23 приложение «Супергерои»

Вы изучили основы Material Design и научились добавлять простые анимации в свое приложение. Теперь пришло время применить полученные знания на практике.

В этом практическом наборе вы будете опираться на концепции, которые вы изучили, создавая приложение **«Супергерои»** . В этом приложении основное внимание уделяется созданию компонентов, необходимых для создания прокручиваемого списка и усовершенствованного пользовательского интерфейса, с использованием принципов Material Design, которые вы изучили ранее.

Что ты построишь

Приложение **«Супергерои»** , отображающее список супергероев.

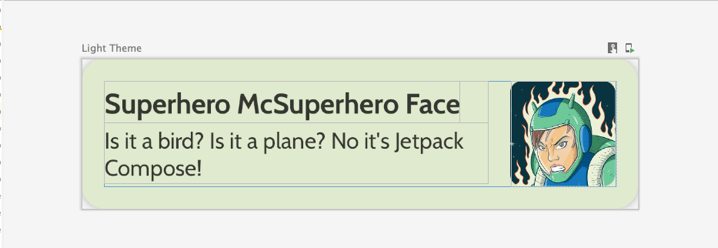
Окончательное приложение будет выглядеть следующим образом как в светлой, так и в темной теме:

|  |  |
| --- | --- |
|  |  |

[2. Начните работу](https://developer.android.com/codelabs/basic-android-kotlin-compose-practice-superheroes?continue=https%3A%2F%2Fdeveloper.android.com%2Fcourses%2Fpathways%2Fandroid-basics-compose-unit-3-pathway-3%23codelab-https%3A%2F%2Fdeveloper.android.com%2Fcodelabs%2Fbasic-android-kotlin-compose-practice-superheroes#1)

В этой задаче вы настраиваете проект и создаете фиктивные данные для супергероев.

1. Создайте новый проект с шаблоном **Empty Activity** и минимум 24 SDK.
2. Загрузите ресурсы для приложения: изображения супергероев и логотип приложения [отсюда](https://github.com/google-developer-training/basic-android-kotlin-compose-training-superheroes/raw/main/Image%20assets.zip). Дополнительную информацию о том, как добавить значок приложения см. в статье [«Изменение значка приложения»](https://developer.android.com/codelabs/basic-android-kotlin-compose-training-change-app-icon) .
3. Загрузите файлы шрифтов Cabin Bold и Cabin Regular с [https://fonts.google.com](https://fonts.google.com/) . Изучите различные доступные файлы шрифтов.
4. Создайте класс данных для хранения данных для каждого супергероя. Создайте новый пакет model с именем класса данных Hero для организации вашего кода. Ваш элемент списка может выглядеть примерно так:



Каждый элемент списка супергероев отображает три части уникальной информации: имя, описание и изображение.

1. В том же пакете model создайте еще один файл для всей информации о героях, которую вы хотите отобразить. Например, имя, описание и ресурс изображения. Ниже приведен пример набора данных для вашего вдохновения.

object HeroesRepository {  
    val heroes = listOf(  
        Hero(  
            nameRes = R.string.hero1,  
            descriptionRes = R.string.description1,  
            imageRes = R.drawable.android\_superhero1  
        ),  
        Hero(  
            nameRes = R.string.hero2,  
            descriptionRes = R.string.description2,  
            imageRes = R.drawable.android\_superhero2  
        ),  
        Hero(  
            nameRes = R.string.hero3,  
            descriptionRes = R.string.description3,  
            imageRes = R.drawable.android\_superhero3  
        ),  
        Hero(  
            nameRes = R.string.hero4,  
            descriptionRes = R.string.description4,  
            imageRes = R.drawable.android\_superhero4  
        ),  
        Hero(  
            nameRes = R.string.hero5,  
            descriptionRes = R.string.description5,  
            imageRes = R.drawable.android\_superhero5  
        ),  
        Hero(  
            nameRes = R.string.hero6,  
            descriptionRes = R.string.description6,  
            imageRes = R.drawable.android\_superhero6  
        )  
    )  
}

1. Добавьте строки имени и описания героев в файл **strings.xml** .

