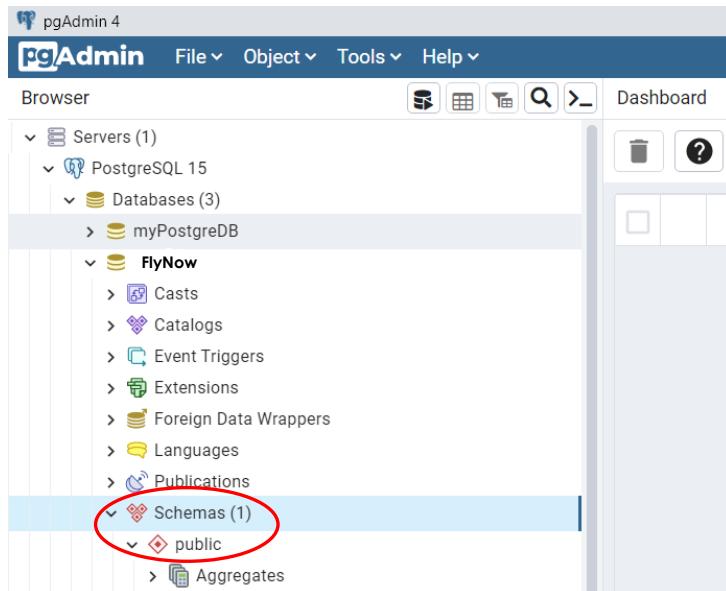
```
8     #connection with database FlyNow,
9     #you must create a database with name FlyNow
10    #and password is different, you must type your own
11    #password of the user postgres
12    conn = psycopg2.connect(
13        database="FlyNow",
14        user="postgres",
15        password="your_password",
16        host="localhost",
17        port="5432"
18    )
```

4. Ανοίγουμε ένα terminal και στο path όπου βρίσκεται το αρχείο “creationOfDatabase.py” τρέχουμε την παρακάτω εντολή για να ολοκληρωθεί η βάση δεδομένων.

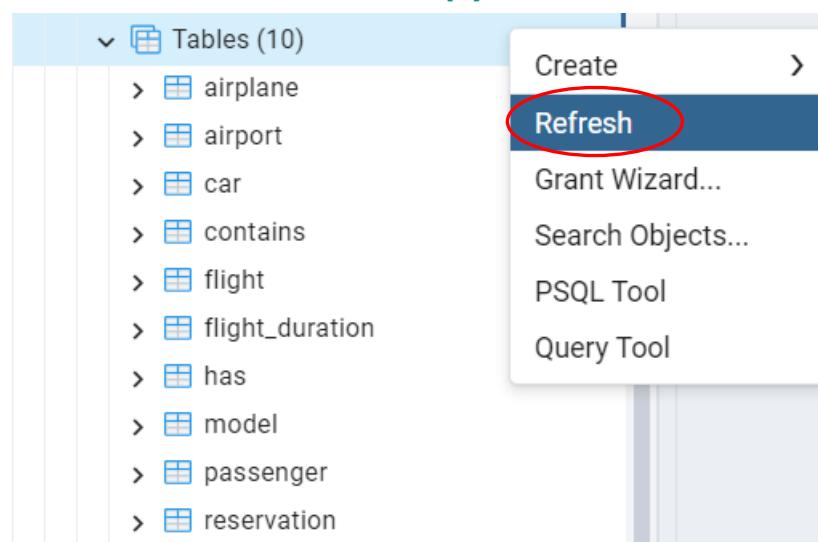
```
C:\Users\garyf\Desktop\FlyNowFiles>python creationOfDatabase.py
```

Προσοχή! Ο φάκελος images που περιέχει τις εικόνες των αυτοκινήτων πρέπει να βρίσκεται στον ίδιο φάκελο με το αρχείο creationOfDatabase.py.

5. Επιλέγουμε το πεδίο “Schemas” στην βάση που δημιουργήθηκε και στην συνέχεια επιλέγουμε το σχήμα “public”.



6. Κάνοντας “Refresh” στο “Tables” εμφανίζονται όλοι οι πίνακες που δημιουργήθηκαν από τις εντολές του αρχείου “creationOfDatabase.py”.



7. Κάνοντας δεξιή κλικ στο πεδίο “Tables” → “Refresh” και στην συνέχεια δεξιή κλικ σε οποιονδήποτε πίνακα “Show/Edit Data” και “All Rows” εμφανίζονται όλες οι εγγραφές του πίνακα.

flightid	flightdate	departuretime	arrivaltime	economyprice	flexprice	businessprice	departureairport	arrivalairport
AM124	2024-02-07	13:50:00+01:00	18:20:00+02:00	85.00	100.00	170.00	AMS	ATH
AM192	2024-02-05	07:20:00+01:00	11:40:00+02:00	90.00	105.00	180.00	AMS	ATH
AM234	2024-01-28	14:50:00+01:00	16:20:00+01:00	50.00	65.00	100.00	AMS	PRG
AT122	2024-02-02	22:00:00+01:00	00:10:00+00:00	75.00	90.00	150.00	AMS	LIS
AT123	2024-01-11	18:50:00+02:00	19:40:00+02:00	95.00	110.00	190.00	ATH	MUT
AT124	2024-01-10	18:50:00+02:00	19:40:00+02:00	50.00	65.00	100.00	ATH	SKG
AT151	2024-02-09	17:50:00+02:00	18:45:00+02:00	60.00	75.00	120.00	ATH	MUT
AT154	2024-02-09	10:50:00+02:00	11:45:00+02:00	60.00	75.00	120.00	ATH	SKG
AT158	2024-02-09	09:50:00+02:00	09:50:00+02:00	70.00	85.00	140.00	ATH	MUT
AT159	2024-02-02	08:35:00+02:00	12:15:00+01:00	105.00	120.00	210.00	AMS	AMS
AT220	2024-01-28	18:25:00+02:00	20:05:00+01:00	60.00	75.00	120.00	PRG	PRG
AT222	2024-01-28	09:25:00+02:00	11:05:00+01:00	80.00	95.00	160.00	ATH	PRG

8. Στην συνέχεια, για να ανοίξουμε τον server επιβεβαιώνουμε αρχικά ότι υπάρχουν τα παρακάτω αρχεία στον φάκελο του.

Όνομα	Ημερομηνία τροποποί...	Τύπος	Μέγεθος
node_modules	25/12/2023 9:27 μμ	Φάκελος αρχείων	
package.json	25/12/2023 9:27 μμ	JSON Source File	1 KB
package-lock.json	25/12/2023 9:27 μμ	JSON Source File	94 KB
server.js	9/1/2024 11:28 μμ	JavaScript Source ...	135 KB

9. Στο “Visual Studio” ή οποιονδήποτε άλλον editor ανοίγουμε το αρχείο “server.js” και βάζουμε το password που θέσαμε στην postgres.

```
7 //connection to the database and settings about the server
8 const pool = new Pool({
9   user: 'postgres',
10  host: 'localhost',
11  database: 'FlyNow',
12  password: 'your_password', //password highlighted with a red oval
13  port: 5432
14 })
```

10. Στο terminal και στο path όπου βρίσκεται το αρχείο “server.js ” τρέχουμε την παρακάτω εντολή για να ανοίξει o server.

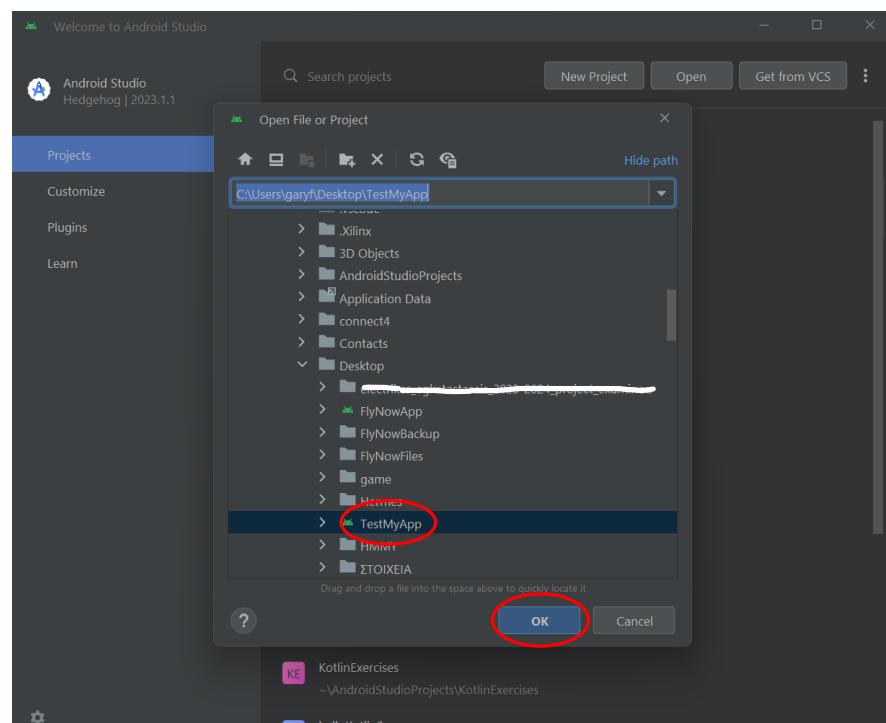
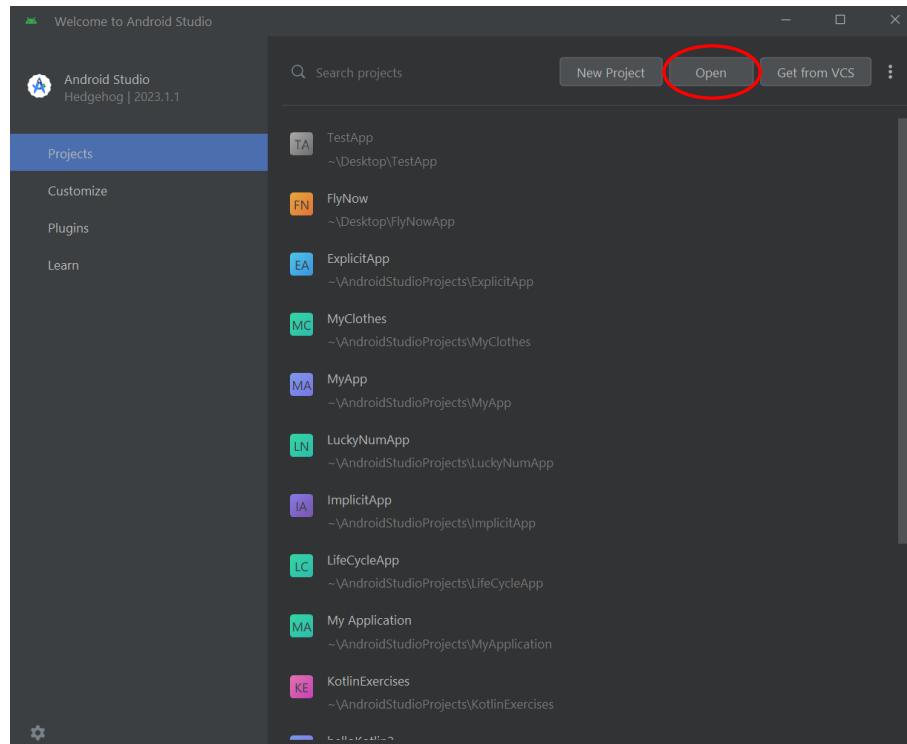
```
C:\Users\garyf\Desktop\FlyNowFiles\FlyNowServer>npm start
```

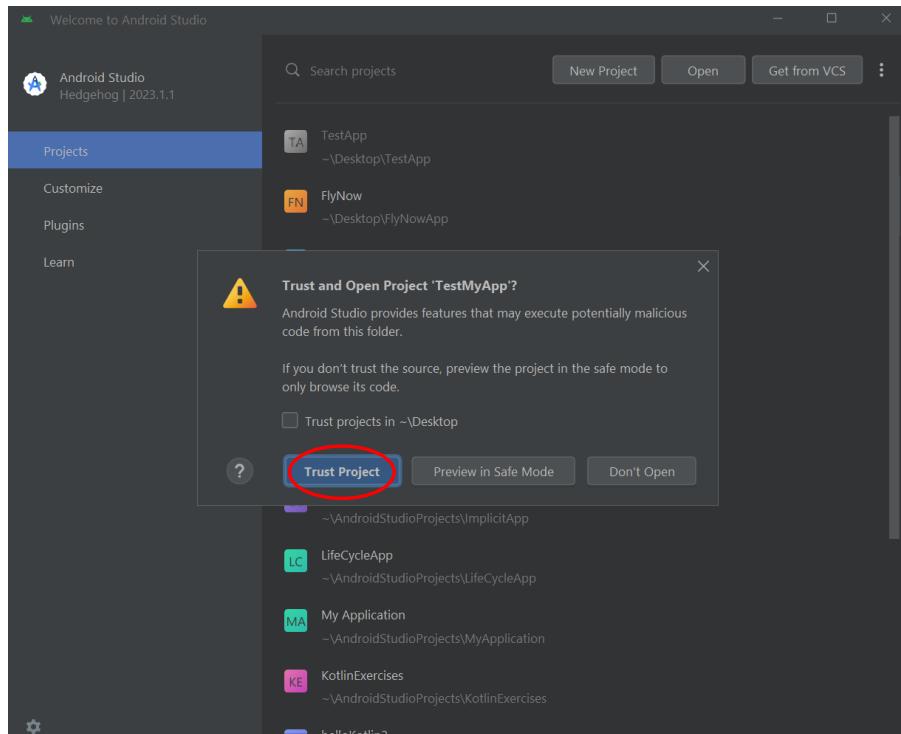
Εάν o server ανοίξει επιτυχώς θα δούμε το παρακάτω μήνυμα:

```
> backend@1.0.0 start
> nodemon server.js

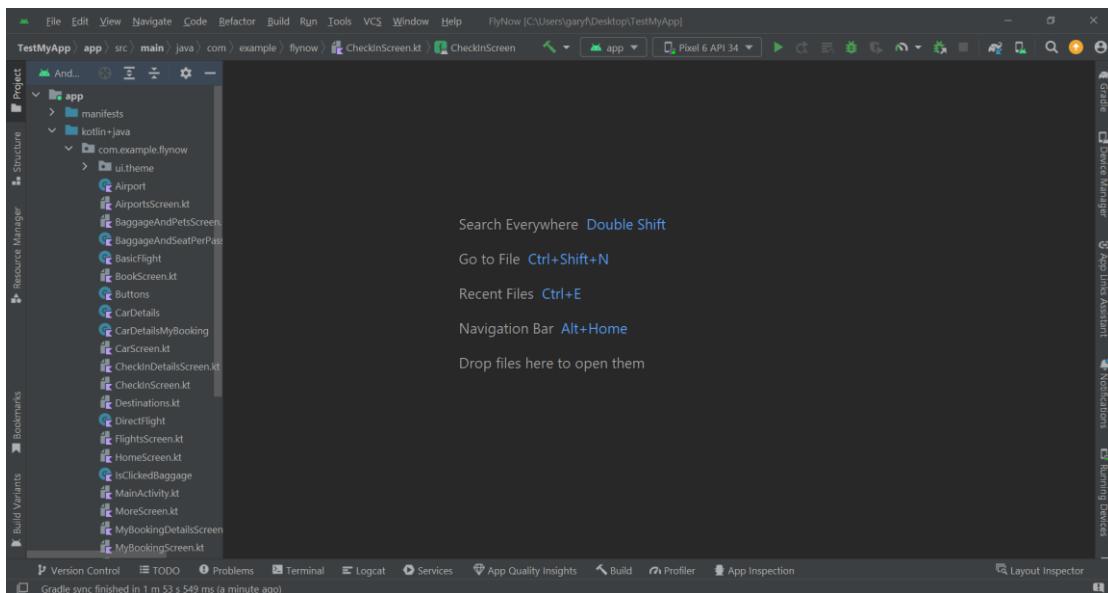
[nodemon] 3.0.2
[nodemon] to restart at any time, enter 'rs'
[nodemon] watching path(s): ***!
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting 'node server.js'
Server is open at 5000
```

11. Ανοιγουμε το “Android Studio”, πατάμε “Open” και επιλέγουμε τον φάκελο “FlyNowApp” (στο παράδειγμα, το όνομα του φακέλου είναι “TestMyApp”).





12. Τώρα μπορούμε να δούμε τον κώδικα της εφαρμογής και την υλοποίηση της από τα παρακάτω αρχεία.



13. Τέλος για να τρέξουμε την εφαρμογή θα πρέπει αφού έχουμε εγκαταστήσει την εφαρμογή "Tailscale" στον υπολογιστή να τρέξουμε την εντολή:

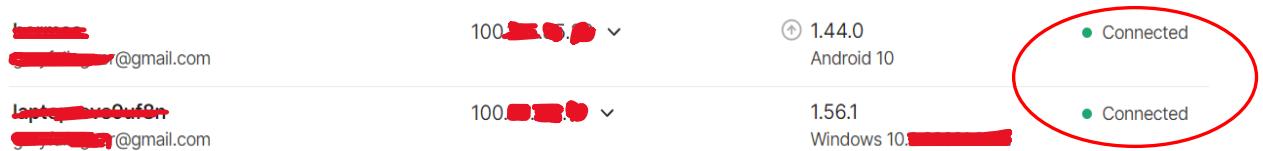
```
C:\Users\garyf\Desktop\FlyNowFiles\FlyNowServer>tailscale up
```

Για να αποσυνδέσουμε τον υπολογιστή, αφού έχουμε τελειώσει με την εφαρμογή, τρέχουμε την εντολή:

“tailscale down”. Εάν αποσυνδεθούμε δεν θα υπάρχει πια σύνδεση με τον server και η εφαρμογή δεν θα μπορεί να αντλήσει δεδομένα από την βάση.

14. Ενεργοποιούμε το “Tailscale” και στο κινητό τηλέφωνο πατώντας το “Active” στην εφαρμογή που έχουμε εγκαταστήσει.

15. Στο link του “Tailscale” βλέπουμε τις συσκευές που είναι συνδεδεμένες, στην περίπτωση μας θέλουμε το κινητό τηλέφωνο και τον υπολογιστή.



16. Αντιγράφουμε την IP διεύθυνση του υπολογιστή που μας δίνει και την αντικαθιστούμε με την IP που υπάρχει, στο πεδίο “url”, σε όλα τα αρχεία με κατάληξη “.kt” του directory “com.example.flynow”.

A screenshot of the Android Studio IDE showing the code for `AirportsScreen.kt`. The code contains a line with a circled URL: `val url = "http://100.10.10.10:8080/flynow/api/airports"`. This URL is part of a function that fetches data from an API. The code is written in Kotlin and uses CoroutinesScope.

```
viewmodelScope.launch { thisCoroutineScope
    fetchDataFromApi { it.listOfAirports
        _airports.value = it
    }
}

//function that runs in the initialization of the airports list that the api returns with the get method
private fun fetchDataFromApi(result: (List<Airport>) -> Unit) {
    val url = "http://100.10.10.10:8080/flynow/api/airports"
    val queue: RequestQueue = volley.newRequestQueue(getApplicationContext())
    val request = JsonArrayRequest(Request.Method.GET, url, jsonObjectRequest: null,
        { response -
            try {
                val airports: List<Airport> = parseJson(response.toString())
                onResult(airports)
            } catch (e: Exception) {
                e.printStackTrace()
            }
        },
        { error -
            Log.d("tag", "Error", error.toString())
            Toast.makeText(getApplicationContext(), "Fail to get response", Toast.LENGTH_SHORT).show()
        }
    )
    queue.add(request)
}
```

Προσοχή! Το “url” πρέπει να έχει την παρακάτω μορφή, χωρίς κενά:

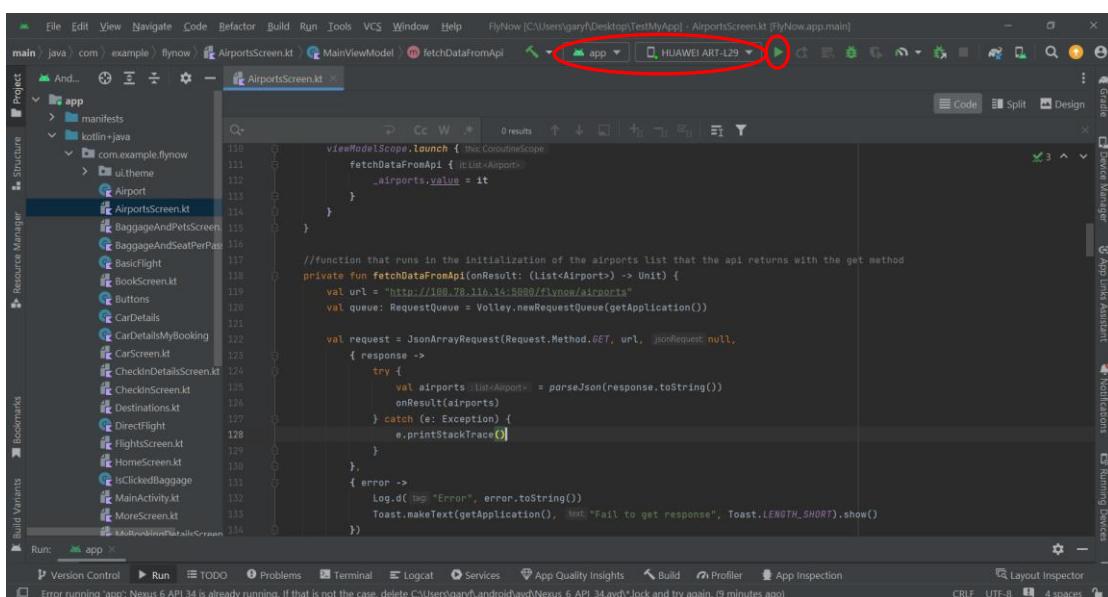
```
val url = "http://ip_ypologisti:5000/flynow/airports"
```

17. Για να τρέξουμε την εφαρμογή στο Android κινητό τηλέφωνο μας, πρέπει να ακολουθήσουμε τις οδηγίες του παρακάτω link για να αποκτήσουμε στο τηλέφωνο μας «Ιδιότητα Προγραμματιστή».

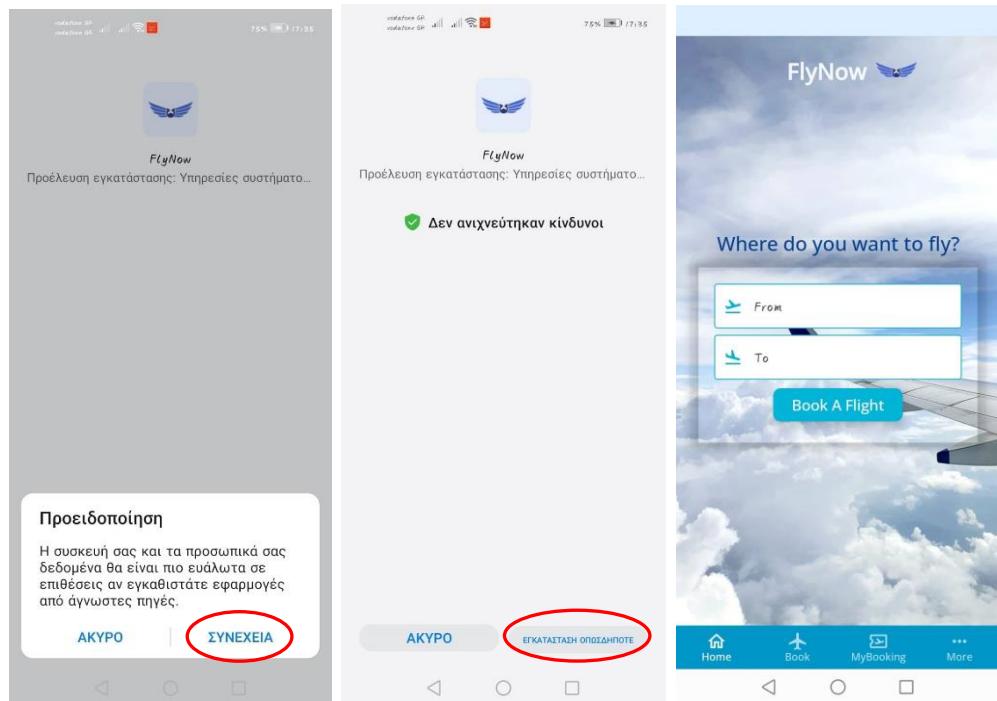
<https://developer.android.com/studio/debug/dev-options>

Προσοχή ακολουθούμε τα βήματα μέχρι το “General options” (όχι το “General options”).

18. Στην συνέχεια συνδέουμε το κινητό μας, με καλώδιο, με τον υπολογιστή, επιλέγουμε στο κινητό «Μεταφορά Αρχείων» (αν μας εμφανίσει σχετικό μήνυμα), επιλέγουμε τον υπολογιστή που εμφανίζεται και στο “Android Studio” πατάμε “Run”. Προσοχή στα δύο πεδία που φαίνονται παρακάτω πρέπει να υπάρχει το “app” και δίπλα η συσκευή τηλεφώνου που συνδέσαμε.



19. Τέλος, εμφανίζονται στην συσκευή τηλεφώνου τα παρακάτω και η εφαρμογή έχει εγκατασταθεί επιτυχώς.



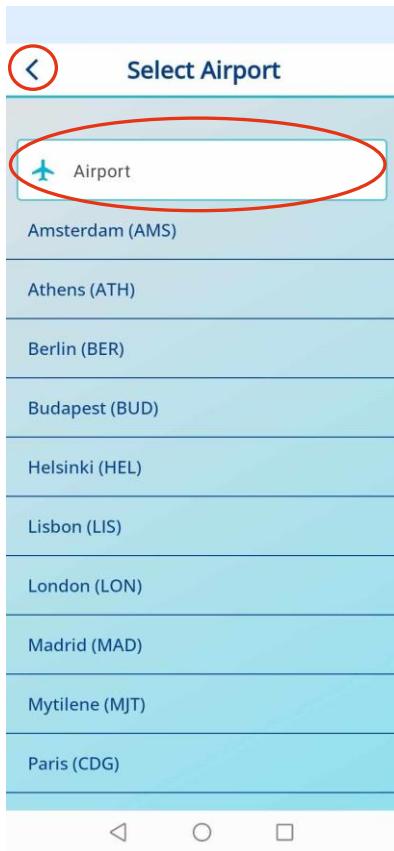
8. ΕΞΗΓΗΣΗ ΤΟΥ BACKEND ΠΑΝΩ ΣΤΗΝ ΕΦΑΡΜΟΓΗ ΚΑΙ ΟΔΗΓΙΕΣ ΧΡΗΣΗΣ

Για την υλοποίηση κάθε σελίδας της εφαρμογής, εκτελούνται κάποια queries μέσα από APIs που έχουν δημιουργηθεί σε JavaScript στο αρχείο “node.js”. Η χρήση της εφαρμογής καθώς και τα queries που τρέχουν από πίσω φαίνονται παρακάτω.

ΑΡΧΙΚΗ ΟΘΟΝΗ ΚΑΙ ΟΛΟΚΛΗΡΩΣΗ ΚΡΑΤΗΣΗΣ



Η αρχική οθόνη “Home” της εφαρμογής “FlyNow” περιέχει δύο πεδία, το “From” που αντιπροσωπεύει το αεροδρόμιο αναχώρησης, και το “To” που αντιπροσωπεύει το αεροδρόμιο άφιξης. Κάνοντας κλικ στο σύμβολο του αεροπλάνου, εμφανίζεται μία λίστα με όλα τα διαθέσιμα αεροδρόμια για πτήσεις. Επιπρόσθετα, στο κάτω μέρος της οθόνης εμφανίζεται το μενού με το οποίο ο χρήστης μπορεί να περιηγηθεί στις διάφορες σελίδες της εφαρμογής, που θα εξηγηθούν στην συνέχεια.



Η λίστα με τα αεροδρόμια φαίνεται στην οθόνη δίπλα και υπάρχει δυνατότητα αναζήτησης συγκεκριμένου αεροδρομίου από το πεδίο “Airport”. Το αεροδρόμιο που επιλέγεται ως αεροδρόμιο αναχώρησης αφαιρείται από την λίστα με τα αεροδρόμια άφιξης στην συνέχεια της κράτησης. Με το κουμπί “πίσω” επιστρέφουμε στην αρχική οθόνη. Το GET API και το query που τρέχει στην βάση φαίνεται παρακάτω.

```
//GET api that makes query to get the airports from the database
app.get('/flynow/airports', async (req, res) => {
  try {
    const result = await pool.query('SELECT * FROM airport ORDER BY city ASC')
    res.json(result.rows)
  } catch (error) {
    console.error(error)
    res.status(500).json({ error: 'Internal Server Error' })
  }
})
```

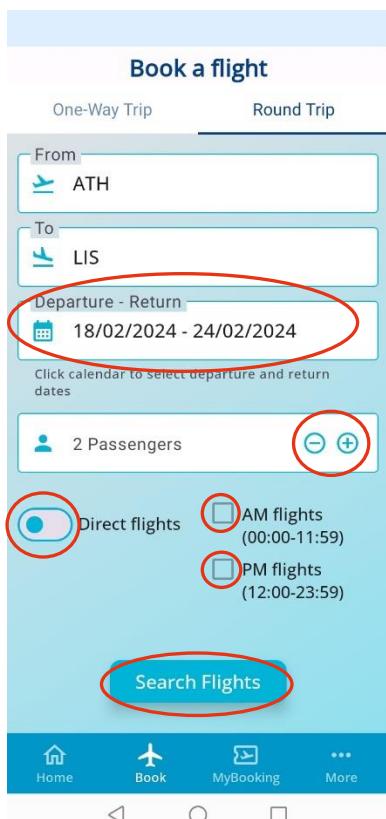


Με τον ίδιο τρόπο επιλέγουμε και το αεροδρόμιο άφιξης και στην συνέχεια πατάμε το κουμπί “Book A Flight” για να μεταφερθούμε στην παρακάτω σελίδα.

