



Android Crew

16 – 20 сентября

Автоматизация разработки

Евгений Мельцайкин

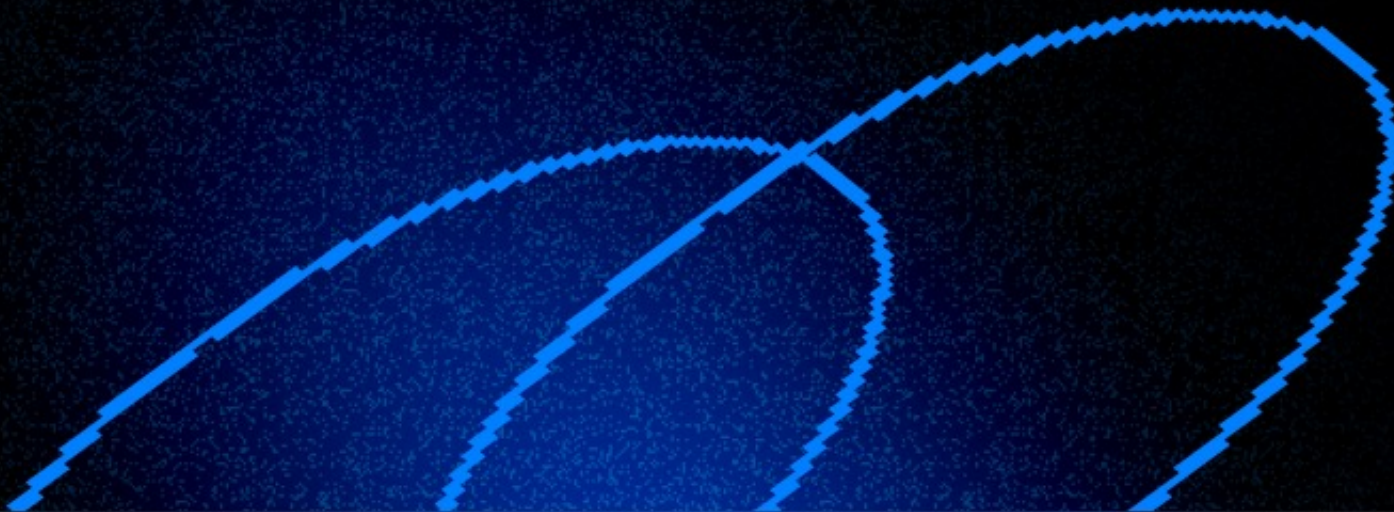
Android-разработчик, СКБ Контур

Доклад:

Генерация шаблонного кода
с помощью Gemini



Генерация шаблонного кода с помощью **GEMINIO**



Контур

Евгений Мельцайкин
Старший инженер-программист

Обо мне

- В коммерческой Android разработке с 2021 года
- Пишу статьи на Хабр
- Спикер Mobile Update



Евгений Мельцайкин



e.meltsaykin

План доклада



Предыстория



Возможные
решения



Geminio

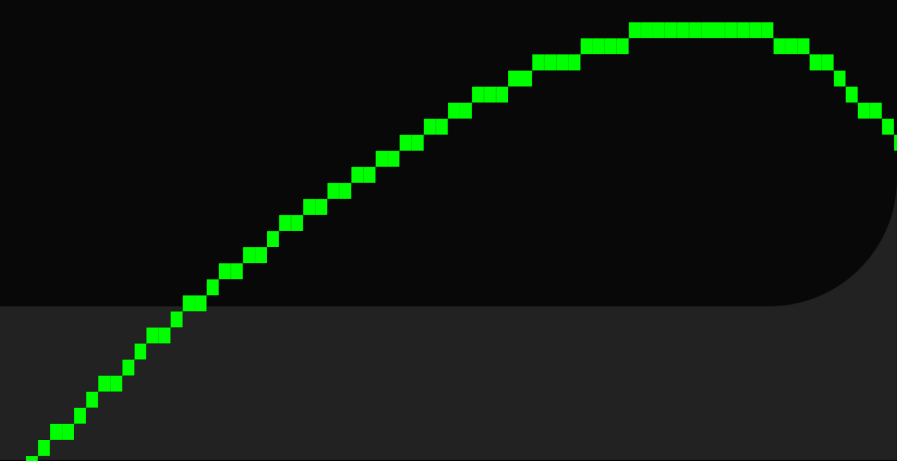


Практика

Проблематика

1. Создание новых фич занимает много времени
2. Копирование шаблонного кода влечет опечатки или ошибки

