TAW system architecture

This document describes the system architecture of the Technology and Web Applications project.

Written by Raffaele Cuniolo.

26 June 2025

GitHub Repository: github.com/CunioloRaffaele/TAW

Index

Docker Containers

The entire application is orchestrated using Docker Compose, which defines and manages multiple containers:

1. **Database Container**
   - **Image**: postgres:latest
   - **Initialization**: Loads schema from airline_database_schema.sql
   - **Exposed Port**: 5431 (host) → 5432 (container)
2. **Backend Container**
   - **Image**: node:22-alpine + chromium headless for puppeteer
   - **Runs**: Node.js/Express server
   - **Exposed Port**: 3000 (host) → 3000 (container)
   - **Depends on**: Database container (waits for DB to be ready)
   - **Entrypoint**: npm run easyStart (starts the server with prisma client initialization)
3. **Frontend Container**
   - **Image**: Multi-stage: builds Angular app with node:22-alpine, then serves the built app with Nginx using nginx:alpine
   - **Exposed Port**: 80 (host) → 80 (container)
   - **Serves**: Static Angular app

All containers are on a shared Docker network, allowing the backend to connect to the database using the service name db. In a production enviroment, the frontend would be served by a reverse proxy (e.g., Nginx) to handle SSL termination and static file serving and the other containers would be behind a firewall, only exposing the frontend ports to the outside world.

DevOps & Automation

To ease the development process, we have implemented a few DevOps practices:

1. **Docker Compose**: Simplifies multi-container management, allowing developers to start the entire application stack with a single command (`docker-compose up`).
2. **Dev Server**: A remote Ubuntu development server was set up to automatically restart the backend when changes are made in the source code productivity, in order to improve productivity for the front-end devs. The server is hosted on a Oracle Cloud VM instance and is accessible via SSH.
   - `deployScript.sh` : Automates pulling the latest code from GitHub and restarting containers
3. **Prisma Client**: Automatic generation of the Prisma client based on the database schema, ensuring type safety and easy database access (and query) for the backend. The backend offers a script in the package.json to automate the prisma client generation and schema update, before starting the server:
   - `npm run easyStart` : Runs `npm run prisma:pull && npm run prisma:generate` before starting the Express server.
4. **Postman Collection**: A Postman collection is provided to test the API endpoints easily, allowing to quickly understand and interact with the backend services.

Database Layer

The airline database schema is designed to manage information related to airlines, airports, aircrafts, flights, bookings, and users. Below is a detailed explanation of each table and its purpose:

Tables

1. **blJWTs:**
   - This table is used to store blacklisted JSON Web Tokens (JWTs) to prevent the use of invalidated tokens.
   - id: Serial primary key.
   - jwt: VARCHAR(255) storing JSON Web Tokens, used for blacklisting invalidated tokens.
2. **Airlines:**
   - name: VARCHAR(255) primary key, representing the airline's name.
   - password: VARCHAR(255) storing the hashed password for airline authentication.
   - country: VARCHAR(100) indicating the airline's country of origin.
   - motto: VARCHAR(300) with a default motto.
   - enrolled: BOOLEAN flag, defaulting to FALSE, possibly indicating enrollment status.
3. **Airports:**
   - id: Serial primary key.
   - name: VARCHAR(100) representing the airport's name.
   - city: VARCHAR(255) where the airport is located.
   - country: VARCHAR(100) of the airport.

    - lat: DOUBLE PRECISION for latitude.
    - lan: DOUBLE PRECISION for longitude.
    - time_zone: VARCHAR for the airport's time zone.
4. **Aircrafts:**
    - id: Serial primary key.
    - model: VARCHAR(100) with a default model 'Boeing 747'.
    - seats_capacity: INTEGER defining the number of seats, with a CHECK constraint ensuring it's between 25 and 750.
    - owner_name: VARCHAR(255) acting as a foreign key referencing Airlines(name), with ON UPDATE CASCADE and ON DELETE CASCADE actions.
5. **Routes:**
    - departure: INTEGER foreign key referencing Airports(id).
    - destination: INTEGER foreign key referencing Airports(id).
    - CHECK (departure <> destination): Ensures departure and destination airports are different.
    - PRIMARY KEY (departure, destination): Composite primary key.
6. **Uses:**
    - id: Serial primary key.
    - airline_name: VARCHAR(255) foreign key referencing Airlines(name).
    - route_departure: INTEGER part of a composite foreign key referencing Routes(departure, destination).
    - route_destination: INTEGER part of a composite foreign key referencing Routes(departure, destination).
    - Foreign key constraints include ON UPDATE CASCADE and ON DELETE CASCADE.
7. **Flights:**
    - code: UUID primary key, defaulting to a randomly generated UUID.
    - duration: INTEGER in minutes, with a CHECK constraint for duration between 90 and 1140 minutes.
    - aircraft_id: INTEGER foreign key referencing Aircrafts(id).
    - liftoff_date: TIMESTAMP indicating the flight's departure time (utc format).
    - route_departure: INTEGER part of a composite foreign key referencing Routes.
    - route_destination: INTEGER part of a composite foreign key referencing Routes.
    - airline_name: VARCHAR(255) foreign key referencing Airlines(name).
    - Foreign key actions are ON UPDATE CASCADE and ON DELETE SET NULL.
8. **Seats:**
    - id: Serial primary key.
    - postion: VARCHAR(4) for seat position (e.g., 'A1'), with a CHECK constraint ensuring at least two characters.
    - aircraft_id: INTEGER foreign key referencing Aircrafts(id), with ON UPDATE CASCADE and ON DELETE CASCADE actions.
9. **Users:**
    - id: Serial primary key.
    - name: VARCHAR(255) for the user's name.
    - email: VARCHAR(255) unique email address, with a CHECK constraint for a valid email format.
    - password: VARCHAR(255) storing the hashed password.
    - role: INTEGER defaulting to 0, with a CHECK constraint limiting values to 0 or 1 (e.g., 0 for regular user, 1 for admin).
10. **Tickets:**
    - code: UUID primary key, defaulting to a randomly generated UUID.
    - type: VARCHAR(50) for ticket class (e.g., 'ECONOMY', 'BUSINESS'), with a CHECK constraint for allowed types.
    - price: DOUBLE PRECISION with a CHECK constraint ensuring a positive price.
    - fligt_code: UUID foreign key referencing Flights(code), with ON UPDATE CASCADE and ON DELETE CASCADE actions.
11. **Trips:**
    - id: Serial primary key.
    - creation_date: TIMESTAMP when the trip was created.
    - user_id: INTEGER foreign key referencing Users(id), with ON UPDATE CASCADE and ON DELETE CASCADE actions.
12. **Extras:**
    - id: Serial primary key.
    - description: VARCHAR(255) describing the extra service.
    - price: DOUBLE PRECISION with a CHECK constraint ensuring a positive price.
13. **Bookings:**
    - id: Serial primary key.
    - ticket_code: UUID foreign key referencing Tickets(code).
    - seat_id: INTEGER foreign key referencing Seats(id).
    - trip_id: INTEGER foreign key referencing Trips(id).
    - extras_id: INTEGER foreign key referencing Extras(id).
    - Foreign key actions are ON UPDATE CASCADE and ON DELETE SET NULL.

Functions and Triggers

1. `Create_trigger_function_if_not_exists(func_name text, func_body text)` :

   Purpose: This is a generic helper function that checks if a trigger function with the given func_name already exists in the pg_proc catalog. If the function does not exist, it dynamically creates it using the provided func_body. This prevents errors when the schema might be applied multiple times, as CREATE FUNCTION IF NOT EXISTS syntax is not directly available for trigger functions.

2. `filter_zombies()` :

   Trigger: unused_booking AFTER INSERT OR UPDATE ON Bookings. **Purpose**: This function is designed to remove "zombie" Bookings records. A booking is considered a zombie if it does not have an associated seat_id, trip_id, or ticket_code. When a booking is inserted or updated and these three foreign key columns are all NULL, the trigger fires, and the Bookings record is deleted. The function returns NULL for INSERT operations to effectively cancel the insertion of the row if it meets the zombie criteria.

3. `generate_seats()` :

   Trigger: seats_generator AFTER INSERT ON Aircrafts. **Purpose**: This function automates the creation of seat entries in the Seats table whenever a new Aircraft record is inserted. For each new aircraft, it iterates up to the seats_capacity of that aircraft. It calculates a seat_position (e.g., 'A1', 'B2') by assigning a row letter (starting from 'A') and a seat number within that row (up to seats_per_row, which is set to 8). This ensures that every newly added aircraft automatically populates its corresponding seats.

4. `limit_bookings_per_trip()` :

   Trigger: limit_bookings_per_trip_trigger BEFORE INSERT OR UPDATE ON Bookings. **Purpose**: This function enforces a business rule that restricts the number of Bookings associated with a single Trip to a maximum of two. **Behavior on INSERT**: Before a new Bookings record is inserted, the function checks if the trip_id for the new booking would result in more than two existing bookings for that trip. If the count is already 2 or more, an exception is raised, preventing the insertion. **Behavior on UPDATE**: If an existing Bookings record is being updated and its trip_id is changed (NEW.trip_id <> OLD.trip_id), the function performs

the same check on the new trip_id. If the new trip_id already has two or more associated bookings, an exception is raised. This ensures that a trip always has a maximum of two bookings linked to it.

Backend Layer

The backend of the TAW system is built using Node.js with the Express framework, providing a RESTful API for the frontend Angular application.

The backend uses three environment variables to configure the server:

- `DATABASE_URL` : Used by Prisma to instantiate the database connection and should be set to the PostgreSQL database URL.
- `JWT_SECRET` : A secret key used for signing and verifying JSON Web Tokens (JWT) for user and airline authentication.
- `JWT_EXPIRATION` : Specifies the expiration time for JWT tokens, set to a duration of '1h'.
- `PORT` : Specifies the port on which the Express server will listen, defaulting to 3000 if not set.

The backend is structured into several modules, each responsible for different aspects of the application. Below is a brief overview of the directory structure and the purpose of each module:

- `app.js`
  - Main entry point for the Express backend server.
- `www`
  - Starts the Express server and listens for incoming requests.
- `users`
  - `auth.js`
  - `controller.js` : Implements user-related business logic (account creation, login, info, deletion, admin actions).
  - `index.js` : Defines Express routes for user endpoints, mapping HTTP verbs and paths to controller functions.
- `airlines`
  - `controller.js` : Business logic for airline management (invitation, enrollment, login, aircraft/route/flight/ticket management/statistics).
  - `index.js` : Defines Express routes for airline endpoints, mapping HTTP verbs and paths to controller functions.
- `navigation`
  - `alg.js` : Contains algorithms for route/path finding (e.g., multi-leg route search) using Breadth-First Search (BFS).
  - `controller.js` : Implements navigation-related business logic (route creation, airport management, flight search).
  - `index.js` : Defines Express routes for navigation endpoints, mapping HTTP verbs and paths to controller functions.
- `bookings`
  - `controller.js` : Implements booking logic (download ticket, booking details, available tickets/seats, extras, trip management).
  - `index.js` : Defines Express routes for booking endpoints, mapping HTTP verbs and paths to controller functions.
  - `ticketGenerator.js.js` : Generates PDF tickets using Puppeteer and HTML templates.
  - `template.html` : HTML template for the ticket PDF, styled with CSS.
- `middlewares`
  - `auth.js` : Middleware for authenticating JWT tokens and attaching user info to requests.
  - `checkAirlineEnrollment.js` : Middleware to verify airline enrollment for protected endpoints.
  - `validateJsonRequest.js` : Middleware to validate incoming JSON request bodies.
- `utils`
  - `jwt.js` : Utility functions for JWT token creation, verification, and blacklisting.
  - `prisma.js` : nitializes and exports a Prisma client instance for database access.
- `prisma`
  - `schema.prisma` : Prisma schema definition for the database (models, relations, constraints).

The backend is designed with security and data integrity in mind: it rigorously validates all incoming requests and does not trust any data received from the frontend. Every endpoint performs thorough checks on input parameters, request bodies, and authentication tokens to prevent invalid, malformed, or unauthorized data from affecting the system. This ensures that even if the frontend is bypassed or manipulated, the backend will reject any invalid or malicious requests, maintaining the reliability and security of the application. Prisma is used as the ORM to interact with the PostgreSQL database, providing type-safe access to the database models and ensuring that all queries are validated against the schema defined in `schema.prisma` .

Endpoints

Accounts

1. Create new account

API Request Description

This endpoint allows you to create a new user account by sending a POST request to the specified URL. The request should include the user's name, surname, email, and password in the request body.

Request Body

- `name` : (text) The user's first name.
- `email` : (text) The user's email address.
- `password` : (text) The user's chosen password.

Response

Upon successful execution, the server will respond with a status code of 200 and a JSON object containing a message and user JWT.

Endpoint:

```
Method: POST
Type: RAW
URL: {{ServerIpAddress}}{{serverPort}}/api/users/user
```

Body:

```
{
    "name": "Admin",
    "email": "adminAccount@gmail.com",
    "password": "ciao"
}
```

2. Get info about account

API Request Description

The `GET` request retrieves the account information for a user from the server. Data is extracted from JWT not from database.

USE THIS ONLY FOR TESTING

Endpoint:

```
Method: GET
Type:
URL: {{ServerIpAddress}}{{serverPort}}/api/users/user
```

3. Login and get JWT token

API Request Description

This endpoint allows you to get JWT token when user needs to log in.

Request Body

- `email` : (text) The user's email address.
- `password` : (text) The user's chosen password.

Response

Upon successful execution, the server will respond with a status code of 200 and a JSON object containing a message and user JWT.

Else error...

From frontend save token to local storage

Endpoint:

```
Method: POST
Type: RAW
URL: {{ServerIpAddress}}{{serverPort}}/api/users/login
```

Body:

```
{
    "email": "adminAccount@gmail.com",
    "password": "ciao"
}
```

More example Requests/Responses:

I. Example Request: Login and get JWT token

Body:

```
{
    "email": "testaccount@gmail.com",
    "password": "ciao"
}
```

I. Example Response: Login and get JWT token

```
{
    "message": "Login successful",
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VySWQiOjUsIm5hbWUiOiJUZXN0QWNjb3VudCIsImVtYWlsIjoidGVzdGFjY291bnRAZ21haWwuY
}
```

Status Code: 200

4. Delete User Account

API Request Description

This endpoint allows you to delete a logged in user.

Endpoint:

```
Method: DELETE
Type:
URL: {{ServerIpAddress}}{{serverPort}}/api/users/user
```

5. Sudo List all accounts

API Request Description

This endpoint allows airline to list all the users registered in the system.

Response

Upon successful execution, the server will respond with a status code of 200 and a JSON object containing a list.

Endpoint:

```
Method: GET
Type:
URL: {{ServerIpAddress}}{{serverPort}}/api/users/accounts
```

6. Sudo delete account

API Request Description

This endpoint allows admin user to delete a user given a user id

Endpoint:

```
Method: DELETE
Type:
URL: {{ServerIpAddress}}{{serverPort}}/api/users/accounts/:id
```

URL variables:

Key	Value	Description
id	6	

Airlines

Airlines management endpoints

1. Create new account

API Request Description

Airlines cannot register themselves but must be added by an admin user who sets a temporary password. Then, at the first login, the new airline must set its password and any relevant information (read more in the documentation of the airline enrollment endpoint).

This endpoints allow admins to create new ailrline account.

Request Body

- `name` : (text) The airline name.

Response

Upon successful execution, the server will respond with a status code of 200 and a JSON object containing the invitation code needed for the first login.

Endpoint:

```
Method: POST
Type: RAW
URL: {{ServerIpAddress}}{{serverPort}}/api/airlines/invite
```

Body:

```
{
    "name": "airline"
}
```

More example Requests/Responses:

I. Example Request: Create new account

Body:

```
{
    "name": "airline"
}
```

I. Example Response: Create new account

```
{
    "message": "Airline account created successfully",
    "name": "airline",
    "invitationCode": "ka27tycfgqj"
}
```

Status Code: 200

2. Airline Enrollment

API Request Description

This endpoints allow airlines to add informations and change password after reciving the invitation id.

