Последнее слово в Android навигации

Mobius
2022 Autumn

Android Navigation at scale

Александр Митропольский

Сбер

Александр Макушев

Сбер

Mobius
2022 Autumn

Навигируемся в Jetpack Compose

Игорь Кареньков

hh.ru

Mobius
2023 Spring

Декларативная архитектура и навигация с Decompose

Алексей Панов

Контур

Mobius
2022 Autumn

Как избежать хаоса: навигация как отдельный логический уровень

Сергей Шардыко

Skyeng

```
// Базовые операции
startActivity()
finish()
onBackPressed()
// Работа со стеком
android:taskAffinity
android:launchMode
Intent.FLAG_ACTIVITY_NEW_TASK
Intent.FLAG_ACTIVITY_CLEAR_TOP
Intent.FLAG_ACTIVITY_SINGLE_TOP
```

Activity

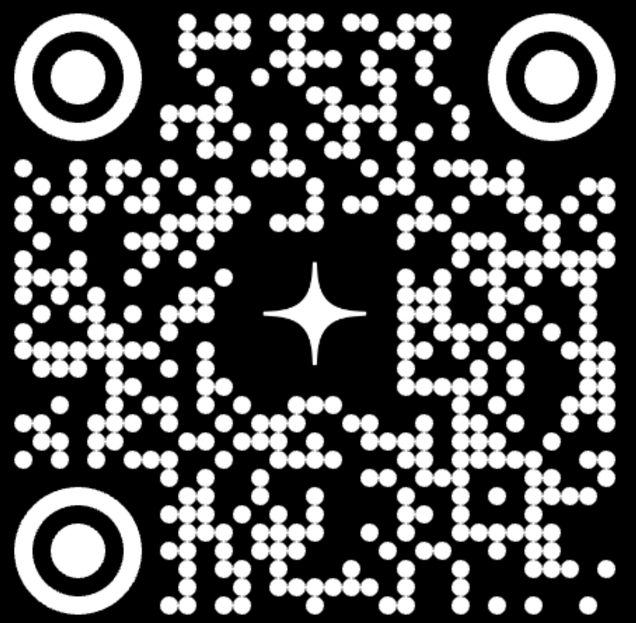
2008

```
// FragmentManager
getSupportFragmentManager()
getChildFragmentManager()
getParentFragmentManager()
// FragmentTransaction
replace() add() remove()
show() hide()
addToBackStack()
setReorderingAllowed()
commit()
commitAllowStateLoss()
```

Fragment

2012

```
// Commands
navigateTo() backTo()
replaceScreen()
newChain() newRootChain()
exit() finishChain()
// CommandBuffer
Roter
Navigator
```

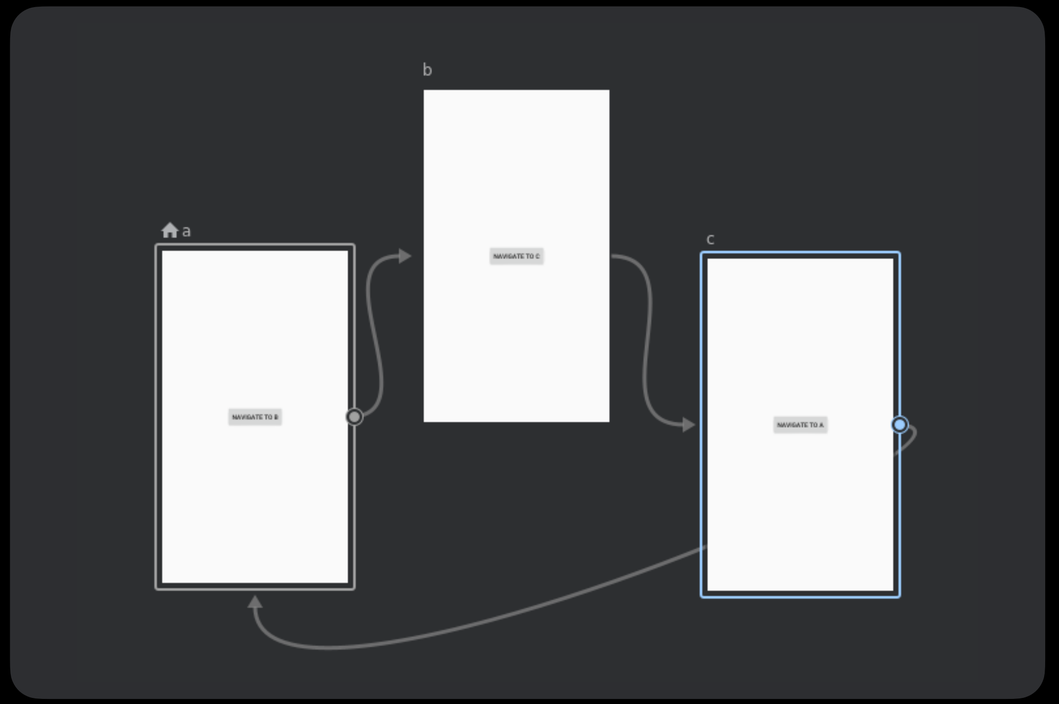


Cicerone — простая навигация в Андроид приложении

Single Activity & Cicerone

2016

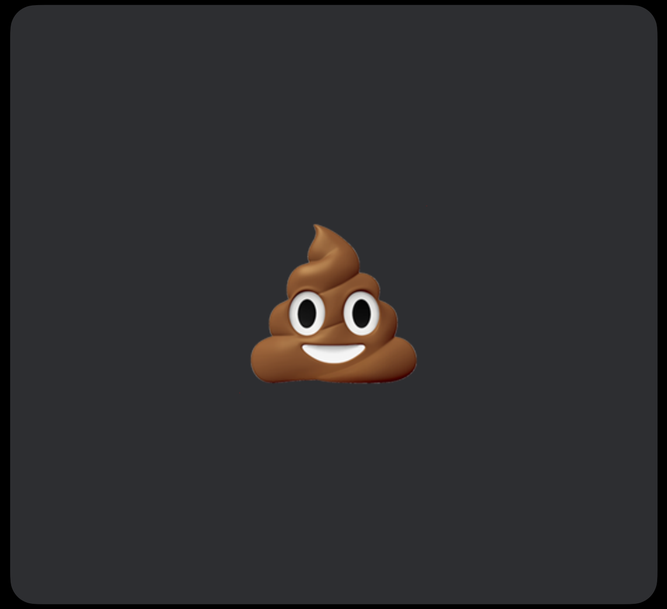
```
// NavigationController
navigate()
navigateUp()
findNavController()
// XML
<navigation/>
<fragment/>
<action/>
<argument/>
```



Navigation Graph

2018

Jetpack Compose Navigation



Compose Navigation

2020

Voyager
Decompose
Modo

Compose Navigation

2020

State

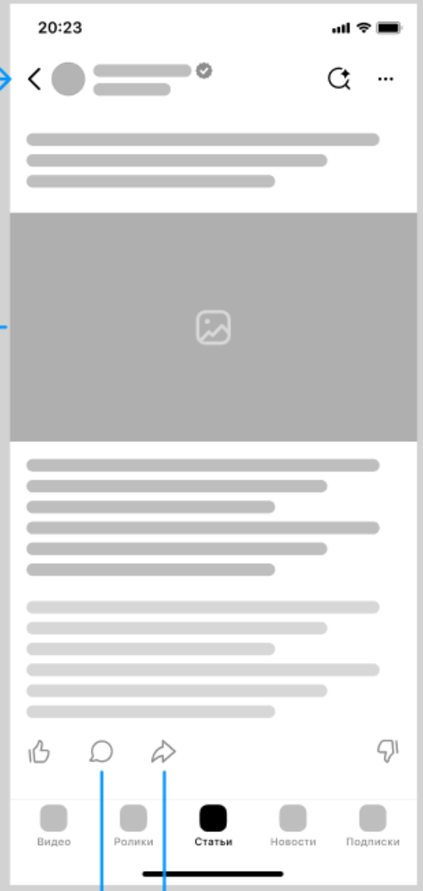
Stack

Screen

Навигация как бизнес-логика

Статья

3.1



Шэринг

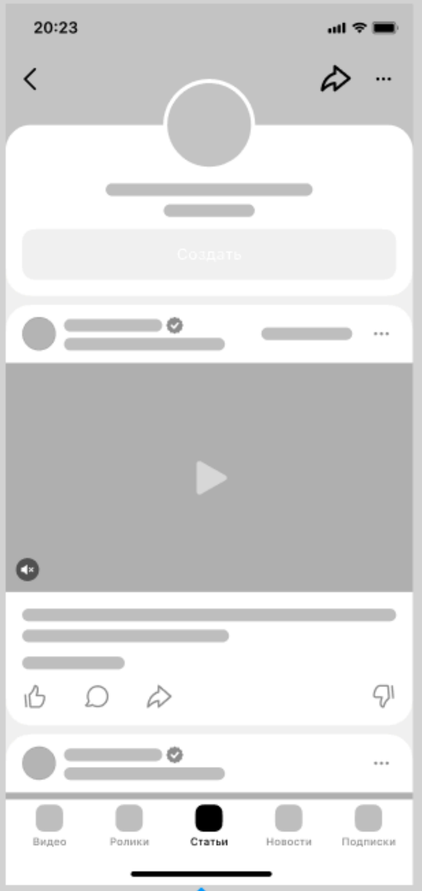
3.1

Комменты

3.2

Канал

3.3



Профиль

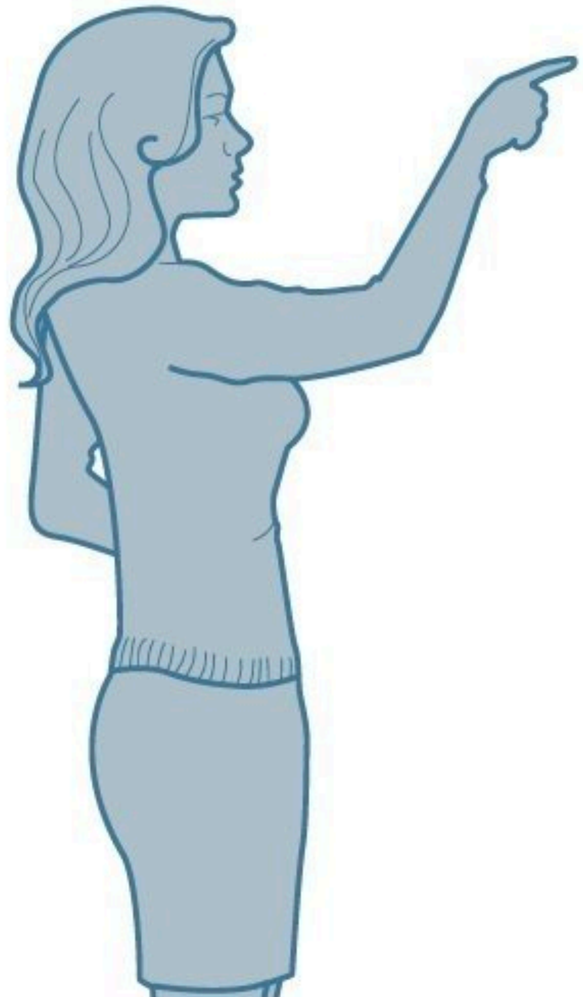
3.1

bottom sheet

bottom sheet

forward and backward

bottom sheet



Открыть

Закрыть

startActivity

showDialog

add

replace

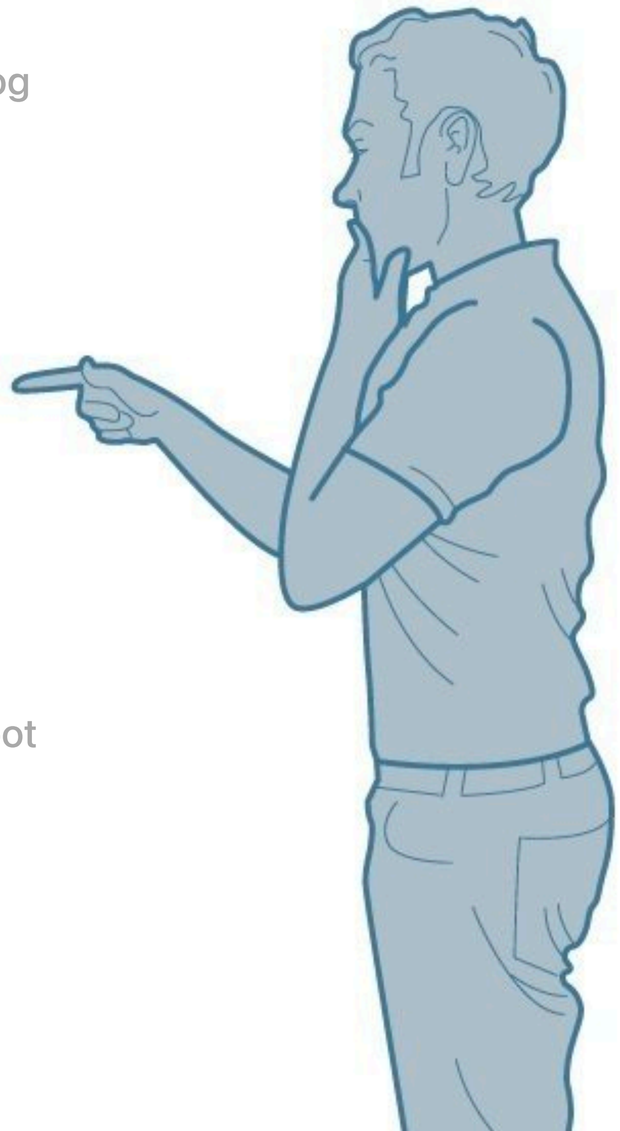
forward

remove

back

backToRoot

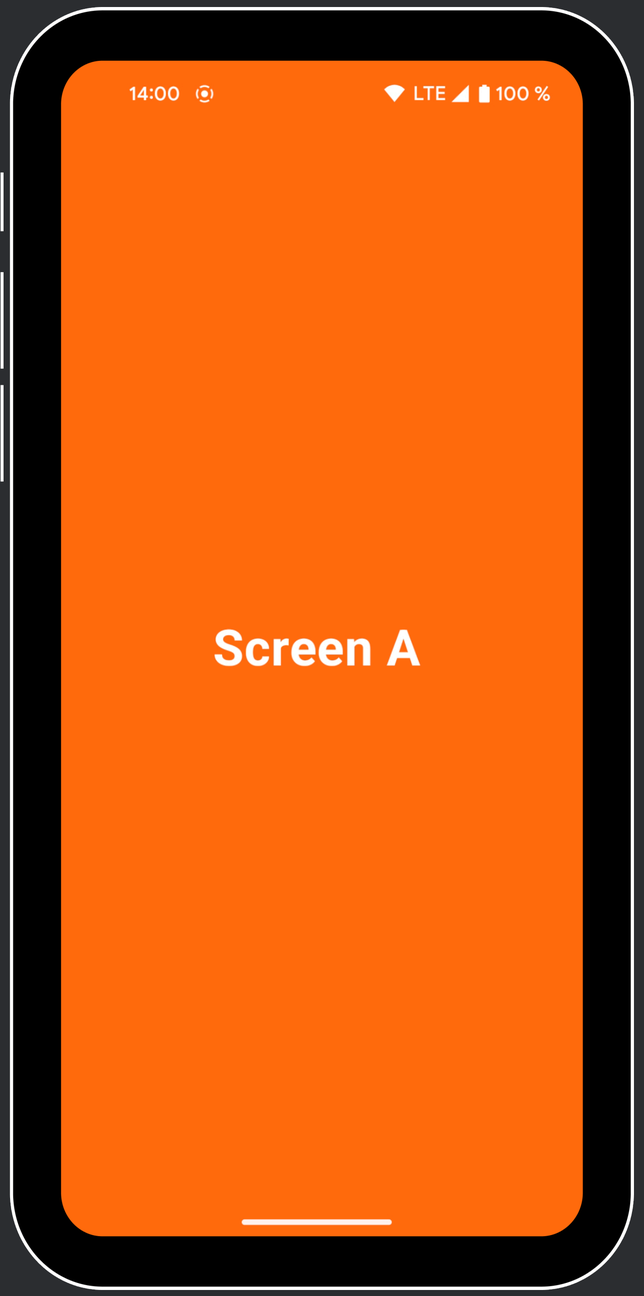
newRoot



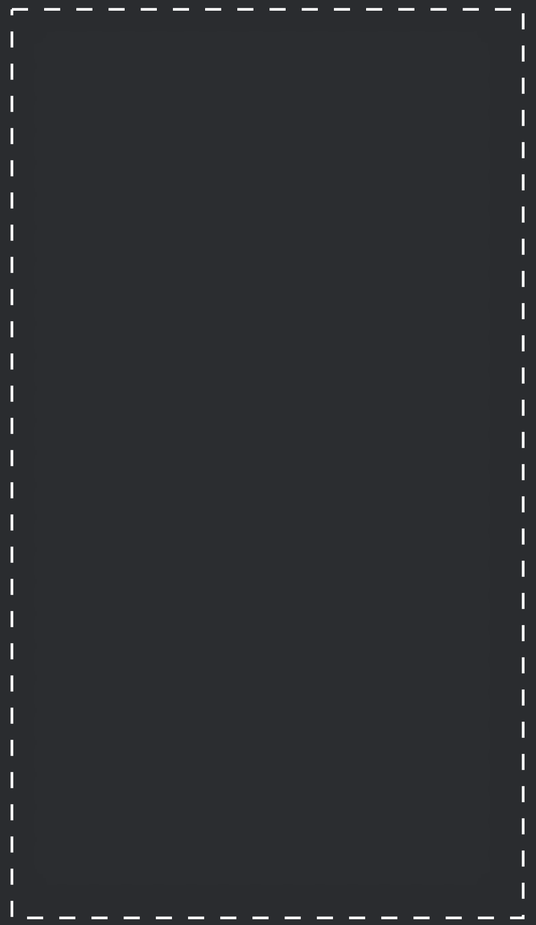
Две команды на всю навигацию

```
// Открыть экран
open(screen)

// Закрыть экран
close(screen)
```

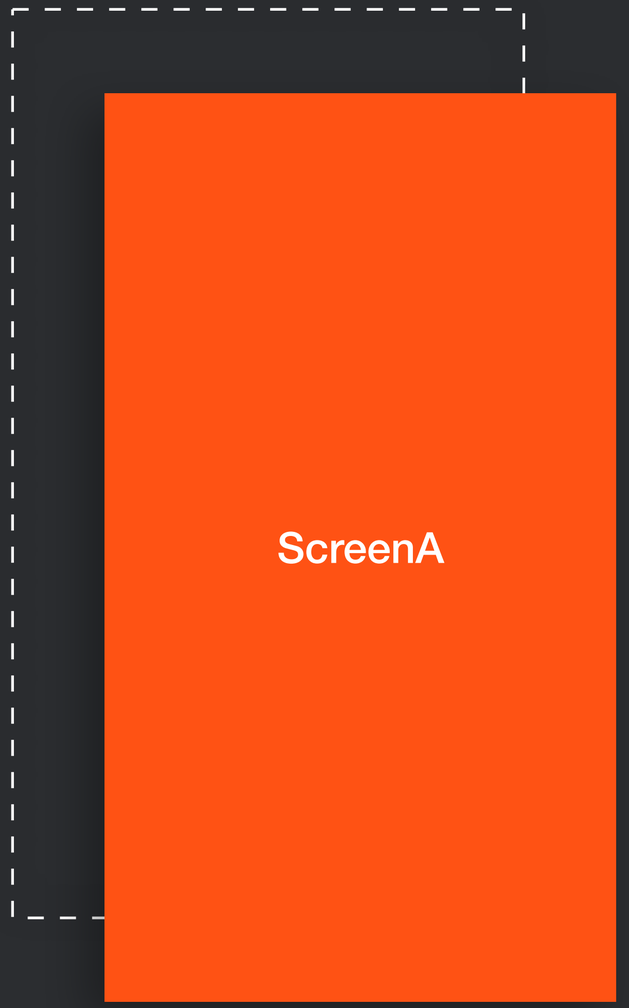
Концепция навигации


14:00
LTE
100 %
Screen A

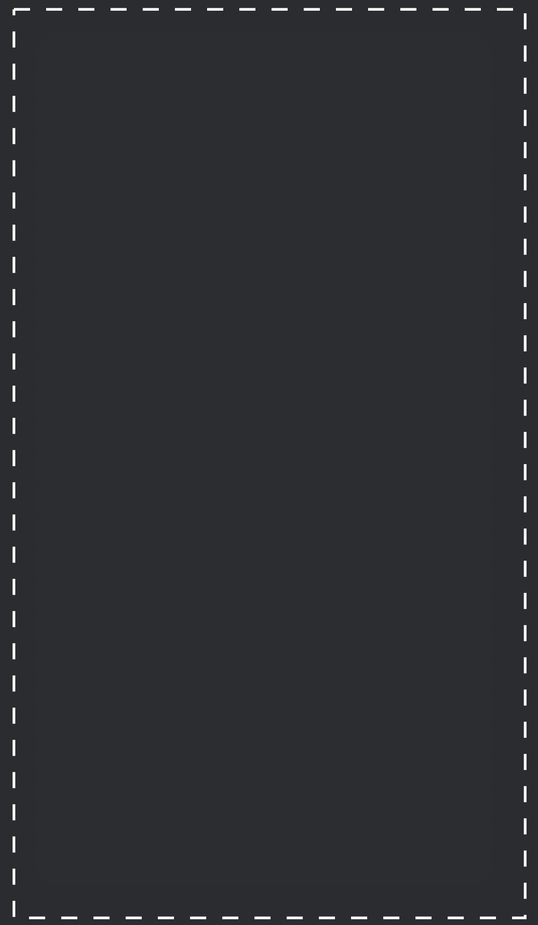
```
<FrameLayout
    android:id="@+id/host"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

</FrameLayout>
```



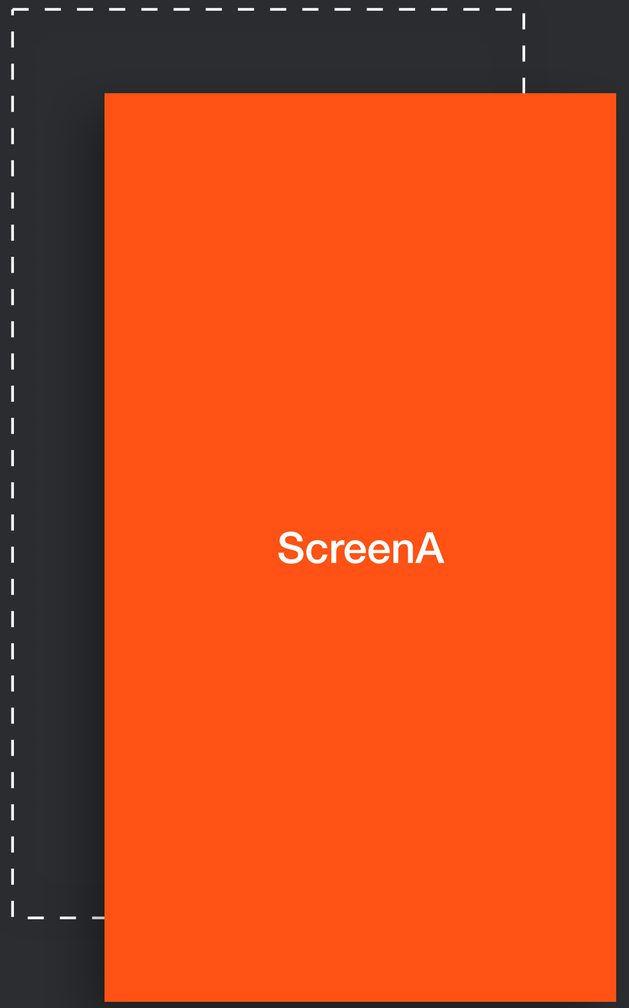
```
<FrameLayout
    android:id="@+id/host"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <FrameLayout
        android:id="@+id/screen_a"
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <TextView
            android:text="ScreenA"/>

    </FrameLayout>

</FrameLayout>
```

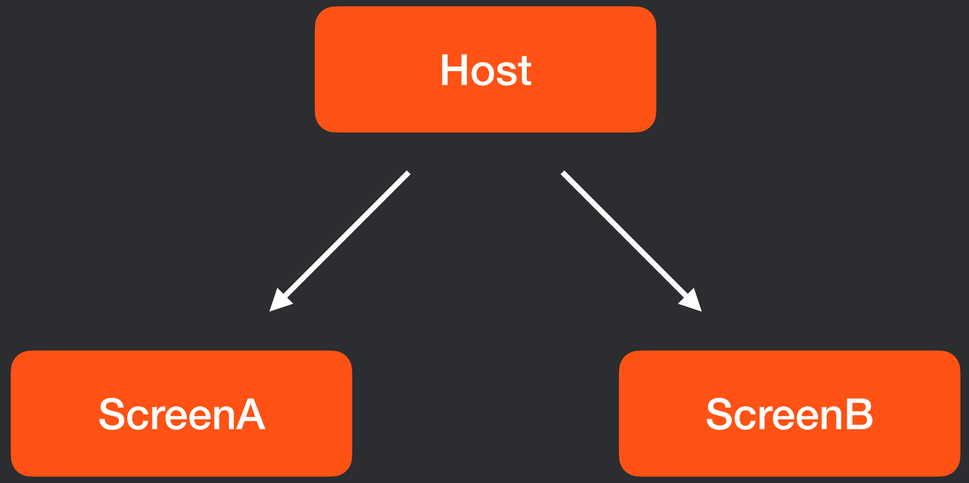


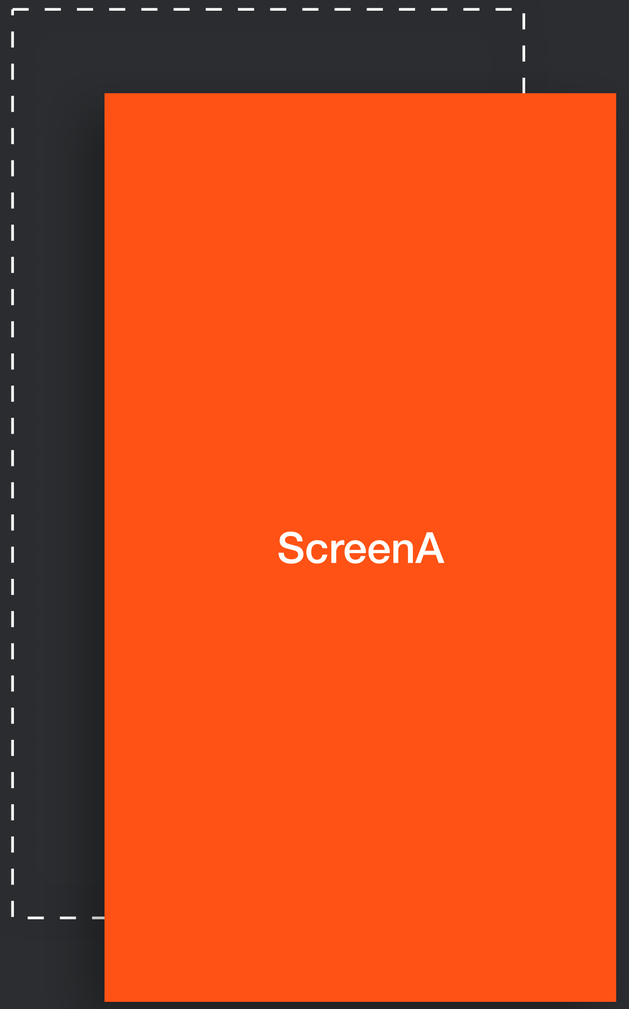
```
@Composable
fun App() {
      // Host
      Box {
         when (screenState) {

         }
    }
}
```

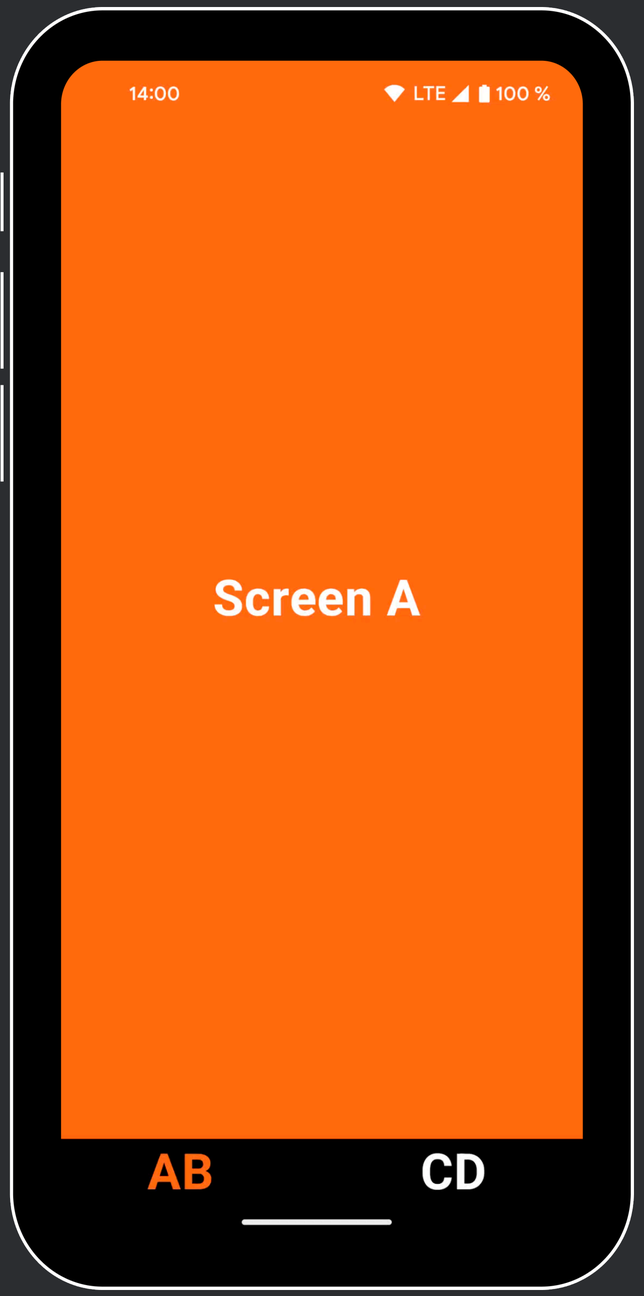


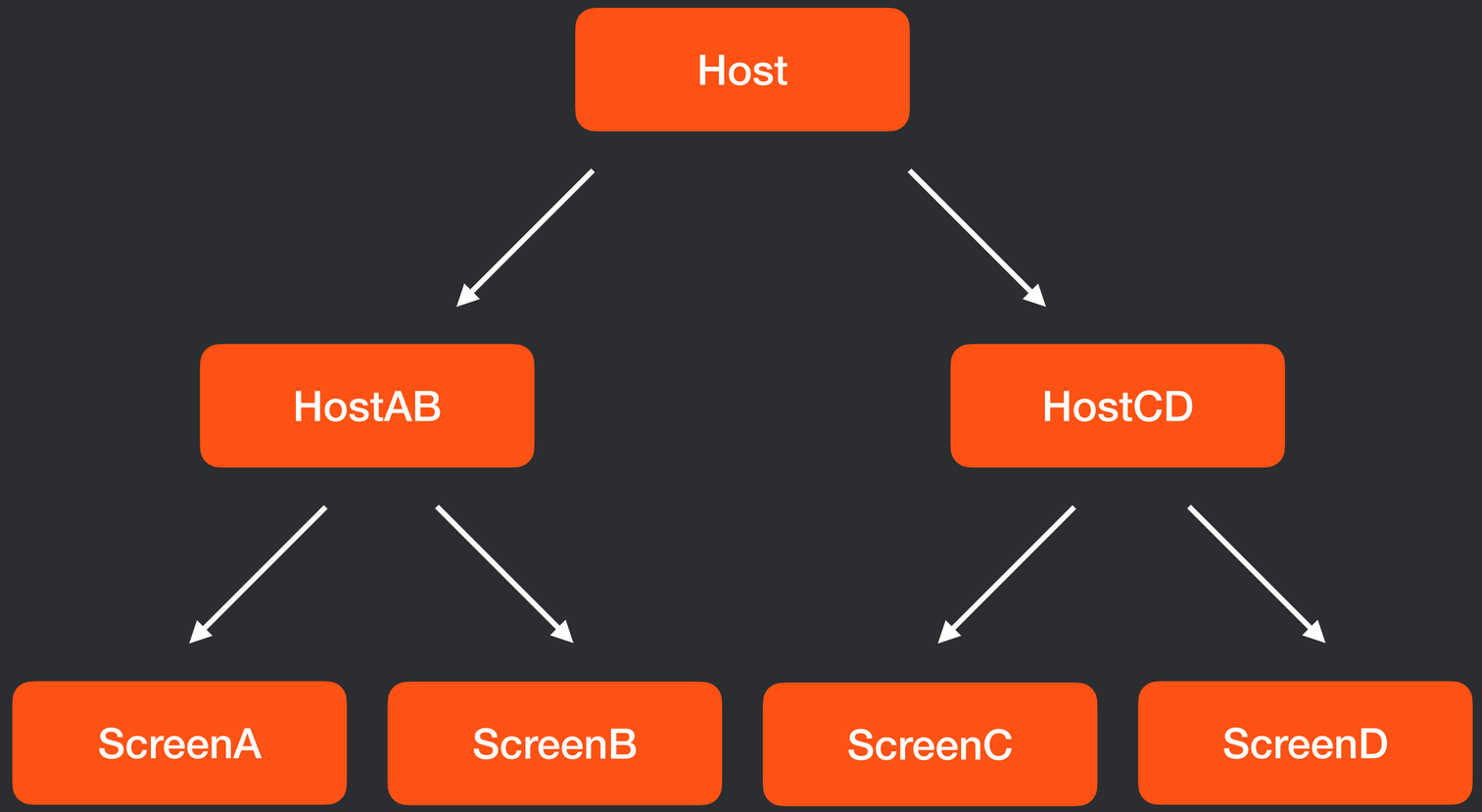
```
@Composable
fun App() {
      // Host
      Box {
         when (screenState) {
             // UI экарана A
             A → Box {
                 Text(text = "Screen A")
             }
             // UI экарана B
             B → Box {
                 Text(text = "Screen B")
             }
         }
    }
}
```

