



## **КУРСОВ ПРОЕКТ**

### **ПО БАЗИ ОТ ДАННИ**

**Студент: Асен Пантелеев Попов**

**ФАК. № 121223246**

**Група: 446**

#### **Тема №3**

Разработете база данни за система за менажиране на полетите на авиокомпания като се съхраняват деня, часа и продължителността на полета, начално и крайно летище, самолет, екипаж, тип на предлаганата храна. Да може да се изважда разписанията на полетите на компанията, нарядите на пилотите, възможностите за заместване на членове на екипажа и т.н.

#### **Задачи:**

- 1.** Да се проектира база от данни и да се представи ER диаграма със съответни CREATE TABLE заявки за средата MySQL.
- 2.** Напишете заявка, в която демонстрирате SELECT с ограничаващо условие по избор.
- 3.** Напишете заявка, в която използвате агрегатна функция и GROUP BY по ваш избор.
- 4.** Напишете заявка, в която демонстрирате INNER JOIN по ваш избор.
- 5.** Напишете заявка, в която демонстрирате OUTER JOIN по ваш избор.
- 6.** Напишете заявка, в която демонстрирате вложен SELECT по ваш избор.
- 7.** Напишете заявка, в която демонстрирате едновременно JOIN и агрегатна функция.
- 8.** Създайте тригер по ваш избор.
- 9.** Създайте процедура, в която демонстрирате използване на курсор.

## Съдържание:

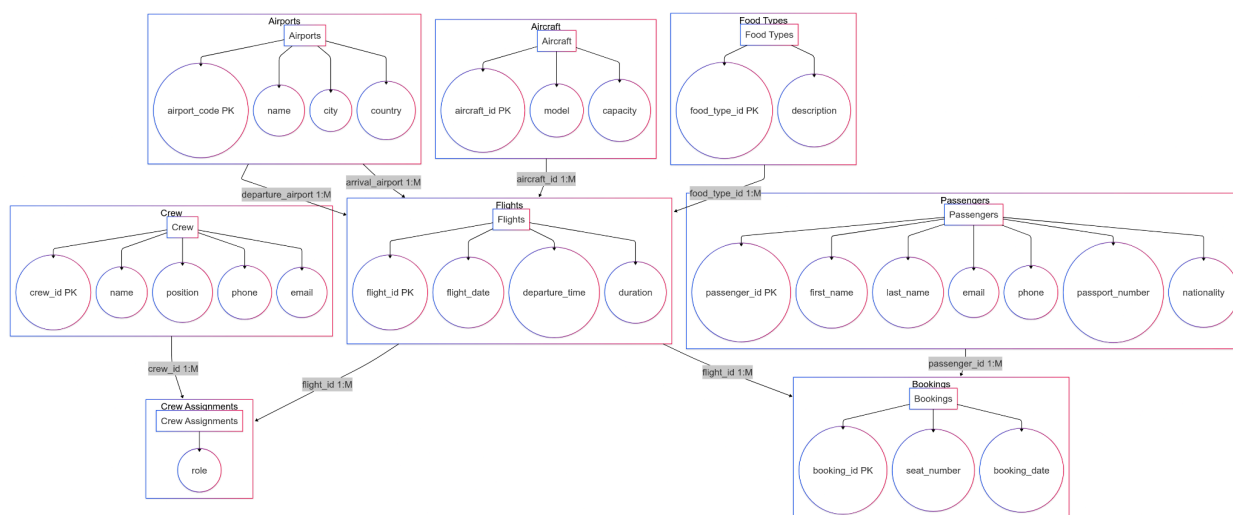
1. [Увод](#)
2. Зад. 1 Да се проектира база от данни и да се представи ER диаграма със съответни CREATE TABLE заявки за средата MySQL
3. Зад. 2 Напишете заявка, в която демонстрирате SELECT с ограничаващо условие по избор
4. Зад. 3 Напишете заявка, в която използвате агрегатна функция и GROUP BY по ваш избор
5. Зад. 4 Напишете заявка, в която демонстрирате INNER JOIN по ваш избор
6. Зад. 5 Напишете заявка, в която демонстрирате OUTER JOIN по ваш избор
7. Зад. 6 Напишете заявка, в която демонстрирате вложен SELECT по ваш избор
8. Зад. 7 Напишете заявка, в която демонстрирате едновременно JOIN и агрегатна функция
9. Зад. 8 Създайте тригер по ваш избор
10. Зад. 9 Създайте процедура, в която демонстрирате използване на курсор
11. [Django приложение за уеб интерфейс към базата](#)
12. Логически поток
13. Шаблони (templates) или Основните HTML страници
14. GitHub репозитория
15. Идеи за надграждане(roadmap)

