

Material Design, Intent

Esempi di utilizzo

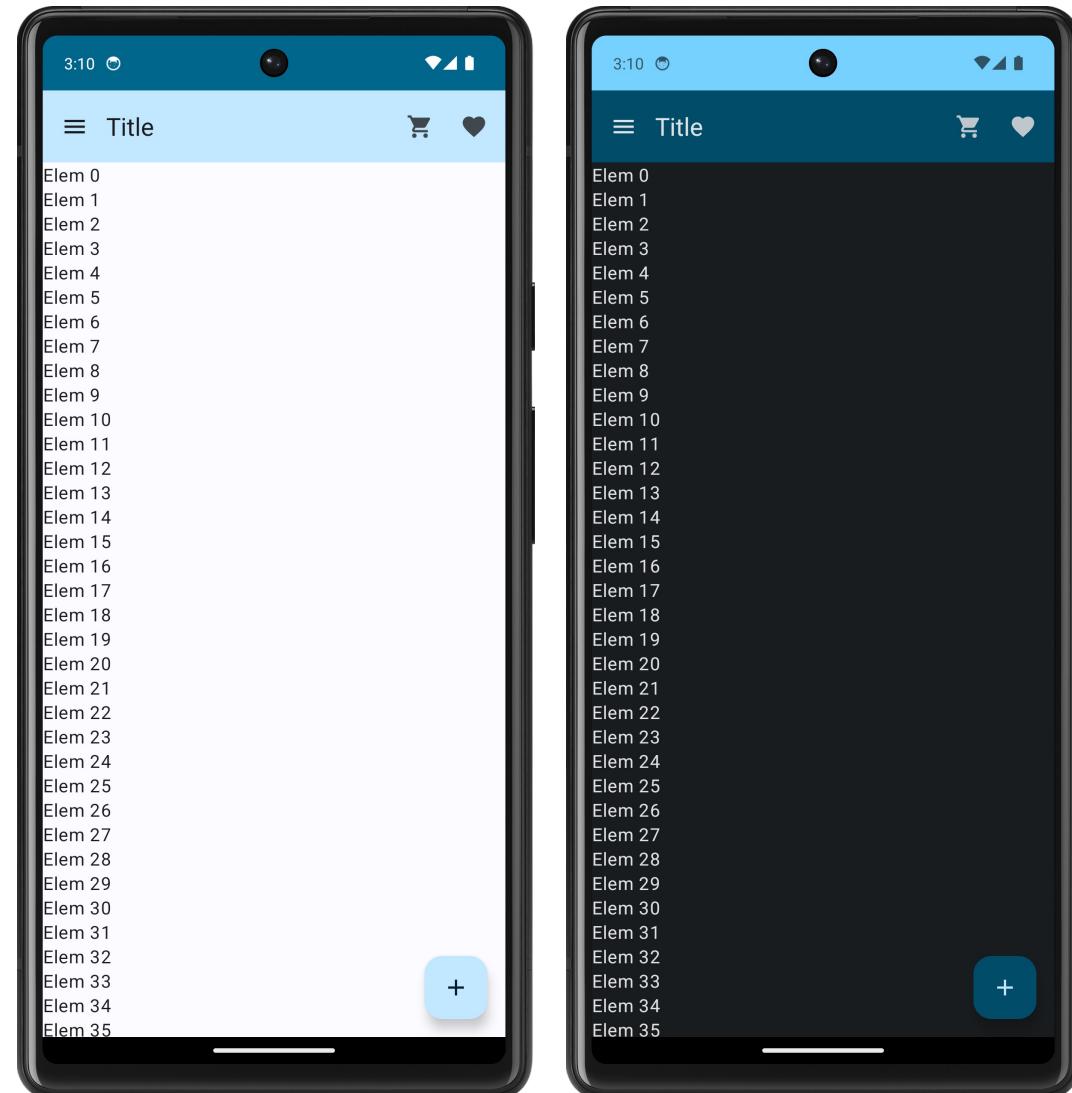
Laboratorio 1

Esercizi

1. Utilizzare elementi di Material Design
2. Avviare un Intent esplicito
3. Avviare un Intent implicito
4. TravelDiary – Schermate dell'applicazione

1.1. Utilizzare elementi di Material Design

- Partire dall'esercizio dello scorso laboratorio (codice disponibile su Virtuale, se necessario)
- Creare un nuovo file **MaterialComponents.kt**
- Implementare nel file uno o più composable per arricchire l'esempio della **LazyColumn** come in figura, aggiungendo:
 - **Scaffold**
 - **TopAppBar**
 - **FloatingActionButton**



1.1. Soluzione

```
@Composable
fun AppLayout() {
    Scaffold(
        topBar = { AppBar() },
        floatingActionButton = {
            FloatingActionButton(onClick = { /*TODO*/ }) {
                Icon(Icons.Filled.Add, "Add item")
            }
        }
    ) { contentPadding ->
        Column(modifier = Modifier.padding(contentPadding)) {
            ScrollableList()
        }
    }
}
```

