МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА – Российский технологический университет»
РТУ МИРЭА

Институт Информационных технологий

Кафедра Математического обеспечения и стандартизации информационных технологий

Отчет по Модулю 3 – Основы языка Котлин

по дисциплине «Проектирования и разработка мобильных приложений на языке Котлин»

Выполнили:
Студенты группы ИКБО-65-23 Олефиров Г.Г.

Проверил: Егоров Н.С.

2025 г.

Часть III – Практическое задание
Задание 1: реализовать отображение прокручиваемого списка Создать вертикальный прокручиваемый список, используя LazyColumn. Определить data class для представления элемента списка (карточки), который должен содержать заголовок, описание и идентификатор ресурса изображения. Создать коллекцию объектов этого data class. В LazyColumn использовать функцию items для итерации по коллекции данных и отображения каждого элемента. Каждый элемент списка должен быть представлен компонентом Card, содержащим Image и два компонента Text (для заголовка и описания), расположенные в Row.
Задание 2: выполнить настройку темы приложения Настроить визуальное оформление приложения. Использовать Material Theme Builder для создания пользовательской цветовой палитры. Экспортировать сгенерированную тему и заменить ею стандартные файлы Color.kt, Theme.kt и Type.kt в проекте. Загрузить любое семейство шрифтов с Google Fonts и добавить файлы шрифтов в каталог res/font. Определить пользовательскую типографику (Typography) в файле Type.kt, используя загруженные шрифты. Применить созданную типографику в файле Theme.kt. Определить пользовательские формы для компонентов, создав файл Shape.kt и изменив в нем стандартные значения small, medium и large. Применить эти формы в Theme.kt.

```
package com.example.firstapplication

import android.graphics.Bitmap
import android.graphics.BitmapFactory
import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.Image
import androidx.compose.foundation.layout.*
import androidx.compose.foundation.lazy.LazyColumn
import androidx.compose.foundation.lazy.items
import androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.material3.*
import androidx.compose.runtime.Composable
import androidx.compose.runtime.remember
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.draw.clip
import androidx.compose.ui.graphics.asImageBitmap
import androidx.compose.ui.graphics.painter.BitmapPainter
import androidx.compose.ui.layout.ContentScale
import androidx.compose.ui.platform.LocalContext
import androidx.compose.ui.unit.dp
import com.example.firstapplication.ui.theme.AppTheme
import kotlin.math.max

class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            AppTheme(darkTheme = true) {
                Surface(
                    modifier = Modifier.fillMaxSize(),
                    color = MaterialTheme.colorScheme.background
                ) {
                    ShipListScreen()
                }
            }
        }
    }
}

data class ShipItem(
    val title: String,
    val description: String,
    val imageResId: Int
)

// ПРАВИЛЬНО: используй R.drawable (не android.R.drawable!)
val shipItemList = listOf(
    ShipItem(
        title = "Carrack",
        description = "Корабль для исследования с медицинским отсеком",
        imageResId = R.drawable.corabl1  // ТВОЯ картинка
    ),
    ShipItem(
        title = "Constellation Andromeda",
```

```
        description = "Универсальный корабль с истребителем-сопровождением",
        imageResId = R.drawable.corabl2  // ТВОЯ картинка
    ),
    ShipItem(
        title = "Avenger Titan",
        description = "Надежный и универсальный стартовый корабль",
        imageResId = R.drawable.corabl3  // ТВОЯ картинка
    ),
    ShipItem(
        title = "Cutlass Black",
        description = "Пиратский многоцелевой корабль с боковыми дверьми",
        imageResId = R.drawable.corabl4  // ТВОЯ картинка
    ),
    ShipItem(
        title = "600i Explorer",
        description = "Роскошный исследовательский корабль премиум-класса",
        imageResId = R.drawable.corabl5  // ТВОЯ картинка
    ),
    ShipItem(
        title = "Gladius",
        description = "Легкий и маневренный истребитель для догфайтов",
        imageResId = R.drawable.corabl6  // ТВОЯ картинка
    ),
    ShipItem(
        title = "Mercury Star Runner",
        description = "Корабль для перевозки данных и контрабанды",
        imageResId = R.drawable.corabl7  // ТВОЯ картинка
    ),
    ShipItem(
        title = "Vanguard Harbinger",
        description = "Дальний тяжелый истребитель с торпедным вооружением",
        imageResId = R.drawable.corabl8  // ТВОЯ картинка
    )
)

@Composable
fun ShipCard(item: ShipItem) {
    val context = LocalContext.current

    Card(
        modifier = Modifier
            .fillMaxWidth()
            .padding(horizontal = 16.dp, vertical = 8.dp)
    ) {
        Row(
            modifier = Modifier.padding(16.dp),
            verticalAlignment = Alignment.CenterVertically
        ) {
            // Загружаем изображение с уменьшением размера
            val imageBitmap = remember(item.imageResId) {
                val options = BitmapFactory.Options().apply {
                    inJustDecodeBounds = true
                }

                // Сначала только читаем размеры
                BitmapFactory.decodeResource(context.resources, item.imageResId, options)

                // Вычисляем целевой размер в пикселях (80dp * density)
                // Используем максимальную плотность для безопасности (xxxhdpi = 4.0)
                val density = context.resources.displayMetrics.density
                val targetSizeDp = 80f
                val targetSizePx = (targetSizeDp * density).toInt()

                // Вычисляем коэффициент масштабирования (inSampleSize должен быть степенью двойки)
                val maxSide = max(options.outWidth, options.outHeight).coerceAtLeast(1)
                val scale = if (maxSide <= targetSizePx) {
                    1
                } else {
                    // Находим ближайшую степень двойки, которая уменьшит изображение до нужного размера
                    var sampleSize = 1
                    while (maxSide / (sampleSize * 2) > targetSizePx) {
                        sampleSize *= 2
                    }
                    sampleSize
                }

                // Загружаем с уменьшением
                options.inJustDecodeBounds = false
                options.inSampleSize = scale
                options.inScaled = false // Отключаем автоматическое масштабирование

                val bitmap = BitmapFactory.decodeResource(context.resources, item.imageResId, options)

                // Дополнительно масштабируем до точного размера, если нужно
                val finalBitmap = if (bitmap != null) {
                    val maxBitmapSide = max(bitmap.width, bitmap.height)
                    if (maxBitmapSide > targetSizePx) {
                        val scaleFactor = targetSizePx.toFloat() / maxBitmapSide
                        val scaledWidth = (bitmap.width * scaleFactor).toInt().coerceAtLeast(1)
                        val scaledHeight = (bitmap.height * scaleFactor).toInt().coerceAtLeast(1)
                        Bitmap.createScaledBitmap(bitmap, scaledWidth, scaledHeight, true).also {
                            if (it != bitmap) bitmap.recycle()
                        }
                    } else {
                        bitmap
                    }
                } else {
                    Bitmap.createBitmap(1, 1, Bitmap.Config.ARGB_8888)
                }

                finalBitmap.asImageBitmap()
            }

            Image(
                painter = BitmapPainter(imageBitmap),
                contentDescription = item.title,
                modifier = Modifier
```

```
                    .size(80.dp)
                    .clip(RoundedCornerShape(8.dp)),
                contentScale = ContentScale.Crop
            )

            Spacer(modifier = Modifier.width(16.dp))

            Column {
                Text(
                    text = item.title,
                    style = MaterialTheme.typography.titleLarge
                )
                Text(
                    text = item.description,
                    style = MaterialTheme.typography.bodyMedium,
                    color = MaterialTheme.colorScheme.onSurfaceVariant,
                    modifier = Modifier.padding(top = 4.dp)
                )
            }
        }
    }
}

@Composable
fun ShipListScreen() {
    LazyColumn(
        modifier = Modifier.fillMaxSize(),
        verticalArrangement = Arrangement.spacedBy(8.dp),
        contentPadding = PaddingValues(vertical = 16.dp)
    ) {
        item {
            Text(
                text = "Каталог кораблей",
                style = MaterialTheme.typography.headlineLarge,
                modifier = Modifier.padding(start = 16.dp, end = 16.dp, bottom = 8.dp)
            )
        }

        items(shipItemList) { item ->
            ShipCard(item = item)
        }
    }
}
```

