w_step_nav.dart

소스 코드

전체 코드

```dart
import 'package:buyoil/common/app_strings.dart';
import 'package:buyoil/common/app_styles.dart';
import 'package:dotted_line/dotted_line.dart';
import 'package:easy_localization/easy_localization.dart';
import 'package:flutter/material.dart';

import '../../common/app_colors.dart';

class StepNavWidget extends StatefulWidget {
  final int currentStep;
  final int totalSteps;

  const StepNavWidget({
    Key? key,
    required this.currentStep,
    required this.totalSteps,
  }) : super(key: key);

  @override
  createState() => StepNavWidgetState();
}

class StepNavWidgetState extends State<StepNavWidget> {
  @override
  Widget build(BuildContext context) {
    return Container(
      padding: EdgeInsets.only(left: 27, top: 33, bottom: 33, right: 39),
      color: AppColors.FF007C5E,
      width: 202,
      child: Stack(
        children: [
          Positioned(
            left: 15,
            top: 0, bottom: 0,
            child: SizedBox(
              width: 5,
              height: double.infinity,
              child: DottedLine(
                direction: Axis.vertical,
                alignment: WrapAlignment.center,
                lineLength: double.infinity,
                lineThickness: 2.0,
                dashLength: 8.0,
                dashColor: AppColors.PRIMARY,
                // dashGradient: [Colors.red, Colors.blue],
```

```
                dashRadius: 2.0,
                dashGapLength: 4.0,
                dashGapColor: Colors.transparent,
                // dashGapGradient: [Colors.red, Colors.blue],
                dashGapRadius: 0.0,
              ),
            ),
          ),
          Positioned(
            child: _body(context),
          )
        ],
      )
    );
  }

  Widget _buildCircularImageStep(int index) {
    return Row(
      mainAxisAlignment: MainAxisAlignment.spaceBetween,
      children: [
        CircleAvatar(
          radius: 15,
          backgroundColor: AppColors.PRIMARY,
          child: CircleAvatar(
            radius: 7.5,
            backgroundColor: widget.currentStep != index ? AppColors.PRIMARY:
Colors.white,
          )
        ),
        Text("${AppStrings.step.tr()} $index", style: AppStyles.tsStepNavText),
      ],
    );
  }

  Widget _body(BuildContext context) {
    return Column(
      mainAxisAlignment: MainAxisAlignment.spaceBetween,
      children: <Widget>[
        _buildCircularImageStep(1),
        _buildCircularImageStep(2),
        _buildCircularImageStep(3),
        _buildCircularImageStep(4),
      ],
    );
  }
}
```

1 파일 개요

파일명: `lib/view/widget/step1/w_step_nav.dart`

클래스명: `StepNavWidget`

역할:

- **단계별 진행 상황 표시(Step Navigation) 위젯**
- 현재 단계와 전체 단계를 시각적으로 표시
- 왼쪽에는 점선(vertical dotted line), 오른쪽에는 단계 표시 및 텍스트

사용 맥락 예시:

- 설치, 결제, 등록, 설문 등 **멀티 스텝 프로세스** 화면
- 현재 진행 중 단계 강조, 완료된 단계/남은 단계 시각화

2️⃣ 주요 기능

1. **점선 표시**
   - `DottedLine` 라이브러리 사용 → 수직 점선
   - 색상: `AppColors.PRIMARY`
   - 점선 두께, 길이, 간격, 반지름 지정
2. **단계 원형 표시**
   - `_buildCircularImageStep(int index)`
   - `CircleAvatar` 두 겹 사용:
     - 바깥 원: `radius 15`, 색상 PRIMARY
     - 안쪽 원: radius 7.5, **현재 단계일 경우 흰색**, 나머지는 PRIMARY
   - 단계 텍스트: `"Step {index}"`
3. **컬럼 배치**
   - `_body()`에서 `Column(mainAxisAlignment: spaceBetween)`
   - 단계 1~4를 순서대로 표시
   - 점선은 `Stack` 안에서 `Positioned`로 왼쪽 배치
4. **스타일**
   - Container 패딩: left 27, top/bottom 33, right 39
   - 배경색: `AppColors.FF007C5E`
   - 고정 너비: 202

