s_step1.dart

소스 코드

전체 코드

```dart
import 'package:buyoil/model/ui_state_step1.dart';
import 'package:buyoil/view/widget/w_left_triangle.dart';
import 'package:buyoil/view/widget/w_step_nav.dart';
import 'package:buyoil/viewmodel/vm_step1.dart';
import 'package:easy_localization/easy_localization.dart';
import 'package:flutter/material.dart';
import 'package:flutter_riverpod/flutter_riverpod.dart';
import 'package:go_router/go_router.dart';

import '../../common/app_colors.dart';
import '../../common/app_strings.dart';
import '../../common/app_styles.dart';
import '../../config.dart';
import '../../router.dart';
import '../../viewmodel/vm_serial_port.dart';
import '../widget/custom_text_input.dart';
import '../widget/step1/buttons_number.dart';
import '../widget/w_header.dart';

class Step1Screen extends ConsumerStatefulWidget {
  const Step1Screen({Key? key}) : super(key: key);

  @override
  ConsumerState<ConsumerStatefulWidget> createState() => Step1ScreenState();
}

class Step1ScreenState extends ConsumerState<Step1Screen> {

  @override
  void initState() {
    super.initState();
  }

  @override
  Widget build(BuildContext context) {
    final notifier = ref.watch(step1Provider.notifier);
    final state = ref.watch(step1Provider);
    return Scaffold(
      body: Column(
        children: [
          HeaderWidget(),
          Expanded(
            child: Row(
              mainAxisAlignment: MainAxisAlignment.spaceBetween,
              children: [
```

```
                StepNavWidget(currentStep: 1, totalSteps: 4),
                Expanded(
                  child: Stack(
                    children: [
                      _body(context),
                      _toast(),
                    ],
                  ),
                )

              ],
            ),
          )
        ],
      )
    );
  }

  Widget _body(BuildContext context) {
    final notifier = ref.watch(step1Provider.notifier);
    final state = ref.watch(step1Provider);

    return Positioned.fill(
      child: Row(
        children: [
          Expanded(
            flex: 512,
            child: Container(
              child: Center(
                  child: Column(
                    mainAxisAlignment: MainAxisAlignment.center,
                    children: [
                      Container(
                        height: 45,
                        alignment: Alignment.topCenter,
                        child: Text(AppStrings.enterPhoneNumber.tr(),
                            style: AppStyles.enterPhoneNumberTextStyle),
                      ),
                      SizedBox(height: 14,),
                      CustomTextInputField(text: state.phoneNumber),
                      SizedBox(height: 14,),
                      SizedBox(height: 45,),
                    ],
                  )
              ),
            ),
          ),
          Container(
            child: RoundedLeftTriangleWidget(width: 36, height: 68, color:
AppColors.D6E7DF),
          ),
          Expanded(
            flex: 625,
```

```
            child: Container(
                color: AppColors.D6E7DF,
                child: Center(
                  child: NumberButtonGroup(
                    onButtonPressed: (String buttonText) {
                      ref.watch(step1Provider.notifier).pressedNumber(buttonText);
                    },
                  ),
                )
            ),
          ),
        ],
      ),
    );
  }

  Widget _toast() {
    return ref.watch(step1Provider).when(
      input: (_, __, showToast) {
        return showToast ? Positioned(
            top: 57,
            left: 0,
            right: 0,
            child: Center(
              child: Container(
                // alignment: Alignment.center,
                padding: EdgeInsets.symmetric(vertical: 23, horizontal: 70),
                decoration: BoxDecoration(
                    borderRadius: BorderRadius.circular(20),
                    border: BoxBorder.all(color: AppColors.D32F2F, width: 1),
                    color: Color(0xfffde3e3)),
                child: Text(AppStrings.checkValidationToast.tr(),
                    textAlign: TextAlign.center,
                    style: AppStyles.tsStep1Toast),
              ),
            )
        ) : Positioned(
          bottom: 0, right: 0,
          child: SizedBox.shrink(),
        );
      },
      completed: (_, _, showToast) {
        return showToast ? Positioned(
            top: 57,
            left: 0,
            right: 0,
            child: Center(
              child: Container(
                // alignment: Alignment.center,
                padding: EdgeInsets.symmetric(vertical: 23, horizontal: 70),
                decoration: BoxDecoration(
                    borderRadius: BorderRadius.circular(20),
                    border: BoxBorder.all(color: AppColors.D32F2F, width: 1),
```

```
                    color: Color(0xfffde3e3)),
                child: Text(AppStrings.checkValidationToast.tr(),
                    textAlign: TextAlign.center,
                    style: AppStyles.tsStep1Toast),
              ),
            )
        ) : Positioned(
          bottom: 0, right: 0,
          child: SizedBox.shrink(),
        );
      },
    );
  }
}
```

1. Step1Screen 역할 개요

`Step1Screen`은 **4단계 중 Step 1: 전화번호 입력** 화면이며 다음 기능을 담당합니다.