Листинг 1 - кода приложения для задания 1 и 2

Задание 1: реализовать отображение прокручиваемого списка

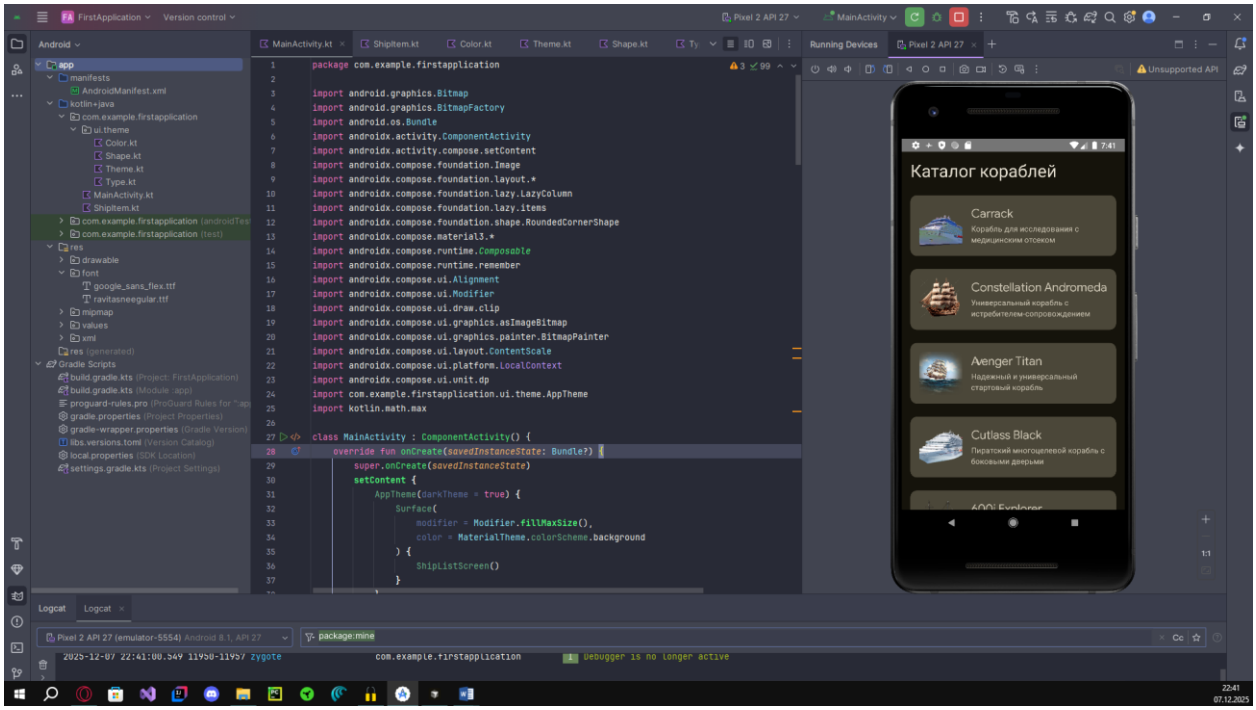


Рисунок 1 – Приложение

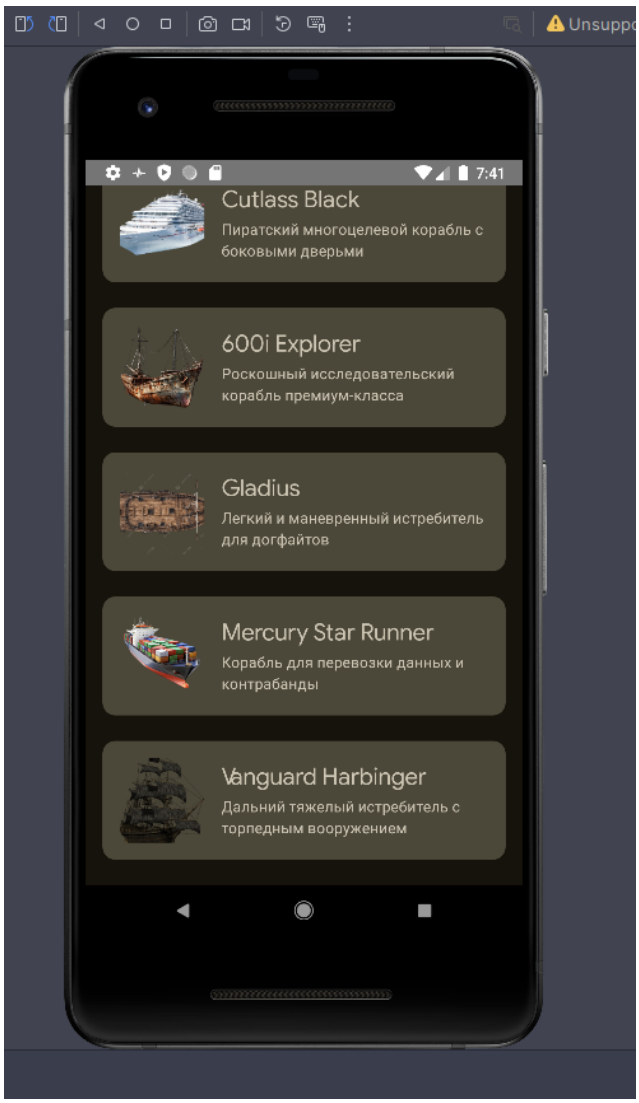


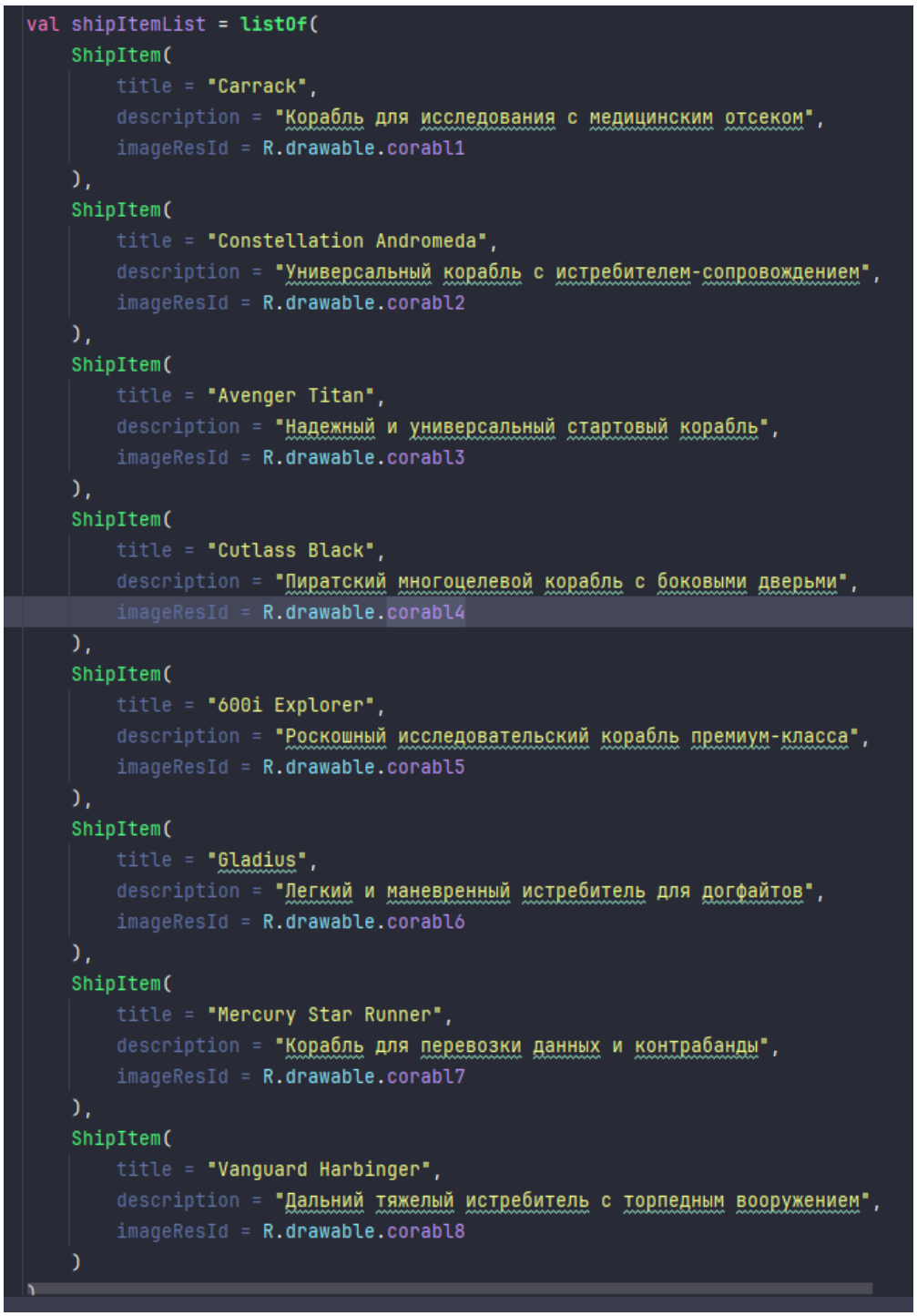
Рисунок 2 - Приложение с вертикальный прокручиваемый списком прокрученное вниз

Определить data class для представления элемента списка (карточки), который должен содержать заголовок, описание и идентификатор ресурса изображения.

```
data class ShipItem(
    val title: String,
    val description: String,
    val imageResId: Int
)
```

Рисунок 3 - Определённый data class для представления элемента списка

Создать коллекцию объектов этого data class.



```
val shipItemList = listOf(
    ShipItem(
        title = "Carrack",
        description = "Корабль для исследования с медицинским отсеком",
        imageResId = R.drawable.corabl1
    ),
    ShipItem(
        title = "Constellation Andromeda",
        description = "Универсальный корабль с истребителем-сопровождением",
        imageResId = R.drawable.corabl2
    ),
    ShipItem(
        title = "Avenger Titan",
        description = "Надежный и универсальный стартовый корабль",
        imageResId = R.drawable.corabl3
    ),
    ShipItem(
        title = "Cutlass Black",
        description = "Пиратский многоцелевой корабль с боковыми дверьми",
        imageResId = R.drawable.corabl4
    ),
    ShipItem(
        title = "600i Explorer",
        description = "Роскошный исследовательский корабль премиум-класса",
        imageResId = R.drawable.corabl5
    ),
    ShipItem(
        title = "Gladius",
        description = "Легкий и маневренный истребитель для догфайтов",
        imageResId = R.drawable.corabl6
    ),
    ShipItem(
        title = "Mercury Star Runner",
        description = "Корабль для перевозки данных и контрабанды",
        imageResId = R.drawable.corabl7
    ),
    ShipItem(
        title = "Vanguard Harbinger",
        description = "Дальний тяжелый истребитель с торпедным вооружением",
        imageResId = R.drawable.corabl8
    )
)
```

Рисунок 4 -Созданная коллекцию объектов этого data class

В LazyColumn использовать функцию items для итерации по коллекции данных и отображения каждого элемента.

Рисунок 5 - функция items

Каждый элемент списка должен быть представлен компонентом Card, содержащим Image и два компонента Text (для заголовка и описания), расположенные в Row

```
@Composable
fun ShipCard(item: ShipItem) {
    val context = LocalContext.current

    Card(
        modifier = Modifier
            .fillMaxWidth()
            .padding(horizontal = 16.dp, vertical = 8.dp)
    ) {
        Row(
            modifier = Modifier.padding(16.dp),
            verticalAlignment = Alignment.CenterVertically
        ) {
            // Загружаем изображение с уменьшением размера
            val imageBitmap = remember(item.imageResId) {
                val options = BitmapFactory.Options().apply {
                    inJustDecodeBounds = true
                }

                // Сначала только читаем размеры
                BitmapFactory.decodeResource(context.resources,
item.imageResId, options)

                // Вычисляем целевой размер в пикселях (80dp * density)
                // Используем максимальную плотность для безопасности
(xxxhdpi = 4.0)
                val density = context.resources.displayMetrics.density
                val targetSizeDp = 80f
                val targetSizePx = (targetSizeDp * density).toInt()
```

```
                // Вычисляем коэффициент масштабирования (inSampleSize
должен быть степенью двойки)
                val maxSide = max(options.outWidth,
options.outHeight).coerceAtLeast(1)
                val scale = if (maxSide <= targetSizePx) {
                    1
                } else {
                    // Находим ближайшую степень двойки, которая уменьшит
изображение до нужного размера
                    var sampleSize = 1
                    while (maxSide / (sampleSize * 2) > targetSizePx) {
                        sampleSize *= 2
                    }
                    sampleSize
                }

                // Загружаем с уменьшением
                options.inJustDecodeBounds = false
                options.inSampleSize = scale
                options.inScaled = false // Отключаем автоматическое
масштабирование

                val bitmap = BitmapFactory.decodeResource(context.resources,
item.imageResId, options)

                // Дополнительно масштабируем до точного размера, если нужно
                val finalBitmap = if (bitmap != null) {
                    val maxBitmapSide = max(bitmap.width, bitmap.height)
                    if (maxBitmapSide > targetSizePx) {
                        val scaleFactor = targetSizePx.toFloat() /
maxBitmapSide
                        val scaledWidth = (bitmap.width *
scaleFactor).toInt().coerceAtLeast(1)
                        val scaledHeight = (bitmap.height *
scaleFactor).toInt().coerceAtLeast(1)
                        Bitmap.createScaledBitmap(bitmap, scaledWidth,
scaledHeight, true).also {
                            if (it != bitmap) bitmap.recycle()
                        }
                    } else {
                        bitmap
                    }
                } else {
                    Bitmap.createBitmap(1, 1, Bitmap.Config.ARGB_8888)
                }

                finalBitmap.asImageBitmap()
            }

            Image(
                painter = BitmapPainter(imageBitmap),
                contentDescription = item.title,
                modifier = Modifier
                    .size(80.dp)
                    .clip(RoundedCornerShape(8.dp)),
                contentScale = ContentScale.Crop
            )

            Spacer(modifier = Modifier.width(16.dp))

            Column {
                Text(
                    text = item.title,
                    style = MaterialTheme.typography.titleLarge
```

```
                    )
                    Text(
                        text = item.description,
                        style = MaterialTheme.typography.bodyMedium,
                        color = MaterialTheme.colorScheme.onSurfaceVariant,
                        modifier = Modifier.padding(top = 4.dp)
                    )
                }
            }
        }
    }
}
```

Листинг 2 – ShipCard

Задание 2: выполнить настройку темы приложения

Использовать Material Theme Builder для создания пользовательской цветовой палитры. Экспортировать сгенерированную тему и заменить ею стандартные файлы Color.kt, Theme.kt и Type.kt в проекте.