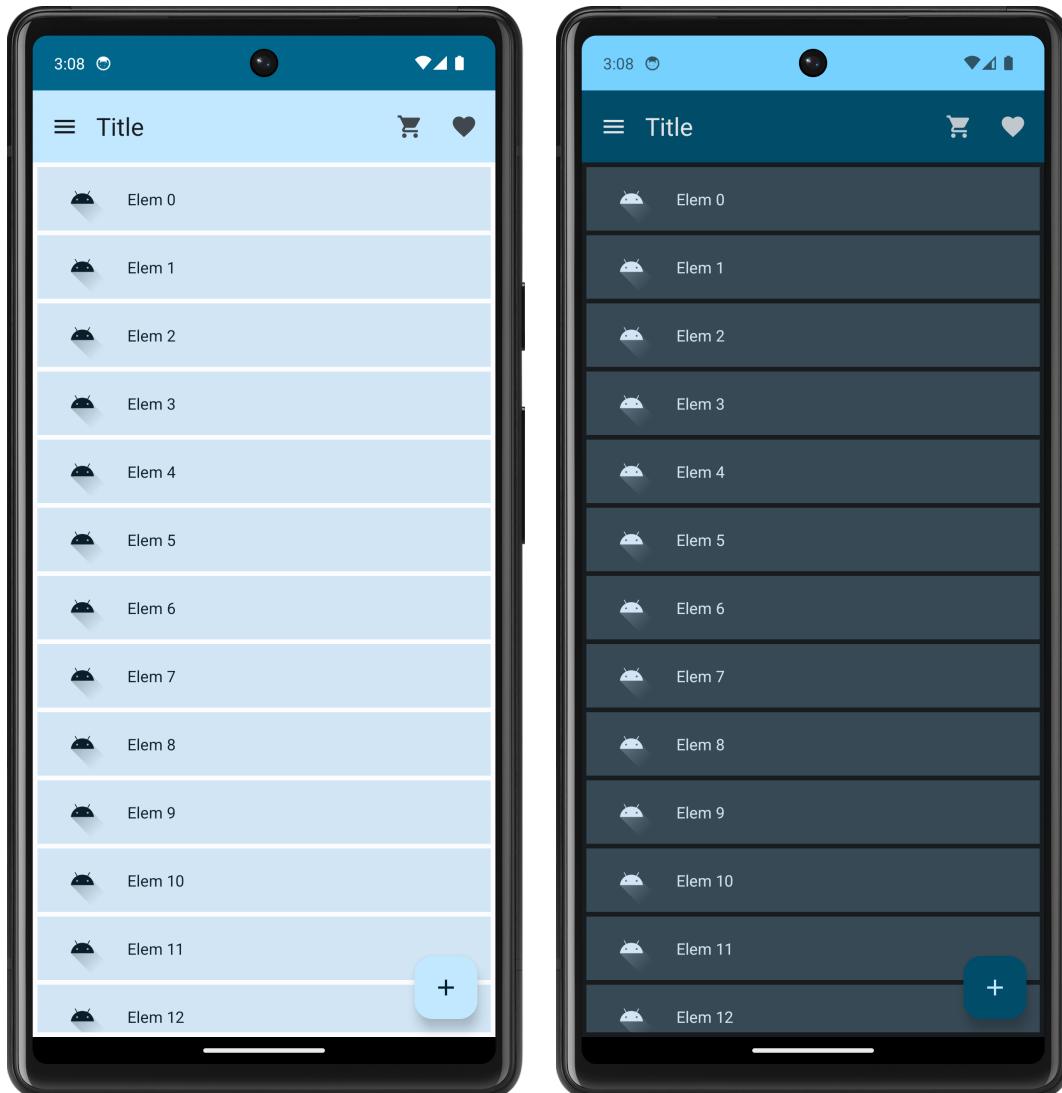
1.1. Soluzione

```
@Composable
fun AppLayout() {
    Scaffold(
        topBar = { AppBar() },
        floatingActionButton = {
            FloatingActionButton(
                Icon(Icons.Filled.Add)
            )
        }
    ) { contentPadding ->
        Column(modifier = Modifier
            .padding(contentPadding))
            .scrollableList()
    }
}
}

@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun AppBar() {
    TopAppBar(
        title = { Text("Title") },
        navigationIcon = {
            IconButton(onClick = { /*TODO*/ }) {
                Icon(Icons.Filled.Menu, "Menu")
            }
        },
        actions = {
            IconButton(onClick = { /*TODO*/ }) {
                Icon(Icons.Filled.ShoppingCart, "Shopping Cart")
            }
            IconButton(onClick = { /*TODO*/ }) {
                Icon(Icons.Filled.Favorite, "Favorites")
            }
        },
        colors = TopAppBarDefaults.topAppBarColors(
            containerColor = MaterialTheme.colorScheme.primaryContainer
        )
    )
}
```

