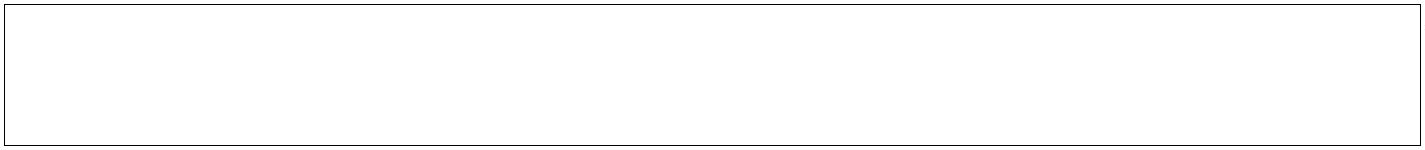
**ПРОФИЛИРАНА ПРИРОДОМАТЕМАТИЧЕСКА ГИМНАЗИЯ**

****



**“АКАДЕМИК НИКОЛА ОБРЕШКОВ” – ГРАД РАЗГРАД**

**ул.Дъбрава 2, тел:0878684362, 0876207320, e-mail:pmgrz@abv.bg, http://www.pmgrz.net**

**ПРОФЕСИЯ „ПРИЛОЖЕН ПРОГРАМИСТ“**

**СПЕЦИАЛНОСТ „ПРИЛОЖНО ПРОГРАМИРАНЕ“**

**ДИПЛОМЕН ПРОЕКТ**

**за придобиване трета степен професионална квалификация**

**ТЕМА: Система за управление на въздушен транспорт**

**Ученик: Виктор Ростиславов Григоров**

**/име, презиме, фамилия/**

**Клас: 12г**

**GitHub-хранилище: https://github.com/Viktor1806/Air\_Transport\_System\_Diploma\_project.git**

**Ръководител-консултант: Венцеслав Кочанов**

**/име и фамилия/**

**Разград**

**2025 година**

**СЪДЪРЖАНИЕ**

Увод………………………………………………………….5

Основна част/Глава първа: Теоретична основа………………………………………………….6

* 1. Уеб базирани информационни системи…………6
  2. Клиент-сървър архитектура………………………6
  3. Flask – микроуеб фреймуърк за Python…………..7
  4. Бази от данни и релационен модел……………….7
  5. Свързване с база данни чрез PyMySQL…………..8
  6. Интерфейс и потребителско изживяване…………8

Основна част/Глава втора: Аналитична част…………9

2.1 Анализ на текущото състояние на процесите във въздушния транспорт…….9

2.2 Модели и подходи за анализ……………………….10

2.3 Интегрирана архитектура…………………………..12

2.4 Заключения от анализа………………………………12

Основна част/Глава трета: Проектна част………………13

3.1 Обща архитектура на системата……………………..13

3.2 Структура на проекта…………………………………13

3.3 Модул 1: Управление на самолети…………………..15

3.4 Модул 2: Управление на полети:…………………….17

3.5 Модул 3: Регистрация и управление на пътници……19

3.6 Модул 4: Управление на резервации…………………23

3.7 Модул 5: Управление на товарен транспорт...25

3.8 Администраторски панел…………27

3.9 Сигурност и сесии…………………28

3.10 Интерфейс и стилове………………29

3.11 Тестване и валидация…………….29

Заключение……………………………..34

Списък на използваната литература…..35

Приложения…………………………….36

**УВОД**

С нарастващата роля на технологиите във всяка сфера на съвременното общество, автоматизацията на процеси и управлението на данни чрез уеб-базирани системи се превръщат в неразделна част от развитието на различни индустрии. Един от най-динамичните и критични сектори, където ефективното управление е от съществено значение, е въздушният транспорт. Този сектор изисква високо ниво на координация, точност и сигурност при изпълнението на дейности, свързани с полети, резервации, превоз на пътници и товари.

Целта на настоящия дипломен проект е разработване на уеб-базирана система зауправление на въздушен транспорт, която да подпомага административните процеси в една авиационна компания. Системата обединява няколко основни функционални модула, които позволяват администриране на самолети, полети, пътници, резервации и товари. Проектът е реализиран с помощта на езика за програмиране Python, уеб фреймуърка Flask, базата данни MySQL и библиотеката PyMySQL за свързване с нея. Осигурен е интуитивен уеб интерфейс, изграден чрез HTML, CSS и шаблони на Jinja2, който позволява лесна навигация и работа с наличните функционалности.

Разработената система се стреми да отговори на реални нужди на администраторите, като същевременно предлага възможност и за пътници да създават резервации самостоятелно. Модулният подход на изграждане позволява както лесно надграждане, така и внедряване в реална среда, като се осигурява гъвкавост при работа с различни типове самолети и полети – пътнически и товарни.

В рамките на дипломната работа ще бъдат представени теоретични основи, анализ на проблема, архитектурата на разработената система, описание на всеки отделен модул, както и екранни снимки, показващи реализацията на функционалностите. Също така ще бъде включен пълният сорс код и структурата на използваната база данни, с цел по-лесна проверка, доразвитие и внедряване в практическа среда.

**ОСНОВНА ЧАСТ/ГЛАВА ПЪРВА:**

**ТЕОРЕТИЧНА ОСНОВА**

**1.1 Уеб базирани информационни системи**

През последните десетилетия уеб технологиите значително промениха начина, по който се създават, използват и управляват информационни системи. Уеб базираните приложения представляват системи, достъпни чрез уеб браузър, като обработват заявки от потребителя към сървър и връщат обратно динамично генерирано съдържание. Според Г. Кулкарни (Kulkarni, 2012), този модел позволява ефективно разпределение на ресурси и осигурява мащабируемост на решенията.

В контекста на транспортния сектор, уеб системите намират широко приложение при управление на логистични процеси, координиране на ресурси и оптимизиране на маршрути. Системите за въздушен транспорт, които събират и обработват информация за самолети, полети, товари и резервации, представляват критична инфраструктура и изискват надеждни софтуерни решения.

**1.2 Клиент–сървър архитектура**

Според моделите, разглеждани от Таненбаум (Tanenbaum, 2003), основна архитектура при уеб приложенията е клиент–сървър. При този модел клиентът (уеб браузър) изпраща заявка, която се обработва от сървър, управляващ логиката и достъпа до базата данни. Това разделение осигурява ясна логика на комуникация, лесна поддръжка и централизирано управление на сигурността.

В настоящата разработка клиентската част е реализирана чрез шаблони в HTML и Jinja2, а сървърната – чрез Python и Flask.

**1.3 Flask – микроуеб фреймуърк за Python**

Flask, създаден от Армин Ронахер (Ronacher, 2010), е лек, модулен и разширяем фреймуърк за разработка на уеб приложения с езика Python. Неговата архитектура следва принципите на минимализма и свобода на избор, като предоставя само основните функции, а допълнителните възможности се добавят чрез разширения.

Според Б. Гронбаум (Grönbaum, 2017), Flask е особено подходящ за образователни и реални проекти, защото позволява изграждане на структуриран бекенд, който може да се мащабира.

**1.4 Бази от данни и релационен модел**

Една от най-важните концепции в разработката на информационни системи е релационният модел, въведен от Е.Ф. Код (Codd, 1970). В основата на този модел стои представянето на информацията чрез свързани таблици, всяка от които съдържа редове (записи) и колони (атрибути). Този подход осигурява:

* Ясна структура на данните
* Поддръжка на връзки между таблици (чрез първични и външни ключове)
* Поддръжка на транзакции и целостта на данните

В настоящия проект е използвана MySQL, една от най-широко използваните СУБД с отворен код, съвместима с езика SQL и подходяща за висока производителност.

**1.5 Свързване с база данни чрез PyMySQL**

За комуникация между уеб приложението и базата от данни е използвана библиотеката PyMySQL. Тя е съвместима с Python и осигурява пълна поддръжка на SQL заявки. Предимство на PyMySQL е възможността за работа с речникови обекти(DictCursor), което улеснява извеждането на данни към шаблони в Jinja2.

Всяка операция – създаване, четене, актуализация и изтриване (CRUD) – се реализира чрез SQL команди, изпращани към MySQL чрез PyMySQL.

**1.6 Интерфейс и потребителско изживяване**

За реализиране на потребителския интерфейс се използват HTML, CSS и Bootstrap. Bootstrap, създаден от Twitter, е CSS фреймуърк за бързо изграждане на респонсив интерфейс. Комбинацията от шаблони в Jinja2 и Bootstrap осигурява динамично и адаптивно съдържание, което улеснява потребителя в навигацията и взаимодействието със системата.

