Техническа Документация

**Система за следене и управление на полетите в летище**

Екип:

Дата: 01.03.2025

**История на техническата документация**

**Автори**

| Фак. номер | Име | Контакт (ел. поща) |
| --- | --- | --- |
| 501222071 | Юмер Местан | omer987@outlook.com |
| 501222056 | Христо Стоилов | hrstoilov@tu-sofia.bg |
| 501222019 | Алекс Тенев | atenev@tu-sofia.bg |
| 501222004 | Димитър Георгиев | dimitar.03@abv.bg |

**История на версиите**

| Версия | Дата | Автор | Описание |
| --- | --- | --- | --- |
| 0.0.1 | 01.03.2025 | Юмер Местан | * Цели на системата: дефинирани са основните цели и предназначението на системата * Начални функционални и нефункционални изисквания: формулирани са първоначалните изисквания към програмната система. * Спецификация: представена е структурата на изискванията и начина, по който ще бъдат документирани. * Концептуален дизайн и декомпозиция: описани са архитектурата на системата, модулите и взаимодействията помежду им. * Акроними: добавен е списък с използваните съкращения в документацията за по-голяма яснота. |
| 0.0.2 | 01.03.2025 | Юмер Местан | * Добавени User Stories в Product Backlog (Упражнение 2): прехвърлени и описани потребителските истории в backlog таблица, включително приоритети, story points и acceptance criteria. * Актуализирани критерии за приемане: разширени с конкретни условия за всяка нова история. * Дребни корекции: поправени правописни/стилистични грешки от предишни секции. |
| 0.0.3 | 25.03.2025 | Юмер Местан | * Анализ на изискванията: въведени са подробни функционални и нефункционални изисквания, ограничения и потребителски роли. * Разширен и актуализиран списък с потребителски истории (User Stories) и критерии за приемане към всяка от тях. * Генерирана концептуална ER диаграма на базата данни. * UML диаграма за архитектурата на системата – MTV модел с Django, REST API, React и сесии. * Добавен раздел „Концептуален дизайн на системата“: клиент-сървър архитектура с Django, MySQL, REST и авторизация. * Декомпозиция на системата на модули: Presentation (UI), Business Logic, Data Access, Security. * Разграничен интерфейс по роли – Django Admin за персонал, React UI за пътници и екипаж. |

**Съдържание**

[**1**](#_3znysh7) **Въведение 4**

[1.1](#_2et92p0) Цели на системата 4

[1.2](#_tyjcwt) Обхват на системата 4

[**2**](#_3dy6vkm) **Спецификация на изискванията 4**

[2.1](#_1t3h5sf) Системни изисквания 4

[2.1.1](#_4d34og8) Функционални изисквания 4

[2.1.2](#_2s8eyo1) Нефункционални изисквания 4

[2.2](#_3rdcrjn) Изследване на потребителските изисквания 4

[2.2.1](#_26in1rg) Потребителски истории 4

[2.2.2](#_lnxbz9) Критерии за приемане на потребителски истории 5

[**3**](#_35nkun2) **Концептуален дизайн на системата 5**

[3.1](#_1ksv4uv) Софтуерна архитектура на система 5

[3.2](#_44sinio) Декомпозиция на системата на модули 5

[3.3](#_2jxsxqh) Концептуален модел на БД 5

[**4**](#_z337ya) **Ресурси 5**

[4.1](#_3j2qqm3) Акроними 5

[4.2](#_1y810tw) Други 5

# Въведение

## Цели на системата

Целта на програмната система е изграждането на база данни и програмни инструменти за нейната актуализация, както и за проследяване на актуална информация за текущи, отминали и предстоящи полети от и към определено летище.

За всеки полет системата предоставя информация за:

* начална и крайна дестинация,
* екипаж,
* номер на полет,
* тип на летателния апарат,
* продължителност на полета,
* брой прикачвания,
* тип на полета (директен или с прекачвания),
* междинни спирки,
* управляваща авиокомпания,
* цена на полета.

Системата предоставя възможност на потребителите (пътници) за търсене на полети по:

* начална и крайна дестинация,
* брой прикачвания,
* тип на летателния апарат,
* продължителност на полета.
* ден/час на излитане и пристигане
* цена на билет

Системата предоставя опция на потребителите (оператори) за търсене на полети по всички полета описани за “пътник”, като ги разширява с:

* член на екипаж
* управляваща компания

Системата позволява създаване на нови полети и редактиране на свързаните с тях данни от оператор, като изборът на стойности е ограничен до предварително зададени стойности от администратора в базата данни за всеки от обектите.

