Android studio và lập trình Jetpack compose

Các công nghệ mới trong phát triển phần mềm

Lê Minh Đức
1612112

Giới thiệu sơ

Nếu đã từng làm về React Native chắc hẳn mọi người sẽ rất thích framework React UI của nó. Các thành phần (component) nhỏ có thể tái sử dụng, lập trình viên có thể phát triển ứng dụng của mình rất tuyệt vời, mang lại sự linh hoạt và tốc độ nhanh chóng.

Với Android, ta cần giữ lo lắng về việc giữ cấu trúc phân cấp view càng phẳng càng tốt. Vì vậy rất khó tiếp cận dựa trên các component như React Native. Kết quả dẫn đến sự rối rắm khi viết code và khó bảo trì mã nguồn.

Cách cài đặt Android Studio

Cài đặt Android studio trên Windows

Cài đặt Android studio trên Linux Ubuntu

Giới thiệu về Jetpack compose

Jetpack compose là công cụ hiện đại để xây dựng giao diện ứng dụng Android. Jetpack compose giúp đơn giản hóa. tăng tốc độ phát triển ứng dụng Android với ít mã nguồn hơn, các tool mạnh mẽ và các API Kotlin trực quan.

Đặc điểm của Jetpack compose

Compose có cơ chế khá giống với các khung UI framework hiện có như React, Litho hoặc Flutter.

Mục tiêu chính của Jetpack compose:

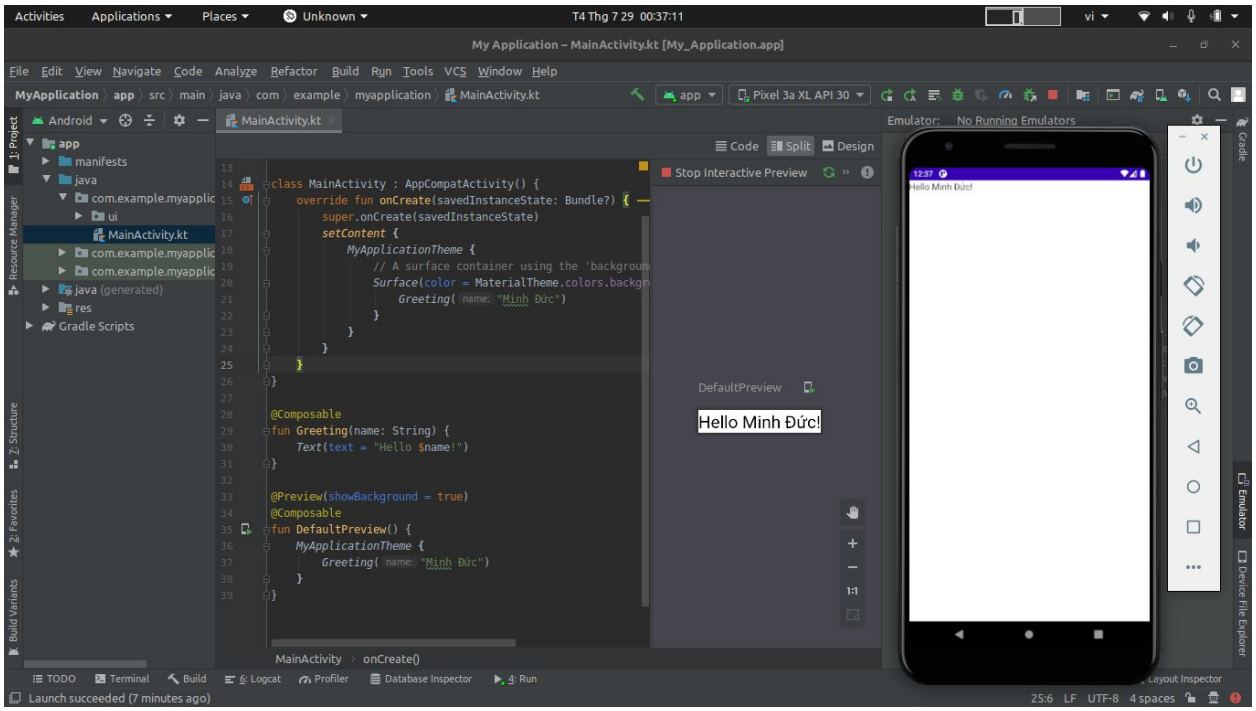
- **Tách biệt khỏi các bản platform release:** Điều này cho phép việc sửa lỗi và triển khai sản phẩm nhanh chóng hơn vì nó độc lập với các bản phát hành Android mới.
- **Ít các thành phần giao diện hơn:** Không bắt buộc sử dụng View hay Fragment khi tạo giao diện người dùng. Tất cả mọi thứ là component và có thể kết hợp tự do với nhau.
- **Làm rõ quyền sở hữu trạng thái (state) và xử lý sự kiện:** Một trong những điều