**ОСНОВНА ЧАСТ/ГЛАВА ВТОРА:**

**АНАЛИТИЧНА ЧАСТ**

**2.1 Анализ на текущото състояние на процесите във въздушния транспорт**

Управлението на въздушен транспорт изисква съгласувани действия между множество звена – авиокомпании, летища, наземни служби, логистични оператори и пътници. Въпреки съществуването на глобални системи, като Amadeus, Sabre и Galileo, тяхната сложност, висока цена и затворен код ги правят неподходящи за малки оператори и учебни цели.

При липса на автоматизирана система се наблюдават следните **основни проблеми**:

* Разпокъсаност на данните и процесите (няколко таблици и документи, поддържани на ръка).
* Трудности при управление на резервации и товари.
* Липса на механизъм за проверка на капацитет и съвместимост на товари със самолети.
* Невъзможност за проследяване на местоположението и състоянието на товарите в реално време.
* Човешки грешки при обработка на информация и резервации.

Това налага нуждата от единна система, която да обединява и автоматизира основните функции, свързани с управлението на въздушния транспорт.

**2.2 Модели и подходи за анализ**

В рамките на този проект бяха приложени следните подходи и модели за анализ и проектиране на системата:

**-Модел на функционално разделение по модули:**

Всеки модул (Самолети, Полети, Пътници, Резервации, Товари) е самостоятелен, но интегриран в общата логика. Това позволява ясно разграничение на отговорности и улеснява поддръжката.

**-Релационен модел на базата данни:**

Използвана е релационна структура с външни ключове, която гарантира консистентност на данните. Например:

* В таблицата reservations има външни ключове към passengers и flights (виж фиг. 18 в ПРИЛОЖЕНИЕ 2).
* Таблицата cargo е обвързана с flights, а чрез тях — и с aircraft (виж фиг. 18 в ПРИЛОЖЕНИЕ 2).

**-Ролеви модел за достъп:**

Системата поддържа два основни типа потребители:

* Администратор – има пълен контрол и достъп до всички данни.
* Пътник – може да създава собствени резервации, които съответно се одобряват от администратора.

**-Модел за валидиране на входните данни:**

Формулярите включват проверка за:

* празни стойности,
* допустими стойности на статуси чрез падащи менюта и ENUM проверки в базата.

**-Модел за автоматизация и логическо проследяване:**

В модула "Товарен транспорт" е реализиран бутон за смяна на статус, който:

* При първо натискане променя статуса на „В движение“
* При второ – на „Доставен“
* При трето – автоматично изтрива товара

**2.3 Интегрирана архитектура**

Цялата система работи на базата на модел на три слоя:

* Представяне (Presentation Layer): HTML шаблони с Bootstrap и Jinja2
* Логика на приложението (Application Layer): Python (Flask)
* База от данни (Data Layer): MySQL с PyMySQL

Това осигурява ясно разделение между интерфейса и логиката, улеснявайки бъдещо разширяване, прехвърляне към друг тип бази или интерфейси (например мобилни приложения).

**2.4 Заключения от анализа**

Чрез приложените подходи, проектът демонстрира възможността да се създаде пълноценна, мащабируема и функционално гъвкава система, отговаряща на реалните нужди на въздушния транспорт. Постигнати са следните ключови резултати:

* Централизиран достъп до всички данни и справки
* Интуитивен потребителски интерфейс
* Автоматизирани проверки за капацитет и статуси
* Реалистично управление на товари с проследимост

**ОСНОВНА ЧАСТ/ГЛАВА ТРЕТА:**

**ПРОЕКТНА ЧАСТ**

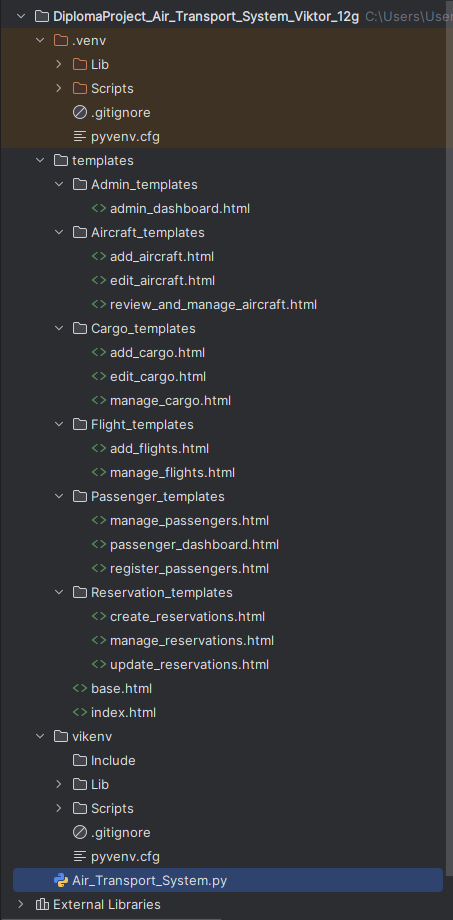
**3.1 Обща архитектура на системата**

Системата е реализирана като уеб приложение с три основни слоя:

* Презентационен слой: HTML + Bootstrap 5 + Jinja2
* Приложен слой: Python (Flask framework)
* Слой за съхранение на данни**:** MySQL, използван чрез PyMySQL

Сърцевината на приложението е файлът Air\_Transport\_System.py, който съдържа всички маршрути и логика. Всеки модул има собствена папка с HTML шаблони, а връзката с базата се осъществява чрез функцията get\_db\_connection().

**3.2 Структура на проекта (виж фиг. 1)**

****

**Фигура 1 – Структурата на проекта**

**3.3 Модул 1: Управление на самолети (фиг. 2)**

Този модул позволява на администратора да:

* Добавя, редактира и изтрива самолети:

@app.route('/add\_aircraft', methods=['GET', 'POST'])  
@admin\_required  
def add\_aircraft():  
cursor.execute("INSERT INTO aircraft (model, capacity, registration\_number, status) VALUES (%s, %s, %s, %s)",  
 (model, capacity, registration\_number, status))

@app.route("/aircraft/edit/<int:id>", methods=["GET", "POST"])  
@admin\_required  
def edit\_aircraft(id):

cursor.execute("""  
 UPDATE aircraft   
 SET model = %s, capacity = %s, registration\_number = %s, status = %s   
 WHERE id = %s  
""", (model, capacity, registration\_number, status, id))

@app.route("/aircraft/<int:id>", methods=["POST"])  
@admin\_required  
def delete\_aircraft(id):

cursor.execute("DELETE FROM aircraft WHERE id = %s", (id,))

* Преглежда списък с всички самолети:

cursor.execute(“SELECT\* FROM aircraft”)

* Филтрира самолетите по статус (изправен, в ремонт и неизправен.)

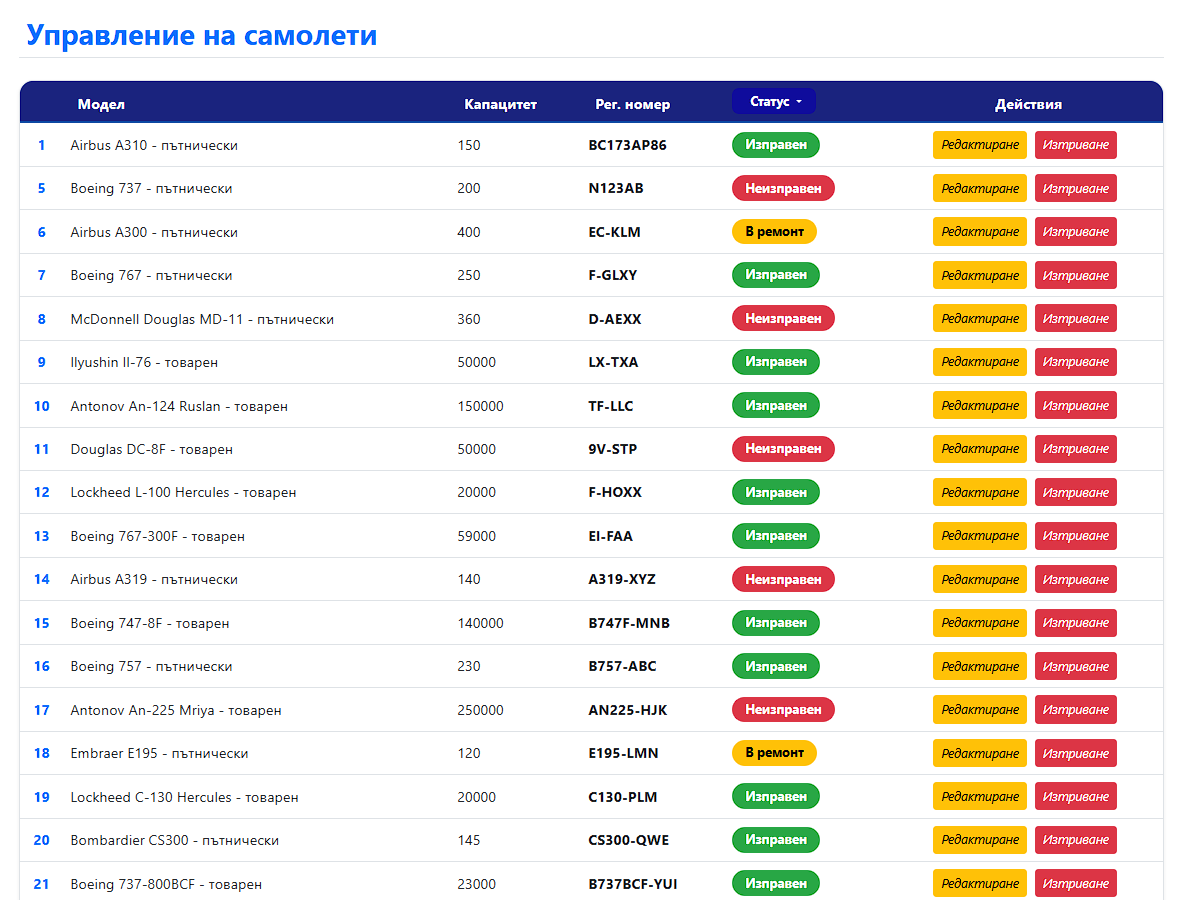
status\_filter = request.args.get("status")