Всяка междинна спирка се описва чрез идентификатор на дестинацията.

Всеки екипаж се състои от различен брой членове и може да бъде променян за всеки отделен полет. Член на екипажа не може да бъде разпределен към застъпващи се по време полети, като допълнително крайната дестинация на последния му полет трябва да съвпада с началната дестинация на следващия (приема се, че не се връща с друг транспорт).

Системата е уеб-базирана и предвижда административен и общ потребителски достъп.

Административният достъп е на две нива: Администратор и Оператор.  
 Администраторът има права да управлява добавянето на записи във всички таблици, докато Операторът може да добавя данни само в таблицата „Полети“ и да преглежда останалите таблици без право на редакция.

## Обхват на системата

Функционален обхват:

* Системата за проследяване и управление на полети в летище ще предоставя следните функционалности:
* Управление на полети – добавяне, редактиране и следене на статуса на полетите.
* Управление на екипажи – регистриране на пилоти и кабинен персонал, свързване с конкретни полети.
* Летищна информация – поддръжка на база данни с летища, междинни спирки и авиокомпании.
* Търсене и филтриране – възможност за намиране на полети по различни критерии.
* Административен панел – контрол на достъпа и управление на потребителските роли.

Технологичен обхват:

* Системата ще бъде уеб-базирано приложение с клиент-сървър архитектура.
* Ще използва релационна база данни за съхранение на информацията.
* Софтуерът ще бъде разработен с помощта на обектно-ориентиран програмен език.
* Ще се прилагат добри практики за сигурност и защита на данните.
* Методология на разработка
* Разработката ще следва Agile методология с итеративен процес на подобрение.
* Използване на Git за контрол на версиите.
* Разделяне на задачите по Scrum подход с дефинирани спринтове.

Тестване на софтуера:

* Системата ще премине през следните тестови етапи:
* Модулно тестване – проверка на отделните компоненти.
* Интеграционно тестване – валидиране на взаимодействието между модулите.
* Функционално тестване – удостоверяване на коректността на всички функции.
* Натоварване и производителност – анализ на системата при висока употреба.
* Тестове за сигурност – оценка на защитата на данните и потребителските права.

Ограничения на системата:

* Първоначалната версия няма да включва мобилно приложение, а само уеб-базирано решение.
* Системата не предоставя резервация на билети – тя е предназначена за вътрешно управление на летищните операции.
* Данните за полетите ще се актуализират ръчно или чрез външни API-та, но не в реално време чрез автоматизирани радари.
* Информацията за полетите се основава на данни от авиокомпаниите и летищните оператори – тя не предоставя автоматични актуализации чрез външни радарни системи.
* Не поддържа директно управление на летателни апарати, а само регистрация на техните полети.

# Спецификация на изискванията

## Системни изисквания

### Функционални изисквания

| **Номер** | **Изисквания** | **Приоритет** |
| --- | --- | --- |
| FR1 | Системата трябва да позволява добавяне, редактиране и изтриване на полети | Висок |
| FR2 | Търсене на полети по дестинация, авиокомпания, дата и статус (излита, каца, отменен, закъснява) | Висок |
| FR3 | Управление на екипажа за всеки полет (добавяне, премахване, преназначаване не членове) | Висок |
| FR4 | Поддържане на база данни с летища и междинни спирки (добавяне и премахване) | Среден |
| FR5 | Автоматично актуализиране на статуса на полета (ако данните идват от външни източници/API) | Среден |
| FR6 | Генериране на справки (отчети) за история на полети, дестинации и статистики за натовареност | Среден |
| FR7 | Администраторът трябва да може да управлява потребителски роли (админ, оператор, наблюдателен) | Висок |
| FR8 | Системата да позволява управление на типовете летателни апарати (добавяне, редактиране, изтриване) | Нисък |

