FriendWork Recruiter API

Версия 1.9.11

**История изменений**

| **Версия** | **Дата** | **Автор** | **Описание изменений** |
| --- | --- | --- | --- |
| 1.9.11 | 28.10.2022 | [Рэми Никиема](mailto:rn@friendwork.ru) | Добавил раздел “История изменений” |
| 1.9.11 | 03.11.2022 | [Александр Скворцов](mailto:askovrcov@friendwork.ru) | Добавлен раздел “Работа со значениями кастомных полей объектов” |
| 1.9.11 | 31.08.2023 | [Александр Бородин](mailto:alb@friendwork.ru) | Добавлена информация об ограничениях в апи |
| 1.9.12. | 18.10.2023 | [Михаил Проказин](mailto:mpro@friendwork.ru) | Обновили раздел [Webhook](#_o5nx5rmqyw7t) |

# Общие сведения

Практически после каждого действия пользователя во FriendWork Recruiter, которое должно изменить данные, хранящиеся в сервисе, можно отправлять уведомление по заранее заданному http/https-адресу, содержащее необходимые данные в формате JSON или письмо на адрес электронной почты, сформированное на основе нашего шаблонизатора писем. См. Раздел [Webhooks](#_o5nx5rmqyw7t).

API также может быть использовано для получения на карьерном портале пользователя списка публикаций и отправки во FriendWork откликов кандидатов с него или новых кандидатов из других систем, при этом дубликаты кандидатов могут быть автоматически объединены (зависит от параметров методов), а также изменения статусов текущих кандидатов (например, при прохождении испытательного срока или увольнении).

Если возникает ситуация, когда код, работающий с API, не дожидается ответа, то стоит увеличить время ожидания. Рекомендованное время ожидания: не менее 60 секунд.

Если ответ все же не был получен или получено сообщение об ошибке, то стоит послать еще один запрос и так далее, увеличивая паузу между отправками каждый раз в 2 раза (1 минута, 2 минуты, 4 минуты, 8 минут, 16 минут, 32 минуты). Подобное поведение стоит поддерживать на случай сбоя DNS (у любой из сторон) или временных неполадок у провайдера (у любой из сторон). Если подобное поведение не поддерживается, то надежность API не гарантируется.

Предполагается, что доступ к API выдается аккаунтам-администраторам с полным перечнем прав доступа к данным.

Для **API** установлено ограничение, которое не позволяет обслуживать более 2 запросов в секунду. Для последующих запросов в рамках одной секунды - ответ будет содержать код ответа - 429 Too many Requests. Аналогично ограничено число запросов в минуту - не более 200000.

# Авторизация

Для всех запросов в обязательном порядке следует устанавливать следующий HTTP header:

сontent-type = application/json

Используется OAuth 2.0.

Для доступа к API во всех запросах требуется установить следующий HTTP header:

Authorization = “Bearer “ + AccessToken

(после слова “Bearer” должен быть как минимум один пробел).

### Пример

Authorization: Bearer IvmJ9p2zsiTIHT8lzAXZHz\_kM77yxUnFaBoC1O1joosvkH3FJZeGdWdT62d83GC4uS54K0IkoqeIL9c0khyibFFAyf\_Kv3gRTATIm7OZwVDiULAOUQGVRfrq0VFDI-H1nBi5HVpThg5bKj6ar8pnXAKRj2AUve1nDV45MZwidAVGv5bBd0g-fORIEUUyEZzExY2nIrM0mnYJnaJT7U839oDS255g1uR-QXzq2KgF2CR9123F

# 

# Методы API

## Рекомендация по работе с API

В методах ниже есть примеры запросов. Вы можете выполнить предоставленные примеры и у вас они заработают. Как правило вам для вашей логики придется изменить набор и значения полей. Если вы поменяли запрос и он не работает, рекомендуем для более быстрого понимания в чем ошибка - исправлять запрос не сразу к итоговому виду, а итеративно. Например, добавили/изменили одно св-во, отправили запрос. Если все работает, можете добавлять следующее свойство. Если же сломалось, то посмотрите на то, что добавляли, проверьте ваше значение и тип свойства в документации.

## Добавление кандидата/Изменение кандидата

Путь: <http://api.friend.work/Candidate/set>

Тип метода: POST

Для внесения кандидата в базу данных необходимо прислать JSON объект следующего вида:

{

Параметр : Значение,

Параметр : Значение,

…

}

Параметры объекта:

**CandidateId**: тип: int, идентификатор кандидата, если хотите его изменить существующего кандидата, а не создать нового

**FirstName**: тип: string, имя кандидата.

**LastName**: тип: string, фамилия кандидата.

**MiddleName**: тип: string, отчество кандидата.

**Photo**: тип string, фотография кандидата в base64

**City**: тип: string, город проживания кандидата.

**BirthDate**: тип: string в формате yyyy-MM-dd, дата рождения кандидата.

**ExternalLink** : тип: string, содержит полную ссылку на кандидата с работного сайта.

**Contacts**: тип: массив элементов ключ-значение, контакты кандидата, например:

“**Contacts**” : [

{ “**E-mail**” : “somebody@example.com” },

{ “**Phone**” : “+79001002030” },

… (может быть задано до двух email’ов и до двух телефонов)

]  
Замечание. Обратите внимание, что Contacts - это массив, а не объект.

**Source**: тип string, может принимать одно из следующих значений (если у клиента не изменен список источников по умолчанию):

* Карьерный портал FriendWork
* Indeed
* HeadHunter
* SuperJob
* Rabota
* Avito
* Zarplata
* Работа@Mail.Ru
* Job-mo.ruзначение
* Rabota66
* LinkedIn
* Мой круг
* ВКонтакте
* Одноклассники
* Facebook
* Twitter
* Google+
* Instagram
* GitHub
* Angel.co
* ДомКадров
* Work.ua
* Rabota.ua
* Farpost.ru - fp
* AmazingHiring
* Worki.ru
* Корпоративный сайт
* Внутренний кандидат
* Внутренняя рекомендация
* Внешняя рекомендация
* Другой сайт
* Другой источник

**AddWay**: тип string, может принимать одно из следующих значений (если у клиента нет специальных настроек):

* Активный поиск
* Отклик

**ResponsibleId**: тип: int, Id аккаунта пользователя, который будет установлен как ответственный за кандидата и его создатель, а также ответственный за статус “Новый” и его создатель. Значение -1 будет соответствовать аккаунту по умолчанию, но без ответственных за кандидата и статуса “Новый”.

