авиасейлс

Как мы в Авиасейлс сделали свою навигацию на Compose

Олег Дволятик,
разработчик Android-платформы

Android в Авиасейлс

- Наши продуктовые команды кросс-функциональные
- Преобладает фичевая разработка
- Так исторически сложилось, что каждая команда создает инструменты себе сама
- Много легаси
- Платформенная команда создает инструменты и вырабатывает общие подходы

Как платформа мы поняли, что стоит уходить от поддержки текущего кода и исправления багов здесь и сейчас в сторону крупных рефакторингов и создания некоторых инструментов с 0

Задача:
решить проблему навигации

Контекст

- Легаси навигация со множеством проблем
- Постепенный переход на Compose
- Экраны в разном состоянии: какие-то уже на Compose, какие-то на MVI, какие-то совсем легаси
- Проблемы с UX: мы не можем открыть один BottomSheet над другим

Нам нужно за раз:

Решить проблему старой навигации

Использовать новые подходы, которые нам даёт Compose

Добавить фичи для дизайнеров

Выбор решения

Решения с рынка

Jetpack Navigation

Voyager

Decompose

Modo

Jetpack Navigation

Плюсы

- Фреймворк от гугла
- Объявление графа

Минусы

- Нетипизированные экраны
- Боль при работе с аргументами
- Мало нужных фичей

Decompose

Плюсы

- Иеархическая структура
- Много возможностей

Минусы

- Требуется изменить собственную архитектуру приложения и экранов

Modo/Voyager

Плюсы

- Готовые реализации для BottomSheet
- Поддержка интеграции с DI из коробки
- Поддержка ViewModel из коробки

Минусы

- Не общепринятое решение

Есть три стула

Почему не готовое решение

- Нет общепринятой библиотеки для всего комьюнити
- Сильная завязка на стороннюю библиотеку. Для этого 100% использовался бы свой api в качестве прокся
- Необходима интеграция под наш проект, поэтому также писался бы дополнительно код
- Размер сторонних библиотек не сильно большой, можно разобраться и форкнуть, что позволить переписать внутреннюю реализацию библиотеки

Почему не форк

- В момент форка мы так или иначе добавляли бы что-то свое
- На рынке нет решения, проверенного годами
- Хотим реализовать свои идеи

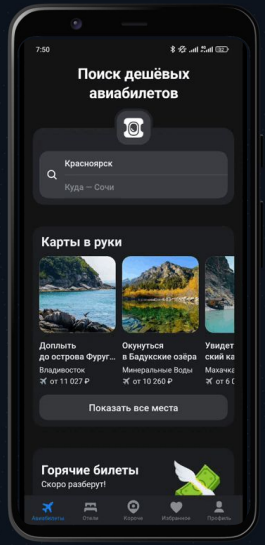
Концепция нашего решения

SingleState-App

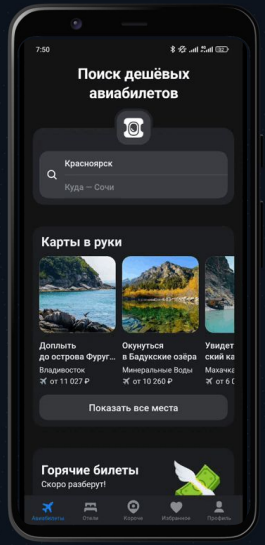
Наш идеал — SingleState для всего приложения

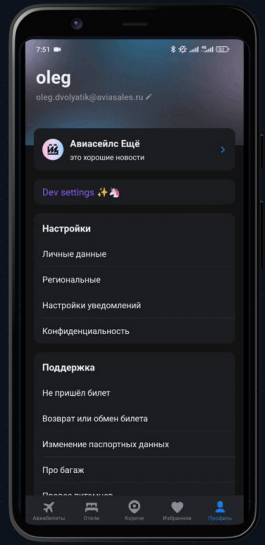
Наша реальность — SingleState для навигации

Что для нас не SingleState-навигация

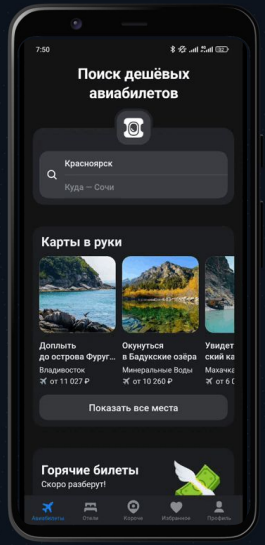


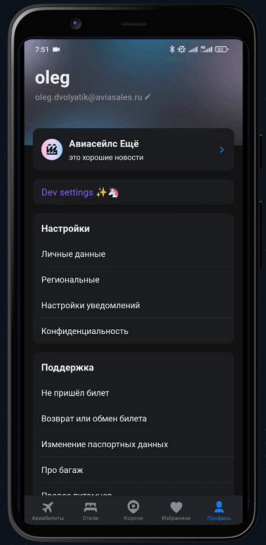
Что для нас не SingleState-навигация

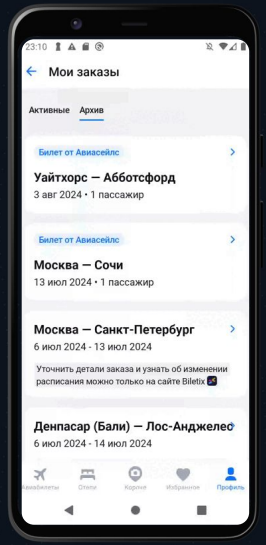




Что для нас не SingleState-навигация





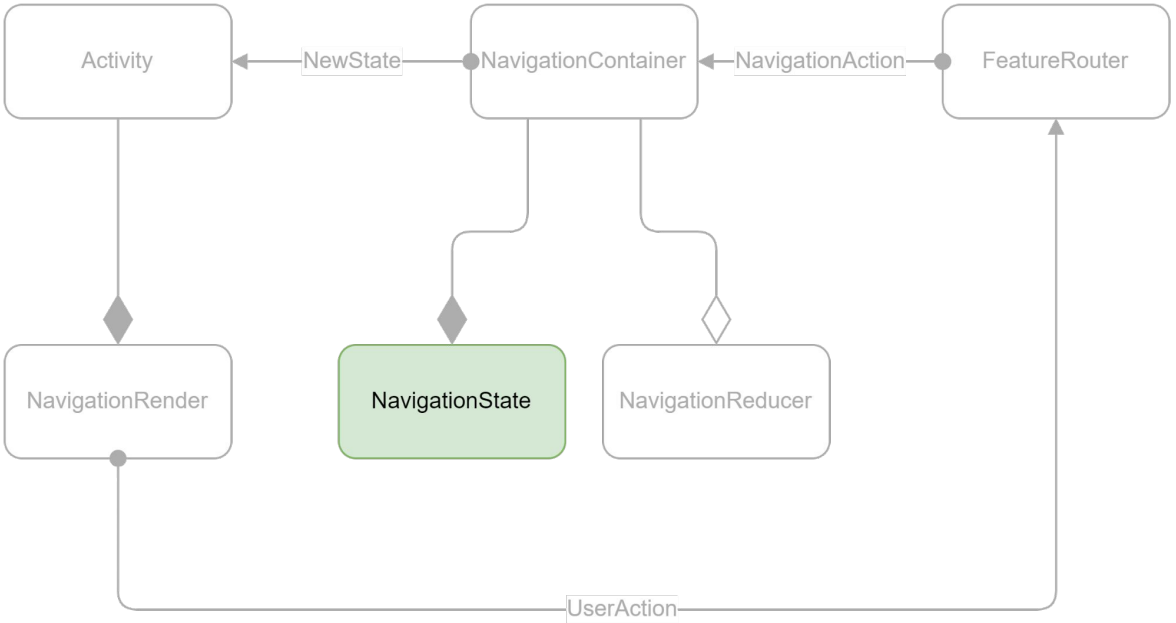


SingleState-навигация

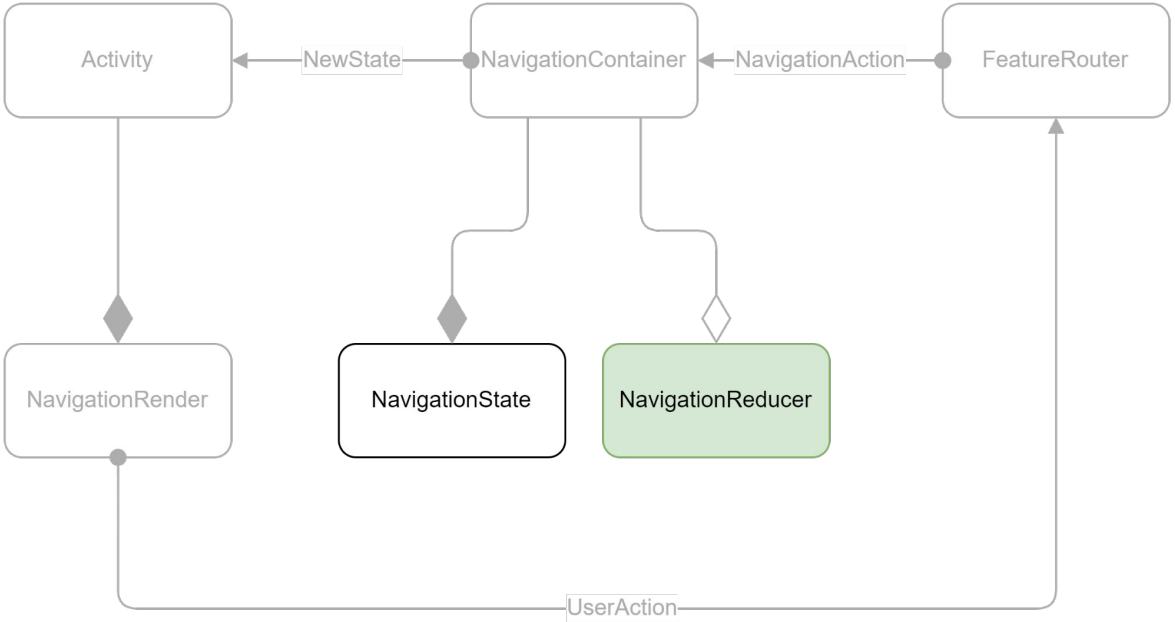
- Трекинг аналитики
- Сохранение состояния
- Обработка диплинков
- Упрощение UI-тестов за счет добавление диплинков
- Работа с одним стейтом из разных мест (так как много команд)