<resources>  
    <string name="app\_name">Superheroes</string>  
    <string name="hero1">Nick the Night and Day</string>  
    <string name="description1">The Jetpack Hero</string>  
    <string name="hero2">Reality Protector</string>  
    <string name="description2">Understands the absolute truth</string>  
    <string name="hero3">Andre the Giant</string>  
    <string name="description3">Mimics the light and night to blend in</string>  
    <string name="hero4">Benjamin the Brave</string>  
    <string name="description4">Harnesses the power of canary to develop bravely</string>  
    <string name="hero5">Magnificent Maru</string>  
    <string name="description5">Effortlessly glides in to save the day</string>  
    <string name="hero6">Dynamic Yasmine</string>  
    <string name="description6">Ability to shift to any form and energize</string>  
</resources>

[3. Тематика материалов](https://developer.android.com/codelabs/basic-android-kotlin-compose-practice-superheroes?continue=https%3A%2F%2Fdeveloper.android.com%2Fcourses%2Fpathways%2Fandroid-basics-compose-unit-3-pathway-3%23codelab-https%3A%2F%2Fdeveloper.android.com%2Fcodelabs%2Fbasic-android-kotlin-compose-practice-superheroes#2)

В этом разделе вы добавите цветовую палитру, типографику и формы приложения, чтобы улучшить внешний вид приложения.

Следующие цвета, тип и форма являются лишь рекомендациями для этой темы. Исследуйте и изменяйте различные цветовые схемы.

Используйте [Material Theme Builder](https://m3.material.io/theme-builder" \l "/custom" \t "_blank) , чтобы создать новую тему для приложения.

**Цвет**

ui.theme/Color.kt

import androidx.compose.ui.graphics.Color

val md\_theme\_light\_primary = Color(0xFF466800)  
val md\_theme\_light\_onPrimary = Color(0xFFFFFFFF)  
val md\_theme\_light\_primaryContainer = Color(0xFFC6F181)  
val md\_theme\_light\_onPrimaryContainer = Color(0xFF121F00)  
val md\_theme\_light\_secondary = Color(0xFF596248)  
val md\_theme\_light\_onSecondary = Color(0xFFFFFFFF)  
val md\_theme\_light\_secondaryContainer = Color(0xFFDDE6C6)  
val md\_theme\_light\_onSecondaryContainer = Color(0xFF161E0A)  
val md\_theme\_light\_tertiary = Color(0xFF396661)  
val md\_theme\_light\_onTertiary = Color(0xFFFFFFFF)  
val md\_theme\_light\_tertiaryContainer = Color(0xFFBCECE6)  
val md\_theme\_light\_onTertiaryContainer = Color(0xFF00201D)  
val md\_theme\_light\_error = Color(0xFFBA1A1A)  
val md\_theme\_light\_errorContainer = Color(0xFFFFDAD6)  
val md\_theme\_light\_onError = Color(0xFFFFFFFF)  
val md\_theme\_light\_onErrorContainer = Color(0xFF410002)  
val md\_theme\_light\_background = Color(0xFFFEFCF5)  
val md\_theme\_light\_onBackground = Color(0xFF1B1C18)  
val md\_theme\_light\_surface = Color(0xFFFEFCF5)  
val md\_theme\_light\_onSurface = Color(0xFF1B1C18)  
val md\_theme\_light\_surfaceVariant = Color(0xFFE1E4D4)  
val md\_theme\_light\_onSurfaceVariant = Color(0xFF45483D)  
val md\_theme\_light\_outline = Color(0xFF75786C)  
val md\_theme\_light\_inverseOnSurface = Color(0xFFF2F1E9)  
val md\_theme\_light\_inverseSurface = Color(0xFF30312C)  
val md\_theme\_light\_inversePrimary = Color(0xFFABD468)  
val md\_theme\_light\_surfaceTint = Color(0xFF466800)  
val md\_theme\_light\_outlineVariant = Color(0xFFC5C8B9)  
val md\_theme\_light\_scrim = Color(0xFF000000)  
  