- 전화번호 입력 UI 표시
- 화면 좌측은 입력된 번호 표시 + 헤더
- 화면 우측은 Keypad(NumberButtonGroup) 제공
- 입력 검증 오류 시 Toast 노출
- UIStateStep1에 따라 상태별 UI 출력
- StepNavWidget을 통해 현재 Step 표시

---

2. 전체 구조 흐름

```
Step1Screen (ConsumerStatefulWidget)
  └── Step1State (UI 상태 감시 + 이벤트 처리)
       ├── state = ref.watch(step1Provider)
       ├── notifier = ref.watch(step1Provider.notifier)
       ├── HeaderWidget()
       ├── StepNavWidget(1/4)
       ├── _body()   ← 전화번호 + 숫자 키패드 UI
       └── _toast()  ← Riverpod의 UIState 활용한 toast 렌더링
```

---

3. build() 분석

3.1 상태 로드 및 Watch

```
final notifier = ref.watch(step1Provider.notifier);
final state = ref.watch(step1Provider);
```

- **state**는 현재 Step1의 상태(UIStateStep1)
- **notifier**는 유저 입력·검증 이벤트 처리

3.2 레이아웃

- 상단: `HeaderWidget()`
- 좌측: `StepNavWidget(currentStep: 1, totalSteps: 4)`
- 중앙: `_body()`
- 최상단 Stack 상단 레이어로 `_toast()` 배치

---

4. _body() 상세 분석

주요 UI 구성:

```
Row
 ├── Expanded(flex: 512)
 |     └── 중앙 정렬
 |          ├── “전화번호 입력” 텍스트
 |          ├── CustomTextInputField
 |          └── spacing
 ├── RoundedLeftTriangleWidget (세그먼트 구분)
 └── Expanded(flex: 625)
       └── 키패드(NumberButtonGroup)
```

전화번호 입력 UI 특징

- `CustomTextInputField(text: state.phoneNumber)`
  → 입력된 번호를 상태에서 그대로 표시

키패드 이벤트

```
onButtonPressed: (String buttonText) {
  ref.watch(step1Provider.notifier).pressedNumber(buttonText);
}
```

- pressedNumber()는 ViewModel에서
  숫자, 삭제, 다음 단계 등 각각 처리할 것

---

5. _toast() 분석

`UIStateStep1`은 두 가지 상태를 가진 것으로 보임:

- `input(phoneNumber, isValid, showToast)`
- `completed(phoneNumber, isValid, showToast)`

snowToast == true

```
Positioned(top: 57)
 └── 빨간 border + 연한 빨강 배경
     └── “검증해주세요” 텍스트
```

showToast == false

```
SizedBox.shrink() (토스트 숨김)
```

즉, ViewModel에서 showToast = true로 변경하면 바로 토스트가 화면 상단에 렌더링됨.

---

6. UIState 활용 구조

UIStateStep1 구조는 Riverpod + Freezed 패턴을 쓰고 있음.

예상 모델:

```
@freezed
class UIStateStep1 with _$UIStateStep1 {
  const factory UIStateStep1.input(String phone, bool isValid, bool showToast) = _Input;
  const factory UIStateStep1.completed(String phone, bool isValid, bool showToast) =
_Completed;
}
```

즉 UI는 상태에 따른 조건 분기만 수행하고, 모든 비즈니스 로직(번호 조합, 길이 제한, validator, toast trigger)은 step1Provider(notifier)에서 처리하는 형태.

---

7. 화면 동작 흐름 (정리)

1. 화면 로드 → state 초기값: input 상태
2. 유저가 키패드를 눌러 `pressedNumber()` 실행
3. ViewModel이 입력 상태 업데이트 → 화면 자동 리빌드
4. 번호가 부족하면 `showToast = true` → 상단에 빨간 Toast 표시
5. 번호가 완성되면 notifier가 UIState를 `completed()`로 변경할 수 있음
6. completed 상태에서도 showToast가 true이면 같은 토스트 컴포넌트 표시

---

8. 코드 관점에서의 개선 제안 (선택)

(요청하시면 개선 버전 코드도 실제로 작성해드립니다.)

1) 중복된 Toast UI 제거 가능

현재 input/completed 두 케이스가 동일한 UI 블록을 중복 렌더링함.
→ Toast 위젯을 하나만 만들고 상태만 받아서 처리하는 방식으로 단일화 가능.

2) Positioned 사용 대신 Overlay 통합 가능

UX 측면에서 Toast 위젯은 AnimatedOpacity나 OverlayEntry가 더 자연스럽습니다.

3) provider 반복 watch 최적화

현재 `_body()` 내부에서 두 번 watch를 호출하고 있습니다.

```
final notifier = ref.watch(step1Provider.notifier);
final state = ref.watch(step1Provider);
```

→ build()에서 받은 값을 그대로 전달하거나 ref.read로 최적화 가능.

4) Row 레이아웃 비율을 Theme 또는 LayoutBuilder로 정리 가능

데스크탑 UI라 가변 레이아웃을 고려하면 더 나은 구조 가능.

---

9. 원하시면 제공 가능한 추가 분석

요청하시면 다음 문서도 제작해드립니다.

- **Step1 ViewModel 내부 pressedNumber() 로직 흐름 해석**
- **전화번호 입력 검증 상태 다이어그램**
- **UIStateStep1 기반 이벤트 시퀀스 다이어그램**
- **Step1 화면 리팩토링 코드 (중복 제거 + 구조 개선)**
- **Step1 전체를 MVVM 구조도로 시각화**