IvanZaycev0717

Мой GitHub - https://github.com/IvanZaycev0717
Связаться со мной в Telegram - https://t.me/ivanzaycev0717
Видео-версия руководства на канале - https://www.youtube.com/@IvanZaycev_0717

РУКОВОДСТВО ПО РЕАЛИЗАЦИИ АВТОРИЗАЦИИ ЧЕРЕЗ СОЦИАЛЬНЫЕ СЕТИ OAUTH 2.0 В ВЕБ-ФРЕЙМВОРКАХ PYTHON

Google
VK (ВКонтакте)
Одноклассники
Яндекс
Telegram

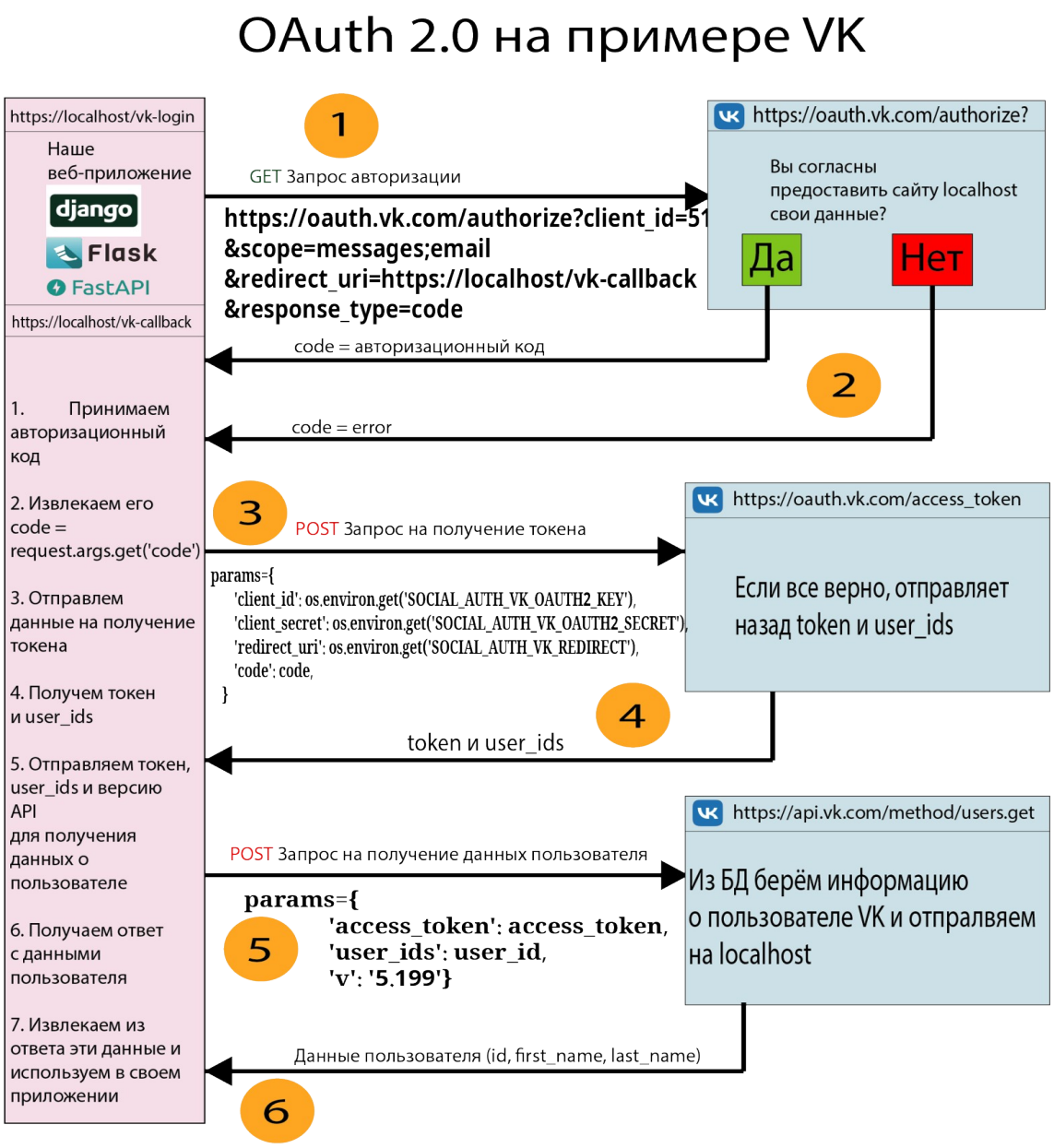
2024

1. Общие положения

Данное руководство может быть использовано для любых веб-фреймворков Python, например, Django, Flask, FastAPI, Starlette и другие.

Конкретно в данном руководстве описан алгоритм авторизации веб-фреймворке Flask.

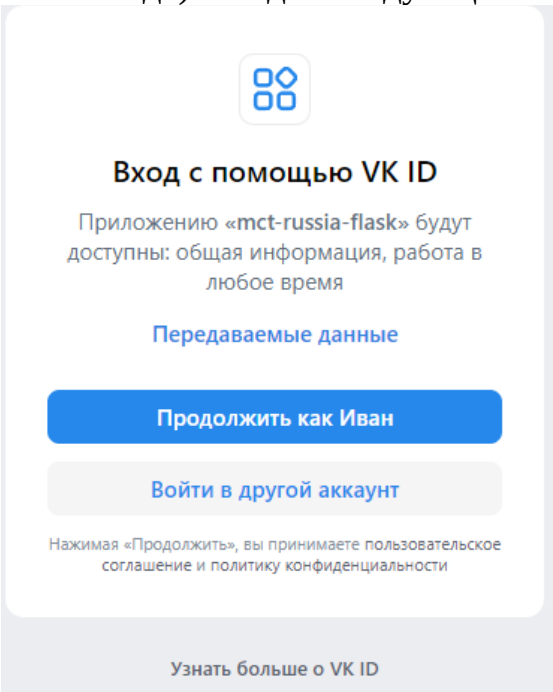
2. Описание процесса авторизации OAuth 2.0



OAuth2 — открытый протокол авторизации, который позволяет пользователям предоставлять сайтам ограниченный доступ к своим данным в других сервисах без передачи логина и пароля. Принцип работы основан на предоставлении временного кода-доступа (токена), который ограничивает права приложения-посредника.

У нас пользователь щелкает на иконку социальной сети, через которую он хочет авторизоваться на нашем сайте. При этом наше приложение отправляет GET-запрос, в URI передается:
- client_id - уникальный идентификатор приложения (об этом подробнее позже)
- scope — запрашиваемые разрешения (доступ к профилю, email и т.д.).
- redirect_uri - URL, на который будет направлен пользователь после авторизации. Несмотря на то, что в названии есть URI, фактически это URL – полный адрес нашего приложения (то есть redirect_uri=https://localhost/vk-callback)
- response_type - тип ответа (обычно code для получения авторизационного кода).

Пользователь перенаправляется по адресу тип ответа (обычно code для получения авторизационного кода) и видит следующее:



Если он нажмет на кнопку «Продолжить как Иван», пользователь будет перенаправлен обратно в веб-приложение, но вместе в ним VK отправит авторизационного код на эндпоинт '/vk-callback'. И здесь наше приложение должно будет отправить еще 2 запроса, прежде чем получить всю информацию о пользователе

Мы получили авторизационный код. Теперь бы должны его извлечь:

```
@auth.route('/vk-callback')
def vk_callback():
  """Fetch a response from VK."""
  code = request.args.get('code')
```

Мы извлекли этот код и теперь мы его должны передать **СТРОГО в POST** запросе на сервер VK для получения токенов (https://oauth.vk.com/access_token).

```
response =
requests.post('https://oauth.vk.com/access_token',
params={
    'client_id':
os.environ.get('SOCIAL_AUTH_VK_OAUTH2_KEY'),
    'client_secret':
os.environ.get('SOCIAL_AUTH_VK_OAUTH2_SECRET'),
    'redirect_uri':
os.environ.get('SOCIAL_AUTH_VK_REDIRECT'),
    'code': code,
  })
```

Теперь ждем ответа от сервиса выдачи токенов ВК. Когда ответ получен, проверяем его статус, если статус 200, то извлекаем токен и id пользователя, чтобы по ним получить информацию о пользователе:

```
if response.status_code == HTTPStatus.OK:
    data = response.json()
    access_token = data['access_token']
    user_id = data['user_id']

    user_info_response = requests.post(
      'https://api.vk.com/method/users.get',
      params={
        'access_token': access_token,
        'user_ids': user_id,
```

```
        'v': '5.199'})
    if user_info_response.status_code == HTTPStatus.OK:
        user_info = user_info_response.json()['response']
[0]
        user_data = {
            'id': user_info.get('id'),
            'first_name': user_info.get('first_name'),
            'last_name': user_info.get('last_name'),
        }
```

Теперь мы можем использовать словарь user_data на свое усмотрение — регистрировать пользователя или дать пользователю залогиниться. Это уже зависит от требований технического задания и логика определяется только вами как разработчиком.

3. Необходимость использования HTTPS и SSL

В процессе разработки мы часто запускаем отладочный сервер Django или Flask по протоколу HTTP. Проблема в том, что некоторые сервисы авторизации через социальные сети (например, Telegram) не работают с этим протоколом. Поэтому нам потребуется, чтобы наше приложение открывалось по протоколу HTTPS. Это легко реализуется на всех фреймворках.

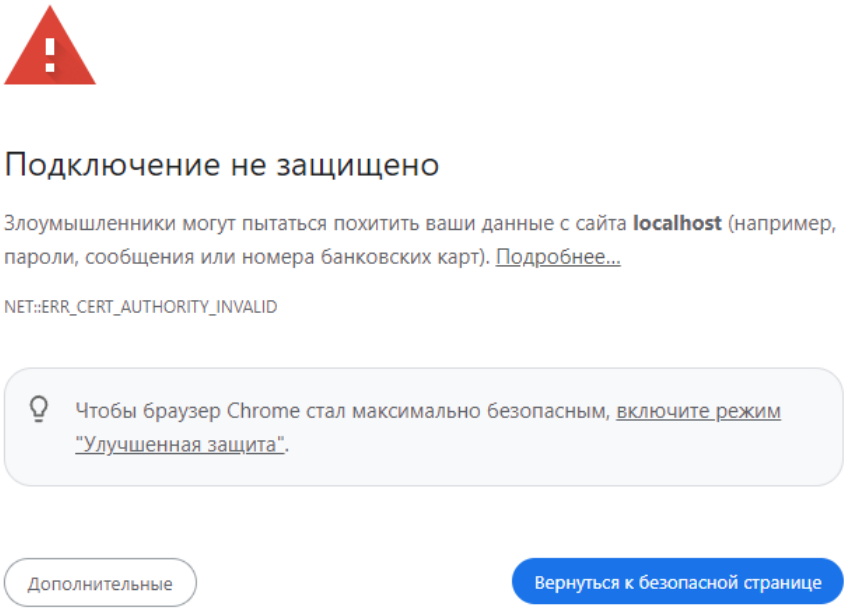
Пример для Flask:
Вначале установим библиотеку

```
pip install pyopenssl
```

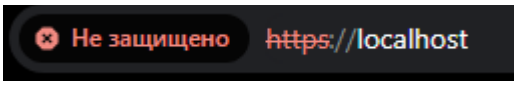
Теперь установим параметры запуска приложения

```
app.run(host='0.0.0.0', port=443, ssl_context='adhoc')
```

Теперь, запустив сервер в строке URL мы должны видеть протокол HTTPS:



Здесь нажимает «Дополнительные» и в появившейся вкладке щелкаем на «Перейти на сайт localhost (небезопасно)»



Обратите внимание на состояние «Не защищено» - это означает, что SSL-сертификат не прошел проверку. Тем не менее у нас HTTPS – а этого вполне достаточно для OAuth.

4. О сторонних библиотеках для авторизации через социальные сети

Для авторизации через Google существуют специальные библиотеки для авторизации через этот сервис. Там все работает хорошо.

Также существуют другие библиотеки для авторизации через социальные сети Python Social Auth или Authlib.

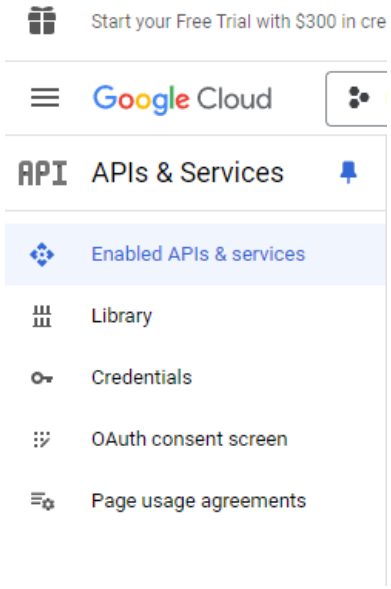
В процессе разработки я ими пытался пользоваться. Но происходил конфликт между сессиями Flask, CSRF-токеном и этими библиотеками. Поэтому я реализовал OAuth 2.0 через GET и POST запросы библиотеки requests.

Наибольший интерес представляет регистрация через Telegram – она ни на что не похожа. Про неё отдельно поговорим.

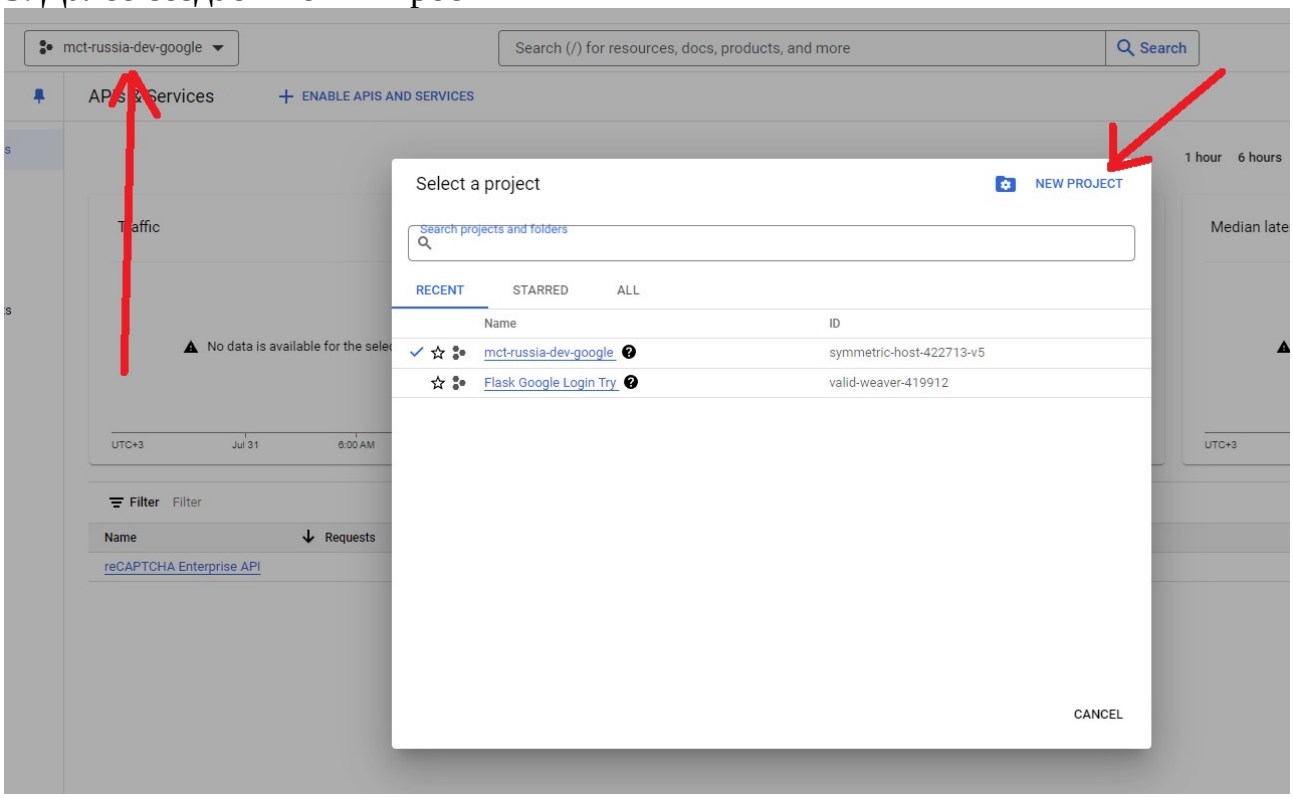
5. Регистрация через Google

Работа с Google выглядит так:

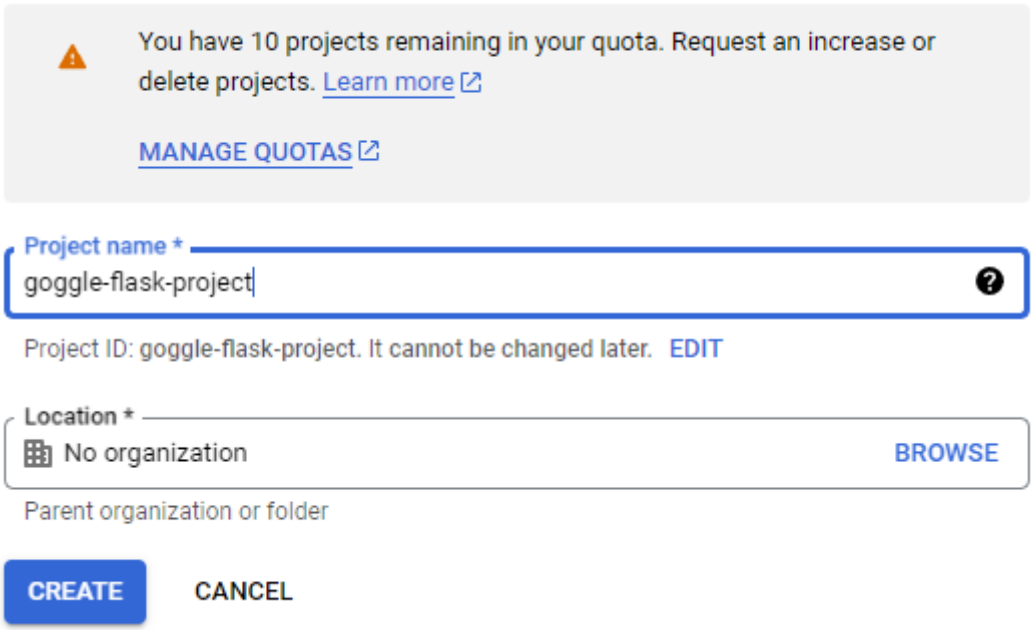
1. Создаем аккаунт Google - https://www.google.com/intl/ru/account/about/
2. Далее переходим в консоль разработчика https://console.cloud.google.com/apis/
Справа должно быть такое меню:



3. Далее создаем новый проект

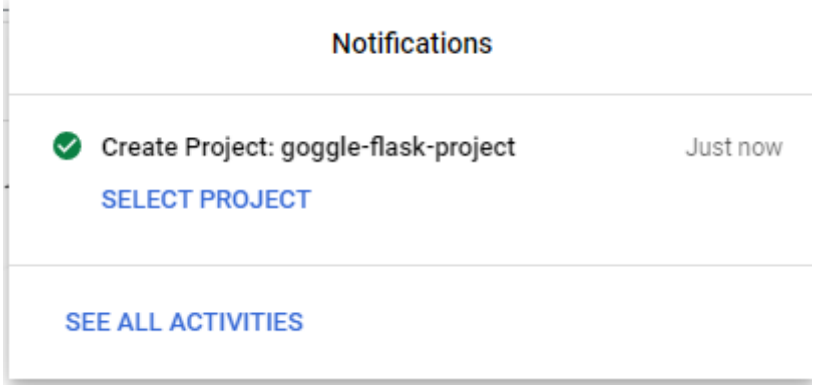


4. Далее вводим название проекта и оставляет пустым поле Location* (No organization)



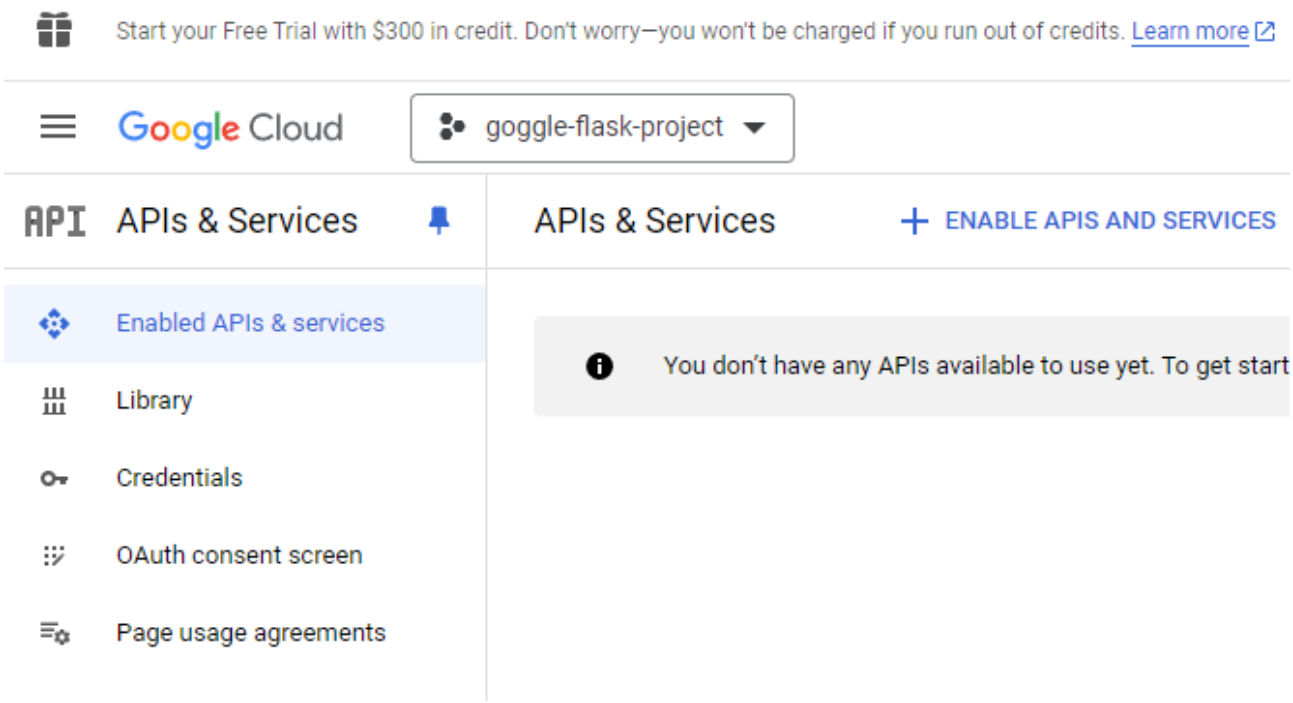
Нажимаем синюю кнопку CREATE

5. Ждем пока создастся проект.



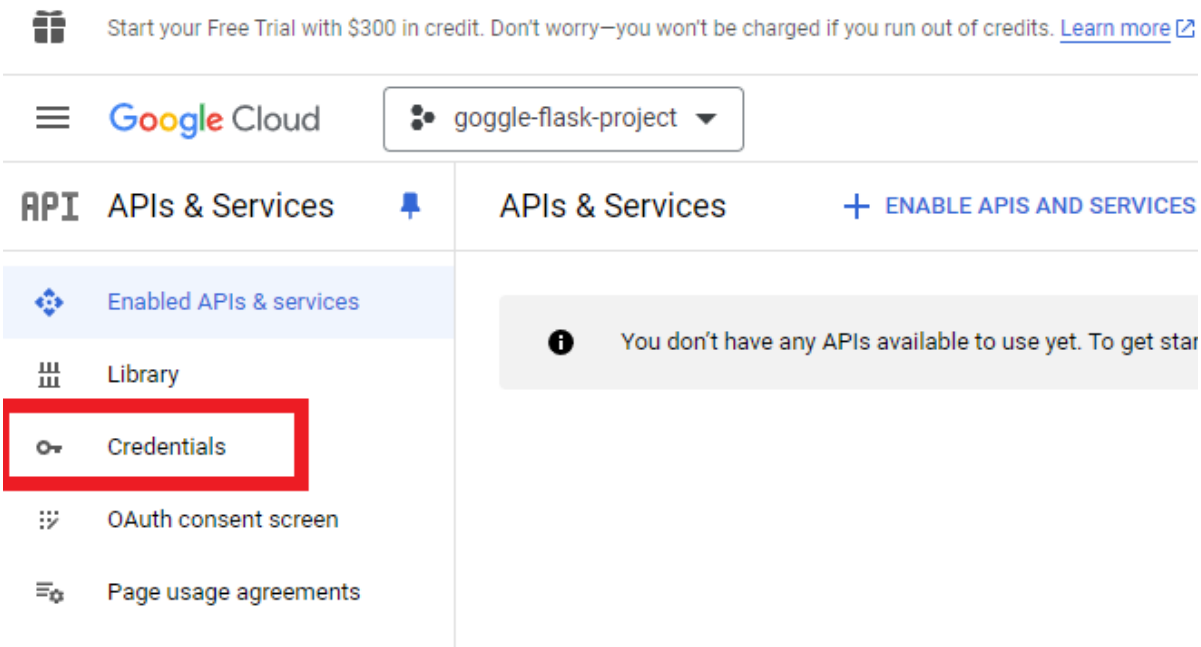
Жмем ссылку SELECT PROJECT под названием проекта

6. Теперь наша страница должна выглядеть так:

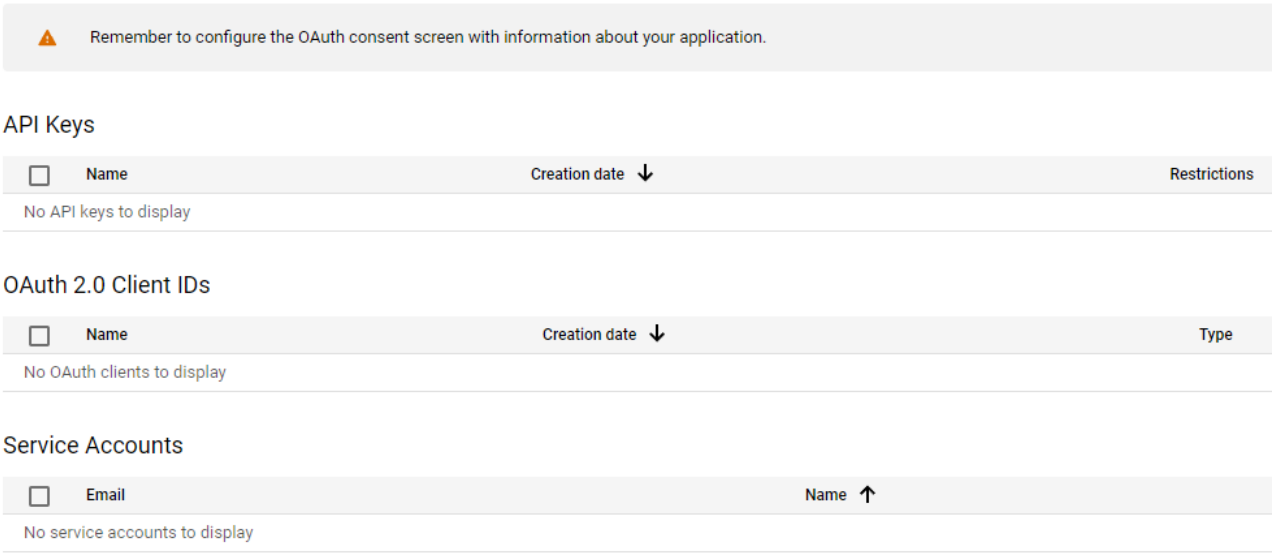


Справа от логотипа Google Cloud название нашего нового проекта

7. Нажимаем на Credentials:

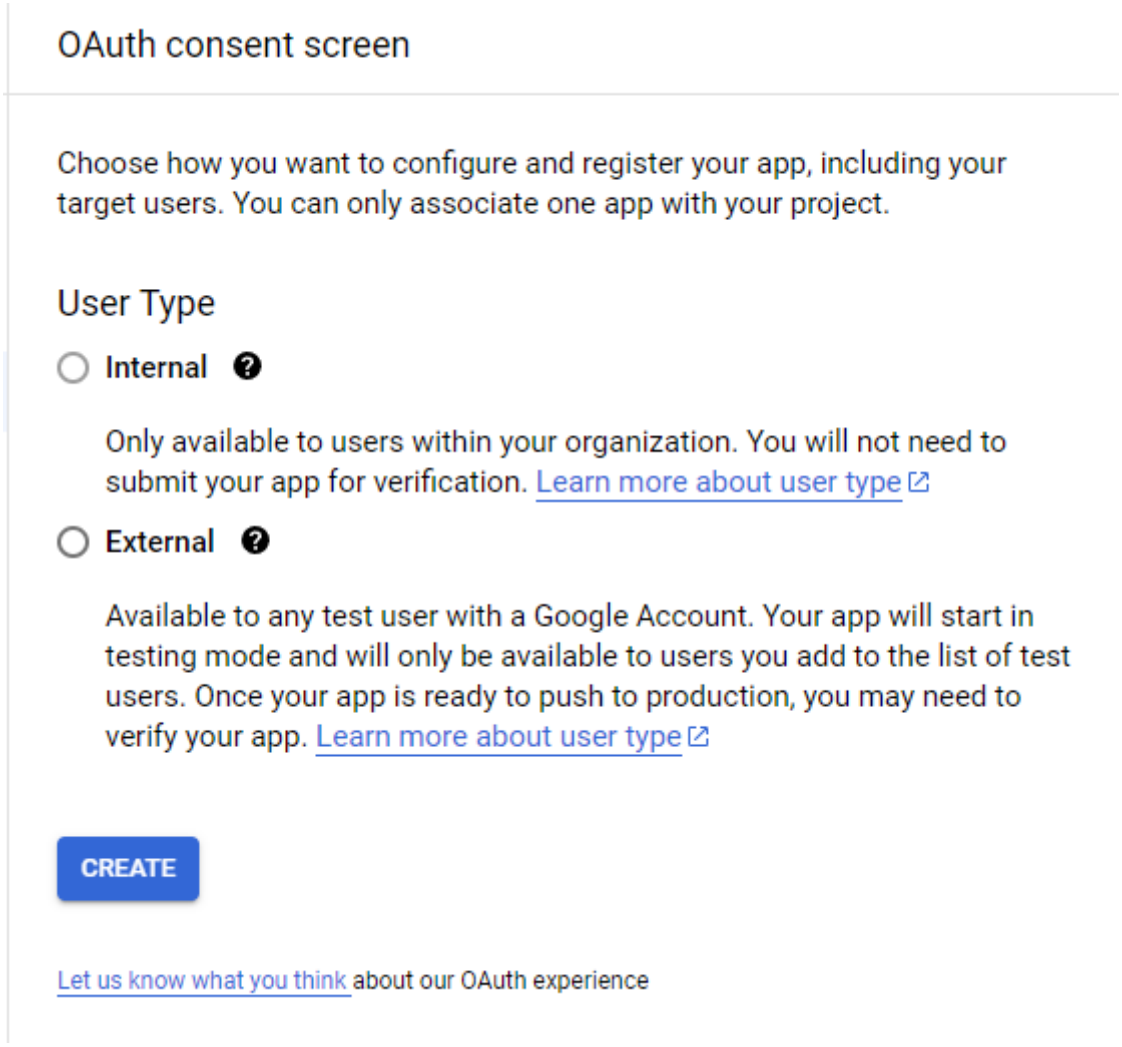


Появится такое окно



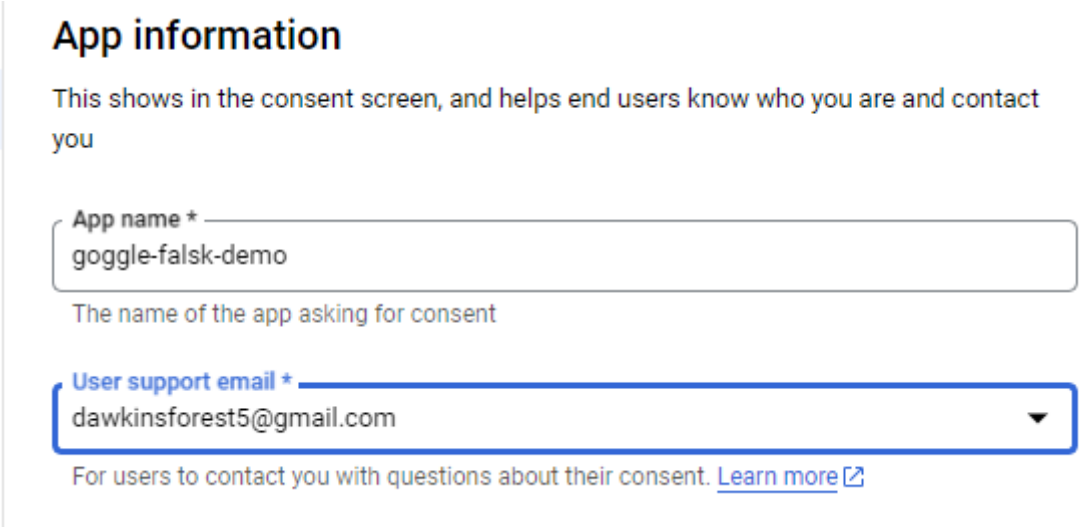
В этом окне слева есть кнопка

Нажимаем на неё
Далее должна появиться такая форма

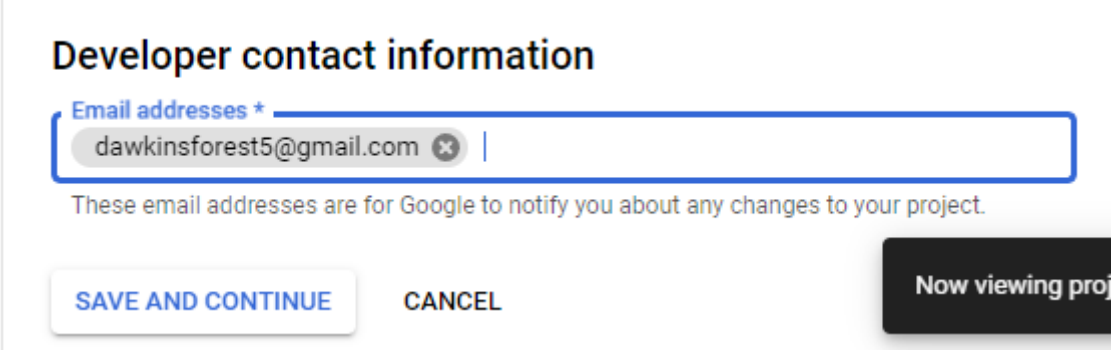


Ставим точку в External и нажимаем CREATE

8. Далее попадаем на страницу App Information
Здесь надо заполнить поля следующим образом:

