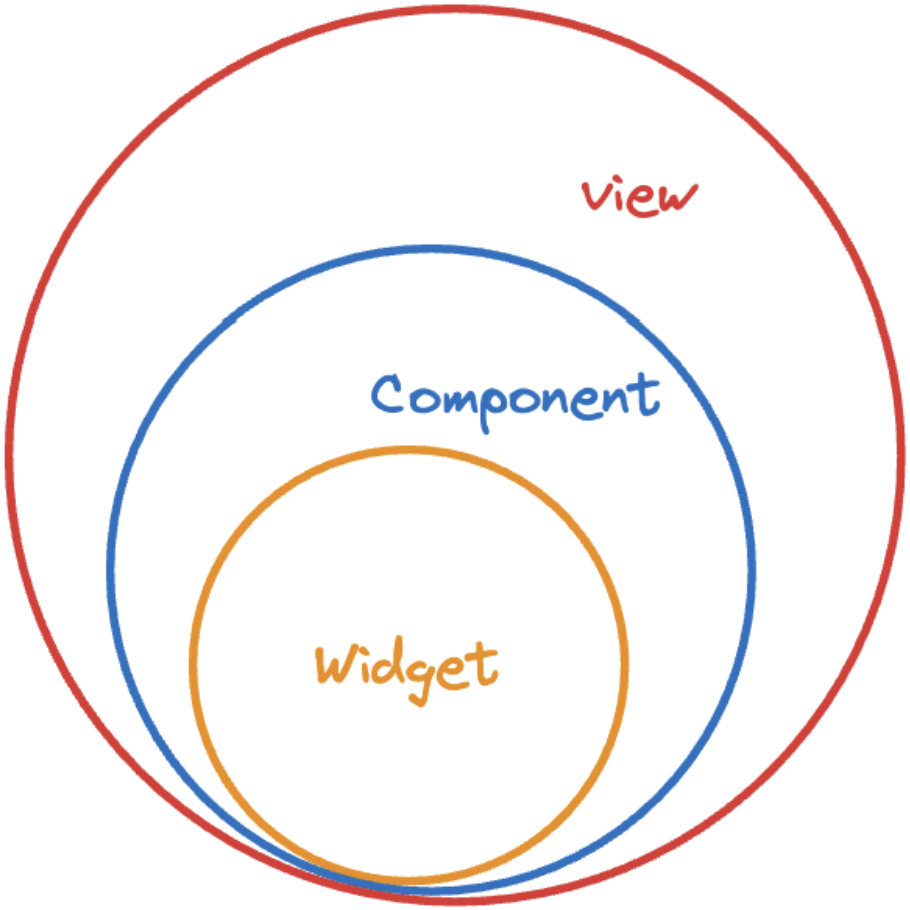
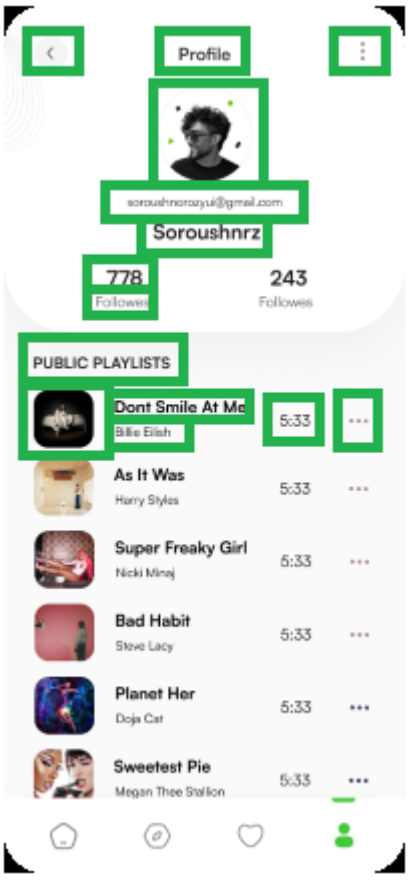
Flutter (위젯,뷰,컴포넌트,스크린)

위젯, 뷰 , 컴포넌트 , 스크린의 차이점은 뭘까 ?