if status\_filter:

cursor.execute("SELECT \* FROM aircraft WHERE status = %s", (status\_filter,))

Всичко се записва в таблицата aircraft (виж фиг. 22 в ПРИЛОЖЕНИЕ 3)

HTML формата включва три шаблона – add\_aircraft.html ,edit\_aircraft.html(на фиг. 10 и фиг. 11 в ПРИЛОЖЕНИЕ 1) и review\_and\_manage\_aircraft.html(виж фиг.2 отново)



**Фигура 2 – Управление на самолети. Администраторът може да редактира и изтрива в зависимост от статуса.**

**3.4 Модул 2: Управление на полети (фиг. 3)**

Модулът дава възможност за:

* Създаване на полети с посочване на самолет, екипаж и маршрут:

@app.route("/flights/add\_flights", methods=["GET", "POST"])

@admin\_required

def add\_flights():

cursor.execute("SELECT id, model, status FROM aircraft")

aircrafts = cursor.fetchall()

cursor.execute(  
 "INSERT INTO flights (flight\_number, departure, arrival, aircraft, crew, status ,departure\_city, arrival\_city) VALUES (%s, %s, %s, %s, %s, %s, %s, %s)",  
 (flight\_number, departure, arrival, aircraft, crew, status,departure\_city,arrival\_city)  
)

* Преглед на всички полети:

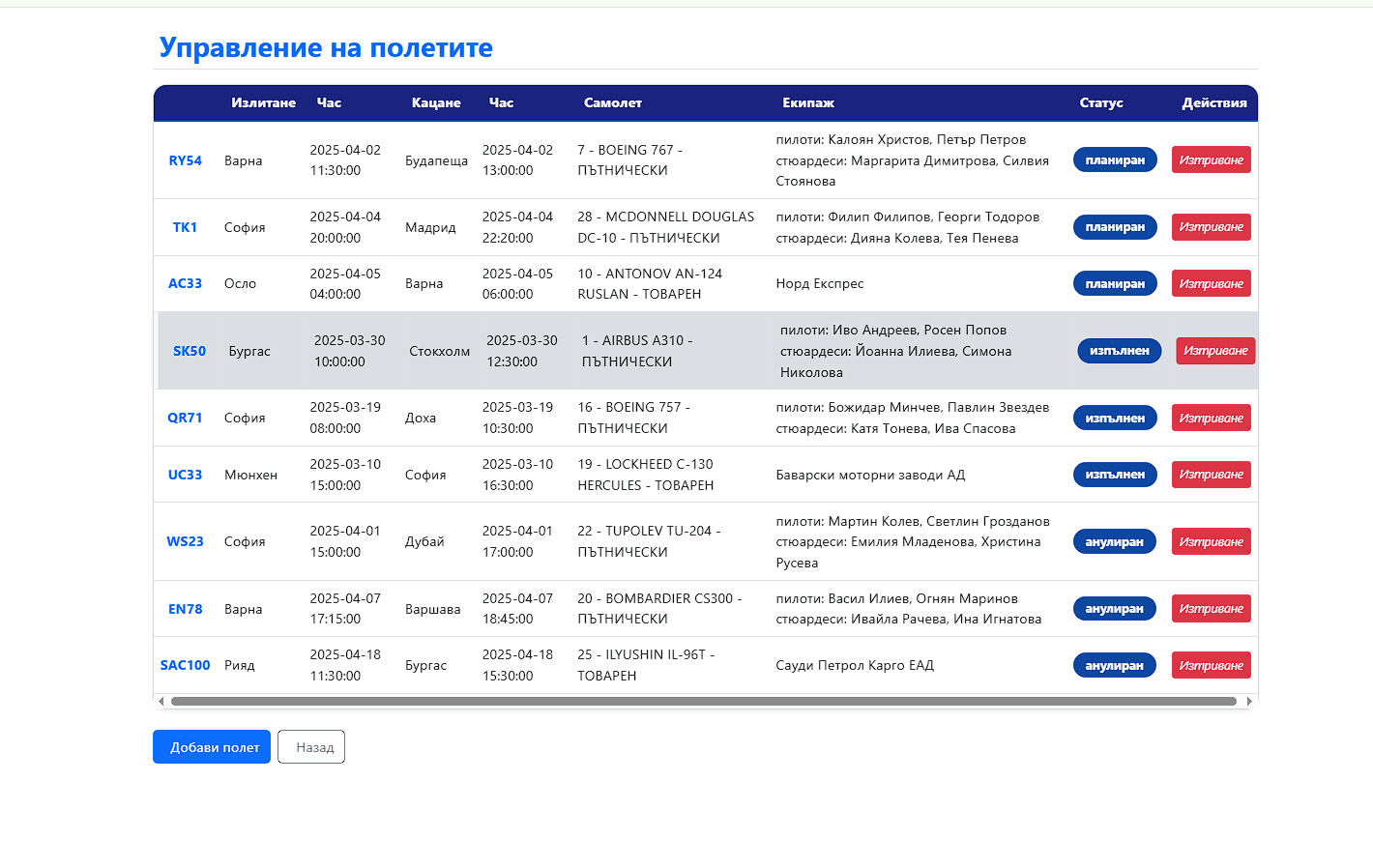
cursor.execute(“SELECT\* FROM flights”)

* Изтриване на анулирани или изпълнени полети

cursor.execute("DELETE FROM flights WHERE id = %s", (id,))

Свързване с таблица aircraft чрез външен ключ.

* Полето aircraft сочи към избран самолет, като се визуализира модела и ID(виж фиг. 18 в ПРИЛОЖЕНИЕ 2).



**Фигура 3 – Управление на полети. Админът има функциите да изтрива отново в зависимост от статуса.**

**3.5 Модул 3: Регистрация и управление на пътници**

Пътниците могат да:

* Създават акаунти чрез регистрационна форма (фиг. 4)

cursor.execute("""

INSERT INTO passengers (name,last\_name, email,password,phone, document\_number)

VALUES (%s, %s,%s, %s,%s,%s)

""", (name,last\_name, email,password,phone, document\_number))

* Влизат със своите данни (фиг.5):

if passenger:

session["passenger\_id"] = passenger['id']

session["passenger\_name"] = passenger['name']

return redirect(url\_for("passenger\_dashboard"))

else:

return redirect(url\_for("index", error="invalid\_passenger\_credentials"))

* Виждат персонализирано табло с опция за създаване на резервации (фиг. 6):

@app.route("/passengers/dashboard")

def passenger\_dashboard():

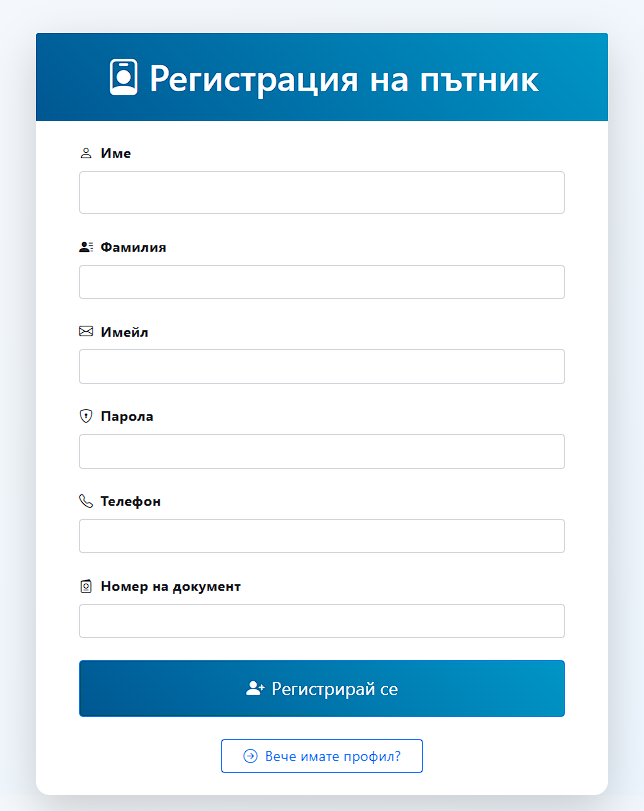
if "passenger\_id" not in session:

return redirect(url\_for("passenger\_login"))