### Нефункционални изисквания

| **Номер** | **Изисквания** | **Приоритет** |
| --- | --- | --- |
| NFR1 | Системата трябва да бъде уеб-базирана с отзивчив дизайн и достъпна през стандартни браузъри (Chrome, Firefox, Edge) | Висок |
| NFR2 | Времето за зареждане на главните функционални страници (търсене, добавяне на полети и т.н.) да не надвишава 2 секунди при нормално натоварване | Висок |
| NFR3 | Системата трябва да поддържа минимум 1000 едновременни потребители без деградация на производителността | Среден |
| NFR4 | Данните да се съхраняват в релационна база данни (например MySQL/PostgreSQL) с възможност за резервиране и архивиране | Висок |
| NFR5 | Достъпът до административната част да бъде защитен чрез двуфакторна автентикация | Среден |
| NFR6 | Системата да е мащабируема, така че да поддържа бъдещо добавяне на нови функционалности (REST архитектура, модулна структура) | Среден |
| NFR7 | При срив или неочаквано прекъсване, системата да се възстановява автоматично или да позволява бързо ръчно възстановяване | Среден |

## Изследване на потребителските изисквания

### Потребителски истории

| **Номер на потр. историята** | **Като** | **аз искам да** | **така, че** | **точки на историята** |
| --- | --- | --- | --- | --- |
| US1 | Администратор | мога да добавям, редактирам и изтривам полети | информацията да е винаги актуална | 5 |
| US2 | Администратор | мога да добавям, редактирам и изтривам потребители | да повишавам или намалям броят на служителите според необходимостта | 5 |
| US3 | Администратор | задавам роли на потребителите (оператор, наблюдателен, член на екипаж) | всеки да има съответните права | 3 |
| US4 | Администратор | да мога да добавям нови дестинации, членове на екипажи, летателни апарати, летища, авиокомпании | да може операторите да имат достъп до тях при управление на записите за конкретни полети |  |
| US5 | Оператор | търся полети по дестинация и дата | бързо да намира подходящи резултати | 3 |
| US6 | Оператор | преглеждам и редактирам данните за всеки полет:  - начална и крайна дестинация,  - екипаж,  - номер на полет,  - тип на летателния апарат,  - продължителност на полета,  - брой прекачвания,  - тип на полета (директен или с прекачвания),  - междинни спирки,  - управляваща авиокомпания,  - цена на полета. | да мога да реагирам на промени в последния момент | 4 |
| US7 | Потребител (пътник) | виждам текущите полети и дали има закъснения | да планирам времето си по-добре | 2 |
| US8 | Потребител (пътник) | да мога да се регистрирам | за да ползвам функционалностите на системата | 3 |
| US9 | Потребител (пътник) | Да мога да преглеждам/ филтрирам/ търся по дестинация, номер на полет, час данни за полетите | да мога бързо и лесно да намеря интересуващата ме информация | 3 |
| US10 | Нерегистриран потребител | Да няма достъп до преглед на данни от системата. | да не злоупотребя | 4 |
| US11 | Регистриран потребител (пътник) | Да може да инициира изтриване на профила си | за да премахне всяка лична информация от системата | 2 |
| US12 | Член на екипаж | Да имам достъп до предстоящите ми полети и детайлна справка за тях, както и информация за останалите членове на екипажа с когото летя | имам бърза справка | 2 |

### Критерии за приемане на потребителски истории

| **Номер на потр. историята** | **Критерии за приемане** |
| --- | --- |
| US1 | - Може да се създава нов полет с всички нужни полета.  - Може да се редактира съществуващ полет.  - Може да се изтрива полет.  - Валидират се данните и се показва съобщение при грешка или успех. |
| US2 | - Може да се добави потребител с роля.  - Може да се редактира потребителски профил.  - Потребител може да бъде изтрит.  - Имейл адресът трябва да е уникален. |
| US3 | - Вижда се списък с потребители и роли.  - Може да се задава и променя роля (оператор, наблюдателен, екипаж).  - Новите права влизат в сила след презареждане. |
| US4 | - Могат да се добавят всички обекти (дестинации и др.).  - Новите обекти са налични за операторите.  - Валидират се данните преди запис. |
| US5 | - Може да се търси по начална/крайна дестинация и дата.  - Резултати се зареждат до 3 секунди.  - Ако няма резултати, се показва съобщение. |
| US6 | - Има достъп до пълен списък с полети.  - Всички полета (екипаж, дестинация и др.) са редактирани.  - Промените се записват успешно.  - Включена е валидация на полетните данни. |
| US7 | - Показват се текущите полети.  - Видима е информация за закъснения (в минути). |
| US8 | - Регистрационна форма е достъпна.  - След регистрация потребителят влиза в профила си.  - Всички полета са задължителни и валидирани. |
| US9 | - Работеща търсачка по три критерия.  - Резултатите са ясни и структурирани.  - Невалидно търсене дава подходящо съобщение. |
| US10 | - Без достъп до данни.  - Опитът води до отказ или пренасочване към вход. |
| US11 | - Има бутон „Изтрий профил“.  - Изисква се потвърждение.  - След това данните се изтриват и системата дава потвърждение. |
| US12 | - Списък с предстоящи полети е наличен.  - Всяка справка съдържа данни за другите членове на екипажа.  - Зареждането е до 3 секунди. |

