Пример интеграции Yii2

В этой статье мы рассмотрим пример реализации авторизации через внешний API с использованием Yii2. 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 Yii2

Для начала нам нужно настроить маршруты для обработки запросов 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 `config/web.php`:

```
'components' => [
    // Прочие компоненты
],
'urlManager' => [
    'enablePrettyUrl' => true,
    'showScriptName' => false,
    'rules' => [
        'POST /proxy/auth/code' => 'auth-proxy/get-code',
        'POST /proxy/auth/result' => 'auth-proxy/get-result',
        'POST /proxy/auth/complete' => 'auth-proxy/complete-auth',
    ],
],
```

Шаг 2: Создание контроллера для обработки запросов

Контроллер будет проксировать запросы к внешнему API с использованием вашего `api_key`. В контроллере мы создадим методы для каждого шага процесса авторизации: получение кода, проверка результата и завершение авторизации.

```php
namespace app\controllers;

use Yii;
use yii\web\Controller;
use yii\web\Response;
use yii\httpclient\Client;

class AuthProxyController extends Controller
{
    private $apiUrl = 'https://api.auth4app.com';
    private $apiKey = 'your_api_key';  // Замените на ваш реальный API-ключ

    // Метод для запроса кода авторизации
    public function actionGetCode()
    {
        Yii::$app->response->format = Response::FORMAT_JSON;

        $phone = Yii::$app->request->post('phone');
        $client = new Client();

        $response = $client->createRequest()
            ->setMethod('POST')
            ->setUrl($this->apiUrl . '/code/get')
            ->setData([
                'api_key' => $this->apiKey,
                'phone' => $phone,
            ])
            ->send();

        return $response->isOk ? $response->data : ['error' => 'API request failed'];
    }

    // Метод для проверки результата авторизации
    public function actionGetResult()
    {
        Yii::$app->response->format = Response::FORMAT_JSON;

        $codeId = Yii::$app->request->post('code_id');
```

```
        $client = new Client();

        $response = $client->createRequest()
            ->setMethod('POST')
            ->setUrl($this->apiUrl . '/code/result')
            ->setData([
                'api_key' => $this->apiKey,
                'code_id' => $codeId,
            ])
            ->send();

        return $response->isOk ? $response->data : ['error' => 'API request failed'];
    }

    // Метод для завершения авторизации и выдачи токена
    public function actionCompleteAuth()
    {
        Yii::$app->response->format = Response::FORMAT_JSON;

        $codeId = Yii::$app->request->post('code_id');
        $client = new Client();

        // 1. Сначала вызываем метод для получения результата авторизации
        $authResponse = $client->createRequest()
            ->setMethod('POST')
            ->setUrl($this->apiUrl . '/code/result')
            ->setData([
                'api_key' => $this->apiKey,
                'code_id' => $codeId,
            ])
            ->send();

        if ($authResponse->isOk && isset($authResponse->data['auth']) && $authResponse->data['auth'] ===
true) {
            // 2. Получаем номер телефона из ответа
            $phone = $authResponse->data['phone'] ?? null;

            if ($phone) {
                // 3. Поиск пользователя по номеру телефона или его создание
                $user = User::findOne(['phone' => $phone]);
```

```
            if (!$user) {
                // Если пользователь не найден, создаем нового
                $user = new User();
                $user->phone = $phone;
                // можно также добавить другие поля, например, имя, email и т.д.
                $user->save();
            }

            // 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 [
                'token' => $token,  // Верните сгенерированный токен здесь
                'user' => $user,    // Возвращаем информацию о пользователе
            ];
        } else {
            return ['error' => 'Phone number not found in the response'];
        }
    } else {
        return ['error' => 'Authorization failed or not completed'];
    }
  }
}
```

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/json',
    },
    body: JSON.stringify({ 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/json',
        },
        body: JSON.stringify({ 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/json',
        },
        body: JSON.stringify({ 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 Yii2 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 #6
Created 9 October 2024 01:34:39 by Agent Auth4App
Updated 9 October 2024 02:23:42 by Agent Auth4App