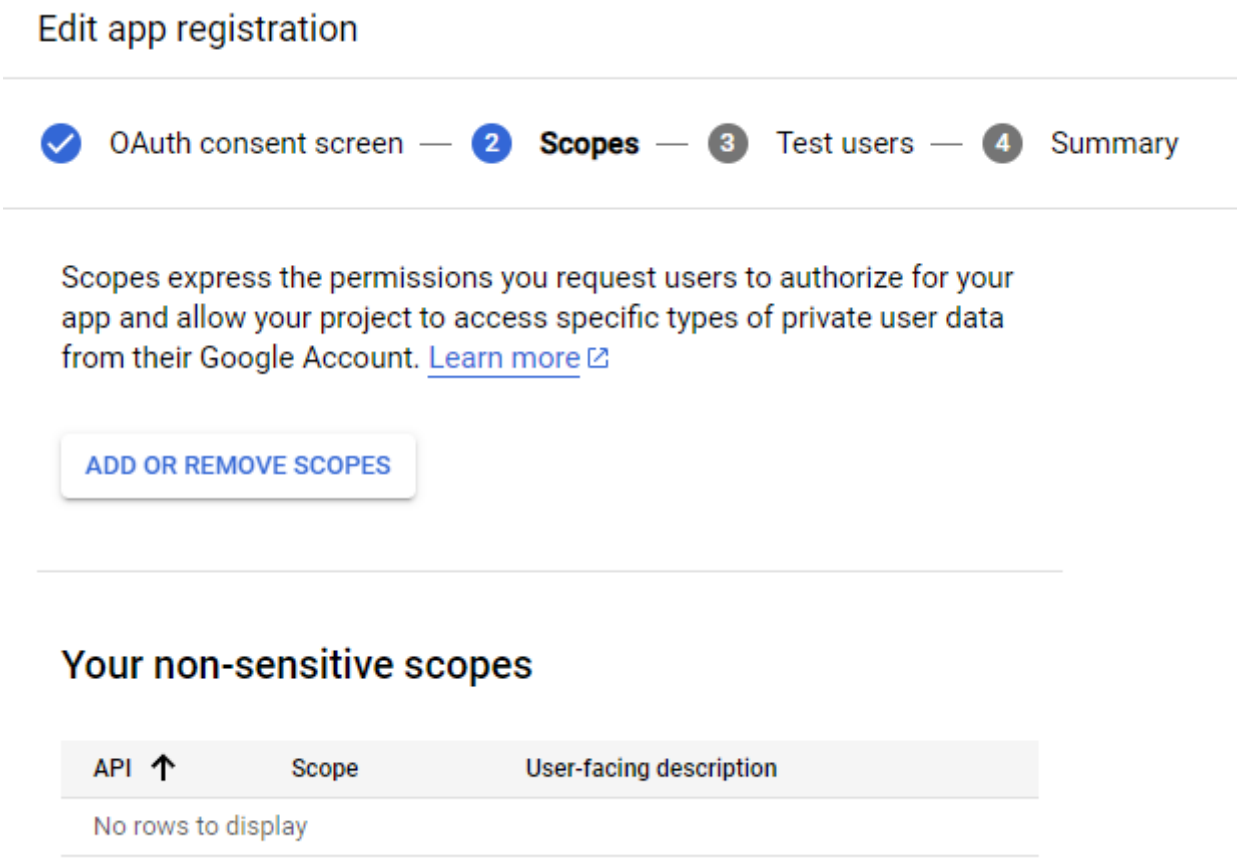


App name – это имя вашего приложения для авторизации. То есть когда пользователь будет перенаправляться в Google он будет видеть именно это название — оно означает какому сервису Goggle предоставляет данные
User support email – это адрес электронной почты Google аккаунта
Далее мы прокручиваем в самый низ и видим Developer contact information
Сюда надо ввести ваш email



Далее нажимаем кнопку SAVE AND CONTINUE

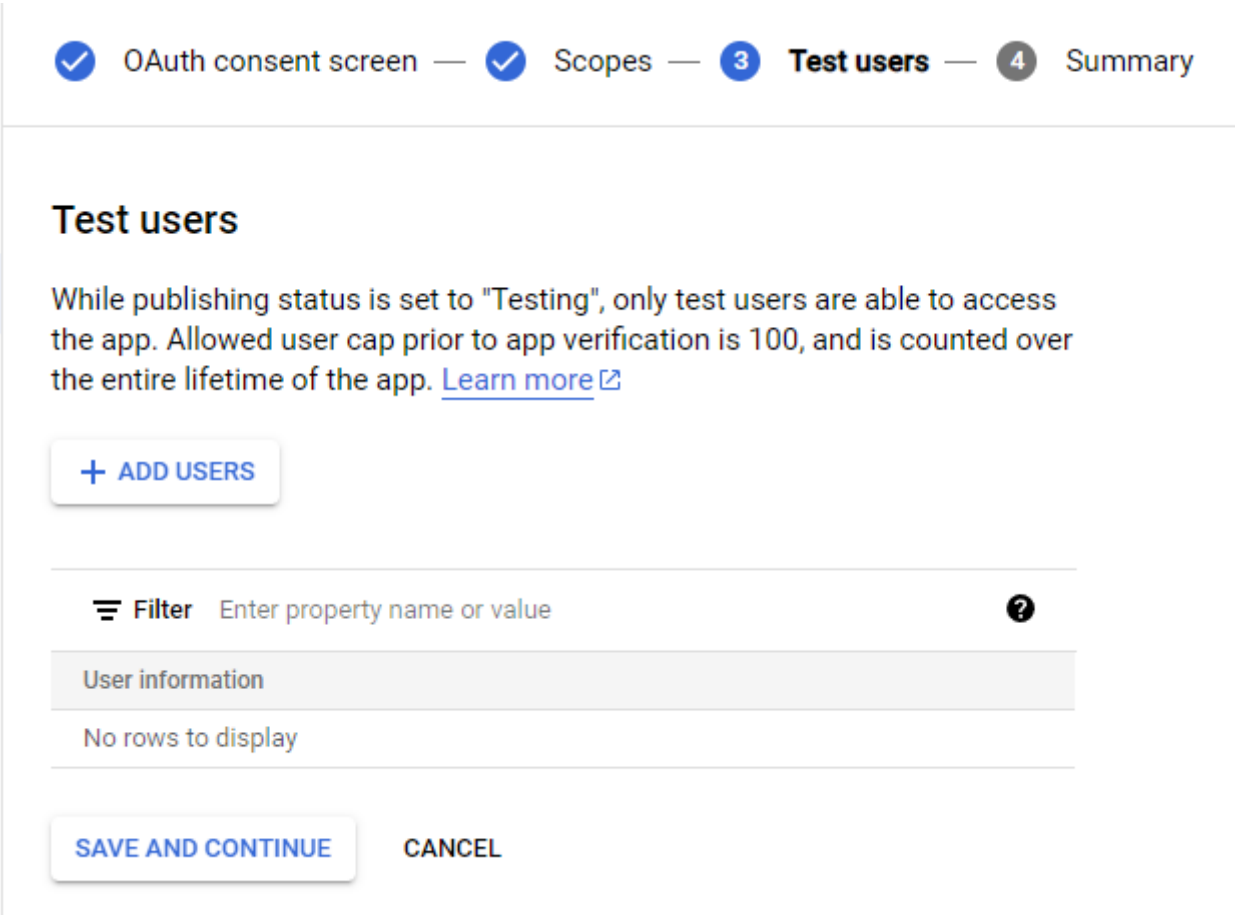
9. Мы попадаем в 2. SCOPES



Здесь настраиваются данные о пользователе, которые мы хотим получить. Если здесь ничего не менять, мы и так получим все необходимые данные. Но если вы хотите какие-то дополнительные данные, такие как «Местоположение», «Возраст», то можете здесь настроить.

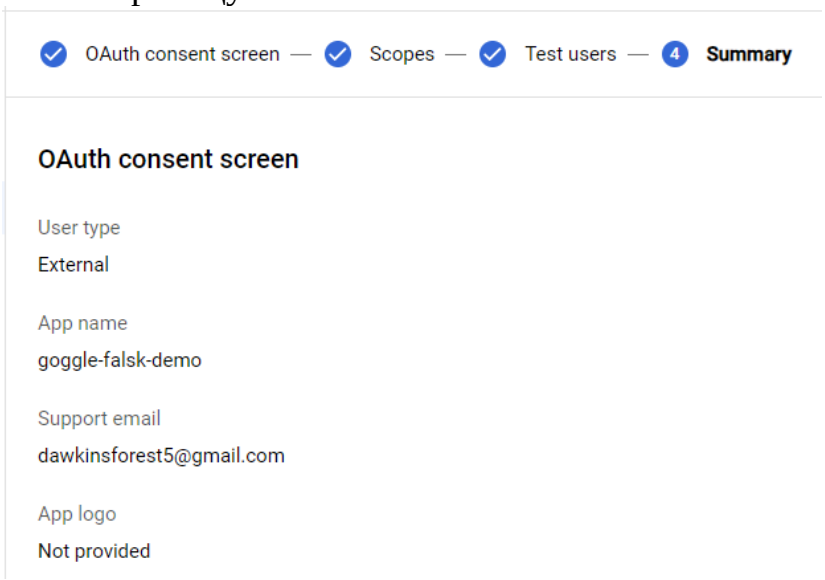
Но в нашем случае это не нужно, поэтому нажимаем кнопку SAVE AND CONTINUE внизу.

10. Мы попадаем в Test users

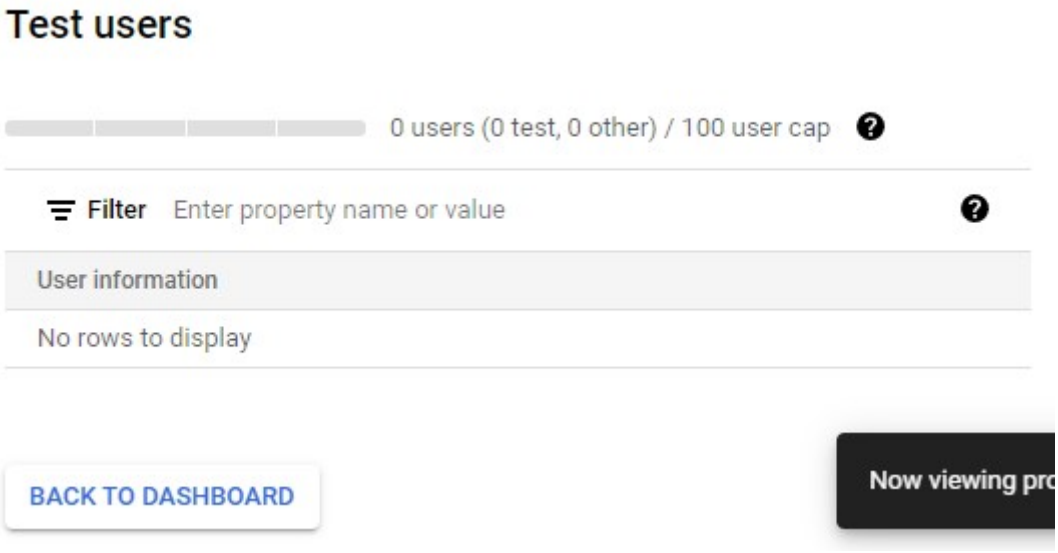


Это специфические параметры, которые нам также не нужны, поэтому нажимаем кнопку SAVE AND CONTINUE.

