Leibniz Universität Hannover

Mobile Interaction

Android: App Architecture

Human-Computer Interaction Group

Prof. Dr. Michael Rohs
michael.rohs@hci.uni-hannover.de

MODIFIER.WEIGHT

Modifier.weight

```
Row(
    modifier = modifier.fillMaxWidth(),
  ) {
    Text(text = "1", modifier = modifier.weight(1f).aspectRatio(2f), textAlign = TextAlign.Center)
    Text(text = "2", modifier = modifier.weight(2f), textAlign = TextAlign.Center)
    Text(text = "3", modifier = modifier.weight(1f), textAlign = TextAlign.Center)
    Text(text = "4")
  }
```

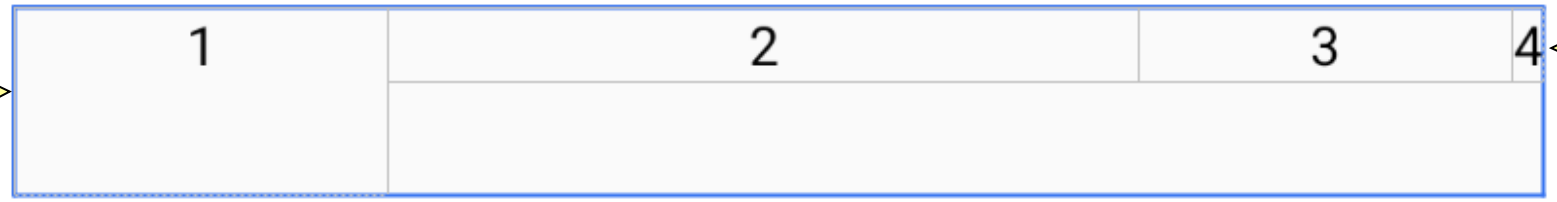
sum of weights = 4

width = 1/4

width = 2/4

width = 1/4

"1": aspect ratio width:height = 2:1



"4" width and height = wrap content