Request Body

- password: (text) The new password.
- country: (text) The airline country.
- motto: (text) The airline motto.

Response

Upon successful execution, the server will respond with a status code of 200 and a JSON object containing the invitation code needed for the first login.

Endpoint:

```
Method: POST
Type: RAW
URL: {{ServerIpAddress}}{{serverPort}}/api/airlines/enroll/:invitationCode/:airlineName
```

URL variables:

Key	Value	Description
invitationCode	6x9klmjjm6x	
airlineName	airline	

Body:

```
{
    "password": "pass",
    "country": "county",
    "motto": "motto"
}
```

3. Login and get JWT token

API Request Description

This endpoint allows airline to get JWT token when user needs to log in.

Airline must be enrolled for this endpoint to work.

Request Body

- `name` : (text) The airline name.
- `password` : (text) The airline's chosen password.

Response

Upon successful execution, the server will respond with a status code of 200 and a JSON object containing a message and airline JWT.

Endpoint:

```
Method: POST
Type: RAW
URL: {{ServerIpAddress}}{{serverPort}}/api/airlines/login
```

Body:

```
{
    "name": "FlySafe",
    "password": "password123"
}
```

4. List all airlines

API Request Description

This endpoint allows airline to list all the airlines registered in the system.

Response

Upon successful execution, the server will respond with a status code of 200 and a JSON object containing a list.

Endpoint:

```
Method: GET
Type:
URL: {{ServerIpAddress}}{{serverPort}}/api/airlines/airlines
```

More example Requests/Responses:

I. Example Request: List all airlines

Body: None

I. Example Response: List all airlines

```
{
    "message": "List of airlines retrieved successfully",
    "airlines": [
        {
            "name": "airline",
            "country": "county",
            "motto": "motto"
        }
    ]
}
```

Status Code: 200

5. Create new aircraft

API Request Description

This endpoint allows airlines to add a new aircraft to the fleet. By sending a POST request to `/api/airlines/aircrafts`, you can specify the details of the aircraft you wish to register.

Request Parameters

The request body must be in JSON format and should include the following parameters:

- **aircraftType** (string): The type of the aircraft being added (e.g., "Aircraft").
- **capacity** (integer): The seating capacity of the aircraft (e.g., 100).

Example Request Body:

```
{
  "aircraftType": "Aircraft",
  "capacity": 100
}
```

Response

Upon a successful request, the server will respond with a status code of 200 and a JSON object containing the following fields:

- **message** (string): A message indicating the result of the operation (may be empty).
- **aircraftType** (string): The type of aircraft that was added (echoed back from the request).
- **capacity** (integer): The capacity of the aircraft that was added (echoed back from the request).

Endpoint:

```
Method: POST
Type: RAW
URL: {{ServerIpAddress}}{{serverPort}}/api/airlines/aircrafts
```

Body:

```
{
    "aircraftType": "Aircraft",
    "capacity": 100
}
```

More example Requests/Responses:

I. Example Request: Create new aircraft

Body:

```
{
    "aircraftType": "Aircraft",
    "capacity": 100
}
```

I. Example Response: Create new aircraft

```
{
    "message": "Aircraft added successfully",
    "aircraftType": "Aircraft",
    "capacity": 100
}
```

Status Code: 200

6. List all aircrafts in airline fleet

API Request Description

This endpoint retrieves a list of aircrafts associated with a specific airline. It is useful for obtaining details about the aircraft models and their seating capacities for a given airline.

Path Parameters

- `airlineName` (string): The name of the airline for which the aircraft details are being requested. This parameter is required and should be URL-encoded if it contains special characters.

Response

- `aircrafts` (array): An array of aircraft objects, where each object contains:
    - id (int): The id of the aircraft.

- `model` (string): The model of the aircraft.
- `seats_capacity` (integer): The total seating capacity of the aircraft.

Endpoint:

```
Method: GET
Type:
URL: {{ServerIpAddress}}{{serverPort}}/api/airlines/aircrafts/:airlineName
```

URL variables:

Key	Value	Description
airlineName	FlySafe	

More example Requests/Responses:

I. Example Request: List all aircrafts in airline fleet

Query:

Key	Value	Description
airlineName	airline	

Body: None

I. Example Response: List all aircrafts in airline fleet

```
{
    "message": "List of aircrafts retrieved successfully",
    "aircrafts": [
        {
            "id": 7,
            "model": "Aircraft",
            "seats_capacity": 100
        },
        {
            "id": 8,
            "model": "Aircraft",
            "seats_capacity": 100
        },
        {
            "id": 9,
            "model": "Aircraft",
            "seats_capacity": 100
        }
    ]
}
```

Status Code: 200

7. Delete aircraft by id

API Request Description

This endpoint allows airlines to delete an aircraft from the system by specifying its unique identifier.

Path Parameter

- `aircraftId` (int): The unique identifier of the aircraft that you wish to delete.

Response

Upon a successful deletion, the API will return a response with the following format:

- **Status Code**: `200 OK`
- **Content-Type**: `application/json`
- { "message": "", "aircraftId": ""}
    - `message` (string): A message indicating the result of the deletion operation (may be empty).
    - `aircraftId` (string): The ID of the aircraft that was deleted.

This response confirms that the aircraft has been successfully removed from the system.

Endpoint:

```
Method: DELETE
Type:
URL: {{ServerIpAddress}}{{serverPort}}/api/airlines/aircrafts/:aircraftId
```

URL variables:

Key	Value	Description
aircraftId	9	

More example Requests/Responses:

I. Example Request: Delete aircraft by id

Query:

Key	Value	Description
aircraftId	9	

Body: None

I. Example Response: Delete aircraft by id

```
{
    "message": "Aircraft deleted successfully",
    "aircraftId": "9"
}
```

Status Code: 200

8. Enroll in route

API Request Description
This endpoint allows an airline to enroll in a specific route by providing the departure and destination airport IDs. The airline must not already be enrolled in the route, and the route must exist.

Request Body

- `departure` (int): The ID of the departure airport.
- `destination` (int): The ID of the destination airport.

Endpoint:

```
Method: POST
Type: RAW
URL: {{ServerIpAddress}}{{serverPort}}/api/airlines/routes
```

Body:

```
{
  "departure": 1,
  "destination": 2
}
```

More example Requests/Responses:

I. Example Request: Enroll in route

Body:

```
{
  "departure": 1,
  "destination": 2
}
```

I. Example Response: Enroll in route

```
{
    "message": "Successfully enrolled in route",
    "routeId": 6
}
```

Status Code: 200

II. Example Request: Enroll in route non succesfull

Body:

```
{
  "departure": 1,
  "destination": 2
}
```

II. Example Response: Enroll in route non succesfull

```
{
    "error": "Airline is already enrolled in this route"
}
```

Status Code: 409

9. UnEnroll from routes

API Request Description

This endpoint allows an airline to unenroll from a specific route by specifying the unique identifier of the enrollment (routeId). The airline must be currently enrolled in the route.

Path Parameter

- `routeId` (int): The unique identifier of the airline's enrollment in the route that you wish to remove.

Endpoint:

```
Method: DELETE
Type:
URL: {{ServerIpAddress}}{{serverPort}}/api/airlines/routes/:routeId
```

URL variables:

Key	Value	Description
routeId	5	

More example Requests/Responses:

I. Example Request: UnEnroll from routes succesfull

Query:

Key	Value	Description
routeId	4	

Body: None

I. Example Response: UnEnroll from routes succesfull

```
{
    "message": "Successfully unenrolled from route",
    "routeId": "4"
}
```

Status Code: 200

II. Example Request: UnEnroll from routes non succesfull

Query:

Key	Value	Description
routeId	3	

Body: None

II. Example Response: UnEnroll from routes non succesfull

```
{
    "error": "No enrollment found for this airline in the specified route. No action taken."
}
```

Status Code: 404

10. List enrolled routes for airline

API Request Description

This endpoint allows an airline to retrieve the list of all routes in which it is currently enrolled. For each route, details about the departure and destination airports are included.

Endpoint:

```
Method: GET
Type:
URL: {{ServerIpAddress}}{{serverPort}}/api/airlines/routes
```

More example Requests/Responses:

I. Example Request: List enrolled routes for airline

Body: None

I. Example Response: List enrolled routes for airline

```
{
    "message": "List of routes retrieved successfully",
    "routes": [
        {
            "id": 4,
            "route_departure": 1,
            "route_destination": 2,
            "departure_details": {
                "name": "Dubai International",
                "city": "Dubai",
                "country": "United Arab Emirates"
            },
            "destination_details": {
                "name": "Frankfurt Airport",
                "city": "Frankfurt",
                "country": "Germany"
            }
        }
    ]
}
```


Status Code: 200

11. List flights of an airline

API Request Description

This endpoint allows an airline to retrieve the list of all flights it operates. For each flight, details about the aircraft and the local departure time (converted from UTC to the departure airport's time zone) are included.

Response

Upon a successful deletion, the API will return a response with the following format:

- `code` (string): The unique identifier of the flight.
- `liftoff_date` (string): The UTC date and time when the flight departs.
- `duration` (int): The duration of the flight in minutes.
- `route_departure` (int): The ID of the departure airport.
- `route_destination` (int): The ID of the destination airport.
- `aircraft_id` (int): The ID of the aircraft used for the flight.
- `lifoffTimeZone` (string): The time zone of the departure airport.
- `liftoff_dateLOCAL` (string): The local date and time of departure at the departure airport.
- `aircraft_details` (object): Details about the aircraft (model and seat capacity).

Endpoint:

```
Method: GET
Type:
URL: {{ServerIpAddress}}{{serverPort}}/api/airlines/flights
```

More example Requests/Responses:

I. Example Request: List flights of an airline

Body: None

I. Example Response: List flights of an airline


```
{
    "message": "List of flights retrieved successfully",
    "flights": [
        {
```

```
        {
            "code": "187f190c-505e-425a-bf49-bebcbd9eaca4",
            "liftoff_date": "2025-10-14T22:00:00.000Z",
            "duration": 91,
            "route_departure": 4,
            "route_destination": 1,
            "aircraft_id": 5,
            "lifoffTimeZone": "Europe/Rome",
            "liftoff_dateLOCAL": "2025-10-15T00:00:00",
            "aircraft_details": {
                "model": "Boeing 777",
                "seats_capacity": 396
            }
        },
        {
            "code": "6ef89afc-ca82-4ed7-b76c-101b5fb10c41",
            "liftoff_date": "2025-10-04T22:00:00.000Z",
            "duration": 91,
            "route_departure": 4,
            "route_destination": 1,
            "aircraft_id": 5,
            "lifoffTimeZone": "Europe/Rome",
            "liftoff_dateLOCAL": "2025-10-05T00:00:00",
            "aircraft_details": {
                "model": "Boeing 777",
                "seats_capacity": 396
            }
        },
        {
            "code": "fbd7353f-dcdc-4498-974e-81e35d11cd79",
            "liftoff_date": "2025-10-04T22:00:00.000Z",
            "duration": 91,
            "route_departure": 4,
            "route_destination": 1,
            "aircraft_id": 5,
            "lifoffTimeZone": "Europe/Rome",
            "liftoff_dateLOCAL": "2025-10-05T00:00:00",
            "aircraft_details": {
                "model": "Boeing 777",
                "seats_capacity": 396
            }
        },
        {
            "code": "49418977-98c9-4d72-afc2-38b25fdb9ca6",
            "liftoff_date": "2025-07-01T08:00:00.000Z",
            "duration": 91,
            "route_departure": 4,
            "route_destination": 1,
            "aircraft_id": 5,
            "lifoffTimeZone": "Europe/Rome",
            "liftoff_dateLOCAL": "2025-07-01T10:00:00",
            "aircraft_details": {
                "model": "Boeing 777",
                "seats_capacity": 396
            }
        },
        {
            "code": "07cccead-eeac-4614-bc46-f095df9d9a7b",
            "liftoff_date": "2025-07-01T10:00:00.000Z",
            "duration": 91,
            "route_departure": 4,
            "route_destination": 1,
            "aircraft_id": 5,
            "lifoffTimeZone": "Europe/Rome",
            "liftoff_dateLOCAL": "2025-07-01T12:00:00",
            "aircraft_details": {
```

```
                "model": "Boeing 777",
                "seats_capacity": 396
            }
        },
        {
            "code": "594edef7-067f-48e4-983e-efd84bae9392",
            "liftoff_date": "2025-06-20T08:00:00.000Z",
            "duration": 91,
            "route_departure": 4,
            "route_destination": 1,
            "aircraft_id": 5,
            "lifoffTimeZone": "Europe/Rome",
            "liftoff_dateLOCAL": "2025-06-20T10:00:00",
            "aircraft_details": {
                "model": "Boeing 777",
                "seats_capacity": 396
            }
        },
        {
            "code": "12f2f821-d9b3-4172-ba69-a2edaec30b48",
            "liftoff_date": "2025-06-20T10:00:00.000Z",
            "duration": 91,
            "route_departure": 4,
            "route_destination": 1,
            "aircraft_id": 2,
            "lifoffTimeZone": "Europe/Rome",
            "liftoff_dateLOCAL": "2025-06-20T12:00:00",
            "aircraft_details": {
                "model": "Boeing 747-8",
                "seats_capacity": 400
            }
        },
        {
            "code": "b6916970-f7a7-452e-b969-1e7d7defbaed",
            "liftoff_date": "2025-06-20T08:00:00.000Z",
            "duration": 91,
            "route_departure": 1,
            "route_destination": 14,
            "aircraft_id": 2,
            "lifoffTimeZone": "Asia/Dubai",
            "liftoff_dateLOCAL": "2025-06-20T12:00:00",
            "aircraft_details": {
                "model": "Boeing 747-8",
                "seats_capacity": 400
            }
        }
    ]
}
```


Status Code: 200

12. List pending flights of an airline

API Request Description

This endpoint allows an airline to retrieve the list of all its **active flights** (i.e., flights with a liftoff date in the future). For each flight, details about the aircraft and the local departure time (converted from UTC to the departure airport's time zone) are included.

Endpoint:

```
Method: GET
Type:
URL: {{ServerIpAddress}}{{serverPort}}/api/airlines/flights/pending
```

More example Requests/Responses:

I. Example Request: List flights of an airline

Body: None

I. Example Response: List flights of an airline

```
{
    "message": "List of active flights retrieved successfully",
    "flights": [
        {
            "code": "ae31cc05-307e-4453-8e86-80145fda9ed2",
            "liftoff_date": "2025-06-21T08:00:00.000Z",
            "duration": 91,
            "route_departure": 1,
            "route_destination": 7,
            "aircraft_id": 2,
            "lifoffTimeZone": "Asia/Dubai",
            "liftoff_dateLOCAL": "2025-06-21T12:00:00",
            "aircraft_details": {
                "model": "Boeing 747-8",
                "seats_capacity": 400
            }
        }
    ]
}
```

Status Code: 200

13. Create tickets for flight

API Request Description

This endpoint allows an airline to create one or more ticket types (e.g., ECONOMY, BUSINESS, BASE, DELUXE, LUXURY) for a specific flight it operates (flight code). Each ticket must specify its type and price. Duplicate ticket types for the same flight are not allowed.

Endpoint:

```
Method: POST
Type: RAW
URL: {{ServerIpAddress}}{{serverPort}}/api/airlines/tickets
```

Body:

```
{
    "flightCode": "e5a39b2f-1320-4e4c-9d10-0cf960a2367c",
    "ticketsArray" : [
        {
            "type": "BASE",
            "price": 300
        },
        {
            "type": "DELUXE",
            "price": 300
        }
    ]
}
```

More example Requests/Responses:

I. Example Request: New ticket

Body:

```
{
    "flightCode": "e5a39b2f-1320-4e4c-9d10-0cf960a2367c",
    "ticketsArray" : [
        {
            "type": "BASE",
            "price": 300
        },
        {
            "type": "DELUXE",
            "price": 300
        }
    ]
}
```

I. Example Response: New ticket

```
{
    "message": "Tickets created successfully",
    "ticketIds": [
        {
            "code": "28a7ea0c-ccd1-4e5e-912f-f082326cb9d4",
            "type": "BASE",
            "price": 300,
            "fligt_code": "e5a39b2f-1320-4e4c-9d10-0cf960a2367c"
        },
        {
            "code": "e736e770-732f-4330-877a-1d91c93af57b",
            "type": "DELUXE",
            "price": 300,
            "fligt_code": "e5a39b2f-1320-4e4c-9d10-0cf960a2367c"
        }
    ],
    "flightCode": "e5a39b2f-1320-4e4c-9d10-0cf960a2367c"
}
```

Status Code: 200

14. Delete ticket

API Request Description

This endpoint allows an airline to delete a specific ticket by providing its unique ticket UUID. The ticket must belong to a flight operated by the airline.