Шаблонизировать код
можно на **любой** стадии
проекта



предыстория

2

3

4



Контур.Диадок: Логистика

6

Погрузка **3**

В пути

Выгрузка

НА ПОДПИСЬ

G9280654-427

ООО "АгроПромТехника-Восток"

г. Санкт-Петербург, ул.Савушкина, д. 126,
лит.Б, пом.62-Н

23 мая 12:00 → 24 мая 14:30

НА ОТПРАВИТЕЛЕ

G7624550191-664

ООО "Сельхозснаб-ВетСтандарт"

г. Санкт-Петербург,
ул.Христорождественская, д. 152а, корп 3

23 мая 13:00 → 24 мая 18:45

НА ПЕРЕВОЗЧИКЕ

G6500928-515

ООО "АсторисТрансКорп-Инвест"

г. Санкт-Петербург, ул.Машинная-2, д. 4,
лит.Б, корп.2

23 мая 14:30 → 24 мая 23:30



Накладная



Филимонов Александр Геннадьевич
Водитель

G928654-427

16 ноября 2023

ПОГРУЗКА



О грузе



Замечания



Контакты



Груз

1864 кг, 96 мест



Температура

от -40°C до +20°C



Влажность

25% – 45%

Маршрут



Принять





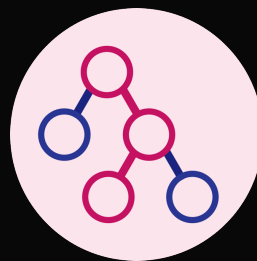
Контур.Диадок: Логистика

Kotlin
Multiplatform



Контур.Диадок: Логистика

Kotlin
Multiplatform

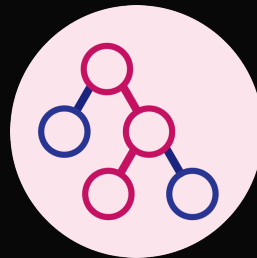


Decompose

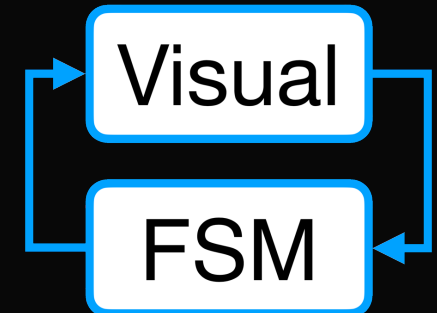


Контур.Диадок: Логистика

Kotlin
Multiplatform



Decompose



предыстория

2

3

4

10

UiState

```
3 3 ↓ sealed class LogisticsFeatureUiState{  
4      data object Loading: LogisticsFeatureUiState()  
5      data object Loaded:  LogisticsFeatureUiState()  
6  }
```

Decompose Component

```
6 ① ↓ interface LogisticsFeatureComponent {
7 ① ↓     val state: AnyStateFlow<LogisticsFeatureUiState>
8      }
```

предыстория

2

3

4

12

Component
Dependencies

```
3 class |LogisticsFeatureDependencies()
```

VFSM State

```
6  ↓ internal sealed class LogisticsFeatureState : State {  
7  
8      data object Loading: LogisticsFeatureState()  
9  
10     companion object {  
11         fun initial(): LogisticsFeatureState {  
12             return Loading  
13         }  
14     }  
15 }
```

предыстория

2

3

4

14

VFSM
BaseAction



```
5  
6 internal sealed class BaseLogisticsFeatureAction : Action<LogisticsFeatureState>()
```


VFSM AsyncWorker

```
8  internal class LogisticsFeatureAsyncWorker() : AsyncWorker<LogisticsFeatureState, BaseLogisticsFeatureAction>() {
9      override fun onNextState(state: LogisticsFeatureState): AsyncWorkerTask<LogisticsFeatureState> {
10         return when (state) {
11             else -> AsyncWorkerTask.Cancel()
12         }
13     }
14 }
```

