

# Proiect Baze de Date II

Companie Aeriană

Catrangiu Radu-Ovidiu

343C1

# Cuprins

<b>Cuprins</b>	<b>1</b>
<b>Descrierea Temei</b>	<b>2</b>
<b>Descrierea Bazei de Date</b>	<b>2</b>
Diagrama Bazei de Date	2
Structura Tabelelor	3
Descrierea Constrângerilor de Integritate	4
Descrierea procedurilor și funcțiilor	5
<b>Descrierea Aplicației</b>	<b>7</b>
Diagrama de Clase	7
Structura Claselor	7
Workflow	10
Modul de Conectare la Baza de Date	10
<b>Capturi de ecran</b>	<b>11</b>
Pagina de Sign in	11
Pagina de Sign up	11
Pagina de căutare a zborului	12
Pagina rezervării	13
Pagina de administrare și statistici	14
<b>Concluzii</b>	<b>17</b>
<b>Bibliografie</b>	<b>17</b>

# Descrierea Temei

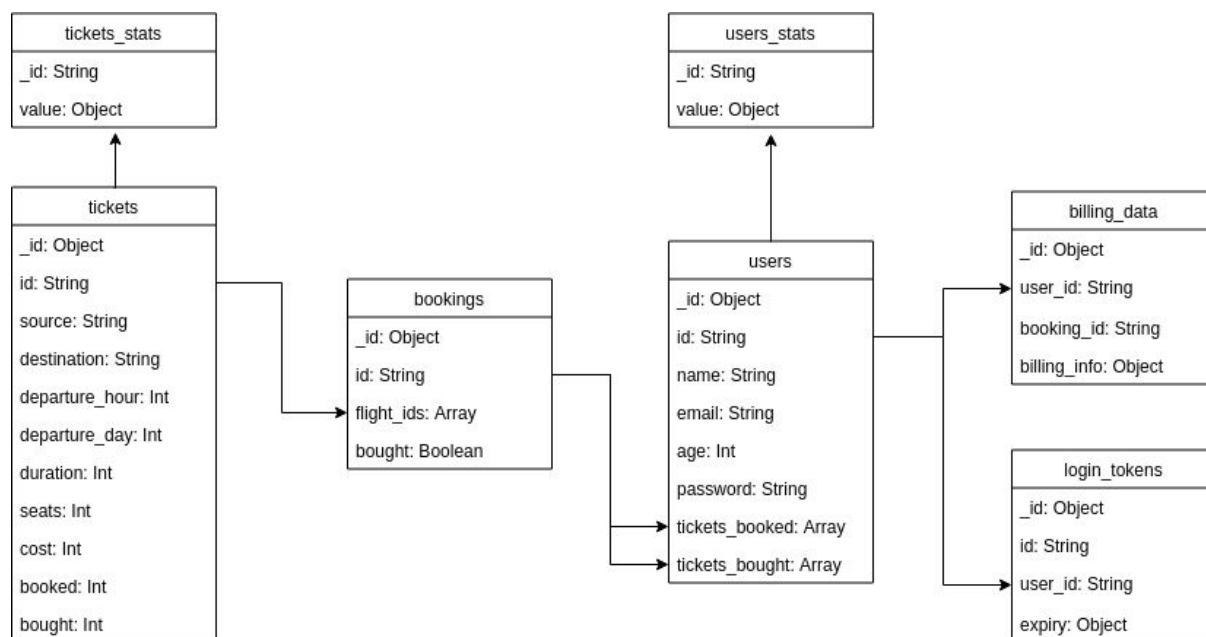
Tema reprezintă o companie de transport aerian. Este implementată în JavaScript folosind NodeJS și MongoDB pentru backend, respectiv Vue.JS și Bootstrap pentru frontend. Serverul de NodeJS expune un API pentru accesarea bazei de date și răspunde la cereri HTTP de tip GET pentru afișarea interfeței grafice.

Pagina de întâmpinare este o pagină de login sau de creare de cont. După autentificare utilizatorul este redirecționat către */client*, unde poate să introducă anumite criterii de căutare a unui zbor, să rezerve zboruri și să vizualizeze toate zborurile rezervate și cumpărate. După rezervarea unui zbor utilizatorul primește un link către o pagină unde poate să vizualizeze detaliile rezervării și să cumpere ulterior biletul sau biletele de avion.