Endpoint:

```
Method: DELETE
Type:
URL: {{ServerIpAddress}}{{serverPort}}/api/airlines/tickets/:ticketUUID
```

URL variables:

Key	Value	Description
ticketUUID	e736e770-732f-4330-877a-1d91c93af57b	

More example Requests/Responses:

I. Example Request: Delete ticket

Query:

Key	Value	Description

Key	Value	Description
ticketUUID	e736e770-732f-4330-877a-1d91c93af57b	

Body: None

I. Example Response: Delete ticket

```
{
    "message": "Ticket deleted successfully",
    "ticketId": {
        "code": "e736e770-732f-4330-877a-1d91c93af57b",
        "type": "DELUXE",
        "price": 300,
        "fligt_code": "e5a39b2f-1320-4e4c-9d10-0cf960a2367c",
        "flights": {
            "code": "e5a39b2f-1320-4e4c-9d10-0cf960a2367c",
            "duration": 91,
            "aircraft_id": 2,
            "liftoff_date": "2025-06-20T08:00:00.000Z",
            "route_departure": 1,
            "route_destination": 7,
            "airline_name": "FlySafe"
        }
    }
}
```

Status Code: 200

15. Stats about routes

Endpoint:

```
Method: GET
Type:
URL: {{ServerIpAddress}}{{serverPort}}/api/airlines/stats/routes
```

More example Requests/Responses:

I. Example Request: Stats route

Body: None

I. Example Response: Stats route

```
{
    "message": "Routes stats retrieved successfully",
    "stats": {
        "1-2": {
            "flightCount": 1,
            "ticketCount": 5,
            "bookingsCount": 1,
            "trendPercentage": 100
        },
        "2-1": {
            "flightCount": 1,
            "ticketCount": 5,
            "bookingsCount": 0,
            "trendPercentage": 0
        },
        "1-3": {
            "flightCount": 0,
            "ticketCount": 0,
            "bookingsCount": 0,
            "trendPercentage": 0
        },
        "3-1": {
            "flightCount": 0,
            "ticketCount": 0,
            "bookingsCount": 0,
            "trendPercentage": 0
        },
        "3-9": {
            "flightCount": 1,
            "ticketCount": 5,
            "bookingsCount": 0,
            "trendPercentage": 0
        },
        "9-3": {
            "flightCount": 1,
            "ticketCount": 5,
            "bookingsCount": 0,
            "trendPercentage": 0
        }
    }
}
```


Status Code: 200

Navigation

1. Create new route

API Request Description
This endpoint allows an airline to create a new route between two airports by specifying the origin and destination airport IDs. The route will only be created if both airports exist and the route does not already exist.

When a route is created, if the airline wants to use it, it must enroll with the enrollment endpoint.

Endpoint:

```
Method: POST
Type: RAW
URL: {{ServerIpAddress}}{{serverPort}}/api/navigate/routes
```

Body:

```
{
    "origin": 1,
    "destination": 7
}
```

More example Requests/Responses:

I. Example Request: new route

Body:

```
{
    "origin": 1,
    "destination": 7
}
```

I. Example Response: new route

```
{
    "message": "Route created successfully",
    "route": {
        "departure": 1,
        "destination": 7
    }
}
```

Status Code: 201

2. List available routes

API Request Description

This endpoint allows anyone to retrieve the list of all available routes in the system. For each route, details about the departure and destination airports are included.

Endpoint:

```
Method: GET
Type:
URL: {{ServerIpAddress}}{{serverPort}}/api/navigate/routes
```

More example Requests/Responses:

I. Example Request: List available routes

Body: None

I. Example Response: List available routes

```
{
    "message": "List of routes retrieved successfully",
    "routes": [
        {
            "departure": 1,
            "destination": 2,
            "airports_routes_departureToairports": {
                "id": 1,
                "name": "Dubai International",
                "city": "Dubai",
                "country": "United Arab Emirates",
                "lat": 25.253174,
                "lan": 55.365673,
                "time_zone": 4
            },
            "airports_routes_destinationToairports": {
```

```
        "airports_routes_destinationToairports": {
            "id": 2,
            "name": "Frankfurt Airport",
            "city": "Frankfurt",
            "country": "Germany",
            "lat": 50.037933,
            "lan": 8.562152,
            "time_zone": 1
        }
    },
    {
        "departure": 2,
        "destination": 3,
        "airports_routes_departureToairports": {
            "id": 2,
            "name": "Frankfurt Airport",
            "city": "Frankfurt",
            "country": "Germany",
            "lat": 50.037933,
            "lan": 8.562152,
            "time_zone": 1
        },
        "airports_routes_destinationToairports": {
            "id": 3,
            "name": "Changi Airport",
            "city": "Singapore",
            "country": "Singapore",
            "lat": 1.36442,
            "lan": 103.991531,
            "time_zone": 8
        }
    },
    {
        "departure": 1,
        "destination": 3,
        "airports_routes_departureToairports": {
            "id": 1,
            "name": "Dubai International",
            "city": "Dubai",
            "country": "United Arab Emirates",
            "lat": 25.253174,
            "lan": 55.365673,
            "time_zone": 4
        },
        "airports_routes_destinationToairports": {
            "id": 3,
            "name": "Changi Airport",
            "city": "Singapore",
            "country": "Singapore",
            "lat": 1.36442,
            "lan": 103.991531,
            "time_zone": 8
        }
    },
    {
        "departure": 4,
        "destination": 5,
        "airports_routes_departureToairports": {
            "id": 4,
            "name": "Malpensa",
            "city": "Milano",
            "country": "Italy",
            "lat": 45.630062,
            "lan": 8.723068,
            "time_zone": 1
        },
```

```
            "airports_routes_destinationToairports": {
                "id": 5,
                "name": "Charles de Gaulle",
                "city": "Paris",
                "country": "France",
                "lat": 49.00969,
                "lan": 2.547925,
                "time_zone": 1
            }
        },
        {
            "departure": 5,
            "destination": 6,
            "airports_routes_departureToairports": {
                "id": 5,
                "name": "Charles de Gaulle",
                "city": "Paris",
                "country": "France",
                "lat": 49.00969,
                "lan": 2.547925,
                "time_zone": 1
            },
            "airports_routes_destinationToairports": {
                "id": 6,
                "name": "Heathrow",
                "city": "London",
                "country": "UK",
                "lat": 51.47002,
                "lan": -0.454295,
                "time_zone": 0
            }
        },
        {
            "departure": 4,
            "destination": 6,
            "airports_routes_departureToairports": {
                "id": 4,
                "name": "Malpensa",
                "city": "Milano",
                "country": "Italy",
                "lat": 45.630062,
                "lan": 8.723068,
                "time_zone": 1
            },
            "airports_routes_destinationToairports": {
                "id": 6,
                "name": "Heathrow",
                "city": "London",
                "country": "UK",
                "lat": 51.47002,
                "lan": -0.454295,
                "time_zone": 0
            }
        },
        {
            "departure": 1,
            "destination": 7,
            "airports_routes_departureToairports": {
                "id": 1,
                "name": "Dubai International",
                "city": "Dubai",
                "country": "United Arab Emirates",
                "lat": 25.253174,
                "lan": 55.365673,
                "time_zone": 4
            }
```

```
        },
        "airports_routes_destinationToairports": {
            "id": 7,
            "name": "John F. Kennedy International",
            "city": "New York",
            "country": "United States",
            "lat": 40.641311,
            "lan": -73.778139,
            "time_zone": -5
        }
    },
    {
        "departure": 7,
        "destination": 8,
        "airports_routes_departureToairports": {
            "id": 7,
            "name": "John F. Kennedy International",
            "city": "New York",
            "country": "United States",
            "lat": 40.641311,
            "lan": -73.778139,
            "time_zone": -5
        },
        "airports_routes_destinationToairports": {
            "id": 8,
            "name": "Los Angeles International",
            "city": "Los Angeles",
            "country": "United States",
            "lat": 33.941589,
            "lan": -118.40853,
            "time_zone": -8
        }
    },
    {
        "departure": 8,
        "destination": 9,
        "airports_routes_departureToairports": {
            "id": 8,
            "name": "Los Angeles International",
            "city": "Los Angeles",
            "country": "United States",
            "lat": 33.941589,
            "lan": -118.40853,
            "time_zone": -8
        },
        "airports_routes_destinationToairports": {
            "id": 9,
            "name": "Tokyo Haneda",
            "city": "Tokyo",
            "country": "Japan",
            "lat": 35.549393,
            "lan": 139.779839,
            "time_zone": 9
        }
    },
    {
        "departure": 9,
        "destination": 10,
        "airports_routes_departureToairports": {
            "id": 9,
            "name": "Tokyo Haneda",
            "city": "Tokyo",
            "country": "Japan",
            "lat": 35.549393,
            "lan": 139.779839,
            "time_zone": 9
```

```
        },
        "airports_routes_destinationToairports": {
            "id": 10,
            "name": "Sydney Kingsford Smith",
            "city": "Sydney",
            "country": "Australia",
            "lat": -33.939922,
            "lan": 151.175276,
            "time_zone": 10
        }
    },
    {
        "departure": 10,
        "destination": 11,
        "airports_routes_departureToairports": {
            "id": 10,
            "name": "Sydney Kingsford Smith",
            "city": "Sydney",
            "country": "Australia",
            "lat": -33.939922,
            "lan": 151.175276,
            "time_zone": 10
        },
        "airports_routes_destinationToairports": {
            "id": 11,
            "name": "Madrid Barajas",
            "city": "Madrid",
            "country": "Spain",
            "lat": 40.498299,
            "lan": -3.5676,
            "time_zone": 1
        }
    },
    {
        "departure": 11,
        "destination": 12,
        "airports_routes_departureToairports": {
            "id": 11,
            "name": "Madrid Barajas",
            "city": "Madrid",
            "country": "Spain",
            "lat": 40.498299,
            "lan": -3.5676,
            "time_zone": 1
        },
        "airports_routes_destinationToairports": {
            "id": 12,
            "name": "Toronto Pearson",
            "city": "Toronto",
            "country": "Canada",
            "lat": 43.677717,
            "lan": -79.624819,
            "time_zone": -5
        }
    },
    {
        "departure": 12,
        "destination": 13,
        "airports_routes_departureToairports": {
            "id": 12,
            "name": "Toronto Pearson",
            "city": "Toronto",
            "country": "Canada",
            "lat": 43.677717,
            "lan": -79.624819,
```

```
            "time_zone": -5
        },
        "airports_routes_destinationToairports": {
            "id": 13,
            "name": "Beijing Capital",
            "city": "Beijing",
            "country": "China",
            "lat": 40.080111,
            "lan": 116.584556,
            "time_zone": 8
        }
    },
    {
        "departure": 13,
        "destination": 14,
        "airports_routes_departureToairports": {
            "id": 13,
            "name": "Beijing Capital",
            "city": "Beijing",
            "country": "China",
            "lat": 40.080111,
            "lan": 116.584556,
            "time_zone": 8
        },
        "airports_routes_destinationToairports": {
            "id": 14,
            "name": "Istanbul Airport",
            "city": "Istanbul",
            "country": "Turkey",
            "lat": 41.275278,
            "lan": 28.751944,
            "time_zone": 3
        }
    },
    {
        "departure": 14,
        "destination": 15,
        "airports_routes_departureToairports": {
            "id": 14,
            "name": "Istanbul Airport",
            "city": "Istanbul",
            "country": "Turkey",
            "lat": 41.275278,
            "lan": 28.751944,
            "time_zone": 3
        },
        "airports_routes_destinationToairports": {
            "id": 15,
            "name": "Amsterdam Schiphol",
            "city": "Amsterdam",
            "country": "Netherlands",
            "lat": 52.310539,
            "lan": 4.768274,
            "time_zone": 1
        }
    },
    {
        "departure": 15,
        "destination": 16,
        "airports_routes_departureToairports": {
            "id": 15,
            "name": "Amsterdam Schiphol",
            "city": "Amsterdam",
            "country": "Netherlands",
            "lat": 52.310539,
            "lan": 4.768274,
```

```
            "lan": 4.768274,
            "time_zone": 1
        },
        "airports_routes_destinationToairports": {
            "id": 16,
            "name": "Zurich Airport",
            "city": "Zurich",
            "country": "Switzerland",
            "lat": 47.458056,
            "lan": 8.555556,
            "time_zone": 1
        }
    },
    {
        "departure": 16,
        "destination": 1,
        "airports_routes_departureToairports": {
            "id": 16,
            "name": "Zurich Airport",
            "city": "Zurich",
            "country": "Switzerland",
            "lat": 47.458056,
            "lan": 8.555556,
            "time_zone": 1
        },
        "airports_routes_destinationToairports": {
            "id": 1,
            "name": "Dubai International",
            "city": "Dubai",
            "country": "United Arab Emirates",
            "lat": 25.253174,
            "lan": 55.365673,
            "time_zone": 4
        }
    },
    {
        "departure": 3,
        "destination": 9,
        "airports_routes_departureToairports": {
            "id": 3,
            "name": "Changi Airport",
            "city": "Singapore",
            "country": "Singapore",
            "lat": 1.36442,
            "lan": 103.991531,
            "time_zone": 8
        },
        "airports_routes_destinationToairports": {
            "id": 9,
            "name": "Tokyo Haneda",
            "city": "Tokyo",
            "country": "Japan",
            "lat": 35.549393,
            "lan": 139.779839,
            "time_zone": 9
        }
    },
    {
        "departure": 2,
        "destination": 12,
        "airports_routes_departureToairports": {
            "id": 2,
            "name": "Frankfurt Airport",
            "city": "Frankfurt",
            "country": "Germany",
            "lat": 50.037933,
```

```
                "lan": 8.562152,
                "time_zone": 1
            },
            "airports_routes_destinationToairports": {
                "id": 12,
                "name": "Toronto Pearson",
                "city": "Toronto",
                "country": "Canada",
                "lat": 43.677717,
                "lan": -79.624819,
                "time_zone": -5
            }
        },
        {
            "departure": 5,
            "destination": 13,
            "airports_routes_departureToairports": {
                "id": 5,
                "name": "Charles de Gaulle",
                "city": "Paris",
                "country": "France",
                "lat": 49.00969,
                "lan": 2.547925,
                "time_zone": 1
            },
            "airports_routes_destinationToairports": {
                "id": 13,
                "name": "Beijing Capital",
                "city": "Beijing",
                "country": "China",
                "lat": 40.080111,
                "lan": 116.584556,
                "time_zone": 8
            }
        },
        {
            "departure": 6,
            "destination": 7,
            "airports_routes_departureToairports": {
                "id": 6,
                "name": "Heathrow",
                "city": "London",
                "country": "UK",
                "lat": 51.47002,
                "lan": -0.454295,
                "time_zone": 0
            },
            "airports_routes_destinationToairports": {
                "id": 7,
                "name": "John F. Kennedy International",
                "city": "New York",
                "country": "United States",
                "lat": 40.641311,
                "lan": -73.778139,
                "time_zone": -5
            }
        }
    ]
}
```


Status Code: 200

3. List available routes for specific airline

API Request Description

This endpoint allows you to retrieve all routes in which a specific airline is enrolled, by providing the airline's name as a path parameter.