# Концептуален дизайн на системата

## Софтуерна архитектура на система

Системата следва клиент-сървър архитектура, където презентационният слой (уеб интерфейс) комуникира с бизнес-логиката чрез REST API. За да постигнем бърза имплементация, ще ползваме Django, с неговия интегриран ORM модел за мапване между Класове и Таблици в базата данни.

Базата данни която ще се ползва ще е MySQL - като за целите на development етапа и проектирането, ще се спрем на sqlite.

Достъпът до базата данни се осъществява чрез ORM модела на Django.

Оторизирането се осъществява чрез стандартен логин middleware с username и парола.

Достъпът до агрегирани и филтрирани данни се осъществява през MVT модела, конкретно през “Views”, които ще конфигурираме да връщат JSON обекти, за да може да се ползват в бъдеще от web + мобилни приложения.

## Декомпозиция на системата на модули

* **Presentation (UI)**: Съдържа всички уеб-страници, форми, бутони.

Тук разделяме презентационни слой на 2 отделни класа - за административния персонал, който въвежда и управлява информацията и за потребителите - обикновени пътници и членове на екипажи.

За административния персонал ще се разчита на Django Admin модула, като за целта е нужно да се предвидят добавянето на филтри и търсена в стандартните Admin Views.

За обикновените потребители - ще се изгради web приложение за достъп до публичната информация, базирано на React framework-а, което да предостави по-интуитивен и подреден UI.

* **Business Logic**: Тук е логиката за валидации, алгоритми за търсене, справки.

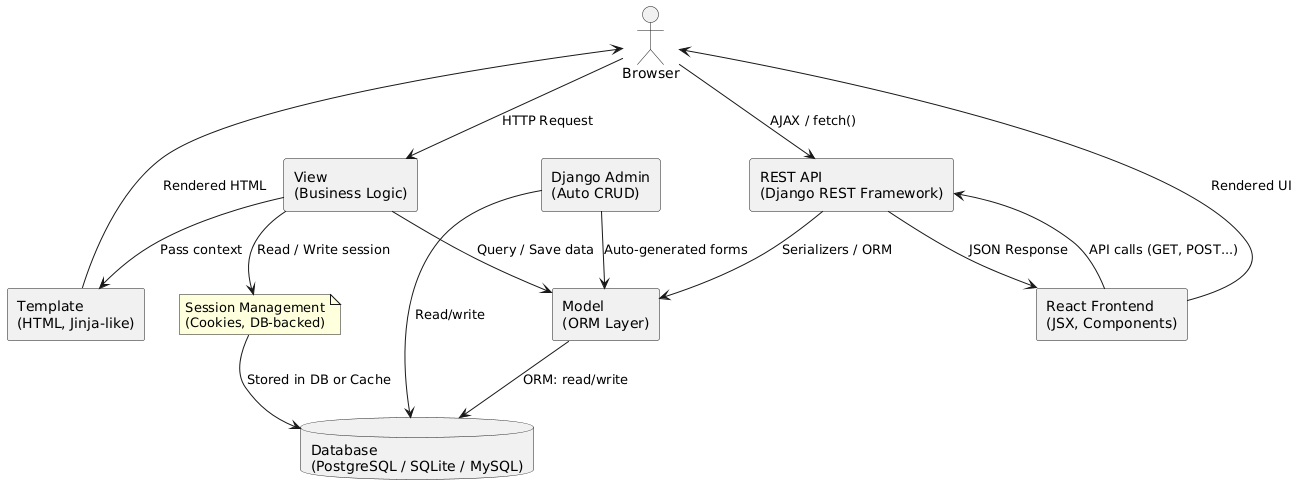
Бизнес логиката ще се изгради на база на MTV модела, като от една страна ще се заложат рестрикции при дефинирането на моделите (за да валидираме входните данни) а от друга, когато не е възможно, ще се ползват API routes привързани към конкретни views от MTV модела на Django, където да се извърши предварителното валидиране на входни данни които се подават за запис или за конфигуриране на заявки за търсене на информация.