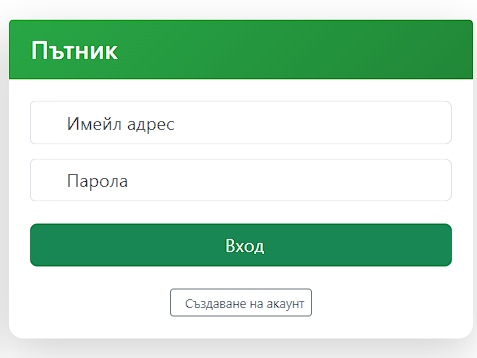
return render\_template("Passenger\_templates/passenger\_dashboard.html", passenger\_name=session["passenger\_name"])

Администраторът има достъп до списък с пътници и възможност за изтриване на техните данни. (фиг. 7):

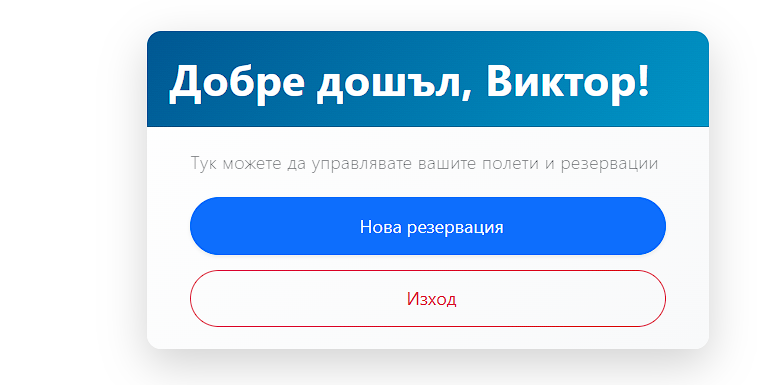
cursor.execute("SELECT \* FROM passengers")



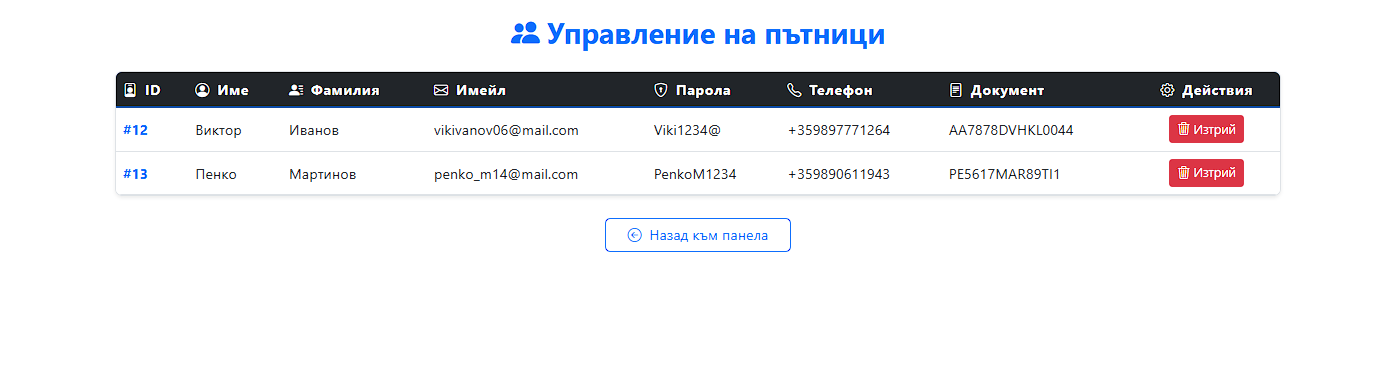
**Фигура 4 – Форма за регистрация на пътник, в която пътникът въвежда своите данни.**



**Фигура 5 – Форма за вход на вече регистрирал се пътник.**



**Фигура 6 – Табло с опция за създаване на резервация от пътник.**



**Фигура 7 – Управление на пътници. Тук администраторът има правото да изтрие цялата информация за пътника.**

**3.6 Модул 4: Управление на резервации (фиг. 8)**

Модулът поддържа:

* Създаване на резервация от логнат пътник:

cursor.execute("""

INSERT INTO reservations (passenger\_id, flight\_id, seat\_number, status)

VALUES (%s, %s, %s, %s)

""", (passenger\_id, flight\_id, seat\_number, status))

* Избор само на пътнически полети с капацитет под 1000:

cursor.execute("""

SELECT flights.id, flights.flight\_number, flights.status, flights.departure\_city, aircraft.capacity

FROM flights

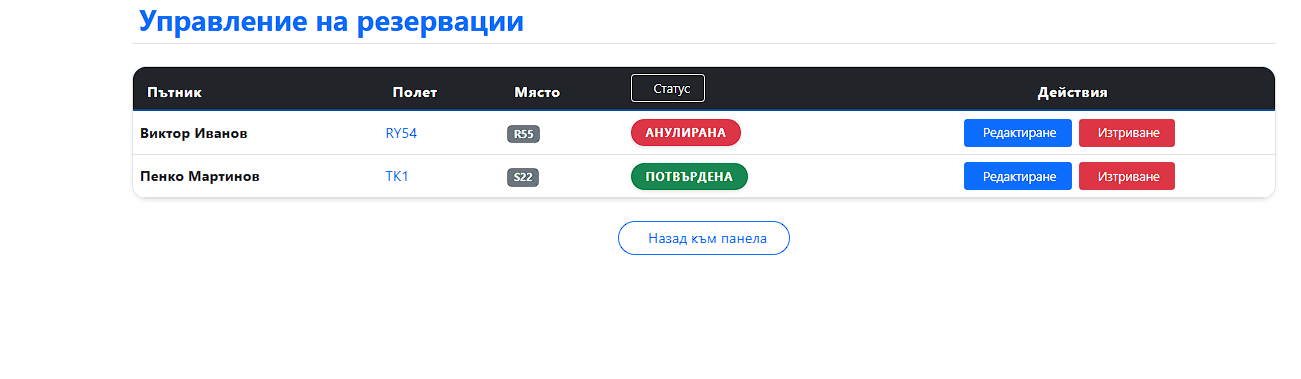
JOIN aircraft ON flights.aircraft = aircraft.id

WHERE aircraft.capacity < 1000 AND flights.status = 'Планиран'

""")

Таблица, с която администраторът може да прави следните неща:

* + редакция на място и статус
  + изтриване на резервации
  + генериране на случаен статус чрез бутон (потвърдена или анулирана)

HTML формата включва три шаблона – create\_reservation.html(за пътник), update\_reservations.html(редакция) и manage\_reservations.html(за админ) (всичките шаблони виж в ПРИЛОЖЕНИЕ 1)

**Фигура 8 – Управление на резервации. Чрез натискане на бутона „Статус“, Админът има правото на случаен принцип да определи дали резервацията е анулирана, или потвърдена.**

**3.7 Модул 5: Управление на товарен транспорт (фиг. 9)**

Най-комплексният модул. Позволява:

* Добавяне на товар, обвързан с товарен полет (самолети с капацитет над 1000):

cursor.execute("""

INSERT INTO cargo (description, weight, volume, flight\_id, status, location)

VALUES (%s, %s, %s, %s, %s, %s)

""", (description, weight, volume, flight\_id, status, location))

* Проследяване на статуса чрез бутон:
  1. Очаква товарене
  2. В движение
  3. Доставено (при което автоматично се изтрива):

track = cursor.fetchone()

if track:

current\_status = track['status']

arrival\_city = track['arrival\_city']

if current\_status == "Очаква товарене":

new\_status = "В движение"

new\_location = None

cursor.execute("""

UPDATE cargo SET status=%s, location=%s WHERE id=%s

""", (new\_status, new\_location, id))

elif current\_status == "В движение":

new\_status = "Доставено"

new\_location = arrival\_city

cursor.execute("""

UPDATE cargo SET status=%s, location=%s WHERE id=%s

""", (new\_status, new\_location, id))

elif current\_status == "Доставено":

cursor.execute("DELETE FROM cargo WHERE id=%s", (id,))

