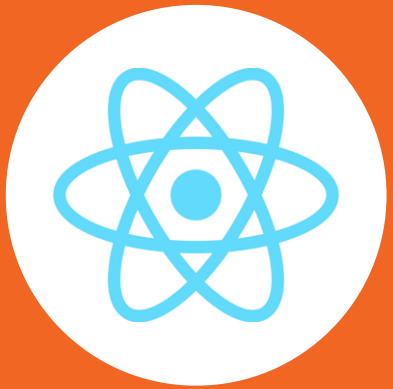
FPT Education

FPT POLYTECHNIC



LẬP TRÌNH ĐA NỀN TẢNG VỚI REACT NATIVE

BÀI 4: GIỚI THIỆU VỀ ỨNG DỤNG CHỤP ẢNH, LẤY ẢNH VÀ PHÁT NHẠC

PHẦN 1: CHỤP ẢNH VÀ LẤY ẢNH TỪ THƯ VIỆN

Mục tiêu

- [ ] Cài đặt thư viện chụp, lấy hình ảnh
- [ ] Tạo ứng dụng chụp ảnh
- [ ] Tạo ứng dụng lấy ảnh từ thư viện

Giới thiệu

- Chụp ảnh và lấy ảnh từ ứng dụng của điện thoại là một chức năng phổ biến, hầu như mọi ứng dụng nào đều có. Để có thể sử dụng được chức năng này trong ứng dụng React Native bạn cần cài đặt thư viện thứ 3, có khá nhiều thư viện hỗ trợ bạn làm điều này. Nhưng khuyến nghị bạn nên sử dụng thư viện react-native-image-picker
  - react-native-image-picker tương thích với iOS/Android với hỗ trợ máy ảnh, video.

Cài đặt thư viện

- Chạy lệnh sau để cài đặt thư viện

```
npm i react-native-image-picker --save
```

- Với thiết bị Android, bạn không cần phải khai báo cấp quyền (saveToPhotos yêu cầu kiểm tra quyền).
- Lưu ý: Thư viện này không yêu cầu Manifest.permission.CAMERA, nếu ứng dụng của bạn khai báo là sử dụng quyền này trong manifest thì bạn phải có quyền trước khi sử dụng launchCamera.

Chụp ảnh

launchCamera()

```
import {launchCamera} from 'react-  native-image-picker';
```

- Khởi động camera để chụp ảnh hoặc quay video

```
launchCamera(options?, callback);

// Bạn có thể sử dụng một promise mà không cần 'callback':
const result = await launchCamera(options?);
```

Chọn ảnh

launchImageLibrary()

```
import {launchImageLibrary} from 'react-  native-image-picker';
```

- Khởi chạy thư viện để chọn hình ảnh hoặc video

```
launchImageLibrary(options?, callback)

// You can also use as a promise without 'callback':
const result = await launchImageLibrary(options?);
```

Options

- Tuỳ vào mục đích sử dụng camera, hay chọn ảnh từ thư viện thì bạn sẽ có những tuỳ chỉnh trong options của module. Dưới đây là danh sách các options tuỳ chỉnh của react-native-image-picker

Option	Mô tả
mediaType	photo hoặc video hoặc mixed (launchCamera trên Android không hỗ trợ 'mixed').
maxWidth	Để thay đổi kích thước hình ảnh
maxHeight	Để thay đổi kích thước hình ảnh

Options

videoQuality	low, medium, hoặc high trên iOS, low hoặc high trên Android.
durationLimit	Thời lượng tối đa của video (tính bằng giây)
quality	0 đến 1
cameraType	'back' or 'front' (Có thể không hỗ trợ trên một số thiết bị Android).
includeBase64	Nếu true, hãy tạo chuỗi base64 của hình ảnh (Tránh sử dụng trên các tệp hình ảnh lớn do ảnh hưởng đến performance).

Options

includeExtra	Nếu đúng, sẽ bao gồm dữ liệu bổ sung, yêu cầu quyền thư viện.
saveToPhotos	Chỉ để khởi chạy launchCamera, lưu tệp hình ảnh / video được chụp vào ảnh công khai.
selectionLimit	Hỗ trợ cung cấp bất kỳ giá trị số nguyên nào. Sử dụng 0 để cho phép bất kỳ số lượng tệp nào trên iOS phiên bản > = 14 & Android phiên bản > = 13. Mặc định là 1.
presentationStyle	Kiểm soát cách trình bày bộ chọn. currentContext, pageSheet, fullScreen, formSheet, popover, overFullScreen, overCurrentContext. Mặc định là currentContext.