val md\_theme\_dark\_primary = Color(0xFFABD468)  
val md\_theme\_dark\_onPrimary = Color(0xFF223600)  
val md\_theme\_dark\_primaryContainer = Color(0xFF344E00)  
val md\_theme\_dark\_onPrimaryContainer = Color(0xFFC6F181)  
val md\_theme\_dark\_secondary = Color(0xFFC1CAAB)  
val md\_theme\_dark\_onSecondary = Color(0xFF2B331D)  
val md\_theme\_dark\_secondaryContainer = Color(0xFF414A32)  
val md\_theme\_dark\_onSecondaryContainer = Color(0xFFDDE6C6)  
val md\_theme\_dark\_tertiary = Color(0xFFA0D0CA)  
val md\_theme\_dark\_onTertiary = Color(0xFF013733)  
val md\_theme\_dark\_tertiaryContainer = Color(0xFF1F4E4A)  
val md\_theme\_dark\_onTertiaryContainer = Color(0xFFBCECE6)  
val md\_theme\_dark\_error = Color(0xFFFFB4AB)  
val md\_theme\_dark\_errorContainer = Color(0xFF93000A)  
val md\_theme\_dark\_onError = Color(0xFF690005)  
val md\_theme\_dark\_onErrorContainer = Color(0xFFFFDAD6)  
val md\_theme\_dark\_background = Color(0xFF1B1C18)  
val md\_theme\_dark\_onBackground = Color(0xFFE4E3DB)  
val md\_theme\_dark\_surface = Color(0xFF1B1C18)  
val md\_theme\_dark\_onSurface = Color(0xFFE4E3DB)  
val md\_theme\_dark\_surfaceVariant = Color(0xFF45483D)  
val md\_theme\_dark\_onSurfaceVariant = Color(0xFFC5C8B9)  
val md\_theme\_dark\_outline = Color(0xFF8F9285)  
val md\_theme\_dark\_inverseOnSurface = Color(0xFF1B1C18)  
val md\_theme\_dark\_inverseSurface = Color(0xFFE4E3DB)  
val md\_theme\_dark\_inversePrimary = Color(0xFF466800)  
val md\_theme\_dark\_surfaceTint = Color(0xFFABD468)  
val md\_theme\_dark\_outlineVariant = Color(0xFF45483D)  
val md\_theme\_dark\_scrim = Color(0xFF000000)

**Форма**

ui.theme/Shape.kt

import androidx.compose.foundation.shape.RoundedCornerShape  
import androidx.compose.material3.Shapes  
import androidx.compose.ui.unit.dp  
  
val Shapes = Shapes(  
    small = RoundedCornerShape(8.dp),  
    medium = RoundedCornerShape(16.dp),  
    large = RoundedCornerShape(16.dp)  
)

**Типография**

ui.theme/Type.kt

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.superheroes.R  
  
  
val Cabin = FontFamily(  
    Font(R.font.cabin\_regular, FontWeight.Normal),  
    Font(R.font.cabin\_bold, FontWeight.Bold)  
)  
// Set of Material typography styles to start with  
val Typography = Typography(  
    bodyLarge = TextStyle(  
        fontFamily = Cabin,  
        fontWeight = FontWeight.Normal,  
        fontSize = 16.sp,  
        lineHeight = 24.sp,  
        letterSpacing = 0.5.sp  
    ),  
    displayLarge = TextStyle(  
        fontFamily = Cabin,  
        fontWeight = FontWeight.Normal,  
        fontSize = 30.sp  
    ),  
    displayMedium = TextStyle(  
        fontFamily = Cabin,  
        fontWeight = FontWeight.Bold,  
        fontSize = 20.sp  
    ),  
    displaySmall = TextStyle(  
        fontFamily = Cabin,  
        fontWeight = FontWeight.Bold,  
        fontSize = 20.sp  
    )  
)

