Описание API проекта WEB\_project.

Сергей Жолудев

2024

Оглавление

[1.1 История изменений документа 3](#_Toc174046173)

[2. Sigin. Процесс аутентификации в системе. 4](#_Toc174046174)

[2.1 Запрос/Ответ 4](#_Toc174046175)

[2.2 Входные параметры 4](#_Toc174046176)

[2.3 Выходные параметры: 5](#_Toc174046177)

[2.4 Проверки 5](#_Toc174046178)

[2.5 Описание интеграции 5](#_Toc174046179)

[3. SigUP. Процесс регистрации в системе. 6](#_Toc174046180)

[3.1 Запрос/ответ. 6](#_Toc174046181)

[3.2 Входные параметры. 7](#_Toc174046182)

[3.3 Выходные параметры 8](#_Toc174046183)

[3.4 Проверки 8](#_Toc174046184)

[3.5 Описание интеграции 8](#_Toc174046185)

[4. Events. 9](#_Toc174046186)

[4.1 Получение списка всех мероприятий 9](#_Toc174046187)

[4.1.1 Запрос/ответ. 9](#_Toc174046188)

[4.1.2 Входные параметры 9](#_Toc174046189)

[4.1.3 Выходные параметры 9](#_Toc174046190)

[4.1.4 Проверки 9](#_Toc174046191)

[4.2 Получение данных конкретного мероприятия 10](#_Toc174046192)

[4.1.1 Запрос/ответ. 10](#_Toc174046193)

[4.2.2 Входные параметры 10](#_Toc174046194)

[4.2.3 Выходные параметры 10](#_Toc174046195)

[4.2.4 Проверки 11](#_Toc174046196)

[4.3 Создание нового мероприятия 11](#_Toc174046197)

[4.3.1 Запрос/ответ 11](#_Toc174046198)

[4.3.2 Входные параметры 12](#_Toc174046199)

[4.3.3 Выходные параметры 12](#_Toc174046200)

[4.3.4 Проверки 13](#_Toc174046201)

[4.4 Обновление существующего мероприятия 13](#_Toc174046202)

[4.4.1 Запрос/ответ 13](#_Toc174046203)

[4.4.2 Входные параметры 13](#_Toc174046204)

[4.4.3 Выходные параметры 14](#_Toc174046205)

[4.4.4 Проверки 14](#_Toc174046206)

[4.5 Удаление существующего мероприятия 14](#_Toc174046207)

[4.1.1 Запрос/ответ. 14](#_Toc174046208)

[4.5.2 Входные параметры 15](#_Toc174046209)

[4.5.3 Выходные параметры 15](#_Toc174046210)

[4.5.4 Проверки 15](#_Toc174046211)

[4.6 Описание интеграции 15](#_Toc174046212)

[5. Comment. Добавление комментария. 16](#_Toc174046213)

[5.1 Запрос/Ответ 16](#_Toc174046214)

[5.2 Входные параметры 16](#_Toc174046215)

[5.3 Выходные параметры 16](#_Toc174046216)

[5.4 Проверки 16](#_Toc174046217)

[5.5 Описание интеграции 17](#_Toc174046218)

1. Основная информация

|  |  |
| --- | --- |
| **Автор** | Сергей Жолудев |
| **Задача** |  |
| **Бизнес постановка** | Это сервис для поиска и регистрации на события в различных городах. |
| **Связанные документы** | <https://xrentv202.atlassian.net/jira/>  Сам проект: <https://github.com/Burchenkov/web_project>  Коды HTTP сообщений:  https://developer.mozilla.org/ru/docs/Web/HTTP/Status |

## 1.1 История изменений документа

|  |  |  |  |
| --- | --- | --- | --- |
| **Версия** | **Дата** | **Автор** | **Описание изменений** |
| 1.0 | 14.07.2024 | Сергей Жолудев | Первая версия документа |
| 1.1 | 21.07.2024 | Сергей Жолудев | Добавлено описание процедуры регистрации signUp, получение списка мероприятий Events |
| 1.2 | 25.07.2024 | Сергей Жолудев | Изменены скрипты для проверки, теперь используется HTTPie |
| 1.3 | 07.08.2024 | Сергей Жолудев | Исправлены выходные данные для signin.  Добавлен API для comment. |
| 1.4 | 08.08.2024 | Сергей Жолудев | Добавлены API для Event: add, update, delete. |
|  |  |  |  |
|  |  |  |  |

# 2. Sigin. Процесс аутентификации в системе.

На WEB-сайте пользователь вводит пароль и логин, данные передаются на Backend. Базовый код проверяет есть ли пользователь в системе, правильный ли пароль указан. Если результат проверки успешен, пользователь входит в систему. Если введён неправильный логин – пользователь получает сообщение: «Нет такого пользователя, сначала зарегистрируйтесь.» Если введён неправильный логин, сообщение: «Неправильный пароль».