Giá trị trả về

- Khi bạn gọi hàm launchCamera hoặc launchImageLibrary. Kết quả trả về sẽ là một object chứa các thông tin sau:
  - didCancel: true nếu người dùng hủy quá trình
  - errorCode: Kiểm tra ErrorCode cho tất cả các mã lỗi

Code	Mô tả
camera_unavailable	Máy ảnh không có sẵn trên thiết bị
permission	Quyền không được đáp ứng
others	Các lỗi khác (kiểm tra errorMessage để biết mô tả)

Giá trị trả về

- errorMessage: Mô tả lỗi, chỉ sử dụng nó cho mục đích gỡ lỗi
- assets: Mảng media đã chọn, tham khảo Asset Object

key	Photo/Video	Mô tả
base64	PHOTO ONLY	Chuỗi base64 của hình ảnh
uri	BOTH	Uri tệp trong bộ nhớ cache dành riêng cho ứng dụng. Ngoại trừ khi chọn video từ thư viện Android, nơi bạn sẽ nhận được uri nội dung chỉ đọc, để lấy uri tệp trong trường hợp này, hãy sao chép tệp vào bộ nhớ cụ thể của ứng dụng bằng bất kỳ thư viện react-native nào.

Giá trị trả về

originalPath (Android)	BOTH	Đường dẫn tập tin gốc
width	BOTH	Kích thước của hình ảnh
height	BOTH	Kích thước của hình ảnh
fileSize	BOTH	Kích thước tệp
type	BOTH	Loại tệp
fileName	BOTH	Tên tệp

Giá trị trả về

duration	VIDEO ONLY	Thời lượng video đã chọn tính bằng giây
bitrate (Android)	VIDEO ONLY	Tốc độ bit trung bình (tính bằng bit/giây) của video đã chọn, nếu có.
timestamp	BOTH	Timestamp của file. Chỉ được bao gồm nếu 'includeExtra' là true
id	BOTH	Số nhận dạng cục bộ của ảnh hoặc video. Trên Android, điều này giống như tên tệp
type	BOTH	Loại tệp
fileName	BOTH	Tên tệp

Chụp ảnh

- Các bạn đã được giới thiệu sơ lược tất cả những gì mà thư viện react-native-image-picker cung cấp. Bây giờ chúng ta sẽ bắt đầu xây dựng một ứng dụng chụp ảnh:
  - Đầu tiên, các bạn cần khai báo các option của camera chúng ta cần.

```
// Đây là option sẽ sử dụng chung cả chụp ảnh và chọn ảnh
const commonOptions: OptionsCommon = {
        mediaType: 'photo',
        maxWidth: 500,
        maxHeight: 500,
    };
```

Chụp ảnh

- cameraOptions: Cung cấp thêm option cameraType: “front” để chụp ảnh bằng camera trước, saveToPhotos: true để lưu hình ảnh trong thư viện

```
const cameraOptions: CameraOptions = {
    cameraType: 'front',
    saveToPhotos: true,
    ...commonOptions,
};
```

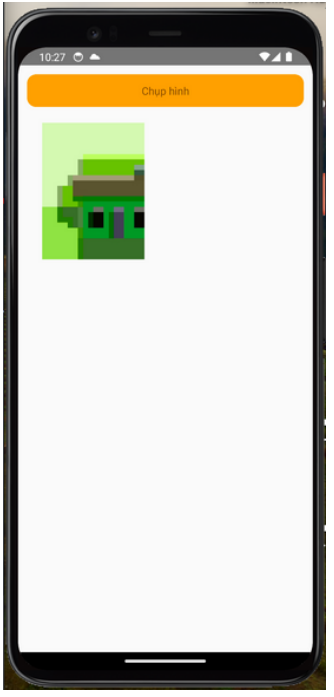
Chụp ảnh

- Hàm sử lý khi người dùng nhấn mở camera, bạn nên viết thêm thông báo lỗi khi người dùng thao tác với camera để người dùng biết được họ đang mắc lỗi gì.

