

Flutterアプリを生成AIで生成する勘所

自己紹介

- りず
和泉田 領一
- 合同会社CAPH TECH 代表社員
- Flutter技術顧問・テックリード
- ゆめみの方から来ました（Flutterテックリード）
- 生成AIを活用した開発の研究

前提条件

使用するAIモデル

- **CursorでClaude 3.5 Sonnetが基本**
- **o1-preview, o1-miniも活用できる**
- Claude 3.5 Sonnetの話

Flutter生成特有のルール

1. 命名規則

- **enumの命名**
 - 大文字とアンダースコアで命名されることが多いため、ルールで命名を規定する。

2. 状態管理とアーキテクチャ

- **Riverpodの利用**

- 何も指定しなければ古い記述形式になるため、Generatorを使用する場合はルールに記述する。

- **アーキテクチャパターンの指定**

- クリーンアーキテクチャとMVVMを利用してくる。
- Riverpodでのアーキテクチャパターンの指示は必須。

3. ウィジェットの管理

- デフォルト生成とカスタム分割

- Widgetはデフォルトで分割せずに生成される。
- カスタムWidgetに分割するには明示的な指示が必要。

- カスタムWidgetの分割ルール

- 「Widgetは単一責任の原則を遵守する」。

- ローカル状態の管理

- Flutter Hooksを使わずにsetStateを利用するコードでも生成なら面倒は少ない。
- わざわざFlutter Hooksを覚える必要が無い。

4. パッケージの利用

- **パッケージ導入の検討**

- 面倒を回避するためにパッケージを導入する場合、生成AIで肩代わりできるか検討。

- **簡易な状態管理パッケージの検討**

- Riverpodが複雑なため、より簡単なパッケージを検討。
- 私はsignalsを利用している。

4. パッケージの利用（続き）

- ルーティング

- Navigator 2.0を利用して、状態の変化により遷移を引き起こす実装を生成できる。
- がんばってgo_routerを利用するよりNavigator 2.0のコードを生成する方が楽に感じている。

生成一般

1. 導入時の活用方法

- **Cursor**を既存プロジェクトで利用を始める場合
 - Copilot+を利用する
 - ドキュメンテーションコメントを生成する
 - テストを生成する
 - リファクタリング
 - Widgetを分割する
 - 設計の問題点を相談する

2. コーディングルール

- 基本的なコーディングルールの遵守
 - 基本ルールはAIが守るため、特別な指示は不要。
 - 命名規則はenumくらい。

3. コメントとアサーションの活用

- **ドキュメンテーションコメントやコードコメントの記述**
 - 生成されたコードを理解する時間が必要。
 - コメントを記述するように指示することで、コードの理解が容易になる。
 - 記述内容を指示することで、ドキュメンテーションコメントを充実させる。
- **アサーションの活用**
 - アサーションを活用することで、コードの信頼性を高める。
 - 事前条件・事後条件・不変条件

4. 設計とコストの注意点

- 設計とAIモデルの選択

- o1-preview, o1-miniで設計、Claude 3.5 Sonnetでコード生成。
- o1-previewは使用量に応じて料金がかかり高価。
- o1-previewが生成するFlutterコードは情報が古い。

5. ドキュメント管理

- **アーキテクチャパターンの記述方法**
 - ルールではなく、ドキュメントファイルに記述。
 - ComposerのProjectでそのファイルを必ず読み込むよう指定。

まとめ

- AIを活用したFlutterアプリ開発のためのルールとベストプラクティス
- 効率的な開発と高品質なコードの生成を目指す

Q&A

ご質問はありますか？