Как переписать приложение с нуля и потерпеть фиаско

Агейченко Александр

Android Teamlead @ Distillery

wackalooon@gmail.com

@Wackaloon

Кто говорит?

Андроид разработчик

Тимлид

Любитель блестящих камушков

7 лет в АйТи

Аутсорсер со стажем

О чем доклад

Важность команды

Плохие решения

Хард-форк проекта

Соединение двух проектов в один

Многомодульность

Dependency Injection

Смена исходников проекта в рантайме

Не повторяйте наших ошибок

Кому будет полезно

Андроид разработчикам Middle-Senior

Менеджерам

Тимлидам

Общий план

С чего все началось

Разделение проектов

Соединение проектов обратно

Шаринг модулей и фич

Соединение 2 проектов в одном АПК

Плавный перевод пользователей между проектами

Место преступления: Проект

Финтех стартап с 7 летним стажем

5 000 000+ скачиваний в маркете*

Работает в развивающихся странах

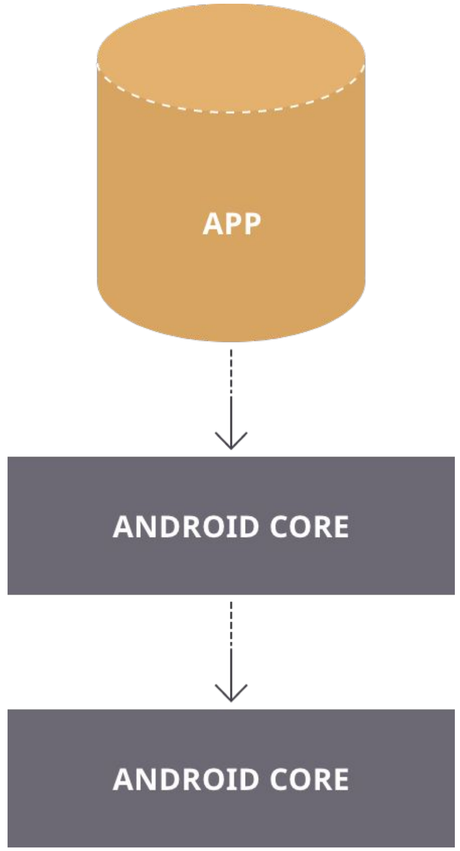
* по мнению менеджеров

Место преступления: Android

3 модуля

Сетевой слой на AsyncTask

~200 000 строк кода



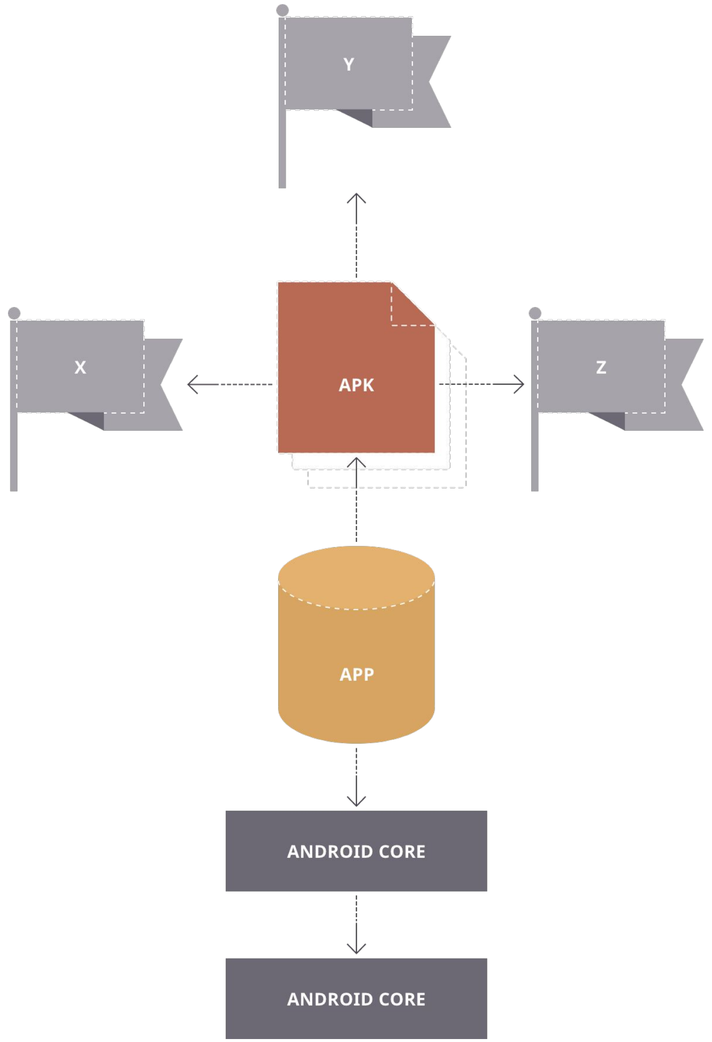
Место преступления: Android

Разные APK для разных стран

Начали затаскивать RxJava

Начали использовать Kotlin

Покрытие тестами ~10%*



* по мнению JaCoCo

Что случилось?

Проект работает в странах X, Y и Z, только что запустились в S

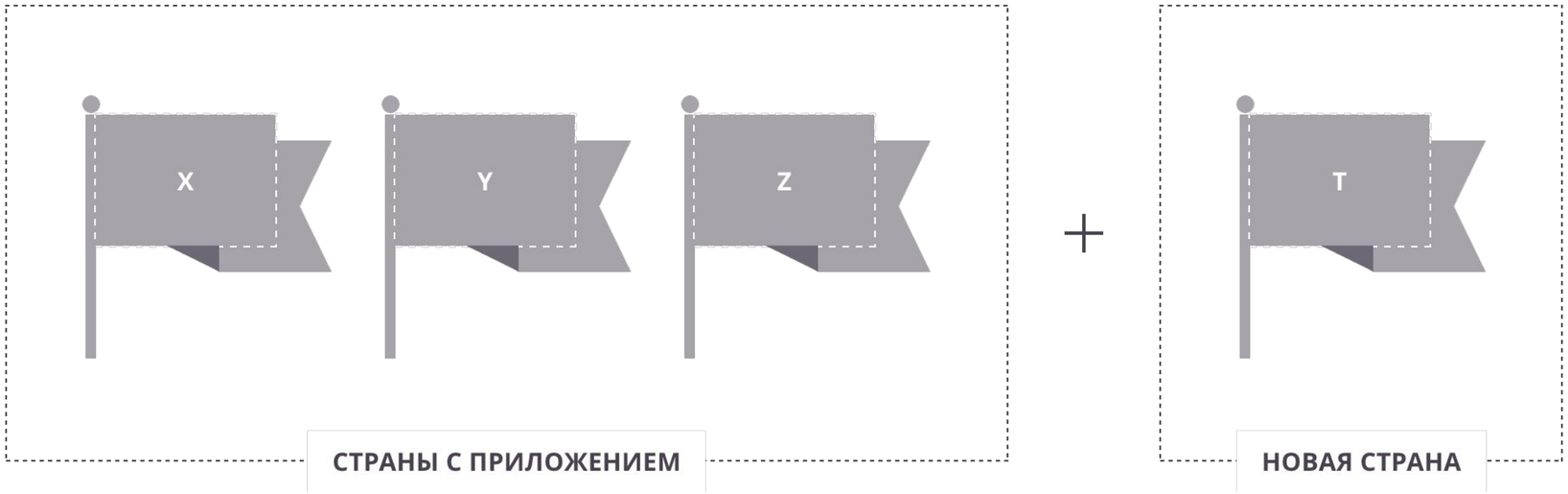
Что случилось?

