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

Оглавление:

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 использование 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:

```
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):

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

Пример ответа - Не правильный запрос

Код HTTP: 400

```
{
  "error": {
      "text": "Bad Request",
      "message": "Проверьте правильность написания идентификатора пользователя"
  }
}
```

Код HTTP: 401

Тело ответа (Body):

```
{
  "error": {
    "text": "Unauthorized",
    "message": "Время действия токена истекло или он введен не корректно"
    }
}
```

Пример ответа - Запись не найдена

Код HTTP: 404

Тело ответа (Body):

```
{
```

```
  "error": {
     "text": "Not Found",
     "message": "Пользователь не найден"
  }
}
```

Пример ответа - Сервер не доступен

Код HTTP: 503

Тело ответа (Body):

```
{
  "error":{
     "text": "Service Unavailable",
     "message": "Сервер не доступен, попробуйте еще раз"
     }
}
```

Требования к реализации

Алгоритм работы:

1. Пользователь наживает 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** проверят `{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.

```
SELECT
```

```
* from
    student s
JOIN
    "country" ctry ON s."country_id" = ctry."id"
JOIN
    "city" city ON s."city_id" = city."id"
WHERE
    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 удается подключиться к БД или происходит внутренняя ошибка. Вернуть пользователю код ошибки HTTP-503 с соответствующим текстом “Сервис временно недоступен”.

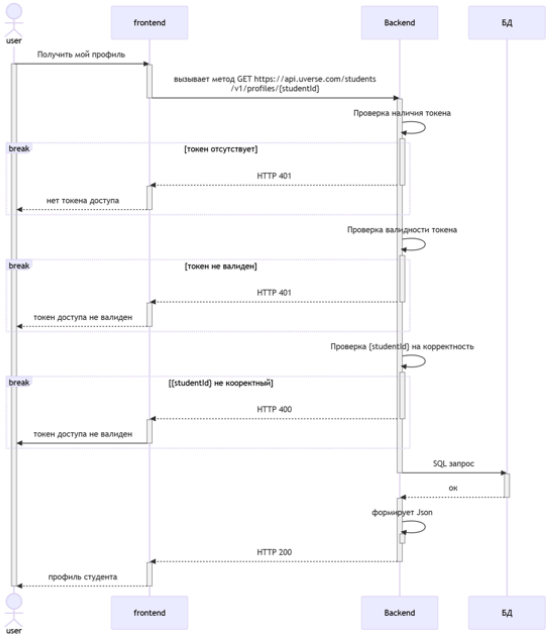
UML-sequence

код для mermaid диаграммы получения профиля студента

```
sequenceDiagram
    actor user
 user->>frontend: Получить мой профиль
activate user
 activate frontend
frontend->>Backend: вызывает метод GET https://api.uverse.com/students<br/>/v1/profiles/{s
deactivate frontend
activate Backend
Backend->>Backend: Проверка наличия токена
activate Backend
break токен отсутствует
Backend-->>frontend: HTTP 401
deactivate Backend
activate frontend
frontend-->>user: нет токена доступа
deactivate frontend
end
Backend->>Backend: Проверка валидности токена
activate Backend
break токен не валиден
Backend-->>frontend: HTTP 401
deactivate Backend
activate frontend
frontend-->>user: токен доступа не валиден
```

```
deactivate frontend
end
Backend->>Backend: Проверка {studentId} на корректность
activate Backend
break {studentId} не коорректный
Backend-->>frontend: HTTP 400
deactivate Backend
activate frontend
frontend->>user: токен доступа не валиден
deactivate frontend
end
Backend->>БД:  SQL запрос
deactivate Backend
activate БД
БД-->>Backend: ок
deactivate БД
activate Backend
Backend->>Backend: формирует Json
activate Backend
deactivate Backend
Backend-->>frontend: HTTP 200
deactivate Backend
activate frontend
frontend-->>user: профиль студента
deactivate frontend
deactivate user

```



Маппинг данных

Для тела ответа:

Параметр в JSON	Описание / комментарий, требования к преобразованию после извлечения из БД или обработки	Обяз.	Тип данных JSON	В БД
`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