

Flutterでフォルダブル対応をしよう！

自己紹介

DEBUG



大阪成蹊大学 データサイエンス学部 B2
ぽちぽちのつどい/Ale Engineer

ここ1年半くらいは主にFlutterで開発
スマホを愛するあまり、開発の沼にハマったオタク

DroidKaigi2025に参加した時に大画面対応のセッションを見てFlutterでもやってみようとなった
[基礎から学ぶ大画面対応
～「Large screen differentiated」認定アプリの開発知見～]

1. 大画面, フォルダブルについて

DEBUG

Android XRの登場

フォルダブルスマホの普及

開発者が意図していないタイミングでのレイアウト崩れもある

Android 16以降、大画面端末でアスペクト比制限が無視される

- minAspectRatio / maxAspectRatioが無効化
- 画面の向き固定(screenOrientation)も無視される
- 強制的にリサイズ可能になり、レターボックス(黒帯)が表示されなくなる

この辺りはDroidKaigi2025で詳しく発表されてました

2. Flutterで対応してみる

DEBUG

今回作成したのは三画面のアプリ

3画面のうち二画面を大画面対応していく

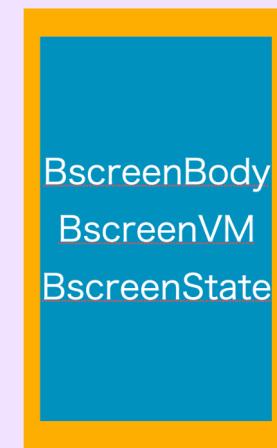
bodyプロパティの中全体をコンポーネント化

ViewModelやStateは普段通り実装する

AScreen



BScreen



```
  saved
    components
      saved_screen_body.dart
      saved_screen_tile.dart
    view_model
      saved_adaptive_screen.dart
      saved_screen.dart
    body: SavedScreenBody(),
); // Scaffold
```

2. Flutterで対応してみる

フォルダブルスマホで表示したいFoldableScreenを実装する

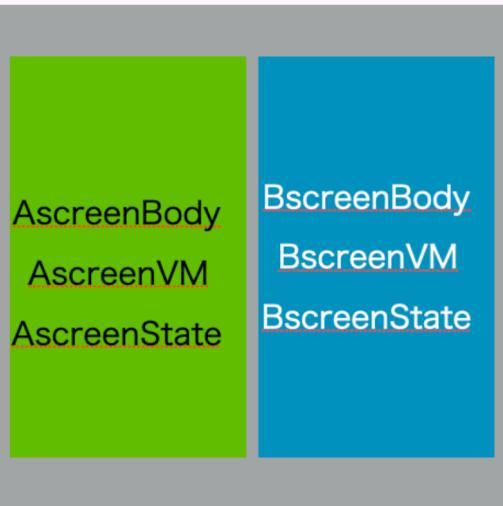
FoldableScreenで先ほど定義したAscreenBodyとBscreenBodyを呼び出す

既存の画面やVMを再利用することができるので工数を削減できる

画面サイズ判定のProviderをwatchするConsumerWidgetを定義してこのwidgetに遷移するようにrouterを定義

この構成だとMVVM + Repositoryパターンを逸脱せず実装できる

→ コンポーネントとVMが1対1で新しい画面でラップしなおしただけと解釈できる



```
class SavedAdaptiveScreen extends HookConsumerWidget {  
  const SavedAdaptiveScreen({super.key});  
  
  @override  
  Widget build(BuildContext context, WidgetRef ref) {  
    final width = MediaQuery.of(context).size.width;  
    useEffect(() {  
      WidgetsBinding.instance.addPostFrameCallback((_) {  
        ref.read(deviceWidthProvider.notifier).updateScreen(width);  
      });  
      return null;  
    }, [width]);  
  
    final isWide = ref.watch(deviceWidthProvider);  
    if (isWide) {  
      return const FoldScreen();  
    } else {  
      return const SavedScreen();  
    }  
  }  
}
```

