Проект для Apple про деревья

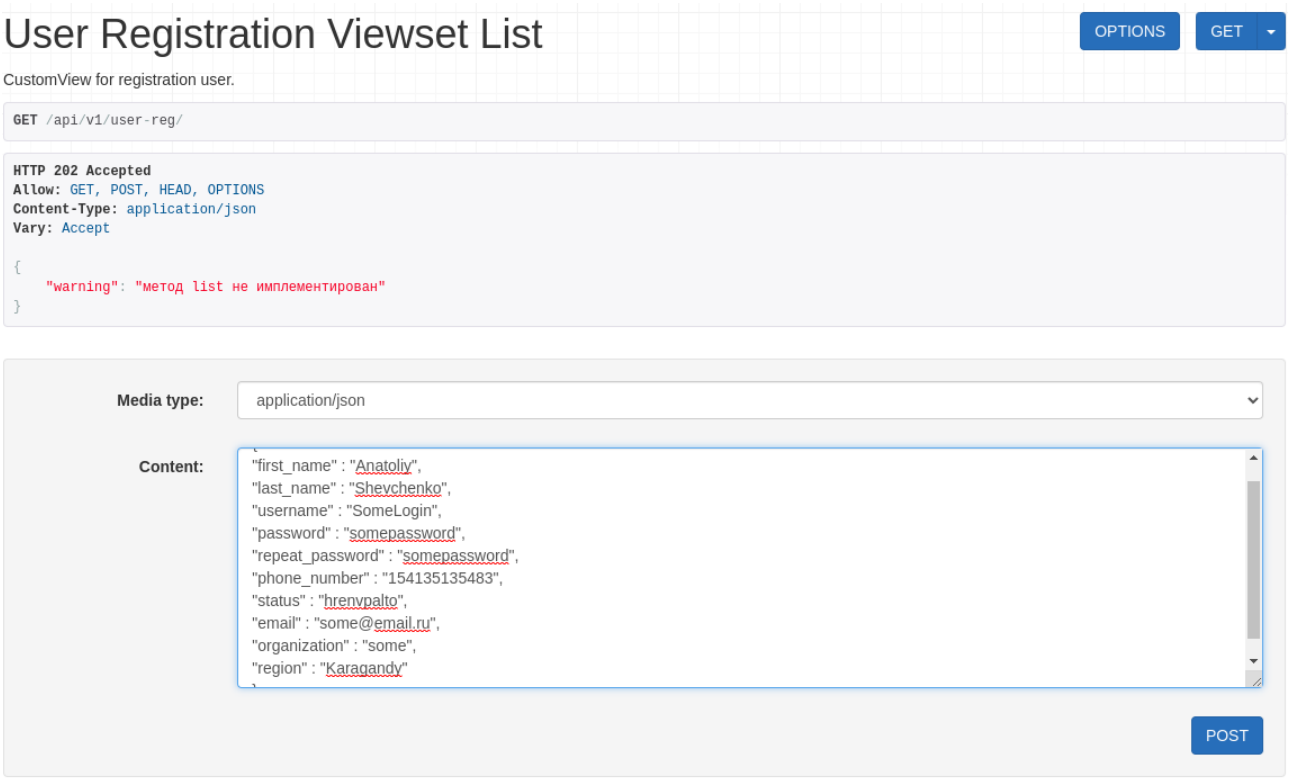
для начала нам понадобится хост: **http://34.22.215.91/**
и будут эндпоинты:

Авторизация:
«api/token/» - эндпоинт так же принимает POST запрос с полями:
«username» - имя пользователя
«password» - пароль аккаунта
В случае если пользователя с указанным именем нет в базе, сервер вернет ответ что такого пользователя не существует. Аналогично если не правильный пароль. В случае успешной авторизации сервер вернет access и refresh токены которые надо бы сохранять с localStorage.

«api/token/refresh/» - access token живет 24 часа, в случае когда его действие закончилось можно использовать данный эндпоинт, он принимает только POST запрос с полем «refresh» значением которого должен быть собственно refresh токен который выдается с access токеном при авторизации. Если refresh токен действителен сервер выдаст новый access токен и таким образом по идее пользователь будет всегда авторизован… ну пока действие refresh токена не закончится.

«api/v1/user-reg/» - Вьюшка Регистрации Пользователя.