VFSM Feature

```
9  @GenerateTransitionsFactory
10  internal class LogisticsFeatureFeature(
11      ⚡ initialState: LogisticsFeatureState,
12      private val asyncWorker: LogisticsFeatureAsyncWorker,
13  ) : InstanceKeeper.Instance, Feature<LogisticsFeatureState, BaseLogisticsFeatureAction> {
14      initialState = initialState,
15      asyncWorker = asyncWorker,
16      transitionsFactory = GeneratedLogisticsFeatureFeatureTransitionsFactory()
17  } {
18      ⚡ override fun onDestroy() {
19          |         asyncWorker.unbind()
20      }
21  }
```



Di Module

```
7  
8 internal val LogisticsFeatureModule = module { this: Module  
9     factoryOf(::LogisticsFeatureAsyncWorker)  
10     factory { initialState -> LogisticsFeatureFeature(initialState = initialState.get(), asyncWorker = get()) }  
11 }
```

Default Decompose Component

```
23 internal class DefaultLogisticsFeatureComponent(  
24     componentContext: ComponentContext,  
25     private val dependencies: LogisticsFeatureDependencies,  
26 ) : LogisticsFeatureComponent,  
27     ComponentContext by componentContext,  
28     CancellableCoroutineScope by componentContext.cancelableCoroutineScope() {  
29  
30     private val koinContext = instanceKeeper.getOrCreate { ComponentKoinContext() }  
31  
32     private val scope = koinContext.getOrCreateKoinScope(  
33         listOf(  
34             LogisticsFeatureModule,  
35             module { this: Module  
36  
37             }  
38         )  
39     )  
40  
41     private val initialState = LogisticsFeatureState.initial()  
42  
43     private val feature = instanceKeeper.getOrCreate {  
44         scope.get<LogisticsFeatureFeature>(parameters = { parametersOf(initialState) })  
45     }  
46  
47     override val state: AnyStateFlow<LogisticsFeatureUiState> = feature.observeState() StateFlow<L  
48         .map { it.toUiState() } Flow<LogisticsFeatureUiState>  
49         .stateIn(  
50             scope = coroutineScope,  
51             started = SharingStarted.Eagerly,  
52             initialValue = initialState.toUiState()
```

UiState

VFSM
State

VFSM
Feature

Compose Screen

Decompose
Component

VFSM
BaseAction

Di Module

Component
Dependencies

VFSM
AsyncWorker

Default
Decompose
Component

Требования к решению

1. Возможность создания нескольких файлов в рамках одного шаблона
2. Возможность создания модулей

Свое решение

Плюсы

- поддержка на нашей стороне
- независимость

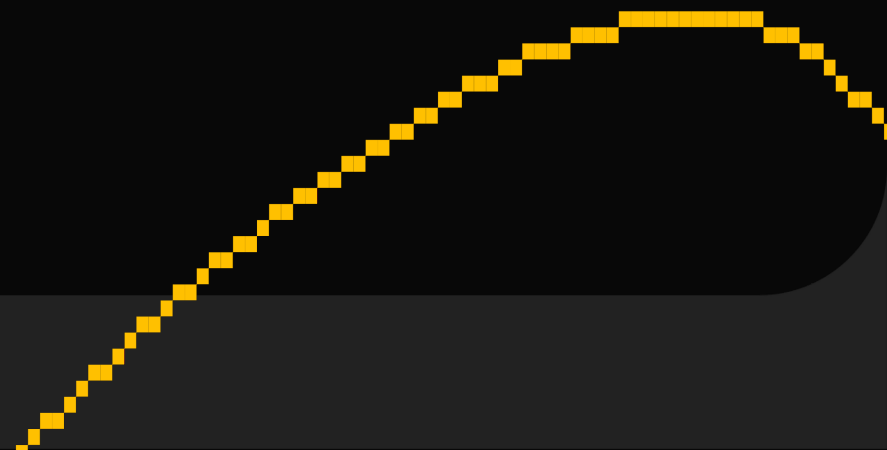
Минусы

- поддержка на нашей стороне
- долгая разработка на старте

Готовые решения

- Live template в Android Studio
- screen-generator-plugin
- Geminio (HH)

