**Московский государственный технический**

**университет им. Н. Э. Баумана**

Факультет «Радиотехнический»

Кафедра РТ5 «Системы обработки информации и управления»

Курс «Парадигмы и конструкции языков программирования»

Отчет по домашней работе

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы РТ5-31Б |  | преподаватель каф. ИУ5 |
| Суровец Е. А. |  | Гапанюк Ю.Е. |
|  |  |  |

Москва, 2024 г.

**Текст программы**

GifsData.kt

package com.example.surovets\_kotlin\_lv1\_2  
  
data class Original(  
 val height: String,  
 val width: String,  
 val size: String,  
 val url: String,  
 val mp4\_size: String,  
 val mp4: String,  
 val webp\_size: String,  
 val webp: String,  
 val frames: String,  
 val hash: String  
)  
  
data class Images(  
 val original: Original  
)  
  
  
data class Gif(  
 val type: String,  
 val id: String,  
 val url: String,  
 val slug: String,  
 val bitly\_gif\_url: String,  
 val bitly\_url: String,  
 val embed\_url: String,  
 val username: String,  
 val source: String,  
 val title: String,  
 val rating: String,  
 val content\_url: String,  
 val source\_tld: String,  
 val source\_post\_url: String,  
 val is\_stecker: String,  
 val import\_datetime: String,  
 val trending\_datetime: String,  
 val images: Images  
)  
  
data class Gifs(  
 val data: List<Gif>  
)

GifsRetrofit.kt

package com.example.surovets\_kotlin\_lv1\_2  
  
import retrofit2.Retrofit  
import retrofit2.converter.gson.GsonConverterFactory  
  
class GifsRetrofit(url: String) {  
 private val retrofit: Retrofit by *lazy* **{** Retrofit.Builder()  
 .baseUrl(url)  
 .addConverterFactory(GsonConverterFactory.create())  
 .build()  
 **}** val gifsService: GifsService by *lazy* **{** retrofit.create(GifsService::class.*java*)  
 **}** suspend fun requestGifs(url: String): Gifs? {  
 val response = gifsService.getGifs()  
 if (response.*isSuccessful*) {  
  
 val body = response.body()  
 if (body != null) {  
 return body  
 }  
 }  
 return null  
 }  
}

GifsService.kt

package com.example.surovets\_kotlin\_lv1\_2  
  
import retrofit2.http.GET  
  
interface GifsService {  
 @GET("v1/gifs/search?api\_key=zIzLK1XZS3iTh2nWVe1LhMdS3QyQcVsj&q=animal")  
 suspend fun getGifs(): retrofit2.Response<Gifs>  
}