Endpoint:

```
Method: GET
Type:
URL: {{ServerIpAddress}}{{serverPort}}/api/navigate/routes/:airlineName
```

URL variables:

Key	Value	Description
airlineName	FlySafe	

More example Requests/Responses:

I. Example Request: List available routes for specific airline

Query:

Key	Value	Description
airlineName	airline	

Body: None

I. Example Response: List available routes for specific airline

```
{
    "message": "Routes found for airline",
    "routes": [
        {
            "route_departure": 1,
            "route_destination": 2
        },
        {
            "route_departure": 1,
            "route_destination": 3
        }
    ]
}
```

Status Code: 200

4. List airports

API Request Description

This endpoint allows you to retrieve a list of airports. If a query parameter is provided, the endpoint will return airports whose name, city, or country starts with the query string (case-insensitive). If no query is provided, all airports are returned with additional location details.

Query Parameter

- `query` (string, optional): A string to filter airports by name, city, or country (case-insensitive, starts with).

Endpoint:

```
Method: GET
Type:
URL: {{ServerIpAddress}}{{serverPort}}/api/navigate/airports
```

More example Requests/Responses:

I. Example Request: List airports

Body: None

I. Example Response: List airports


```
{
    "message": "List of airports retrieved successfully",
    "airports": [
        {
            "id": 1,
            "name": "Dubai International",
            "city": "Dubai",
            "country": "United Arab Emirates",
            "time_zone": "Asia/Dubai",
            "lan": 55.365673,
            "lat": 25.253174
        },
        {
            "id": 2,
            "name": "Frankfurt Airport",
            "city": "Frankfurt",
            "country": "Germany",
            "time_zone": "Europe/Berlin",
            "lan": 8.562152,
            "lat": 50.037933
        },
        {
            "id": 3,
            "name": "Changi Airport",
            "city": "Singapore",
            "country": "Singapore",
            "time_zone": "Asia/Singapore",
            "lan": 103.991531,
            "lat": 1.36442
        },
        {
            "id": 4,
            "name": "Malpensa",
            "city": "Milano",
            "country": "Italy",
            "time_zone": "Europe/Rome",
            "lan": 8.723068,
            "lat": 45.630062
        },
        {
            "id": 5,
            "name": "Charles de Gaulle",
            "city": "Paris",
            "country": "France",
            "time_zone": "Europe/Paris",
            "lan": 2.547925,
            "lat": 49.00969
        },
        {
            "id": 6,
            "name": "Heathrow",
            "city": "London",
            "country": "UK",
            "time_zone": "Europe/London",
            "lan": -0.454295,
            "lat": 51.47002
        },
        {
            "id": 7,
            "name": "John F. Kennedy International",
            "city": "New York",
```

```
        "country": "United States",
        "time_zone": "America/New_York",
        "lan": -73.778139,
        "lat": 40.641311
    },
    {
        "id": 8,
        "name": "Los Angeles International",
        "city": "Los Angeles",
        "country": "United States",
        "time_zone": "America/Los_Angeles",
        "lan": -118.40853,
        "lat": 33.941589
    },
    {
        "id": 9,
        "name": "Tokyo Haneda",
        "city": "Tokyo",
        "country": "Japan",
        "time_zone": "Asia/Tokyo",
        "lan": 139.779839,
        "lat": 35.549393
    },
    {
        "id": 10,
        "name": "Sydney Kingsford Smith",
        "city": "Sydney",
        "country": "Australia",
        "time_zone": "Australia/Sydney",
        "lan": 151.175276,
        "lat": -33.939922
    },
    {
        "id": 11,
        "name": "Madrid Barajas",
        "city": "Madrid",
        "country": "Spain",
        "time_zone": "Europe/Madrid",
        "lan": -3.5676,
        "lat": 40.498299
    },
    {
        "id": 12,
        "name": "Toronto Pearson",
        "city": "Toronto",
        "country": "Canada",
        "time_zone": "America/Toronto",
        "lan": -79.624819,
        "lat": 43.677717
    },
    {
        "id": 13,
        "name": "Beijing Capital",
        "city": "Beijing",
        "country": "China",
        "time_zone": "Asia/Shanghai",
        "lan": 116.584556,
        "lat": 40.080111
    },
    {
        "id": 14,
        "name": "Istanbul Airport",
        "city": "Istanbul",
        "country": "Turkey",
        "time_zone": "Europe/Istanbul",
        "lan": 28.751944,
```

```
            "lat": 41.275278
        },
        {
            "id": 15,
            "name": "Amsterdam Schiphol",
            "city": "Amsterdam",
            "country": "Netherlands",
            "time_zone": "Europe/Amsterdam",
            "lan": 4.768274,
            "lat": 52.310539
        },
        {
            "id": 16,
            "name": "Zurich Airport",
            "city": "Zurich",
            "country": "Switzerland",
            "time_zone": "Europe/Zurich",
            "lan": 8.555556,
            "lat": 47.458056
        }
    ]
}
```


Status Code: 200

5. Get Airport details by its id

API Request Description

This endpoint allows you to retrieve detailed information about a specific airport by providing its unique airport ID.

Endpoint:

```
Method: GET
Type:
URL: {{ServerIpAddress}}{{serverPort}}/api/navigate/airports/:airportId
```

URL variables:

Key	Value	Description
airportId	1	

More example Requests/Responses:

I. Example Request: Get Airport details by its id

Query:

Key	Value	Description
airportId	1	

Body: None

I. Example Response: Get Airport details by its id

```
{
    "message": "Airport details retrieved successfully",
    "airport": {
        "id": 1,
        "name": "Dubai International",
        "city": "Dubai",
        "country": "United Arab Emirates",
        "time_zone": "Asia/Dubai",
        "lan": 55.365673,
        "lat": 25.253174
    }
}
```


Status Code: 200

6. List airports autocomplete

API Request Description

This endpoint allows you to retrieve a list of airports. If a query parameter is provided, the endpoint will return airports whose name, city, or country starts with the query string (case-insensitive). If no query is provided, all airports are returned with additional location details.

Query Parameter

- `query` (string, optional): A string to filter airports by name, city, or country (case-insensitive, starts with).

Endpoint:

```
Method: GET
Type:
URL: {{ServerIpAddress}}{{serverPort}}/api/navigate/airports
```

Query params:

Key	Value	Description
query	i	

More example Requests/Responses:

I. Example Request: list airports with autocomplete

Query:

Key	Value	Description
query	mi	

Body: None

I. Example Response: list airports with autocomplete

```
{
    "message": "List of airports retrieved successfully",
    "airports": [
        {
            "id": 4,
            "name": "Malpensa",
            "city": "Milano",
            "country": "Italy",
            "time_zone": 1
        },
        {
            "id": 7,
            "name": "Linate",
            "city": "Milano",
            "country": "Italy",
            "time_zone": 1
        }
    ]
}
```


Status Code: 200

7. Get list of destinations given a departure

API Request Description

This endpoint allows you to retrieve all possible destination airports for a given departure airport. For each destination, details about the destination airport are included.

Endpoint:

```
Method: GET
Type:
URL: {{ServerIpAddress}}{{serverPort}}/api/navigate/airports/:departure/routes
```

URL variables:

Key	Value	Description
departure	1	

More example Requests/Responses:

I. Example Request: Get list of destinations given a departure

Query:

Key	Value	Description
departure	1	

Body: None

I. Example Response: Get list of destinations given a departure

```
{
    "message": "List of destinations retrieved successfully",
    "airports": [
        {
            "destination": 2,
            "destinationAirport": {
                "name": "Frankfurt Airport",
                "city": "Frankfurt",
                "country": "Germany",
                "time_zone": 1
            }
        },
        {
            "destination": 3,
            "destinationAirport": {
                "name": "Changi Airport",
                "city": "Singapore",
                "country": "Singapore",
                "time_zone": 8
            }
        }
    ]
}
```


Status Code: 200

8. Get list of departures given a destination

API Request Description

This endpoint allows you to retrieve all possible departure airports for a given destination airport. For each departure, details about the departure airport are included.

Endpoint:

```
Method: GET
Type:
URL: {{ServerIpAddress}}{{serverPort}}/api/navigate/airports/:destination/incoming-routes
```

URL variables:

Key	Value	Description
destination	3	

More example Requests/Responses:

I. Example Request: Get list of departures given a destination

Query:

Key	Value	Description
destination	3	

Body: None

I. Example Response: Get list of departures given a destination

```
{
    "message": "List of departures retrieved successfully",
    "id": "3",
    "airports": [
        {
            "departure": 2,
            "departureAirport": {
                "name": "Frankfurt Airport",
                "city": "Frankfurt",
                "country": "Germany",
                "time_zone": 1
            }
        },
        {
            "departure": 1,
            "departureAirport": {
                "name": "Dubai International",
                "city": "Dubai",
                "country": "United Arab Emirates",
                "time_zone": 4
            }
        }
    ]
}
```


Status Code: 200

9. Get multi leg routes

API Request Description

This endpoint finds and returns the path (sequence of airports) between two airports, if a route exists. The search is performed using the provided from and to airport IDs as query parameters.

This response confirms that a path between the specified airports has been found and provides the sequence of airports connected by routes.

Endpoint:

```
Method: GET
Type:
URL: {{ServerIpAddress}}{{serverPort}}/api/navigate/routes/path
```

Query params:

Key	Value	Description
from	1	
to	16	

More example Requests/Responses:

I. Example Request: muli leg

Query:

Key	Value	Description
from	1	
to	16	

Body: None

I. Example Response: muli leg

```
{
    "message": "Route found",
    "stepsCount": 7,
    "steps": [
        1,
        2,
        12,
        13,
        14,
        15,
        16
    ],
    "from": 1,
    "to": 16,
    "path": [
        {
            "id": 1,
            "name": "Dubai International",
            "city": "Dubai",
            "country": "United Arab Emirates",
            "time_zone": 4
        },
        {
            "id": 2,
            "name": "Frankfurt Airport",
            "city": "Frankfurt",
            "country": "Germany",
            "time_zone": 1
        },
        {
            "id": 12,
            "name": "Toronto Pearson",
            "city": "Toronto",
            "country": "Canada",
            "time_zone": -5
        },
        {
            "id": 13,
            "name": "Beijing Capital",
            "city": "Beijing",
            "country": "China",
            "time_zone": 8
        },
        {
            "id": 14,
            "name": "Istanbul Airport",
            "city": "Istanbul",
            "country": "Turkey",
            "time_zone": 3
        },
        {
            "id": 15,
            "name": "Amsterdam Schiphol",
            "city": "Amsterdam",
            "country": "Netherlands",
            "time_zone": 1
        },
        {
            "id": 16,
            "name": "Zurich Airport",
            "city": "Zurich",
            "country": "Switzerland",
            "time_zone": 1
        }
    ]
```

```
}
```

Status Code: 200

10. Create new flights

API Request Description

This endpoint allows an airline to create a new flight by sending a POST request to the specified URL. The request must include the local departure date/time, flight duration, departure and destination airport IDs, and the aircraft ID. The system will convert the local departure time to UTC based on the departure airport's time zone and store it in the database.

BE AWARE

This endpint does verify if the aircraft is avaliable (not in use during the flight time period) but it does not verify if it is geographically available (last landing)...

Request Body

- `liftOffDateLOCAL` : (text, ISO 8601) The local departure date and time of the departure airport (e.g., `"2025-07-01T10:00:00"`).
- `duration` : (integer) The flight duration in minutes.
- `routeDeparture` : (integer) The ID of the departure airport.
- `routeDestination` : (integer) The ID of the destination airport.
- `aircraftId` : (integer) The ID of the aircraft to be used.

Response

Upon successful execution, the server will respond with a status code of 201 and a JSON object containing a message and the created flight object (with the UTC liftoff date).

Endpoint:

```
Method: POST
Type: RAW
URL: {{ServerIpAddress}}{{serverPort}}/api/navigate/flights
```

Body:

```
{
    "liftOffDateLOCAL": "2025-06-21T12:00:00",
    "duration": 91,
    "routeDeparture": 1,
    "routeDestination": 7,
    "aircraftId": 2
}
```

More example Requests/Responses:

I. Example Request: Fail route enrollment

Body:

```
{
    "liftOffDateLOCAL": "2025-10-15 ",
    "duration": 91,
    "routeDeparture": 1,
    "routeDestination": 4,
    "aircraftId": 5
}
```

I. Example Response: Fail route enrollment

```
{
    "error": "Airline is not enrolled in the specified route"
}
```

Status Code: 403

II. Example Request: Fail route error

Body:

```
{
    "liftOffDateLOCAL": "2025-10-15 ",
    "duration": 91,
    "routeDeparture": 1,
    "routeDestination": 4,
    "aircraftId": 5
}
```

II. Example Response: Fail route error

```
{
    "error": "Route not found between the specified airports"
}
```

Status Code: 404

III. Example Request: Fail Date error

Body:

```
{
    "liftOffDateLOCAL": "2025-01-05 ",
    "duration": 91,
    "routeDeparture": 4,
    "routeDestination": 1,
    "aircraftId": 5
}
```

III. Example Response: Fail Date error

```
{
    "error": "Liftoff date cannot be in the past"
}
```

Status Code: 400

IV. Example Request: Create new flights

Body:

```
{
    "liftOffDateLOCAL": "2025-07-01T10:00:00",
    "duration": 91,
    "routeDeparture": 4,
    "routeDestination": 1,
    "aircraftId": 5
}
```

IV. Example Response: Create new flights

```
{
    "message": "New flight created successfully",
    "flight": {
        "code": "49418977-98c9-4d72-afc2-38b25fdb9ca6",
        "duration": 91,
        "aircraft_id": 5,
        "liftoff_date": "2025-07-01T08:00:00.000Z",
        "route_departure": 4,
        "route_destination": 1,
        "airline_name": "FlySafe"
    }
}
```


Status Code: 201

V. Example Request: Fail aircraft in use

Body:


```
{
    "liftOffDateLOCAL": "2025-07-01T12:00:00",
    "duration": 91,
    "routeDeparture": 4,
    "routeDestination": 1,
    "aircraftId": 5
}
```


V. Example Response: Fail aircraft in use


```
{
    "error": "Aircraft is already scheduled for a flight at the specified lift-off date"
}
```


Status Code: 400

VI. Example Request: Fail lift off date in the past

Body:


```
{
    "liftOffDateLOCAL": "2025-06-21T12:00:00",
    "duration": 91,
    "routeDeparture": 1,
    "routeDestination": 7,
    "aircraftId": 2
}
```


VI. Example Response: Fail lift off date in the past


```
{
    "error": "Lift-off date must be in the future"
}
```


Status Code: 400

11. Fetch flights based on user search parameters

API Request Description

This endpoint allows you to search for available flights (single-leg or multi-leg) based on departure/destination airports, date range, number of passengers, and class. The search window is converted from local time to UTC using the departure airport's time zone.

