Описание API проекта WEB\_project.

Сергей Жолудев

2024

Оглавление

[1. Основная информация 3](#_Toc174538191)

[1.1 История изменений документа 3](#_Toc174538192)

[2. User. 4](#_Toc174538193)

[2.1 Sigin. Процесс аутентификации в системе. 4](#_Toc174538194)

[2.1.1 Запрос/Ответ 4](#_Toc174538195)

[2.1.2 Входные параметры 4](#_Toc174538196)

[2.1.3 Выходные параметры: 5](#_Toc174538197)

[2.1.4 Проверки 5](#_Toc174538198)

[2.1.5 Описание интеграции 6](#_Toc174538199)

[2.2 SignUP. Процесс регистрации в системе. 7](#_Toc174538200)

[2.2.1 Запрос/ответ. 7](#_Toc174538201)

[2.2.2 Входные параметры. 8](#_Toc174538202)

[2.2.3 Выходные параметры 9](#_Toc174538203)

[2.2.4 Проверки 9](#_Toc174538204)

[2.2.5 Описание интеграции 9](#_Toc174538205)

[2.3 Получение данных всех пользователей 10](#_Toc174538206)

[2.3.1 Запрос/ответ. 10](#_Toc174538207)

[2.3.2 Входные параметры. 10](#_Toc174538208)

[2.3.3 Выходные параметры. 10](#_Toc174538209)

[2.3.4 Проверки. 11](#_Toc174538210)

[2.4 Получение данных конкретного пользователя 11](#_Toc174538211)

[2.4.1 Запрос/ответ. 11](#_Toc174538212)

[2.4.2 Входные параметры. 11](#_Toc174538213)

[2.4.3 Выходные параметры. 11](#_Toc174538214)

[2.4.4 Проверки. 12](#_Toc174538215)

[2.5 Обновление данных конкретного пользователя 12](#_Toc174538216)

[2.5.1 Запрос/ответ. 12](#_Toc174538217)

[2.5.2 Входные параметры 12](#_Toc174538218)

[2.5.3 Выходные параметры 13](#_Toc174538219)

[2.5.4 Проверки 13](#_Toc174538220)

[4.5 Удаление существующего пользователя 13](#_Toc174538221)

[4.5.1 Запрос/ответ. 13](#_Toc174538222)

[4. Events. 13](#_Toc174538223)

[4.1 Получение списка всех мероприятий 14](#_Toc174538224)

[4.1.1 Запрос/ответ. 14](#_Toc174538225)

[4.1.2 Входные параметры 14](#_Toc174538226)

[4.1.3 Выходные параметры 14](#_Toc174538227)

[4.1.4 Проверки 14](#_Toc174538228)

[4.2 Получение данных конкретного мероприятия 14](#_Toc174538229)

[4.1.1 Запрос/ответ. 15](#_Toc174538230)

[4.2.2 Входные параметры 15](#_Toc174538231)

[4.2.3 Выходные параметры 15](#_Toc174538232)

[4.2.4 Проверки 16](#_Toc174538233)

[4.3 Создание нового мероприятия 16](#_Toc174538234)

[4.3.1 Запрос/ответ 16](#_Toc174538235)

[4.3.2 Входные параметры 17](#_Toc174538236)

[4.3.3 Выходные параметры 17](#_Toc174538237)

[4.3.4 Проверки 18](#_Toc174538238)

[4.4 Обновление существующего мероприятия 18](#_Toc174538239)

[4.4.1 Запрос/ответ 18](#_Toc174538240)

[4.4.2 Входные параметры 18](#_Toc174538241)

[4.4.3 Выходные параметры 19](#_Toc174538242)

[4.4.4 Проверки 19](#_Toc174538243)

[4.5 Удаление существующего мероприятия 19](#_Toc174538244)

[4.1.1 Запрос/ответ. 19](#_Toc174538245)

[4.5.2 Входные параметры 20](#_Toc174538246)

[4.5.3 Выходные параметры 20](#_Toc174538247)

[4.5.4 Проверки 20](#_Toc174538248)

[4.6 Описание интеграции 20](#_Toc174538249)

[5. Comment. Добавление комментария. 21](#_Toc174538250)

[5.1 Запрос/Ответ 21](#_Toc174538251)

[5.2 Входные параметры 21](#_Toc174538252)

[5.3 Выходные параметры 21](#_Toc174538253)

[5.4 Проверки 21](#_Toc174538254)

[5.5 Описание интеграции 22](#_Toc174538255)

# 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 описания API для Event: add, update, delete. |
| 1.5 | 14.08.2024 | Сергей Жолудев | Добавлены описания API для User: GET пользователя по id, update, delete. |
|  |  |  |  |