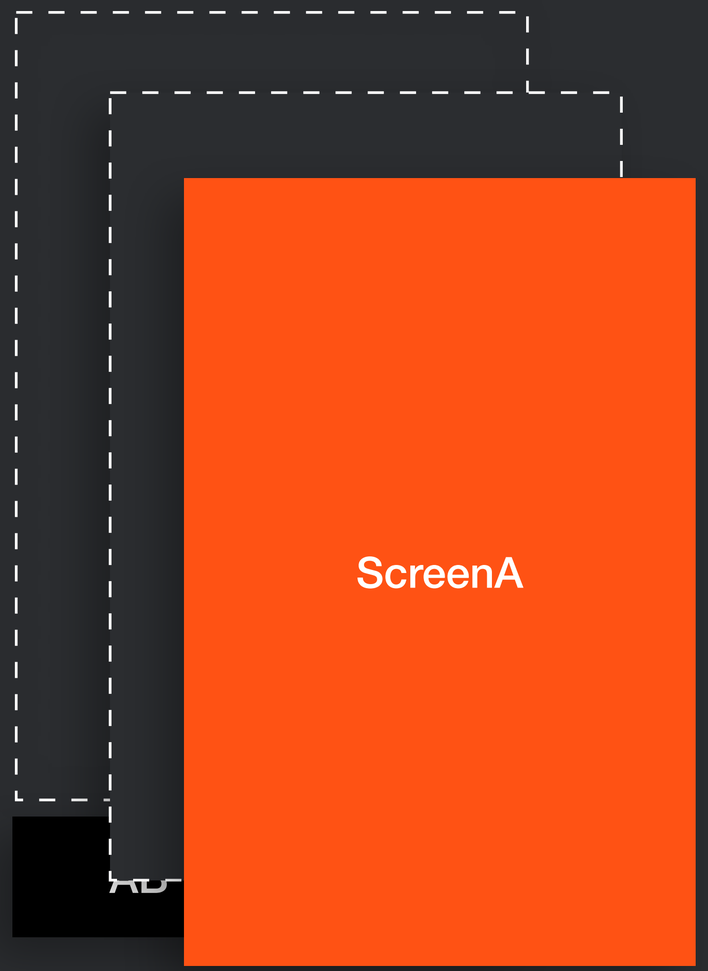



Host
ScreenA
ScreenB


ScreenA


14:00
LTE
100 %
Screen A
AB
CD

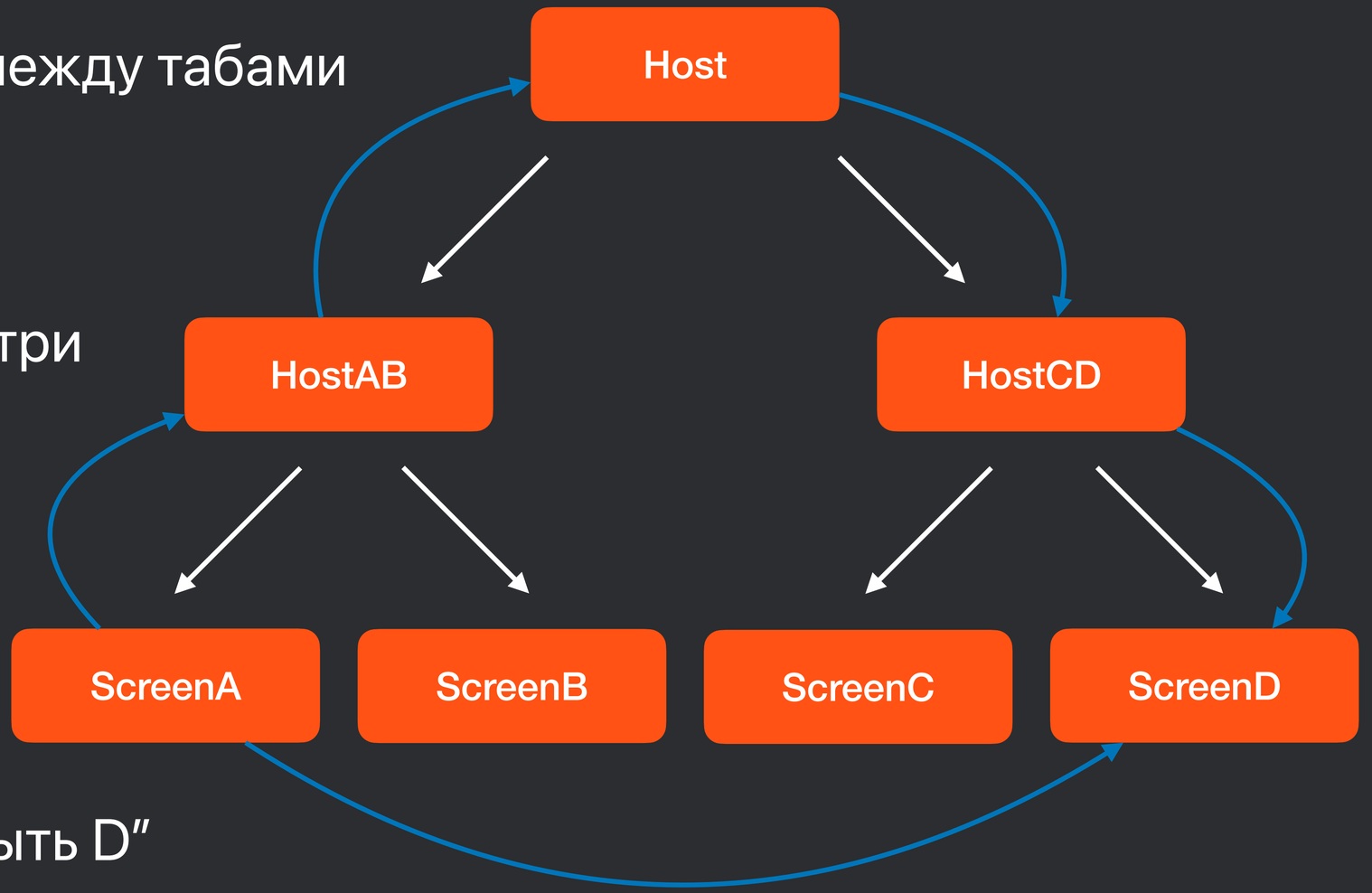

Host
HostAB
HostCD
ScreenA
ScreenB
ScreenC
ScreenD

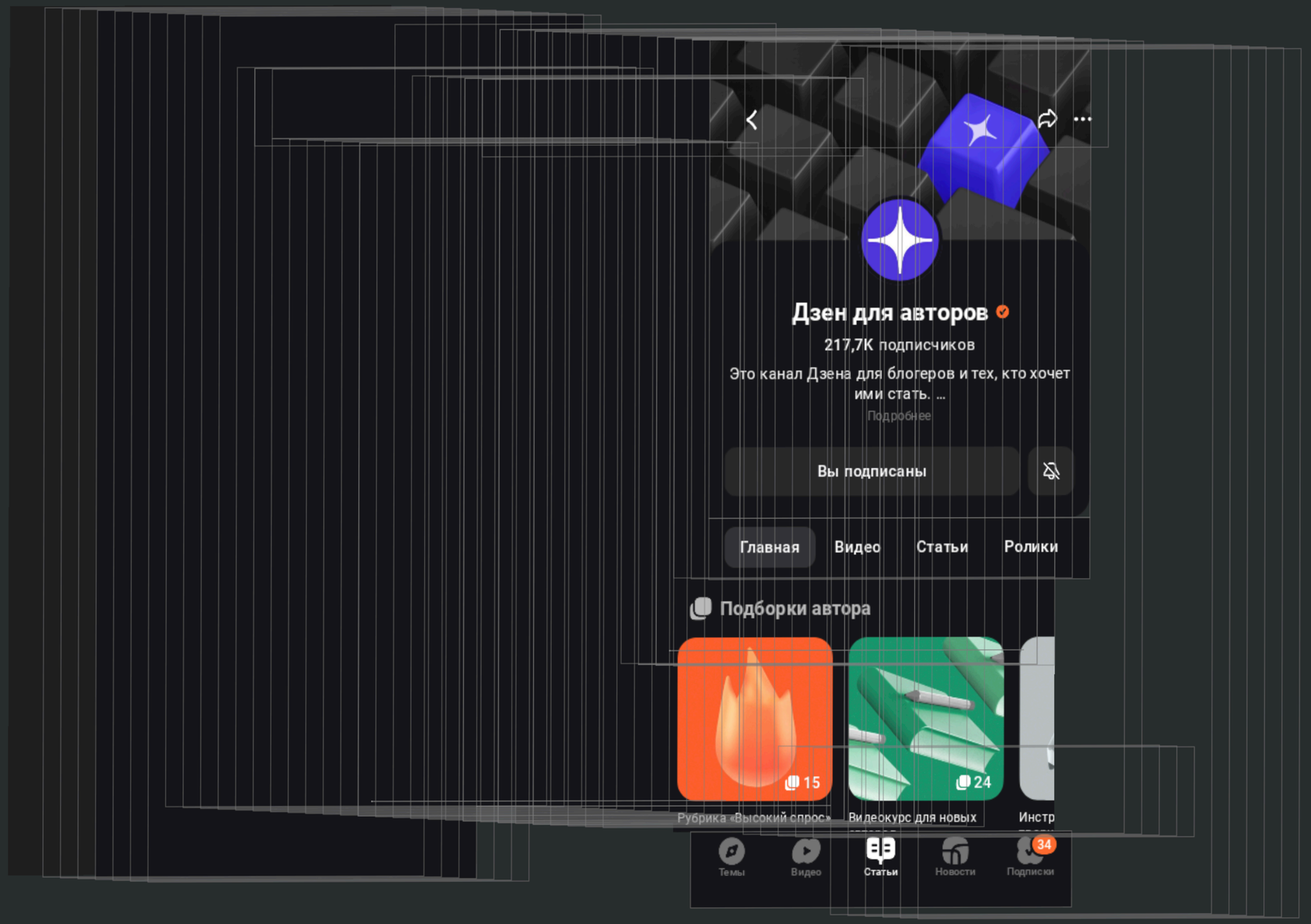

ScreenA

Навигация между табами

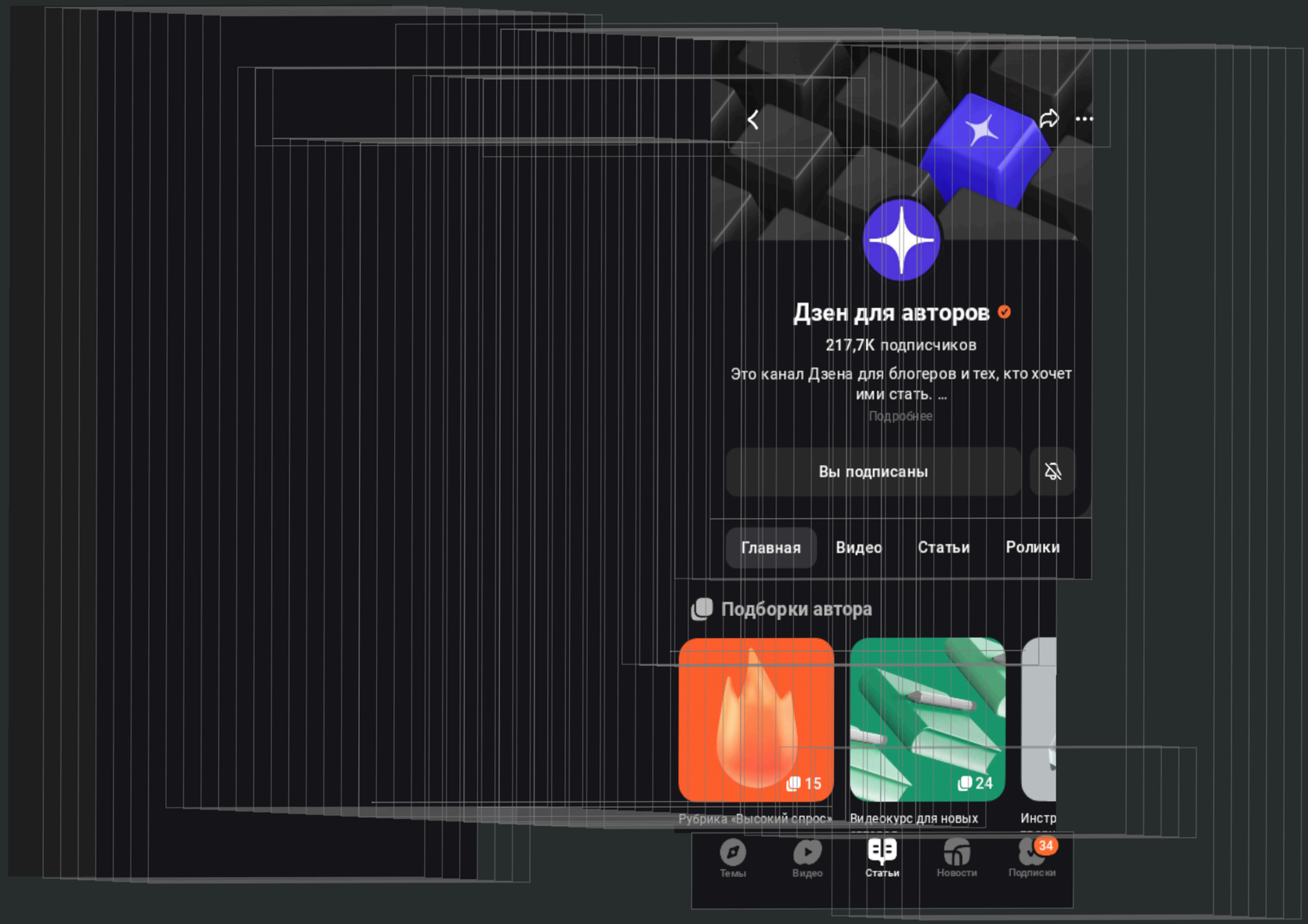
Навигация внутри таба

Кнопка “Открыть D”



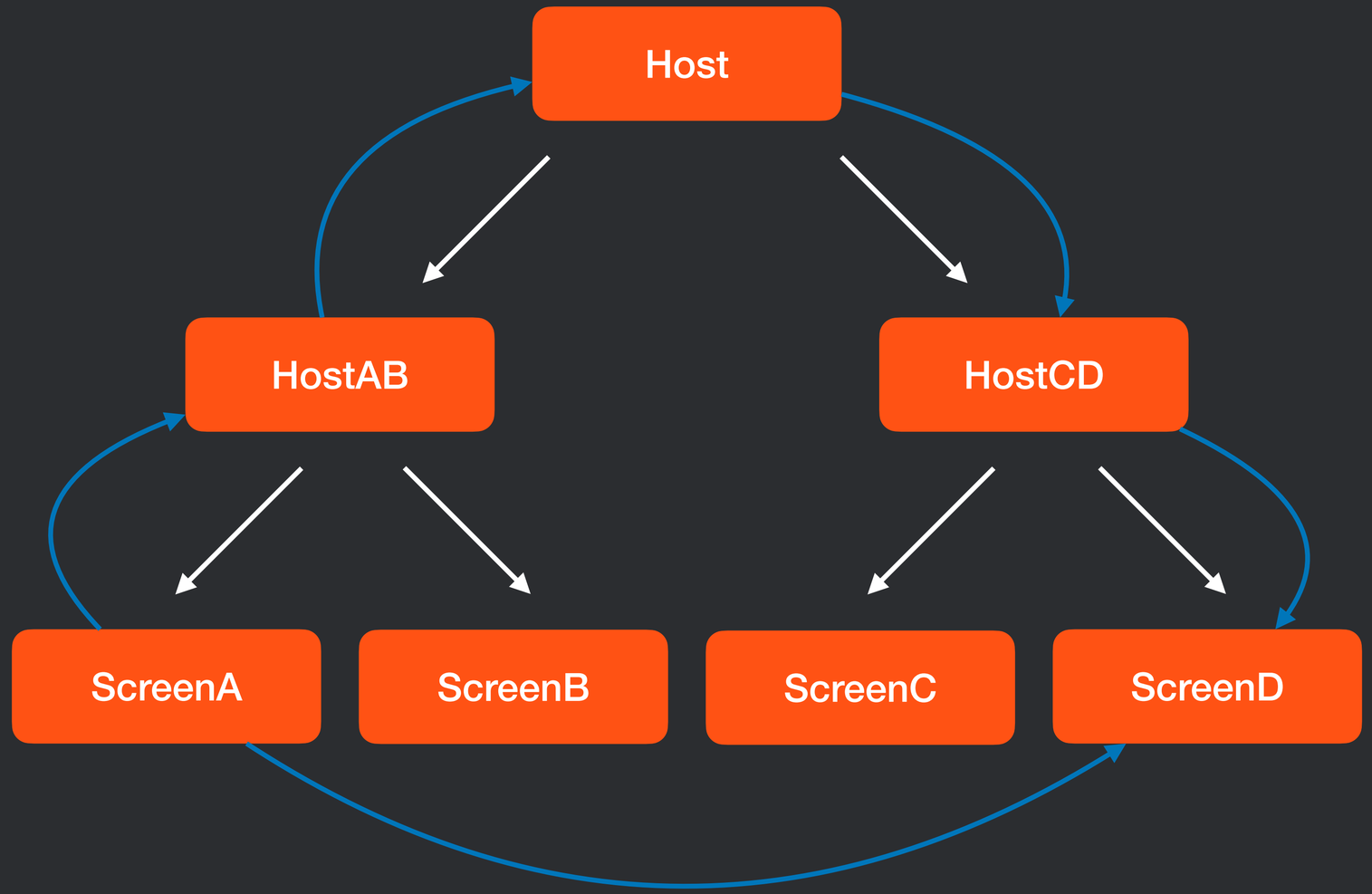

Дзен для авторов
217,7K подписчиков
Это канал Дзена для блогеров и тех, кто хочет ими стать. ...
Подробнее
Вы подписаны
Главная
Видео
Статьи
Ролики
Подборки автора
15
Рубрика «Высокий спрос»
24
Видеокурс для новых
Инстр
Темы
Видео
Статьи
Новости
Подписки
34

```
// Открыть экран
open(screen)

// Закрыть экран
close(screen)
```



Что если навигация пройдет путь за нас?

```
class ScreenA {
...
    open(ScreenD)
...
}
```



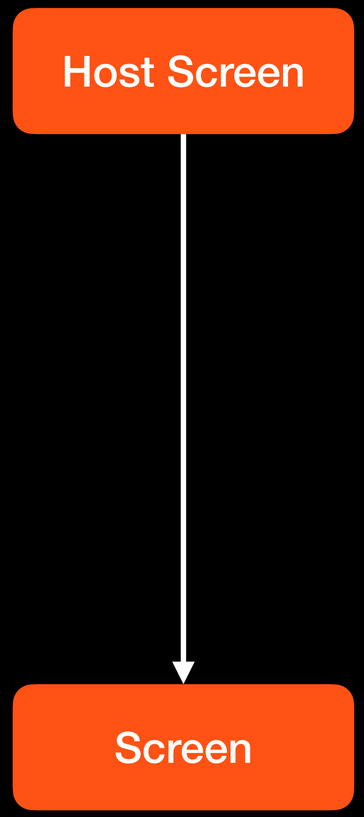
Навигация пройдет путь за нас!

1. Иерархия экранов

2. Алгоритм поиска

3. Изменить UI

Иерархия экранов



Родительский или хостовый экран

Это экран содержащий UI контейнер, в котором будут открываться другие экраны

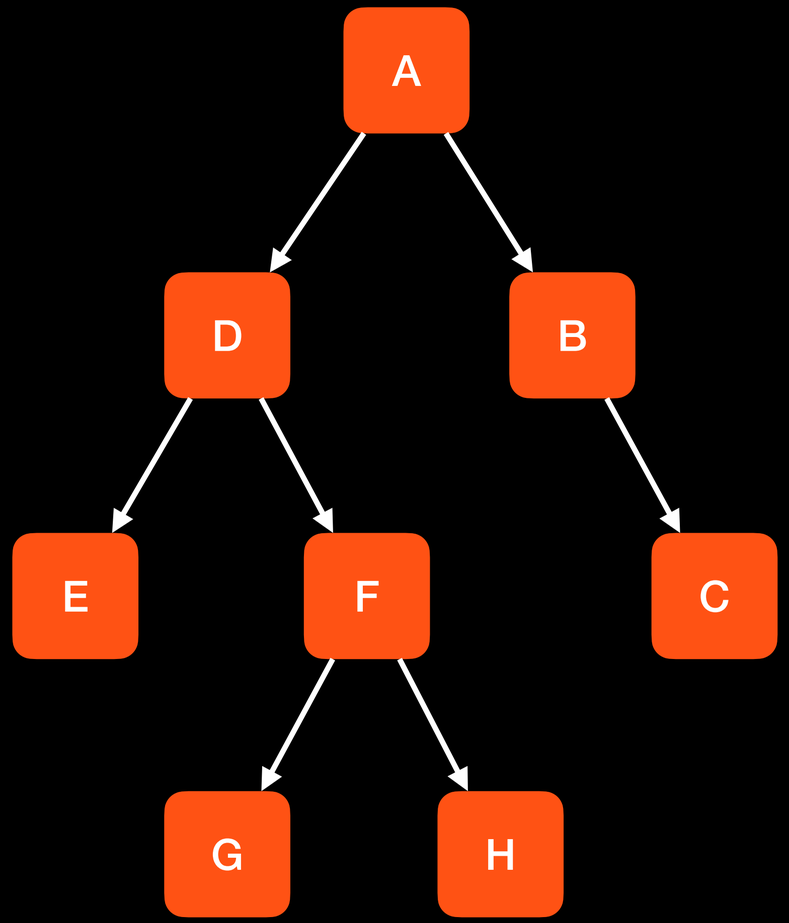
Связь

В UI контейнере родительского экрана будет открываться дочерний экран

**необязательно открыт в данный момент*

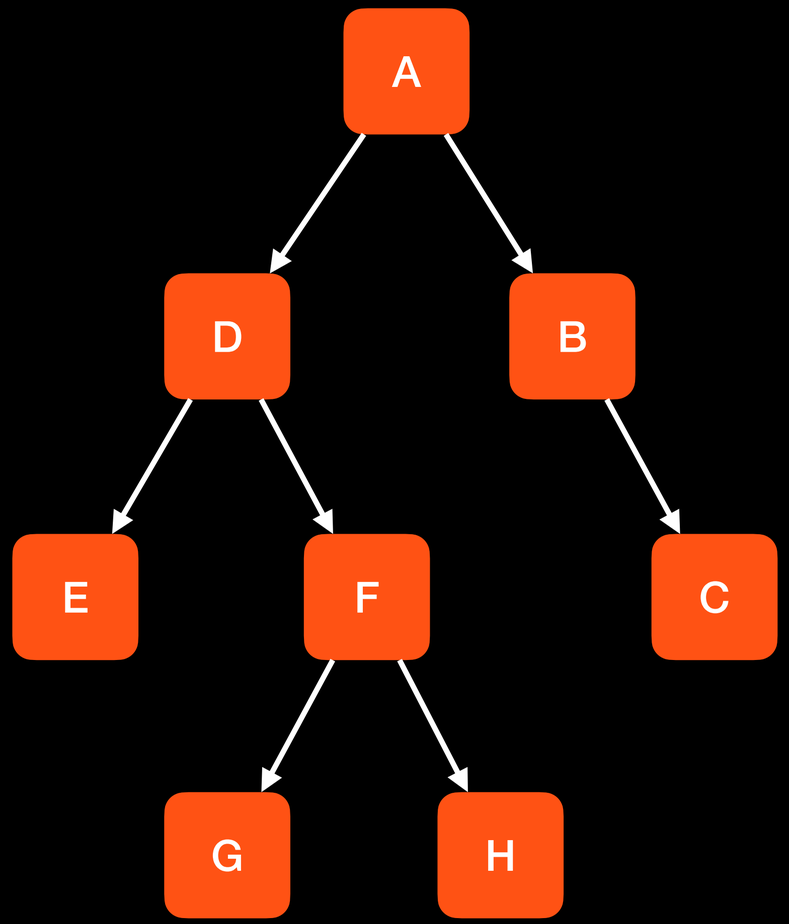
Экран

Содержит логику фичи и UI,
между ними навигурится юзер

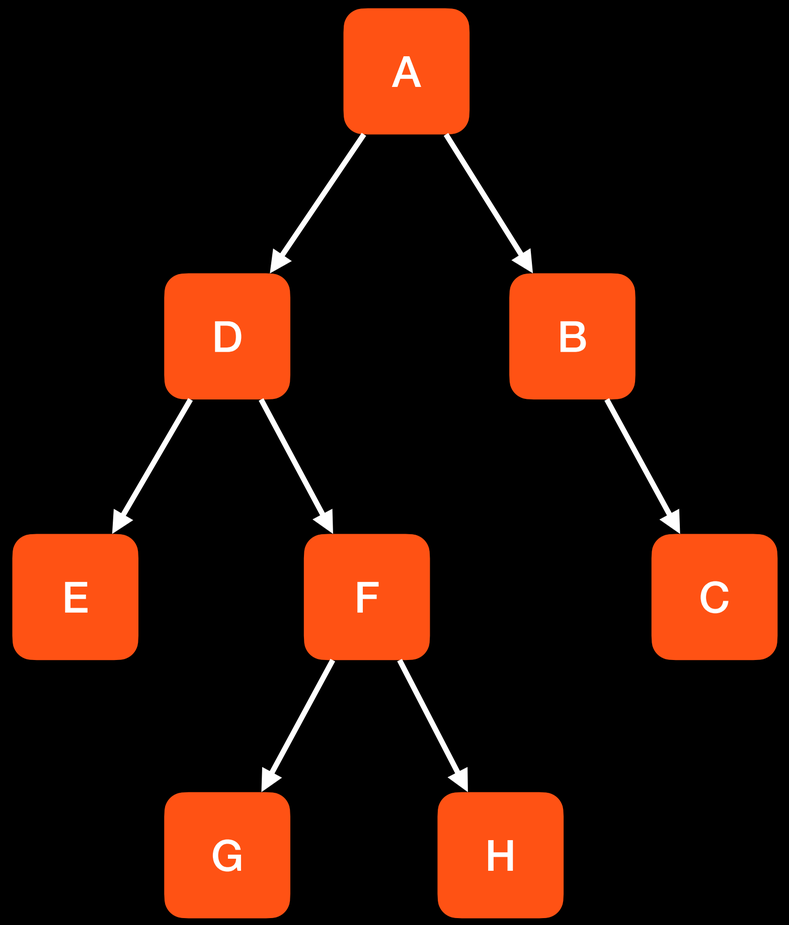


Иерархия экранов

“Карта с адресами расположения каждого экрана приложения”

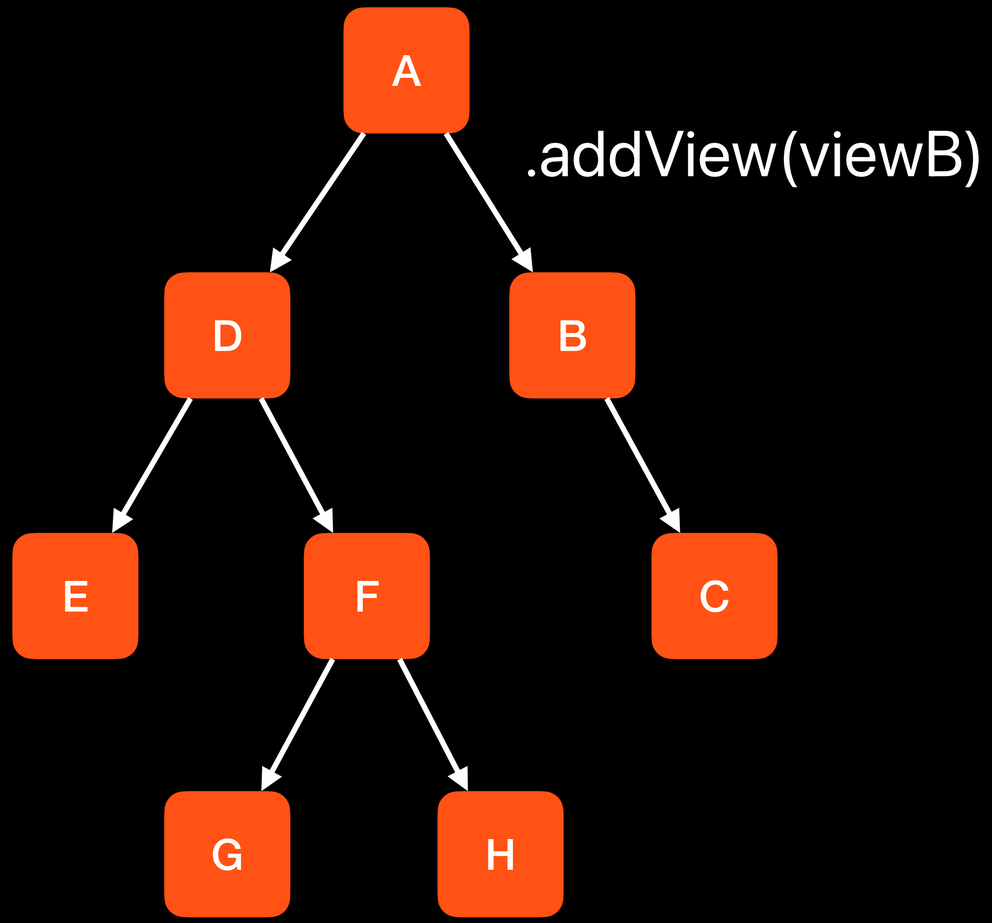



```
registerNavigation(A::class, listOf(B::class, D::class))
registerNavigation(B::class, listOf(C::class))
registerNavigation(D::class, listOf(E::class, F::class))
registerNavigation(F::class, listOf(G::class, H::class))
```