Проект работает в странах X, Y и Z, только что запустились в S

Планируется выходить в новую страну T

Следующая страна потенциально x9000 по нагрузке от текущих

Сервера текущую нагрузку держат средне



Первые сложности

Сервер говорит клиенту какой экран открыть

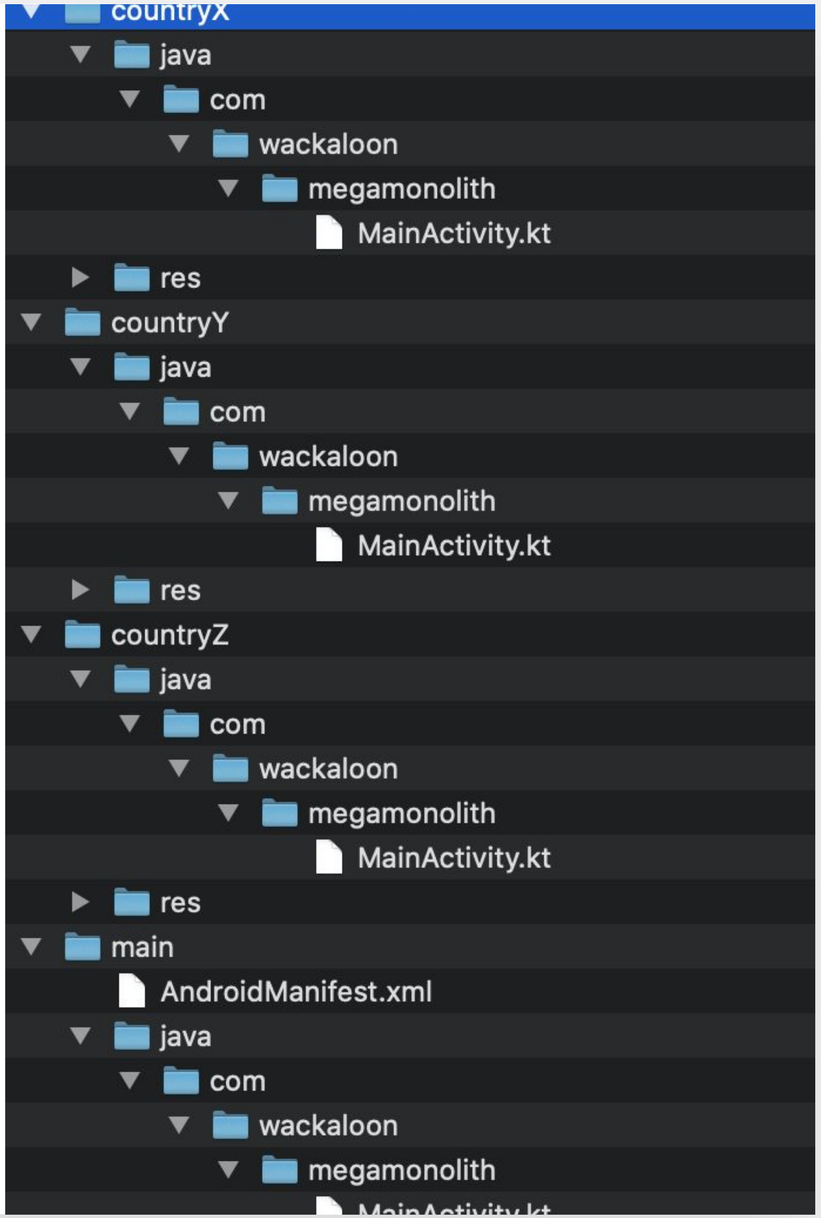
Десятки экранов которые отличаются в разных странах

src/**main**/java/MainActivity.java

src/**countryX**/java/MainActivity.java

src/**countryY**/java/MainActivity.java

src/**countryZ**/java/MainActivity.java



А может не надо?

Технодемка и аналитика замечательные

Конкуренты на рынке слабые

Расширение неизбежно

Сделаем все правильно

Общий план

- ✓ С чего все началось
- **Разделение проектов**
- Соединение проектов обратно
- Шаринг модулей и фич
- Соединение 2 проектов в одном АПК
- Плавный перевод пользователей между проектами

Сложный шаг

Масштабируемость

Необходимо переписать бекенд

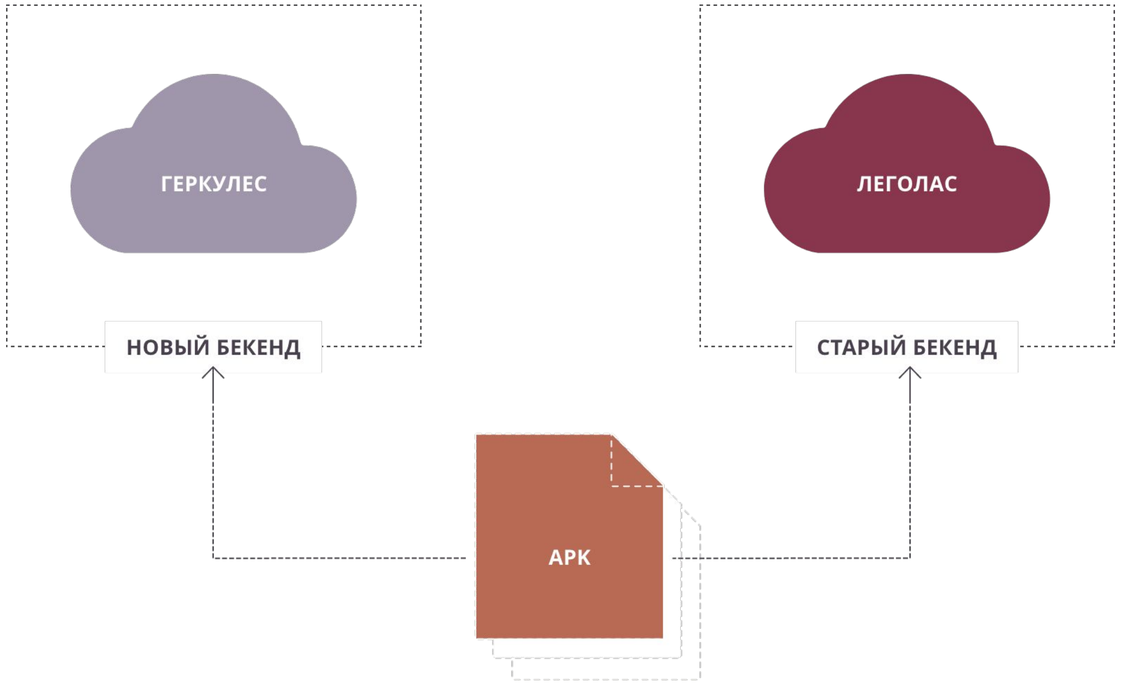
Сможем выходить на новый рынок хоть каждый день