Архитектура

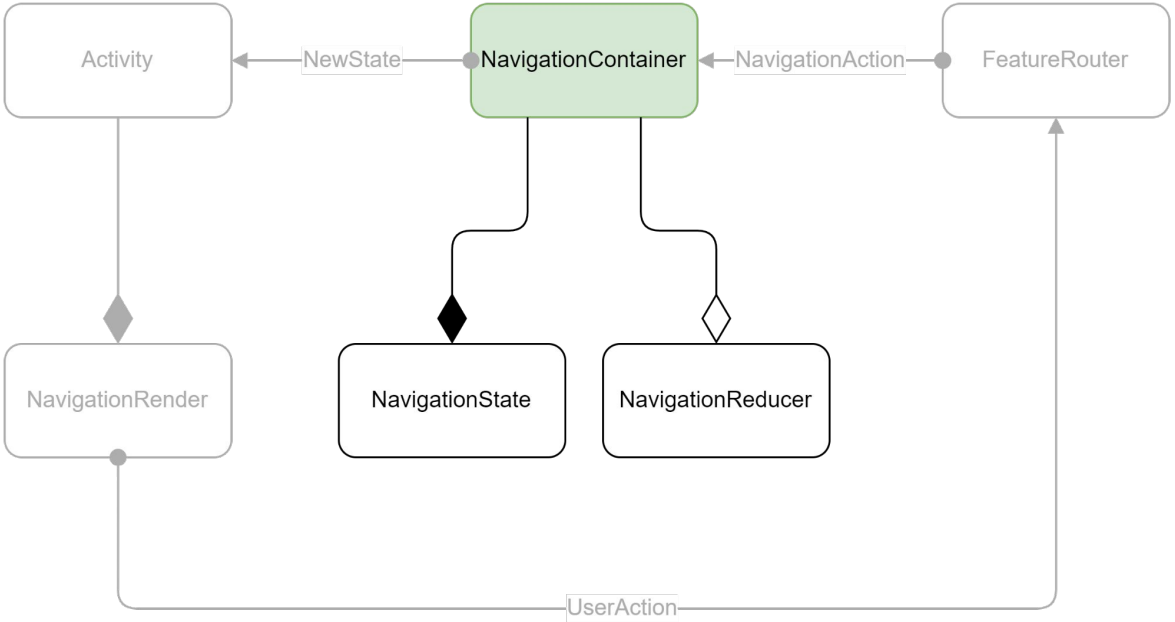
UDF-подход


Activity
NewState
NavigationContainer
NavigationAction
FeatureRouter
NavigationRender
NavigationState
NavigationReducer
UserAction

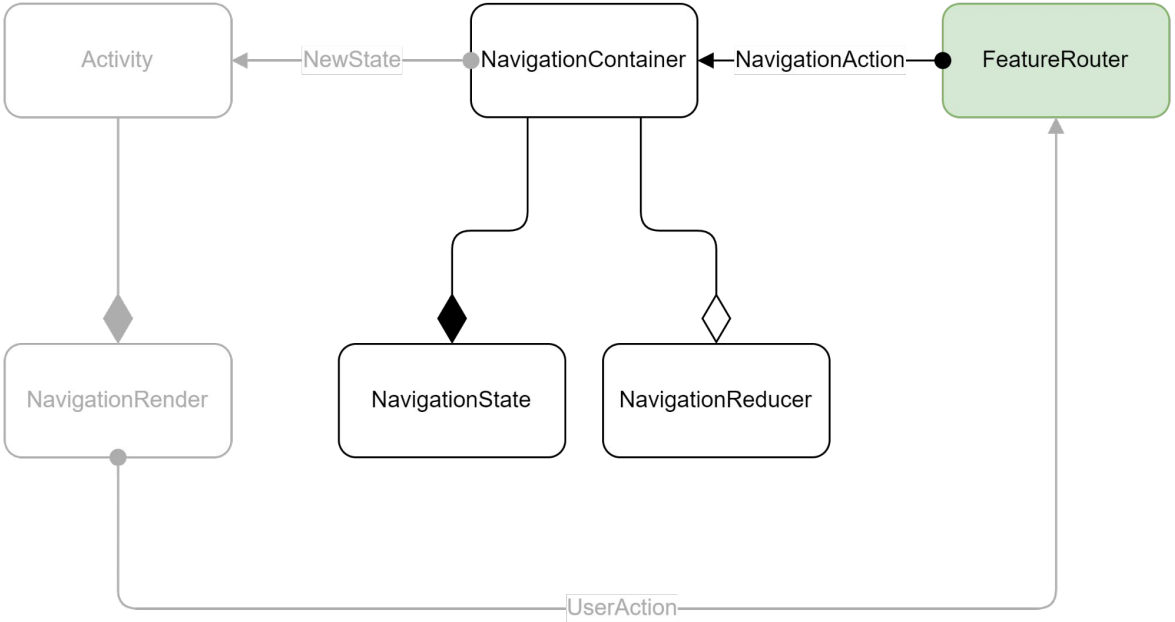
UDF-подход


Activity
NewState
NavigationContainer
NavigationAction
FeatureRouter
NavigationRender
NavigationState
NavigationReducer
UserAction

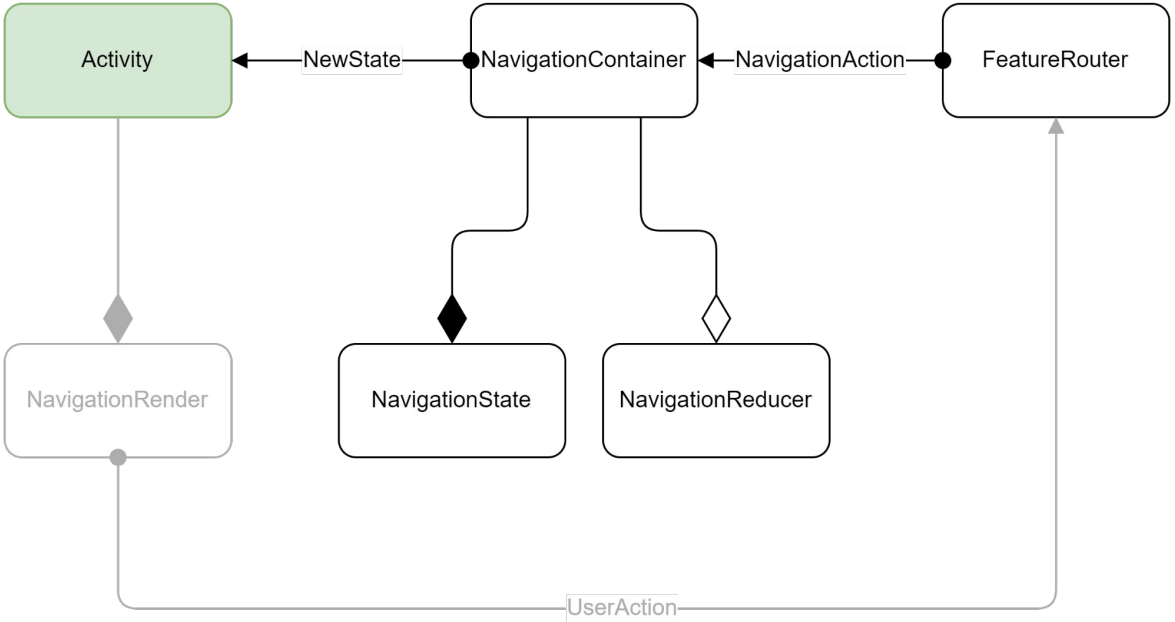
UDF-подход


Activity
NewState
NavigationContainer
NavigationAction
FeatureRouter
NavigationRender
NavigationState
NavigationReducer
UserAction

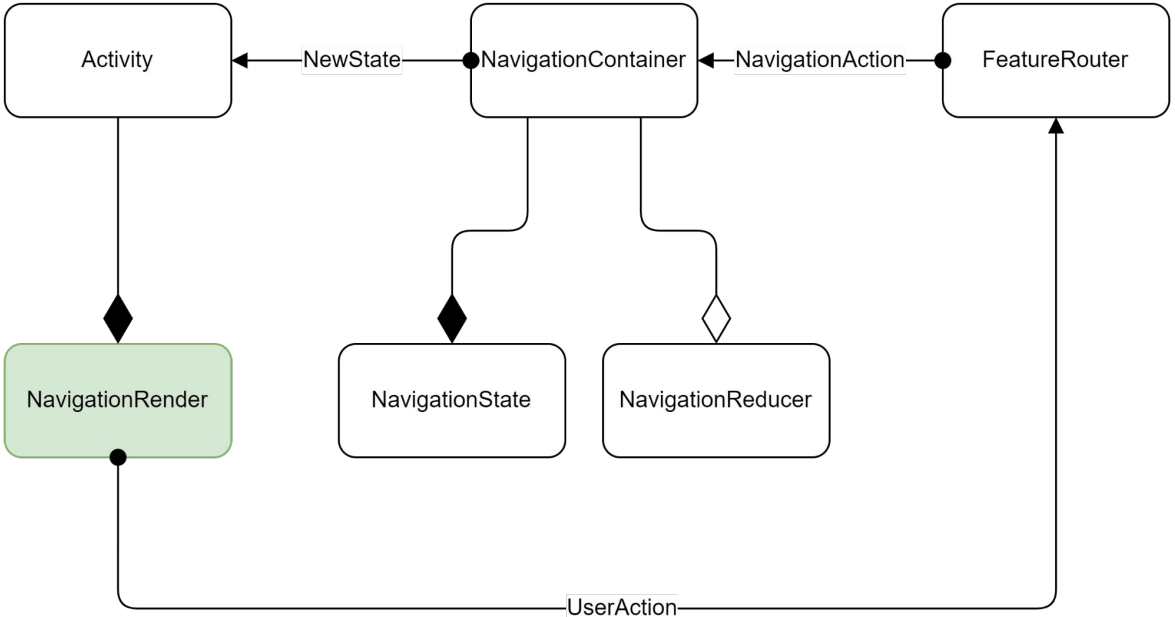
UDF-подход


Activity
NewState
NavigationContainer
NavigationAction
FeatureRouter
NavigationRender
NavigationState
NavigationReducer
UserAction

