# Заказ авиабилетов

# 

[Заказ авиабилетов](#_j5671zetm34w)

[0. Изменения](#_nqb45dyqhjqw)

[1. Введение](#_1a6w12bcocrt)

[2. Описание системы.](#_4y2tcc1qjc3)

[2.1. Администраторы и клиенты.](#_fi8rksfwpnyv)

[2.2. Самолеты и рейсы](#_vavevkxiztr7)

[2.3. Заказ билетов](#_erheo3bu61x)

[3. REST-интерфейс](#_6jvau9gx1251)

[3.1. Общие правила](#_wy0g9gcjf5g1)

[3.2. Регистрация администратора.](#_do68cxn1jmgc)

[3.3. Регистрация клиента.](#_n2yba9rxn4oj)

[3.4. Login.](#_2ywm16oqmg2t)

[3.5. Logout.](#_h2ytru4qs5ac)

[3.6. Получение информации о текущем пользователе](#_k8vgnop2evz5)

[3.7 Администратор - получение информации о клиентах](#_r506hjulte0w)

[3.8. Редактирование профиля администратора](#_op2uu5i18h9i)

[3.9. Редактирование профиля клиента](#_aklot67ll02c)

[3.10 Администратор - получение информации о типах самолетов](#_ypj7vm3p3mgi)

[3.11. Администратор - добавление рейса.](#_6quari9x6cht)

[3.12. Администратор - изменение рейса.](#_gw5qbcr9vkrs)

[3.13. Администратор - удаление рейса.](#_nnebm4vlta2)

[3.14. Администратор - получение информации о рейсе.](#_uhei365eaxwc)

[3.15. Администратор - утверждение рейса.](#_di43188w75fi)

[3.16. Администратор и клиент - получение списка рейсов.](#_4mhsjesb4bf9)

[3.17. Получение списка стран (гражданство)](#_fzow78f5nycy)

[3.18. Клиент - заказ билетов.](#_p3nqeyydq2oj)

[3.19. Администратор и клиент - получение списка заказов.](#_ss9flv6gdrk6)

[3.20. Клиент - получение списка свободных мест для регистрации.](#_ugdr2eoz5gl7)

[3.21. Клиент - регистрация пассажира на рейс.](#_7w8wpcw7m945)

[3.22. Получение настроек сервера](#_yrr7vn3c6fkr)

[3.23. Очистка базы данных.](#_kb147ss7gy5k)

[Приложение 1. Конфигурационный файл сервера.](#_nitfmae1ixwi)

[Таблица 1. Параметры конфигурационного файла](#_mf19ltjfmpsn)

# 

### 0. Изменения

В 3.12 удалено поле “planeName” из Response и “approved” из Request и добавлено поле “approved” в Response

В 3.16 в schedule имя поля “dates” изменено на “period”

В 3.20 уточнен формат места

Исправлены имена полей в возвращаемом json - везде placesInBusinessRow и placesInEconomyRow

В 3.20 убраны ненужные {} в возвращаемом json

Поправлен формат списка дат в запросах и ответах. Должно быть так **"день1”, “день2”,..”деньN"**

# 

## 1. Введение

Авиакомпании “TT Airlines”, приступающей к выполнению пассажирских перевозок, требуется сайт, на котором авиапассажиры могли бы заказывать билеты на рейсы.

Авиакомпания не выполняет рейсы с пересадками и не разрешает провоз животных.

Администраторы авиакомпании должны иметь возможность добавлять и удалять рейсы, определять цену билетов и т.п. Клиенты должны иметь возможность купить билеты и выбирать места при регистрации.

Вам предстоит написать серверную часть приложения.

Руководство авиакомпании предполагает, что высказанные им требования к сайту являются логичными и непротиворечивыми. Так ли это в действительности или нет - должно выясниться в процессе разработки и пробной эксплуатации сайта. Если в ходе разработки или пробной эксплуатации появится необходимость корректировки тех или иных пунктов задания, то этот вопрос должен обсуждаться совместно представителями заказчика и разработчика с внесением в случае необходимости изменений в техническое задание. Изменения возможны как на клиентской, так и на серверной стороне.

Все значения времени на сервере представляются в GMT, локальное (местное) время не используется.

## 2. Описание системы.

### 2.1. Администраторы и клиенты.

Администраторы и клиенты должны зарегистрироваться на сайте. При регистрации они указывают

Администратор :

* фамилию, имя, отчество (последнее - если имеется)
* должность
* логин и пароль для входа на сайт.