## Увод

В тази документация е представена разработката на курсова работа по бази данни, Тема №3: „Система за менажиране на полетите на авиокомпания“. Проектът включва:

- Проектиране на релационна база данни в MySQL.
- Реализация на SQL скриптове за създаване, тестване, заявки, тригери и процедури.
- Django приложение за уеб интерфейс към базата.

**Зад. 1** Да се проектира база от данни и да се представи ER диаграма със съответни CREATE TABLE заявки за средата MySQL:



### Описание на таблици в ER-диаграмата:

#### - Airports

Съдържа справочна информация за летища.

- airport\_code (PK): уникален IATA код
- name, city, country: описателни полета  
Използва се от **Flights** при дефиниране на изходящо и входящо летище.

#### - Aircraft

Записва наличните самолети на авиокомпанията.

- aircraft\_id (PK)
- model, capacity: модел и брой седалки  
Свързва се едно-към-много към **Flights**, за да определи с кой самолет е изпълнен всеки полет.

### **-Food\_Types**

Каталог с типове кетъринг (например „Economy“, „Business“).

- food\_type\_id (PK)
- description: описание на менюто  
Използва се от **Flights** за асоцииране на хранителния пакет по полет.

### **-Flights**

Централната таблица за полети.

- flight\_id (PK)
- flight\_date, departure\_time, duration (в минути)
- FK(Външен ключ) към **Aircraft**, **Food\_Types**, **Airports** (departure/arrival)  
Позволява генериране на разписание, търсене по дата/час и изчисляване на натовареност.

### **-Crew**

Персонал – пилоти и стюардеси.

- crew\_id (PK)
- name, position, phone, email  
Използва се за планиране на екипажи и филтриране по роля (например само „Pilot“).

### **-Crew\_Assignments**

Планиране на кой екипаж (от **Crew**) с кой полет (**Flights**) работи и с каква роля.

- composite PK: (crew\_id, flight\_id, role)
- role: „Main Pilot“, „Cabin Crew“, и т.н.  
Позволява назначаване на неограничен брой екипажни позиции за всеки полет.

### **-Passengers**

Регистър на пътници.

- passenger\_id (PK)
- first\_name, last\_name, email, phone, passport\_number, nationality

### **-Bookings**

Резервации: кой пътник на кой полет е и на кое място.

- booking\_id (PK)
- FK към **Passengers** и **Flights**
- seat\_number: уникално място на полета

- booking\_date: времето на резервация  
Таблицата гарантира, че еднакво място не може да се резервира два пъти за един и същи полет.