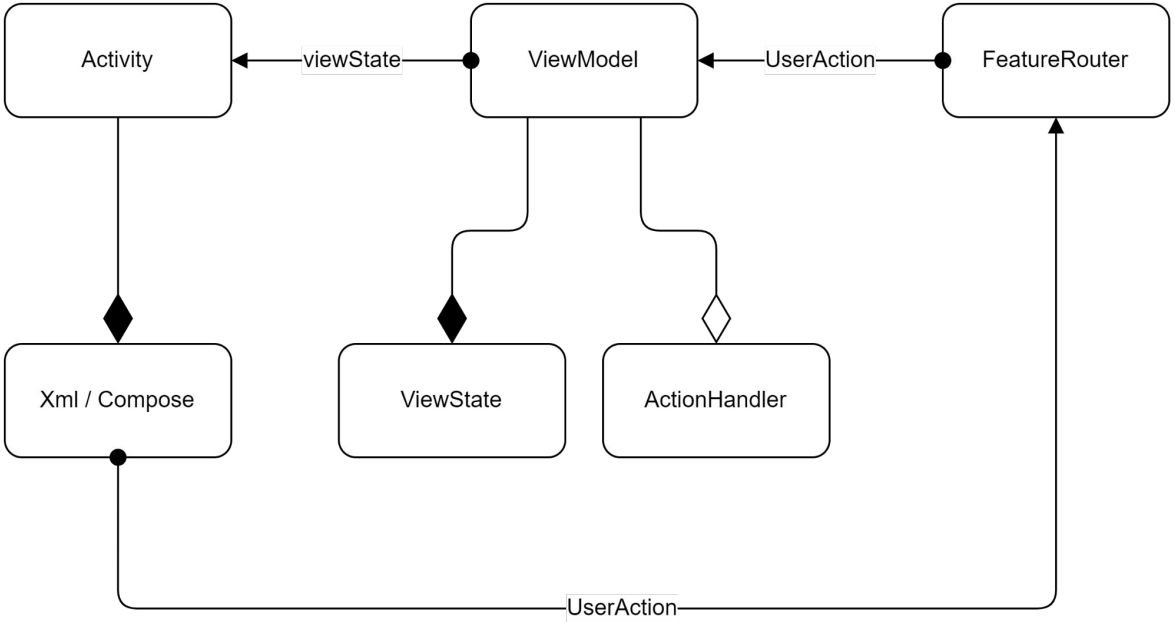
UDF-подход


Activity
NewState
NavigationContainer
NavigationAction
FeatureRouter
NavigationRender
NavigationState
NavigationReducer
UserAction

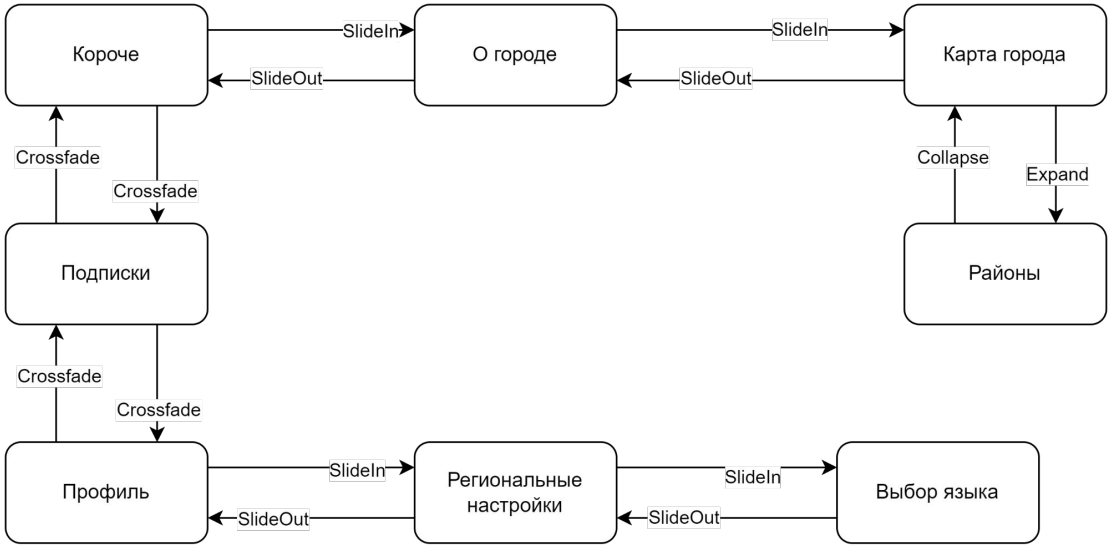
UDF-подход


Activity
NewState
NavigationContainer
NavigationAction
FeatureRouter
NavigationRender
NavigationState
NavigationReducer
UserAction

UDF-подход


Activity
viewState
ViewModel
UserAction
FeatureRouter
Xml / Compose
ViewState
ActionHandler
UserAction

Состояние


Короче
SlideIn
SlideOut
О городе
SlideIn
SlideOut
Карта города
Crossfade
Crossfade
Collapse
Expand
Подписки
Районы
Crossfade
Crossfade
Профиль
SlideIn
SlideOut
Региональные настройки
SlideIn
SlideOut
Выбор языка

Состояние

```
<navigation xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/trap_nav_graph">

    <fragment android:id="@+id/trapMainScreenFragment"
        android:name="aviasales.context.trap.product.ui.main.TrapMainFragment"
        android:label="TrapMainScreenFragment">
        <action android:id="@+id/action_trapMainScreenFragment_to_trapPoiDetailsFragment"
            app:destination="@id/trapPlaceDetailsFragment"/>
        <action android:id="@+id/action_trapMainScreenFragment_to_galleryFragment"
            app:destination="@id/galleryFragment"/>
    </fragment>

    <fragment android:id="@+id/trapPlaceDetailsFragment"
        android:name="aviasales.context.trap.feature.poi.details.ui.TrapPlaceDetailsFragment"
        android:label="TrapPlaceDetailsFragment"
        tools:layout="@layout/fragment_trap_poi_details">
    </fragment>

    <fragment android:id="@+id/galleryFragment"
        android:name="aviasales.shared.gallery.ui.GalleryFragment"
        android:label="GalleryFragment"/>

</navigation>
```

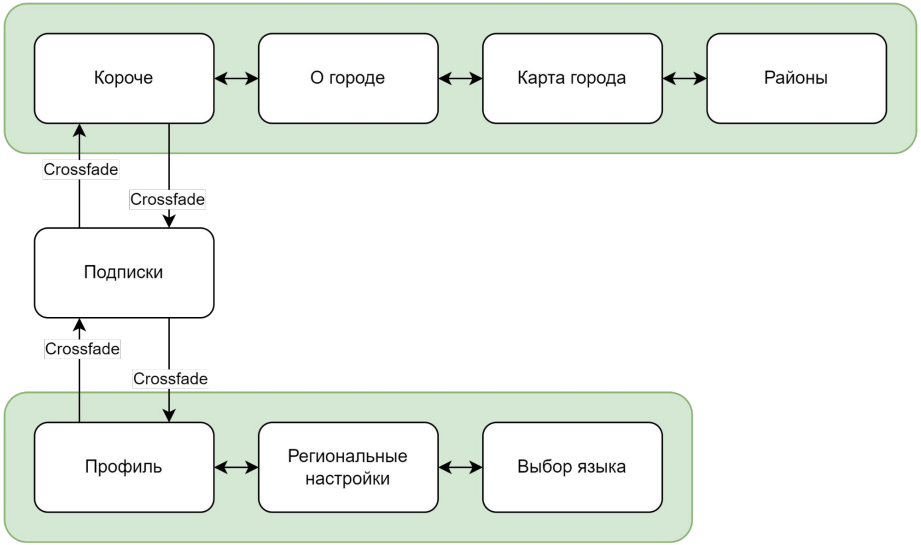
Состояние

```
<navigation xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/trap_nav_graph">

  <fragment android:id="@+id/trapMainScreenFragment"
      android:name="aviasales.context.trap.product.ui.main.TrapMainFragment"
      android:label="TrapMainScreenFragment">
    <action android:id="@+id/action_trapMainScreenFragment_to_trapPoiDetailsFragment"
        app:destination="@id/trapPlaceDetailsFragment"/>
    <action android:id="@+id/action_trapMainScreenFragment_to_galleryFragment"
        app:destination="@id/galleryFragment"/>
  </fragment>

  <fragment android:id="@+id/trapPlaceDetailsFragment"
      android:name="aviasales.context.trap.feature.poi.details.ui.TrapPlaceDetailsFragment"
      android:label="TrapPlaceDetailsFragment"
      tools:layout="@layout/fragment_trap_poi_details">
  </fragment>

  <fragment android:id="@+id/galleryFragment"
      android:name="aviasales.shared.gallery.ui.GalleryFragment"
      android:label="GalleryFragment"/>

</navigation>
```

Почему не выбрали сырой граф

- Kotlin не предоставляет реализации графа
- Сторонние библиотеки нагружены функционалом и нет оптимизаций под наш конкретно случай
- Свое решение дорого в реализации и дорого в поддержке

