

Пример интеграции Laravel

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

Для начала нам нужно создать маршруты, 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.

Пример маршрутов в файле `routes/web.php`:

```
use App\Http\Controllers\AuthProxyController;

Route::post('/proxy/auth/code', [AuthProxyController::class, 'getCode']);
Route::post('/proxy/auth/result', [AuthProxyController::class, 'getResult']);
Route::post('/proxy/auth/complete', [AuthProxyController::class, 'completeAuth']);
```

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.

```
namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Support\Facades\Http;
```

```
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 getCode(Request $request)
    {
        $response = Http::post($this->apiUrl . '/code/get', [
            'api_key' => $this->apiKey,
            'phone' => $request->input('phone'), // Номер телефона от клиента
        ]);

        return response()->json($response->json());
    }

    // Метод для проверки результата авторизации
    public function getResult(Request $request)
    {
        $response = Http::post($this->apiUrl . '/code/result', [
            'api_key' => $this->apiKey,
            'code_id' => $request->input('code_id'), // Используем code_id для проверки результата
        ]);

        return response()->json($response->json());
    }

    // Метод для завершения авторизации и выдачи токена
    public function completeAuth(Request $request)
    {
        $code_id = $request->input('code_id');

        // 1. Сначала вызываем метод для получения результата авторизации
        $authResponse = Http::post($this->apiUrl . '/code/result', [
            'api_key' => $this->apiKey,
            'code_id' => $code_id,
        ]);

        // Преобразуем ответ в JSON
        $authData = $authResponse->json();
    }
}
```

```

// Проверяем, если авторизация успешна
if (isset($authData['auth']) && $authData['auth'] === true) {
    // 2. Получаем номер телефона из ответа
    $phone = $authData['phone'] ?? null;

    if ($phone) {
        // 3. Поиск пользователя по номеру телефона или его создание
        $user = User::where('phone', $phone)->first();

        if (!$user) {
            // Если пользователь не найден, создаем нового
            $user = User::create([
                'phone' => $phone,
                // можно также добавить другие поля, например, 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 response()->json([
            'token' => $token, // Верните сгенерированный токен здесь
            'user' => $user,    // Возвращаем информацию о пользователе
        ]);
    } else {
        // Если номер телефона не был возвращен
        return response()->json(['error' => 'Phone number not found in the response'], 400);
    }
} else {
    // Если авторизация не успешна
    return response()->json(['error' => 'Authorization failed or not completed'], 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/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 Laravel 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 #8

Created 9 October 2024 01:28:03 by Agent Auth4App

Updated 9 October 2024 02:23:07 by Agent Auth4App