DaNenTang2 - PhotoScreen.tsx

```
1  const onOpenCamera = async () => {
2    const response: ImagePickerResponse = await launchCamera(cameraOptions);
3    if (response?.assets) {
4      setImages(response.assets);
5    } else {
6      Alert.alert('Có lỗi xảy ra', response.errorMessage);
7    }
8  };
```

- Chạy chương trình và chúng ta có kết quả sau:



Chọn ảnh từ thư viện

- Thư viện react-native-image-picker cung cấp tuỳ chọn có thể chọn nhiều ảnh trong thư viện. Để làm điều này bạn cần khai báo option trong launchImageLibrary.
  - libraryOptions: Cung cấp thêm option selectionLimit: 10, giới hạn người dùng, chọn tối đa 10 ảnh

```
const libraryOptions: ImageLibraryOptions = {
 selectionLimit: 10,
 ...commonOptions,
};
```

Chọn ảnh từ thư viện

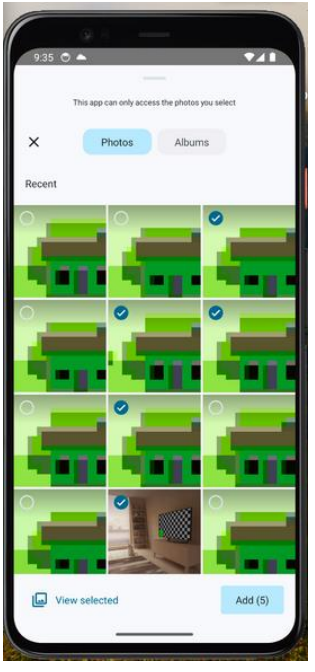
- Hàm sử lý khi người dùng nhấn mở thư viện ảnh, bạn nên viết thêm thông báo lỗi khi người dùng thao tác với thư viện ảnh để người dùng biết được họ đang mắc lỗi gì.

DaNenTang2 - PhotoScreen.tsx

```
1  const onOpenLibrary = async () => {
2    const response: ImagePickerResponse = await launchImageLibrary(
3      libraryOptions,
4    );
5    if (response?.assets) {
6      setImages(response.assets);
7    } else {
8      Alert.alert('Có lỗi xảy ra', response.errorMessage);
9    }
10 };
```

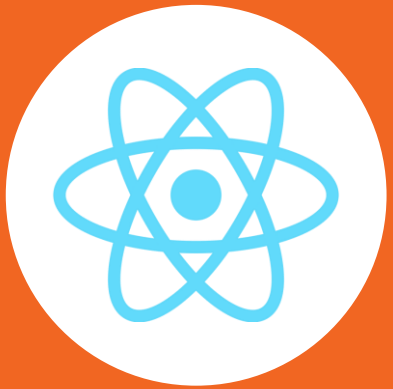
Chọn ảnh từ thư viện

- Reload lại ứng dụng và chúng ta có kết quả sau:



FPT Education

FPT POLYTECHNIC



LẬP TRÌNH ĐA NỀN TẢNG VỚI REACT NATIVE

BÀI 4: GIỚI THIỆU VỀ ỨNG DỤNG CHỤP ẢNH, LẤY ẢNH VÀ PHÁT NHẠC

PHẦN 2: TẠO ỨNG DỤNG PHÁT NHẠC

Mục tiêu

- [ ] Tìm hiểu về thư viện react-native-track-player
- [ ] Tạo một ứng dụng phát nhạc

Giới thiệu

- Các ứng dụng phát nhạc, là một phần không thể thiếu trong thế giới lập trình ứng dụng di động. Để phát triển các ứng dụng phát nhạc bạn cần tạo các native module, để có thể phát được âm thanh. Việc viết bằng native module này khá tốn nguồn lực. May thay, đã có một số thư viện đã hỗ trợ rất tốt cho việc phát âm thanh này như:
  - React Native Track Player: Đây là một thư viện mạnh mẽ cho phép phát nhạc từ local hoặc từ internet. Nó cung cấp các tính năng như phát, tạm dừng, tua nhạc, và thậm chí cả tạo playlist.
  - React Native Sound: Thư viện này cho phép phát các tập tin âm thanh đơn giản như MP3, WAV, và các định dạng khác. Nó cũng hỗ trợ các chức năng cơ bản như phát, tạm dừng và ngừng.

