

# Методы управления данными - StudentCompetitionsApi

В разделе представлены сценарии REST-методов API для управления данными о студентах, соревнованиях и участиях.

## Эндпоинты

Для работы с сервисами зарегистрированы следующие эндпоинты:

Эндпоинт	Описание
{host}/api/students	Работа с каталогом студентов
{host}/api/competitions	Работа с каталогом соревнований
{host}/api/participations	Работа с каталогом участий
{host}/api/levels	Работа со справочником уровней
{host}/api/results	Работа со справочником результатов

## Создать студента - createStudent

Параметр	Значение
Название сценария	Создать студента
Название метода	createStudent
Эндпоинт	{host}/api/students
Глагол	POST
Описание	Метод создает нового студента
Ожидаемый результат	Возвращена ссылка на созданного студента ИЛИ Возвращено сообщение об ошибке
Ограничения	Отсутствуют

### Входные параметры

Название параметра	Тип данных	Обязательное	Размещение	Описание
name	String	Да	Body	Имя студента
surname	String	Да	Body	Фамилия студента
middleName	String	Нет	Body	Отчество студента
cardNumber	Long	Да	Body	Номер студенческого билета
faculty	String	Да	Body	Факультет
department	String	Нет	Body	Кафедра
studyGroup	String	Нет	Body	Учебная группа
email	String	Нет	Body	Электронная почта
phone	String	Нет	Body	Номер телефона

### Выходные параметры

В случае успешного выполнения метода возвращается URI созданного студента в заголовке Location. При возникновении ошибок возвращается сообщение об ошибке.

Название параметра	Тип данных	Обязательное	Размещение	Описание
Location	URI	Да	Header	URI созданного студента

### Сообщения об ошибках

HTTP-код ответа	Описание ошибки	Комментарий
400	Ошибка валидации данных	Не заполнены обязательные поля или данные не соответствуют ограничениям (cardNumber < 0, невалидный email или phone, недопустимый faculty)
500	Внутренняя ошибка сервера	Ошибка при работе с базой данных

## Описание сценария

### Базовый сценарий. Создать студента

№	Действие	АС
1	Система извлекает данные из запроса метода и проверяет, что заполнены все обязательные поля.	АС.1
2	Система проверяет валидность значений полей: <ul style="list-style-type: none"><li>- cardNumber &gt;= 0</li><li>- email соответствует формату электронной почты</li><li>- phone соответствует формату номера телефона</li><li>- faculty входит в список допустимых факультетов</li></ul>	АС.1
3	Система создает новую запись в таблице student. Поля заполняются по следующему правилу: student.id = сгенерированный UUID student.name = name student.surname = surname student.middle_name = middleName student.card_number = cardNumber student.faculty = faculty student.department = department student.study_group = studyGroup student.email = email student.phone = phone	АС.2
4	Система формирует URI созданного студента в формате {host}/api/students/{id} и помещает его в заголовок Location.	
5	Результат работы метода возвращается запрашивающей системе с http-кодом 201.	
6	Сценарий завершен.	

### Альтернативные сценарии

#### АС.1 Не заполнены обязательные поля или данные невалидны

№	Действие системы
1	Система формирует ответ с http-кодом 400 и описанием ошибок валидации.
2	Сценарий завершен

#### АС.2 При записи в базу данных произошла ошибка

№	Действие системы
1	Система возвращает ответ с http-кодом 500.
2	Сценарий завершен.

Получить студента по идентификатору - `getStudent`

Параметр	Значение
Название сценария	Получить студента по идентификатору
Название метода	<code>getStudent</code>
Эндпоинт	<code>{host}/api/students/{id}</code>
Глагол	GET
Описание	Метод возвращает информацию о студенте по идентификатору
Ожидаемый результат	Возвращена информация о студенте ИЛИ Возвращено сообщение об ошибке
Ограничения	Отсутствуют

Входные параметры

Название параметра	Тип данных	Обязательное	Размещение	Описание
<code>id</code>	String (UUID)	Да	Path	Идентификатор студента

Выходные параметры

В случае успешного выполнения метода возвращается информация о студенте.