Структура хранения данных



Структура хранения данных

Структура ElementNavigation

NavigationElement Kotlin

```kotlin
sealed interface NavigationElement {

  val key: Key

  @JvmInline
  value class Key(val origin: String)
}
```

Структура ElementNavigation

StackElement Kotlin

```kotlin
data class StackElement(
  override val key: NavigationElement.Key,
  val elements: ElementList
) : NavigationElement

val StackElement.topElement: NavigationElement
  get() = elements.last()

val StackElement.topFullScreenElement: NavigationElement
  get() = elements.last { it !is DialogElement && it !is BottomSheetElement }
```

Структура ElementNavigation

DialogElement — Kotlin

```kotlin
data class DialogElement(
  override val key: NavigationElement.Key,
  val element: NavigationElement
) : NavigationElement
```

Структура ElementNavigation

NavigationElement Kotlin

```kotlin
// Kotlin-модуль
interface ScreenElement : NavigationElement

// Android-модуль
interface ComposeScreenElement : ScreenElement {

  @Composable
  fun Content()
}
```

Reducer

- Чистая функция
- Получается на вход текущее состояние и экшн
- На выходе новое состояние

Reducer

NavigationStateReducer — Kotlin

```kotlin
object NavigationStateReducer {

 operator fun invoke(element: Element, action: Action): NavigationElement {
    val targetElement = if (action is DirectNavigationAction) {
      element.find(action.key) ?: return element
    } else {
      element.findLast()
    }

    while (!handled) {
      val parent = elementHashMap[selector.key] ?: break

      val result = parent.reduce(action)
      selector = result.element
      handled = result is Handled
    }

    while (true) {
      val parent = elementHashMap[selector.key] ?: break
      val result = parent.updateChild(selector)
      selector = result
    }

    return selector
  }
```

Reducer

NavigationStateReducer — Kotlin

```kotlin
object NavigationStateReducer {

 operator fun invoke(element: Element, action: Action): NavigationElement {
    val targetElement = if (action is DirectNavigationAction) {
      element.find(action.key) ?: return element
    } else {
      element.findLast()
    }

    while (!handled) {
      val parent = elementHashMap[selector.key] ?: break

      val result = parent.reduce(action)
      selector = result.element
      handled = result is Handled
    }

    while (true) {
      val parent = elementHashMap[selector.key] ?: break
      val result = parent.updateChild(selector)
      selector = result
    }

    return selector
  }
```

Поиск целевого элемента

Reducer

NavigationStateReducer — Kotlin

```kotlin
object NavigationStateReducer {

 operator fun invoke(element: Element, action: Action): NavigationElement {
    val targetElement = if (action is DirectNavigationAction) {
      element.find(action.key) ?: return element
    } else {
      element.findLast()
    }

    while (!handled) {
      val parent = elementHashMap[selector.key] ?: break

      val result = parent.reduce(action)
      selector = result.element
      handled = result is Handled
    }

    while (true) {
      val parent = elementHashMap[selector.key] ?: break
      val result = parent.updateChild(selector)
      selector = result
    }

    return selector
  }
```

Расчет нового состояния

Reducer

NavigationStateReducer — Kotlin

```kotlin
object NavigationStateReducer {

 operator fun invoke(element: Element, action: Action): NavigationElement {
    val targetElement = if (action is DirectNavigationAction) {
      element.find(action.key) ?: return element
    } else {
      element.findLast()
    }

    while (!handled) {
      val parent = elementHashMap[selector.key] ?: break

      val result = parent.reduce(action)
      selector = result.element
      handled = result is Handled
    }

    while (true) {
      val parent = elementHashMap[selector.key] ?: break
      val result = parent.updateChild(selector)
      selector = result
    }

    return selector
  }
```

Пересборка дерева

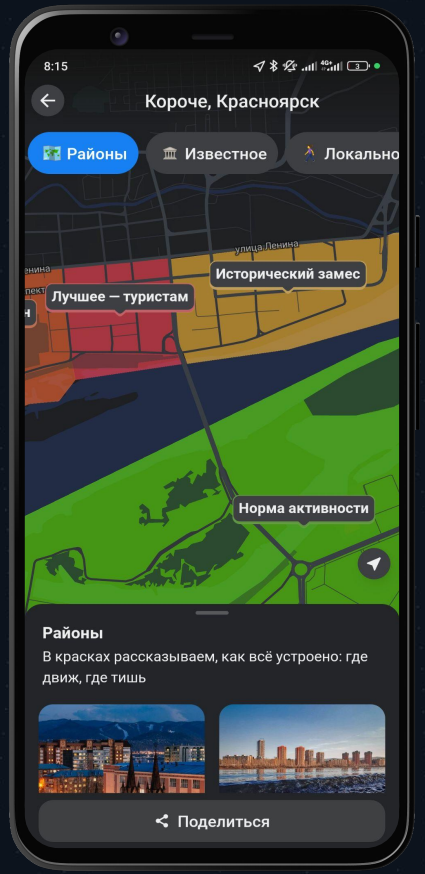
Reducer

ElementStackReducer — Kotlin

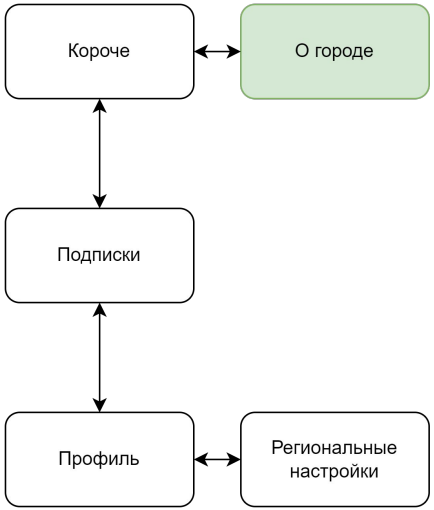
```kotlin
object ElementStackReducer : ElementReducer<StackElement, StackNavigationAction> {

  override fun invoke(element: StackElement, action: StackNavigationAction): ReducerResult =
    when (action) {
      is Open -> element.update { push(action.elements) }
      is Replace -> element.update { replaceLast(action.elements) }
      is Back -> element.updateIf(element.elements.size > 1) { pop() }
    }
}
```

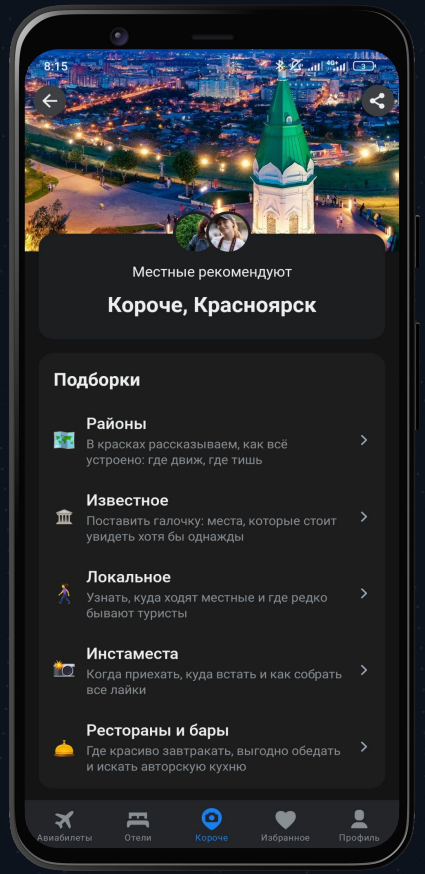
Reducer


Короче
О городе
Карта города
Подписки
Профиль
Региональные настройки

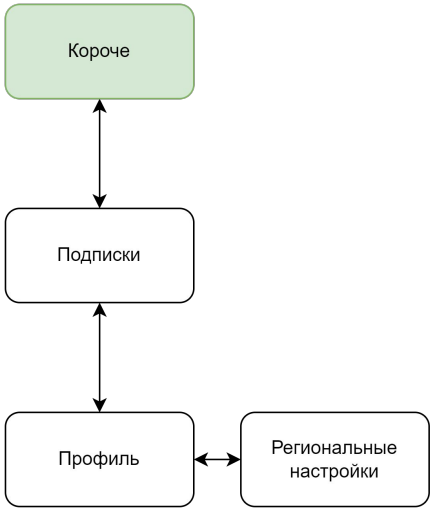

8:15
Короче, Красноярск
Районы
Известное
Локально
Исторический замес
Лучшее — туристам
Норма активности
Районы
В красках рассказываем, как всё устроено: где движ, где тишь
Поделиться

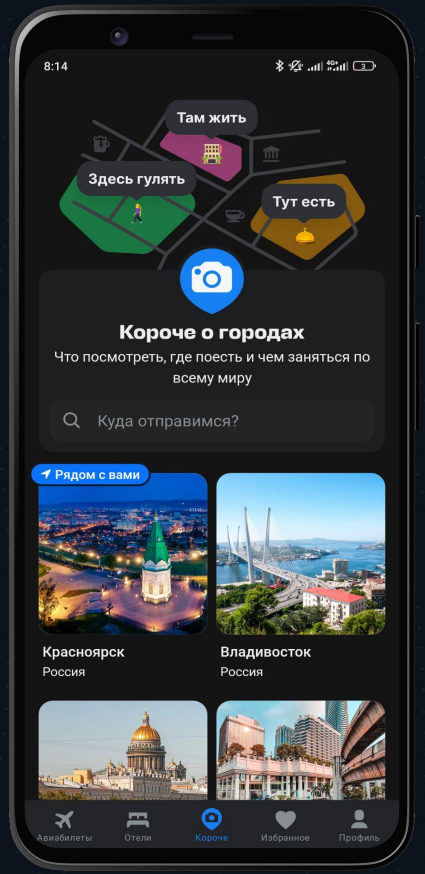
Reducer


Короче
О городе
Подписки
Профиль
Региональные настройки

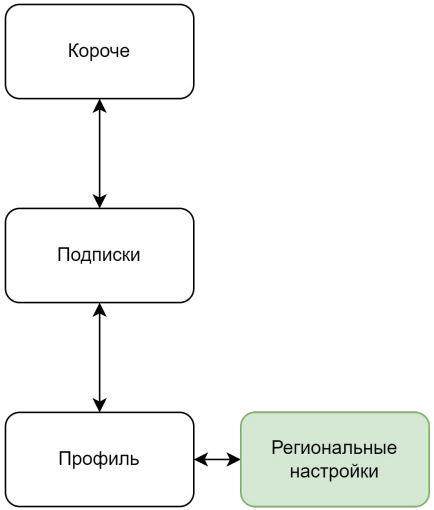

8:15
Местные рекомендуют
Короче, Красноярск
Подборки
Районы
В красках рассказываем, как всё устроено: где движ, где тишь
Известное
Поставить галочку: места, которые стоит увидеть хотя бы однажды
Локальное
Узнать, куда ходят местные и где редко бывают туристы
Инстаместа
Когда приехать, куда встать и как собрать все лайки
Рестораны и бары
Где красиво завтракать, выгодно обедать и искать авторскую кухню
Авиабилеты
Отели
Короче
Избранное
Профиль