**Тема**

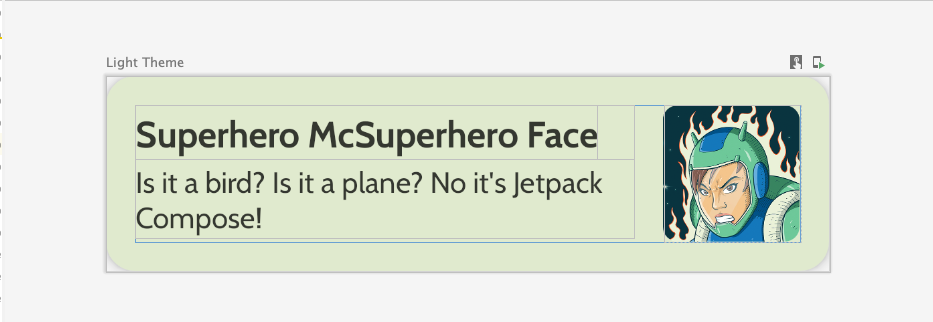
ui.theme/Theme.kt

import android.app.Activity  
import android.os.Build  
import androidx.compose.foundation.isSystemInDarkTheme  
import androidx.compose.material3.MaterialTheme  
import androidx.compose.material3.darkColorScheme  
import androidx.compose.material3.dynamicDarkColorScheme  
import androidx.compose.material3.dynamicLightColorScheme  
import androidx.compose.material3.lightColorScheme  
import androidx.compose.runtime.Composable  
import androidx.compose.runtime.SideEffect  
import androidx.compose.ui.graphics.toArgb  
import androidx.compose.ui.platform.LocalContext  
import androidx.compose.ui.platform.LocalView  
import androidx.core.view.WindowCompat  
  
private val LightColors = lightColorScheme(  
   primary = md\_theme\_light\_primary,  
   onPrimary = md\_theme\_light\_onPrimary,  
   primaryContainer = md\_theme\_light\_primaryContainer,  
   onPrimaryContainer = md\_theme\_light\_onPrimaryContainer,  
   secondary = md\_theme\_light\_secondary,  
   onSecondary = md\_theme\_light\_onSecondary,  
   secondaryContainer = md\_theme\_light\_secondaryContainer,  
   onSecondaryContainer = md\_theme\_light\_onSecondaryContainer,  
   tertiary = md\_theme\_light\_tertiary,  
   onTertiary = md\_theme\_light\_onTertiary,  
   tertiaryContainer = md\_theme\_light\_tertiaryContainer,  
   onTertiaryContainer = md\_theme\_light\_onTertiaryContainer,  
   error = md\_theme\_light\_error,  
   errorContainer = md\_theme\_light\_errorContainer,  
   onError = md\_theme\_light\_onError,  
   onErrorContainer = md\_theme\_light\_onErrorContainer,  
   background = md\_theme\_light\_background,  
   onBackground = md\_theme\_light\_onBackground,  
   surface = md\_theme\_light\_surface,  
   onSurface = md\_theme\_light\_onSurface,  
   surfaceVariant = md\_theme\_light\_surfaceVariant,  
   onSurfaceVariant = md\_theme\_light\_onSurfaceVariant,  
   outline = md\_theme\_light\_outline,  
   inverseOnSurface = md\_theme\_light\_inverseOnSurface,  
   inverseSurface = md\_theme\_light\_inverseSurface,  
   inversePrimary = md\_theme\_light\_inversePrimary,  
   surfaceTint = md\_theme\_light\_surfaceTint,  
   outlineVariant = md\_theme\_light\_outlineVariant,  
   scrim = md\_theme\_light\_scrim,  
)  
  
