**Jetpack Compose**

Введение в Jetpack Compose

Jetpack Compose представляет собой инновационный инструментарий для создания декларативных пользовательских интерфейсов в Android-приложениях. Вместо традиционного подхода, основанного на XML-разметке, Jetpack Compose использует язык программирования Kotlin для описания пользовательского интерфейса в виде кода. Этот подход делает процесс разработки более гибким, интуитивным и обеспечивает лучшую поддержку при изменении данных.

Сравнение с XML

Одной из ключевых особенностей Jetpack Compose является его отличие от традиционного использования XML для описания макетов. В Compose элементы интерфейса автоматически перерисовываются при изменении данных, что значительно упрощает управление состоянием пользовательского интерфейса. В то время как в XML требуется явно отслеживать состояния элементов и вызывать соответствующие функции для их обновления, в Jetpack Compose это происходит более естественным образом благодаря декларативному стилю программирования.

Использование функции превью

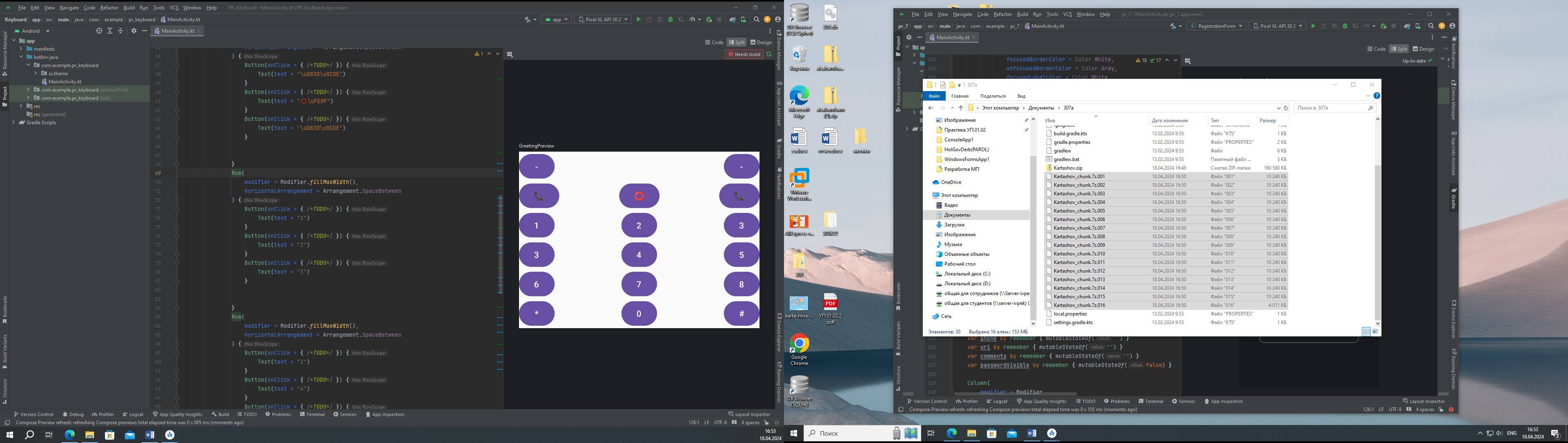
Функция превью является одним из ключевых инструментов в разработке с Jetpack Compose. Эта функция позволяет разработчикам непосредственно в коде видеть предварительный просмотр того, как будет выглядеть пользовательский интерфейс при изменении кода. Такой подход обеспечивает мгновенную обратную связь и ускоряет процесс разработки, позволяя разработчикам быстро экспериментировать с внешним видом и структурой интерфейса.

Практическая работа №1 Использование контейнеров Column и Row

Введение

Jetpack Compose предоставляет мощные инструменты для создания динамичных пользовательских интерфейсов, и одним из ключевых элементов являются контейнеры Column и Row. Эти контейнеры позволяют удобно располагать элементы в определенном порядке и с заданными расстояниями между ними.

package com.example.pr\_keyboard  
  
import android.os.Bundle  
import androidx.activity.ComponentActivity  
import androidx.activity.compose.setContent  
import androidx.compose.foundation.layout.Arrangement  
import androidx.compose.foundation.layout.Box  
import androidx.compose.foundation.layout.Column  
import androidx.compose.foundation.layout.Row  
import androidx.compose.foundation.layout.fillMaxWidth  
import androidx.compose.material3.Button  
import androidx.compose.material3.Text  
import androidx.compose.runtime.Composable  
import androidx.compose.ui.Modifier  
import androidx.compose.ui.tooling.preview.Preview  
  
class MainActivity : ComponentActivity() {  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 *setContent* **{** GreetingPreview()  
 **}** }  
  
 @Composable  
 fun Greeting(name: String, modifier: Modifier = Modifier) {  
 Text(  
 text = "Hello $name!",  
 modifier = modifier  
 )  
 }  
 }  
  