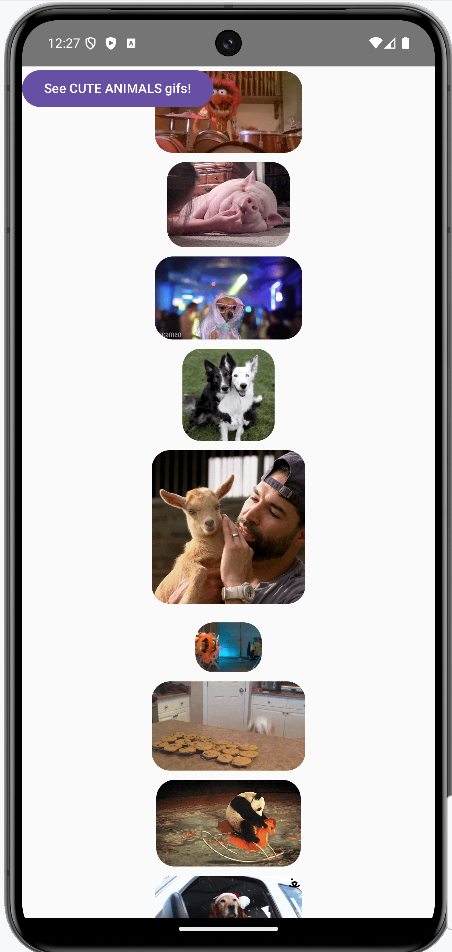
ImageWork.kt

package com.example.surovets\_kotlin\_lv1\_2  
  
import android.content.Context  
import android.content.res.Configuration  
import android.media.Image  
import android.provider.ContactsContract.CommonDataKinds.Im  
import android.util.Log  
import androidx.compose.foundation.clickable  
import androidx.compose.foundation.layout.Box  
import androidx.compose.foundation.layout.Column  
import androidx.compose.foundation.layout.PaddingValues  
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.lazy.LazyColumn  
import androidx.compose.foundation.lazy.grid.GridCells  
import androidx.compose.foundation.lazy.grid.LazyVerticalGrid  
import androidx.compose.foundation.lazy.grid.items  
import androidx.compose.foundation.lazy.items  
import androidx.compose.foundation.shape.RoundedCornerShape  
import androidx.compose.material.icons.Icons  
import androidx.compose.material.icons.filled.Refresh  
import androidx.compose.material3.Button  
import androidx.compose.material3.CircularProgressIndicator  
import androidx.compose.material3.Icon  
import androidx.compose.material3.Text  
import androidx.compose.runtime.Composable  
import androidx.compose.runtime.getValue  
import androidx.compose.runtime.mutableStateOf  
import androidx.compose.runtime.rememberCoroutineScope  
import androidx.compose.runtime.saveable.rememberSaveable  
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.layout.ContentScale  
import androidx.compose.ui.platform.LocalConfiguration  
import androidx.compose.ui.platform.LocalContext  
import androidx.compose.ui.unit.dp  
import coil.compose.AsyncImage  
import coil.decode.GifDecoder  
import coil.request.ImageRequest  
import kotlinx.coroutines.CoroutineExceptionHandler  
import kotlinx.coroutines.CoroutineScope  
import kotlinx.coroutines.launch  
  
  
@Composable  
fun GifsScreen(gifsRetrofit: GifsRetrofit) {  
 var gifsList by rememberSaveable **{** mutableStateOf<List<Gif>?>(null) **}** var isLoading by rememberSaveable **{** mutableStateOf(false) **}** var isFailed by rememberSaveable **{** mutableStateOf(false) **}** var isList by rememberSaveable **{** mutableStateOf(false) **}** var isOneGif by rememberSaveable **{** mutableStateOf(false) **}***// var oneGif by rememberSaveable { mutableStateOf(false) }* var oneGifImageRequest by rememberSaveable **{** mutableStateOf<ImageRequest?>(null) **}** val isLandscape = LocalConfiguration.current.orientation == Configuration.ORIENTATION\_LANDSCAPE  
 val coroutineScope = rememberCoroutineScope()  
 val handler = CoroutineExceptionHandler **{** \_, exception **->** run **{** isLoading = false  
 isFailed = true  
 isList = true  
 isOneGif = false  
 **}  
 }***// var oneGif : Gif? = null  
// var oneGifImageRequest : ImageRequest? = null* val BASE\_URL =  
 "https://api.giphy.com/"  
  
 Column(modifier = Modifier.fillMaxSize()) **{** when {  
 isLoading -> {  
 Log.i("loading", "load")  
 Box(  
 modifier = Modifier  
 .fillMaxSize()  
 .weight(1f),  
 contentAlignment = Alignment.Center,  
 ) **{** CircularProgressIndicator()  
 **}** }  
  
 isFailed -> {  
 Box(  
 modifier = Modifier.fillMaxSize(),  
 contentAlignment = Alignment.Center  
 ) **{** Column() **{** Box(modifier = Modifier.clickable(onClick = **{** ShowGifs(BASE\_URL,  
 coroutineScope,  
 gifsRetrofit,  
 handler,  
 **{** gifsList = it **}**,  
 **{**isList = it**}**,  
 **{** isLoading = it **}**,  
 **{** isFailed = it **}** )  
 **}**)) **{** Icon(  
 imageVector = Icons.Default.Refresh,  
 contentDescription = null,  
 modifier = Modifier.align(Alignment.Center),  
 )  
 **}  
 }  
 }** }  
  
 isOneGif -> {  
 DrawGif(oneGifImageRequest)  
 }  
  
 isList -> {  
 Log.i("ELSE", "else")  
 if (isLandscape) {  
 LazyVerticalGrid(  
 columns = GridCells.Fixed(3),  
 modifier = Modifier.weight(1f),  
 contentPadding = PaddingValues(10.dp)  
 )  
 **{** items(gifsList ?: emptyList()) **{** gif **->** val imageRequest = ImageRequest.Builder(LocalContext.current)  
 .data(gif.images.original.url)  
 .decoderFactory(GifDecoder.Factory())  
 .build()  
  
