Билеты на автобусы

Билеты на автобусы
- 0. Изменения.
- 1. Введение
- 2. Описание системы.
  - 2.1. Администраторы и клиенты.
  - 2.2. Автобусы и рейсы
  - 2.3. Заказ билетов
- 3. REST-интерфейс
  - 3.1. Общие правила
  - 3.2. Регистрация администратора.
  - 3.3. Регистрация клиента.
  - 3.4. Login.
  - 3.5. Logout.
  - 3.6. Уход пользователя.
  - 3.7. Получение информации о текущем пользователе
  - 3.8. Администратор - получение информации о клиентах
  - 3.9. Редактирование профиля администратора
  - 3.10. Редактирование профиля клиента
  - 3.11 Администратор - получение информации о марках автобусов
  - 3.12. Администратор - добавление рейса.
  - 3.13. Администратор - изменение рейса.

3.14. Администратор - удаление рейса.
3.15. Администратор - получение информации о рейсе.
3.16. Администратор - утверждение рейса.
3.17. Администратор и клиент - получение списка рейсов.
3.18. Клиент - заказ билетов.
3.19. Администратор и клиент - получение списка заказов.
3.20. Клиент - получение списка свободных мест.
3.21. Клиент - выбор места .
3.22. Клиент - отказ от заказа.
3.23. Получение настроек сервера
3.24. Очистка базы данных.
Приложение 1. Конфигурационный файл сервера.
Таблица 1. Параметры конфигурационного файла

0. Изменения.

№	Дата	Содержание

1. Введение

Автобусной компании, занимающейся междугородными перевозками пассажирова, требуется сайт, на котором пассажиры могли бы заказывать билеты на рейсы.

Все рейсы выполняются из начального пункта до конечного пункта, без промежуточных остановок. Автобусная компания не выполняет рейсы с пересадками. Каждому пассажиру выделяется свое место, проезд пассажиров “стоя” не предусмотрен.

Администраторы компании должны иметь возможность добавлять и удалять рейсы, определять цену билетов и т.п. Клиенты должны иметь возможность покупать билеты.

Вам предстоит написать серверную часть приложения.

Руководство компании 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.

Для того, чтобы иметь возможность выполнять действия, пользователь (администратор или клиент) должен зарегистрироваться в системе, выбрав уникальное имя (логин) и указав пароль. Если регистрация прошла успешно, для этого пользователя автоматически выполняется операция входа на сервер (login). После выполнения операции “login” считается, что пользователь создал сеанс работы с сервером, и такой пользователь называется активным пользователем. Активный пользователь может выйти из системы, выполнив операцию logout. Если активный пользователь не выполняет никаких действий в течение ***user_idle_timeout***[1] секунд**,** операция “logout” выполняется для него автоматически системой. Пользователь, выполнивший операцию “logout”, может в дальнейшем вновь выполнить операцию “login”.

---

[1] Здесь и далее слова, выделенные жирным курсивом, обозначают параметры конфигурационного файла (Приложение 1)

Пользователь может изменить свои регистрационные данные, но не может изменить логин. Если пользователь изменяет пароль, его сеанс не прекращается, но для следующего входа он должен указать новый пароль.

Пользователь может иметь только один сеанс работы с сервером. Если пользователь пытается войти на сервер , не выполнив операцию “logout”, для него создается новый сеанс, а старый автоматически прекращается.

Пользователь, прошедший регистрацию, может со временем покинуть сервер. Если пользователь покидает сервер, для него автоматически выполняется операция “logout”, и в дальнейшем он не может выполнять операцию “login”. Новый пользователь не может в качестве своего имени выбрать имя пользователя, покинувшего сервер. Если пользователь является администратором, то он может покинуть сервер только если на сервере после его ухода останется хотя бы один администратор.

2.2. Автобусы и рейсы

Компания располагает парком автобусов, который не изменяется. Марки автобусов задаются в таблице БД , которая не подлежит редактированию средствами данного сайта. Названия всех марок автобусов уникальные. Для каждой марки автобуса в этой таблице указывается его название и количество мест . Количество автобусов каждой марки в таблице не присутствует, так как предполагается, что компания может арендовать дополнительный автобус 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 автобуса 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.

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[2].

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

[2] Реализация отсылки писем и смс в сервере в настоящее время не требуется

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/admins

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*** 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	<pre>{
	  "id": идентификационный номер,
	  "firstName": "имя",
	  "lastName": "фамилия",
	  "patronymic": "отчество",
	  "position": "должность",
	  "userType": "admin"
	}</pre>

3.3. Регистрация клиента.

POST /api/clients

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/sessions

Request

json	{
	"login": "логин",
	"password": "пароль"
	}

Response

Тот же, что и для 3.2 или 3.3 соответственно

3.5. Logout.

DELETE /api/sessions

Request

cookie	JAVASESSIONID

Response

json	{}

3.6. Уход пользователя.