private val DarkColors = darkColorScheme(  
   primary = md\_theme\_dark\_primary,  
   onPrimary = md\_theme\_dark\_onPrimary,  
   primaryContainer = md\_theme\_dark\_primaryContainer,  
   onPrimaryContainer = md\_theme\_dark\_onPrimaryContainer,  
   secondary = md\_theme\_dark\_secondary,  
   onSecondary = md\_theme\_dark\_onSecondary,  
   secondaryContainer = md\_theme\_dark\_secondaryContainer,  
   onSecondaryContainer = md\_theme\_dark\_onSecondaryContainer,  
   tertiary = md\_theme\_dark\_tertiary,  
   onTertiary = md\_theme\_dark\_onTertiary,  
   tertiaryContainer = md\_theme\_dark\_tertiaryContainer,  
   onTertiaryContainer = md\_theme\_dark\_onTertiaryContainer,  
   error = md\_theme\_dark\_error,  
   errorContainer = md\_theme\_dark\_errorContainer,  
   onError = md\_theme\_dark\_onError,  
   onErrorContainer = md\_theme\_dark\_onErrorContainer,  
   background = md\_theme\_dark\_background,  
   onBackground = md\_theme\_dark\_onBackground,  
   surface = md\_theme\_dark\_surface,  
   onSurface = md\_theme\_dark\_onSurface,  
   surfaceVariant = md\_theme\_dark\_surfaceVariant,  
   onSurfaceVariant = md\_theme\_dark\_onSurfaceVariant,  
   outline = md\_theme\_dark\_outline,  
   inverseOnSurface = md\_theme\_dark\_inverseOnSurface,  
   inverseSurface = md\_theme\_dark\_inverseSurface,  
   inversePrimary = md\_theme\_dark\_inversePrimary,  
   surfaceTint = md\_theme\_dark\_surfaceTint,  
   outlineVariant = md\_theme\_dark\_outlineVariant,  
   scrim = md\_theme\_dark\_scrim,  
)  
  
@Composable  
fun SuperheroesTheme(  
   darkTheme: Boolean = isSystemInDarkTheme(),  
   // Dynamic color is available on Android 12+  
   // Dynamic color in this app is turned off for learning purposes  
   dynamicColor: Boolean = false,  
   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 -> DarkColors  
       else -> LightColors  
   }  
   val view = LocalView.current  
   if (!view.isInEditMode) {  
       SideEffect {  
           val window = (view.context as Activity).window  
           window.statusBarColor = colorScheme.background.toArgb()  
           WindowCompat.getInsetsController(window, view).isAppearanceLightStatusBars = !darkTheme  
       }  
   }  
  
   MaterialTheme(  
       colorScheme = colorScheme,  
       typography = Typography,  
       shapes = Shapes,  
       content = content  
   )  
}

[4. Список отображения](https://developer.android.com/codelabs/basic-android-kotlin-compose-practice-superheroes?continue=https%3A%2F%2Fdeveloper.android.com%2Fcourses%2Fpathways%2Fandroid-basics-compose-unit-3-pathway-3%23codelab-https%3A%2F%2Fdeveloper.android.com%2Fcodelabs%2Fbasic-android-kotlin-compose-practice-superheroes#3)

Первым шагом в создании списка является создание элемента списка.

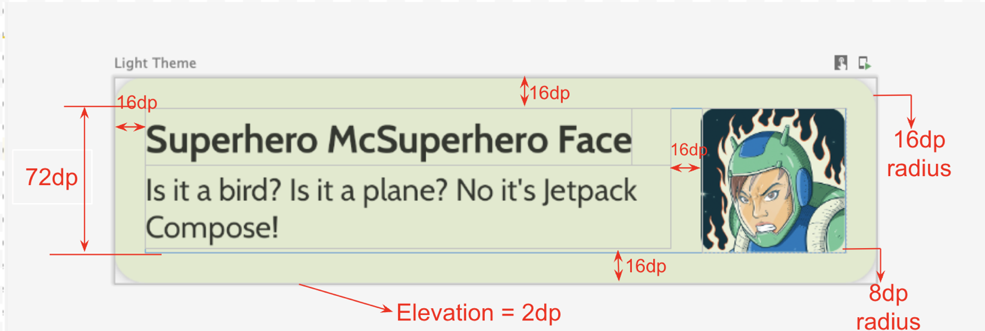
1. Создайте файл с именем HeroesScreen.kt, в пакете com.example.superheroes. В этом файле вы будете создавать элемент списка и составные элементы списка.
2. Создайте составной элемент, представляющий элемент списка супергероев, который выглядит как следующий снимок экрана и спецификации пользовательского интерфейса.