quan trọng và phức tạp nhất để có trong các ứng dụng lớn là việc xử lý luồng dữ liệu và trạng thái trong giao diện người dùng của bạn. Compose cho ta biết rõ cái gì đang chịu trách nhiệm về trạng thái và cách xử lý các sự kiện, tương tự như cách React xử lý việc này.

- **Viết mã ít hơn**

Điều này giúp dễ dàng tạo ra các thành phần độc lập sau đó kết nối chúng lại với nhau và có thể tái sử dụng cho các màn hình khác hoặc các ứng dụng khác. Việc này giúp bạn có nhiều thời gian hơn để tập trung vào trải nghiệm người dùng.

Ví dụ một ứng dụng compose đơn giản



Trong phương thức **onCreate**, ta đặt nội dung của ứng dụng compose bằng cách gọi setContent. Đây là một phương thức khởi tạo cây widge và bọc nó trong FrameLayout.

Để làm cho mọi thứ hoạt động, ta cần bọc mọi thứ của ứng dụng trong MaterialTheme. MaterialTheme cung cấp màu sắc, styles và fonts cho các widget. Sau đó có thể thêm một composable function, trong composable function đó chỉ đơn giản hiển thị 1 dòng text ra ngoài màn hình.

Composable functions

Jetpack compose được xây dựng dựa trên các hàm tổng hợp (composable function). Những hàm này cho phép xác định UI của ứng dụng bằng cách thiết lập các component, liên kết chúng lại và liên kết với data để hiển thị ra những nội dung cần thiết. Composable function tập trung vào UI và phụ thuộc giữa UI và dữ liệu thay vì tập trung vào xây dựng giao diện người dùng.

Để tạo 1 composable function, ta chỉ cần thêm **@Composable** vào đầu mỗi hàm, ví dụ:

```
27
28      @Composable
29      fun Greeting(name: String) {
30          Text(text = "Hello $name!")
31      }
32
```

Hàm này đơn giản chỉ in ra 1 câu Hello + tham số **name** được truyền vào.

Sau đó ta gọi hàm này ở trong **setContent** của hàm main đồng thời truyền tham số **name** vào:

```
14  class MainActivity : AppCompatActivity() {
15      override fun onCreate(savedInstanceState: Bundle?) {
16          super.onCreate(savedInstanceState)
17          setContent {
18              MyApplicationTheme {
19                  // A surface container using the 'background' color from the theme
20                  Surface(color = MaterialTheme.colors.background) {
21                      Greeting( name: "Minh Đức")
22                  }
23              }
24          }
25      }
26  }
```

Layout

Các thành phần UI trong Jetpack compose được phân cấp (1 component này có thể chứa các component khác), developer có thể xây dựng 1 hệ thống phân cấp bằng cách gọi một hàm tổng hợp (composable function) từ 1 hàm tổng hợp khác.

Bắt đầu với các Text đơn giản

Ở đây có một hàm đơn giản là **MyFunction()** để hiển thị ra thông tin của một sinh viên, sau đó ta gọi hàm này ở **DefaultPreview()** để xem trước kết quả (mà chưa cần chạy trên 1 thiết bị Android):

```
@Composable
fun MyFunction() {
    Text( text: "Lê Minh Đức")
    Text( text: "MSSV: 1612112")
    Text( text: "Số điện thoại: 0378654897")
}

@Preview(showBackground = true)
@Composable
fun DefaultPreview() {
    MyApplicationTheme {
        MyFunction()
    }
}
```