**Sex**: тип: int, может принимать одно из следующих значений: 0 - пол кандидата неизвестен, 1 - женский пол, 2 - мужской пол.

**Age**: тип: int, показывает сколько лет кандидату, 0 - аналогично отсутствию параметра; если есть дата рождения, то значение параметра Age игнорируется.

**Position**: тип: string, желаемая позиция соискателя (название резюме).

**Salary**: тип int, желаемая зарплата. Значение 0 - неизвестно или по договоренности.

**SalaryCurrency**: тип int, валюта желаемой зарплаты. Может принимать одно из следующих значений: 0 - рос. рубль, 1 - доллар, 2 - евро, 3 - гривна, 4 - тенге, 5 - бел. рубль.

**RelocationReadiness**: тип int, готовность к переезду, 0 - не готов/неизвестно, 1 - готов.

**Resume**: тип: string, текст резюме кандидата. Может содержать HTML (картинки должны быть встроены).

**DuplicateProcessing**: тип string, может принимать одно из следующих значений:

* Ignore
  + Если у кандидата нет дубликата, то добавляем его. Возвращаем его идентификатор и сообщение "Candidate was added"
  + Если у кандидата есть дубликат, то не добавляем его. Возвращаем идентификаторы дубликатов и сообщение "Candidate has duplicates"
* (TODO) Replace
  + Если у кандидата нет дубликата, то добавляем его. Возвращаем его идентификатор и сообщение "Candidate was added".
  + Если дубликаты есть, то заменяет первый дубликат данными из нового резюме, но при этом происходит объединение данных.
* Add - добавить дубликат
  + Если дубликатов нет, то создает нового кандидата. Идентификатор кандидата - ID нового кандидата, идентификаторы дубликатов null сообщение "Candidate was added".
  + (TODO) Если дубликаты есть, то заменяет первый дубликат данными из нового резюме, но при этом происходит слияние данных, т.е. данные которые мы не передали будут из старого резюме.
  + Если дубликаты есть, то создает копию кандидата. Идентификатор кандидата - ID нового кандидата, идентификаторы дубликатов - Массив идентификаторов дубликатов. Сообщение "Candidate was added".

Объединяем кандидатов на основе совпадения (если совпадает хоть одно правило ниже):

* номеров телефонов
* адресов электронных почт
* ссылок в соц.сетях

**FileName**: тип: string, название прикрепленного файла.

**FileContent**: файл в формате [Data URI](https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/Data_URIs).

Как конвертировать файл в формат Data URI с помощью File API вы можете узнать из [примера](https://drive.google.com/file/d/1WssH_pSqekfaKTxbusQ9JFtllRSro4b4/view?usp=sharing). (html файл с формой, которая отправляет данные кандидата и прикрепленный файл к нам в сервис, для проверки не забудьте вставить свой token)

**SocialLinks**: тип: объект, социальные сети кандидата. Свойства объекта:

* VK: тип: string, ссылка на ВКонтакте
* Facebook: тип: string, ссылка на Facebook
* LinkedIn: тип: string, ссылка на LinkedIn
* GooglePlus: тип: string, ссылка на GooglePlus
* Twitter: тип: string, ссылка на Twitter
* Skype: тип: string, ссылка на Skype
* MoiKrug: тип: string, ссылка на MoiKrug
* OK: тип: string, ссылка на Одноклассники

**Place**: тип int, место работы, 1 - у работодателя, 2 - удаленно, 4 - не важно. Если выбрано несколько пунктов, то значения складываются.

**Employment**: тип int, занятость, 1 - полная, 2 - частичная, 4 - проектная, 8 - стажировка, 16 - волонтерство, 32 - не важно. Если выбрано несколько пунктов, то значения складываются.

**Schedule**: тип int, график работы, 1 - полный, 2 - гибкий, 4 - вахтовый, 8 - сменный, 16 - неполный, 32 - не важно. Если выбрано несколько пунктов, то значения складываются.

**Citizenship**: тип: string, гражданство кандидата.

**AboutMe**: тип: string, дополнительная информация о кандидадте.

**BusinessTrip**: тип int, готовность к командировкам, -1 - не готов(а), 0 - неизвестно, 1 - готов(а).

**Childs**: тип int, наличие детей, 0 - неизвестно, 1 - есть, 2 - нет.

**MaritalStatus**: тип int, семейное положение, 0 - неизвестно, 1 - в браке, 2 - не в браке.

**HealthBook**: тип int, наличие санкнижки, 0 - неизвестно, 1 - есть, 2 - нет.

**DriverLicence**: тип: string, водительские права.

**Courses**: тип: массив объектов, курсы кандидата. Свойства объекта:

* Name: тип: string, название
* Organization: тип: string, организация
* Year: тип: int, год

**Recommendations**: тип: массив объектов, рекомендации кандидата. Свойства объекта:

* Company: тип: string, компания
* Phone: тип: string, контакты
* Name: тип: string, имя представителя
* Result: тип: string, результат проверки

**Experience**: тип: массив объектов, опыт работы. Свойства объекта:

* Company: тип: string, компания
* Position: тип: string, должность
* City: тип: string, регион
* FromMonth: тип: int, месяц начала работы, где январь - 0, а декабрь - 12
* ToMonth: тип: int, месяц окончания работы, где январь - 0, а декабрь - 12
* FromYear: тип: int, год начала работы
* ToYear: тип: int, год окончания работы
* Description: тип: string, должностные обязанности

**Education**: тип: массив объектов, образование. Свойства объекта:

* Level: тип int, уровень образования, 0 - высшее, 1 - неполное высшее, 2 - средне-специальное, 3 - среднее
* University: тип: string, название учебного заведения
* Faculty: тип: string, факультет/специальность
* GraduateYear: тип: int, год окончания

**Skills**: тип: массив string, ключевые навыки

**CustomFieldsValues**: CustomFieldValue[] - значения настаиваемых (кастомных) полей для кандидата. Перед использованием поля должны быть созданы сотрудники FriendWork. Для этого обратитесь в службу поддержки.