Для тестирование используется утилиты HHTPie.

## 2.1 Запрос/Ответ

|  |  |
| --- | --- |
| **Запрос** | **Ответ** |
| http POST :8000/signin/ login="Someone3" password="12345" | Response code: 200, 401  Response body:  For 200: “HTTP\_200\_OK”  For 400: “HTTP\_401\_UNAUTHORIZED”  body of message:  “Login does not exist. Please SingUP at first”  Or  “The password is incorrect” |

## 2.2 Входные параметры

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№** | **Параметр** | **Тип данных** | **Обязательность** | **Описание** | **Варианты значений** |
| 1 | Login | Str | Да | Логин пользователя | MasterOfTheUniverse |
| 2 | Passwrod | Str | Да | Пароль для входа в систему. | SXM8V6WLR |

## 2.3 Выходные параметры:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Параметр** | **Тип данных** | **Описание** | **Варианты значений** |
| 1 | HTTP response | Str | Ответ системы об успешности аутентфификации | HTTP\_200\_OK |
| 2 | HTTP response | Str | Ответ системы об неуспешности аутентфификации, с уточнением что конкретно не так. | HTTP\_401\_UNAUTHORIZED  + message:  “Login does not exist. Please SingUP at first” or  "The password is incorrect" |

## 2.4 Проверки

|  |  |
| --- | --- |
| **Параметр** | **Проверка** |
| email | * Логин, в нашем случае аутентифицируемся с помощью email, должен быть в формате [xxxxxxx@xxx.xxx](mailto:xxxxxxx@xxx.xxx). * Логин не должен превыщать N символов. * Пароль не должен превышать N симолов. |

## 2.5 Описание интеграции

Изображение выглядит как текст, снимок экрана, диаграмма, число

Автоматически созданное описание

# 3. SigUP. Процесс регистрации в системе.

Если пользователь не зарегистрирован в системе, он нажимает кнопку «Зарегистрироваться». Перед ним появляется модальное окно, куда он вводит данные:

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Данные через POST запрос направляются на сервер backend. Происходит проверка, есть ли такой login, есть ли такой email. Если есть – возвращается сообщение об ошибки с информационным сообщением. В случае уникальности предоставленных данных, добавляем данным аккаунт(экземплят класс User в БД).

## 3.1 Запрос/ответ.

Успешный:

|  |  |
| --- | --- |
| **Запрос** | **Ответ** |
| http POST :8000/signup/ email="someone@example.com" login="MegaUser" role="user" password= "123456" | Response code: 201 “HTTP\_201\_CREATED” |

Не успешный:

|  |  |
| --- | --- |
| **Запрос** | **Ответ** |
| http POST :8000/signup/ email="someone@example.com" login="MegaUser" role="user" password= "123456" | Response code: 400 Bad Request  Response body:  ‘{  “message” : “login is already exist”  }’  Or  ‘{  “message” : “email is already exist”  }’ |

## 3.2 Входные параметры.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Параметр** | **Тип данных** | **Описание** | **Варианты значений** |
| 1 | email | str | Email пользователя | some@example.com |
| 2 | login | str | login пользователя | MasterOfTheUniverse |
| 3 | role | str | роль пользователя в системе. От Frontend всегда user. | user |
| 7 | password | str | пароль пользователя | SXM8V6WLR |

## 3.3 Выходные параметры

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Параметр** | **Тип данных** | **Описание** | **Варианты значений** |
| 1 | HTTP response | Str | Ответ системы об успешности регистрации пользователя | HTTP\_201\_CREATED |
| 3 | HTTP response | Str | Ответ системы об неуспешности регистрации | 400 |
| 4 | Body of response | Str | Ответ системы, что конкретно не так логин или email уже существуют | ‘{  “message” : “login is already exist”  }’ |

## 3.4 Проверки

При регистрации со стороны backend должна выполняться проверка на уникальность email и login. Делаем запрос в БД, если предоставленные данные не уникальны – возвращаем HTTP\_400\_BAD\_REQUEST, в теле сообщения уточнение, что именно неправильно.