Giới thiệu

- ExoPlayer: Mặc dù không phải là một thư viện chính thức của React Native, ExoPlayer là một trình phát đa phương tiện mạnh mẽ từ Google và có thể được tích hợp vào ứng dụng React Native thông qua các gói bổ sung.

- Ở bài học này, các bạn sẽ hướng dẫn sử dụng thư viện react-native-track-player, ở thời điểm viết slide này, đây là thư viện hỗ trợ mãnh mẽ nhất cho việc phát nhạc.

React Native Track Player

Giới thiệu

Một mô-đun âm thanh chính thức được tạo cho các ứng dụng âm nhạc. Cung cấp phát lại âm thanh, điều khiển phương tiện bên ngoài, chế độ nền và hơn thế nữa!

Các tính năng

- Lightweight: Tối ưu hóa để sử dụng ít tài nguyên nhất theo nhu cầu của bạn
- Feels native: Vì mọi thứ được xây dựng cùng nhau, nó tuân theo các nguyên tắc thiết kế giống như các ứng dụng âm nhạc thực sự
- Multi-platform: Hỗ trợ Android, iOS và Windows

React Native Track Player

- Hỗ trợ Media Controls: Cung cấp các sự kiện để điều khiển ứng dụng từ thiết bị bluetooth, màn hình khóa, thông báo, đồng hồ thông minh hoặc thậm chí là ô tô
- Local hoặc network, files hoặc streams: Không quan trọng phương tiện thuộc về đâu, chúng tôi luôn sẵn sàng hỗ trợ bạn
- Hỗ trợ phát trực tuyến tốc độ bit thích ứng: Hỗ trợ DASH, HLS hoặc SmoothStreaming
- Hỗ trợ bộ nhớ đệm: Bộ nhớ cache các tệp phương tiện để phát lại chúng mà không cần kết nối internet
- Hỗ trợ background: Tiếp tục phát âm thanh ngay cả sau khi ứng dụng ở chế độ nền

- Tuỳ chỉnh tối đa: Ngay cả các biểu tượng thông báo cũng có thể tùy chỉnh!
- Hỗ trợ truyền: Sử dụng kết hợp với react-native-track-casting (WIP) để chuyển đổi liền mạch sang bất kỳ thiết bị tương thích nào của Google Cast hỗ trợ bộ thu phương tiện tùy chỉnh

Cài đặt

- Chạy lệnh sau để cài đặt thư viện:

```
npm install --save react-native-track-player
```

Setup

- Cài đặt hoàn tất, bây giờ các bạn cần một số setup ban đầu để thư viện có thể hoạt động được
- Dưới đây là 2 bước setup cơ bản bạn cần thực hiện:
- Bước 1, bạn cần đăng ký **playback service** ngay sau khi đăng ký thành phần chính của ứng dụng (thường là trong tệp index.js ở gốc dự án):

```
// AppRegistry.registerComponent(...);
TrackPlayer.registerPlaybackService(() => require('./service'));
```

```
// service.js
module.exports = async function() {
   // Service này cần được đăng ký để module hoạt động
   // nhưng nó sẽ được sử dụng sau trong phần 'Receiving Events'
}
```

- Bước 2, bạn cần thiết lập trình phát. Quá trình này thường mất ít hơn một giây:

```
import TrackPlayer from 'react-native-track-player';

await TrackPlayer.setupPlayer()
// Trình phát sẵn sàng để sử dụng
```

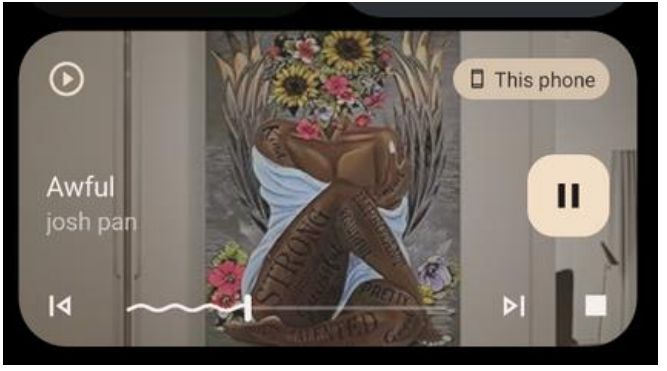
Setup

- Đảm bảo rằng phương pháp thiết lập đã hoàn tất trước khi tương tác với bất kỳ chức năng nào khác trong TrackPlayer để tránh sự mất ổn định.