 @Preview(showBackground = true)  
 @Composable  
 fun GreetingPreview() {  
 Box(modifier = Modifier.*fillMaxWidth*()) **{** Column **{** Row(  
 modifier = Modifier.*fillMaxWidth*(),  
 horizontalArrangement = Arrangement.SpaceBetween  
 ) **{** Button(onClick = **{** /\**TODO*\*/ **}**) **{** Text(text = "-")  
 **}** Button(onClick = **{** /\**TODO*\*/ **}**) **{** Text(text = "-")  
 **}  
  
  
  
 }** Row(  
 modifier = Modifier.*fillMaxWidth*(),  
 horizontalArrangement = Arrangement.SpaceBetween  
 ) **{** Button(onClick = **{** /\**TODO*\*/ **}**) **{** Text(text = "\uD83D\uDCDE")  
 **}** Button(onClick = **{** /\**TODO*\*/ **}**) **{** Text(text = "⭕\uFE0F")  
 **}** Button(onClick = **{** /\**TODO*\*/ **}**) **{** Text(text = "\uD83D\uDCDE")  
 **}  
  
  
 }** Row(  
 modifier = Modifier.*fillMaxWidth*(),  
 horizontalArrangement = Arrangement.SpaceBetween  
 ) **{** Button(onClick = **{** /\**TODO*\*/ **}**) **{** Text(text = "1")  
 **}** Button(onClick = **{** /\**TODO*\*/ **}**) **{** Text(text = "2")  
 **}** Button(onClick = **{** /\**TODO*\*/ **}**) **{** Text(text = "3")  
 **}  
  
  
 }** Row(  
 modifier = Modifier.*fillMaxWidth*(),  
 horizontalArrangement = Arrangement.SpaceBetween  
 ) **{** Button(onClick = **{** /\**TODO*\*/ **}**) **{** Text(text = "3")  
 **}** Button(onClick = **{** /\**TODO*\*/ **}**) **{** Text(text = "4")  
 **}** Button(onClick = **{** /\**TODO*\*/ **}**) **{** Text(text = "5")  
 **}  
  
  
 }** Row(  
 modifier = Modifier.*fillMaxWidth*(),  
 horizontalArrangement = Arrangement.SpaceBetween  
 ) **{** Button(onClick = **{** /\**TODO*\*/ **}**) **{** Text(text = "6")  
 **}** Button(onClick = **{** /\**TODO*\*/ **}**) **{** Text(text = "7")  
 **}** Button(onClick = **{** /\**TODO*\*/ **}**) **{** Text(text = "8")  
 **}  
  
 }** Row(  
 modifier = Modifier.*fillMaxWidth*(),  
 horizontalArrangement = Arrangement.SpaceBetween  
 ) **{** Button(onClick = **{** /\**TODO*\*/ **}**) **{** Text(text = "\*")  
 **}** Button(onClick = **{** /\**TODO*\*/ **}**) **{** Text(text = "0")  
 **}** Button(onClick = **{** /\**TODO*\*/ **}**) **{** Text(text = "#")  
 **}  
  
  
 }  
  
 }  
 }** }



Практическая работа №2-3 Card, Box, Image

Card

Card в Jetpack Compose представляет собой контейнер, обычно используемый для размещения других элементов и создания карточек в пользовательском интерфейсе. Карточки обычно служат для выделения информации и создания структурированного макета.

package com.example.pr2  
  
  
  
import android.os.Bundle  
import androidx.activity.ComponentActivity  
import androidx.activity.compose.setContent  
import androidx.compose.foundation.Image  
import androidx.compose.foundation.background  
import androidx.compose.foundation.layout.Arrangement  
import androidx.compose.foundation.layout.Column  
import androidx.compose.foundation.layout.Row  
import androidx.compose.foundation.layout.fillMaxSize  
import androidx.compose.foundation.layout.fillMaxWidth  
import androidx.compose.foundation.layout.padding  
import androidx.compose.foundation.layout.size  
import androidx.compose.foundation.rememberScrollState  
import androidx.compose.foundation.shape.*CircleShape*import androidx.compose.foundation.verticalScroll  
import androidx.compose.material3.Card  
import androidx.compose.material3.CardDefaults  
import androidx.compose.material3.Text  
import androidx.compose.runtime.Composable  
import androidx.compose.ui.Modifier  
import androidx.compose.ui.draw.clip  
import androidx.compose.ui.graphics.Color  
import androidx.compose.ui.res.painterResource  
import androidx.compose.ui.tooling.preview.Preview  
import androidx.compose.ui.unit.dp  
  
  
class MainActivity2 : ComponentActivity() {  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 *setContent* **{** Prog2()  
  
 **}** }  
  
  
  