**Заявките, с които създаваме базата данни и таблиците са:**

```
CREATE DATABASE airline_management;
USE airline_management;

CREATE TABLE Airports (
    airport_code CHAR(3) PRIMARY KEY,
    name VARCHAR(45),
    city VARCHAR(45),
    country VARCHAR(45)
);

CREATE TABLE Aircraft (
    aircraft_id INT AUTO_INCREMENT PRIMARY KEY,
    model VARCHAR(45),
    capacity INT
);

CREATE TABLE Food_Types (
    food_type_id INT AUTO_INCREMENT PRIMARY KEY,
    description VARCHAR(100)
);
```

```
CREATE TABLE Flights (  
    flight_id INT AUTO_INCREMENT PRIMARY KEY,  
    flight_date DATE,  
    departure_time TIME,  
    duration INT,  
    aircraft_id INT,  
    food_type_id INT,  
    departure_airport CHAR(3),  
    arrival_airport CHAR(3),  
    FOREIGN KEY (aircraft_id) REFERENCES Aircraft(aircraft_id),  
    FOREIGN KEY (food_type_id) REFERENCES Food_Types(food_type_id),  
    FOREIGN KEY (departure_airport) REFERENCES Airports(airport_code),  
    FOREIGN KEY (arrival_airport) REFERENCES Airports(airport_code)  
);
```

```
CREATE TABLE Crew (  
    crew_id INT AUTO_INCREMENT PRIMARY KEY,  
    name VARCHAR(100),  
    position VARCHAR(45),  
    phone VARCHAR(15),  
    email VARCHAR(100)  
);
```

```
CREATE TABLE Crew_Assignments (  
    crew_id INT,  
    flight_id INT,  
    role VARCHAR(45),  
    PRIMARY KEY (crew_id, flight_id),  
    FOREIGN KEY (crew_id) REFERENCES Crew(crew_id),  
    FOREIGN KEY (flight_id) REFERENCES Flights(flight_id)  
);
```

```

CREATE TABLE Passengers (
    passenger_id INT AUTO_INCREMENT PRIMARY KEY,
    first_name VARCHAR(45),
    last_name VARCHAR(45),
    email VARCHAR(100) UNIQUE,
    phone VARCHAR(15),
    passport_number VARCHAR(20) UNIQUE,
    nationality VARCHAR(45)
);

CREATE TABLE Bookings (
    booking_id INT AUTO_INCREMENT PRIMARY KEY,
    passenger_id INT,
    flight_id INT,
    seat_number VARCHAR(5),
    booking_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (passenger_id) REFERENCES Passengers(passenger_id),
    FOREIGN KEY (flight_id) REFERENCES Flights(flight_id),
    UNIQUE (flight_id, seat_number)
);

```

## Попълване на базата с тестови данни

```

INSERT INTO Airports VALUES
('SOF', 'Sofia Airport', 'Sofia', 'Bulgaria'),
('LHR', 'Heathrow', 'London', 'UK');

```

```

INSERT INTO Aircraft (model, capacity) VALUES
('Boeing 737', 180),
('Airbus A320', 150);

```

```

INSERT INTO Food_Types (description) VALUES
('Economy Class Meal'),
('Business Class Meal');

```

```

INSERT INTO Flights (flight_date, departure_time, duration, aircraft_id, food_type_id, departure_airport, arrival_airport) VALUES
('2024-06-01', '10:00', 120, 1, 1, 'SOF', 'LHR'),
('2024-06-02', '15:00', 180, 2, 2, 'LHR', 'SOF');

```

```

INSERT INTO Crew (name, position, phone, email) VALUES
('Ivan Petrov', 'Pilot', '0888123456', 'ivan.petrov@example.com'),
('Anna Dimitrova', 'Flight Attendant', '0888654321', 'anna.dimitrova@example.com');

INSERT INTO Crew (crew_id, name, position, phone, email) VALUES
(3, 'Asen Popov', 'Pilot', '0888123478', 'asen.popov@example.com');

INSERT INTO Crew_Assignments (crew_id, flight_id, role) VALUES
(1, 1, 'Main Pilot'),
(2, 1, 'Cabin Crew'),
(3, 1, 'Main Pilot');

INSERT INTO Passengers (first_name, last_name, email, phone, passport_number, nationality) VALUES
('Petar', 'Ivanov', 'petar.ivanov@example.com', '0888123000', 'BG12345', 'Bulgaria'),
('Elena', 'Stoyanova', 'elena.stoyanova@example.com', '0888999888', 'BG67890', 'Bulgaria');

INSERT INTO Bookings (passenger_id, flight_id, seat_number) VALUES
(1, 1, '15A'),
(2, 1, '15B');

```

**Задача 2.** Напишете заявка, в която демонстрирате **SELECT** с ограничаващо условие по избор:

```

-- Извеждане на нарядите на пилотите
SELECT c.name as pilot_name,
       f.flight_id,
       f.flight_date,
       f.departure_time,
       ca.role
FROM Crew c
JOIN Crew_Assignments ca ON ca.crew_id = c.crew_id
JOIN Flights f ON f.flight_id = ca.flight_id
WHERE c.position = 'Pilot'
ORDER BY f.flight_date, f.departure_time;