1.2. Utilizzare elementi di Material Design

- Creare un nuovo composable **ListItem** da utilizzare al posto di **Text** per gli elementi della lista
 - Parametro: testo da mostrare
 - Colore di sfondo: **secondaryContainer**
 - Colore del contenuto: **onSecondaryContainer**
 - Mostrare un'immagine accanto al testo



1.2. Soluzione

```
@Composable
fun MaterialList() {
    val elems = (0..100).toList().map { "Elem $it" }

    LazyColumn(
        verticalArrangement = Arrangement.spacedBy(4.dp),
        modifier = Modifier.padding(4.dp)
    ) {
        items(elems) {
            MaterialListItem(it)
        }
    }
}
```

1.2. Soluzione

```
@Composable
fun MaterialList() {
    val elems = (0..100).toList().map { "Elem $it" }

    LazyColumn(
        verticalArrangement = Arrangement.Center,
        modifier = Modifier.padding(16.dp)
    ) {
        items(elems) {
            MaterialListItem(it)
        }
    }
}

@Composable
fun MaterialListItem(item: String, modifier: Modifier = Modifier) {
    Row(
        verticalAlignment = Alignment.CenterVertically,
        modifier = modifier
            .background(MaterialTheme.colorScheme.secondaryContainer)
            .fillMaxWidth()
            .padding(vertical = 4.dp, horizontal = 16.dp)
    ) {
        Image(
            painter = painterResource(R.drawable.ic_launcher_foreground),
            contentDescription = "Android Logo",
            colorFilter = ColorFilter.tint(MaterialTheme.colorScheme.onSecondaryContainer),
            modifier = Modifier.size(48.dp)
        )
        Spacer(Modifier.size(16.dp))
        Text(
            item,
            style = MaterialTheme.typography.bodyMedium,
            color = MaterialTheme.colorScheme.onSecondaryContainer
        )
    }
}
```

