#POST /lessons/{id}/reschedule — Перенести занятие

Обновлено 7 декабря 2025, 19:06 Дмитрий Ермаков

Назначение

Переводит урок **x** в статус RESCHEDULED (для всех его Student_Lessons) и **атомарно** создаёт новый урок **y** на указанную дату/время с тем же составом участников.

URL и метод

- Method: POST
- Path: /lessons/{id}/reschedule

Входные параметры

№	Тип параметра (Headers, Query, Body)	Параметр	Тип	Обяз.	Ограничения	Описание	Пример
1	Headers	Authorization	string	да	Bearer <token>	JWT токен пользователя (учитель)	Bearer eyJhbGciOi...
2	Path	id	integer	да	id > 0	Идентификатор урока, который нужно перенести	601
3	Body	dateStart	string	да	YYYY-MM-DD	Новая дата урока (локальная дата пользователя)	2025-11-25
4	Body	timeStart	string	да	HH:mm	Время начала урока (локальное время пользователя)	10:30
5	Body	timeEnd	string	да	HH:mm	Время окончания урока (локальное время пользователя)	11:30
6	Body	description	string	нет	≤256	Новое описание (если нужно)	Перенос

Предикаты/проверки

Правило	Если нарушено
Урок принадлежит другому пользователю	403
Все Student_Lessons урока x имеют статус PLANNED	409
dateEnd > dateStart, корректный ISO	422
Ресурс (урок не найден)	422/404

Пример запроса

```
POST /lessons/601/reschedule HTTP/1.1
Authorization: Bearer <token>
Content-Type: application/json

{
  "dateStart": "2025-11-25",
  "timeStart": "10:30",
  "timeEnd": "11:30"
}
```

Ответы

201 Created

```
{
 "id": 650,
 "seriesId": 501,
 "dateStart": "2025-11-05T18:00:00Z",
 "dateEnd": "2025-11-05T18:50:00Z",
 "lessonType": "MULTI",
 "description": "Перенос из-за занятости",
 "isRegular": false,
 "createdDate": "2025-10-22"
}
```


Ошибки

Код	Когда
400	Неверный формат запроса
401	Нет/неверный токен
403	Урок не принадлежит пользователю
404	Урок не найден или указанные studentIds не найдены
409	Урок ≠ PLANNED (перенос невозможен)
422	Валидация дат/полей.

Таблица выходных параметров

Параметр	Тип	Обяз.	Описание
id	integer	да	Идентификатор **нового** созданного занятия.
seriesId	integer	нет	Идентификатор серии, к которой относится занятие. Может быть null, если урок не входит в серию.
dateStart	string	да	Дата и время начала занятия в формате ISO 8601 (UTC), например 2025-11-05T18:00:00Z.
dateEnd	string	да	Дата и время окончания занятия в формате ISO 8601 (UTC), например 2025-11-05T18:50:00Z.
lessonType	string	да	Тип занятия. Например: SOLO — индивидуальное, MULTI — групповое (точный список значений см. в справочнике типов).
description	string | null	нет	Описание занятия / комментарий преподавателя. Может отсутствовать или быть null.
isRegular	boolean	да	Признак того, что занятие было создано как часть регулярной серии (true / false).
createdDate	string	да	Дата создания занятия в формате YYYY-MM-DD (локальная дата пользователя на момент создания).

Формат ответа `POST /lessons/{id}/reschedule`


```
{
    "id": 650,
    "seriesId": 501,
    "dateStart": "2025-11-05T18:00:00Z",
    "dateEnd": "2025-11-05T18:50:00Z",
    "lessonType": "MULTI",
    "description": "Перенос из-за занятости",
    "isRegular": false,
    "createdDate": "2025-10-22"
  }
```

Алгоритм `POST /lessons/{id}/reschedule` (версия «старый + новый»)

0. Вход

Тело запроса (как форма создания):

```
{
  "date": "2025-11-05",
  "timeStart": "18:00",
  "timeEnd": "18:50"
}
```

1. Найти старый урок и проверить права

1.1. Берём `userId` из JWT.
1.2. Ищем в БД урок:

- `id = {id}` (из path),
- `user_id = userId`,
- `is_deleted = false`.

Если не нашли → `404`.

1.3. Проверяем, что урок можно переносить (статус `PLANNED`):

- если статус не подходит → `409` / `422` (ошибка «нельзя перенести урок в этом статусе»).

2. Собрать новый интервал времени (локально)

Из body: `dateStart`, `timeStart`, `timeEnd`.
Берём `tz` юзера из настроек.