Pagina de administrare afișează toate zborurile, oferă posibilitatea de ștergere sau de adăugare a unor noi zboruri și oferă date statistice legate de Utilizatori și Zboruri.

## Descrierea Bazei de Date

Diagrama Bazei de Date



## Structura Tabelelor

### **tickets:**

- `_id`: Index generat de MongoDB
- `id`: String unic generat programatic (uuidv4)
- `source`: Sursa zborului.
- `destination`: Destinația zborului.
- `departure_hour`: Ora din zi în care are loc zborul.
- `departure_day`: Ziua din an în care are loc zborul.
- `duration`: Durata în ore a zborului
- `seats`: Numărul de locuri total disponibil în cadrul zborului.
- `booked`: Numărul de bilete rezervate pentru zborul curent.
- `bought`: Numărul total de bilete cumpărate pentru zborul curent.
- `cost`: Prețul unui bilet.

### **bookings:**

- `_id`: Index generat de MongoDB
- `id`: String unic generat programatic (uuidv4)
- `flight_ids`: Array de id-uri ale biletelor ce trebuie cumpărate pentru a zbura de la sursă la destinație. Zboruri cu escală.
- `bought`: Boolean necesar pentru a genera corect pagina de booking astfel încât să nu poată fi cumpărat de două ori.

### **users:**

- `_id`: Index generat de MongoDB
- `id`: String unic generat programatic (uuidv4)
- `name`: Numele utilizatorului
- `email`: Email-ul utilizatorului
- `age`: Vârsta utilizatorului
- `password`: Parola utilizatorului. Stocată în clar în această implementare rudimentară. Altfel ar trebui adăugat ``salt`` și criptată, apoi stocat hash-ul.
- `tickets_booked`: Array de obiecte unde un obiect conține date despre un zbor rezervat.
- `tickets_bought`: Array de obiecte unde un obiect conține date despre un zbor cumpărat.

**billing\_data:**

- `_id`: Index generat de MongoDB
- `user_id`: ID-ul userului ale cărui date de cumpărare sunt stocate în `billing_info`.
- `booking_id`: ID-ul rezervării pentru care sunt folosite datele din `billing_info` pentru a efectua o plată.
- `billing_info`: Datele cardului utilizat pentru a efectua plata de către utilizator.

**login\_tokens:**

- `_id`: Index generat de MongoDB
- `id`: String unic generat programatic (uuidv4). Reprezintă cookie-ul ce facilitează autentificarea unui utilizator.
- `user_id`: ID-ul userului ce este autentificat.
- `expiry`: Obiect de tip `Date` utilizat de MongoDB pentru a șterge documentul din colecție după 3600 de secunde de la creare.

## Descrierea Constrângerilor de Integritate

Pentru implementarea constrângerilor de integritate am folosit un validator în instanța de MongoDB în NodeJS. Acest validator primește o schemă JSON pe care am configurat-o folosind documentația oficială. Această configurație poate fi găsită în fișierul *config.js*.

**tickets:**

- `id`: String unic
- `source`: String de lungime minimă 2
- `destination`: String de lungime minimă 2
- `departure_hour`: Integer în intervalul [0, 23]
- `departure_day`: Integer în intervalul [1, 365]
- `duration`: Integer mai mare sau egal cu 1
- `seats`: Integer
- `booked`: Integer
- `bought`: Integer
- `cost`: Integer mai mare sau egal cu 1

**bookings:**

- `id`: String unic
- `flight_ids`: Array ce conține numai obiecte de tip String
- `bought`: Boolean

**users:**

- id: String unic
- name: String de lungime minimă 3
- email: String unic de lungime minimă 3
- age: Integer
- password: String de lungime minimă 3
- tickets\_booked: Array ce conține numai obiecte. Un obiect conține următoarele proprietăți: 'ticket' : String, 'flights': Integer, 'cost': Integer
- tickets\_bought: Array ce conține numai obiecte. Un obiect conține următoarele proprietăți: 'ticket' : String, 'flights': Integer, 'cost': Integer

**billing\_data:**

- user\_id: String
- booking\_id: String
- billing\_info: Obiect de forma: 'card\_number': String de lungime minima 2, 'name': String de lungime minima 2, 'expiry\_month': Integer in intervalul [1, 12], 'expiry\_year': Integer in intervalul [18, 40], 'card\_cvv': String de fix 3 caractere.