`DELETE /api/accounts`

Request

cookie	JAVASESSIONID

Response

json	{}

3.7. Получение информации о текущем пользователе

`GET /api/accounts`

Request

cookie	JAVASESSIONID

Response

Для администратора

json	<pre>{
	"Id": идентификационный номер,
	"firstName": "имя",
	"lastName": "фамилия",
	"patronymic": "отчество",
	"position": "должность",
	"userType": "admin"
	}</pre>

Для клиента

json	<pre>{
	"id": идентификационный номер,</pre>

	<pre>"firstName": "имя",
	"lastName": "фамилия",
	"patronymic": "отчество",
	"email": "адрес",
	"phone": "номер",
	"userType": "client"
	}</pre>

3.8. Администратор - получение информации о клиентах

GET /api/clients

Request

cookie	JAVASESSIONID

Response

json	<pre>[
	{
	"id": идентификационный номер,
	"firstName": "имя",
	"lastName": "фамилия",
	"patronymic": "отчество",
	"email": "адрес",
	"phone": "номер",
	"userType": "client"
	}
	]</pre>

3.9. Редактирование профиля администратора

PUT /api/admins

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"
	}

3.10. Редактирование профиля клиента

PUT /api/clients

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" }

3.11 Администратор - получение информации о марках автобусов

GET /api/buses

Request

cookie	JAVASESSIONID

Response

json	<pre>[
	  {
	  "busName": "марка автобуса",
	  "placeCount": количество мест
	  },
	  …
	  {
	  "busName": "марка автобуса",
	  "placeCount": количество мест
	
	  }
	]</pre>

3.12. Администратор - добавление рейса.

POST /api/trips

Request

cookie	JAVASESSIONID

json	<pre>{
	  "busName": "марка автобуса",
	  "fromStation": "начальная станция",
	  "toStation": "конечная станция",
	  "start": "время отправления",
	  "duration": "время в пути",
	  "price": цена билета,
	  "schedule": {
	  "fromDate": "дата начала выполнения маршрута",
	  "toDate": "дата окончания выполнения маршрута", // включительно
	  "period": "дни рейсов",
	  }
	  "dates": [
	  "день1", "день2",.."деньN" // дни выполнения маршрута",
	  ]
	}</pre>

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

В запросе должен присутствовать либо раздел schedule, либо раздел dates, но не оба вместе.
Если имеется раздел “schedule” - задается расписание рейсов. Указываются дата начала рейса, дата окончания и дни выезда
Дни рейсов (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”.
Цена билета указывается в рублях.
В серверной части программы не предусмотрен контроль по дубликатам рейсов. Например, можно ввести два рейса с одними и теми же всеми или некоторыми характеристиками.

Response

json	{ "tripId": номер рейса, "fromStation": "начальная станция", "toStation": "конечная станция", "start": "время отправления", "duration": "время в пути",

	"price": цена билета,
	"bus": {
	"busName": "марка автобуса",
	"places": количество мест
	},
	"approved": false,
	"schedule": {
	"fromDate": "дата начала выполнения маршрута",
	"toDate": "дата окончания выполнения маршрута", // включительно
	"period": "дни выезда",
	}
	"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.

3.13. Администратор - изменение рейса.

`PUT /api/trips/номер_рейса`

Request

cookie	JAVASESSIONID

json	<pre>{
	  "busName": "марка автобуса",
	  "fromStation": "начальная станция",
	  "toStation": "конечная станция",
	  "start": "время отправления",
	  "duration": "время в пути",
	  "price": цена билета,
	  "schedule": {
	  "fromDate": "дата начала выполнения маршрута",
	  "toDate": "дата окончания выполнения маршрута", // включительно
	  "period": "дни рейсов",
	  }
	  "dates": [
	  "день1”, “день2”, ..”деньN": "дни выполнения маршрута",
	  ]
	}</pre>

Требования к параметрам запроса те же, что и для 3.12.

Response

Тот же, что и для 3.12

3.14. Администратор - удаление рейса.

DELETE /api/trips/номер_рейса

Request

cookie	JAVASESSIONID

Response

json	{}

3.15. Администратор - получение информации о рейсе.

`GET /api/trips/номер_рейса`

Request

cookie	JAVASESSIONID

Response

Тот же, что и для 3.12

3.16. Администратор - утверждение рейса.

PUT /api/trips/номер_рейса/approve

Request

cookie	JAVASESSIONID

Response

Тот же, что и для 3.12, но поле “approved” имеет значение “true”

3.17. Администратор и клиент - получение списка рейсов.

GET /api/trips

Request

cookie	JAVASESSIONID
RequestParam	**fromStation =** “начальная станция”
RequestParam	**toStation =** “конечная станция”
RequestParam	**busName =** “название марки автобуса”
RequestParam	**fromDate =** "дата начала"
RequestParam	**toDate =** "дата окончания"

Все параметры необязательные.
Если присутствует “fromStation”, но нет “toStation” - выдаются только рейсы, отправляющиеся из fromStation.
Если присутствует “toStation”, но нет “fromStation” - выдаются только рейсы, прибывающие в toStation.
Если присутствуют одновременно “fromStation” и “toStation” - выдаются только рейсы, выполняемые по этому маршруту
Если присутствует “busName” - выдаются только рейсы, выполняемые на данном типе автобуса.
Если присутствует “fromDate” - выдаются только рейсы, выполняемые, начиная с указанной даты.
Если присутствует “toDate” - выдаются только рейсы, выполняемые до указанной даты (включительно).

