Пример интеграции PHP (без фреймворков)

В этом примере вся логика обработки запроса на авторизацию, проверка результата и завершение авторизации реализованы в одном PHP-файле. Клиент передает параметр `action` для выбора нужного метода.

PHP логика для обработки запросов

```
$apiUrl = "https://api.auth4app.com";
$apiKey = "your_api_key"; // Замените на ваш реальный API-ключ

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $action = $_POST['action'] ?? null;

    switch ($action) {
        case 'get_code':
            getCode($_POST['phone']);
            break;
        case 'get_result':
            getResult($_POST['code_id']);
            break;
        case 'complete_auth':
            completeAuth($_POST['code_id']);
            break;
        default:
            http_response_code(400);
            echo json_encode(['error' => 'Invalid action']);
    }
} else {
    http_response_code(405); // Метод не разрешен
    echo json_encode(['error' => 'Method not allowed']);
}

// Функция для запроса кода авторизации
function getCode($phone) {
    global $apiUrl, $apiKey;
```

```
    $url = $apiUrl . "/code/get";
    $postData = json_encode(['api_key' => $apiKey, 'phone' => $phone]);

    $response = sendPostRequest($url, $postData);
    echo $response;
}

// Функция для проверки результата авторизации
function getResult($codeId) {
    global $apiUrl, $apiKey;
    $url = $apiUrl . "/code/result";
    $postData = json_encode(['api_key' => $apiKey, 'code_id' => $codeId]);

    $response = sendPostRequest($url, $postData);
    echo $response;
}

// Функция для завершения авторизации и выдачи токена
function completeAuth($codeId) {
    global $apiUrl, $apiKey;
    $url = $apiUrl . "/code/result";
    $postData = json_encode(['api_key' => $apiKey, 'code_id' => $codeId]);

    // 1. Получаем результат авторизации
    $response = sendPostRequest($url, $postData);
    $authData = json_decode($response, true);

    if ($authData['auth'] === true) {
        $phone = $authData['phone'] ?? null;

        if ($phone) {
            // 2. Поиск пользователя по номеру телефона или его создание
            $user = findUserByPhone($phone);
            if (!$user) {
                $user = createUser($phone);
            }

            // 3. Генерация токена (это можно сделать через JWT или другой метод)
            $token = "ТУТ ВАШ ТОКЕН"; // На этом этапе можно создать токен
```

```
            // 4. Возвращаем токен и данные пользователя
            echo json_encode([
                'token' => $token,
                'user' => [
                    'id' => $user['id'],
                    'phone' => $user['phone']
                ]
            ]);
        } else {
            http_response_code(400);
            echo json_encode(['error' => 'Phone number not found']);
        }
    } else {
        http_response_code(400);
        echo json_encode(['error' => 'Authorization failed']);
    }
}

// Функция для отправки POST-запросов
function sendPostRequest($url, $postData) {
    $ch = curl_init($url);
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
    curl_setopt($ch, CURLOPT_HTTPHEADER, ['Content-Type: application/json']);
    curl_setopt($ch, CURLOPT_POST, true);
    curl_setopt($ch, CURLOPT_POSTFIELDS, $postData);

    $response = curl_exec($ch);
    curl_close($ch);

    return $response;
}

// Пример функций для поиска и создания пользователя
function findUserByPhone($phone) {
    // Здесь должна быть логика поиска пользователя по номеру телефона в базе данных
    return null; // Если пользователь не найден
}

function createUser($phone) {
    // Здесь должна быть логика создания нового пользователя
```

```
    return ['id' => 1, 'phone' => $phone]; // Пример возврата пользователя
}
```

Шаг 3: Логика клиентской стороны

Теперь на клиентской стороне передаем параметр `action` для выбора нужного действия, используя `FormData`.

1. Отправка номера телефона для получения кода

Клиент отправляет POST-запрос с действием `get_code`, передавая номер телефона:

```
const formData = new FormData();
formData.append('action', 'get_code');
formData.append('phone', '+1234567890'); // Замените на реальный номер телефона

fetch('/auth.php', {
    method: 'POST',
    body: formData
})
.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 отправляет POST-запрос с действием `get_result`, передавая `code_id`:

```
function checkAuthStatus(codeId, intervalId) {
    const formData = new FormData();
    formData.append('action', 'get_result');
```

```
    formData.append('code_id', codeId);

    fetch('/auth.php', {
        method: 'POST',
        body: formData
    })
    .then(response => response.json())
    .then(data => {
        if (data.auth === true) {
            // Останавливаем проверку
            clearInterval(intervalId);

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

3. Завершение авторизации и получение токена

Клиент отправляет POST-запрос с действием `complete_auth`, передавая `code_id`:

```
function completeAuth(codeId) {
    const formData = new FormData();
    formData.append('action', 'complete_auth');
    formData.append('code_id', codeId);

    fetch('/auth.php', {
        method: 'POST',
        body: formData
    })
    .then(response => response.json())
    .then(data => {
        if (data.token) {
            // Токен получен, можно использовать для дальнейшей работы
            console.log('Authorization successful, token:', data.token);
        }
    });
```

```
}
```

Заключение

В этом примере мы рассмотрели, как объединить всю логику обработки запросов на авторизацию, проверку результата и завершение авторизации в одном PHP-файле. Клиент передает действие через параметр `action` для выполнения нужного метода.

Порядок действий:

1. Клиент отправляет номер телефона через действие `get_code`.
2. Клиент запрашивает результат авторизации через действие `get_result`, передавая `code_id`.
3. Как только авторизация завершена (`auth: true`), клиент вызывает метод завершения авторизации через действие `complete_auth`.
4. Сервер выдает клиенту токен для дальнейшей работы.

Revision #3
Created 9 October 2024 01:54:08 by Agent Auth4App
Updated 9 October 2024 02:29:49 by Agent Auth4App