3️⃣ 구조 분석

```
StepNavWidget (StatefulWidget)
 └ StepNavWidgetState (State)
       ├ build()
       |    └ Container
       |         ├ Stack
       |         |    ├ Positioned(left:15) → DottedLine(vertical)
       |         |    └ Positioned → _body()
       ├ _body()
       |    └ Column(mainAxisAlignment: spaceBetween)
```

```
        |         ├─ _buildCircularImageStep(1)
        |         ├─ _buildCircularImageStep(2)
        |         ├─ _buildCircularImageStep(3)
        |         └─ _buildCircularImageStep(4)
        └─ _buildCircularImageStep(index)
              ├─ Row(spaceBetween)
              |    ├─ CircleAvatar(바깥+안쪽)
              |    └─ Text("Step $index")
```

- 점선과 단계 표시를 **Stack**으로 겹쳐서 표현
- 단계는 **Column**으로 배치, `spaceBetween`으로 일정 간격 유지

---

4 동작 흐름

1. `StepNavWidget(currentStep: 2, totalSteps: 4)` 호출
2. Container 내부 Stack 구성:
    - 왼쪽에 점선 표시
    - 오른쪽에 `_body()` → Column에 4단계 표시
3. `_buildCircularImageStep()` 호출 시:
    - 현재 단계와 비교 → 안쪽 원 색상 결정
    - `"Step {index}"` 텍스트 표시
4. 화면에 수직 점선과 단계 원형 표시

---

5 장점

- **시각적 단계 표시**: 현재 단계 강조, 완료/미완료 구분 가능
- **점선 라이브러리 활용**: 수직 점선 간단 구현
- **Stack + Positioned 구조**: 점선과 단계 원형 겹치기 용이
- **국제화 지원**: `easy_localization` 사용 → Step 텍스트 번역 가능

---

6 단점 / 개선점

1. **단계 수 하드코딩**
    - `_body()`에서 4단계 직접 호출
    - 개선: `widget.totalSteps` 기반으로 반복 생성
2. **고정 너비**
    - Container width 202 → 반응형 화면에서는 부적합
    - 개선: 부모 constraints 활용 또는 `double.infinity`
3. **StatefulWidget 불필요**
    - 현재 상태 관리 없음 → StatelessWidget으로 변경 가능

4. **점선 위치 고정**
    - `Positioned(left:15)` → 화면 크기에 따라 위치 조정 어려움
    - 개선: LayoutBuilder나 padding 사용
5. **재사용성**
    - 단계 텍스트, 색상, 원 크기 하드코딩 → 매개변수화 가능

7️⃣ 개선 예시 (반응형 + 단계 수 동적 생성)

```
class StepNavWidget extends StatelessWidget {
  final int currentStep;
  final int totalSteps;

  const StepNavWidget({Key? key, required this.currentStep, required this.totalSteps})
: super(key: key);

  @override
  Widget build(BuildContext context) {
    return Container(
      padding: const EdgeInsets.symmetric(vertical: 33, horizontal: 27),
      color: AppColors.FF007C5E,
      child: Stack(
        children: [
          Positioned(
            left: 15,
            top: 0, bottom: 0,
            child: SizedBox(
              width: 5,
              child: DottedLine(
                direction: Axis.vertical,
                lineLength: double.infinity,
                lineThickness: 2.0,
                dashLength: 8.0,
                dashColor: AppColors.PRIMARY,
                dashGapLength: 4.0,
              ),
            ),
          ),
          Positioned(
            left: 40,
            right: 0,
            top: 0,
            bottom: 0,
            child: Column(
              mainAxisAlignment: MainAxisAlignment.spaceBetween,
              children: List.generate(totalSteps, (index) =>
_buildCircularImageStep(index + 1)),
            ),
          ),
        ],
      ),
```

```
    );
  }

  Widget _buildCircularImageStep(int index) {
    return Row(
      children: [
        CircleAvatar(
          radius: 15,
          backgroundColor: AppColors.PRIMARY,
          child: CircleAvatar(
            radius: 7.5,
            backgroundColor: currentStep == index ? Colors.white : AppColors.PRIMARY,
          ),
        ),
        const SizedBox(width: 8),
        Text("${AppStrings.step.tr()} $index", style: AppStyles.tsStepNavText),
      ],
    );
  }
}
```

- `totalSteps` 기반으로 단계 동적 생성
- StatelessWidget 사용 → 불필요한 상태 제거
- 점선과 단계 간격 반응형 처리 가능