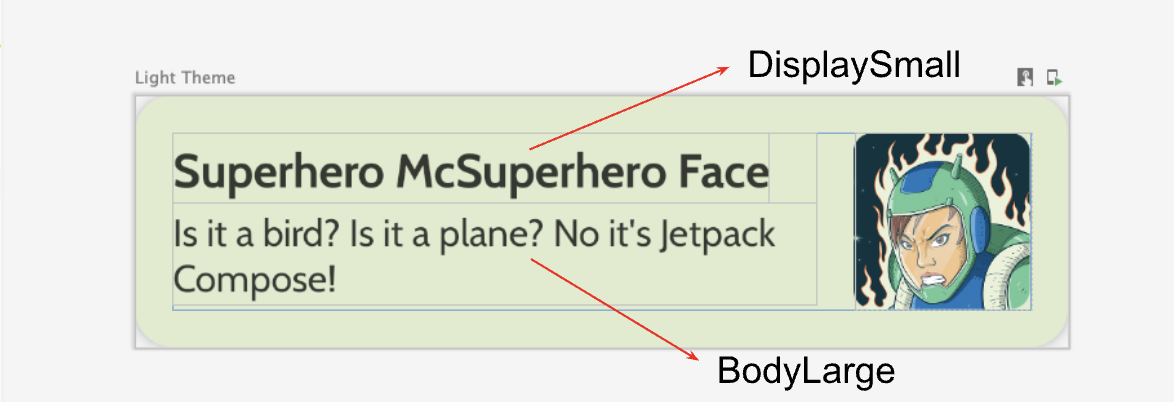


Следуйте этой спецификации пользовательского интерфейса или проявите творческий подход и создайте свой собственный элемент списка:

* Повышение уровня карты 2dp
* Высота элемента списка 72dp с отступом 16dp
* Радиус отсечения элемента списка равен 16dp
* макет Box с изображением с размером 72dp
* Радиус обрезки изображения 8dp
* Расстояние между изображением и текстом 16dp
* Стиль имени супергероя: DisplaySmall
* Стиль описания супергероя: BodyLarge

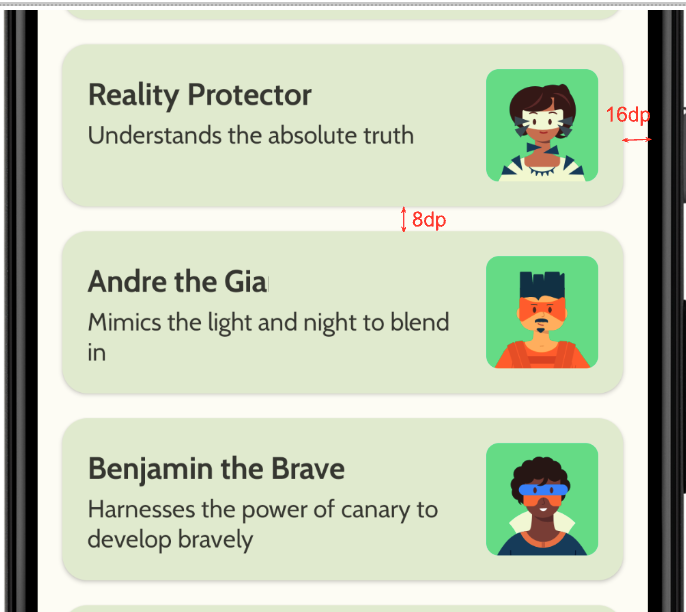
Изучите различные варианты отступов и размеров. Согласно рекомендациям Material 3, отступы должны быть с шагом 4dp.





**Создайте LazyColumn**

1. Создайте еще один составной объект, который принимает список героев и отображает его. Здесь вы используете файл [LazyColumn](https://developer.android.com/reference/kotlin/androidx/compose/foundation/lazy/package-summary#LazyColumn(androidx.compose.ui.Modifier,androidx.compose.foundation.lazy.LazyListState,androidx.compose.foundation.layout.PaddingValues,kotlin.Boolean,androidx.compose.foundation.layout.Arrangement.Vertical,androidx.compose.ui.Alignment.Horizontal,androidx.compose.foundation.gestures.FlingBehavior,kotlin.Boolean,kotlin.Function1)).
2. Используйте следующие спецификации пользовательского интерфейса для заполнения.