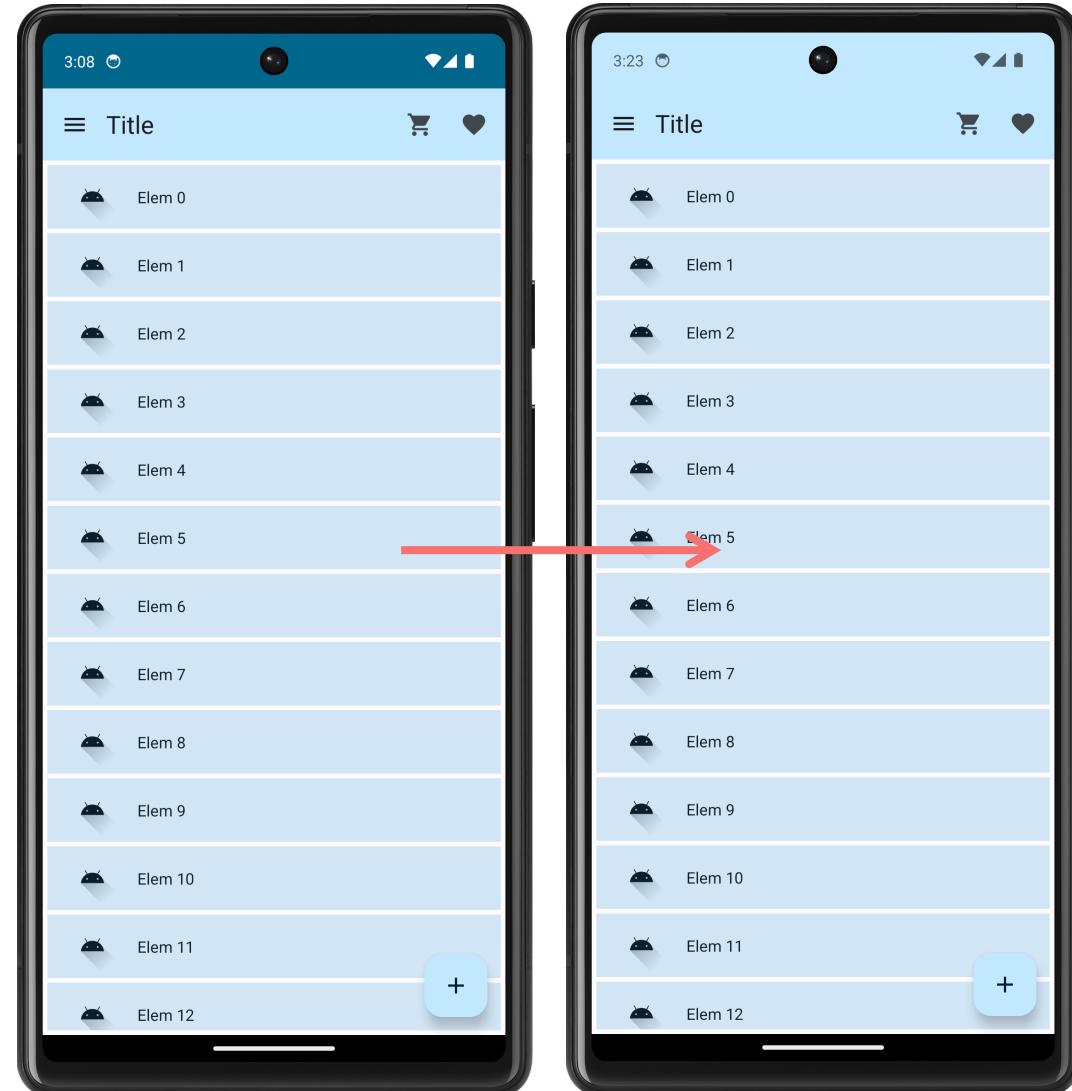
1.2. Bonus

- Per cambiare il colore della status bar in modo che sia identico a quello dell'app bar:
 - Modifichiamo le seguenti righe di **ui.theme.Theme.kt**:

```
window.statusBarColor = colorScheme.primary.toArgb()
WindowCompat.getInsetsController(window, view)
    .isAppearanceLightStatusBars = darkTheme
```

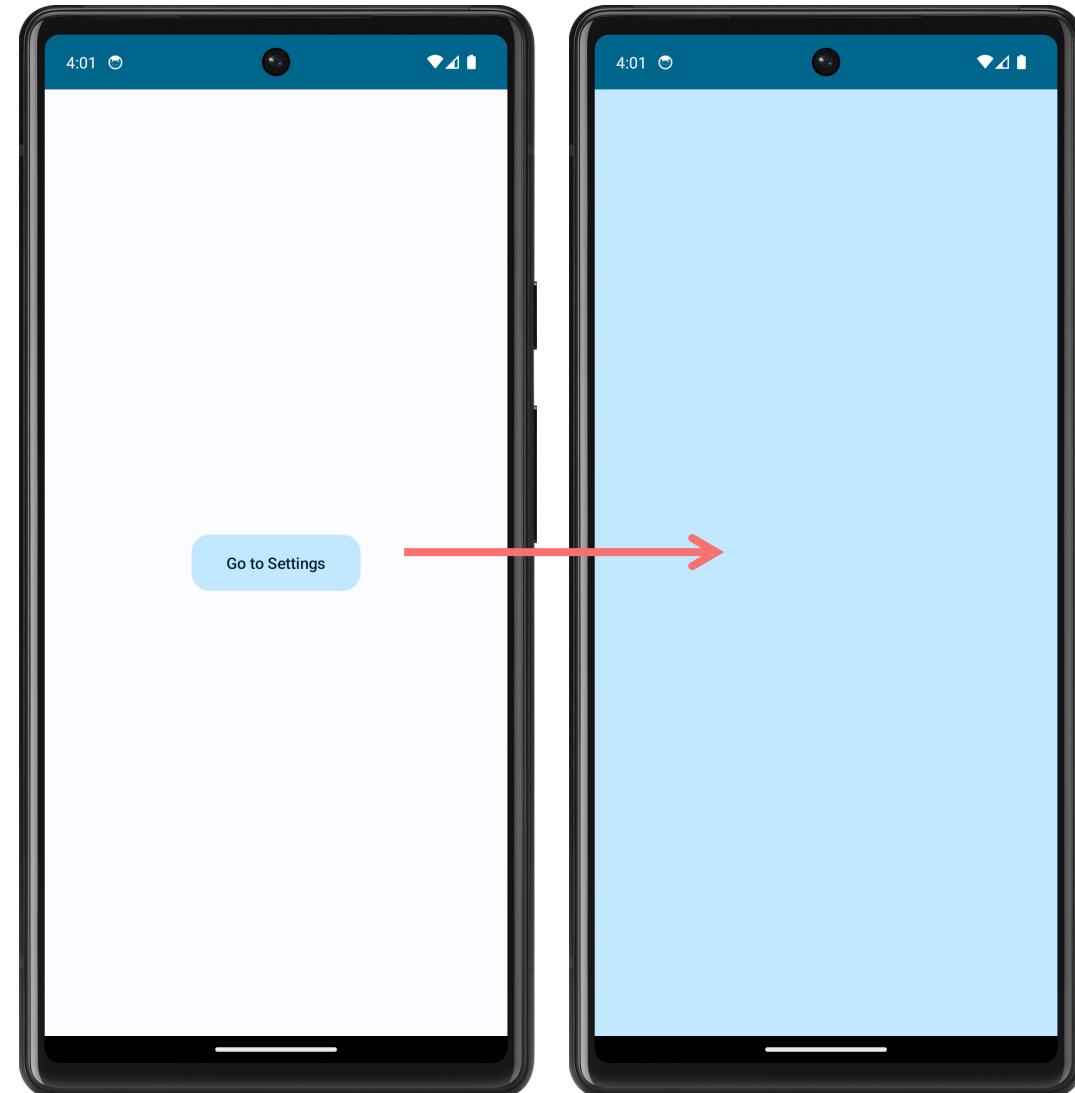
- In:

```
window.statusBarColor = colorScheme.primaryContainer.toArgb()
WindowCompat.getInsetsController(window, view)
    .isAppearanceLightStatusBars = !darkTheme
```