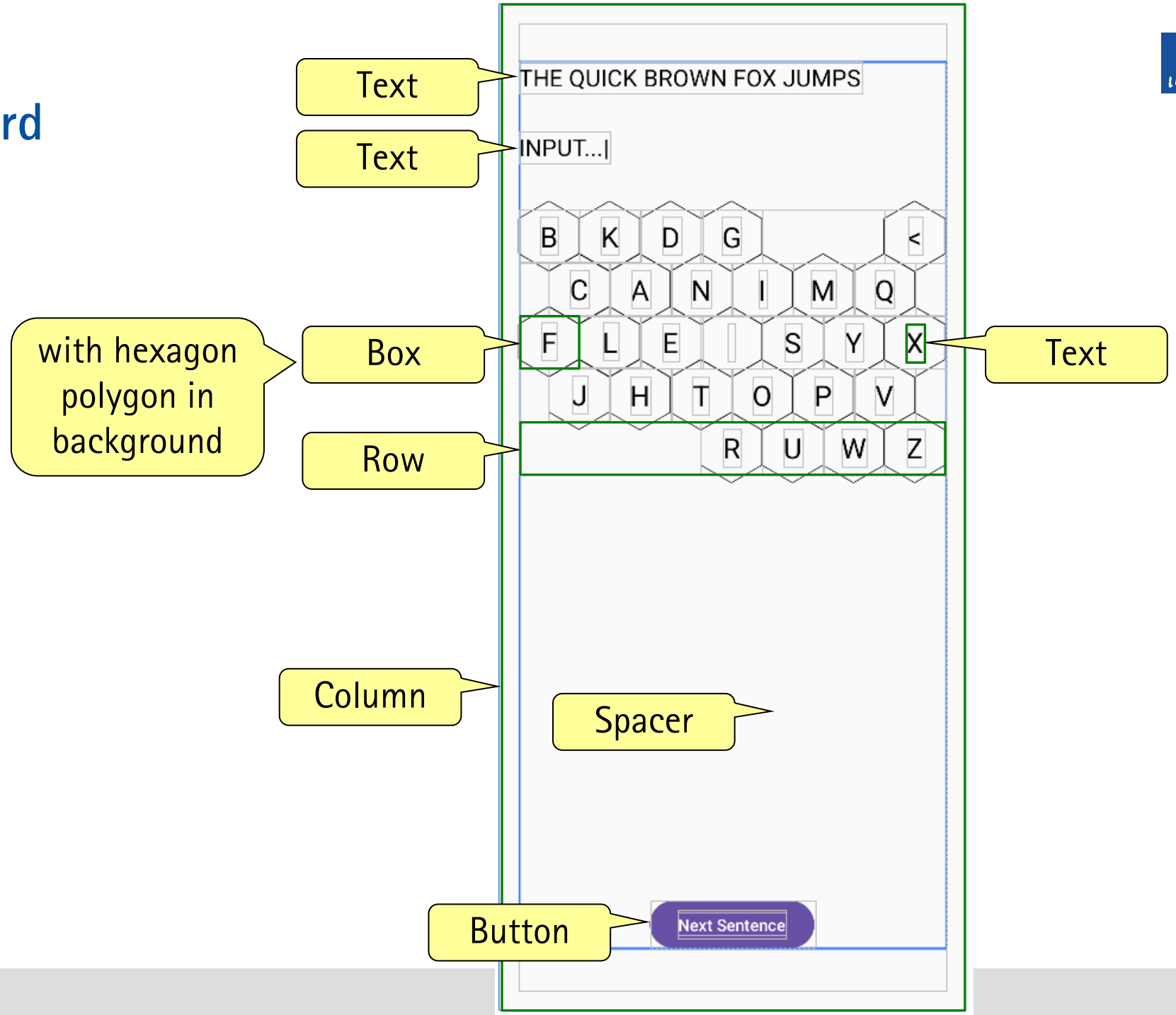
Modifier.weight and RowScope

```
Row(
        modifier = modifier.fillMaxWidth(),
    ) { // this: RowScope
        Text(text = "1", modifier = modifier.weight(1f).aspectRatio(2f), textAlign = TextAlign.Center)
        Text(text = "2", modifier = modifier.weight(2f), textAlign = TextAlign.Center)
        Text(text = "3", modifier = modifier.weight(1f), textAlign = TextAlign.Center)
        Text(text = "4")
    }
```

```
@Composable
inline fun Row(
    modifier: Modifier = Modifier,
    ...
    content: @Composable RowScope.() -> Unit
) {...}
```

```
interface RowScope {
    ...
    fun Modifier.weight(
        weight: Float,
        fill: Boolean = true
    ): Modifier
    ...
}
```