Geminio



Конфигурация плагина

1. Создать файл конфигурации плагина `geminioConfig.yaml`

GeminioConfig.yaml

```
templatesRootDirPath: /geminio/templates
modulesTemplatesRootDirPath: /geminio/modulesTemplates


groupsNames:
  forNewGroup: Podłodka Templates
  forNewModulesGroup: Podłodka Modules
```

Конфигурация плагина

1. Создать файл конфигурации плагина `geminioConfig.yaml`
2. В плагине `Geminio` указать путь до файла конфигурации

Конфигурация плагина

Settings

Appearance & Behavior > Geminio plugin 

Config file path:

Templates paths

Templates:

Modules templates:

Groups names

Templates group:

Modules templates group:

Устройство рецепта

1. requiredParams
2. widgets
3. globals
4. recipe

requiredParams

`requiredParams:`

`name:` Podlodka data, domain layer feature

`description:` Генерирует объекты для data и domain слоя

widgets

```
widgets:
  - stringParameter:
    id: feature
    name: Имя фичи
    constraints:
      - class
      - nonempty
      - unique
    default: FeatureName

  - booleanParameter:
    id: includeNetwork
    name: Сгенерировать объекты для сети?
    help: Генерирует объекты для работы с сетью
    default: false
```

Globals

- `stringParameter:`
 - `id:` `serviceApiName`
 - `value:` `${feature}ServiceApi`
- `stringParameter:`
 - `id:` `defaultServiceApiName`
 - `value:` `Default${feature}ServiceApi`
- `booleanParameter:`
 - `id:` `isTestingTemplate`
 - `value:` `true`

recipe

```
- instantiateAndOpen:  
  from: domain/repository/Repository.kt.ftl  
  to:  ${srcOut}/domain/repository/${repositoryName}.kt
```

recipe

```
- predicate:  
  validIf: ${includeMock}  
  commands:  
    - instantiate:  
      from: data/network/MockServiceApi.kt.ftl  
      to: ${srcOut}/data/network/Mock${serviceName}.kt  
  elseCommands:  
    - instantiate:  
      from: data/network/DefaultServiceApi.kt.ftl  
      to: ${srcOut}/data/network/${defaultServiceName}.kt
```