Клиент :

* фамилию, имя, отчество (последнее - если имеется)
* e-mail
* номер сотового телефона
* логин и пароль для входа на сайт.

Администраторы и клиенты могут изменить свои регистрационные данные, но не могут изменять логин.

### 2.2. Самолеты и рейсы

Авиакомпания располагает парком самолетов, который не изменяется. Типы самолетов задаются в таблице БД , которая не подлежит редактированию средствами данного сайта. Названия всех типов самолетов уникальные. Для каждого типа самолета в этой таблице указывается его название и количество рядов в бизнес-классе и в экономическом классе. Количество самолетов каждого вида в таблице не присутствует, так как предполагается, что авиакомпания может арендовать дополнительный самолет для рейса в случае необходимости.

Зарегистрированный администратор может добавить новый рейс. После добавления рейса он находится в состоянии “не утвержден”. Пока рейс находится в этом состоянии, администратор может изменить расписание рейса, тип воздушного судна и т.д. , а также удалить рейс, а клиенты не могут заказывать билеты на него. После того, как администратор убедился, что информация о рейсе верна, он утверждает рейс. Теперь клиенты могут заказывать билеты, а изменение параметров рейса и его удаление не разрешается.

### 2.3. Заказ билетов

Зарегистрированный клиент может купить билеты и пройти регистрацию на рейс.

При покупке билета клиент имеет возможность выбрать рейс , указывая дату и маршрут, после чего система предлагает ему подходящие рейсы. Он может купить билеты для любого количества пассажиров. После покупки билетов ему высылаются письмо на его e-mail и смс на его сотовый телефон[[1]](#footnote-0).

Все финансовые вопросы, связанные с заказом или изменением заказа, в рамках данной задачи не рассматриваются.

Клиент может выполнить регистрацию на рейс пассажира, для которого он купил билет. В ходе регистрации определяется место в самолете. После регистрации клиенту высылаются письмо на его e-mail и смс на его сотовый телефон. Если для пассажира регистрация на сайте не будет произведена, его место в самолете определяется при регистрации в аэропорту, у сотрудников которого есть специальная программа для этой цели.

## 3. REST-интерфейс

### 3.1. Общие правила

Для взаимодействия клиентской и серверной частей приложения используется REST-интерфейс. Клиент посылает REST - запрос (Request), передавая в нем строку в формате JSON. Некоторые параметры передаются через заголовок HTTP-запроса. В ответ (Response) сервер возвращает свою строку, также в формате JSON, которая содержит информацию о результате выполненной операции.

Для идентификации администратора или клиента используются cookie. При выполнении операции “Login” сервер возвращает cookie с именем “JAVASESSIONID”. Во всех остальных запросах предполагается, что клиент передает эту cookie.

Далее в документе принята следующая система обозначений.

Текст, содержащий латинские буквы и цифры, должен присутствовать в Request или Response в точности так, как написано. Текст, содержащий русские буквы, должен быть заменен на некоторое допустимое значение.

Например, "name":"имя пользователя" означает, что имя параметра должно быть “name”, а вместо “имя пользователя” должно быть указано некоторое допустимое имя.

Если значение параметра приведено в кавычках, параметр является текстовым, если без кавычек - целый типа int или long.

Порядок следования полей в Request и Response может быть произвольным.

При успешном выполнении всегда возвращается код HTTP 200.

В случае, если запрос не был выполнен из-за какой-то ошибки в нем, возвращается код HTTP 400 (для запроса по несуществующему URL - 404), и json следующего вида

{

"errors" : [  
 {  
 **"errorCode"**: "код ошибки",   
 **"field"**: "поле запроса, являющееся причиной ошибки",   
 **"message"**: "причина ошибки"  
 }

...  
]

}

Например

{

"errors" : [  
 {  
 **"errorCode"**: "LOGIN\_ALREADY\_EXISTS" ,  
 **"field"**: "login",   
 **"message"**: "User Ivanov already exists"   
 }  
]

}

### 3.2. Регистрация администратора.

POST /api/admin

Request

|  |  |
| --- | --- |
| json | {  **"firstName"**: "имя",   **"lastName"**: "фамилия",   **"patronymic"**: "отчество", // необязателен  **"position"**: "должность",  **"login"**: "логин",   **"password"**: "пароль"  } |

Логин может содержать только латинские и русские буквы и цифры и не может быть пустым. Пароль может содержать любые символы и тоже не может быть пустым. Максимальная длина логина, пароля, фамилии , имени и отчества не более ***max\_name\_length[[2]](#footnote-1)*** символов. Минимальная длина пароля ***min\_password\_length*** символов. Логин должен храниться так, как он задан, но не является case-sensitive при дальнейшей работе. Например, если администратор регистрировался с логином “Иванов”, он может впоследствии заходить на сервер, используя логины “Иванов”, “иванов”, “иВаНоВ” и т.д. Пароль является case-sensitive.