2. Avviare un Intent esplicito

- In un nuovo progetto Android, creare due activity come negli screenshot a lato
 - **MainActivity** (sinistra)
 - **SettingsActivity** (destra)
- Al click sul tasto di **MainActivity**, passare a **SettingsActivity**



2. Caratteristiche del tasto

- Il tasto “Go to Settings” deve avere le seguenti proprietà:
 - Dimensioni: 150dp * 50dp
 - Arrotondamento angoli: 16dp
 - Colore di sfondo: primaryContainer
 - Colore del contenuto: onPrimaryContainer

2. Soluzione

```
class SettingsActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            IntentTheme {
                Surface(
                    modifier = Modifier.fillMaxSize(),
                    color = MaterialTheme.colorScheme.primaryContainer
                ) {
                }
            }
        }
    }
}
```

2. Soluzione

```
class SettingsActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            IntentTheme {
                Surface(
                    modifier = Modifier
                        .color(MaterialTheme.colors.surface)
                ) {
                    Text("Go to Settings")
                }
            }
        }
    }
}

@Composable
fun SettingsButton() {
    val ctx = LocalContext.current

    Button(
        modifier = Modifier.requiredSize(150.dp, 50.dp),
        shape = RoundedCornerShape(16.dp),
        colors = ButtonDefaults.buttonColors(
            containerColor = MaterialTheme.colorScheme.primaryContainer,
            contentColor = MaterialTheme.colorScheme.onPrimaryContainer,
        ),
        onClick = {
            val intent = Intent(ctx, SettingsActivity::class.java)
            ctx.startActivity(intent)
        }
    )
}
```

2. Soluzione

```
class SettingsActivity : ComponentActivity() {  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContent {  
            IntentTheme {  
                Surface(  
                    modifier = Modifier.fillMaxSize()  
                ) {  
                    Column(modifier = Modifier.padding(16.dp)) {  
                        SettingsButton()  
                        Text("Go to Settings")  
                    }  
                }  
            }  
        }  
    }  
}
```

LocalContext.current permette di ottenere un riferimento al contesto corrente, sul quale è possibile richiamare startActivity

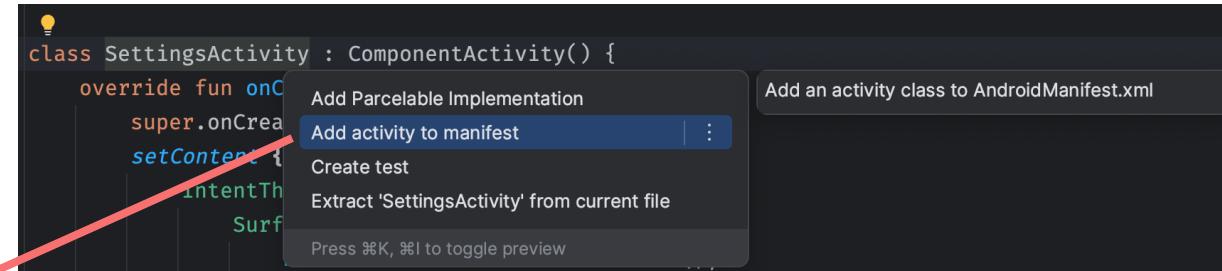
```
@Composable  
fun SettingsButton() {  
    val ctx = LocalContext.current  
    Button(  
        modifier = Modifier.requiredSize(150.dp, 50.dp),  
        shape = RoundedCornerShape(16.dp),  
        colors = ButtonDefaults.buttonColors(  
            containerColor = MaterialTheme.colorScheme.primaryContainer,  
            contentColor = MaterialTheme.colorScheme.onPrimaryContainer,  
        ),  
        onClick = {  
            val intent = Intent(ctx, SettingsActivity::class.java)  
            ctx.startActivity(intent)  
        }  
    ) {  
        Text("Go to Settings")  
    }  
}
```

requiredSize fa l'override delle dimensioni del padre

2. Errore!

android.content.ActivityNotFoundException: Unable to find explicit activity class {com.example.intent/com.example.intent.SettingsActivity}; have you declared this activity in your AndroidManifest.xml, or does your intent not match its declared <intent-filter>?