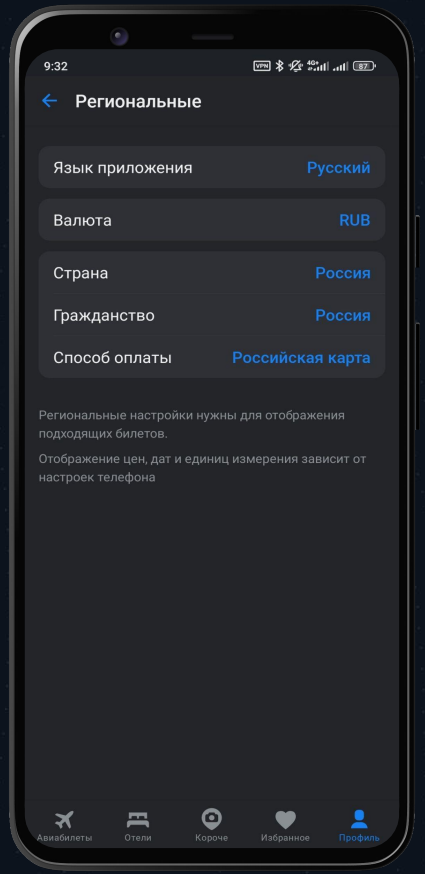
Reducer


Короче
Подписки
Профиль
Региональные настройки


8:14
Там жить
Здесь гулять
Тут есть
Короче о городах
Что посмотреть, где поесть и чем заняться по всему миру
Куда отправимся?
Рядом с вами
Красноярск
Россия
Владивосток
Россия
Авиабилеты
Отели
Короче
Избранное
Профиль

Reducer


Короче
Подписки
Профиль
Региональные настройки


9:32
Региональные
Язык приложения
Русский
Валюта
RUB
Страна
Россия
Гражданство
Россия
Способ оплаты
Российская карта
Региональные настройки нужны для отображения подходящих билетов.
Отображение цен, дат и единиц измерения зависит от настроек телефона
Авиабилеты
Отели
Короче
Избранное
Профиль

Render

ElementState

```kotlin
@Composable
fun ElementState(current: NavigationElement?, target: NavigationElement) {
  when (target) {
    is StackElement -> StackElementState(current as? StackElement, target)
    is DialogElement -> DialogElementState(current as? DialogElement, target)
    is BottomSheetElement -> BottomSheetElementState(current as? BottomSheetElement, target)
    is OverlayElement -> OverlayElementState(current as? OverlayElement, target)
    is ComposeScreenElement -> ComposeScreenElementState(target = target)

    is ScreenElement -> error("Not have render for ${target::class}")
  }
}
```

Render

StackElementState — Kotlin

```kotlin
@Composable
fun StackElementState(current: StackElement?, target: StackElement) {
  StackAnimation(
    current = current,
    target = target,
    content = { targetAnimationState -> ElementState(current?.topElement,
targetAnimationState) }
  )

  val top = target.topElement
  if (top is DialogElement || top is BottomSheetElement || top is OverlayElement) {
    ElementState(current = current?.topElement, target = top)
  }
}
```

Render

ComposableScreenElementState — Kotlin

```kotlin
@Composable
fun ComposeScreenElementState(target: ComposeScreenElement) {
  CompositionLocalProvider(LocalCurrentElement provides target) {
    target.SaveableContent()
  }
}
```

Решение других задач

Сохранение состояния экрана

SaveableStateHolder:

- Скролл
- Чекбоксы/свитчи
- Прочие состояния UI-элементов

Сохранение состояния экрана

ScreenModel:

- ViewState
- Репозитории

Сохранение состояния экрана

Переиспользуем ViewModel?

Менеджер состояния 

Сохранение состояния экрана

Переиспользуем ViewModel?

Менеджер состояния 

Хранилище 

Сохранение состояния экрана

Переиспользуем ViewModel?

Менеджер состояния

Хранилище

Интерфейс ViewModel

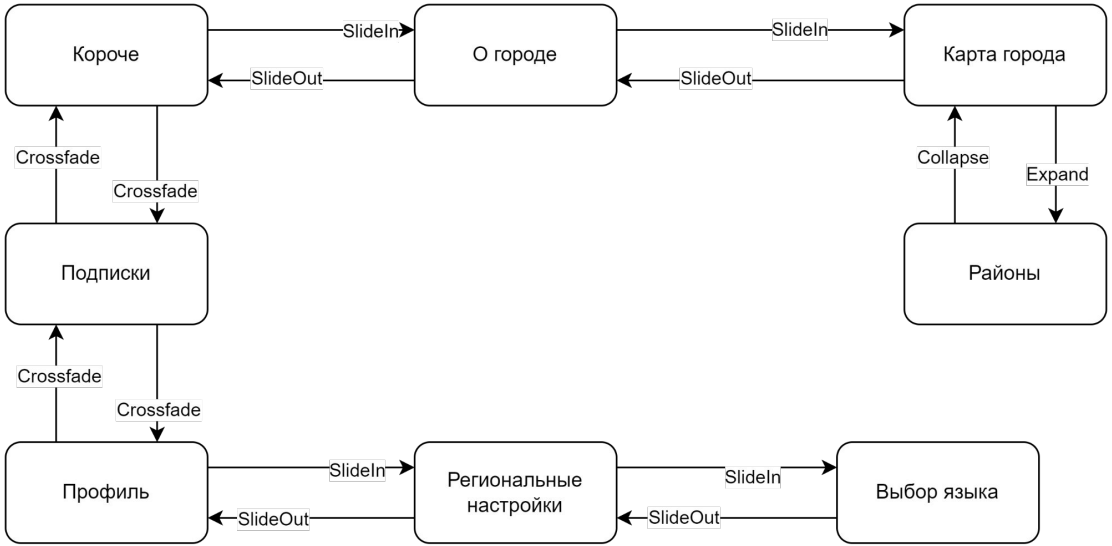
Системная кнопка назад

→ BackHandler

ElementBackHandler — Kotlin

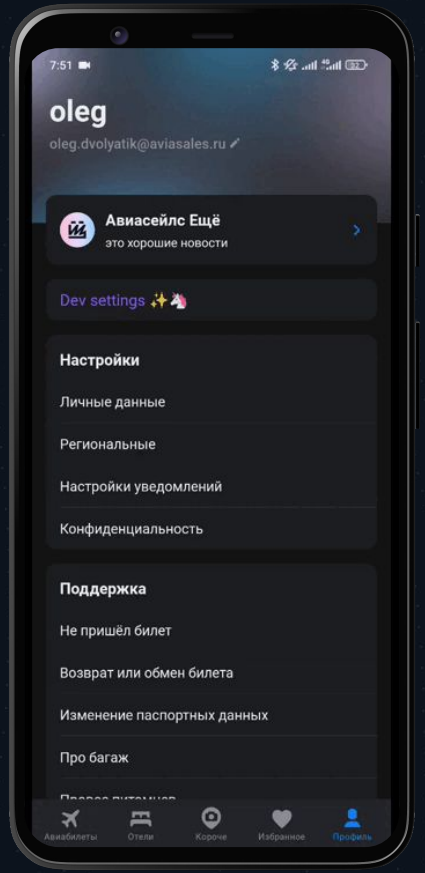
```kotlin
@Composable
fun ElementBackHandler(state: ComposeNavigationState) {
  val actor = LocalActionEmitterProvider.current

  BackHandler(
    enabled = state.target.canGoBack(),
    onBack = { actor.back() })
}
```

Анимации


Короче
SlideIn
SlideOut
О городе
SlideIn
SlideOut
Карта города
Crossfade
Crossfade
Collapse
Expand
Подписки
Районы
Crossfade
Crossfade
Профиль
SlideIn
SlideOut
Региональные настройки
SlideIn
SlideOut
Выбор языка

Анимации

- Тип контейнера
- Тип перехода



Анимации

Animation

Kotlin

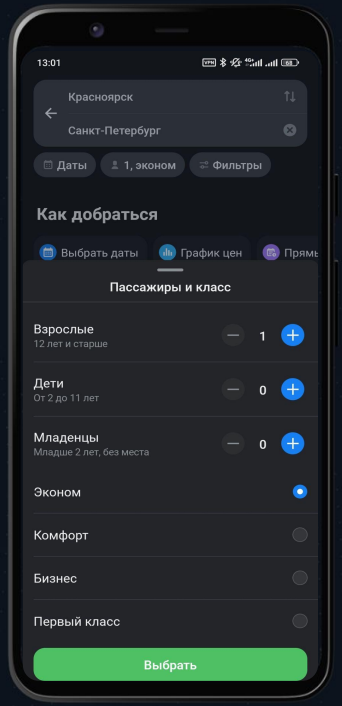
```kotlin
data class StackElementAnimations(
  val idle: ElementTransitionSpec,
  val enter: ElementTransitionSpec,
  val exit: ElementTransitionSpec,
  val replace: ElementTransitionSpec,
)

 CompositionLocalProvider(
    LocalStackElementAnimations provides slideAnimations,
  ) { <...> }
```