Endpoint:

```
Method: POST
Type: RAW
URL: {{ServerIpAddress}}{{serverPort}}/api/navigate/flights/search
```

Body:

```
{
    "routes" : [
        {
            "departure": 1,
            "destination": 3
        },
        {
            "departure": 1,
            "destination": 14
        }
    ],
    "searchStartDateLOCAL": "2025-06-25 18:00:00",
    "searchEndDateLOCAL": "2025-07-25 20:00:00",
    "passengers": 1,
    "classType": "ECONOMY"
}
```

More example Requests/Responses:

I. Example Request: Fail example

Body:

```
{
    "routes" : [
        {
        "departure": 1,
        "destination": 2
        },
        {
            "departure": 2,
            "destination": 12
        }
    ],
    "searchStartDate": "2025-05-14 18:00:00",
    "searchEndDate": "2025-05-14 19:00:00",
    "passengers": 1,
    "classType": "ECONOMY"
}
```

I. Example Response: Fail example

```
{
    "message": "Flights with multiple legs retrieved successfully",
    "flights": []
}
```

Status Code: 200

II. Example Request: Fetch flights single route

Body:

```
{
    "routes" : [
        {
        "departure": 4,
        "destination": 1
        }
    ],
    "searchStartDateLOCAL": "2025-06-20 18:00:00",
    "searchEndDateLOCAL": "2025-07-25 20:00:00",
    "passengers": 1,
    "classType": "ECONOMY"
}
```


II. Example Response: Fetch flights single route


```
{
    "message": "Flights for single leg retrieved successfully",
    "startDateUtc": "2025-06-20T16:00:00.000Z",
    "endDateUtc": "2025-07-25T18:00:00.000Z",
    "departureAirport": {
        "time_zone": "Europe/Rome"
    },
    "flights": [
        {
            "code": "49418977-98c9-4d72-afc2-38b25fdb9ca6",
            "duration": 91,
            "aircraft_id": 5,
            "liftoff_date": "2025-07-01T08:00:00.000Z",
            "route_departure": 4,
            "route_destination": 1,
            "airline_name": "FlySafe",
            "aircrafts": {
                "id": 5,
                "model": "Boeing 777",
                "seats_capacity": 396,
                "owner_name": "FlySafe"
            },
            "routes": {
                "departure": 4,
                "destination": 1,
                "airports_routes_departureToairports": {
                    "id": 4,
                    "name": "Malpensa",
                    "city": "Milano",
                    "country": "Italy",
                    "lat": 45.630062,
                    "lan": 8.723068,
                    "time_zone": "Europe/Rome"
                },
                "airports_routes_destinationToairports": {
                    "id": 1,
                    "name": "Dubai International",
                    "city": "Dubai",
                    "country": "United Arab Emirates",
                    "lat": 25.253174,
                    "lan": 55.365673,
                    "time_zone": "Asia/Dubai"
                }
            },
            "liftoff_date_LOCAL": "2025-07-01T10:00:00.000+02:00"
        },
        {
            "code": "07cccead-eeac-4614-bc46-f095df9d9a7b",
            "duration": 91,
```

```
            "aircraft_id": 5,
            "liftoff_date": "2025-07-01T10:00:00.000Z",
            "route_departure": 4,
            "route_destination": 1,
            "airline_name": "FlySafe",
            "aircrafts": {
                "id": 5,
                "model": "Boeing 777",
                "seats_capacity": 396,
                "owner_name": "FlySafe"
            },
            "routes": {
                "departure": 4,
                "destination": 1,
                "airports_routes_departureToairports": {
                    "id": 4,
                    "name": "Malpensa",
                    "city": "Milano",
                    "country": "Italy",
                    "lat": 45.630062,
                    "lan": 8.723068,
                    "time_zone": "Europe/Rome"
                },
                "airports_routes_destinationToairports": {
                    "id": 1,
                    "name": "Dubai International",
                    "city": "Dubai",
                    "country": "United Arab Emirates",
                    "lat": 25.253174,
                    "lan": 55.365673,
                    "time_zone": "Asia/Dubai"
                }
            },
            "liftoff_date_LOCAL": "2025-07-01T12:00:00.000+02:00"
        }
    ]
}
```


Status Code: 200

III. Example Request: Fetch flights multi leg

Body:


```
{
    "routes" : [
        {
        "departure": 4,
        "destination": 1
        },
        {
            "departure": 1,
            "destination": 14
        }
    ],
    "searchStartDateLOCAL": "2025-06-25 18:00:00",
    "searchEndDateLOCAL": "2025-07-25 20:00:00",
    "passengers": 1,
    "classType": "ECONOMY"
}
```


III. Example Response: Fetch flights multi leg


```
{
```

```
{
    "message": "Flights with multiple legs retrieved successfully",
    "startDateUtc": "2025-06-25T16:00:00.000Z",
    "endDateUtc": "2025-07-25T18:00:00.000Z",
    "departureAirport": {
        "time_zone": "Europe/Rome"
    },
    "flights": [
        [
            {
                "code": "49418977-98c9-4d72-afc2-38b25fdb9ca6",
                "duration": 91,
                "aircraft_id": 5,
                "liftoff_date": "2025-07-01T08:00:00.000Z",
                "route_departure": 4,
                "route_destination": 1,
                "airline_name": "FlySafe",
                "aircrafts": {
                    "id": 5,
                    "model": "Boeing 777",
                    "seats_capacity": 396,
                    "owner_name": "FlySafe"
                },
                "routes": {
                    "departure": 4,
                    "destination": 1,
                    "airports_routes_departureToairports": {
                        "id": 4,
                        "name": "Malpensa",
                        "city": "Milano",
                        "country": "Italy",
                        "lat": 45.630062,
                        "lan": 8.723068,
                        "time_zone": "Europe/Rome"
                    },
                    "airports_routes_destinationToairports": {
                        "id": 1,
                        "name": "Dubai International",
                        "city": "Dubai",
                        "country": "United Arab Emirates",
                        "lat": 25.253174,
                        "lan": 55.365673,
                        "time_zone": "Asia/Dubai"
                    }
                }
            },
            {
                "code": "b6916970-f7a7-452e-b969-1e7d7defbaed",
                "duration": 91,
                "aircraft_id": 2,
                "liftoff_date": "2025-07-01T10:00:00.000Z",
                "route_departure": 1,
                "route_destination": 14,
                "airline_name": "FlySafe",
                "aircrafts": {
                    "id": 2,
                    "model": "Boeing 747-8",
                    "seats_capacity": 400,
                    "owner_name": "FlySafe"
                },
                "routes": {
                    "departure": 1,
                    "destination": 14,
                    "airports_routes_departureToairports": {
                        "id": 1,
                        "name": "Dubai International",
```

```
                        "city": "Dubai",
                        "country": "United Arab Emirates",
                        "lat": 25.253174,
                        "lan": 55.365673,
                        "time_zone": "Asia/Dubai"
                    },
                    "airports_routes_destinationToairports": {
                        "id": 14,
                        "name": "Istanbul Airport",
                        "city": "Istanbul",
                        "country": "Turkey",
                        "lat": 41.275278,
                        "lan": 28.751944,
                        "time_zone": "Europe/Istanbul"
                    }
                }
            }
        ]
    ]
}
```


Status Code: 200

12. Get flight details on UUID

API Request Description

This endpoint allows you to retrieve detailed information about a specific flight by providing its unique flight UUID. The response includes aircraft details, route information, and the local departure time based on the departure airport's time zone.

Endpoint:

```
Method: GET
Type: RAW
URL: {{ServerIpAddress}}{{serverPort}}/api/navigate/flights/:flightUUID
```

URL variables:

Key	Value	Description
flightUUID	187f190c-505e-425a-bf49-bebcbd9eaca4	

Body:


```
{
    "routes" : [
        {
        "departure": 4,
        "destination": 1
        }
    ],
    "searchStartDateLOCAL": "2025-06-20 18:00:00",
    "searchEndDateLOCAL": "2025-07-25 20:00:00",
    "passengers": 1,
    "classType": "ECONOMY"
}
```


More example Requests/Responses:

I. Example Request: Get flight details on UUID

Query:

Key	Value	Description

Key	Value	Description
flightUUID	187f190c-505e-425a-bf49-bebcbd9eaca4	

Body:

```
{
    "routes" : [
        {
        "departure": 4,
        "destination": 1
        }
    ],
    "searchStartDateLOCAL": "2025-06-20 18:00:00",
    "searchEndDateLOCAL": "2025-07-25 20:00:00",
    "passengers": 1,
    "classType": "ECONOMY"
}
```

I. Example Response: Get flight details on UUID

```
{
    "message": "Flight details retrieved successfully",
    "flight": {
        "code": "187f190c-505e-425a-bf49-bebcbd9eaca4",
        "duration": 91,
        "aircraft_id": 5,
        "liftoff_date": "2025-10-14T22:00:00.000Z",
        "route_departure": 4,
        "route_destination": 1,
        "airline_name": "FlySafe",
        "aircrafts": {
            "id": 5,
            "model": "Boeing 777",
            "seats_capacity": 396,
            "owner_name": "FlySafe"
        },
        "routes": {
            "departure": 4,
            "destination": 1,
            "airports_routes_departureToairports": {
                "id": 4,
                "name": "Malpensa",
                "city": "Milano",
                "country": "Italy",
                "lat": 45.630062,
                "lan": 8.723068,
                "time_zone": "Europe/Rome"
            },
            "airports_routes_destinationToairports": {
                "id": 1,
                "name": "Dubai International",
                "city": "Dubai",
                "country": "United Arab Emirates",
                "lat": 25.253174,
                "lan": 55.365673,
                "time_zone": "Asia/Dubai"
            }
        },
        "liftoff_date_LOCAL": "2025-10-15T00:00:00.000+02:00"
    }
}
```

Status Code: 200

13. Delete flight

API Request Description

This endpoint allows an airline to delete a specific flight by providing its unique flight UUID. The flight must belong to the airline making the request.

Endpoint:

```
Method: DELETE
Type:
URL: {{ServerIpAddress}}{{serverPort}}/api/navigate/flights/:flightUUID
```

URL variables:

Key	Value	Description
flightUUID	fbd7353f-dcdc-4498-974e-81e35d11cd79	

More example Requests/Responses:

I. Example Request: Fail

Query:

Key	Value	Description
flightUUID	86f9c095-5142-4214-8fe3-1e5602f8824a	

Body: None

I. Example Response: Fail

```
{
    "error": "Flight not found or does not belong to the airline"
}
```

Status Code: 404

II. Example Request: Deletion

Query:

Key	Value	Description
flightUUID	fbd7353f-dcdc-4498-974e-81e35d11cd79	

Body: None

II. Example Response: Deletion

```
{
    "message": "Flight deleted successfully"
}
```

Status Code: 200

Bookings

1. Download booking

API Request Description

This endpoint allows an authenticated user to download a PDF ticket for a specific booking by providing the booking ID. The booking must belong to the requesting user. The PDF contains all relevant ticket, flight, and passenger details.

Endpoint:

```
Method: GET
Type:
URL: {{ServerIpAddress}}{{serverPort}}/api/bookings/booking/:id/download
```

URL variables:

Key	Value	Description
id	3	

More example Requests/Responses:

I. Example Request: Error

Query:

Key	Value	Description
UUID	e9e42b62-6e91-4136-b46d-85b88c4fae6d	

Body: None

I. Example Response: Error

```
{
    "error": "Ticket not found or does not belong to this user"
}
```

Status Code: 404

2. Get booking details

API Request Description

This endpoint allows an authenticated user to retrieve detailed information about a specific booking by providing the booking ID. The booking must belong to the requesting user. The response includes ticket, flight, seat, extras, and passenger details, as well as the local departure date.

Endpoint:

```
Method: GET
Type:
URL: {{ServerIpAddress}}{{serverPort}}/api/bookings/booking/:id
```

URL variables:

Key	Value	Description
id	2	

More example Requests/Responses:

I. Example Request: Error

Query:

Key	Value	Description
UUID	e9e42b62-6e91-4136-b46d-85b88c4fae6d	

Body: None

I. Example Response: Error

```
{
    "error": "Ticket not found or does not belong to this user"
}
```

Status Code: 404

II. Example Request: Get ticket details

Query:

Key	Value	Description
id	2	

Body: None

II. Example Response: Get ticket details

```
{
    "id": 2,
    "ticket_code": "36fb125b-3ee1-4340-a4af-10900e283c60",
    "seat_id": 2,
    "trip_id": 2,
    "extras_id": 2,
    "trips": {
        "id": 2,
        "creation_date": "2024-05-16T15:30:00.000Z",
        "user_id": 3,
        "users": {
            "id": 3,
            "name": "Admin",
            "email": "adminAccount@gmail.com",
            "password": "$2b$10$dTV3A7cZwWjBiqKrxEXwuuIJGDkIKqSJOizMDkdwL9s0jQK4bsNx6",
            "role": 1
        }
    },
    "tickets": {
        "code": "36fb125b-3ee1-4340-a4af-10900e283c60",
        "type": "BUSINESS",
        "price": 1200,
        "fligt_code": "e6f4746c-ee45-4a8d-9a52-2d2c4a6cf874",
        "flights": {
            "code": "e6f4746c-ee45-4a8d-9a52-2d2c4a6cf874",
            "duration": 728,
            "aircraft_id": 2,
            "liftoff_date": "2024-06-02T15:30:00.000Z",
            "route_departure": 2,
            "route_destination": 3,
            "airline_name": "Lufthansa",
            "routes": {
                "departure": 2,
                "destination": 3,
                "airports_routes_departureToairports": {
                    "id": 2,
                    "name": "Frankfurt Airport",
                    "time_zone": "Europe/Berlin"
                },
                "airports_routes_destinationToairports": {
                    "id": 3,
                    "name": "Changi Airport",
                    "time_zone": "Asia/Singapore"
                }
            }
        }
    },
    "extras": {
        "id": 2,
        "description": "Airport Lounge",
        "price": 75
    },
    "seats": {
        "id": 2,
        "postion": "A2",
        "aircraft_id": 1
    },
    "localDepartureDate": "2024-06-02T17:30:00.000+02:00"
}
```


Status Code: 200

3. List available tickets for flight

API Request Description

This endpoint allows you to retrieve all available ticket types for a specific flight by providing the flight UUID. Each ticket includes its type, price, and unique code.

BE AWARE:
This endpoints return even when the flight is already departured

Endpoint:

```
Method: GET
Type:
URL: {{ServerIpAddress}}{{serverPort}}/api/bookings/tickets/:flightUUID
```

URL variables:

Key	Value	Description
flightUUID	e5a39b2f-1320-4e4c-9d10-0cf960a2367d	

More example Requests/Responses:

I. Example Request: Tickets

Query:

Key	Value	Description
flightUUID	e5a39b2f-1320-4e4c-9d10-0cf960a2367c	

Body: None

I. Example Response: Tickets

```
[
    {
        "code": "9e2be6f0-c9a9-4b42-a076-e3feba827d7c",
        "type": "ECONOMY",
        "price": 300,
        "fligt_code": "e5a39b2f-1320-4e4c-9d10-0cf960a2367c"
    },
    {
        "code": "8c7c396a-d3e2-4082-9578-9931959d8455",
        "type": "BUSINESS",
        "price": 300,
        "fligt_code": "e5a39b2f-1320-4e4c-9d10-0cf960a2367c"
    }
]
```

Status Code: 200

II. Example Request: Error

Query:

Key	Value	Description
flightUUID	e5a39b2f-1320-4e4c-9d10-0cf960a2367d	

Body: None

II. Example Response: Error

```
{
    "error": "No tickets found for this flight"
}
```

Status Code: 404

4. List available seats for flight

API Request Description

This endpoint allows you to retrieve all available (not yet booked) seats for a specific flight by providing the flight UUID. Only seats that are not already booked for the flight are returned.

BE AWARE:
This endpoints return even when the flight is already departured

Endpoint:

```
Method: GET
Type:
URL: {{ServerIpAddress}}{{serverPort}}/api/bookings/seats/:flightUUID
```

URL variables:

Key	Value	Description
flightUUID	e5a39b2f-1320-4e4c-9d10-0cf960a2367c	

5. Get list of extras

API Request Description

This endpoint allows you to retrieve the list of all available extras (such as additional services or products) that can be added to a flight booking.

Endpoint:

```
Method: GET
Type:
URL: {{serverEndpointDev}}{{serverPort}}/api/bookings/extras
```

More example Requests/Responses:

I. Example Request: extras

Body: None

I. Example Response: extras

```
[
    {
        "id": 1,
        "description": "Extra Baggage",
        "price": 50
    },
    {
        "id": 2,
        "description": "Airport Lounge",
        "price": 75
    }
]
```

Status Code: 200

6. Book flights

API Request Description

This endpoint allows an authenticated user to create a new trip by booking one or more tickets, each with a selected seat and optional extras. The endpoint validates

ticket, seat, and extra IDs, ensures flights are not departed, and prevents double-booking of seats.

This response confirms that the trip and all associated bookings have been successfully created. If any validation fails (invalid ticket, seat, extra, or already booked/departed), an error message is returned.