**Relocations**: тип: массив string, регионы переезда

**Subway**: тип: string, название станции метро

**Langs**: тип: объект коллекции (ключ - значение), языки и их уровни владения, например:

"**Langs**": { "Русский": "родной", "Английский": "свободно владею" }

**TaskJobId**: тип int, id вакансии, на которую необходимо назначить задачу

**CallInfo**: тип: массив объектов, информация о звонках. Свойства объекта:

* CreationTime: тип string, время осуществления звонка, формат (yyyy-MM-dd hh:mm:ss).
* Status: тип int, статус звонка, 0 - интересен, 1 - не интересен, 2 - не удалось получить ответ по любой причине

**Score**: тип int, оценка кандидата, 0 - неизвестно, 1 - пройдена, 2 - не пройдена.

**ApprovePersonalData**: тип bool, флаг, указывающий одобрена ли обработка персональных данных, true - да, false - нет.

Примечания:

При обнаружении дубликата, кандидат не будет внесён в базу данных, если параметр DuplicateProcessing не указан или принимает соответствующее значение.

При успешном добавлении кандидата в базу данных, в возвращаемом значении API будет содержаться свойство CandidateId (тип: int) с идентификатором кандидата.

### Тип CustomFieldValue

{

SystemName: string - название кастомного поля

Value: object - значение кастомного поля

}

### Примеры с минимальным набором данных

{

"LastName": "Иванов31",

"Contacts": [

{

"E-mail": "test121131@mail.ru"

}

]

}

Или

{

"LastName": "Иванов32",

"Contacts": [

{

"Phone": "69999999999"

}

]

}

### Пример 2

{

"FirstName": "Иван",

"MiddleName": "Иванович",

"LastName": "Иванов",

"City": "Москва",

"BirthDate": "1980-01-01",

"Source": "HeadHunter",

"ExternalLink": "https://spb.hh.ru/resume/d6bf16ee7891db73d60039ed1f4f523238395a",

"AddWay": "Активный поиск",

"Contacts": [

{

"E-mail": "test@mail.ru"

},

{

"Phone": "79999999999"

},

]

}

### Пример 3

{

"FirstName": "Иван",

"MiddleName": "Иванович",

"LastName": "Иванов",

"City": "Москва",

"BirthDate": "1980-01-01",

"Source": "HeadHunter",

"ExternalLink": "https://spb.hh.ru/resume/d6bf16ee7891db73d60039ed1f4f523238395a",

"AddWay": "Активный поиск",

"Contacts": [

{

"E-mail": "test@mail.ru"

},

{

"Phone": "79999999999"

}

]

}

### Пример 4

{

"FirstName": "Иван",

"MiddleName": "Иванович",

"LastName": "Иванов",

"City": "Санкт-Петербург",

"BirthDate": "1988-04-16",

"Age": 21,

"Sex": 2,

"Position": ".Net developer",

"Salary": 10000,

"AddWay": "Отклик",

"Source": "Instagram",

"Contacts" : [

{

"E-mail" : "somebody@example.com"

},

{

"Phone" : "+79001002030"

}

],

"RelocationReadiness": 1,

"Resume": "Мое резюме",

"DuplicateProcessing": "Add",

"SocialLinks":

{

"VK" : "vk",

"Facebook": "Facebook",

"LinkedIn": "LinkedIn",

"GooglePlus": "GooglePlus",

"Twitter": "Twitter",

"Skype": "Skype",

"MoiKrug": "MoiKrug",

"OK": "OK"

},

"Place": 2,

"Employment": 4,

"Schedule": 16,

"Citizenship": "Россия",

"AboutMe": "Обо мне",

"BusinessTrip": 1,

"Childs": 2,

"MaritalStatus": 2,

"Courses":[

{

"Name": "Название",

"Organization": "Организация",

"Year": 2018

},

{

"Name": "Название2",

"Organization": "Организация2",

"Year": 2019

}

],

"Recommendations":[

{

"Company": "Компания",

"Phone": "Контакты",

"Name": "Имя представителя",

"Result": "Результат проверки"

}

],

"Experience":[

{

"Company": "Компания",

"Position": "Должность",

"City": "Москва",

"FromMonth": 1,

"ToMonth": 2,

"FromYear": 1998,

"ToYear": 2001,

"Description": "Должностные обязанности" }

],

"Education":[

{

"Level": 1,

"University": "Название учебного заведения",

"Faculty": "Факультет/специальность",

"GraduateYear": 1998

}

],

"Skills": [ "Skills1", "Skills2" ],

"HealthBook": 2,

"DriverLicence": "A,B,D",

"Subway": "Адмиралтейская",

"Relocations": [ "Москва", "Омск" ],

"Langs":

{

"Русский": "родной",

"Английский": "свободно владею"

},

"TaskJobId": 17,

"CallInfo": [

{

"CreationTime": "2019-12-19 18:05:45",

"Status": 1

}

],

"CustomFieldsValues": [ { “systemName”: “myCustomField”, “value”: 123 } ]

}

## Редактирование кандидата

Путь: <http://api.friend.work/Candidate/set>

Тип метода: POST

Синтаксис полностью идентичен с добавление кандидата, за одним исключением:  
для того, чтобы отредактировать кандидата в объект необходимо передать поле

**CandidateId**: тип: int, идентификатор кандидата.

### Пример с минимальным набором данных

{

"CandidateId": 2015657,

"LastName": "Иванов31",

"Contacts": [

{

"E-mail": "test121131@mail.ru"

}

]

}

Примечания:

При обнаружении дубликата, кандидат не будет внесён в базу данных, если параметр DuplicateProcessing не указан или принимает соответствующее значение.