Имя, фамилия и отчество администратора могут содержать только русские буквы , пробелы и знак “минус” (используемый как тире).

Для зарегистрировавшегося администратора автоматически выполняется операция “Login” (п.3.4)

Response

|  |  |
| --- | --- |
| cookie | JAVASESSIONID |
| json | **{**  **"id"**: "идентификационный номер",  **"firstName"**: "имя",   **"lastName"**: "фамилия",   **"patronymic"**: "отчество",   **"position"**: "должность",   **"userType"**: "admin"  **}** |

### 

### 3.3. Регистрация клиента.

POST /api/client

Request

|  |  |
| --- | --- |
| json | {   **"firstName"**: "имя",   **"lastName"**: "фамилия",   **"patronymic"**: "отчество", // необязателен  **"email"**: "адрес",  **"phone"**: "номер",   **"login"**: "логин",  **"password"**: "пароль",  } |

Требования к логину, паролю клиента - те же, что и для администратора.

E-mail должен соответствовать требованиям, предъявляемым к формату e-mail.

Допустимые телефонные номера - сотовые номера любых операторов России. Номер может начинаться как с “8”, так и с “+7”. Наличие в номере знаков “-” (дефис) ошибкой не является, но перед записью в БД все знаки “-” удаляются. Номера телефонов стационарной связи указывать не разрешается.

Для зарегистрировавшегося клиента автоматически выполняется операция “Login” (п.3.4)

Response

|  |  |
| --- | --- |
| cookie | JAVASESSIONID |
| json | {  **"id"**: "идентификационный номер",  **"firstName"**: "имя",   **"lastName"**: "фамилия",   **"patronymic"**: "отчество",   **"email"**: "адрес",  **"phone"**: "номер",   **"userType"**: "client" } |

### 

### 3.4. Login.

POST /api/session

Request

|  |  |
| --- | --- |
| json | {  **"login"**: "логин",   **"password"**: "пароль" } |

Response

|  |  |
| --- | --- |
| json | Тот же, что и для 3.2 или 3.3 соответственно |

### 

### 3.5. Logout.

DELETE /api/session

Request

|  |  |
| --- | --- |
| cookie | JAVASESSIONID |

### 

Response

|  |  |
| --- | --- |
| json | **{}** |

### 

### 3.6. Получение информации о текущем пользователе

GET /api/account

Request

|  |  |
| --- | --- |
| cookie | JAVASESSIONID |

### 

Response

Для администратора

|  |  |
| --- | --- |
| json | **{**  **"id"**: "идентификационный номер",   **"firstName"**: "имя",   **"lastName"**: "фамилия",   **"patronymic"**: "отчество",   **"position"**: "должность",   **"userType"**: "admin"  **}** |

Для клиента

|  |  |
| --- | --- |
| json | **{**  **"id"**: "идентификационный номер",   **"firstName"**: "имя",   **"lastName"**: "фамилия",   **"patronymic"**: "отчество",   **"email"**: "адрес",  **"phone"**: "номер",   **"userType"**: "client"  **}** |

### 

### 

### 3.7 Администратор - получение информации о клиентах

GET /api/clients

Request

|  |  |
| --- | --- |
| cookie | JAVASESSIONID |

### 

Response

|  |  |
| --- | --- |
| json | **[**  **{**  **"id"**: "идентификационный номер",  **"firstName"**: "имя",   **"lastName"**: "фамилия",   **"patronymic"**: "отчество",   **"email"**: "адрес",  **"phone"**: "номер",   **"userType"**: "client"  **}**  **]** |

### 3.8. Редактирование профиля администратора

PUT /api/admin

Request

|  |  |
| --- | --- |
| cookie | JAVASESSIONID |

### 

|  |  |
| --- | --- |
| json | {  **"firstName"**: "имя",   **"lastName"**: "фамилия",   **"patronymic"**: "отчество",   **"position"**: "должность",   **"oldPassword"**: "прежний пароль",  **"newPassword"**: "новый пароль"  } |