Оптимизм

Единая APK

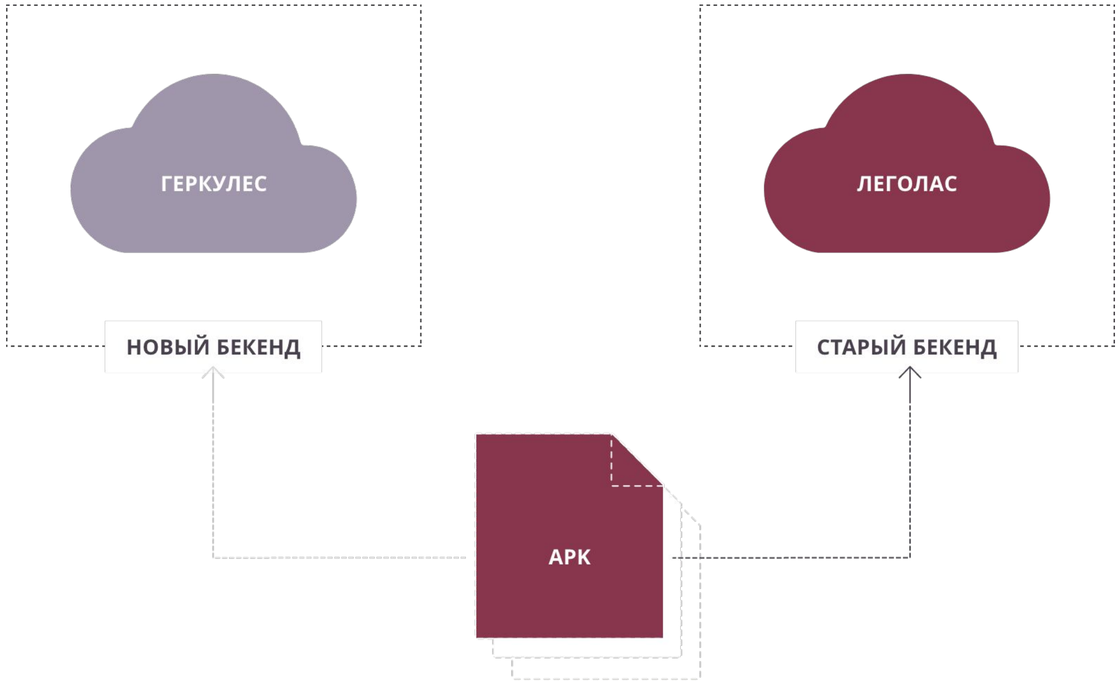
Сделаем архитектуру нормально

MVP через несколько месяцев



Геркулес и Леголас

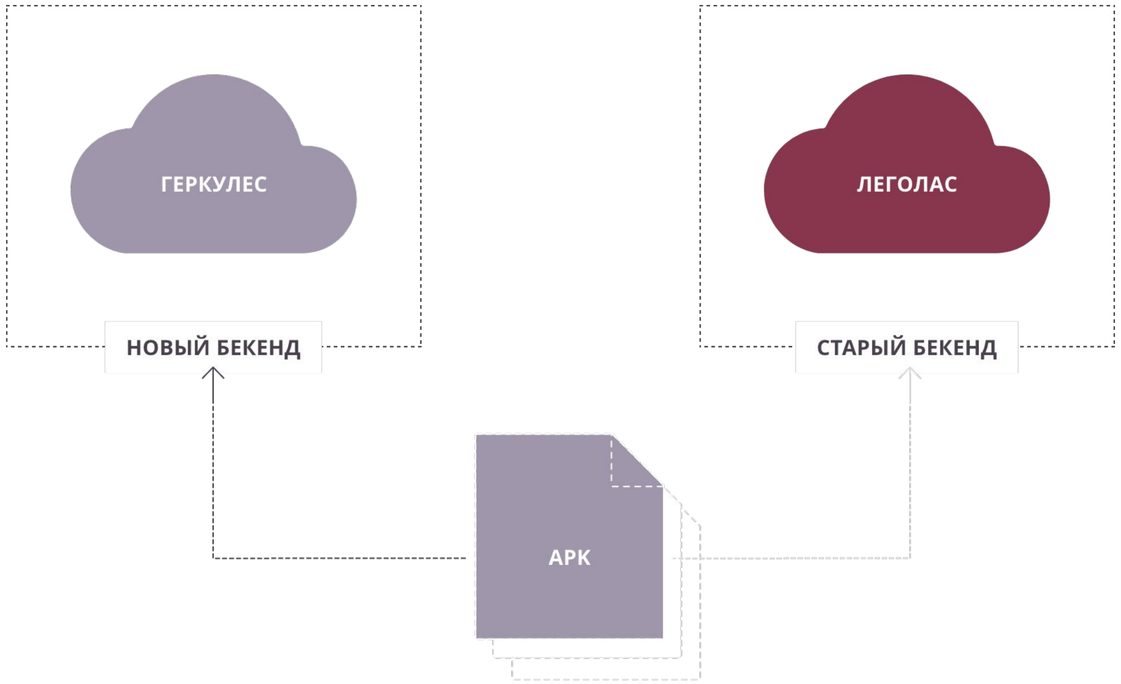
Новая архитектура бекенда



Геркулес и Леголас

Новая архитектура бекенда

Постепенно рефакторить

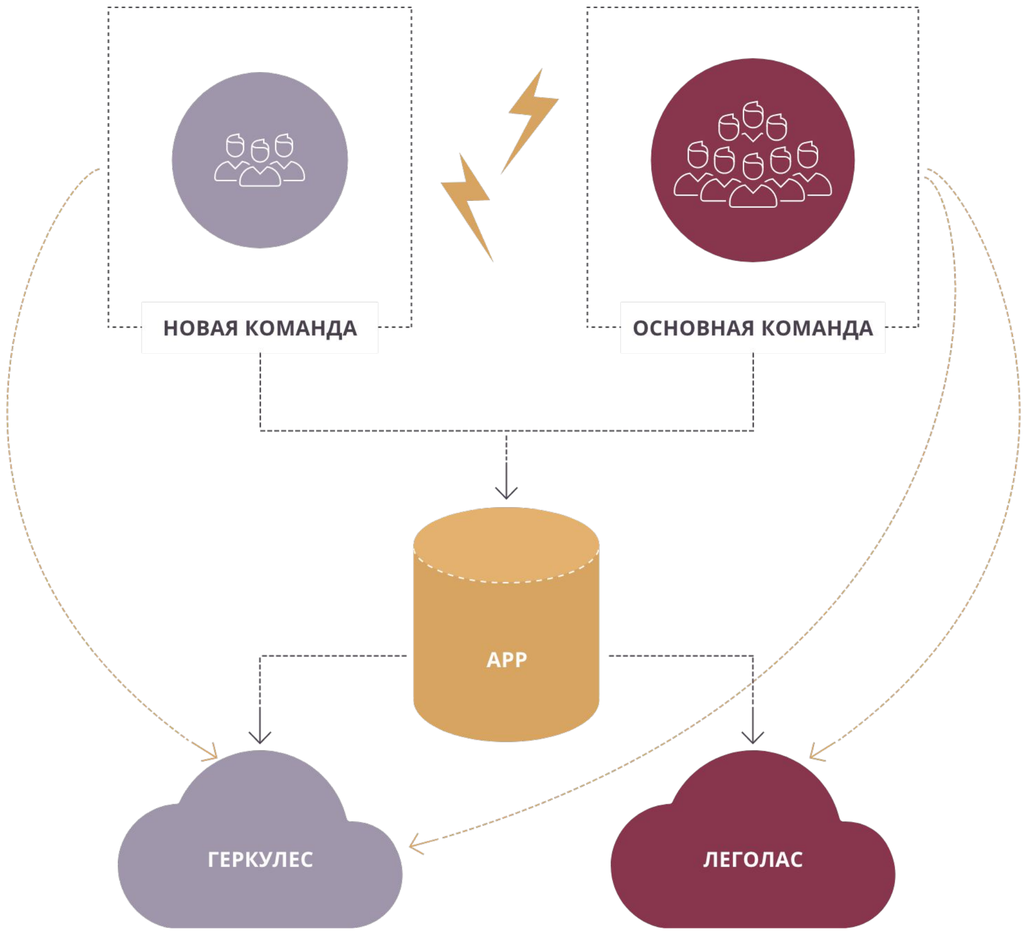