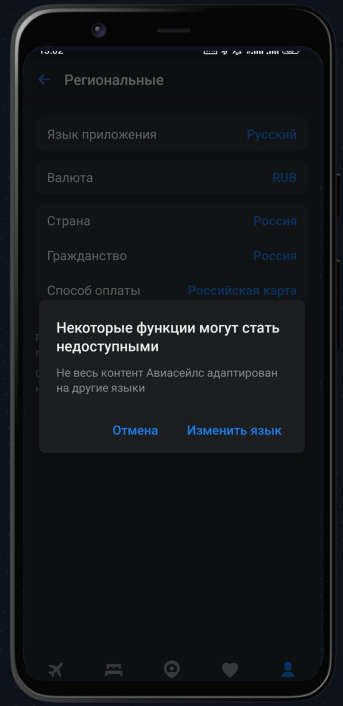
Анимации

Animation Kotlin

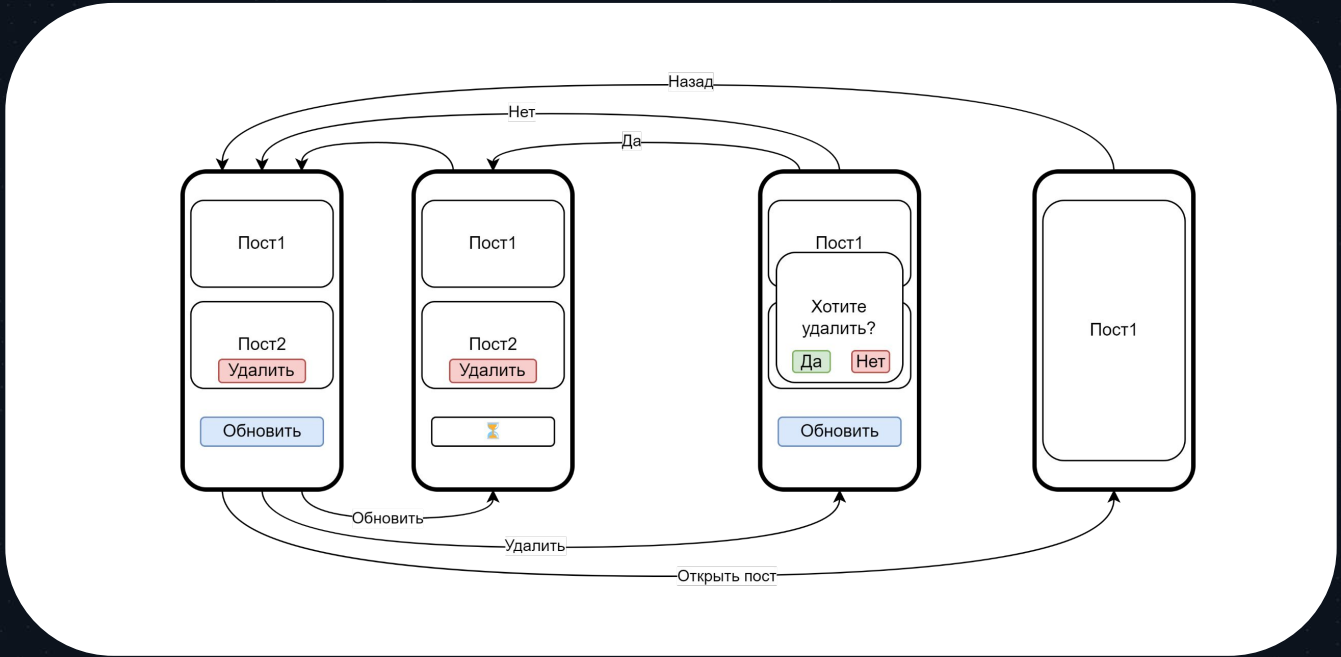
```kotlin
@Composable
private fun calculateTransitionSpec(from: StackElement?, to: StackElement):
ElementTransitionSpec =
  when {
    from == null -> LocalStackElementAnimations.current.idle
    from.topElement.key == to.topElement.key -> LocalStackElementAnimations.current.idle
    from.elements.size == to.elements.size -> LocalStackElementAnimations.current.replace
    from.elements.size > to.elements.size -> LocalStackElementAnimations.current.exit
    from.elements.size < to.elements.size -> LocalStackElementAnimations.current.enter
    else -> LocalStackElementAnimations.current.idle
  }
```