11. Мы попадаем на страницу SUMMARY

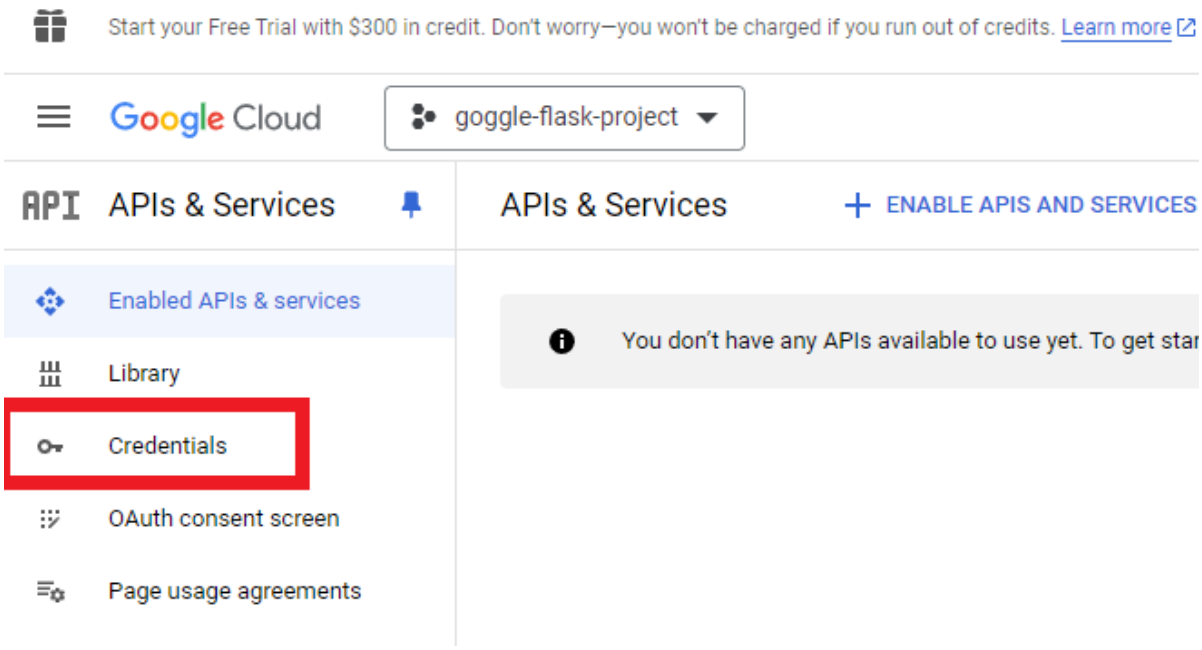


Проверяем правильность данных, если все верно перелистываем вниз

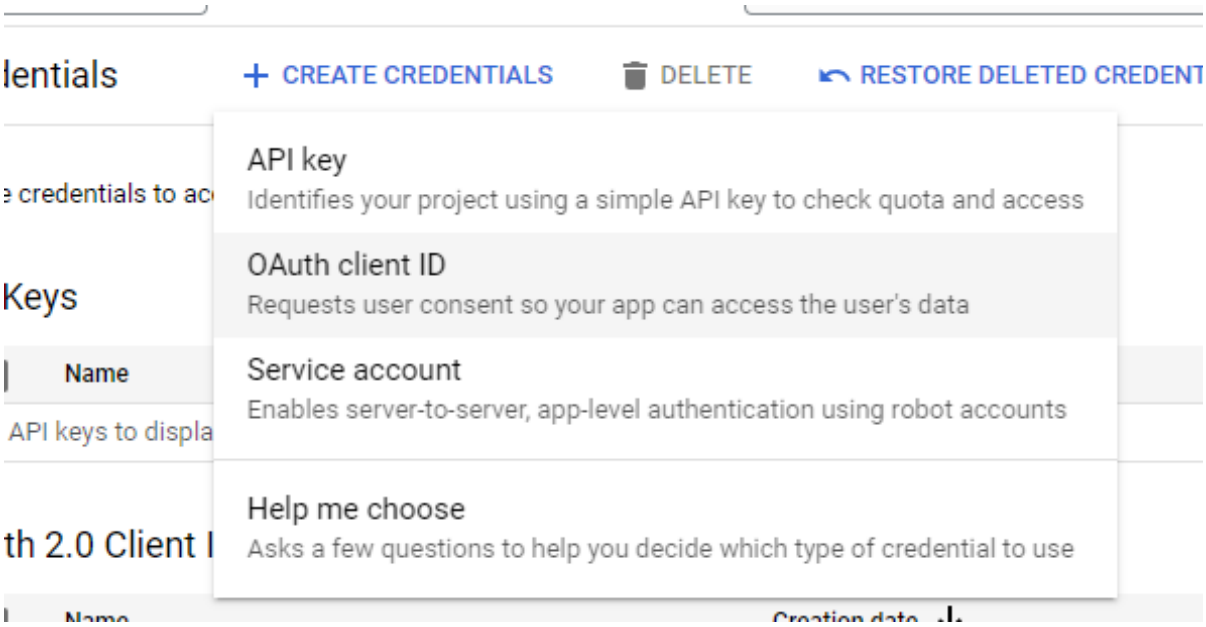


И нажимаем на кнопку BACK TO DASHBOARD

12. Далее переходим назад в Credentials

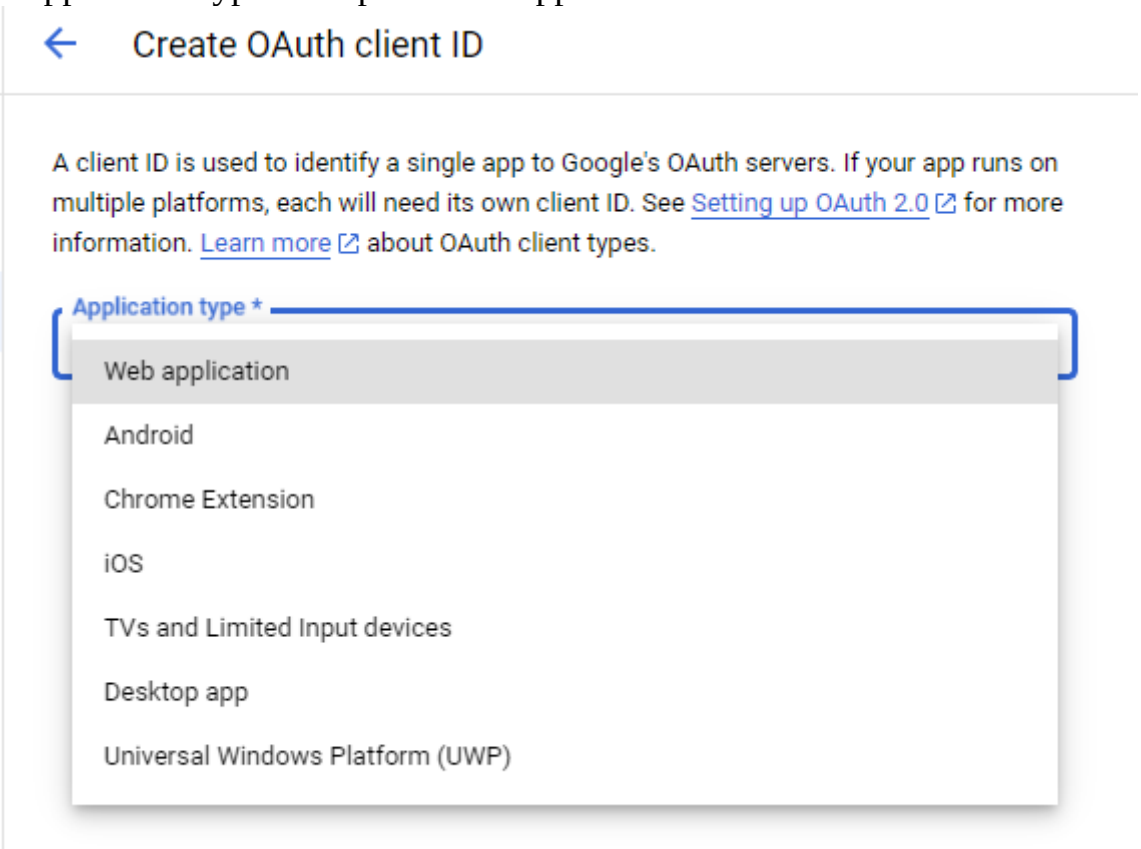


Вверху щелкаем CREATE CREDENTIAL

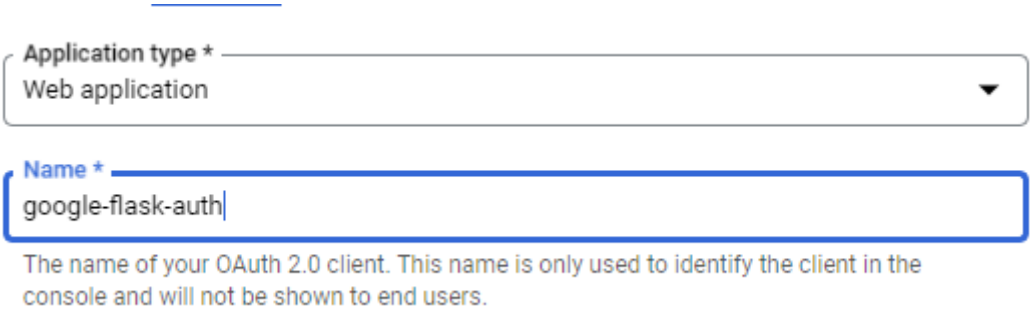


Выбираем OAuth client ID

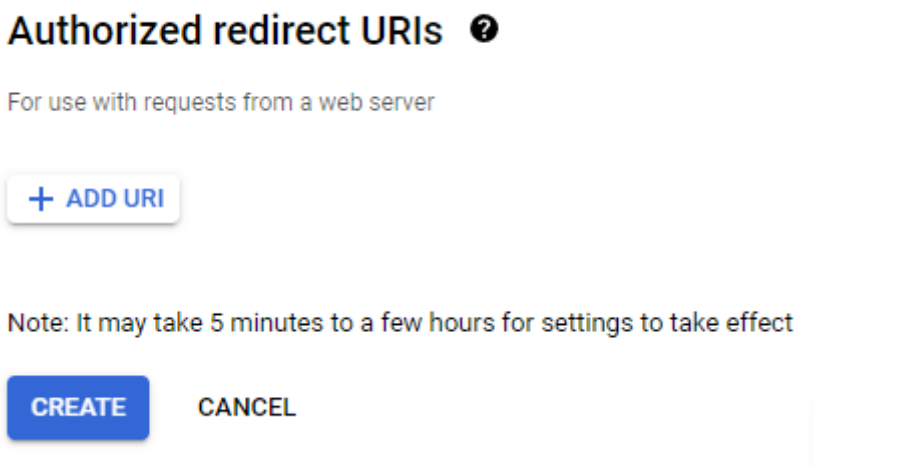
13. В Application type выбираем Web application



Далее вводим имя приложение (Оно не будет показано конечному пользователю, который будет авторизоваться через Goggle)

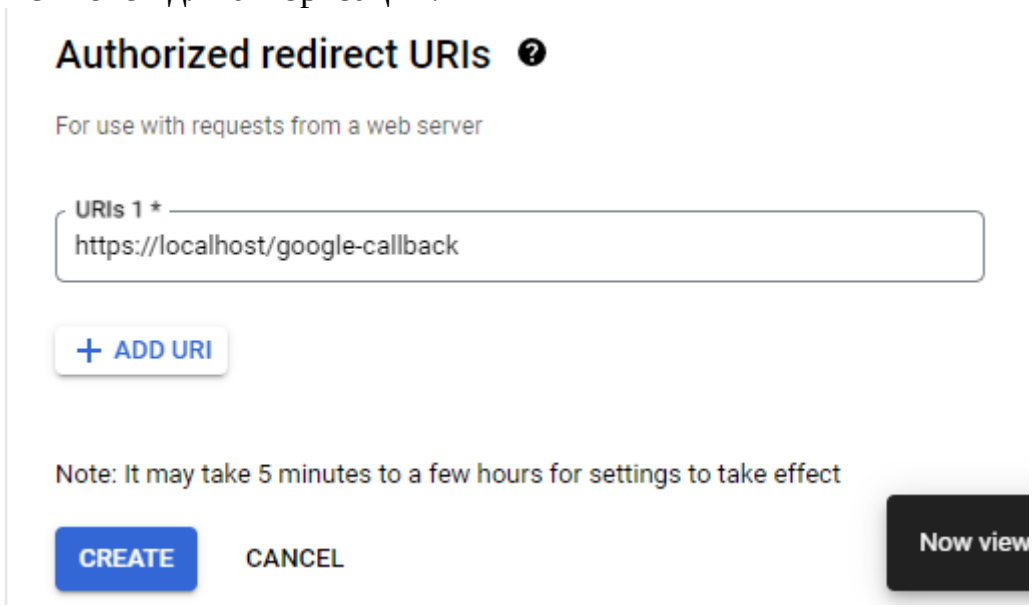


14. САМЫЙ ГЛАВНЫЙ МОМЕНТ



Мы должны нажать + ADD URI

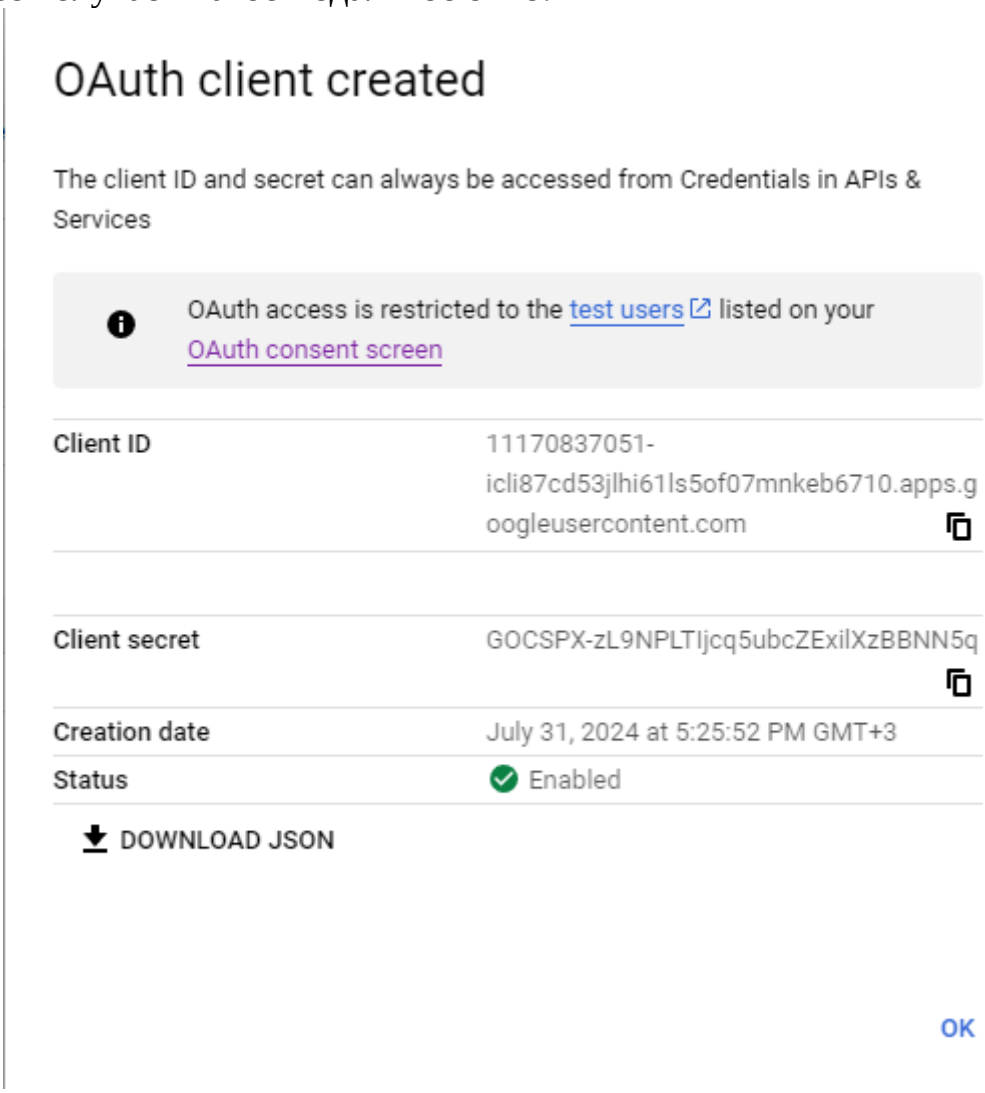
Сюда мы должны ввести имя эндроинта в нашем приложении, куда будет оправляться токен для авторизации:



У нас в нашем приложении должен быть такой эндпоин. В моем приложении это выглядит так:

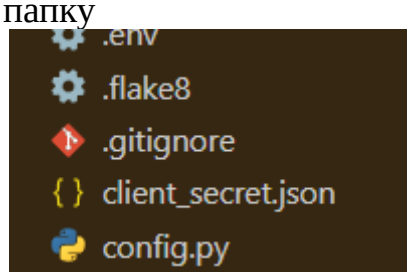
```python
@auth.route("/google-login")
def google_login():
  """Reginster or sign in by means of Google
Account."""
  authorization_url, state = flow.authorization_url()
  return redirect(authorization_url)
```

15. Далее получаем такое модальное окно:



Здесь нам обязательно надо нажать DOWNLOAD JSON.

16. Мы скачаем JSON файл со всеми секретными ключами, которые будут использованы в проекте. Теперь этот файлик добавляем в нащ проект, скопировав его в корневую папку

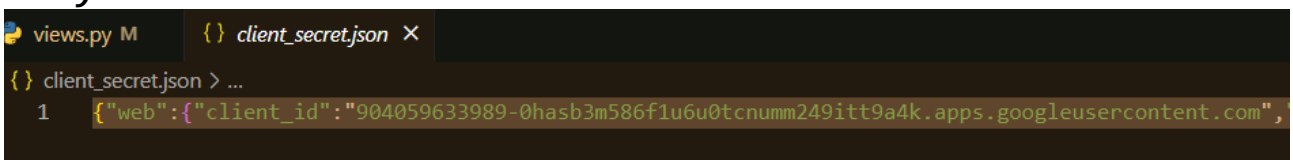


17. Теперь можно закрыть консоль Google и дальше продолжить работать в нашем проекте.

18. В переменные окружения — файлик .env необходимо записать следующую информацию

```
# Goggle OAuth
GOOGLE_CLIENT_ID=904059633989-
0hasb3m586f1u6u0tcnumm249itt9a4k.apps.googleuser
```

Это берется из того самого JSON-Файла, что мы скачали в пункте 16.



19. Далее необходимо скачать следующие библиотеки Python:

```
pip install google-auth
pip install google-auth-oauthlib
```

20. Переходим в файл с view-функциями (обычно views.py)
И делаем необходимые импорты

```
from google_auth_oauthlib.flow import Flow
import google.auth.transport.requests
from google.oauth2 import id_token
```

21. Там же в глобальной области прописываем следующий код

```
client_secrets_file = os.path.join(basedir,
'client_secret.json')

flow = Flow.from_client_secrets_file(
  client_secrets_file=client_secrets_file,
  scopes=[
    'https://www.googleapis.com/auth/userinfo.profile',
    'https://www.googleapis.com/auth/userinfo.email',
    'openid'],
```

```
    redirect_uri='https://localhost/google-callback'
)
```

Вначале мы указываем путь к нашему JSON-файлу из пункта 16. Далее настраиваем Flow как в примере. Эндпоинты Goggle взяты из официальной документации

22. Далее создаем view-функцию для перенаправления пользователя на авторизацию

```
@auth.route("/google-login")
def google_login():
    """Reginster or sign in by means of Google
Account."""
    authorization_url, state = flow.authorization_url()
    return redirect(authorization_url)
```

23. Теперь создаем view-функцию для callback

```
@auth.route("/google-callback")
def google_callback():
    """Fetch a response from Google."""
    flow.fetch_token(authorization_response=request.url)
    credentials = flow.credentials
    request_session = requests.session()
    cached_session =
cachecontrol.CacheControl(request_session)
    token_request =
google.auth.transport.requests.Request(
        session=cached_session)
    id_info = id_token.verify_oauth2_token(
        id_token=credentials._id_token,
        request=token_request,
        audience=os.environ.get('GOOGLE_CLIENT_ID'),
        clock_skew_in_seconds=10
    )
    social_uid = str(id_info.get('sub'))
```

```
username = id_info.get('name')
email = id_info.get('email')
```

Мы получили уникальный id клиента Goggle (“sub”), его имя «name” и «email”. Теперь эти данные можно использовать в для реализации логики регистрации через социальные сети в нашем приложении.