Response

Массив (список) из структур, описанных в 3.12

Поле approved в каждом элементе выдается только для при запросе администратора. При запросе клиента неутвержденные рейсы не выдаются.

3.18. Клиент - заказ билетов.

POST /api/orders

Request

cookie	JAVASESSIONID

json	<pre>{
	  "tripId": номер рейса,
	  "date": "дата",
	  "passengers": [
	  {
	  "firstName": "имя",
	  "lastName": "фамилия",
	  "passport": "номер паспорта"
	  },
	  ...
	  ]
	}</pre>

Response

json	<pre>{
	  "orderId": номер заказа,
	  "tripId": "номер рейса",
	  "fromStation": "начальная станция",</pre>

	<pre>"toStation": "конечная станция",
	"busName": "марка автобуса",
	"date": "дата поездки",
	"start": "время отправления",
	"duration": "время в пути",
	"price": цена билета,
	"totalPrice": цена заказа,
	"passengers": [
	{
	"firstName": "имя",
	"lastName": "фамилия",
	"passport": "номер паспорта",
	},
	...
	],
	}</pre>

Номер паспорта, имя и фамилия не могут быть пустыми, других требований к ним не предъявляется.
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
RequestParam	**fromStation =** “начальная станция”
RequestParam	**toStation =** “конечная станция”
RequestParam	**busName =** “марка автобуса”
RequestParam	**fromDate =** "дата начала"
RequestParam	**toDate =** "дата окончания"
RequestParam	**clientId =** "номер клиента"

Все параметры необязательные.

Если присутствует “fromStation”, но нет “toStation” - выдаются только заказы на рейсы, выезжающие из fromStation.
Если присутствует “toStation”, но нет “fromStation” - выдаются только заказы на рейсы, приезжающие в toStation.
Если присутствуют одновременно “fromStation” и “toStation” - выдаются только заказы на рейсы по этому маршруту
Если присутствует “busName” - выдаются только заказы на рейсы, выполняемые на данной марке автобуса.
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

Массив (список) из структур, описанных в 3.18

3.20. Клиент - получение списка свободных мест.

`GET /api/places/номер_заказа`

Request

cookie	JAVASESSIONID

Response

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

3.21. Клиент - выбор места .

POST /api/places

Request

cookie	JAVASESSIONID

json	<pre>{
	  "orderId": номер заказа,
	  "lastName": "фамилия",
	  "firstName": "имя",
	  "passport": "номер паспорта",
	  "place": "место в автобусе"
	}</pre>

Response

json	<pre>{
	  "orderId": номер заказа,
	  "ticket": "номер билета",
	  "lastName": "фамилия",
	  "firstName": "имя",
	  "passport": "номер паспорта",
	  "place": "место в автобусе"
	}</pre>

Если для данного пассажира место уже было выбрано ранее, при выполнении этого запроса прежнее место помечается как свободное и выделяется новое место.
Если клиент сделает заказ, но не выберет места для кого-то из своих пассажиров, места будут назначены водителем автобуса при посадке на рейс.

Номер билета формируется следующим образом : Билет рейс_место
где
Билет - слово “Билет”
рейс - номер рейса
место - номер места

Например, “Билет 123_15” (билет на рейс 123, место 15)

3.22. Клиент - отказ от заказа.

DELETE /api/orders/номер_заказа

Request

cookie	JAVASESSIONID

Response

json	{}

Примечание. Если для этого заказа клиент уже выбрал места, они помечаются как свободные.
Отказаться можно только от заказа целиком. Частичный отказ (для некоторых пассажиров) не предумотрен.

3.23. 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,*** **}**

3.24. 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 (application.properties) используется для задания параметров работы сервера и содержит элементы вида “параметр = значение”. Он должен находится в каталоге src\main\resources 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, проверять не требуется. В файле могут быть и другие параметры, например, параметры для настройки Spring.

Параметры конфигурационного файла недоступны пользователям. Во время работы сервера изменение параметров конфигурационного файла на работу сервера не влияет. Для того, чтобы измененные параметры вступили в действие, необходим перезапуск сервера. После перезапуска измененные значения параметров используются только для новых запросов, никакие изменения в БД для ранее выполненных запросов не производятся.

Таблица 1. Параметры конфигурационного файла

Параметр	Тип	Назначение
server.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
user_idle_timeout	int	Время неактивности пользователя (в секундах), по истечении которого его сессия автоматически закрывается.

Пример конфигурационного файла

```
server.port = 8888
max_name_length = 50
min_password_length = 8
user_idle_timeout = 60
```