- Non abbiamo dichiarato **SettingsActivity** nel manifest
- Alt+Invio sul nome dell'activity

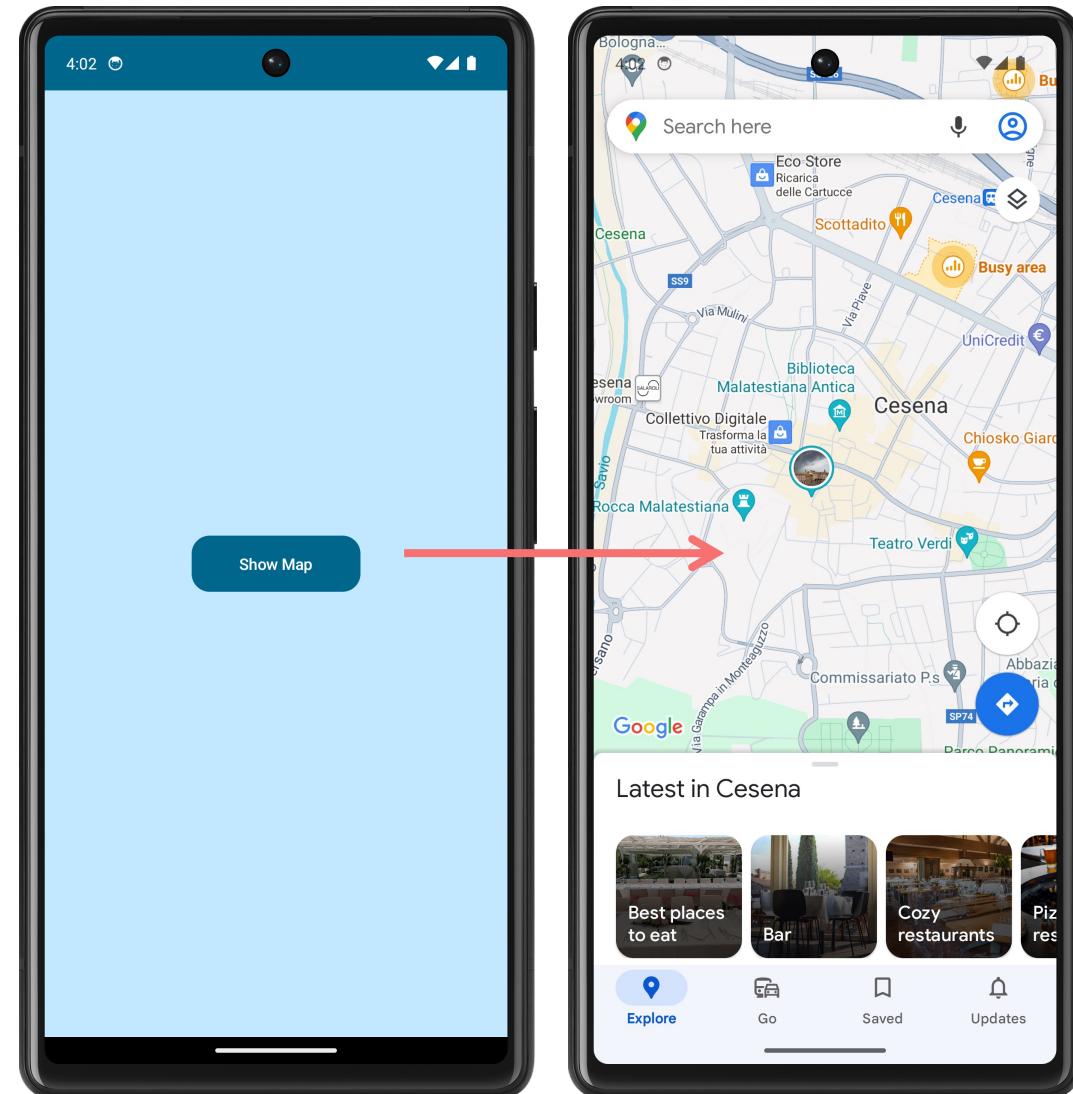


```
<activity android:name="com.example.intent.SettingsActivity" />  
↓  
<activity android:name=".SettingsActivity" android:exported="false" />
```

Evita che l'activity possa essere avviata da altre app

3. Avviare un Intent implicito

- In **SettingsActivity**, aggiungere un tasto “Show Map”
- Al click sul tasto, aprire la mappa sulla posizione di Cesena
 - Coordinate:
geo:44.1391, 12.24315



3. Caratteristiche del tasto

- Il tasto “Show Map” deve avere le seguenti proprietà:
 - Dimensioni: 150dp * 50dp
 - Arrotondamento angoli: 16dp
 - Colore di sfondo: primary
 - Colore del contenuto: onPrimary

3. Soluzione

```
@Composable
fun MapButton() {
    val ctx = LocalContext.current

    Button(
        modifier = Modifier.requiredSize(150.dp, 50.dp),
        shape = RoundedCornerShape(16.dp),
        colors = ButtonDefaults.buttonColors(
            containerColor = MaterialTheme.colorScheme.primary,
            contentColor = MaterialTheme.colorScheme.onPrimary,
        ),
        onClick = {
            val geolocation = Uri.parse("geo:44.1391, 12.24315")
            val intent = Intent(Intent.ACTION_VIEW).apply { data = geolocation }
            if (intent.resolveActivity(ctx.packageManager) != null) {
                ctx.startActivity(intent)
            }
        }
    ) {
        Text("Show Map")
    }
}
```