Endpoint:

```
Method: POST
Type: RAW
URL: {{serverEndpointDev}}{{serverPort}}/api/bookings/trips
```

Body:

```
{
"tickets": [
            {
                "ticketUUID": "138f5e6a-8770-4e79-a78c-6dc2960d41c5",
                "seatId": 2625,
                "extras": [1]
            }
            /*{
                "ticketUUID": "5d7fe786-8e13-4657-846a-30621a0c3afd",
                "seatId": 2078
            }*/
        ]
}
```

More example Requests/Responses:

I. Example Request: Invalid request

Body:

```
{
"tickets": [
            {
                "ticketUUID": "5d7fe786-8e13-4657-846a-30621a0c3afd"
            },
            {
                "ticketUUID": "5d7fe786-8e13-4657-846a-30621a0c3afd",
                "seatId": 2078
            }
        ]
}
```

I. Example Response: Invalid request

```
{
    "error": "Each ticket must have a ticketUUID and seatId"
}
```

Status Code: 400

II. Example Request: Invalid ticket uuid

Body:

```
{
"tickets": [
            {
                "ticketUUID": "5d7fe786-8e13-4657-846a-30621a0c3afd",
                "seatId": 2078
            },
            {
                "ticketUUID": "4d7fe786-8e13-4657-846a-30621a0c3afd",
                "seatId": 2078
            }
        ]
}
```

II. Example Response: Invalid ticket uuid

```
{
    "error": "Invalid ticket UUIDs provided"
}
```

Status Code: 400

III. Example Request: Flight already departured

Body:

```
{
"tickets": [
            {
                "ticketUUID": "5d7fe786-8e13-4657-846a-30621a0c3afd",
                "seatId": 2076
            },
            {
                "ticketUUID": "e8752daf-7f5d-43af-9ee8-54c08e12255b",
                "seatId": 2078
            }
        ]
}
```

III. Example Response: Flight already departured

```
{
    "error": "Some flights yout are trying to book have already departed"
}
```

Status Code: 400

IV. Example Request: Seat not present in aircraft for ticket booking

Body:

```
{
"tickets": [
            {
                "ticketUUID": "5d7fe786-8e13-4657-846a-30621a0c3afd",
                "seatId": 2076
            },
            {
                "ticketUUID": "5d7fe786-8e13-4657-846a-30621a0c3afd",
                "seatId": 2078
            }
        ]
}
```

IV. Example Response: Seat not present in aircraft for ticket booking

```
{
    "error": "Invalid seat IDs for the selected flights"
}
```

Status Code: 400

V. Example Request: Seats already booked

Body:

```
{
"tickets": [
            {
                "ticketUUID": "5d7fe786-8e13-4657-846a-30621a0c3afd",
                "seatId": 2077
            },
            {
                "ticketUUID": "5d7fe786-8e13-4657-846a-30621a0c3afd",
                "seatId": 2078
            }
        ]
}
```

V. Example Response: Seats already booked

```
{
    "error": "Seat 2077,2078 is already booked for flight 5d7fe786-8e13-4657-846a-30621a0c3afd,5d7fe786-8e13-4657-846a-30621a0c3afd"
}
```

Status Code: 400

VI. Example Request: Trip created succesfully

Body:

```
{
"tickets": [
            {
                "ticketUUID": "5d7fe786-8e13-4657-846a-30621a0c3afd",
                "seatId": 2077
            },
            {
                "ticketUUID": "5d7fe786-8e13-4657-846a-30621a0c3afd",
                "seatId": 2078
            }
        ]
}
```

VI. Example Response: Trip created succesfully

```
{
    "message": "Trip created successfully",
    "tripId": 3,
    "tickets": [
        {
            "ticketUUID": "5d7fe786-8e13-4657-846a-30621a0c3afd",
            "seatId": 2077,
            "extras": []
        },
        {
            "ticketUUID": "5d7fe786-8e13-4657-846a-30621a0c3afd",
            "seatId": 2078,
            "extras": []
        }
    ]
}
```

Status Code: 200

7. list booked flights

API Request Description

This endpoint allows an authenticated user to retrieve a list of all their trips. Each trip includes its unique ID and a boolean flag indicating whether the trip is active (i.e., at least one associated flight has not yet departed).

Endpoint:

```
Method: GET
Type:
URL: {{ServerIpAddress}}{{serverPort}}/api/bookings/trips
```

More example Requests/Responses:

I. Example Request: list booked flights

Body: None

I. Example Response: list booked flights

```
[
    {
        "id": 3,
        "isActive": true
    }
]
```

Status Code: 200

8. get booking details

API Request Description

This endpoint allows an authenticated user to retrieve detailed information about a specific trip by providing the trip ID. The trip must belong to the requesting user. The response includes all bookings for the trip, each with ticket, flight, and extras details, as well as an isActive flag indicating if any flight in the trip is in the future.

Endpoint:

```
Method: GET
Type:
URL: {{ServerIpAddress}}{{serverPort}}/api/bookings/trips/:tripId
```

URL variables:

Key	Value	Description
tripId	3	

More example Requests/Responses:

I. Example Request: Error

Query:

Key	Value	Description
tripId	4	

Body: None

I. Example Response: Error

```
{
    "error": "Trip not found or does not belong to this user"
}
```

Status Code: 404

II. Example Request: list booked flights

Query:

Key	Value	Description
tripId	3	

Body: None

II. Example Response: list booked flights

```
{
    "id": 3,
    "creation_date": "2025-06-21T10:40:46.008Z",
    "user_id": 4,
    "bookings": [
        {
            "id": 3,
            "ticket_code": "5d7fe786-8e13-4657-846a-30621a0c3afd",
            "seat_id": 2077,
            "trip_id": 3,
            "extras_id": null,
            "tickets": {
                "code": "5d7fe786-8e13-4657-846a-30621a0c3afd",
                "type": "BASE",
                "price": 22,
                "fligt_code": "a4431347-a2bb-491d-87ea-ecacaaefeb1f",
                "flights": {
                    "code": "a4431347-a2bb-491d-87ea-ecacaaefeb1f",
                    "duration": 230,
                    "aircraft_id": 7,
                    "liftoff_date": "2025-06-23T18:00:00.000Z",
                    "route_departure": 1,
                    "route_destination": 7,
                    "airline_name": "FlySafe"
                }
            },
            "extras": null
        },
        {
            "id": 4,
            "ticket_code": "5d7fe786-8e13-4657-846a-30621a0c3afd",
            "seat_id": 2078,
            "trip_id": 3,
            "extras_id": null,
            "tickets": {
                "code": "5d7fe786-8e13-4657-846a-30621a0c3afd",
                "type": "BASE",
                "price": 22,
                "fligt_code": "a4431347-a2bb-491d-87ea-ecacaaefeb1f",
                "flights": {
                    "code": "a4431347-a2bb-491d-87ea-ecacaaefeb1f",
                    "duration": 230,
                    "aircraft_id": 7,
                    "liftoff_date": "2025-06-23T18:00:00.000Z",
                    "route_departure": 1,
                    "route_destination": 7,
                    "airline_name": "FlySafe"
                }
            },
            "extras": null
        }
    ],
    "isActive": true
}
```


Status Code: 200

Ungrouped

1. Get UTC time

API Request Description

This endpoint retrieves the current UTC time. It does not require any query parameters or request body.

Response

The response will be in JSON format and includes the following key:

- **`utcTime`** : A string representing the current UTC time.

Endpoint:

```
Method: GET
Type:
URL: {{serverEndpointDev}}{{serverPort}}/api/utcTime
```

More example Requests/Responses:

I. Example Request: UtcTime

Body: None

I. Example Response: UtcTime

```
{
    "utcTime": "2025-06-19T09:24:53.728Z"
}
```

Status Code: 200

Frontend Layer

The frontend layer is responsible for providing a user interface for the application. It allows users to interact with the backend services, view flight information, book tickets, and manage their bookings.

As per project requirements, the frontend layer is implemented using AngularJS. It communicates with the backend services through the RESTful APIs descripted above.

The only environment variable used in the frontend is the `apiUrl`, which is set to the IP address of the server where the backend services are hosted. This variable is used to construct the URLs for API requests.

The frontend is organized into several components, each responsible for a specific part of the user interface.

Components

- `aircraft-card`
  - Displays information about an aircraft (ID, model, number of seats) and allows deletion with confirmation.
- `airline-card`
  - Shows details about an airline, such as name, country, and motto.
- `airline-flight-card`
  - Displays information about a single flight operated by an airline
- `airport-card`
  - Displays detailed information about a single airport. It typically shows the airport's name, city, country, and possibly additional details such as its time zone or geographic coordinates.
- `flight-card`
  - Displays detailed information about a single flight. It typically shows key flight details.
- `flight-list`
  - Responsible for displaying a list of flights.
- `flight-search`
  - Provides a user interface for searching flights allowing users to specify search parameters such as departure and destination airports, date range, number of passengers, and class type.
- `map`
  - Displays a map with airports.
- `map-routing`
  - Handles routing on the map, allowing users to visualize flight routes between airports.
- `route-card`
  - Displays detailed information about a specific route between two airports.
- `toolbar`
  - Navigation bar
- `trip-card`
  - Displays detailed information about a user's trip (booking).
- `user-card`
  - Handles user-related information, such as user details and actions like deleting the user profile.

These components include are then placed inside different pages.

Pages

- `admin-dashboard`
- `airline-enrollment`
- `airline-login`
- `flights-display`
- `forbitten`
- `homepage`
- `homepage-airline`
- `profiles`
- `server-error`
- `singin`
- `ticket-booking`
- `user-registration`

The `roleAuthGuard` is used to protect routes that require authentication and specific user roles. If a user tries to access a protected route without the required role, they are redirected to the `/forbidden` page. Angular HTTP interceptors are used to detect if a user's JWT (JSON Web Token) has expired or to handle server errors globally.

Authentication and Authorization

The application uses JWT (JSON Web Token) for authentication and authorization. The JWT is stored in the browser's local storage after a user or an airline logs in or registers.

Airline JWT

```
{
  "airlineName": "FlySafe",
  "country": "Francia",
  "motto": "Vive la france",
  "role": 2,
  "iat": 1750707581,
  "exp": 1750711181
}
```

User JWT (admin with role 1)

```
{
  "userId": 4,
  "name": "Admin",
  "email": "adminAccount@gmail.com",
  "role": 1,
  "iat": 1750707775,
  "exp": 1750711375
}
```

Application live examples

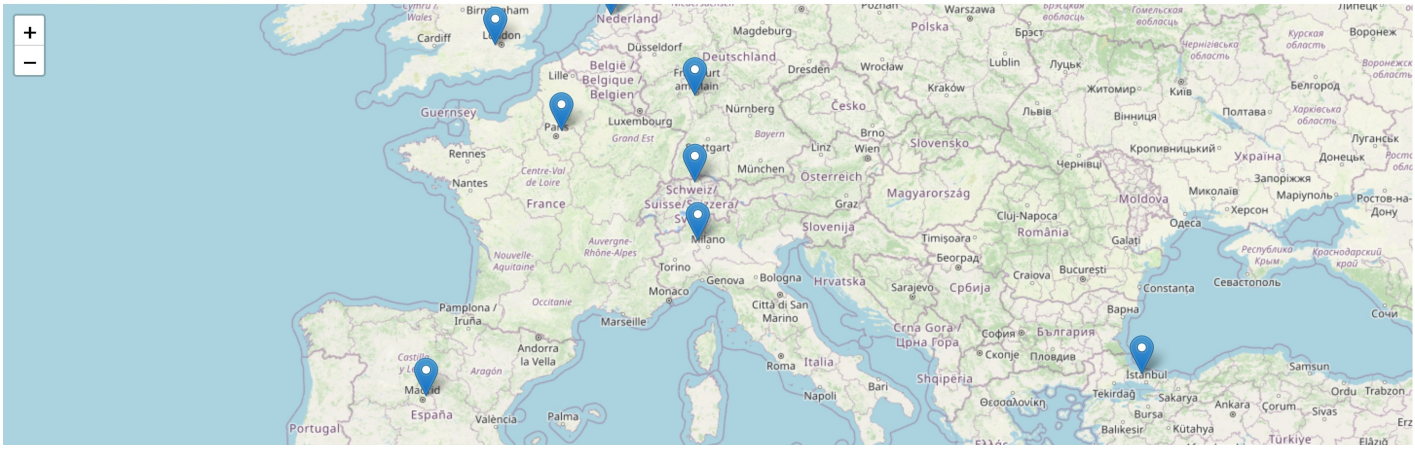
No auth:

Benvenuto su TAW Flights!

Cerca voli tra centinaia di destinazioni. Acquista biglietti e gestisci le tue prenotazioni.

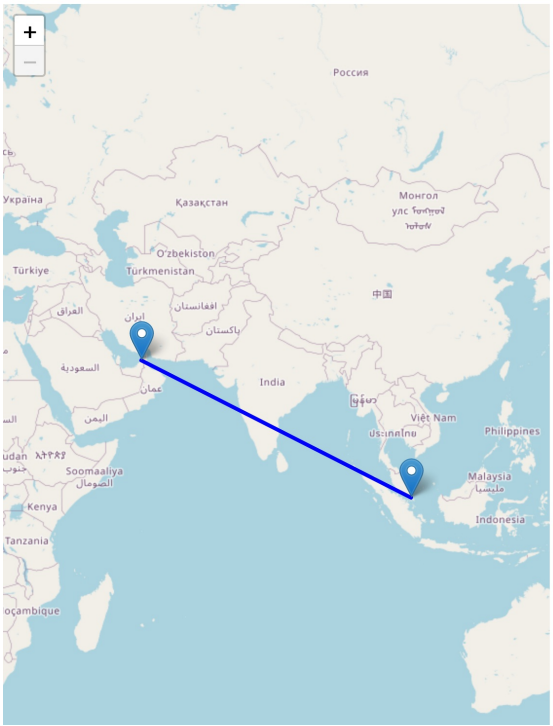
Tipo viaggio* Solo andata | Da* | A* | Data di partenza*