**login\_tokens:**

- id: String unic
- user\_id: String
- expiry: Obiect

## Descrierea procedurilor și funcțiilor

### Funcții stocate pe server

**get\_tickets\_stats\_description():**

Întoarce o descriere pentru tipul de statistică. *(folosită în mapReduce)*

**trunct\_two\_decimals():**

Truncchează o valoare de tip double la două zecimale. *(folosită în mapReduce)*

**compute\_avg():**

Calculează valoarea medie a costului pentru cheile dintr-un obiect. *(folosită în funcția finalize din mapReduce)*

## Funcții API

### **/admin - insert():**

Inserează un nou zbor în colecția *tickets*.

### **/admin - delete():**

Șterge un zbor din colecția *tickets*.

### **/admin - list():**

Întoarce toate zborurile din colecția *tickets*.

### **/stats - update():**

Apelează două funcții **mapReduce** în funcție de parametrul *collection*. Dacă are valoarea *tickets* funcția rulează mapReduce pe colecția *tickets*, altfel rulează mapReduce pe colecția *users*. Prima variantă generează date în colecția *tickets\_stats*, iar a doua în colecția *users\_stats*.

### **/stats - list():**

În funcție de parametrul *collection* întoarce datele din colecțiile *tickets\_stats* sau *users\_stats*.

### **/account - login():**

Caută userul în colecția *users* după email, verifică corectitudinea parolei, generează un token și îl adaugă în colecția *login\_tokens*.

### **/account - create():**

Inserează un user nou în colecția *users* apoi apelează funcția *login()*.

### **/account - check\_token():**

Caută un token primit ca parametru în colecția *login\_tokens*. Dacă îl găsește întoarce răspuns "Valid token", altfel eroare "Invalid token".

### **/client - get\_user\_info():**

Caută și întoarce datele stocate în colecția *users* despre utilizatorul care face apelul. Sunt omise câmpurile "password", "\_id" și "id".

### **/client - get\_optimal\_route():**

Caută în colecția *tickets* și întoarce zborurile care satisfac condițiile utilizatorului.

### /client - book\_ticket():

Caută zborurile din lista primită ca parametru în colecția *tickets*, le incrementează proprietatea *booked*, apoi inserează în colecția *bookings* noua rezervare și inserează în proprietatea de tip array *tickets\_booked* a utilizatorului datele despre rezervare (id, număr de zboruri, cost total).

### /client - buy\_ticket():

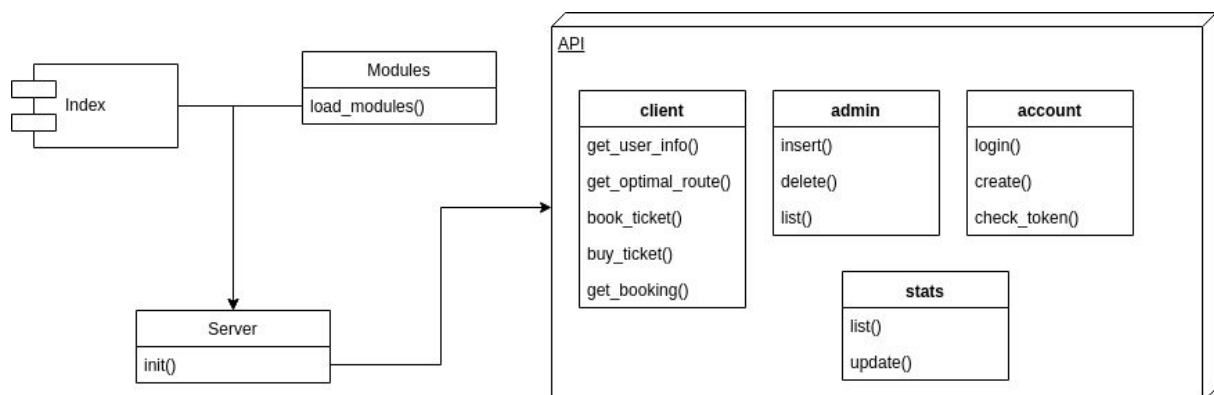
Inserează în colecția *billing\_data* datele cardului utilizatorului. Caută în colecția *bookings* rezervarea. Pentru toate zborurile din rezervare actualizează proprietățile *booked* și *bought*, decrementând-o pe prima și incrementând-o pe a doua. Actualizează rezervarea din colecția *bookings*, setând proprietatea *bought* ca *true*. Caută userul cumpărător în colecția *users*. Caută datele rezervării în datele utilizatorului. Se actualizează documentul utilizatorului din colecția *users* eliminând datele rezervării din proprietatea array *tickets\_booked* și adăugându-le în proprietatea array *tickets\_bought*.