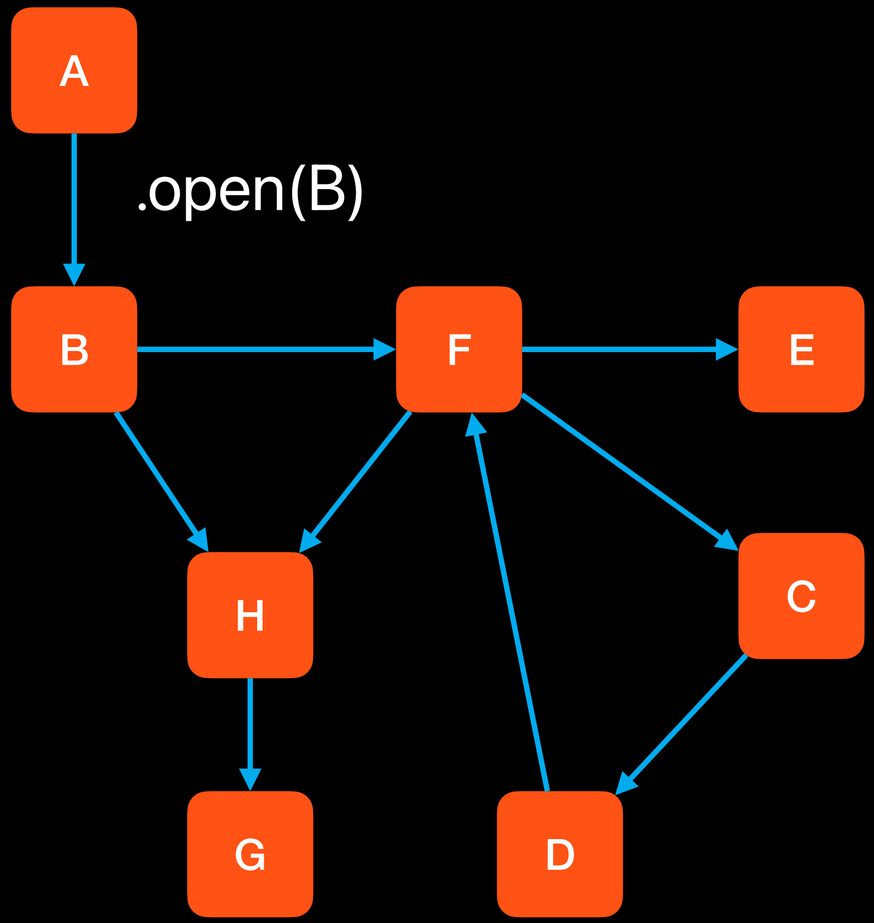


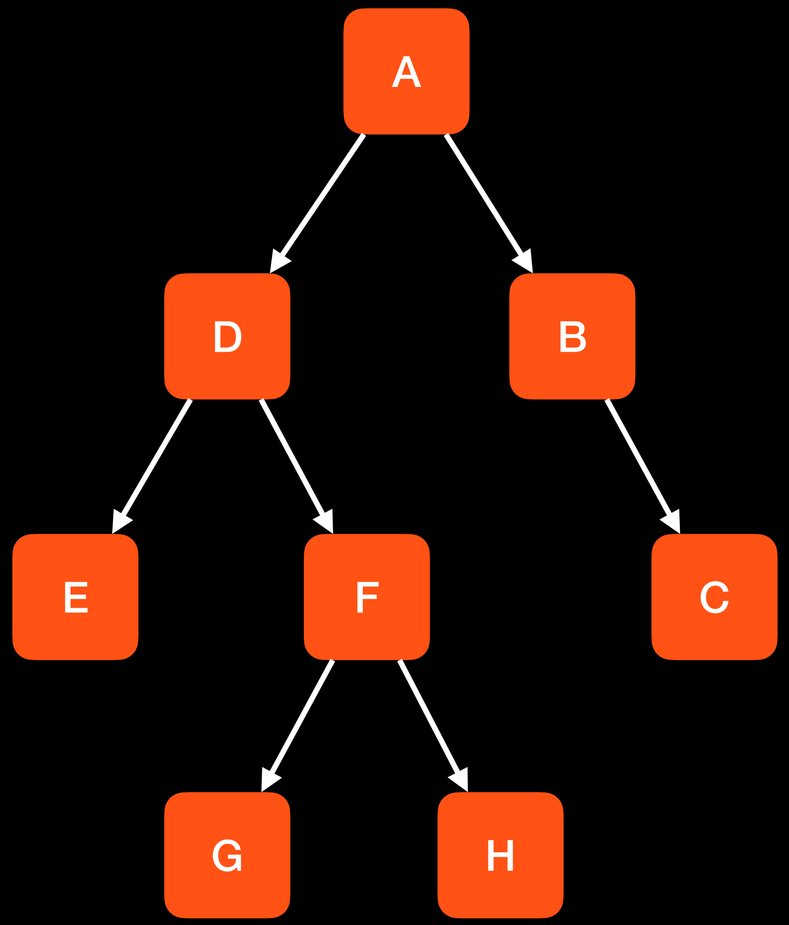
2. Это дерево с одним корнем

Иерархия экранов

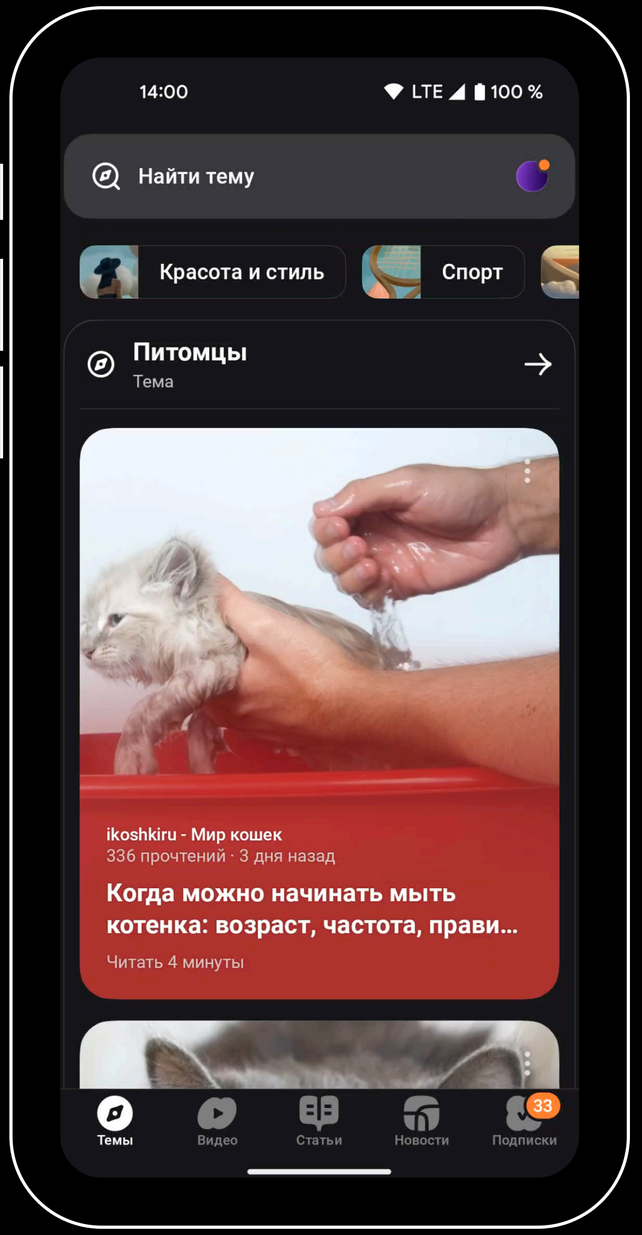


Навигационный граф

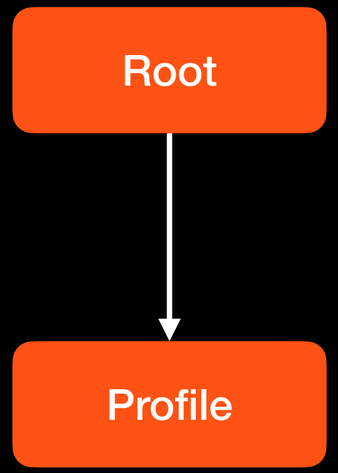




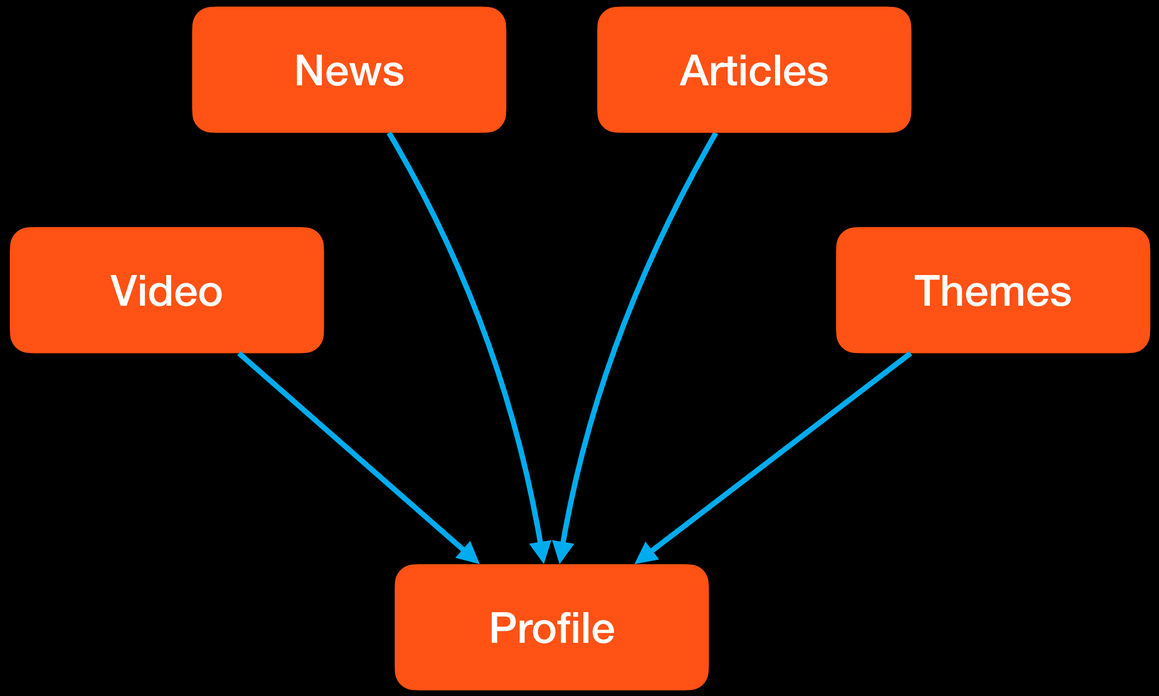
3. Стабильная. Редко изменяется

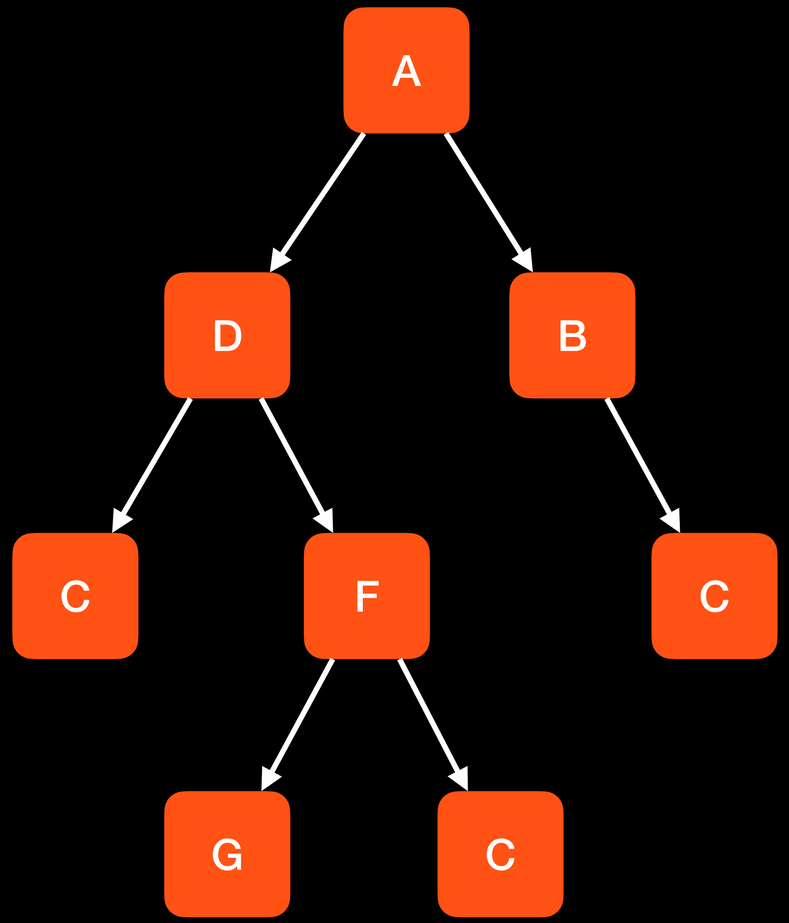

14:00
LTE
100 %
Найти тему
Красота и стиль
Спорт
Питомцы
Тема
ikoshkiru - Мир кошек
336 прочтений · 3 дня назад
Когда можно начинать мыть котенка: возраст, частота, прави...
Читать 4 минуты
Темы
Видео
Статьи
Новости
Подписки
33

Иерархия экранов

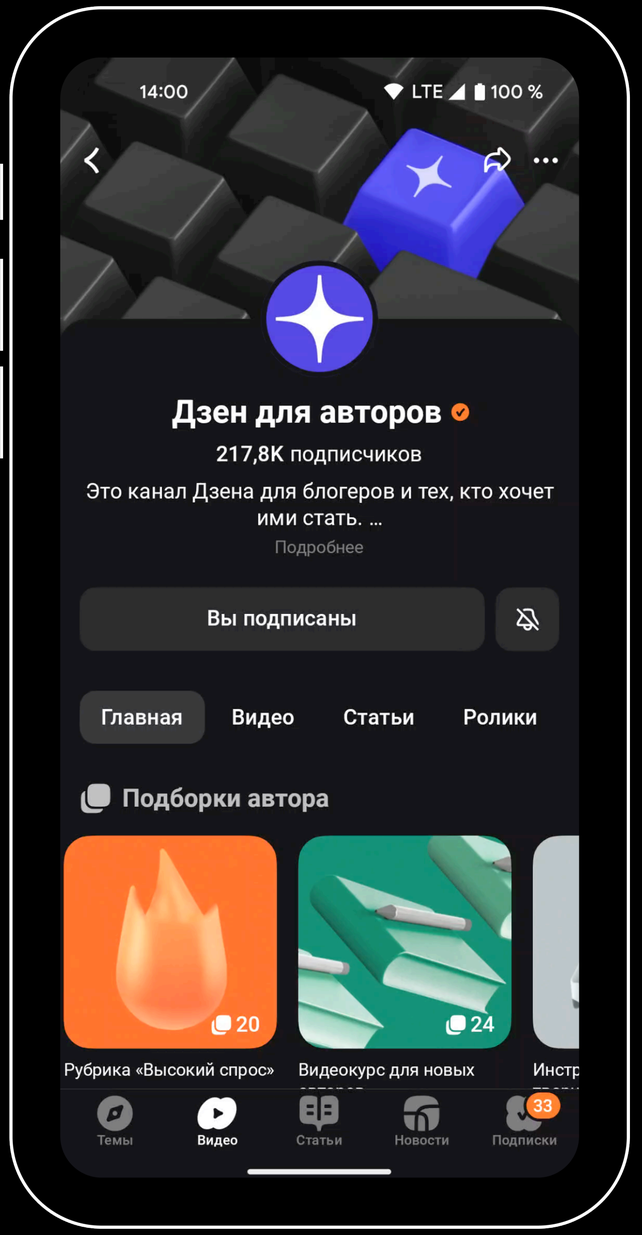


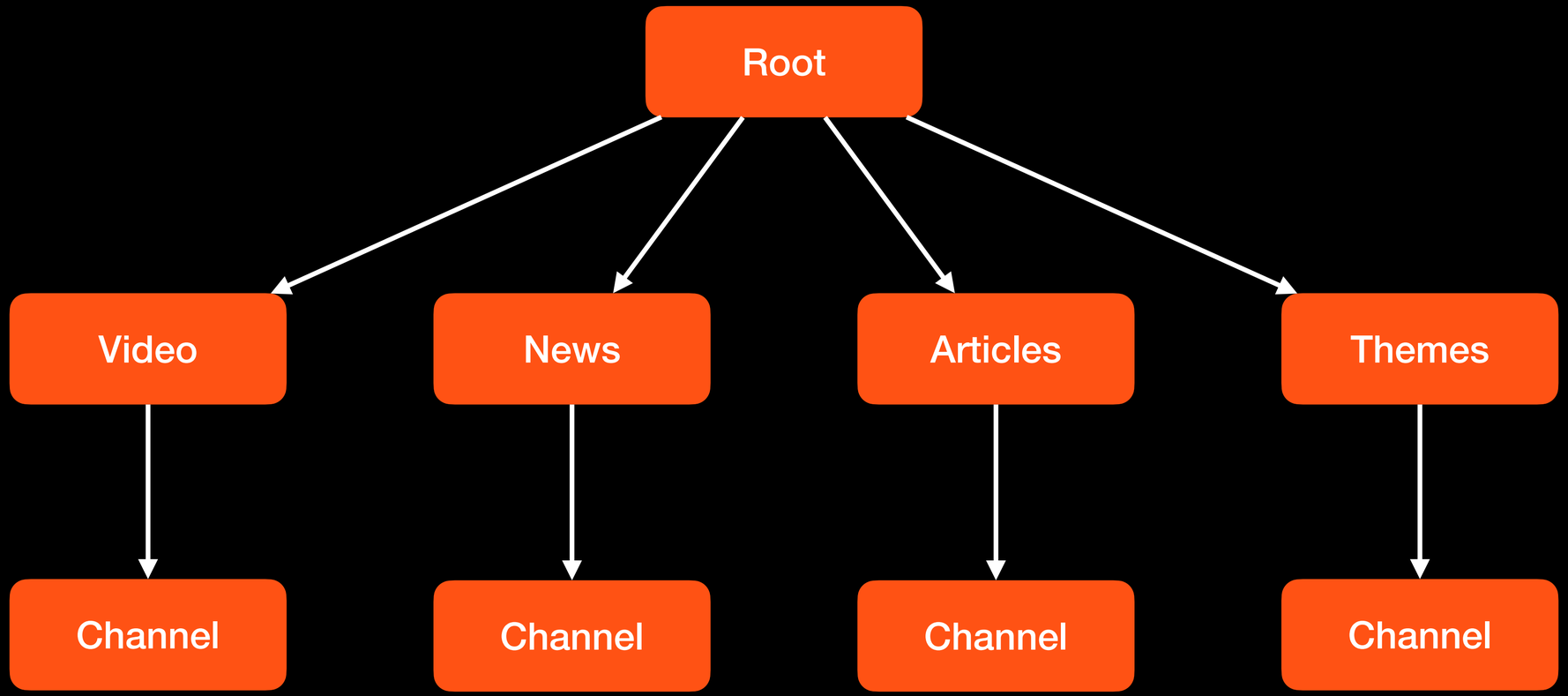
Навигационный граф





4. Может содержать дубликаты


14:00
LTE
100 %
Дзен для авторов
217,8K подписчиков
Это канал Дзена для блогеров и тех, кто хочет ими стать. ...
Подробнее
Вы подписаны
Главная
Видео
Статьи
Ролики
Подборки автора
20
Рубрика «Высокий спрос»
24
Видеокурс для новых
Темы
Видео
Статьи
Новости
33
Подписки


Root
Video
News
Articles
Themes
Channel
Channel
Channel
Channel

```
// Зарегистрировать во всех табах
registerNavigationInTab(listOf(Channel::class))

fun registerNavigationInTab(screens: List) {
    // Для каждого таба
    tabs.forEach { tab →
        registerNavigation(tab, screens)
    }
}
```

Навигация пройдет путь за нас!

✅ 1. Иерархия экранов

2. Алгоритм поиска

3. Изменить UI

Алгоритм поиска

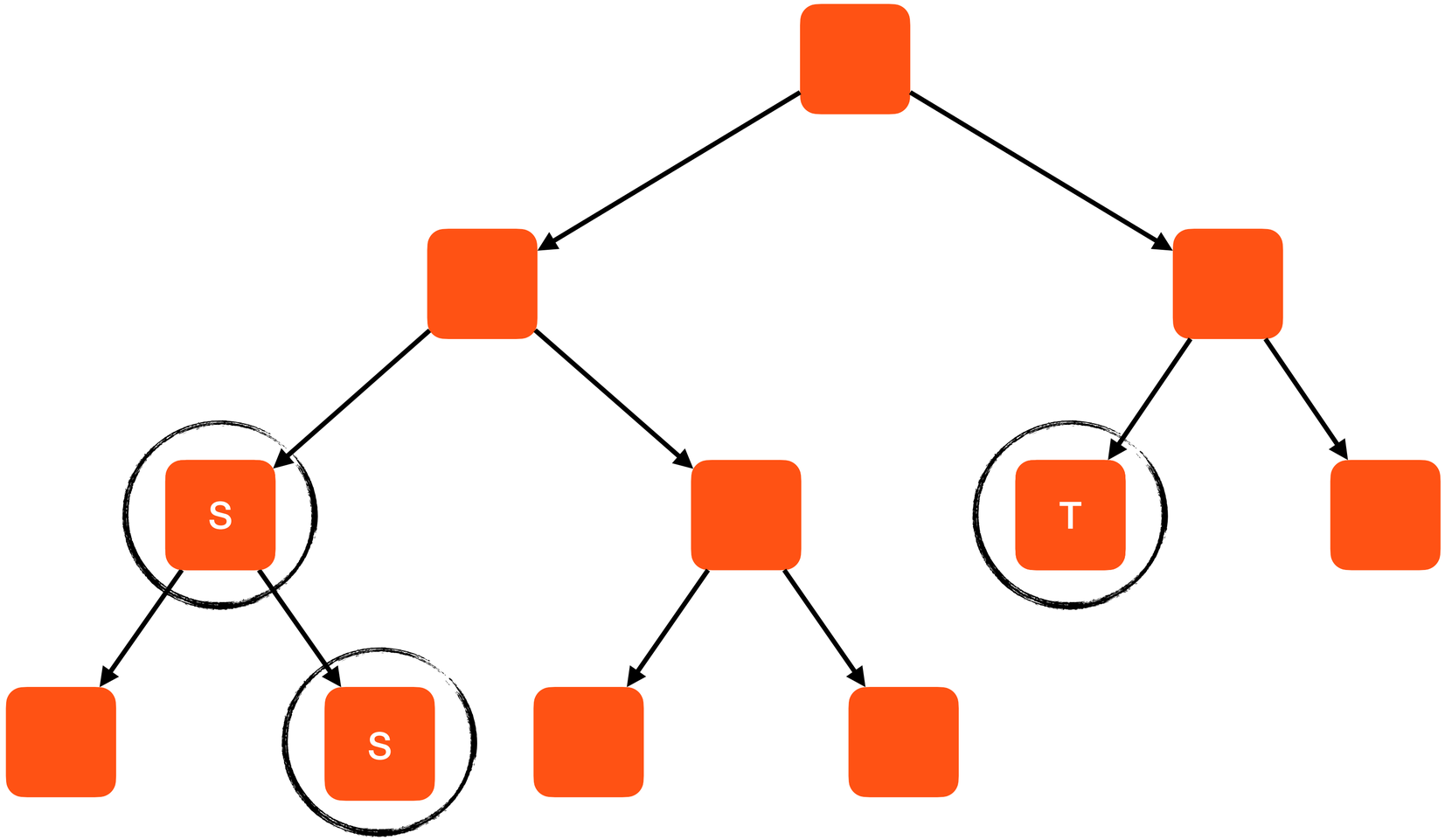
Input

Иерархия экранов

Стартовый экран S

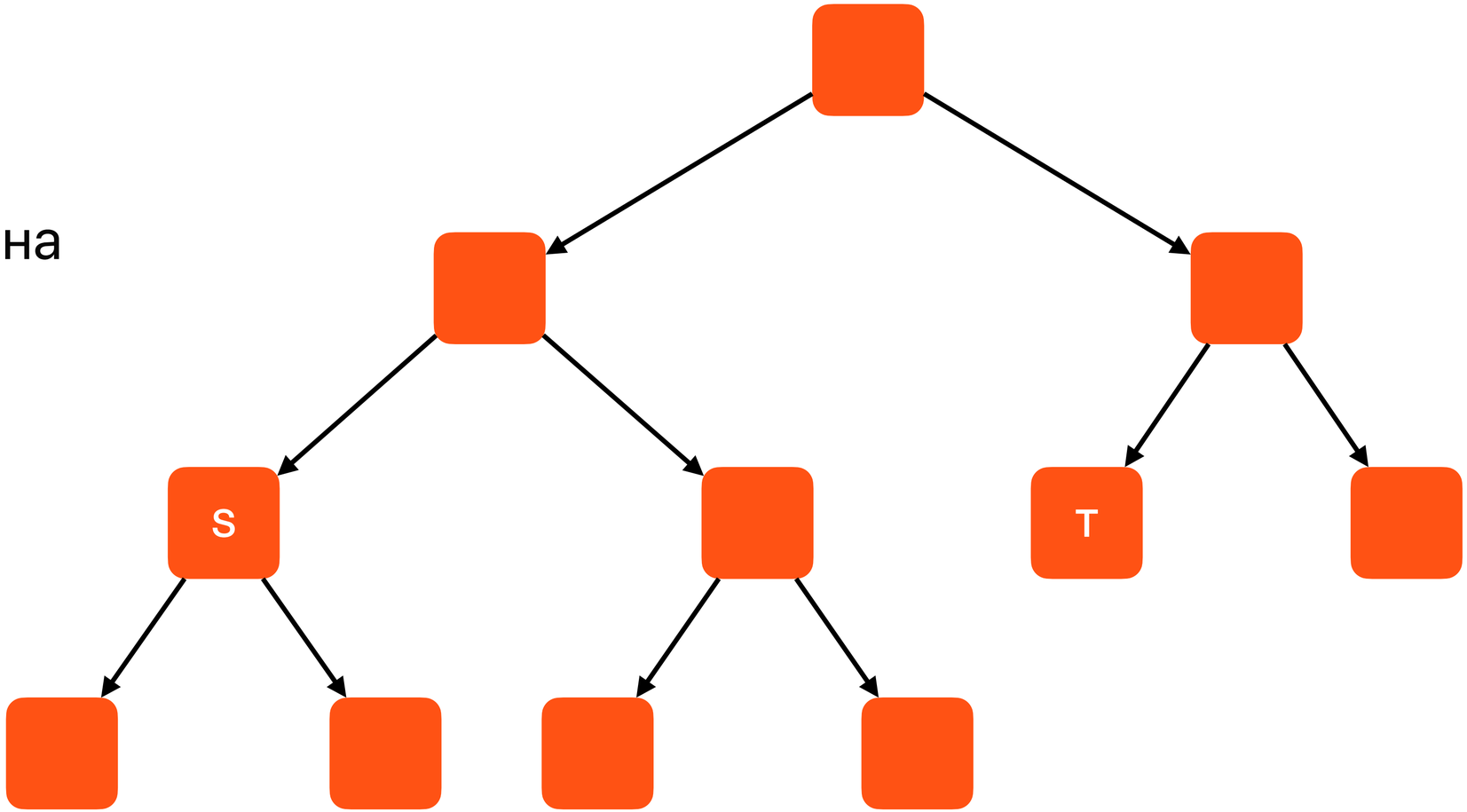
Искомый экран T

Путь до стартового экрана



Output

Путь до искомого экрана



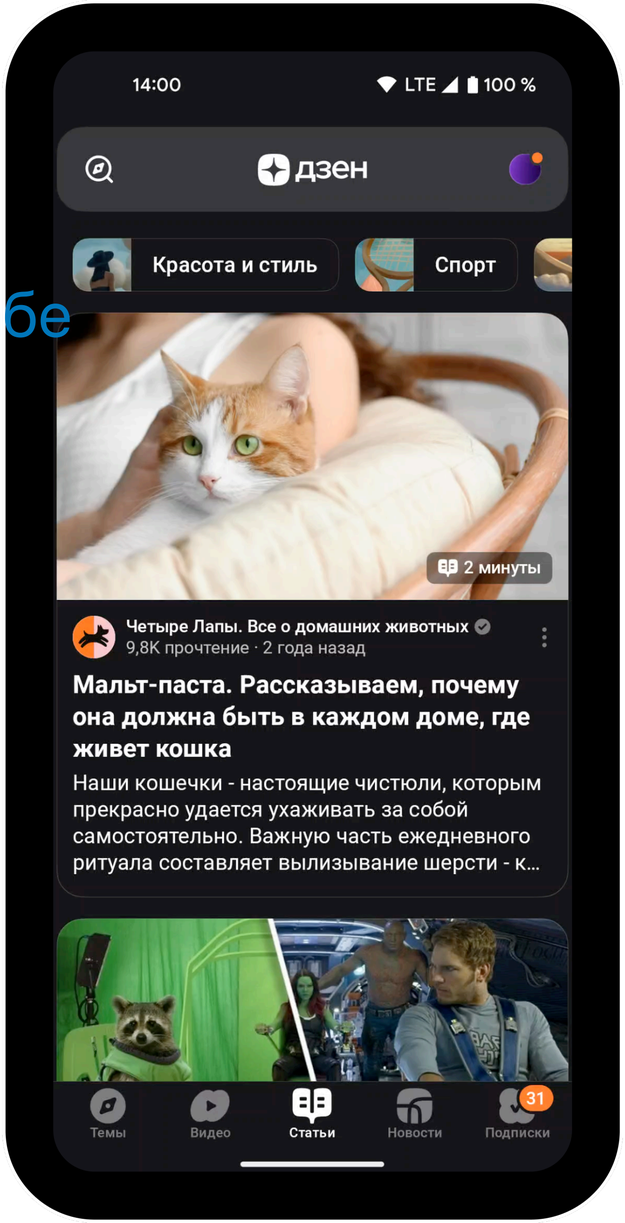
Алгоритм должен быть
интуитивным

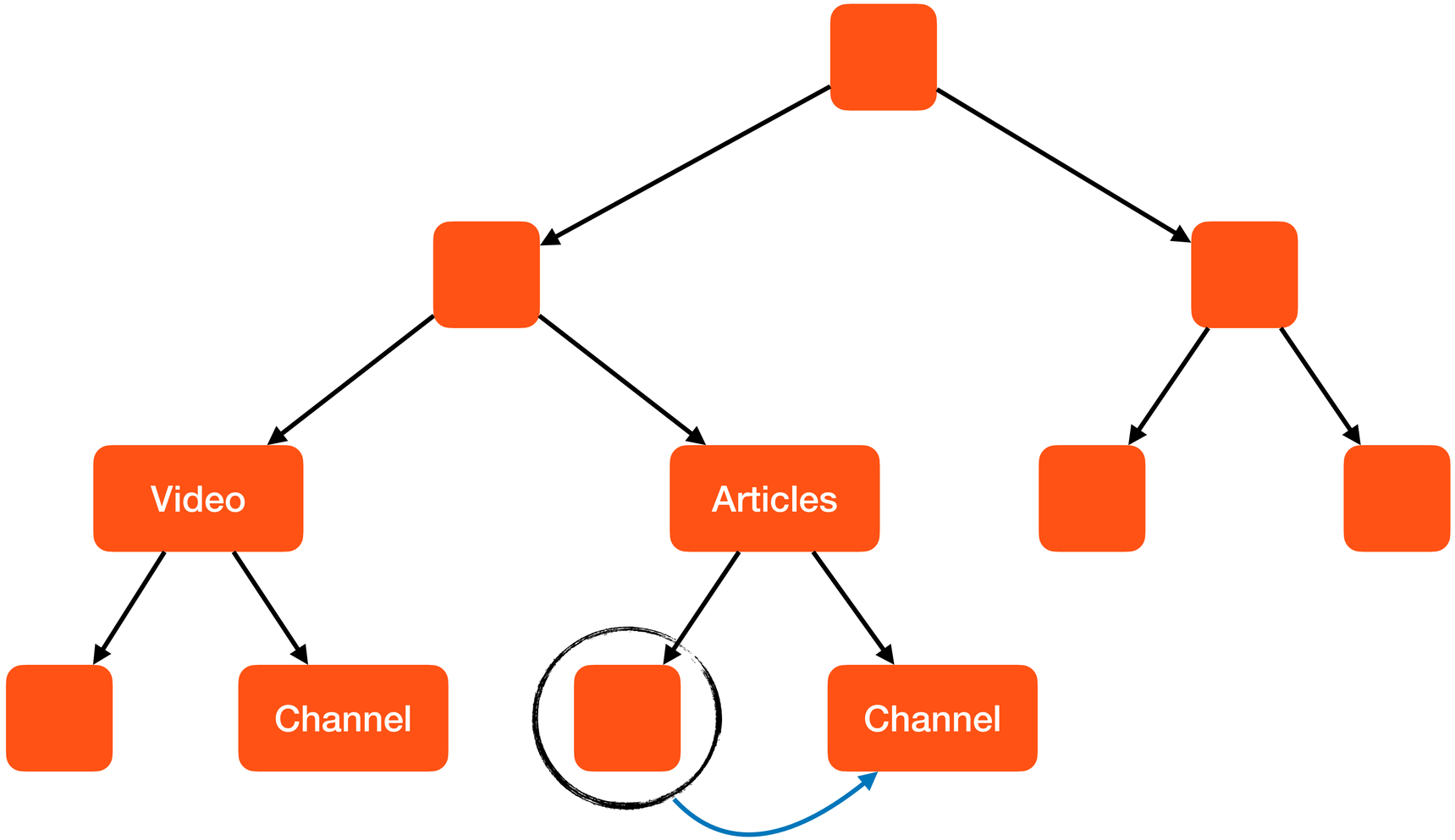
ТЗ: Открой канал в том же табе

```
open(Channel(id = "12345"))
```