Τώρα έχουμε μεταφερθεί στην σελίδα “Book A Flight” οπού αφού επιλέξουμε “One-Way Trip” ή “Round Trip”, είτε κάνοντας κλικ πάνω είτε σκρολάροντας στο πλάι, συμπληρώνουμε επιπλέον στοιχεία για την πτήση. Με κλικ πάνω στο σύμβολο ημερολόγιο εμφανίζεται ένα ημερολόγιο



με δυνατότητα επιλογής της ημερομηνίας αναχώρησης (για One-Way Trip) ή και επιστροφής (για Round Trip). Κάνοντας κλικ στο κουμπί “OK” η/οι ημερομηνία/ες που επιλέχθηκε/αν φαίνονται στο πεδίο “Departure - Return”.



Τα σύμβολα “+” και “-“ αυξάνουν ή μειώνουν τον αριθμό των επιβατών με ελάχιστη τιμή το 1 και μέγιστη το 10. Το κουμπί “Direct flights” όταν ενεργοποιηθεί δείχνει στον χρήστη μόνο απευθείας πτήσεις, χωρίς ενδιάμεση στάση, σε αντίθετη περίπτωση εμφανίζονται όλες οι πτήσεις. Εάν ο χρήστης κάνει κλικ στο πεδίο “AM flights” τότε εμφανίζονται πτήσεις με αναχώρηση από τις 00:00 έως τις 11:59, ενώ αν κάνει κλικ στο πεδίο “PM flights” τότε εμφανίζονται πτήσεις με αναχώρηση από τις 12:00 έως τις 23:59. Εάν δεν γίνει κλικ σε κανένα από τα δύο πεδία τότε εμφανίζονται όλες οι πτήσεις για τις ημερομηνίες που επέλεξε.

Τα πεδία “AM flights” και “PM flights” μπορούν να επιλεχθούν μόνο εάν έχει επιλεχτεί το πεδίο “Direct flights”.

Τέλος κάνοντας κλικ στο κουμπί “Search Flights” ξεκινούν τα queries για αναζήτηση πτήσεων με τα δεδομένα κριτήρια και ο χρήστης μεταφέρεται στην σελίδα “Flights”.

Τρέχει το ακόλουθο POST API:

```
app.post('/flynow/flights', async (req, res) => {
```

Εάν επιλεχθεί AM flights,

Η αναχώρηση:

```
result = await pool.query(`SELECT f.flightid, TO_CHAR(f.flightdate, 'DD/MM/YYYY') AS flightdate, f.departuretime, f.arrivaltime, f.economyprice, f.flexprice, f.businessprice, f.departureairport, f.arrivalairport, a1.city as departurecity, a2.city as arrivalcity, ap.model as airplanemodel, fd.duration AS flightduration
FROM flight f, airport a1, airport a2, airplane ap, flight_duration fd, model m
WHERE f.departuretime=fd.departuretime
AND f.arrivaltime=fd.arrivaltime
AND f.departureairport=a1.name
AND f.arrivalairport=a2.name
AND ap.airplaneid=f.airplane
AND m.modelid=ap.model
AND f.departureairport='${jsonArray[0].from}'
AND f.arrivalairport='${jsonArray[0].to}'
AND f.flightdate=TO_DATE('${jsonArray[0].departureDate}', 'DD/MM/YYYY')
AND EXTRACT(HOUR FROM f.departuretime) * 60 + EXTRACT(MINUTE FROM f.departuretime) >= 0
AND EXTRACT(HOUR FROM f.departuretime) * 60 + EXTRACT(MINUTE FROM f.departuretime) <= 11 * 60 + 59
AND ${jsonArray[0].passengersCount} <= m.capacity -
    SELECT COUNT(DISTINCT(h.seatnumber))
    FROM has h
    WHERE f.flightid=h.flightid
`)
```

Η επιστροφή:

```
result = await pool.query(`SELECT f.flightid, TO_CHAR(f.flightdate, 'DD/MM/YYYY') AS flightdate, f.departuretime, f.arrivaltime, f.economyprice, f.flexprice, f.businessprice, f.departureairport, f.arrivalairport, a1.city as departurecity, a2.city as arrivalcity, ap.model as airplanemodel, fd.duration AS flightduration
FROM flight f, airport a1, airport a2, airplane ap, flight_duration fd, model m
WHERE f.departuretime=fd.departuretime
AND f.arrivaltime=fd.arrivaltime
AND f.departureairport=a1.name
AND f.arrivalairport=a2.name
AND ap.airplaneid=f.airplane
AND m.modelid=ap.model
AND f.departureairport='${jsonArray[0].to}'
AND f.arrivalairport='${jsonArray[0].from}'
AND f.flightdate=TO_DATE('${jsonArray[0].returnDate}', 'DD/MM/YYYY')
AND EXTRACT(HOUR FROM f.departuretime) * 60 + EXTRACT(MINUTE FROM f.departuretime) >= 0
AND EXTRACT(HOUR FROM f.departuretime) * 60 + EXTRACT(MINUTE FROM f.departuretime) <= 11 * 60 + 59
AND ${jsonArray[0].passengersCount} <= m.capacity -
    SELECT COUNT(DISTINCT(h.seatnumber))
    FROM has h
    WHERE f.flightid=h.flightid
`)
```

Εάν επιλεχθεί PM flights,

Η αναζόρηση:

```
result = await pool.query(`  
SELECT f.flightid, TO_CHAR(f.flightdate, 'DD/MM/YYYY') AS flightdate, f.departuretime, f.arrivaltime,  
f.economyprice, f.flexprice, f.businessprice, f.departureairport,  
f.arrivalairport, a1.city as departurecity, a2.city as arrivalcity,  
ap.model as airplanemodel, fd.duration AS flightduration  
FROM flight f, airport a1, airport a2, airplane ap, flight_duration fd, model m  
WHERE f.departuretime=fd.departuretime  
AND f.arrivaltime=fd.arrivaltime  
AND f.departureairport=a1.name  
AND f.arrivalairport=a2.name  
AND ap.airplaneid=f.airplane  
AND m.modelid=ap.model  
AND departureairport='${jsonArray[0].from}'  
AND arrivalairport='${jsonArray[0].to}'  
AND flightdate=TO_DATE('${jsonArray[0].departureDate}', 'DD/MM/YYYY')  
AND EXTRACT(HOUR FROM f.departuretime) * 60 + EXTRACT(MINUTE FROM f.departuretime) >= 12 * 60  
AND EXTRACT(HOUR FROM f.departuretime) * 60 + EXTRACT(MINUTE FROM f.departuretime) <= 23 * 60 + 59  
AND ${jsonArray[0].passengersCount} <= m.capacity - (  
    SELECT COUNT(DISTINCT(h.seatnumber))  
    FROM has h  
    WHERE f.flightid=h.flightid)  
`)
```

Η επιστροφή:

```
result = await pool.query(`  
SELECT f.flightid, TO_CHAR(f.flightdate, 'DD/MM/YYYY') AS flightdate, f.departuretime, f.arrivaltime,  
f.economyprice, f.flexprice, f.businessprice, f.departureairport,  
f.arrivalairport, a1.city as departurecity, a2.city as arrivalcity,  
ap.model as airplanemodel, fd.duration AS flightduration  
FROM flight f, airport a1, airport a2, airplane ap, flight_duration fd, model m  
WHERE f.departuretime=fd.departuretime  
AND f.arrivaltime=fd.arrivaltime  
AND f.departureairport=a1.name  
AND f.arrivalairport=a2.name  
AND ap.airplaneid=f.airplane  
AND m.modelid=ap.model  
AND f.departureairport='${jsonArray[0].to}'  
AND f.arrivalairport='${jsonArray[0].from}'  
AND f.flightdate=TO_DATE('${jsonArray[0].returnDate}', 'DD/MM/YYYY')  
AND EXTRACT(HOUR FROM f.departuretime) * 60 + EXTRACT(MINUTE FROM f.departuretime) >= 12 * 60  
AND EXTRACT(HOUR FROM f.departuretime) * 60 + EXTRACT(MINUTE FROM f.departuretime) <= 23 * 60 + 59  
AND ${jsonArray[0].passengersCount} <= m.capacity - (  
    SELECT COUNT(DISTINCT(h.seatnumber))  
    FROM has h  
    WHERE f.flightid=h.flightid)  
`)
```

Εάν επιλεχθεί μόνο Direct flights και τίποτα άλλο,

Η αναχώρηση:

```
result = await pool.query(`  
SELECT f.flightid, TO_CHAR(f.flightdate, 'DD/MM/YYYY') AS flightdate, f.departuretime, f.arrivaltime,  
f.economyprice, f.flexprice, f.businessprice, f.departureairport,  
f.arrivalairport, a1.city as departurecity, a2.city as arrivalcity,  
ap.model as airplanemodel, fd.duration AS flightduration  
FROM flight f, airport a1, airport a2, airplane ap, flight_duration fd, model m  
WHERE f.departuretime=fd.departuretime  
AND f.arrivaltime=fd.arrivaltime  
AND f.departureairport=a1.name  
AND f.arrivalairport=a2.name  
AND ap.airplaneid=f.airplane  
AND m.modelid=ap.model  
AND departureairport='${jsonArray[0].from}'  
AND arrivalairport='${jsonArray[0].to}'  
AND flightdate=TO_DATE('${jsonArray[0].departureDate}', 'DD/MM/YYYY')  
AND ${jsonArray[0].passengersCount} <= m.capacity - (  
    SELECT COUNT(DISTINCT(h.seatnumber))  
    FROM has h  
    WHERE f.flightid=h.flightid)  
`)
```

Η επιστροφή:

```
result = await pool.query(`  
SELECT f.flightid, TO_CHAR(f.flightdate, 'DD/MM/YYYY') AS flightdate, f.departuretime, f.arrivaltime,  
f.economyprice, f.flexprice, f.businessprice, f.departureairport,  
f.arrivalairport, a1.city as departurecity, a2.city as arrivalcity,  
ap.model as airplanemodel, fd.duration AS flightduration  
FROM flight f, airport a1, airport a2, airplane ap, flight_duration fd, model m  
WHERE f.departuretime=fd.departuretime  
AND f.arrivaltime=fd.arrivaltime  
AND f.departureairport=a1.name  
AND f.arrivalairport=a2.name  
AND ap.airplaneid=f.airplane  
AND m.modelid=ap.model  
AND departureairport='${jsonArray[0].to}'  
AND arrivalairport='${jsonArray[0].from}'  
AND flightdate=TO_DATE('${jsonArray[0].returnDate}', 'DD/MM/YYYY')  
AND ${jsonArray[0].passengersCount} <= m.capacity - (  
    SELECT COUNT(DISTINCT(h.seatnumber))  
    FROM has h  
    WHERE f.flightid=h.flightid)  
`)
```

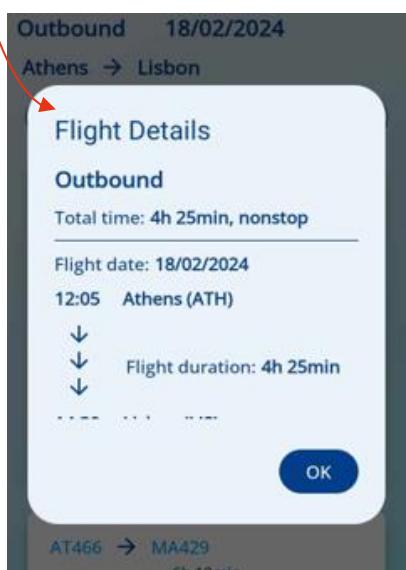
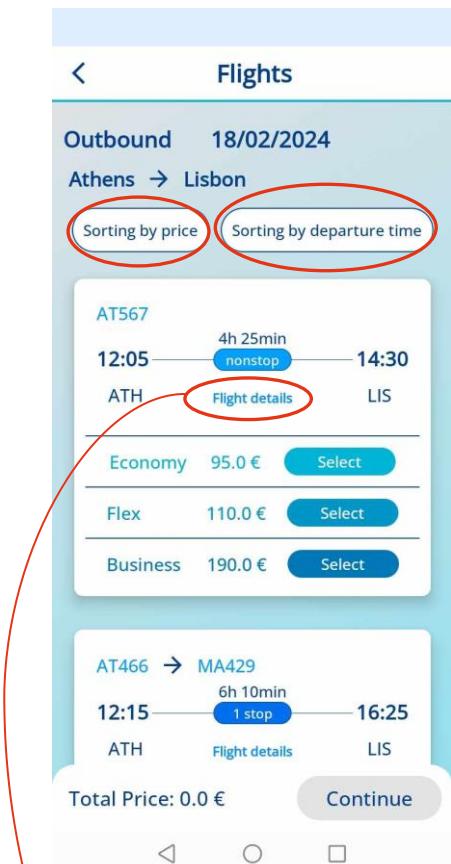
Εάν δεν επιλεχθεί ούτε Direct flights τρέχει το προηγούμενο query + το επόμενο,

Η αναχώρηση:

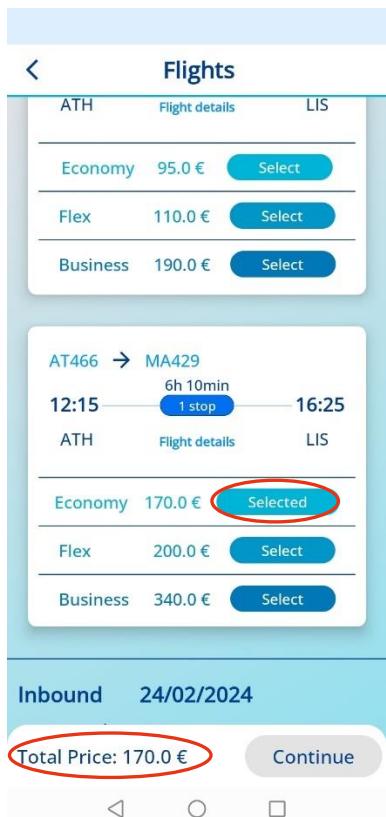
```
result1 = await pool.query(`  
SELECT  
f1.flightid AS first_flightid,  
TO_CHAR(f1.flightdate, 'DD/MM/YYYY') AS first_flightdate,  
f1.departuretime AS first_flight_departuretime,  
f1.arrivaltime AS first_flight_arrivaltime,  
f1.economyprice AS first_flight_economyprice,  
f1.flexprice AS first_flight_flexprice,  
f1.businessprice AS first_flight_businessprice,  
f1.departureairport AS first_flight_departureairport,  
f1.arrivalairport AS first_flight_arrivalairport,  
a1.city AS first_flight_departurecity,  
a2.city AS first_flight_arrivalcity,  
ap1.model AS first_flight_airplanemodel,  
fd1.duration AS first_flightduration,  
  
f2.flightid AS second_flightid,  
TO_CHAR(f2.flightdate, 'DD/MM/YYYY') AS second_flightdate,  
f2.departuretime AS second_flight_departuretime,  
f2.arrivaltime AS second_flight_arrivaltime,  
f2.economyprice AS second_flight_economyprice,  
f2.flexprice AS second_flight_flexprice,  
f2.businessprice AS second_flight_businessprice,  
f2.departureairport AS second_flight_departureairport,  
f2.arrivalairport AS second_flight_arrivalairport,  
a3.city AS second_flight_departurecity,  
a4.city AS second_flight_arrivalcity,  
ap2.model AS second_flight_airplanemodel,  
fd2.duration AS second_flightduration  
  
FROM  
flight f1  
JOIN airport a1 ON f1.departureairport = a1.name  
JOIN airport a2 ON f1.arrivalairport = a2.name  
JOIN airplane ap1 ON ap1.airplaneid = f1.airplane  
JOIN flight_duration fd1 ON f1.departuretime = fd1.departuretime AND f1.arrivaltime = fd1.arrivaltime  
JOIN flight f2 ON f2.departureairport = a2.name  
JOIN airport a3 ON f2.departureairport = a3.name  
JOIN airport a4 ON f2.arrivalairport = a4.name  
JOIN airplane ap2 ON ap2.airplaneid = f2.airplane  
JOIN flight_duration fd2 ON f2.departuretime = fd2.departuretime AND f2.arrivaltime = fd2.arrivaltime  
JOIN model m1 ON m1.modelid=ap1.model  
JOIN model m2 ON m2.modelid=ap2.model  
  
WHERE  
f1.departureairport = '${jsonArray[0].from}'  
AND f1.arrivalairport = f2.departureairport  
AND f2.arrivalairport = '${jsonArray[0].to}'  
AND f1.flightdate = TO_DATE('${jsonArray[0].departureDate}', 'DD/MM/YYYY')  
AND f2.flightdate = f1.flightdate  
AND EXTRACT(HOUR FROM f2.departuretime) * 60 + EXTRACT(MINUTE FROM f2.departuretime) >  
EXTRACT(HOUR FROM f1.arrivaltime) * 60 + EXTRACT(MINUTE FROM f1.arrivaltime)  
AND ${jsonArray[0].passengersCount} <= m1.capacity - (  
    SELECT COUNT(DISTINCT(h.seatnumber))  
    FROM has h  
    WHERE f1.flightid=h.flightid)  
AND ${jsonArray[0].passengersCount} <= m2.capacity - (  
    SELECT COUNT(DISTINCT(h.seatnumber))  
    FROM has h  
    WHERE f2.flightid=h.flightid)  
`)
```

Η επιστροφή:

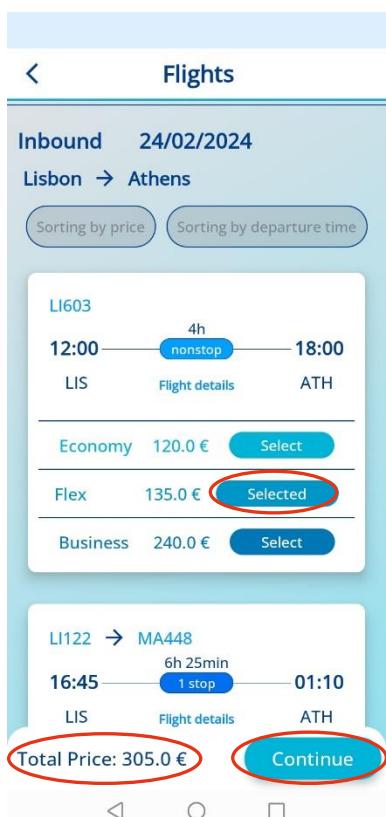
```
result1 = await pool.query(`  
SELECT  
f1.flightid AS first_flightid,  
TO_CHAR(f1.flighthdate, 'DD/MM/YYYY') AS first_flighthdate,  
f1.departuretime AS first_flight_departuretime,  
f1.arrivaltime AS first_flight_arrivaltime,  
f1.economyprice AS first_flight_economyprice,  
f1.flexprice AS first_flight_flexprice,  
f1.businessprice AS first_flight_businessprice,  
f1.departureairport AS first_flight_departureairport,  
f1.arrivalairport AS first_flight_arrivalairport,  
a1.city AS first_flight_departurecity,  
a2.city AS first_flight_arrivalcity,  
ap1.model AS first_flight_airplanemodel,  
fd1.duration AS first_flighthduration,  
  
f2.flightid AS second_flightid,  
TO_CHAR(f2.flighthdate, 'DD/MM/YYYY') AS second_flighthdate,  
f2.departuretime AS second_flight_departuretime,  
f2.arrivaltime AS second_flight_arrivaltime,  
f2.economyprice AS second_flight_economyprice,  
f2.flexprice AS second_flight_flexprice,  
f2.businessprice AS second_flight_businessprice,  
f2.departureairport AS second_flight_departureairport,  
f2.arrivalairport AS second_flight_arrivalairport,  
a3.city AS second_flight_departurecity,  
a4.city AS second_flight_arrivalcity,  
ap2.model AS second_flight_airplanemodel,  
fd2.duration AS second_flighthduration  
  
FROM  
flight f1  
JOIN airport a1 ON f1.departureairport = a1.name  
JOIN airport a2 ON f1.arrivalairport = a2.name  
JOIN airplane ap1 ON ap1.airplaneid = f1.airplane  
JOIN flight_duration fd1 ON f1.departuretime = fd1.departuretime AND f1.arrivaltime = fd1.arrivaltime  
JOIN flight f2 ON f2.departureairport = a2.name  
JOIN airport a3 ON f2.departureairport = a3.name  
JOIN airport a4 ON f2.arrivalairport = a4.name  
JOIN airplane ap2 ON ap2.airplaneid = f2.airplane  
JOIN flight_duration fd2 ON f2.departuretime = fd2.departuretime AND f2.arrivaltime = fd2.arrivaltime  
JOIN model m1 ON m1.modelid=ap1.model  
JOIN model m2 ON m2.modelid=ap2.model  
  
WHERE  
f1.departureairport = '${jsonArray[0].to}'  
AND f1.arrivalairport = f2.departureairport  
AND f2.arrivalairport = '${jsonArray[0].from}'  
AND f1.flighthdate = TO_DATE('${jsonArray[0].returnDate}', 'DD/MM/YYYY')  
AND f2.flighthdate = f1.flighthdate  
AND EXTRACT(HOUR FROM f2.departuretime) * 60 + EXTRACT(MINUTE FROM f2.departuretime) >  
EXTRACT(HOUR FROM f1.arrivaltime) * 60 + EXTRACT(MINUTE FROM f1.arrivaltime)  
AND ${jsonArray[0].passengersCount} <= m1.capacity - (  
    SELECT COUNT(DISTINCT(h.seatnumber))  
    FROM has h  
    WHERE f1.flightid=h.flightid)  
AND ${jsonArray[0].passengersCount} <= m2.capacity - (  
    SELECT COUNT(DISTINCT(h.seatnumber))  
    FROM has h  
    WHERE f2.flightid=h.flightid)  
`)
```



Στην σελίδα “Flights” εμφανίζονται όλες οι πτήσεις (εάν υπάρχουν) που πληρούν τα κριτήρια αναζήτησης. Το κουμπί “Sorting by price” ταξινομεί τις πτήσεις από την φθηνότερη στην ακριβότερη, ενώ το κουμπί “Sorting by departure time” ταξινομεί τις πτήσεις από αυτήν που αναχωρεί πιο νωρίς σε αυτή που αναχωρεί πιο αργά. Επιπρόσθετα υπάρχουν τρεις κατηγορίες πτήσης, η “Economy”, η “Flex” και η “Business” Class, η κάθε μία με διαφορετικά προνόμια και διαφορετική τιμή. Το κουμπί “Flight details”, όταν πατηθεί ανοίγει ένα παράθυρο με περισσότερες λεπτομέρειες για την πτήση.



Κάνοντας κλικ στο “Select” οποιασδήποτε κατηγορίας πτήσης, η πτήση επιλέγεται και η τιμή της προστίθεται στο πεδίο “Total Price”. Έστω ότι για την αναχώρηση επιλέξαμε την πτήση με μία στάση, σε “Economy” Class που φαίνεται δίπλα.



Στην συνέχεια με τον ίδιο τρόπο επιλέγουμε την πτήση της επιστροφής, έστω ότι επιλέγουμε την απευθείας πτήση σε “Flex Class” και η τιμή της προστίθεται στο “Total Price” (η τιμή αυτή αφορά έναν επιβάτη). Τέλος πατάμε το “Continue” για να συνεχίσουμε στην επόμενη σελίδα.

The screenshot shows the 'Passenger 1' section of a form titled 'Passengers'. It includes fields for Gender, First Name, Last Name, Birthdate, Email, and Phone Number, all marked with asterisks (*) indicating they are required. The 'Birthdate' field has a calendar icon. Below the form, it says 'Total Price: 610.0 €' and has a 'Continue' button.

Στην σελίδα “Passengers”, ο χρήστης συμπληρώνει τα στοιχεία όλων των επιβατών. Συγκεκριμένα, κάνοντας κλικ στο πεδίο “Gender*”, εμφανίζεται μία λίστα από την οποία ο χρήστης μπορεί να επιλέξει το φύλο του επιβάτη ανάμεσα σε “Male” και “Female”. Στην συνέχεια συμπληρώνει το όνομα και το επώνυμο του επιβάτη στα πεδία “First Name*” και “Last Name*”, την ημερομηνία γέννησης κάνοντας κλικ στο εικονίδιο του ημερολογίου στο πεδίο “Birthdate*”, όπως φαίνεται παρακάτω, το email στο πεδίο “Email*” και το τηλέφωνο στο πεδίο “Phone Number*”. Το “*” δηλώνει ότι όλα τα πεδία είναι υποχρεωτικά.

The image contains three screenshots of the 'Passenger' form. The first screenshot shows the 'Birthdate' input field with a date picker overlay. The second screenshot shows 'Passenger 1' filled with data (Female, Garyfalia, Georgitziki, 02/02/2002, garyfaliageor@gmail.com, +306943045362). The third screenshot shows 'Passenger 2' filled with data (Male, Vasilis, Anagnostopoulos, 22/04/2002, basilisanagnostopoulos02@gmail.com, +306985886422). Both passengers have 'Continue' buttons highlighted with red circles.

Κάνοντας κλικ στο “Continue”, και αφού έχουν συμπληρωθεί όλα τα στοιχεία των επιβατών, ο χρήστης μεταφέρεται στην επόμενη σελίδα.

The screenshot shows the "Seats" section for the Outbound journey. It displays a photograph of airplane seats and a message: "Select a seat from 1x to 30x where x: A,B,C,D,E,F. There is no charge for seat selection." Below this, there are two dropdown menus for "Seat" selection:

- Outbound:**
 - Mrs Garyfalia Georgitziki
 - Seat dropdown: Athens → Madrid (highlighted with a red circle)
 - Seat dropdown: Madrid → Lisbon

Total Price: 610.0 € Continue

Η επόμενη σελίδα “Seats” αφορά την επιλογή θέσης για κάθε πτήση και για κάθε επιβάτη. Κάνοντας κλικ στο πεδίο “Seats” εμφανίζεται μία λίστα με όλες τις κενές θέσεις του συγκεκριμένου αεροπλάνου, και κάθε φορά που μια θέση επιλέγεται, δεν μπορεί να επιλεχθεί ξανά από τον επόμενο επιβάτη στο ίδιο αεροπλάνο και στην ίδια πτήση. Οι θέσεις ξεκινάνε από την 1A και τελειώνουν στην 30F. Αφού έχουμε επιλέξει θέσεις για όλους τους επιβάτες πατάμε “Continue” για να μεταφερθούμε στην επόμενη σελίδα.

The image contains three screenshots of the "Seats" section during the booking process:

- Left Screenshot (Outbound Journey):** Shows the seat selection for Mrs. Garyfalia Georgitziki. The seat 1A is highlighted with a red circle. The dropdown menu shows "Seat" and "Athens → Madrid".
- Middle Screenshot (Outbound Journey):** Shows the seat selection for Mr. Vasilis Anagnostopoulos. The dropdown menu shows "Seat" and "Madrid → Lisbon".
- Right Screenshot (Inbound Journey):** Shows the seat selection for Mr. Vasilis Anagnostopoulos. The dropdown menu shows "Seat" and "Lisbon → Athens".

In all three screenshots, the total price is listed as 610.0 € and the "Continue" button is highlighted with a red circle.

Για τις λίστες των θέσεων τρέχουν τα παρακάτω POST APIs ώστε να βρεθούν οι κατειλημμένες θέσεις του αεροπλάνου της πτήσης και να αφαιρεθούν από τις συνολικές θέσεις.

Για την χωρητικότητα του αεροπλάνου της πτήσης:

```
//POST api that returns the airplane capacity according to the airplane model
app.post('/flynow/airplane-capacity', async (req, res) => {
  const jsonArray = req.body

  try {
    const result = await pool.query(`SELECT capacity FROM model WHERE modelid='${jsonArray[0].airplaneModel}'`)

    res.json(result.rows)
  } catch (error) {
    console.error(error)
    res.status(500).json({ error: 'Internal Server Error' })
  }
})
```

Για τις κατειλημμένες θέσεις του αεροπλάνου:

```
//POST api that returns the not available seats in a flight
app.post('/flynow/seats', async (req, res) => {
  const jsonArray = req.body

  try {
    const result = await pool
      .query(`SELECT DISTINCT seatnumber FROM has WHERE flightid='${jsonArray[0].flightId}'`)

    res.json(result.rows)
  } catch (error) {
    console.error(error)
    res.status(500).json({ error: 'Internal Server Error' })
  }
})
```

< Baggage&Pets

Baggage

All passengers are entitled to a free 8kg baggage in the aircraft cabin.

Outbound

Mrs Garyfalia Georgitziki

Baggage 1	23kg	32kg
	15€	25€

Add a new piece of baggage

Mr Vasilis Anagnostopoulos

Baggage 1	23kg	32kg
	15€	25€

Add a new piece of baggage

Total Price: 610.0 € **Continue**

Στην τελευταία σελίδα "Baggage&Pets", ο χρήστης επιλέγει τις βαλίτσες που επιθυμεί ο κάθε επιβάτης, με μέγιστο αριθμό τις 5 βαλίτσες ανά επιβάτη σε κάθε πτήση. Μπορεί να επιλέξει ανάμεσα σε 23kg και 32kg βάρος βαλίτσας, με ανάλογες τιμές που διαφέρουν ανάλογα την κλάση της πτήσης ("Economy", "Flex", "Business"). Από το κουμπί "+" ("Add a new piece of baggage"). Από το εικονίδιο της διαγραφής μπορούμε να διαγράψουμε μία επιπλέον βαλίτσα που προσθέσαμε. Η τιμή της κάθε βαλίτσας που έχει επιλεχθεί προστίθεται στο "Total Price".

< Baggage&Pets

Baggage

All passengers are entitled to a free 8kg baggage in the aircraft cabin.

Outbound

Mrs Garyfalia Georgitziki

Baggage 1	23kg	32kg
	15€	25€

Baggage 2	23kg	32kg
	15€	25€

Add a new piece of baggage

Mr Vasilis Anagnostopoulos

Baggage 1	23kg	32kg
	15€	25€

Total Price: 650.0 € **Continue**

< Baggage&Pets

Baggage

Mr Vasilis Anagnostopoulos

Baggage 1	23kg	32kg
	15€	25€

Baggage 2	23kg	32kg
	15€	25€

Baggage 3	23kg	32kg
	15€	25€

Add a new piece of baggage

Inbound

Mrs Garyfalia Georgitziki

Baggage 1	23kg	32kg
	Free	25€

Baggage 2	23kg	32kg
	15€	25€

Baggage 3	23kg	32kg
	15€	25€

Baggage 4	23kg	32kg
	15€	25€

Baggage 5	23kg	32kg
	15€	25€

Add a new piece of baggage

Mr Vasilis Anagnostopoulos

Baggage 1	23kg	32kg
	15€	25€

Total Price: 705.0 € **Continue**

< Baggage&Pets

Baggage

Mr Vasilis Anagnostopoulos

Baggage 1	23kg	32kg
	15€	25€

Total Price: 785.0 € **Continue**

Baggage&Pets

Mr Vasilis Anagnostopoulos

Baggage 1	23kg	32kg
	Free	25€
Baggage 2	23kg	32kg
	15€	25€

+ Add a new piece of baggage

Pets

Are you travelling with pet?