### /client - get\_booking():

Folosind codul rezervării primit ca parametru caută în colecția *bookings* rezervarea, apoi caută în colecția *tickets* datele despre zborul/zborurile din rezervare și le întoarce.

## Descrierea Aplicației

### Diagrama de Clase





## Structura Claselor

În index se apelează clasa Modules prin metoda `load_modules()` pentru a inițializa modulele de mediu (environment), mai precis colecțiile specificate în config și funcția care identifică utilizatorul după token la fiecare apel de API.

### Modules

Citește fișierul de configurare și inițializează conexiunea la baza de date. Creează colecțiile adăugând opțiuni pentru constrângeri și pentru fiecare colecție apelează `createIndex()` dacă este specificat în fișierul de configurare. În cazul colecției *login\_tokens* este setat indexul *expiry* astfel încât fiecare document să fie șters automat în 3600 de secunde.

De asemenea, clasa Modules inițializează și funcția de autentificare care primește un token, mai precis cookie-ul cu care este apelat API-ul, și verifică în colecția *login\_tokens* dacă există acest token și cui îi aparține, setând în environment parametrul *user\_id*, astfel fiecare funcție din API să aibă acces la *user\_id*-ul apelantului.

### Server

Serverul expune o funcție de inițializare care primește următorii parametri: *port*, *rpc\_config*, *modules* și *callback*. *Port*-ul este cel din configurație, *port*-ul pe care va porni serverul, *rpc\_config* este un obiect în care sunt definite fișierele în care se află funcțiile (handler), dacă serviciile sunt accesate prin autentificare (*useAuth*) și calea fiecărui serviciu din API. *Modules* este obiectul cu modulele inițializate. *Callback* este funcția apelată la finalul inițializării. De asemenea, în server sunt inițializate și toate handler-urile pentru cererile de tip GET.

### Client

Clasa *client* expune 5 funcții publice și 2 funcții private:

- ***get\_user\_info*** : nu primește niciun parametru și caută în baza de date informații despre userul apelant și le întoarce.
- ***get\_optimal\_route*** : primește ca parametri datele de căutare introduse de user, caută în baza de date toate zborurile și calculează zborul optim folosind funcția privată *find\_optimal\_route*.
- ***book\_ticket*** : primește ca parametru lista de zboruri ce trebuiesc rezervate. Actualizează toate zborurile din listă incrementându-le

proprietatea *booked*. Inserează în colecția *bookings* noua rezervare și adaugă în lista de rezervări ale userului noua rezervare.

- **buy\_ticket**: primește ca parametru id-ul rezervării și informațiile necesare pentru efectuarea plății. Actualizează toate zborurile din listă incrementându-le proprietatea *bought* și decrementându-le proprietatea *booked*. Marchează rezervarea în baza de date ca fiind cumpărată și o elimină din lista de rezervări ale userului, adăugând-o în lista de bilete cumpărate de către user.
- **get\_booking**: primește ca parametru id-ul rezervării. Caută în baza de date rezervarea, apoi toate zborurile menționate în rezervare și întoarce datele despre aceste zboruri.
- **find\_optimal\_route**: este funcția care calculează zborurile necesare pentru a ajunge de la sursă la destinație.
- **compute\_distance**: este o funcție ce calculează metrica pentru sortarea zborurilor în *find\_optimal\_route*.

## Admin

Clasa *admin* expune 3 funcții publice:

- **insert**: funcție care primește ca parametru datele unui nou zbor ce este introdus în baza de date.
- **delete**: funcție care primește ca parametru id-ul unui zbor ce este șters din baza de date.
- **list**: funcție care nu primește niciun parametru și care întoarce toate zborurile.

