

Using AI Powered Tool

Haram Lee

이게 서버 상태야

리액트 네이티브 프론트에서 파일 전송하는법 알려줘



React Native에서 파일을 Express 서버로 전송하려면, **Axios** 또는 **fetch**를 사용하여 파일을 **FormData** 형식으로 서버에 보내야 합니다. 위에서 보여주신 서버 코드에서 파일을 다중으로 처리하고 있으므로, 파일 배열을 서버에 전송하는 방식으로 구현할 수 있습니다.

1. Axios를 사용한 파일 전송

React Native 클라이언트 코드

다음은 **Axios**를 사용해 여러 파일을 서버로 전송하는 예시입니다. 여기서는 **react-native-image-picker**로 파일을 선택한 후 서버로 전송하는 과정을 설명합니다.

FileUploadScreen.tsx

```
tsx 코드 복사

import React, { useState } from 'react';
import { View, Button, Image, Text } from 'react-native';
import { launchImageLibrary } from 'react-native-image-picker';
import axios from 'axios';

const FileUploadScreen = () => {
  const [selectedImages, setSelectedImages] = useState<any[]>([]);
  const [uploadResponse, setUploadResponse] = useState<string | null>(null);

  // 파일 선택 함수
  const selectImages = () => {
    launchImageLibrary(
      {
        mediaType: 'photo',
        selectionLimit: 100, // 다중 파일 선택 허용
      },
    ),
  },
};
```

2. 설명

1. 파일 선택:

- launchImageLibrary**를 사용하여 여러 이미지를 선택합니다.
- selectionLimit** 옵션을 사용하여 사용자가 한 번에 여러 이미지를 선택할 수 있도록 설정합니다.

2. FormData 구성:

- 선택된 이미지를 **FormData**에 추가합니다. **uri**, **name**, **type** 필드를 지정하여 서버로 전송할 수 있도록 설정합니다.
- files**라는 필드 이름을 사용하여 서버로 전송하는데, 이는 서버의 **upload.array("files", 100)**와 매칭됩니다.

3. Axios 요청:

- POST** 요청으로 파일을 서버에 전송하며, 요청의 헤더에 **Content-Type: multipart/form-data**를 설정합니다.
- 파일 업로드가 성공하면 서버의 응답 메시지를 받아서 클라이언트에 표시합니다.

3. 서버 측 파일 처리

서버에서는 **Multer** 미들웨어를 통해 파일 배열을 처리하고, 이를 원하는 저장소에 업로드한 후 응답을 반환합니다. 이 부분은 이미 서버 코드에서 처리하고 있기 때문에 React Native 클라이언트는 이에 맞게 파일을 전송하기만 하면 됩니다.

결론

React Native에서 여러 파일을 선택하여 **Axios**를 통해 **FormData**로 서버에 전송하는 방법을 설명했습니다. 서버는 **Multer**를 사용하여 다중 파일을 처리하고 있으므로, 클라이언트는 **files** 필드로 파일 배열을 전송하면 됩니다.

최근 프로젝트에서 AI(특히 OpenAI의 ChatGPT)를 활용하여 React Native 파일 업로드 기능을 구현하고, 이를 Express 백엔드와 연동하였습니다. 클라이언트에서는 Axios를 사용해 파일을 전송하고, 서버에서는 Multer로 파일을 처리하는 구조였습니다. AI는 파일 업로드를 위한 컴포넌트 구성과 서버 연동 방법을 이해하고 구현하는 데 큰 도움이 되었습니다.

문제가 발생한 부분은 React Native에서 FormData를 올바르게 구성해 서버에서 파일을 정상적으로 받을 수 있도록 하는 과정이었습니다. 처음에는 이미지의 URI 경로와 MIME 타입 처리가 잘못되어 파일 업로드가 실패했었습니다. AI의 도움으로 문제의 원인을 빠르게 파악하고, 안드로이드와 iOS 플랫폼별로 FormData의 경로 처리를 수정하였습니다.

또한, 요청 헤더 설정에서 발생한 문제도 AI가 제공한 명확한 지침을 통해 해결할 수 있었습니다. AI는 복잡한 파일 처리 로직을 단순화하고, 정확한 코드 예시와 디버깅 팁을 제공해 시간을 절약할 수 있었습니다. 이를 통해 파일 업로드 기능을 성공적으로 구현할 수 있었고, 추가적인 자료 조사 없이 빠르게 문제를 해결할 수 있었습니다.