* Редакция на детайлите по товарите:

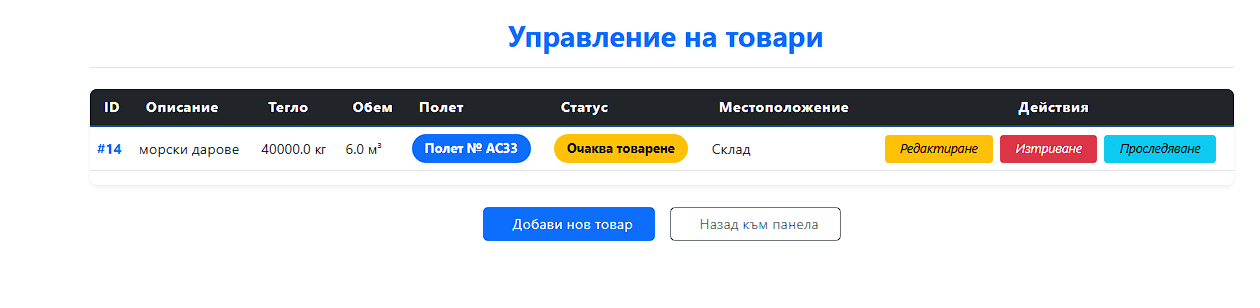
cursor.execute("""

UPDATE cargo

SET description = %s, weight = %s, volume = %s, flight\_id = %s, status = %s, location = %s

WHERE id = %s

""", (description, weight, volume, flight\_id, status, location, id))



.

**Фигура 9 – Управление на товари. Тук чрез няколко натискания на бутона „Проследяване“ се извършват автоматизирани процеси**.

**3.8 Администраторски панел(фиг. 17)**

Системата за управление на въздушен транспорт съдържа интегриран административен панел, който предоставя възможност за централизирано управление на всички компоненти на проекта. Достъпът до панела е защитен чрез система за аутентикация с потребителско име и парола. Само администраторите могат да управляват следните модули:

* Управление на самолети – добавяне, редакция, изтриване и филтриране по статус.
* Управление на полети – създаване на нови полети, актуализация на данни за полети, изтриване на невалидни записи.
* Управление на пътници – преглед на списък с регистрирани пътници, редактиране на данни или премахване на акаунти.
* Управление на резервации – редактиране, изтриване и генериране на случаен статус ("потвърдена" или "анулирана") на резервации.
* Управление на товарни превози – добавяне на товари, следене на тяхното местоположение, промяна на статус и премахване на доставени товари.

Панелът е реализиран с помощта на Flask, PyMySQL и шаблони на Jinja2, което позволява лесна навигация, визуално управление на данните и високо ниво на сигурност.

Дизайнът на административния панел използва:

* Bootstrap за визуална стилизация.
* Икони за яснота на действията (редактиране, изтриване, проследяване).
* Защита на действията чрез потвърждения ("Сигурни ли сте, че искате да изтриете записа?").

**3.9 Сигурност и сесии**

Използвана е сесийна система чрез Flask.session:

* За администраторите: session["admin\_logged\_in"]
* За пътниците: session["passenger\_id"], session["passenger\_name"]

Достъп до административни маршрути се осъществява чрез @admin\_required декоратор.

**3.10 Интерфейс и стилове**

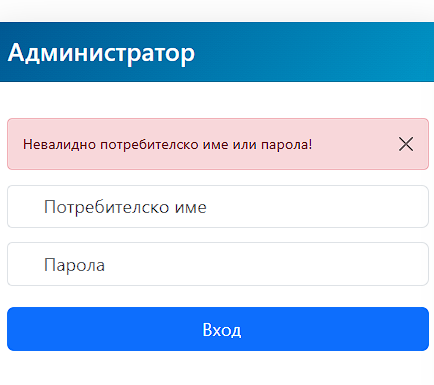
Използвани са:

* Bootstrap 5 – за отзивчив и стилен интерфейс
* Bootstrap Icons – за визуално подобрение
* Персонализирани стилове – за оцветяване на статуси и интерактивни таблици

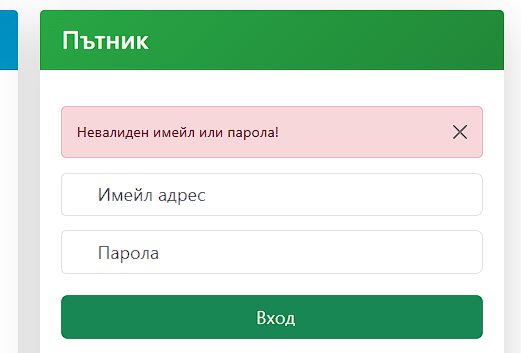
Всеки модул има ясно разграничени зони – таблици, форми и бутони за действия.

**3.11 Тестване и валидация**

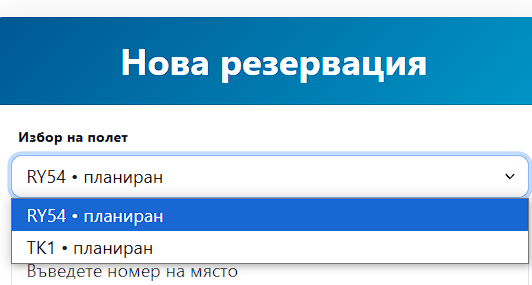
Системата е тествана неколкократно като например въвеждане на грешни входни данни за админ и пътник(фиг.10 и фиг.11), при резервация от пътник се показват само пътнически полети (фиг. 12), при създаване на полет се показват само самолетите, които са в изправност(фиг.13) и последно при добавянето на товари се показват единствено товарни полети (фиг. 14).



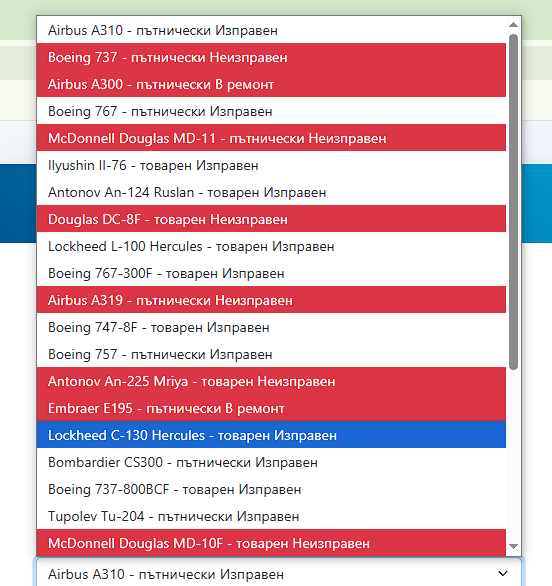
**Фигура 10 – След въвеждане на грешни данни за администратор**



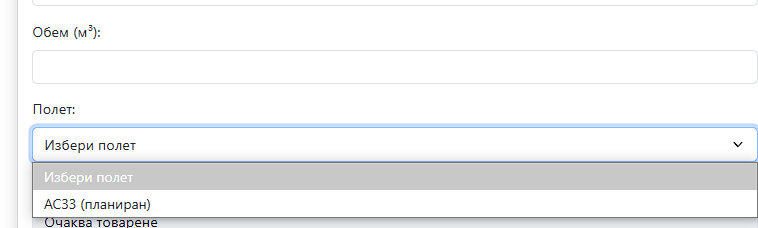
**Фигура 11 – След въвеждане на грешни данни за пътник**

****

**Фигура 12 – Показват се само планираните пътнически полети**

****

**Фигура 13 – При създаване на нов полет се показват само изправните самолети.**

****

**Фигура 14 – Показва се само товарен полет.**

**ЗАКЛЮЧЕНИЕ**

Разработеният дипломен проект представя една цялостна, функционална и интуитивна уеб базирана система за управление на въздушен транспорт, изградена с помощта на Python (Flask)**,** HTML/CSS (Bootstrap)**,** MySQLиPyMySQL. Чрез прилагане на теоретични принципи, добри практики в програмирането и обективен анализ на съществуващите нужди, проектът успява да отговори на реални изисквания в контекста на управлението на въздушни превози – както за пътнически, така и за товарни полети.

По време на реализирането на системата се постигна:

* Внедряване на автентикация и администраторски панел за контрол върху всички модули;
* Разделяне на потребителски и административни функционалности;
* Автоматизация на процеси като резервации, товарене, проследяване и статуси;
* Реализация на проверки и валидации за безопасно и коректно въвеждане на данни;
* Възможност за разширение на системата с допълнителни функционалности (например известяване, API интеграции и др.).

Изградената система намира пряко приложение в реални летищни или логистични среди, в учебни институции за обучение по бази данни и уеб технологии, както и като основа за бъдещо разширяване до пълнофункционална информационна система.