## Account

Clasa *account* expune 3 funcții publice:

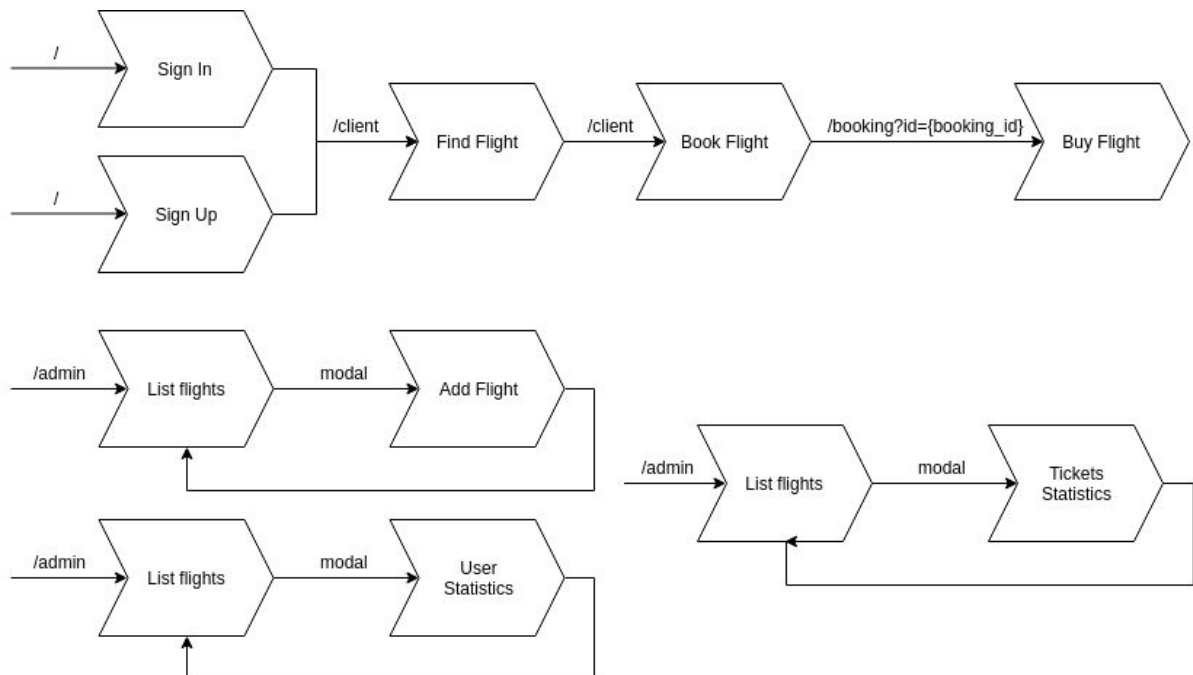
- **login**: funcție care primește ca parametru un email și o parolă, caută în baza de date după email, dacă găsește user-ul verifică să coincidă parola, apoi creează un token de login pe care îl introduce în colecția *login\_tokens*.
- **create**: funcție care primește ca parametru numele, email-ul, vârsta și parola cu care creează un utilizator nou în colecția *users*. După introducerea noului utilizator în baza de date este apelată funcția *login*.
- **check\_token**: funcție care primește ca parametru un token pe care îl caută în baza de date pentru a-i determina validitatea.

## Stats

Clasa *stats* expune 2 funcții publice și 2 funcții private:

- **update**: funcție care primește un parametru în funcție de care apelează una dintre funcțiile private *update\_users\_stats* sau *update\_tickets\_stats*.
- **list**: funcție care primește un parametru în funcție de care listează conținutul uneia dintre colecțiile *tickets\_stats* sau *users\_stats*.
- *update\_users\_stats*: funcție care apelează **mapReduce** pentru a genera un raport în colecția *users\_stats*.
- *update\_tickets\_stats*: funcție care apelează **mapReduce** pentru a genera un raport în colecția *tickets\_stats*.

## Workflow



## Modul de Conectare la Baza de Date

La inițializare serverul se conectează la baza de date folosind un URL de forma ``mongodb://${user}:${password}@${login_url}``. De asemenea se specifică baza de date la care se dorește conectarea și colecțiile din baza de date respectivă, pentru a fi inițializate, cu constrângerile specificate în fișierul de configurare, în cazul în care nu există.

# Capturi de ecran

## Pagina de **Sign in**

### Sign in or Sign up

Sign in [Sign up](#)

#### SIGN IN

## Pagina de Sign up

### Sign in or Sign up

[Sign in](#) Sign up

#### SIGN UP

You will receive an email to verify your account.

## Pagina de căutare a zborului

[LOGOUT](#)[Radu Catrangiu](#)

Search for a flight!