Придумать реализацию модальных экранов

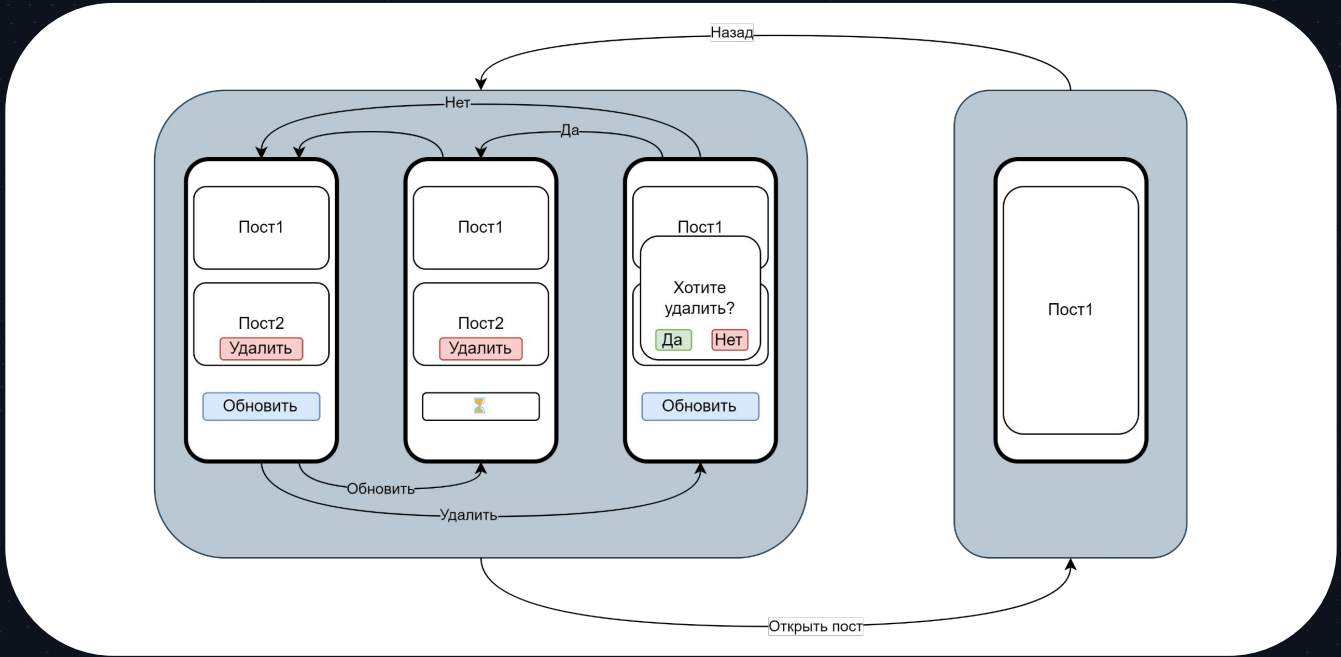




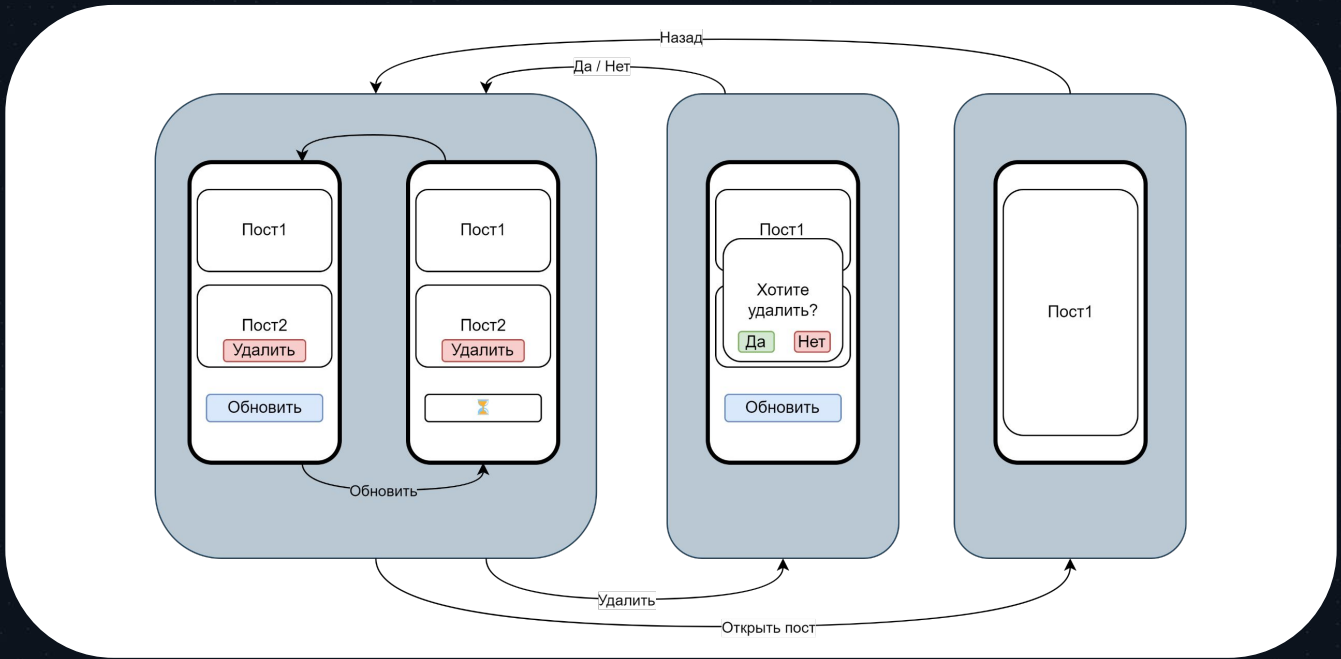
Придумать реализацию модальных экранов



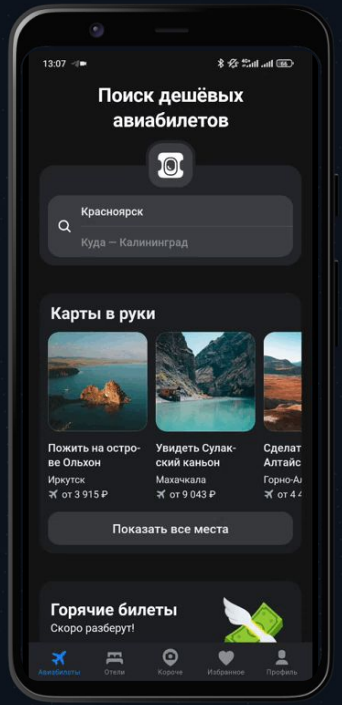
Придумать реализацию модальных экранов



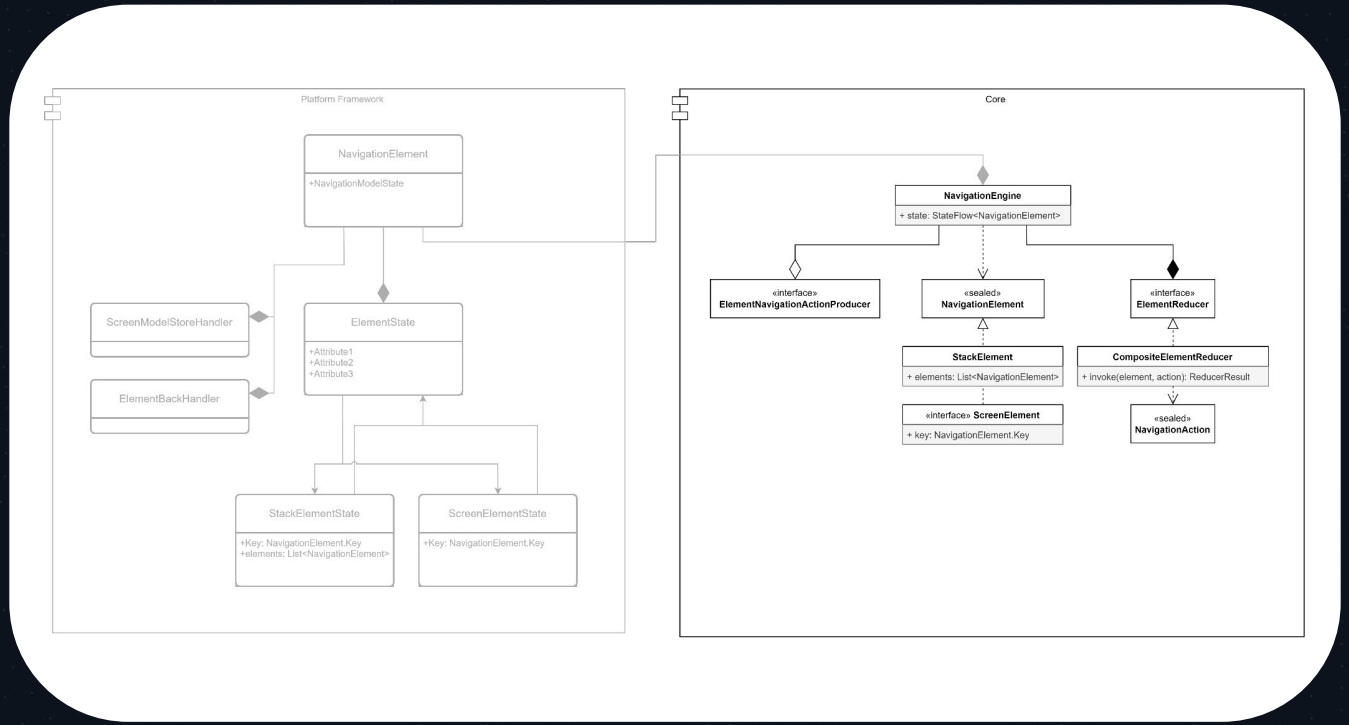
Придумать реализацию модальных экранов



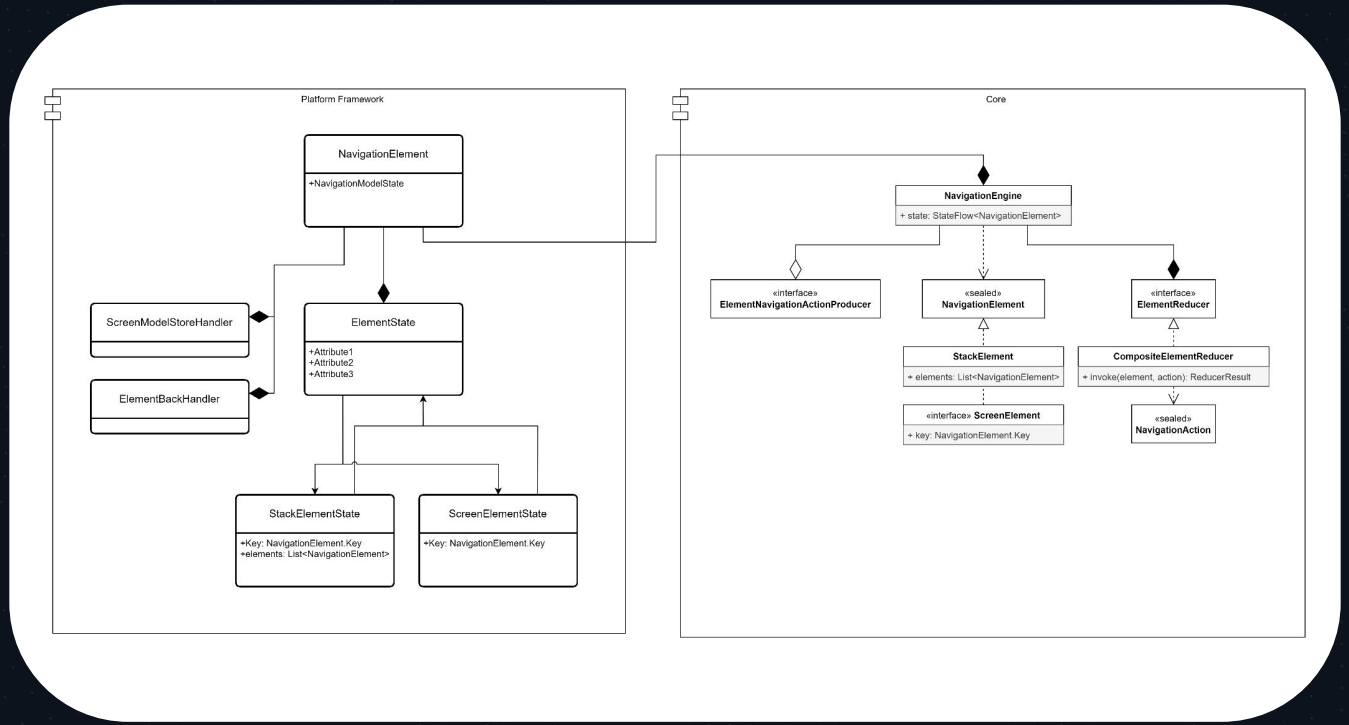
Придумать реализацию модальных экранов



Разбивка на модули



Многомодульность



Сахар

ElementNavigation Kotlin

```kotlin
@Composable
fun ElementNavigation(
  actionProducer: ElementNavigationActionProducer,
  actionEmitter: ElementNavigationActionEmitter,
  coroutineScope: CoroutineScope,
  element: NavigationElement,
) {
  val engine: NavigationEngine = remember {
    NavigationEngine(
      actionProducer = actionProducer,
      coroutineScope = coroutineScope,
      initialElement = element,
    )
  }
  <..>
}
```

Сахар

ElementNavigation Kotlin

```kotlin
@Composable
fun ElementNavigation(
  actionProducer: ElementNavigationActionProducer,
  actionEmitter: ElementNavigationActionEmitter,
  coroutineScope: CoroutineScope,
  element: NavigationElement,
) {
  <..>
  CompositionLocalProvider(
    LocalSaveableStateHolder provides saveableStateHolder,
    LocalActionEmitterProvider provides actionEmitter,
    LocalStackElementAnimations provides slideAnimations,
  ) {
    <..>
  }
}
```

Сахар

ElementNavigation — Kotlin

```kotlin
@Composable
fun ElementNavigation(
  actionProducer: ElementNavigationActionProducer,
  actionEmitter: ElementNavigationActionEmitter,
  coroutineScope: CoroutineScope,
  element: NavigationElement,
) {
  <..>
  CompositionLocalProvider(<..>) {
    <..>
    ElementBackHandler(state.value)
    ScreenModelStoreHandler(state.value)
  }
}
```