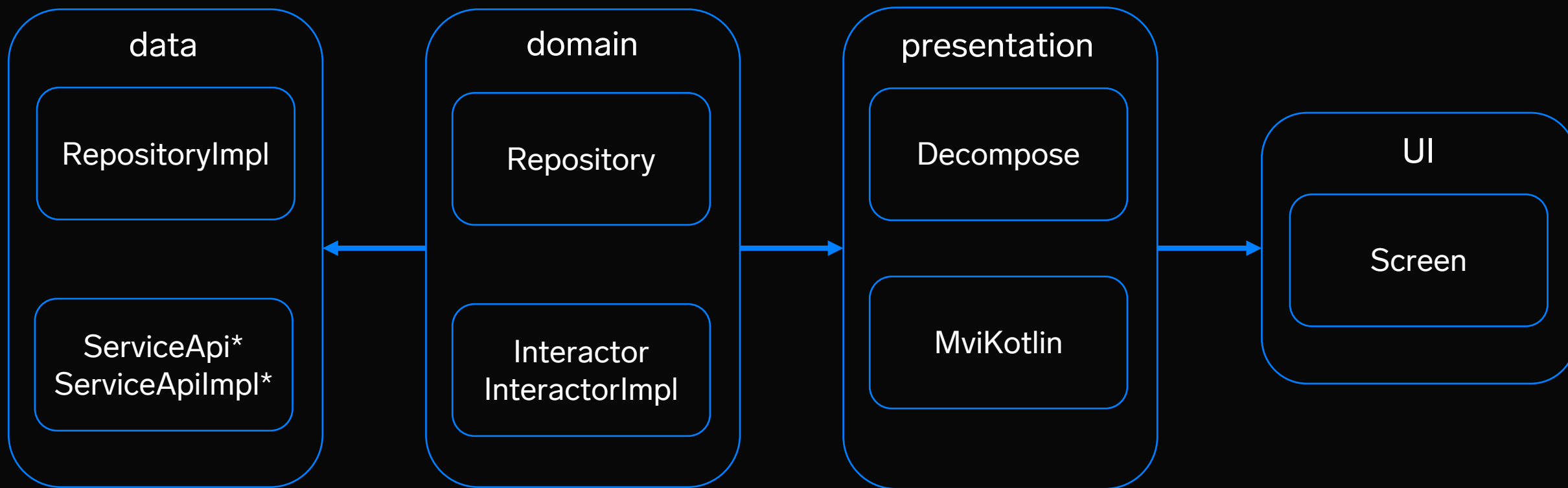
Возможности

1. Создание настраиваемых шаблонов файла
2. Создание настраиваемых шаблонов модулей
3. Создание неограниченного количества файлов в рамках одного шаблона

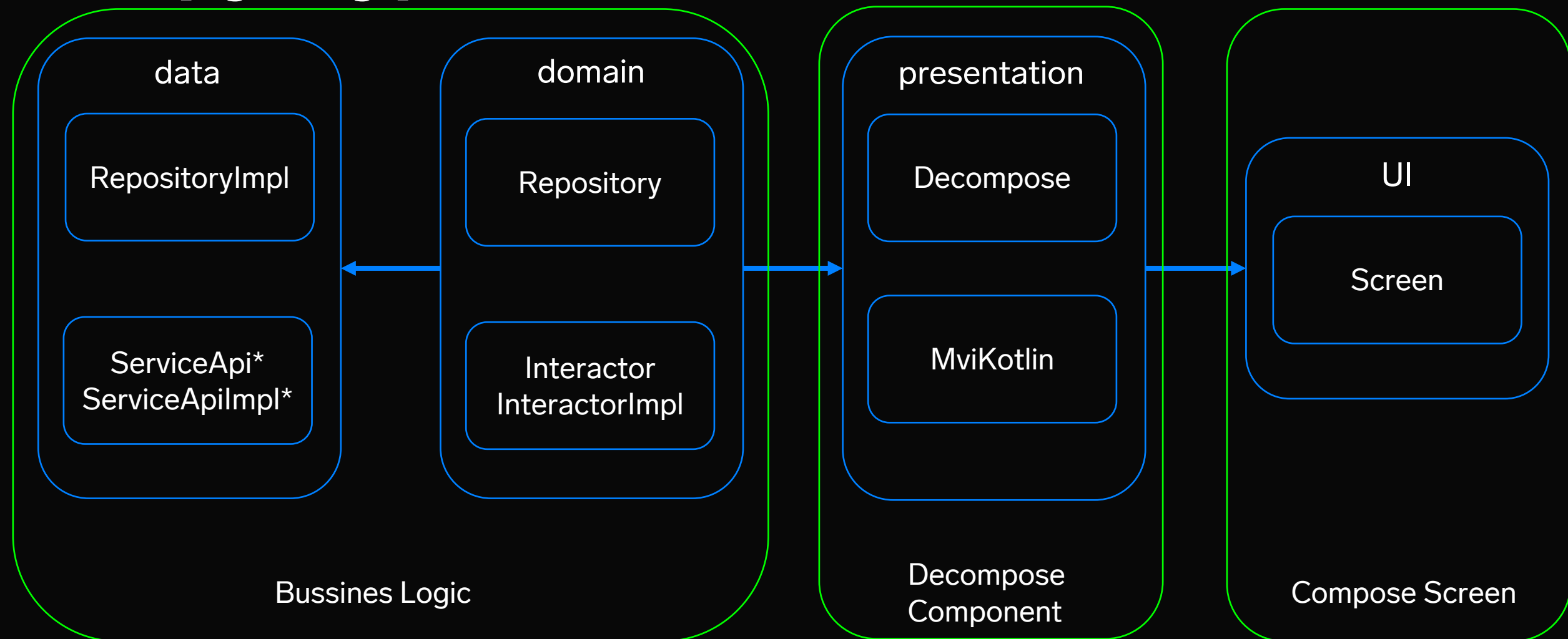
Написание шаблонов



Структура



Структура



Трудности

1. Плагин работает не всегда идеально
2. Генерация кода работает только в android Source set

Спасибо за внимание



Контур

Евгений Мельцайкин