ТЗ: Закрой канал

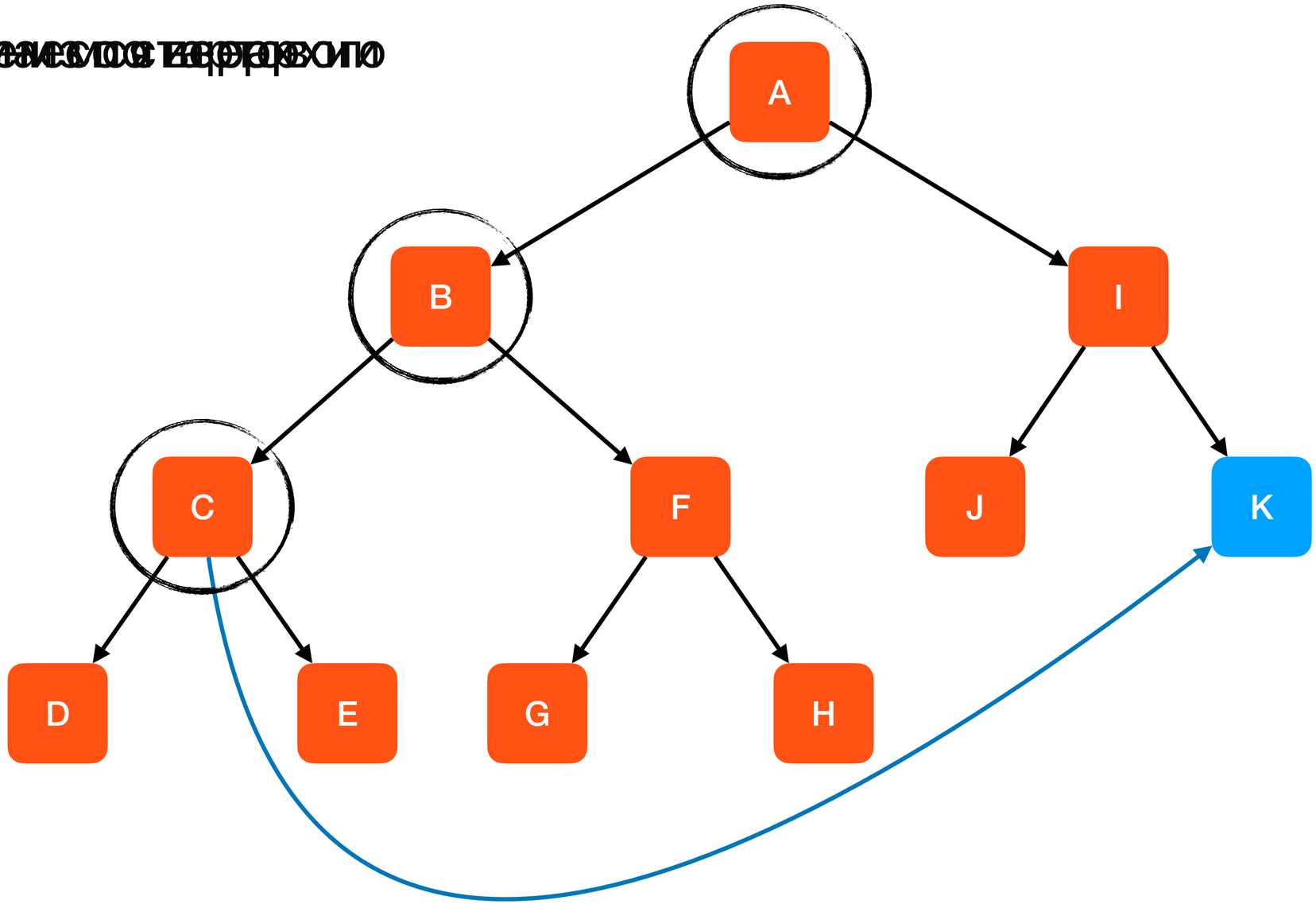
```
close(Channel(id = "12345"))
```




Video
Articles
Channel
Channel

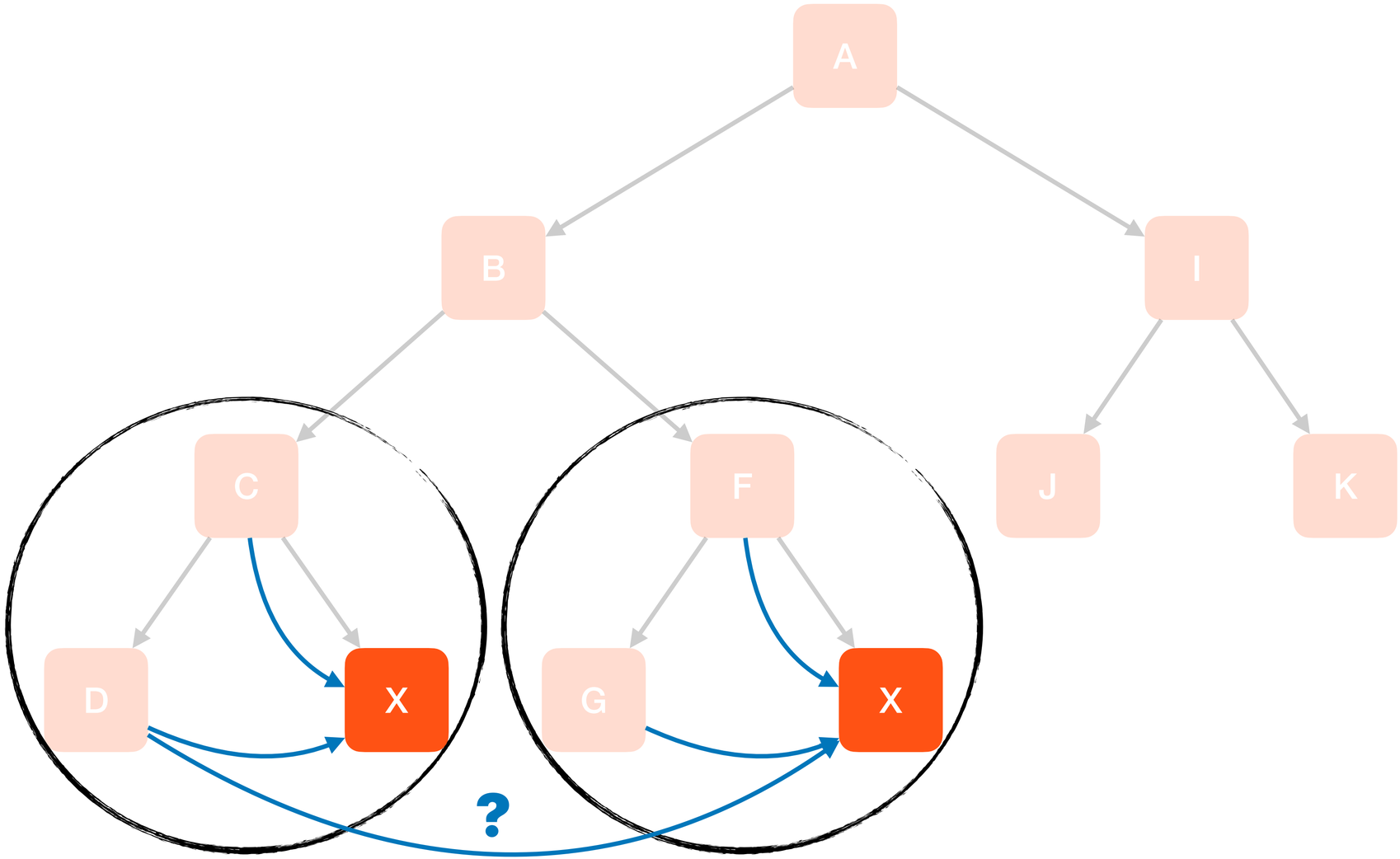
Алгоритм должен
находить ближайший экран от стартового

C.open(K)



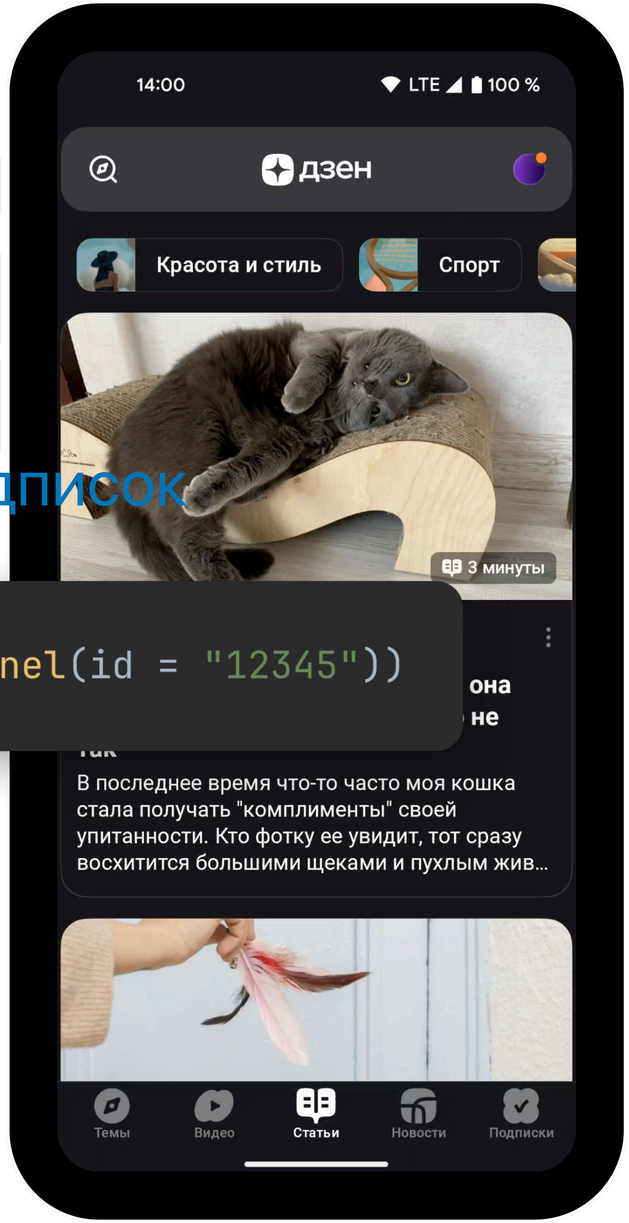
Дубликаты экранов

open(X)



ТЗ: Открой канал внутри подписок

```
inside(SubscriptionsTab).open(Channel(id = "12345"))
```

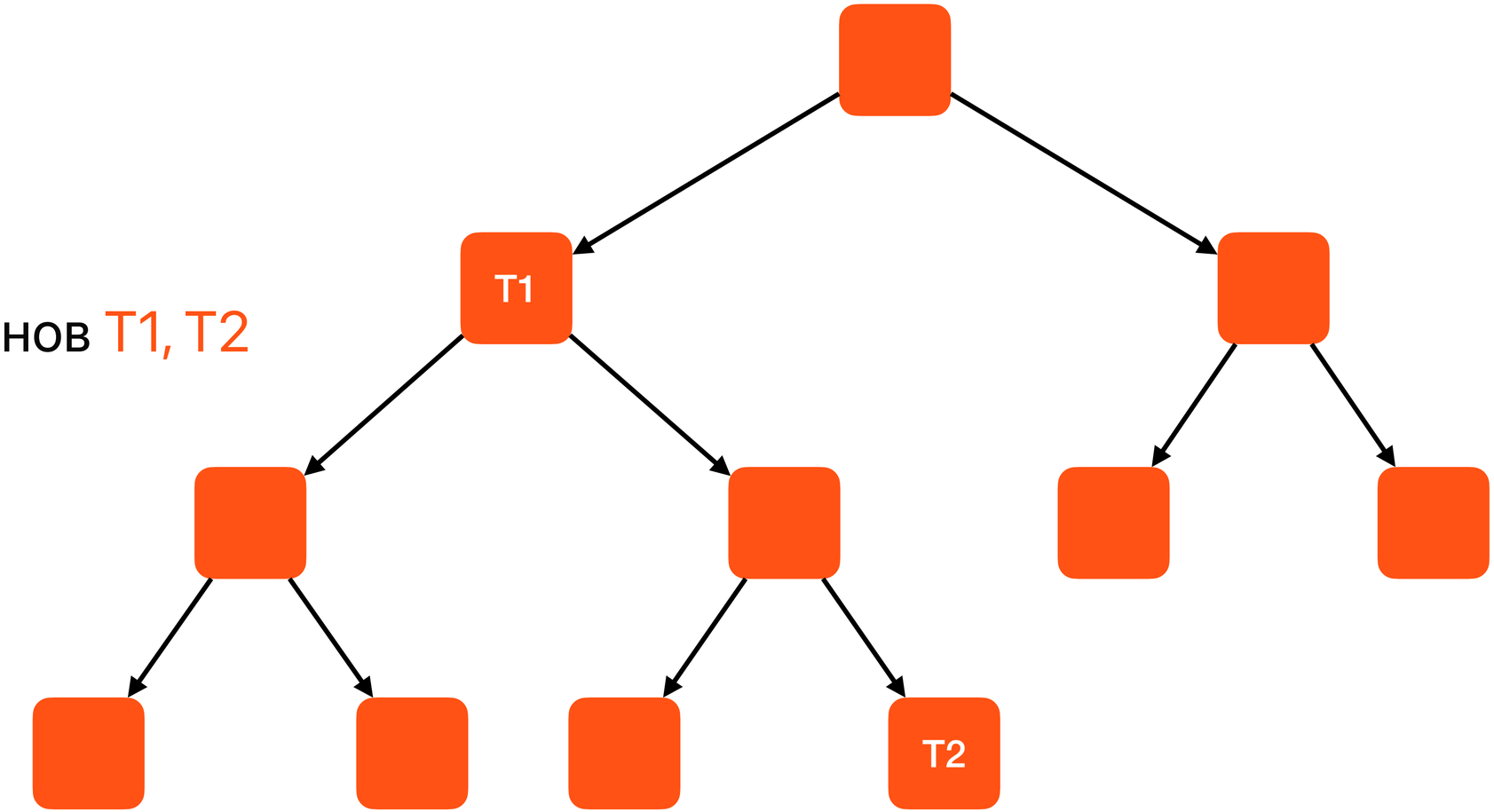


```
// Открыть экран внутри ...
inside(...).inside(...).inside(...).open(screen)

// Закрыть экран внутри ...
inside(...).inside(...).inside(...).close(screen)
```

Input

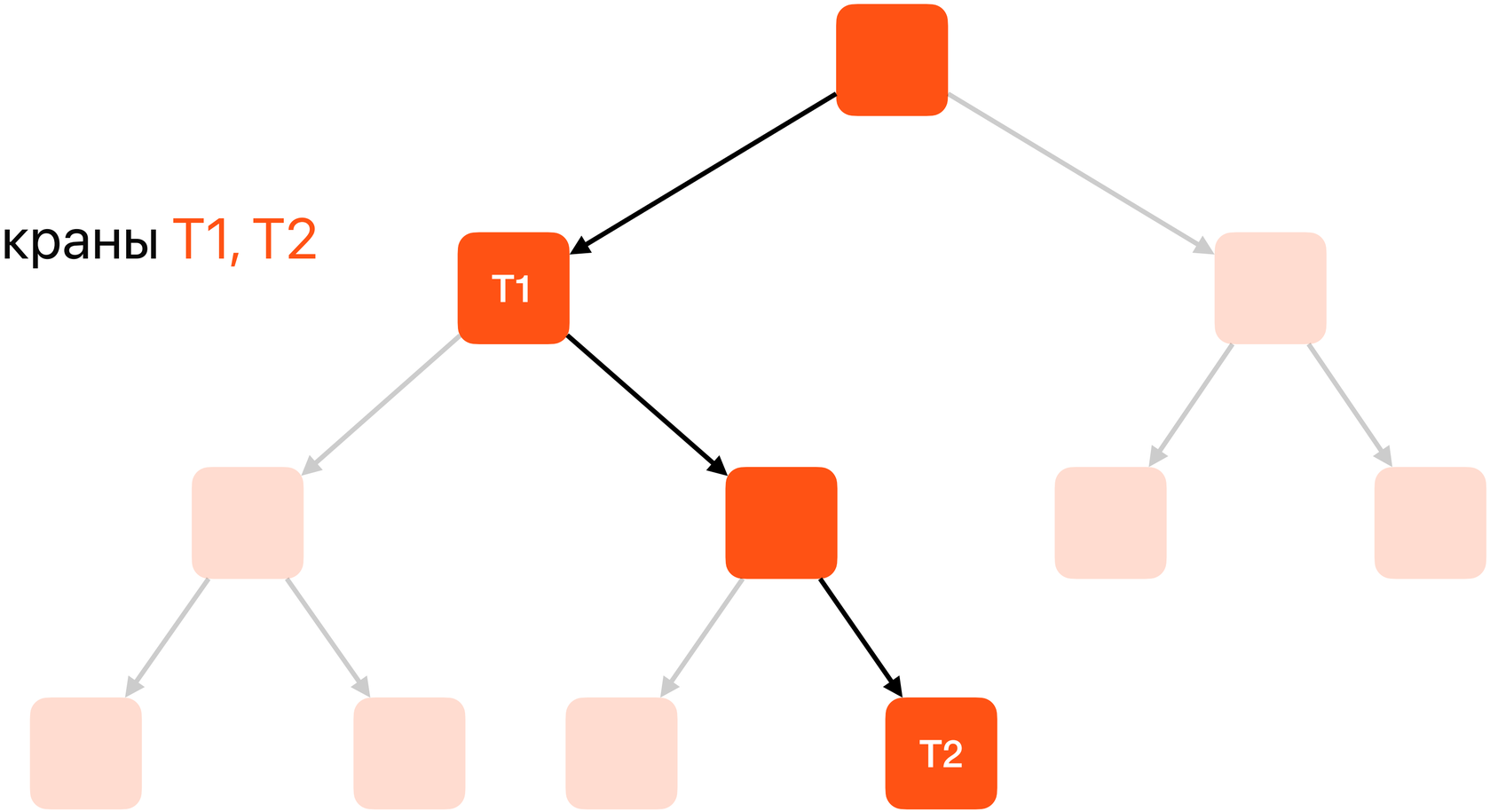
~~Искомый экран T~~

Список искомых экранов T1, T2



Output

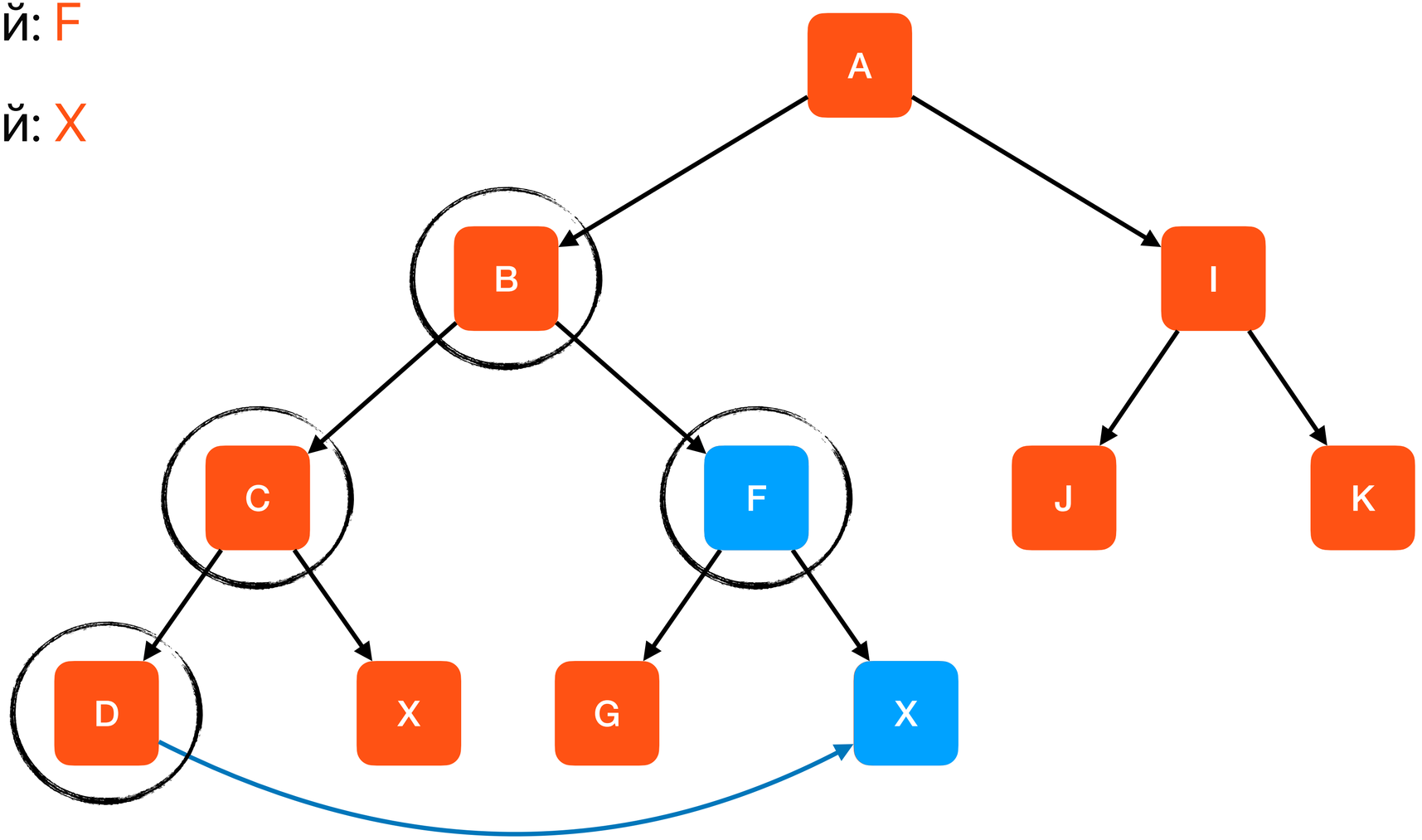
Путь через искомые экраны T1, T2



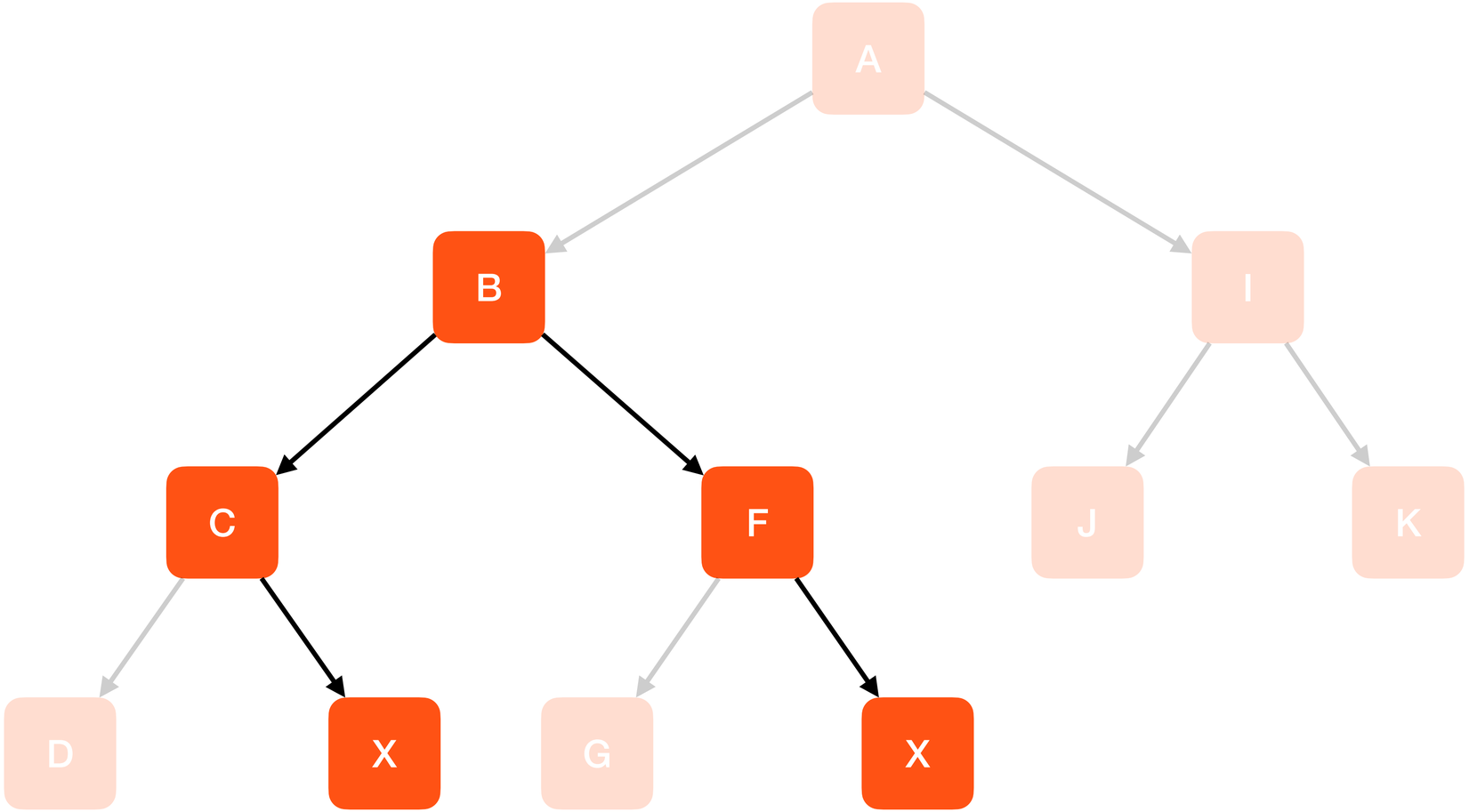
D.inside(F).open(X)

Искомый: F

Искомый: X



D.inside(B).open(X)



Навигация пройдет путь за нас!