**СПИСЪК НА ИЗПОЛЗВАНАТА ЛИТЕРАТУРА**

1. Flask Documentation. https://flask.palletsprojects.com

2. PyMySQL Documentation. <https://pymysql.readthedocs.io>

3. MySQL 8.0 Reference Manual. <https://dev.mysql.com/doc/refman/8.0/en/>

4. Bootstrap 5 Documentation. https://getbootstrap.com/docs/5.3/

5. Jinja2 Template Engine Documentation. https://jinja.palletsprojects.com

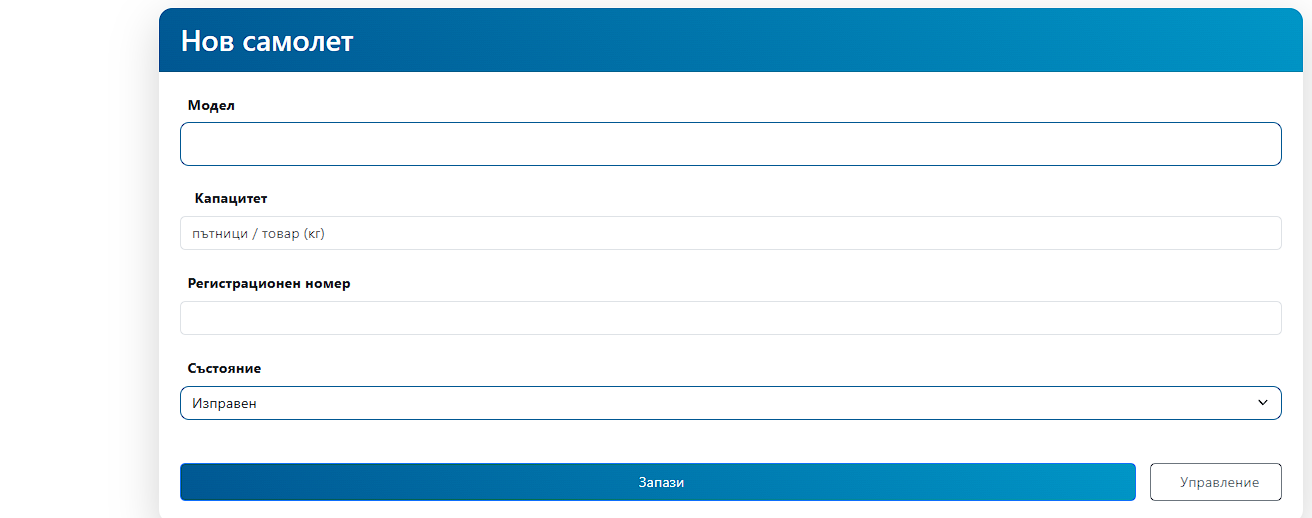
6. Stack Overflow (различни технически решения по конкретни проблеми)

7. ChatGPT - Генериране на структурата по задание

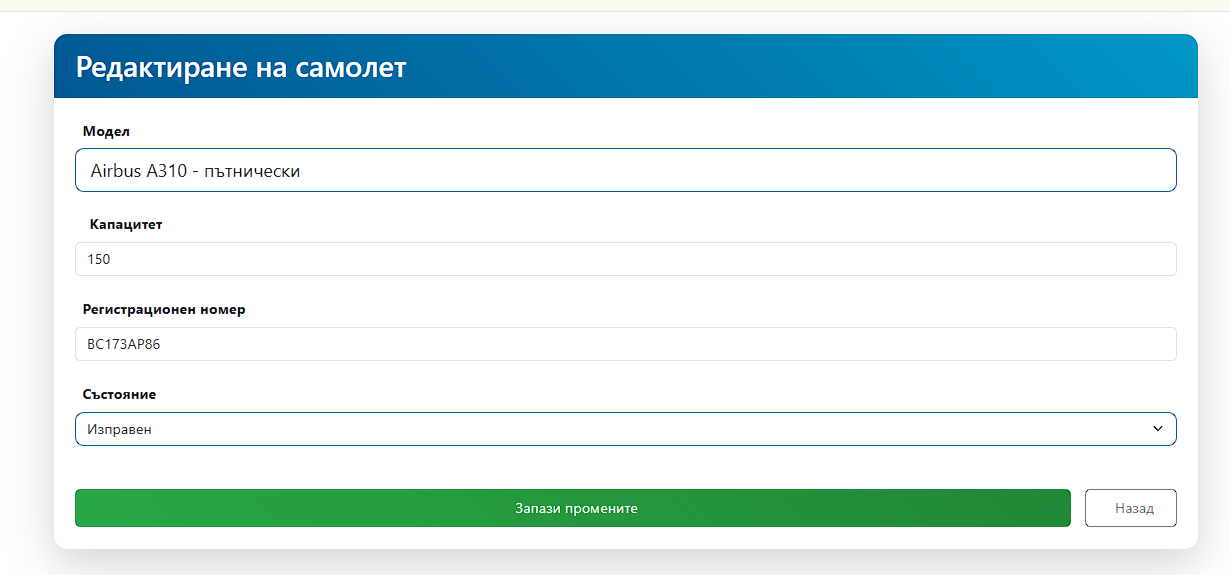
**ПРИЛОЖЕНИЯ**

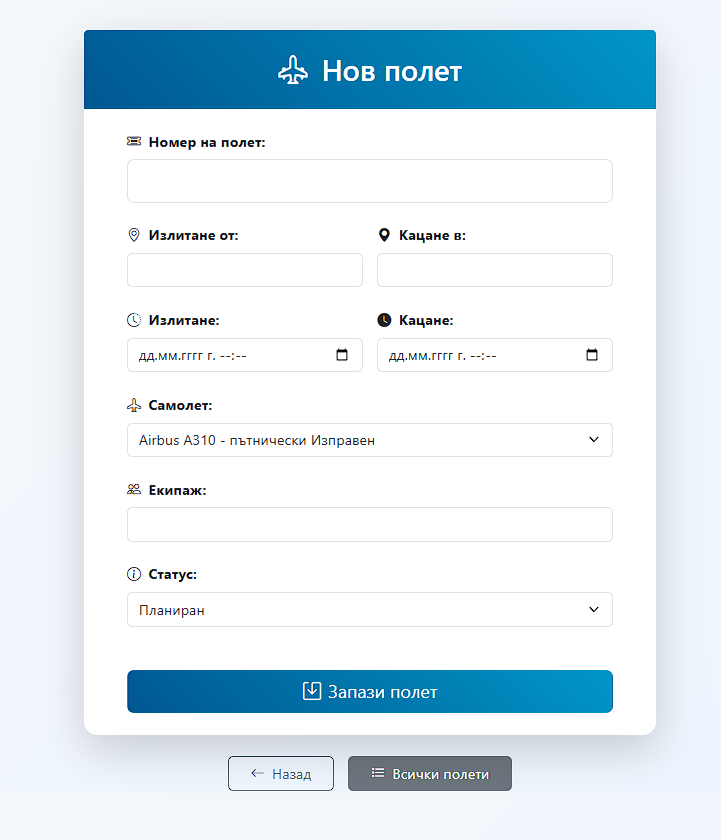
***ПРИЛОЖЕНИЕ 1/HTML шаблони***

**Фигура 15 – Шаблон за добавяне на нов самолет.**

****

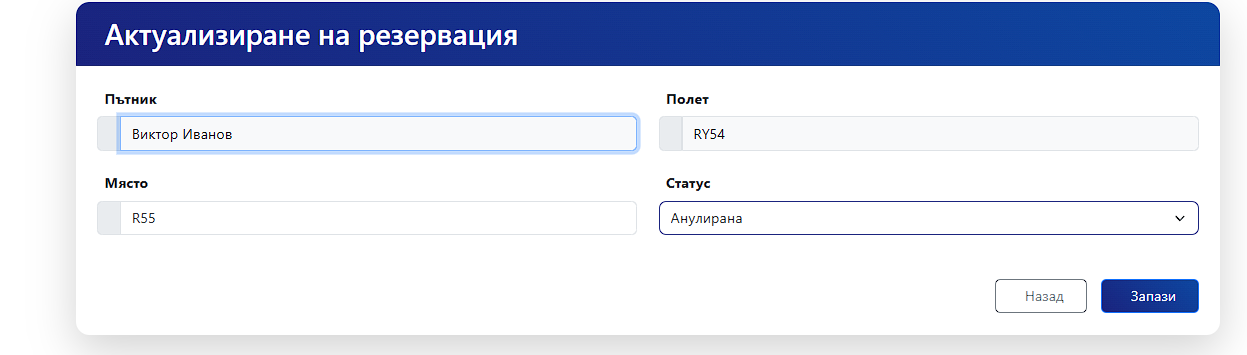
**Фигура 16 – Шаблон за редакция на самолет**.