```



Това е заявка с ограничаващо условие **Where**, която връща списък с нарядите на всички пилоти в системата:

	pilot_name	flight_id	flight_date	departure_time	role
►	Ivan Petrov	1	2024-06-01	10:00:00	Main Pilot
	Asen Popov	1	2024-06-01	10:00:00	Main Pilot
	Panteley Popov	2	2024-06-02	15:00:00	Main Pilot

**Задача 3. Напишете заявка, в която използвате агрегатна функция и GROUP BY по ваш избор:**

```
-- Извеждане на всички полети с техните заети и свободни места
SELECT f.flight_id, f.flight_date, COUNT(b.booking_id) AS booked_seats, a.capacity,
(a.capacity - COUNT(b.booking_id)) AS available_seats
FROM Flights f
LEFT JOIN Bookings b ON f.flight_id = b.flight_id
JOIN Aircraft a ON f.aircraft_id = a.aircraft_id
GROUP BY f.flight_id;
```

Това е заявка с агрегатна функция **COUNT()** и **GROUP BY**, която за всеки полет смята колко места са заети и колко остават свободни:

	flight_id	flight_date	booked_seats	capacity	available_seats
►	1	2024-06-01	3	180	177
	3	2025-05-20	2	150	148
	2	2024-06-02	2	150	148

**Задача 4. Напишете заявка, в която демонстрирате INNER JOIN по ваш избор:**

```
-- Извеждане на разписанията на полетите
SELECT f.flight_id, f.flight_date, f.departure_time, f.duration,
       da.name as departure_airport,
       aa.name as arrival_airport,
       a.model as aircraft_model,
       ft.description as food_type
FROM Flights f
JOIN Airports da ON f.departure_airport = da.airport_code
JOIN Airports aa on f.arrival_airport = aa.airport_code
JOIN Aircraft a on f.aircraft_id = a.aircraft_id
JOIN Food_Types ft ON f.food_type_id = ft.food_type_id
ORDER BY f.flight_date, f.departure_time;
```

Това е заявка с няколко **INNER JOIN**, при която данните за полети, начални и крайни летища, самолет и тип храна се събират в един резултат и така се извеждат разписанията на полетите:

	flight_id	flight_date	departure_time	duration	departure_airport	arrival_airport	aircraft_model	food_type
	1	2024-06-01	10:00:00	120	Sofia Airport	Heathrow	Boeing 737	Economy Class Meal
	2	2024-06-02	15:00:00	180	Heathrow	Sofia Airport	Airbus A320	Business Class Meal
	3	2025-05-20	12:00:00	120	Sofia Airport	Paris Airport	Airbus A320	Economy Class Meal

**Задача 5. Напишете заявка, в която демонстрирате OUTER JOIN по ваш избор:**

```
-- Извеждане на всички полети с техните заети и свободни места
SELECT f.flight_id, f.flight_date, COUNT(b.booking_id) AS booked_seats, a.capacity,
(a.capacity - COUNT(b.booking_id)) AS available_seats
FROM Flights f
LEFT JOIN Bookings b ON f.flight_id = b.flight_id
JOIN Aircraft a ON f.aircraft_id = a.aircraft_id
GROUP BY f.flight_id;
```

За тази задача може да използваме същата заявка като от задача 3, защото тя съдържа **Outer Join** (конкретно **LEFT JOIN**), при който за всеки полет се извеждат общият брой заети места (от Bookings) и капацитетът на самолета. Дори полети без резервации ще присъстват в резултата, тъй като използваме **LEFT JOIN**:

	flight_id	flight_date	booked_seats	capacity	available_seats
▶	1	2024-06-01	3	180	177
	3	2025-05-20	2	150	148
	2	2024-06-02	2	150	148

**Задача 6. Напишете заявка, в която демонстрирате вложен SELECT по ваш избор:**

```
-- Намиране на свободни членове на екипажа за даден полет и роля
SELECT crew_id, name, position FROM CREW
WHERE position = 'Pilot' -- позицията, за която търсим заместник
AND crew_id NOT IN(
    SELECT ca.crew_id FROM Crew_Assignments ca
    JOIN Flights f ON f.flight_id = ca.flight_id
    WHERE f.flight_date = '2024-06-02' -- дата на полета, за който търсим заместник
);
```