Наивность разрушается

Новая команда работает только над Геркулесом

Они не знают что и как работает в бизнесе

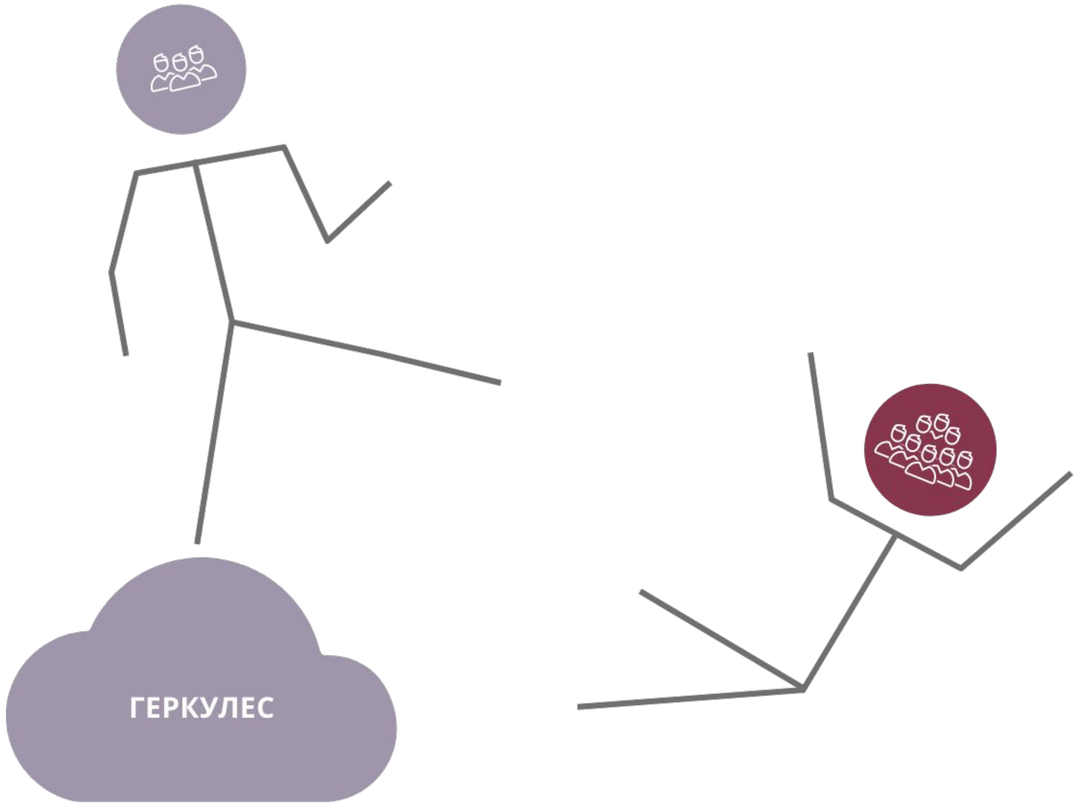
Начинаются конфликты



Первый звонок

Всех потихоньку вытесняют с Геркулеса

У новой команды свой менеджмент



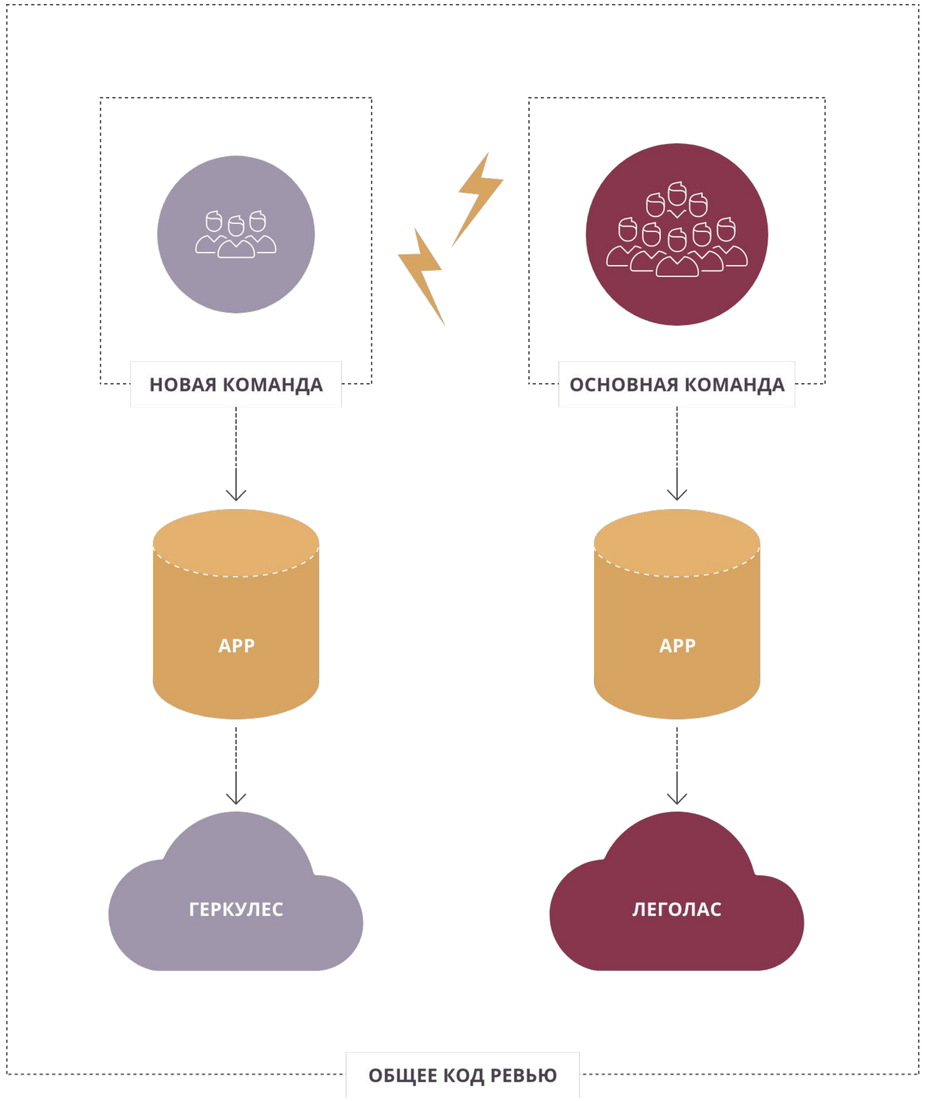
А давайте...