Kết quả sẽ là:

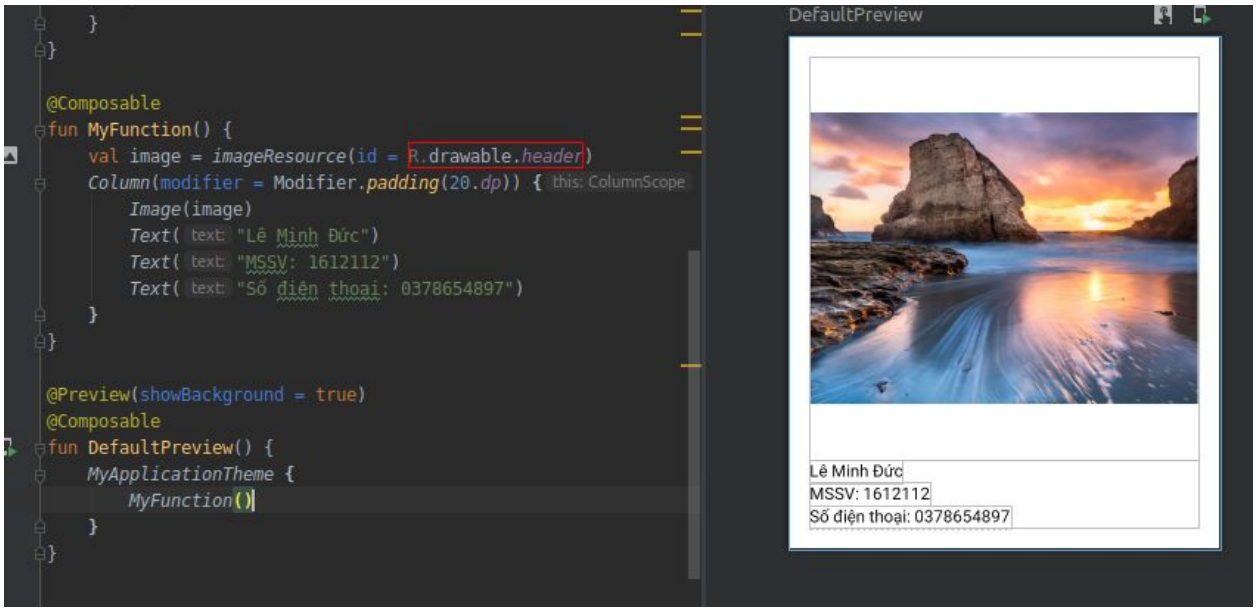
Nội dung các Text bị đè lên nhau, để khắc phục thì sẽ cần bao bọc các Text này trong 1 **Column** và kết quả sẽ như sau:

Ta cũng có thể thêm các style setting cho **Column** này:

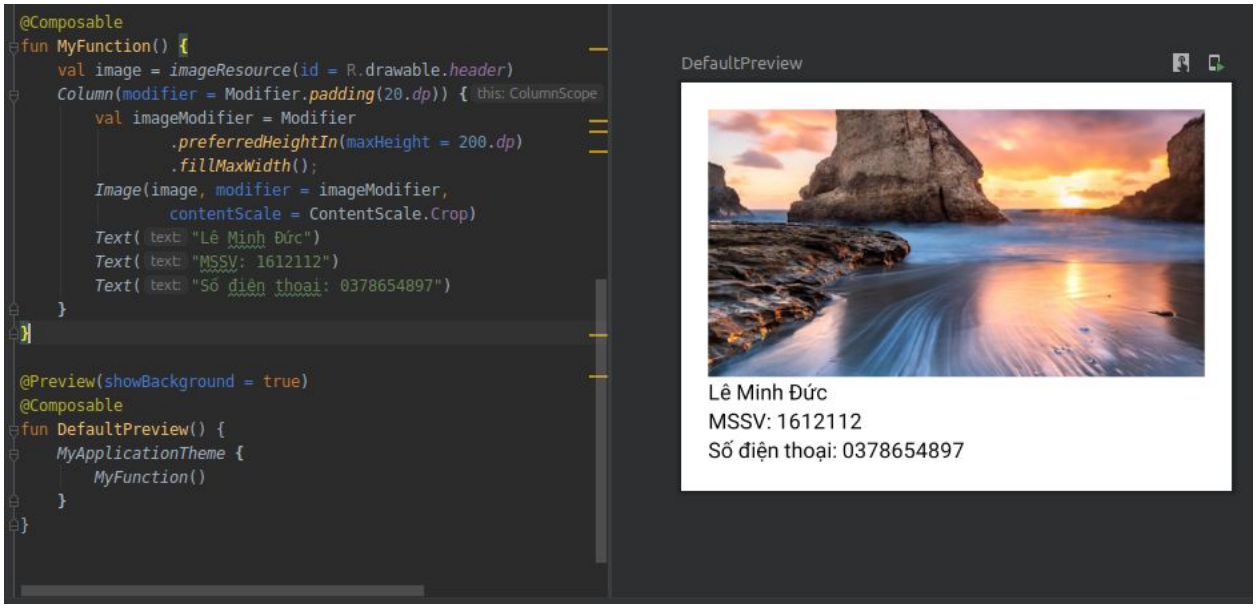
Dùng từ khóa **modifier** để cho phép cấu hình các style, trong trường hợp này thì xét **padding** cho nội dung trong **column** đó, kết quả là các **text** trong **column** cách viền của **column** 20 dp.