Това е заявка с вложен **SELECT**, при която първо взимаме всички **crew\_id**, вече заети на полет на определена дата, и чрез **NOT IN** изключваме тези записи от основния резултат. Така получаваме свободните (неназначени) пилоти за конкретния полет:

	crew_id	name	position
▶	1	Ivan Petrov	Pilot
	3	Asen Popov	Pilot
●	NULL	NULL	NULL

**Задача 7. Напишете заявка, в която демонстрирате едновременно JOIN и агрегатна функция:**

```
-- Извеждане на всички полети с техните заети и свободни места
SELECT f.flight_id, f.flight_date, COUNT(b.booking_id) AS booked_seats, a.capacity,
(a.capacity - COUNT(b.booking_id)) AS available_seats
FROM Flights f
LEFT JOIN Bookings b ON f.flight_id = b.flight_id
JOIN Aircraft a ON f.aircraft_id = a.aircraft_id
GROUP BY f.flight_id;
```

За тази задача може да използваме същата заявка като от задача 3, защото тя съдържа едновременно **Join** и агрегатната функция **Count()**:

	flight_id	flight_date	booked_seats	capacity	available_seats
▶	1	2024-06-01	3	180	177
	3	2025-05-20	2	150	148
	2	2024-06-02	2	150	148

**Задача 8. Създайте тригер по ваш избор:**

```
-- Тригер върху таблицата Bookings, който автоматично превръща
-- seat_number в главни букви и не използва допълнителна таблица.
DELIMITER $$

CREATE TRIGGER trg_before_booking_insert
BEFORE INSERT ON Bookings
FOR EACH ROW
BEGIN
    -- Превръщаме seat_number в главни букви
    SET NEW.seat_number = UPPER(NEW.seat_number);
END$$

DELIMITER ;
```

В този проект е реализиран **BEFORE INSERT** тригер върху таблицата **Bookings**, който гарантира, че въведеното място (**seat\_number**) винаги ще бъде записано с главни букви. Това опростява търсене и осигурява консистентност на данните.

### Тестване на тригера:

```
-- Тестова INSERT заявка
INSERT INTO Bookings (passenger_id, flight_id, seat_number)
VALUES (1, 1, 'b3c');
-- Резултат
SELECT booking_id, seat_number
FROM Bookings
WHERE booking_id = LAST_INSERT_ID();
```

Резултат:

	booking_id	seat_number
▶	9	B3C
*	NULL	NULL

**Задача 9.** Създайте процедура, в която демонстрирате използване на курсор:

```
DELIMITER $$

CREATE PROCEDURE sp_monthly_meal_report(
    IN p_month INT,
    IN p_year INT
)
BEGIN
    DECLARE done INT DEFAULT 0;
    DECLARE v_ft_id INT;
    DECLARE v_count INT;
    DECLARE v_desc VARCHAR(100);

    -- 1) курсор за всички различни food_type_id в зададения период
    DECLARE cur_ft CURSOR FOR
        SELECT DISTINCT food_type_id
        FROM Flights
        WHERE MONTH(flight_date) = p_month
        AND YEAR(flight_date) = p_year;

    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;
```

```
-- 2) временна таблица за съхранение на резултатите
CREATE TEMPORARY TABLE IF NOT EXISTS temp_meal_stats (
    food_type_id INT PRIMARY KEY,
    description VARCHAR(100),
    meal_count INT
) ENGINE=MEMORY;
```

```
-- 3) обхождаме типовете храна и смятаме броя поръчки
OPEN cur_ft;
read_loop: LOOP
    FETCH cur_ft INTO v_ft_id;
    IF done THEN
        LEAVE read_loop;
    END IF;

    SELECT description INTO v_desc
        FROM Food_Types
        WHERE food_type_id = v_ft_id;

    SELECT COUNT(b.booking_id) INTO v_count
        FROM Flights f
        LEFT JOIN Bookings b ON f.flight_id = b.flight_id
        WHERE f.food_type_id = v_ft_id
            AND MONTH(f.flight_date) = p_month
            AND YEAR(f.flight_date) = p_year;

    INSERT INTO temp_meal_stats
        VALUES (v_ft_id, v_desc, v_count);
END LOOP;
CLOSE cur_ft;
```

```
-- 4) връщаме резултатите
SELECT * FROM temp_meal_stats;
```

```
-- 5) изчистваме временната таблица
DROP TEMPORARY TABLE temp_meal_stats;
END$$
```

```
DELIMITER ;
```

Създадена е процедура **sp\_monthly\_meal\_report**, която за даден месец и година връща броя на поръчаните ястия по типове (на база направените резервации). Използва курсор, временна таблица.