Response

|  |  |
| --- | --- |
| json | **{**  **"firstName"**: "имя",   **"lastName"**: "фамилия",   **"patronymic"**: "отчество",   **"position"**: "должность",   **"userType"**: "admin" **}** |

### 

### 3.9. Редактирование профиля клиента

PUT /api/client

Request

|  |  |
| --- | --- |
| cookie | JAVASESSIONID |

### 

|  |  |
| --- | --- |
| json | {  **"firstName"**: "имя",   **"lastName"**: "фамилия",   **"patronymic"**: "отчество",   **"email"**: "адрес",  **"phone"**: "номер",   **"oldPassword"**: "прежний пароль",  **"newPassword"**: "новый пароль"  } } |

Response

|  |  |
| --- | --- |
| json | **{**  **"firstName"**: "имя",   **"lastName"**: "фамилия",   **"patronymic"**: "отчество",   **"email"**: "адрес",  **"phone"**: "номер",   **"userType"**: "client" **}** |

### 

### 

### 3.10 Администратор - получение информации о типах самолетов

GET /api/planes

Request

|  |  |
| --- | --- |
| cookie | JAVASESSIONID |

### 

Response

|  |  |
| --- | --- |
| json | **[**   **{**  **"name"**: "название самолета",  **"businessRows"**: количество рядов в бизнес-классе,  **"economyRows"**: количество рядов в эконом-классе,  **"placesInBusinessRow"**: количество мест в ряду бизнес-класса,  **"placesInEconomyRow"**: количество мест в ряду эконом-класса  **},**  **…**  **{**  **"name"**: "название самолета",  **"businessRows"**: количество рядов в бизнес-классе,  **"economyRows"**: количество рядов в эконом-классе  **"placesInBusinessRow"**: количество мест в ряду бизнес-класса,  **"placesInEconomyRow"**: количество мест в ряду эконом-класса  **}  ]** |

### 3.11. Администратор - добавление рейса.

POST /api/flights

Request

|  |  |
| --- | --- |
| cookie | JAVASESSIONID |
| json | {  **"flightName"**: "название рейса",   **"planeName"**: "название самолета",   **"fromTown"**: "город вылета",   **"toTown"**: "город прилета",  **"start"**: "время вылета",  **"duration"**: "время в пути",  **"priceBusiness"**: цена билета в бизнес-классе,  **"priceEconomy"**: цена билета в эконом-классе,  **"schedule"**: **{**  **"fromDate"**: "дата начала выполнения маршрута",  **"toDate"**: "дата окончания выполнения маршрута", // включительно  **"period"**: "дни вылета",  **}**  **"dates"**: **[**  **"день1”, “день2”,..”деньN" //** дни выполнения маршрута",  **]**  **}** |

Время вылета и время в пути задаются в формате “HH:MM”. Время вылета и время в пути одно и то же в течение всего периода полетов.

Все названия рейсов должны быть уникальные

В запросе должен присутствовать либо раздел schedule, либо раздел dates, но не оба вместе.

Если имеется раздел “schedule” - задается расписание полетов. Указываются дата начала рейса, дата окончания и дни вылета

Дни вылета (period) могут задаваться в одном из следующих форматов (регистр букв не учитывается, кроме названий дней недели)

* “daily” - рейс выполняется ежедневно
* “odd” - рейс выполняется по нечетным дням месяца. Если два нечетных дня следуют друг за другом (например, 31 марта и 1 апреля), рейс выполняется в оба эти дня
* “even” - рейс выполняется по четным дням месяца.
* “день-недели-1, день-недели-2… , день-неделиN” - рейсы выполняются по дням недели, указанным в списке. Для дней недели используются обозначения Sun,Mon,Tue,Wed,Thu,Fri,Sat. Порядок элементов в списке произвольный. Например, если указано “Sun,Tue,Sat” - это значит, что рейс выполняется по воскресеньям, вторникам и субботам”
* “день1”, “день2”, … “деньN” - рейсы выполняются в указанные дни каждого месяца. Если в каком-то месяце дней меньше, чем максимальный номер дня в этом списке, лишние для этого месяца дни игнорируются. Нумерация дней начинается с 1.Например, если указано “4, 12, 30, 31” , то рейсы будут выполняться 4, 12 февраля, 4, 12, 30, 31 марта и 4, 12, 30 апреля

Если имеется раздел “dates” - указывается список дат, по которым выполняется рейс.

Даты задаются в формате “YYYY-MM-DD”.