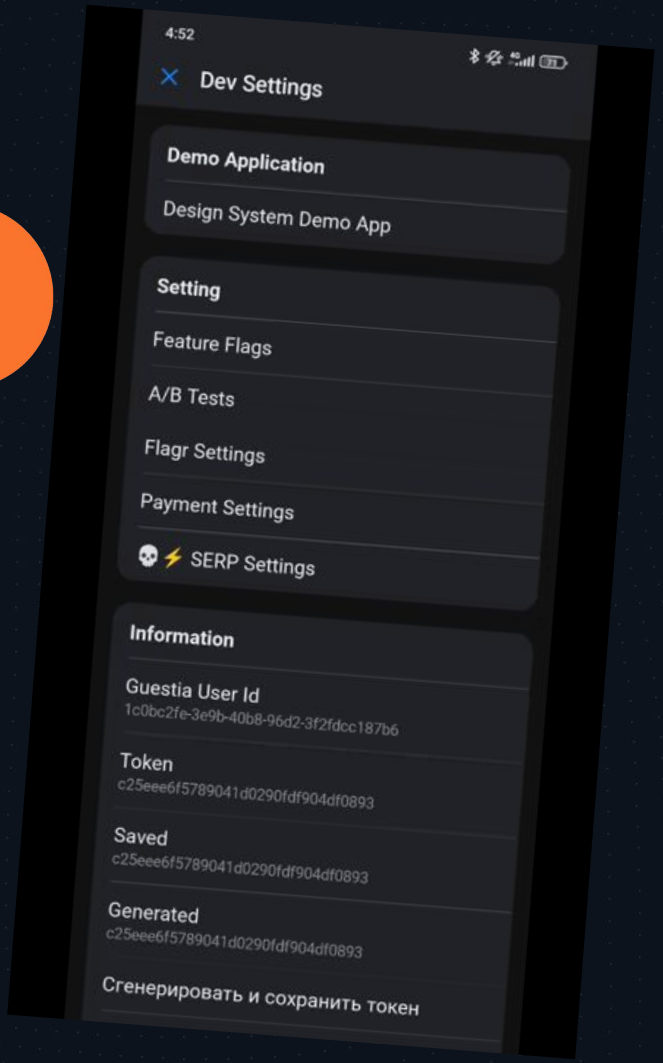
Сахар

ElementNavigation Kotlin

```kotlin
@Composable
fun ElementNavigation(
  actionProducer: ElementNavigationActionProducer,
  actionEmitter: ElementNavigationActionEmitter,
  coroutineScope: CoroutineScope,
  element: NavigationElement,
) {
  CompositionLocalProvider(<..>) {
    val state: State<ComposeNavigationState> = engine.state
        .flowAsComposeNavigationState()
        .collectAsState()

    ElementState(current = state.value.current, target = state.value.target)

    <...>
  }
}
```

Интеграция

DevSettingsScreen



Интеграция

DevSettingsProductFragment — Kotlin

```kotlin
override fun onCreateView(<..>): View =
    ThemedComposeView {
      val root = stack("Root", DevSettingsHomeScreenElement)

      ElementNavigation(
        actionProducer = component.composeNavigator,
        actionEmitter = component.composeNavigator,
        coroutineScope = lifecycleScope,
        element = root
      )
    }
```

Создаем Composable Content

Интеграция

DevSettingsProductFragment — Kotlin

```kotlin
override fun onCreateView(<..>): View =
    ThemedComposeView {
      val root = stack("Root", DevSettingsHomeScreenElement)

      ElementNavigation(
        actionProducer = component.composeNavigator,
        actionEmitter = component.composeNavigator,
        coroutineScope = lifecycleScope,
        element = root
      )
    }
```

Определяем начальное состояние

Интеграция

DevSettingsProductFragment — Kotlin

```kotlin
override fun onCreateView(<..>): View =
    ThemedComposeView {
      val root = stack("Root", DevSettingsHomeScreenElement)

      ElementNavigation(
        actionProducer = component.composeNavigator,
        actionEmitter = component.composeNavigator,
        coroutineScope = lifecycleScope,
        element = root
      )
    }
```

Вызываем главную функцию

DevSettingsScreen

DevSettingsHomeRouterImpl — Kotlin

```kotlin
override fun openFeatureFlags() {
  navigator open FeatureFlagsSettingsScreenElement
}
```

DevSettingsScreen

ElementNavigatorImpl — Kotlin

```kotlin
private val _composeActions = Channel<NavigationAction>(Channel.UNLIMITED)
override val composeActions: Flow<NavigationAction> = _composeActions.receiveAsFlow()


override fun open(element: NavigationElement) {
  _composeActions.trySend(Open(element))
}
```

Автотесты

Reducer Test

ElementStackReducerTest — Kotlin

```kotlin
@RunWith(JUnitParamsRunner::class)
class ElementStackReducerTest {

  @Parameters(method = "getElementStackReducerDataSet")
  @Test
  fun `stack reducer with initial state should reduce as expected for passed action`(
    initialState: StackElement,
    expected: ReducerResult,
    action: StackNavigationAction,
  ) {
    assertThat(ElementStackReducer(initialState, action)).isEqualTo(expected)
  }

 <..>
}
```

Reducer Test

ElementStackReducerTest — Kotlin

```kotlin
private fun getElementStackReducerDataSet() = navigationParametrizedTest {
  testsFromInitialState(stack(STACK_ALT_NAVIGATION_KEY, Screen1)) {
    Open(Screen2) expectHandled stack(STACK_ALT_NAVIGATION_KEY, Screen1, Screen2)

    Replace(Screen2) expectHandled stack(STACK_ALT_NAVIGATION_KEY, Screen2)
    Replace(Screen2, Screen3) expectHandled stack(STACK_ALT_NAVIGATION_KEY, Screen2, Screen3)
  }

  testsFromInitialState(stack(STACK_ALT_NAVIGATION_KEY, Screen1, Screen2, Screen3)) {
    Back expectHandled stack(STACK_ALT_NAVIGATION_KEY, Screen1, Screen2)
  }
}
```

Reducer Test

ElementStackReducerTest — Kotlin

```kotlin
private fun getElementStackReducerDataSet() = navigationParametrizedTest {
  testsFromInitialState(stack(STACK_ALT_NAVIGATION_KEY, Screen1)) {
    Open(Screen2) expectHandled stack(STACK_ALT_NAVIGATION_KEY, Screen1, Screen2)

    Replace(Screen2) expectHandled stack(STACK_ALT_NAVIGATION_KEY, Screen2)
    Replace(Screen2, Screen3) expectHandled stack(STACK_ALT_NAVIGATION_KEY, Screen2, Screen3)
  }

  testsFromInitialState(stack(STACK_ALT_NAVIGATION_KEY, Screen1, Screen2, Screen3)) {
    Back expectHandled stack(STACK_ALT_NAVIGATION_KEY, Screen1, Screen2)
  }
}
```

Reducer Test

ElementStackReducerTest — Kotlin

```kotlin
private fun getElementStackReducerDataSet() = navigationParametrizedTest {
  testsFromInitialState(stack(STACK_ALT_NAVIGATION_KEY, Screen1)) {
    Open(Screen2) expectHandled stack(STACK_ALT_NAVIGATION_KEY, Screen1, Screen2)

    Replace(Screen2) expectHandled stack(STACK_ALT_NAVIGATION_KEY, Screen2)
    Replace(Screen2, Screen3) expectHandled stack(STACK_ALT_NAVIGATION_KEY, Screen2, Screen3)
  }

  testsFromInitialState(stack(STACK_ALT_NAVIGATION_KEY, Screen1, Screen2, Screen3)) {
    Back expectHandled stack(STACK_ALT_NAVIGATION_KEY, Screen1, Screen2)
  }
}
```

Тесты вспомогательных классов

ElementKeysSubtractTest — Kotlin

```kotlin
  @Test
  fun `should return empty set if elements equals`() {
    val firstElement = stack(STACK_ROOT_NAVIGATION_KEY, Screen1, stack(STACK_NAVIGATION_KEY,
Screen2), Screen4)
    val secondElement = stack(STACK_ROOT_NAVIGATION_KEY, Screen1, stack(STACK_NAVIGATION_KEY,
Screen2), Screen4)

    val actual = firstElement.subtract(secondElement)
    val expected = emptySet<NavigationElement.Key>()

    assertThat(actual).assertThat(expected)
  }
```

Compose-Testing

StackElementStateTest — Kotlin

```kotlin
@get:Rule val composeTestRule: ComposeContentTestRule = createComposeRule()

@Test
fun StackElementStateTest() {
  <..>
}
```

Compose-Testing

StackElementStateTest — Kotlin

```kotlin
@get:Rule val composeTestRule: ComposeContentTestRule = createComposeRule()

@Test
fun StackElementStateTest() {
  val target = stack("Stack", MockScreen, TestScreen)

  <..>
}
```

Compose-Testing

StackElementStateTest — Kotlin

```kotlin
@get:Rule val composeTestRule: ComposeContentTestRule = createComposeRule()

@Test
fun StackElementStateTest() {
  val target = stack("Stack", MockScreen, TestScreen)

  composeTestRule.setContent {
    <..>
  }

  <..>
}
```

Compose-Testing

StackElementStateTest — Kotlin

```kotlin
@get:Rule val composeTestRule: ComposeContentTestRule = createComposeRule()

@Test
fun StackElementStateTest() {
  val target = stack("Stack", MockScreen, TestScreen)

  composeTestRule.setContent {
    AppTheme() {
      CompositionLocalProvider() {
        <..>
      }
    }
  }

  <..>
}
```

Compose-Testing

StackElementStateTest — Kotlin

```kotlin
@get:Rule val composeTestRule: ComposeContentTestRule = createComposeRule()

@Test
fun StackElementStateTest() {
  val target = stack("Stack", MockScreen, TestScreen)

  composeTestRule.setContent {
    AppTheme() {
      CompositionLocalProvider() {
        StackElementState(current = null, target = target)
      }
    }
  }

  <..>
}
```

Compose-Testing

StackElementStateTest — Kotlin

```kotlin
@get:Rule val composeTestRule: ComposeContentTestRule = createComposeRule()

@Test
fun StackElementStateTest() {
  val target = stack("Stack", MockScreen, TestScreen)

  composeTestRule.setContent {
    AppTheme() {
      CompositionLocalProvider() {
        StackElementState(current = null, target = target)
      }
    }
  }

  composeTestRule.onNode(ScreenMatcher).assertIsDisplayed()
}
```

Суммирование

Внедрение

Мы выбрали потенциальные фичи для внедрения
и поэкспериментировали с ними

Поняли, какие нужны доработки

Реализовали новую навигацию и раскатили
на пользователей через эксперимент

После успешного эксперимента оставили
только новую навигацию

Далее в планах «расходиться» от этих фичей
и покрывать более крупные куски поэтапно,
через эксперименты

Спасибо!