Сделаем хард форк!

Так будет быстрее

Выдадим MVP в срок

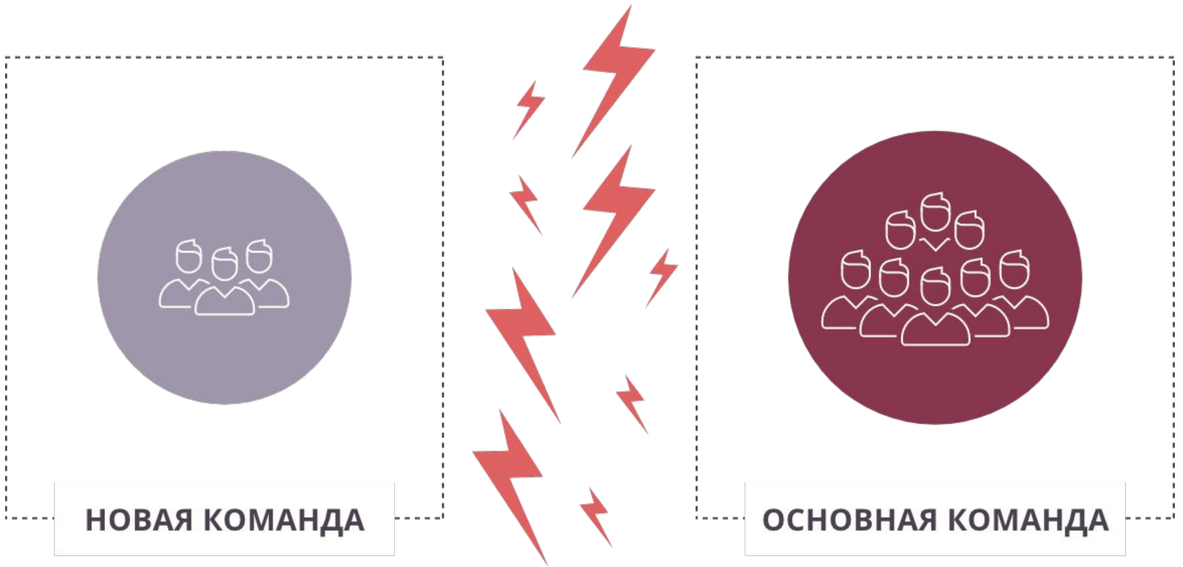
Скорость важнее



Второй звонок

Конфликты усиливаются

Недовольство растет



Мягкая комната

Команды изолируют

Мягкая комната

Конфликтов стало меньше

Дедлайны продлеваются

N лет спустя

Прошло 1.5 года

А мы пользователей будем переводить?

.

. .

. . .

AAAAAA!!!

Брейнштормим варианты

Big bang

Глобальный APK

Плавный переход

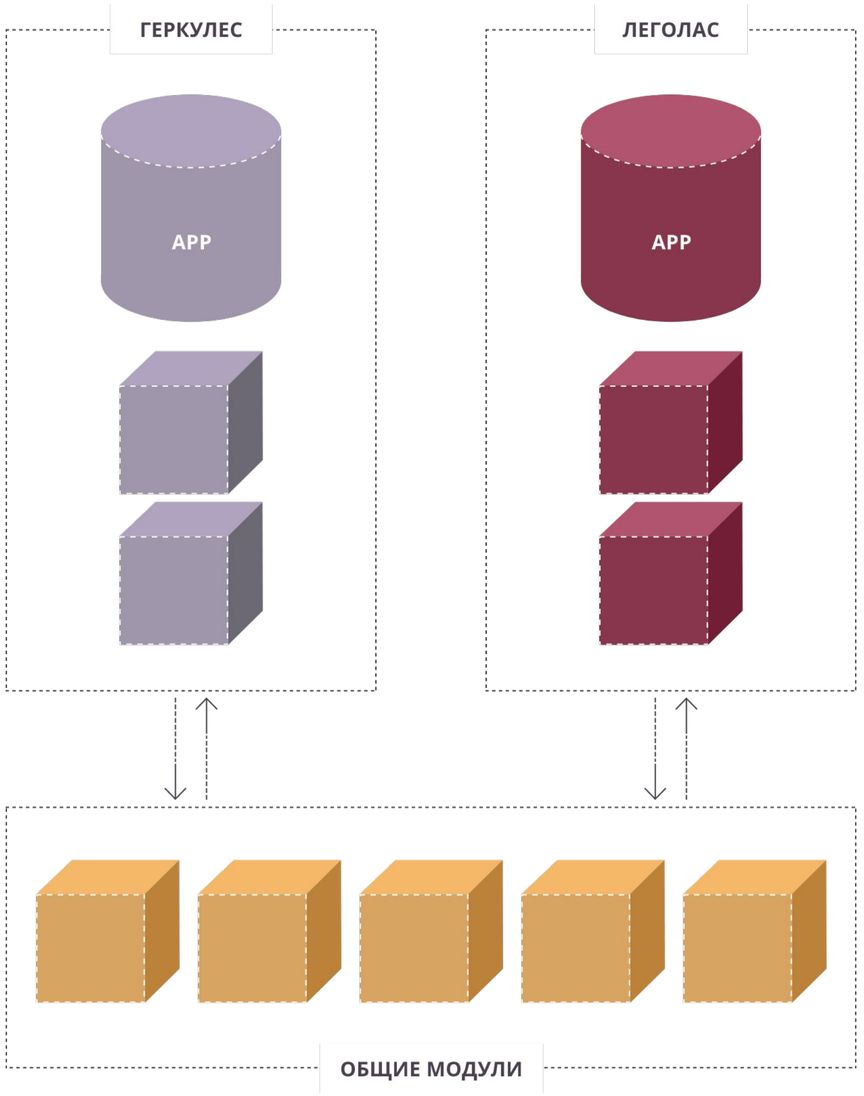
Общий план

- ✓ С чего все началось
- ✓ Разделение проектов
- **Соединение проектов обратно**
- Шаринг модулей и фич
- Соединение 2 проектов в одном АПК
- Плавный перевод пользователей между проектами

Просветление

Нивелировать разницу в коде

Общий код в общие модули



Вопросы

Поддержка версионирования?

Как шарить?

Как хранить?