✅ 1. Иерархия экранов

✅ 2. Алгоритм поиска

3. Изменить UI

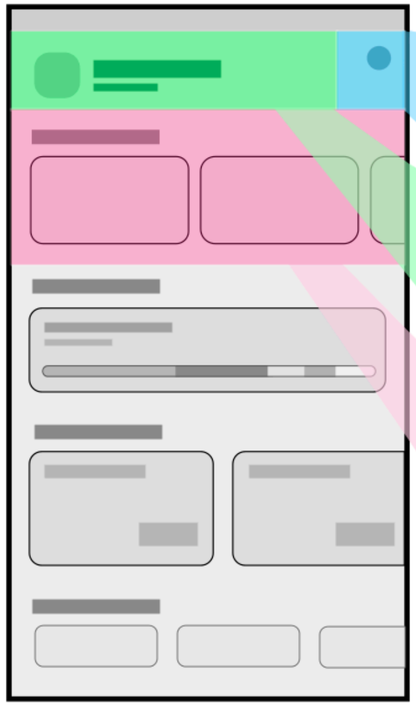
Изменить UI

Decompose

=

Декомпозиция

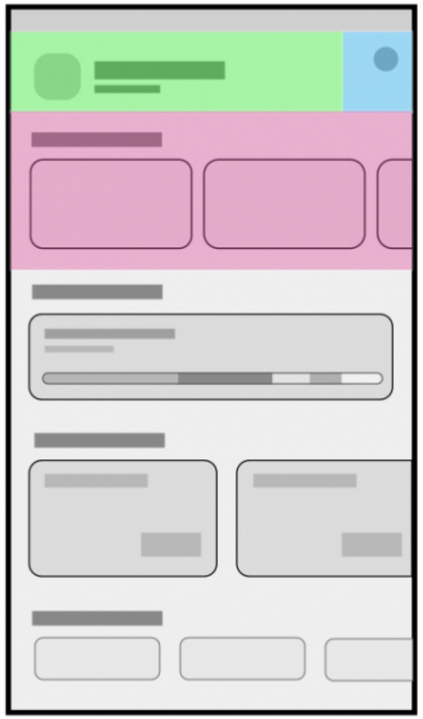
Проблема массивных ViewModel

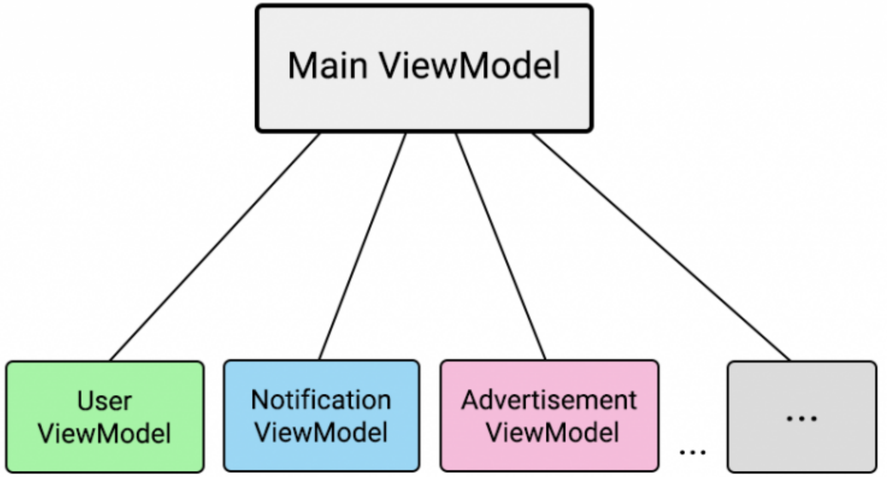


```
class MainViewModel {

    val isNotificationBadgeVisible: StateFlow<Boolean>

    fun onNotificationIconClick(){
        ...
    }

    val profile: StateFlow<Profile>

    fun onAvatarClick(){
        ...
    }

    val advertisementItems: StateFlow<List<AdvertisementItem>>

    fun onAdvertisementItemClick(advertisementId: AdvertizementId) {
        ...
    }

    ...

    ...

    ...

}
```

Компонентный подход.
Боремся со сложностью
в Android-приложениях

Компонентный подход





Компонентный подход. Боремся со сложностью в Android-приложениях

Decompose компонент

Функционал как у Fragment

```
interface ComponentContext :
    LifecycleOwner,       // Аналог LifecycleOwner. ЖЦ компонента
    StateKeeperOwner,     // Аналог SavedStateHandle. Сохранение состояния
    InstanceKeeperOwner,  // Аналог ViewModel. Сохранение объекта
    BackHandlerOwner      // Аналог OnBackPressedDispatcher. Кнопка Back
```

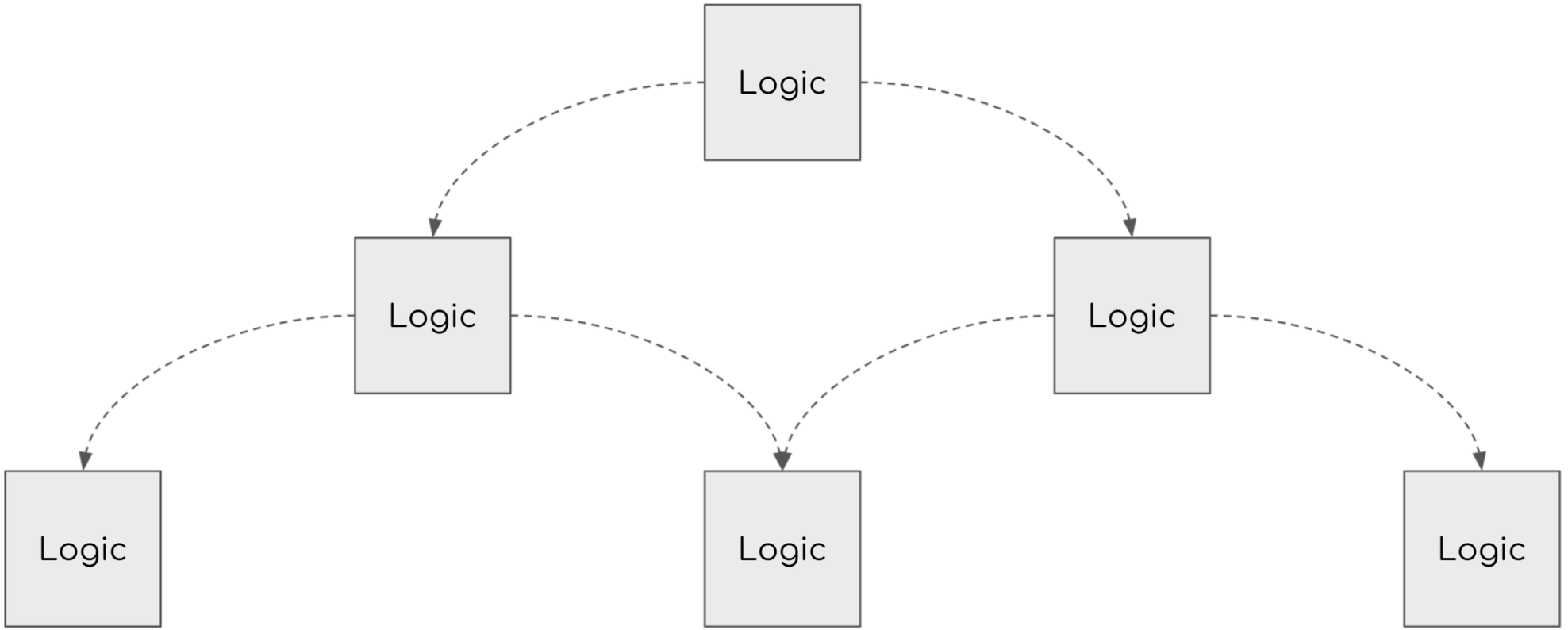
Decompose дочерний компонент

```
class ProfileComponent(
    componentContext: ComponentContext
) : ComponentContext by componentContext
```

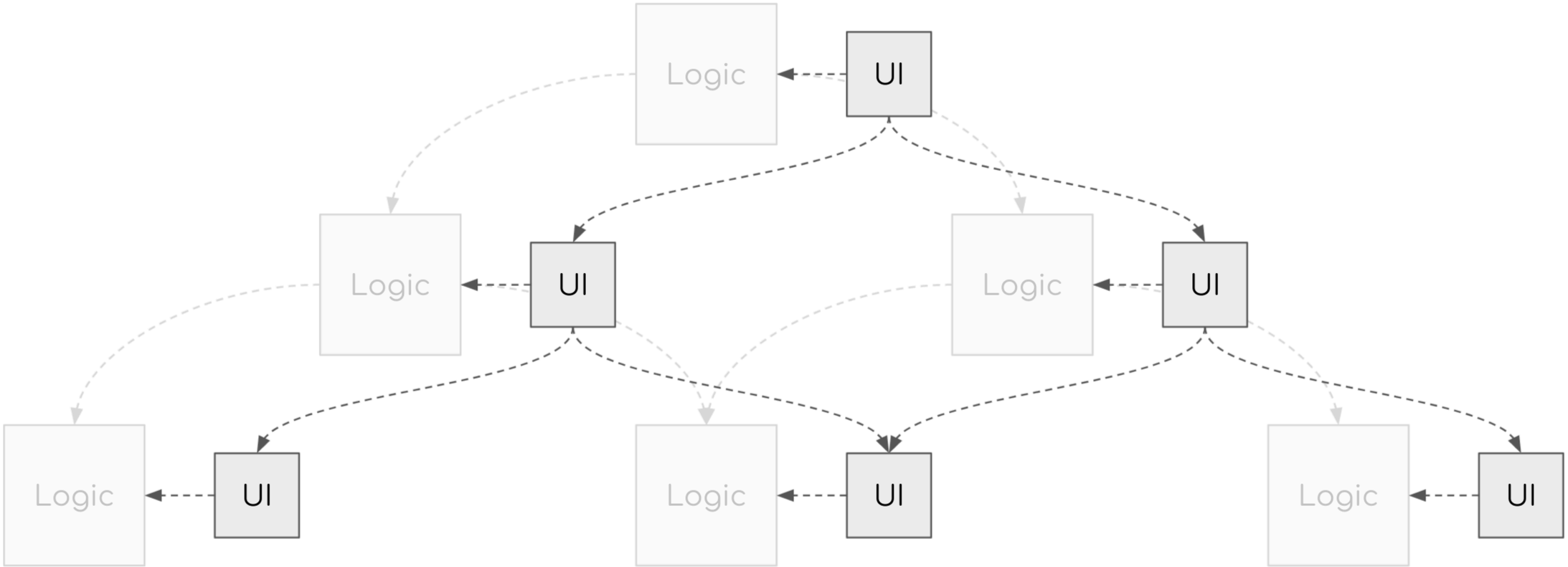
```
    childContext(key = "profile")
)
```

```
class NotifyComponent(
    componentContext: ComponentContext
) : ComponentContext by componentContext
```

Иерархия компонентов



Иерархия UI



Screen

```
abstract class ScreenComponent
```

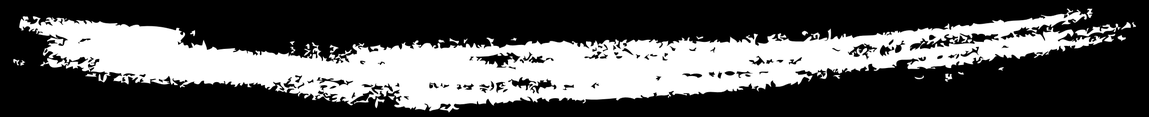
```
abstract clas
    context:
) : ComponentContext by context,
    Lifecycle.Callbacks {

    init {
```

```
abstract class ViewScreenComponent(
    context: ComponentContext,
) : ScreenComponent(context) {

    // Создание экрана
    abstract fun onCreateView(
        layoutInflater: LayoutInflater,
        parent: ViewGroup,
    ): View
}
```

```
abstract class ComposeScreenComponent(
    context: ComponentContext,
) : ScreenComponent(context) {

    // Создание экрана
    @Composable
    abstract fun Content()
}
```

Decompose
=
Fragment здорового человека 😄

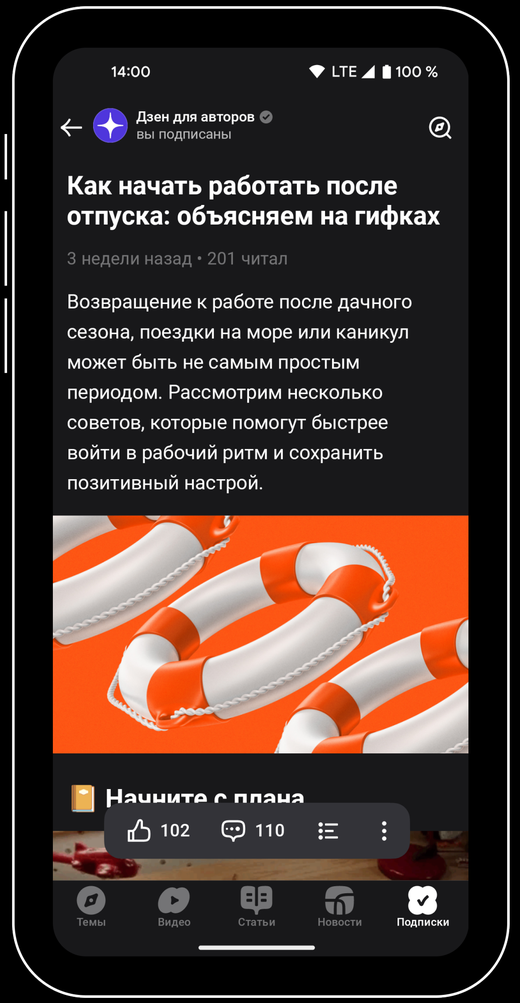
Decompose State Navigation



ScreenParams

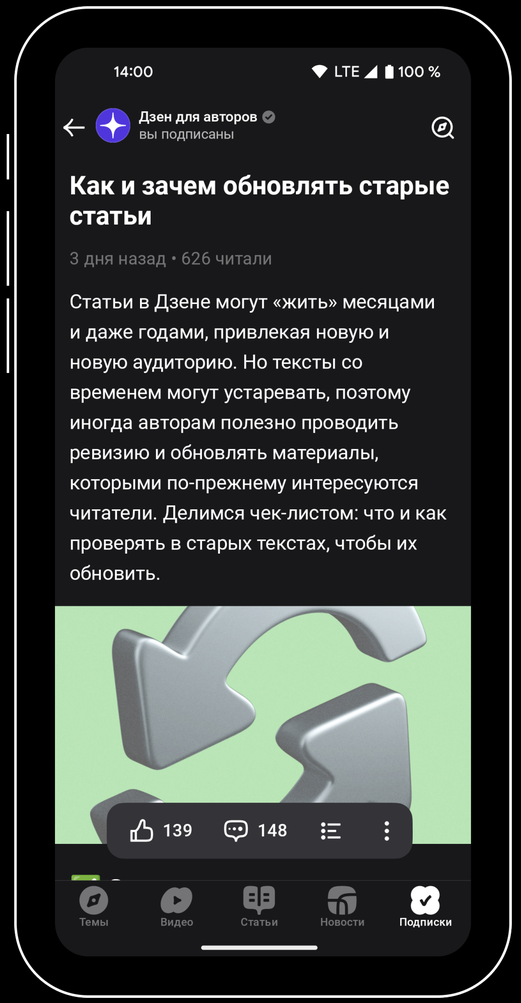
Это аргументы, с которыми открывается экран

```
ArticleScreenParams(id = "12345")
```



Один экран с разными параметрами

```
ArticleScreenParams(id = "98760")
```



```kotlin
interface ScreenParams : Parcelable

// С параметрами
@Parcelize
data class FooScreenParams(
  val id: String,
  val number: Int,
  val flag: Boolean,
) : ScreenParams

// Без параметров
@Parcelize
data object BarScreenParams : ScreenParams
```

Для навигации ScreenParams - это id экрана

```
abstract class ScreenComponent<P : ScreenParams>(
    context: ComponentContext,
    val params: P,
) : ScreenContext by context
```

StackNavigation

Через него вызываем команды навигации

```
class SampleComponent(
    context: ComponentContext
) : ComponentContext by context {

    private val navigation = StackNavigation<ScreenParams>()

}
```

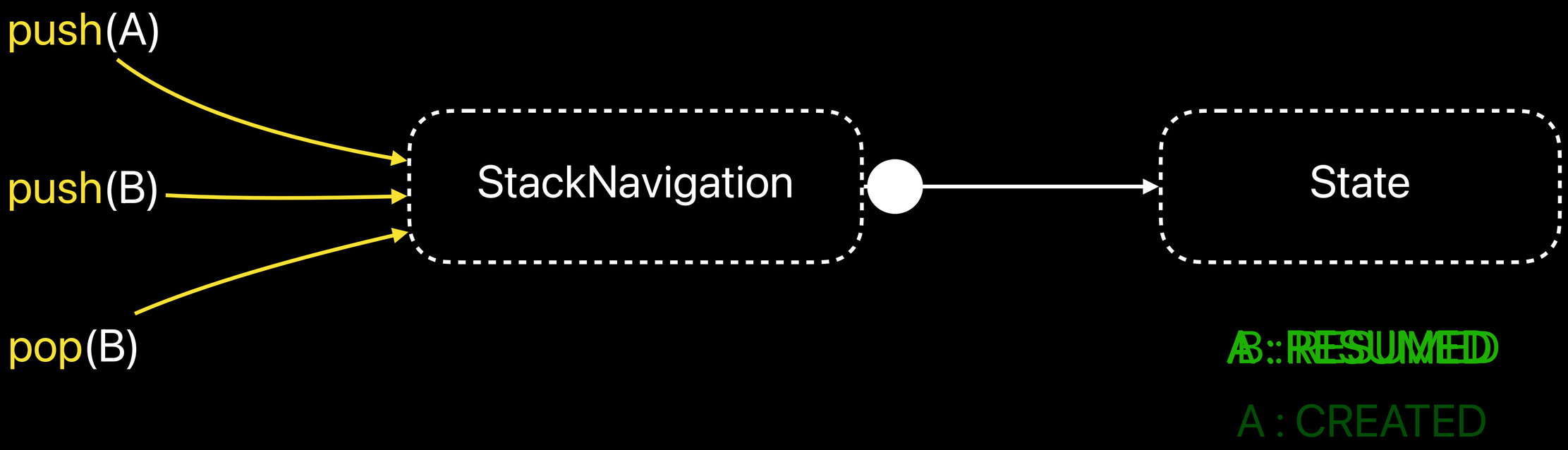
```
class StackNavigation<P> {

    fun push(params: P) { ... }

    fun pop(params: P) { ... }

    val state: Flow<StackState>
}
```

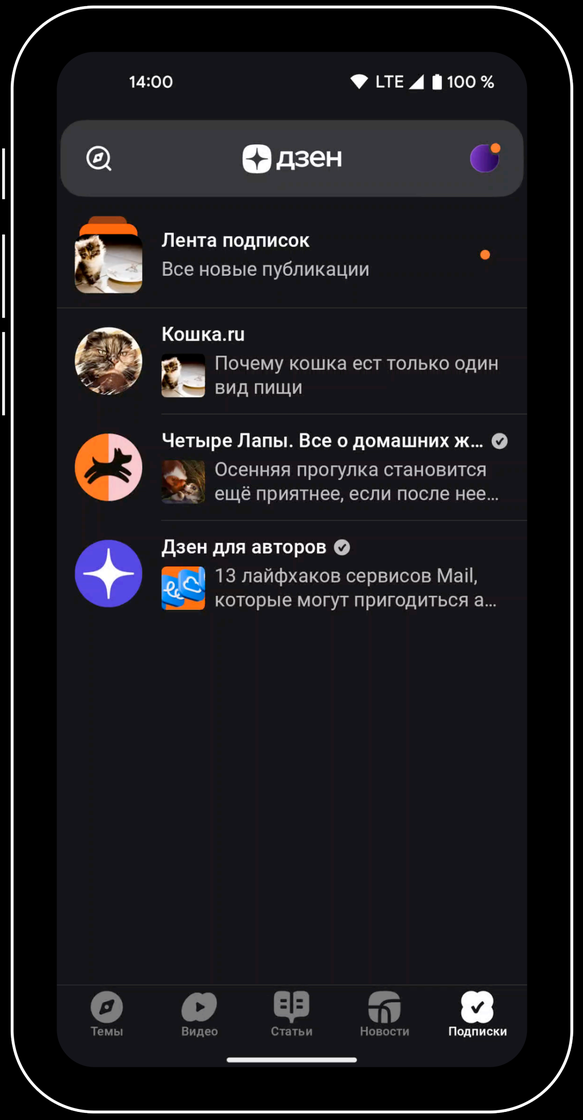
- Открыть экран сверху стека

- Удалить экран из стека

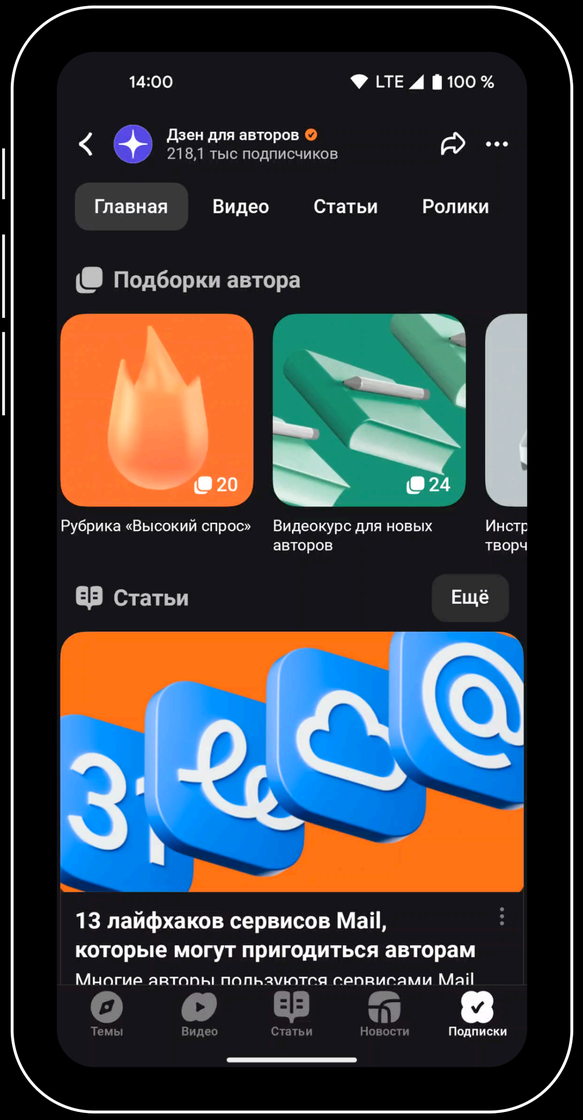
- Поток с состоянием стека

```
data class StackState<P>(val items: List<P>) {
    val children: List<Pair<P, Status>> = items.mapIndexed { index, params ->
        Pair(
            // Параметры экрана
            params,
            // ЖЦ экрана. Если верхний, то RESUMED. Остальные в backStack, те CREATED
            if (index == items.lastIndex) Status.RESUMED else Status.CREATED,
        )
    }
}
```

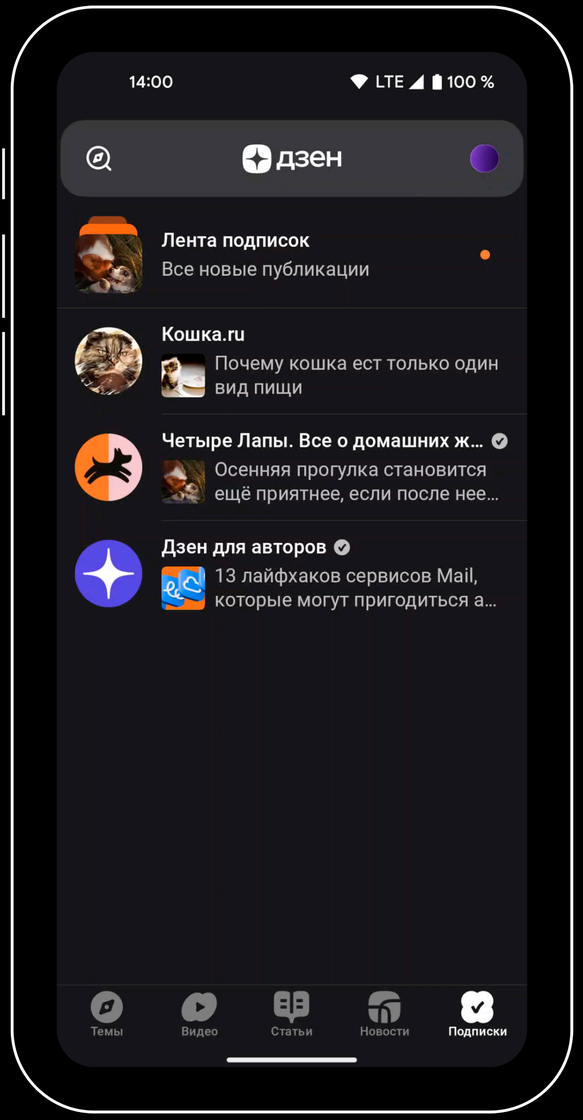

push(A)
push(B)
pop(B)
StackNavigation
State
A : CREATED


14:00
LTE
100 %
дзен
Лента подписок
Все новые публикации
Кошка.ru
Почему кошка ест только один вид пищи
Четыре Лапы. Все о домашних ж...
Осенняя прогулка становится ещё приятнее, если после нее...
Дзен для авторов
13 лайфхаков сервисов Mail, которые могут пригодиться а...
Темы
Видео
Статьи
Новости
Подписки

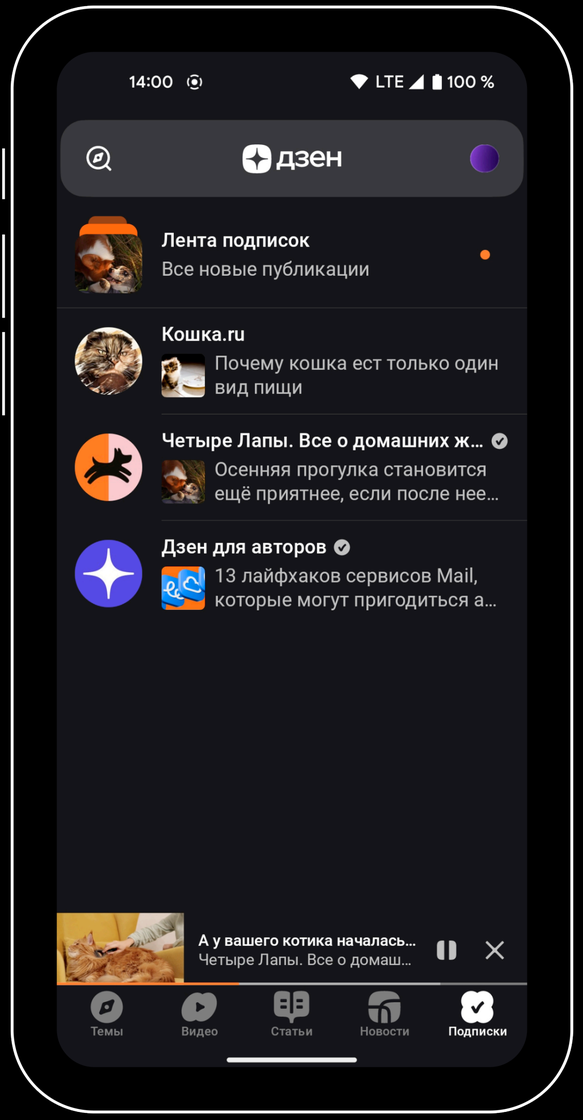
Stack


14:00
LTE
100 %
Дзен для авторов
218,1 тыс подписчиков
Главная
Видео
Статьи
Ролики
Подборки автора
20
Рубрика «Высокий спрос»
24
Видеокурс для новых авторов
Статьи
Ещё
13 лайфхаков сервисов Mail, которые могут пригодиться авторам
Многие авторы пользуются сервисами Mail
Темы
Видео
Статьи
Новости
Подписки

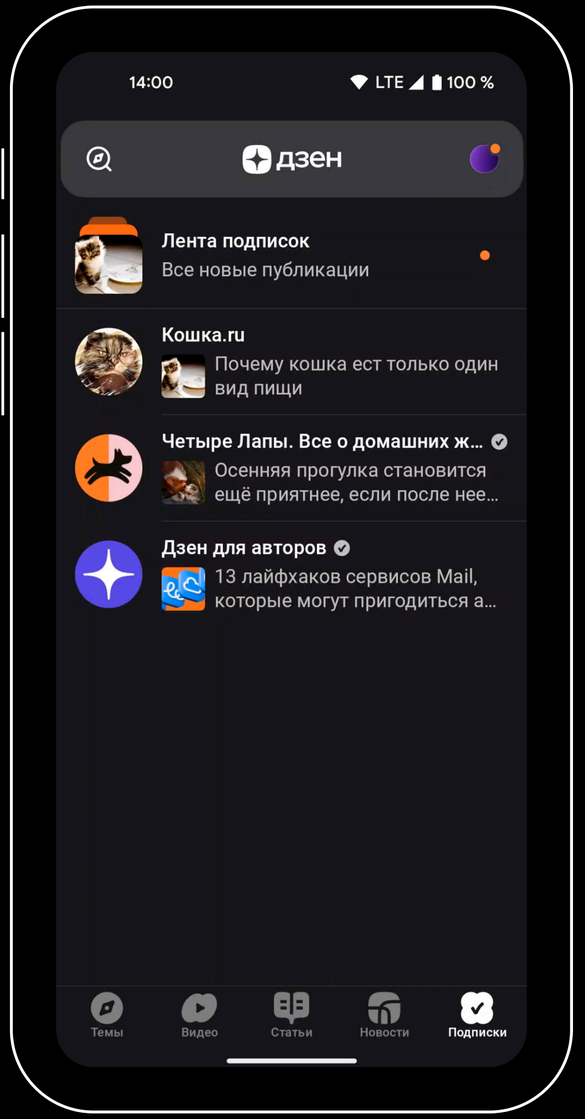
Pages

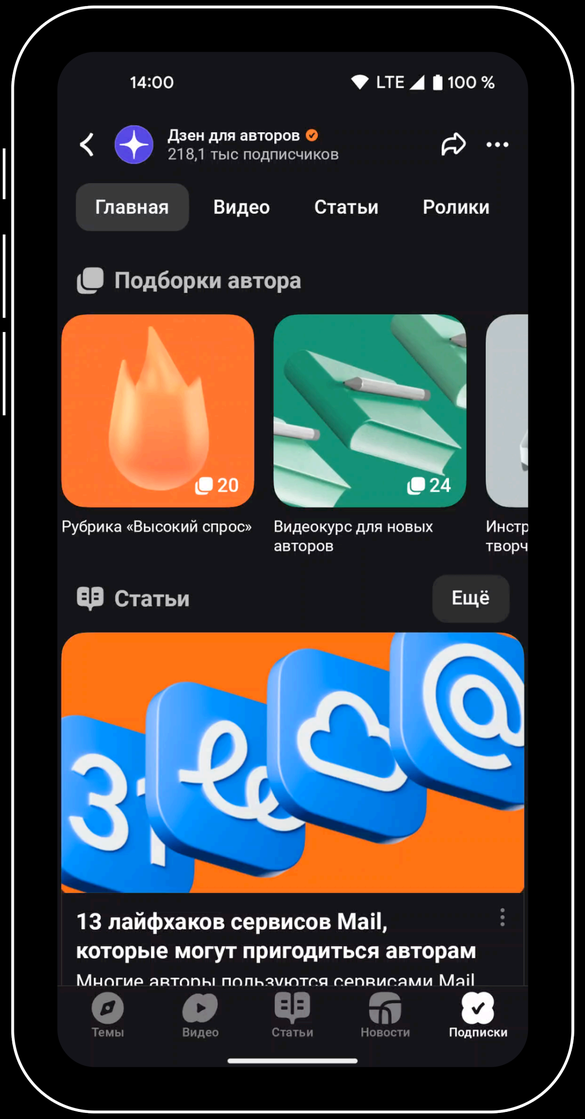

14:00
LTE
100 %
дзен
Лента подписок
Все новые публикации
Кошка.ru
Почему кошка ест только один вид пищи
Четыре Лапы. Все о домашних ж...
Осенняя прогулка становится ещё приятнее, если после нее...
Дзен для авторов
13 лайфхаков сервисов Mail, которые могут пригодиться а...
Темы
Видео
Статьи
Новости
Подписки

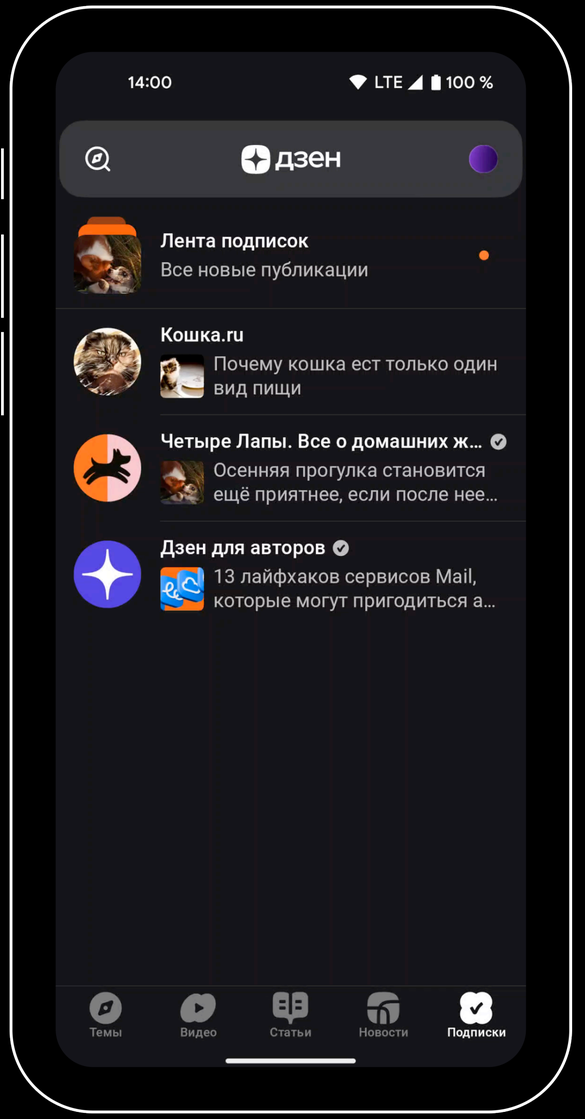
Dialogs

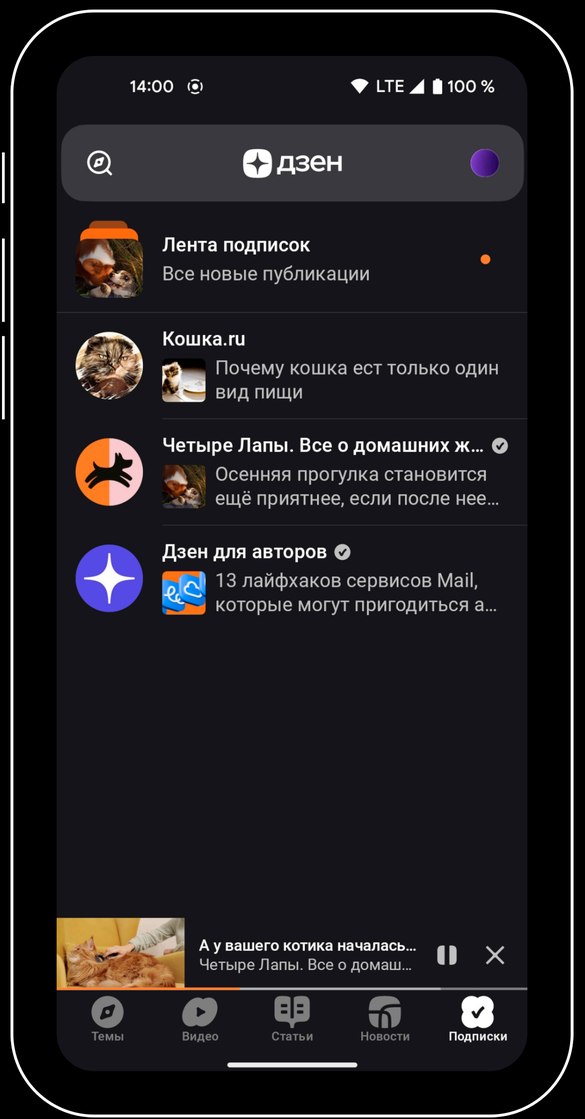

14:00
LTE
100 %
дзен
Лента подписок
Все новые публикации
Кошка.ru
Почему кошка ест только один вид пищи
Четыре Лапы. Все о домашних ж...
Осенняя прогулка становится ещё приятнее, если после нее...
Дзен для авторов
13 лайфхаков сервисов Mail, которые могут пригодиться а...
А у вашего котика началась...
Четыре Лапы. Все о домаш...
Темы
Видео
Статьи
Новости
Подписки