 @Preview(showBackground = true)  
 @Composable  
 fun Prog2() {  
  
 Column(modifier = Modifier  
 .*fillMaxSize*()  
 .*background*(Color.Gray)  
 .*verticalScroll*(rememberScrollState())  
 )**{** ListItem("String", "String")  
 ListItem("String", "String")  
 ListItem("String", "String")  
 ListItem("String", "String")  
 ListItem("String", "String")  
 ListItem("String", "String")  
 ListItem("String", "String")  
  
  
 **}** }  
}  
  
@Composable  
fun ListItem(title:String, content: String) {  
 Card(  
  
 modifier = Modifier  
 .*fillMaxWidth*()  
 .*padding*(20.*dp*),  
 elevation = CardDefaults.cardElevation(5.*dp*),  
  
 )  
 **{** Row(  
 modifier = Modifier  
 .*fillMaxWidth*(),  
 horizontalArrangement = Arrangement.Start  
 )  
 **{** Image(  
  
 painter = painterResource(id = R.drawable.*ic\_launcher\_background*),  
 contentDescription = "Описание изображения",  
 modifier = Modifier  
 .*clip*(*CircleShape*)  
 .*size*(80.*dp*)  
  
 )  
 Column(  
 modifier = Modifier  
 .*size*(90.*dp*)  
  
 .*padding*(end = 10.*dp*),  
  
  
 ) **{** Text(text = title)  
 Text(text = content)  
 **}  
  
  
 }  
 }**}

Практическая работа №4 Взаимодействие с кодом Kotlin и состояния

В Jetpack Compose, разработанном на языке Kotlin, взаимодействие с кодом становится более выразительным и удобным. В этом контексте использование переменных, условных конструкций, циклов и функций играет ключевую роль в создании динамичных пользовательских интерфейсов. Кроме того, Jetpack Compose предоставляет механизмы управления состоянием компонентов, в частности, с помощью MutableState и функции remember.

package com.example.swetof\_2  
  
import android.os.Bundle  
import androidx.activity.ComponentActivity  
import androidx.activity.compose.setContent  
import androidx.compose.foundation.background  
import androidx.compose.foundation.layout.Arrangement  
import androidx.compose.foundation.layout.Column  
import androidx.compose.foundation.layout.fillMaxSize  
import androidx.compose.foundation.layout.size  
import androidx.compose.foundation.shape.*CircleShape*import androidx.compose.material3.Button  
import androidx.compose.material3.ButtonDefaults  
import androidx.compose.material3.Surface  
import androidx.compose.material3.Text  
import androidx.compose.runtime.Composable  
import androidx.compose.runtime.getValue  
import androidx.compose.runtime.mutableStateOf  
import androidx.compose.runtime.remember  
import androidx.compose.runtime.setValue  
import androidx.compose.ui.Alignment  
import androidx.compose.ui.Modifier  
import androidx.compose.ui.draw.clip  
import androidx.compose.ui.graphics.Color  
import androidx.compose.ui.graphics.Color.Companion.*Green*import androidx.compose.ui.graphics.Color.Companion.*Red*import androidx.compose.ui.graphics.Color.Companion.*Yellow*import androidx.compose.ui.tooling.preview.Preview  
import androidx.compose.ui.unit.dp  
  
class MainActivity : ComponentActivity() {  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 *setContent* **{** Pr4()  
  
 **}** }  
}  
  
@Preview(showBackground = true)  
@Composable  
fun Pr4() {  
 var color by remember **{** *mutableStateOf*(Color.White) **}** Surface(  
 color = color,  
 modifier = Modifier.*fillMaxSize*()  
 ) **{** Column(modifier = Modifier.*fillMaxSize*(),  
 verticalArrangement = Arrangement.SpaceAround,  
 horizontalAlignment = Alignment.CenterHorizontally)**{** Button(  
 onClick = **{** color = *Red* **}**,  
 modifier = Modifier  
 .*size*(210.*dp*)  
 .*clip*(*CircleShape*),  
 colors = ButtonDefaults  
 .buttonColors(  
 containerColor = Color.Red  
 )  
 ) **{  
  
  
  
 }** Button(  
 onClick = **{** color = *Yellow* **}**,  
 modifier = Modifier  
 .*size*(210.*dp*)  
 .*clip*(*CircleShape*),  
 colors = ButtonDefaults  
 .buttonColors(  
 containerColor = Color.Yellow  
 )  
 ) **{  
  
 }** Button(  
 onClick = **{** color = *Green* **}**,  
 modifier = Modifier  
 .*size*(210.*dp*)  
 .*clip*(*CircleShape*),  
 colors = ButtonDefaults  
 .buttonColors(  
 containerColor = Color.Green  
 )  
 ) **{  
  
 }  
  
 }  
 }**}