Response

|  |  |
| --- | --- |
| json | **{**  **"flightId"**: номер рейса,  **"flightName"**: "название рейса",   **"fromTown"**: "город вылета",   **"toTown"**: "город прилета",  **"start"**: "время вылета",  **"duration"**: "время в пути",  **"priceBusiness"**: цена билета в бизнес-классе,  **"priceEconomy"**: цена билета в эконом-классе,  **"plane": {**  **"name"**: "название самолета",  **"businessRows"**: количество рядов в бизнес-классе,  **"economyRows"**: количество рядов в эконом-классе  **"placesInBusinessRow"**: количество мест в ряду бизнес-класса,  **"placesInEconomyRow"**: количество мест в ряду эконом-класса  **},**  **"approved"**: false,  **"schedule"**: **{**  **"fromDate"**: "дата начала выполнения маршрута",  **"toDate"**: "дата окончания выполнения маршрута", // включительно  **"period"**: "дни вылета",  **}**  **"dates"**: **[**  **"день1”, день2”, ..”деньN"** " //дни выполнения маршрута",  **]**  **}** |

Если в Request был задан раздел “schedule”, в Response должен присутствовать как раздел “schedule”, так и раздел “date” (созданный на сервере по этому “schedule”). Если в Request был задан раздел “dates”, раздел “schedule” в Response должен отсутствовать.

Примечание. В серверной части программы не предусмотрен контроль по дубликатам рейсов. Например, можно ввести два рейса с одним и теми же всеми или некоторыми характеристиками.

### 

### 3.12. Администратор - изменение рейса.

PUT /api/flights/номер\_рейса

Request

|  |  |
| --- | --- |
| cookie | JAVASESSIONID |
| json | {  **"flightName"**: "название рейса",   **"planeName"**: "название самолета",   **"fromTown"**: "город вылета",   **"toTown"**: "город прилета",  **"start"**: "время вылета",  **"duration"**: "время в пути",  **"priceBusiness"**: цена билета в бизнес-классе,  **"priceEconomy"**: цена билета в эконом-классе,  **"schedule"**: **{**  **"fromDate"**: "дата начала выполнения маршрута",  **"toDate"**: "дата окончания выполнения маршрута", // включительно  **"period"**: "дни вылета",  **}**  **"dates"**: **[**  **"день1”, “день2”, ..”деньN"**: "дни выполнения маршрута",  **]**  **}** |

Требования к параметрам запроса те же, что и для 3.10.

Response

|  |  |
| --- | --- |
| json | **{**  **"flightId"**: номер рейса,  **"flightName"**: "название рейса",   **“plane” :{**  **"name"**: "название самолета",  **"businessRows"**: количество рядов в бизнес-классе,  **"economyRows"**: количество рядов в эконом-классе,  **"placesInBusinessRow"**: количество мест в ряду бизнес-класса,  **"placesInRowEconomy"**: количество мест в ряду эконом-класса  **},**  **"approved"**: false,  **"fromTown"**: "город вылета",   **"toTown"**: "город прилета",  **"start"**: "время вылета",  **"duration"**: "время в пути",  **"priceBusiness"**: цена билета в бизнес-классе,  **"priceEconomy"**: цена билета в эконом-классе,  **"schedule"**: **{**  **"fromDate"**: "дата начала выполнения маршрута",  **"toDate"**: "дата окончания выполнения маршрута", // включительно  **"period"**: "дни вылета",  **}**  **"dates"**: **[**  **"день1”, “день2”,..”деньN"**: "дни выполнения маршрута",  **]**  **}** |

Пояснение к Response - см.3.10

### 3.13. Администратор - удаление рейса.

DELETE /api/flights/номер\_рейса

Request

|  |  |
| --- | --- |
| cookie | JAVASESSIONID |

### 

Response

|  |  |
| --- | --- |
| json | **{}** |

### 3.14. Администратор - получение информации о рейсе.

GET /api/flights/номер\_рейса

Request

|  |  |
| --- | --- |
| cookie | JAVASESSIONID |

Response

|  |  |
| --- | --- |
| json | **{**  **"flightId"**: номер рейса,  **"flightName"**: "название рейса",    **“plane” :{**  **"name"**: "название самолета",  **"businessRows"**: количество рядов в бизнес-классе,  **"economyRows"**: количество рядов в эконом-классе,  **"placesInBusinessRow"**: количество мест в ряду бизнес-класса,  **"placesInEconomyRow"**: количество мест в ряду эконом-класса  **},**  **"fromTown"**: "город вылета",   **"toTown"**: "город прилета",  **"start"**: "время вылета",  **"duration"**: "время в пути",  **"priceBusiness"**: цена билета в бизнес-классе,  **"priceEconomy"**: цена билета в эконом-классе,  **"approved"**: false | true, // утвержден ли  **"schedule"**: **{**  **"fromDate"**: "дата начала выполнения маршрута",  **"toDate"**: "дата окончания выполнения маршрута", // включительно  **"period"**: "дни вылета",  **}**  **"dates"**: **[**  **"день1”, “день2”,..”деньN" //** дни выполнения маршрута",  **]**  **}** |
|  |  |