Тестване на процедурата: `CALL sp_monthly_meal_report(6, 2024);`

Резултати:

food_type_id	description	meal_count
1	Economy Class Meal	3
2	Business Class Meal	2

## Django приложение за уеб интерфейс към базата

След изграждането на **MySQL** схемата, нейният уеб интерфейс е реализиран като Django-приложение **Airline Management**. По-долу са описани основните функционалности на платформата. Основната цел на приложението е да се осигури интуитивен уеб интерфейс, който опростява администрирането на базата и демонстрира работещ прототип на система за управление на полети и екипажи.

### 1. Логически поток

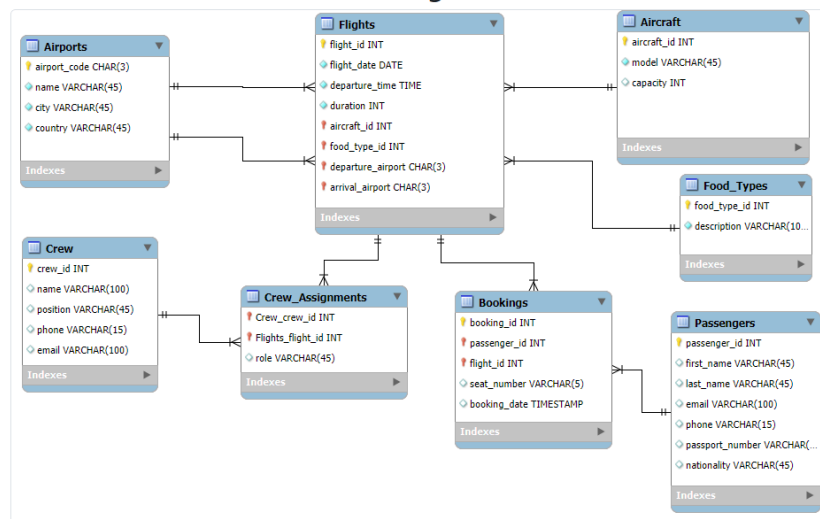
- Начална страница - показва сурови таблици, които се обновяват динамично + ER-диаграмата.



Welcome to Airline Management 

Тук можеш да разглеждаш всички данни и ER-диаграмата.

### ER Diagram



Изглед на таблиците в системата:

Airports

airport_code	name	city	country
CDG	Paris Airport	Paris	France
LHR	Heathrow	London	UK
SOF	Sofia Airport	Sofia	Bulgaria

Aircraft

aircraft_id	model	capacity
1	Boeing 737	180
2	Airbus A320	150

Food Types

food_type_id	description
1	Economy Class Meal
2	Business Class Meal

Flights

flight_id	flight_date	departure_time	duration	aircraft_id	food_type_id	departure_airport_id	arrival_airport_id
1	June 1, 2024	10 a.m.	120	1	1	SOF	LHR
2	June 2, 2024	3 p.m.	180	2	2	LHR	SOF
3	May 20, 2025	noon	120	2	1	SOF	CDG
4	June 15, 2024	6 p.m.	90	1	1	SOF	LHR

- Разписания на полетите (Flight Schedule) - извличане чрез ORM със `select_related` за летища, самолет и храна:

Flight Schedule

Flight ID	Date	Departure Time	Departure Airport	Arrival Airport	Aircraft	Food Type	Passengers	Book
1	June 1, 2024	10 a.m.	Sofia Airport	Heathrow	Boeing 737	Economy Class Meal	<div>View</div>	<div>Book</div>
2	June 2, 2024	3 p.m.	Heathrow	Sofia Airport	Airbus A320	Business Class Meal	<div>View</div>	<div>Book</div>
4	June 15, 2024	6 p.m.	Sofia Airport	Heathrow	Boeing 737	Economy Class Meal	<div>View</div>	<div>Book</div>
3	May 20, 2025	noon	Sofia Airport	Paris Airport	Airbus A320	Economy Class Meal	<div>View</div>	<div>Book</div>