При успешном добавлении кандидата в базу данных, в возвращаемом значении API будет содержаться свойство CandidateId (тип: int) с идентификатором кандидата.

## Добавление события/статуса для кандидата

### Адрес запроса

[https://api.friend.work/Candidate/{id}/CandidateHistories/set](https://api.friend.work/Candidate/%7Bid%7D/CandidateHistories/set)

Где {id}: int, это идентификатор кандидата которому нужно добавить событие

### Тип запроса

POST

### Формат запроса

{

"CandidateHistoryId": long, - идентификатор события кандидата. При добавлении не заполняйте его, оно возможно вам может пригодится при [получении событий](#_e1il1kqowyh2),

"Timestamp": long, - дата и время в формате Unix Time (в миллисекундах). Служит для однозначной идентификации времени поступления информации об обрабатываем событии и предотвращения перезаписи актуальных данных их устаревшей версией. Этот параметр может отсутствовать, но тогда актуальность данных не гарантируется

"Name": string, - название события (статуса) из доступных в аккаунте пользователя

"JobId": int, - Id вакансии. Нельзя создавать статус без вакансии.

"FromDate": string, - дата начала события (может отсутствовать, если в интерфейсе у этого события не нужно устанавливать)

"ToDate": string,- дата окончания события (может отсутствовать, если в интерфейсе у этого события не нужно устанавливать)

"IsAllDay": bool, - означающий будет ли событие длиться весь день

"Description": string, - комментарий к событию

"Rating": int, - рейтинг события. Может принимать значения от 0 до 5, где 0 - рейтинг не выставлен.

"ResponsibleId": int, - Id пользователя, который будет установлен как ответственный за статус и его создатель. Значение -1 будет соответствовать аккаунту по умолчанию в качестве создателя и “Без ответственного”.

"Additional": // Deprecated - поле будет удалено в будущих релизах

AdditionalItem[] // Deprecated - Необязательное поле. Используется при интеграциях с внешними сервисами. Представляет собой дополнительную информацию, которую следует добавить в событие для кандидата

}

### Формат ответа

{

Code: int,

Message: string

}

### Пример запроса без Additional

{

"Timestamp": 1576686743443,

"Name": "Новый",

"JobId": 123761,

"FromDate": "30.12.2019 12:00",

"ToDate": "10.02.2020 12:00",

"IsAllDay": false,

"Description": "Описание события",

"Rating": 2,

"ResponsibleId": -1,

"Additional": []

}

### Пример запроса с Additional

{

"Timestamp": 1576686743443,

"Name": "Новый",

"JobId": 92196,

"FromDate": "30.12.2019 12:00",

"ToDate": "10.02.2020 12:00",

"IsAllDay": false,

"Description": "Описание события",

"Rating": 2,

"ResponsibleId": -1,

"Additional": // Deprecated - поле будет удалено в будущих релизах

[{

"PrivateName": "VcvId",

"Alias": "Вакансия VCV",

"Value": {

"id": "abcd1234",

"display": "Вакансия Стажер"

}

}

]

}

### Пример ответа:

{

"CandidateHistoryId": 23892324,

"Code": 0,

"Message": "CandidateHistory was added"

}

## (TODO) Добавление отклика существующему кандидату

Путь: **/Candidate/{id}/Applications/set**

Тип метода: POST

Где {id} = идентификатор кандидата

Параметры объекта:

Timestamp: тип: long, дата и время в формате Unix Time (в миллисекундах). Служит для однозначной идентификации времени поступления информации об обрабатываемом отклике и предотвращения дублирования информации. Этот параметр может отсутствовать, но тогда есть вероятность дублирования отклика.

PublicationSource: тип string, см. Source в добавлении кандидата.

PublicationLink: тип string, ссылка на адрес публикации (с ведущим http или https), на которую откликнулся кандидат на работном сайте.

PublicationName: тип: string, название публикации (вакансии на работном сайте).

PublicationId: тип string, Id публикации.

JobId: тип int, Id вакансии во FriendWork, если вакансии нет, то можно не передавать или отправить 0.

DateCreated: тип: string в формате “dd.MM.yyyy HH:mm” (UTC), дата создания отклика (может отсутствовать, тогда берется текущая).

LastUpdate: тип: string в формате “dd.MM.yyyy HH:mm” (UTC), дата обновления отклика (может отсутствовать).

CoverLetter: тип: string, сопроводительное письмо.

Rating: тип: int, рейтинг события. Может принимать значения от 0 до 5, где 0 - рейтинг не выставлен.

Additional: тип: массив объектов. Представляет собой дополнительную информацию, которую следует добавить в отклик кандидата. Смотри такой же параметр в добавлении события.

Примечания:

При успешном добавлении отклика в базу данных, в возвращаемом значении API будет содержаться свойство ApplicationId с Id события.

Пример входных параметров:

{

"PublicationId" : "12345",

"PublicationSource" : "HeadHunter",

"PublicationLink" : "https://hh.ru/12345",

"PublicationName" : "Новый отклик",

"DateCreated" : "16.10.2016 11:07",

"LastUpdate" : "16.10.2016 11:07",

"CoverLetter" : "Сопроводительное письмо"

}

## (TODO) Добавление отклика вместе с новым кандидатом

[Альтернативный вариант](https://docs.google.com/document/d/1FT3hJ1grvrOM5EuMgiAovAi1xelwoyDfeHlAevsITKU/edit#heading=h.u5waf0ei6z5u) **(реализован)**

Путь: **/Candidate/Applications/set**

Тип метода: POST

Параметры объекта:

Аналогичны параметрам объекта при добавлении отклика существующему кандидату. При этом обязательно должно присутствовать следующее свойство:

Candidate: кандидат, смотри аналогичный объект в добавлении кандидата.

Примечания:

При успешном добавлении отклика в базу данных, в возвращаемом значении API будет содержаться свойство ApplicationId с Id события.

Пример входных параметров:

{

"PublicationId" : "12345",

"PublicationSource" : "HeadHunter",

"PublicationLink" : "https://hh.ru/12345",

"PublicationName" : "Новый отклик",

"DateCreated" : "16.10.2016 11:07",

"LastUpdate" : "16.10.2016 11:07",

"CoverLetter" : "Содержание отклика",

"Candidate" : {

...

