**Документация модуля “Пассажиры”**

**Содержание**

Глава 1. Генерация пассажиров………………………………………….……..3

1.1 Ручное создание пассажиров………………………………………...3

1.2 Авто-генерация пассажиров…………………………………………4

Глава 2. Поведение пассажиров………………………………………………..5

2.1 Типы поведения………………………………………………………5

2.2 Статус действия………………………………………………………5

Глава 3. API……………………………………………………………………...7

3.1 Основные эндпоинты………………………………………………...7

Глава 4. База данных……………………………………………………………9

Глава 5. Логирование………………………………………………………….10

Приложение…………………………………………………………………….11

**Глава 1. Генерация пассажиров**

Реализован веб-интерфейс с помощью которого пользователь может генерировать пассажиров. Пользователю предлагается 2 варианта создания пассажиров: ручное создание и авто-генерация.

Ручное создание пассажиров позволяет создать ограниченное количество людей с одинаковыми или случайными характеристиками. Также предусмотрена возможность создание пассажиров всех поведений. Можно задать поведение, вес багажа и само число пассажиров.

Авто-генерация запускает поток, который с определённой частой создаёт заданное количество пассажиров. Интерфейс предоставляет пользователю возможность задать количество пассажиров и частоту их создания, а также параметры самих пассажиров, аналогичные ручному созданию. Авто-генерация работает в отдельном потоке, дабы не мешать основному приложению. Присутствует статистика, демонстрирующая пользователю текущие параметры авто-генерации.

Для обоих видов генерации реализованы подсказки в выборе параметров, а также уведомления об успешности их запроса.

**1.1 Ручное создание пассажиров**

* Метод: POST /create\_passengers
* Параметры запроса (JSON):

1. num\_passengers: Количество пассажиров (Число, обязательное).
2. behavior: Тип поведения (Случайно, Все, Обычный, Возврат, и др.).
3. baggage\_weight: Вес багажа (Число или null для случайного значения).

* Пример запроса:

{  
 "num\_passengers": 5,  
 "behavior": "Случайно",  
 "baggage\_weight": null  
}

* Ответ:

1. Успех: 200, Успешно было создано {num\_passengers} пассажиров.
2. Ошибка: 400 Bad Request, Один из параметров не был заполнен.

**1.2 Авто-генерация пассажиров**

* Методы

1. Метод запуска: POST /start\_auto\_generation
2. Метод остановки: POST /stop\_auto\_generation

* Параметры запроса (JSON):

1. num\_passengers: Количество пассажиров за интервал (Число, обязательное).
2. interval: Интервал генерации в секундах (Число, обязательное).
3. behavior и baggage\_weight: аналогично ручному созданию.

* Пример запроса:

{  
 "num\_passengers": 2,  
 "interval": 60,  
 "behavior": "Обычный"  
}

* Ответ
* Успех: 200, Была запущена авто-генерация.
* Ошибка: 400 Bad Request, Один из параметров не был заполнен.

**Глава 2. Поведение пассажиров**

**2.1 Типы поведения**

Пассажиры имеют предопределенные типы поведения с разной вероятностью:

Таблица 1. Паттерны поведения пассажиров

|  |  |  |
| --- | --- | --- |
| **Тип** | **Вероятность создания** | **Описание** |
| Обычный | 55% | Стандартный сценарий поведения пассажира. После поиска и получения билета, пассажир идёт на регистрацию, затем ожидает транспортировки, перемещается на борт и улетает. |
| Возврат | 20% | Данный сценарий после поиска покупки билета старается его вернуть и уйти. |
| Мошенник касса | 5% | Пассажир получает данные о билете на момент его поиска, после чего тут же старается его вернуть. |
| Мошенник регистрация | 5% | Мошенник получает данные о билете на момент его поиска, после чего отправляется на регистрацию, минуя этап покупки. |
| Опоздавший касса | 10% | Пассажир, после поиска билета, приходит слишком поздно и не успевает купить билет. С шансом в 30% попробует купить другой билет или с 70% уйти. |
| Опоздавший регистрация | 5% | Пассажир, который нашёл и купил билет, но не успел на регистрацию. Уходит без попытки купить другой билет. |

**2.2 Статус действия**

Помимо поведения, пассажиры руководствуются статусом и временем действия для своих поступков. Время действия определяет момент, когда пассажир что-то сделает. Статус определяет действие, которое совершит пассажир. Он может иметь следующие значения:

Таблица 2. Статусы пассажиров

|  |  |
| --- | --- |
| **Наименование статуса** | **Описание** |
| Поиск билета | Статус, с которым появляются новые пассажиры. Все пассажиры с этим статусом запрашивают случайные рейсы из соответствующей таблицы, после чего приобретают статус ‘Покупка билета’. Если таковых нет, удаляются. |
| Покупка билета | Пассажиры с этим статусом отправляются на кассу и приобретает билет на рейс, выбранный в предыдущем пункте. В случае успеха, приобретают статус ‘Ожидание регистрации’. В случае неудачи, с некоторым шансом приобретают статус ‘Поиск билета’ и вновь пробуют его купить, иначе статус ‘Удаление’. |
| Ожидание покупки билета | Специальный статус для пассажиров с поведением ‘Опоздавший касса’. Данный статус не даёт пассажиру купить билет пока рейс не начнёт регистрацию. Как только настаёт этот момент, пассажир получает статус ‘Покупка билета’. |
| Возврат билета | Пассажиры с этим статусом обращаются на кассу и пробуют вернуть билет. В случае успеха, получают статус ‘Удаление’. В противном случае, если есть рейсы с открытой кассой, получает статус ‘Поиск билета’. Если таковых нет, статус ‘Удаление’. |
| Ожидание регистрации | Этот статус получает все пассажиры, на рейс которых ещё не открыта регистрация, но билет уже куплен ими (Или мошенник). Как только открывается регистрация, регистрация, меняют свой статус на соответствующий. Если пассажир имеет поведение ‘Опоздавший регистрация', то статус меняется лишь по окончанию регистрации на соответствующий рейс. |
| Регистрация | Пассажиры с данным статусом отправляются на регистрацию. В случае успеха, приобретают статус ‘На посадку’, в противном случае ‘Удаление’. |
| На посадку | Служебный статус, показывающий какие пассажиры прошли регистрацию и ожидают транспортировку на рейс. |
| Транспортировка | Служебный статус, показывающий какие пассажиры транспортируются на свой рейс. |
| На борту | Служебный статус, показывающий какие пассажиры находятся на борту. По истечении некоторого модельного времени приобретают статус ‘Удаление’. |
| Удаление | Служебный статус, показывающий какие пассажиры будут удаление в следующий тик модельного времени. |

**Глава 3. API**

**3.1 Основные эндпоинты**

1. Получение доступных рейсов

* Метод: POST /passenger/available-flights
* Назначение: Табло передаёт информацию о новом рейсе на который пассажиры могут купить билет.
* Входные данные: Список рейсов в формате:

[  
 {  
 "flightID": "123", *# ID рейса.* "airplaneId: "123" *# ID самолёта.* }  
]

1. Уведомление о начале регистрации на рейс

* Метод: POST /passenger/check-in/start/<int:flightId>
* Назначение: Табло уведомляет о начале регистрации на рейс, после чего пассажиры со статусом ‘Ожидание регистрации’ могут начать регистрироваться на рейс.
* Входные данные: ID рейса в URL.

1. Уведомление о конце регистрации на рейс

* Метод: POST /passenger/check-in/end/<int:flightId>
* Назначение: Табло уведомляет о конце регистрации на рейс.
* Входные данные: ID рейса в URL.

1. Покупка билета

* Метод: POST /passenger/ticket
* Назначение: Обработка успешной/неуспешной покупки.
* Входные данные:

[  
 {  
 "PassengerId": 1, *# ID пассажира* "Status": "Successful" *# Статус покупки.* }  
]

1. Возврат билета

* Метод: POST /passenger/return-ticket
* Назначение: Обработка возврата билета.
* Входные данные:

[  
 {  
 "PassengerId": 1, *# ID пассажира* "Status": "Unsuccessful" *# Статус возврата* }  
]

1. Регистрация

* Метод: POST /passenger/check-in
* Назначение: Обработка регистрации пассажиров.
* Входные данные:

[  
 {  
 "PassengerId": 1, *# ID пассажира* "Status": "Unsuccessful" *# Статус возврата* }  
]

1. Транспортировка на борт

* Метод: POST /passenger/transporting
* Назначение: Подтверждение транспортировки пассажира.
* Входные данные не требуются.