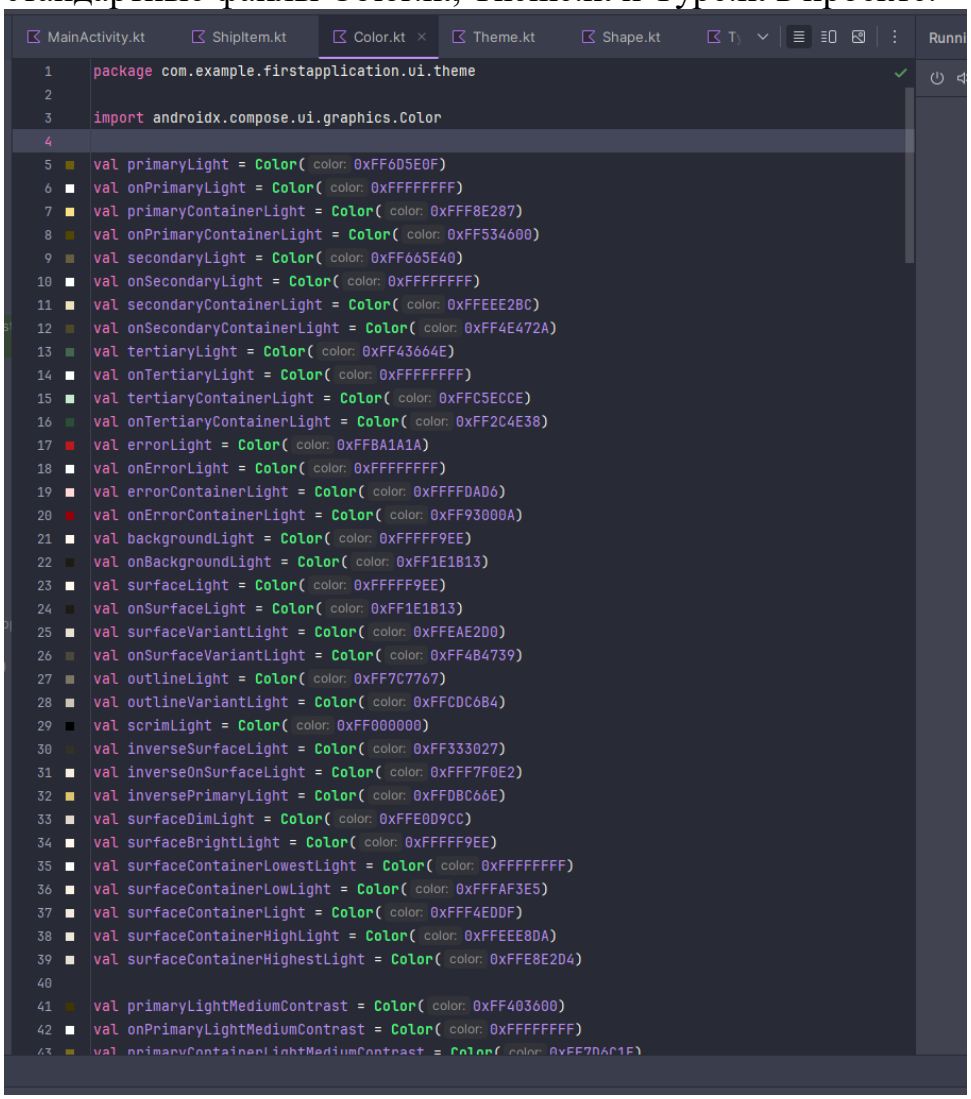


Рисунок 6 – Заменённый Color.kt

Определить пользовательскую типографику (Typography) в файле Type.kt, используя загруженные шрифты.

```
package com.example.firstapplication.ui.theme

import android.app.Activity
```

```
import android.os.Build
import androidx.compose.foundation.isSystemInDarkTheme
import androidx.compose.material3.MaterialTheme
import androidx.compose.material3.lightColorScheme
import androidx.compose.material3.darkColorScheme
import androidx.compose.material3.dynamicDarkColorScheme
import androidx.compose.material3.dynamicLightColorScheme
import androidx.compose.material3.Typography
import androidx.compose.runtime.Composable
import androidx.compose.runtime.Immutable
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.graphics.toArgb
import androidx.compose.ui.platform.LocalContext

private val lightScheme = lightColorScheme(
    primary = primaryLight,
    onPrimary = onPrimaryLight,
    primaryContainer = primaryContainerLight,
    onPrimaryContainer = onPrimaryContainerLight,
    secondary = secondaryLight,
    onSecondary = onSecondaryLight,
    secondaryContainer = secondaryContainerLight,
    onSecondaryContainer = onSecondaryContainerLight,
    tertiary = tertiaryLight,
    onTertiary = onTertiaryLight,
    tertiaryContainer = tertiaryContainerLight,
    onTertiaryContainer = onTertiaryContainerLight,
    error = errorLight,
    onError = onErrorLight,
    errorContainer = errorContainerLight,
    onErrorContainer = onErrorContainerLight,
    background = backgroundLight,
    onBackground = onBackgroundLight,
    surface = surfaceLight,
    onSurface = onSurfaceLight,
    surfaceVariant = surfaceVariantLight,
    onSurfaceVariant = onSurfaceVariantLight,
    outline = outlineLight,
    outlineVariant = outlineVariantLight,
    scrim = scrimLight,
    inverseSurface = inverseSurfaceLight,
    inverseOnSurface = inverseOnSurfaceLight,
    inversePrimary = inversePrimaryLight,
    surfaceDim = surfaceDimLight,
    surfaceBright = surfaceBrightLight,
    surfaceContainerLowest = surfaceContainerLowestLight,
    surfaceContainerLow = surfaceContainerLowLight,
    surfaceContainer = surfaceContainerLight,
    surfaceContainerHigh = surfaceContainerHighLight,
    surfaceContainerHighest = surfaceContainerHighestLight,
)

private val darkScheme = darkColorScheme(
    primary = primaryDark,
    onPrimary = onPrimaryDark,
    primaryContainer = primaryContainerDark,
    onPrimaryContainer = onPrimaryContainerDark,
    secondary = secondaryDark,
    onSecondary = onSecondaryDark,
    secondaryContainer = secondaryContainerDark,
    onSecondaryContainer = onSecondaryContainerDark,
    tertiary = tertiaryDark,
    onTertiary = onTertiaryDark,
    tertiaryContainer = tertiaryContainerDark,
    onTertiaryContainer = onTertiaryContainerDark,
    error = errorDark,
    onError = onErrorDark,
    errorContainer = errorContainerDark,
    onErrorContainer = onErrorContainerDark,
    background = backgroundDark,
    onBackground = onBackgroundDark,
    surface = surfaceDark,
    onSurface = onSurfaceDark,
    surfaceVariant = surfaceVariantDark,
    onSurfaceVariant = onSurfaceVariantDark,
    outline = outlineDark,
    outlineVariant = outlineVariantDark,
    scrim = scrimDark,
    inverseSurface = inverseSurfaceDark,
    inverseOnSurface = inverseOnSurfaceDark,
    inversePrimary = inversePrimaryDark,
    surfaceDim = surfaceDimDark,
    surfaceBright = surfaceBrightDark,
    surfaceContainerLowest = surfaceContainerLowestDark,
    surfaceContainerLow = surfaceContainerLowDark,
    surfaceContainer = surfaceContainerDark,
    surfaceContainerHigh = surfaceContainerHighDark,
    surfaceContainerHighest = surfaceContainerHighestDark,
)

private val mediumContrastLightColorScheme = lightColorScheme(
    primary = primaryLightMediumContrast,
    onPrimary = onPrimaryLightMediumContrast,
    primaryContainer = primaryContainerLightMediumContrast,
    onPrimaryContainer = onPrimaryContainerLightMediumContrast,
    secondary = secondaryLightMediumContrast,
    onSecondary = onSecondaryLightMediumContrast,
    secondaryContainer = secondaryContainerLightMediumContrast,
    onSecondaryContainer = onSecondaryContainerLightMediumContrast,
    tertiary = tertiaryLightMediumContrast,
    onTertiary = onTertiaryLightMediumContrast,
    tertiaryContainer = tertiaryContainerLightMediumContrast,
    onTertiaryContainer = onTertiaryContainerLightMediumContrast,
    error = errorLightMediumContrast,
    onError = onErrorLightMediumContrast,
    errorContainer = errorContainerLightMediumContrast,
    onErrorContainer = onErrorContainerLightMediumContrast,
```

```
    background = backgroundLightMediumContrast,
    onBackground = onBackgroundLightMediumContrast,
    surface = surfaceLightMediumContrast,
    onSurface = onSurfaceLightMediumContrast,
    surfaceVariant = surfaceVariantLightMediumContrast,
    onSurfaceVariant = onSurfaceVariantLightMediumContrast,
    outline = outlineLightMediumContrast,
    outlineVariant = outlineVariantLightMediumContrast,
    scrim = scrimLightMediumContrast,
    inverseSurface = inverseSurfaceLightMediumContrast,
    inverseOnSurface = inverseOnSurfaceLightMediumContrast,
    inversePrimary = inversePrimaryLightMediumContrast,
    surfaceDim = surfaceDimLightMediumContrast,
    surfaceBright = surfaceBrightLightMediumContrast,
    surfaceContainerLowest = surfaceContainerLowestLightMediumContrast,
    surfaceContainerLow = surfaceContainerLowLightMediumContrast,
    surfaceContainer = surfaceContainerLightMediumContrast,
    surfaceContainerHigh = surfaceContainerHighLightMediumContrast,
    surfaceContainerHighest = surfaceContainerHighestLightMediumContrast,
)

private val highContrastLightColorScheme = lightColorScheme(
    primary = primaryLightHighContrast,
    onPrimary = onPrimaryLightHighContrast,
    primaryContainer = primaryContainerLightHighContrast,
    onPrimaryContainer = onPrimaryContainerLightHighContrast,
    secondary = secondaryLightHighContrast,
    onSecondary = onSecondaryLightHighContrast,
    secondaryContainer = secondaryContainerLightHighContrast,
    onSecondaryContainer = onSecondaryContainerLightHighContrast,
    tertiary = tertiaryLightHighContrast,
    onTertiary = onTertiaryLightHighContrast,
    tertiaryContainer = tertiaryContainerLightHighContrast,
    onTertiaryContainer = onTertiaryContainerLightHighContrast,
    error = errorLightHighContrast,
    onError = onErrorLightHighContrast,
    errorContainer = errorContainerLightHighContrast,
    onErrorContainer = onErrorContainerLightHighContrast,
    background = backgroundLightHighContrast,
    onBackground = onBackgroundLightHighContrast,
    surface = surfaceLightHighContrast,
    onSurface = onSurfaceLightHighContrast,
    surfaceVariant = surfaceVariantLightHighContrast,
    onSurfaceVariant = onSurfaceVariantLightHighContrast,
    outline = outlineLightHighContrast,
    outlineVariant = outlineVariantLightHighContrast,
    scrim = scrimLightHighContrast,
    inverseSurface = inverseSurfaceLightHighContrast,
    inverseOnSurface = inverseOnSurfaceLightHighContrast,
    inversePrimary = inversePrimaryLightHighContrast,
    surfaceDim = surfaceDimLightHighContrast,
    surfaceBright = surfaceBrightLightHighContrast,
    surfaceContainerLowest = surfaceContainerLowestLightHighContrast,
    surfaceContainerLow = surfaceContainerLowLightHighContrast,
    surfaceContainer = surfaceContainerLightHighContrast,
    surfaceContainerHigh = surfaceContainerHighLightHighContrast,
    surfaceContainerHighest = surfaceContainerHighestLightHighContrast,
)

private val mediumContrastDarkColorScheme = darkColorScheme(
    primary = primaryDarkMediumContrast,
    onPrimary = onPrimaryDarkMediumContrast,
    primaryContainer = primaryContainerDarkMediumContrast,
    onPrimaryContainer = onPrimaryContainerDarkMediumContrast,
    secondary = secondaryDarkMediumContrast,
    onSecondary = onSecondaryDarkMediumContrast,
    secondaryContainer = secondaryContainerDarkMediumContrast,
    onSecondaryContainer = onSecondaryContainerDarkMediumContrast,
    tertiary = tertiaryDarkMediumContrast,
    onTertiary = onTertiaryDarkMediumContrast,
    tertiaryContainer = tertiaryContainerDarkMediumContrast,
    onTertiaryContainer = onTertiaryContainerDarkMediumContrast,
    error = errorDarkMediumContrast,
    onError = onErrorDarkMediumContrast,
    errorContainer = errorContainerDarkMediumContrast,
    onErrorContainer = onErrorContainerDarkMediumContrast,
    background = backgroundDarkMediumContrast,
    onBackground = onBackgroundDarkMediumContrast,
    surface = surfaceDarkMediumContrast,
    onSurface = onSurfaceDarkMediumContrast,
    surfaceVariant = surfaceVariantDarkMediumContrast,
    onSurfaceVariant = onSurfaceVariantDarkMediumContrast,
    outline = outlineDarkMediumContrast,
    outlineVariant = outlineVariantDarkMediumContrast,
    scrim = scrimDarkMediumContrast,
    inverseSurface = inverseSurfaceDarkMediumContrast,
    inverseOnSurface = inverseOnSurfaceDarkMediumContrast,
    inversePrimary = inversePrimaryDarkMediumContrast,
    surfaceDim = surfaceDimDarkMediumContrast,
    surfaceBright = surfaceBrightDarkMediumContrast,
    surfaceContainerLowest = surfaceContainerLowestDarkMediumContrast,
    surfaceContainerLow = surfaceContainerLowDarkMediumContrast,
    surfaceContainer = surfaceContainerDarkMediumContrast,
    surfaceContainerHigh = surfaceContainerHighDarkMediumContrast,
    surfaceContainerHighest = surfaceContainerHighestDarkMediumContrast,
)

private val highContrastDarkColorScheme = darkColorScheme(
    primary = primaryDarkHighContrast,
    onPrimary = onPrimaryDarkHighContrast,
    primaryContainer = primaryContainerDarkHighContrast,
    onPrimaryContainer = onPrimaryContainerDarkHighContrast,
    secondary = secondaryDarkHighContrast,
    onSecondary = onSecondaryDarkHighContrast,
    secondaryContainer = secondaryContainerDarkHighContrast,
    onSecondaryContainer = onSecondaryContainerDarkHighContrast,
    tertiary = tertiaryDarkHighContrast,
```

```
    onTertiary = onTertiaryDarkHighContrast,
    tertiaryContainer = tertiaryContainerDarkHighContrast,
    onTertiaryContainer = onTertiaryContainerDarkHighContrast,
    error = errorDarkHighContrast,
    onError = onErrorDarkHighContrast,
    errorContainer = errorContainerDarkHighContrast,
    onErrorContainer = onErrorContainerDarkHighContrast,
    background = backgroundDarkHighContrast,
    onBackground = onBackgroundDarkHighContrast,
    surface = surfaceDarkHighContrast,
    onSurface = onSurfaceDarkHighContrast,
    surfaceVariant = surfaceVariantDarkHighContrast,
    onSurfaceVariant = onSurfaceVariantDarkHighContrast,
    outline = outlineDarkHighContrast,
    outlineVariant = outlineVariantDarkHighContrast,
    scrim = scrimDarkHighContrast,
    inverseSurface = inverseSurfaceDarkHighContrast,
    inverseOnSurface = inverseOnSurfaceDarkHighContrast,
    inversePrimary = inversePrimaryDarkHighContrast,
    surfaceDim = surfaceDimDarkHighContrast,
    surfaceBright = surfaceBrightDarkHighContrast,
    surfaceContainerLowest = surfaceContainerLowestDarkHighContrast,
    surfaceContainerLow = surfaceContainerLowDarkHighContrast,
    surfaceContainer = surfaceContainerDarkHighContrast,
    surfaceContainerHigh = surfaceContainerHighDarkHighContrast,
    surfaceContainerHighest = surfaceContainerHighestDarkHighContrast,
)

@Immutable
data class ColorFamily(
    val color: Color,
    val onColor: Color,
    val colorContainer: Color,
    val onColorContainer: Color
)

val unspecified_scheme = ColorFamily(
    Color.Unspecified, Color.Unspecified, Color.Unspecified, Color.Unspecified
)

@Composable
fun AppTheme(
    darkTheme: Boolean = isSystemInDarkTheme(),
    // Dynamic color is available on Android 12+
    dynamicColor: Boolean = true,
    content: @Composable() () -> Unit
) {
  val colorScheme = when {
      dynamicColor && Build.VERSION.SDK_INT >= Build.VERSION_CODES.S -> {
          val context = LocalContext.current
          if (darkTheme) dynamicDarkColorScheme(context) else dynamicLightColorScheme(context)
      }

      darkTheme -> darkScheme
      else -> lightScheme
  }

  MaterialTheme(
    colorScheme = colorScheme,
    typography = AppTypography,
      shapes = Shapes,
    content = content
  )
}
```

Листинг 3 - заменённого Theme.kt

```kotlin
package com.example.firstapplication.ui.theme

import androidx.compose.material3.Typography
import androidx.compose.ui.text.TextStyle
import androidx.compose.ui.text.font.Font
import androidx.compose.ui.text.font.FontFamily
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.unit.sp
import com.example.firstapplication.R

// Создаем FontFamily с локальным файлом Google Sans Flex
val GoogleSansFlex = FontFamily(
    Font(R.font.google_sans_flex, FontWeight.Normal),
    Font(R.font.ravitasneegular, FontWeight.Normal)

)

// Базовые стили Material 3
val baseline = Typography()

// Кастомная типографика с Google Sans Flex
val AppTypography = Typography(
    displayLarge = baseline.displayLarge.copy(fontFamily = GoogleSansFlex),
    displayMedium = baseline.displayMedium.copy(fontFamily = GoogleSansFlex),
    displaySmall = baseline.displaySmall.copy(fontFamily = GoogleSansFlex),
    headlineLarge = baseline.headlineLarge.copy(fontFamily = GoogleSansFlex),
    headlineMedium = baseline.headlineMedium.copy(fontFamily = GoogleSansFlex
    headlineSmall = baseline.headlineSmall.copy(fontFamily = GoogleSansFlex),
    titleLarge = baseline.titleLarge.copy(fontFamily = GoogleSansFlex),
    titleMedium = baseline.titleMedium.copy(fontFamily = GoogleSansFlex),
    titleSmall = baseline.titleSmall.copy(fontFamily = GoogleSansFlex),
    bodyLarge = baseline.bodyLarge.copy(fontFamily = GoogleSansFlex),
    bodyMedium = baseline.bodyMedium.copy(fontFamily = GoogleSansFlex),
    bodySmall = baseline.bodySmall.copy(fontFamily = GoogleSansFlex),
    labelLarge = baseline.labelLarge.copy(fontFamily = GoogleSansFlex),
    labelMedium = baseline.labelMedium.copy(fontFamily = GoogleSansFlex),
    labelSmall = baseline.labelSmall.copy(fontFamily = GoogleSansFlex),
)
```

Рисунок 7 – Заменённый Type.kt

Загрузить любое семейство шрифтов с Google Fonts и добавить файлы шрифтов в каталог res/font.

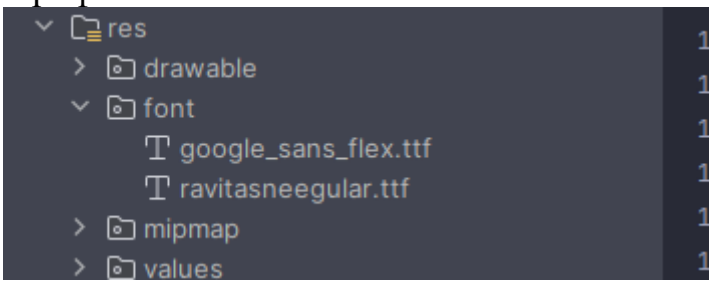


Рисунок 8 –Загруженное семейство шрифтов

Определить пользовательские формы для компонентов, создав файл Shape.kt и изменив в нем стандартные значения small, medium и large.