## 3.5 Описание интеграции

Изображение выглядит как текст, снимок экрана, диаграмма, Параллельный

Автоматически созданное описание

# 4. Events.

Описание создания, редактирования, удаления и получения списка всех мероприятий.

## 4.1 Получение списка всех мероприятий

### 4.1.1 Запрос/ответ.

|  |  |
| --- | --- |
| **Запрос** | **Ответ** |
| http :8000/events/ | Response code: HTTP\_200\_OK  Response body:  ‘{  "pk":3,  "title":"Box fight",  "category":"Sport",  "date":"2024/01/01 19:00:00",  "location":"Moscow",  "description":"this is a fight of the century"  }’ |

### 4.1.2 Входные параметры

Входных параметров нет, просто отправляем запрос GET на получение списка Events.

### 4.1.3 Выходные параметры

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Параметр** | **Тип данных** | **Описание** | **Варианты значений** |
| 1 | HTTP response | Str | Ответ системы об успешности | 200 |
| 2 | Body of response | JSON | Тело ответа системы | ‘{  "pk":3,  "title":"Box fight",  "category":"Sport","date":"2024-07-30T19:00:00Z",  "location":"Moscow",  "description":"this is a fight of the century"  }’ |

### 4.1.4 Проверки

## 4.2 Получение данных конкретного мероприятия

Для получения данных о конкретном мероприятии надо знать его id(или pk).

### 4.1.1 Запрос/ответ.

|  |  |
| --- | --- |
| **Запрос** | **Ответ** |
| http :8000/event\_change/1 | Response code: HTTP\_200\_OK  Response body:  {  "age\_limit": 6,  "city": "Moscow",  "description": "This is MMA Fight",  "duration": 2,  "event\_name": "MMA Fight",  "image": "c:/image\_MMAFight.jpeg",  "pk": 1,  "price": 0,  "start\_time": "10:00:00",  "user\_limit": 2000  } |

### 4.2.2 Входные параметры

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Параметр** | **Тип данных** | **Описание** | **Варианты значений** |
| 1 | pk | int | id мероприятия в DB | 1 |

### 4.2.3 Выходные параметры

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Параметр** | **Тип данных** | **Описание** | **Варианты значений** |
| 1 | event\_name | str | Название события | MMA Fight |
| 2 | date | str | Дата меропрития | 2024-01-01 |
| 3 | city | str | Город проведения мероприятия | Zelenograd |
| 4 | description | str | Описание мероприятия | This is MMA Fight |
| 5 | price | int | Стоимость входного билета | 0 |
| 6 | image | str | Место хранения картинки мероприятия | c:\image\_MMAFight.jpeg |
| 7 | age | int | Возрастное ограничение | 14 |
| 8 | start\_time | str | Время начала мероприятия | 10:00:00 |
| 9 | user\_limit | int | Максимальное количество мест | 1000 |

### 4.2.4 Проверки

## 4.3 Создание нового мероприятия

### 4.3.1 Запрос/ответ

|  |  |
| --- | --- |
| **Запрос** | **Ответ** |
| http POST :8000/event\_add/ event\_name="MMA Fight" date="2024-01-01" city="Ekaterinburg" description="This is MMA Fight" price="0" image="c:\image\_MMAFight.jpeg" age\_limit="14" start\_time="10:00:00" duration="4" user\_limit="1000" | Response code: HTTP\_201\_CREATED  Response body:  {  "age\_limit": 14,  "city": "Zelenograd",  "description": "This is MMA Fight",  "duration": 4,  "event\_name": "MMA Fight",  "image": "c:\\image\_MMAFight.jpeg",  "pk": 8,  "price": 0,  "start\_time": "10:00:00",  "user\_limit": 1000  }  Response code: HTTP\_400\_BAD\_REQUEST |

### 4.3.2 Входные параметры

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Параметр** | **Тип данных** | **Описание** | **Варианты значений** |
| 1 | event\_name | str | Название события | MMA Fight |
| 2 | date | str | Дата меропрития | 2024-01-01 |
| 3 | city | str | Город проведения мероприятия | Zelenograd |
| 4 | description | str | Описание мероприятия | This is MMA Fight |
| 5 | price | int | Стоимость входного билета | 0 |
| 6 | image | str | Место хранения картинки мероприятия | c:\image\_MMAFight.jpeg |
| 7 | age | int | Возрастное ограничение | 14 |
| 8 | start\_time | str | Время начала мероприятия | 10:00:00 |
| 9 | user\_limit | int | Максимальное количество мест | 1000 |

### 4.3.3 Выходные параметры

В качестве выходных параметров backend возвращает статус выполнения HTTP\_201\_CREATED в случае успешного создания мероприятия и HTTP\_400\_BAD\_REQUEST в случае неуспеха, в этом случае надо смотреть что присылает Frontend.