3. Soluzione

```
@Composable
fun MapButton() {
    val ctx = LocalContext.current

    Button(
        modifier = Modifier.requiredSize(150.dp, 50.dp),
        shape = RoundedCorneredShape(10.dp),
        colors = ButtonDefaults.buttonColors(containerColor = Color.Blue, contentColor = Color.White),
        onClick = {
            val geolocation = LatLng(latitude, longitude)
            val intent = Intent(Intent.ACTION_VIEW, Uri.parse("geo:$latitude,$longitude"))
            if (intent.resolveActivity(ctx.packageManager) != null) {
                ctx.startActivity(intent)
            }
        }
    ) {
        Text("Show Map")
    }
}
```

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">

    <!-- ... -->

    <queries>
        <intent>
            <action android:name="android.intent.action.VIEW" />
            <data android:scheme="geo" />
            <category android:name="android.intent.category.DEFAULT" />
        </intent>
    </queries>

</manifest>
```

4. TravelDiary – Schermate dell'applicazione

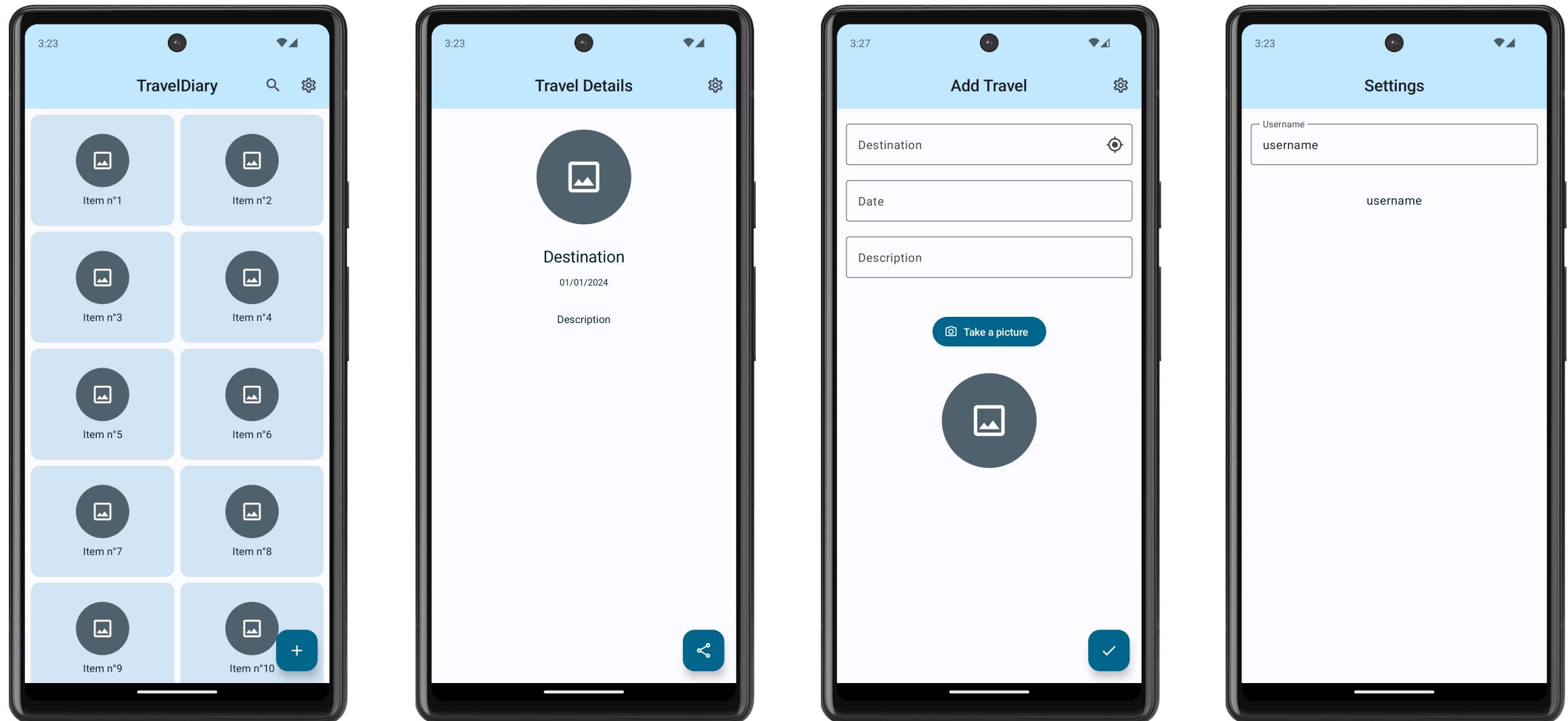
- Creare un nuovo progetto col nome **TravelDiary**
- Ad ogni esercitazione, lo arricchiremo con nuove funzionalità