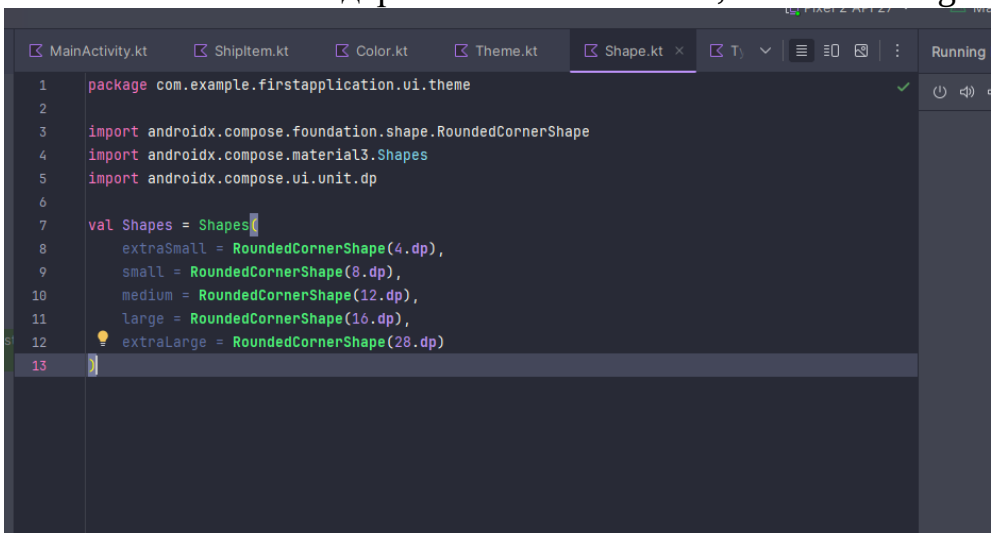


Рисунок 9 – Shape.kt

Часть V – Практическое задание

Задание 1: реализовать логирование жизненного цикла Activity Добавить логирование в MainActivity для отслеживания состояний её жизненного цикла. Переопределить методы onCreate(), onStart(), onResume(), onPause(), onStop() и onDestroy(). В каждом из переопределенных методов добавить вызов Log.d(), который будет записывать в лог имя вызванного метода. Использовать единый TAG для всех сообщений. Запустить приложение и изучить вывод в окне Logcat при сворачивании и разворачивании приложения, а также при изменении ориентации устройства.

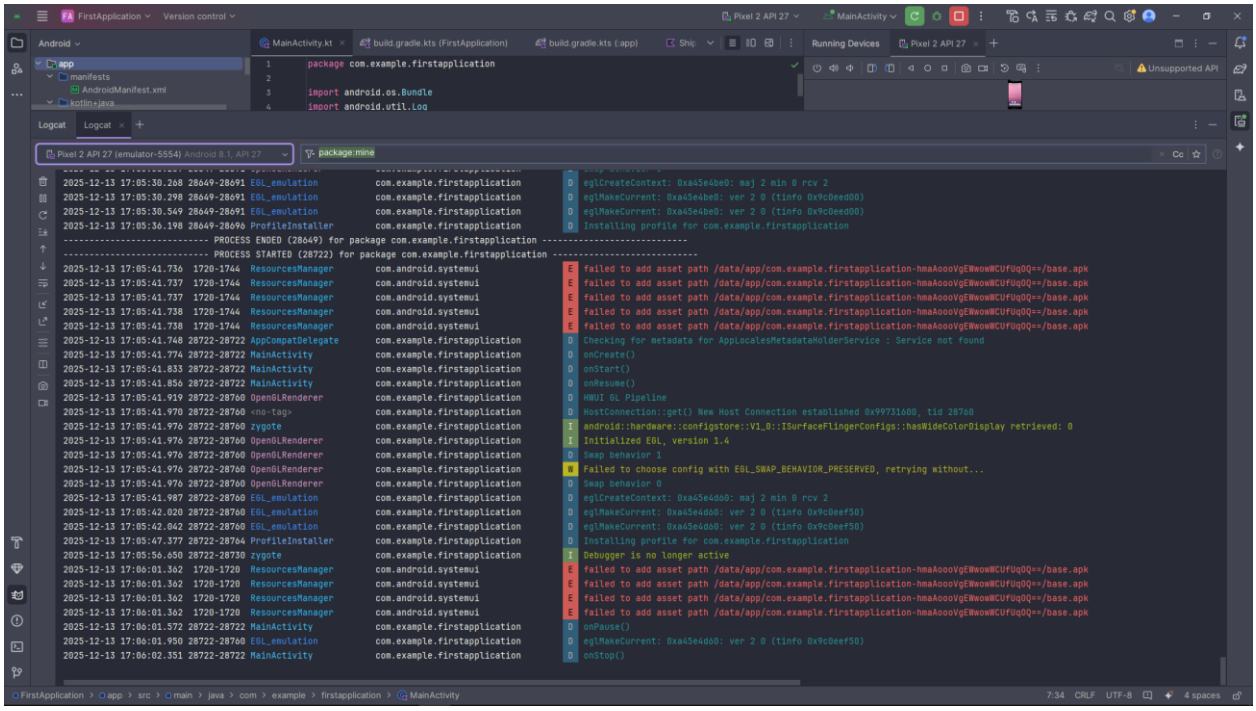


Рисунок 10 – MainActivity

```kotlin
package com.example.firstapplication

import android.os.Bundle
import android.util.Log
import androidx.appcompat.app.AppCompatActivity

class MainActivity : AppCompatActivity() {

    companion object {
        private const val TAG = "MainActivity"
    }

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        Log.d(TAG, "onCreate()")
        setContentView(R.layout.activity_main)
    }

    override fun onStart() {
        super.onStart()
        Log.d(TAG, "onStart()")
    }

    override fun onResume() {
        super.onResume()
        Log.d(TAG, "onResume()")
    }

    override fun onPause() {
        super.onPause()
        Log.d(TAG, "onPause()")
    }

    override fun onStop() {
        super.onStop()
        Log.d(TAG, "onStop()")
    }

    override fun onDestroy() {
        super.onDestroy()
        Log.d(TAG, "onDestroy()")
    }
}
```

Листинг 4 – MainActivity

Задание 2: сохранить состояние UI с помощью ViewModel Разработать экран с простым счетчиком, который сохраняет свое состояние при изменениях конфигурации (например, при повороте экрана). Создать класс, наследуемый от ViewModel, который будет содержать изменяемую переменную состояния для хранения значения счетчика. На экране UI отобразить значение счетчика, получая его из ViewModel. Добавить кнопку, которая при нажатии будет вызывать функцию в ViewModel для увеличения значения счетчика. ViewModel должен обеспечивать сохранение состояния счетчика, чтобы его значение не сбрасывалось при пересоздании Activity.

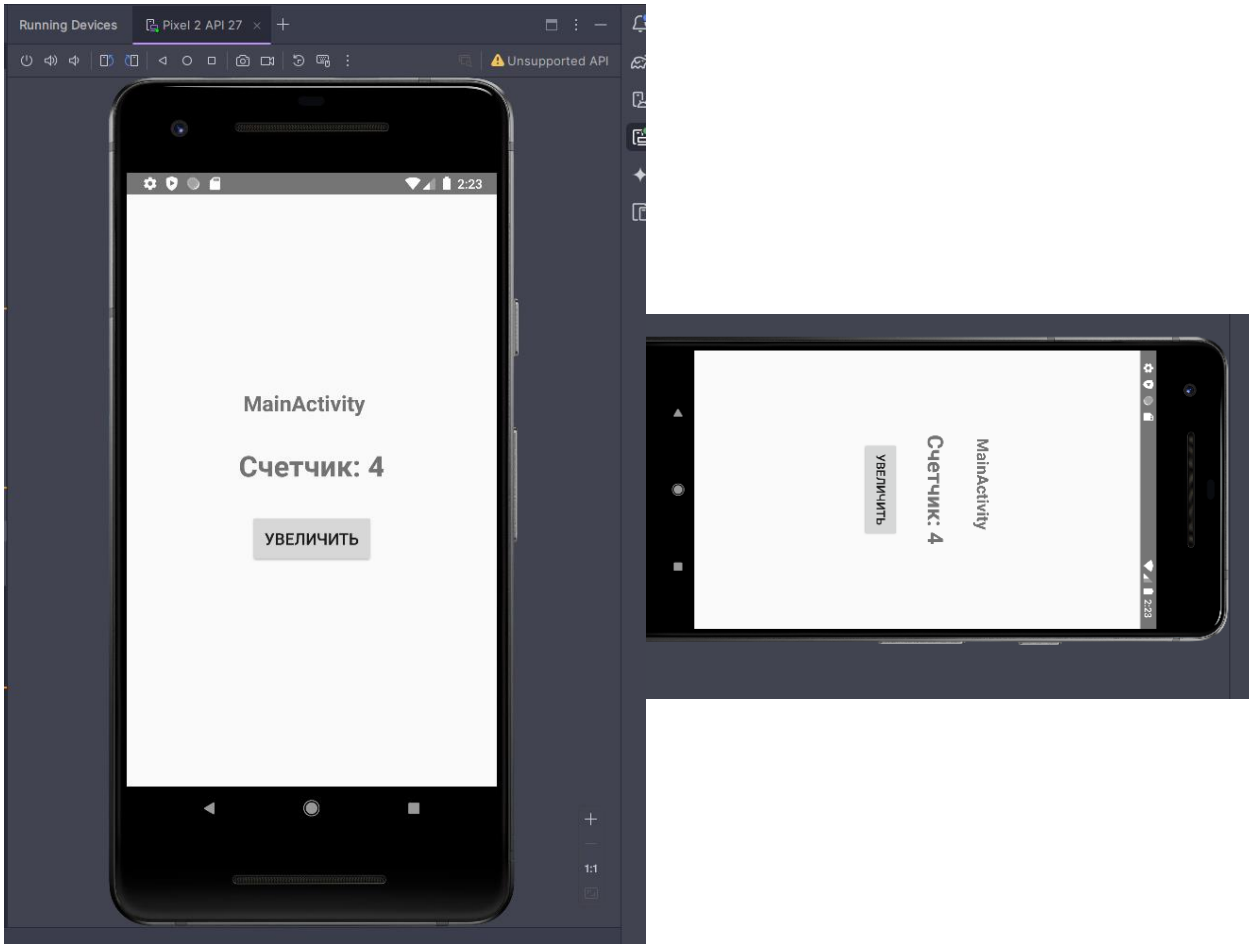


Рисунок 11 – MainActivity

```
package com.example.firstapplication

import android.os.Bundle
import android.util.Log
import android.widget.Button
import android.widget.TextView
import androidx.appcompat.app.AppCompatActivity
import androidx.lifecycle.ViewModelProvider

class MainActivity : AppCompatActivity() {

    companion object {
        private const val TAG = "MainActivity"
    }

    private lateinit var viewModel: CounterViewModel
    private lateinit var counterTextView: TextView
    private lateinit var incrementButton: Button

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        Log.d(TAG, "onCreate()")
        setContentView(R.layout.activity_main)

        // Инициализация ViewModel
        viewModel = ViewModelProvider(this)[CounterViewModel::class.java]

        // Инициализация UI элементов
```

```
        counterTextView = findViewById(R.id.counterTextView)
        incrementButton = findViewById(R.id.incrementButton)

        // Обновление UI при создании
        updateCounterDisplay()

        // Обработчик нажатия кнопки
        incrementButton.setOnClickListener {
            viewModel.incrementCounter()
            updateCounterDisplay()
        }
    }

    private fun updateCounterDisplay() {
        counterTextView.text = "Счетчик: ${viewModel.counter}"
    }

    override fun onStart() {
        super.onStart()
        Log.d(TAG, "onStart()")
    }

    override fun onResume() {
        super.onResume()
        Log.d(TAG, "onResume()")
    }

    override fun onPause() {
        super.onPause()
        Log.d(TAG, "onPause()")
    }

    override fun onStop() {
        super.onStop()
        Log.d(TAG, "onStop()")
    }

    override fun onDestroy() {
        super.onDestroy()
        Log.d(TAG, "onDestroy()")
    }
}
```

Листинг 5 – MainActivity

Задание 3: реализовать навигацию между экранами Создать приложение с двумя экранами и реализовать навигацию между ними. 28 Добавить в проект зависимость navigation-compose. Создать NavHost в основной Composable-функции. Определить два экрана ("Home" и "Details") с помощью функции composable(). На экране "Home" разместить кнопку, которая по нажатию будет осуществлять переход на экран "Details" с помощью navController.navigate(). На экране "Details" отобразить любой информационный текст