2.1. Собираем локальные даты-время:

- `localStart = dateStart + timeStart @ tz`
- `localEndCandidate = dateStart + timeEnd @ tz`

2.2. Определяем, на какой день приходится конец:

- если **`timeEnd > timeStart`** → `localEnd = localEndCandidate` (тот же день),
- если **`timeEnd < timeStart`** → `localEnd = localEndCandidate + 1 day` (через полночь),
- если **`timeEnd == timeStart`** → `422 INVALID_TIME_RANGE`
  (урок нулевой длины — не даём так делать).

2.3. Переводим в UTC:

- `newDateStart = localStart → UTC`,
- `newDateEnd = localEnd → UTC`.

3. Проверка конфликтов

Проверяем, не влезает ли новый слот в существующее расписание:

```
SELECT id
FROM lessons
WHERE user_id = :userId
  AND is_deleted = false
  AND id <> :currentLessonId          -- НЕ сравниваем с самим собой
  AND date_start < :newDateEnd
  AND date_end   > :newDateStart;
  AND lessons.status = 'PLANNED'
```

Примечание: При проверке конфликтов учитывать только уроки со статусом `PLANNED` (не учитывать `RESCHEDULED`, `SKIPPED`).

- Если список **пустой** → ок, идём дальше.

- Если **есть конфликты** →
  собираем:

```
{
  "error": "LESSON_CONFLICT",
  "message": "Конфликт уроков в расписании",
  "conflicts": [
    {
      "dateStart": "2025-11-05T18:00:00Z",
      "dateEnd": "2025-11-05T18:50:00Z",
      "conflictWith": [123, 124]
    }
  ]
}
```

  и возвращаем `409`, **ничего не меняя в БД**.

---

4. Создать новый урок и обновить старый (в транзакции)

4.1. `BEGIN;`

4.2. **Обновляем старый урок**:

- в `Lessons`:
  - `status` → `RESCHEDULED`
- в `Student_Lessons` для этого урока:
  - `status` → `RESCHEDULED`.

4.3. **Создаём новый урок**:

- в `Lessons` вставляем новую запись:
  - `user_id = userId`,
  - `series_id`:
    - либо копируем `seriesId` старого,
    - либо, если его не было, можно взять `seriesId = старый id`,
  - `date_start = newDateStart`, при этом новая дата создания урока должна быть >= текущего времени `(User.LocalDateTime)` (иначе возвращаем 422 ошибку ,см таблицу ошибок)
  - `date_end = newDateEnd`,
  - `lessonType`, `description` и т.п. копируем из старого или берём из body,
  - `isRegular` — false
  - `createdDate` — `userNow.toISODate` или текущая дата.

Получаем `newLessonId`.

4.4. **Копируем студентов**:

- Берём всех студентов из `Student_Lessons`, привязанных к старому уроку (`status` был `PLANNED`, стал `RESCHEDULED`).
- Для каждого студента создаём запись в `Student_Lessons` для `newLessonId` со статусом `PLANNED`.

4.5. `COMMIT;`

---

5 Создаем транзакцию

Текущий \ Новый	CONDUCTED (−1)	RESCHEDULED (0)	SKIPPED (−1)
PLANNED (0)	−1	0	**−1**
CONDUCTED (−1)	0	**+1**	0
RESCHEDULED (0)	-	-	**-**
SKIPPED (−1)	0	**+1**	0

Маппинг параметров при создании транзакции

Параметр в таблице https://wiki.yandex.ru/homepage/aad767133c7b/arxitektura-prilozhenija/5289b7fa606a/5d9bd74d95d8/database/0f08c89c3d98/	Source/Value	
id	[генерится на бэке]	

Balance_id	[привязаный счет к ученику]	
Type (manual/lesson)	Тип транзакции • manual (транзакция была создана пользователем "ручное списание/ пополнение") • lesson (урок)	
Direction (+ / -)	в зависимости от матрицы • -1 = CREDIT • 0 = NEUTRAL • 1 = DEBIT	
Comment	Смена статуса с [status x0] to [status x] где статус x0 - статус до изменения стаутс x - статус после изменения	
Amount	см матрицу (1 или 0)	
User_id	[привязаный учитель к ученику]	
Create_Date	[дата создания на бэке]	

6. Ответ

Формируем и возвращаем JSON:

```
    {
  "id": 650,
  "seriesId": 501,
  "dateStart": "2025-11-05T18:00:00Z",
  "dateEnd": "2025-11-05T18:50:00Z",
  "lessonType": "MULTI",
  "description": "Перенос из-за занятости",
  "isRegular": false,
  "createdDate": "2025-10-22"
}
```

Сайд-эффекты

- На шаге 3 генерируются student_lesson.status_changed (→ транзакция 0)