### 4.3.4 Проверки

## 4.4 Обновление существующего мероприятия

Для обновления существующего мероприятия нужно знать его **pk**. Для создания запросов используется метод **PUT**.

### 4.4.1 Запрос/ответ

|  |  |
| --- | --- |
| **Запрос** | **Ответ** |
| http PUT :8000/event\_change/4 event\_name="Boxing Fight" date="2024-10-10" city="Москва" description="Roy Jones vs Mike Tyson" price="0" image="c:\image\_BOXING.jpeg" age\_limit="8" start\_time="11:00:00" duration="2" user\_limit="1000" | Response code: HTTP\_200\_OK  Response body:  {  "age\_limit": 8,  "city": "Москва",  "description": "Roy Jones vs Mike Tyson",  "duration": 2,  "event\_name": "Boxing Fight",  "image": "c:\\image\_BOXING.jpeg",  "pk": 4,  "price": 0,  "start\_time": "11:00:00",  "user\_limit": 1000  }  Response code: HTTP\_400\_BAD\_REQUEST |

### 4.4.2 Входные параметры

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Параметр** | **Тип данных** | **Описание** | **Варианты значений** |
| 1 | event\_name | str | Название события | MMA Fight |
| 2 | date | str | Дата меропрития | 2024-01-01 |
| 3 | city | str | Город проведения мероприятия | Zelenograd |
| 4 | description | str | Описание мероприятия | This is MMA Fight |
| 5 | price | int | Стоимость входного билета | 0 |
| 6 | image | str | Место хранения картинки мероприятия | c:\image\_MMAFight.jpeg |
| 7 | age | int | Возрастное ограничение | 14 |
| 8 | start\_time | str | Время начала мероприятия | 10:00:00 |
| 9 | user\_limit | int | Максимальное количество мест | 1000 |

### 4.4.3 Выходные параметры

В качестве выходных параметров backend возвращает статус выполнения HTTP\_200\_OK в случае успешного изменения мероприятия и HTTP\_400\_BAD\_REQUEST в случае неуспеха, в этом случае надо смотреть что присылает Frontend.

### 4.4.4 Проверки

## 4.5 Удаление существующего мероприятия

Для удаления существующего мероприятия надо знать его **pk**.

### 4.1.1 Запрос/ответ.

|  |  |
| --- | --- |
| **Запрос** | **Ответ** |
| http DELETE :8000/event\_change/2 | Response code: HTTP\_204\_NO\_CONTENT |

### 4.5.2 Входные параметры

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Параметр** | **Тип данных** | **Описание** | **Варианты значений** |
| 1 | pk | int | id мероприятия в DB | 1 |

### 4.5.3 Выходные параметры

Выходных параметров нет, возвращается только сообщение HTTP.status

### 4.5.4 Проверки

## 4.6 Описание интеграции

Изображение выглядит как текст, снимок экрана, диаграмма, Параллельный

Автоматически созданное описание

# 5. Comment. Добавление комментария.

API для добавления комментария отправляем JSON определённого формата. Для тестирование используется утилиты HHTPie.

## 5.1 Запрос/Ответ

|  |  |
| --- | --- |
| **Запрос** | **Ответ** |
| http :8000/comment text=”This was great!” create\_at=5 | Response code: “HTTP\_201\_CREATED” |

## 5.2 Входные параметры

В качестве входных параметров принимает текст комментария и параметр created\_at.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Параметр** | **Тип данных** | **Описание** | **Варианты значений** |
| 1 | text | str | текст комментария к событию | “That was awasome!” |
| 2 | created\_at | str |  | MasterOfTheUniverse |

## 5.3 Выходные параметры

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Параметр** | **Тип данных** | **Описание** | **Варианты значений** |
| 1 | HTTP response | Str | Ответ системы об успешности | HTTP\_201\_CREATED |

## 5.4 Проверки

## 5.5 Описание интеграции