  
 AsyncImage(  
 model = imageRequest,  
 contentDescription = null,  
 modifier = Modifier  
 .padding(5.dp)  
 .clip(RoundedCornerShape(20.dp)),  
 contentScale = ContentScale.Crop  
 )  
 **}  
 }** } else {  
*// if (isList) {* Log.i("COL", "col")  
 LazyColumn(modifier = Modifier.fillMaxSize()) **{** items(gifsList ?: emptyList()) **{** gif **->** val imageRequest = ImageRequest.Builder(LocalContext.current)  
 .data(gif.images.original.url)  
 .decoderFactory(GifDecoder.Factory())  
 .build()  
  
 Box(  
 modifier = Modifier.fillMaxWidth(),  
 contentAlignment = Alignment.Center  
 ) **{** AsyncImage(  
 model = imageRequest,  
 contentDescription = null,  
 modifier = Modifier  
 .clickable **{** isOneGif = true  
*// oneGif = gif* oneGifImageRequest = imageRequest  
 **}** .padding(5.dp)  
 .clip(RoundedCornerShape(20.dp)),  
 alignment = Alignment.Center,  
 contentScale = ContentScale.Crop,  
 )  
 **}  
 }  
 }***// } else {  
// DrawGif(oneGifImageRequest)  
// }* }  
 }  
 }  
 **}** Row(modifier = Modifier.fillMaxWidth()) **{** Button(onClick = **{** coroutineScope.launch(handler) **{}** ShowGifs(  
 BASE\_URL,  
 coroutineScope,  
 gifsRetrofit,  
 handler,  
 **{** gifsList = it **}**,  
 **{**isList = it**}**,  
 **{** isLoading = it **}**,  
 **{** isFailed = it **}** )  
 **}**, modifier = Modifier.align(Alignment.Bottom)) **{** Text(text = "See CUTE ANIMALS gifs!")  
 **}  
 }**}  
  
  
fun ShowGifs(  
 url: String,  
 coroutineScope: CoroutineScope,  
 gifsRetrofit: GifsRetrofit,  
 handler: CoroutineExceptionHandler,  
 setGifsList: (List<Gif>?) -> Unit,  
 setList: (Boolean) -> Unit,  
 setLoading: (Boolean) -> Unit,  
 setFailed: (Boolean) -> Unit  
) {  
 Log.i("SHOW", "show")  
 coroutineScope.launch(handler) **{** setLoading(true)  
 setFailed(false)  
 setList(true)  
 try {  
 val response = gifsRetrofit.requestGifs(url)  
 setGifsList(response?.data)  
 } catch (e: Exception) {  
 setFailed(true)  
 } finally {  
 setLoading(false)  
 }  
 **}**}  
  
  
  
@Composable  
fun DrawGif(imageRequest: ImageRequest?) {  
 Box(  
 modifier = Modifier.fillMaxSize(),  
 contentAlignment = Alignment.Center  
 ) **{** AsyncImage(  
 model = imageRequest,  
 contentDescription = null,  
 modifier = Modifier  
 .padding(5.dp)  
 .clip(RoundedCornerShape(20.dp)),  
 alignment = Alignment.Center,  
 contentScale = ContentScale.Crop,  
 )  
 **}**}

MainActivity.kt

package com.example.surovets\_kotlin\_lv1\_2  
  
import android.os.Bundle  
import androidx.activity.ComponentActivity  
import androidx.activity.compose.setContent  
  
class MainActivity : ComponentActivity() {  
 val BASE\_URL =  
 "https://api.giphy.com/"  
  
 private val gifsRetrofit by *lazy* **{** GifsRetrofit(BASE\_URL) **}** override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 *setContent* **{** GifsScreen(gifsRetrofit)  
 **}** }  
}

**Результат выполнения**

****