# 2. User.

## 2.1 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 утилита HHTPie.

### 2.1.1 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” |

### 2.1.2 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 |

### 2.1.3 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" |

### 2.1.4 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. |

### 2.1.5 Описание интеграции

Изображение выглядит как текст, снимок экрана, диаграмма, число

Автоматически созданное описание

## 2.2 SignUP. 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).

### 2.2.1 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”  }’ |

### 2.2.2 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 |

### 2.2.3 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”  }’ |

### 2.2.4 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.

### 2.2.5 Описание интеграции

Изображение выглядит как текст, снимок экрана, диаграмма, Параллельный

Автоматически созданное описание

## 2.3 Получение данных всех пользователей

### 2.3.1 Запрос/ответ.

Для получения списка всех пользователей из которого можно узнать id конкрентного пользователя определен URL …/users/

|  |  |
| --- | --- |
| **Запрос** | **Ответ** |
| http :8000/users/ | HTTP/1.1 200 OK  …  Response body:  {  "avatar": "c:\\avatar\_admin.jpeg",  "email": "admin@example.com",  "login": "admin",  "password": "q1w2e3r4",  **"pk": 2**,  "role": "Admin",  "secret\_phrase": "Restore password"  },  {  "avatar": "c:\\avatar.jpeg",  "email": "USER4@example.com",  "login": "user4",  "password": "q1w2e3r44545",  "pk": 4,  "role": "user",  "secret\_phrase": "Restore password"  }, |

### 2.3.2 Входные параметры.

В данном случае входных параметров не требуется, поскольку просто отправляем GET запрос на URL http :8000/users/.

### 2.3.3 Выходные параметры.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Параметр** | **Тип данных** | **Описание** | **Варианты значений** |
| 1 | HTTP response | Str | Ответ системы об успешности | HTTP/1.1 200 OK |
| 2 | Body of response | JSON | Тело ответа системы | {  "avatar": "c:\avatar\_admin.jpeg",  "email": "admin@example.com",  "login": "admin",  "password": "q1w2e3r4",  "pk": 2,  "role": "Admin",  "secret\_phrase": "secret"  }, |

### 2.3.4 Проверки.

## 2.4 Получение данных конкретного пользователя

Для работы с данными конкретного пользователя надо знать его pk(id) под которым он находится в БД. Процедура получения даных всех пользователей описана выше.

### 2.4.1 Запрос/ответ.

|  |  |
| --- | --- |
| **Запрос** | **Ответ** |
| http PUT :8000/user\_change/4 | HTTP/1.1 200 OK  …  Response body:  {  "avatar": "c:\u0007vatar\_admin.jpeg",  "email": "admin@example.com",  "login": "admin",  "password": "q1w2e3r4",  **"pk": 4**,  "role": "Admin",  "secret\_phrase": "Restore password"  }, |

### 2.4.2 Входные параметры.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Параметр** | **Тип данных** | **Описание** | **Варианты значений** |
| 1 | pk | int | id пользователя под которым он находится в БД | 4 |

### 2.4.3 Выходные параметры.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Параметр** | **Тип данных** | **Описание** | **Варианты значений** |
| 1 | HTTP response | Str | Ответ системы об успешности | HTTP/1.1 200 OK |
| 2 | Body of response | JSON | Тело ответа системы с данными конкретного пользователя | {  "avatar": "c:\\avatar.jpeg",  "email": "USER4@example.com",  "login": "user4",  "password": "q1w2e3r44545",  "pk": 4,  "role": "user",  "secret\_phrase": "Restore password"  } |

### 2.4.4 Проверки.

## 2.5 Обновление данных конкретного пользователя

Для работы с данными конкретного пользователя надо знать его pk(id) под которым он находится в БД. Процедура получения даных всех пользователей описана выше. Теперь когда у нас есть id конкретного пользователя мы можем обновить его данные.

### 2.5.1 Запрос/ответ.

|  |  |
| --- | --- |
| **Запрос** | **Ответ** |
| http PUT :8000/user\_change/4 email="USER4@example.com" login="user4" role="user" password="q1w2e3r455" secret\_phrase="Restore password" avatar = "c:\avatar.jpeg" | HTTP/1.1 200 OK  …  Response body:  {  "avatar": "c:\u0007vatar\_admin.jpeg",  "email": "admin@example.com",  "login": "admin",  "password": "q1w2e3r4",  **"pk": 2**,  "role": "Admin",  "secret\_phrase": "Restore password"  }, |

### 2.5.2 Входные параметры

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Параметр** | **Тип данных** | **Описание** | **Варианты значений** |
| 1 | email | str | email пользователя | USER4@example.com |
| 2 | login | str | логин пользователя | user4 |
| 3 | role | str | роль пользователя, по умолчанию user | user |
| 4 | password | str | пароль пользователя | q1w2e3r44545 |
| 5 | secret\_phrase | str | секретная фраза для восстановления пароля | Restore password" |
| 6 | avatar | str | ссылка на файл с аватар пользователя | c:\avatar.jpeg |

### 2.5.3 Выходные параметры

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

### 2.5.4 Проверки

## 4.5 Удаление существующего пользователя

Для удаления существующего мероприятия надо знать его **pk**. Входных и выходных параметров нет. Указываем только pk(id) пользователя отправляя HTTP DELETE на указанные ниже URL.

### 4.5.1 Запрос/ответ.

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

# 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