- Flutter 를 처음 접하게 되면 가장 먼저 알게 되는 게 위젯이 아닐까라고 생각합니다. 위젯을 이용해서 Flutter 에서는 화면의 간단한 UI 를 제작합니다. 하지만 이제 프로젝트로 가게 되면 위젯만을 이용해서 작업을 하게 되면 효율도 많이 떨어지고 복잡하게 됩니다. 그래서 알고 가야하는 부분이 **위젯 , 뷰 ,컴포넌트, 스크린** 입니다.



1. Widget (위젯)



Flutter의 기본 단위로, UI를 구성하는 가장 작은 요소입니다.

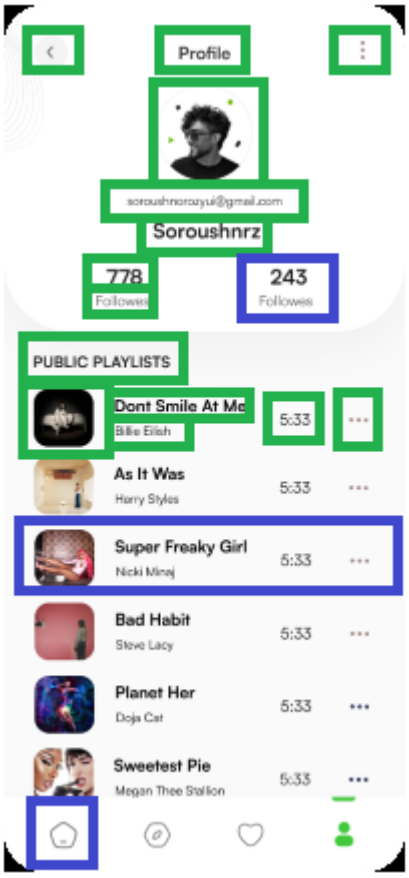
Flutter에서 화면에 표시되는 모든 것은 위젯입니다. (위 화면 예시로 들면 초록색 박스)

위젯을 조합하여 더 큰 UI를 만들며, 재사용 가능한 단위로 커스텀 위젯을 만들 수 있습니다.

- **특징**:
  - Flutter의 기본 빌딩 블록.
  - `StatelessWidget` 또는 `StatefulWidget`으로 상태를 관리.
  - 예: `Text`, `Button`, `Row`, `Column` 등.
- **사용 목적**:
  - UI 요소를 정의하거나 조합.
  - **개발 초기**에 간단한 UI를 만들 때 주로 사용.

---

2. Component (컴포넌트)



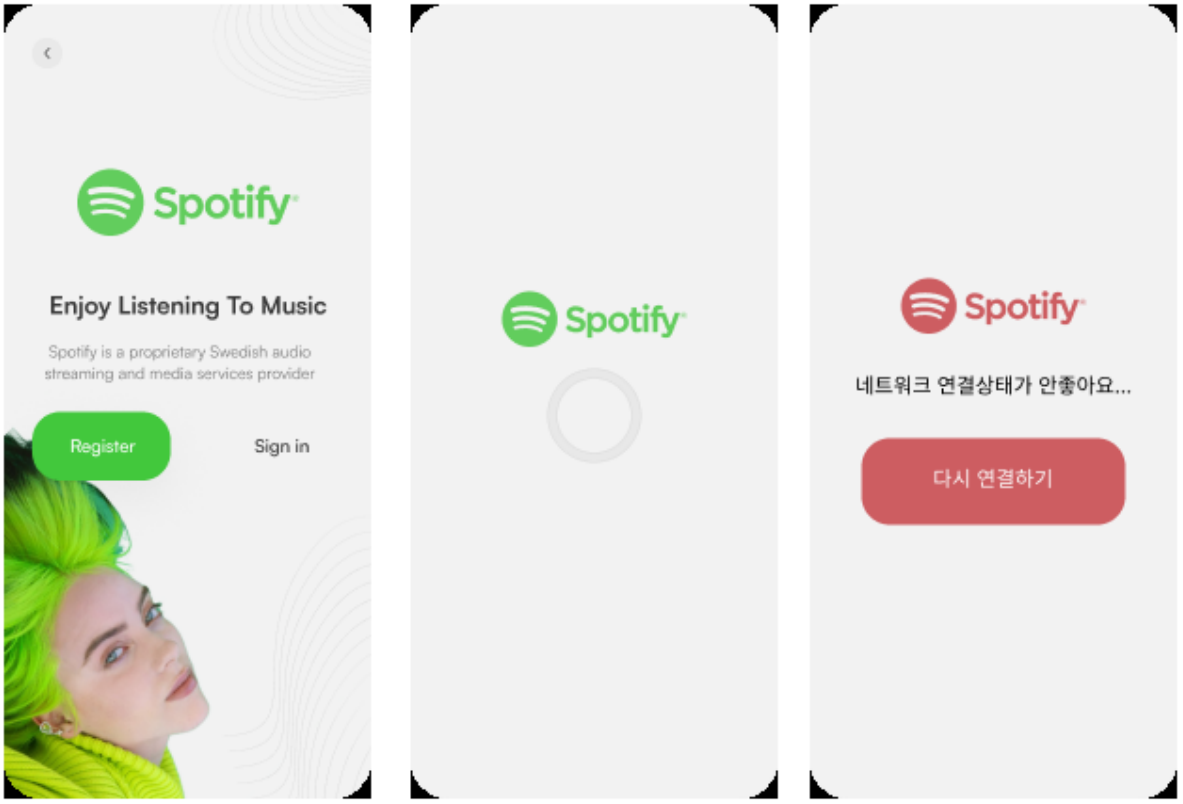
재사용 가능한 UI 조각으로, 여러 위젯을 조합하여 기능적으로 독립적인 단위를 만듭니다. (위 화면에서 파란색 박스)