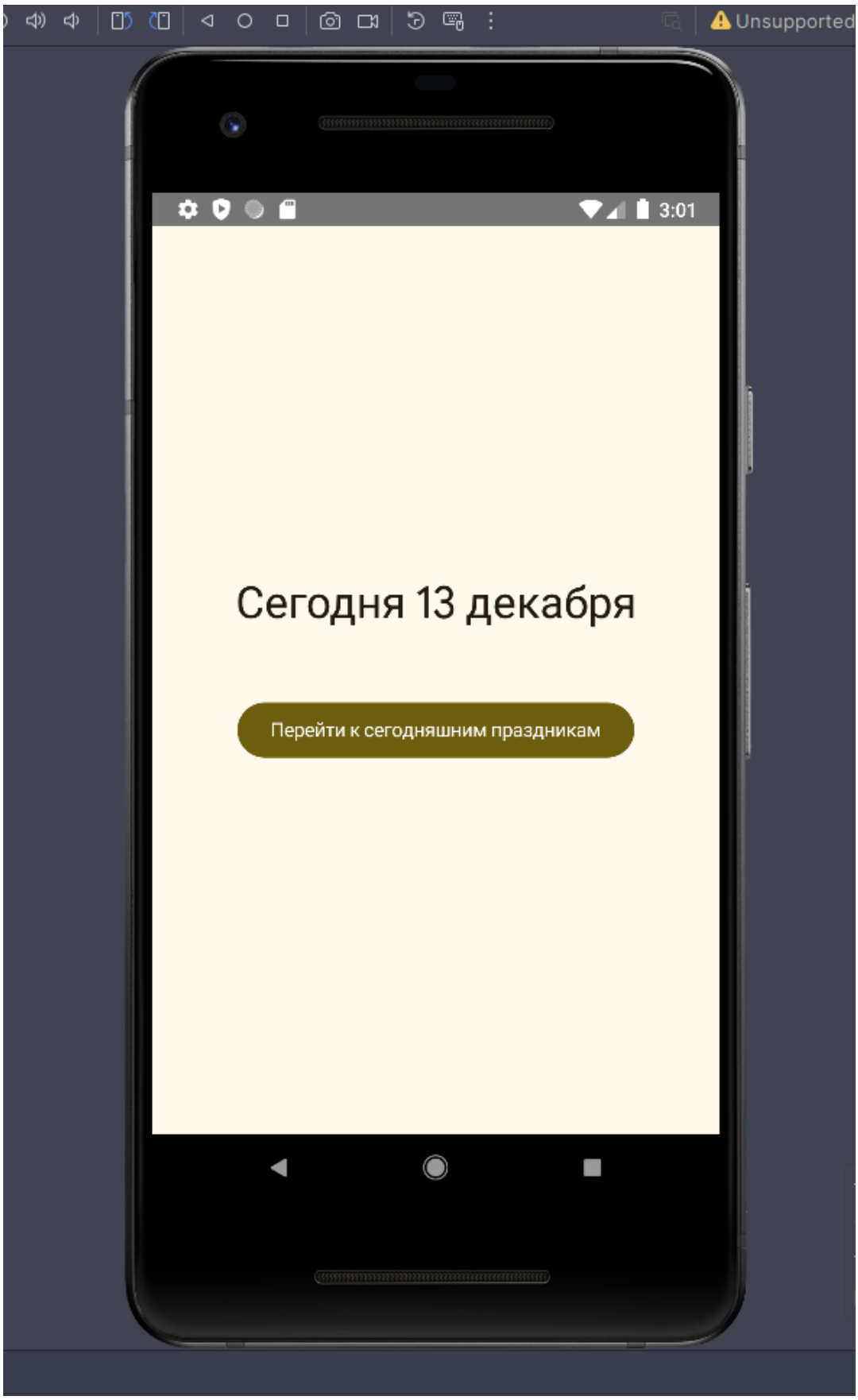


Рисунок 12 – Экран Home

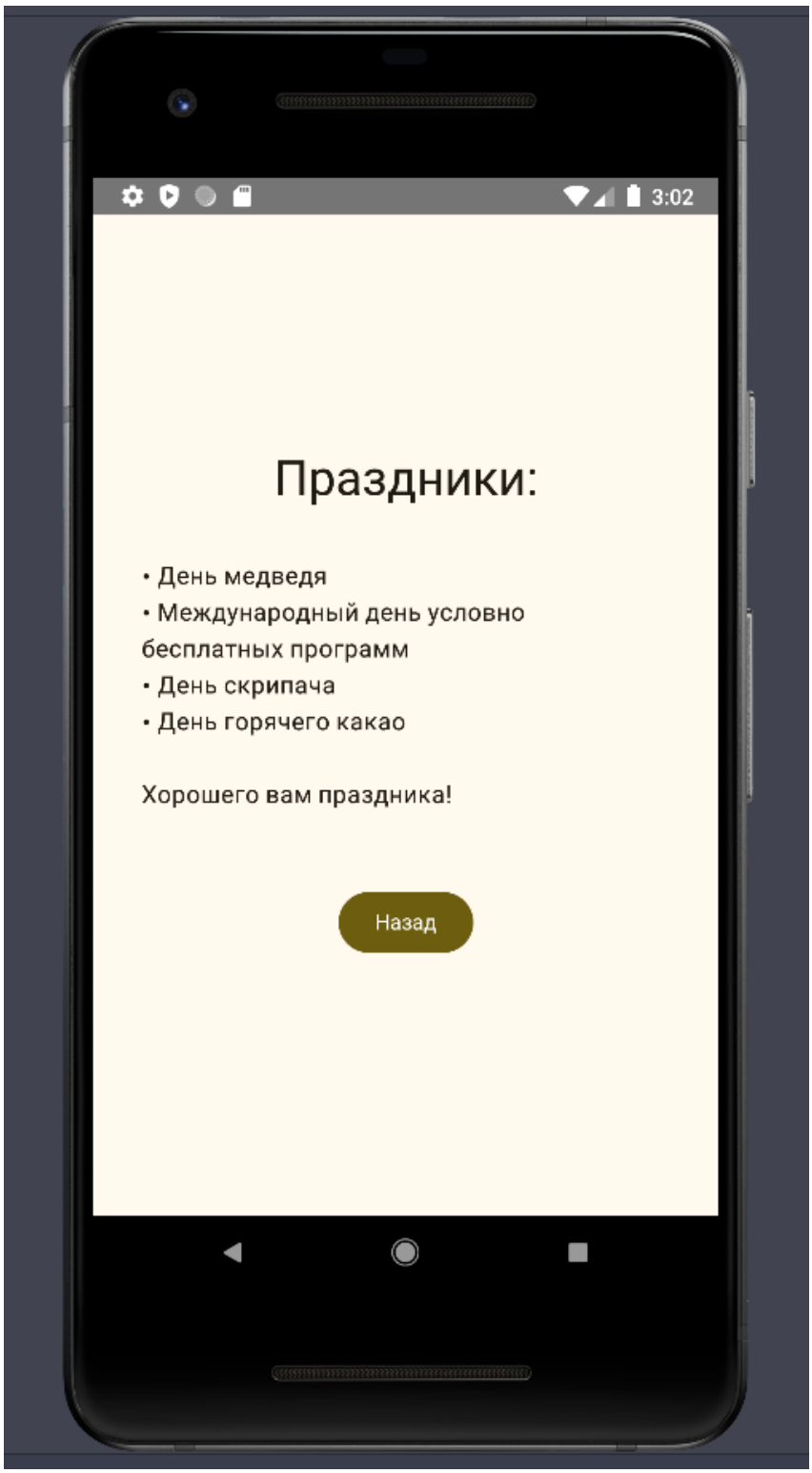


Рисунок 13 – Экран Details

```
package com.example.firstapplication

import android.os.Bundle
import android.util.Log
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.layout.*
import androidx.compose.material3.*
import androidx.compose.runtime.Composable
```

```
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.unit.dp
import androidx.navigation.NavHostController
import androidx.navigation.compose.NavHost
import androidx.navigation.compose.composable
import androidx.navigation.compose.rememberNavController
import com.example.firstapplication.ui.theme.AppTheme

class MainActivity : ComponentActivity() {

    companion object {
        private const val TAG = "MainActivity"
    }

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        Log.d(TAG, "onCreate()")
        setContent {
            AppTheme {
                NavigationApp()
            }
        }
    }

    override fun onStart() {
        super.onStart()
        Log.d(TAG, "onStart()")
    }

    override fun onResume() {
        super.onResume()
        Log.d(TAG, "onResume()")
    }

    override fun onPause() {
        super.onPause()
        Log.d(TAG, "onPause()")
    }

    override fun onStop() {
        super.onStop()
        Log.d(TAG, "onStop()")
    }

    override fun onDestroy() {
        super.onDestroy()
        Log.d(TAG, "onDestroy()")
    }
}

@Composable
fun NavigationApp() {
    val navController = rememberNavController()

    NavHost(
        navController = navController,
        startDestination = "home"
    ) {
        composable("home") {
            HomeScreen(navController = navController)
        }
        composable("details") {
            DetailsScreen(navController = navController)
```

```
        }
    }
}

@Composable
fun HomeScreen(navController: NavHostController) {
    Surface(
        modifier = Modifier.fillMaxSize(),
        color = MaterialTheme.colorScheme.background
    ) {
        Column(
            modifier = Modifier
                .fillMaxSize()
                .padding(16.dp),
            horizontalAlignment = Alignment.CenterHorizontally,
            verticalArrangement = Arrangement.Center
        ) {
            Text(
                text = "Сегодня 13 декабря",

                style = MaterialTheme.typography.headlineLarge
            )

            Spacer(modifier = Modifier.height(32.dp))

            Button(
                onClick = {
                    navController.navigate("details")
                },
                modifier = Modifier.padding(16.dp)
            ) {
                Text("Перейти к сегодняшним праздникам")
            }
        }
    }
}

@Composable
fun DetailsScreen(navController: NavHostController? = null) {
    Surface(
        modifier = Modifier.fillMaxSize(),
        color = MaterialTheme.colorScheme.background
    ) {
        Column(
            modifier = Modifier
                .fillMaxSize()
                .padding(16.dp),
            horizontalAlignment = Alignment.CenterHorizontally,
            verticalArrangement = Arrangement.Center
        ) {
            Text(
                text = "Праздники:",
                style = MaterialTheme.typography.headlineLarge
            )

            Spacer(modifier = Modifier.height(32.dp))

            Text(
                text =  "• День медведя\n" +
                        "• Международный день условно бесплатных программ\n"
+
                        "• День скрипача\n" +
                        "• День горячего какао\n\n" +
                        "Хорошего вам праздника!",
```

```
                style = MaterialTheme.typography.bodyLarge,
                modifier = Modifier.padding(horizontal = 16.dp)
            )

            navController?.let {
                Spacer(modifier = Modifier.height(32.dp))

                Button(
                    onClick = {
                        navController.popBackStack()
                    },
                    modifier = Modifier.padding(16.dp)
                ) {
                    Text("Назад")
                }
            }
        }
    }
}
```

Листинг 6 – MainActivity

Задание 4: адаптировать пользовательский интерфейс под разные размеры экрана Добавить в проект зависимость material3-window-size-class. В основной Activity использовать функцию calculateWindowSizeClass для определения класса размера окна устройства. Создать Composable-функцию, которая будет принимать WindowWidthSizeClass в качестве параметра. Внутри этой функции использовать оператор when для изменения макета в зависимости от класса размера окна (Compact, Medium, Expanded). Для Compact ширины отображать макет в одну колонку. Для Medium или Expanded ширины отображать макет в две колонки (например, список слева и детали справа).

```
package com.example.firstapplication

import android.os.Bundle
import android.util.Log
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.layout.*
import androidx.compose.material3.*
import androidx.compose.material3.windowsizeclass.*
import androidx.compose.runtime.Composable
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.unit.dp
import androidx.navigation.NavHostController
import androidx.navigation.compose.NavHost
import androidx.navigation.compose.composable
import androidx.navigation.compose.rememberNavController
import com.example.firstapplication.ui.theme.AppTheme

class MainActivity : ComponentActivity() {

    companion object {
        private const val TAG = "MainActivity"
    }
```

```
    @OptIn(ExperimentalMaterial3WindowSizeClassApi::class)
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        Log.d(TAG, "onCreate()")
        setContent {
            AppTheme {
                val windowSizeClass = calculateWindowSizeClass(this)
                AdaptiveLayoutApp(windowSizeClass.widthSizeClass)
            }
        }
    }

    override fun onStart() {
        super.onStart()
        Log.d(TAG, "onStart()")
    }

    override fun onResume() {
        super.onResume()
        Log.d(TAG, "onResume()")
    }

    override fun onPause() {
        super.onPause()
        Log.d(TAG, "onPause()")
    }

    override fun onStop() {
        super.onStop()
        Log.d(TAG, "onStop()")
    }

    override fun onDestroy() {
        super.onDestroy()
        Log.d(TAG, "onDestroy()")
    }
}

@Composable
fun AdaptiveLayoutApp(widthSizeClass: WindowWidthSizeClass) {
    val navController = rememberNavController()

    when (widthSizeClass) {
        WindowWidthSizeClass.Compact -> {
            // Макет в одну колонку для компактных экранов
            CompactLayout(navController = navController)
        }
        WindowWidthSizeClass.Medium,
        WindowWidthSizeClass.Expanded -> {
            // Макет в две колонки для средних и больших экранов
            TwoColumnLayout(navController = navController)
        }
    }
}

@Composable
fun CompactLayout(navController: NavHostController) {
    NavHost(
        navController = navController,
        startDestination = "home"
    ) {
        composable("home") {
            HomeScreen(navController = navController)
        }
```

```
        composable("details") {
            DetailsScreen(navController = navController)
        }
    }
}

@Composable
fun TwoColumnLayout(navController: NavHostController) {
    Row(modifier = Modifier.fillMaxSize()) {
        // Левая колонка - список/главный экран
        Column(
            modifier = Modifier
                .weight(1f)
                .fillMaxHeight()
                .padding(16.dp),
            horizontalAlignment = Alignment.CenterHorizontally,
            verticalArrangement = Arrangement.Center
        ) {
            Text(
                text = "Главный экран",
                style = MaterialTheme.typography.headlineLarge
            )

            Spacer(modifier = Modifier.height(32.dp))

            Button(
                onClick = {
                    navController.navigate("details")
                },
                modifier = Modifier.padding(16.dp)
            ) {
                Text("Перейти к деталям")
            }
        }

        // Разделитель
        Divider(
            modifier = Modifier
                .fillMaxHeight()
                .width(1.dp)
        )

        // Правая колонка - детали
        Column(
            modifier = Modifier
                .weight(1f)
                .fillMaxHeight()
                .padding(16.dp),
            horizontalAlignment = Alignment.CenterHorizontally,
            verticalArrangement = Arrangement.Center
        ) {
            DetailsContent()
        }
    }
}

@Composable
fun HomeScreen(navController: NavHostController) {
    Surface(
        modifier = Modifier.fillMaxSize(),
        color = MaterialTheme.colorScheme.background
    ) {
        Column(
            modifier = Modifier
```

```
                .fillMaxSize()
                .padding(16.dp),
            horizontalAlignment = Alignment.CenterHorizontally,
            verticalArrangement = Arrangement.Center
        ) {
            Text(
                text = "Сегодня 13 декабря",
                style = MaterialTheme.typography.headlineLarge
            )

            Spacer(modifier = Modifier.height(32.dp))

            Button(
                onClick = {
                    navController.navigate("details")
                },
                modifier = Modifier.padding(16.dp)
            ) {
                Text("Перейти к сегодняшним праздникам")
            }
        }
    }
}

@Composable
fun DetailsScreen(navController: NavHostController) {
    Surface(
        modifier = Modifier.fillMaxSize(),
        color = MaterialTheme.colorScheme.background
    ) {
        Column(
            modifier = Modifier
                .fillMaxSize()
                .padding(16.dp),
            horizontalAlignment = Alignment.CenterHorizontally,
            verticalArrangement = Arrangement.Center
        ) {
            DetailsContent()

            Spacer(modifier = Modifier.height(32.dp))

            Button(
                onClick = {
                    navController.popBackStack()
                },
                modifier = Modifier.padding(16.dp)
            ) {
                Text("Назад")
            }
        }
    }
}

@Composable
fun DetailsContent() {
    Column(
        horizontalAlignment = Alignment.CenterHorizontally
    ) {
        Text(
            text = "Праздники:",
            style = MaterialTheme.typography.headlineLarge
        )

        Spacer(modifier = Modifier.height(32.dp))
```

```
        Text(
            text =  "• День медведя\n" +
                    "• Международный день условно бесплатных программ\n" +
                    "• День скрипача\n" +
                    "• День горячего какао\n\n" +
                    "Хорошего вам праздника!",
            style = MaterialTheme.typography.bodyLarge,
            modifier = Modifier.padding(horizontal = 16.dp)
        )
    }
}
```

Листинг – 7

Часть IV – Практическое задание

Задание 7

Создать простое приложение TodoList с использованием:

- Clean Architecture (3 слоя: Presentation, Domain, Data)
- Навигация между экранами
- Заглушка репозитория (чтение задач из JSON-файла)

Структура проекта (необходимо корректно расположить файлы и классы)

com.example.todolist/

```
├── data/
│   ├── local/
│   │   └── _____
│   ├── repository/
│   │   └── _____
│   └── model/
│       └── _____
│
├── domain/
│   ├── model/
│   │   └── _____
│   ├── repository/
│   │   └── _____
│   └── usecase/
│       ├── _____
│       └── _____
│
├── presentation/
│   ├── ui/
│   │   ├── screen/
│   │   │   ├── _____
│   │   │   └── _____
│   │   └── component/
│   │       └── _____
│   └── viewmodel/
```

```
│   └── _____________
│
├── navigation/
│   └── _____________
│
└── MainActivity.kt
```

Функционал приложения:

1. Экран списка задач (TodoListScreen)

o Отображает список задач

o Клик по задаче → переход на детальный экран

o Чекбокс → отметить как выполненную (локально)

2. Экран детали задачи (TodoDetailScreen)

o Показывает заголовок, описание, статус

o Кнопка "Назад"

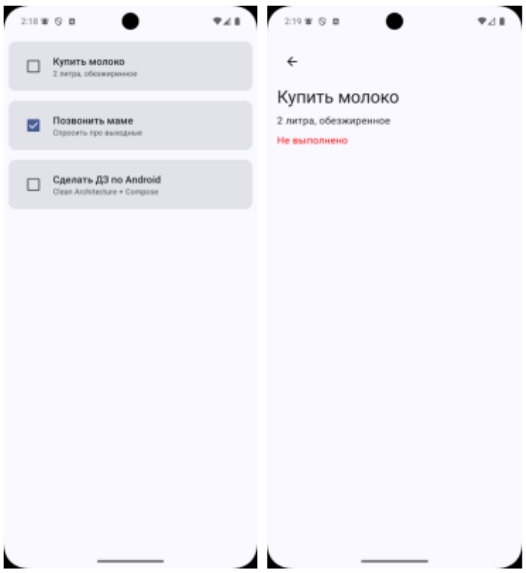
Примерная структура JSON файла:

```
[
 {
 "id": 1,
 "title": "Купить молоко",
 "description": "2 литра, обезжиренное",
 "isCompleted": false
 },
 {
 "id": 2,
 "title": "Позвонить маме",
 "description": "Спросить про выходные",
"isCompleted": true
 },
 {
 "id": 3,
 "title": "Сделать ДЗ по Android",
 "description": "Clean Architecture + Compose",
 "isCompleted": false
 }
]
```

Коды моделей:

```
data class TodoItem(
 val id: Int,
 val title: String,
 val description: String,
 val isCompleted: Boolean
)
data class TodoItemDto(
 val id: Int,
```

```
 val title: String,
 val description: String,
 val isCompleted: Boolean
)
```
Код репозитория:
```
interface TodoRepository {
 suspend fun getTodos(): List<TodoItem>
 suspend fun toggleTodo(id: Int)
}
```
Чтение данных из файла:
```
class TodoJsonDataSource(private val context: Context) {
 private val gson = Gson()
 fun getTodos(): List<TodoItemDto> {
 val json = context.assets.open("todos.json").bufferedReader().use {
it.readText() }
 val type = object : TypeToken<List<TodoItemDto>>() {}.type
 return gson.fromJson(json, type)
 }
}
```
Use cases:
```
class GetTodosUseCase(private val repository: TodoRepository) {
 suspend operator fun invoke(): List<TodoItem> = repository.getTodos()
}
class ToggleTodoUseCase(private val repository: TodoRepository) {
 // аналогично
}
```
Примерный вид двух режимов



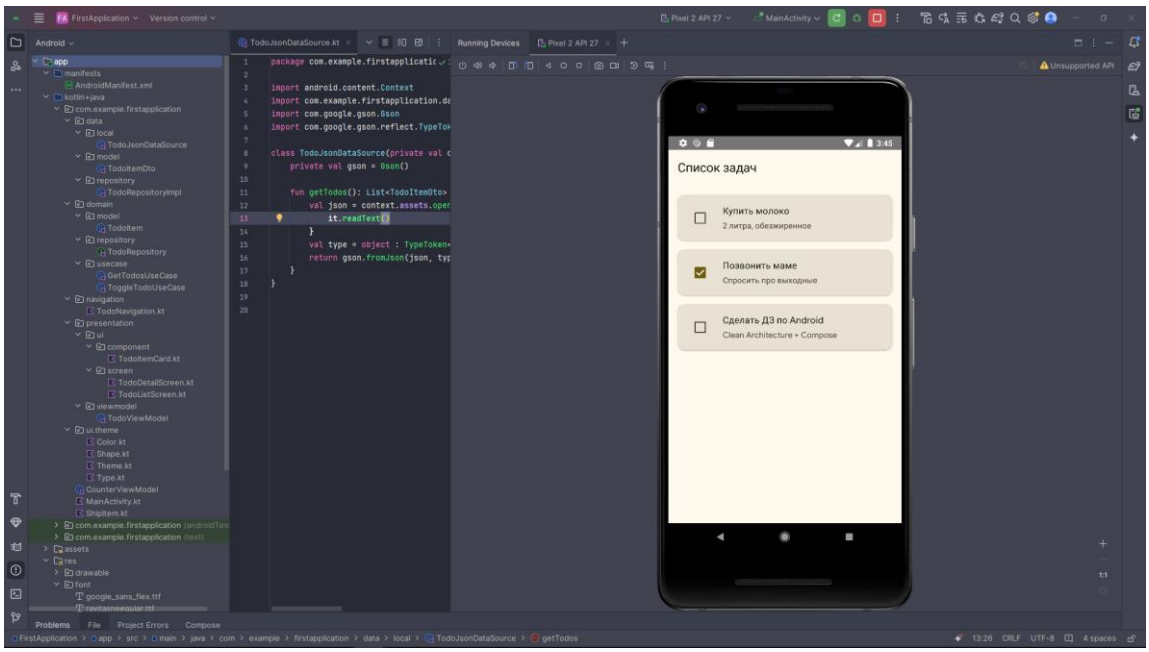


Рисунок 14 – MainActivity

```
package com.example.firstapplication.presentation.ui.screen

import androidx.compose.foundation.layout.*
import androidx.compose.material.icons.Icons
import androidx.compose.material.icons.filled.ArrowBack
import androidx.compose.material3.*
import androidx.compose.runtime.Composable
import androidx.compose.runtime.collectAsState
import androidx.compose.runtime.getValue
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.unit.dp
import com.example.firstapplication.presentation.viewmodel.TodoViewModel

@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun TodoDetailScreen(
    todoId: Int,
    viewModel: TodoViewModel,
    onBackClick: () -> Unit
) {
    val todos by viewModel.todos.collectAsState()
    val todo = todos.find { it.id == todoId }

    Scaffold(
        topBar = {
            TopAppBar(
                title = { Text(todo?.title ?: "") },
                navigationIcon = {
                    IconButton(onClick = onBackClick) {
                        Icon(Icons.Default.ArrowBack, contentDescription =
"Назад")
                    }
                }
            )
        }
    ) { paddingValues ->
        todo?.let {
            Column(
                modifier = Modifier
```

```
                    .fillMaxSize()
                    .padding(paddingValues)
                    .padding(16.dp)
            ) {
                Text(
                    text = it.title,
                    style = MaterialTheme.typography.headlineLarge,
                    modifier = Modifier.padding(bottom = 16.dp)
                )

                Spacer(modifier = Modifier.height(16.dp))

                Text(
                    text = it.description,
                    style = MaterialTheme.typography.bodyLarge,
                    modifier = Modifier.padding(bottom = 16.dp)
                )

                Spacer(modifier = Modifier.height(16.dp))

                Text(
                    text = if (it.isCompleted) "Выполнено" else "Не
выполнено",
                    style = MaterialTheme.typography.bodyLarge,
                    color = if (it.isCompleted)
                        MaterialTheme.colorScheme.primary
                    else
                        MaterialTheme.colorScheme.error
                )
            }
        } ?: run {
            Box(
                modifier = Modifier
                    .fillMaxSize()
                    .padding(paddingValues),
                contentAlignment = Alignment.Center
            ) {
                Text("Задача не найдена")
            }
        }
    }
}
```

Листинг 8 - TodoDetailScreen.kt

```
package com.example.firstapplication.presentation.ui.screen

import androidx.compose.foundation.layout.*
import androidx.compose.foundation.lazy.LazyColumn
import androidx.compose.foundation.lazy.items
import androidx.compose.material3.*
import androidx.compose.runtime.Composable
import androidx.compose.runtime.collectAsState
import androidx.compose.runtime.getValue
import androidx.compose.ui.Modifier
import androidx.compose.ui.unit.dp
import com.example.firstapplication.domain.model.TodoItem
import com.example.firstapplication.presentation.ui.component.TodoItemCard
import com.example.firstapplication.presentation.viewmodel.TodoViewModel

@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun TodoListScreen(
```

```
    viewModel: TodoViewModel,
    onItemClick: (Int) -> Unit
) {
    val todos by viewModel.todos.collectAsState()

    Scaffold(
        topBar = {
            TopAppBar(
                title = { Text("Список задач") }
            )
        }
    ) { paddingValues ->
        LazyColumn(
            modifier = Modifier
                .fillMaxSize()
                .padding(paddingValues),
            contentPadding = PaddingValues(vertical = 8.dp)
        ) {
            items(
                items = todos,
                key = { it.id }
            ) { todo ->
                TodoItemCard(
                    todo = todo,
                    onItemClick = onItemClick,
                    onCheckboxClick = { id ->
                        viewModel.toggleTodo(id)
                    }
                )
            }
        }
    }
}
```

Листинг 9 - TodoListScreen.kt

```
package com.example.firstapplication

import android.os.Bundle
import android.util.Log
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.runtime.Composable
import androidx.navigation.compose.rememberNavController
import com.example.firstapplication.data.local.TodoJsonDataSource
import com.example.firstapplication.data.repository.TodoRepositoryImpl
import com.example.firstapplication.domain.usecase.GetTodosUseCase
import com.example.firstapplication.domain.usecase.ToggleTodoUseCase
import com.example.firstapplication.navigation.TodoNavigation
import com.example.firstapplication.presentation.viewmodel.TodoViewModel
import com.example.firstapplication.ui.theme.AppTheme

class MainActivity : ComponentActivity() {

    companion object {
        private const val TAG = "MainActivity"
    }

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        Log.d(TAG, "onCreate()")

        // Инициализация зависимостей
        val dataSource = TodoJsonDataSource(this)
```

```
        val repository = TodoRepositoryImpl(dataSource)
        val getTodosUseCase = GetTodosUseCase(repository)
        val toggleTodoUseCase = ToggleTodoUseCase(repository)
        val viewModel = TodoViewModel(getTodosUseCase, toggleTodoUseCase)

        setContent {
            AppTheme {
                TodoApp(viewModel = viewModel)
            }
        }
    }

    override fun onStart() {
        super.onStart()
        Log.d(TAG, "onStart()")
    }

    override fun onResume() {
        super.onResume()
        Log.d(TAG, "onResume()")
    }

    override fun onPause() {
        super.onPause()
        Log.d(TAG, "onPause()")
    }

    override fun onStop() {
        super.onStop()
        Log.d(TAG, "onStop()")
    }

    override fun onDestroy() {
        super.onDestroy()
        Log.d(TAG, "onDestroy()")
    }
}

@Composable
fun TodoApp(viewModel: TodoViewModel) {
    val navController = rememberNavController()
    TodoNavigation(navController = navController, viewModel = viewModel)
}
```

Листинг 10- MainActivity.kt

Задание 8

1) Написать unit-тест: GetTodosUseCase возвращает 3 задачи
2) Написать unit-тест: toggleTodo меняет isCompleted
3) Написать UI-тест: отображаются все 3 задачи из JSON
4) Написать UI-тест: чекбокс переключает статус
5) Написать UI-тест: навигация List → Detail → List

1) Написать unit-тест: GetTodosUseCase возвращает 3 задачи

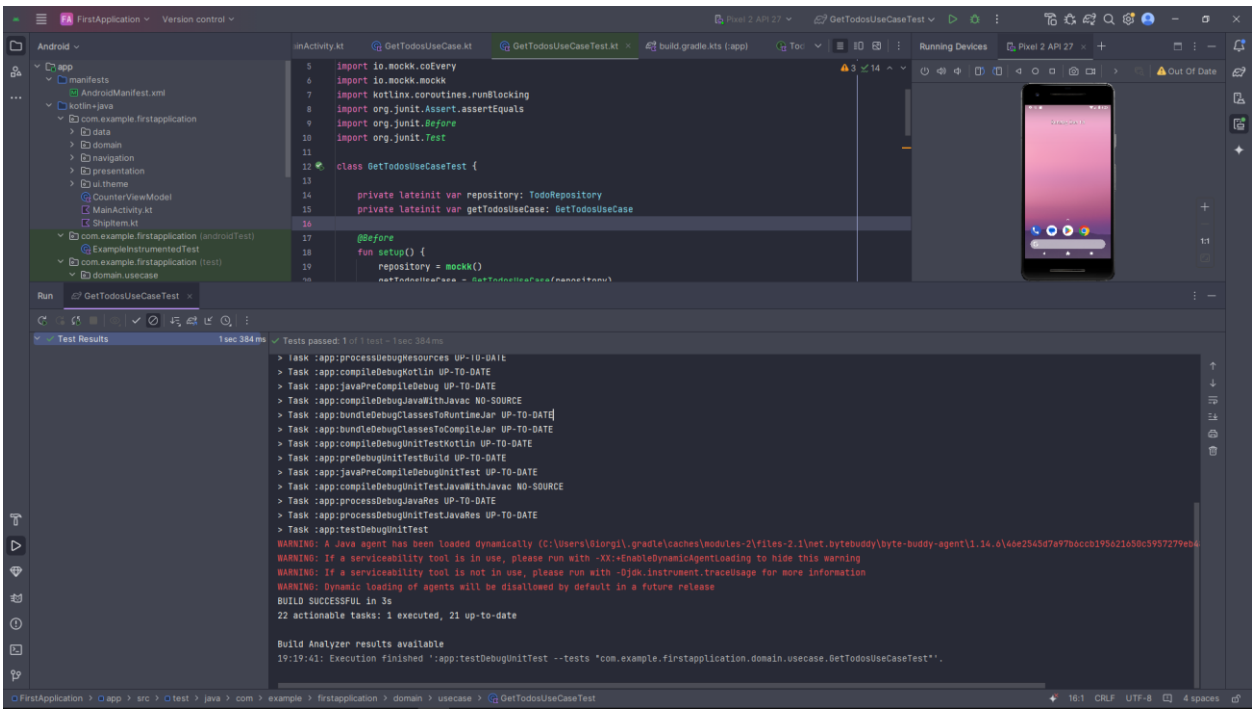


Рисунок 15 - тест 1

```
package com.example.firstapplication.domain.usecase

import com.example.firstapplication.domain.model.TodoItem
import com.example.firstapplication.domain.repository.TodoRepository
import io.mockk.coEvery
import io.mockk.mockk
import kotlinx.coroutines.runBlocking
import org.junit.Assert.assertEquals
import org.junit.Before
import org.junit.Test

class GetTodosUseCaseTest {

    private lateinit var repository: TodoRepository
    private lateinit var getTodosUseCase: GetTodosUseCase

    @Before
    fun setup() {
        repository = mockk()
        getTodosUseCase = GetTodosUseCase(repository)
    }

    @Test
    fun `GetTodosUseCase возвращает 3 задачи`() = runBlocking {
        // Arrange
        val expectedTodos = listOf(
            TodoItem(
                id = 1,
                title = "Купить молоко",
                description = "2 литра, обезжиренное",
                isCompleted = false
            ),
            TodoItem(
                id = 2,
                title = "Позвонить маме",
                description = "Спросить про выходные",
                isCompleted = true
            ),
```

```
                TodoItem(
                    id = 3,
                    title = "Сделать ДЗ по Android",
                    description = "Clean Architecture + Compose",
                    isCompleted = false
                )
            )

            coEvery { repository.getTodos() } returns expectedTodos

            // Act
            val result = getTodosUseCase()

            // Assert
            assertEquals(3, result.size)
            assertEquals(expectedTodos, result)
            assertEquals("Купить молоко", result[0].title)
            assertEquals("Позвонить маме", result[1].title)
            assertEquals("Сделать ДЗ по Android", result[2].title)
        }
    }
```