**BUCURESTI to SIBIU**  
Departure hour: 8  
Flight duration: 1

Departure day: 1  
Remaining seats: 25

**SIBIU to BARCELONA**  
Departure hour: 8  
Flight duration: 3

Departure day: 3  
Remaining seats: 23

**BUCURESTI to SIBIU**  
Departure hour: 8  
Flight duration: 1

Departure day: 1  
Remaining seats: 25

**SIBIU to BARCELONA**  
Departure hour: 8  
Flight duration: 3

Departure day: 3  
Remaining seats: 23

/booking?id=d4c951b1-2e94-42fd-90fd-ef22f6a25fdd

[LOGOUT](#)[Radu Catrangiu](#)

Source

Departure Day

Tickets

Booked

BUCURESTI to BARCELONA

Bought

BARCELONA to SIBIU

## Pagina rezervării

ID: a0d5f316-9732-44dc-a00b-239327548820

### BUCURESTI to SIBIU

Price: 163

Departure hour: 8  
Flight duration: 1

Departure day: 1  
Remaining seats: 25

### SIBIU to BARCELONA

Price: 157

Departure hour: 8  
Flight duration: 3

Departure day: 3  
Remaining seats: 23

Buy tickets

ID: a0d5f316-9732-44dc-a00b-239327548820

### BUCURESTI to SIBIU

Price: 163

Departure hour: 8  
Flight duration: 1

Departure day: 1  
Remaining seats: 25

### SIBIU to BARCELONA

Price: 157

Departure hour: 8  
Flight duration: 3

Departure day: 3  
Remaining seats: 23

Buy tickets

#### Buy tickets

×

Card Number

Name On Card

Expiry Month

Expiry Year

CVV

Close

Buy

ID: a0d5f316-9732-44dc-a00b-239327548820

### BUCURESTI to SIBIU

Price: 163

Departure hour: 8  
Flight duration: 1

Departure day: 1  
Remaining seats: 25

### SIBIU to BARCELONA

Price: 157

Departure hour: 8  
Flight duration: 3

Departure day: 3  
Remaining seats: 23

Tickets bought!

## Pagina de administrare și statistici

User Statistics	Tickets Statistics	New FLIGHT	
<b>BUCURESTI to SIBIU</b> Departure hour: 8 Flight duration: 1 Tickets Booked: 5 Price: 163		<b>TIMISOARA to BARCELONA</b> Departure hour: 15 Flight duration: 1 Tickets Booked: 0 Price: 199	
Delete Flight		Delete Flight	
<b>BUCURESTI to CLUJ</b> Departure hour: 9 Flight duration: 1 Tickets Booked: 0 Price: 229		<b>CLUJ to TIMISOARA</b> Departure hour: 12 Flight duration: 1 Tickets Booked: 0 Price: 122	
Delete Flight		Delete Flight	
<b>SIBIU to BARCELONA</b> Departure hour: 8 Flight duration: 3 Tickets Booked: 4 Price: 157		<b>BUCURESTI to PRAGA</b> Departure hour: 13 Flight duration: 0 Tickets Booked: 0 Price: 238	
Delete Flight		Delete Flight	
<b>BARCELONA to SIBIU</b> Departure hour: 9 Flight duration: 2		<b>PRAGA to LONDRA</b> Departure hour: 10 Flight duration: 6	
Departure day: 1 Flight capacity: 25 Tickets Bought: 8		Departure day: 1 Flight capacity: 12 Tickets Bought: 0	
Departure day: 3 Flight capacity: 23 Tickets Bought: 6		Departure day: 321 Flight capacity: 5 Tickets Bought: 0	
Departure day: 2 Flight capacity: 24		Departure day: 320 Flight capacity: 21	