Название параметра	Тип данных	Обязательное	Размещение	Описание
<code>student</code>	StudentDto	Да	Body	Информация о студенте

## Сообщения об ошибках

HTTP-код ответа	Описание ошибки	Комментарий
404	Студент не найден	Студент с указанным идентификатором не существует
500	Внутренняя ошибка сервера	Ошибка при работе с базой данных

## Описание сценария

### Базовый сценарий. Получить студента по идентификатору

№	Действие	АС
1	Система извлекает идентификатор из пути запроса.	
2	Система осуществляет поиск студента в базе данных при условии <code>student.id = id</code> и запоминает результат.	АС.1
3	Система проверяет, что результат поиска с шага 2 не пуст.	АС.2
4	Система заполняет результат работы метода данными найденного студента.	
5	Результат работы метода возвращается запрашивающей системе с http-кодом 200.	
6	Сценарий завершен.	

## Альтернативные сценарии

### АС.1 При чтении из базы данных произошла ошибка

№	Действие системы
1	Система возвращает ответ с http-кодом 500.
2	Сценарий завершен.

### АС.2 Студент по переданному идентификатору не найден

№	Действие системы
1	Система возвращает ответ с http-кодом 404.
2	Сценарий завершен.

## Получить список студентов - getStudents

Параметр	Значение
Название сценария	Получить список студентов
Название метода	getStudents
Эндпоинт	{host}/api/students
Глагол	GET
Описание	Метод осуществляет получение списка студентов с пагинацией и сортировкой
Ожидаемый результат	Возвращена коллекция студентов ИЛИ Возвращена пустая коллекция ИЛИ Возвращено сообщение об ошибке
Ограничения	Отсутствуют

## Входные параметры

Название параметра	Тип данных	Обязательное	Размещение	Описание
page	Integer	Нет	Query	Номер страницы. Нумерация zero-based. Значение по умолчанию: 0
size	Integer	Нет	Query	Размер страницы. Значение по умолчанию: 10
sort	String	Нет	Query	Сортировка в формате "поле,направление". Например: "surname,asc" или "cardNumber,desc". Можно передать несколько параметров sort

## Выходные параметры

В случае успешного выполнения метода возвращается коллекция студентов и информация о странице.

Название параметра	Тип данных	Обязательное	Размещение	Описание
content	List<StudentDto>	Да	Body	Коллекция студентов
page	PageDto	Да	Body	Информация о странице

### Сообщения об ошибках

HTTP-код ответа	Описание ошибки	Комментарий
500	Внутренняя ошибка сервера	Ошибка при работе с базой данных

### Описание сценария

#### Базовый сценарий. Получить список студентов

№	Действие	АС
1	Система извлекает параметры запроса: page, size, sort.	
2	Система осуществляет поиск студентов в базе данных. Примечание: К поиску применяется пагинация и сортировка на стороне базы данных.	АС.1
3	Система заполняет результат работы метода по следующему правилу: content = записи, найденные на предыдущем шаге page.number = номер страницы из параметров запроса page.size = количество записей из content page.totalElements = общее количество элементов page.totalPages = общее количество страниц page.first = результат проверки условия: page.number == 0 page.last = результат проверки условия: page.number == page.totalPages - 1 page.empty = результат проверки условия: page.totalElements == 0 page.sorts = примененные сортировки	
4	Результат работы метода возвращается запрашивающей системе с http-кодом 200.	
5	Сценарий завершен.	

### Альтернативные сценарии

#### АС.1 При чтении из базы данных произошла ошибка

№	Действие системы
1	Система возвращает ответ с http-кодом 500.
2	Сценарий завершен.