Yes No

Total Price: 835.0 €

Continue

Τέλος στο πεδίο “Pets” μπορούμε να προσθέσουμε εάν επιθυμούμε ένα κατοικίδιο στην κράτησή μας, όπου ανάλογα με το μέγεθός (βάρος) του υπάρχει και η ανάλογη χρέωση. Αφού έχουμε ολοκληρώσει τις επιλογές μας στο “Total Price” βλέπουμε την τελική τιμή της κράτησής μας και πατώντας το “Continue” μας εμφανίζει το παρακάτω μήνυμα επιβεβαίωσης της κράτησης. Εάν η κράτηση ολοκληρωθεί με επιτυχία, εμφανίζεται ο κωδικός κράτησης με τον οποίο ο/οι επιβάτης/ες μπορούν να αναζητήσουν την κράτησή τους. Πατώντας “OK” ο χρήστης μεταφέρεται στην σελίδα “Home”.

Baggage&Pets

Baggage 2	15€	25€	
-----------	-----	-----	--

+ Add a new piece of baggage

Pets

Are you travelling with pet?

Yes No

Pet Size

- Small (<8kg) - 35€
- Medium (<25kg) - 50€
- Large (>25kg) - 90€

Total Price: 885.0 €

Continue

Baggage&Pets

Baggage 2	15€	25€	
-----------	-----	-----	--

+ Add a new piece of baggage

Pets

Confirm Reservation?

Are you sure you want to book this flight?

No Yes

Pet Size

- Small (<8kg) - 35€
- Medium (<25kg) - 50€
- Large (>25kg) - 90€

Total Price: 885.0 €

Continue

Baggage&Pets

Baggage 2	15€	25€	
-----------	-----	-----	--

+ Add a new piece of baggage

Pets

Reservation Booked Successfully!

Your booking reference is **NADCVX**

OK

Pet Size

- Small (<8kg) - 35€
- Medium (<25kg) - 50€
- Large (>25kg) - 90€

Total Price: 885.0 €

Continue

ΠΡΟΣΟΧΗ! Για να αποτύχει η ολοκλήρωση κράτησης σημαίνει ότι ταυτόχρονα κάποιος άλλος χρήστης έχει προλάβει να κρατήσει την συγκεκριμένη θέση στην ίδια πτήση, οπότε ο χρήστης θα μεταφερθεί ξανά στην επιλογή νέων θέσεων.

Για την ολοκλήρωση της κράτησης εκτελείται το παρακάτω POST API:

```
//POST api to use the previous functions to insert the new booking in the database
app.post('/flynow/new-booking', async (req, res) => {
    const jsonArray = req.body
    let passengersId, reservationId

    try {
        const result = await checkIfSeatsAreOk(jsonArray[0].seats)
        if(result) {
            passengersId = await insertPassengers(jsonArray[0].passengers)
            reservationId = await insertReservation(jsonArray[0].petSize,jsonArray[0].price)
            await insertInHashTable(jsonArray[0].seats,jsonArray[0].baggage,jsonArray[0].passengers,
                passengersId,reservationId,jsonArray[0].classTypeOutbound,jsonArray[0].classTypeInbound
            )
            res.json([ { success: true, bookingId: reservationId } ])
        }
        else {
            res.json([ { success: false, bookingId: reservationId } ])
        }
    } catch (error) {
        console.error(error)
        res.status(500).json({ error: 'Internal Server Error' })
    }
})
```

Η συνάρτηση **generateReservationId()** δημιουργεί ένα μοναδικό reservationid, η οποία εκτελείται μέσα στην συνάρτηση **insertReservation()**:

```
//function that generates a random reservation id for the completion of the booking
function generateReservationId() {
    const characters = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ';
    let reservationId = '';

    for (let i = 0; i < 6; i++) {
        const randomIndex = Math.floor(Math.random() * characters.length);
        reservationId += characters.charAt(randomIndex);
    }

    return reservationId;
}
```

Η συνάρτηση **checkIfSeatsAreOk()** ελέγχει εάν οι θέσεις που επιλέχθηκαν για τους επιβάτες είναι ακόμα ελεύθερες:

```
//function that checks if selected seats of the passengers in the booking before the completion are all ok
//and no one's seat has been taken from someone else
async function checkIfSeatsAreOk(seats) {

    for(let i=0; i<seats.length; i++) {

        if(seats[i].outbound?.direct) {
            const result = await pool.query(`

                SELECT seatnumber
                FROM has
                WHERE flightid='${seats[i].outbound.direct.flightid_1}'
                AND seatnumber='${seats[i].outbound.direct.seat_1}'`)

            if(result.rows.length != 0) {
                return false
            }
        }

        else if(seats[i].outbound?.oneStop) {
            let result = await pool.query(`

                SELECT seatnumber
                FROM has
                WHERE flightid='${seats[i].outbound.oneStop.flightid_1}'
                AND seatnumber='${seats[i].outbound.oneStop.seat_1}'`)

            if(result.rows.length != 0) {
                return false
            }

            result = await pool.query(`

                SELECT seatnumber
                FROM has
                WHERE flightid='${seats[i].outbound.oneStop.flightid_2}'
                AND seatnumber='${seats[i].outbound.oneStop.seat_2}'`)

            if(result.rows.length != 0) {
                return false
            }
        }

        if(seats[i].inbound) {
            if(seats[i].inbound?.direct) {
                const result = await pool.query(`

                    SELECT seatnumber
                    FROM has
                    WHERE flightid='${seats[i].inbound.direct.flightid_1}'
                    AND seatnumber='${seats[i].inbound.direct.seat_1}'`)

                if(result.rows.length != 0) {
                    return false
                }
            }

            else if(seats[i].inbound?.oneStop) {
                let result = await pool.query(`

                    SELECT seatnumber
                    FROM has
                    WHERE flightid='${seats[i].inbound.oneStop.flightid_1}'
                    AND seatnumber='${seats[i].inbound.oneStop.seat_1}'`)

                if(result.rows.length != 0) {
                    return false
                }

                result = await pool.query(`

                    SELECT seatnumber
                    FROM has
                    WHERE flightid='${seats[i].inbound.oneStop.flightid_2}'
                    AND seatnumber='${seats[i].inbound.oneStop.seat_2}'`)

                if(result.rows.length != 0) {
                    return false
                }
            }
        }
    }

    return true
}
```

Η συνάρτηση ***insertPassengers()*** εισάγει τους επιβάτες στον πίνακα **passenger**:

```
//function that inserts the passengers of the booking in the database
async function insertPassengers(passengers) {
    let passengersId = []

    for(let i=0; i<passengers.length; i++) {
        try{
            await pool.query(`INSERT INTO passenger(firstname,lastname,email,birthdate,sex,phonenumber)
                VALUES('${passengers[i].firstname}','${passengers[i].lastname}','${passengers[i].email}',
                '${passengers[i].birthdate}','${passengers[i].gender}','${passengers[i].phonenumber}')`)
        }catch (error) {
            if (error.code === '23505' && error.constraint === 'passenger_email_key') {
                console.log('Passenger with this email already exists. Continuing with other queries.')
            }
            else {
                // Handle other errors
                console.error('Error inserting passenger:', error)
            }
        }

        const result = await pool.query(`SELECT passengerid FROM passenger WHERE email='${passengers[i].email}'`)
        passengersId.push({[passengers[i].email]: result.rows[0].passengerid})
    }
    return passengersId
}
```

Η συνάρτηση ***insertReservation()*** εισάγει την κράτηση στον πίνακα **reservation**:

```
//function that generates the reservation id with function generateReservationId()
//and checks if this reservation id exists already in the database and if exists
//makes again the reservation id and do again this process and finally insert the reservation to the database
async function insertReservation(petSize,price) {
    let reservationId

    do {
        reservationId = generateReservationId()
        const result = await pool.query(`SELECT COUNT(*) FROM reservation WHERE reservationid='${reservationId}'`)
        const count = result.rows[0].count

        if (count === '0') {
            break
        }
    } while (true)

    if(petSize != "") {
        await pool.query(`INSERT INTO reservation(reservationid, petsize, wifionboard, price)
            VALUES('${reservationId}', '${petSize}', ${0}, ${price})`)
    }
    else {
        await pool.query(`INSERT INTO reservation(reservationid, petsize, wifionboard, price)
            VALUES('${reservationId}', ${null}, ${0}, ${price})`)
    }

    return reservationId
}
```

Η συνάρτηση **insertInHasTable()** εισάγει τους επιβάτες, την κράτηση, τις θέσεις, τις βαλίτσες και τις πτήσεις στον πίνακα **has**:

```
//function that inserts data to has table that has info for all the stuff about booking
async function insertInHasTable(seats,baggage,passengers,passengerId,reservationId,classTypeOutbound,classTypeInbound) {
    for(let i=0; i<seats.length; i++) {
        const foundPassenger = passengers.find(passenger => Object.keys(passenger)[0] === passengers[i].email)
        const passengerId = foundPassenger[passenger[i].email]

        //outbound direct flight
        if(seats[i].outbound?.direct) {
            await pool.query(`INSERT INTO has(passengerid, flightid, reservationid, seatnumber, classtype, baggage, numofbaggagepercategory, checkin)
                VALUES(${passengerId}, '${seats[i].outbound.direct.flightid_1}', '${reservationId}',
                '${seats[i].outbound.direct.seat_1}', '${classTypeOutbound.toUpperCase()} CLASS', 'baggage23kg', ${baggage[i].outbound.baggage23kg}, ${false})`);
        }
        await pool.query(`INSERT INTO has(passengerid, flightid, reservationid, seatnumber, classtype, baggage, numofbaggagepercategory, checkin)
            VALUES(${passengerId}, '${seats[i].outbound.direct.flightid_1}', '${reservationId}',
            '${seats[i].outbound.direct.seat_1}', '${classTypeOutbound.toUpperCase()} CLASS', 'baggage32kg', ${baggage[i].outbound.baggage32kg}, ${false})`);

        //outbound one stop flight
        else if(seats[i].outbound?.oneStop) {
            await pool.query(`INSERT INTO has(passengerid, flightid, reservationid, seatnumber, classtype, baggage, numofbaggagepercategory, checkin)
                VALUES(${passengerId}, '${seats[i].outbound.oneStop.flightid_1}', '${reservationId}',
                '${seats[i].outbound.oneStop.seat_1}', '${classTypeOutbound.toUpperCase()} CLASS', 'baggage23kg', ${baggage[i].outbound.baggage23kg}, ${false})`);

            await pool.query(`INSERT INTO has(passengerid, flightid, reservationid, seatnumber, classtype, baggage, numofbaggagepercategory, checkin)
                VALUES(${passengerId}, '${seats[i].outbound.oneStop.flightid_1}', '${reservationId}',
                '${seats[i].outbound.oneStop.seat_1}', '${classTypeOutbound.toUpperCase()} CLASS', 'baggage32kg', ${baggage[i].outbound.baggage32kg}, ${false})`);

            await pool.query(`INSERT INTO has(passengerid, flightid, reservationid, seatnumber, classtype, baggage, numofbaggagepercategory, checkin)
                VALUES(${passengerId}, '${seats[i].outbound.oneStop.flightid_2}', '${reservationId}',
                '${seats[i].outbound.oneStop.seat_2}', '${classTypeOutbound.toUpperCase()} CLASS', 'baggage23kg', ${baggage[i].outbound.baggage23kg}, ${false})`);

            await pool.query(`INSERT INTO has(passengerid, flightid, reservationid, seatnumber, classtype, baggage, numofbaggagepercategory, checkin)
                VALUES(${passengerId}, '${seats[i].outbound.oneStop.flightid_2}', '${reservationId}',
                '${seats[i].outbound.oneStop.seat_2}', '${classTypeOutbound.toUpperCase()} CLASS', 'baggage32kg', ${baggage[i].outbound.baggage32kg}, ${false})`);

            //inbound direct flight
            if(seats[i].inbound) {
                await pool.query(`INSERT INTO has(passengerid, flightid, reservationid, seatnumber, classtype, baggage, numofbaggagepercategory, checkin)
                    VALUES(${passengerId}, '${seats[i].inbound.direct.flightid_1}', '${reservationId}',
                    '${seats[i].inbound.direct.seat_1}', '${classTypeInbound.toUpperCase()} CLASS', 'baggage23kg', ${baggage[i].inbound.baggage23kg}, ${false})`);

                await pool.query(`INSERT INTO has(passengerid, flightid, reservationid, seatnumber, classtype, baggage, numofbaggagepercategory, checkin)
                    VALUES(${passengerId}, '${seats[i].inbound.direct.flightid_1}', '${reservationId}',
                    '${seats[i].inbound.direct.seat_1}', '${classTypeInbound.toUpperCase()} CLASS', 'baggage32kg', ${baggage[i].inbound.baggage32kg}, ${false})`);

            }
            //inbound one stop flight
            else if(seats[i].inbound?.oneStop) {
                await pool.query(`INSERT INTO has(passengerid, flightid, reservationid, seatnumber, classtype, baggage, numofbaggagepercategory, checkin)
                    VALUES(${passengerId}, '${seats[i].inbound.oneStop.flightid_1}', '${reservationId}',
                    '${seats[i].inbound.oneStop.seat_1}', '${classTypeInbound.toUpperCase()} CLASS', 'baggage23kg', ${baggage[i].inbound.baggage23kg}, ${false})`);

                await pool.query(`INSERT INTO has(passengerid, flightid, reservationid, seatnumber, classtype, baggage, numofbaggagepercategory, checkin)
                    VALUES(${passengerId}, '${seats[i].inbound.oneStop.flightid_1}', '${reservationId}',
                    '${seats[i].inbound.oneStop.seat_1}', '${classTypeInbound.toUpperCase()} CLASS', 'baggage32kg', ${baggage[i].inbound.baggage32kg}, ${false})`);

                await pool.query(`INSERT INTO has(passengerid, flightid, reservationid, seatnumber, classtype, baggage, numofbaggagepercategory, checkin)
                    VALUES(${passengerId}, '${seats[i].inbound.oneStop.flightid_2}', '${reservationId}',
                    '${seats[i].inbound.oneStop.seat_2}', '${classTypeInbound.toUpperCase()} CLASS', 'baggage23kg', ${baggage[i].inbound.baggage23kg}, ${false})`);

                await pool.query(`INSERT INTO has(passengerid, flightid, reservationid, seatnumber, classtype, baggage, numofbaggagepercategory, checkin)
                    VALUES(${passengerId}, '${seats[i].inbound.oneStop.flightid_2}', '${reservationId}',
                    '${seats[i].inbound.oneStop.seat_2}', '${classTypeInbound.toUpperCase()} CLASS', 'baggage32kg', ${baggage[i].inbound.baggage32kg}, ${false})`);

            }
        }
    }
    return true
}
}
```

My Booking



Για να δει ο χρήστης την σελίδα που του δείχνει την κράτησή του, μπορεί να κάνει κλικ στο κουμπί "MyBooking" του μενού της αρχικής οθόνης ή να κάνει κλικ στο "More" και μετά στο κουμπί "Change your Booking". Ανοίγει η σελίδα "Find your Booking" και για την εύρεση της κράτησης, ο χρήστης αρκεί να πληκτρολογήσει τον κωδικό της κράτησής του και το επώνυμο του είτε σε κεφαλαία είτε σε μικρά γράμματα στα πεδία "Booking reference" και "Last name" αντίστοιχα. Κάνοντας κλικ στο κουμπί "Continue" η εφαρμογή τρέχει ένα query για να ελέγξει αν υπάρχει κράτηση με αυτά τα στοιχεία και αν υπάρχει τρέχει επιμέρους queries για να ανακτήσει τα στοιχεία της κράτησης από την βάση δεδομένων.

The screenshots illustrate the steps for finding a booking:

- The first screenshot shows the 'More' menu with the 'Change your Booking' option highlighted.
- The second screenshot shows the 'Find your Booking' screen. The 'Booking reference' field contains 'nadcvx' and the 'Last name' field contains 'georgitziki'. Both fields are circled in red.
- The third screenshot shows the results screen. The 'Continue' button is highlighted with a red circle. Below it, a keyboard is shown with the input 'georgitziki'.

Για τον έλεγχο εάν υπάρχει κράτηση με τα στοιχεία που πληκτρολόγησε ο χρήστης:

```
app.post('/flynow/check-booking', async (req, res) => {
  const jsonArray = req.body

  try {
    const result = await pool
      .query(`SELECT p.lastname, h.reservationid
              FROM passenger p, has h
              WHERE LOWER(p.lastname) ='${jsonArray[0].lastname.toLowerCase()}' 
                    AND h.reservationid = '${jsonArray[0].bookingid.toUpperCase()}' 
                    AND p.passengerid = h.passengerid`)
    if(result.rows[0] != null){
      res.json([{ success: true}])
    }
    else{
      res.json([{ success: false}])
    }
  } catch (error) {
    console.error(error)
    res.status(500).json({ error: 'Internal Server Error' })
  }
})
```

Τρέχει το POST API:

```
app.post('/flynow/booking-details', async (req, res) => {
```

Για να βρει πόσες πτήσεις περιέχει η κράτηση:

```
const numOfFlights = await pool
  .query(`SELECT COUNT(DISTINCT h.flightid)
          FROM has h
          WHERE h.reservationid = '${jsonArray[0].bookingid.toUpperCase()}'`)
```

Ανάλογα με τον αριθμό των πτήσεων τρέχει τα εξής queries:

- Για μία πτήση:

Για τα στοιχεία της πτήσης:

```
const direct_flight = await pool
.query(`SELECT DISTINCT TO_CHAR(f.flightdate, 'DD/MM/YYYY') AS flightdate, f.departuretime,
    f.arrivaltime, a1.city AS departurecity, a2.city AS arrivalcity, a1.name AS departureairport,
    a2.name AS arrivalairport, fd.duration, f.flightid, ap.model as airplanemodeL, h.classtype
    FROM flight f, has h, airport a1, airport a2, flight_duration fd, airplane ap
    WHERE f.flightid = h.flightid
        AND h.reservationid = '${jsonArray[0].bookingid.toUpperCase()}'  
        AND f.departureairport = a1.name
        AND f.arrivalairport = a2.name
        AND f.departuretime = fd.departuretime
        AND f.arrivaltime = fd.arrivaltime
        AND f.departuretime < f.arrivaltime
        AND ap.airplaneid=f.airplane`)
```

Για τις βαλίτσες και την θέση κάθε επιβάτη:

```
const result = await pool
.query(`SELECT h1.baggage AS baggage23kg, h2.baggage AS baggage32kg,
    h1.numofbaggagepercategory AS numofbaggage23kg,
    h2.numofbaggagepercategory AS numofbaggage32kg, h1.flightid , h1.reservationid,
    p.firstname, p.lastname, p.sex, h1.seatnumber, a1.city AS departurecity, a2.city AS arrivalcity,
    h1.checkin AS checkin
    FROM flight f, has h1, has h2, passenger p, airport a1, airport a2
    WHERE f.flightid = h1.flightid
        AND h1.reservationid = '${jsonArray[0].bookingid.toUpperCase()}'  
        AND h2.reservationid = '${jsonArray[0].bookingid.toUpperCase()}'  
        AND h1.baggage = 'baggage23kg'
        AND h2.baggage = 'baggage32kg'
        AND h1.passengerid = h2.passengerid
        AND h1.flightid = h2.flightid
        AND h1.passengerid = p.passengerid
        AND h2.passengerid = p.passengerid
        AND f.departureairport = a1.name
        AND f.arrivalairport = a2.name
        ORDER BY p.lastname ASC`)
```

- Για δύο πτήσεις:

Εάν είναι με αναχώρηση και επιστροφή απευθείας πτήσεις για τα στοιχεία τους:

```
const two_way_direct = await pool
.query(`SELECT DISTINCT TO_CHAR(f1.flightdate, 'DD/MM/YYYY') AS flightdate1, f1.departuretime AS departuretime1,
f1.arrivaltime AS arrivaltime1, a1.city AS departurecity1, a2.city AS arrivalcity1, a1.name AS departureairport1,
a2.name AS arrivalairport1, fd1.duration AS duration1, f1.flightid AS flightid1, ap1.model AS airplanemode1, h1.classtype AS classtype1,
TO_CHAR(f2.flightdate, 'DD/MM/YYYY') AS flightdate2, f2.departuretime AS departuretime2, f2.arrivaltime AS arrivaltime2,
a3.city AS departurecity2, a4.city AS arrivalcity2, a3.name AS departureairport2, a4.name AS arrivalairport2, fd2.duration AS duration2,
f2.flightid AS flightid2, ap2.model AS airplanemode2, h2.classtype AS classtype2
FROM
flight f1
JOIN has h1 ON f1.flightid = h1.flightid
JOIN airport a1 ON f1.departureairport = a1.name
JOIN airport a2 ON f1.arrivalairport = a2.name
JOIN flight_duration fd1 ON f1.departuretime = fd1.departuretime AND f1.arrivaltime = fd1.arrivaltime
JOIN airplane ap1 ON ap1.airplaneid = f1.airplane,
flight f2
JOIN has h2 ON f2.flightid = h2.flightid
JOIN airport a3 ON f2.departureairport = a3.name
JOIN airport a4 ON f2.arrivalairport = a4.name
JOIN flight_duration fd2 ON f2.departuretime = fd2.departuretime AND f2.arrivaltime = fd2.arrivaltime
JOIN airplane ap2 ON ap2.airplaneid = f2.airplane

WHERE h1.reservationid = '${jsonArray[0].bookingid.toUpperCase()}'  

AND h2.reservationid = '${jsonArray[0].bookingid.toUpperCase()}'  

AND f1.arrivalairport = f2.departureairport  

AND f1.flightdate <= f2.flightdate  

AND f2.arrivalairport = f1.departureairport`)
```

Για τις βαλίτσες και τις θέσεις κάθε επιβάτη:

```
const result = await pool
.query(`SELECT h1.baggage AS baggage23kg, h2.baggage AS baggage32kg, h1.numofbaggagepercategory AS numofbaggage23kg,
h2.numofbaggagepercategory AS numofbaggage32kg, h1.flightid , h1.reservationid, p.firstname, p.lastname, p.sex, h1.seatnumber,
a1.city AS departurecity, a2.city AS arrivalcity, h1.checkin AS checkin
FROM has h1, has h2, passenger p, flight f1, flight f2, airport a1, airport a2
WHERE h1.reservationid = '${jsonArray[0].bookingid.toUpperCase()}'  

AND h2.reservationid = '${jsonArray[0].bookingid.toUpperCase()}'  

AND h1.baggage = 'baggage23kg'  

AND h2.baggage = 'baggage32kg'  

AND h1.passengerid = h2.passengerid  

AND h1.flightid = h2.flightid  

AND h1.passengerid = p.passengerid  

AND h2.passengerid = p.passengerid  

AND f1.flightid = h1.flightid  

AND f2.flightid = h2.flightid  

AND f1.flightid = h2.flightid  

AND f2.flightid = h1.flightid  

AND f1.flightdate <= f2.flightdate  

AND f1.departureairport = a1.name  

AND f1.arrivalairport = a2.name
ORDER BY f1.flightdate, f2.flightdate, p.lastname ASC`)
```

Εάν οι πτήσεις είναι αναχώρηση με μία ενδιάμεση στάση για τα στοιχεία τους:

```
const one_stop_flight = await pool
.query(`SELECT DISTINCT TO_CHAR(f1.flighthdate, 'DD/MM/YYYY') AS flighthdate1, f1.departuretime AS departuretime1,
f1.arrivaltime AS arrivaltime1, a1.city AS departurecity1, a2.city AS arrivalcity1, a1.name AS departureairport1,
a2.name AS arrivalairport1, fd1.duration AS duration1, f1.flighthid AS flighthid1, ap1.model AS airplanemode1, h1.classtype AS classtype1,
TO_CHAR(f2.flighthdate, 'DD/MM/YYYY') AS flighthdate2, f2.departuretime AS departuretime2, f2.arrivaltime AS arrivaltime2,
a3.city AS departurecity2, a4.city AS arrivalcity2, a3.name AS departureairport2, a4.name AS arrivalairport2,
fd2.duration AS duration2, f2.flighthid AS flighthid2, ap2.model AS airplanemode2, h2.classtype AS classtype2
FROM
flight f1
JOIN has h1 ON f1.flighthid = h1.flighthid
JOIN airport a1 ON f1.departureairport = a1.name
JOIN airport a2 ON f1.arrivalairport = a2.name
JOIN flight_duration fd1 ON f1.departuretime = fd1.departuretime AND f1.arrivaltime = fd1.arrivaltime
JOIN airplane ap1 ON ap1.airplaneid = f1.airplane,
flight f2
JOIN has h2 ON f2.flighthid = h2.flighthid
JOIN airport a3 ON f2.departureairport = a3.name
JOIN airport a4 ON f2.arrivalairport = a4.name
JOIN flight_duration fd2 ON f2.departuretime = fd2.departuretime AND f2.arrivaltime = fd2.arrivaltime
JOIN airplane ap2 ON ap2.airplaneid = f2.airplane
WHERE h1.reservationid = '${jsonArray[0].bookingid.toUpperCase()}'  

AND h2.reservationid = '${jsonArray[0].bookingid.toUpperCase()}'  

AND f1.arrivalairport = f2.departureairport  

AND f2.flighthdate = f1.flighthdate  

AND f1.departureairport <> f2.arrivalairport`)
```

Για τις βαλίτσες και τις θέσεις κάθε επιβάτη:

```
const result = await pool
.query(`SELECT h1.baggage AS baggage23kg, h2.baggage AS baggage32kg, h1.numofbaggagepercategory AS numofbaggage23kg,
h2.numofbaggagepercategory AS numofbaggage32kg, h1.flighthid , h1.reservationid, p.firstname, p.lastname,
p.sex, h1.seatnumber, a1.city AS departurecity, a2.city AS arrivalcity, h1.checkin AS checkin
FROM has h1, has h2, passenger p, flight f1, flight f2, airport a1, airport a2
WHERE h1.reservationid = '${jsonArray[0].bookingid.toUpperCase()}'  

AND h2.reservationid = '${jsonArray[0].bookingid.toUpperCase()}'  

AND h1.baggage = 'baggage23kg'  

AND h2.baggage = 'baggage32kg'  

AND h1.passengerid = h2.passengerid  

AND h1.flighthid = h2.flighthid  

AND h1.passengerid = p.passengerid  

AND h2.passengerid = p.passengerid  

AND f1.flighthid = h1.flighthid  

AND f2.flighthid = h2.flighthid  

AND f1.flighthid = h1.flighthid  

AND f1.flighthdate = f2.flighthdate  

AND f1.departureairport = a1.name  

AND f1.arrivalairport = a2.name
ORDER BY f1.departuretime, p.lastname ASC`)
```

- Για τρεις πτήσεις:

Εάν οι πτήσεις είναι αναχώρηση με μία ενδιάμεση στάση και επιστροφή απευθείας για τα στοιχεία τους:

```
const one_stop_go_direct_return= await pool
.query(`SELECT DISTINCT TO_CHAR(f1.flighthdate, 'DD/MM/YYYY') AS flighthdate1, f1.departuretime AS departuretime1,
f1.arrivaltime AS arrivaltime1, a1.city AS departurecity1, a2.city AS arrivalcity1, a1.name AS departureairport1,
| a2.name AS arrivalairport1, fd1.duration AS duration1, f1.flighthid AS flightid1, ap1.model as airplanemode1, h1.classtype AS classtype1,
TO_CHAR(f2.flighthdate, 'DD/MM/YYYY') AS flighthdate2, f2.departuretime AS departuretime2, f2.arrivaltime AS arrivaltime2,
a3.city AS departurecity2, a4.city AS arrivalcity2, a3.name AS departureairport2, a4.name AS arrivalairport2,
fd2.duration AS duration2, f2.flighthid AS flightid2, ap2.model as airplanemode2, h2.classtype AS classtype2,
TO_CHAR(f3.flighthdate, 'DD/MM/YYYY') AS flighthdate3, f3.departuretime AS departuretime3, f3.arrivaltime AS arrivaltime3,
a5.city AS departurecity3, a6.city AS arrivalcity3, a5.name AS departureairport3, a6.name AS arrivalairport3,
fd3.duration AS duration3, f3.flighthid AS flightid3, ap3.model as airplanemode3, h3.classtype AS classtype3
FROM
flight f1
JOIN has h1 ON f1.flighthid = h1.flighthid
JOIN airport a1 ON f1.departureairport = a1.name
JOIN airport a2 ON f1.arrivalairport = a2.name
JOIN flight_duration fd1 ON f1.departuretime = fd1.departuretime AND f1.arrivaltime = fd1.arrivaltime
JOIN airplane ap1 ON ap1.airplaneid = f1.airplane,
flight f2
JOIN has h2 ON f2.flighthid = h2.flighthid
JOIN airport a3 ON f2.departureairport = a3.name
JOIN airport a4 ON f2.arrivalairport = a4.name
JOIN flight_duration fd2 ON f2.departuretime = fd2.departuretime AND f2.arrivaltime = fd2.arrivaltime
JOIN airplane ap2 ON ap2.airplaneid = f2.airplane,
flight f3
JOIN has h3 ON f3.flighthid = h3.flighthid
JOIN airport a5 ON f3.departureairport = a5.name
JOIN airport a6 ON f3.arrivalairport = a6.name
JOIN flight_duration fd3 ON f3.departuretime = fd3.departuretime AND f3.arrivaltime = fd3.arrivaltime
JOIN airplane ap3 ON ap3.airplaneid = f3.airplane
WHERE h1.reservationid = '${jsonArray[0].bookingid.toUpperCase()}'  

AND h2.reservationid = '${jsonArray[0].bookingid.toUpperCase()}'  

AND h3.reservationid = '${jsonArray[0].bookingid.toUpperCase()}'  

AND f1.arrivalairport = f2.departureairport  

AND f2.flighthdate = f1.flighthdate  

AND f2.flighthdate <= f3.flighthdate  

AND f1.departureairport <> f2.arrivalairport  

AND f2.arrivalairport = f3.departureairport  

AND f1.departureairport = f3.arrivalairport  

AND h1.classtype = h2.classtype`)
```