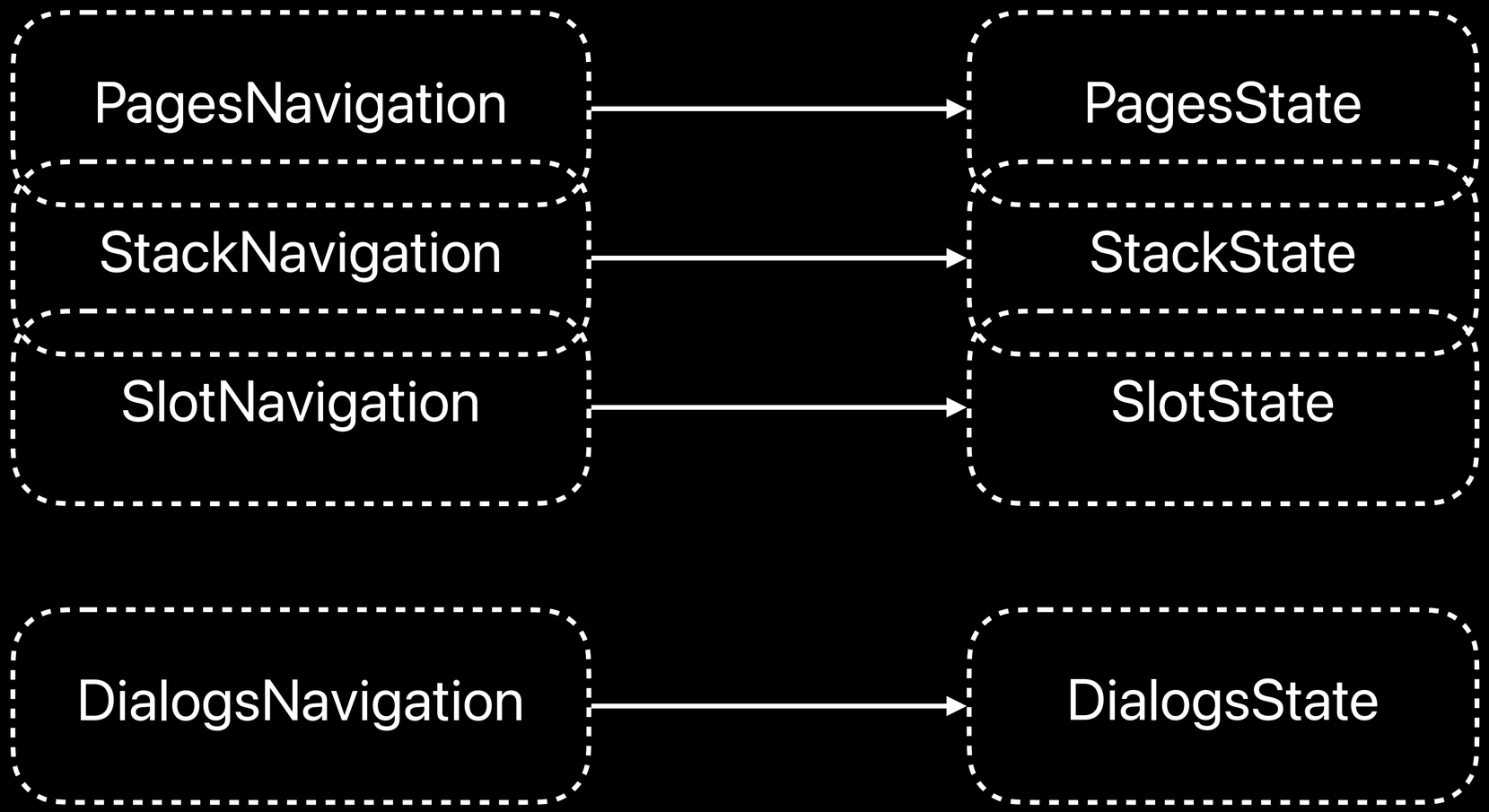
Player


14:00
LTE 100 %
дзен
Лента подписок
Все новые публикации
Кошка.ru
Почему кошка ест только один вид пищи
Четыре Лапы. Все о домашних ж...
Осенняя прогулка становится ещё приятнее, если после нее...
Дзен для авторов
13 лайфхаков сервисов Mail, которые могут пригодиться а...
Темы
Видео
Статьи
Новости
Подписки


14:00
LTE 100 %
Дзен для авторов
218,1 тыс подписчиков
Главная
Видео
Статьи
Ролики
Подборки автора
20
Рубрика «Высокий спрос»
24
Видеокурс для новых авторов
Статьи
Ещё
13 лайфхаков сервисов Mail, которые могут пригодиться авторам
Многие авторы пользуются сервисами Mail
Темы
Видео
Статьи
Новости
Подписки


14:00
LTE 100 %
дзен
Лента подписок
Все новые публикации
Кошка.ru
Почему кошка ест только один вид пищи
Четыре Лапы. Все о домашних ж...
Осенняя прогулка становится ещё приятнее, если после нее...
Дзен для авторов
13 лайфхаков сервисов Mail, которые могут пригодиться а...
Темы
Видео
Статьи
Новости
Подписки


14:00
LTE 100 %
дзен
Лента подписок
Все новые публикации
Кошка.ru
Почему кошка ест только один вид пищи
Четыре Лапы. Все о домашних ж...
Осенняя прогулка становится ещё приятнее, если после нее...
Дзен для авторов
13 лайфхаков сервисов Mail, которые могут пригодиться а...
А у вашего котика началась...
Четыре Лапы. Все о домаш...
Темы
Видео
Статьи
Новости
Подписки


PagesNavigation
PagesState
StackNavigation
StackState
SlotNavigation
SlotState
DialogsNavigation
DialogsState

```
data class StackState<P>(
    val items: List<P>
)
```

```
data class SlotState<P>(
    val item: P?
)
```

```
data class PagesState<P>(
    val items: List<P>,
    val selected: Int,
)
```

```
data class DialogsState<P>(
    // Может быть пустым
    val items: List<P>?
)
```

StackHost

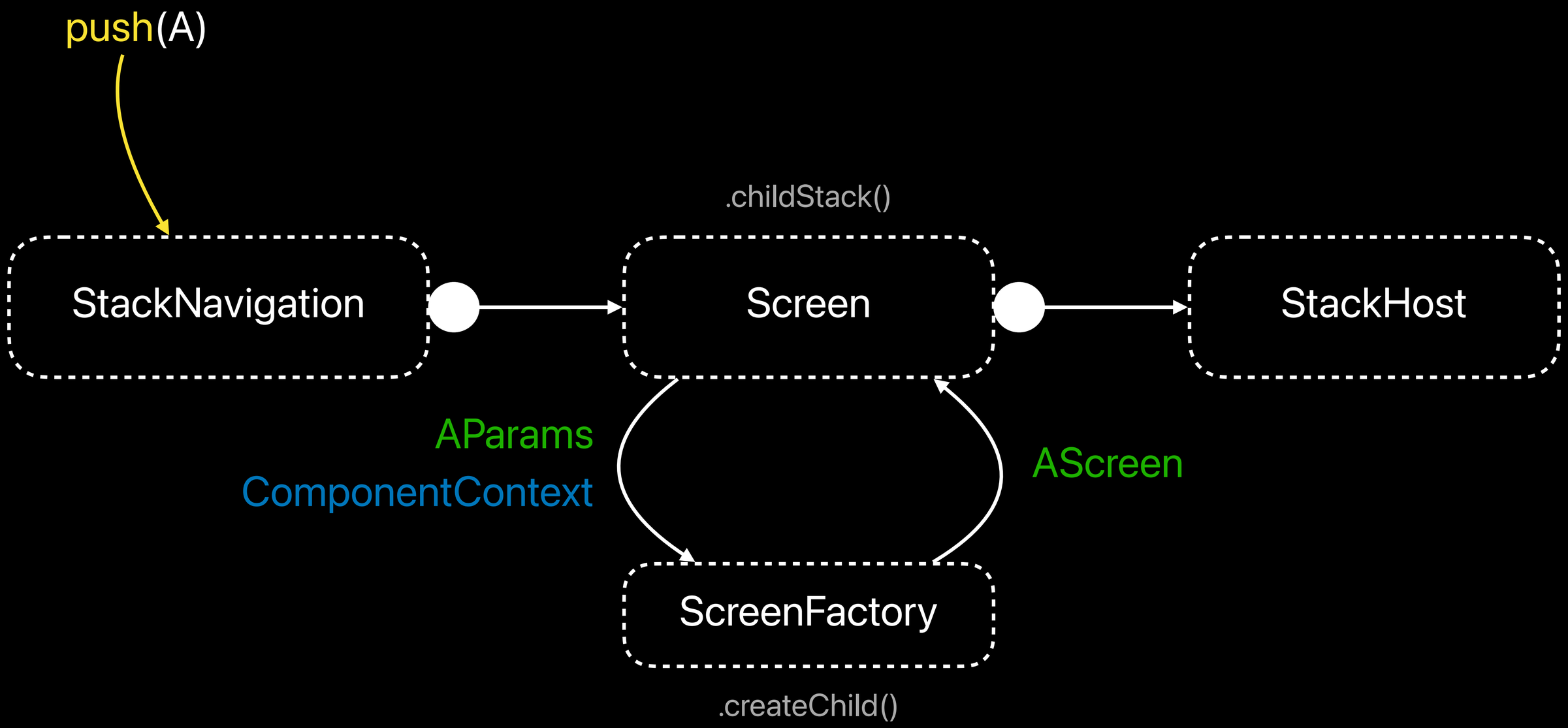
Через него отображаем экраны

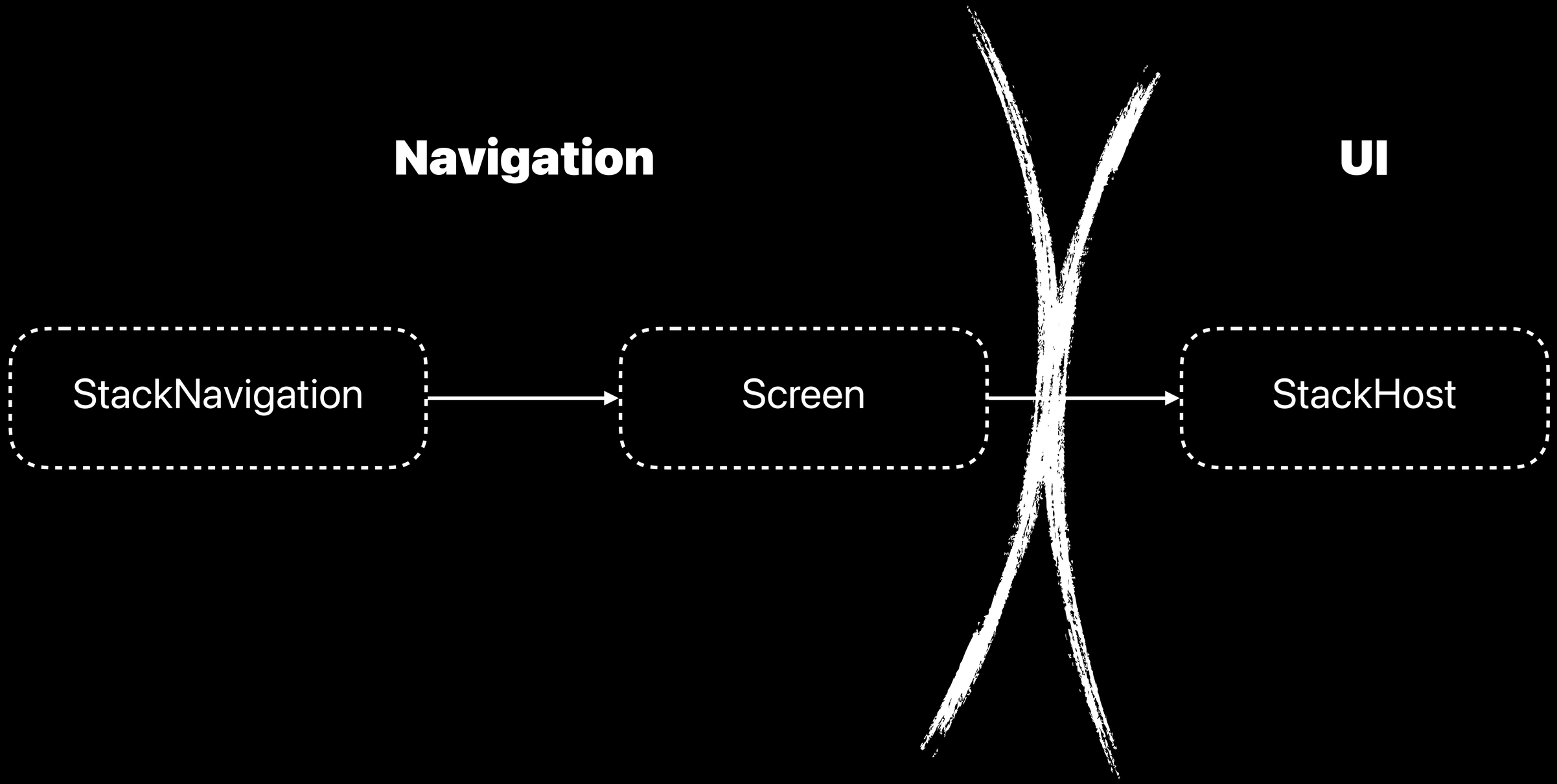
```
class SampleComponent(
    context: ComponentContext
) : ComponentContext by context {

    private val navigation = StackNavigation<ScreenParams>()

    private val childStack: StateFlow<ChildStack> =
        childStack(
            source = navigation,
            childFactory = ::createChild,
        )

    fun createChild(params: ScreenParams, context: ComponentContext) =
        when (params) {
            is AParams -> AScreenComponent(context, params)
            is BParams -> BScreenComponent(context, params)
        }
}
```

Дочерний context

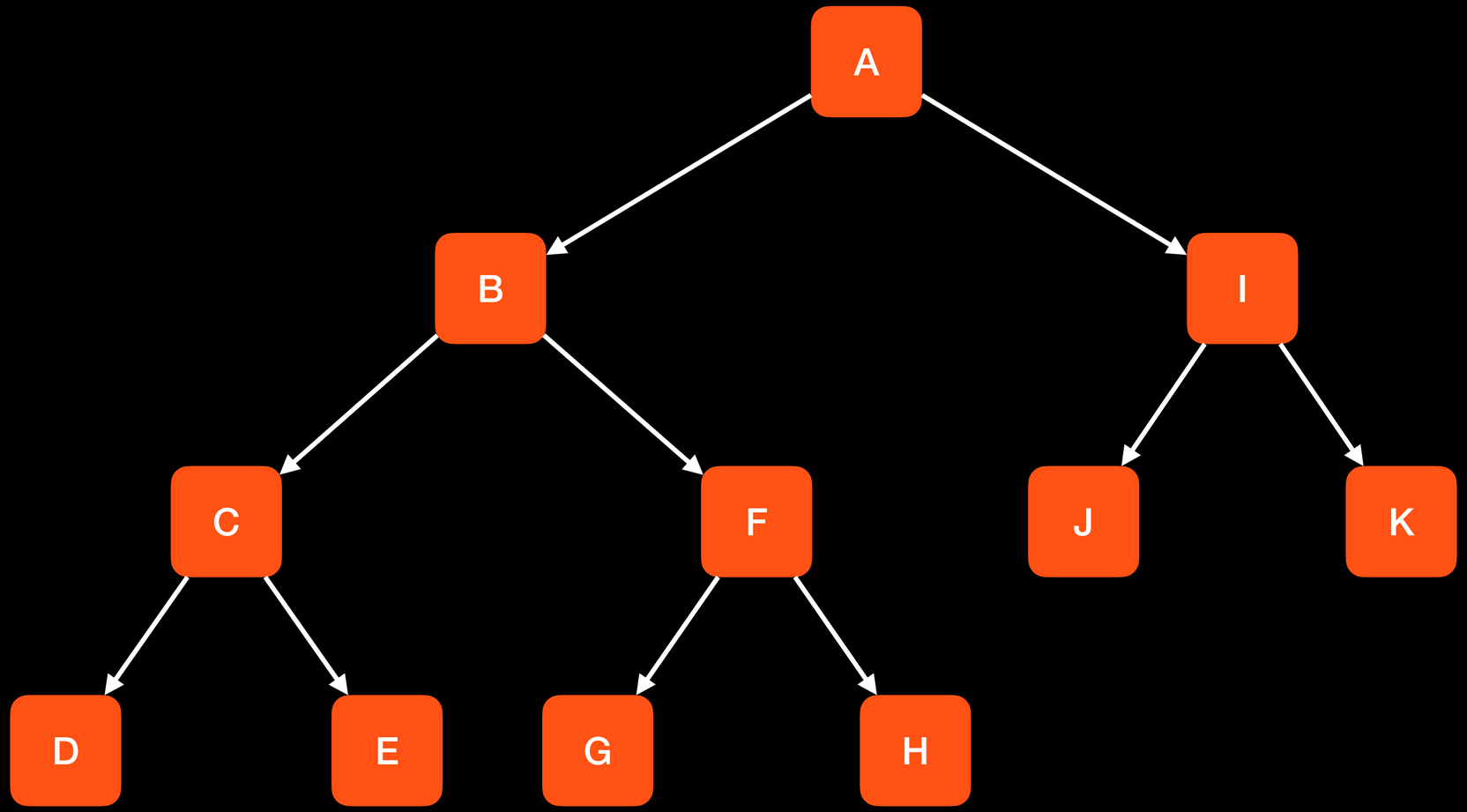
```
class SampleComponent(
    context: ComponentContext
) : ComponentContext by context {

    private val navigation = StackNavigation<ScreenParams>()

    private val childStack: StateFlow<ChildStack> = childStack(...)

    @Composable
    fun Content() {
        val stack by childStack.subscribeAsState()
        StackHost(stack)
    }
}
```

```
@Composeble
fun StackHost(stack: ChildStack) {
    val active = stack.last()
    when(active) {
        is ComposeScreenComponent -> {
            active.Content()
        }
        is ViewScreenComponent -> {
            AndroidView { context ->
                active.createView(context)
            }
        }
    }
}
```

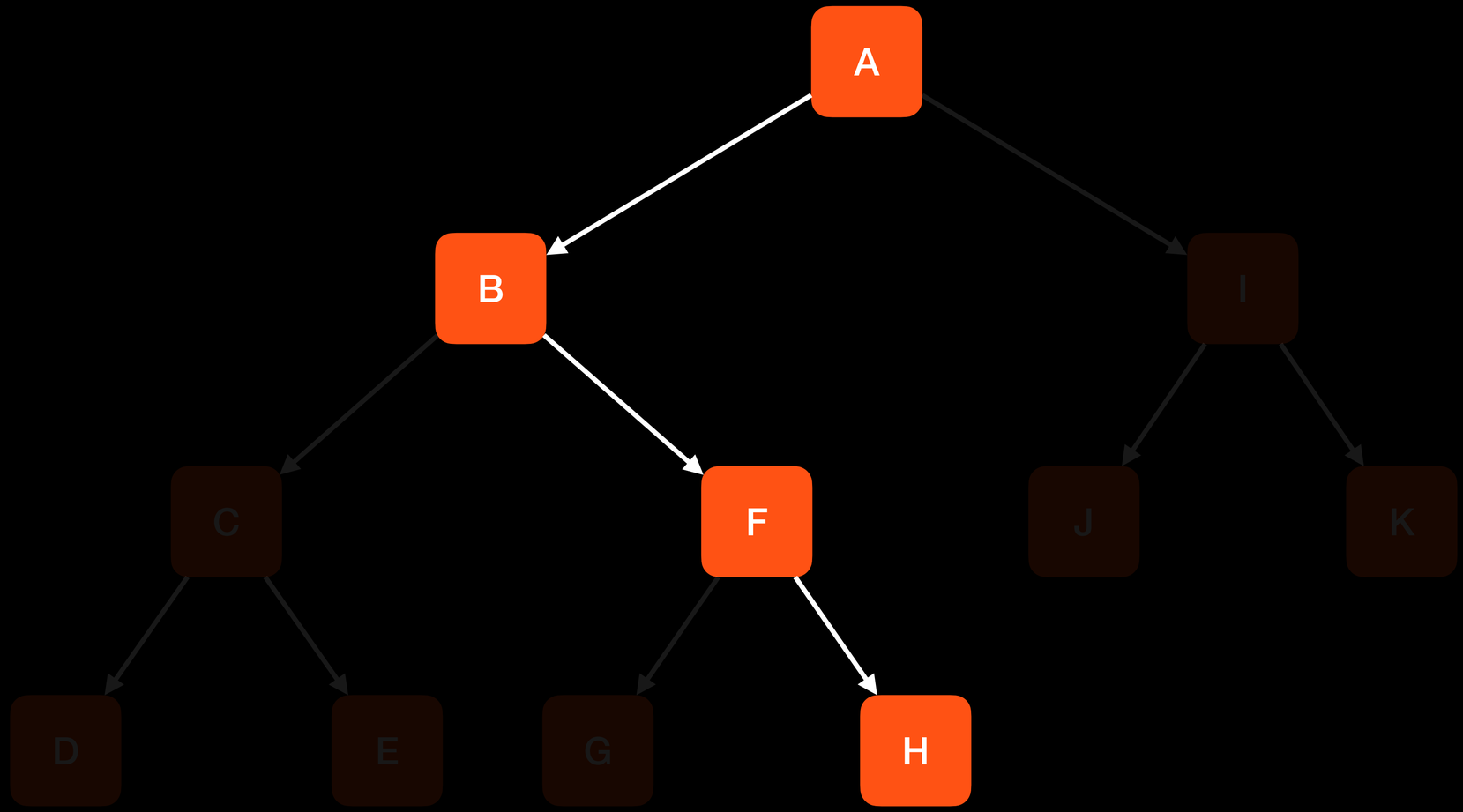

push(A)
.childStack()
StackNavigation
Screen
StackHost
AParams
ComponentContext
AScreen
ScreenFactory
.createChild()