Принимает POST запрос с полями:
«first_name» - имя пользователя(str)
«last_name» - фамилия пользователя(str)
«username» - username(str) для авторизации, уникальное поле
«password» - пароль(str)
«repeat_password» - повторить пароль(str), там есть кое какая валидация, если пароли не совпадают сервер выдаст исключение.
«phone_number» - номер телефона(str), уникальное поле
«email» - электронная почта(str), уникальное поле
«status» - принимает строку.
«region» - принимает строку.
«**organization» -** название организации, когда пользователь регистрируется происходит проверка, если такой организации нет сервер отдаст ответ «ERROR: manager does not exist». Если организация есть сервер ответит «registration success».



«api/v1/staff-reg/» - Вьюшка Регистрации Куратора.

Принимает POST запрос с полями:

«first_name» - имя пользователя(str)

«last_name» - фамилия пользователя(str)

«username» - username(str) для авторизации, уникальное поле

«password» - пароль(str)

«repeat_password» - повторить пароль(str), там есть кое какая валидация, если пароли не совпадают сервер выдаст исключение.

«phone_number» - номер телефона(str), уникальное поле

«card» - номер карты, принимает строку, ровно 16 символов, уникальное поле

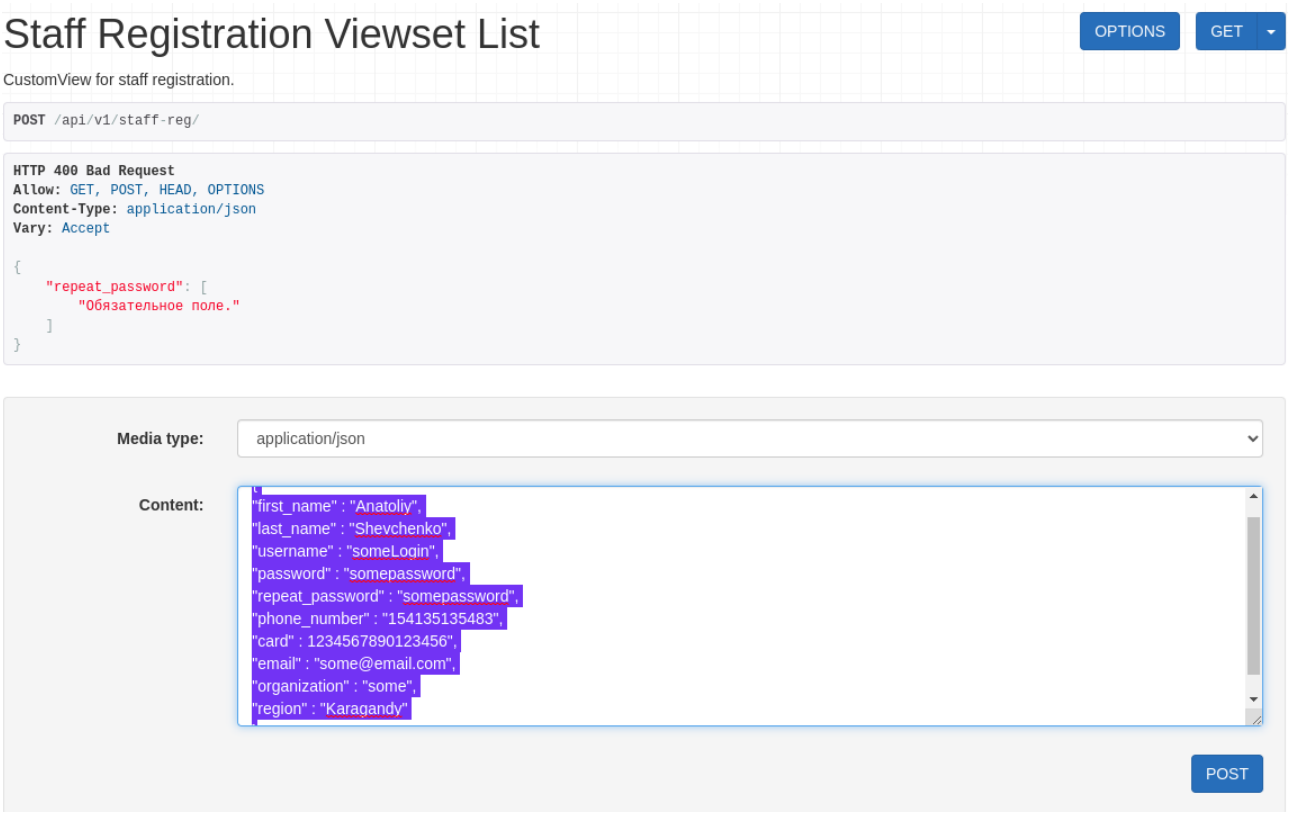
«email» - электронная почта(str), уникальное поле

«status» - принимает строку.