Για τις βαλίτσες και τις θέσεις κάθε επιβάτη:

```
const result = await pool
.query(`SELECT h1.baggage AS baggage23kg, h2.baggage AS baggage32kg, h1.numofbaggagepercategory AS numofbaggage23kg,
h2.numofbaggagepercategory AS numofbaggage32kg, h1.flighthid , h1.reservationid, p.firstname, p.lastname, p.sex,
h1.seatnumber, a1.city AS departurecity, a2.city AS arrivalcity, h1.checkin AS checkin
FROM has h1, has h2, passenger p, flight f1, flight f2, flight f3, airport a1, airport a2
WHERE h1.reservationid = '${jsonArray[0].bookingid.toUpperCase()}'  

AND h2.reservationid = '${jsonArray[0].bookingid.toUpperCase()}'  

AND h1.baggage = 'baggage23kg'  

AND h2.baggage = 'baggage32kg'  

AND h1.passengerid = h2.passengerid  

AND h1.flighthid = h2.flighthid  

AND h1.passengerid = p.passengerid  

AND h2.passengerid = p.passengerid  

AND f1.flighthid = h1.flighthid  

AND f2.flighthid = h2.flighthid  

AND f1.flighthid = h1.flighthid  

AND f3.flighthid = h1.flighthid  

AND f3.flighthid = h2.flighthid  

AND f2.flighthdate = f1.flighthdate  

AND f2.flighthdate <= f3.flighthdate  

AND f1.departureairport = a1.name  

AND f1.arrivalairport = a2.name
ORDER BY f1.flighthdate, f2.flighthdate, f3.flighthdate, f1.departuretime, f2.departuretime, p.lastname ASC`)
```

Εάν οι πτήσεις είναι αναχώρηση απευθείας και επιστροφή με μία ενδιάμεση στάση για τα στοιχεία τους:

```
const direct_go_one_stop_return = await pool
.query(`SELECT DISTINCT TO_CHAR(f1.flightdate, 'DD/MM/YYYY') AS flightdate1, f1.departuretime AS departuretime1,
    f1.arrivaltime AS arrivaltime1, a1.city AS departurecity1, a2.city AS arrivalcity1, a1.name AS departureairport1,
    a2.name AS arrivalairport1, fd1.duration AS duration1, f1.flightid AS flightid1, ap1.model as airplanemode1, h1.classtype AS classtype1,
    TO_CHAR(f2.flightdate, 'DD/MM/YYYY') AS flightdate2, f2.departuretime AS departuretime2, f2.arrivaltime AS arrivaltime2,
    a3.city AS departurecity2, a4.city AS arrivalcity2, a3.name AS departureairport2, a4.name AS arrivalairport2,
    fd2.duration AS duration2, f2.flightid AS flightid2, ap2.model as airplanemode2, h2.classtype AS classtype2,
    TO_CHAR(f3.flightdate, 'DD/MM/YYYY') AS flightdate3, f3.departuretime AS departuretime3, f3.arrivaltime AS arrivaltime3,
    a5.city AS departurecity3, a6.city AS arrivalcity3, a5.name AS departureairport3, a6.name AS arrivalairport3, fd3.duration AS duration3,
    f3.flightid AS flightid3, ap3.model as airplanemode3, h3.classtype AS classtype3
FROM
flight f1
JOIN has h1 ON f1.flightid = h1.flightid
JOIN airport a1 ON f1.departureairport = a1.name
JOIN airport a2 ON f1.arrivalairport = a2.name
JOIN flight_duration fd1 ON f1.departuretime = fd1.departuretime AND f1.arrivaltime = fd1.arrivaltime
JOIN airplane ap1 ON ap1.airplaneid = f1.airplane,
flight f2
JOIN has h2 ON f2.flightid = h2.flightid
JOIN airport a3 ON f2.departureairport = a3.name
JOIN airport a4 ON f2.arrivalairport = a4.name
JOIN flight_duration fd2 ON f2.departuretime = fd2.departuretime AND f2.arrivaltime = fd2.arrivaltime
JOIN airplane ap2 ON ap2.airplaneid = f2.airplane,
flight f3
JOIN has h3 ON f3.flightid = h3.flightid
JOIN airport a5 ON f3.departureairport = a5.name
JOIN airport a6 ON f3.arrivalairport = a6.name
JOIN flight_duration fd3 ON f3.departuretime = fd3.departuretime AND f3.arrivaltime = fd3.arrivaltime
JOIN airplane ap3 ON ap3.airplaneid = f3.airplane
WHERE h1.reservationid = '${jsonArray[0].bookingid.toUpperCase()}' AND h2.reservationid = '${jsonArray[0].bookingid.toUpperCase()}' AND h3.reservationid = '${jsonArray[0].bookingid.toUpperCase()}' AND f1.arrivalairport = f2.departureairport AND f1.flightdate <= f2.flightdate AND f2.arrivalairport = f3.departureairport AND f3.arrivalairport = f1.departureairport AND f2.flightdate = f3.flightdate`)
```

Για τις βαλίτσες και τις θέσεις κάθε επιβάτη:

```
const result = await pool
.query(`SELECT h1.baggage AS baggage23kg, h2.baggage AS baggage32kg, h1.numofbaggagepercategory AS numofbaggage23kg,
    h2.numofbaggagepercategory AS numofbaggage32kg, h1.flightid , h1.reservationid, p.firstname, p.lastname, p.sex,
    h1.seatnumber, a1.city AS departurecity, a2.city AS arrivalcity, h1.checkin AS checkin
FROM has h1, has h2, passenger p, flight f1, flight f2, flight f3, airport a1, airport a2
WHERE h1.reservationid = '${jsonArray[0].bookingid.toUpperCase()}' AND h2.reservationid = '${jsonArray[0].bookingid.toUpperCase()}' AND h1.baggage = 'baggage23kg' AND h2.baggage = 'baggage32kg' AND h1.passengerid = h2.passengerid AND h1.flightid = h2.flightid AND h1.passengerid = p.passengerid AND h2.passengerid = p.passengerid AND f1.flightid = h1.flightid AND f2.flightid = h2.flightid AND f1.flightid = h2.flightid AND f2.flightid = h1.flightid AND f3.flightid = h1.flightid AND f3.flightid = h2.flightid AND f1.flightdate <= f2.flightdate AND f2.flightdate = f3.flightdate AND f1.departureairport = a1.name AND f1.arrivalairport = a2.name
ORDER BY f1.flightdate, f2.flightdate, f3.flightdate , f1.departuretime, f2.departuretime, p.lastname ASC`)
```

- Για τέσσερις πτήσεις:

Για τα στοιχεία των πτήσεων που είναι αναχώρηση και επιστροφή με μία ενδιάμεση στάση η κάθε πτήση:

```
const one_stop_go_one_stop_return = await pool
  .query(`SELECT DISTINCT TO_CHAR(f1.flightdate, 'DD/MM/YYYY') AS flightdate1, f1.departuretime AS departuretime1,
  f1.arrivaltime AS arrivaltime1, a1.city AS departurecity1, a2.city AS arrivalcity1, a1.name AS departureairport1,
  a2.name AS arrivalairport1, fd1.duration AS duration1, f1.flightid AS flightid1, ap1.model as airplanemode1, h1.classtype AS classtype1,
  TO_CHAR(f2.flightdate, 'DD/MM/YYYY') AS flightdate2, f2.departuretime AS departuretime2, f2.arrivaltime AS arrivaltime2,
  a3.city AS departurecity2, a4.city AS arrivalcity2, a3.name AS departureairport2, a4.name AS arrivalairport2, fd2.duration AS duration2,
  f2.flightid AS flightid2, ap2.model as airplanemode2, h2.classtype AS classtype2,
  TO_CHAR(f3.flightdate, 'DD/MM/YYYY') AS flightdate3, f3.departuretime AS departuretime3, f3.arrivaltime AS arrivaltime3, a5.city AS departurecity3,
  a6.city AS arrivalcity3, a5.name AS departureairport3, a6.name AS arrivalairport3, fd3.duration AS duration3, f3.flightid AS flightid3,
  ap3.model as airplanemode3, h3.classtype AS classtype3,
  TO_CHAR(f4.flightdate, 'DD/MM/YYYY') AS flightdate4, f4.departuretime AS departuretime4, f4.arrivaltime AS arrivaltime4, a7.city AS departurecity4,
  a8.city AS arrivalcity4, a7.name AS departureairport4, a8.name AS arrivalairport4, fd4.duration AS duration4, f4.flightid AS flightid4,
  ap4.model as airplanemode4, h4.classtype AS classtype4
  FROM
  flight f1
  JOIN has h1 ON f1.flightid = h1.flightid
  JOIN airport a1 ON f1.departureairport = a1.name
  JOIN airport a2 ON f1.arrivalairport = a2.name
  JOIN flight_duration fd1 ON f1.departuretime = fd1.departuretime AND f1.arrivaltime = fd1.arrivaltime
  JOIN airplane ap1 ON ap1.airplaneid = f1.airplane,
  flight f2
  JOIN has h2 ON f2.flightid = h2.flightid
  JOIN airport a3 ON f2.departureairport = a3.name
  JOIN airport a4 ON f2.arrivalairport = a4.name
  JOIN flight_duration fd2 ON f2.departuretime = fd2.departuretime AND f2.arrivaltime = fd2.arrivaltime
  JOIN airplane ap2 ON ap2.airplaneid = f2.airplane,
  flight f3
  JOIN has h3 ON f3.flightid = h3.flightid
  JOIN airport a5 ON f3.departureairport = a5.name
  JOIN airport a6 ON f3.arrivalairport = a6.name
  JOIN flight_duration fd3 ON f3.departuretime = fd3.departuretime AND f3.arrivaltime = fd3.arrivaltime
  JOIN airplane ap3 ON ap3.airplaneid = f3.airplane,
  flight f4
  JOIN has h4 ON f4.flightid = h4.flightid
  JOIN airport a7 ON f4.departureairport = a7.name
  JOIN airport a8 ON f4.arrivalairport = a8.name
  JOIN flight_duration fd4 ON f4.departuretime = fd4.departuretime AND f4.arrivaltime = fd4.arrivaltime
  JOIN airplane ap4 ON ap4.airplaneid = f4.airplane
  WHERE h1.reservationid = '${jsonArray[0].bookingid.toUpperCase()}'  

    AND h2.reservationid = '${jsonArray[0].bookingid.toUpperCase()}'  

    AND h3.reservationid = '${jsonArray[0].bookingid.toUpperCase()}'  

    AND h4.reservationid = '${jsonArray[0].bookingid.toUpperCase()}'  

    AND f1.arrivalairport = f2.departureairport  

    AND f2.arrivalairport = f3.departureairport  

    AND f1.flightdate = f2.flightdate  

    AND f1.flightdate <= f3.flightdate  

    AND f3.arrivalairport = f4.departureairport  

    AND f4.arrivalairport = f1.departureairport  

    AND f3.flightdate = f4.flightdate`)
  
```

Για τις βαλίτσες και τις θέσεις κάθε επιβάτη:

```
const result = await pool
  .query(`SELECT h1.baggage AS baggage23kg, h2.baggage AS baggage32kg, h1.numofbaggagepercategory AS numofbaggage23kg,
    h2.numofbaggagepercategory AS numofbaggage32kg, h1.flightid , h1.reservationid, p.firstname, p.lastname, p.sex,
    h1.seatnumber, a1.city AS departurecity, a2.city AS arrivalcity, h1.checkin AS checkin
  FROM has h1, has h2, passenger p, flight f1, flight f2, flight f3, flight f4, airport a1, airport a2
  WHERE h1.reservationid = '${jsonArray[0].bookingid.toUpperCase()}'  

    AND h2.reservationid = '${jsonArray[0].bookingid.toUpperCase()}'  

    AND h1.baggage = 'baggage23kg'  

    AND h2.baggage = 'baggage32kg'  

    AND h1.passengerid = h2.passengerid  

    AND h1.flightid = h2.flightid  

    AND h1.passengerid = p.passengerid  

    AND h2.passengerid = p.passengerid  

    AND f1.flightid = h1.flightid  

    AND f2.flightid = h2.flightid  

    AND f1.flightid = h2.flightid  

    AND f2.flightid = h1.flightid  

    AND f3.flightid = h1.flightid  

    AND f3.flightid = h2.flightid  

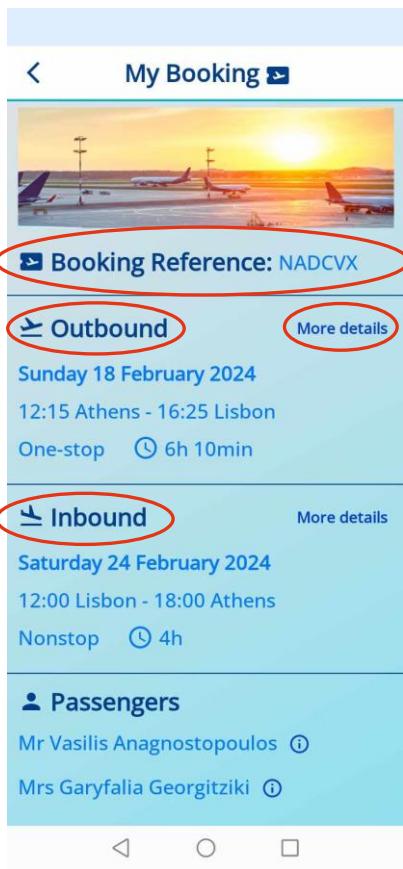
    AND f4.flightid = h1.flightid  

    AND f4.flightid = h2.flightid  

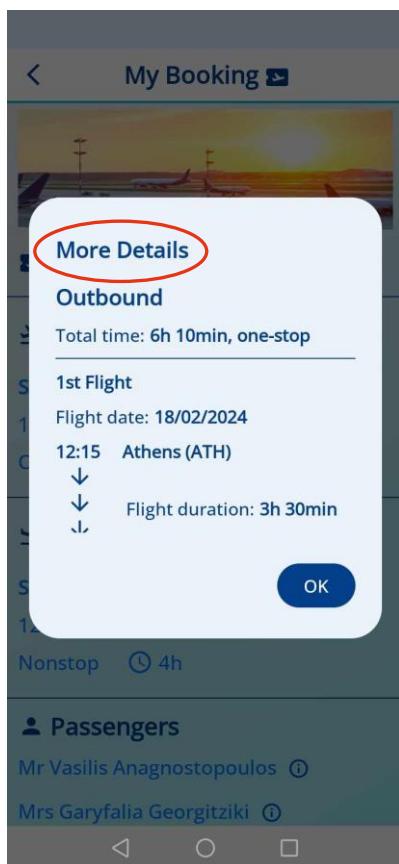
    AND f1.flightdate <= f2.flightdate  

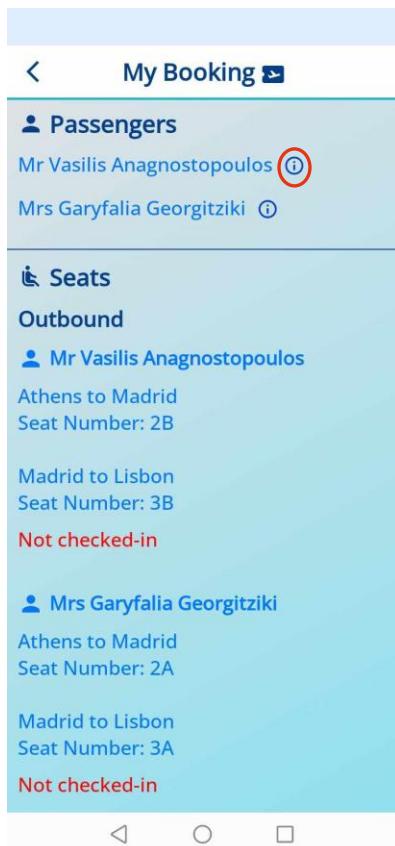
    AND f1.departureairport = a1.name  

    AND f1.arrivalairport = a2.name
  ORDER BY f1.flightdate, f2.flightdate, f3.flightdate, f4.flightdate, f1.departuretime, f2.departuretime, p.lastname ASC`)
```

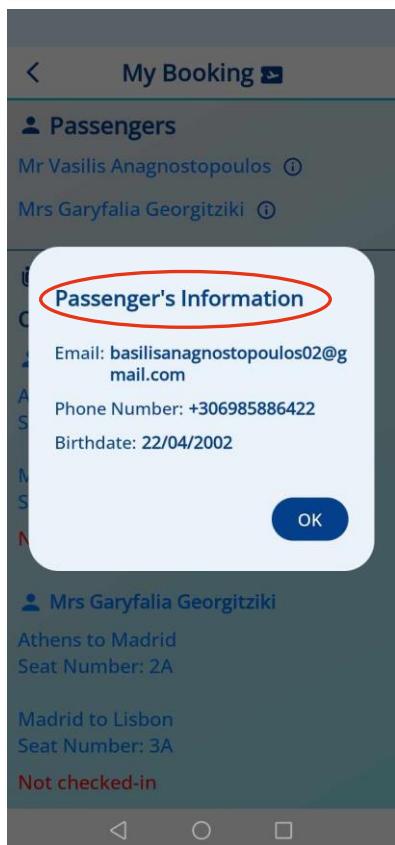


Εφόσον τα στοιχεία που πληκτρολόγησε ο χρήστης ήταν σωστά και η αναζήτηση ήταν επιτυχής, ανοίγει η σελίδα "My Booking" με όλα τα στοιχεία της κράτησης. Αρχικά, στο πάνω μέρος της σελίδας εμφανίζεται ο κωδικός της κράτησης στο πεδίο "Booking Reference". Έπειτα, βλέπουμε την πτήση αναχώρησης στο πεδίο "Outbound" και κάποια στοιχεία της, ενώ για επιπλέον πληροφορίες κάνοντας κλικ στο "More details" εμφανίζονται περισσότερες πληροφορίες για την/τις πτήση/εις. Κατά τον ίδιο τρόπο ακολουθεί η πτήση επιστροφής, εάν υπάρχει.





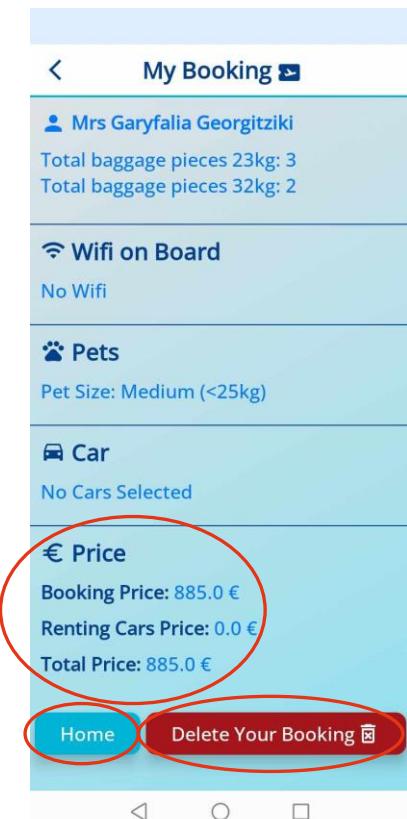
Στην συνέχεια, στο πεδίο “Passengers” βλέπουμε τους επιβάτες της κράτησης και κάνοντας κλικ στο εικονίδιο “Info” μπορούμε να δούμε επιπλέον πληροφορίες για τον κάθε επιβάτη.



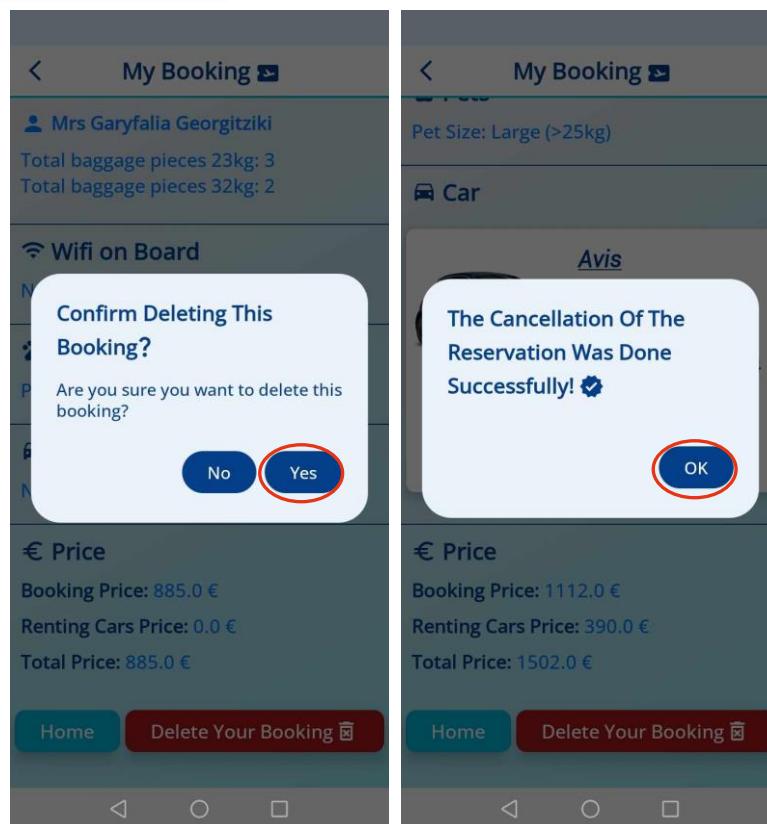


Παρακάτω, στο πεδίο "Seats" βλέπουμε την θέση που έχει επιλέξει να έχει ο κάθε επιβάτης σε κάθε πτήση, και εάν έχει γίνει check-in για την αναχώρηση συνολικά, και για την επιστροφή αν υπάρχει. Έπειτα ακολουθεί ο συνολικός αριθμόν βαλιτσών 23 και 32 κιλών που έχει επιλέξει κάθε επιβάτης για την αναχώρηση και την επιστροφή. Ακολουθεί το πεδίο "Wifi on Board" όπου εάν ο χρήστης έχει επιλέξει για την κράτηση κάποιον τύπο ίντερνετ από το "More" θα εμφανιστεί εδώ. Στο πεδίο "Pets" εμφανίζεται ο τύπος κατοικίδιου που έχει επιλεχθεί για την κράτηση αν υπάρχει, και τέλος εμφανίζεται στο πεδίο "Cars" το/τα αυτοκίνητο/α της κράτησης που προστέθηκαν από το "More" (εάν υπάρχουν).

Inbound	Outbound
Mr Vasilis Anagnostopoulos Lisbon to Athens Seat Number: 12F Not checked-in	Mr Vasilis Anagnostopoulos Total baggage pieces 23kg: 0 Total baggage pieces 32kg: 2
Mrs Garyfalia Georgitziki Lisbon to Athens Seat Number: 12E Not checked-in	Mrs Garyfalia Georgitziki Total baggage pieces 23kg: 3 Total baggage pieces 32kg: 2
Baggage	Wifi on Board
Outbound	No Wifi
Mr Vasilis Anagnostopoulos Total baggage pieces 23kg: 2 Total baggage pieces 32kg: 1	Pets
Mrs Garyfalia Georgitziki Total baggage pieces 23kg: 1 Total baggage pieces 32kg: 1	Pet Size: Medium (<25kg)
Car	Car
	No Cars Selected
Price	Price
	Booking Price: 885.0 €
	Renting Cars Price: 0.0 €



Τέλος, υπάρχει το πεδίο “Price”, στο οποίο φαίνονται αναλυτικά οι τιμές για την κράτηση των πτήσεων, την κράτηση των αυτοκινήτων και η συνολική τιμή όλων. Στο κάτω μέρος της σελίδας, υπάρχουν τα κουμπιά “Home” και “Delete Your Booking”, όπου με το κουμπί “Home” ο χρήστης πλοηγείται στην αρχική σελίδα, ενώ με το κουμπί “Delete Your Booking” ο χρήστης μπορεί να ακυρώσει την κράτησή του αφού πατήσει “Yes” στο σχετικό μήνυμα. Έπειτα εμφανίζεται μήνυμα ότι η κράτηση διαγράφηκε επιτυχώς και πατώντας το “OK” μεταφέρεται στην αρχική σελίδα(Home Screen).



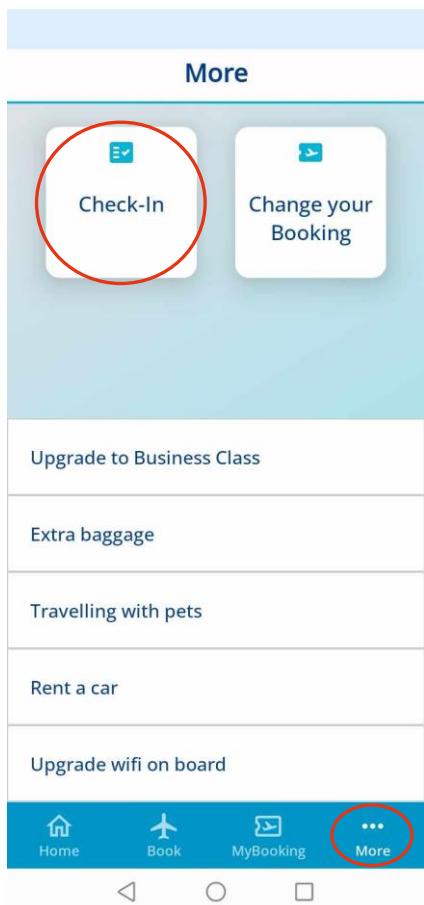
Για την διαγραφή της κράτησης εκτελείται το παρακάτω POST API:

```
//POST api to delete the booking of the user from the database
app.post('/flynow/delete-booking', async (req, res) => {
    const jsonArray = req.body

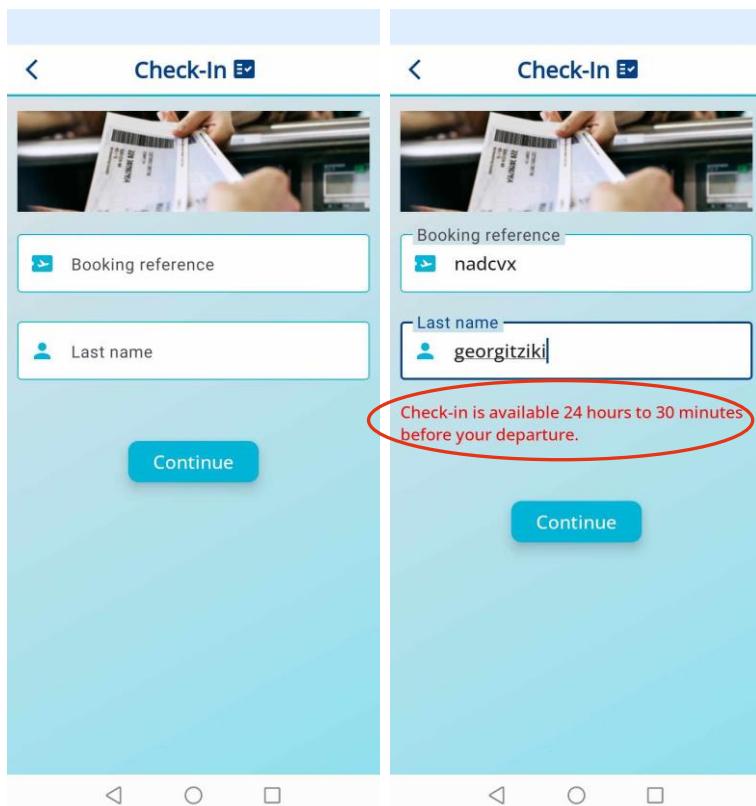
    try {
        await pool.query(`DELETE FROM has WHERE reservationid='${jsonArray[0].bookingId.toUpperCase()}'`)
        await pool.query(`DELETE FROM contains WHERE reservationid='${jsonArray[0].bookingId.toUpperCase()}'`)
        await pool.query(`DELETE FROM reservation WHERE reservationid='${jsonArray[0].bookingId.toUpperCase()}'`)

        res.json([{} success: true])
    } catch (error) {
        console.error(error)
        res.status(500).json({ error: 'Internal Server Error' })
    }
})
```

CHECK-IN



Ο χρήστης έχει την δυνατότητα μέσω της εφαρμογής να κάνει check-in την πτήση που επιθυμεί. Για να το κάνει αυτό αρκεί να κάνει ένα κλικ στο κουμπί "Check-In" και να μεταφερθεί στην σελίδα "Check-In", όπου πάλι θα του ζητηθεί να πληκτρολογήσει τον κωδικό της κράτησής του και το επώνυμο του είτε σε κεφαλαία είτε σε μικρά γράμματα στα πεδία "Booking reference" και "Last name" αντίστοιχα. Κάνοντας κλικ στο κουμπί "Continue" η εφαρμογή τρέχει το query που αναλύσαμε παραπάνω για να ελέγξει αν υπάρχει κράτηση με αυτά τα στοιχεία και αν υπάρχει τρέχει επιμέρους queries για να ανακτήσει τα στοιχεία της κράτησης από την βάση δεδομένων.