Поиск студентов по фильтру - searchStudents

Параметр	Значение
Название сценария	Поиск студентов по фильтру
Название метода	searchStudents
Эндпоинт	{host}/api/students/search
Глагол	POST
Описание	Метод осуществляет поиск студентов по заданным фильтрам
Ожидаемый результат	Возвращена коллекция студентов ИЛИ Возвращена пустая коллекция ИЛИ Возвращено сообщение об ошибке
Ограничения	Отсутствуют

Входные параметры

Название параметра	Тип данных	Обязательное	Размещение	Описание
page	Integer	Нет	Query	Номер страницы. Значение по умолчанию: 0
size	Integer	Нет	Query	Размер страницы. Значение по умолчанию: 10
sort	String	Нет	Query	Сортировка в формате "поле,направление"
filters	List<Filter>	Да	Body	Список фильтров для поиска



Filter			
Название параметра	Тип данных	Обязательное	Описание
field	String	Да	Поле для фильтрации (например, "surname", "faculty", "cardNumber")
operation	String	Да	Операция фильтрации (equals, like, greater, less, between и т.д.)
value	Object	Условно	Первое значение для сравнения (обязательно для большинства операций)
value2	Object	Нет	Второе значение (используется для операций between, not between)

Выходные параметры

Название параметра	Тип данных	Обязательное	Размещение	Описание
content	List<StudentDto>	Да	Body	Коллекция студентов, подходящих под условия поиска
page	PageDto	Да	Body	Информация о странице

Сообщения об ошибках

HTTP-код ответа	Описание ошибки	Комментарий
500	Внутренняя ошибка сервера	Ошибка при работе с базой данных

Описание сценария

Базовый сценарий. Поиск студентов по фильтру

№	Действие	АС
1	Система извлекает параметры запроса (page, size, sort) и тело запроса (filters).	
2	Система формирует условия поиска на основе переданных фильтров: <ul style="list-style-type: none"> <li>- Для каждого фильтра применяется соответствующая операция</li> <li>- Между фильтрами применяется логическое И</li> <li>- Если filters пуст, возвращаются все записи</li> </ul>	
3	Система осуществляет поиск студентов в базе данных с применением сформированных условий, пагинации и сортировки.	АС.1
4	Система заполняет результат работы метода аналогично методу getStudents.	
5	Результат работы метода возвращается запрашивающей системе с http-кодом 200.	
6	Сценарий завершен.	

### Альтернативные сценарии

#### АС.1 При чтении из базы данных произошла ошибка

№	Действие системы
1	Система возвращает ответ с http-кодом 500.
2	Сценарий завершен.

### Обновить студента (полное) - updateStudent

Параметр	Значение
Название сценария	Обновить студента (полное обновление)
Название метода	updateStudent
Эндпоинт	{host}/api/students/{id}
Глагол	PUT
Описание	Метод полностью обновляет информацию о студенте
Ожидаемый результат	Возвращена информация об обновленном студенте ИЛИ

Параметр	Значение
	Возвращено сообщение об ошибке
Ограничения	Отсутствуют

Входные параметры

Название параметра	Тип данных	Обязательное	Размещение	Описание
id	String (UUID)	Да	Path	Идентификатор студента
name	String	Да	Body	Имя студента
surname	String	Да	Body	Фамилия студента
middleName	String	Нет	Body	Отчество студента
cardNumber	Long	Да	Body	Номер студенческого билета
faculty	String	Да	Body	Факультет
department	String	Нет	Body	Кафедра
studyGroup	String	Нет	Body	Учебная группа
email	String	Нет	Body	Электронная почта
phone	String	Нет	Body	Номер телефона

Выходные параметры

Название параметра	Тип данных	Обязательное	Размещение	Описание
student	StudentDto	Да	Body	Информация об обновленном студенте

Сообщения об ошибках

HTTP-код ответа	Описание ошибки	Комментарий
400	Ошибка валидации данных	Не заполнены обязательные поля или данные невалидны
404	Студент не найден	Студент с указанным идентификатором не существует
500	Внутренняя ошибка сервера	Ошибка при работе с базой данных

## Описание сценария

### Базовый сценарий. Обновить студента (полное обновление)

№	Действие	АС
1	Система извлекает идентификатор из пути и данные из тела запроса.	
2	Система проверяет валидность всех переданных значений (аналогично созданию).	АС.1
3	Система осуществляет поиск студента в базе данных при условии <code>student.id = id</code> .	АС.2
4	Система проверяет, что результат поиска с шага 3 не пуст.	АС.3
5	Система обновляет запись в таблице <code>student</code> , полностью заменяя все поля новыми значениями.	АС.2
6	Система заполняет результат работы метода данными обновленного студента.	
7	Результат работы метода возвращается запрашивающей системе с <code>http</code> -кодом 200.	
8	Сценарий завершен.	