1. Подтверждение посадки

* Метод: POST /passenger/on-board
* Назначение: Фиксация окончательного статуса "На борту".
* Входные данные не требуются.

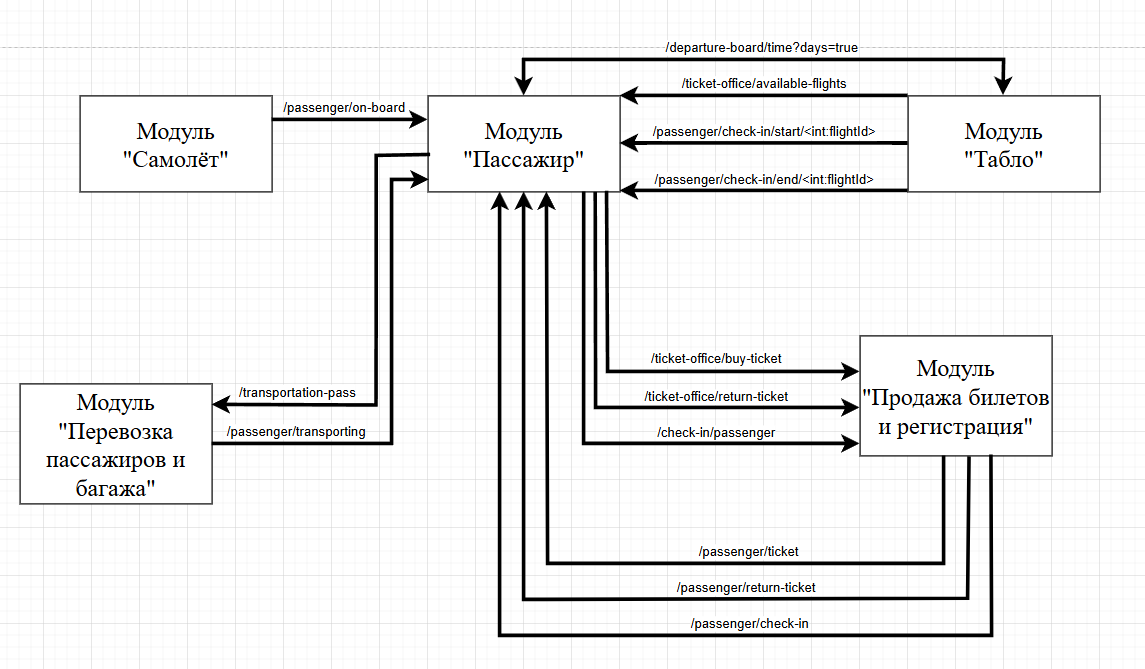


Рисунок 1. Сетевое взаимодействие модуля “Пассажиры”

**Глава 4. База данных**

Данный модуль использует следующую базу данных, состоящую из двух таблиц: пассажиры (passengers) и рейсы (flights).

Таблица 3. Passengers

|  |  |  |
| --- | --- | --- |
| **Столбец** | **Тип данных** | **Описание** |
| ID | Int, primary key autoincrement | Уникальный идентификатор пассажиров. |
| Behavior | Text not null | Паттерн поведения пассажиров. Принимает 1 из 6 значений, упомянутых ранее, с определённым шансом. |
| Status | Text not null | Текущее состояние пассажира, на основе которого решается его следующий шаг. |
| Baggage\_weight | Int not null | Вес багажа пассажира. |
| Action\_time | Text not null | Время действия пассажира при наступлении которого происходит событие. |
| Flight\_id | Int | ID борта билета пассажира. |
| Check\_in\_end\_time | Text | Время конца регистрации на выбранный пассажиром борт. |

Таблица 4. Flights

|  |  |  |
| --- | --- | --- |
| **Столбец** | **Тип данных** | **Описание** |
| Flight\_id | Int not null | ID рейса, на который пассажиры покупают билет. |
| Airplane\_id | Int not null | ID самолёта, на котором пассажиры полетят. |
| Is\_check\_in | Int not null | Булева переменная, показывающая состояние регистрации. При 0 работает касса, но не регистрация. При 1 наоборот. |
| Check\_in\_end\_time | Text | Время конца регистрации после которого строка рейса будет удалена из таблицы. |

**Глава 5. Логирование**

Данный модуль логирует 3 вида событий: действия пользователя, действия пассажиров и действия рейсов. К действиям пользователя относится создание пассажиров и запуск/отмена авто-генерации. Строки данного лога содержат данные о времени действия и параметрах создания пассажиров. Выглядит он следующим образом:

*2025-03-14 17:35:49,003* - Пользователь сгенерировал *1* пассажиров с характеристиками: поведение - Возврат, вес багажа – Случайно

*2025-03-14 23:57:30,219* - Пользователь включил авто-генерацию пассажиров со следующими параметрами: число генерируемых пассажиров - *5*, интервал генераций - *2*, поведение - Обычный, вес багажа – Случайно

*2025-03-14 23:57:33,162* - Пользователь отключил авто-генерацию пассажиров.