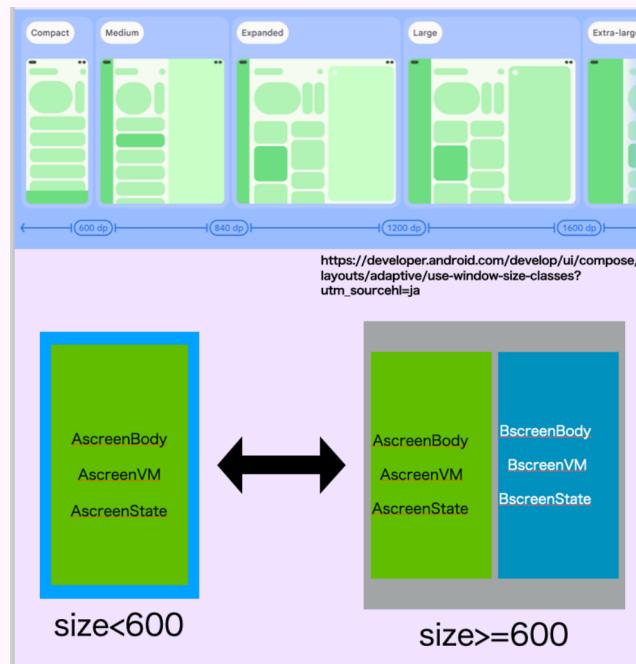
2. Flutterで対応してみる

画面を切り替えるために画面サイズを判定するためのProviderを定義する

切り替えの参考画面サイズはGoogleが提唱している600px以上を大画面とする

→Android developerページにある"ウィンドウ サイズクラスを使用する"というページを参考にしました

また横向きやさらに大きいタブレットなどで対応する場合横は840px以上とする



```
@riverpod
class DeviceWidth extends _$DeviceWidth {
  @override
  bool build() {
    return false;
  }

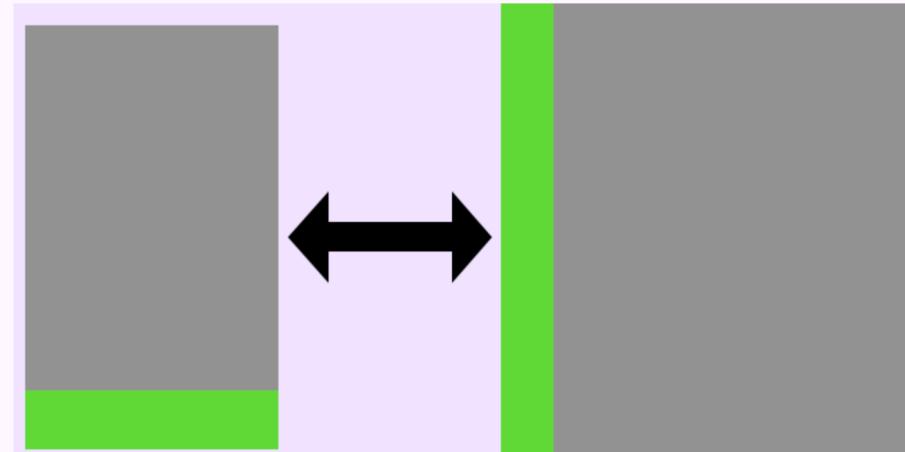
  void updateScreen(double width) {
    state = width >= 600;
  }
}
```