6. Регистрация через VK

Регистрируем VK ID - https://id.vk.com/
Переходим в консоль разработчика - https://id.vk.com/about/business/go/accounts/90508/apps

1. Далее необходимо нажать на кнопку «Добавить приложение»

2. Вводим название приложения (оно будет отображаться у пользователя вашего сайта) и выбрать Web

Шаг 1 из 2

Регистрация приложения

Введите название приложения 6 / 48

my_app

Выберите нужные платформы

Web

Android

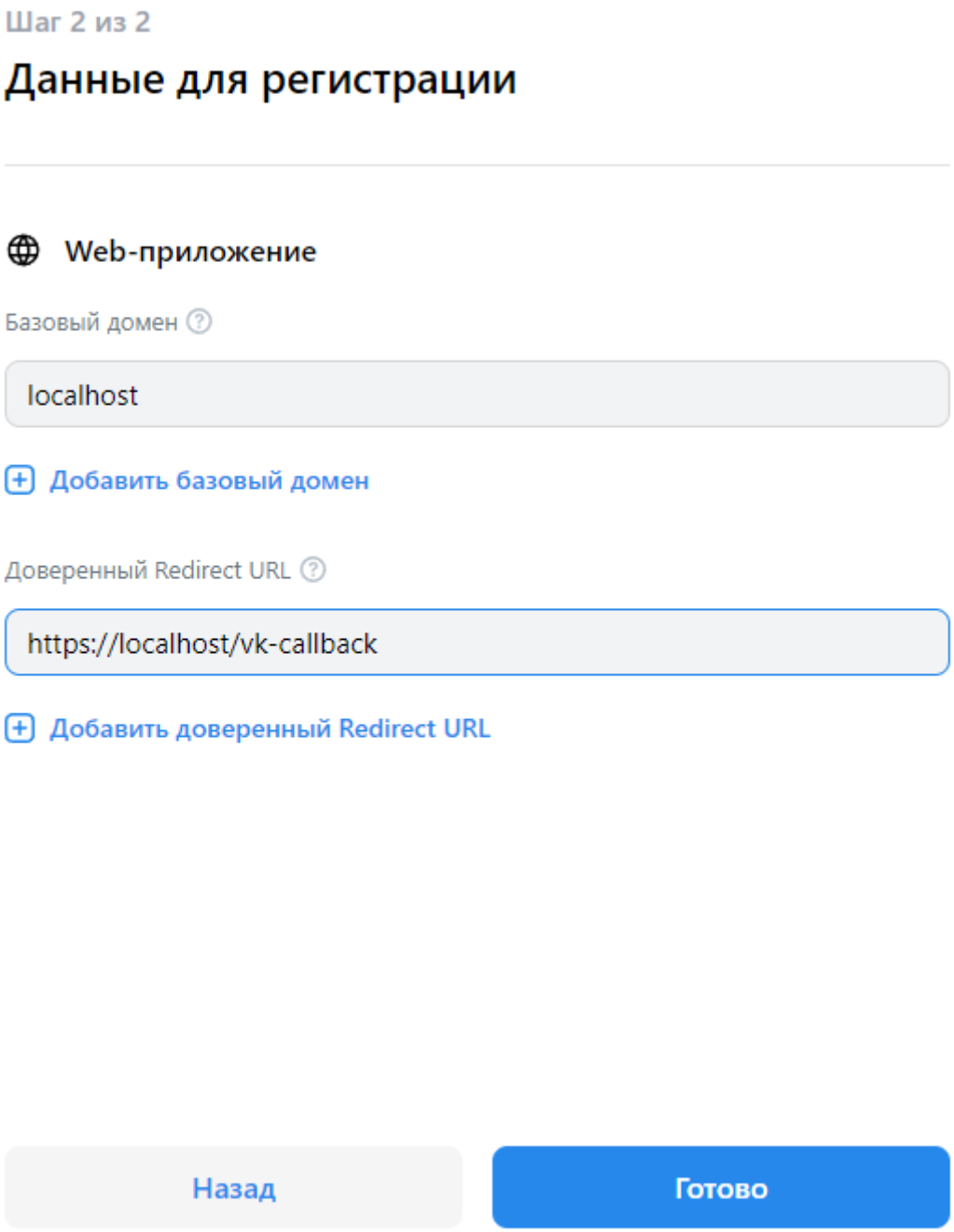
iOS

Выберите изображение

Для окон авторизации и иконки приложения
в личном кабинете пользователя VK

Отменить Далее

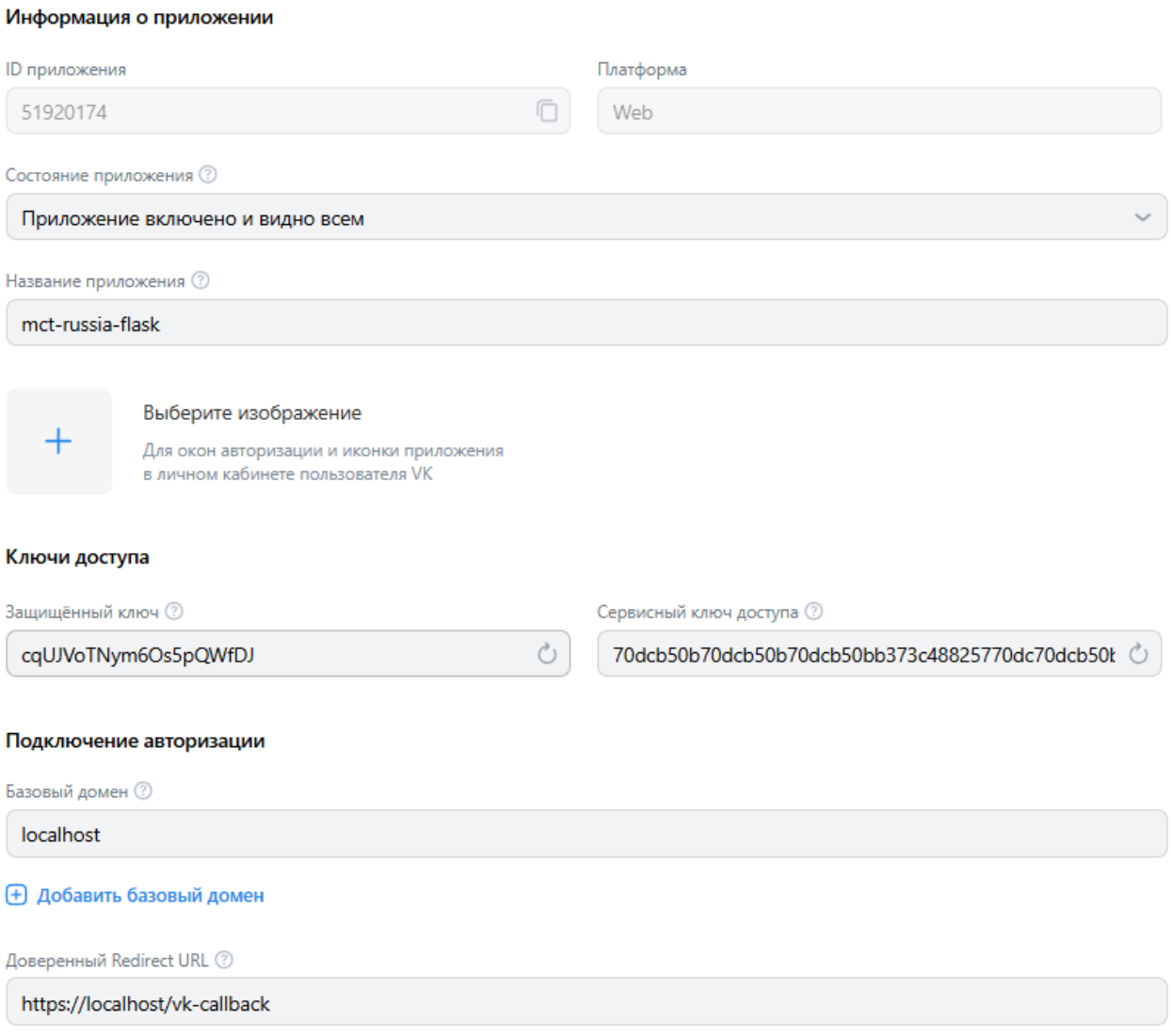
Нажать кнопку «Далее»
3. Переходим на страницу «Данные для регистрации». Здесь вводим данные как на картинке ниже:



Нажимаем кнопку «Готово»
4. Теперь нам доступны секреты приложения:
Их надо будет записать **в .env**

Вначале сформируем строку

```
VK_API_REQUEST=https://oauth.vk.com/authorize?
client_id=51920174&scope=messages;email&redirect_uri=h
ttps://localhost/vk-callback&response_type=code
```



`client_id – это ID нашего VK приложения`

Далее прописываем в .env еще дополнительную информацию, которая берется из информации о приложении

```
SOCIAL_AUTH_VK_OAUTH2_KEY=51920174
SOCIAL_AUTH_VK_OAUTH2_SECRET=cqUJVoTNym6Os5pQWfDJ
SOCIAL_AUTH_VK_REDIRECT=https://localhost/vk-callback
```

5. Теперь переходим во views нашего приложения.

6. Вначале установим библиотеку

```
pip install requests
```

7. Теперь реализуем логику приложения:

```
@auth.route('/vk-login')
def vk_login():
  """Reginster or sign in by means of VK Account."""
  redirect_url = os.environ.get('VK_API_REQUEST')
  return redirect(redirect_url)
```

```
@auth.route('/vk-callback')
def vk_callback():
  """Fetch a response from VK."""
  code = request.args.get('code')
  response =
requests.post('https://oauth.vk.com/access_token',
params={
    'client_id':
os.environ.get('SOCIAL_AUTH_VK_OAUTH2_KEY'),
    'client_secret':
     os.environ.get('SOCIAL_AUTH_VK_OAUTH2_SECRET'),
    'redirect_uri':
os.environ.get('SOCIAL_AUTH_VK_REDIRECT'),
    'code': code,
  })
  if response.status_code == HTTPStatus.OK:
    data = response.json()
    access_token = data['access_token']
    user_id = data['user_id']
    user_info_response = requests.post(
      'https://api.vk.com/method/users.get',
```

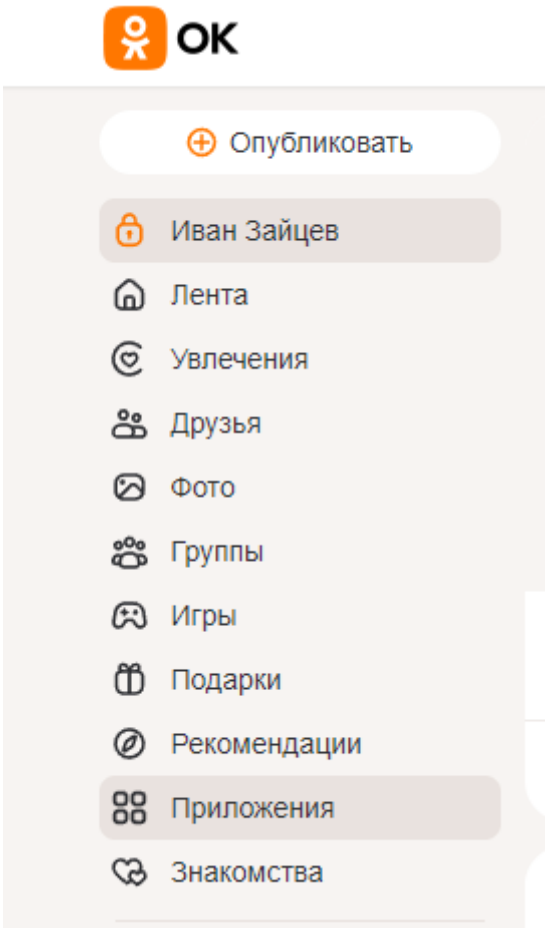
```
    params={
        'access_token': access_token,
        'user_ids': user_id,
        'v': '5.199'})
    if user_info_response.status_code == HTTPStatus.OK:
        user_info = user_info_response.json()['response']
[0]
        user_data = {
            'id': user_info.get('id'),
            'first_name': user_info.get('first_name'),
            'last_name': user_info.get('last_name'),
        }
        username = f'{user_data['first_name']}
           {user_data['last_name']}'

        email = get_random_email()
        social_uid = str(user_data['id'])
```

Теперь мы получили информацию о пользователи VK и можем дальше реализовывать логику его регистрации и авторизации.

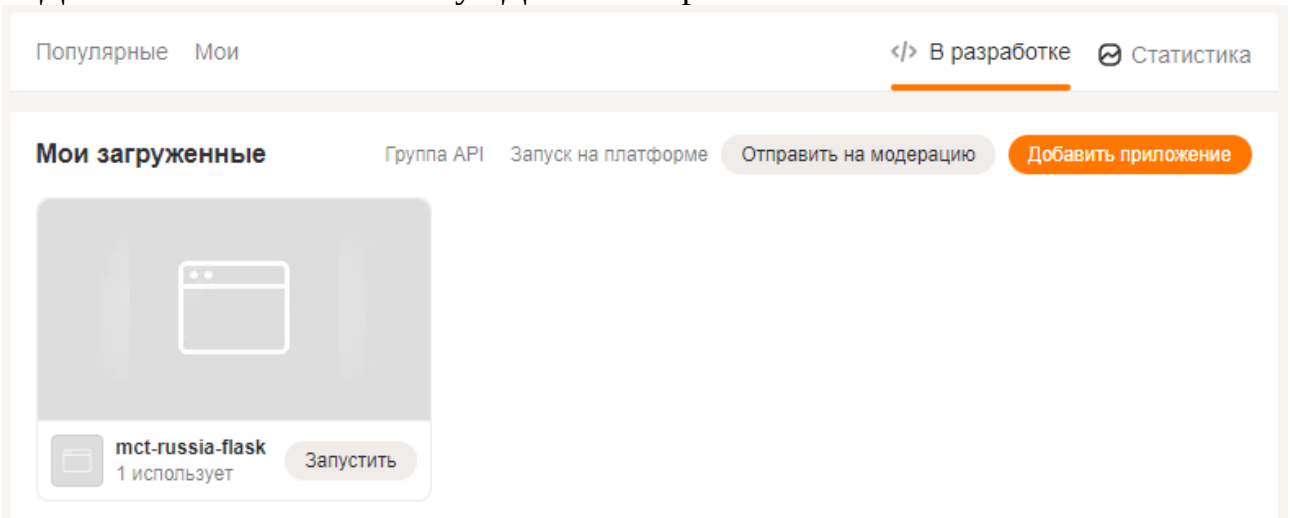
7. Регистрация через Одноклассники

Если у вас есть VK ID, то вы сможете спокойно получить доступ к сайту «Одноклассники», использую этот ID.

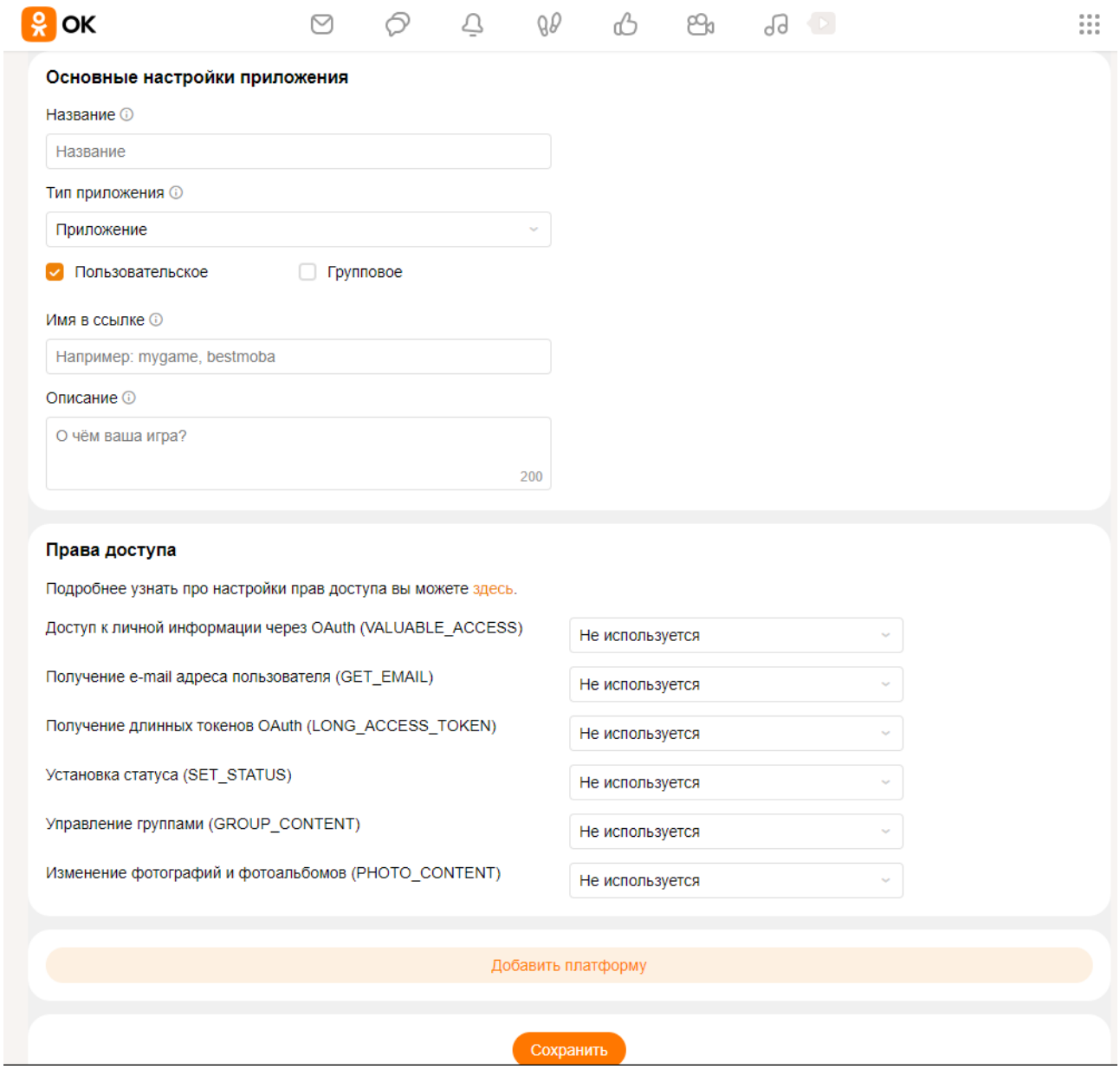
1. Зайдите в ваш профайл https://ok.ru/profile/
2. Обязательно зарегистрируйте свою электронную почту и подтвердите её
3. В меню слева есть много различных вкладок, нам нужно приложение