### Альтернативные сценарии

#### АС.1 Данные не прошли валидацию

№	Действие системы
1	Система возвращает ответ с <code>http</code> -кодом 400.
2	Сценарий завершен.

#### АС.2 При работе с базой данных произошла ошибка

№	Действие системы
1	Система возвращает ответ с <code>http</code> -кодом 500.
2	Сценарий завершен.

#### АС.3 Студент по переданному идентификатору не найден

№	Действие системы
1	Система возвращает ответ с http-кодом 404.
2	Сценарий завершен.

Обновить студента (частичное) - patchStudent

Параметр	Значение
Название сценария	Обновить студента (частичное обновление)
Название метода	patchStudent
Эндпоинт	{host}/api/students/{id}
Глагол	PATCH
Описание	Метод частично обновляет информацию о студенте
Ожидаемый результат	Возвращена информация об обновленном студенте ИЛИ Возвращено сообщение об ошибке
Ограничения	Отсутствуют

Входные параметры

Название параметра	Тип данных	Обязательное	Размещение	Описание
id	String (UUID)	Да	Path	Идентификатор студента
name	String	Нет	Body	Имя студента
surname	String	Нет	Body	Фамилия студента
middleName	String	Нет	Body	Отчество студента
cardNumber	Long	Нет	Body	Номер студенческого билета
faculty	String	Нет	Body	Факультет
department	String	Нет	Body	Кафедра
studyGroup	String	Нет	Body	Учебная группа

Название параметра	Тип данных	Обязательное	Размещение	Описание
email	String	Нет	Body	Электронная почта
phone	String	Нет	Body	Номер телефона

Выходные параметры

Название параметра	Тип данных	Обязательное	Размещение	Описание
student	StudentDto	Да	Body	Информация об обновленном студенте

Сообщения об ошибках

HTTP-код ответа	Описание ошибки	Комментарий
400	Ошибка валидации данных	Переданные данные невалидны
404	Студент не найден	Студент с указанным идентификатором не существует
500	Внутренняя ошибка сервера	Ошибка при работе с базой данных

Описание сценария

Базовый сценарий. Обновить студента (частичное обновление)

№	Действие	АС
1	Система извлекает идентификатор из пути и данные из тела запроса.	
2	Система осуществляет поиск студента в базе данных при условии student.id = id.	АС.1
3	Система проверяет, что результат поиска с шага 2 не пуст.	АС.2
4	Система проверяет валидность переданных значений.	АС.3
5	Система обновляет только те поля записи, которые были переданы в теле запроса. Примечание: Если поле не передано, его значение остается неизменным.	АС.1
6	Система заполняет результат работы метода данными обновленного студента.	
7	Результат работы метода возвращается запрашивающей системе с http-кодом 200.	
8	Сценарий завершен.	

Альтернативные сценарии

АС.1 При работе с базой данных произошла ошибка

№	Действие системы
1	Система возвращает ответ с http-кодом 500.
2	Сценарий завершен.

АС.2 Студент по переданному идентификатору не найден

№	Действие системы
1	Система возвращает ответ с http-кодом 404.
2	Сценарий завершен.

АС.3 Данные не прошли валидацию

№	Действие системы
1	Система возвращает ответ с http-кодом 400.
2	Сценарий завершен.

Удалить студента - deleteStudent

Параметр	Значение
Название сценария	Удалить студента
Название метода	deleteStudent
Эндпоинт	{host}/api/students/{id}
Глагол	DELETE
Описание	Метод удаляет студента по идентификатору
Ожидаемый результат	Студент удален ИЛИ Возвращено сообщение об ошибке

Параметр	Значение
Ограничения	Отсутствуют

Входные параметры

Название параметра	Тип данных	Обязательное	Размещение	Описание
id	String (UUID)	Да	Path	Идентификатор студента

Выходные параметры

В случае успешного удаления возвращается пустое тело с http-кодом 204.

