

GET/profiles/{userId} - получить профиль ученика для МП

Оглавление:

- [Общее описание](#)
 - [Доступы и ограничения](#)
- [Запрос](#)
 - [Пример запроса:](#)
 - [Тело запроса \(Body\).](#)
 - [Пример ответа - Успех](#)
 - [Пример ответа - Ошибка авторизации](#)
 - [Пример ответа - Запись не найдена](#)
- [Требования к реализации](#)
- [Маппинг данных](#)

История изменений

Список ссылок (ссылка) на задачи в Jira.
В основном: 1 метод = 1 задача = 1
Confluence.

Связанные документы (внутренние)

Тест-кейсы, описания БД и другие статьи
по проекту, которыми пользовались в
ходе разработки

Общее описание

Метод GET `https://api.uverse.com/students/v1/profiles/{studentId}`.

Предназначен для авторизованных пользователей в системе U-Verse, при использовании мобильного приложения. С помощью этого метода можно получить подробную информацию о студенте.

Входными данными будет служить `{studentId}` передаваемый в URL-адресе запроса.

Ожидаемым результатом будет `JSON-объект` с информацией о профиле студента.

Доступы и ограничения

Метод доступен только авторизованным пользователям с ролями преподаватель, администратор, студент, модератор.

Для доступа к методу пользователей с ролью студент, нужна покупка не менее одного платного/бесплатного курса.

Запрос

Method type: GET

BaseURL: `https://api.uverse.com`

API name:

- students

API version: v1

Method name: `/profiles/{studentId}`

Authorization:

- Тип: Token (Bearer Token)
- Обязательная.
- Роли: Студент
- Для роли студент доступна после приобретения не менее одного курса в платформе.

Входные данные:

HTTP params: *HEADER, PATH*

Выходные данные:

BODY

Headers, Cookies:

- стандартные

Пример запроса:

```
1 GET https://api.uverse.com/students/v1/profiles/5629636a-ae0c-4c7f-a319-68c8bf3df3dd
```

query-parametrs:

- нет

Тело запроса (Body)

- не поддерживается

Пример ответа - Успех

Код HTTP: 200

Тело ответа (Body):

```
1 {
```

```
2  "id": "4242c5e0-e348-4fd0-bdf9-9abd5182f220",
3  "firstName": "Максим",
4  "lastName": "Смирнов",
5  "middleName": null,
6  "avatarUrl": "/images/profileStudent/profileStudentId/avatar/name.png",
7  "country": {
8    "id": "92e9a899-c126-4f05-b25c-7329d15eecbd",
9    "name": "Россия"
10 },
11 "birthday": "2000-12-31",
12 "gender": 0,
13 "city": {
14   "id": "e778484c-6936-493f-846d-f37dcb0e3b14",
15   "name": "Perm"
16 },
17 "phone": "79631258596",
18 "email": "tghryu/87@bk.ru",
19 "aboutMe": "Text"
20 }
```

Пример ответа - Не правильный запрос

Код HTTP: 400

```
1  {
2    "error": {
3      "text": "Bad Request",
4      "message": "Проверьте правильность написания идентификатора пользователя"
5    }
6  }
```

Код HTTP: 401

Тело ответа (Body):

```
1  {
2    "error": {
3      "text": "Unauthorized",
4      "message": "Время действия токена истекло или он введен не корректно"
5    }
6  }
```

Пример ответа - Запись не найдена

Код HTTP: 404

Тело ответа (Body):

```
1  {
2    "error": {
3      "text": "Not Found",
4      "message": "Пользователь не найден"
5    }
6  }
```

Пример ответа - Сервер не доступен

Код HTTP: 503

Тело ответа (Body):

```
1 {  
2   "error":{  
3     "text": "Service Unavailable",  
4     "message": "Сервер не доступен, попробуйте еще раз"  
5   }  
6 }
```

Требования к реализации

Алгоритм работы:

1. Пользователь нажимает на мой профиль для получения информации о студенте.
2. **Frontend** дергает метод `GET`
`https://api.uverse.com/students/v1/profiles/{studentId}`
3. **Backend** проверяет наличие токена доступа в заголовке `Authorization` :
 - a. Если токена нет в наличии пользователю возвращаем ошибку HTTP 401 текст смотри выше;
 - b. Если с токен в наличие и он валиден идем дальше.
4. **Backend** проверяет валидность токена доступа в заголовке `Authorization` :
 - a. Если токен не валиден, пользователю возвращаем ошибку HTTP 401 текст смотри выше;
 - b. Если с токен в наличие и он валиден идем дальше.
5. **Backend** проверяет `{studentId}` на корректность:
 - a. Если `{studentId}` не корректен, пользователю возвращаем ошибку HTTP 400, текст смотри выше;
 - b. Если `{studentId}` корректен идем дальше.
6. После проведения проверок **Backend** отправляет SQL запрос к **БД** для получения данных.

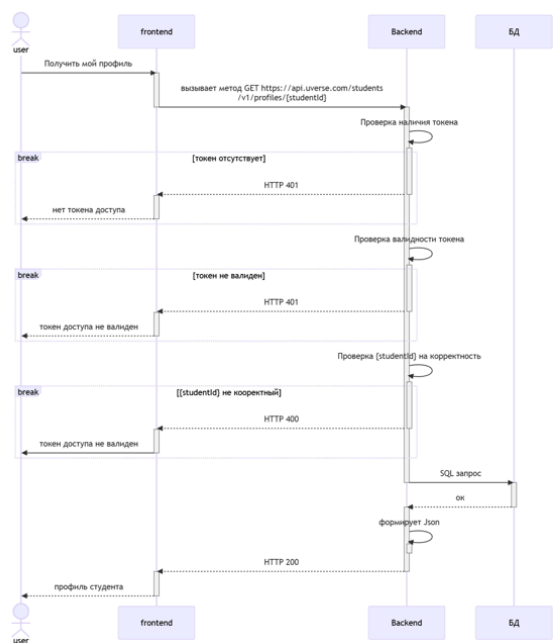
```
1 SELECT  
2 * from  
3   student s  
4 JOIN  
5   "country" ctry ON s."country_id" = ctry."id"  
6 JOIN  
7   "city" city ON s."city_id" = city."id"  
8 WHERE  
9   s."id" = {userId};
```

- 7. **БД** возвращает данные на **Backend**.
- 8. **Backend** формирует JSON (описан выше).
- 9. **Backend** возвращает его на **Frontend**.
- 10. **Frontend** отображает данные пользователю с код ответом HTTP 200, описан выше.

Обработка исключительных ситуаций:

- 1. Если при ответе на запрос поля будут пустыми, возвращать пустое поле в ответе.
- 2. Если id пользователя валиден, но такого пользователя нет в системе возвращать http 404? текст описан выше.
- 3. Внутренняя ошибка.

В процессе работы алгоритма не удастся подключиться к БД или происходит внутренняя ошибка. Вернуть пользователю код ошибки HTTP-503 с соответствующим текстом “Сервис временно недоступен”.



Маппинг данных

Для тела ответа:

Параметр в JSON	Описание / комментарий, требования к преобразованию	Обяз.	Тип данны x JSON	В БД

	после извлечения из БД или обработки			
id	Уникальный идентификатор пользователя	+	String	student.id
firstName	Имя пользователя	+	String	student.firstName
lastName	Фамилия пользователя	+	String	student.lastName
middleName	Отчество пользователя	-	String	student.middleName
avatarUrl	Ссылка на аватар пользователя	-	String	student.avatarUrl
country	Информация о стране	+	object	
country.id	уникальный идентификатор страны	+	String	student_country.id
country.name	Наименование страны	+	String	country.name
birthday	День рождения	+	String	student.birthday
gender	Пол	+	Number	student.gender
city	Город	+	object	
city.id	уникальный идентификатор города	+	String	student_sity.id
city.name	Наименование города	+	String	sity.name

phone	Номер телефона	+	String	student.phone
email	Адрес электронной почты	+	String	student.email
aboutMe	Информация о себе	+	String	student.aboutMe