### Insert new FLIGHT

Source

Destination

Departure Hour

Departure Day

Flight Duration

Number of seats

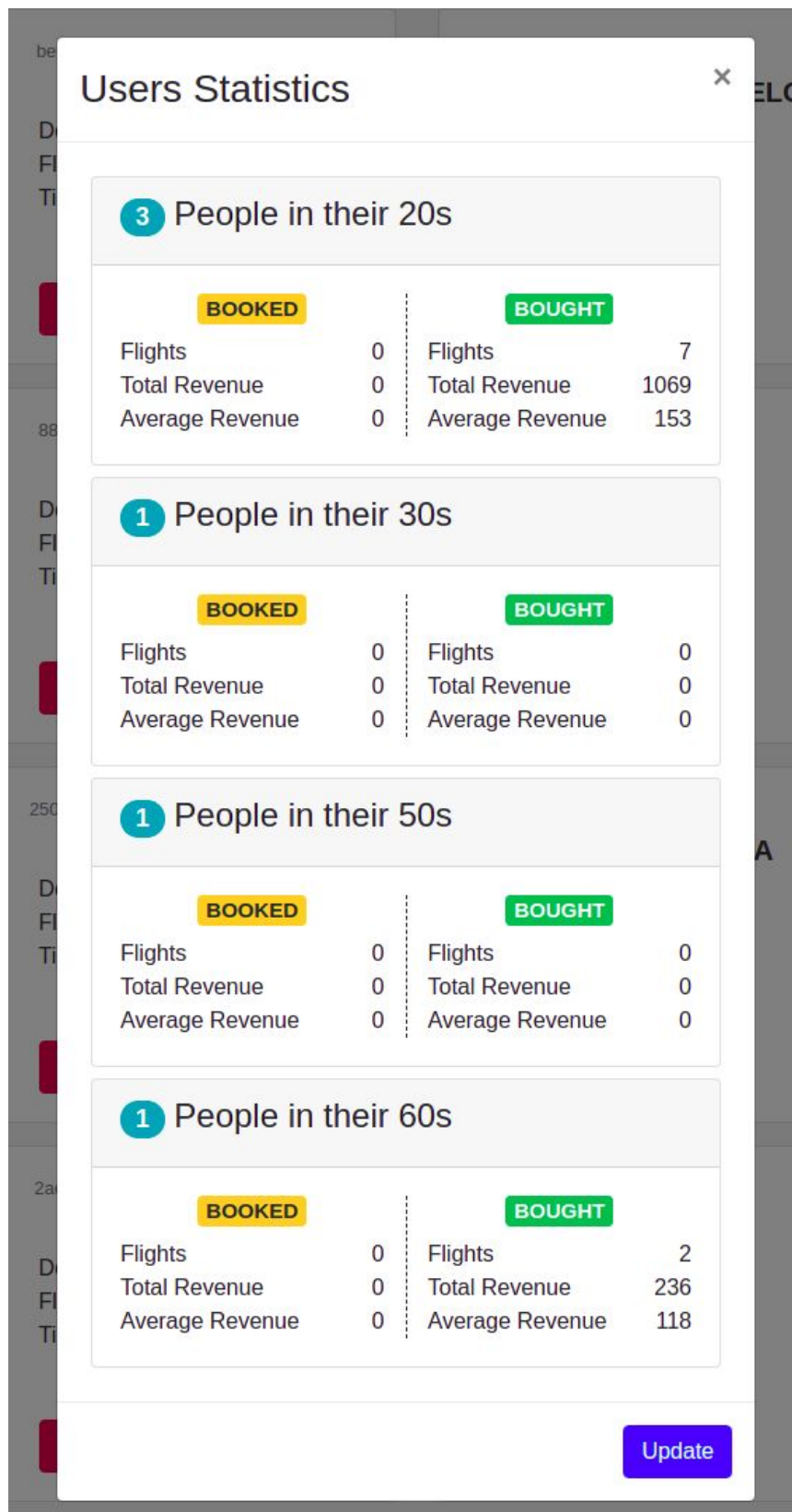
Price

☐ Clear input after use

Cancel

Create

Departure day: 3  
Flight capacity: 23  
Tickets Bought: 6Departure hour: 13  
Flight duration: 0  
Tickets Booked: 0  
Price: 238





## Raport Bilete

Tickets Statistics

TICKET COST

Minimum	65
Maximum	238
Average	160.22

REVENUE IF ALL CURRENT BOOKINGS ARE BOUGHT

Minimum	0
Maximum	489
Average	54.27

REVENUE IF AIRPLANE FULL

Minimum	588
Maximum	4075
Average	2290.27

CURRENT REVENUE

Minimum	0
Maximum	1141
Average	230.72

Update

## Concluzii

Proiectul a fost unul util deoarece nu există aplicație din domeniu care să nu folosească cel puțin o bază de date, iar ca programator trebuie să interacționezi cu acestea măcar o dată pentru a te familiariza cu modul de lucru.

## Bibliografie

<https://docs.mongodb.com/>

<https://getbootstrap.com/docs/4.2/>

<https://developer.mozilla.org/ro/docs/Web/JavaScript>

<https://vuejs.org/v2/guide/>