

Пример интеграции Django (Python)

В этой статье мы рассмотрим пример реализации авторизации через внешний API с использованием Django. Клиент отправляет номер телефона, получает код для авторизации через прокси-сервер, а затем проверяет результат авторизации с регулярными запросами. В конце, после успешной авторизации, пользователь получает токен.

Шаг 1: Настройка маршрутов в Django

Для начала нам нужно настроить маршруты (URLs), которые будут обрабатывать запросы от клиента. Мы создадим три маршрута:

- **/proxy/auth/code** — для запроса кода авторизации на основе номера телефона.
- **/proxy/auth/result** — для проверки статуса авторизации через `code_id`.
- **/proxy/auth/complete** — для завершения авторизации и выдачи токена после успешного завершения процесса.

Настройка маршрутов производится в файле `urls.py`:

```
from django.urls import path
from . import views

urlpatterns = [
    path('proxy/auth/code', views.get_code, name='get_code'),
    path('proxy/auth/result', views.get_result, name='get_result'),
    path('proxy/auth/complete', views.complete_auth, name='complete_auth'),
]
```

Шаг 2: Создание представлений (views) для обработки запросов

В Django мы будем использовать представления для обработки запросов. Мы создадим три представления: для запроса кода, проверки результата и завершения авторизации с выдачей токена.

```
import requests
from django.http import JsonResponse
```

```

from django.views.decorators.csrf import csrf_exempt

API_URL = 'https://api.auth4app.com'
API_KEY = 'your_api_key' # Замените на ваш реальный API-ключ

@csrf_exempt
def get_code(request):
    if request.method == 'POST':
        phone = request.POST.get('phone')
        response = requests.post(f'{API_URL}/code/get', data={
            'api_key': API_KEY,
            'phone': phone
        })
        return JsonResponse(response.json())
    return JsonResponse({'error': 'Invalid request method'}, status=400)

@csrf_exempt
def get_result(request):
    if request.method == 'POST':
        code_id = request.POST.get('code_id')
        response = requests.post(f'{API_URL}/code/result', data={
            'api_key': API_KEY,
            'code_id': code_id
        })
        return JsonResponse(response.json())
    return JsonResponse({'error': 'Invalid request method'}, status=400)

@csrf_exempt
def complete_auth(request):
    if request.method == 'POST':
        code_id = request.POST.get('code_id')

        # 1. Сначала вызываем метод для получения результата авторизации
        response = requests.post(f'{API_URL}/code/result', data={
            'api_key': API_KEY,
            'code_id': code_id
        })
        auth_data = response.json()

        if auth_data.get('auth') == True:

```

```
# 2. Получаем номер телефона из ответа
```

```
phone = auth_data.get('phone')
```

```
if phone:
```

```
    # 3. Поиск пользователя по номеру телефона или его создание
```

```
    user = User.objects.filter(phone=phone).first()
```

```
    if not user:
```

```
        # Если пользователь не найден, создаем нового
```

```
        user = User.objects.create(phone=phone)
```

```
        # Можно добавить другие поля, такие как имя, email и т.д.
```

```
    # 4. Здесь можно сгенерировать токен для пользователя
```

```
    token = "ТУТ ВАШ ТОКЕН" # На этом этапе можно создать токен
```

```
    # Возвращаем ответ с токеном и информацией о пользователе
```

```
    return JsonResponse({
```

```
        'token': token, # Верните сгенерированный токен здесь
```

```
        'user': {
```

```
            'id': user.id,
```

```
            'phone': user.phone,
```

```
            # можно вернуть другие данные пользователя
```

```
        }
```

```
    })
```

```
    return JsonResponse({'error': 'Phone number not found in the response'}, status=400)
```

```
else:
```

```
    return JsonResponse({'error': 'Authorization failed or not completed'}, status=400)
```

```
return JsonResponse({'error': 'Invalid request method'}, status=400)
```

Шаг 3: Логика клиентской стороны

На клиентской стороне необходимо реализовать логику для выполнения последовательных шагов:

1. Отправка номера телефона для получения кода авторизации.
2. Параллельное отправление запросов для проверки результата авторизации с интервалом в 3 секунды.
3. Как только в ответе появится `auth: true`, запрос завершения авторизации с передачей `code_id`.

1. Отправка номера телефона для запроса кода

Клиент отправляет POST-запрос на `/proxy/auth/code`, передавая номер телефона:

```
fetch('/proxy/auth/code', {
  method: 'POST',
  headers: {
    'Content-Type': 'application/x-www-form-urlencoded',
  },
  body: new URLSearchParams({ phone: '+1234567890' }) // Замените на реальный номер телефона
})
.then(response => response.json())
.then(data => {
  if (data.code_id) {
    // Начинаем проверку статуса авторизации каждые 3 секунды
    const codeId = data.code_id;
    const intervalId = setInterval(() => {
      checkAuthStatus(codeId, intervalId);
    }, 3000);
  }
});
```

2. Периодическая проверка статуса авторизации

Клиент начинает проверять статус авторизации, отправляя запросы на `/proxy/auth/result`:

```
function checkAuthStatus(codeId, intervalId) {
  fetch('/proxy/auth/result', {
    method: 'POST',
    headers: {
      'Content-Type': 'application/x-www-form-urlencoded',
    },
    body: new URLSearchParams({ code_id: codeId })
  })
  .then(response => response.json())
  .then(data => {
    if (data.auth === true) {
      // Останавливаем проверку
    }
  });
}
```

```
clearInterval(intervalId);

// Завершаем авторизацию
completeAuth(codeId);
}
});
}
```

3. Завершение авторизации и получение токена

Как только авторизация будет успешной, клиент отправляет запрос на `/proxy/auth/complete`:

```
function completeAuth(codeId) {
  fetch('/proxy/auth/complete', {
    method: 'POST',
    headers: {
      'Content-Type': 'application/x-www-form-urlencoded',
    },
    body: new URLSearchParams({ code_id: codeId })
  })
  .then(response => response.json())
  .then(data => {
    if (data.token) {
      // Токен получен, можно использовать для дальнейшей работы
      console.log('Authorization successful, token:', data.token);
    }
  });
}
```

Заключение

Мы рассмотрели пример интеграции Django с внешним API для авторизации. В процессе клиент отправляет номер телефона, получает код для авторизации и затем проверяет результат с использованием `code_id`. Когда авторизация завершается успешно, клиент получает токен для дальнейшей работы.

Порядок действий:

1. Клиент отправляет номер телефона через прокси-метод `/proxy/auth/code`.

2. Клиент запрашивает результат авторизации через `/proxy/auth/result`, передавая `code_id`.
 3. Как только авторизация завершена (`auth: true`), клиент вызывает метод завершения авторизации `/proxy/auth/complete`.
 4. Сервер выдает клиенту токен для дальнейшей работы.
-

Revision #4

Created 9 October 2024 01:51:48 by Agent Auth4App

Updated 9 October 2024 02:22:34 by Agent Auth4App