Как шарить артефакты

Библиотека

Менеджер

Git gut

Репка

Git Subtree

Git Subrepo

Git Slave

В одной куче

Рижские шпроты или jar’ники в томате

Все фичи в отдельных репозиториях

Скомпилированные бинарники

Фичи как библиотеки

Библиотека: Плюсы

Очень быстрые локальные билды

Полная обратная совместимость

Обновление по мере необходимости

Инкрементальная разработка фич с альфа/бета версиями

Библиотека: Минусы

Можно забыть обновить библиотеку

Поддерживать очень много репозиториев

Постоянно держать несколько открытых андроид студий

Как шарить артефакты

- ✓ Библиотека
- **Менеджер**
- Git gut
- Репка
- Git Subtree
- Git Subrepo
- Git Slave
- В одной куче

Менеджер для шпрот

Repository Manager

- JFrog Artifactory
- Sonatype Nexus

Repository Manager: Плюсы

Просто и понятно

Всем знакомо

Все фичи в одном месте

Легкое переключение версий

Не надо компилировать руками

Repository Manager: Минусы

Рабочий процесс меняется

Как проверять изменения?

Еще один инструмент

Как шарить артефакты

- ✓ Библиотека
- ✓ Менеджер
- **Git gut**
- Репка
- Git Subtree
- Git Subrepo
- Git Slave
- В одной куче