- Nhưng trong thực các dự án thực tế bạn nên viết trình phát âm thanh ra một hook riêng, nơi đây chứa các setup, function điều khiển, trạng thái âm thanh mà bạn muốn. Cách viết ra hook cũng khiến code bạn clean hơn và dễ bảo trì nâng cấp sau này.
- Bây giờ chúng ta sẽ bắt đầu thực hiện các bước setup bằng custom hook:
  - Bước 1: Thêm các listener Media Controls vào registerPlaybackService

    Media Controls là các trình điều khiển bên ngoài ứng dụng, giúp bạn có thể bật, tắt, thay đổi thời lượng....

Setup



Viết hàm playbackService để khai báo các listen của sự kiện Media Controls, hàm này sẽ được import trong file gốc index.js

DaNenTang2 - trackService.ts

```
export async function playbackService() {
  TrackPlayer.addEventListener(Event.RemotePause, () => {
    TrackPlayer.pause();
  });

  TrackPlayer.addEventListener(Event.RemotePlay, () => {
    TrackPlayer.play();
  });
```

DaNenTang2 - trackService.ts

```
    TrackPlayer.addEventListener(Event.RemoteNext, () => {
      TrackPlayer.skipToNext();
    });

    TrackPlayer.addEventListener(Event.RemotePrevious, () => {
      TrackPlayer.skipToPrevious();
    });

    TrackPlayer.addEventListener(Event.RemoteSeek, ({position}) => {
      TrackPlayer.seekTo(position);
    });
}
```

Sau đó bạn sẽ import function này vào registerPlaybackService để đăng ký các event cho Media Controls

DaNenTang2 - index.js

```
AppRegistry.registerComponent(appName, () => App);

TrackPlayer.registerPlaybackService(() => playbackService);

```

- Bước 2: Viết hook usePlayTrack để gọi các hàm điều khiển âm thanh cho ứng dụng

Đầu tiên, viết hàm startPlayer để setup ban đầu cho trình phát nhạc, và khai báo các trình điều khiển cho Media Controls.

Setup

DaNenTang2 - trackService.ts

```
const startPlayer = async (setSetupDone: (isSetupDone: boolean) => void) => {
  try {
    await TrackPlayer.setupPlayer().finally(() => setSetupDone(true));
    await TrackPlayer.updateOptions({
      capabilities: [
        Capability.Play,
        Capability.Pause,
        Capability.Stop,
        Capability.SeekTo,
        Capability.SkipToNext,
        Capability.SkipToPrevious,
      ],
    });
    await TrackPlayer.setRepeatMode(RepeatMode.Off);
  } catch (error) {
    console.log('[Error player] ', error);
  }
};
```

Setup

Bạn luôn phải gọi hàm setupPlayer() đầu tiên, trước khi bắt đầu sử dụng các hàm nào của react-native-track-player.

updateOptions là các nút tính năng của Media Controls. Như bật, tắt, chuyển bài....

setRepeatMode cho phép lặp lại của âm thanh.

- Để sử dụng custom hook usePlayTrack, bạn cần phải truyền vào danh sách âm thanh cần phát (playListData). Hook usePlaybackState dùng để lấy trạng thái phát nhạc hiện tại, đang phát, hay đang dừng..., isSetupDone là true nếu như đã setup thành công

DaNenTang2 - trackService.ts

```
export const usePlayTrack = (playListData: AddTrack[]) => {
  const playBackState = usePlaybackState();
  const [isSetupDone, setSetupDone] = useState(false);
```

- Ở trong hook usePlayTrack, bạn cần lưu trữ những state của đoạn âm thanh như: vị trí, thời lượng, tên, hình ảnh... của đoạn âm thanh. Hook useProgress dùng để lấy thời lượng track đang phát.