}

}

## Получение событий кандидата

Метод возвращает список всех событий (кроме удаленных) кандидата, включая статус без вакансии (обычно называется “Новый”) и статусы, которые были перекрыты их подстатусами (пример: после “Интервью” поставили статус “Интервью состоялось”, оба будут в выдаче). При необходимости их можно отфильтровать уже после получения.

### Адрес запроса

[https://api.friend.work/Candidate/{id}/CandidateHistories](https://api.friend.work/Candidate/%7Bid%7D/CandidateHistories)

Где {id}: int, это идентификатор кандидата которому нужно добавить событие

### Тип запроса

GET

### Формат запроса

Не нужно передавать дополнительных данных

### Формат ответа

{

Code: int,

Message: string,

CandidateHistories: [] - массив объектов, каждый из которых аналогичен параметру переданному в “[Добавление события для кандидата](#_jj746fpd24eg)”

}

### Пример ответа

{

"Code": 0,

"Message": null,

"CandidateHistories": [

{

"CandidateHistoryId": 123,

"Timestamp": 1576686743443,

"Name": "Новый",

"JobId": 123761,

"FromDate": "30.12.2019 12:00",

"ToDate": "10.02.2020 12:00",

"IsAllDay": false,

"Description": "Описание события",

"Rating": 2,

"ResponsibleId": -1,

"AccountId": 0,

"Additional": [ // Deprecated - поле будет удалено в будущих релизах

{

"PrivateName": "VcvId",

"Alias": "Вакансия VCV",

"Value": {

"id": "abcd1234",

"display": "Вакансия Стажер"

}

}

]

}

]

}

## Получение событий кандидатов по фильтру

Метод возвращает список всех событий (кроме удаленных) кандидатов, соответствующих фильтру, включая статусы без вакансии (обычно называется "Новый") и статусы, которые были перекрыты их подстатусами (пример: после "Интервью" поставили статус "Интервью состоялось", оба будут в выдаче).

### Адрес запроса

<https://api.friend.work/Candidate/CandidatesHistories>

### Тип запроса

POST

### Формат запроса

{

**CustomersIds**: int[]? - массив идентификаторов заказчиков, [0] - истории по вакансиям без заказчика.

**StatusesIds**: int[]? - массив идентификаторов статусов.

**FromDate**: string? - дата и время начала периода, в который должна войти дата начала события (Формат yyyy-MM-dd / yyyy-MM-dd hh:mm). При наличии в запросе этого параметра в ответ будут переданы *только* события, имеющие поле объекта FromDate (т.е. события с типами “Дата и время” и “Диапазон дат”).

**ToDate**: string? - дата и время окончания периода, в который должна войти дата начала события (Формат yyyy-MM-dd / yyyy-MM-dd hh:mm). При наличии в запросе этого параметра в ответ будут переданы *только* события, имеющие поле объекта FromDate (т.е. события с типами “Дата и время” и “Диапазон дат”).

**FromDateCreated**: string? - дата и время начала периода, в котором событие должно быть создано (Формат yyyy-MM-dd / yyyy-MM-dd hh:mm). Значение по умолчанию: 1970-01-01 00:00.

**ToDateCreated**: string? - дата и время окончания периода, в котором событие должно быть создано (Формат yyyy-MM-dd / yyyy-MM-dd hh:mm). Значение по умолчанию: дата и время на момент отправки запроса.

**MinCandidateHistoryId**: int? - идентификатор события, начиная с которого события могут попасть в выдачу.

**Count**: int? - количество возвращаемых историй кандидатов. Значение по умолчанию (оно же максимальное) - 1000.

}

### Пример запроса

{

"CustomersIds": [0],

"StatusesIds" : [],

"FromDate" : "2014-06-20",

"ToDate": "2014-07-19"

}

### Формат ответа

{

**CandidateHistories**: [] - массив объектов, каждый из которых аналогичен параметру переданному в “[Добавление события для кандидата](#_jj746fpd24eg)”, а также идентификатор кандидата в поле CandidateId,

**CandidateHistoriesCount**: int - количество возвращаемых историй кандидатов,

**Code**: int - [код ошибки](#_latsdh4uhyrb)

**Message**: string - [текст сообщения об ошибке](#_latsdh4uhyrb).

}

### Пример ответа

{

"CandidateHistories": [

{

"TimeStamp": null,

"DateCreated": "19.06.2014 14:08",

"Name": "",

"JobId": 3,

"JobName": null,

"FromDate": "29.06.2014 21:00",

"ToDate": "13.07.2014 21:00",

"IsAllDay": false,

"Description": null,

"Rating": null,

"ResponsibleId": 2,

"AccountId": 0,

"Additional": [],

"CandidateId": 584

}

],

"CandidateHistoriesCount": 1000

"Code": 0,

"Message": "OK"

}

## (TODO) Список аккаунтов

Путь: **/Account**

Тип метода: GET

При успешном выполнении в возвращаемом значении будет содержаться список аккаунтов, относящихся к текущему клиенту.

Пример возвращаемого списка:

[

{

"AccountId" : "12345",

"UserName" : "e@mail.com",

"FirstName" : "Иван",

"LastName" : "Иванов",

"Phone" : "7777777",

"Department" : "Example"

},

{

"AccountId" : "12346",

"UserName" : "e@mail.ru",

"FirstName" : "Иван",

"LastName" : "Иванов",

"Phone" : "+7 (921) 921-21-21",

"Department" : "Пример"

}

]

## Получение публикации

**Публикация** - это вакансия, которая опубликована на работном сайт (hh, superjob и др.) или на карьерном портале.

Путь: **/Publication/{id}**

Тип метода: GET

При успешном выполнении в возвращаемом значении будет содержаться публикация по данному id.