«region» - принимает строку.

«**organization»** - название организации, когда куратор регистрируется происходит проверка если организация с таким названием в указанном регионе уже существует сервер выдаст ответ «ERROR: ORGANIZATION ALREADY EXIST».

В остальных случаях сервер ответит «registration success». Картинка ниже, то есть есть валидация на пароле, repeat_password обязательное поле.



«api/v1/per-cab/» - Личный кабинет. Тут пользователь может посмотреть свой профиль и изменить свои данные или вообще удалить свой профиль, принимает методы:

1) GET метод, просто возвращает данные пользователя, получить данные можно по ключу «user».

2) PATCH метод, можно указать конкретные поля которые надо изменить, абсолютно любые, список полей есть выше в регистрации, сервер ответит «update: success».

3) DELETE метод, удаляет пользователя из базы данных полностью, сервер ответит «success: user has been removed».

Персональная кабина

УДАЛИТЬ ПАРАМЕТРЫ ПОЛУЧИТЬ

Вид для личного кабинета.

```
GET /api/v1/per-cab/
```

```
HTTP 200 OK
Allow: GET, PATCH, DELETE, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "staff": {
        "last_login": "2023-06-14T20:02:06.472739+06:00",
        "first_name": "Anatoliy",
        "last_name": "Shevchenko",
        "username": "someLogin",
        "phone_number": "154135135483",
        "email": "some@email.com",
        "region": "Karagandy",
        "photo": null,
        "organization": "some",
        "card": "1234567890123456",
        "cardholder_name": null
    }
}
```

Тип носителя: application/json

Содержание:

ЗАПЛАТА

Персональная кабина

УДАЛИТЬ ПАРАМЕТРЫ ПОЛУЧИТЬ

Вид для личного кабинета.

```
GET /api/v1/per-cab/
```

```
HTTP 200 OK
Allow: GET, PATCH, DELETE, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "user": {
        "last_login": "2023-06-14T20:05:08.066417+06:00",
        "first_name": "Anatoliy",
        "last_name": "Shevchenko",
        "username": "SOMELogin",
        "phone_number": "4513213518",
        "email": "some@email.ru",
        "status": "aegag",
        "region": "Karagandy",
        "photo": null,
        "coins": 0,
        "organization": "some"
    }
}
```

Тип носителя: application/json

Содержание:

ЗАПЛАТА

«api/v1/my-org/» - Организация, доступ только у кураторов. Тут так же можно посмотреть данные изменить их и удалить организацию, принимает методы:

1) GET метод. Просто возвращает данные организации, получить можно по ключу «my_organization».