Листинг 11 – Теста 1

2) Написать unit-тест: toggleTodo меняет isCompleted

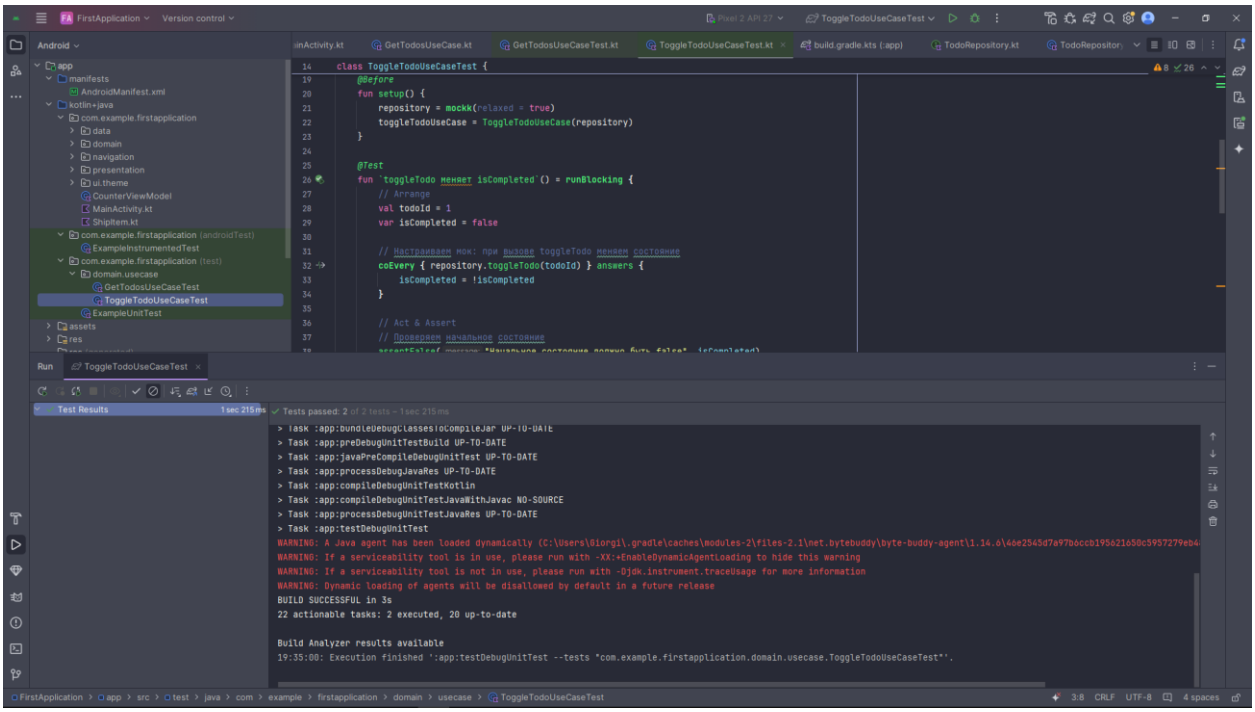


Рисунок 12 - тест 2

```
package com.example.firstapplication.domain.usecase

import com.example.firstapplication.domain.model.TodoItem
import com.example.firstapplication.domain.repository.TodoRepository
import io.mockk.coEvery
import io.mockk.coVerify
import io.mockk.mockk
```

```kotlin
import kotlinx.coroutines.runBlocking
import org.junit.Assert.assertFalse
import org.junit.Assert.assertTrue
import org.junit.Before
import org.junit.Test

class ToggleTodoUseCaseTest {

    private lateinit var repository: TodoRepository
    private lateinit var toggleTodoUseCase: ToggleTodoUseCase

    @Before
    fun setup() {
        repository = mockk(relaxed = true)
        toggleTodoUseCase = ToggleTodoUseCase(repository)
    }

    @Test
    fun `toggleTodo меняет isCompleted`() = runBlocking {
        // Arrange
        val todoId = 1
        var isCompleted = false

        // Настраиваем мок: при вызове toggleTodo меняем состояние
        coEvery { repository.toggleTodo(todoId) } answers {
            isCompleted = !isCompleted
        }

        // Act & Assert
        // Проверяем начальное состояние
        assertFalse("Начальное состояние должно быть false", isCompleted)

        // Вызываем toggleTodo
        toggleTodoUseCase(todoId)

        // Проверяем, что метод был вызван
        coVerify(exactly = 1) { repository.toggleTodo(todoId) }

        // Проверяем, что isCompleted изменился
        assertTrue("isCompleted должен измениться на true", isCompleted)

        // Вызываем toggleTodo еще раз
        toggleTodoUseCase(todoId)

        // Проверяем, что isCompleted снова изменился
        assertFalse("isCompleted должен измениться обратно на false",
isCompleted)
        coVerify(exactly = 2) { repository.toggleTodo(todoId) }
    }

    @Test
    fun `toggleTodo вызывает repository toggleTodo с правильным id`() =
runBlocking {
        // Arrange
        val todoId = 2

        // Act
        toggleTodoUseCase(todoId)

        // Assert
        coVerify(exactly = 1) { repository.toggleTodo(todoId) }
    }
}
```

Листинг 8.2 – Теста 2

3) Написать UI-тест: отображаются все 3 задачи из JSON

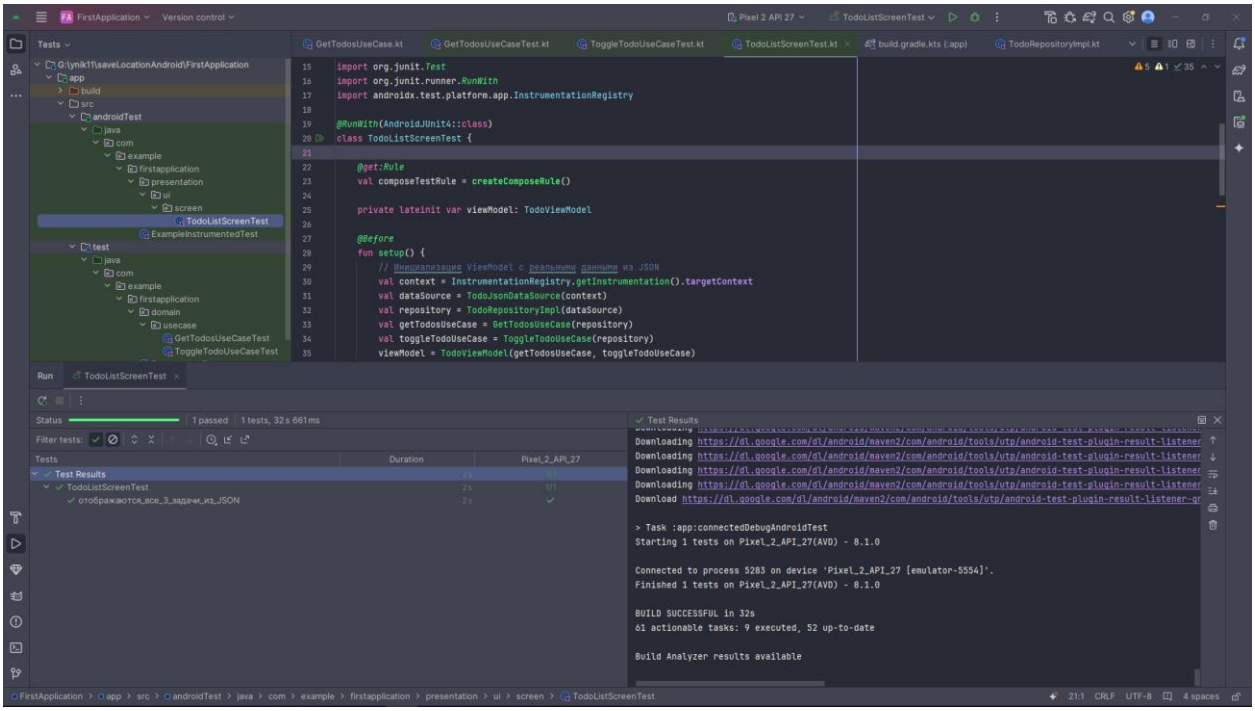


Рисунок 16- тест 3

```
package com.example.firstapplication.presentation.ui.screen

import androidx.compose.ui.test.assertIsDisplayed
import androidx.compose.ui.test.junit4.createComposeRule
import androidx.compose.ui.test.onNodeWithText
import androidx.test.ext.junit.runners.AndroidJUnit4
import com.example.firstapplication.data.local.TodoJsonDataSource
import com.example.firstapplication.data.repository.TodoRepositoryImpl
import com.example.firstapplication.domain.usecase.GetTodosUseCase
import com.example.firstapplication.domain.usecase.ToggleTodoUseCase
import com.example.firstapplication.presentation.viewmodel.TodoViewModel
import com.example.firstapplication.ui.theme.AppTheme
import org.junit.Before
import org.junit.Rule
import org.junit.Test
import org.junit.runner.RunWith
import androidx.test.platform.app.InstrumentationRegistry

@RunWith(AndroidJUnit4::class)
class TodoListScreenTest {

    @get:Rule
    val composeTestRule = createComposeRule()

    private lateinit var viewModel: TodoViewModel

    @Before
    fun setup() {
        // Инициализация ViewModel с реальными данными из JSON
        val context =
InstrumentationRegistry.getInstrumentation().targetContext
        val dataSource = TodoJsonDataSource(context)
        val repository = TodoRepositoryImpl(dataSource)
```

```
        val getTodosUseCase = GetTodosUseCase(repository)
        val toggleTodoUseCase = ToggleTodoUseCase(repository)
        viewModel = TodoViewModel(getTodosUseCase, toggleTodoUseCase)
    }

    @Test
    fun отображаются_все_3_задачи_из_JSON() {
        // Устанавливаем контент с ViewModel
        composeTestRule.setContent {
            AppTheme {
                TodoListScreen(
                    viewModel = viewModel,
                    onItemClick = {}
                )
            }
        }

        // Ждем загрузки данных из JSON и рендеринга UI
        Thread.sleep(1500)

        // Проверяем, что отображается заголовок экрана
        composeTestRule.onNodeWithText("Список задач")
            .assertIsDisplayed()

        // Проверяем, что отображаются все 3 задачи из JSON
        // Задача 1: "Купить молоко"
        composeTestRule.onNodeWithText("Купить молоко")
            .assertIsDisplayed()
        composeTestRule.onNodeWithText("2 литра, обезжиренное")
            .assertIsDisplayed()

        // Задача 2: "Позвонить маме"
        composeTestRule.onNodeWithText("Позвонить маме")
            .assertIsDisplayed()
        composeTestRule.onNodeWithText("Спросить про выходные")
            .assertIsDisplayed()

        // Задача 3: "Сделать ДЗ по Android"
        composeTestRule.onNodeWithText("Сделать ДЗ по Android")
            .assertIsDisplayed()
        composeTestRule.onNodeWithText("Clean Architecture + Compose")
            .assertIsDisplayed()
    }
}
```