Chèn hình ảnh vào ứng dụng

Trên **Android Studio**, click chọn **Resource manager**, sau đó click vào dấu **+** để thêm 1 hình ảnh vào trong project, sau đó có thể chèn hình ảnh đó vào UI như sau:



Ngoài ra cũng có thể chỉnh sửa kích thước của hình ảnh đó:



Ở đây khai báo biến **imageModifier** để định nghĩa chiều dài và rộng của hình ảnh, với:

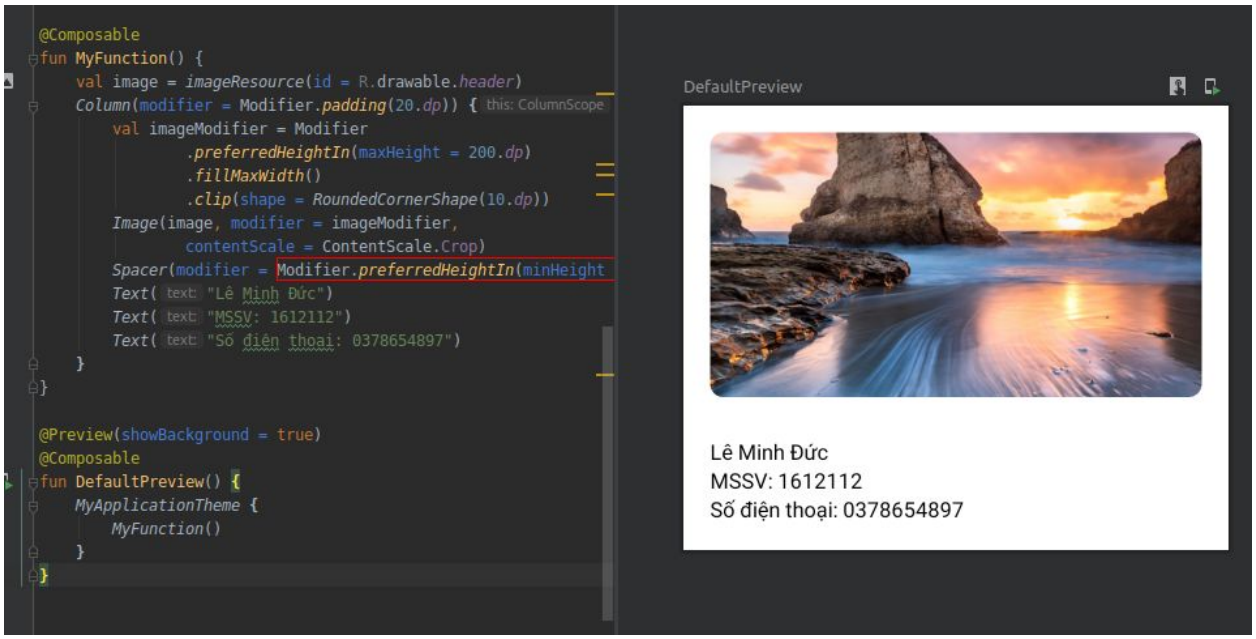
- **preferredHeightIn**: chỉ định chiều cao tối đa của hình ảnh la 200 dp.
- **fillMaxWidth**: hình ảnh có chiều rộng lớn nhất để phù hợp với thành phần cha của nó (ở đây là **column**)

