1年半放置したExpo製アプリを最新化してみた

Press Space for next page →

自己紹介

- 📝 飯野陽平（wheatandcat)
- 🏢 会社: 合同会社UNICORN 代表社員
- 📚 Blog: https://www.wheatandcat.me/
- 🛠️ 今までに作ったもの
  - memoir
  - OOMAKA
  - MarkyLinky

あらすじ①

- 2022年に**memoir**をリリース
- 参考リンク
  - memoir
  - 【夫婦で開発】1年かけて1週間を振り返えるアプリを本気で開発してみた

あらすじ②

- 技術スタック
  - Expo
- Expoとは？
  - React Nativeを使用したアプリ開発を支える代表的なエコシステム
  - Expo SDKを使用することで、本来必要な開発をかなり省略できる
    - 省略できる開発の例
      - 公式がBuild環境のサーバーを提供
      - テストアプリの配布機能をサポート
      - Push通知等のプラットフォーム固有の開発をExpo側で抽象化して提供
      - etc...
  - 現在はReact Nativeの公式でもExpoを使用した開発を推奨している

あらすじ③

- 2023年までは定期的に更新していたが、それ以降は他の個人開発に時間を割いていて更新を止めていた
- 間に作っていた個人制作物
  - 自分専用todo
  - MarkyLinky
  - OOMAKA
  - STOCK KEEPER

開発を1年半サボると、どういうことが起きるのか？

🤮 ローカルでビルドが出来なくなる

- Expo SDK 48 → 52までの間でEAS Buildの対応が必須になった
- 元々あったExpo Goベースの開発はあくまでデバッグ用という立ち位置になり、ネイティブ周りの機能を修正したアプリはビルドできなくなった
- Expo SDK 48の時点で本番ビルドはEAS Buildに移行済みだったが、EAS Buildで実行してもエラーになり、ローカルでもビルドできない
- そもそもデバッグできない状態になり、詰みの状態が発生

😭 Expoの一部ライブラリが非推奨になる

- Expo SDK 48 → 52までの間でexpo-auth-sessionの`expo-auth-session/providers/google`が非推奨になった
- 具体的には、以前まではExpo更新のライブラリのみGoogleログインが可能だったが、出来なくなり代替で@react-native-google-signin/google-signinの使用が必須になった
- その他の一部ライブラリでも廃止、仕様変更が発生がしていた

😭 ナビゲーションライブラリのトレンドが変わる

- 以前はreact-navigationが最もよく使われているライブラリだった
- 現在はexpo-routerが公式が提供しているライブラリとして推奨されている
- しかも、expo-routerは`ファイルベースルーティング`を採用しているので、ディレクトリ構造の変更も必要
- ファイルベースルーティングとは？
  - ディレクトリ構造が、そのまま画面のルーティングに反映される方式
  - Next.jsやNuxt.js等の採用されている方式

😕 Reactのバージョンが変わり、非推奨が発生

- Expo SDK 48 → 52の移行でReact、React Nativeのバージョンが上がった
- `defaultProps`の非推奨化など警告がでるようになり、一部コード修正が発生

🤔 ディレクトリ構造のトレンドが変わる

- 以前はフロントエンドのディレクトリ構造はAtomic Designが流行っていた
- 現状のトレンドは以下
  - Feature Sliced Design
  - Screaming Architecture（日本だとFeature型と呼ばれている事が多い）
- トレンドが変わった理由
  - Atomic Designの定義が複雑でチーム開発でうまく回せないケースが多い
  - 後からの開発でコンポーネントの機能拡張した際に、ディレクトリ構造が変わる可能性があり、長期の開発では管理が難しくなる
  - 各画面との依存関係がわかりにくい等々

🤔 Lint、Formatterのトレンドが変わる

- 以前はESLintとPrettierを使っていた
- 現在はBiomeがトレンド
- トレンドが変わった理由
  - 長期の開発をしていくとESLint & Prettierの設定の複雑になっていくケースが発生しやすい
  - ESLint & Prettierを使用するとPCに負荷がかかり開発に支障が出るケースがある
  - よりシンプル且つ、一括管理できるBiomeの方が採用されるようになった

🤔 使っていたライブラリが開発停止になる

- 開発時はグローバルな状態管理はRecoilが最もよく使われていた
- Facebookが開発しているライブラリで、ほぼ公式扱いだったが...
- その後の大規模レイオフで全開発者が解雇され開発停止になった
- React 19では動作しないため、今後の移行が必須
- 今後はJotaiが使われていきそう

このままでは開発できないので、諸々最新化してみる

最新化後の構成

- Expo SDK: 48 → 52
- ナビゲーションライブラリ: react-navigation → expo-router
- ディレクトリ構造: Atomic Design → Feature型
- Lint、Formatter: ESLint、Prettier → Biome
- パッケージマネージャ: yarn → pnpm
- ※まだReact19まで猶予があるので、**Recoil**の移行は一旦後回し