Если при последнем по времени изменении рейса (или при создании, если изменений не было) в Request был задан раздел “schedule”, в Response должен присутствовать как раздел “schedule”, так и раздел “dates” (созданный на сервере по этому “schedule”). Если в Request был задан раздел “dates”, раздел “schedule” в Response должен отсутствовать.

### 3.15. Администратор - утверждение рейса.

PUT /api/flights/номер\_рейса/approve

Request

|  |  |
| --- | --- |
| cookie | JAVASESSIONID |

Response

|  |  |
| --- | --- |
| json | **{**  **"flightId"**: номер рейса,  **"flightName"**: "название рейса",    **“plane” :{**  **"name"**: "название самолета",  **"businessRows"**: количество рядов в бизнес-классе,  **"economyRows"**: количество рядов в эконом-классе,  **"placesInBusinessRow"**: количество мест в ряду бизнес-класса,  **"placesInEconomyRow"**: количество мест в ряду эконом-класса  **},**  **"fromTown"**: "город вылета",   **"toTown"**: "город прилета",  **"start"**: "время вылета",  **"duration"**: "время в пути",  **"priceBusiness"**: цена билета в бизнес-классе,  **"priceEconomy"**: цена билета в эконом-классе,  **"approved"**: true,  **"schedule"**: **{**  **"fromDate"**: "дата начала выполнения маршрута",  **"toDate"**: "дата окончания выполнения маршрута", // включительно  **"period"**: "дни вылета",  **}**  **"dates"**: **[**  **"день1”, “день2”,..”деньN"** // дни выполнения маршрута",  **]**  **}** |

Пояснение к Response - см.3.13.

### 3.16. Администратор и клиент - получение списка рейсов.

GET /api/flights

Request

|  |  |
| --- | --- |
| cookie | JAVASESSIONID |
| QueryParam | **fromTown =** “город” |
| QueryParam | **toTown =** “город” |
| QueryParam | **flightName =** “название рейса” |
| QueryParam | **planeName =** “название самолета” |
| QueryParam | **fromDate =** "дата начала" |
| QueryParam | **toDate =** "дата окончания" |

Все параметры необязательные.

Если присутствует “fromTown”, но нет “toTown” - выдаются только рейсы, вылетающие из fromTown.

Если присутствует “toTown”, но нет “fromTown” - выдаются только рейсы, прилетающие в toTown.

Если присутствуют одновременно “fromTown” и “toTown” - выдаются только рейсы, летающие между этими городами

Если присутствует “flightName” - выдается только рейсы с указанным названием.

Если присутствует “planeName” - выдаются только рейсы, выполняемые на данном типе самолета.

Если присутствует “fromDate” - выдаются только рейсы, выполняемые, начиная с указанной даты.

Если присутствует “toDate” - выдаются только рейсы, выполняемые до указанной даты (включительно).

Response

|  |  |
| --- | --- |
| json | **[**  **{**  **"flightId"**: номер рейса,  **"flightName"**: "название рейса",    **“plane” :{**  **"name"**: "название самолета",  **"businessRows"**: количество рядов в бизнес-классе,  **"economyRows"**: количество рядов в эконом-классе,  **"placesInBusinessRow"**: количество мест в ряду бизнес-класса,  **"placesInEconomyRow"**: количество мест в ряду эконом-класса  **},**  **"fromTown"**: "город вылета",   **"toTown"**: "город прилета",  **"start"**: "время вылета",  **"priceBusiness"**: цена билета в бизнес-классе,  **"priceEconomy"**: цена билета в эконом-классе,  **"duration"**: "время в пути",  **"approved"**: false | true, // утвержден ли  **"schedule"**: **{**  **"fromDate"**: "дата начала выполнения маршрута",  **"toDate"**: "дата окончания выполнения маршрута", // включительно  **"period"**: "дни вылета",  **},**  **"dates"**: **[**  **"день1”, “день2”,..”деньN"** " // дни выполнения маршрута",  **],**  **}**  **...**    **]** |

Пояснение к Response - см.3.13.