Ngoài ra cần phải định nghĩa **contentScale = contentScale.Crop** để hình ảnh có thể lấp đầy chiều rộng của màn hình và được cắt đi (nếu cần) với chiều cao phù hợp.

Material design

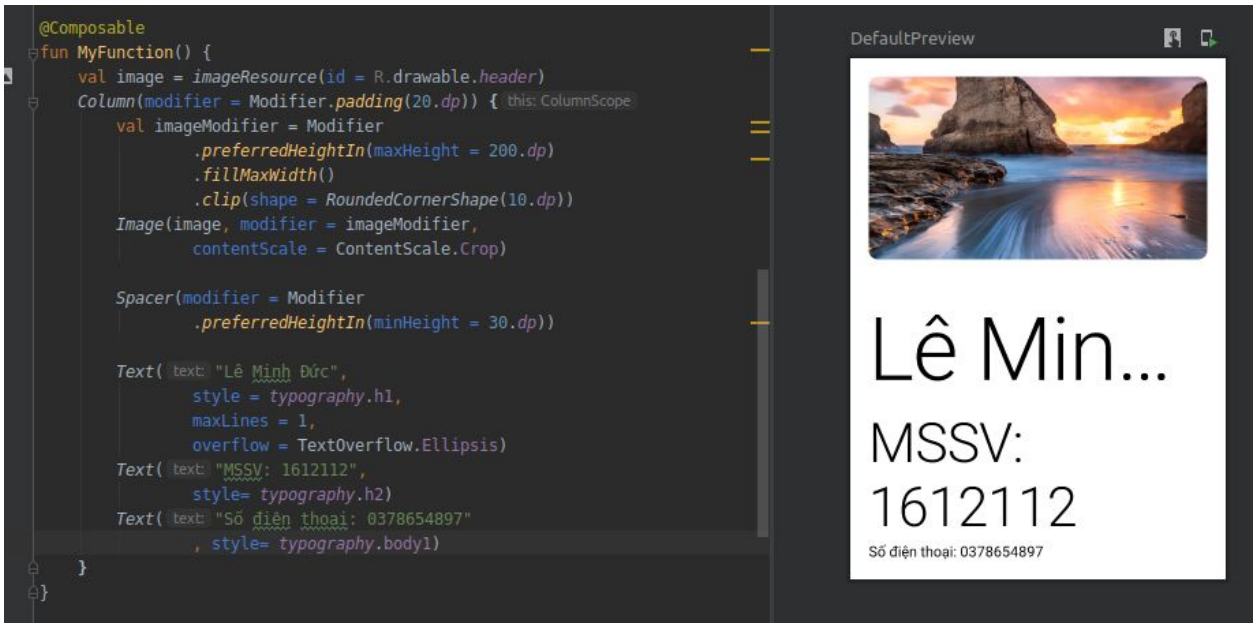
Chỉnh sửa hình dạng

Một trong những thành phần chính của material design là chỉnh sửa hình dạng của các component. Sử dụng hàm **click()** với thuộc tính **shape = RoundedCornerShape()** để bo tròn các góc của 1 hình ảnh và ta có kết quả như sau:



Chỉnh sửa style cho các text

Ta có thể thêm các style cho các text như sau:



Ở dòng text đầu tiên, ta định nghĩa text này như 1 thẻ <h1>, chỉ được nằm tối đa ở 1 dòng, những ký tự nào không được hiển thị ra sẽ được thay bằng dấu …

Tương tự cho các dòng text tiếp theo.

Source code ứng dụng demo

https://github.com/minhduc1612112/Jetpack-Compose-Android

Tham khảo

Nội dung của bài báo cáo này follow theo: Jetpack compose basic

Tham khảo khác: Jetpack Compose - UI framework mới của Android