Сообщения об ошибках

HTTP-код ответа	Описание ошибки	Комментарий
404	Студент не найден	Студент с указанным идентификатором не существует (опционально, зависит от реализации)
500	Внутренняя ошибка сервера	Ошибка при работе с базой данных

Описание сценария

Базовый сценарий. Удалить студента

№	Действие	АС
1	Система извлекает идентификатор из пути запроса.	
2	Система осуществляет удаление студента из базы данных при условии student.id = id.	АС.1
3	Результат работы метода возвращается запрашивающей системе с http-кодом 204.	
4	Сценарий завершен.	

Альтернативные сценарии

АС.1 При удалении из базы данных произошла ошибка



№	Действие системы
1	Система возвращает ответ с http-кодом 500.
2	Сценарий завершен.

## Аналогичные методы для других сущностей

Для сущностей **Competition**, **Participation**, **Level** и **Result** применяются аналогичные методы со следующими операциями:

### Competitions (Соревнования)

- **POST** /api/competitions - Создать соревнование
- **GET** /api/competitions/{id} - Получить соревнование по идентификатору
- **GET** /api/competitions - Получить список соревнований (с пагинацией и сортировкой)
- **POST** /api/competitions/search - Поиск соревнований по фильтру
- **PUT** /api/competitions/{id} - Полное обновление соревнования
- **PATCH** /api/competitions/{id} - Частичное обновление соревнования
- **DELETE** /api/competitions/{id} - Удалить соревнование

### Participations (Участия)

- **POST** /api/participations - Создать запись об участии
- **GET** /api/participations/{id} - Получить участие по идентификатору
- **GET** /api/participations - Получить список участий (с пагинацией и сортировкой)
- **POST** /api/participations/search - Поиск участий по фильтру
- **PUT** /api/participations/{id} - Полное обновление участия (недоступно если isBlocked = true)
- **PATCH** /api/participations/{id} - Частичное обновление участия (недоступно если isBlocked = true)
- **DELETE** /api/participations/{id} - Удалить участие

**Примечание:** Поле isBlocked устанавливается системой и не может быть изменено через API.

### Levels (Уровни)

- **POST** /api/levels - Создать уровень
- **GET** /api/levels/{code} - Получить уровень по коду

- **GET** /api/levels - Получить список уровней (с пагинацией, сортировкой и фильтрацией по name и order)
- **POST** /api/levels/search - Поиск уровней по фильтру
- **PUT** /api/levels/{code} - Полное обновление уровня
- **DELETE** /api/levels/{code} - Удалить уровень

**Примечание:** Для уровней в качестве идентификатора используется поле code (String), а не UUID.

## Results (Результаты)

- **POST** /api/results - Создать результат
- **GET** /api/results/{code} - Получить результат по коду
- **GET** /api/results - Получить список результатов (с пагинацией, сортировкой и фильтрацией по name и order)
- **POST** /api/results/search - Поиск результатов по фильтру
- **PUT** /api/results/{code} - Полное обновление результата
- **DELETE** /api/results/{code} - Удалить результат

**Примечание:** Для результатов в качестве идентификатора используется поле code (String), а не UUID.

---

## Объекты сервиса

### StudentDto

Объект содержит информацию о студенте.

Поле	Тип данных	Обязательное	Описание
id	String (UUID)	Да	Идентификатор студента
name	String	Да	Имя
surname	String	Да	Фамилия
middleName	String	Нет	Отчество
cardNumber	Long	Да	Номер студенческого билета
faculty	String	Да	Факультет
department	String	Нет	Кафедра
studyGroup	String	Нет	Учебная группа
email	String	Нет	Электронная почта
phone	String	Нет	Номер телефона

**CompetitionDto**

Объект содержит информацию о соревновании.

Поле	Тип данных	Обязательное	Описание
id	String (UUID)	Да	Идентификатор соревнования
name	String	Да	Название соревнования
organizer	String	Нет	Организатор
type	String	Да	Тип соревнования (Olympiad, Contest, Championship, Competition)

**ParticipationDto**

Объект содержит информацию об участии студента в соревновании.