Разделы approved и plane выдаются только для при запросе администратора. При запросе клиента неутвержденные рейсы не выдаются.

### 

### 

### 

### 

### 

### 3.17. Получение списка стран (гражданство)

GET /api/countries

Request

|  |  |
| --- | --- |
| cookie | JAVASESSIONID |

Response

|  |  |
| --- | --- |
| json | **[**  **{**  **"iso3166"**: двухбуквенный код страны, ISO 3166-1 Alpha2,  **"name"**: название страны  **}**  **...**  **]** |

Коды ISO-3166 см. <https://ru.wikipedia.org/wiki/ISO_3166-1>.

Названия стран приводятся на русском языке.

### 

### 

### 

### 3.18. Клиент - заказ билетов.

POST /api/orders

Request

|  |  |
| --- | --- |
| cookie | JAVASESSIONID |
| json | **{**  **"flightId"**: номер рейса,   **"date"**: "дата",   **"passengers"**: **[**  **{**  **"firstName"**: "имя",  **"lastName"**: "фамилия",  **"nationality"**: "гражданство",  **"passport"**: "номер паспорта"  **"class"**: "BUSINESS" | “ECONOMY”,  **},**  **...**  **]**  **}** |

Response

|  |  |
| --- | --- |
| json | **{**  **"orderId"**: номер заказа,  **"flightId"**: "номер рейса",   **"fromTown"**: "город вылета",   **"toTown"**: "город прилета",  **"flightName"**: "название рейса",   **"planeName"**: "название самолета",   **"date"**: "дата полета",  **"start"**: "время вылета",  **"duration"**: "время в пути",  **"passengers"**: **[**  **{**  **"ticket"**: "номер билета",  **"firstName"**: "имя",  **"lastName"**: "фамилия",  **"nationality"**: "гражданство",  **"passport"**: "номер паспорта",  **"class"**: "BUSINESS" | “ECONOMY”,  **"price"**: цена билета  **},**  **...**  **],**  **"totalPrice"**: цена заказа  **}** |

Номер паспорта, имя и фамилия не могут быть пустыми, других требований к ним не предъявляется.

Гражданство в Request указывается кодом ISO-3166, в Response возвращается в виде пары “ISO-3166” и названия страны , объединенных с помощью знака “тире” (минус)

В серверном коде не предусмотрены никакие проверки содержимого заказов. В частности, клиент может указать в заказе одного и того же пассажира несколько раз или даже сделать один и тот же заказ несколько раз.

### 

### 3.19. Администратор и клиент - получение списка заказов.

GET /api/orders

Request

|  |  |
| --- | --- |
| cookie | JAVASESSIONID |
| QueryParam | **fromTown =** “город” |
| QueryParam | **toTown =** “город” |
| QueryParam | **flightName =** “название рейса” |
| QueryParam | **planeName =** “название самолета” |
| QueryParam | **fromDate =** "дата начала" |
| QueryParam | **toDate =** "дата окончания" |
| QueryParam | **clientId =** "номер клиента" |

Все параметры необязательные.

Если присутствует “fromTown”, но нет “toTown” - выдаются только заказы на рейсы, вылетающие из fromTown.

Если присутствует “toTown”, но нет “fromTown” - выдаются только заказы на рейсы, прилетающие в toTown.

Если присутствуют одновременно “fromTown” и “toTown” - выдаются только заказы на рейсы, летающие между этими городами

Если присутствует “flightName” - выдается только заказы на рейсы с указанным названием.

Если присутствует “planeName” - выдаются только заказы на рейсы, выполняемые на данном типе самолета.

Если присутствует “fromDate” - выдаются только заказы на рейсы, выполняемые, начиная с указанной даты.

Если присутствует “toDate” - выдаются только заказы на рейсы, выполняемые до указанной даты (включительно).