Viaggiatori* 1 | Classe* ECONOMY | Solo voli diretti | Cerca voli

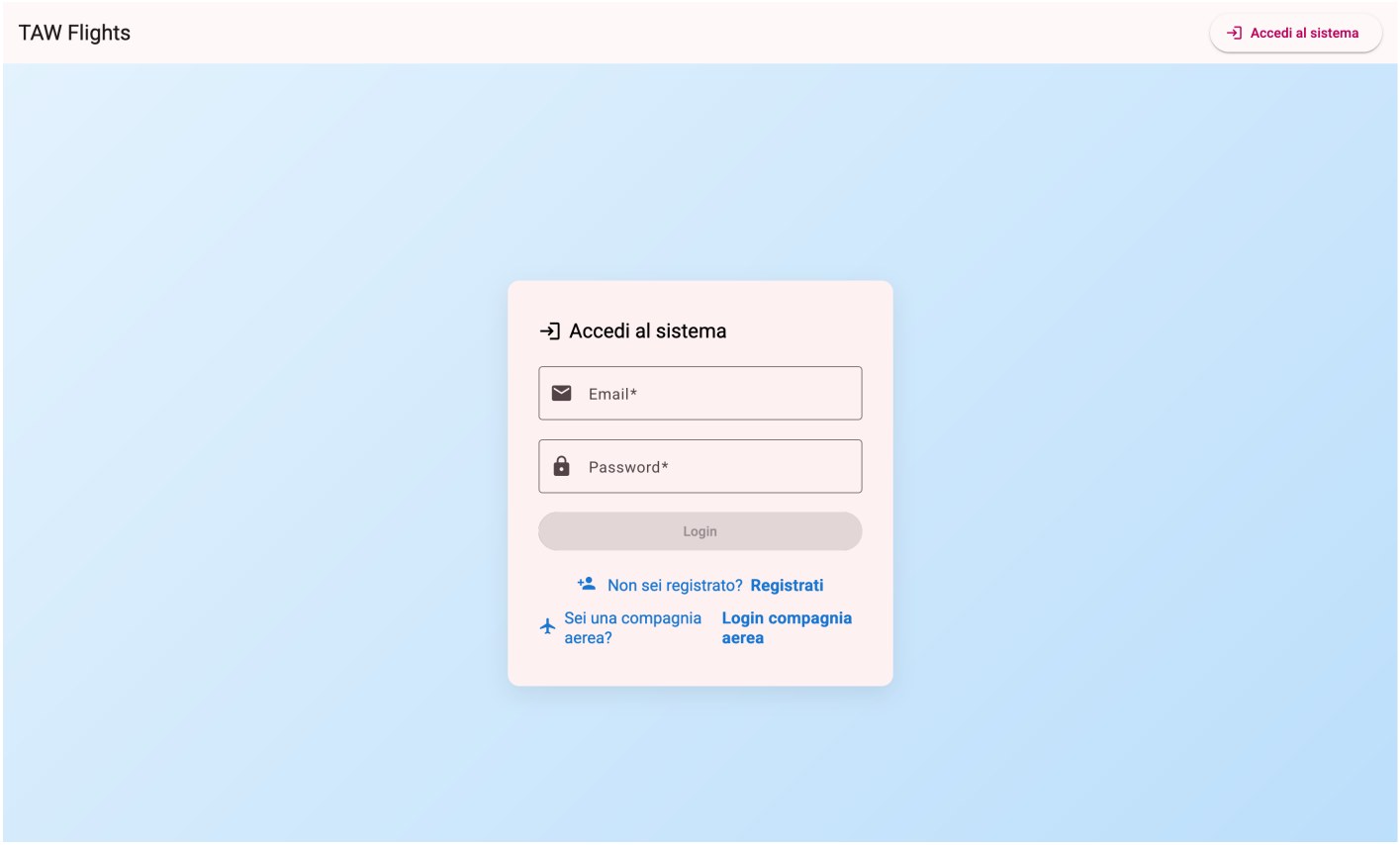


TAW Flights — Accedi al sistema

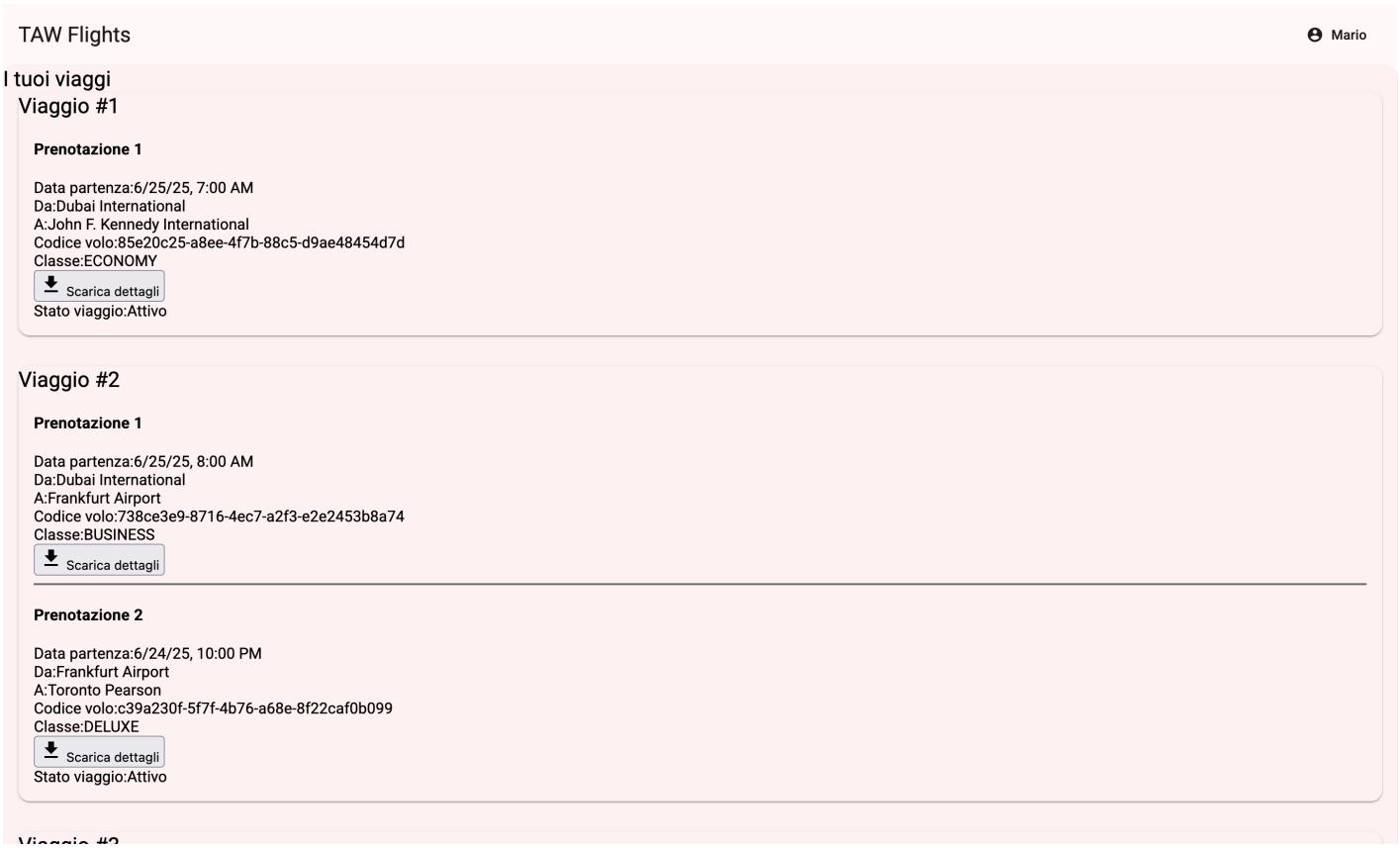
Risultati ricerca voli

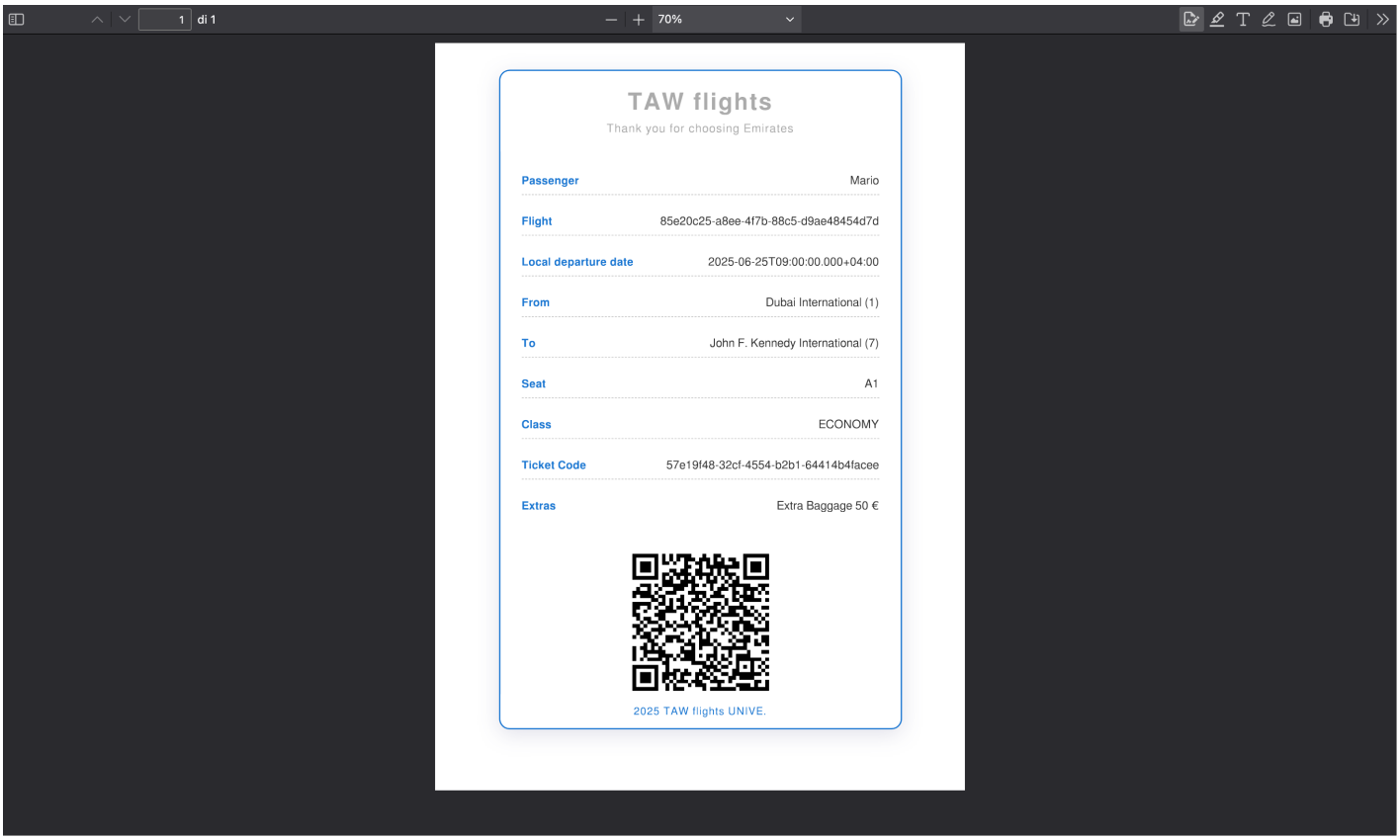


Singapore Airlines	11:00 Dubai International (Dubai)	6h 30m — Diretto	16:30 Changi Airport (Singapore)	€264

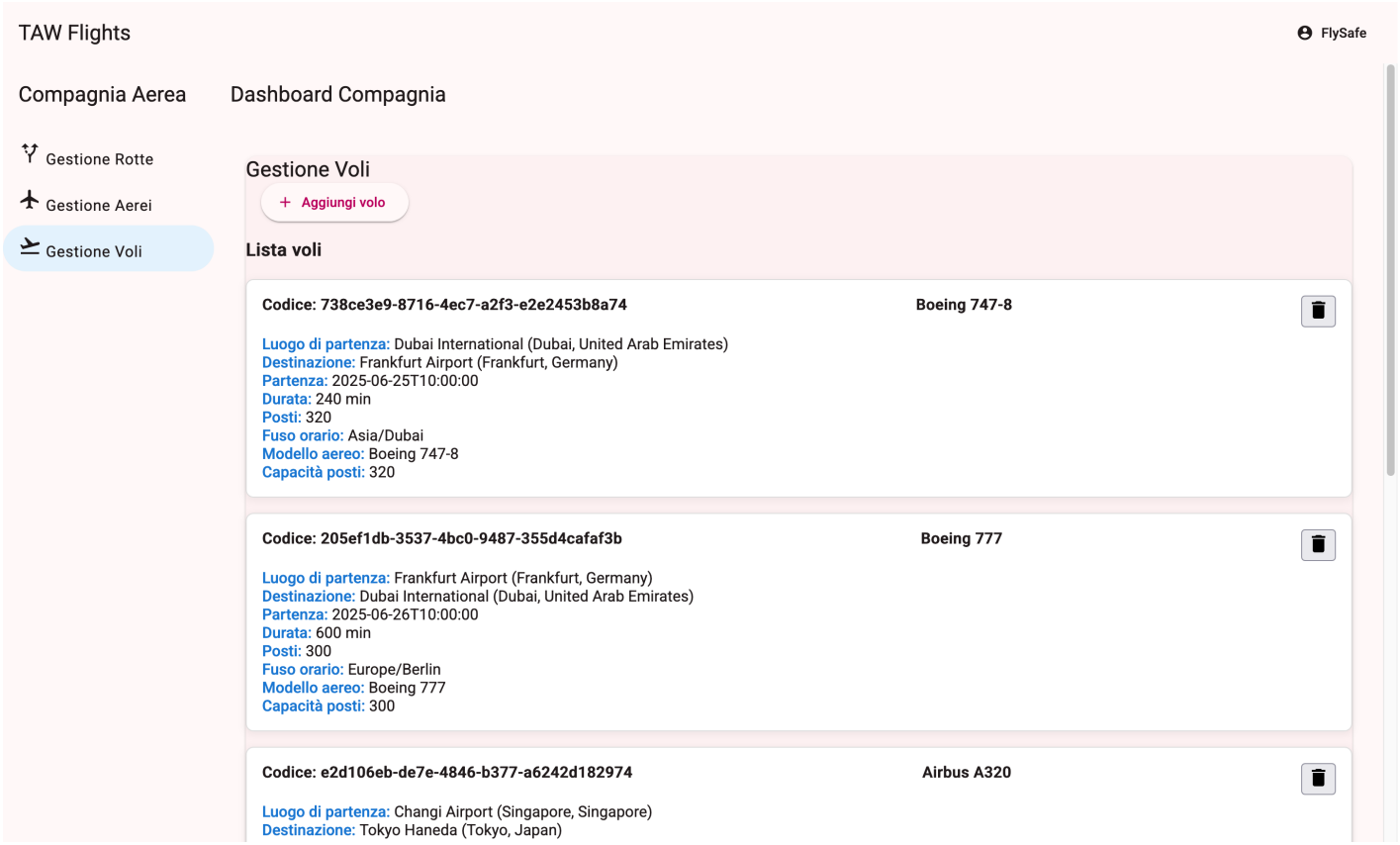

TAW Flights
Accedi al sistema
Accedi al sistema
Email*
Password*
Login
Non sei registrato? Registrati
Sei una compagnia aerea?
Login compagnia aerea

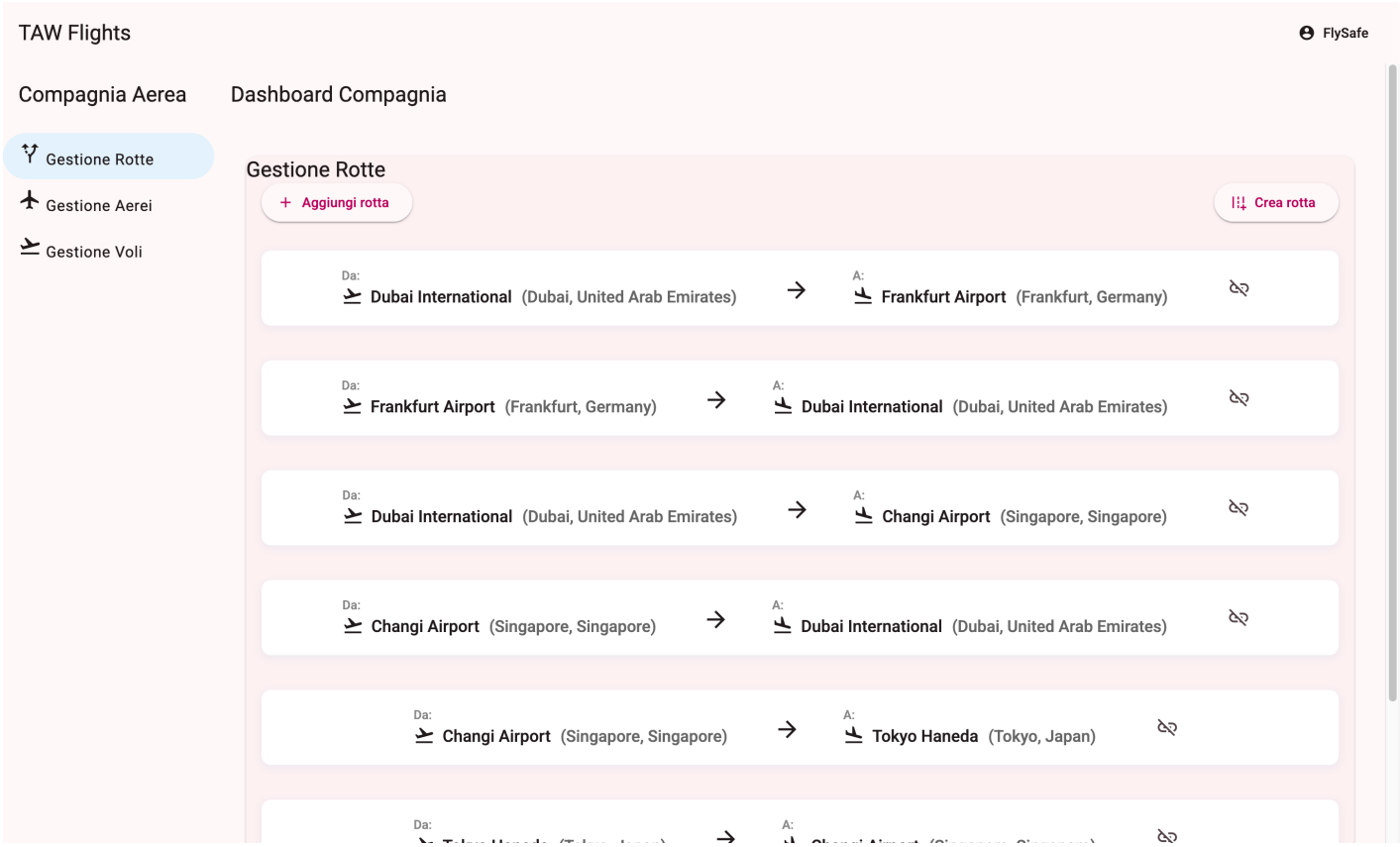
User pages:

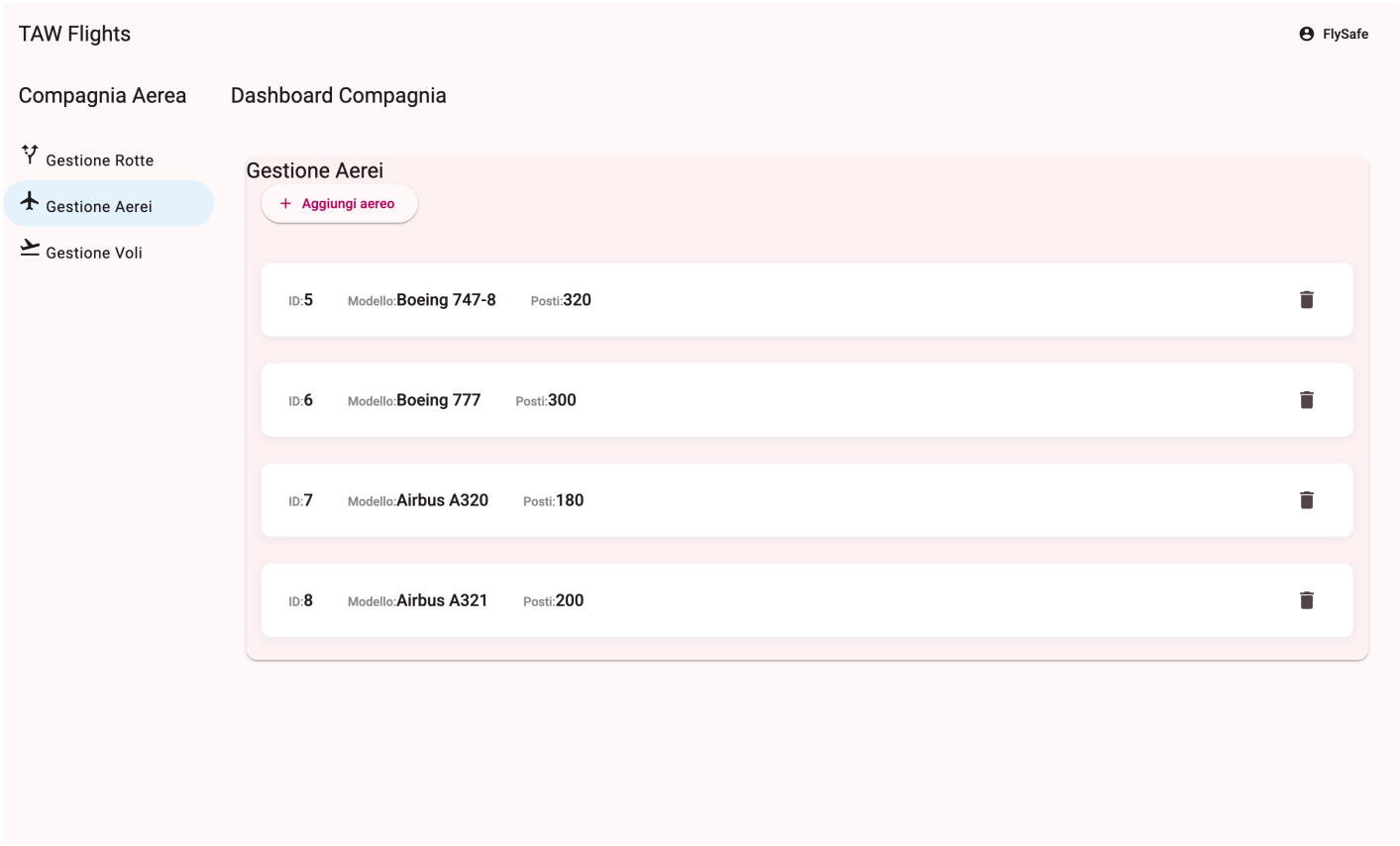

TAW Flights
Mario
I tuoi viaggi
Viaggio #1
Prenotazione 1
Data partenza:6/25/25, 7:00 AM
Da:Dubai International
A:John F. Kennedy International
Codice volo:85e20c25-a8ee-4f7b-88c5-d9ae48454d7d
Classe:ECONOMY
Scarica dettagli
Stato viaggio:Attivo
Viaggio #2
Prenotazione 1
Data partenza:6/25/25, 8:00 AM
Da:Dubai International
A:Frankfurt Airport
Codice volo:738ce3e9-8716-4ec7-a2f3-e2e2453b8a74
Classe:BUSINESS
Scarica dettagli
Prenotazione 2
Data partenza:6/24/25, 10:00 PM
Da:Frankfurt Airport
A:Toronto Pearson
Codice volo:c39a230f-5f7f-4b76-a68e-8f22caf0b099
Classe:DELUXE
Scarica dettagli
Stato viaggio:Attivo



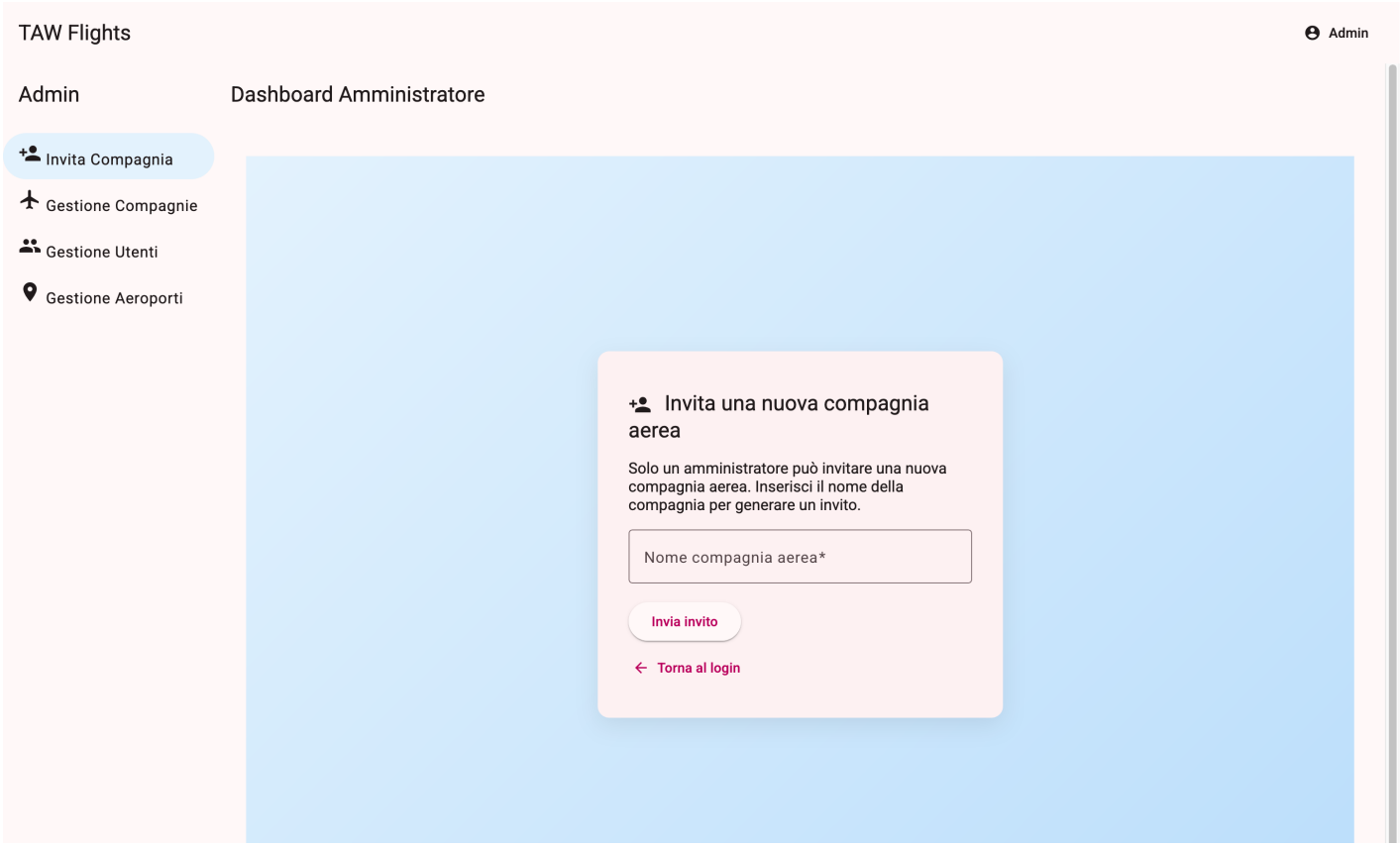
Airline pages:

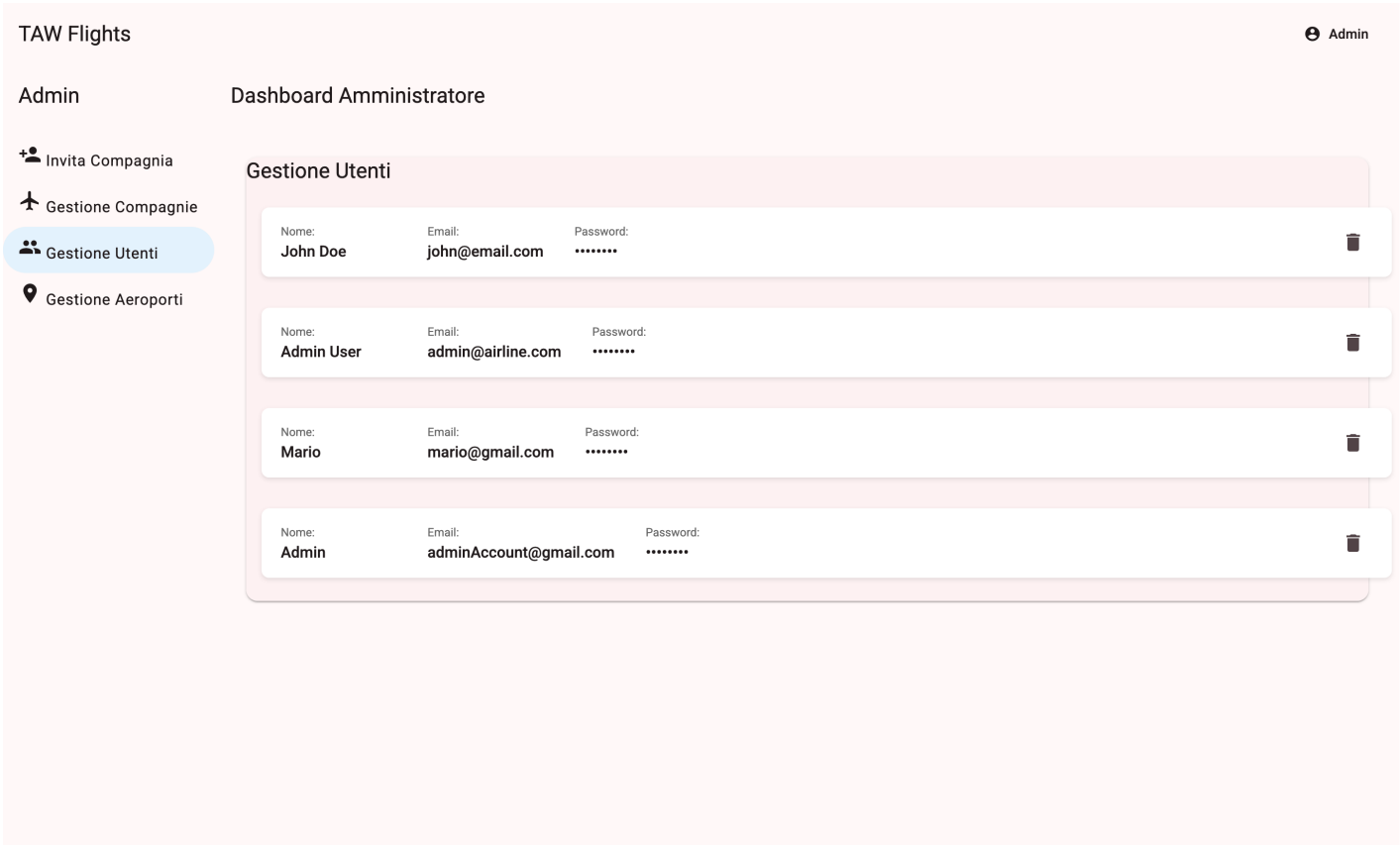



TAW Flights
FlySafe
Compagnia Aerea
Dashboard Compagnia
Gestione Rotte
Gestione Aerei
Gestione Voli
Gestione Rotte
Aggiungi rotta
Crea rotta
Da: Dubai International (Dubai, United Arab Emirates)
A: Frankfurt Airport (Frankfurt, Germany)
Da: Frankfurt Airport (Frankfurt, Germany)
A: Dubai International (Dubai, United Arab Emirates)
Da: Dubai International (Dubai, United Arab Emirates)
A: Changi Airport (Singapore, Singapore)
Da: Changi Airport (Singapore, Singapore)
A: Dubai International (Dubai, United Arab Emirates)
Da: Changi Airport (Singapore, Singapore)
A: Tokyo Haneda (Tokyo, Japan)


TAW Flights
FlySafe
Compagnia Aerea
Dashboard Compagnia
Gestione Rotte
Gestione Aerei
Gestione Voli
Gestione Aerei
Aggiungi aereo
ID:5 Modello:Boeing 747-8 Posti:320
ID:6 Modello:Boeing 777 Posti:300
ID:7 Modello:Airbus A320 Posti:180
ID:8 Modello:Airbus A321 Posti:200

Admin pages:


TAW Flights
Admin
Admin
Dashboard Amministratore
Invita Compagnia
Gestione Compagnie
Gestione Utenti
Gestione Aeroporti
Invita una nuova compagnia aerea
Solo un amministratore può invitare una nuova compagnia aerea. Inserisci il nome della compagnia per generare un invito.
Nome compagnia aerea*
Invia invito
Torna al login


TAW Flights
Admin
Admin
Dashboard Amministratore
Invita Compagnia
Gestione Compagnie
Gestione Utenti
Gestione Aeroporti
Gestione Utenti
Nome: John Doe
Email: john@email.com
Password: ••••••••
Nome: Admin User
Email: admin@airline.com
Password: ••••••••
Nome: Mario
Email: mario@gmail.com
Password: ••••••••
Nome: Admin
Email: adminAccount@gmail.com
Password: ••••••••

TAW Flights

Admin

Admin

Dashboard Amministratore

- Invita Compagnia
- Gestione Compagnie
- Gestione Utenti
- Gestione Aeroporti

Gestione Compagnie Aeree

Nome:	Paese:	Motto:
FlySafe	Francia	Vive la france
Emirates	United Arab Emirates	Fly Better
Lufthansa	Germany	Say yes to the world
Singapore Airlines	Singapore	A Great Way to Fly

TAW Flights

Admin

Dashboard Amministratore

- Invita Compagnia
- Gestione Compagnie
- Gestione Utenti
- Gestione Aeroporti

Gestione Aeroporti

Ordina per: Nome

Nome:	Città:	Nazione:	Fuso orario:
Amsterdam Schiphol	Amsterdam	Netherlands	UTCEurope/Amsterdam
Beijing Capital	Beijing	China	UTCAsia/Shanghai
Changi Airport	Singapore	Singapore	UTCAsia/Singapore
Charles de Gaulle	Paris	France	UTCEurope/Paris
Dubai International	Dubai	United Arab Emirates	UTCAsia/Dubai
Frankfurt Airport	Frankfurt	Germany	UTCEurope/Berlin
Heathrow	London	UK	UTCEurope/London
Istanbul Airport	Istanbul	Turkey	UTCEurope/Istanbul