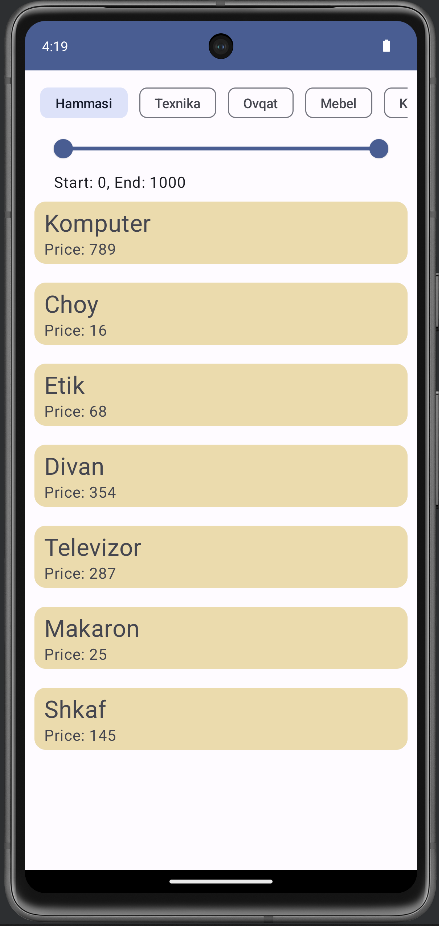
**Recycler view sorting(filter)**



**Data toyyorlash**

Birinchi bo`lib recycler(LazyColumn) uchun model yaratamiz

data class Product(  
 var name: String,  
 var price: Int,  
 var category: String  
)

Shu bilan birga category va bir nechta elementlik massiv yaratamiz

val *categories* = *listOf*<String>(  
 "Hammasi",  
 "Texnika",  
 "Ovqat",  
 "Mebel",  
 "Kiyim"  
)

var *products* = *mutableListOf*<Product>(  
 Product(  
 "Komputer",  
 789,  
 *categories*[1]  
 ),  
  
 Product(  
 "Choy",  
 16,  
 *categories*[2]  
 ),

…)

var *sliderRangeValue* = 0f..1000f

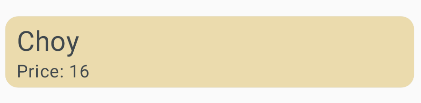
Sliderni qiymatini saqlash uchun kerak

Tepadagilarni hammasini bitta filega yozsa bo`ladi.

**Item view yig`ish**

@Composable  
fun ProductView(product: Product) {  
 Box(  
 modifier = Modifier  
 .*fillMaxWidth*()  
 .*padding*(vertical = 10.*dp*)  
 ) **{** Card **{** Column(  
 modifier = Modifier  
 .*fillMaxWidth*()  
 .*background*(*productItemBG*)  
 .*padding*(vertical = 5.*dp*, horizontal = 10.*dp*)  
 ) **{** Text(  
 text = product.name,  
 fontSize = 25.*sp*,  
 )  
 Text(text = "Price: ${product.price}")  
 **}  
 }  
 }**}

Bu funksiyada oddiy Box Card Column elementlaridan bitta product uchun dizayn yigamiz. Ko`rinishi shunaqa:



**Chips yasash**



Tepadagi uni ko`rinishi. Qanday yasash pastdagi kodda bo`ladi:

@SuppressLint("MutableCollectionMutableState")  
@OptIn(ExperimentalMaterial3Api::class)  
@Composable  
fun Chips(  
  
): MutableList<Product> {  
  
 var selectedCategory by remember **{** *mutableStateOf*(*categories*[0])  
 **}** var productList by remember **{** *mutableStateOf*(*products*)  
 **}** LazyRow(modifier = Modifier.*fillMaxWidth*()) **{** *items*(*categories*) **{** item **->** FilterChip(modifier = Modifier.*padding*(horizontal = 6.*dp*),   
 selected = (item == selectedCategory), onClick = **{** selectedCategory = item  
  
 var tempProducts = *mutableListOf*<Product>()  
  
 for (p in *products*) {  
 if (selectedCategory == *categories*[0]) {  
 tempProducts = *products* } else if (selectedCategory == p.category) {  
 tempProducts.add(p)  
 }  
 }  
  
 productList = *sortByPrice*(*sliderRangeValue*, tempProducts)  
  
 **}**, label = **{** Text(text = item)  
 **}**)  
 **}  
 }** return productList  
}

selectedCategory – bosilgandagi elementni saqlaydi

productList - Chips()funksiyasi qaytarish kerak bolgan list

tempProducts – qaysi product tanlangan category ga kirishini saralash uchun

*sortByPrice*()– funksiyasi pastroqda chuntirilgan

fun sortByPrice(  
 range: ClosedFloatingPointRange<Float>,  
 list: MutableList<Product>):MutableList<Product> {  
  
 var tempProducts = *mutableListOf*<Product>()  
  
 for (p in list){  
 if (p.price.toFloat() in range){  
 tempProducts.add(p)  
 }  
 }  
 return tempProducts  
}

**Slider yasash**



Slider shunaqa narsa

Rasmdagi RangeSlider, chinki ikkita tarafdan tanlasa boladi(start, end)

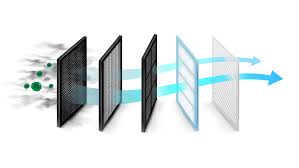
Slider qiymatiga faqat float qabul qilarkan

@SuppressLint("MutableCollectionMutableState")  
@OptIn(ExperimentalMaterial3Api::class)  
@Composable  
fun RangeSliderView(  
 categorisedProducts: MutableList<Product>  
): MutableList<Product> {  
  
 var productList by remember **{** *mutableStateOf*(*products*)  
 **}** productList = categorisedProducts  
  
 Column(  
 modifier = Modifier.*fillMaxWidth*().*padding*(horizontal = 20.*dp*)  
 ) **{** RangeSlider(  
 value = *sliderRangeValue*,

onValueChange = **{** newValues **->** *sliderRangeValue* = newValues  
 **}**,

onValueChangeFinished = **{** productList = *sortByPrice*(*sliderRangeValue*, categorisedProducts)  
  
 **}**, valueRange = 0f..1000f, steps = 0  
 )  
 Text(  
 text = "Start: ${*sliderRangeValue*.start.toInt()}, " +  
 "End: ${*sliderRangeValue*.endInclusive.toInt()}"  
 )  
 **}** return productList  
}

**Hammasini birlashtirish**



Listni saralash shu filtrga oxshab ishlidi

Avval category boyicha saralidi, keyin narx boyicha

@SuppressLint("MutableCollectionMutableState")  
@Composable  
fun MyView() {  
  
 var categorisedProducts by remember **{** *mutableStateOf*(*products*)  
 **}** var sortedProducts by remember **{** *mutableStateOf*(categorisedProducts)  
 **}** Column(  
 modifier = Modifier  
 .*fillMaxSize*()  
 .*padding*(10.*dp*)  
 ) **{** categorisedProducts = Chips()  
  
 sortedProducts = RangeSliderView(categorisedProducts)  
  
LazyColumn **{** *items*(sortedProducts) **{** product **->** ProductView(product = product)  
 **}  
 }  
 }**}

[GitHub](https://github.com/sqljava/RecyclerSort)