Пример возвращаемого списка:

{

"Status" : "2", // Active = 0, Deleted = 1, Archived = 2, Draft = 3

"Title" : "Example",

"SalaryTo" : "50000",

"SalaryFrom" : "40000",

"DateCreated" : "12.06.2017",

"LastUpdate" : "13.06.2017",

"Subway" : "Example"

"JobId" : "123",

"Company" : "Example",

"SalaryTo" : "50000",

"SalaryFrom" : "40000",

"DateCreated" : "12.06.2017",

"LastUpdate" : "13.06.2017",

"RegionId" : "123"

"City" : "Москва",

"Address" : "Адрес офиса",

"ExpYearsMin" : "1", // Опыт работы от (лет)

"ExpYearsMax" : "2", // Опыт работы до (лет)

"EmploymentType" : "1" // Fulltime = 0, Halftime = 1, Temporary = 2,

Internship = 3

"ApplicationBaseFields" : "{“Name”: “Example”, “Label”: “Example” ,

“Requirement” : “1” }" // Обязательные поля при

отклике

}

## Список публикаций

**Не рекомендуется использовать для своего карьерного портала.**

Путь: **/Publication?page=1&perpage=20&status=active**

Тип метода: GET

При успешном выполнении в возвращаемом значении будет содержаться список публикаций карьерного портала, относящихся к текущему клиенту.

Пример возвращаемого списка:

[

{

"Status" : "2", // Active = 0, Deleted = 1, Archived = 2, Draft = 3

"Title" : "Example",

"SalaryTo" : "50000",

"SalaryFrom" : "40000",

"DateCreated" : "12.06.2017",

"LastUpdate" : "13.06.2017",

"Subway" : "Example"

"JobId" : "123",

"Company" : "Example",

"SalaryTo" : "50000",

"SalaryFrom" : "40000",

"DateCreated" : "12.06.2017",

"LastUpdate" : "13.06.2017",

"RegionId" : "123"

"City" : "Москва",

"Address" : "Адрес офиса",

"ExpYearsMin" : "1", // Опыт работы от (лет)

"ExpYearsMax" : "2", // Опыт работы до (лет)

"EmploymentType" : "1" // Fulltime = 0, Halftime = 1, Temporary = 2,

Internship = 3

"ApplicationBaseFields" : "{“Name”: “Example”, “Label”: “Example” ,

“Requirement” : “1” }" // Обязательные поля при

отклике

},

{

"Status" : "2", // Active = 0, Deleted = 1, Archived = 2, Draft = 3

"Title" : "Example",

"SalaryTo" : "50000",

"SalaryFrom" : "40000",

"DateCreated" : "12.06.2017",

"LastUpdate" : "13.06.2017",

"Subway" : "Example"

"JobId" : "123",

"Company" : "Example",

"SalaryTo" : "50000",

"SalaryFrom" : "40000",

"DateCreated" : "12.06.2017",

"LastUpdate" : "13.06.2017",

"RegionId" : "123"

"City" : "Москва",

"Address" : "Адрес офиса",

"ExpYearsMin" : "1",

"ExpYearsMax" : "2",

"EmploymentType" : "1" // Fulltime = 0, Halftime = 1, Temporary = 2,

Internship = 3

"ApplicationBaseFields" : "{“Name”: “Example”, “Lable”: “Example” ,

“Requirement” : “1” }"

}

]

## Список сокращенных публикаций

**Рекомендуется использовать для своего карьерного портала.**

**Сокращенная публикация** - это та же самая публикация, только возвращаем не все свойства, а их часть.

Путь: **/Publication/Brief?page=1&perpage=20&status=active**

Тип метода: GET

При успешном выполнении в возвращаемом значении будет содержаться список публикаций карьерного портала, относящихся к текущему клиенту.

Пример возвращаемого списка:

[

{

"Status" : "2", // Active = 0, Deleted = 1, Archived = 2, Draft = 3

"Title" : "Example",

"SalaryTo" : "50000",

"SalaryFrom" : "40000",

"DateCreated" : "12.06.2017",

"LastUpdate" : "13.06.2017",

"Subway" : "Example"

"JobId" : "123",

"Company" : "Example",

"SalaryTo" : "50000",

"SalaryFrom" : "40000",

"DateCreated" : "12.06.2017",

"LastUpdate" : "13.06.2017",

},

{

"Status" : "2", // Active = 0, Deleted = 1, Archived = 2, Draft = 3

"Title" : "Example",

"SalaryTo" : "50000",

"SalaryFrom" : "40000",

"DateCreated" : "12.06.2017",

"LastUpdate" : "13.06.2017"

"Subway" : "Example"

"JobId" : "123",

"Company" : "Example",

"SalaryTo" : "50000",

"SalaryFrom" : "40000",

"DateCreated" : "12.06.2017",

"LastUpdate" : "13.06.2017"

}

]

## Список категорий и специализаций

Путь: **/Publication/Specializations**

Тип метода: GET

При успешном выполнении в возвращаемом значении будет содержаться список категорий и специализаций, относящихся к текущему клиенту.

Пример возвращаемого списка:

{

"Categories":[

{

"Id":9,

"Name":"Категория1"

},

{

"Id":8,

"Name":"Категория2"

},

{

"Id":7,

"Name":"Категория3"

}

],

"Specializations":[

{

"Id":1,

"Name":"Специализация1",

"CategoryId":9

},

{

"Id":2,

"Name":"Специализация2",

"CategoryId":8

},

{

"Id":3,

"Name":"Специализация3",

"CategoryId":7

}

],

"Code":0,

"Message":null

}

Замечание: категории пока не поддерживаются.