Логи пассажиров, вместе со временем действия, отображают ID пассажира и информацию об изменении его статуса. Выглядит такой лог следующим образом:

*2025-03-14 19:16:50,497* – Пассажир *126* изменил свой статус с 'Поиск билета' на 'Покупка билета'.

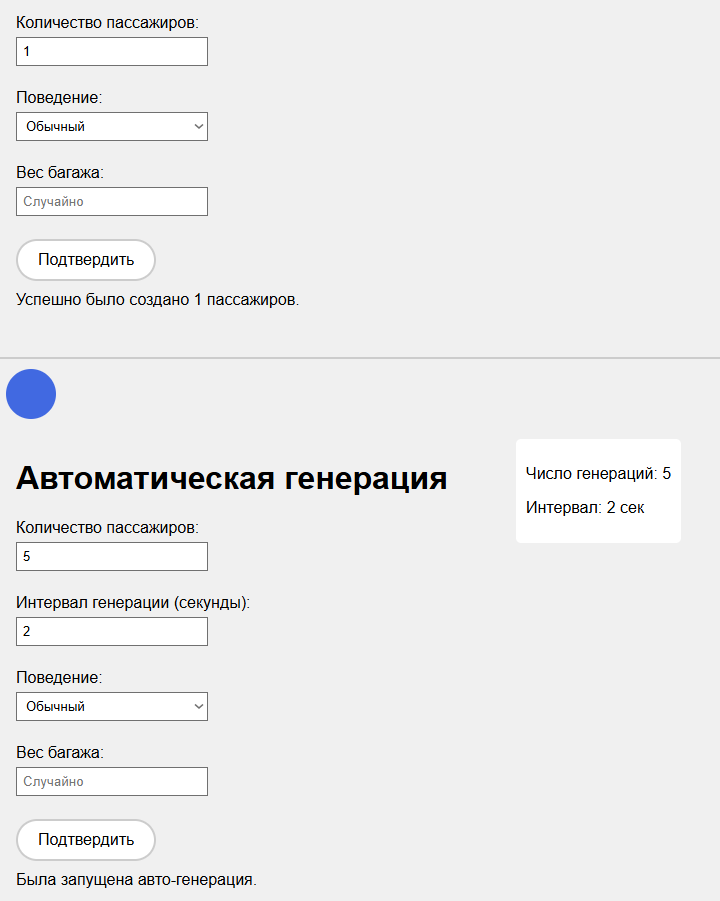
Действия рейсов отображают информацию о начале продаж билетов на рейс, а также о начале и конце регистрации на него.

*2025-03-20 22:19:49,994* - Рейс №*1* стал доступен для покупки билетов.  
*2025-03-20 22:20:03,792* - Началась регистрация на рейс №*1*.  
*2025-03-20 22:20:39,892* - Закончилась регистрация на рейс №*1*.