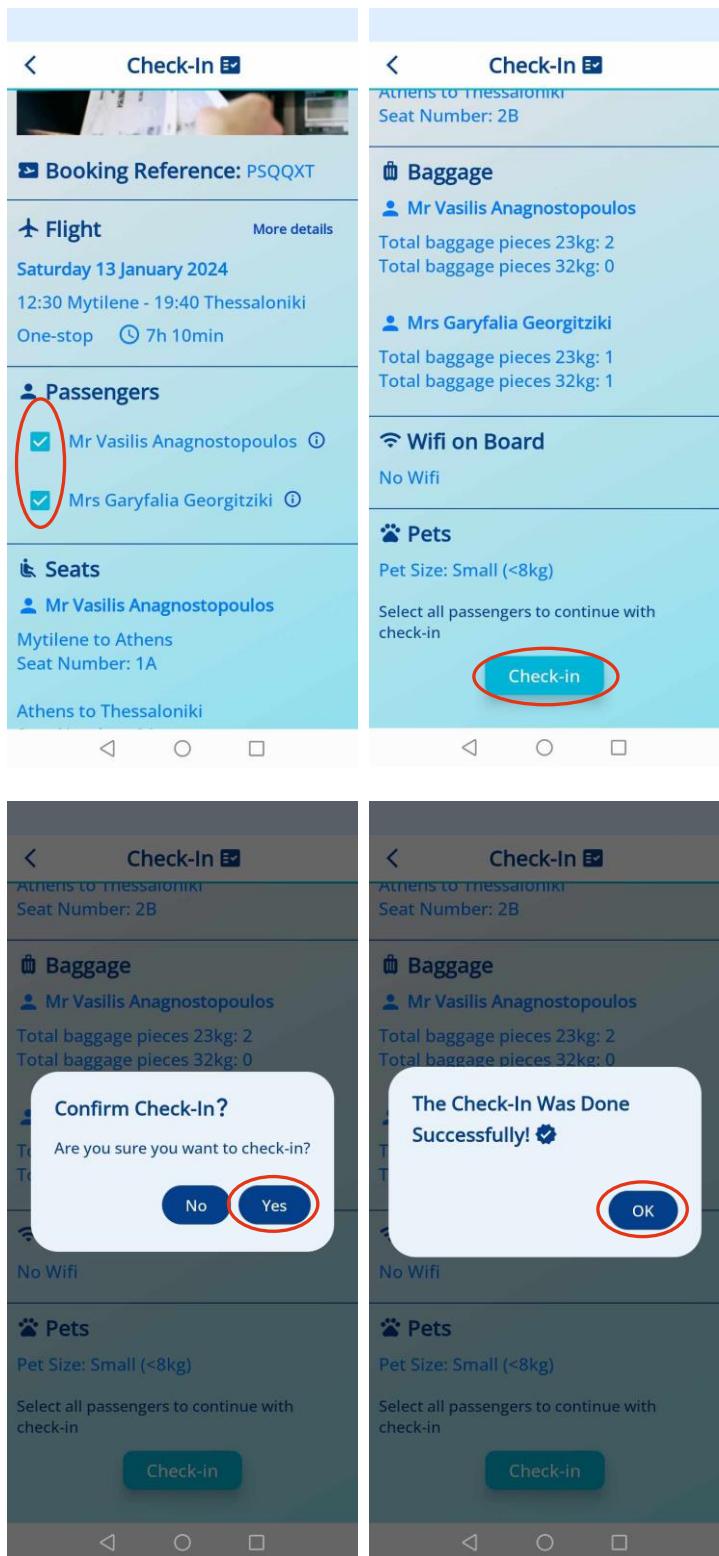


To check-in ανοίγει πάντα 24 ώρες πριν την πτήση και κλείνει μίση ώρα πριν την αναχώρησή της. Στο παράδειγμα αυτό, η πτήση αναχωρεί στις 18/02/2024 και ώρα 12:15, οπότε το check-in δεν έχει ακόμα ανοίξει γι αυτό και εμφανίζεται το ανάλογο μήνυμα.

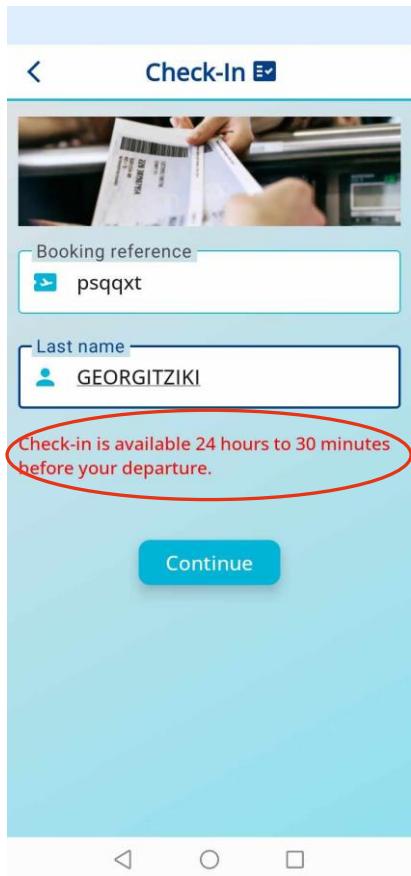
Σε ένα δεύτερο παράδειγμα το check-in έχει ανοίξει καθώς η πτήση απογειώνεται μέσα στο επόμενο 24ωρο, όπως φαίνεται παρακάτω:

The image displays three sequential screenshots of a mobile application for flight check-in. The first screenshot shows the main check-in screen with a booking reference (PSQQXT), flight details (Flight from Mytilene to Athens on Saturday, 13 January 2024, duration 7h 10min), and passenger selection. Two passengers are listed: Mr Vasilis Anagnostopoulos and Mrs Garyfalia Georgitziki. The second screenshot shows the 'Seats' section where both passengers are assigned seats: Mr Vasilis Anagnostopoulos is seated at 1A and Mrs Garyfalia Georgitziki is seated at 1B. The third screenshot shows the final step of the check-in process, where the 'Check-in' button is highlighted with a red oval, indicating the user needs to confirm the selection.

Τσεκάροντας όλους τους επιβάτες της πτήσης το κουμπί "Check-in" γίνεται enabled και ο χρήστης μπορεί να ολοκληρώσει το Check-in.



Πατώντας το κουμπί “Check-in” εμφανίζεται το dialog για την επιβεβαίωση του check-in και εφόσον ο χρήστης πατήσει “Yes” εμφανίζεται το μήνυμα ότι το check-in έγινε με επιτυχία. Κάνοντας κλικ στο κουμπί “OK” ο χρήστης μεταφέρεται στην αρχική σελίδα. Σε περίπτωση που ο χρήστης προσπαθήσει να ξανακάνει check-in την ίδια πτήση θα εμφανιστεί το παρακάτω μήνυμα λάθους, καθώς πλέον η πτήση που απομένει για check-in είναι της επιστροφής και δεν έχει ανοίξει ακόμα.



Με το πάτημα του “Continue”, το POST API και τα queries που τρέχουν και στις δύο περιπτώσεις για την ανάκτηση των πτήσεων και άλλων πληροφοριών που χρειάζονται στο check-in (εάν είναι διαθέσιμο) είναι:

To POST API που καλείται:

```
app.post('/flynow/checkin-details', async (req, res) => {
```

```
//find flights for check in
const flight1 = await pool
.query('WITH CurrentTimeFormatted AS (
    SELECT
        TO_CHAR(CURRENT_TIMESTAMP, 'HH24:MI:SS') || '+' ||
        LPAD(EXTRACT(TIMEZONE_HOUR FROM CURRENT_TIMESTAMP)::TEXT, 2, '0') || ':' ||
        LPAD(EXTRACT(TIMEZONE_MINUTE FROM CURRENT_TIMESTAMP)::TEXT, 2, '0') AS formatted_current_time
)
SELECT DISTINCT f.flightid, f.departuretime, f.flighthdate, f.arrivaltime
FROM
    flight f
JOIN has h ON h.flightid = f.flightid
WHERE
    h.reservationid = '${jsonArray[0].bookingid.toUpperCase()}' 
    AND f.flighthdate = CURRENT_DATE + INTERVAL '1 day'
    AND f.departuretime <= (CAST((SELECT formatted_current_time FROM CurrentTimeFormatted) AS TIME WITH TIME ZONE) + INTERVAL '24 hours' - INTERVAL '1 second')
    AND h.checkin = 'false'
    OR (f.flighthdate = CURRENT_DATE
        AND f.departuretime >= (CAST((SELECT formatted_current_time FROM CurrentTimeFormatted) AS TIME WITH TIME ZONE)+INTERVAL '30 minutes')
        AND h.checkin = 'false')
GROUP BY f.flightid, f.flighthdate, f.arrivaltime
ORDER BY f.departuretime`)
```

```

const flight2 = await pool
  .query(`SELECT DISTINCT f2.flightid, f2.departuretime, f2.flighthdate, f2.arrivaltime
  FROM
    flight f1
  JOIN has h1 ON h1.flightid = f1.flightid,
    flight f2
  JOIN has h2 ON h2.flightid = f2.flightid
WHERE f1.flightid = '${flight1.rows[0].flightid}'
  AND h1.reservationid = '${jsonArray[0].bookingid.toUpperCase()}''
  AND h2.reservationid = '${jsonArray[0].bookingid.toUpperCase()}''
  AND f2.flighthdate = f1.flighthdate
  AND f2.departureairport = f1.arrivalairport
  AND f2.departuretime > f1.arrivaltime
GROUP BY f2.flightid, f2.flighthdate, f2.arrivaltime
ORDER BY f2.departuretime`)

```

```

//one way direct flight
const direct_flight = await pool
  .query(`SELECT DISTINCT TO_CHAR(f.flighthdate, 'DD/MM/YYYY') AS flighthdate, f.departuretime,
  f.arrivaltime, a1.city AS departurecity, a2.city AS arrivalcity, a1.name AS departureairport,
  a2.name AS arrivalairport, fd.duration, f.flightid, ap.model as airplanemodel, h.classtype
  FROM flight f, has h, airport a1, airport a2, flight_duration fd, airplane ap
  WHERE f.flightid = '${numOfFlightsToCheckin[0].flightid}'
    AND h.reservationid = '${jsonArray[0].bookingid.toUpperCase()}''
    AND f.flightid = h.flightid
    AND f.departureairport = a1.name
    AND f.arrivalairport = a2.name
    AND f.departuretime = fd.departuretime
    AND f.arrivaltime = fd.arrivaltime
    AND f.departuretime < f.arrivaltime
    AND ap.airplaneid=f.airplane`)

```

```

//returns the baggage per passenger
const result = await pool
  .query(`SELECT h1.baggage AS baggage23kg, h2.baggage AS baggage32kg,
  h1.numofbaggagepercategory AS numofbaggage23kg, h2.numofbaggagepercategory AS numofbaggage32kg,
  h1.flighthid , h1.reservationid, p.firstname, p.lastname, p.sex, h1.seatnumber,
  a1.city AS departurecity, a2.city AS arrivalcity, h1.checkin
  FROM flight f, has h1, has h2, passenger p, airport a1, airport a2
  WHERE h1.flighthid = '${numOfFlightsToCheckin[0].flightid}'
    AND h2.flighthid = '${numOfFlightsToCheckin[0].flightid}'
    AND f.flighthid = h1.flighthid
    AND h1.reservationid = '${jsonArray[0].bookingid.toUpperCase()}''
    AND h2.reservationid = '${jsonArray[0].bookingid.toUpperCase()}''
    AND h1.baggage = 'baggage23kg'
    AND h2.baggage = 'baggage32kg'
    AND h1.passengerid = h2.passengerid
    AND h1.flighthid = h2.flighthid
    AND h1.passengerid = p.passengerid
    AND h2.passengerid = p.passengerid
    AND f.departureairport = a1.name
    AND f.arrivalairport = a2.name
  ORDER BY p.lastname ASC`)

```

```

//returns oneway one-stop flight
const one_stop_flight = await pool
.query(`SELECT DISTINCT TO_CHAR(f1.flightdate, 'DD/MM/YYYY') AS flightdate1,
f1.departuretime AS departuretime1, f1.arrivaltime AS arrivaltime1,
a1.city AS departurecity1, a2.city AS arrivalcity1, a1.name AS departureairport1,
a2.name AS arrivalairport1, fd1.duration AS duration1, f1.flightid AS flightid1,
ap1.model AS airplanemode1, h1.classtype AS classtype1,
TO_CHAR(f2.flightdate, 'DD/MM/YYYY') AS flightdate2, f2.departuretime AS departuretime2,
f2.arrivaltime AS arrivaltime2, a3.city AS departurecity2, a4.city AS arrivalcity2,
a3.name AS departureairport2, a4.name AS arrivalairport2, fd2.duration AS duration2,
f2.flightid AS flightid2, ap2.model AS airplanemode2, h2.classtype AS classtype2
FROM
    flight f1
    JOIN has h1 ON f1.flightid = h1.flightid
    JOIN airport a1 ON f1.departureairport = a1.name
    JOIN airport a2 ON f1.arrivalairport = a2.name
    JOIN flight_duration fd1 ON f1.departuretime = fd1.departuretime AND f1.arrivaltime = fd1.arrivaltime
    JOIN airplane ap1 ON ap1.airplaneid = f1.airplane,
    flight f2
    JOIN has h2 ON f2.flightid = h2.flightid
    JOIN airport a3 ON f2.departureairport = a3.name
    JOIN airport a4 ON f2.arrivalairport = a4.name
    JOIN flight_duration fd2 ON f2.departuretime = fd2.departuretime AND f2.arrivaltime = fd2.arrivaltime
    JOIN airplane ap2 ON ap2.airplaneid = f2.airplane

WHERE f1.flightid = '${numOfFlightsToCheckin[0].flightid}'
    AND f2.flightid = '${numOfFlightsToCheckin[1].flightid}'
    AND h1.reservationid = '${jsonArray[0].bookingid.toUpperCase()}'
    AND h2.reservationid = '${jsonArray[0].bookingid.toUpperCase()}'
    AND f1.arrivalairport = f2.departureairport
    AND f2.flightdate = f1.flightdate
    AND f1.departureairport <> f2.arrivalairport`)

//returns the baggage per passenger
const result = await pool
.query(`SELECT DISTINCT h1.baggage AS baggage23kg1, h2.baggage AS baggage32kg1,
h1.numofbaggagepercategory AS numofbaggage23kg1, h2.numofbaggagepercategory AS numofbaggage32kg1,
h1.flightid AS flightid1, h1.reservationid AS reservationid1, p.firstname AS firstname1,
p.lastname AS lastname1, p.sex AS sex1, h1.seatnumber AS seatnumber1, a1.city AS departurecity1,
a2.city AS arrivalcity1, h1.checkin AS checkin1,
h1.baggage AS baggage23kg2, h2.baggage AS baggage32kg2, h1.numofbaggagepercategory AS numofbaggage23kg2,
h2.numofbaggagepercategory AS numofbaggage32kg2, h2.flightid AS flightid2, h2.reservationid AS reservationid2,
p.firstname AS firstname2, p.lastname AS lastname2, p.sex AS sex2, h2.seatnumber AS seatnumber2, a3.city AS departurecity2,
a4.city AS arrivalcity2, h2.checkin AS checkin2, f1.departuretime, f2.departuretime
FROM passenger p,
flight f1
JOIN has h1 ON f1.flightid = h1.flightid
JOIN airport a1 ON f1.departureairport = a1.name
JOIN airport a2 ON f1.arrivalairport = a2.name
JOIN flight_duration fd1 ON f1.departuretime = fd1.departuretime AND f1.arrivaltime = fd1.arrivaltime
JOIN airplane ap1 ON ap1.airplaneid = f1.airplane,
flight f2
JOIN has h2 ON f2.flightid = h2.flightid
JOIN airport a3 ON f2.departureairport = a3.name
JOIN airport a4 ON f2.arrivalairport = a4.name
JOIN flight_duration fd2 ON f2.departuretime = fd2.departuretime AND f2.arrivaltime = fd2.arrivaltime
JOIN airplane ap2 ON ap2.airplaneid = f2.airplane

WHERE f1.flightid = '${numOfFlightsToCheckin[0].flightid}'
    AND f2.flightid = '${numOfFlightsToCheckin[1].flightid}'
    AND h1.reservationid = '${jsonArray[0].bookingid.toUpperCase()}'
    AND h2.reservationid = '${jsonArray[0].bookingid.toUpperCase()}'
    AND h1.baggage = 'baggage23kg'
    AND h2.baggage = 'baggage32kg'
    AND h1.passengerid = h2.passengerid
    AND h1.passengerid = p.passengerid
    AND h2.passengerid = p.passengerid
    AND f1.arrivalairport = f2.departureairport
    AND f2.flightdate = f1.flightdate
    AND f1.departureairport <> f2.arrivalairport
ORDER BY f1.departuretime, f2.departuretime, p.lastname ASC`)

```

```

//returns the passengers info
const result = await pool
.query(`SELECT DISTINCT COUNT(DISTINCT p.passengerid), p.sex, p.lastname, p.firstname,
TO_CHAR(p.birthdate, 'DD/MM/YYYY') AS birthdate, p.email, p.phonenumber, h.reservationid
FROM passenger p, has h
WHERE p.passengerid = h.passengerid AND h.reservationid = '${jsonArray[0].bookingid.toUpperCase()}' 
GROUP BY p.sex, p.lastname, p.firstname, h.reservationid, p.birthdate, p.email, p.phonenumber
ORDER BY p.lastname ASC`)

//returns the petsize if exists
const result = await pool
.query(`SELECT r.petsize
FROM reservation r
WHERE r.reservationid = '${jsonArray[0].bookingid.toUpperCase()}'`)

//returns the wifi type
const result = await pool
.query(`SELECT r.wifionboard
FROM reservation r
WHERE r.reservationid = '${jsonArray[0].bookingid.toUpperCase()}'`)

```

Κάνοντας κλικ στο κουμπί “Yes” εκτελείται το παρακάτω POST API και τα queries:

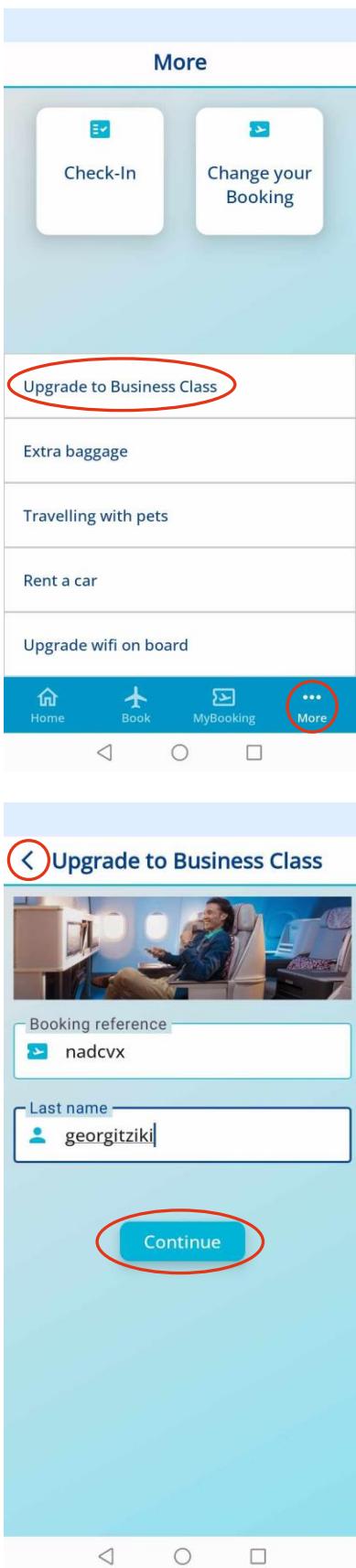
```

//POST api that update the check in info and make the flight for all passengers in the reservation checked in
//otherwise is not checked in
app.post('/flynow/update-checkin', async (req, res) => {
  const jsonArray = req.body

  try {
    if(jsonArray[0].numofflights === 1) {
      await pool.query(`UPDATE has
      SET checkin = 'true'
      WHERE reservationid='${jsonArray[0].bookingid.toUpperCase()}' 
      AND flightid = '${jsonArray[0].flightid1.toUpperCase()}'`)
      res.json([{ success: true}])
    }
    else if(jsonArray[0].numofflights === 2) {
      await pool.query(`UPDATE has
      SET checkin = 'true'
      WHERE reservationid='${jsonArray[0].bookingid.toUpperCase()}' 
      AND (flightid = '${jsonArray[0].flightid1.toUpperCase()}' OR flightid = '${jsonArray[0].flightid2.toUpperCase()}')`)
      res.json([{ success: true}])
    }
    else{
      res.json([{ success: false}])
    }
  } catch (error) {
    console.error(error)
    res.status(500).json({ error: 'Internal Server Error' })
  }
})

```

UPGRADE TO BUSINESS CLASS



Ο χρήστης έχει την δυνατότητα μέσω της εφαρμογής να αναβαθμίσει την κλάση της πτήσης. Για να το κάνει αυτό αρκεί να κάνει ένα κλικ στο κουμπί "Upgrade To Business Class" και να μεταφερθεί στην σελίδα "Upgrade To Business Class", όπου θα του ζητηθεί να πληκτρολογήσει τον κωδικό της κράτησής του και το επώνυμό του είτε σε κεφαλαία είτε σε μικρά γράμματα στα πεδία "Booking reference" και "Last name" αντίστοιχα. Κάνοντας κλικ στο κουμπί "Continue" η εφαρμογή τρέχει ένα query για να ελέγξει αν υπάρχει κράτηση με αυτά τα στοιχεία και αν υπάρχει τρέχει query για να δει τι κλάση υπάρχει στο outbound και στο inbound (αν υπάρχει) της κράτησης. Αν ο χρήστης έχει business class σε όλες τις πτήσεις της κράτησής του δεν μπορεί να αναβαθμίσει κάτι παραπάνω. Με το πάτημα του κουμπιού "Continue" εκτελείται το POST API που φαίνεται παρακάτω.

To POST API είναι:

```
app.post('/flynow/upgrade-to-business', async (req, res) => {
```

Πρώτα, υπολογίζουμε τον αριθμό πτήσεων και έπειτα βλέπουμε τι query θα εκτελεστεί κάθε φορά.

Για μία πτήση:

```
const numOfflights = await pool.query(`  
SELECT COUNT(DISTINCT flightid)  
from has  
WHERE reservationid='${jsonArray[0].bookingId.toUpperCase()}'`)  
  
if(numOfflights.rows[0].count === '1') {  
    const result = await pool.query(`SELECT DISTINCT h.classtype FROM flight f, has h  
WHERE f.flightid=h.flightid  
AND h.reservationid='${jsonArray[0].bookingId.toUpperCase()}'`)  
  
    res.json([  
        oneWay: true,  
        outbound: {  
            classType: result.rows[0].classtype  
        }  
    ])  
}
```

Για δύο πτήσεις(πρώτο με επιστροφή-direct,δεύτερο με αναχώρηση-one stop):

```
else if(numOfflights.rows[0].count === '2') {  
    let result = await pool.query(`SELECT DISTINCT h1.classtype AS classtype1, h2.classtype AS classtype2  
FROM flight f1, flight f2, has h1, has h2  
WHERE f1.flightid=h1.flightid AND f2.flightid=h2.flightid  
AND h1.reservationid='${jsonArray[0].bookingId.toUpperCase()}'  
AND h2.reservationid='${jsonArray[0].bookingId.toUpperCase()}'  
AND f1.arrivalairport=f2.departureairport AND  
f2.arrivalairport=f1.departureairport  
AND f1.flighthdate<=f2.flighthdate`)  
  
    if(result.rows.length == 0) {  
        let result = await pool.query(`SELECT DISTINCT h1.classtype AS classtype  
FROM flight f1, flight f2, has h1, has h2  
WHERE f1.flightid=h1.flightid AND f2.flightid=h2.flightid  
AND h1.reservationid='${jsonArray[0].bookingId.toUpperCase()}'  
AND h2.reservationid='${jsonArray[0].bookingId.toUpperCase()}'  
AND f1.arrivalairport=f2.departureairport AND  
f2.arrivalairport<>f1.departureairport`)
```

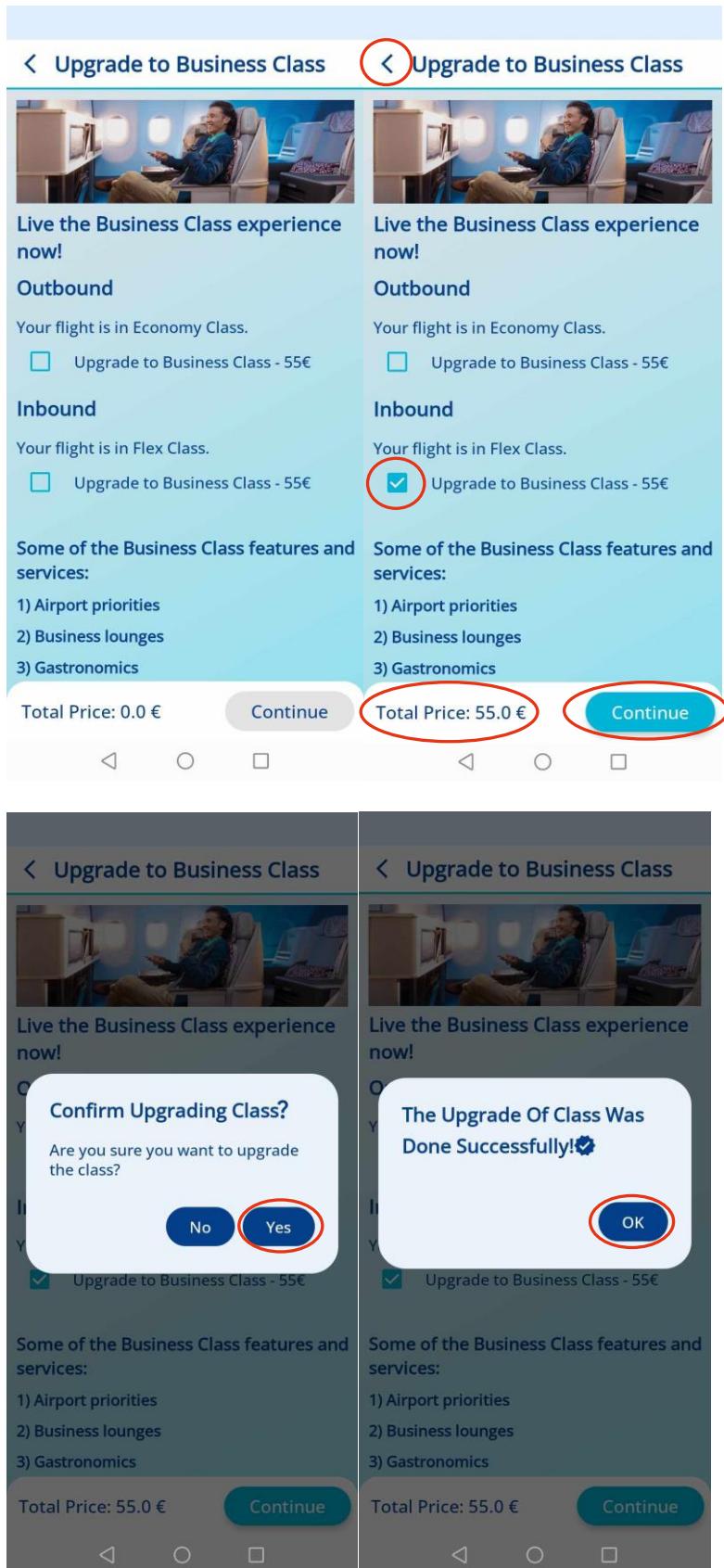
Για τρεις πτήσεις(πρώτο με αναχώρηση-direct και επιστροφή-one stop, δεύτερο με αναχώρηση-one stop και επιστροφή-direct):

```
else if(numOfFlights.rows[0].count === '3') {
    let result = await pool.query(`SELECT DISTINCT h1.classtype AS classtype1, h2.classtype AS classtype2
    FROM flight f1, flight f2, flight f3,
    has h1, has h2, has h3
    WHERE f1.flightid=h1.flightid AND f2.flightid=h2.flightid
    AND f3.flightid=h3.flightid
    AND h1.reservationid='${jsonArray[0].bookingId.toUpperCase()}' 
    AND h2.reservationid='${jsonArray[0].bookingId.toUpperCase()}' 
    AND h3.reservationid='${jsonArray[0].bookingId.toUpperCase()}' 
    AND f1.arrivalairport=f2.departureairport AND f2.arrivalairport=f3.departureairport
    AND f3.arrivalairport=f1.departureairport
    AND f2.flighthdate=f3.flighthdate
    AND f1.flighthdate<=f2.flighthdate`)

    if(result.rows.length == 0) {
        let result = await pool.query(`SELECT DISTINCT h1.classtype AS classtype1, h2.classtype AS classtype2
        FROM flight f1, flight f2, flight f3,
        has h1, has h2, has h3
        WHERE f1.flightid=h1.flightid AND f2.flightid=h2.flightid
        AND f3.flightid=h3.flightid
        AND h1.reservationid='${jsonArray[0].bookingId.toUpperCase()}' 
        AND h2.reservationid='${jsonArray[0].bookingId.toUpperCase()}' 
        AND h3.reservationid='${jsonArray[0].bookingId.toUpperCase()}' 
        AND f1.arrivalairport=f2.departureairport AND f2.arrivalairport=f3.departureairport
        AND f3.arrivalairport=f1.departureairport
        AND f2.flighthdate=f3.flighthdate
        AND f1.flighthdate>=f2.flighthdate`)
```

Για τέσσερις πτήσεις(με αναχώρηση και επιστροφή-one stop):

```
else if(numOfFlights.rows[0].count === '4') {
    const result = await pool.query(`SELECT DISTINCT h1.classtype AS classtype1, h3.classtype AS classtype2
    FROM flight f1, flight f2, flight f3,
    flight f4, has h1, has h2, has h3, has h4
    WHERE f1.flightid=h1.flightid AND f2.flightid=h2.flightid
    AND f3.flightid=h3.flightid AND f4.flightid=h4.flightid
    AND h1.reservationid='${jsonArray[0].bookingId.toUpperCase()}' 
    AND h2.reservationid='${jsonArray[0].bookingId.toUpperCase()}' 
    AND h3.reservationid='${jsonArray[0].bookingId.toUpperCase()}' 
    AND h4.reservationid='${jsonArray[0].bookingId.toUpperCase()}' 
    AND f1.arrivalairport=f2.departureairport AND f2.arrivalairport=f3.departureairport
    AND f3.arrivalairport=f4.departureairport AND f4.arrivalairport=f1.departureairport
    AND f1.flighthdate=f2.flighthdate AND f3.flighthdate=f4.flighthdate
    AND f1.flighthdate<=f3.flighthdate`)
```



Ο χρήστης μπορεί να αναβαθμίσει την κλάση των πτήσεων, για παράδειγμα αν επιλέξει να αναβαθμίσει το inbound και πατήσει το κουμπί "Continue". Θα εμφανιστεί το dialog που φαίνεται παρακάτω. Επίσης, παρατηρείται ότι υπάρχει περαιτέρω χρέωση για την αναβάθμιση. Μόλις ο χρήστης πατήσει το κουμπί "Yes" εκτελείται το POST API που φαίνεται παρακάτω για την ενημέρωση της κράτησης με αυτήν την αναβάθμιση και ενημερώνεται και η συνολική τιμή της κράτησης. Με το πάτημα του κουμπιού "OK" ο χρήστης πηγαίνει στο Home Screen.