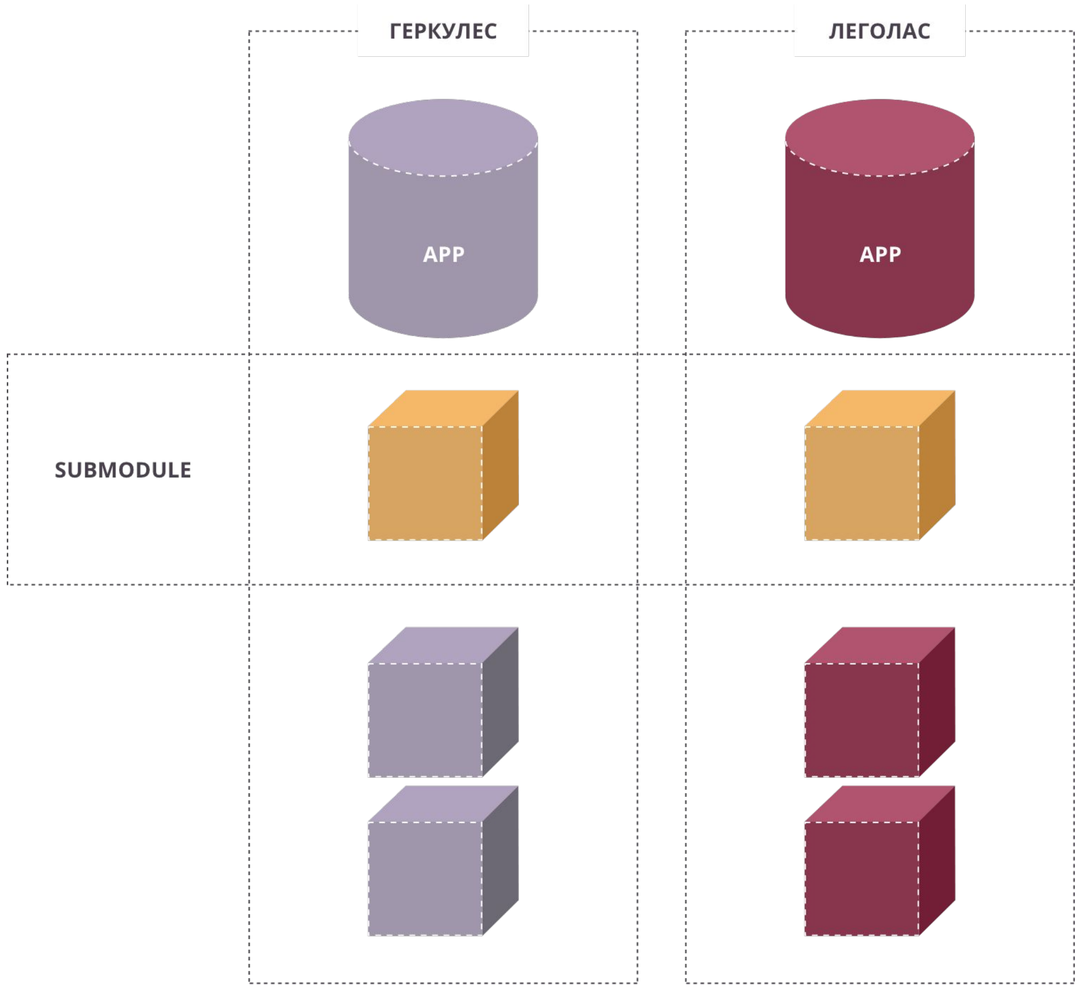
Git Gut

Git Submodules как папки основного репозитория

Версионирование по коммитам в сабмодуле

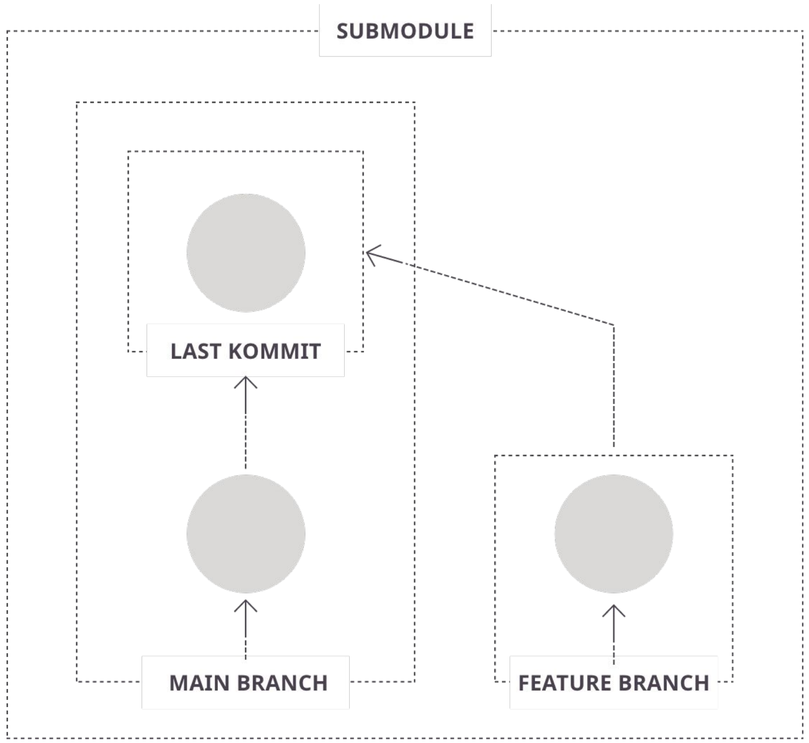
Можно отслеживать ветки

Один или несколько выделенных репозиториев



Git Submodules: Как мержить

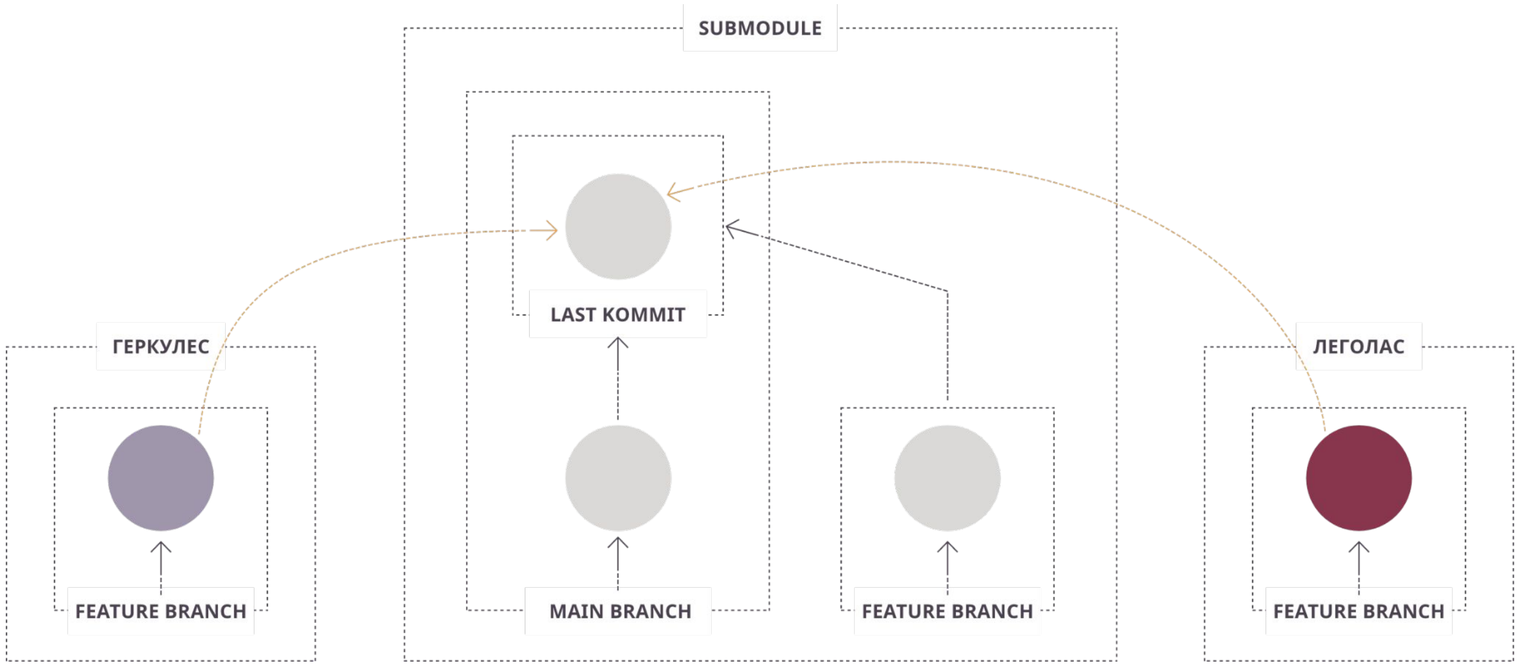
Мержим фича PR



Git Submodules: Как мержить

Мержим фича PR

Обновляем репозитории хостов

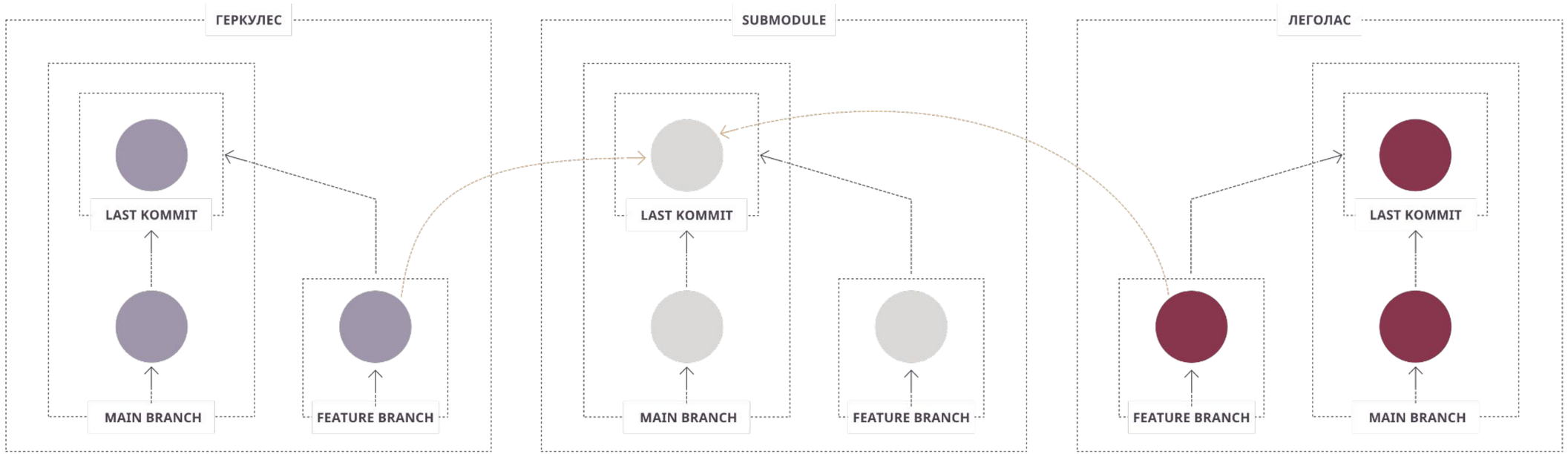


Git Submodules: Как мержить

Мержим фича PR

Обновляем репозитории хостов

Мержим 1-2 PR хостов



Git Submodules: Минусы

Нет контроля валидности сборки перед мержем

Хосты могут указывать на старую ревизию

Процесс разработки выглядит сильно сложнее

Нужно изучать новый инструмент

Git Submodules: Плюсы

Изоляция фич

Иллюзия работы в одном репо

Удобство сборки

Как шарить артефакты

- ✓ Библиотека
- ✓ Менеджер
- ✓ Git gut
- **Репка**
- Git Subtree
- Git Subrepo
- Git Slave
- В одной куче

Репка

Repo от Google

Атомарно мержит

Отслеживает ветки

Работает только с Gerrit

Репка: Плюсы

Может мержить все PR сразу

Поддерживает множество репозиториев

Есть плагины

Репка: Минусы

Трудно поднять и запустить

Экзотический инструмент

Сложность сборки проекта

Отсутствует GUI

Как шарить артефакты

- ✓ Библиотека
- ✓ Менеджер
- ✓ Git gut
- ✓ Репка

Git Subtree

Git Subrepo

Git Slave

В одной куче

Как шарить артефакты

- ✓ Библиотека
- ✓ Менеджер
- ✓ Git gut
- ✓ Репка
- ✓ Git Subtree
- ✓ Git Subrepo
- ✓ Git Slave