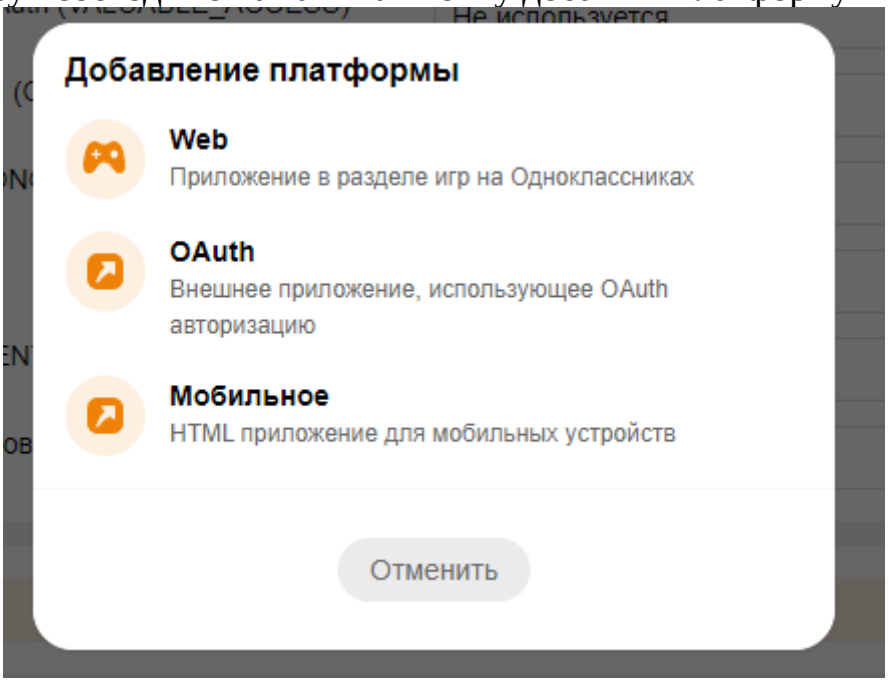
4. Далее нажимаем на кнопку «Добавить приложение»



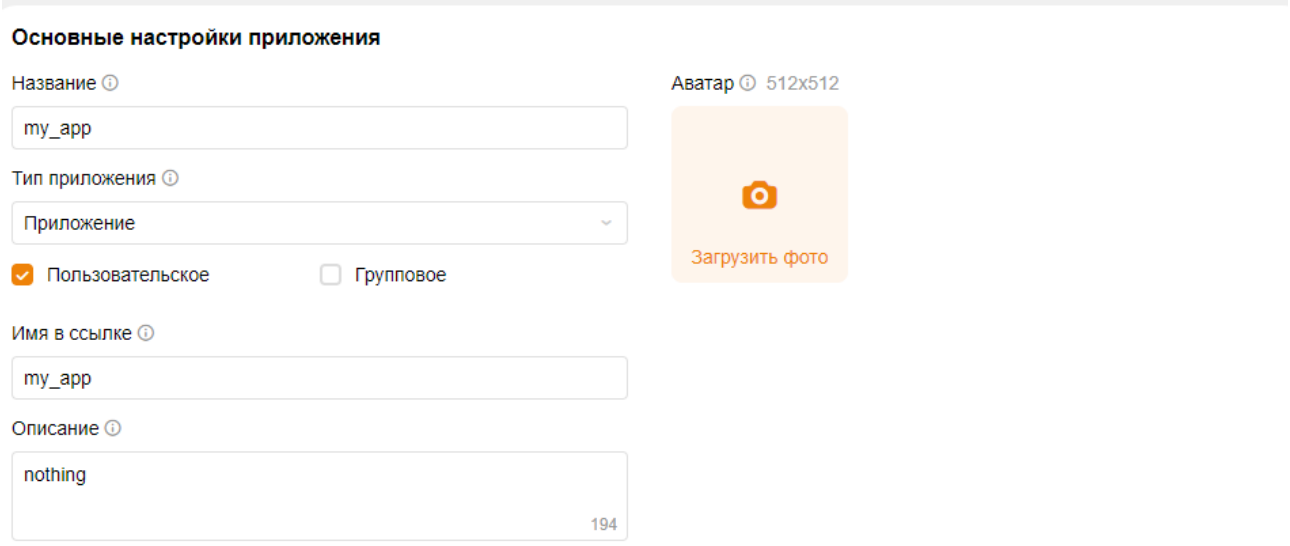
У нас появится такое окно



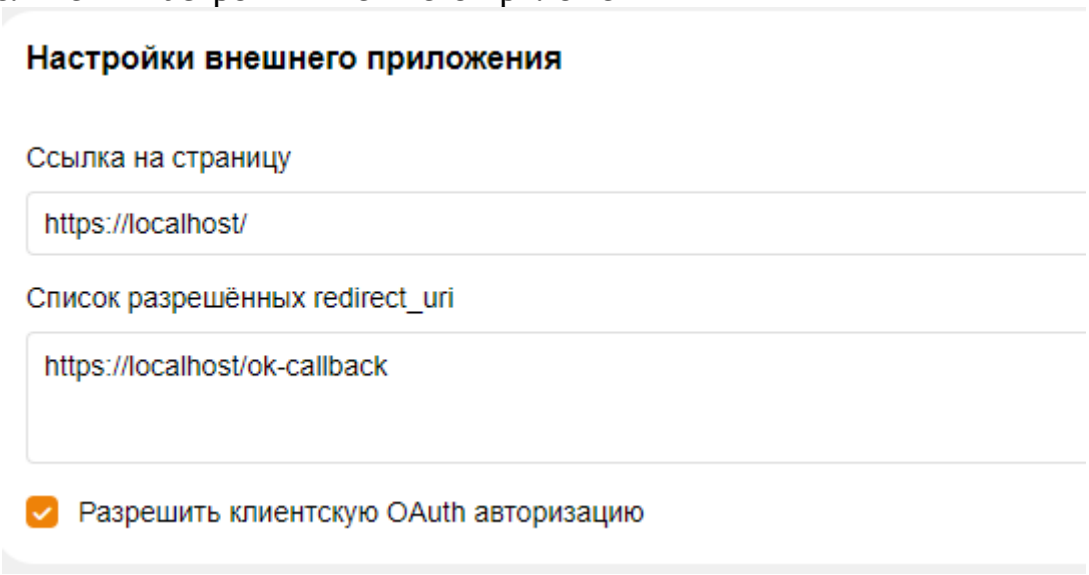
5. Здесь внизу необходимо нажать на кнопку добавить платформу



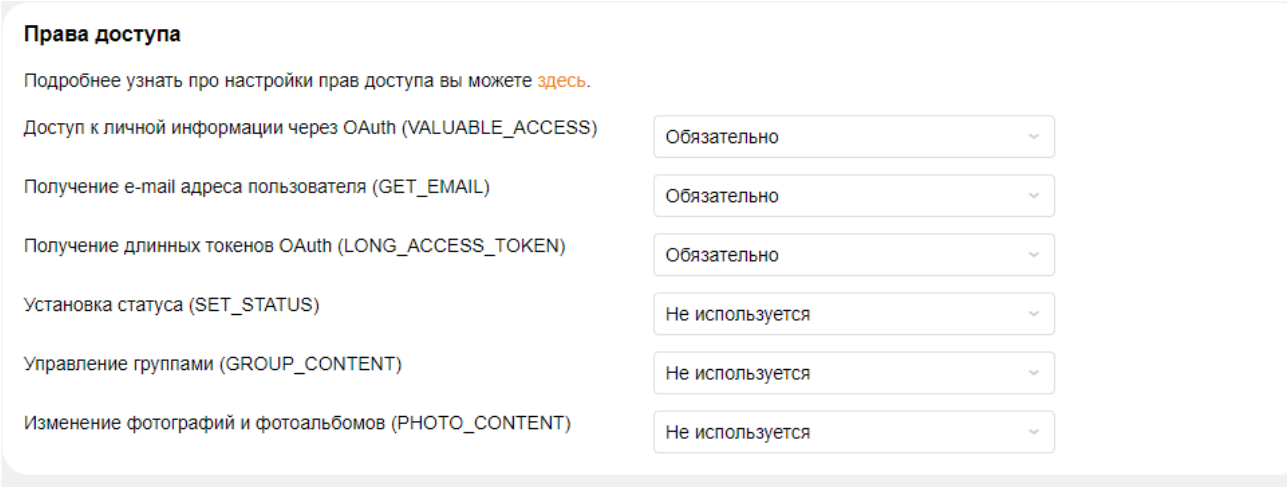
Нажимаем на OAuth.
6. Заполняем «Основные настройки приложения»



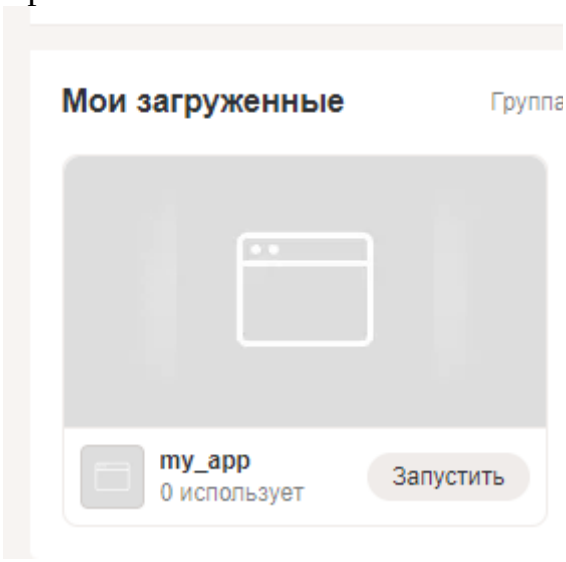
7. Заполняем «Настройки внешнего приложения»



8. Выбираем права доступа — это та информация о пользователя, которая придем с «Одноклассников»



Нажимаем кнопку «Сохранить»
9. Приложение готово к работе



10. Теперь нам надо узнать некоторые данные этого приложения и записать их в .env файл:

К нам на электронную почту к которой привязан аккаунт Одноклассники должно прийти письмо со всеми необходимыми данными

Здравствуйте, Иван Зайцев!

Ваше приложение mct-russia-flask успешно зарегистрировано на Одноклассниках.

Application ID: 512002593594.
Публичный ключ приложения: CLDEJMLGDIHBABABA.
Секретный ключ приложения: 0AF67F589594B24F2F41BDE6.
Ссылка на приложение: https://ok.ru/game/512002593594

--
С уважением,
Служба поддержки OK.ru

Все эти данные записываем в наш файл .env

```
# OK OAuth
OK_API_REQUEST=https://connect.ok.ru/oauth/authorize?
client_id=512002593594&scope=VALUABLE_ACCESS;GET_EMAIL
&response_type=token&redirect_uri=https://localhost/
ok-callback
OK_CLIENT_ID=512002593594
OK_CLIENT_SECRET=0AF67F589594B24F2F41BDE6
OK_PUBLIC_KEY=CLDEJMLGDIHBABABA
OK_REDIRECT_URI=https://localhost/ok-callback
```

11. Мы получили все необходимые данные. Проблема в том, что «Одноклассники» используют устаревшую модель OAuth 1.0 или неявную авторизацию. Токен доступа и секретный ключ вам будет выслан не на сервер, а в адресную строку. Поэтому мы должны создать промежуточный шаблон, откуда мы будем извлекать этот токен с помощью JavaScript и отправлять его на сервер Flask

Скрипт выглядит так odnoklassniki.js:

```
var fragment = window.location.hash.substr(1);
var params = new URLSearchParams(fragment);
var token = params.get('access_token');
var session = params.get('session_secret_key');
var xhr = new XMLHttpRequest();
```

```
xhr.open('GET', '/ok-callback?access_token=' + token +
'&session_secret_key=' + session, true);
xhr.onreadystatechange = function() {
 if (xhr.readyState === 4 && xhr.status === 200) {
  console.log(xhr.responseText);
 }
};
xhr.send();
```

12. Теперь мы должны подключить этот скрипт в промежуточном шаблоне

```
{% block scripts %}
  {{ super() }}
  <script src="{{ url_for('static',
filename='js/odnoklassniki.js') }}"></script>
{% endblock %}
```

13. Теперь реализуем 3 view-функции — для перенаправления на сервер Одноклассников, для промежуточного шаблона с кнопкой регистрации через Одноклассники (который извлекает токен из адресной строки), и callback, где мы получаем все необходимые данные

```
@auth.route('/ok-login')
def ok_login():
  """Reginster or sign in by means of OK Account."""
  redicrect_url = os.environ.get('OK_API_REQUEST')
  return redirect(redicrect_url)
```

```
@auth.route('/ok-callback')
def ok_callback():
  """Fetch a response from Odnoklassniki."""
  access_token = request.args.get('access_token')
  session_secret_key = request.args.get('session_secret_key')
  if access_token and session_secret_key \
    and 'null' not in access_token \
      and 'null' not in session_secret_key:
    url = 'https://api.ok.ru/fb.do'
    params = {
      'application_key': os.environ.get('OK_PUBLIC_KEY'),
      'method': 'users.getCurrentUser',
      'access_token': access_token,
      'sig': session_secret_key,
      'format': 'json'
```

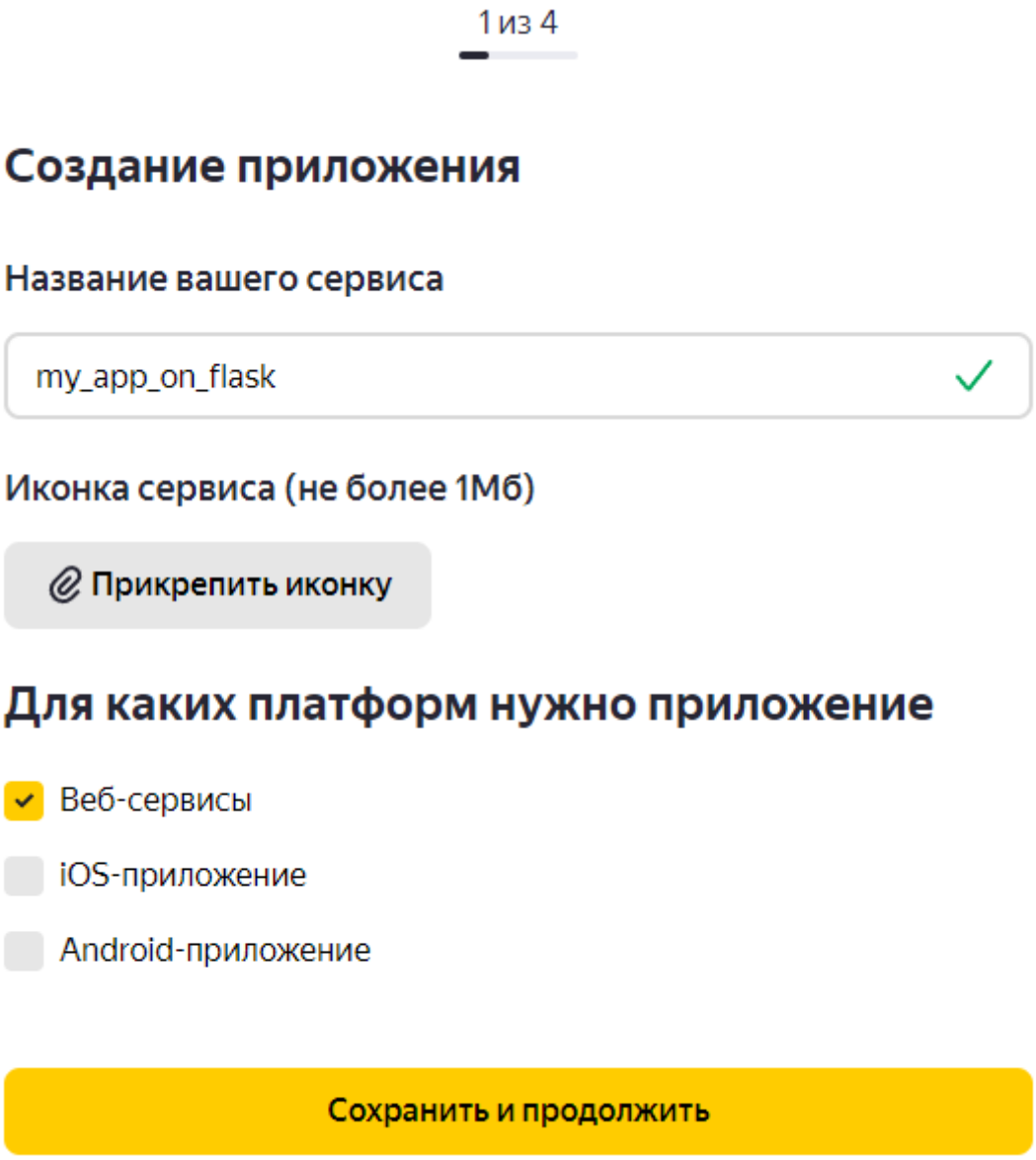
```
    }
    ok_response = requests.post(url, params=params)
    if ok_response.status_code == HTTPStatus.OK:
        ok_user_data = ok_response.json()
        username = ok_user_data.get('name')
        email = ok_user_data.get('email')
        social_uid = str(ok_user_data.get('uid'))
```

Мы получили все данные о пользователе из Одноклассников. Теперь их можно использовать для реализации логики регистрации в нашем приложении.