## (TODO) Добавление вакансии

Путь: **/Jobs**

Тип метода: POST

Тело метода:

{

"Name": "Тестовая вакансия с API",

"PositionCount": 1,

"RegionId": 4999,

"StatusId": 1,

"OpenDate": "01.01.2022"

"PlannedClosingDate": "10.10.2022",

"SuspensePeriods": [

{

"FromDate": "05.05.2022",

"Comment": "123",

"SuspendReason": "321",

"ToDate": "06.05.2022"

}

],

"CustomerId": 11549,

"ResponsibleId": 2,

"EmployeeIds": [ 2 ],

"Description": "description",

"Comment": "123",

"CustomFieldsValues": [

{

"SystemName": "OneAndOnly",

"Value": 3

},

{

"SystemName": "OneAndOnly2",

"Value": [

3,

4

]

},

{

"SystemName": "OneAndOnly3",

"Value": "156"

},

{

"SystemName": "MacBlr\_I\_can\_work",

"Value": 0

}

]

}

При успешном добавлении вакансии получаем ответ со статусом 200 и вакансией, которую добавили

Тело ответа:

{

"JobId": 202789,

"Status": 1,

"IsDraft": 0,

"Name": "Тестовая вакансия с API",

"RegionId": 4999,

"Customer": 11549,

"OrganizationUnitId": null,

"Responsible": 2,

"IsDeleted": 0,

"Description": "description",

"Comment": "123",

"FilesHash": null,

"ClientId": 1,

"DayCount": null,

"Recruiters": [

2

],

"Additional": null,

"FromDate": "2022-07-04T00:00:00",

"ToDate": null,

"PlannedToDate": null,

"SuspensePeriods": "[{\"fromDate\":\"Thu May 05 2022\",\"toDate\":\"Sun Jun 05 2022\",\"comment\":\"123\",\"suspendReason\":\"321\"}]",

"AllowedCandidateStatuses": [],

"PositionCount": 1,

"JobTypeId": null,

"JobPriorityId": 0,

"IsConfidential": false,

"CustomFieldsValues": null,

"AssignedJobTests": null,

"StaffPositionId": null,

"RecruitmentApplicationId": null

}

## Прикрепление файла к кандидату/вакансии

Путь: **/Files/{objectType}/{entityId}**

Тип метода: POST

Описание параметров из запроса

ObjectType - Candidate = 1, Job = 2

EntityId - Идентификатор сущности (кандидата, вакансии)

Тело метода: файл

При успешном добавлении файла получаем ответ со статусом 200

## Работа со значениями кастомных полей объектов

Для кандидата (или вакансии), созданного в системе возможно создание/обновление значений кастомных полей. Необходимое для этого API описано в [документации по API Организационно-штатной структуры - 2.6. CustomFields](https://docs.google.com/document/d/1GP1BiTkb6-UC1o_ibWHVp23NfBVjZkJuq1FKR1ksP64/edit#heading=h.hdebsh9cgn8f).

# Возвращаемые значения API

В общем случае API вернет сообщение в формате JSON следующего вида:

{

Code = код ошибки,

Message = сообщение

}

Code: тип: int, может принимать следующие значения:

0 = Ошибка отсутствует.

1 = Общая ошибка API.

2 = Ошибка авторизации.

3 = Ошибка при сохранении/редактировании данных.

4 = Ошибка при получении устаревших данных.

5 = Обрабатываемый объект удален.

Message: тип: string, опциональный параметр, может содержать подробное описание ошибки или результат выполненного действия.

В следующих версиях документации будет учтено, что по API можно прислать файлы и другие данные, которые требуют специфической обработки.

# 

# Webhooks

Актуализированная документация для вебхуков

## Методы внешнего API

Данный раздел станет актуален после влития [мержа](https://gitlab-99.rc2.friendwork.ru/friendwork/recruiter/-/merge_requests/6145).

### GET /webhooks | Возвращает список вебхуков

Не имеет параметров. Возвращает список *определений вебхуков* для текущего клиента. Состав данных в ответе описан в [разделе “Определение вебхука”](#_1xapouxgrbee).

Пример ответа:

| [  {  "id": 443,  "additionalData": 1,  "authorization": null,  "jobId": null,  "candidateStatusId": null,  "content": null,  "url": "https://webhook.site/095b6e3b-7c21-42d4-ba3d-b2a992ba648d",  "method": 1,  "name": "Test 5",  "rule": 2,  "senderEmail": null,  "type": 1,  "dataItems": ["webhook.name"]  }  ] |
| --- |

### POST /webhooks | Создаёт новый вебхук

В теле запроса ожидает json *определение вебхука*.

Состав данных в ответе описан в [разделе “Определение вебхука”](#_1xapouxgrbee).

Пример тела запроса:

| {  "additionalData": 1,  "authorization": null,  "jobId": null,  "candidateStatusId": null,  "content": null,  "url": "http://localhost",  "method": 1,  "name": "Test 4",  "rule": 2,  "senderEmail": null,  "type": 1,  "dataItems": ["webhook.name"]  } |
| --- |

Пример ответа:

| {  "id": 444  } |
| --- |

### DELETE /webhooks | Удаляет вебхук с указанным Id

В query-параметре с именем id указывается число - идентификатор вебхука, который следует удалить.

## Определение вебхука

Определение вебхука состоит описания состава передаваемых данных, правила при котором вебхук будет вызываться, а также данных о том как вызывать вебхук (например URL по которому следует обращаться).

### Поля определения

### Поля, показывающие базовые данные о вебхуке

* Name - строка или null. Наименование хука. Используется только для удобства поддержки.
* Url - строка. Адрес, по которому будет совершено обращение при вызове хука.
* Method - число или строка, показывающая какой HTTP-метод использовать при обращении: Get (0) или Post (1).
* Authorization - значения для HTTP-заголовка Authorization в запросе, исполняемом при вызове вебхука.
* Type - строка или число. Возможные значения:
  + “sendRequest” (1) - при вызове хука отправлять HTTP-запрос
  + “sendEmail” (2) - при вызове хука отправлять письмо по электронной почтой. Содержимое письма - фиксированный набор данных о событии.
  + “sendEmailTemplate” (3) - при вызове хука отправлять письмо по электронной почтой. Содержимое письма - EmailTemplate, заполненный данными.
* Content - строка, представляющая число, или null. Показывает идентификатор EmailTemplate’а, который следует использовать для отправки письма при указании Type = 2 или Type = 3.
* Email - строка или null, адрес электронной почты, на который следует отправить письмо при указании Type = 2 или Type = 3.
* SenderEmail - строка или null, адрес электронной почты (с опциональным указанием читаемого имени), который который следует использовать как адрес отправителя для письма при указании Type = 2 или Type = 3.