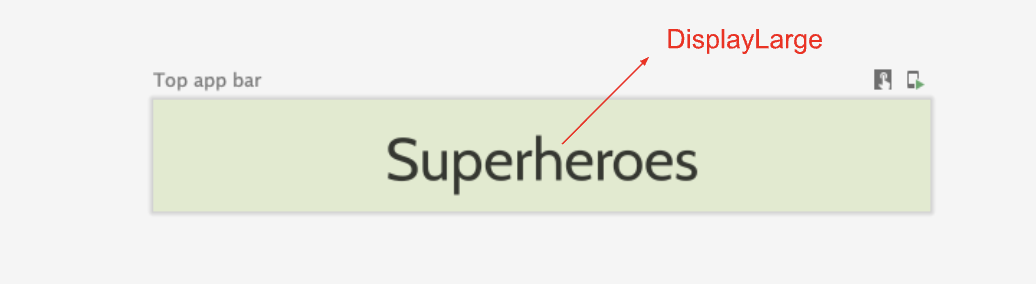
После завершения реализации ваше приложение должно соответствовать следующему снимку экрана:



[5. Добавьте верхнюю панель приложений.](https://developer.android.com/codelabs/basic-android-kotlin-compose-practice-superheroes?continue=https%3A%2F%2Fdeveloper.android.com%2Fcourses%2Fpathways%2Fandroid-basics-compose-unit-3-pathway-3%23codelab-https%3A%2F%2Fdeveloper.android.com%2Fcodelabs%2Fbasic-android-kotlin-compose-practice-superheroes#4)

Добавьте верхнюю панель приложения для своего приложения.

1. В MainActivity.kt добавьте составной элемент для отображения верхней панели приложения. Добавьте текст в верхнюю панель приложения; это может быть название приложения. Выровняйте его по центру как по горизонтали, так и по вертикали.
2. Вы можете установить верхнюю панель приложения со стилем DisplayLarge.



1. Используйте scaffold для отображения верхней панели приложения. При необходимости обратитесь к [верхней панели приложения — документации Material Design 3 .](https://m3.material.io/components/top-app-bar/overview)

Настроить цвет строки состояния

Чтобы сделать ваше приложение [безрамочным](https://developer.android.com/develop/ui/views/layout/edge-to-edge) , вы можете настроить цвет строки состояния в соответствии с цветом фона.

1. В Theme.kt добавьте этот новый метод, чтобы изменить цвета строки состояния и панели навигации от края до края.

/\*\*  
 \* Sets up edge-to-edge for the window of this [view]. The system icon colors are set to either  
 \* light or dark depending on whether the [darkTheme] is enabled or not.  
 \*/  
private fun setUpEdgeToEdge(view: View, darkTheme: Boolean) {  
    val window = (view.context as Activity).window  
    WindowCompat.setDecorFitsSystemWindows(window, false)  
    window.statusBarColor = Color.Transparent.toArgb()  
    val navigationBarColor = when {  
        Build.VERSION.SDK\_INT >= 29 -> Color.Transparent.toArgb()  
        Build.VERSION.SDK\_INT >= 26 -> Color(0xFF, 0xFF, 0xFF, 0x63).toArgb()  
        // Min sdk version for this app is 24, this block is for SDK versions 24 and 25  
        else -> Color(0x00, 0x00, 0x00, 0x50).toArgb()  
    }  
    window.navigationBarColor = navigationBarColor  
    val controller = WindowCompat.getInsetsController(window, view)  
    controller.isAppearanceLightStatusBars = !darkTheme  
    controller.isAppearanceLightNavigationBars = !darkTheme  
}

1. В функции SuperheroesTheme()вызовите функцию setUpEdgeToEdge() изнутри блока SideEffect.

fun SuperheroesTheme(  
    darkTheme: Boolean = isSystemInDarkTheme(),  
    // Dynamic color is available on Android 12+  
    // Dynamic color in this app is turned off for learning purposes  
    dynamicColor: Boolean = false,  
    content: @Composable () -> Unit  
) {  
    //...  
    val view = LocalView.current  
    if (!view.isInEditMode) {  
        SideEffect {  
            setUpEdgeToEdge(view, darkTheme)  
        }  
    }  
  
    //...  
}

|  |  |
| --- | --- |
|  |  |