Navigation
UI
StackNavigation
Screen
StackHost

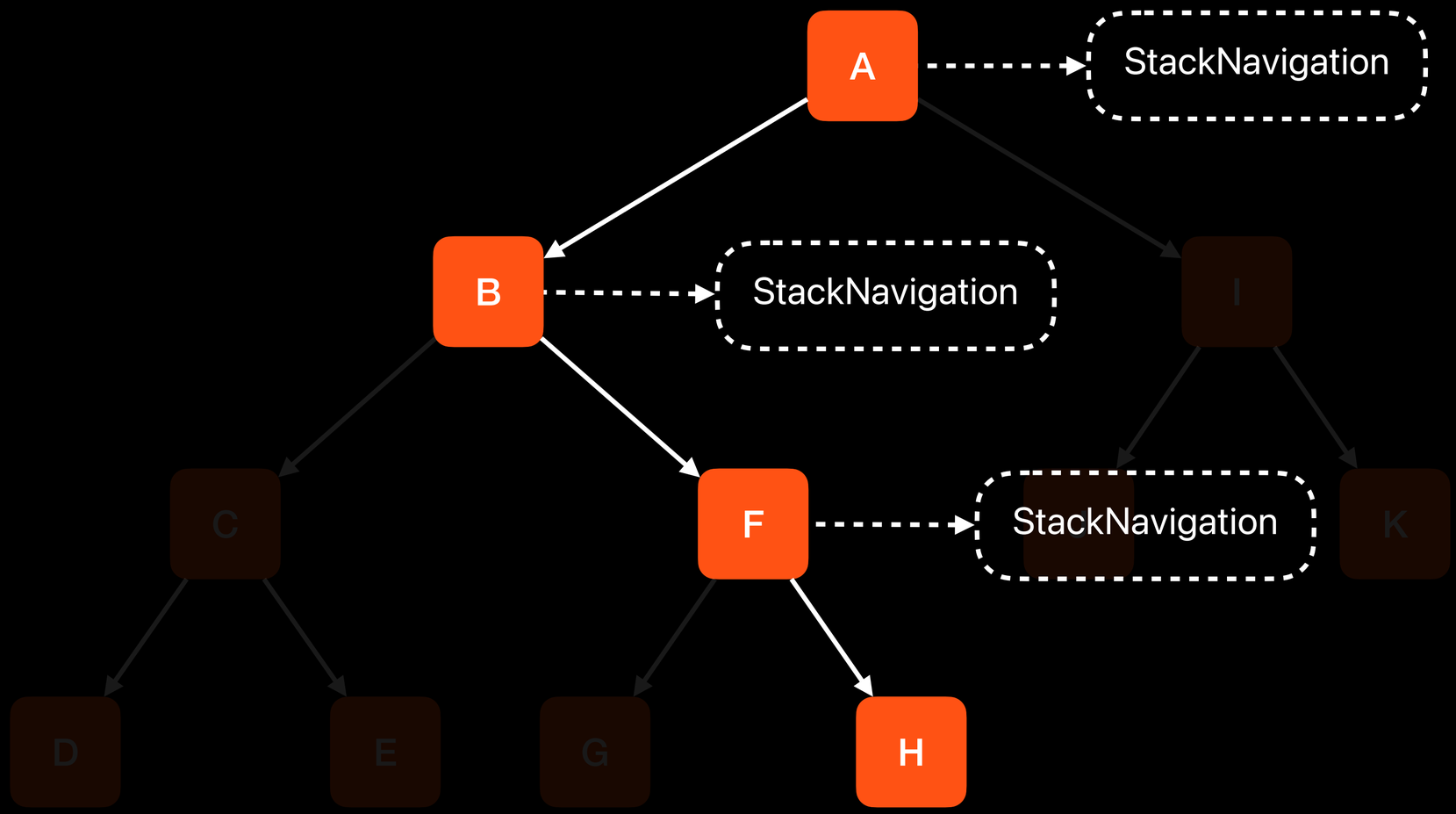
Иерархия экранов



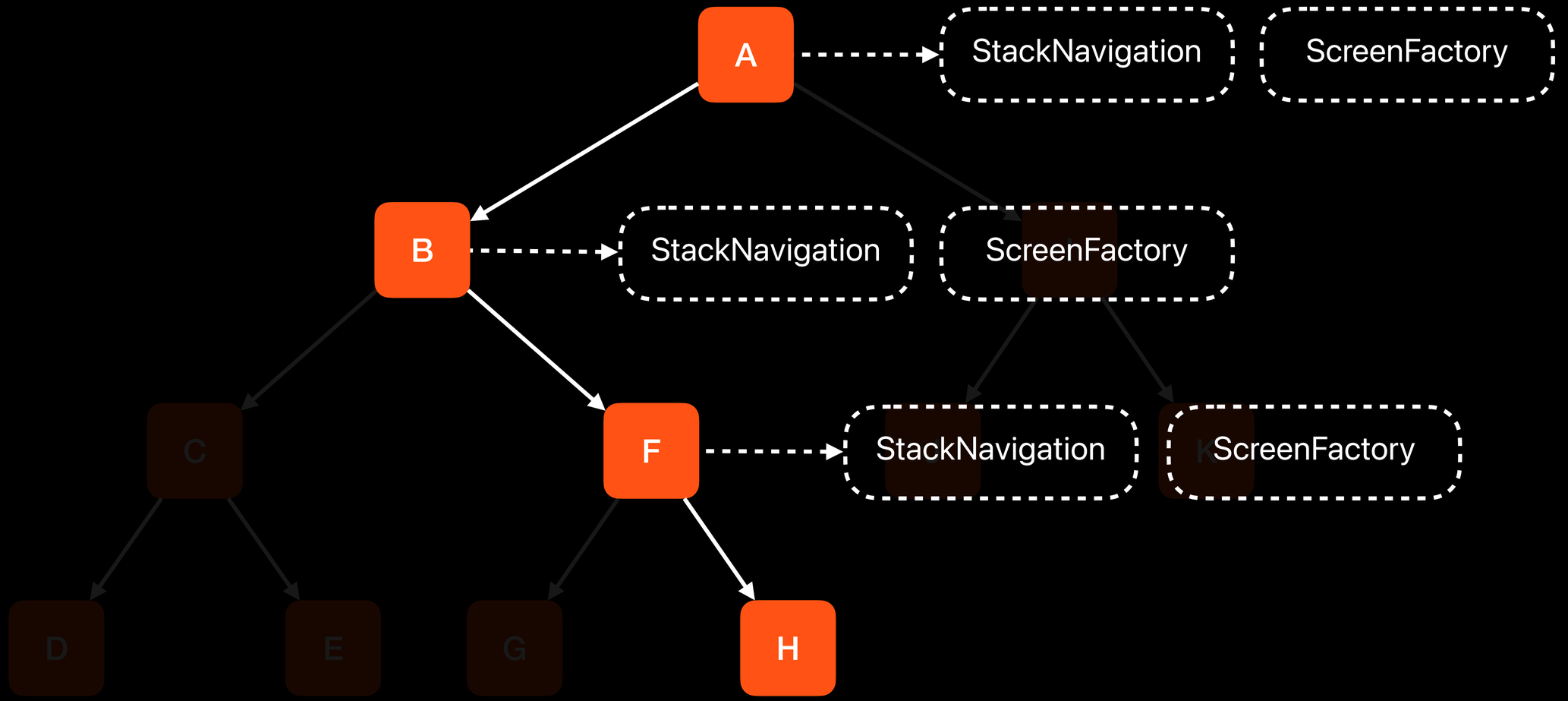
Открытые экраны



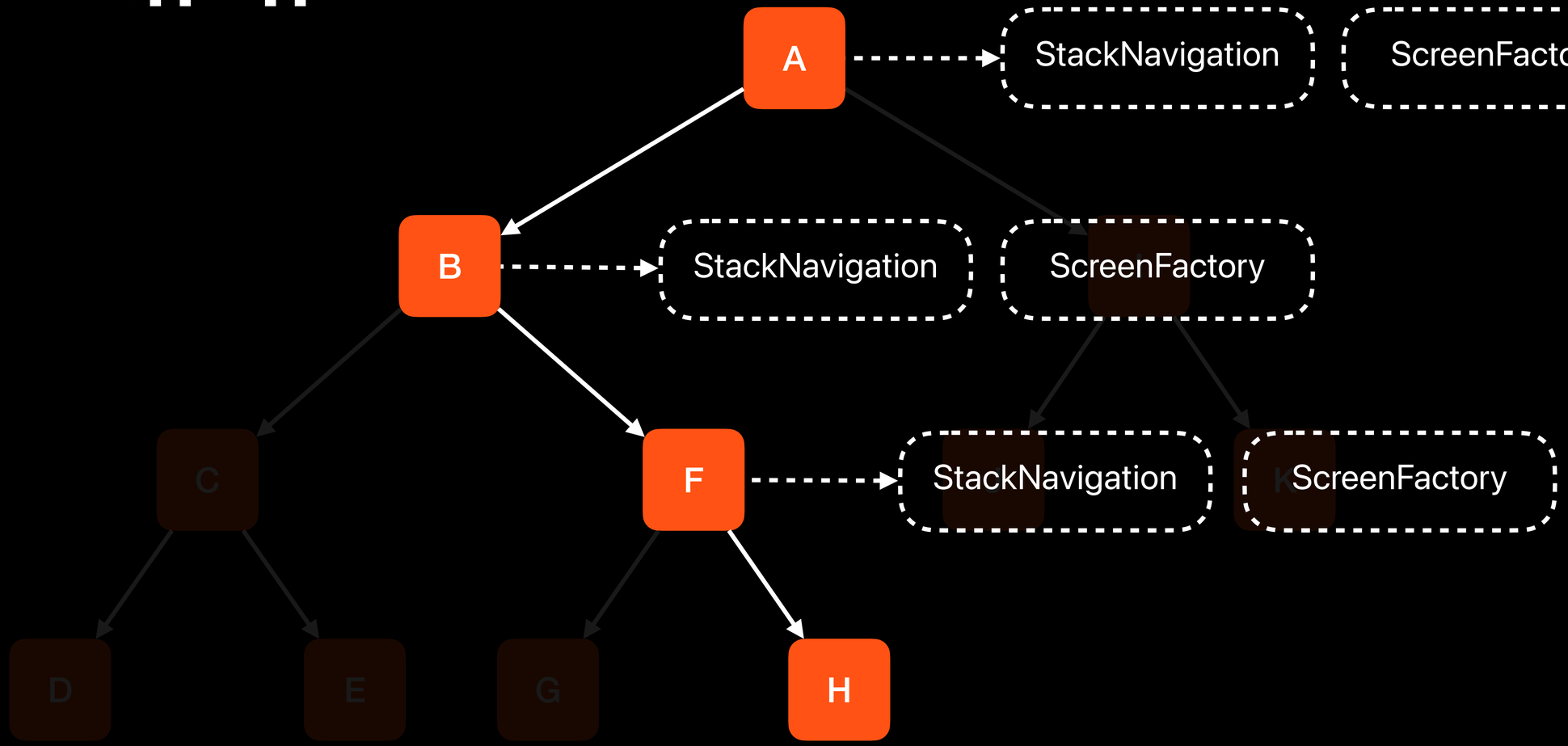
Navigation

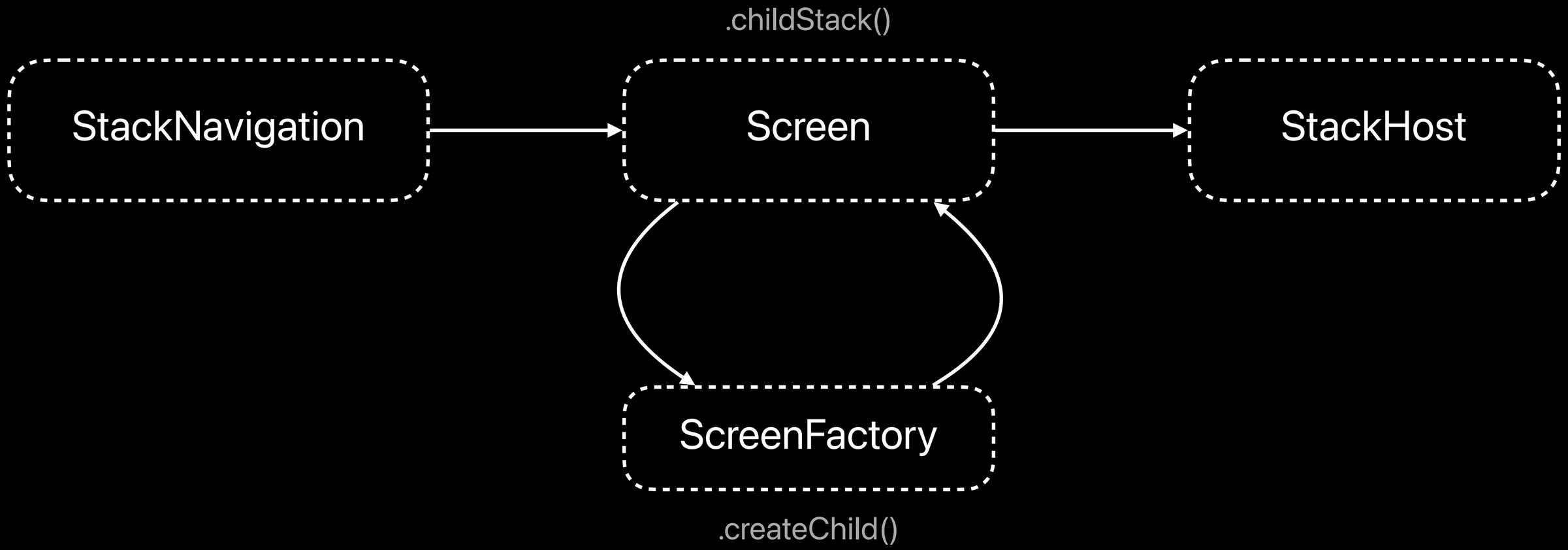

A
StackNavigation
B
StackNavigation
I
C
F
StackNavigation
K
D
E
G
H

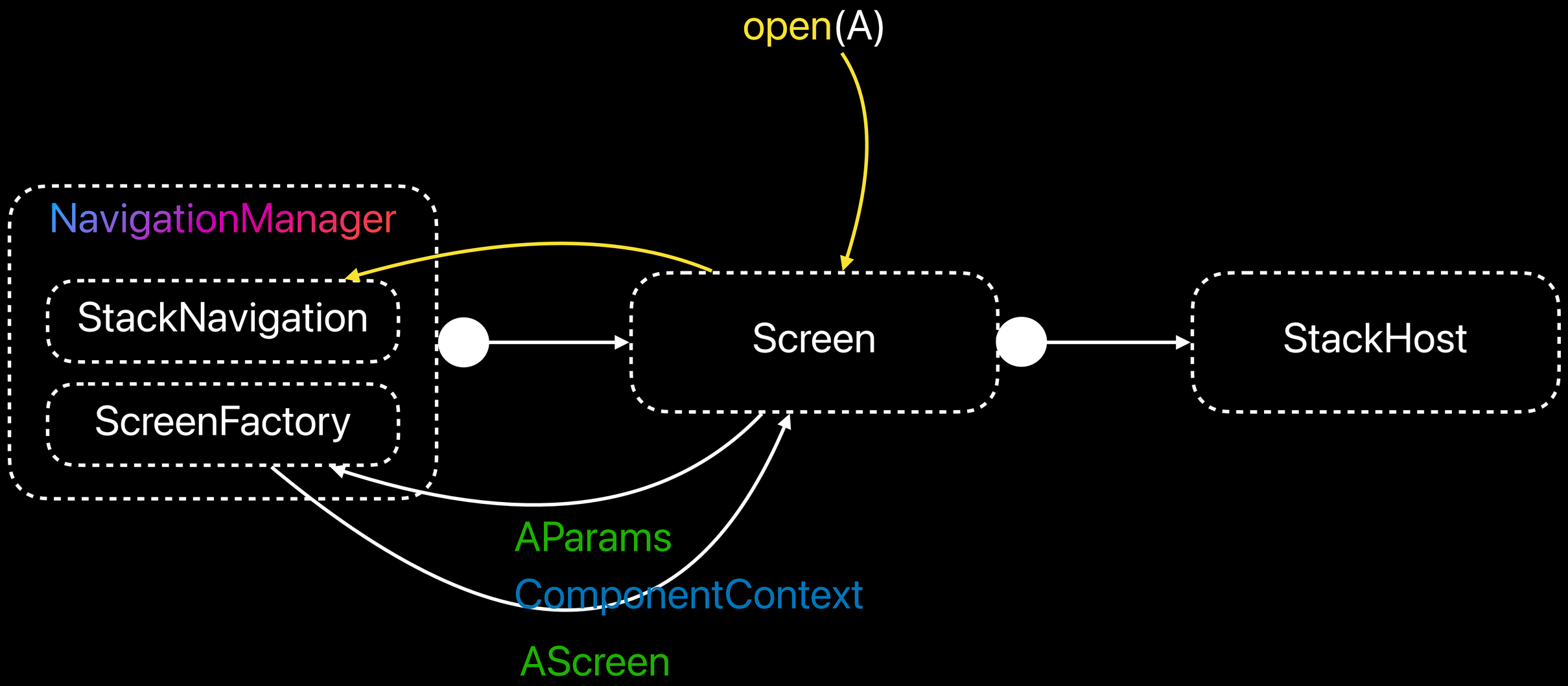
ScreenFactory



Не подходит


A
StackNavigation
ScreenFactory
B
StackNavigation
ScreenFactory
C
F
StackNavigation
ScreenFactory
D
E
G
H


.childStack()
StackNavigation
Screen
StackHost
ScreenFactory
.createChild()


open(A)
NavigationManager
StackNavigation
ScreenFactory
Screen
StackHost
AParams
ComponentContext
AScreen

```kotlin
// Контекст с навигацией
interface ScreenContext : ComponentContext {
    val navigationManager: NavigationManager
}
```

- NavigationManager в Context

```kotlin
abstract class ScreenComponent<P : ScreenParams>(
    context: ComponentContext,
    context: ScreenContext,
    val params: P,
) : ComponentContext by context
) : ScreenContext by context
```

- Наследуем от ScreenContext

```
fun ScreenContext.stack(): StateFlow<ChildStack> {
    return childStack(
        source = navigationManager.getStackNavigation(),
        childFactory = navigationManager::createScreen,
    )
}
```

- Navigation и Factory через NavigationManager

```
class SampleComponent(
    context: ScreenContext,
    params: SampleScreenParams
): ScreenComponent<SampleScreenParams>(context, params) {

    private val stack: StateFlow<ChildStack> = stack()

}
```

~~Push~~

~~Pop~~

Open&Close

Stack

Текущий	+	Команда	->	Новый
[A, B]		open(C)		[A, B, C]
[A, B]		close(B)		[A]

Stack

Текущий	+	Команда	->	Новый
[A, B, C]		open(A)		[A]
[A, B, C]		close(D)		[A, B, C]

Pages

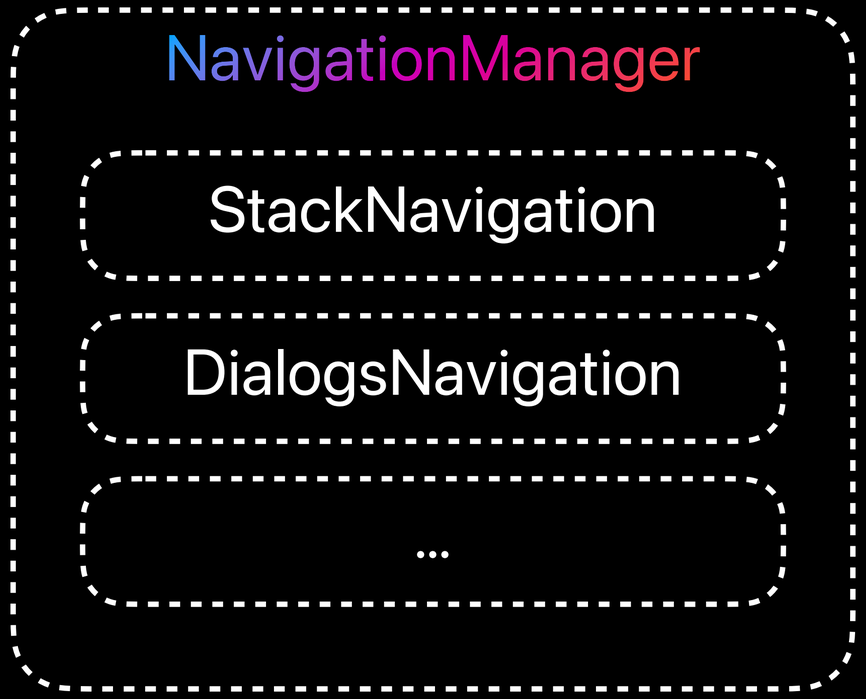
Текущий	+	Команда	->	Новый
[A, B, C] Selected: 2		open(A)		[A, B, C] Selected: 0
[A, B, C] Selected: 2		close(C) Выбираем предыдущий		[A, B, C] Selected: 1

```
class SmpleComponent(
    context: ScreenContext,
    params: SampleScreenParams,
): ScreenComponent<SampleScreenParams>(context, params) {

    val stack = stack()
    val dialogs = dialogs()

    @Composable
    fun Content() {
        Box {
            StackHost(stack)
            DialogsHost(dialogs)
        }
    }
}
```

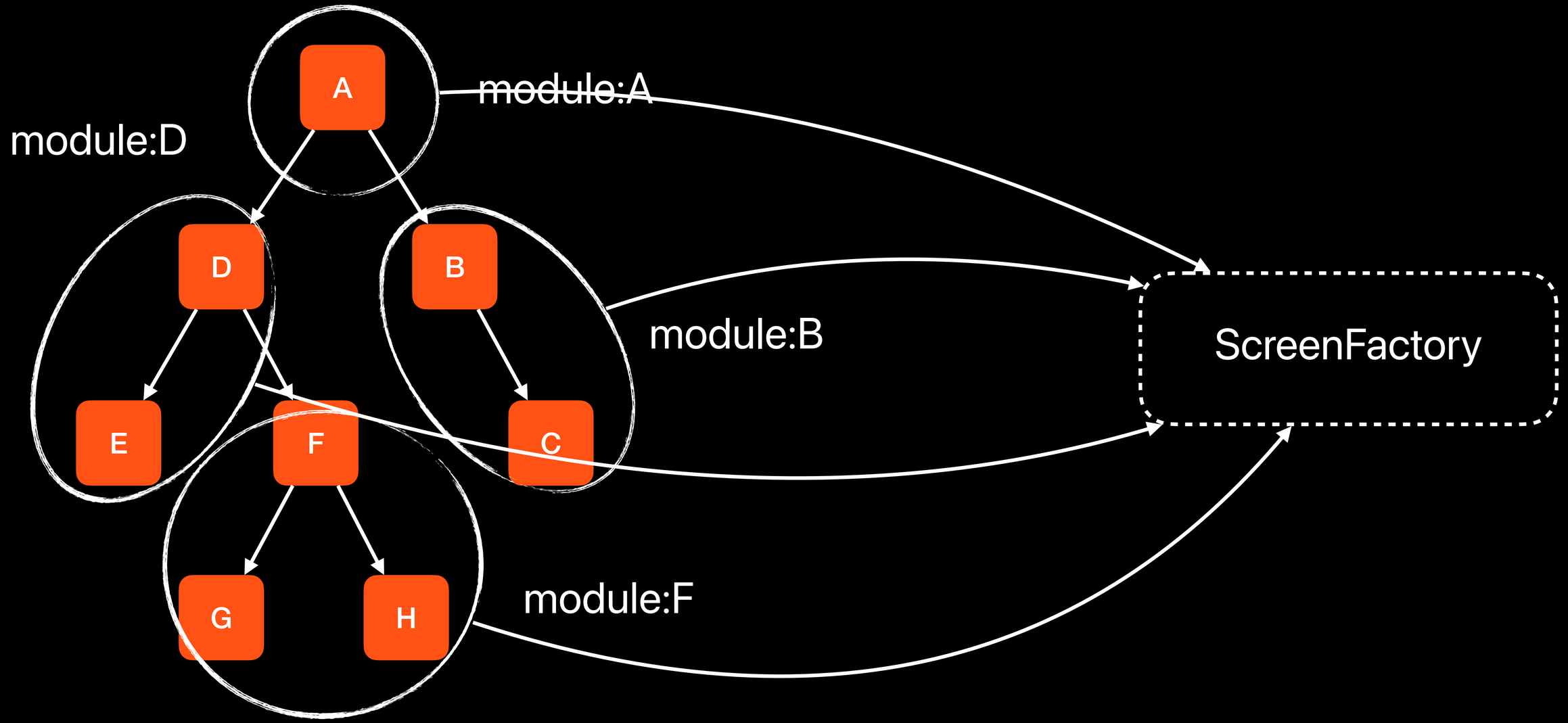
- Содержит одновременно Stack и Dialogs

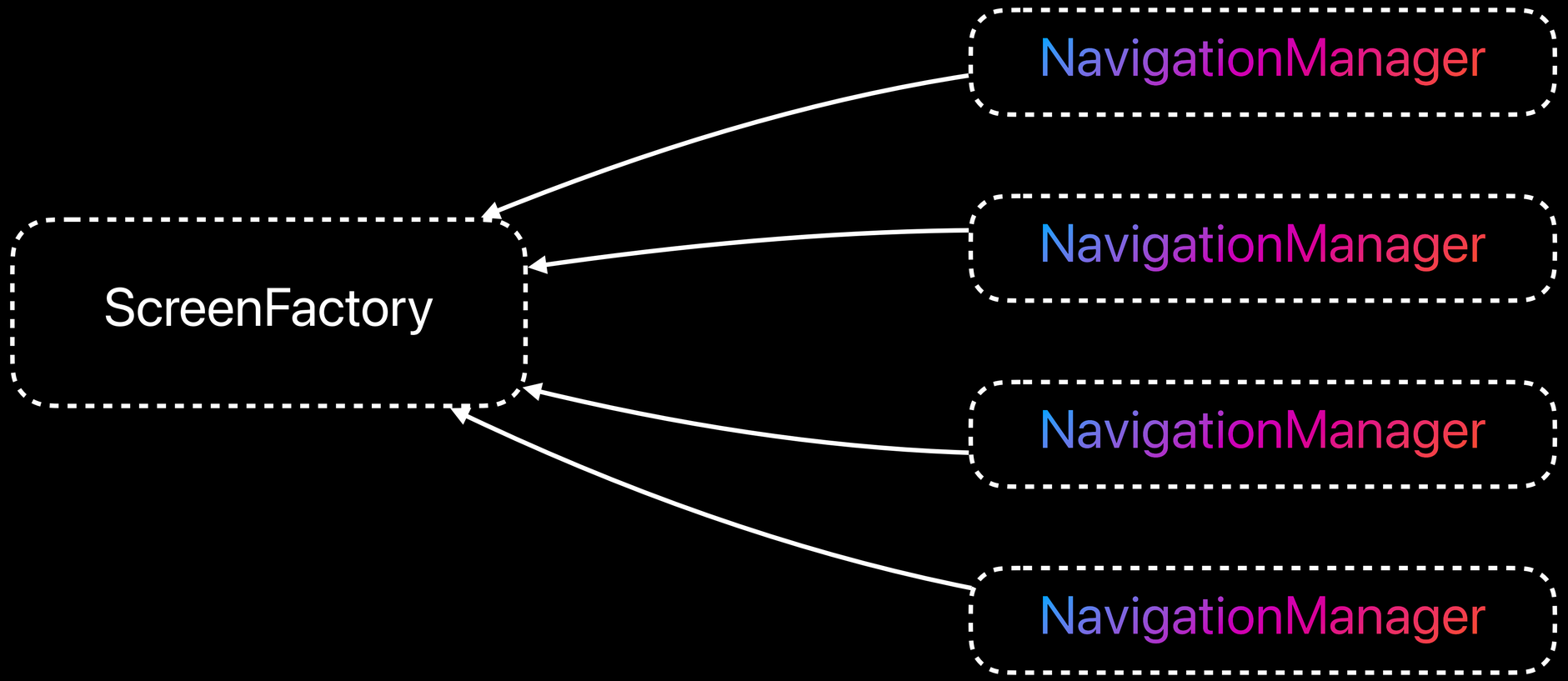
- UI содержит два Host

```
registerNavigation(SampleScreenParams::class) {
    // A и B откроются в StackHost
    stack(A::class, B::class)
    // C и D откроются в DialogsHost
    dialogs(C::class, D::class)
}
```


NavigationManager
StackNavigation
DialogsNavigation
...

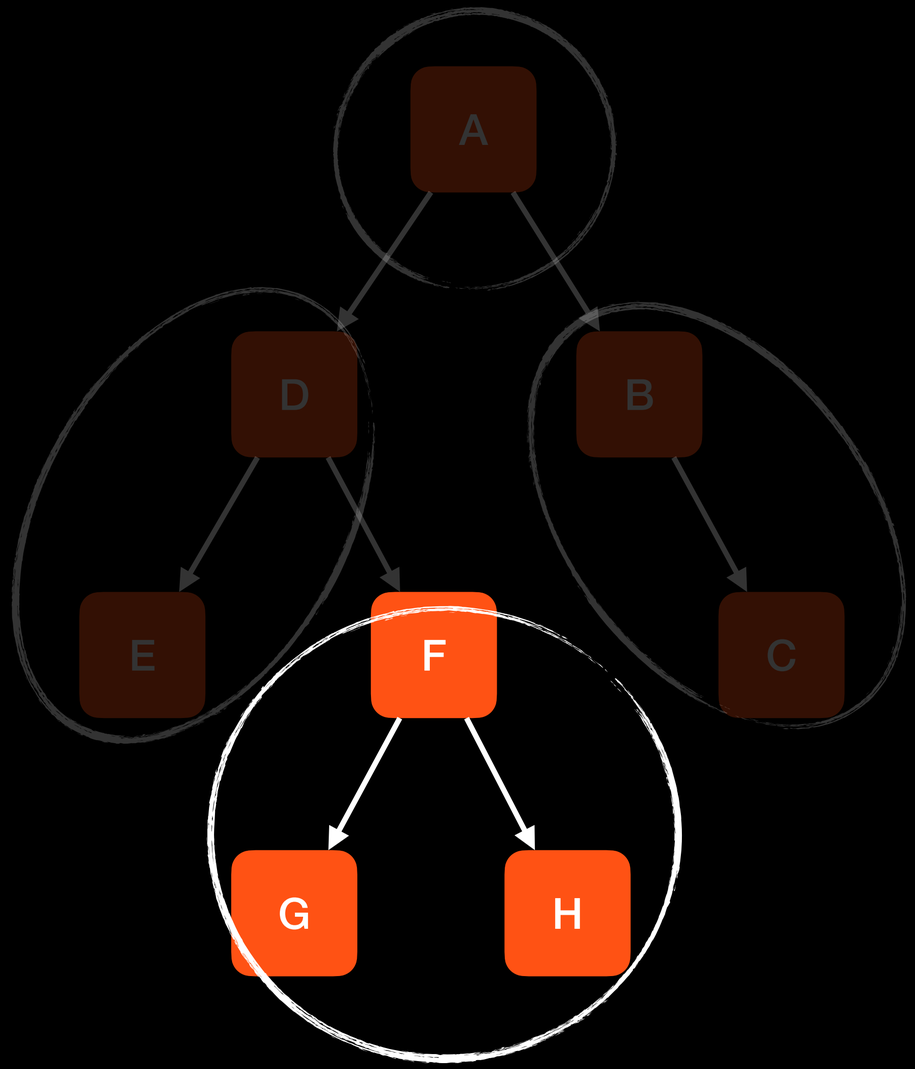
Глобальная фабрика


A
module:A
module:D
D
B
module:B
ScreenFactory
E
F
C
G
H
module:F

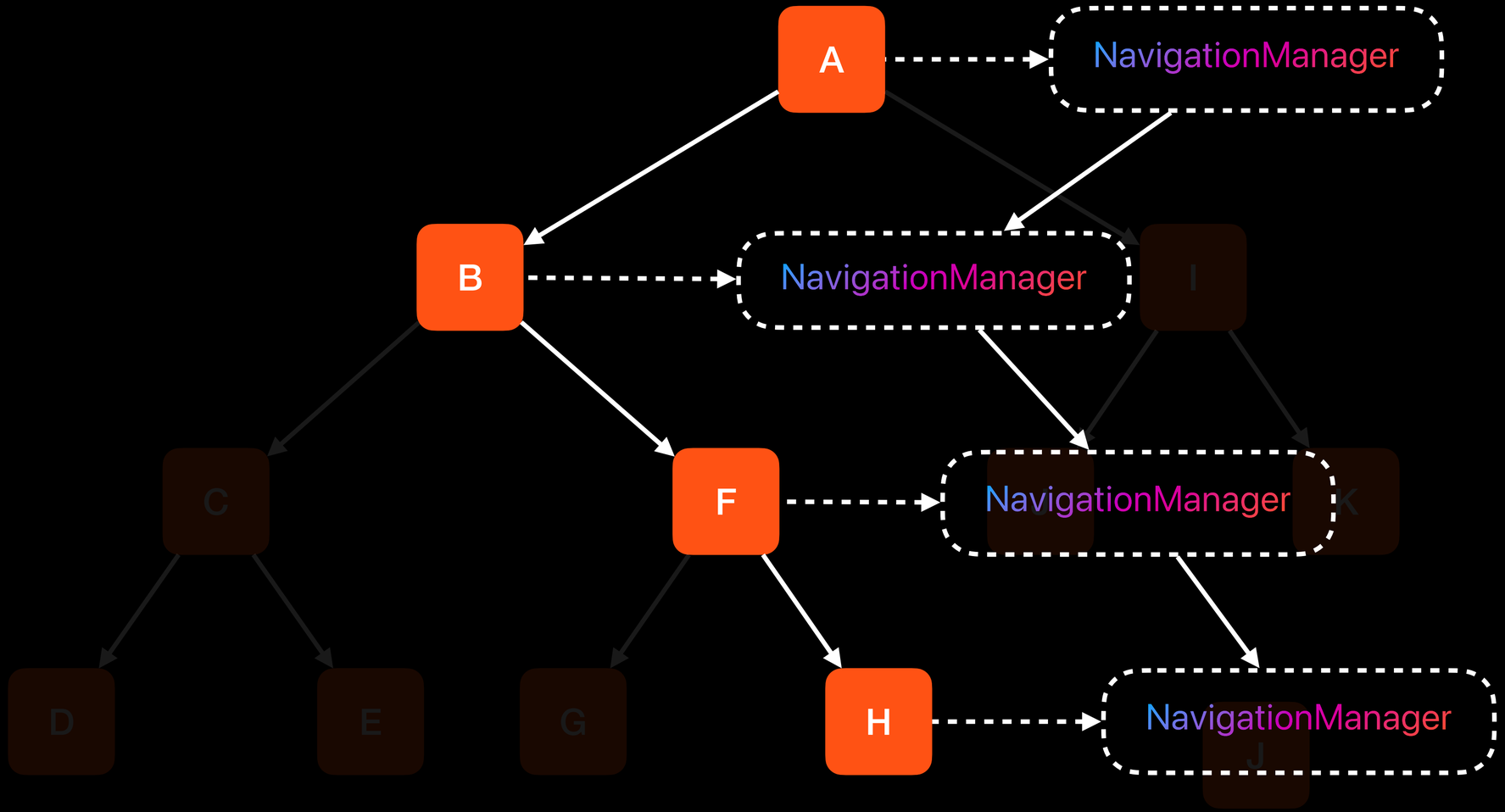

NavigationManager
NavigationManager
ScreenFactory
NavigationManager
NavigationManager

Регистрация создания экрана в фабрике

```
registerFactory(FScreenParams::class) { context: ScreenContex, params: FScreenParams →
    FScreenComponent(context, params)
}
```




```
// module:F
registerFactory(FScreenParams::class) { ... }
registerFactory(GScreenParams::class) { ... }
registerFactory(HScreenParams::class) { ... }

registerNavigation(FScreenParams::class) {
    stack(GScreenParams::class, HScreenParams::class)
}
```


A
NavigationManager
B
NavigationManager
C
F
NavigationManager
D
E
G
H
NavigationManager

NavigationManager

- Повторяют иерархию экранов
- Знают какие экраны могут открыть
- Общий интерфейс навигации open&close

Навигация пройдет путь за нас!