### Поля, отвечающие за условие вызова хука

* CandidateStatusId - число или null. Если не null, то вебхук будет вызываться только при переходе кандидата на указанный статус. Если null - то не ограничивает вызов хука конкретным статусом (но и не разрешает, см. поле Rule)
* Rule - стока или число, показывающее условие для вызова хука. Возможные значения:
  + None (0) - вебхук может быть вызван только при переходе кандидата на определенный CandidateStatusId.
  + OnlyOnAdd (1) - показывает что следует вызывать вебхук при создании кандидата, если CandidateStatusId равен null. Иначе действует как None (0).
  + FireOnAnyStatus (2) - показывает, что вебхук следует вызывать при назначении любого статуса кандидату. Если установлено это значение, то CandidateStatusId игнорируется.
* JobId - число или null, показывает идентификатор вакансии, по который должен быть назначен статус, для отправки сообщения по электронной почте при указании Type = 2 или Type = 3.

### Поля, отвечающие за состав данных в запросе

* AdditionalData - строка или число. Показывает, следует ли включать информацию о кандидате в json, отправляемый в запросе при вызове хука. Допустимые значения: None (0) или WithCandidate (1)  
  *Примечание*: данное поле по сути ничем не отличается от спецификатора, добавляющего в запрос элемент данных “[candidate](#_8vfmczt08i0d)”.
* DataItems - массив строк. Каждая строка является спецификацией элемента данных. Возможные спецификаторы:
  + ~~webhook.name~~
  + candidateHistory.creator
  + candidateHistory.responsible
  + job.organization\_unit
  + job.organization\_unit.custom\_fields

## Структура передаваемых вебхуком данных

В зависимости от конфигурации и данных события, запустившего вебхук, состав передаваемых данных может различаться.

Передаваемый объект может иметь одно или более полей, показывающих элементы данных:

* candidate
* candidateHistory
* webhook
* candidateHistory.creator
* candidateHistory.responsible
* job.organizationUnit
* job.organizationUnit.customFields

### Candidate | Информация о кандидате

Передаётся в случае, если в определении хука указано AdditionalData = WithCandidate. Структура объекта соответствует [представлению кандидата в API](#_opmm7j5ooyk).

### CandidateHistory | Информация о новом/измененном статусе

Передаётся в случае, если событием, приведшим к вызову к вебхука, было назначение статуса кандидату, или изменение записи в его ленте. Структура объекта соответствует [представлению истории кандидата в API](#_ogprd8a87gb0).

### Webhook | Информация о сработавшем вебхуке

Добавляет элемент данных, показывающий наименование вызванного вебхука. Содержит только одно поле: name - строка или null, название вебхука (определяется при создании).

### CandidateHistory.creator | Информация о создателе истории кандидата

Передаётся в случае, если событием, приведшим к вызову к вебхука, было назначение статуса кандидату, или изменение записи в его ленте, при условии что соответствующий спецификатор был установлен в dataItems при определении вебхука.

Структура объекта:

* firstName - строка, имя пользователя
* lastName - строка, фамилия пользователя
* userName - строка, логин (email) пользователя

### CandidateHistory.responsible | Информация об отственном за историю кандидата

Передаётся в случае, если событием, приведшим к вызову к вебхука, было назначение статуса кандидату, или изменение записи в его ленте, при условии что соответствующий спецификатор был установлен в dataItems при определении вебхука.

Структура объекта:

* firstName - строка, имя пользователя
* lastName - строка, фамилия пользователя
* userName - строка, логин (email) пользователя

### Job.organization\_unit | Информация о подразделении, связанном с вакансией, по которой был установлен статус

Передаётся в случае, если событием, приведшим к вызову к вебхука, было назначение статуса кандидату, или изменение записи в его ленте, при условии что соответствующий спецификатор был установлен в dataItems при определении вебхука, и вакансия, связанная с установленным статусом кандидата, имела связанное подразделение.

Структура объекта:

* name - строка, наименование подразделения
* foreignKey - строка или null, внешний ключ подразделения
* Id - число, уникальный идентификатор подразделения

### Job.organization\_unit.custom\_fields | Значения настраиваемых полей подразделения, связанного с вакансией, по которой был установлен статус

Передаётся в случае, если событием, приведшим к вызову к вебхука, было назначение статуса кандидату, или изменение записи в его ленте, при условии что соответствующий спецификатор был установлен в dataItems при определении вебхука, и вакансия, связанная с установленным статусом кандидата, имела связанное подразделение.

Структура объекта соответствует [представлению значений настраиваемых полей в API](#_q4qytiarutgs).

### Особенности структуры передаваемого JSON в связи с разными составами данных.

Базовым элементом данных является информация о статусе кандидата или (при назначении статуса) или о самом кандидате (при создании). Если в итоговом хуке есть только один элемент данных, и он явлдяется информацией о назначенном статусе или кандидате, то этот элемент будет **корнем отправляемого json**.

Во всех иных случаях, отправляться будет json-объект, полями которого являются элементы данных. Пример такого json:

| {  "candidate": { ... },  "candidateHistory": { ... },  "candidateHistory.creator": {  "username": "abc@xyz.biz",  "firstName": "Миша",  "lastName": "Проказин"  },  "job.organization\_unit": {  "foreing\_key": ...,  "name": ...  }  } |
| --- |

Однако если в вебхуке не указано что следует передавать информацию о кандидате, а так же отсутствуют спецификаторы "job.organization\_unit" и "candidateHistory.creator", то единственным элементом данных окажется candidateHistory, тогда итоговый отправленный json **НЕ будет** выглядеть так:

| {  "candidateHistory": {  "candidateHistoryId": 5,  "name": "На связи",  "jobId": 211306,  "jobName": "Оператор крана",  ...  }  } |
| --- |

Корнем json-объекта станет значение единственного элемента данных, т.е. отправлен будет следующий json:

| {  "candidateHistoryId": 5,  "name": "На связи",  "jobId": 211306,  "jobName": "Оператор крана",  ...  } |
| --- |