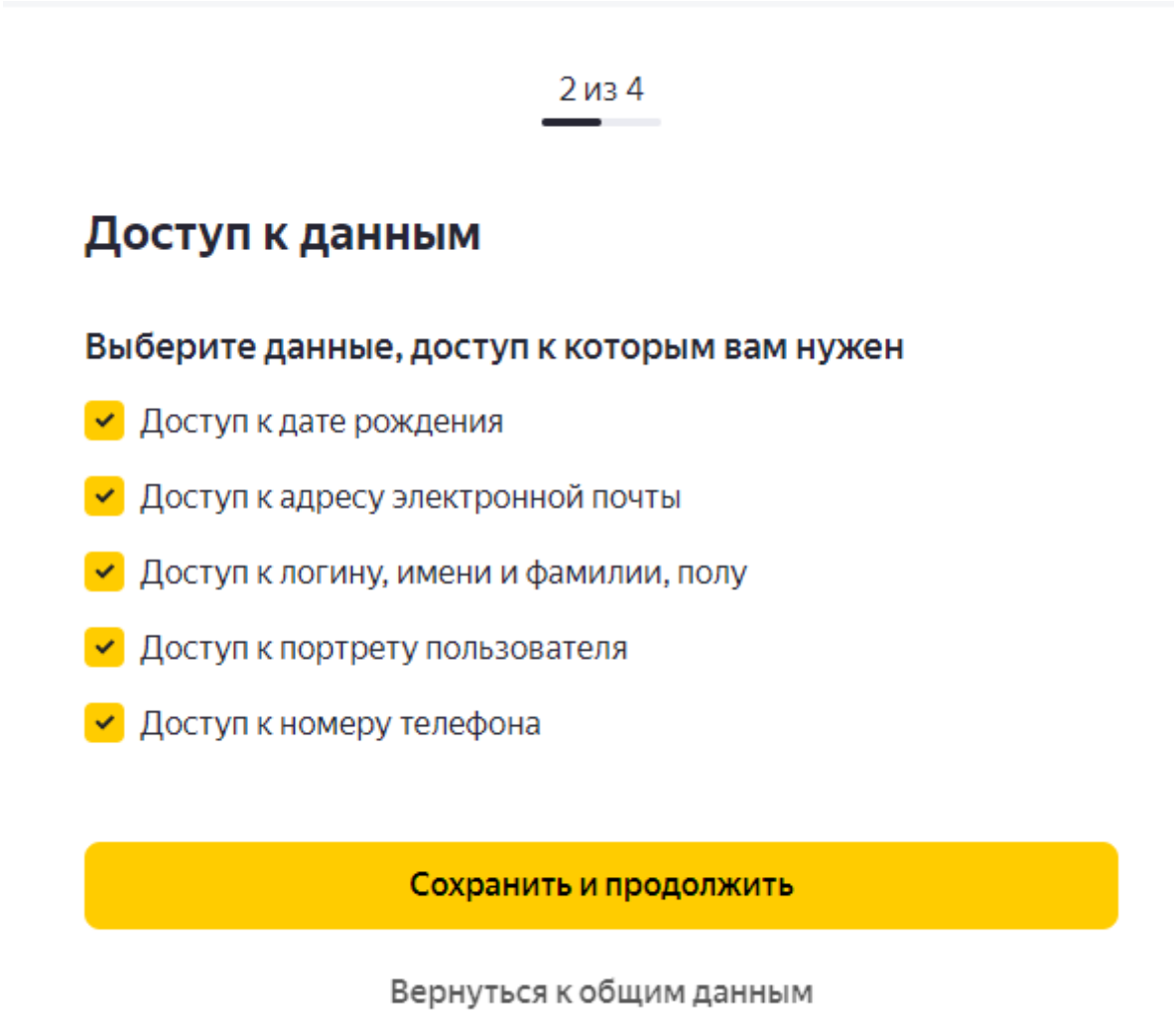
8. Регистрация через Яндекс

1. Регистрируем Яндекс ID - https://id.yandex.ru/
2. Переходим по ссылке - https://oauth.yandex.ru/
3. В нижней части страницы есть кнопка «Создать приложение»

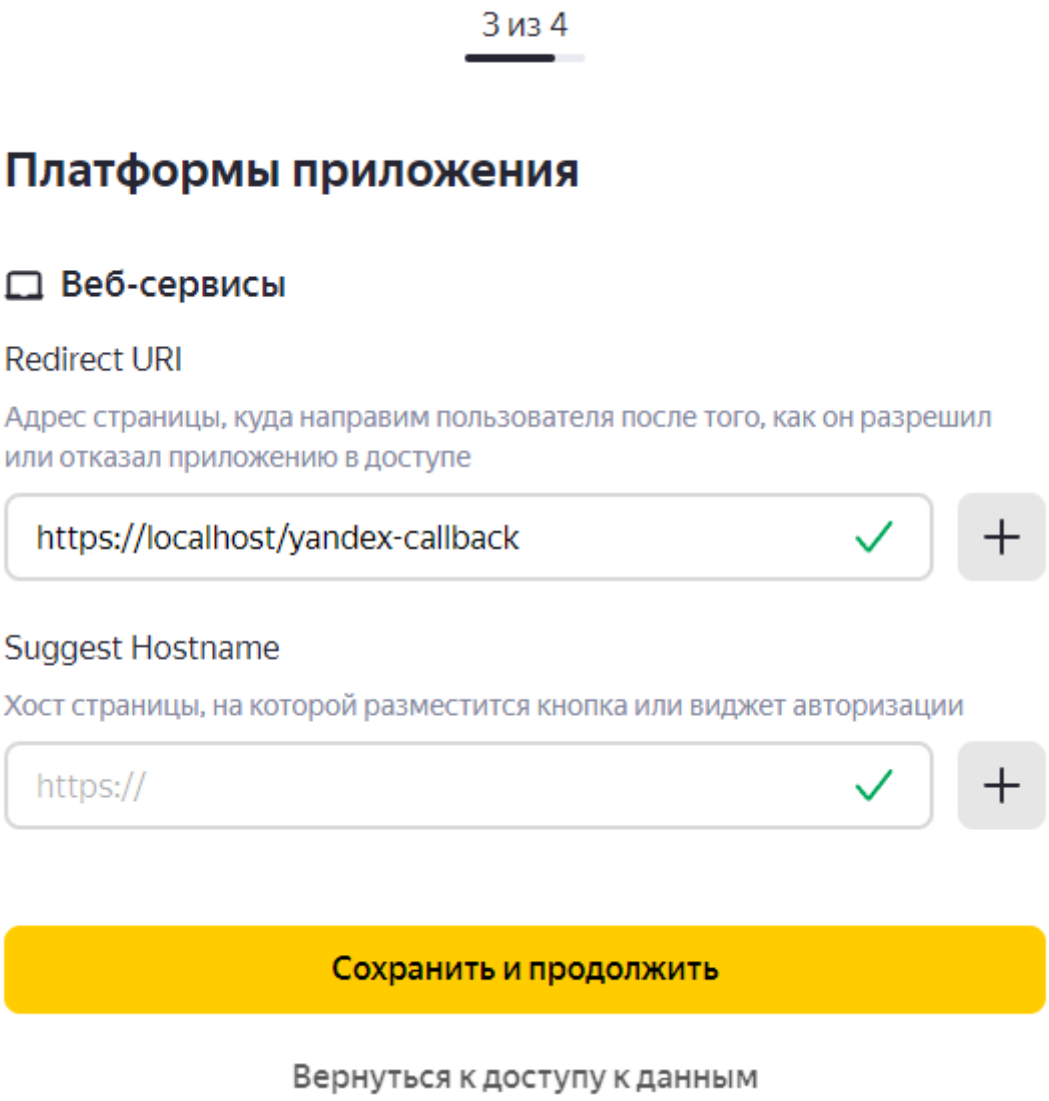
4. Придумываем название приложению и выбираем для какой платформы оно предназначено



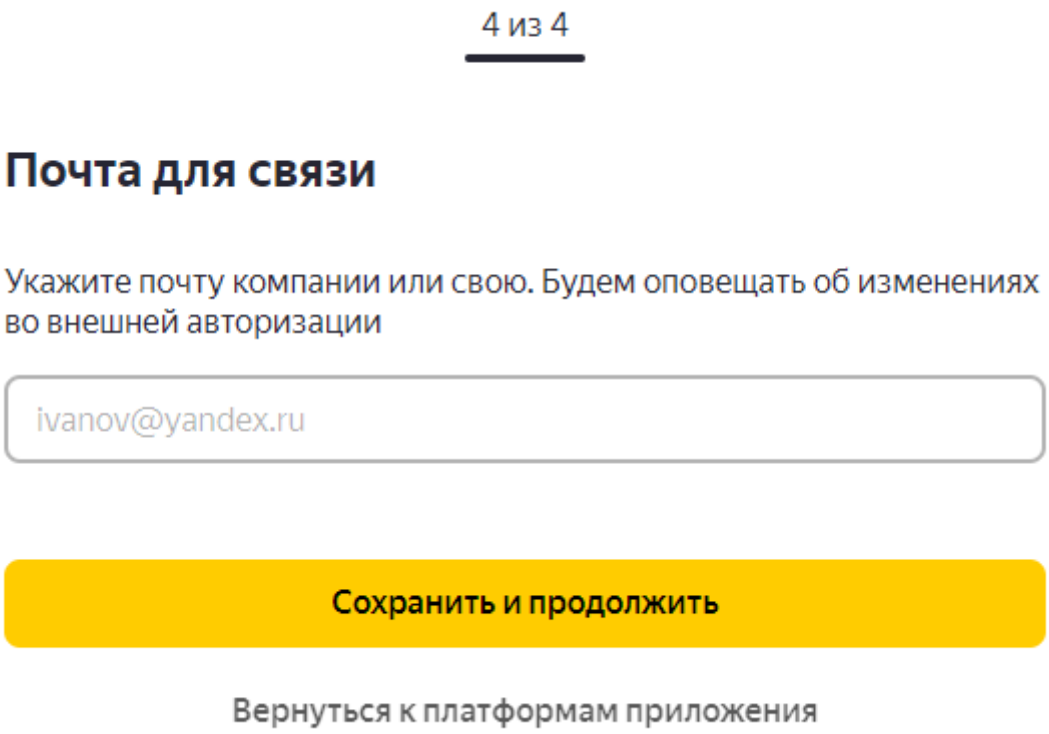
5. Выбираем ту информацию о пользователе, которую хотим получить



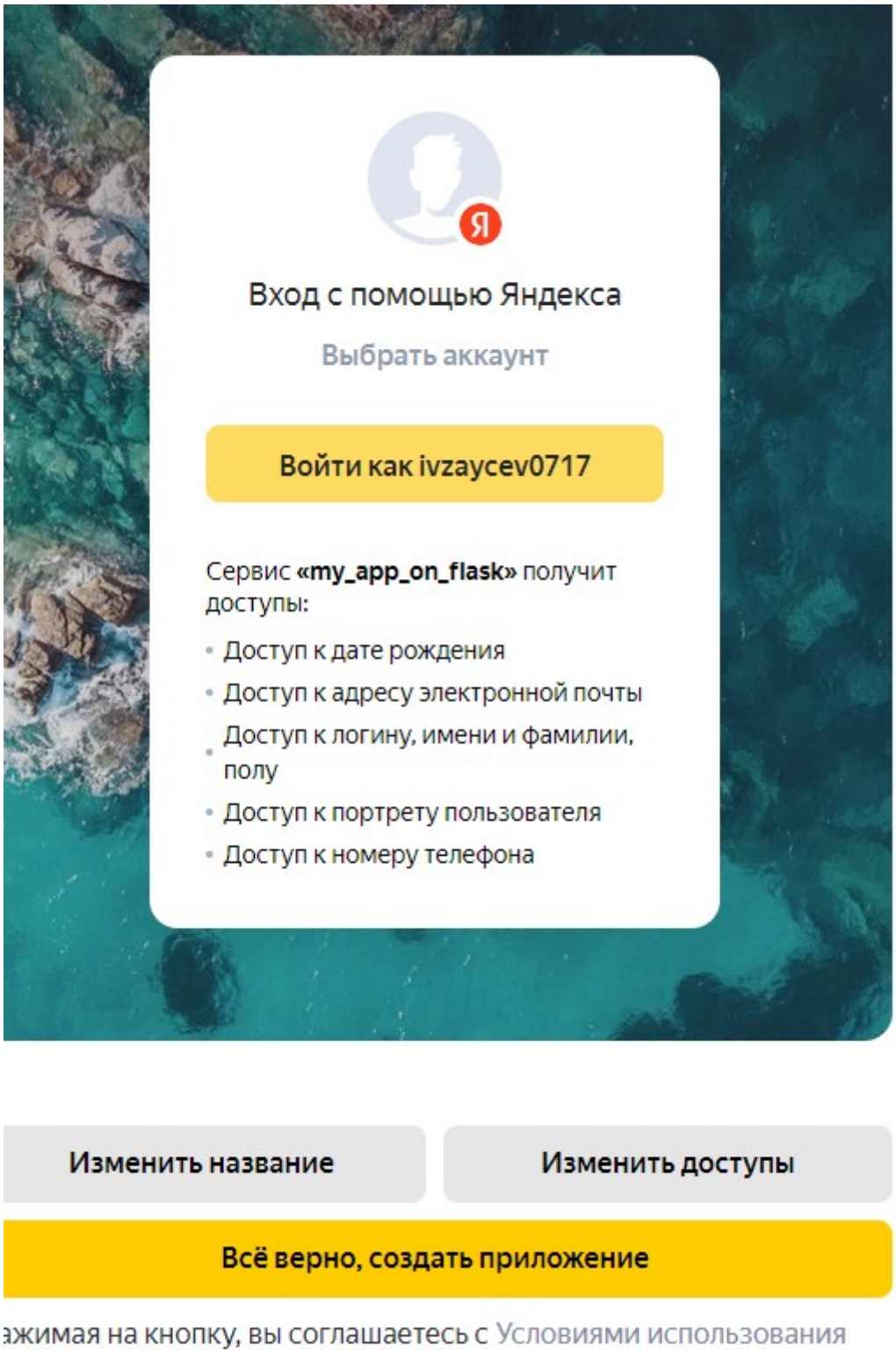
6. Вводим редирект



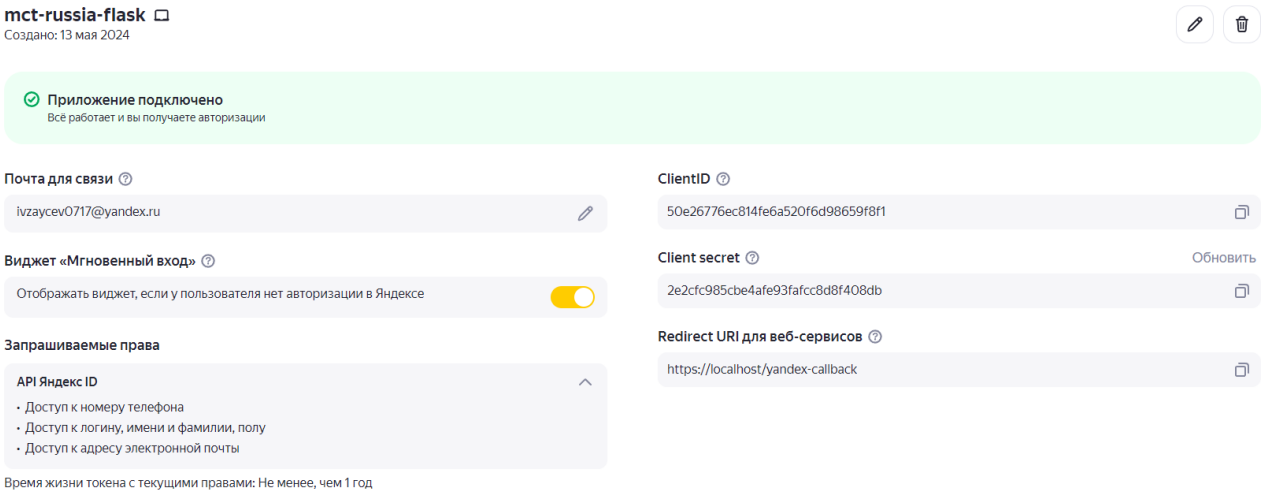
7. Вводим свой email



8. Проверяем, что все правильно



9. Теперь получаем доступ ко всем ключам



10. Копируем эти ключи в файл .env

```
# Yandex OAuth
YANDEX_API_REQUEST=https://oauth.yandex.ru/authorize?response_type=code&client_id=50e26776ec814fe6a520f6d98659f8f1&redirect_uri=https://localhost/yandex-callback
YA_CLIENT_ID=50e26776ec814fe6a520f6d98659f8f1
YA_CLIENT_SECRET=2e2cfc985cbe4afe93fafcc8d8f408db
YA_REDIRECT_URI=https://localhost/yandex-callback
```

11. Реализуем соответствующие view-функции

```
@auth.route('/yandex-login')
def yandex_login():
  """Reginster or sign in by means of Yandex Account."""
  redirect_url = os.environ.get('YANDEX_API_REQUEST')
  return redirect(redirect_url)
```

```
@auth.route('/yandex-callback')
def yandex_callback():
  """Fetch a response from Yandex."""
  code = request.args.get('code')
  if code:
    token_url = 'https://oauth.yandex.ru/token'
    token_params = {
      'grant_type': 'authorization_code',
      'code': code,
      'client_id': os.environ.get('YA_CLIENT_ID'),
      'client_secret': os.environ.get('YA_CLIENT_SECRET')
```

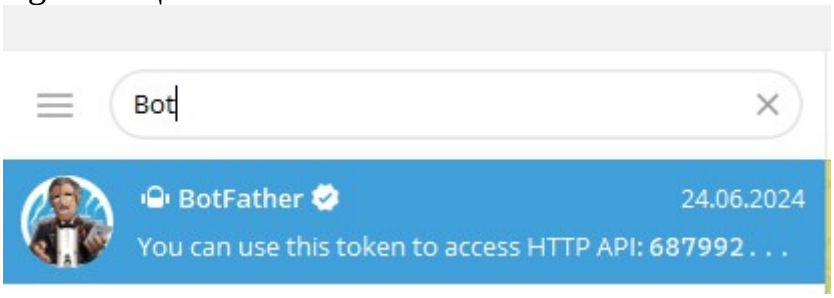
```
    }
    response = requests.post(token_url, data=token_params)
    data = response.json()
    if 'access_token' in data:
        user_info_url = 'https://login.yandex.ru/info'
        headers = {'Authorization': f'OAuth
     {data['access_token']}'}
        user_info_response = requests.post(user_info_url,
headers=headers)
        user_info = user_info_response.json()
        username = user_info.get('login')
        email = user_info.get('default_email')
        social_uid = str(user_info.get('id'))
        phone = user_info.get('default_phone').get('number')
```

Мы получили от Яндексу всю информацию о пользователе и можем приступать к реализации алгоритма регистрации.