- Резервации с избран полет (**Book**) - **BookFlightView** показва форма с вече избран полет; след успешен запис потребителят се връща към списъка с пътници на този полет:

Book Flight #1

Passenger

-----

▼

Flight

#1 on 2024-06-01 at 10:00:00

▼

Seat number

e.g. 12A

Запази

- Списък с пасажерите(**View**) - **FlightBookingsView** извежда списък с всички пътници на избрания полет:

Bookings for Flight 1 on June 1, 2024

Seat	Passenger	Email	Phone	Booking Date
B3C	Petar Ivanov	petar.ivanov@example.com	0888123000	April 20, 2025, 3:37 p.m.
15A	Petar Ivanov	petar.ivanov@example.com	0888123000	April 18, 2025, 12:22 p.m.
15B	Elena Stoyanova	elena.stoyanova@example.com	0888999888	April 18, 2025, 12:22 p.m.

- Назначения на екипажа(**Crew Assignments**) - комбиниран изглед (**List + Create**), позволяващ едновременно преглед и добавяне на назначения:



Crew Assignments

Crew:  
-----

Flight:  
-----

Role:  
e.g. Main Pilot

Назначи

Pilot Name	Flight ID	Date	Departure Time	Role
Ivan Petrov	1	June 1, 2024	10 a.m.	Main Pilot
Anna Dimitrova	1	June 1, 2024	10 a.m.	Cabin Crew
Asen Popov	1	June 1, 2024	10 a.m.	Main Pilot
Panteley Popov	2	June 2, 2024	3 p.m.	Main Pilot
Petya Georgieva	3	May 20, 2025	noon	Cabin Crew
Raya Georgieva	4	June 15, 2024	6 p.m.	Assistant Pilot

- Добави Пътник(**Add Passenger**) - Интерфейс за **INSERT** заявка директно към базата, която, при нов запис, се обновява динамично в приложението:

Добави Пътник

First name  
-----

Last name  
-----

Email  
-----

Phone  
-----

Passport number  
-----

Nationality  
-----

Запази



- Додати Самолет(Add Aircraft):

## Додати Самолет

Model

Capacity

Зберегти

- Додати Полет(Add Flight):

## Додати Полет

Flight date

mm/dd/yyyy



Departure time

--:-- --



Duration (minutes)

Aircraft

-----



Food type

-----



Departure airport

-----



Arrival airport

-----



Зберегти

- **Добави Член на Екипажа(Add Crew):**

## Добави Член на Екипажа

Name

Position

Phone

Email



Запази

- **Добави Летище(Add Airport):**

## Добави Летище

Airport code

Name

City

Country

Запази

- Резервирай Полет(Book a Flight) - Форма за резервация на полет по избор. Позволява избор на пътник, полет и място; валидността на мястото се гарантира от unique\_together:

## Book a Flight

Passenger

Flight

Seat number

Запази

## 2. Шаблони (templates) или Основните HTML страници





- **base.html** – Bootstrap layout, единна навигация.
- **home.html** – показва кратък dashboard с всички таблици (+ изображението на ER-диаграмата).
- **schedule.html** – таблица с полети.
- **crew\_assignments.html** – страница, която комбинира списък и форма за назначения.
- **form.html** – универсален шаблон за всички CRUD форми.

## 3. GitHub репозитория

- Проектът е публикуван в публичен **GitHub** репозиториум - линк

## 4. Идеи за надграждане(roadmap)

Фаза	Описание	Статус
v0.1	Базова функционалност: базата, CRUD през Django, ER диаграма	✓ Завършена

v0.2	Добавяне на аутентикация (Django auth, потребителски роли)	 Планирано
v0.3	Генериране на PDF билети за пътниците	 Планирано
v0.4	Роля "Dispatcher" – екран за редакция на разписанията на полетите	 Планирано
v0.5	REST API (Django REST Framework) за мобилно приложение	 Планирано