2. Flutterで対応してみる

DEBUG

Providerが定義できればBottomNavigationをNavigationRailにする

Googleは640px以上の大画面アプリではNavigationRailを推奨している



3. アプリにくみこんでみた

DEBUG

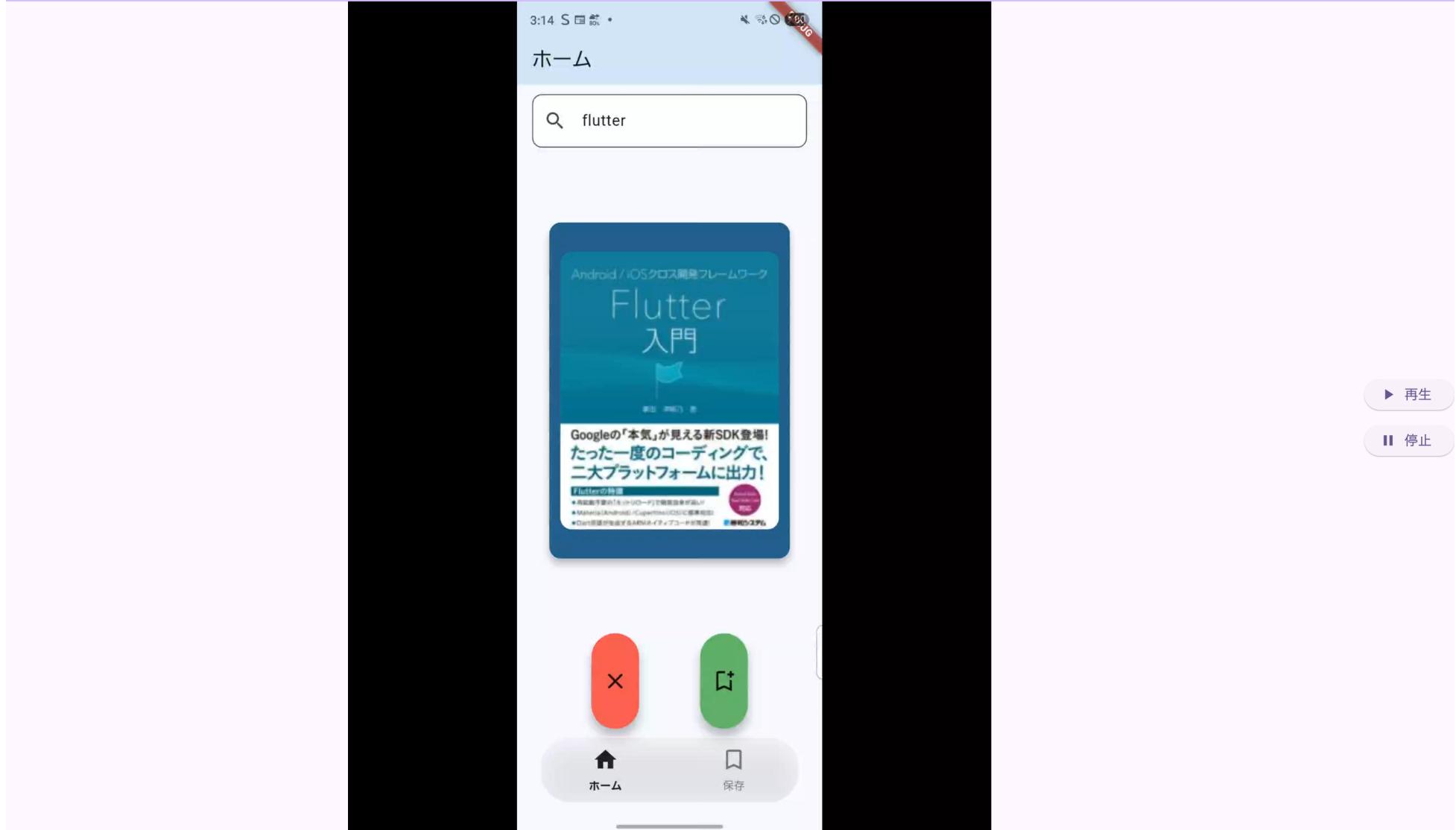
DEMO

3. アプリにくみこんでみた

DEBUG



3. アプリにくみこんでみた



4. 感想,まとめ

- 今あるコンポーネントやVMを再利用できるので工数がそこまでかからず実装できる
- AppBar共通化したかったのでこのような実装になったがScaffoldごとよびだして並べる実装だともっと楽にできると思います
- 全部の画面をやるというよりチャット画面や設定画面など一部の画面を対応させるのが良さそう
- Androidだけじゃなくて2026年後半にはAppleもフォルダブルを出すという噂もある