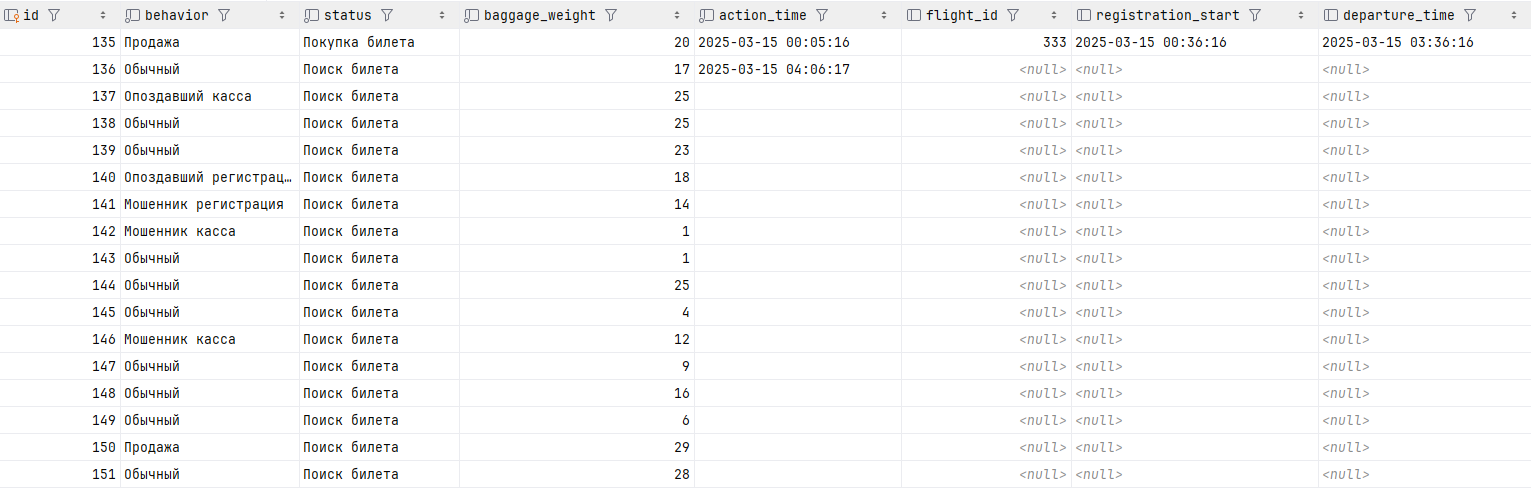
**Приложение**

Приложение 1. Пользовательский интерфейс

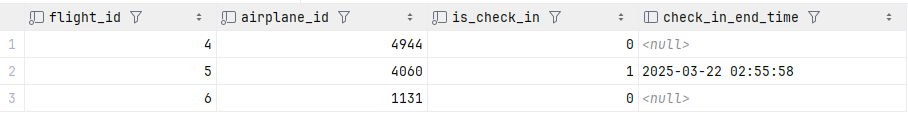
модуля



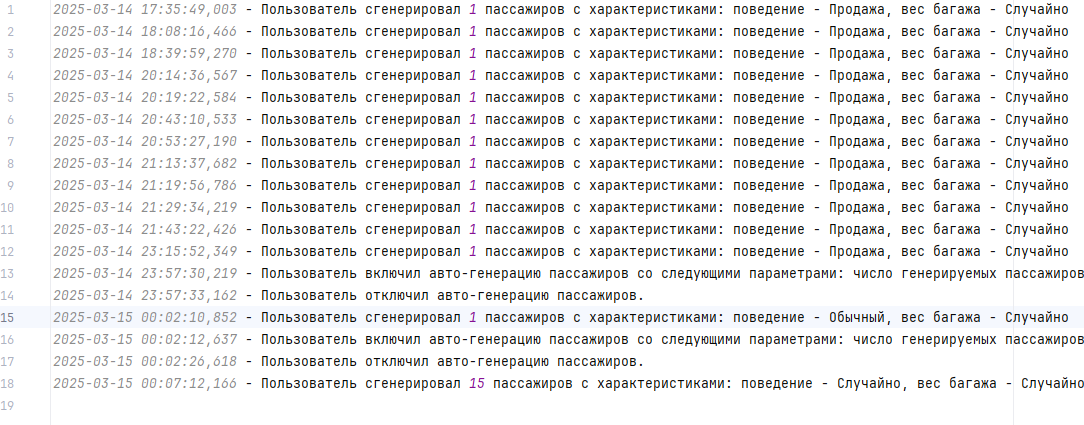
Приложение 2. Таблица Passengers



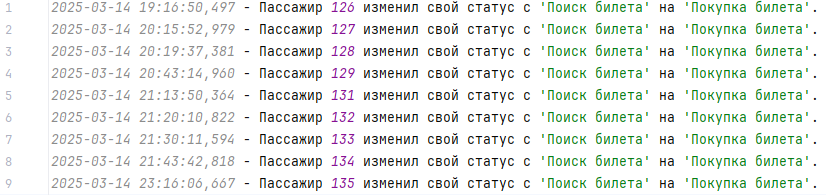
Приложение 3. Таблица Flights



Приложение 4. Логи пользователя



Приложение 5. Логи пассажиров



Приложение 6. Логи рейсов