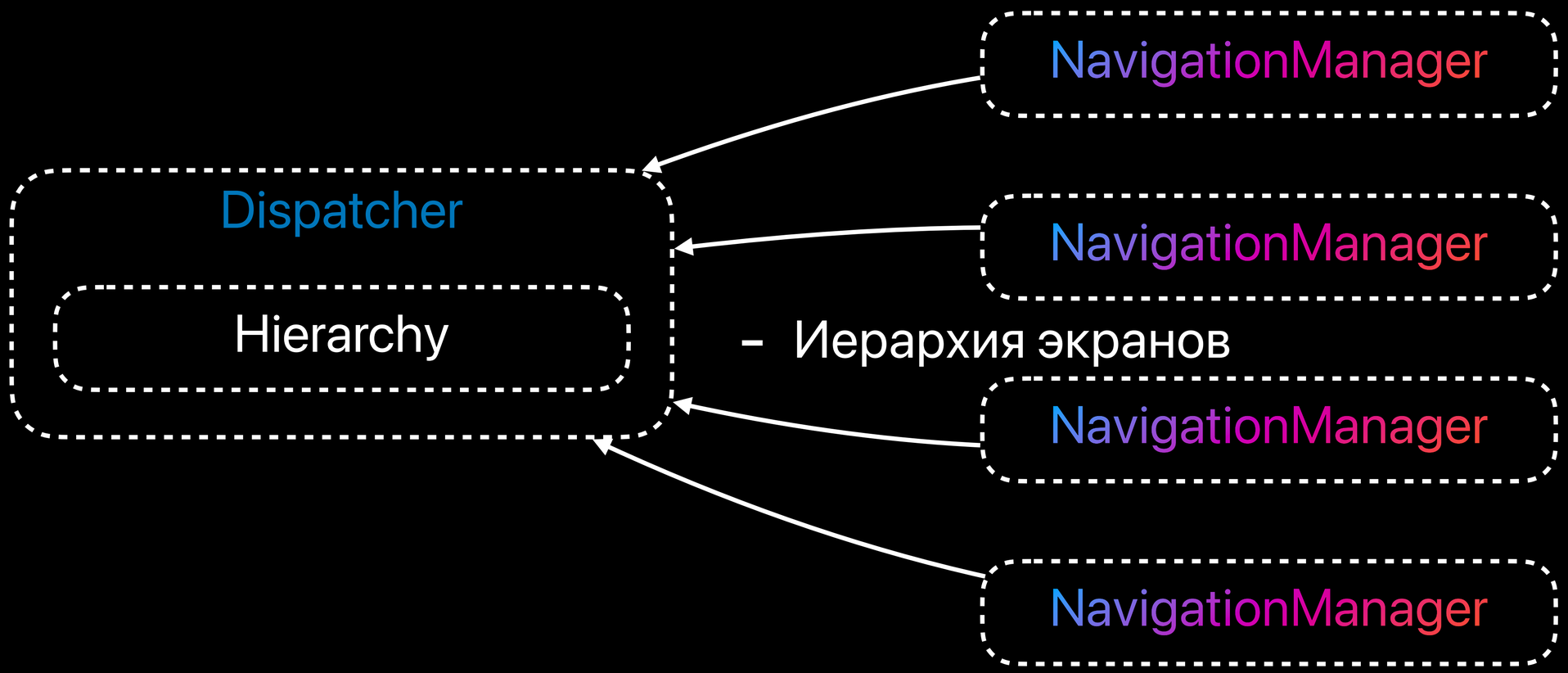
✅ 1. Иерархия экранов

✅ 2. Алгоритм поиска

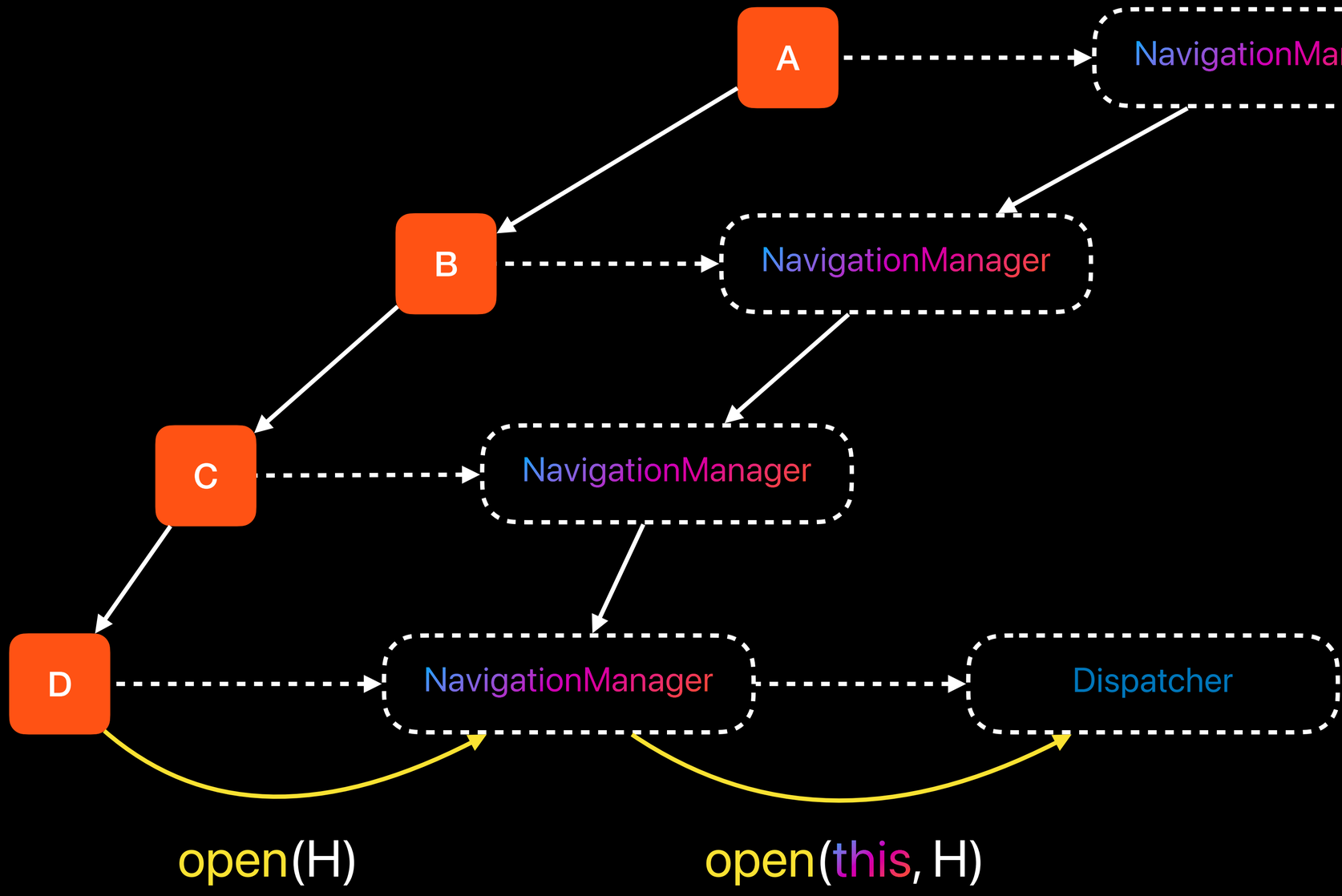
✅ 3. Изменить UI

Все вместе

Управление навигацией



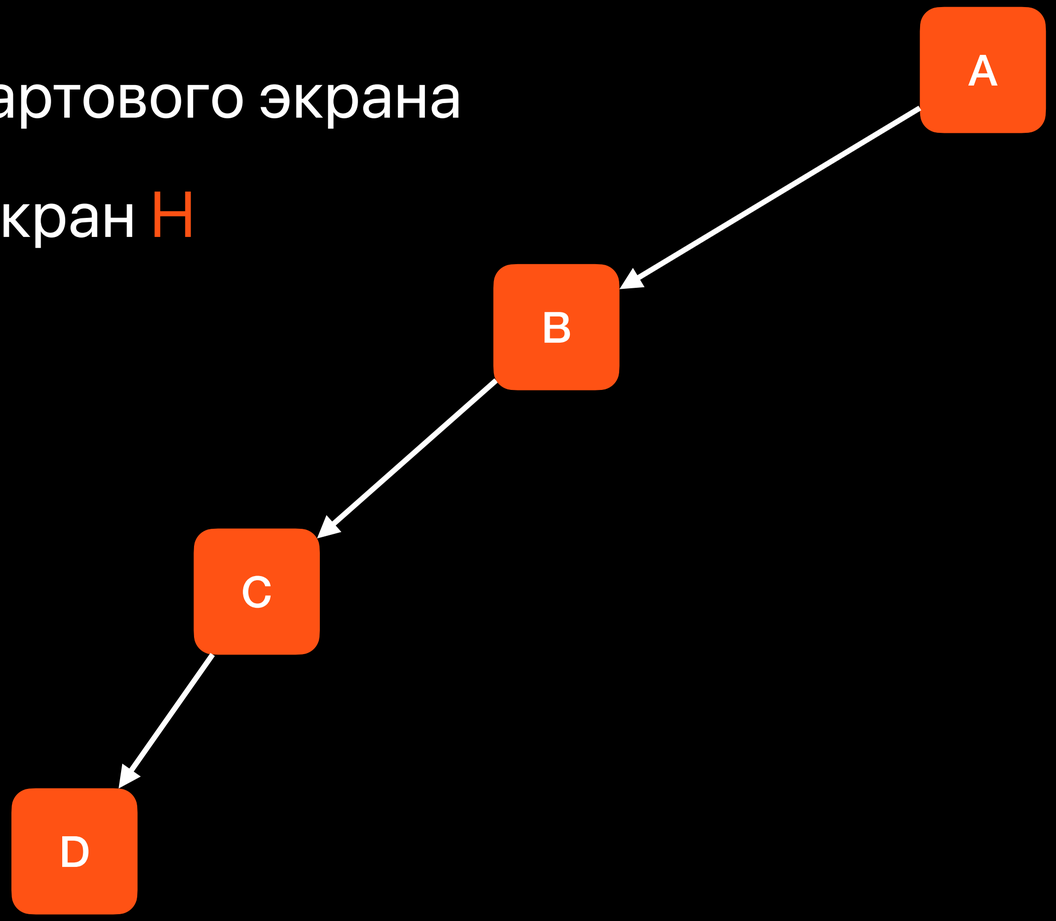
1. Вызов команды



1. Вызов команды

Путь до стартового экрана

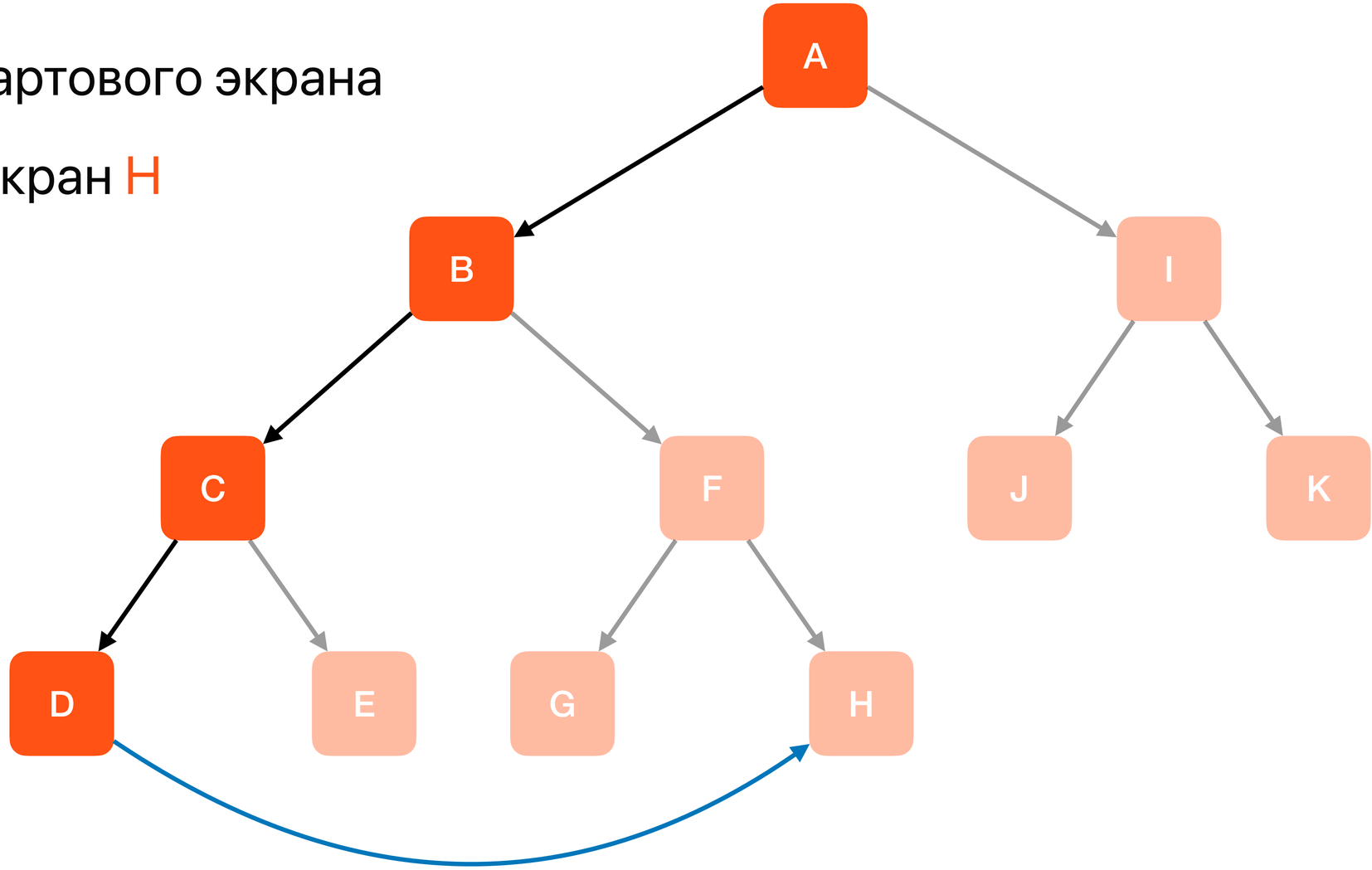
Искомый экран H



2. Поиск

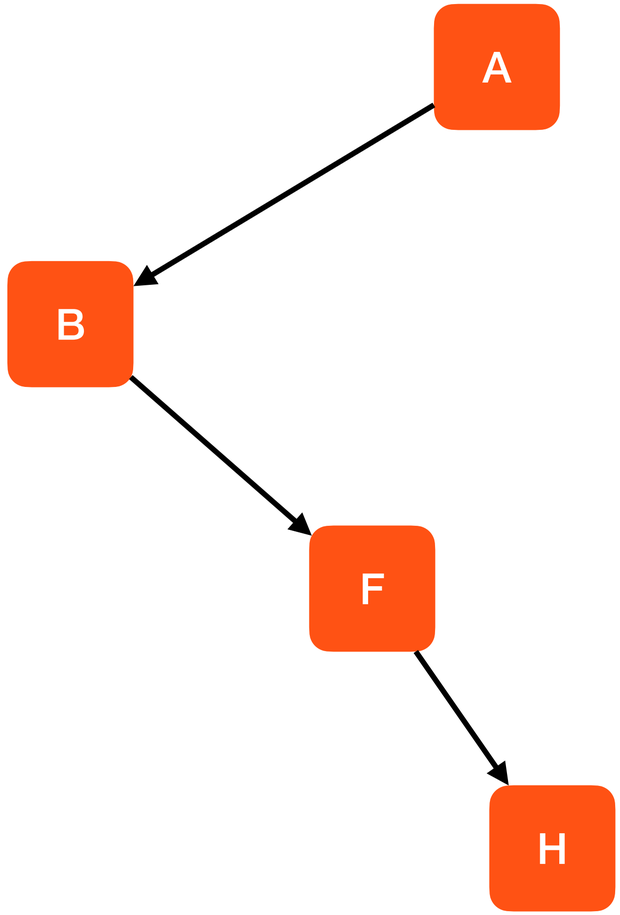
Путь до стартового экрана

Искомый экран H

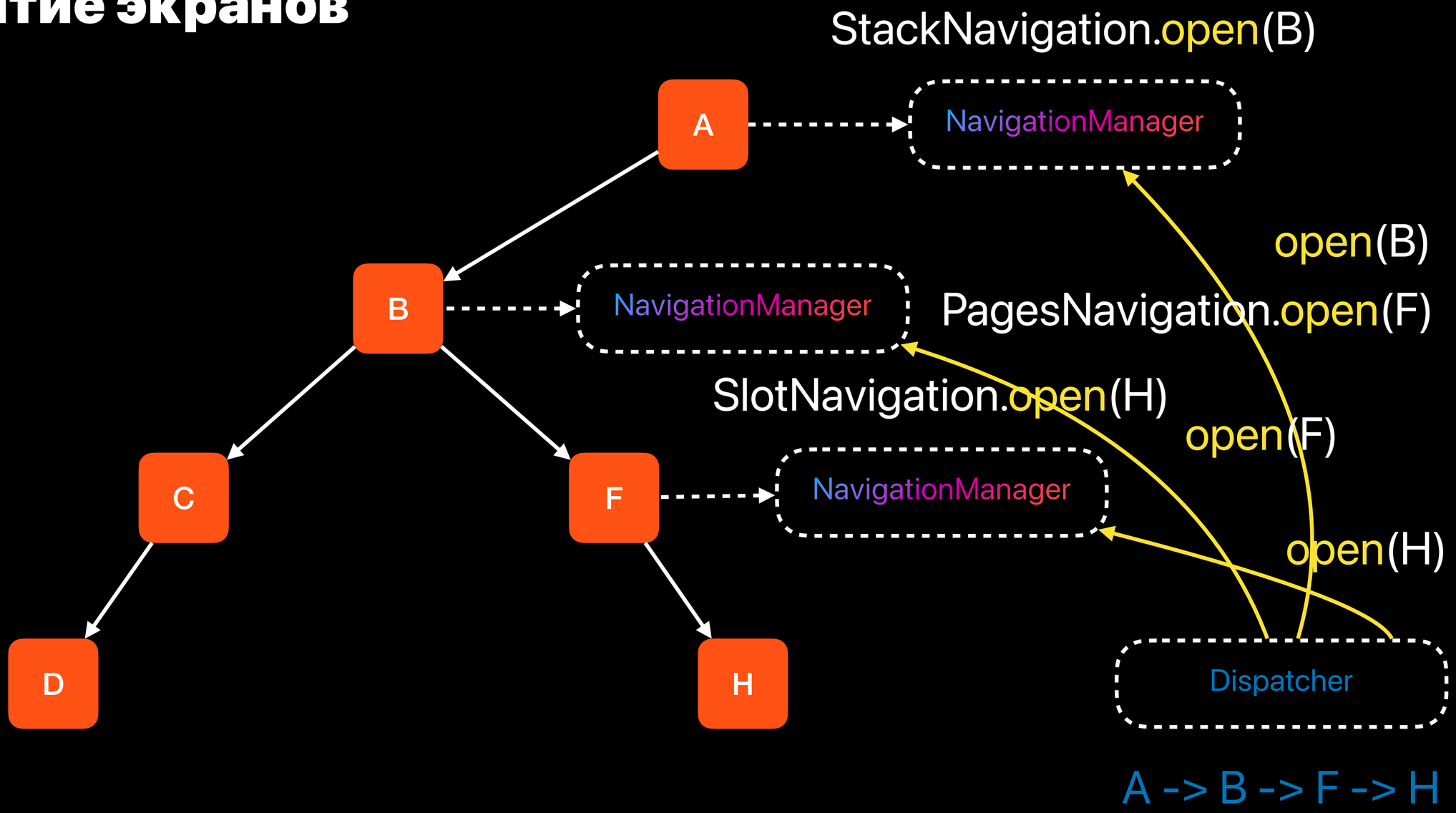


2. Поиск

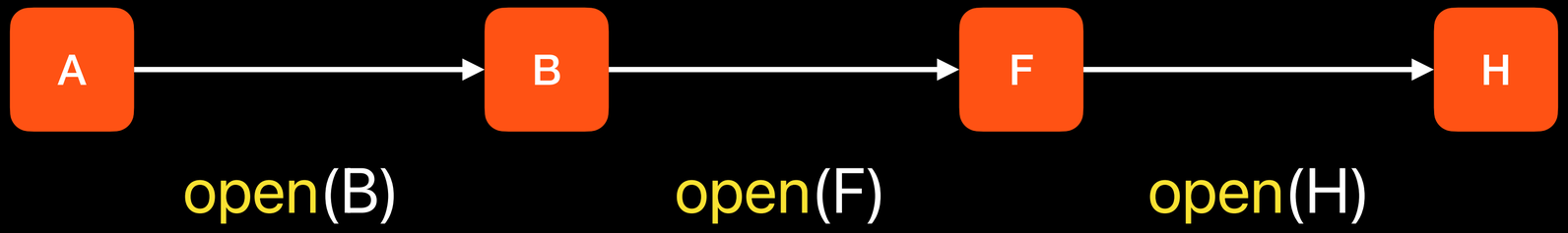
Путь до искомого экрана



3. Открытие экранов



Открытие экранов



Закрытие экранов

Создание экрана

Создание экрана

1. Экран и параметры

2. Фабрика и иерархия

```
@Parcelable
data class SampleScreenParams(...): ScreenParams
```

```
registerFactory(SampleScreenParams::class) { context, params →
    SampleScreenParams(context, params)
}

registerNavigation(RootScreenParams::class) {
    stack(SampleScreenParams::class)
}
```

```
        ...
    }
}
```

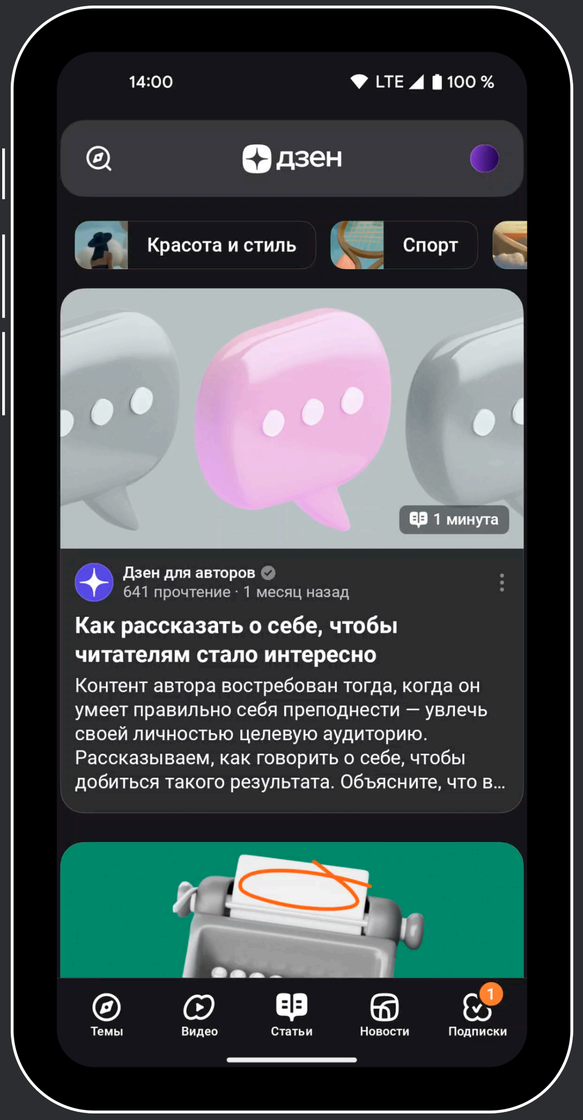
Создание host экрана

Создание host экрана

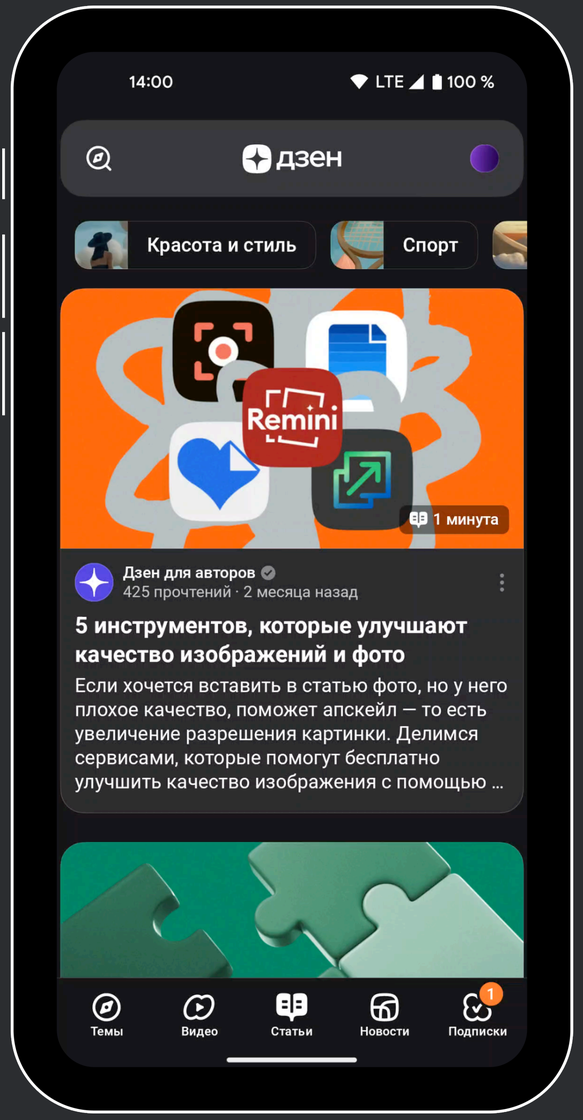
1. Экран и параметры
2. Фабрика и иерархия
3. State и Host
4. Дочерняя иерархия

```
class SampleComponent(
    context: ScreenContext,
    params: SampleScreenParams,
): ScreenComponent<SampleScreenParams>(context, params) {

    val pages = pages()

    // Навигация внутри SampleScreen
    registerNavigation(SampleScreenParams::class) {
        pages(...)
    }
}
```

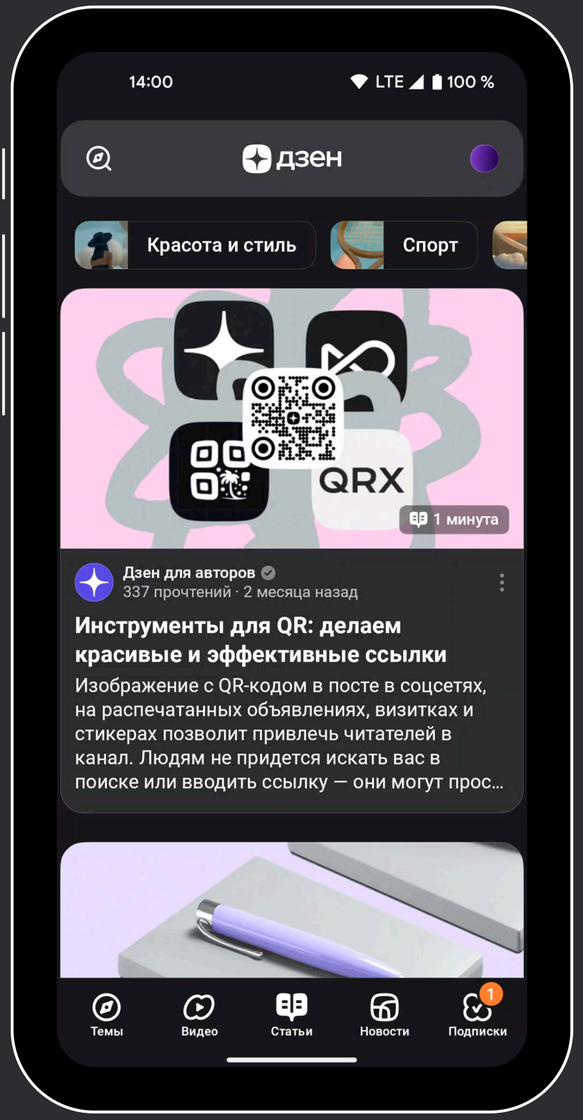
Примеры



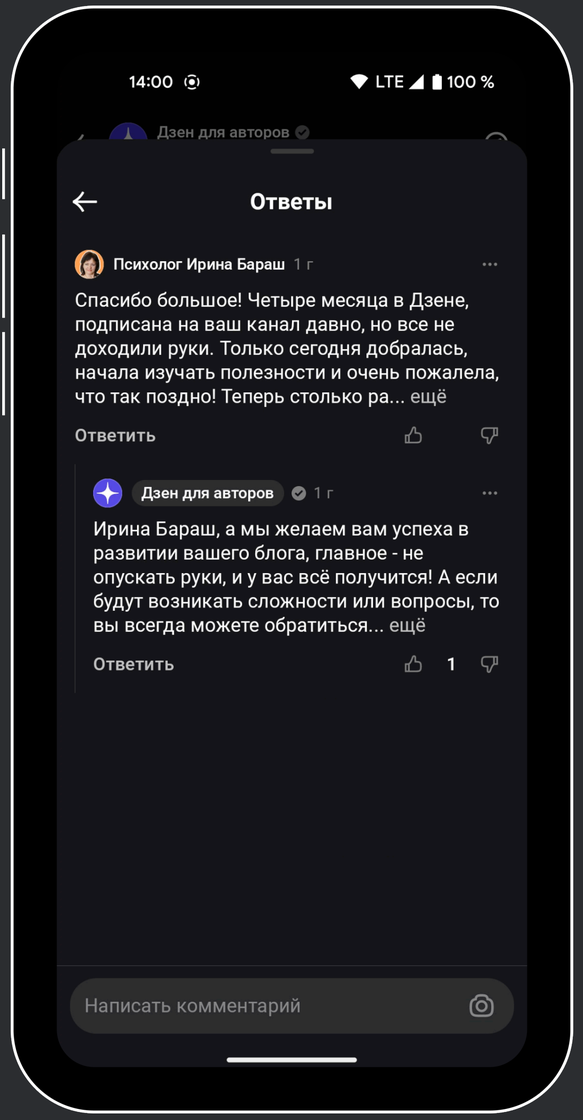

```
open(ChannelScreenParams(id = "1q2w3e4r"))
```

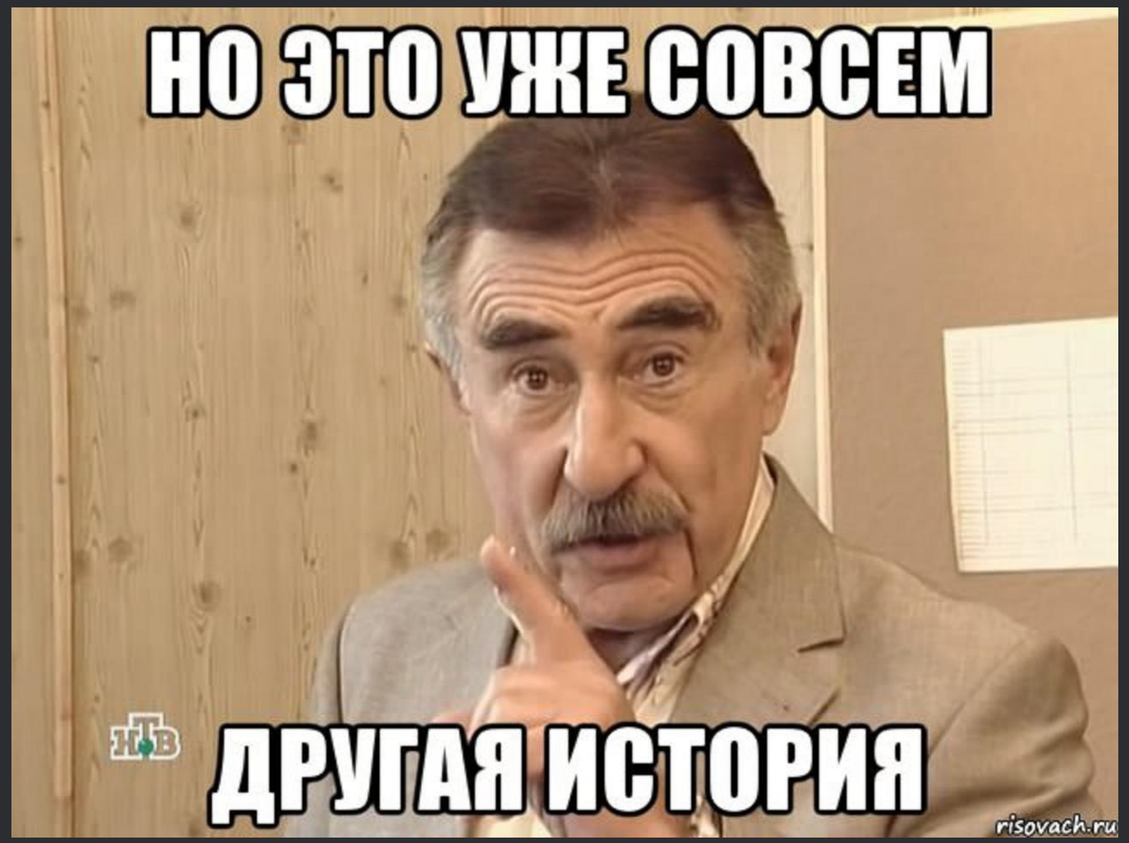



```
inside(SubscriptionsScreenParams)
open(ChannelScreenParams(id = "1q2w3e4r"))
```




```
inside(SubscriptionsScreenParams)
inside(ChannelScreenParams(id = "1q2w3e4r"))
open(ArticlesScreenParams)
```


14:00
LTE
100 %
Дзен для авторов
Ответы
Психолог Ирина Бараш 1 г
Спасибо большое! Четыре месяца в Дзене, подписана на ваш канал давно, но все не доходили руки. Только сегодня добралась, начала изучать полезности и очень пожалела, что так поздно! Теперь столько ра... ещё
Ответить
Дзен для авторов 1 г
Ирина Бараш, а мы желаем вам успеха в развитии вашего блога, главное - не опускать руки, и у вас всё получится! А если будут возникать сложности или вопросы, то вы всегда можете обратиться... ещё
Ответить
1
Написать комментарий


НО ЭТО УЖЕ СОВСЕМ
НТВ
ДРУГАЯ ИСТОРИЯ
risovach.ru

Разработчики остались довольны)

Q&A

@DanilKoles

ScreenFactory

```
Map<KClass<ScreenParams>, (context: ScreenContext, params: P) → ScreenComponent<P>>
```

```
class NavigationManager {

    fun createScreen(
        params: ScreenParams,
        context: ComponentContext,
    ): ScreenComponent {
        val screenContext = ScreenContext(NavigationManager(), context)
        ...
    }

}
```

```
class NavigationManager {

    val screenFactory: ScreenFactory

    fun createScreen(
        params: ScreenParams,
        context: ComponentContext,
    ): ScreenComponent {
        val screenContext = ScreenContext(NavigationManager(), context)
        return screenFactory.create(screenContext, params)
    }

}
```

Dagger без lateinit

```
class FScreenComponent @AssistedInject(
    @Assisted context: ScreenContext,
    @Assisted params: FScreenParams,
    val foo: Foo,
    val bar: Bar,
): ScreenComponent<FScreenParams>(context, params) {

    @AssistedFactory
    interface Factory {
        fun create(
            context: ScreenContext,
            params: FScreenParams,
        ): FScreenComponent
    }
}
```

- Предоставляются извне DI графа
- Зависимости из DI

- Интерфейс фабрики, Её генерирует Dagger

Dagger без lateinit

```
@FScope
@Component
interface FDaggerComponent {
    // Dagger генерирует фабрику
    val factory: FScreenComponent.Factory
}

// Регистрируем фабрику из Dagger
val factory = fDaggerComponent.factory
registerFactory(FScreenParams::class, factory::create)
```

```
class NavigationManager(
    val parent: NavigationManager? = null,
    val params: ScreenParams,
) {

    val children: MutableMap<ScreenParams, NavigationManager>

    fun createScreen(
        context: ComponentContext, // Дочерний context
        params: ScreenParams, // Дочерние params
    ): ScreenComponent {
        val childManager = NavigationManager(this, params)
        children[params] = childManager
        // Удаляем при уничтожении экрана
        context.lifecycle.onDestroy { children.remove(params) }
        ...
    }
}
```

```
class NavigationManager {

    val navigations: MutableMap<String, Navigation>

    fun provideNavigation(
        type: String,
        factory: () → Navigation,
    ): Navigation {
        return navigations.getOrPut(tag) { factory() }
    }
}

fun ScreenContext.stack(): StateFlow<ChildStack> {
    childStack(
        navigationManager.provideNavigation(
            type = "stack"
            factory = { StackNavigation() }
        )
    )
}
```

- Все Navigator экрана
- Lazy создание
- Создаем StackNavigator при вызове stack()

Управление навигацией

```
class Dispatcher {

  val hierarchy: Hierarchy

  // from - с какого экрана вызывается команда
  // path - параметры из команды inside(...).inside(...).open(...)
  fun open(from: NavigationManager, path: Path<ScreenParams>) { ... }

  fun close(from: NavigationManager, path: Path<ScreenParams>) { ... }

}
```