DaNenTang2 - trackService.ts

```
const {duration, position} = useProgress();
const [trackTitle, setTrackTitle] = useState<string>();
const [trackArtist, setTrackArtist] = useState<string>();
const [trackArtwork, setTrackArtwork] = useState<string>();

```

Listen event

- Sử dụng hook useTrackPlayerEvents để bạn có thể lấy tên, hình ảnh,... của đoạn âm thanh phát hiện tại

DaNenTang2 - trackService.ts

```
useTrackPlayerEvents([Event.PlaybackActiveTrackChanged], async event => {
  const {title, artwork, artist} = event?.track || {};
  if (event.type === Event.PlaybackActiveTrackChanged && !!event?.track) {
    setTrackTitle(title);
    setTrackArtist(artist);
    setTrackArtwork(artwork);
  }
});
```

Khởi tạo trình phát âm thanh

- Sử dụng useEffect để gọi hàm startPlayer để setup cho trình phát nhạc. Nếu screen unmout gọi hàm reset() để xoá tất cả đoạn âm thanh trong trình phát nhạc.

DaNenTang2 - trackService.ts

```
1  useEffect(() => {
2    startPlayer(setSetupDone);
3    return () => {
4      TrackPlayer.reset();
5    };
6  }, []);
7
```

Thêm danh sách âm thanh

- Sau khi setup thành công, các bạn thêm danh sách nhạc của mình vào hàng chờ của TrackPlayer.

DaNenTang2 - trackService.ts

```
useEffect(() => {
  if (!!isSetupDone && !!playListData) {
    TrackPlayer.getActiveTrack().then(async activeTrack => {
      if (!activeTrack) {
        await TrackPlayer.add(playListData);
      }
    });
  }
}, [isSetupDone, playListData]);
```

Hàm phát/dừng âm thanh

- Sử dụng playBackState để xác định trạng thái phát nhạc hiện tại để dừng hoặc phát nhạc.

DaNenTang2 - trackService.ts

```
1  const onTogglePlayTrack = async () => {
2    if (playBackState.state === State.Playing) {
3      await TrackPlayer.pause();
4    } else {
5      await TrackPlayer.play();
6    }
7  };
8
```

Tua âm thanh đang phát

- Gọi hàm seekTo để tua đến đoạn âm thanh mong muốn. toTime là thời lượng tính bằng giây, mà bạn muốn tua đến

DaNenTang2 - trackService.ts

```
1  const onSeekTo = (toTime: number) => {
2    TrackPlayer.seekTo(toTime);
3  };
```

Tiếp theo/Trở lại bài

- Gọi hàm skipToNext để chuyển đến bài tiếp theo, gọi hàm skipToPrevious để trở lại bài trước.
- initialPosition là vị trí của bài trong hàng chờ mà bạn muốn chuyển đến.

DaNenTang2 - trackService.ts

```
1  const onSkipToNext = (initialPosition?: number) => {
2    TrackPlayer.skipToNext(initialPosition);
3  };
4
5  const onSkipToPrevious = (initialPosition?: number) => {
6    TrackPlayer.skipToPrevious(initialPosition);
7  };
```

return value

- Return các giá trị trong hook, bạn có thể lấy giá trị bất kì trong hook được return để sử dụng vào trình phát âm thanh của mình

DaNenTang2 - trackService.ts

```
return {
    onTogglePlayTrack,
    onSeekTo,
    onSkipToNext,
    onSkipToPrevious,
    playBackState: playBackState.state,
    duration,
    position,
    trackTitle,
    trackArtist,
    trackArtwork,
};
```

Kết luận

- Tới đây, bạn đã hoàn thành viết một custom hook để phát nhạc. Công việc còn lại của bạn chỉ cần gọi hook ra, và truyền thêm vào danh sách âm thanh mà bạn muốn phát. Tuỳ vào từng yêu cầu mà bạn sẽ xây dựng giao diện gắn với từng function, giá trị mà đã được xây dựng trong hook của chúng ta

Tổng kết

- Cài đặt thư viện chụp, lấy hình ảnh
- Tạo ứng dụng chụp ảnh
- Tạo ứng dụng lấy ảnh từ thư viện
- Tìm hiểu về thư viện react-native-track-player
- Tạo một ứng dụng phát nhạc

Kết thúc