****

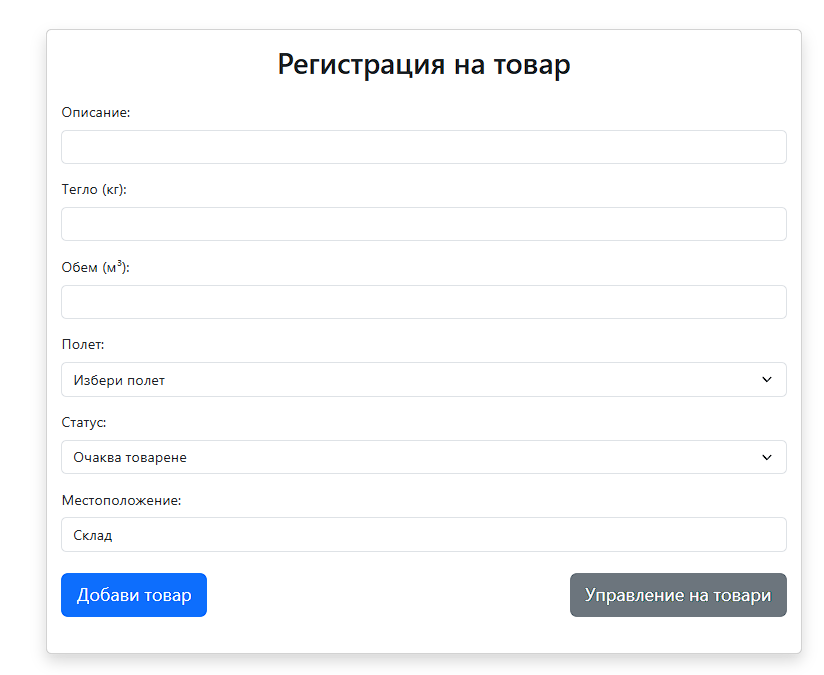
****

**Фигура 17 – Шаблон за създаване на нов полет**.

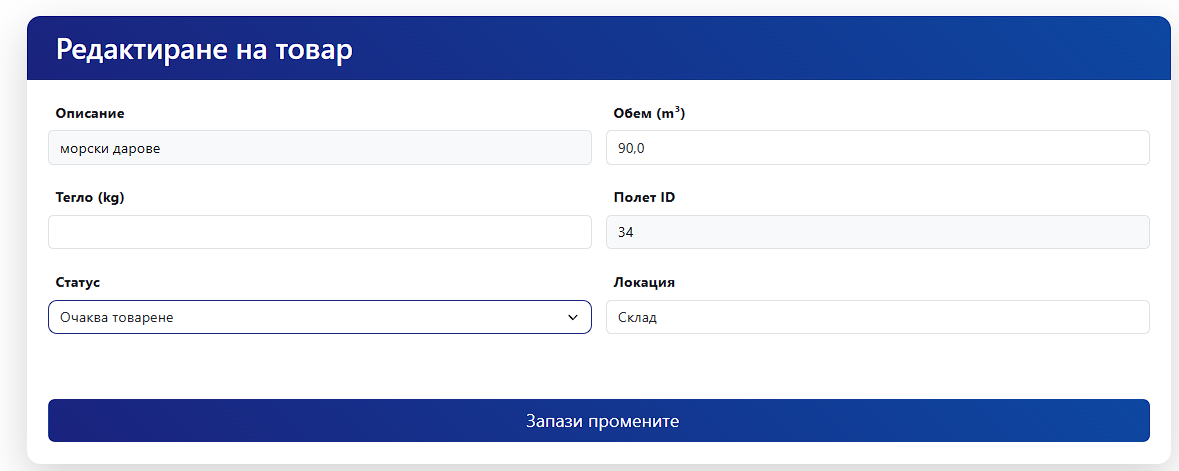
**Фигура 18 – Шаблон за актуализация на резервация.**

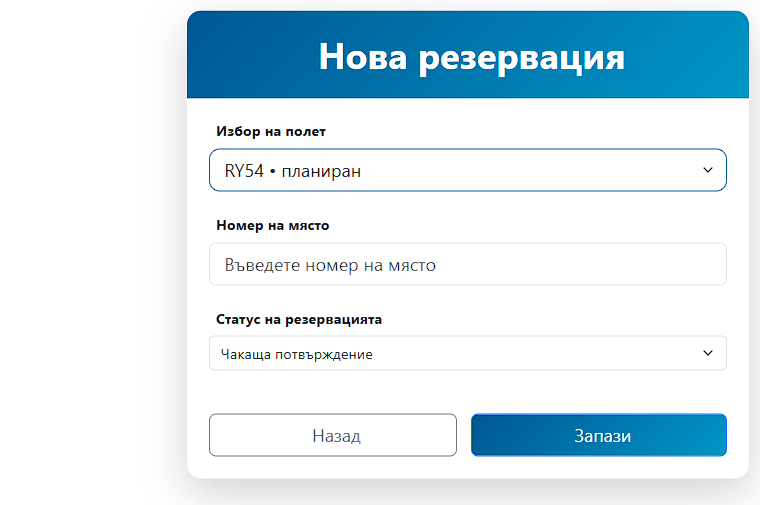


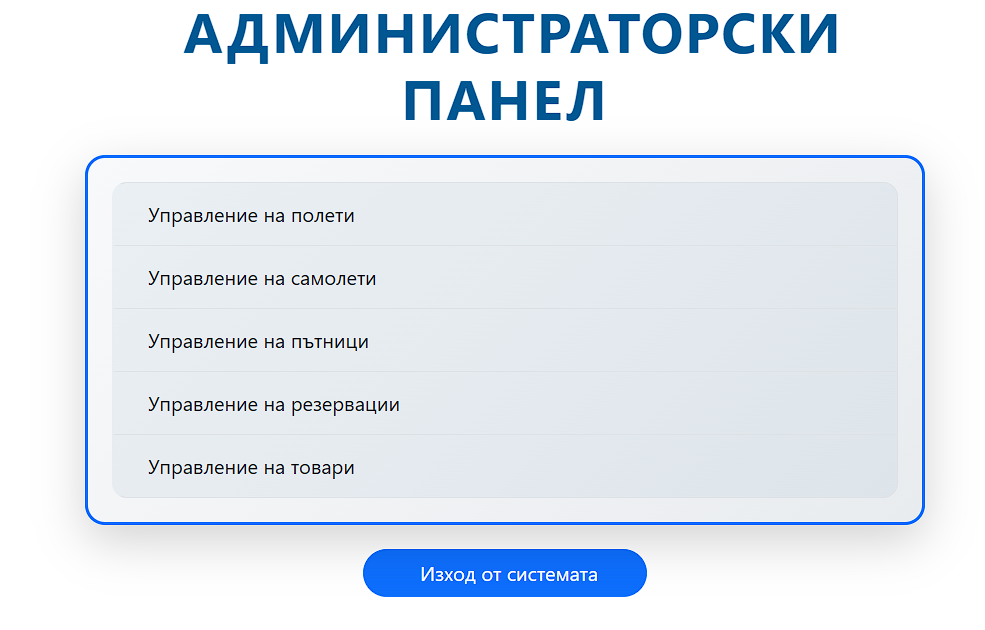
**Фигура 19 – Шаблон за регистрация на товар.**