AtomikKeyboard

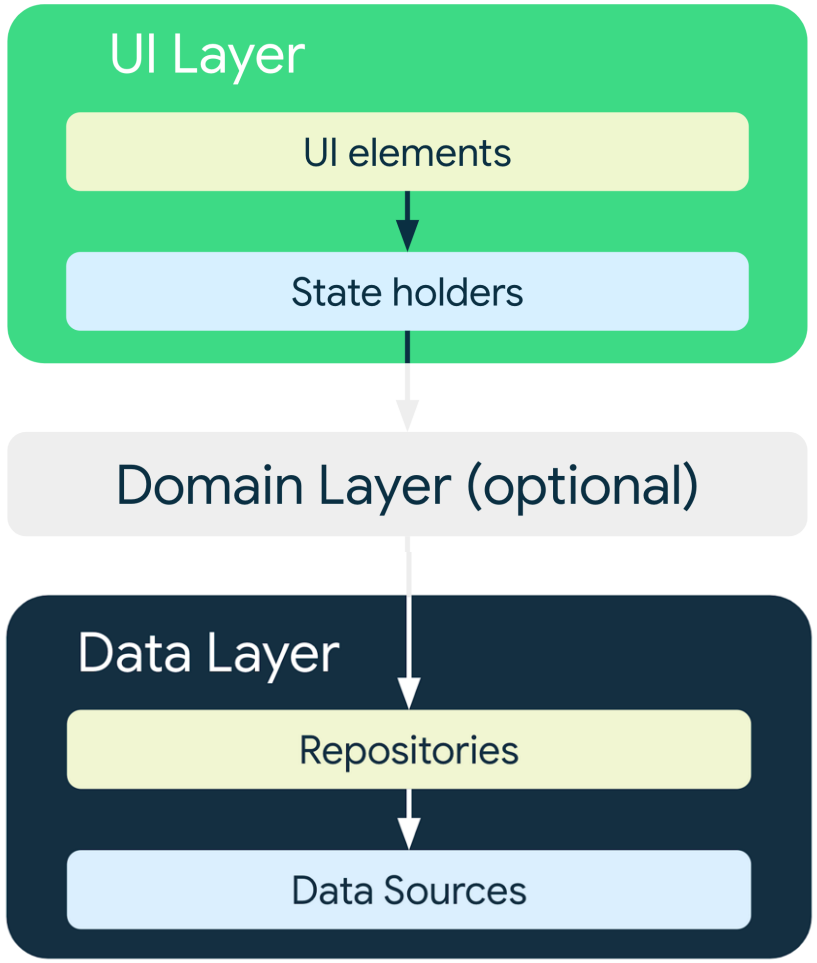


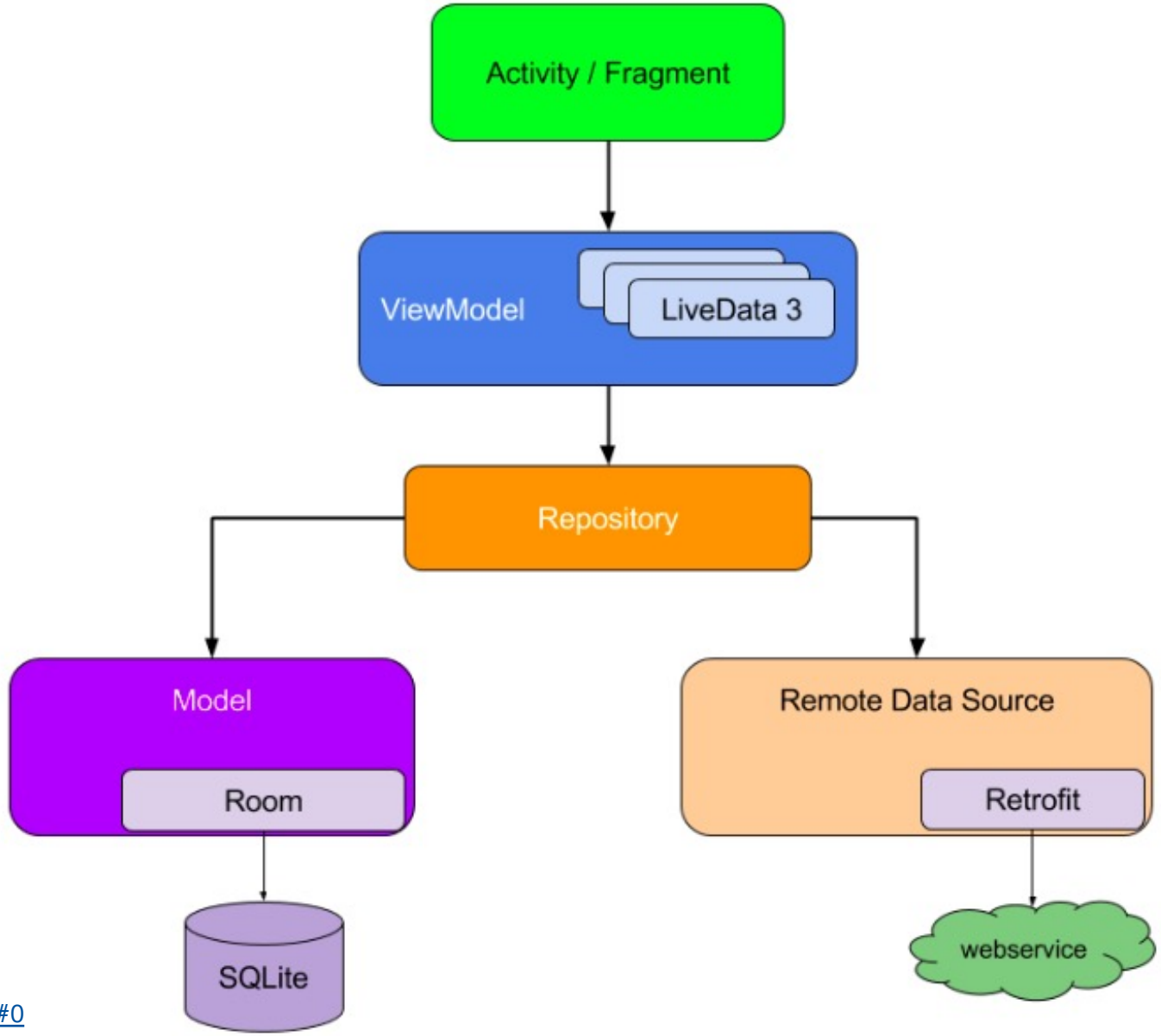
APP ARCHITECTURE CASE STUDY: TODOAPP

Learning Questions

- How to structure a larger app
  - Data layer: repository (interface), SQLite database (data source implementation)
  - UI layer: UI elements (view), state holders (view model)
- How to navigate between multiple pages
  - NavHost, NavHostController

Android App Architecture



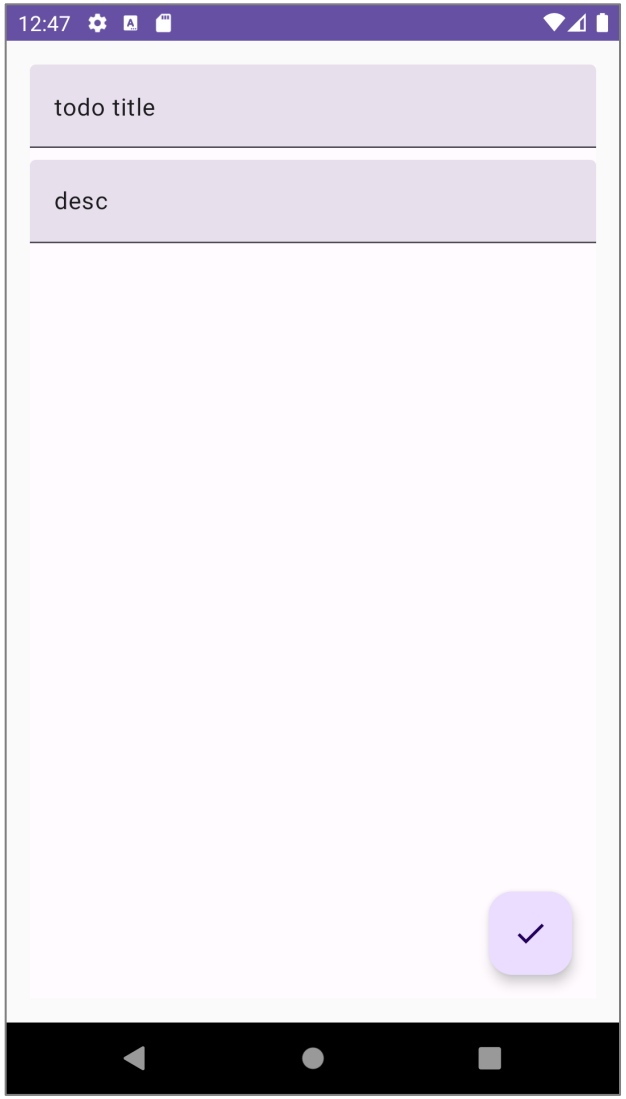


https://developer.android.com/topic/architecture
https://developer.android.com/codelabs/basic-android-kotlin-training-repository-pattern#0

Stud.IP: MI-04-TodoApp.zip

App Architecture Case Study: TodoApp

- manifests
  - AndroidManifest.xml
- java
  - de.luh.hci.mi.todoapp
    - data — data layer
      - Todo
      - TodoDatabase
      - TodoRepository
      - TodoRepositoryImpl
    - ui — UI layer
      - add_edit
        - AddEditTodoScreen.kt
        - AddEditTodoViewModel
      - list
        - TodoItem.kt
        - TodoListScreen.kt
        - TodoListViewModel
      - theme
      - MainActivity — navigation
      - Routes
      - TodoApp — data model instances