Если присутствует “clientId” - выдаются только заказы на рейсы, сделанные данным клиентом. Этот параметр имеет смысл только если запрос делается администратором. Для клиента этот параметр игнорируется, клиент может получить только свои собственные заказы

Response

|  |  |
| --- | --- |
| json | **[**  **{**  **"orderId"**: номер заказа,  **"flightId"**: номер рейса,   **"fromTown"**: "город вылета",   **"toTown"**: "город прилета",  **"flightName"**: "название рейса",   **"planeName"**: "название самолета",   **"date"**: "дата полета",  **"start"**: "время вылета",  **"duration"**: "время в пути",  **"passengers"**: **[**  **{**  **"ticket"**: "номер билета",  **"firstName"**: "имя",  **"lastName"**: "фамилия",  **"nationality"**: "гражданство",  **"passport"**: "номер паспорта",  **"class"**: "BUSINESS" | “ECONOMY”,  **"price"**: цена билета  **},**  **...**  **],**  **"totalPrice"**: цена заказа  **}**  **…**  **]** |

### 

### 

### 3.20. Клиент - получение списка свободных мест для регистрации.

GET /api/places/номер\_заказа

Request

|  |  |
| --- | --- |
| cookie | JAVASESSIONID |

Response

|  |  |
| --- | --- |
| json | **[**  **место1,место2.. местоN**  **]** |

### Места в самолете указываются следующим образом : вначале номер ряда, затем место в ряде. Ряды нумеруются, начиная с 1, места - начиная с буквы A (латинская). Между номером ряда и номером места ничего не ставится.

### Например

### 1A - место A в 1 ряду

12B - место B в 12 ряду

Ряды с 1 по businessRows для этого типа самолета принадлежат бизнес-классу, ряды с businessRows + 1 и до последнего - экономическому классу. Если businessRows == 0, места в экономическом классе нумеруются с 1.

### 3.21. Клиент - регистрация пассажира на рейс.

POST /api/places

Request

|  |  |
| --- | --- |
| cookie | JAVASESSIONID |
| json | **{**  **"orderId"**: номер заказа,  **"ticket"**: "номер билета",  **"lastName"**: "фамилия",  **"firstName"**: "имя",  **"place"**: "место в самолете"  **}** |

Response

|  |  |
| --- | --- |
| json | **{**  **"orderId"**: номер заказа,  **"ticket"**: "номер билета",  **"lastName"**: "фамилия",  **"firstName"**: "имя",  **"place"**: "место в самолете"  **}** |

### Формат места - см. 3.20

### 3.22. Получение настроек сервера

GET /api/settings

Request

|  |  |
| --- | --- |
| cookie | JAVASESSIONID |

Параметр “cookie” для этого запроса не является обязательным. Если он передается, то для администратора выдаются доступные ему настройки, а для клиента - доступные ему. Если cookie не передается в запросе, возвращается список настроек, доступных до выполнения операции “Login”

Response

В настоящее время для всех 3 случаев выдается один и тот же результат. Это поведение может быть в дальнейшем изменено.

|  |  |
| --- | --- |
| json | **{**  **"maxNameLength"**: значение ***max\_name\_length,***  **"minPasswordLength"**: значение ***min\_password\_length,*** **}** |

### 3.23. Очистка базы данных.

POST /api/debug/clear

Request

Отсутствует

Response

|  |  |
| --- | --- |
| json | {} |

### 

Удаляет все записи в БД. Метод предназначен для отладки, в production должен быть отключен.

### 

## Приложение 1. Конфигурационный файл сервера.

Конфигурационный файл используется для задания параметров работы сервера и содержит элементы вида “параметр = значение”. Каждый элемент находится в отдельной строке файла (таблица 1). Порядок следования элементов в файле произвольный. Все параметры обязательные. Формат данных в конфигурационном файле всегда правильный и не проверяется, поэтому установка недопустимых по типу или величине значений может привести к неопределенному поведению.

Во время работы сервера изменение параметров конфигурационного файла на работу сервера не влияет. Если значения параметров были изменены, необходимо перезапустить сервер для того, чтобы новые значения вступили в действие,.

Имя конфигурационного файла передается через командную строку при запуске сервера. В случае, если командная строка не содержит имени конфигурационного файла, или файл не найден, или не содержит хотя бы одного параметра, сервер выводит сообщение в лог и завершает свою работу.

### Таблица 1. Параметры конфигурационного файла

|  |  |  |
| --- | --- | --- |
| Параметр | Тип | Назначение |
| rest\_http\_port | int | Порт, на котором работает REST-сервер. |
| max\_name\_length | int | Максимальная длина имени, логина и пароля. |
| min\_password\_length | int | Минимальная длина пароля |

Пример конфигурационного файла

rest\_http\_port = 8888

max\_name\_length = 50

min\_password\_length = 8

1. Реализация отсылки писем и смс в сервере в настоящее время не требуется [↑](#footnote-ref-0)
2. Здесь и далее слова, выделенные жирным курсивом, обозначают параметры конфигурационного файла (Приложение 1) [↑](#footnote-ref-1)