**Фигура 20 – Шаблон за редакция на товар.**

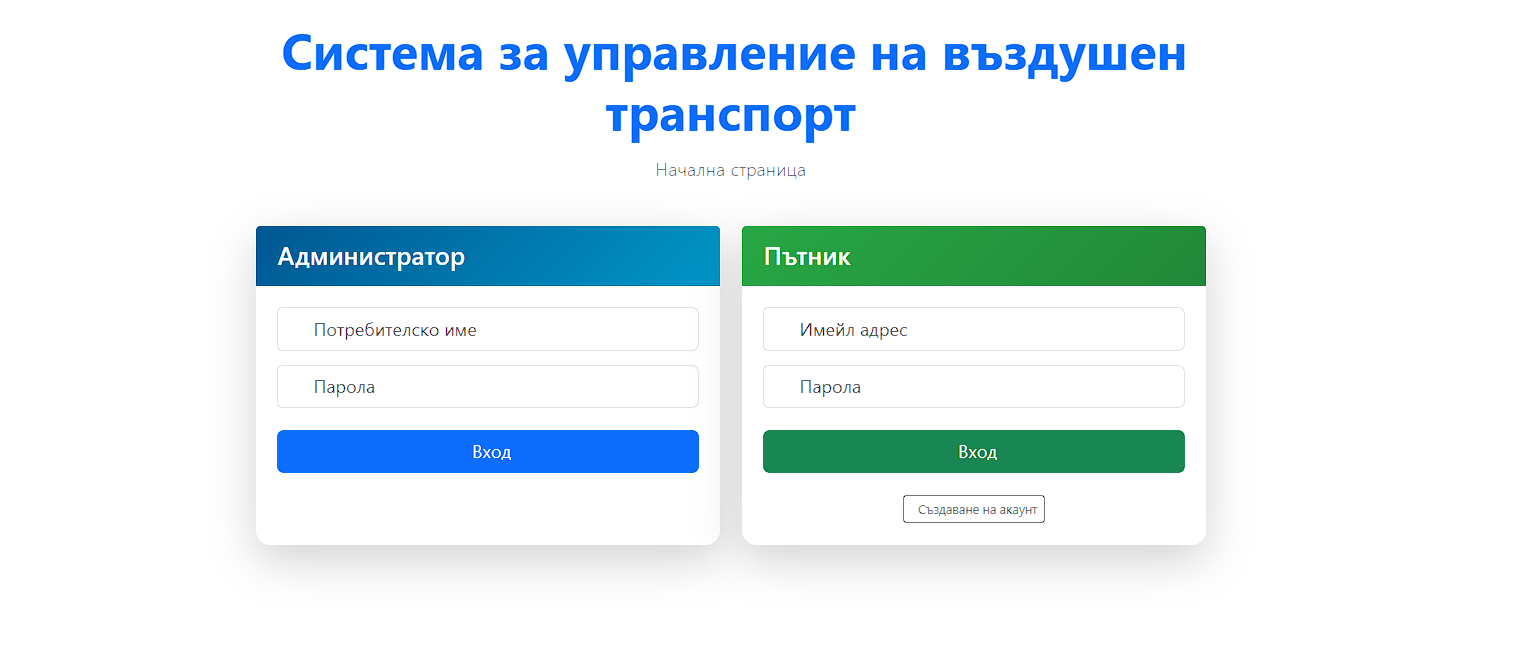


**Фигура 22 – Шаблон за създаване на нова резервация от пътник**.



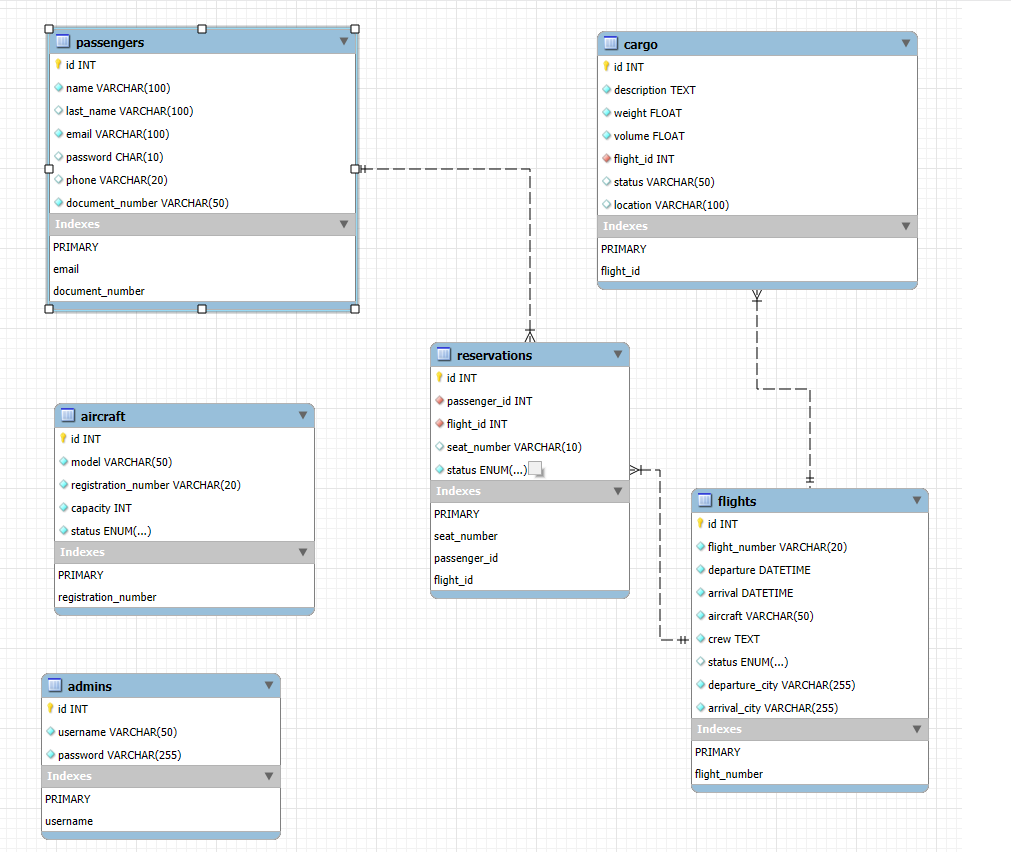
**Фигура 21 – Админ панел**

**Фигура 23 – Шаблон за входни данни на админ и пътник**.



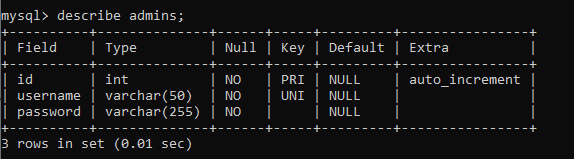
***ПРИЛОЖЕНИЕ 2/Структура на базата данни(ER диаграма)***

**Фигура 24 – ER Диаграма – връзки между таблиците в базата данни.**

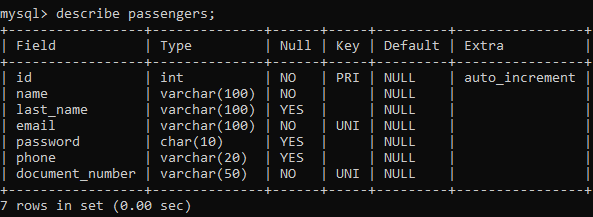
****

***ПРИЛОЖЕНИЕ 3/Основните таблици в базата данни***

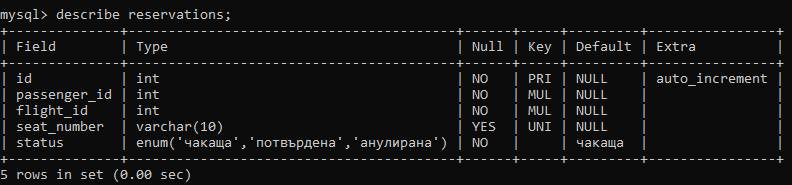
**Фигура 25 – Таблица admins в база данни air\_transport\_system\_db**

****

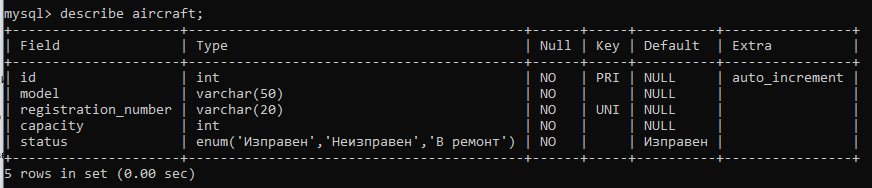
**Фигура 26 – Таблица passengers в база данни air\_transport\_system\_db**

****

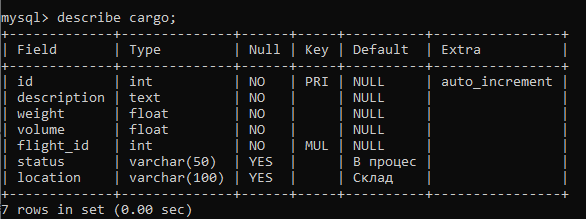
**Фигура 27 – Таблица reservations в база данни air\_transport\_system\_db**

****

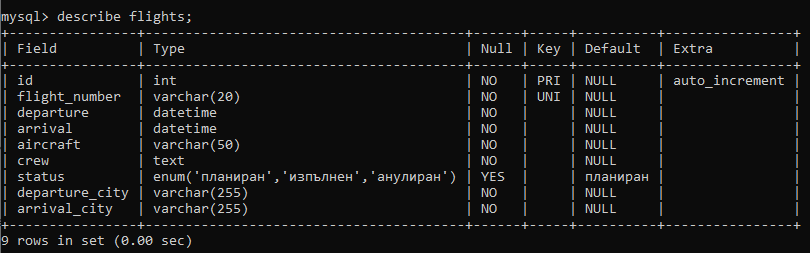
**Фигура 28 - Таблица aircraft в база данни air\_transport\_system\_db**

****

**Фигура 29 - Таблица cargo в база данни air\_transport\_system\_db**

****

**Фигура 30 - Таблица flights в база данни air\_transport\_system\_db**

****