Αν μια πτήση είναι ήδη στην Business Class η εφαρμογή δεν επιτρέπει περαιτέρω ενημέρωση της.

Προσοχή!

To Outbound και το Inbound αντίστοιχα μπορεί να αφορά και 2 πτήσεις δηλαδή με one-stop.

Το POST API που τρέχει για την ενημέρωση της κλάσης στην κράτηση είναι το παρακάτω:

```
app.post('/flynow/update-business', async (req, res) => {
```

Αν ο χρήστης έχει επιλέξει και το outbound και το inbound να κάνει Business Class εκτελεί το παρακάτω query:

```
if(jsonArray[0].outbound&&j jsonArray[0].inbound) {
  await pool.query(`UPDATE has
    SET classtype='BUSINESS CLASS'
    WHERE reservationid='${jsonArray[0].bookingId.toUpperCase()}'`)
}
```

Διαφορετικά, πρώτα υπολογίζουμε τον αριθμό πτήσεων όπως πριν και έπειτα βλέπουμε τι query θα εκτελεστεί κάθε φορά.

Για μία πτήση:

```
const numOfFlights = await pool.query(`  
SELECT COUNT(DISTINCT flightid)  
from has  
WHERE reservationid='${jsonArray[0].bookingId.toUpperCase()}'`)  
  
if(numOfFlights.rows[0].count === '1') {  
    if(jsonArray[0].outbound) {  
        await pool.query(`UPDATE has  
SET classtype='BUSINESS CLASS'  
WHERE reservationid='${jsonArray[0].bookingId.toUpperCase()}'`)  
    }  
}
```

Για δύο πτήσεις(πρώτο με επιστροφή-direct,δεύτερο με αναχώρηση-one stop):

```
else if(numOfFlights.rows[0].count === '2') {  
    let result = await pool.query(`SELECT DISTINCT f1.flightid AS flightid1, f2.flightid AS flightid2  
FROM flight f1, flight f2, has h1, has h2  
WHERE f1.flightid=h1.flightid AND f2.flightid=h2.flightid  
AND h1.reservationid='${jsonArray[0].bookingId.toUpperCase()}'  
AND h2.reservationid='${jsonArray[0].bookingId.toUpperCase()}'  
AND f1.arrivalairport=f2.departureairport AND  
f2.arrivalairport=f1.departureairport  
AND f1.flighdate<=f2.flighdate`)  
  
    if(result.rows.length == 0) {  
        if(jsonArray[0].outbound) {  
            await pool.query(`UPDATE has  
SET classtype='BUSINESS CLASS'  
WHERE reservationid='${jsonArray[0].bookingId.toUpperCase()}'`)  
        }  
    } else {  
        if(jsonArray[0].outbound) {  
            await pool.query(`UPDATE has  
SET classtype='BUSINESS CLASS'  
WHERE reservationid='${jsonArray[0].bookingId.toUpperCase()}'  
AND flightid='${result.rows[0].flightid1}'`)  
        }  
        else if(jsonArray[0].inbound) {  
            await pool.query(`UPDATE has  
SET classtype='BUSINESS CLASS'  
WHERE reservationid='${jsonArray[0].bookingId.toUpperCase()}'  
AND flightid='${result.rows[0].flightid2}'`)  
        }  
    }  
}
```

Για τρεις πτήσεις(πρώτο με αναχώρηση-direct και επιστροφή-one stop, δεύτερο με αναχώρηση-one stop και επιστροφή-direct):

```

else if(numOfFlights.rows[0].count === '3') {
    let result = await pool.query(`SELECT DISTINCT f1.flightid AS flightid1, f2.flightid AS flightid2, f3.flightid AS flightid3
    FROM flight f1, flight f2, flight f3,
    has h1, has h2, has h3
    WHERE f1.flightid=h1.flightid AND f2.flightid=h2.flightid
    AND f3.flightid=h3.flightid
    AND h1.reservationid='${jsonArray[0].bookingId.toUpperCase()}' 
    AND h2.reservationid='${jsonArray[0].bookingId.toUpperCase()}' 
    AND h3.reservationid='${jsonArray[0].bookingId.toUpperCase()}' 
    AND f1.arrivalairport=f2.departureairport AND f2.arrivalairport=f3.departureairport
    AND f3.arrivalairport=f1.departureairport
    AND f2.flightdate=f3.flightdate
    AND f1.flightdate<=f2.flightdate`)

    if(result.rows.length == 0) {
        let result = await pool.query(`SELECT DISTINCT f1.flightid AS flightid1, f2.flightid AS flightid2, f3.flightid AS flightid3
        FROM flight f1, flight f2, flight f3,
        has h1, has h2, has h3
        WHERE f1.flightid=h1.flightid AND f2.flightid=h2.flightid
        AND f3.flightid=h3.flightid
        AND h1.reservationid='${jsonArray[0].bookingId.toUpperCase()}' 
        AND h2.reservationid='${jsonArray[0].bookingId.toUpperCase()}' 
        AND h3.reservationid='${jsonArray[0].bookingId.toUpperCase()}' 
        AND f1.arrivalairport=f2.departureairport AND f2.arrivalairport=f3.departureairport
        AND f3.arrivalairport=f1.departureairport
        AND f2.flightdate=f3.flightdate
        AND f1.flightdate>=f2.flightdate`)

        if(jsonArray[0].outbound) {
            await pool.query(`UPDATE has
            SET classtype='BUSINESS CLASS'
            WHERE reservationid='${jsonArray[0].bookingId.toUpperCase()}' 
            AND (flightid='${result.rows[0].flightid2}' 
            OR flightid='${result.rows[0].flightid3}'`)

        }
        else if(jsonArray[0].inbound) {
            await pool.query(`UPDATE has
            SET classtype='BUSINESS CLASS'
            WHERE reservationid='${jsonArray[0].bookingId.toUpperCase()}' 
            AND flightid='${result.rows[0].flightid1}'`)

        }
    }
    else {
        if(jsonArray[0].outbound) {
            await pool.query(`UPDATE has
            SET classtype='BUSINESS CLASS'
            WHERE reservationid='${jsonArray[0].bookingId.toUpperCase()}' 
            AND flightid= ${result.rows[0].flightid1}`)

        }
        else if(jsonArray[0].inbound) {
            await pool.query(`UPDATE has
            SET classtype='BUSINESS CLASS'
            WHERE reservationid='${jsonArray[0].bookingId.toUpperCase()}' 
            AND (flightid='${result.rows[0].flightid2}' 
            OR flightid='${result.rows[0].flightid3}'`)

        }
    }
}

```

Για τέσσερις πτήσεις(με αναχώρηση και επιστροφή-one stop):

```
else if(numOfFlights.rows[0].count === '4') {
    const result = await pool.query(`SELECT DISTINCT f1.flightid AS flightid1, f2.flightid AS flightid2,
f3.flightid AS flightid3, f4.flightid AS flightid4
FROM flight f1, flight f2, flight f3,
flight f4, has h1, has h2, has h3, has h4
WHERE f1.flightid=h1.flightid AND f2.flightid=h2.flightid
AND f3.flightid=h3.flightid AND f4.flightid=h4.flightid
AND h1.reservationid='${jsonArray[0].bookingId.toUpperCase()}' 
AND h2.reservationid='${jsonArray[0].bookingId.toUpperCase()}' 
AND h3.reservationid='${jsonArray[0].bookingId.toUpperCase()}' 
AND h4.reservationid='${jsonArray[0].bookingId.toUpperCase()}' 
AND f1.arrivalairport=f2.departureairport AND f2.arrivalairport=f3.departureairport
AND f3.arrivalairport=f4.departureairport AND f4.arrivalairport=f1.departureairport
AND f1.flighthdate=f2.flighthdate AND f3.flighthdate=f4.flighthdate
AND f1.flighthdate<=f3.flighthdate`)

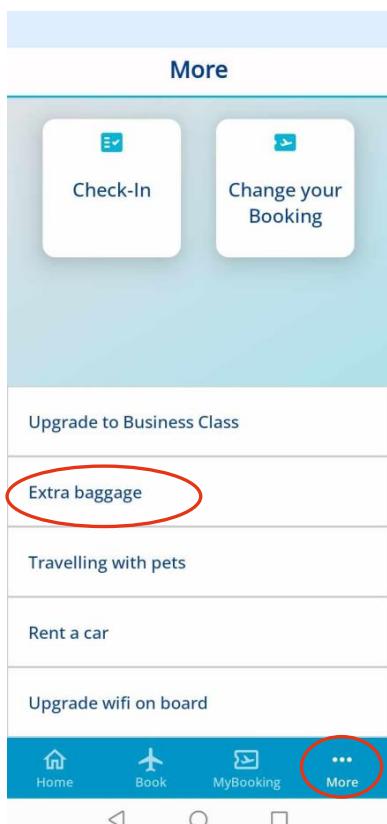
if(jsonArray[0].outbound) {
    await pool.query(' UPDATE has
SET classtype='BUSINESS CLASS'
WHERE reservationid='${jsonArray[0].bookingId.toUpperCase()}' 
AND (flightid='${result.rows[0].flightid1}'
OR flightid='${result.rows[0].flightid2}')')
}
else if(jsonArray[0].inbound) {
    await pool.query(' UPDATE has
SET classtype='BUSINESS CLASS'
WHERE reservationid='${jsonArray[0].bookingId.toUpperCase()}' 
AND (flightid='${result.rows[0].flightid3}'
OR flightid='${result.rows[0].flightid4}')')
}
}
```

Update query για την ενημέρωση της τιμής της κράτησης:

```
await pool.query(`

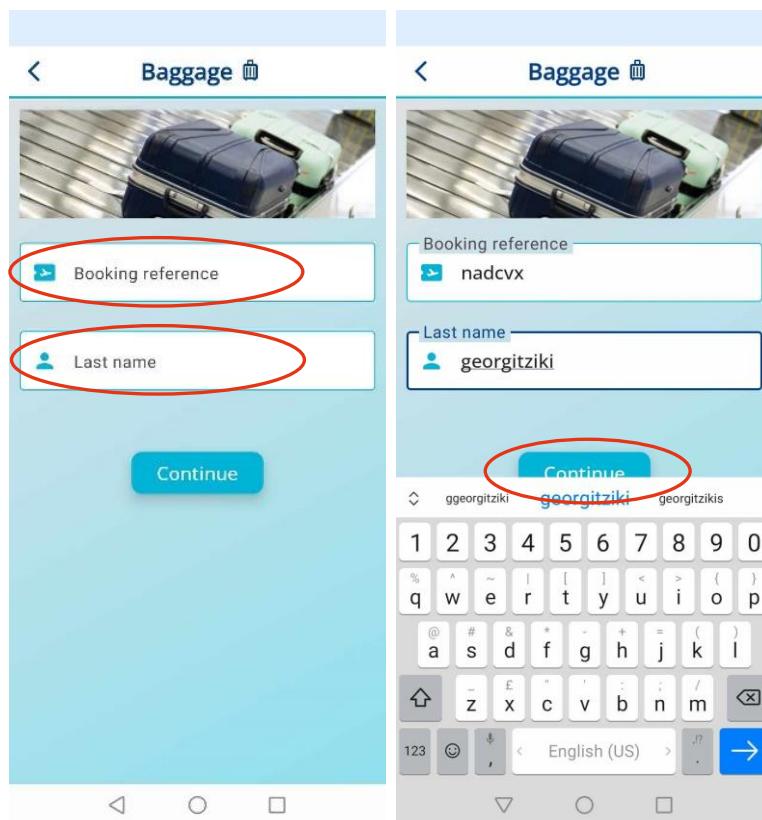
UPDATE reservation
SET price = price + ${jsonArray[0].price}
WHERE reservationid='${jsonArray[0].bookingId.toUpperCase()}'`)
```

EXTRA BAGGAGE



Για να προσθέσουμε επιπλέον βαλίτσες στην κράτησή μας, μπορούμε από το “More” κάνοντας κλικ στο “Extra baggage” να επιλέξουμε τις επιπλέον βαλίτσες για κάθε επιβάτη.

Πληκτρολογώντας τον κωδικό της κράτησής και το επώνυμο μας είτε σε κεφαλαία είτε σε μικρά γράμματα στα πεδία “Booking reference” και “Last name” αντίστοιχα, και κάνοντας κλικ στο κουμπί “Continue” μπορούμε να εισάγουμε τις επιπλέον βαλίτσες στην κράτησή μας. Τρέχει το ίδιο query με παραπάνω για τον έλεγχο ύπαρξης της κράτησης και εφόσον υπάρχει συνεχίζουμε στην επόμενη σελίδα.



Ταυτόχρονα, τρέχουν queries για την ανάκτηση των ήδη υπαρχουσών βαλιτσών για κάθε επιβάτη σε κάθε πτήση.

To POST API είναι:

```
app.post('/flynow/baggage-from-more', async (req, res) => {
```

To query που μετράει τον αριθμό των πτήσεων είναι:

```
const numOfFlights = await pool
  .query(`SELECT COUNT(DISTINCT flightid)
    |   |   FROM has
    |   | WHERE reservationid='${jsonArray[0].bookingId.toUpperCase()}'`)
```

Ανάλογα με τον αριθμό των πτήσεων τρέχει τα εξής queries:

- Για μία πτήση:

```
result = await pool
  .query(`SELECT DISTINCT p.firstname, p.lastname, p.sex, p.email, p.birthdate, p.phonenumber,
h1.numofbaggagepercategory AS numofbaggage23kg, h2.numofbaggagepercategory AS numofbaggage32kg
FROM flight f, has h1, has h2, passenger p
WHERE h1.reservationid = '${jsonArray[0].bookingId.toUpperCase()}'
AND h2.reservationid = '${jsonArray[0].bookingId.toUpperCase()}'
AND h1.baggage = 'baggage23kg'
AND h2.baggage = 'baggage32kg'
AND h1.flightid = f.flightid
AND h2.flightid = f.flightid
AND h1.passengerid = p.passengerid
AND h2.passengerid = p.passengerid`)
```

- Για δύο πτήσεις:

Εάν είναι με αναχώρηση μόνο με ενδιάμεση στάση, για τις βαλιτσές έχουμε:

```
result = await pool.query(`SELECT DISTINCT p.firstname, p.lastname, p.sex, p.email, p.birthdate, p.phonenumber,
h1.numofbaggagepercategory AS numofbaggage23kg, h2.numofbaggagepercategory AS numofbaggage32kg
FROM flight f1, flight f2, has h1, has h2, passenger p
WHERE f1.flightid = h1.flightid AND f2.flightid = h2.flightid
AND h1.reservationid = '${jsonArray[0].bookingId.toUpperCase()}'
AND h2.reservationid = '${jsonArray[0].bookingId.toUpperCase()}'
AND h1.baggage = 'baggage23kg'
AND h2.baggage = 'baggage32kg'
AND h1.passengerid = p.passengerid
AND h2.passengerid = p.passengerid
AND f1.arrivalairport=f2.departureairport AND
f2.arrivalairport<>f1.departureairport`)
```

Εάν είναι με αναχώρηση και επιστροφή απευθείας πτήσεις,
για τις βαλίτσες έχουμε:

```
result = await pool.query(`SELECT DISTINCT p.firstname, p.lastname, p.sex, p.email, p.birthdate, p.phonenumber,
h1.numofbaggagepercategory AS numofbaggage23kgOutbound, h2.numofbaggagepercategory AS numofbaggage32kgOutbound
,h3.numofbaggagepercategory AS numofbaggage23kgInbound, h4.numofbaggagepercategory AS numofbaggage32kgInbound
    FROM flight f1, flight f2, has h1, has h2, has h3, has h4, passenger p
    WHERE f1.flightid=h1.flightid AND f2.flightid=h3.flightid
    AND f1.flightid=h2.flightid AND f2.flightid=h4.flightid
    AND p.passengerid=h1.passengerid AND p.passengerid=h2.passengerid
    AND p.passengerid=h3.passengerid AND p.passengerid=h4.passengerid
    AND h1.reservationid='${jsonArray[0].bookingId.toUpperCase()}'  

    AND h2.reservationid='${jsonArray[0].bookingId.toUpperCase()}'  

    AND h3.reservationid='${jsonArray[0].bookingId.toUpperCase()}'  

    AND h4.reservationid='${jsonArray[0].bookingId.toUpperCase()}'  

    AND h1.baggage='baggage23kg'  

    AND h2.baggage='baggage32kg'  

    AND h3.baggage='baggage23kg'  

    AND h4.baggage='baggage32kg'  

    AND f1.arrivalairport=f2.departureairport AND
f2.arrivalairport=f1.departureairport  

    AND f1.flightdate<=f2.flightdate`)
```

- Για τρεις πτήσεις:

Εάν οι πτήσεις είναι αναχώρηση απευθείας και επιστροφή με
μία ενδιάμεση στάση, για τις βαλίτσες έχουμε:

```
result = await pool
.query(`SELECT DISTINCT p.firstname, p.lastname, p.sex, p.email, p.birthdate, p.phonenumber,
h1.numofbaggagepercategory AS numofbaggage23kgOutbound, h2.numofbaggagepercategory AS numofbaggage32kgOutbound
,h3.numofbaggagepercategory AS numofbaggage23kgInbound, h4.numofbaggagepercategory AS numofbaggage32kgInbound
FROM flight f1, flight f2, flight f3, has h1, has h2, has h3, has h4, has h5, has h6, passenger p
WHERE f1.flightid=h1.flightid AND f2.flightid=h3.flightid AND f1.flightid=h2.flightid
AND f2.flightid=h4.flightid AND f3.flightid=h5.flightid AND f3.flightid=h6.flightid
AND p.passengerid=h1.passengerid AND p.passengerid=h2.passengerid
AND p.passengerid=h3.passengerid AND p.passengerid=h4.passengerid
AND p.passengerid=h5.passengerid AND p.passengerid=h6.passengerid
AND h1.reservationid='${jsonArray[0].bookingId.toUpperCase()}'  

AND h2.reservationid='${jsonArray[0].bookingId.toUpperCase()}'  

AND h3.reservationid='${jsonArray[0].bookingId.toUpperCase()}'  

AND h4.reservationid='${jsonArray[0].bookingId.toUpperCase()}'  

AND h5.reservationid='${jsonArray[0].bookingId.toUpperCase()}'  

AND h6.reservationid='${jsonArray[0].bookingId.toUpperCase()}'  

AND h1.baggage='baggage23kg'  

AND h2.baggage='baggage32kg'  

AND h3.baggage='baggage23kg'  

AND h4.baggage='baggage32kg'  

AND h5.baggage='baggage23kg'  

AND h6.baggage='baggage32kg'  

AND f1.arrivalairport=f2.departureairport AND f2.arrivalairport=f3.departureairport
AND f3.arrivalairport=f1.departureairport
AND f2.flightdate=f3.flightdate
AND f1.flightdate<=f2.flightdate`)
```

Εάν οι πτήσεις είναι αναχώρηση με μία ενδιάμεση στάση και επιστροφή απευθείας, για τις βαλίτσες έχουμε:

```
result = await pool.query(`SELECT DISTINCT p.firstname, p.lastname, p.sex, p.email, p.birthdate, p.phonenumber,
h3.numofbaggagepercategory AS numofbaggage23kgOutbound, h4.numofbaggagepercategory AS numofbaggage32kgOutbound
,h1.numofbaggagepercategory AS numofbaggage23kgInbound, h2.numofbaggagepercategory AS numofbaggage32kgInbound
FROM flight f1, flight f2, flight f3, has h1, has h2, has h3, has h4, has h5, has h6, passenger p
WHERE f1.flightid=h1.flightid AND f2.flightid=h3.flightid AND f1.flightid=h2.flightid
AND f2.flightid=h4.flightid AND f3.flightid=h5.flightid AND f3.flightid=h6.flightid
AND p.passengerid=h1.passengerid AND p.passengerid=h2.passengerid
AND p.passengerid=h3.passengerid AND p.passengerid=h4.passengerid
AND p.passengerid=h5.passengerid AND p.passengerid=h6.passengerid
AND h1.reservationid='${jsonArray[0].bookingId.toUpperCase()}'  

AND h2.reservationid='${jsonArray[0].bookingId.toUpperCase()}'  

AND h3.reservationid='${jsonArray[0].bookingId.toUpperCase()}'  

AND h4.reservationid='${jsonArray[0].bookingId.toUpperCase()}'  

AND h5.reservationid='${jsonArray[0].bookingId.toUpperCase()}'  

AND h6.reservationid='${jsonArray[0].bookingId.toUpperCase()}'  

AND h1.baggage='baggage23kg'  

AND h2.baggage='baggage32kg'  

AND h3.baggage='baggage23kg'  

AND h4.baggage='baggage32kg'  

AND h5.baggage='baggage23kg'  

AND h6.baggage='baggage32kg'  

AND f1.arrivalairport=f2.departureairport AND f2.arrivalairport=f3.departureairport  

AND f3.arrivalairport=f1.departureairport  

AND f2.flighthdate=f3.flighthdate  

AND f1.flighthdate>=f2.flighthdate`)
```

- Για τέσσερις πτήσεις:

```
result = await pool
.query(`SELECT DISTINCT p.firstname, p.lastname, p.sex, p.email, p.birthdate, p.phonenumber,
h1.numofbaggagepercategory AS numofbaggage23kgOutbound, h2.numofbaggagepercategory AS numofbaggage32kgOutbound
,h5.numofbaggagepercategory AS numofbaggage23kgInbound, h6.numofbaggagepercategory AS numofbaggage32kgInbound
FROM flight f1, flight f2, flight f3, flight f4, has h1, has h2, has h3, has h4, has h5, has h6, has h7, has h8, passenger p
WHERE f1.flightid=h1.flightid AND f2.flightid=h3.flightid AND f1.flightid=h2.flightid
AND f2.flightid=h4.flightid AND f3.flightid=h5.flightid AND f3.flightid=h6.flightid
AND f4.flightid=h7.flightid AND f4.flightid=h8.flightid
AND p.passengerid=h1.passengerid AND p.passengerid=h2.passengerid
AND p.passengerid=h3.passengerid AND p.passengerid=h4.passengerid
AND p.passengerid=h5.passengerid AND p.passengerid=h6.passengerid
AND p.passengerid=h7.passengerid AND p.passengerid=h8.passengerid
AND h1.reservationid='${jsonArray[0].bookingId.toUpperCase()}'  

AND h2.reservationid='${jsonArray[0].bookingId.toUpperCase()}'  

AND h3.reservationid='${jsonArray[0].bookingId.toUpperCase()}'  

AND h4.reservationid='${jsonArray[0].bookingId.toUpperCase()}'  

AND h5.reservationid='${jsonArray[0].bookingId.toUpperCase()}'  

AND h6.reservationid='${jsonArray[0].bookingId.toUpperCase()}'  

AND h7.reservationid='${jsonArray[0].bookingId.toUpperCase()}'  

AND h8.reservationid='${jsonArray[0].bookingId.toUpperCase()}'  

AND h1.baggage='baggage23kg'  

AND h2.baggage='baggage32kg'  

AND h3.baggage='baggage23kg'  

AND h4.baggage='baggage32kg'  

AND h5.baggage='baggage23kg'  

AND h6.baggage='baggage32kg'  

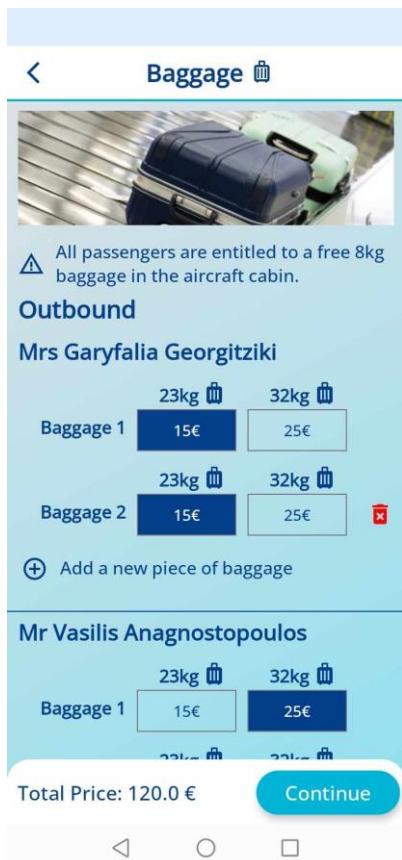
AND h7.baggage='baggage23kg'  

AND h8.baggage='baggage32kg'  

AND f1.arrivalairport=f2.departureairport AND f2.arrivalairport=f3.departureairport  

AND f3.arrivalairport=f4.departureairport AND f4.arrivalairport=f1.departureairport  

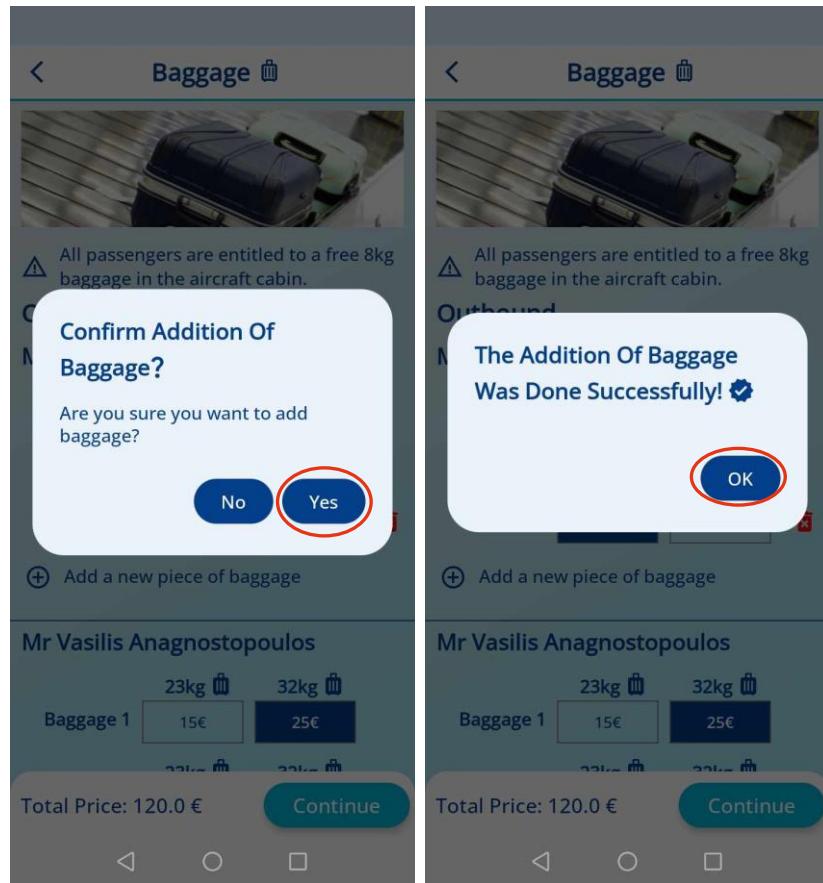
AND f1.flighthdate=f2.flighthdate AND f3.flighthdate=f4.flighthdate
AND f1.flighthdate<=f3.flighthdate`)
```



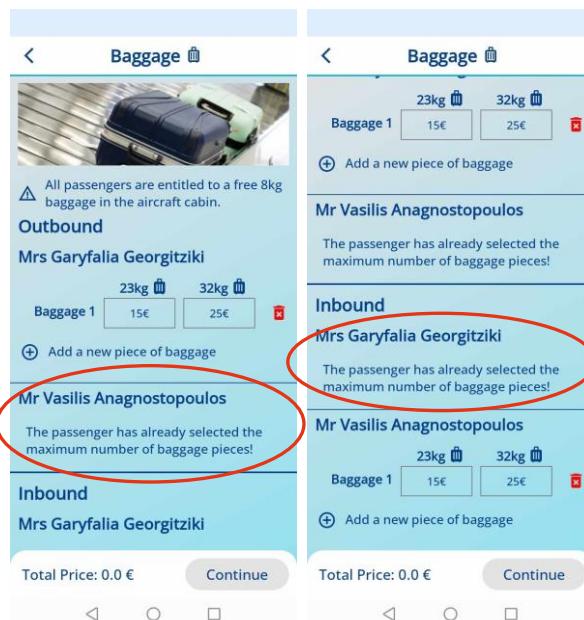
Τώρα οι βαλίτσες των 23 και 32 κιλών έχουν την σταθερή τιμή ανεξαρτήτως της κλάσης της κράτησης. Το όριο των 5 βαλίτσων ανά άτομο και ανά πτήση εξακολουθεί να υπάρχει και μάλιστα εάν ένας επιβάτης έχει συμπληρώσει τις 5 βαλίτσες κατά την διάρκεια της κράτησης δεν μπορεί να προσθέσει κάποια επιπλέον. Παρόμοια εάν έχει 4 βαλίτσες, μπορεί να προσθέσει ακόμη μία και ούτω καθεξής. Στις εικόνες δίπλα και κάτω βλέπουμε την προσθήκη κάποιων επιπλέον βαλίτσων. Η τιμή τους έχει προστεθεί στο "Total Price" και κάνοντας κλικ στο "Continue" γίνεται η επιβεβαίωση της προσθήκης των βαλίτσων.

	23kg	32kg
Baggage 1 (Mrs. Garyfalia)	15€	25€
Baggage 2 (Mrs. Garyfalia)	15€	25€
Baggage 1 (Mr. Vasilis)	15€	25€
Baggage 2 (Mr. Vasilis)	15€	25€

Total Price: 120.0 € Continue



Οπότε τώρα εάν πάμε να ξαναπροσθέσουμε βαλίτσες βλέπουμε ότι δεν μας αφήνει να προσθέσουμε επιπλέον βαλίτσες στους επιβάτες που συμπλήρωσαν το μέγιστο όριο των 5 βαλιτσών ανά πτήση.