프로젝트에서 **반복적**으로 사용되는 UI를 컴포넌트화하면 중복 코드를 줄이고 유지보수성을 높일 수 있습니다.

- **특징**:
  - 단일 책임 원칙: 하나의 역할만 담당.
  - 재사용 가능한 커스텀 위젯.
  - 예: 공통 `Card`, `CustomButton`, `ListItem`.
- **사용 목적**:
  - 여러 화면에서 공통으로 쓰이는 UI를 반복 작성하지 않고 재사용.
  - 코드 가독성과 유지보수를 높임.

---

3. View (뷰)



성공 **로딩** **실패**

특정 상태에 따라 렌더링되는 UI 구조를 나타냅니다.

로딩, 데이터, 에러 상태 등에서 다른 UI를 보여줘야 할 때 상태별 UI를 구성하는 데 사용됩니다.

- **특징**:
  - 상태(`loading`, `error`, `data`)에 따라 다르게 렌더링되는 UI 단위.
  - 독립적으로 동작하며, **데이터 상태와 밀접하게 연결**.
- **사용 목적**:
  - 상태별로 화면을 나누어 관리.
  - 네트워크 요청이나 데이터 처리에 따라 동적인 UI를 만들 때.

4. Screen (스크린)

앱의 한 화면을 구성하는 최상위 단위입니다.

Screen은 여러 컴포넌트와 뷰를 조합하여 완전한 화면을 구성하며, 라우팅(Route) 단위로 사용됩니다.

즉, 위 쪽에서 어떤 상태인지를 감지하여 뷰를 골라서 보여주는 곳을 스크린이라고 합니다.

- **특징**:
  - Flutter의 `Scaffold`로 기본 UI를 감싸는 최상위 단위.
  - ViewModel 또는 Controller에서 상태를 구독하고, View를 렌더링.
  - 라우팅을 통해 화면 간 이동을 처리.
- **사용 목적**:
  - 앱의 주요 화면(Home, Login, Detail 등)을 구성.
  - 상태 관리 도구(Riverpod, Provider, Bloc 등)와 연동.

결론

위젯을 효율적으로 관리하기 위해서는 '컴포넌트화' 가 중요할 것 같다고 생각합니다.

컴포넌트화를 이용해 최대한 중복 코드를 줄이고, 유지보수성을 높이며, 코드의 일관성과 가독성을 향상 시킬 수 있습니다. 따라서 개발을 시작하는 단계에서 초반 설계에 위젯을 잘 나누고 컴포넌트화의 구조를 잘 생각해서 프로젝트를 진행한다면 완성도가 더 높은 프로젝트가 될 것 입니다.