Поле	Тип данных	Обязательное	Описание
id	String (UUID)	Да	Идентификатор записи
student	URI	Да	Ссылка на студента
competition	URI	Да	Ссылка на соревнование
level	URI	Да	Ссылка на уровень
result	URI	Да	Ссылка на результат
year	Integer	Да	Год участия
supervisor	String	Нет	Научный руководитель
points	Integer	Нет	Количество баллов
description	String	Нет	Описание
isBlocked	Boolean	Да	Флаг блокировки редактирования (устанавливается системой)

## LevelDto

Объект содержит информацию об уровне соревнования.

Поле	Тип данных	Обязательное	Описание
code	String	Да	Код уровня
name	String	Да	Название уровня
order	Integer	Нет	Порядок сортировки

## ResultDto

Объект содержит информацию о результате участия.

Поле	Тип данных	Обязательное	Описание
code	String	Да	Код результата
name	String	Да	Название результата
order	Integer	Нет	Порядок сортировки

## PageDto

Объект содержит информацию о результатах пагинации.

Поле	Тип данных	Обязательное	Описание
number	Integer	Да	Номер страницы (zero-based)
size	Integer	Да	Количество элементов на текущей странице
sorts	List<SortDto>	Нет	Используемые сортировки
totalElements	Long	Да	Количество всех элементов
totalPages	Integer	Да	Количество всех страниц
first	Boolean	Да	Первая ли страница
last	Boolean	Да	Последняя ли страница
empty	Boolean	Да	Пустая ли страница

## SortDto

Объект содержит информацию о сортировке.

Поле	Тип данных	Обязательное	Описание
property	String	Да	Поле, по которому происходит сортировка
direction	String	Да	Направление сортировки (ASC или DESC)

## FilterChain

Объект используется для задания цепочки фильтров.

Поле	Тип данных	Обязательное	Описание
filters	List<Filter>	Да	Список фильтров

## Filter

Объект используется для задания фильтра.

Поле	Тип данных	Обязательное	Описание
field	String	Да	Поле для фильтрации
operation	String	Да	Операция (equals, like, greater, less, between и т.д.)
value	Object	Условно	Первое значение
value2	Object	Нет	Второе значение (для операций between, not between)

**SheetDto<T>**

Объект используется для получения контента и страницы для сущности T.

Поле	Тип данных	Обязательное	Описание
content	List<T>	Да	Запрашиваемые объекты
page	PageDto	Да	Информация о странице

**Операции фильтрации**

Поддерживаемые операции для фильтров (можно использовать русский аналог):

Операция	Русский аналог	Описание
equals	равно	Точное совпадение
not equals	не равно	Не равно значению
greater	больше	Больше значения
greater or equals	больше или равно	Больше или равно значению
less	меньше	Меньше значения
less or equals	меньше или равно	Меньше или равно значению
like	содержит	Содержит подстроку (без учета регистра)
not like	не содержит	Не содержит подстроку
starts with	начинается с	Начинается с подстроки
not starts with	не начинается с	Не начинается с подстроки
ends with	заканчивается	Заканчивается подстрокой
not ends with	не заканчивается	Не заканчивается подстрокой
is null	не определено	Значение NULL
is not null	определено	Значение не NULL
is empty	пусто	Пустое значение
is not empty	не пусто	Непустое значение
is true	истина	Булево значение true
is false	ложь	Булево значение false
in	в списке	Значение входит в список
not in	не в списке	Значение не входит в список
between	между	Значение между двумя границами
not between	не между	Значение не между двумя границами
before	до	Дата до указанной
after	после	Дата после указанной

Справочник факультетов

Допустимые значения для поля `faculty` (можно передавать русский или английский вариант):

Английский	Русский
Faculty of Biology and Ecology	Факультет биологии и экологии
Faculty of Computer Science and Engineering	Факультет информатики и вычислительной техники
Faculty of History	Исторический факультет
Faculty of Mathematics	Математический факультет
Faculty of Psychology	Факультет психологии
Faculty of Social and Political Sciences	Факультет социально-политических наук
Faculty of Foreign Languages	Факультет иностранных языков
Faculty of Physics and Technology	Физико-технический факультет
Faculty of Philology and Communication	Факультет филологии и коммуникации
Faculty of Economics	Экономический факультет
Faculty of Law	Юридический факультет