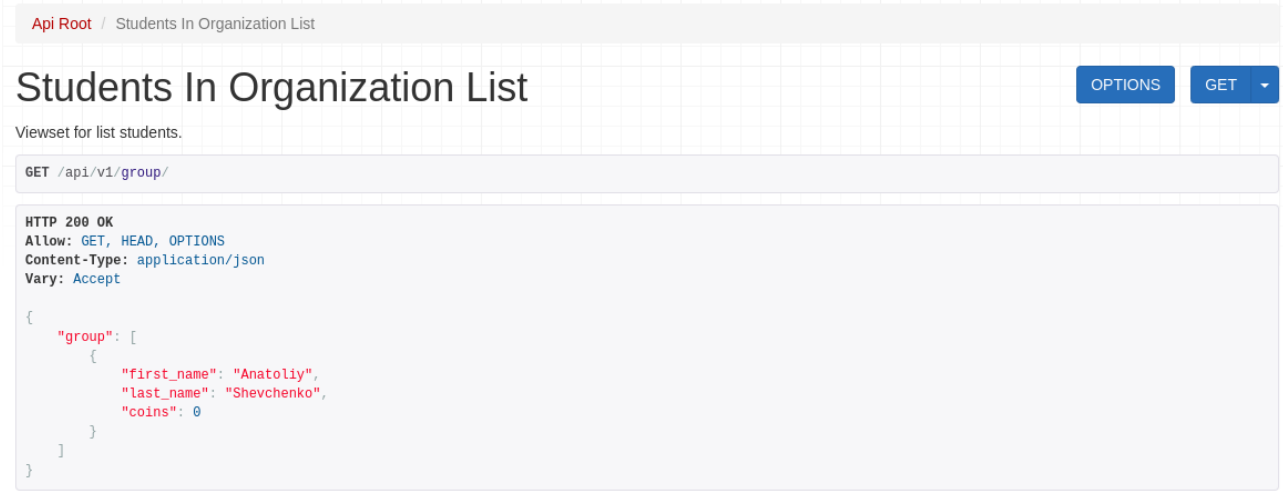
2) PATCH метод. Можно изменить название и/или регион. Не знаю нафига оно надо но может пригодиться, сервер ответит «info: organization updated», принимает поля:

«title» - Название организации(str)

«region» - принимает число, смотри как в регистрации.

3) DELETE метод. Удаляет организацию. Сервер ответит: «info: organization deleted».

«api/v1/group/» - Список только студентов с рейтингом. ТУТ ЕСТЬ ПАГИНАЦИЯ. Достать данные пагинатора такие как счетчик страниц или ссылки на предыдущую или следующую страницу можно по ключу «pagination». Принимает только метод GET. Вы спросите нафига там пагинатор? А вдруг там +100500 студентов и тогда ваше приложение ляжет. Студенты сразу же сортируются по убыванию коинов.



«api/v1/student-report/» - Тут можно создать отчет и посмотреть их список, этот эндпоинт только для студентов. В методе GET есть пагинация, опять же отчетов может быть много и мы пока не решили будем ли мы их удалять, будущее туманно. Тут отчеты только конкретного студента который в данный момент авторизован. Принимает методы:

1) GET метод. Пагинация так же по ключу «pagination». А отчеты по ключу «reports», если отчетов нет то сервер даст ответ «error: reports does not exist».

2) POST метод. Принимает ключи:

«title» - название отчета(str)

«report» - название операции, принимает цифру аналогично с регионом в регистрации, пример:

(1, 'Полив дерева'),
(2, 'Разрыхление дерева'),
(3, 'Побелка дерева'),
(4, 'Удобрение дерева')

«photo» - принимает фотографию, и при загрузке сжимает для экономии места на сервере.

Если все передано правильно сервер ответит «success: report created».

«api/v1/student-report/id:str/» - Тут можно провалиться в конкретный отчет, вместо «id:str» надо подставить id отчета. Принимает методы:

1) GET метод. Покажет отчет студента, полностью, получить можно по ключу «report».

2) PATCH метод. Можно редактировать любые поля отчета. Если поля будут заполнены правильно сервер ответит «success: report updated».

3) DELETE метод. Удаляет отчет. Кстати при удалении отчета удаляется и фото с сервера. Сервер ответит «success: report has been removed».

«api/v1/check-report/» - Это эндпоинт только для кураторов. Принимает только GET метод и возвращает все отчеты студентов своей организации по ключу «reports». Думали всё? **ТУТ ЕСТЬ ПАГИНАЦИЯ!** Получить все данные пагинатора как всегда можно по ключу «pagination». Если отчетов нет сервер ответит «error: reports not found».

«api/v1/check-report/id:str/» - Тут тоже проваливаемся в конкретный отчет, но как вы могли догадаться, тут мы это делаем от имени авторизованного куратора, и да вместо «id:str» надо подставить id отчета.

Принимает методы:

1) GET метод. Показывает полный отчет, получить можно по ключу «report».

2) PATCH метод. Принимает поле:

«action» - в котором надо указать «approved» в случае одобрения отчета и «reject» в случае отклонения отчета. Балы будут начислены в случае одобрения соответственно благодаря отчету где поле report будет 1 из 4 вариантов ну вы видели выше где создание отчета. И сервер вернет соответственно «approved» или «reject» операция будет выполнена.