Ta queries που τρέχουν για την προσθήκη των βαλιτσών εξαρτώνται πάλι από τον αριθμό των πτήσεων και φαίνονται παρακάτω:

To POST API:

```

app.post('/flynow/update-baggage', async (req, res) => {
  if(jsonArray[0].oneWay) {
    for(let i=0; i<j jsonArray[0].baggage.length; i++) {
      await pool
        .query(`UPDATE has_h
          SET numofbaggagepercategory = numofbaggagepercategory + ${j jsonArray[0].baggage[i].outbound.baggage23kg}
          FROM passenger p
          WHERE h.reservationid='${j jsonArray[0].bookingId.toUpperCase()}' 
          AND p.passengerid=h.passengerid
          AND p.email='${j jsonArray[0].baggage[i].email}'
          AND h.baggage='baggage23kg'`)

      await pool
        .query(`UPDATE has_h
          SET numofbaggagepercategory = numofbaggagepercategory + ${j jsonArray[0].baggage[i].outbound.baggage32kg}
          FROM passenger p
          WHERE h.reservationid='${j jsonArray[0].bookingId.toUpperCase()}' 
          AND p.passengerid=h.passengerid
          AND p.email='${j jsonArray[0].baggage[i].email}'
          AND h.baggage='baggage32kg'`)

    }
  }

  else {
    const numOfFlights = await pool.query(`SELECT COUNT(DISTINCT flightid) from has WHERE reservationid='${j jsonArray[0].bookingId.toUpperCase()}'`)
    if(numOfFlights.rows[0].count === '2') {
      let result = await pool.query(`SELECT DISTINCT f1.flightid AS flightid1, f2.flightid AS flightid2
        FROM flight f1, flight f2, has h1, has h2
        WHERE f1.flightid=h1.flightid AND f2.flightid=h2.flightid
        AND h1.reservationid='${j jsonArray[0].bookingId.toUpperCase()}' 
        AND h2.reservationid='${j jsonArray[0].bookingId.toUpperCase()}' 
        AND f1.arrivalairport=f2.departureairport 
        AND f2.arrivalairport=f1.departureairport
        AND f1.flightdate<=f2.flightdate`)

      for(let i=0; i<j jsonArray[0].baggage.length; i++) {
        await pool
          .query(`UPDATE has_h
            SET numofbaggagepercategory = h.numofbaggagepercategory + ${j jsonArray[0].baggage[i].outbound.baggage23kg}
            FROM passenger p
            WHERE h.reservationid='${j jsonArray[0].bookingId.toUpperCase()}' 
            AND p.passengerid=h.passengerid
            AND p.email='${j jsonArray[0].baggage[i].email}'
            AND h.baggage='baggage23kg'
            AND h.flightid='${result.rows[0].flightid1}'`)

        await pool
          .query(`UPDATE has_h
            SET numofbaggagepercategory = numofbaggagepercategory + ${j jsonArray[0].baggage[i].outbound.baggage32kg}
            FROM passenger p
            WHERE h.reservationid='${j jsonArray[0].bookingId.toUpperCase()}' 
            AND p.passengerid=h.passengerid
            AND p.email='${j jsonArray[0].baggage[i].email}'
            AND h.baggage='baggage32kg'
            AND h.flightid='${result.rows[0].flightid1}'`)
      }
    }
  }
})

```

```
        await pool
        .query(`UPDATE has_h
                SET numofbaggagepercategory = numofbaggagepercategory + ${jsonArray[0].baggage[i].inbound.baggage23kg}
                FROM passenger p
                WHERE h.reservationid='${jsonArray[0].bookingId.toUpperCase()}' 
                AND p.passengerid=h.passengerid
                AND p.email='${jsonArray[0].baggage[i].email}'
                AND h.baggage='baggage23kg'
                AND h.flightid='${result.rows[0].flightid2}'`)

        await pool
        .query(`UPDATE has_h
                SET numofbaggagepercategory = numofbaggagepercategory + ${jsonArray[0].baggage[i].inbound.baggage32kg}
                FROM passenger p
                WHERE h.reservationid='${jsonArray[0].bookingId.toUpperCase()}' 
                AND p.passengerid=h.passengerid
                AND p.email='${jsonArray[0].baggage[i].email}'
                AND h.baggage='baggage32kg'
                AND h.flightid='${result.rows[0].flightid2}'`)

    }
}
```

```

else if(numOfflights.rows[0].count === '3') {
  let result = await pool.query(`SELECT DISTINCT f1.flightid AS flightid1, f2.flightid AS flightid2, f3.flightid AS flightid3
    FROM flight f1, flight f2, flight f3,
    has h1, has h2, has h3
    WHERE f1.flightid=h1.flightid AND f2.flightid=h2.flightid
    AND f3.flightid=h3.flightid
    AND h1.reservationid='${jsonArray[0].bookingId.toUpperCase()}' 
    AND h2.reservationid='${jsonArray[0].bookingId.toUpperCase()}' 
    AND h3.reservationid='${jsonArray[0].bookingId.toUpperCase()}' 
    AND f1.arrivalairport=f2.departureairport AND f2.arrivalairport=f3.departureairport
    AND f3.arrivalairport=f1.departureairport
    AND f2.flighthdate=f3.flighthdate
    AND f1.flighthdate<=f2.flighthdate`)

  if(result.rows.length == 0) {
    let result = await pool.query(`SELECT DISTINCT f1.flightid AS flightid1, f2.flightid AS flightid2, f3.flightid AS flightid3
      FROM flight f1, flight f2, flight f3,
      has h1, has h2, has h3
      WHERE f1.flightid=h1.flightid AND f2.flightid=h2.flightid
      AND f3.flightid=h3.flightid
      AND h1.reservationid='${jsonArray[0].bookingId.toUpperCase()}' 
      AND h2.reservationid='${jsonArray[0].bookingId.toUpperCase()}' 
      AND h3.reservationid='${jsonArray[0].bookingId.toUpperCase()}' 
      AND f1.arrivalairport=f2.departureairport AND f2.arrivalairport=f3.departureairport
      AND f3.arrivalairport=f1.departureairport
      AND f2.flighthdate=f3.flighthdate
      AND f1.flighthdate>=f2.flighthdate`)

    for(let i=0; i<jsonArray[0].baggage.length; i++) {
      await pool
      .query(`UPDATE has h
        SET numofbaggagepercategory = numofbaggagepercategory + ${jsonArray[0].baggage[i].outbound.baggage23kg}
        FROM passenger p
        WHERE h.reservationid='${jsonArray[0].bookingId.toUpperCase()}' 
        AND p.passengerid=h.passengerid
        AND p.email='${jsonArray[0].baggage[i].email}' 
        AND h.baggage='baggage23kg'
        AND (h.flightid='${result.rows[0].flightid2}' 
        OR h.flightid='${result.rows[0].flightid3}')`)

      await pool
      .query(`UPDATE has h
        SET numofbaggagepercategory = numofbaggagepercategory + ${jsonArray[0].baggage[i].outbound.baggage32kg}
        FROM passenger p
        WHERE h.reservationid='${jsonArray[0].bookingId.toUpperCase()}' 
        AND p.passengerid=h.passengerid
        AND p.email='${jsonArray[0].baggage[i].email}' 
        AND h.baggage='baggage32kg'
        AND (h.flightid='${result.rows[0].flightid2}' 
        OR h.flightid='${result.rows[0].flightid3}')`)
    }
  }
}

```

```

        await pool
        .query(`UPDATE has_h
                SET numofbaggagepercategory = numofbaggagepercategory + ${jsonArray[0].baggage[i].inbound.baggage23kg}
                FROM passenger p
                WHERE h.reservationid='${jsonArray[0].bookingId.toUpperCase()}'
                AND p.passengerid=h.passengerid
                AND p.email='${jsonArray[0].baggage[i].email}'
                AND h.baggage='baggage23kg'
                AND h.flightid='${result.rows[0].flightid1}'`)

        await pool
        .query(`UPDATE has_h
                SET numofbaggagepercategory = numofbaggagepercategory + ${jsonArray[0].baggage[i].inbound.baggage32kg}
                FROM passenger p
                WHERE h.reservationid='${jsonArray[0].bookingId.toUpperCase()}'
                AND p.passengerid=h.passengerid
                AND p.email='${jsonArray[0].baggage[i].email}'
                AND h.baggage='baggage32kg'
                AND h.flightid='${result.rows[0].flightid1}'`)

    }
} else {
    for(let i=0; i<jsonArray[0].baggage.length; i++) {
        await pool
        .query(`UPDATE has_h
                SET numofbaggagepercategory = numofbaggagepercategory + ${jsonArray[0].baggage[i].outbound.baggage23kg}
                FROM passenger p
                WHERE h.reservationid='${jsonArray[0].bookingId.toUpperCase()}'
                AND p.passengerid=h.passengerid
                AND p.email='${jsonArray[0].baggage[i].email}'
                AND h.baggage='baggage23kg'
                AND h.flightid='${result.rows[0].flightid1}'`)

        await pool
        .query(`UPDATE has_h
                SET numofbaggagepercategory = numofbaggagepercategory + ${jsonArray[0].baggage[i].outbound.baggage32kg}
                FROM passenger p
                WHERE h.reservationid='${jsonArray[0].bookingId.toUpperCase()}'
                AND p.passengerid=h.passengerid
                AND p.email='${jsonArray[0].baggage[i].email}'
                AND h.baggage='baggage32kg'
                AND h.flightid='${result.rows[0].flightid1}'`)

        await pool
        .query(`UPDATE has_h
                SET numofbaggagepercategory = numofbaggagepercategory + ${jsonArray[0].baggage[i].inbound.baggage23kg}
                FROM passenger p
                WHERE h.reservationid='${jsonArray[0].bookingId.toUpperCase()}'
                AND p.passengerid=h.passengerid
                AND p.email='${jsonArray[0].baggage[i].email}'
                AND h.baggage='baggage23kg'
                AND (h.flightid='${result.rows[0].flightid2}'
                OR h.flightid='${result.rows[0].flightid3}')`)

        await pool
        .query(`UPDATE has_h
                SET numofbaggagepercategory = numofbaggagepercategory + ${jsonArray[0].baggage[i].inbound.baggage32kg}
                FROM passenger p
                WHERE h.reservationid='${jsonArray[0].bookingId.toUpperCase()}'
                AND p.passengerid=h.passengerid
                AND p.email='${jsonArray[0].baggage[i].email}'
                AND h.baggage='baggage32kg'
                AND (h.flightid='${result.rows[0].flightid2}'
                OR h.flightid='${result.rows[0].flightid3}')`)

    }
}

```

```

else if(numOfFlights.rows[0].count === '4') {
    const result = await pool.query(`SELECT DISTINCT f1.flightid AS flightid1,
    f2.flightid AS flightid2, f3.flightid AS flightid3, f4.flightid AS flightid4
    FROM flight f1, flight f2, flight f3,
    flight f4, has h1, has h2, has h3, has h4
    WHERE f1.flightid=h1.flightid AND f2.flightid=h2.flightid
    AND f3.flightid=h3.flightid AND f4.flightid=h4.flightid
    AND h1.reservationid='${jsonArray[0].bookingId.toUpperCase()}'
    AND h2.reservationid='${jsonArray[0].bookingId.toUpperCase()}'
    AND h3.reservationid='${jsonArray[0].bookingId.toUpperCase()}'
    AND h4.reservationid='${jsonArray[0].bookingId.toUpperCase()}'
    AND f1.arrivalairport=f2.departureairport AND f2.arrivalairport=f3.departureairport
    AND f3.arrivalairport=f4.departureairport AND f4.arrivalairport=f1.departureairport
    AND f1.flightdate=f2.flightdate AND f3.flightdate=f4.flightdate
    AND f1.flightdate<=f3.flightdate`)

    for(let i=0; i<jsonArray[0].baggage.length; i++) {
        await pool
        .query(`UPDATE has h
                SET numofbaggagepercategory = numofbaggagepercategory + ${jsonArray[0].baggage[i].outbound.baggage23kg}
                FROM passenger p
                WHERE h.reservationid='${jsonArray[0].bookingId.toUpperCase()}'
                AND p.passengerid=h.passengerid
                AND p.email='${jsonArray[0].baggage[i].email}'
                AND h.baggage='baggage23kg'
                AND (h.flightid='${result.rows[0].flightid1}'
                OR h.flightid='${result.rows[0].flightid2}')`)

        await pool
        .query(`UPDATE has h
                SET numofbaggagepercategory = numofbaggagepercategory + ${jsonArray[0].baggage[i].outbound.baggage32kg}
                FROM passenger p
                WHERE h.reservationid='${jsonArray[0].bookingId.toUpperCase()}'
                AND p.passengerid=h.passengerid
                AND p.email='${jsonArray[0].baggage[i].email}'
                AND h.baggage='baggage32kg'
                AND (h.flightid='${result.rows[0].flightid1}'
                OR h.flightid='${result.rows[0].flightid2}')`)

        await pool
        .query(`UPDATE has h
                SET numofbaggagepercategory = numofbaggagepercategory + ${jsonArray[0].baggage[i].inbound.baggage23kg}
                FROM passenger p
                WHERE h.reservationid='${jsonArray[0].bookingId.toUpperCase()}'
                AND p.passengerid=h.passengerid
                AND p.email='${jsonArray[0].baggage[i].email}'
                AND h.baggage='baggage23kg'
                AND (h.flightid='${result.rows[0].flightid3}'
                OR h.flightid='${result.rows[0].flightid4}')`)

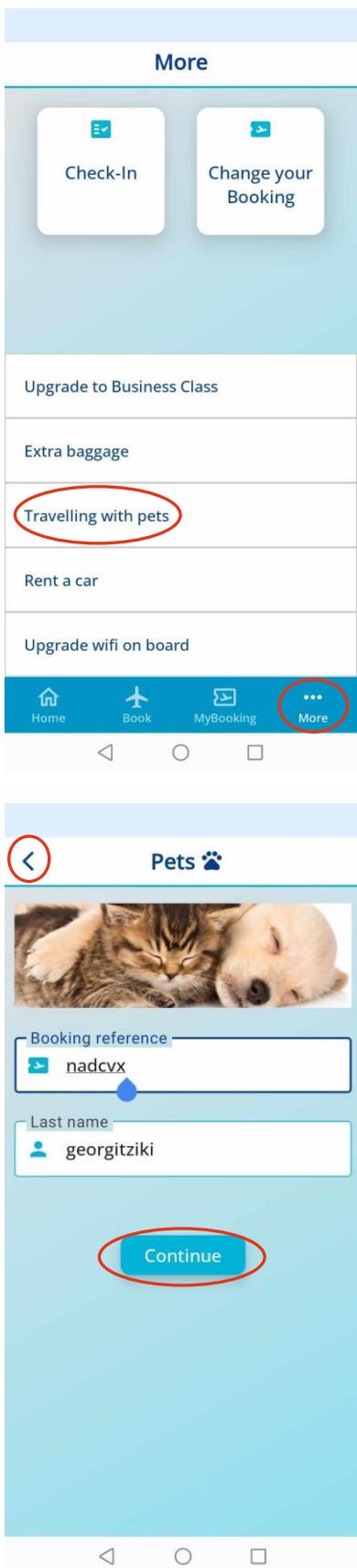
        await pool
        .query(`UPDATE has h
                SET numofbaggagepercategory = numofbaggagepercategory + ${jsonArray[0].baggage[i].inbound.baggage32kg}
                FROM passenger p
                WHERE h.reservationid='${jsonArray[0].bookingId.toUpperCase()}'
                AND p.passengerid=h.passengerid
                AND p.email='${jsonArray[0].baggage[i].email}'
                AND h.baggage='baggage32kg'
                AND (h.flightid='${result.rows[0].flightid3}'
                OR h.flightid='${result.rows[0].flightid4}')`)

    }
}

await pool
.query(`UPDATE reservation
        SET price= price + ${jsonArray[0].price}
        WHERE reservationid='${jsonArray[0].bookingId.toUpperCase()}'`)

```

TRAVELLING WITH PETS



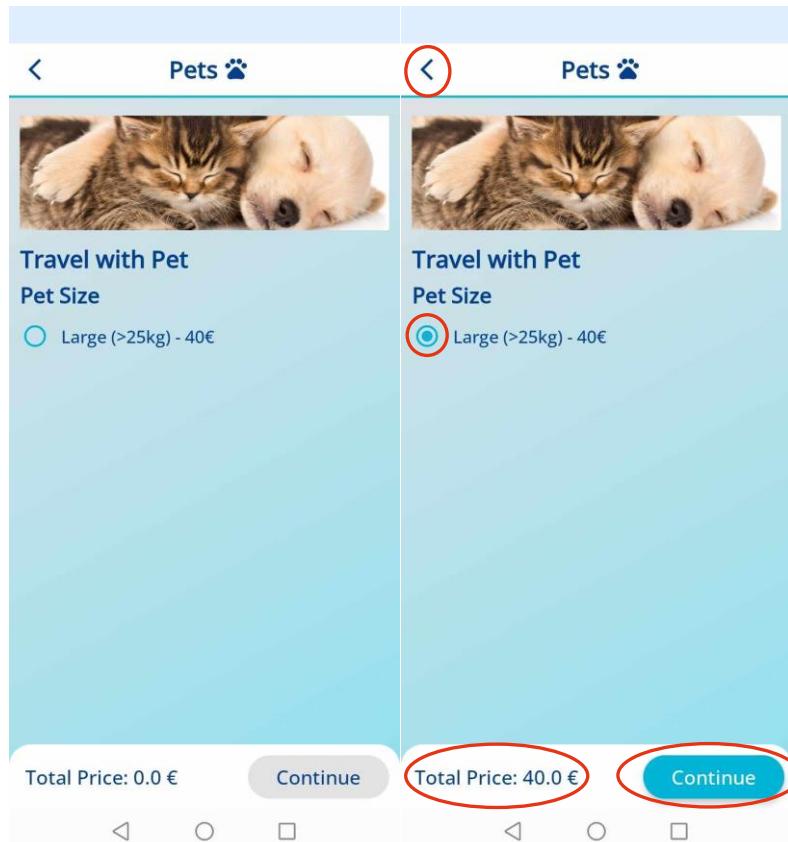
Ο χρήστης έχει την δυνατότητα μέσω της εφαρμογής να προσθέσει ένα κατοικίδιο ή να αλλάξει μέγεθος κατοικίδιου εάν έχει προσθέσει ήδη. Για να το κάνει αυτό αρκεί να κάνει ένα κλικ στο κουμπί "Travelling with pets" και να μεταφερθεί στην σελίδα "Pets", όπου θα του ζητηθεί να πληκτρολογήσει τον κωδικό της κράτησής του και το επώνυμο του είτε σε κεφαλαία είτε σε μικρά γράμματα στα πεδία "Booking reference" και "Last name" αντίστοιχα. Κάνοντας κλικ στο κουμπί "Continue" η εφαρμογή τρέχει ένα query για να ελέγξει αν υπάρχει κράτηση με αυτά τα στοιχεία και αν υπάρχει τρέχει query για να δει αν υπάρχει pet ήδη στην κράτηση. Η αλλαγή του μεγέθους του κατοικίδιου γίνεται με μεγαλύτερο μέγεθος πάντα με επιπλέον πρόσθεση στην τελική τιμή όπου προστίθεται τελικά στην συνολική τιμή της κράτησης. Αν τελικά ο χρήστης έχει επιλέξει το μεγαλύτερο μέγεθος δεν μπορεί να επιλέξει κάτι παραπάνω.

Οπότε με το πάτημα του κουμπιού “Continue” τρέχει το παρακάτω POST API:

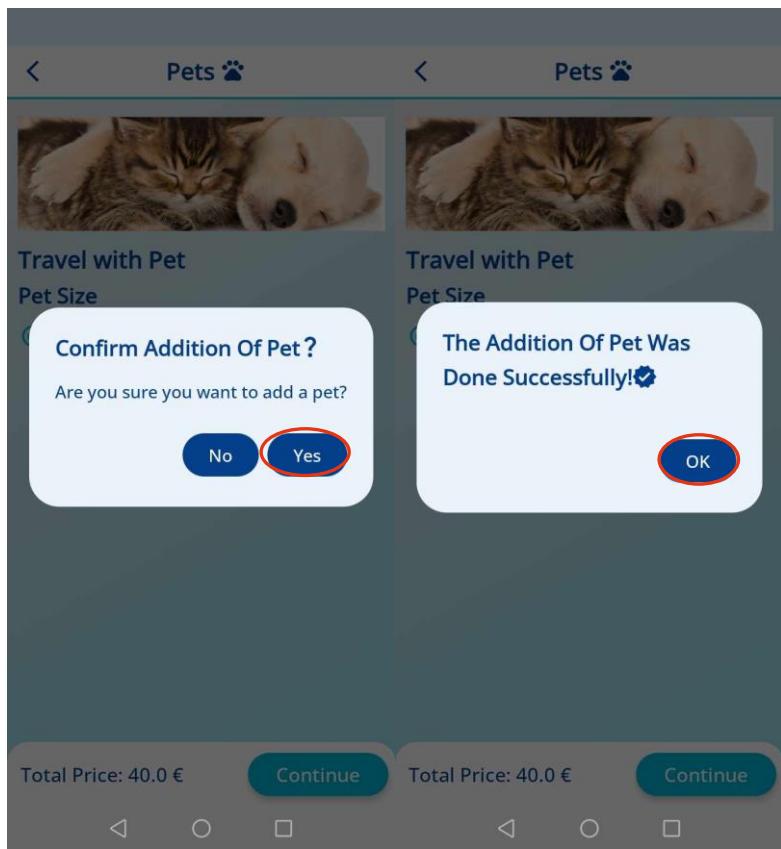
```
//POST api that returns the value from the petsize field of the database for the pets from more screen
app.post('/flynow/pets-from-more', async (req, res) => {
  const jsonArray = req.body

  try {
    const result = await pool
      .query(`SELECT r.petsize
              FROM reservation r
              WHERE r.reservationid='${jsonArray[0].bookingid.toUpperCase()}'`)

    if(result.rows[0].petsize == null) {
      result.rows[0].petsize = ""
    }
    res.json([{"petSize: result.rows[0].petsize}])
  } catch (error) {
    console.error(error)
    res.status(500).json({ error: 'Internal Server Error' })
  }
})
```



Ο χρήστης πρέπει να επιλέξει κάποιο μέγεθος στην προκειμένη περίπτωση Large διότι ήδη η κράτηση έχει Medium και πατώντας το κουμπί “Continue” εμφανίζεται όπως φαίνεται το dialog παρακάτω. Ο χρήστης πρέπει να πατήσει “Yes” για να ενημερωθεί το κατοικίδιο στην κράτηση οπότε μετά αν όλα πάνε καλά εμφανίζεται επιτυχημένη η αλλαγή και πατώντας το “OK” ο χρήστης μεταφέρεται στο “Home Screen”.



Εάν ο χρήστης έχει επιλέξει το μεγαλύτερο μέγεθος δεν μπορεί να κάνει κάποια περαιτέρω ενημέρωση σχετικά με το κατοικίδιο στην κράτηση.

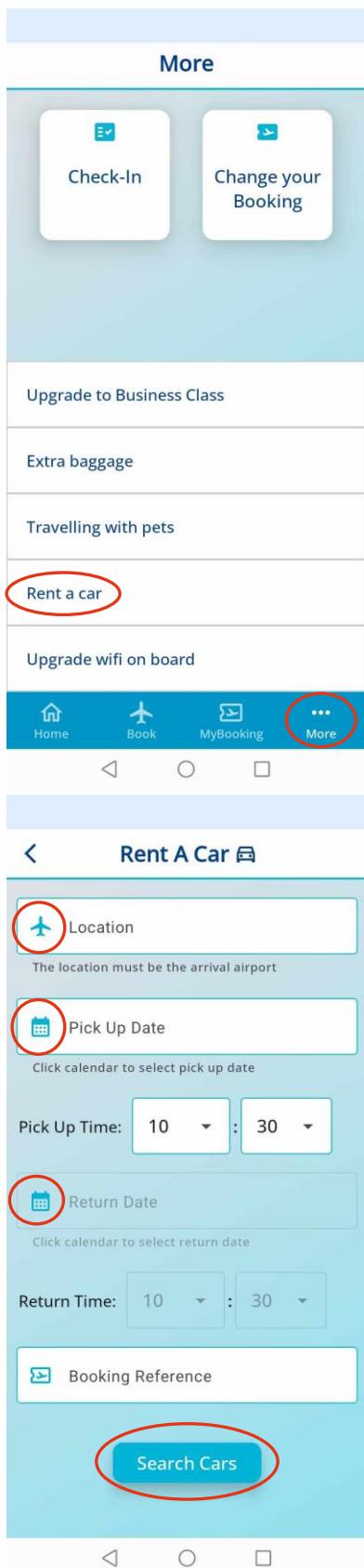
Για την ενημέρωση του κατοικίδιου στην κράτηση τρέχει το παρακάτω POST API:

```
//POST api that updates the petsize field of the database that select the user from the pets from more screen
//+ the new price of the reservation that is priceOld + priceNew of the new pet size
app.post('/flynow/update-pets', async (req, res) => {
    const jsonArray = req.body

    try {
        const result = await pool
        .query(` UPDATE reservation
                SET petsize= '${jsonArray[0].petSize}', price= price + ${jsonArray[0].price}
                WHERE reservationid='${jsonArray[0].bookingid.toUpperCase()}'`)

        res.json([{ success: true }])
    } catch (error) {
        console.error(error)
        res.status(500).json({ error: 'Internal Server Error' })
    }
})
```

RENT A CAR



Ο χρήστης έχει την δυνατότητα μέσω της εφαρμογής να κάνει ενοικίαση ενός αυτοκινήτου. Κάνοντας κλικ στο κουμπί "Rent a car" μεταφέρεται στην σελίδα "Rent A Car", όπου θα του ζητηθεί να επιλέξει το αεροδρόμιο προορισμού με τον ίδιο τρόπο που έγινε και στην ολοκλήρωση κράτησης πχ. αν το ταξίδι είναι One Way ή Round Trip το αεροδρόμιο προορισμού είναι πάντα το αεροδρόμιο προορισμού του Outbound(δηλαδή LIS). Έπειτα, γίνεται επιλογή pick up και return(πρέπει πρώτα να επιλεχθεί pick up) ημερομηνίας με την επιλογή του εικονιδίου ημερολόγιο και ώρας/λεπτών(η λίστα της ώρας είναι από το 00 μέχρι το 23 και η λίστα των λεπτών είναι με βήμα 15 από 00 μέχρι το 45), η μέγιστη διαφορά ημερών μπορεί να είναι μέχρι 10 ημέρες. Τέλος πληκτρολογεί τον κωδικό κράτησης και κάνοντας κλικ στο κουμπί "Search Cars" η εφαρμογή τρέχει ένα query για να ελέγξει αν υπάρχει η κράτηση και αν το αεροδρόμιο είναι όντως του προορισμού της κράτησης, διαφορετικά δείχνει ότι υπάρχει error. Αν όλα πάνε καλά τότε θα πρέπει να εμφανισθούν τα αυτοκίνητα που είναι διαθέσιμα στο αεροδρόμιο που επιλέχθηκε και δεν βρίσκονται σε κάποια κράτηση.

The first screenshot shows an error for an incorrect arrival airport (ATH) and a booking reference (nadcvd). The second shows another incorrect arrival airport (ATH) and reference (nadcvx). The third shows errors for both arrival airport (MAD) and reference (nadcvx).

Location	Pick Up Date	Return Date	Booking Reference
ATH	18/02/2024	23/02/2024	nadcvd
ATH	18/02/2024	23/02/2024	nadcvx
MAD	18/02/2024	23/02/2024	nadcvx

Search Cars

The first screenshot shows errors for arrival airport (LIS), pick-up date (18/02/2024), return time (17:30), and booking reference (nadcvx). The second shows errors for arrival airport (LIS), pick-up date (16/02/2024), return time (10:30), and booking reference (NADCVX). The third shows errors for arrival airport (LIS), pick-up date (18/02/2024), return time (10:30), and booking reference (NADCVX).

Location	Pick Up Date	Return Date	Booking Reference
LIS	18/02/2024	18/02/2024	nadcvx
LIS	16/02/2024	25/02/2024	NADCVX
LIS	18/02/2024	25/02/2024	NADCVX

Search Cars

Για τον έλεγχο της κράτησης και του αεροδρομίου τρέχει το παρακάτω POST API:

```
app.post('/flynow/car-booking-exists', async (req, res) => {
```

Έλεγχος αν υπάρχει το booking reference:

```
const result = await pool.query(`  
SELECT reservationid  
FROM reservation  
WHERE reservationid='${jsonArray[0].bookingId.toUpperCase()}'`)  
  
if(result.rows.length==0) {  
    res.json([{success: false, successAirport: true, successTime: true}])  
}
```

Έλεγχος για το αεροδρόμιο προορισμού και για το αν το pick-up και το return βρίσκονται μέσα στα όρια μετά την άφιξη της outbound πτήσης και πριν την αναχώρηση της inbound πτήσης(αν υπάρχει) της κράτησης.

Πρώτα, υπολογίζουμε τον αριθμό πτήσεων και έπειτα βλέπουμε αν το αεροδρόμιο προορισμού είναι αυτό που επέλεξε ο χρήστης και αν το pick-up/return datetime είναι σωστά.