Справочник типов соревнований

Допустимые значения для поля `type` в `CompetitionDto` (можно передавать русский или английский вариант):

Английский	Русский
Olympiad	Олимпиада
Contest	Состязание
Championship	Чемпионат
Competition	Соревнование



## Примеры запросов

### Создание студента

http

POST /api/students HTTP/1.1

Content-Type: application/json

```
{
  "name": "Александр",
  "surname": "Слелин",
  "middleName": "Владимирович",
  "cardNumber": 1036575,
  "faculty": "Faculty of Mathematics",
  "studyGroup": "PMI-11MO",
  "email": "a.slelin.work@mail.ru"
}
```

### Получение списка студентов с сортировкой

http

GET /api/students?page=0&size=10&sort=surname,asc&sort=name,asc HTTP/1.1

### Поиск студентов по фильтру

http

POST /api/students/search?page=0&size=10 HTTP/1.1

Content-Type: application/json

```
{
  "filters": [
    {
      "field": "surname",
      "operation": "like",
      "value": "Слелин"
    },
    {
      "field": "cardNumber",
      "operation": "greater or equals",
      "value": 1000000
    }
  ]
}
```

### Создание записи об участии

```
http

POST /api/participations HTTP/1.1
Content-Type: application/json

{
  "student": "http://localhost:8080/api/students/8dd01f45-f038-4d88-8606-5d04714b5ed5",
  "competition": "http://localhost:8080/api/competitions/5dd01f45-f038-4d88-8606-5d04714b5ed5",
  "level": "http://localhost:8080/api/levels/RUS",
  "result": "http://localhost:8080/api/results/WIN",
  "year": 2023,
  "points": 60
}
```

### Получение уровней с фильтрацией

```
http

GET /api/levels?name=Всероссийский&order=3 HTTP/1.1
```

### Формат ошибок

При возникновении ошибки возвращается объект ErrorResponse со следующей структурой:

Поле	Тип данных	Описание
path	String	Путь к endpoint
method	String	HTTP-метод
httpStatus	Integer	HTTP-статус ошибки
debugMessage	String	Сообщения/советы по исправлению ошибки
message	String	Сообщение об ошибке
exception	String	Тип ошибки
causeException	String	Первопричинная ошибка (опционально)
details	Map<String, Object>	Дополнительные детали (опционально)
timestamp	LocalDateTime	Время возникновения ошибки

### Пример ответа с ошибкой (404)

```
json
{
  "path": "/api/students/8dd01f45-f038-4d88-8606-5d04714b5ed5",
  "method": "GET",
  "httpStatus": 404,
  "debugMessage": "Entity not found.",
  "message": "Entity \"Student\" not found by field = \"id\" with value = \"8dd01f45-f038-4d88-8606-5d04714b5ed5\".",
  "exception": "EntityNotFoundException",
  "causeException": null,
  "details": {
    "class": "Student",
    "field": "id",
    "value": "8dd01f45-f038-4d88-8606-5d04714b5ed5"
  },
  "timeStamp": "2025-12-23T16:25:58.083832251"
}
```

### Пример ответа с ошибкой валидации (400)

```
json
```

```
{
  "path": "/api/students",
  "method": "POST",
  "httpStatus": 400,
  "debugMessage": "Validation failed.",
  "message": "Validation failed for classes [a.slelin.work.backend.data.entity.Student]...",
  "exception": "ConstraintViolationException",
  "causeException": null,
  "details": {
    "errors": [
      {
        "field": "email",
        "message": "must be a well-formed email address",
        "invalidValue": "@mail.ru",
        "constraintType": "Email"
      },
      {
        "field": "cardNumber",
        "message": "must be greater than or equal to 0",
        "invalidValue": -34545,
        "constraintType": "Min"
      }
    ]
  },
  "timeStamp": "2025-12-23T16:31:18.641493014"
}
```