■ リポジトリ

- 更新前: リンク
- 更新後: リンク

起動まで①

Expoには、バージョンアップ時のマイグレーションを自動で行うコマンドが用意されている

以下のコマンドでSDKの最新化 & ライブラリのバージョンの依存関係を更新

```
$ npx expo install expo@latest
$ npx expo install --fix
```

以下のコマンドでドキュメントに記載されているチェックを行い、NGが出ている箇所を修正

```
$ npx expo-doctor
```

- 参考
  - Expo Doctor

起動まで①

Expoには、バージョンアップ時のマイグレーションを自動で行うコマンドが用意されている

以下のコマンドでSDKの最新化 & ライブラリのバージョンの依存関係を更新

```
$ npx expo install expo@latest
$ npx expo install --fix
```

以下のコマンドでドキュメントに記載されているチェックを行い、NGが出ている箇所を修正

```
$ npx expo-doctor
```

- 参考
  - Expo Doctor

ただ今回のケースでは、これをやってもビルドが通らなかった 😢
なので、新規でExpoのプロジェクトを作成して徐々に既存のコードを移行する手法で対応

起動まで②

以下のページを参考に、Expo Routerのサンプルアプリを作成

- Install Expo Router - Expo Documentation

ここから徐々に既存のコードを移行していった

ディレクトリ構造①

■ 以前のディレクトリ構造

```
.
├── assets
└── src
    ├── components
    │   ├── atoms
    │   ├── molecules
    │   ├── organisms
    │   ├── pages
    │   └── templates
    ├── containers
    ├── hooks
    ├── img
    ├── lib
    ├── queries
    └── store
```

ディレクトリ構造①

■ 以前のディレクトリ構造

```
.
├── assets
└── src
    ├── components
    │   ├── atoms
    │   ├── molecules
    │   ├── organisms
    │   ├── pages
    │   └── templates
    ├── containers
    ├── hooks
    ├── img
    ├── lib
    ├── queries
    └── store
```

ディレクトリ構造①

■ 以前のディレクトリ構造

```
.
├── assets
└── src
    ├── components
    │   ├── atoms
    │   ├── molecules
    │   ├── organisms
    │   ├── pages
    │   └── templates
    ├── containers
    ├── hooks
    ├── img
    ├── lib
    ├── queries
    └── store
```

ディレクトリ構造②

■ 現在のディレクトリ構造

```
.
├── app
│   └── (app)
│       └── search
├── assets
│   ├── fonts
│   └── img
├── components
│   ├── elements
│   └── layouts
├── containers
├── features
│   ├── home
│   │   └── components
│   └── search
│       ├── components
│       └── hooks
├── hooks
├── lib
├── queries
└── store
```

ディレクトリ構造②

■ 現在のディレクトリ構造

```
.
├── app
│   └── (app)
│       └── search
├── assets
│   ├── fonts
│   └── img
├── components
│   ├── elements
│   └── layouts
├── containers
├── features
│   ├── home
│   │   └── components
│   └── search
│       ├── components
│       └── hooks
├── hooks
├── lib
├── queries
└── store
```

ディレクトリ構造②

■ 現在のディレクトリ構造

```
.
├── app
│   └── (app)
│       └── search
├── assets
│   ├── fonts
│   └── img
├── components
│   ├── elements
│   └── layouts
├── containers
├── features
│   ├── home
│   │   └── components
│   └── search
│       ├── components
│       └── hooks
├── hooks
├── lib
├── queries
└── store
```

ディレクトリ構造②

■ 現在のディレクトリ構造

```
.
├── app
│   └── (app)
│       └── search
├── assets
│   ├── fonts
│   └── img
├── components
│   ├── elements
│   └── layouts
├── containers
├── features
│   ├── home
│   │   └── components
│   └── search
│       ├── components
│       └── hooks
├── hooks
├── lib
├── queries
└── store
```

Lint、Formatterの移行

- 以下のページを参考にBiomeに移行
  - Getting Started | Biome
- 今回のコードでは軽微な修正のみだったので基本は移行後に、以下のコマンドで自動で修正できた
  - ```
    $ npx @biomejs/biome check --write ./
    ```
    ```

デモ

現在の動作するところまで確認

まとめ

- Expoのアプリの開発を1年半放置すると、結構な地獄になる
- 正直、ローカルビルドができなくなるのは想定していなかったので、かなり困った
- フレームワークとしての強みはかなりあるが、反面フレームワークで強制される部分が大きいので、開発に間を空けすぎると復帰が難しくなる
- バックエンドと比較してフロントエンドはトレンドの移り変わりが激しいのを再認識した

ご清聴ありがとうございました🎉