Για μία πτήση:

```
const numOfflights = await pool.query(`  
SELECT COUNT(DISTINCT flightid)  
from has  
WHERE reservationid='${jsonArray[0].bookingId.toUpperCase()}'`)  
  
if(numOfflights.rows[0].count === '1') {  
    const result = await pool.query(`SELECT DISTINCT f.flightid FROM flight f, has h  
    WHERE f.flightid=h.flightid  
    AND h.reservationid='${jsonArray[0].bookingId.toUpperCase()}'  
    AND f.arrivalairport='${jsonArray[0].location}'`)  
  
    if(result.rows.length == 0) {  
        check = 1  
        res.json([{success: true, successAirport: false, successTime: true}])  
    }  
    else {  
        const result1 = await pool.query(`SELECT DISTINCT f.flightid FROM flight f  
        WHERE f.flightid='${result.rows[0].flightid}'  
        AND (EXTRACT(HOUR FROM f.arrivaltime) * 60 + EXTRACT(MINUTE FROM f.arrivaltime) <  
        ${jsonArray[0].pickUpHours} * 60 + ${jsonArray[0].pickUpMinutes})  
        AND f.flightdate = TO_DATE('${jsonArray[0].pickUpDate}', 'DD/MM/YYYY')  
        OR f.flightdate < TO_DATE('${jsonArray[0].pickUpDate}', 'DD/MM/YYYY')`)  
  
        if(result1.rows.length == 0) {  
            check = 1  
            res.json([{success: true, successAirport: true, successTime: false}])  
        }  
    }  
}  
}
```

Για δύο πτήσεις:

```
else if(numOfFlights.rows[0].count === '2') {
    const result = await pool.query(`SELECT DISTINCT f1.flightid AS "flightid1", f2.flightid AS "flightid2"
    FROM flight f1, flight f2, has h1, has h2
    WHERE f1.flightid=h1.flightid AND f2.flightid=h2.flightid
    AND h1.reservationid='${jsonArray[0].bookingId.toUpperCase()}' 
    AND h2.reservationid='${jsonArray[0].bookingId.toUpperCase()}' 
    AND f1.arrivalairport=f2.departureairport AND
    (f2.arrivalairport=f1.departureairport AND f1.flightdate<=f2.flightdate
    AND f1.arrivalairport='${jsonArray[0].location}'`)

    const result1 = await pool.query(`SELECT DISTINCT f2.flightid FROM flight f1, flight f2, has h1, has h2
    WHERE f1.flightid=h1.flightid AND f2.flightid=h2.flightid
    AND h1.reservationid='${jsonArray[0].bookingId.toUpperCase()}' 
    AND h2.reservationid='${jsonArray[0].bookingId.toUpperCase()}' 
    AND f1.arrivalairport=f2.departureairport
    AND (f2.arrivalairport<>f1.departureairport
    AND f2.arrivalairport='${jsonArray[0].location}'`)

    if(result.rows.length == 0 && result1.rows.length == 0) {
        check = 1
        res.json([{success: true, successAirport: false, successTime: true}])
    }
    else if(result1.rows.length != 0) {
        const result2 = await pool.query(`SELECT DISTINCT f1.flightid FROM flight f1
        WHERE (f1.flightid='${result1.rows[0].flightid}'
        AND ((EXTRACT(HOUR FROM f1.arrivaltime) * 60 + EXTRACT(MINUTE FROM f1.arrivaltime) <
        ${jsonArray[0].pickUpHours} * 60 + ${jsonArray[0].pickUpMinutes}
        AND f1.flightdate = TO_DATE('${jsonArray[0].pickUpDate}', 'DD/MM/YYYY'))
        OR f1.flightdate < TO_DATE('${jsonArray[0].pickUpDate}', 'DD/MM/YYYY'))`)

        if(result2.rows.length == 0) {
            check = 1
            res.json([{success: true, successAirport: true, successTime: false}])
        }
    }
    else if(result.rows.length != 0) {
        const result3 = await pool.query(`SELECT DISTINCT f1.flightid FROM flight f1, flight f2
        WHERE (f1.flightid='${result.rows[0].flightid1}'
        AND ((EXTRACT(HOUR FROM f1.arrivaltime) * 60 + EXTRACT(MINUTE FROM f1.arrivaltime) <
        ${jsonArray[0].pickUpHours} * 60 + ${jsonArray[0].pickUpMinutes}
        AND f1.flightdate = TO_DATE('${jsonArray[0].pickUpDate}', 'DD/MM/YYYY'))
        OR f1.flightdate < TO_DATE('${jsonArray[0].pickUpDate}', 'DD/MM/YYYY'))
        AND (f2.flightid='${result.rows[0].flightid2}'
        AND ((EXTRACT(HOUR FROM f2.departuretime) * 60 + EXTRACT(MINUTE FROM f2.departuretime) >
        ${jsonArray[0].returnHours} * 60 + ${jsonArray[0].returnMinutes}
        AND f2.flightdate = TO_DATE('${jsonArray[0].returnDate}', 'DD/MM/YYYY'))
        OR f2.flightdate > TO_DATE('${jsonArray[0].returnDate}', 'DD/MM/YYYY'))`)

        if(result3.rows.length == 0) {
            check = 1
            res.json([{success: true, successAirport: true, successTime: false}])
        }
    }
}
```

Για τρεις πτήσεις:

```
else if(numOfflights.rows[0].count === '3') {
    const result = await pool.query(`SELECT DISTINCT f1.flightid AS "flightid1", f2.flightid AS "flightid2" FROM flight f1, flight f2, flight f3,
    has h1, has h2, has h3
    WHERE f1.flightid=h1.flightid AND f2.flightid=h2.flightid
    AND f3.flightid=h3.flightid
    AND h1.reservationid='${jsonArray[0].bookingId.toUpperCase()}' 
    AND h2.reservationid='${jsonArray[0].bookingId.toUpperCase()}' 
    AND h3.reservationid='${jsonArray[0].bookingId.toUpperCase()}' 
    AND f1.arrivalairport=f2.departureairport AND f2.arrivalairport=f3.departureairport
    AND f3.arrivalairport=f1.departureairport
    AND f2.flightdate=f3.flightdate
    AND (f1.flightdate<=f2.flightdate AND f1.arrivalairport='${jsonArray[0].location}')`)

    const result1 = await pool.query(`SELECT DISTINCT f3.flightid AS "flightid1", f1.flightid AS "flightid2" FROM flight f1, flight f2, flight f3,
    has h2, has h3
    WHERE f1.flightid=h1.flightid AND f2.flightid=h2.flightid
    AND f3.flightid=h3.flightid
    AND h1.reservationid='${jsonArray[0].bookingId.toUpperCase()}' 
    AND h2.reservationid='${jsonArray[0].bookingId.toUpperCase()}' 
    AND h3.reservationid='${jsonArray[0].bookingId.toUpperCase()}' 
    AND f1.arrivalairport=f2.departureairport AND f2.arrivalairport=f3.departureairport
    AND f3.arrivalairport=f1.departureairport
    AND f2.flightdate=f3.flightdate
    AND (f1.flightdate>=f2.flightdate AND f1.departureairport='${jsonArray[0].location}')`)

    if(result.rows.length == 0 && result1.rows.length == 0) {
        check = 1
        res.json([{success: true, successAirport: false, successTime: true}])
    }
    else if(result1.rows.length != 0) {
        const result2 = await pool.query(`SELECT DISTINCT f1.flightid FROM flight f1, flight f2
        WHERE (f1.flightid='${result1.rows[0].flightid1}' 
        AND ((EXTRACT(HOUR FROM f1.arrivaltime) * 60 + EXTRACT(MINUTE FROM f1.arrivaltime) <
        ${jsonArray[0].pickUpHours} * 60 + ${jsonArray[0].pickUpMinutes}
        AND f1.flightdate = TO_DATE('${jsonArray[0].pickUpDate}', 'DD/MM/YYYY'))
        OR f1.flightdate < TO_DATE('${jsonArray[0].pickUpDate}', 'DD/MM/YYYY')))
        AND (f2.flightid='${result1.rows[0].flightid2}' 
        AND ((EXTRACT(HOUR FROM f2.departuretime) * 60 + EXTRACT(MINUTE FROM f2.departuretime) >
        ${jsonArray[0].returnHours} * 60 + ${jsonArray[0].returnMinutes}
        AND f2.flightdate = TO_DATE('${jsonArray[0].returnDate}', 'DD/MM/YYYY'))
        OR f2.flightdate > TO_DATE('${jsonArray[0].returnDate}', 'DD/MM/YYYY'))`)

        if(result2.rows.length == 0) {
            check = 1
            res.json([{success: true, successAirport: true, successTime: false}])
        }
    }
    else if(result.rows.length != 0) {
        const result3 = await pool.query(`SELECT DISTINCT f1.flightid FROM flight f1, flight f2
        WHERE (f1.flightid='${result.rows[0].flightid1}' 
        AND ((EXTRACT(HOUR FROM f1.arrivaltime) * 60 + EXTRACT(MINUTE FROM f1.arrivaltime) <
        ${jsonArray[0].pickUpHours} * 60 + ${jsonArray[0].pickUpMinutes}
        AND f1.flightdate = TO_DATE('${jsonArray[0].pickUpDate}', 'DD/MM/YYYY'))
        OR f1.flightdate < TO_DATE('${jsonArray[0].pickUpDate}', 'DD/MM/YYYY')))
        AND (f2.flightid='${result.rows[0].flightid2}' 
        AND ((EXTRACT(HOUR FROM f2.departuretime) * 60 + EXTRACT(MINUTE FROM f2.departuretime) >
        ${jsonArray[0].returnHours} * 60 + ${jsonArray[0].returnMinutes}
        AND f2.flightdate = TO_DATE('${jsonArray[0].returnDate}', 'DD/MM/YYYY'))
        OR f2.flightdate > TO_DATE('${jsonArray[0].returnDate}', 'DD/MM/YYYY'))`)

        if(result3.rows.length == 0) {
            check = 1
            res.json([{success: true, successAirport: true, successTime: false}])
        }
    }
}
```

Για τέσσερις πτήσεις:

```
else if(numOfflights.rows[0].count === '4') {
    const result = await pool.query(`SELECT DISTINCT f2.flightid AS "flightid1", f3.flightid AS "flightid2" FROM flight f1, flight f2, flight f3,
    flight f4, has h1, has h2, has h3, has h4
    WHERE f1.flightid=h1.flightid AND f2.flightid=h2.flightid
    AND f3.flightid=h3.flightid AND f4.flightid=h4.flightid
    AND h1.reservationid='${jsonArray[0].bookingId.toUpperCase()}' 
    AND h2.reservationid='${jsonArray[0].bookingId.toUpperCase()}' 
    AND h3.reservationid='${jsonArray[0].bookingId.toUpperCase()}' 
    AND h4.reservationid='${jsonArray[0].bookingId.toUpperCase()}' 
    AND f1.arrivalairport=f2.departureairport AND f2.arrivalairport=f3.departureairport
    AND f3.arrivalairport=f4.departureairport AND f4.arrivalairport=f1.departureairport
    AND f1.flightdate=f2.flightdate AND f3.flightdate=f4.flightdate
    AND f1.flightdate<=f3.flightdate AND f2.arrivalairport='${jsonArray[0].location}'`)

    if(result.rows.length == 0) {
        check = 1
        res.json([{success: true, successAirport: false, successTime: true}])
    }
    else {
        const result1 = await pool.query(`SELECT DISTINCT f1.flightid FROM flight f1, flight f2
        WHERE (f1.flightid='${result.rows[0].flightid1}' 
        AND ((EXTRACT(HOUR FROM f1.arrivaltime) * 60 + EXTRACT(MINUTE FROM f1.arrivaltime) <
        ${jsonArray[0].pickUpHours} * 60 + ${jsonArray[0].pickUpMinutes}) 
        AND f1.flightdate = TO_DATE('${jsonArray[0].pickUpDate}', 'DD/MM/YYYY')) 
        OR f1.flightdate < TO_DATE('${jsonArray[0].pickUpDate}', 'DD/MM/YYYY'))) 
        AND (f2.flightid='${result.rows[0].flightid2}' 
        AND ((EXTRACT(HOUR FROM f2.departuretime) * 60 + EXTRACT(MINUTE FROM f2.departuretime) >
        ${jsonArray[0].returnHours} * 60 + ${jsonArray[0].returnMinutes}) 
        AND f2.flightdate = TO_DATE('${jsonArray[0].returnDate}', 'DD/MM/YYYY')) 
        OR f2.flightdate > TO_DATE('${jsonArray[0].returnDate}', 'DD/MM/YYYY'))`)

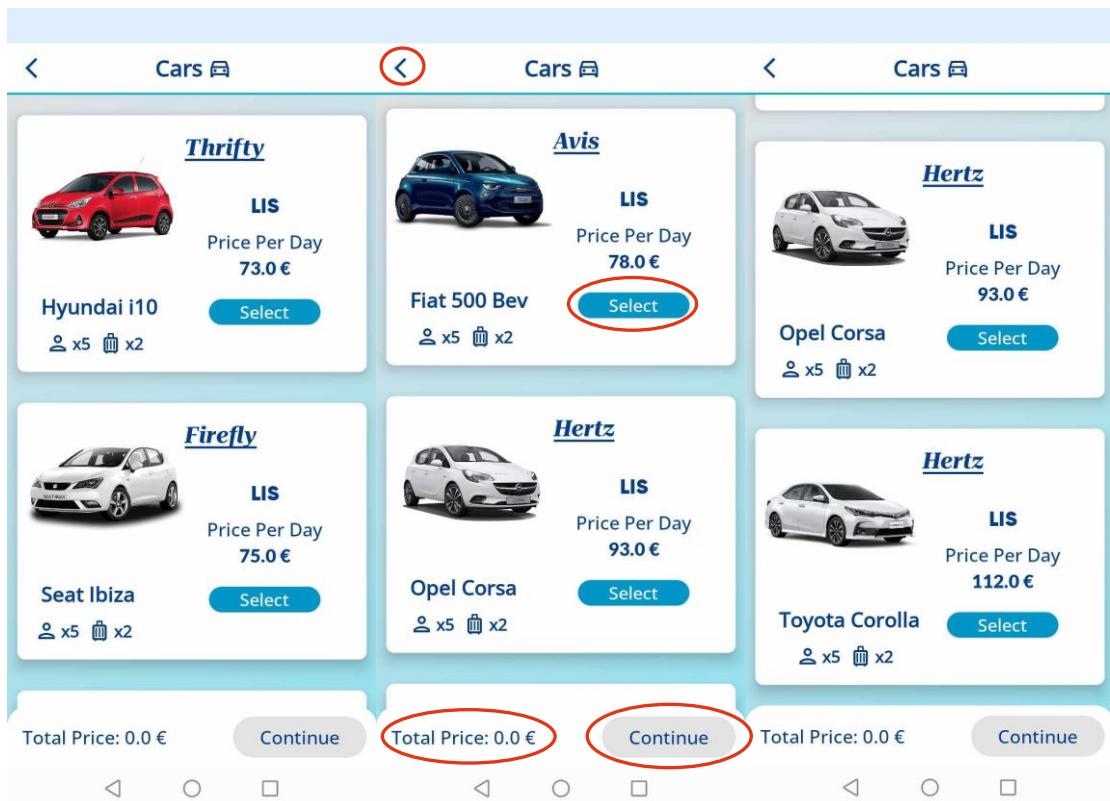
        if(result1.rows.length == 0) {
            check = 1
            res.json([{success: true, successAirport: true, successTime: false}])
        }
    }
}
if(check === 0) {
    res.json([{success: true, successAirport: true, successTime: true}])
}
```

Αν όλα πάνε καλά τρέχει το POST API για αναζήτηση αυτοκινήτων:

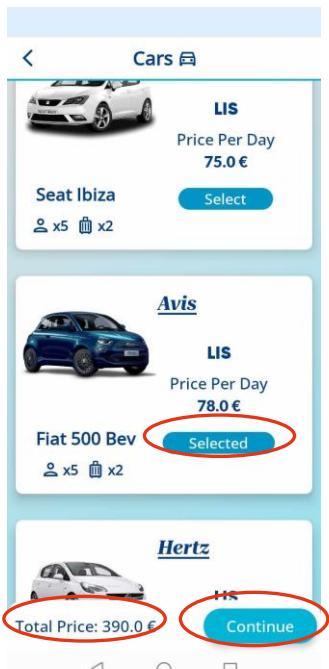
```
app.post('/flynow/cars', async (req, res) => {
    const jsonArray = req.body

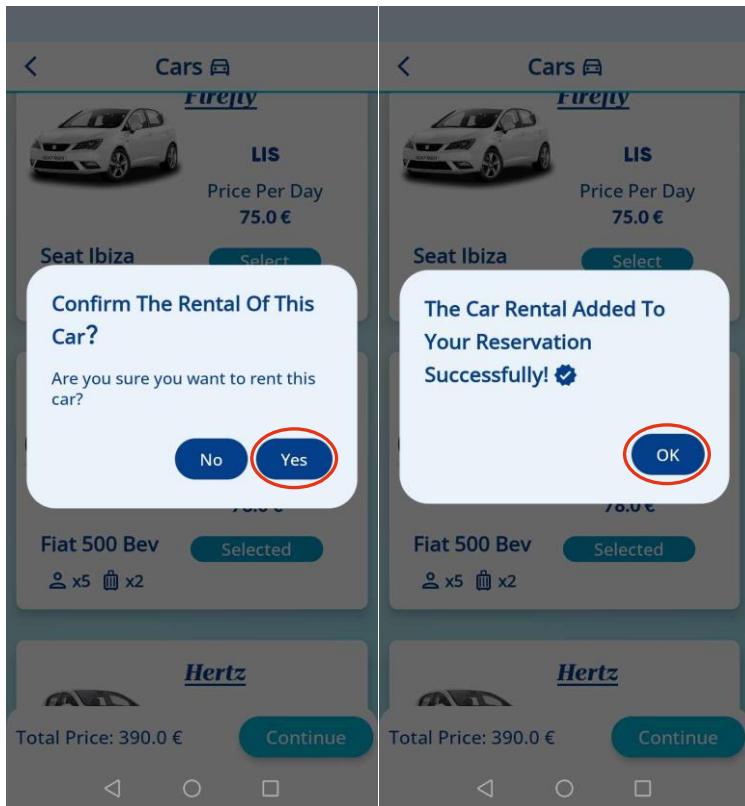
    try {
        const result = await pool
        .query(`

        SELECT c.carimage, c.company, c.model, c.price, c.carid
        FROM car c
        LEFT JOIN contains con ON c.carid=con.carid
        AND con.pickup <= TIMESTAMP '${jsonArray[0].return}'
        AND con.return >= TIMESTAMP '${jsonArray[0].pickUp}'
        WHERE c.location='${jsonArray[0].location}'
        AND con.carid IS NULL
        ORDER BY c.price ASC`)
```



Ο χρήστης έχει την δυνατότητα να πάει πίσω με το back button, να επιλέξει κάποιο αυτοκίνητο που προτιμά και κάτω φαίνεται η τελική τιμή του αυτοκινήτου που είναι η τιμή του αυτοκινήτου για μία μέρα επί τις ημέρες που θα ενοικιαστεί καθώς και το κουμπί "Continue" γίνεται enabled όπως φαίνεται παρακάτω και πατώντας το εμφανίζεται το dialog παρακάτω.





Ο χρήστης πρέπει να πατήσει το κουμπί “Yes” ώστε να ολοκληρωθεί η ενοικίαση του αυτοκινήτου και να του εμφανίσει την επιτυχή ολοκλήρωση και πατώντας το κουμπί “OK” να πάει στο “Home Screen”. Πατώντας το “Yes” button τρέχει το POST API που φαίνεται παρακάτω που εκτελεί insert query.

```
//POST api that inserts the renting of car in a reservation that finally select the user
app.post('/flynow/renting-car', async (req, res) => {
  const jsonArray = req.body

  try {
    await pool.query(`INSERT INTO contains(carid, reservationid, pickup, return, rentingprice)
VALUES(${jsonArray[0].carId}, '${jsonArray[0].bookingId.toUpperCase()}', 
TIMESTAMP '${jsonArray[0].pickUp}',TIMESTAMP '${jsonArray[0].return}', ${jsonArray[0].price});`)

    res.json([{ success: true}])
  } catch (error) {
    console.error(error)
    res.status(500).json({ error: 'Internal Server Error' })
  }
})
```

Στο My Booking η ενοικίαση αυτοκινήτου φαίνεται όπως παρακάτω:

The screenshot displays a mobile application interface for managing bookings. At the top, there is a header bar with a back arrow, the text "My Booking" with a mail icon, and a search icon. Below the header, there are two sections: "Pets" (with a paw icon) and "Car". The "Car" section is expanded, showing a booking for an Avis car. The details are as follows:

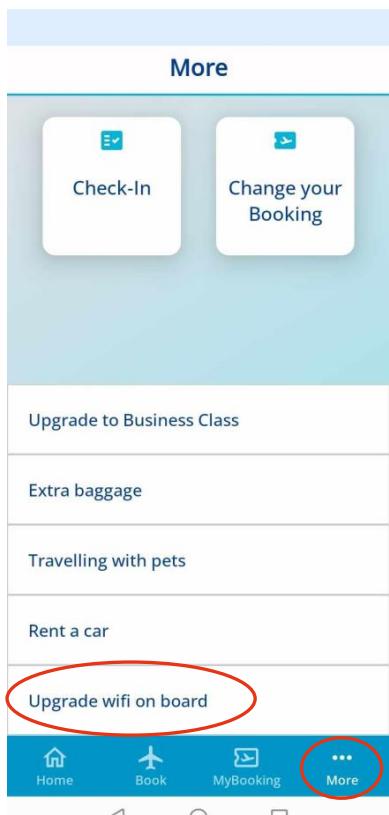
- Avis** (car brand)
- Location LIS** (pick-up location)
- Pick Up** 18/02/2024 17:30
- Return** 23/02/2024 20:00
- Fiat 500 Bev** (car model)
- Passenger**: 5 people (indicated by a person icon and "x5")
- Baggage**: 2 bags (indicated by a suitcase icon and "x2")

Below the car details, there is a section for **€ Price** with the following breakdown:

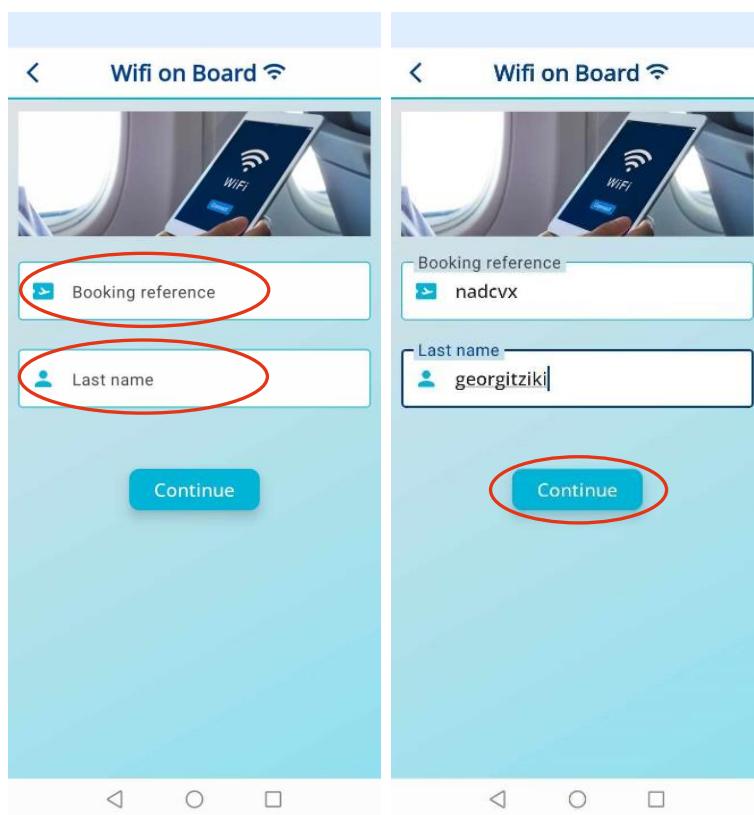
- Booking Price:** 1005.0 €
- Renting Cars Price:** 390.0 €
- Total Price:** 1395.0 €

At the bottom of the screen, there are two buttons: "Home" (blue) and "Delete Your Booking" (red). Navigation icons (back, home, and recent apps) are visible at the very bottom.

UPGRADE WIFI ON BOARD



Για να προσθέσει ο χρήστης κάποιον τύπο ίντερνετ στην κράτηση του, μπορεί κάνοντας κλικ στο κουμπί "Upgrade wifi on Board" να ανακατευθυνθεί στην αντίστοιχη σελίδα όπου θα του εμφανιστούν οι επιλογές για ίντερνετ. Πληκτρολογώντας τον κωδικό της κράτησής και το επώνυμο μας είτε σε κεφαλαία είτε σε μικρά γράμματα στα πεδία "Booking reference" και "Last name" αντίστοιχα, και κάνοντας κλικ στο κουμπί "Continue" τρέχει το ίδιο query με παραπάνω για τον έλεγχο ύπαρξης της κράτησης και ένα query που επιστρέφει τον τύπο ίντερνετ που έχει επιλεχθεί, αν υπάρχει. Εφόσον υπάρχει η κράτηση συνεχίζουμε στην επόμενη σελίδα.



To POST API που καλείται και το query φαίνονται παρακάτω:

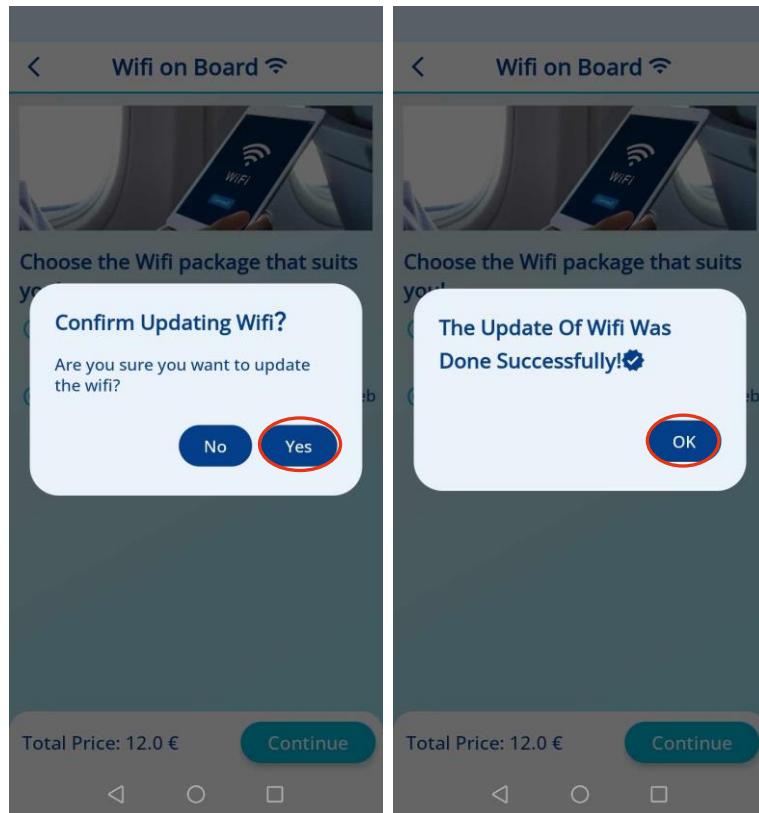
```
//POST api that returns the value from the wifionboard field of the database for the wifi on board screen
app.post('/flynow/wifi-on-board', async (req, res) => {
  const jsonArray = req.body

  try {
    const result = await pool
      .query(`SELECT r.wifionboard
              FROM reservation r
              WHERE r.reservationid='${jsonArray[0].bookingid.toUpperCase()}'`)

    res.json([{ wifiOnBoard: result.rows[0].wifionboard }])
  } catch (error) {
    console.error(error)
    res.status(500).json({ error: 'Internal Server Error' })
  }
})
```



Οι επιλογές ίντερνετ φαίνονται στην σελίδα “Wifi on Board” και οι τιμές διαφοροποιούνται ανάλογα με την παροχή του κάθε πακέτου ίντερνετ. Διαλέγοντας το ακριβότερο πακέτο, και πατώντας “Continue” εμφανίζεται το μήνυμα επιβεβαίωσης αγοράς ίντερνετ και η ανανέωση της κράτησης ολοκληρώνεται καθώς τρέχει το ανάλογο query.



To POST API και το query που τρέχει όταν ο χρήστης πατήσει "Yes":

```
//POST api that updates the wifionboard field of the database that select the user from the wifi on board screen
//+ the new price of the reservation that is priceOld + priceNew of the new selection of the wifi on board
app.post('/flynow/update-wifi', async (req, res) => {
  const jsonArray = req.body

  try {
    const result = await pool
      .query(`UPDATE reservation
              |   SET wifionboard=${jsonArray[0].wifionBoard}, price= price + ${jsonArray[0].price}
              |   WHERE reservationid='${jsonArray[0].bookingid.toUpperCase()}'`)

    res.json([{ success: true }])
  } catch (error) {
    console.error(error)
    res.status(500).json({ error: 'Internal Server Error' })
  }
})
```



Εδώ φαίνεται πως αφού ο χρήστης προηγουμένως έκανε update στο ακριβότερο πακέτο ίντερνετ, δεν μπορεί να κάνει περαιτέρω αναβάθμιση.

Wifi on Board
Audio/Video streaming,
High speed web browsing & Social Media,
up to 15Mbps

Pets
Pet Size: Large (>25kg)

Car

Avis
Location LIS
Fiat 500 Bev
Pick Up 18/02/2024 17:30
Return 23/02/2024 20:00
Σ x5 Ε x2

€ Price

Πηγαίνοντας πάλι στην σελίδα "My Booking" βλέπουμε ότι το πεδίο "Wifi on Board" έχει ανανεωθεί από "No Wifi" που έγραφε πριν στο νέο πακέτο που επιλέχθηκε.

9. ΒΙΒΛΙΟΓΡΑΦΙΑ

Για την υλοποίηση της εργασίας χρησιμοποιήθηκαν τα links που ακολουθούν παρακάτω:

- Για την δομή της ΒΔ και ιδέες σχετικά με το Θέμα:
<https://en.aegeanair.com/>

https://www.esky.gr/?gclid=CjwKCAiA8OmdBhAgEiwAShr409hyN5X0GvE8WDCMSwhLLxZwhRK_RoT20pNX1PDuXcNjU KbHuxepjRoCJksQAvD_BwE

- Για την εκμάθηση του Jetpack Compose του Android Studio σε Kotlin βοήθησε η παρακάτω σειρά μαθημάτων-βίντεο:

<https://www.youtube.com/watch?v=cDabx3SjuOY&list=PLQkwcJG4YTCSpJ2NLhDTHi6XBNfk9WiC>