- Oggi partiamo con la creazione delle quattro schermate che compongono l'app

4. TravelDiary – Schermate dell'applicazione

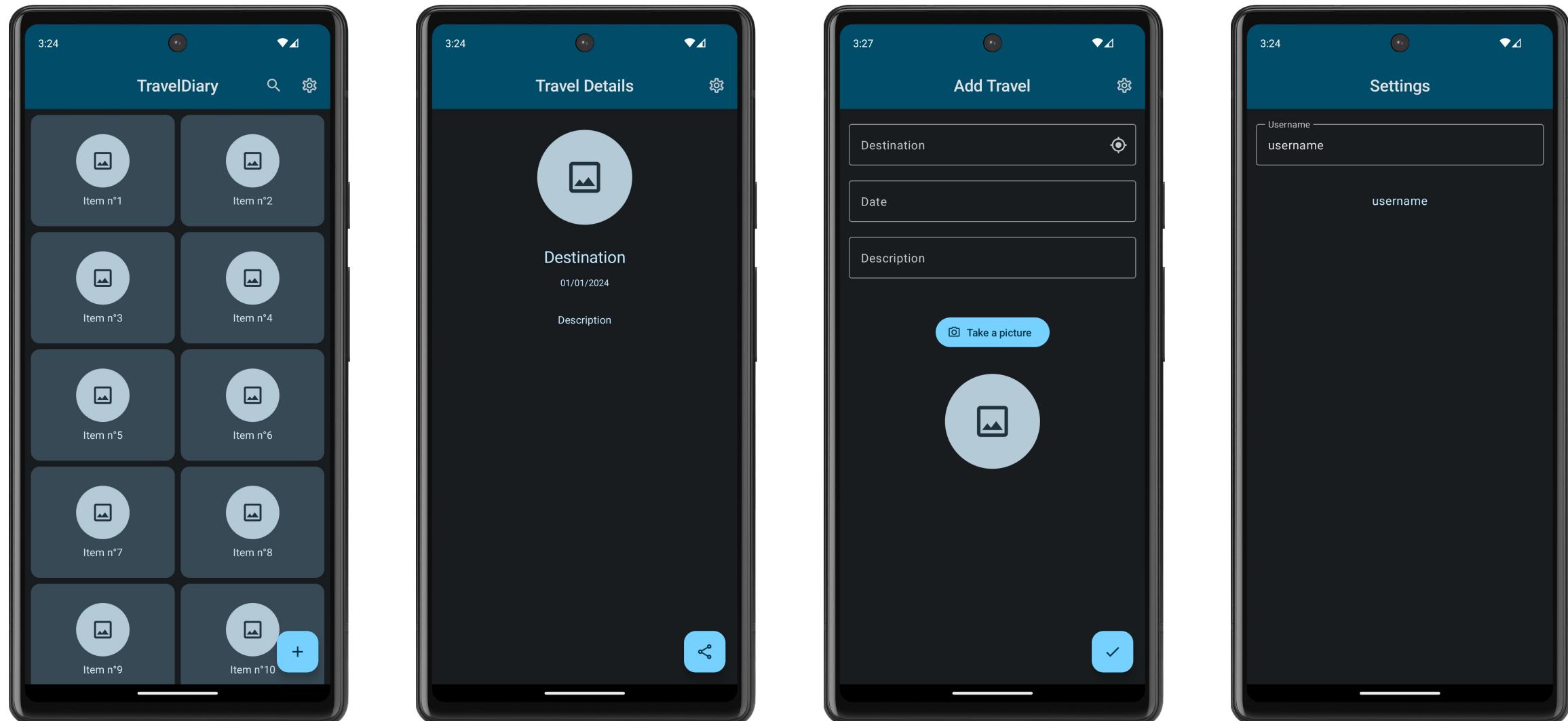
- Aggiungere un package **ui.screens**, con al suo interno i file:
 - **HomeScreen.kt**
 - **TravelDetailsScreen.kt**
 - **AddTravelScreen.kt**
 - **SettingsScreen.kt**
- Fare riferimento agli screenshot delle prossime slide per il design e ai riferimenti nell'ultima slide per suggerimenti sui componenti da utilizzare
- Per utilizzare le icone di Material Design che compaiono negli screenshot, aggiungere la seguente riga al blocco **dependencies** del file **build.gradle.kts** (modulo :app)

```
implementation("androidx.compose.material:material-icons-extended")
```

4. TravelDiary – Tema chiaro



4. TravelDiary – Tema scuro



Riferimenti

- Componenti di Material Design 3
<https://m3.material.io/components>
- App bars
<https://developer.android.com/jetpack/compose/components/app-bars>
- Card
<https://developer.android.com/jetpack/compose/components/card>
- Images
<https://developer.android.com/jetpack/compose/graphics/images>
- Text fields
<https://developer.android.com/jetpack/compose/text/user-input>