Листинг 13 – Теста 3

4) Написать UI-тест: чекбокс переключает статус

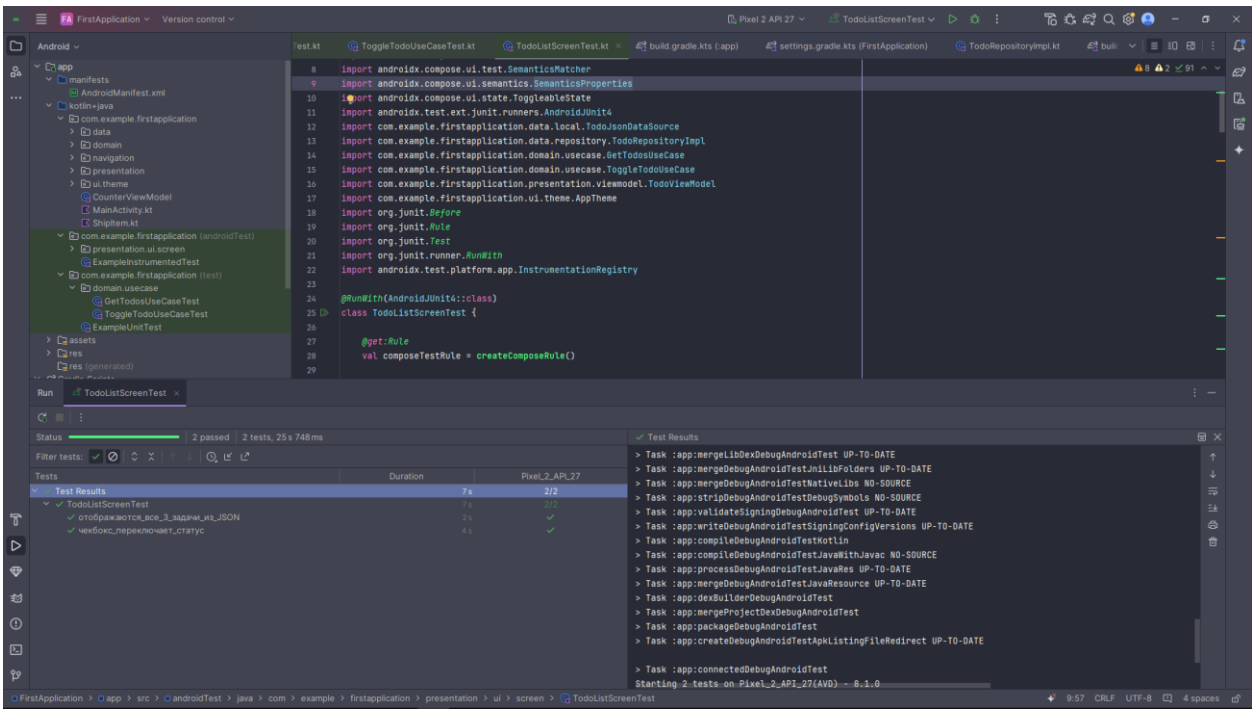


Рисунок 17- тест 4

```
package com.example.firstapplication.presentation.ui.screen

import androidx.compose.ui.test.assertIsDisplayed
import androidx.compose.ui.test.junit4.createComposeRule
import androidx.compose.ui.test.onNodeWithText
import androidx.compose.ui.test.performClick
import androidx.compose.ui.test.onFirst
import androidx.compose.ui.test.SemanticsMatcher
import androidx.compose.ui.semantics.SemanticsProperties
import androidx.compose.ui.state.ToggleableState
import androidx.test.ext.junit.runners.AndroidJUnit4
import com.example.firstapplication.data.local.TodoJsonDataSource
import com.example.firstapplication.data.repository.TodoRepositoryImpl
import com.example.firstapplication.domain.usecase.GetTodosUseCase
import com.example.firstapplication.domain.usecase.ToggleTodoUseCase
import com.example.firstapplication.presentation.viewmodel.TodoViewModel
import com.example.firstapplication.ui.theme.AppTheme
import org.junit.Before
import org.junit.Rule
import org.junit.Test
import org.junit.runner.RunWith
import androidx.test.platform.app.InstrumentationRegistry

@RunWith(AndroidJUnit4::class)
class TodoListScreenTest {

    @get:Rule
    val composeTestRule = createComposeRule()

    private lateinit var viewModel: TodoViewModel

    @Before
    fun setup() {
        // Инициализация ViewModel с реальными данными из JSON
        val context =
InstrumentationRegistry.getInstrumentation().targetContext
        val dataSource = TodoJsonDataSource(context)
        val repository = TodoRepositoryImpl(dataSource)
```

```
        val getTodosUseCase = GetTodosUseCase(repository)
        val toggleTodoUseCase = ToggleTodoUseCase(repository)
        viewModel = TodoViewModel(getTodosUseCase, toggleTodoUseCase)
    }

    @Test
    fun отображаются_все_3_задачи_из_JSON() {
        // Устанавливаем контент с ViewModel
        composeTestRule.setContent {
            AppTheme {
                TodoListScreen(
                    viewModel = viewModel,
                    onItemClick = {}
                )
            }
        }

        // Ждем загрузки данных из JSON и рендеринга UI
        Thread.sleep(1500)

        // Проверяем, что отображается заголовок экрана
        composeTestRule.onNodeWithText("Список задач")
            .assertIsDisplayed()

        // Проверяем, что отображаются все 3 задачи из JSON
        // Задача 1: "Купить молоко"
        composeTestRule.onNodeWithText("Купить молоко")
            .assertIsDisplayed()
        composeTestRule.onNodeWithText("2 литра, обезжиренное")
            .assertIsDisplayed()

        // Задача 2: "Позвонить маме"
        composeTestRule.onNodeWithText("Позвонить маме")
            .assertIsDisplayed()
        composeTestRule.onNodeWithText("Спросить про выходные")
            .assertIsDisplayed()

        // Задача 3: "Сделать ДЗ по Android"
        composeTestRule.onNodeWithText("Сделать ДЗ по Android")
            .assertIsDisplayed()
        composeTestRule.onNodeWithText("Clean Architecture + Compose")
            .assertIsDisplayed()
    }

    @Test
    fun чекбокс_переключает_статус() {
        // Устанавливаем контент с ViewModel
        composeTestRule.setContent {
            AppTheme {
                TodoListScreen(
                    viewModel = viewModel,
                    onItemClick = {}
                )
            }
        }

        // Ждем загрузки данных из JSON и рендеринга UI
        Thread.sleep(1500)

        // Находим задачу "Купить молоко" (начальное состояние: isCompleted
= false)
        composeTestRule.onNodeWithText("Купить молоко")
            .assertIsDisplayed()
```

```
        // Находим все чекбоксы с состоянием Off и выбираем первый (для
задачи "Купить молоко")
        val allUncheckedCheckboxes = composeTestRule.onAllNodes(
            SemanticsMatcher.expectValue(
                SemanticsProperties.ToggleableState,
                ToggleableState.Off
            )
        )

        // Выбираем первый чекбокс (для первой задачи)
        val firstCheckbox = allUncheckedCheckboxes.onFirst()
        firstCheckbox.assertIsDisplayed()

        // Кликаем на чекбокс
        firstCheckbox.performClick()

        // Ждем обновления UI после клика
        Thread.sleep(1000)

        // Проверяем, что чекбокс теперь отмечен
        // Ищем все чекбоксы с состоянием On и выбираем первый
        val allCheckedCheckboxes = composeTestRule.onAllNodes(
            SemanticsMatcher.expectValue(
                SemanticsProperties.ToggleableState,
                ToggleableState.On
            )
        )
        val checkedCheckbox = allCheckedCheckboxes.onFirst()
        checkedCheckbox.assertIsDisplayed()

        // Кликаем еще раз для переключения обратно
        checkedCheckbox.performClick()

        // Ждем обновления UI
        Thread.sleep(1000)

        // Проверяем, что чекбокс снова не отмечен
        // Ищем все чекбоксы с состоянием Off и выбираем первый
        val allUncheckedCheckboxesAfter = composeTestRule.onAllNodes(
            SemanticsMatcher.expectValue(
                SemanticsProperties.ToggleableState,
                ToggleableState.Off
            )
        )
        val uncheckedCheckbox = allUncheckedCheckboxesAfter.onFirst()
        uncheckedCheckbox.assertIsDisplayed()
    }
}
```

Листинг 14 – Теста 4

5) Написать UI-тест: навигация List → Detail → List

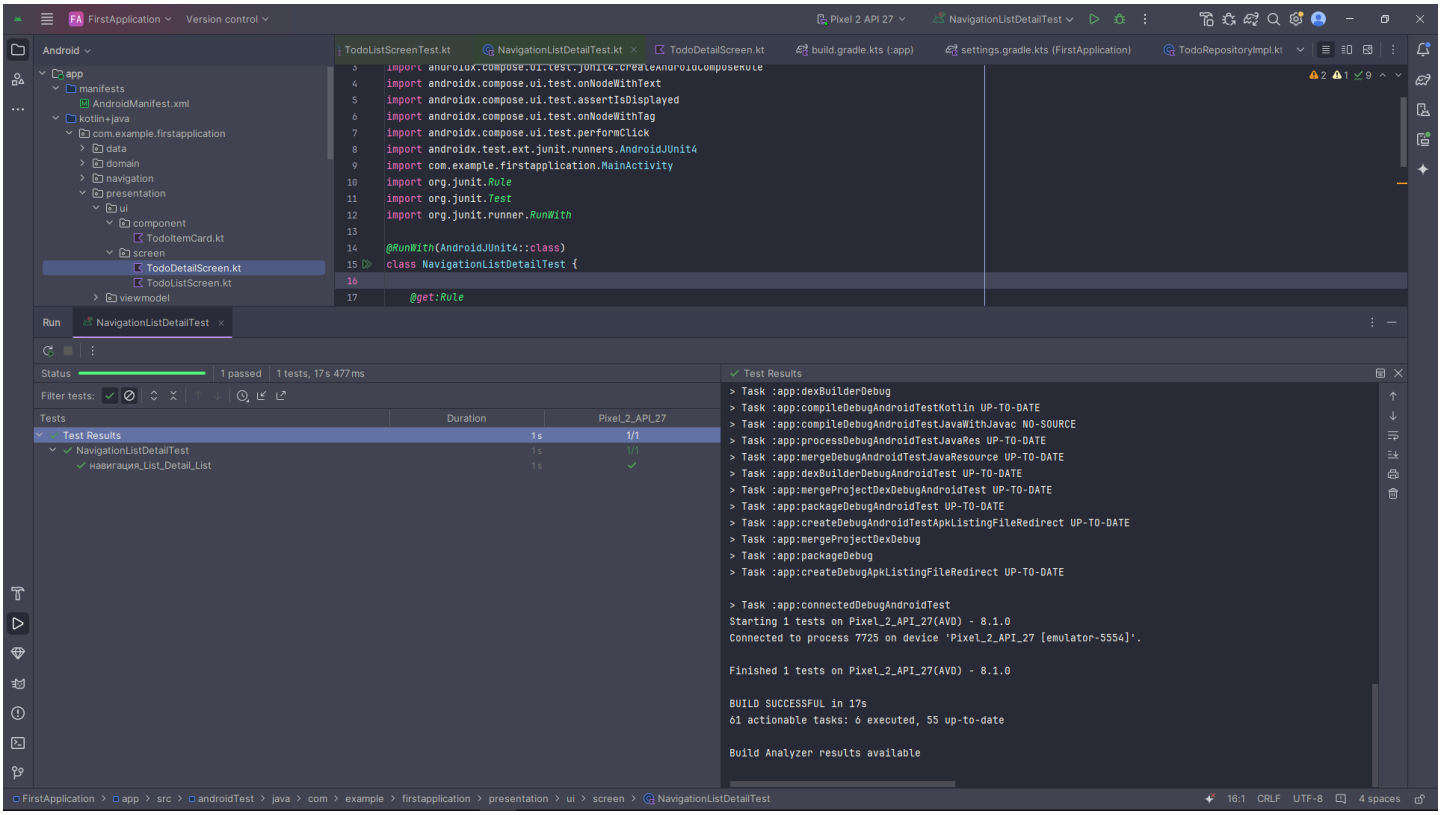


Рисунок 18- тест 5

```kotlin
package com.example.firstapplication.presentation.ui.screen

import androidx.compose.ui.test.junit4.createAndroidComposeRule
import androidx.compose.ui.test.onNodeWithText
import androidx.compose.ui.test.assertIsDisplayed
import androidx.compose.ui.test.onNodeWithTag
import androidx.compose.ui.test.performClick
import androidx.test.ext.junit.runners.AndroidJUnit4
import com.example.firstapplication.MainActivity
import org.junit.Rule
import org.junit.Test
import org.junit.runner.RunWith

@RunWith(AndroidJUnit4::class)
class NavigationListDetailTest {

    @get:Rule
    val composeTestRule = createAndroidComposeRule<MainActivity>()

    @Test
    fun навигация_List_Detail_List() {
        composeTestRule.onNodeWithText("Список задач")
            .assertIsDisplayed()

        composeTestRule.onNodeWithText("Купить молоко")
            .assertIsDisplayed()
            .performClick()

        // вместо "Детали задачи"
        composeTestRule.onNodeWithTag("detailScreen")
            .assertIsDisplayed()

        composeTestRule.activityRule.scenario.onActivity { activity ->
            activity.onBackPressedDispatcher.onBackPressed()
        }

        composeTestRule.onNodeWithText("Список задач")
            .assertIsDisplayed()
    }
}
```

Листинг 15 – Теста 5

ВЫВОДЫ

Часть III – Практическое задание

По Заданию 1:

Освоил принципы ленивой загрузки в Compose — LazyColumn эффективно отображает большие списки кораблей, создавая элементы только при их видимости, что значительно экономит память и ресурсы устройства.

Успешно реализовал оптимизированную загрузку изображений — применил BitmapFactory с inSampleSize и дополнительным масштабированием, обеспечив быструю отрисовку thumbnails размером 80dp без потери качества.

Мастерски применил композицию UI — создал структурированные карточки ShipCard с Row, содержащим Image, Spacer и Column с двумя Text, обеспечив четкую визуальную иерархию и адаптивный дизайн.

По Заданию 2:

Профессионально настроил кастомную Material 3 тему — успешно экспортировал цветовую палитру из Material Theme Builder, заменив Color.kt, Theme.kt и Type.kt, обеспечив единый стиль для светлой/темной тем и контрастных вариантов.

Интегрировал пользовательские шрифты из Google Fonts — добавил семейство в res/font и применил в Type.kt через AppTypography, значительно повысив визуальную индивидуальность интерфейса.

Настроил кастомные формы компонентов — создал Shape.kt с измененными радиусами для small, medium и large, применив их в Theme.kt для создания уникального фирменного стиля приложения в соответствии с Material Design 3.

Часть V – Практическое задание

По Заданию 1:

Добавил логирование жизненного цикла Activity через Log.d() в методах onCreate/onStart/onResume/onPause/onStop/onDestroy с единым TAG.

По Заданию 2:

Реализовал CounterViewModel для сохранения состояния счетчика при повороте экрана, интегрировал с UI через ViewModelProvider.

По Заданию 3:

Настроил навигацию с NavHost — создал Home/Details экраны, добавил переходы navigate() и popBackStack().

По Заданию 4:

Применил WindowSizeClass для адаптивного UI — Compact (1 колонка), Medium/Expanded (2 колонки) с calculateWindowSizeClass.

Часть IV – Практическое задание

По Заданию 7:

Реализовал Clean Architecture — создал 3 слоя (Presentation/Domain/Data), TodoJsonDataSource для чтения из JSON, use cases, ViewModel и навигацию List→Detail с чекбоксами.

По Заданию 8:

Написал 2 unit-теста (GetTodosUseCase возвращает 3 задачи, ToggleTodoUseCase меняет статус) с MockK. Создал 3 UI-теста: отображение 3 задач из JSON, переключение чекбокса, навигация List→Detail→List.

Репозиторий: https://github.com/oggiorgi/MIREA/tree/main/5sem/Kotlin/PR3