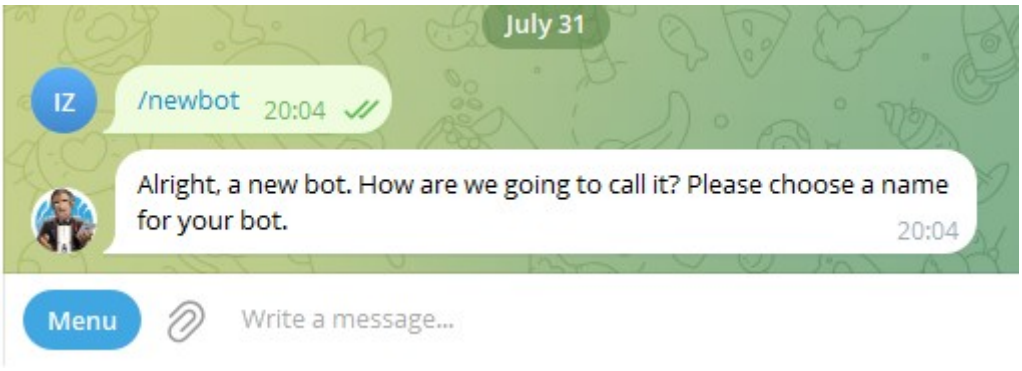
9. Регистрация через Telegram

Регистрация через Telegram очень сильно отличается от других сервисов.

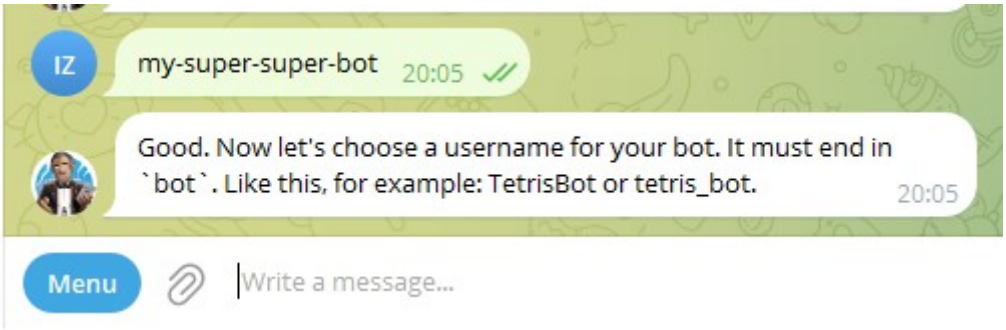
1. Надо зарегистрироваться в мессенджере Telegram
2. В поиске Telegram ищем BotFather



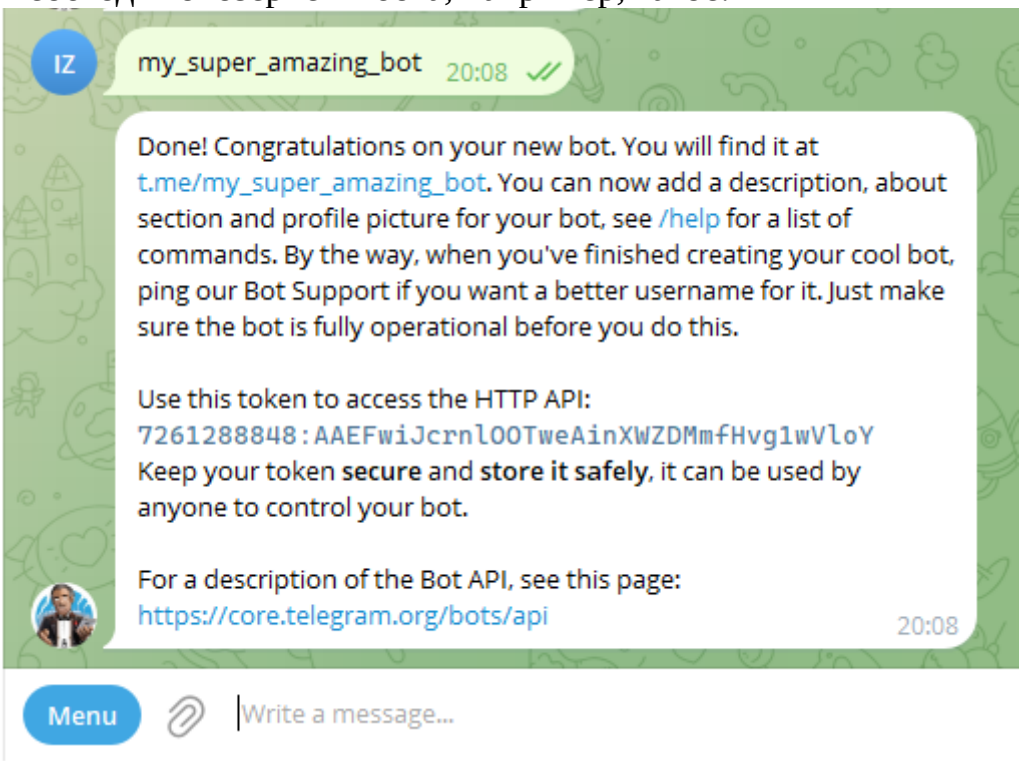
3. Заходим на этот бот и вводим команду /newbot



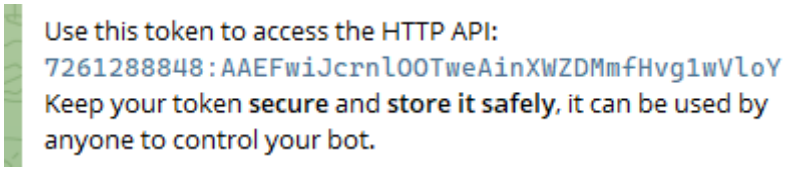
4. Имя может быть таким



5. Далее необходимо юзернейм бота, например, такое:



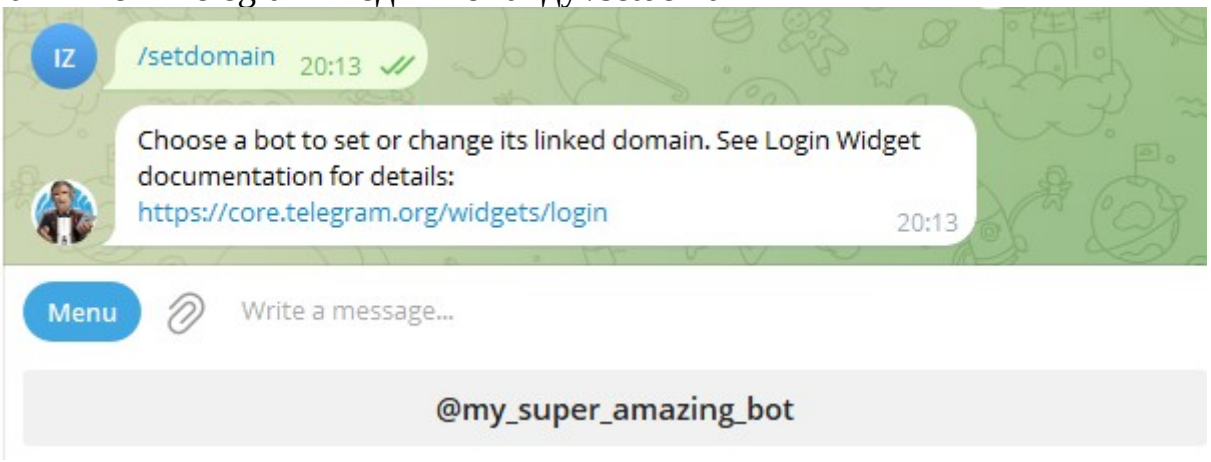
6. Из этого сообщения копируем токен



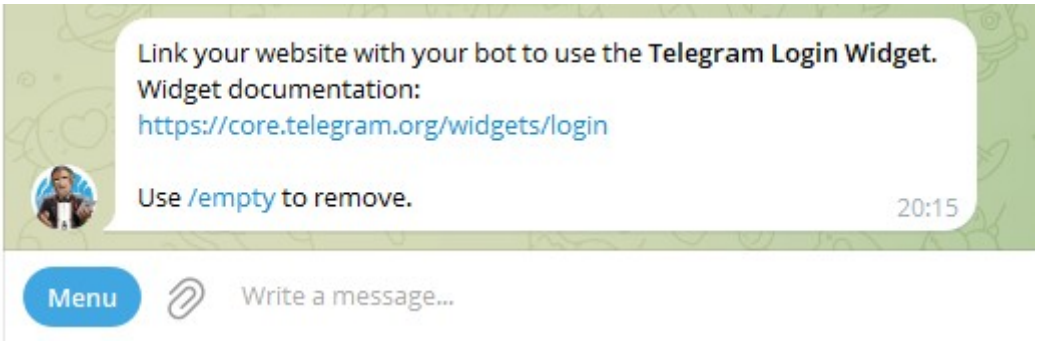
7. Вставляем этот токен в .env файл

```
# Telegram OAuth
BOT_TOKEN=7261288848:AAEFwiJcrnlOOTweAinXWZDMmfHvg1wVloY
```

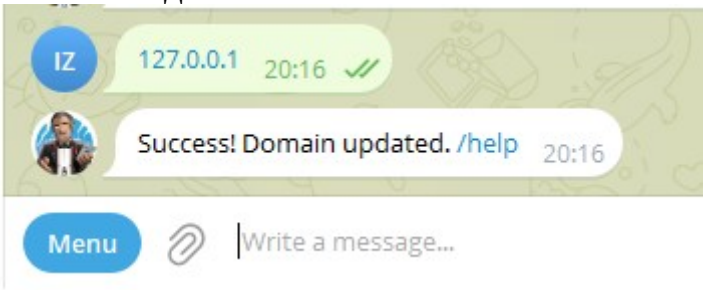
8. Там в же в Telegram вводим команду /setdomain



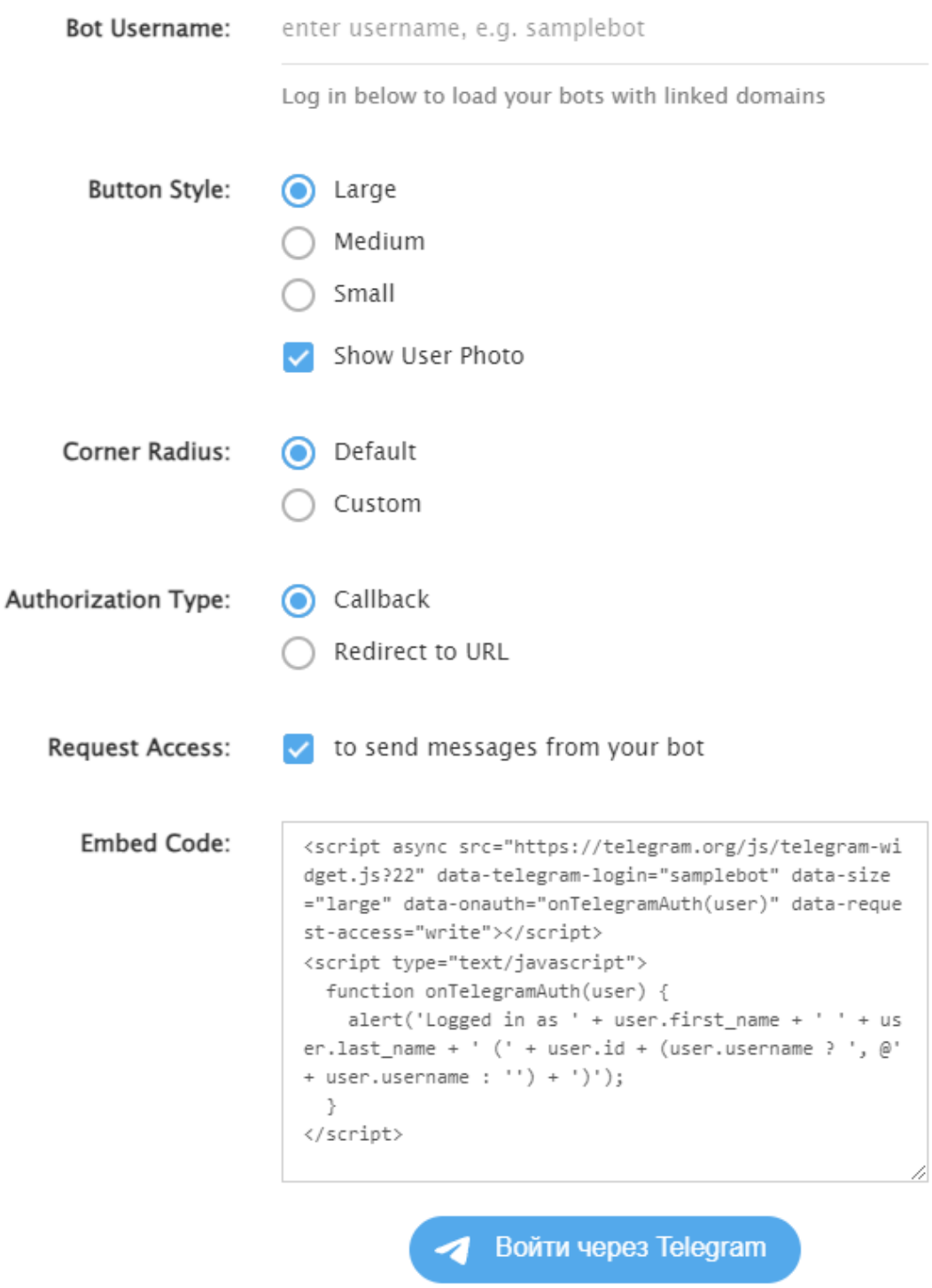
9. Выбираем нашего бота



10. Вводим ссылку на наш эндпоинт



11. Переходим по ссылке https://core.telegram.org/widgets/login



Здесь нам надо настроить внешний вид кнопки на нашем сайте.

12. Копируем скрипт и вставляем туда данные нашего бота:

```
<script async src="https://telegram.org/js/telegram-widget.js?22" data-telegram-login="my_super_amazing_bot" data-
```

```
size="large" data-auth-url="https://localhost/telegram-
callback" data-request-access="write"></script>
```

13. Теперь этот скрипт помещаем в тот шаблон, где хотим видеть кнопку «Войти через Telegram»

14. Теперь работаем во views.py
И для начала нам надо импортировать библиотеки для проверки безопасности:

```
import hashlib
import hmac
```

15. В глобальной области производим кодирование токена

```
BOT_TOKEN_HASH =
hashlib.sha256(os.environ['BOT_TOKEN'].encode())
```

16. Далее реализуем view-функцию для получения данных от Telegram:

```
@auth.route('/telegram-callback')
def telegram_callback():
  """Reginster or sign in by means of Telegram
Account."""
  username = request.args.get('username')
  social_uid = str(request.args.get('id'))
  email = get_random_email()
  if username:
    # check security
    query_hash = request.args.get('hash')
    params = request.args.items()
    data_check_string = '\n'.join(
      sorted(f'{x}={y}' for x, y in params if x not in
('hash', 'next')))
    computed_hash = hmac.new(
      BOT_TOKEN_HASH.digest(),
      data_check_string.encode(),
      'sha256').hexdigest()
    is_correct = hmac.compare_digest(computed_hash,
query_hash)
    if not is_correct:
```

```
        current_app.logger.exception('Telegram HASH
problem')
        abort(403)
```

Мы проверили безопасность (это требование официально документации Telegram) и получили от Telegram все данные о пользователе.

10. Заключение

Обратите внимание, что дополнительная проверка требуется только для Telegram потому, что мы отправляем только GET запрос.
В других сервисах мы отправляем POST запросы по зашифрованному протоколу HTTPS, что гарантирует сохранность данных.

С течением времени описание реализации может меняться, поэтому необходимо всегда обращаться к официальной документации по OAuth 2.0. Особое внимание следует уделять требованиям безопасности.