* **Data Access**: Слоят за комуникация с базата данни.

Достъпът до базата данни се осъществява от ORM-a на Django.

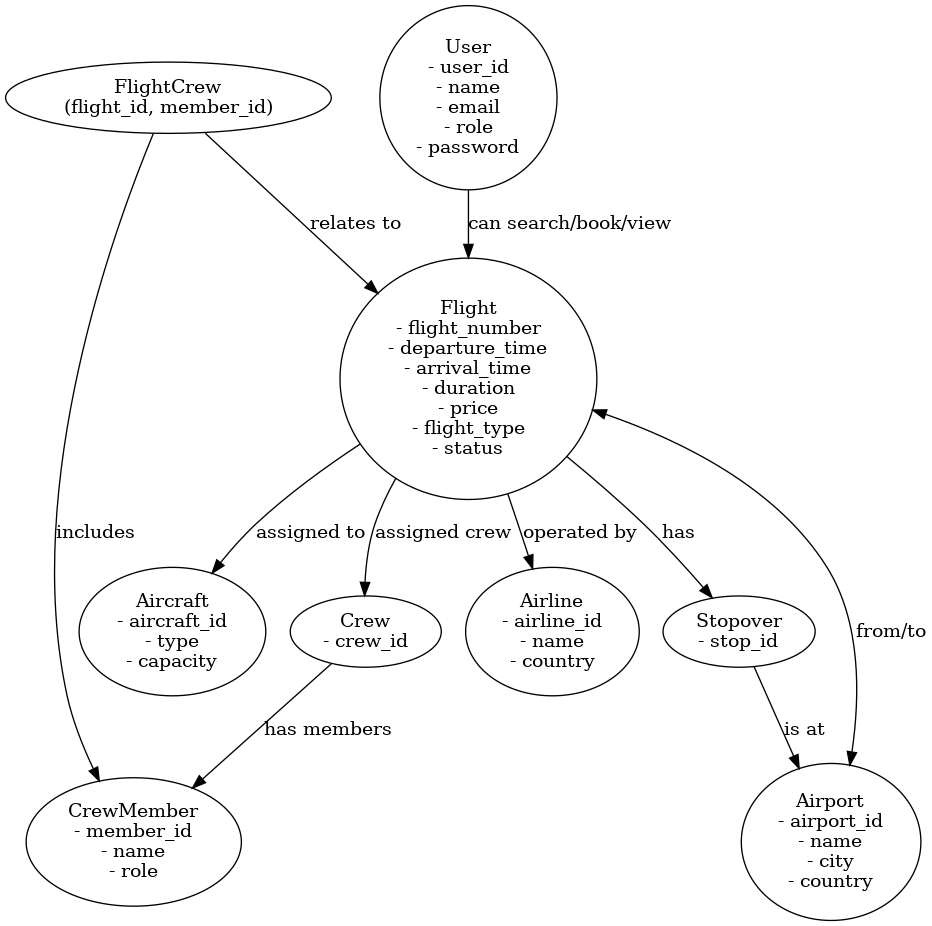
* **Security**: Автентификация (JWT, сесии), ролево базирана система.

Тук отново разчитаме на Django session middleware. Освен това предвиждаме отделни роли с ограничен достъп до ресурсите по отношение на правене на промени.

Диаграма на свързаността на отделните компоненти на системата

## Концептуален модел на БД

*ER диаграма*



# Ресурси

## Акроними

*An explanation of any specific terms / acronyms used in this document.*

| **Акроним** | **Описание** |
| --- | --- |
| API | Application Programing Interface - интерфейс за комуникация |
| CRUD | Create, Read, Update, Delete - основни операции над данни |
| ERD | Entity-Relationship Diagram - диаграма за модел на базата данни |
| UI | User Interface - потребителски интерфейс |
| DB | Database - база данни |
| NFR | Non-Functional Requirements - нефункционални изисквания |
| FR | Functional Requirements - функционални изисквания |
| US | User Story - потребителска история |

## Други

*Интернет сайтове откъдето се е взимала информацията.*