TodoApp: Data Layer

```
data class Todo(
    val title: String,
    val description: String,
    val isDone: Boolean,
    val id: Int? = null // primary key for database
)


interface TodoRepository {
    suspend fun insertTodo(todo: Todo)
    suspend fun deleteTodo(todo: Todo)
    suspend fun getTodoById(id: Int): Todo?
    fun getTodos(): Flow<List<Todo>>
}
```

```
class TodoRepositoryImpl(
    private val db: TodoDatabase
) : TodoRepository {
    override suspend fun insertTodo(todo: Todo) {
        db.insertTodo(todo)
    }
    override suspend fun deleteTodo(todo: Todo) {
        db.deleteTodo(todo)
    }
    override suspend fun getTodoById(id: Int): Todo? {
        return db.getTodoById(id)
    }
    override fun getTodos(): Flow<List<Todo>> {
        return db.getTodosAsFlow()
    }
}
```

TodoApp: Data Layer

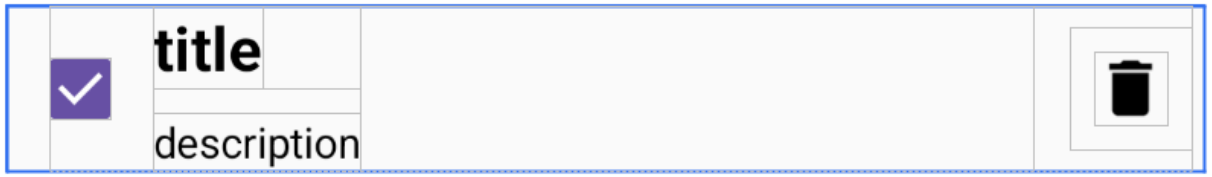
```
class TodoApp : Application() {
    private lateinit var db: TodoDatabase
    lateinit var repository: TodoRepository
        private set

    override fun onCreate() {
        super.onCreate()
        val databaseFile = applicationContext.getDatabasePath("todos.db")
        db = TodoDatabase(databaseFile)
        repository = TodoRepositoryImpl(db)
    }

}
```

TodoApp: UI Layer

```
// A screen to add a new todo item
// or to edit an existing one.
@Composable
fun AddEditTodoScreen(
    navigateBack: () -> Unit,
    viewModel: AddEditTodoViewModel
) {...}
```

```
// ViewModel for adding new todo items
// and for editing todo items.
class AddEditTodoViewModel(
    private val repository: TodoRepository,
    savedStateHandle: SavedStateHandle
) : ViewModel() {...}
```

```
// A single todo item.
@Composable
fun TodoItem(
    todo: Todo,
    deleteTodo: (Todo) -> Unit,
    todoDone: (todo: Todo, isDone: Boolean) -> Unit,
    modifier: Modifier = Modifier
) {...}
```



TodoApp: UI Layer

```
// A screen with a list of todo items.
@Composable
fun TodoListScreen(
    onNavigate: (route: String) -> Unit,
    viewModel: TodoListViewModel
) {...}


// ViewModel for showing a list of todo items.
class TodoListViewModel(
    private val repository: TodoRepository,
) : ViewModel() {...}
```

coordinates typical parts of an app screen (floating action button, bottom bar, etc.)

```
Scaffold(
    snackbarHost = {...},
    modifier = Modifier...,
    floatingActionButton = {
        FloatingActionButton(onClick = {...}) {
            Icon(
                imageVector = Icons.Default.Check,
                contentDescription = "Save"
            )
        }
    }
) { Column(
        modifier = Modifier
            .fillMaxSize()
            .padding(paddingValues),
    ) {...}
}
```

a popup message

main button

main content

TodoApp: Navigation

```
object Routes {
    const val TODO_LIST = "todo_list"
    const val ADD_EDIT_TODO = "add_edit_todo"
}
```

```
class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            TodoAppTheme {
                val navController = rememberNavController()
                NavHost(
                    navController = navController,
                    startDestination = Routes.TODO_LIST
                ) {
                    // navigation destination 1
                    // navigation destination 2
                }
            }
        }
    }
}
```

```
// navigation destination 1
composable(route = Routes.TODO_LIST) {
    TodoListScreen(
        navController::navigate,
        viewModel(factory = TodoListViewModel.Factory)
    )
}
// navigation destination 2
composable(
    route = Routes.ADD_EDIT_TODO + "?todoId={todoId}",
    arguments = listOf(navArgument("todoId") {...})
) {
    AddEditTodoScreen(
        navController::popBackStack,
        viewModel(factory = AddEditTodoViewModel.Factory)
    )
}
```