В одной куче

Монорельсы: Плюсы

Единая ветка мастер

Два проекта в одной студии

Компилировать оба проекта

Один репозиторий

Монорельсы: Минусы

Нет управления версиями

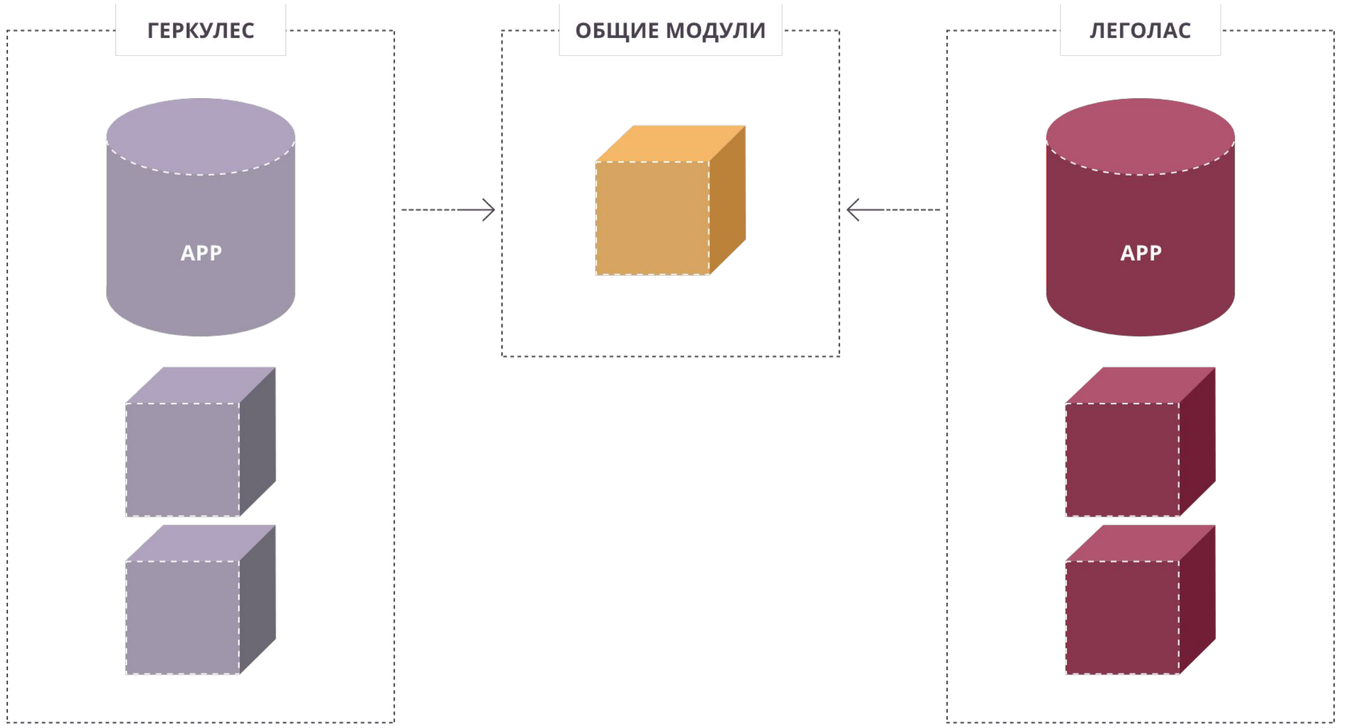
Свалка из модулей

Как шарить артефакты

- ✓ Библиотека
- ✓ Менеджер
- ✓ Git gut
- ✓ Репка
- ✓ Git Subtree
- ✓ Git Subrepo
- ✓ Git Slave
- ✓ В одной куче

Что делать с общими зависимостями?

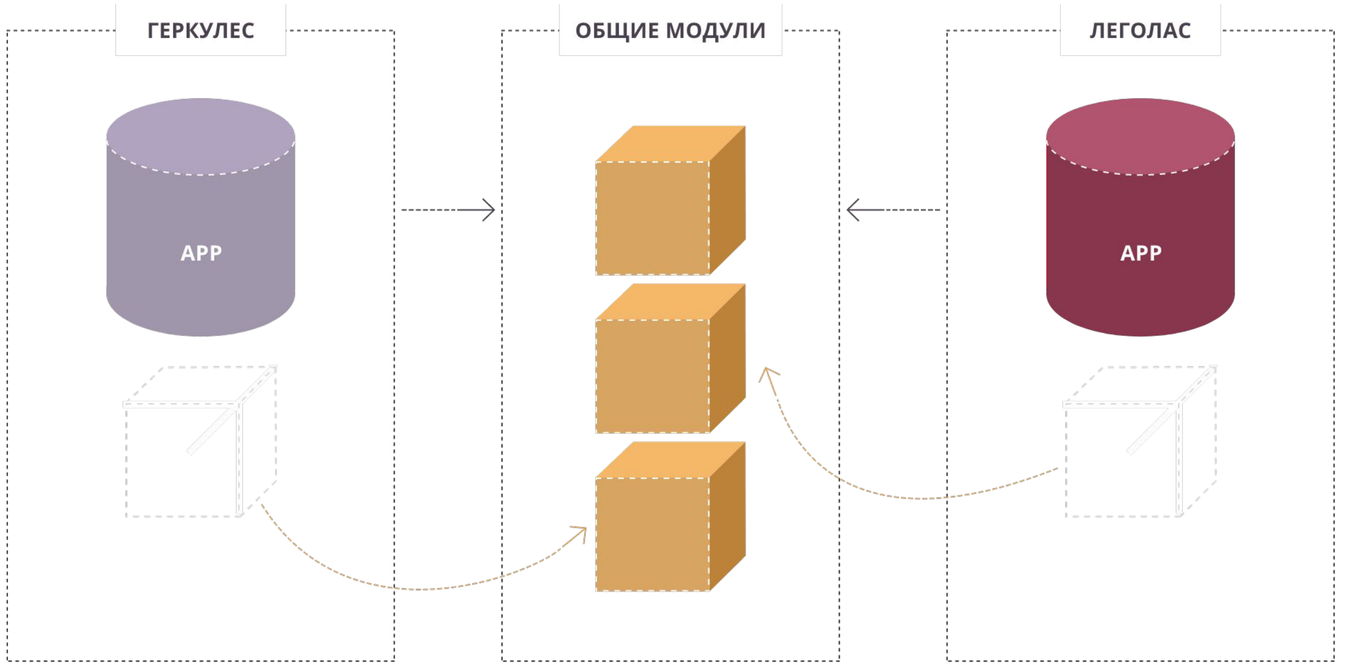
Выделить общие модули



Что делать с общими зависимостями?

Выделить общие модули

Постепенно перетащить модули

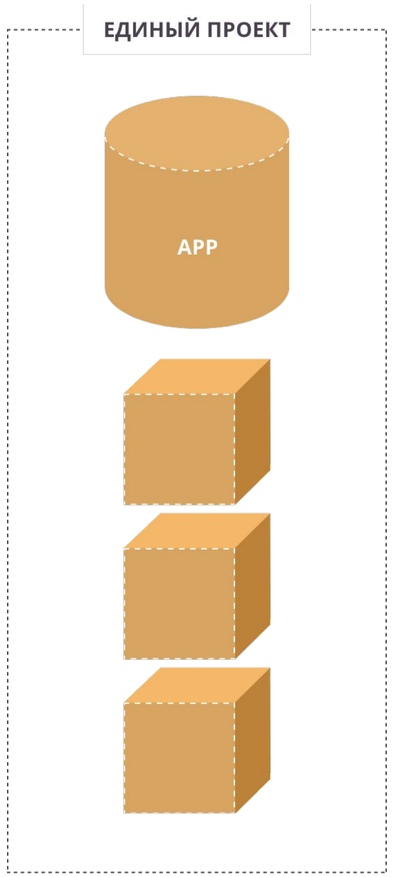


Что делать с общими зависимостями?

Выделить общие модули

Постепенно перетащить модули

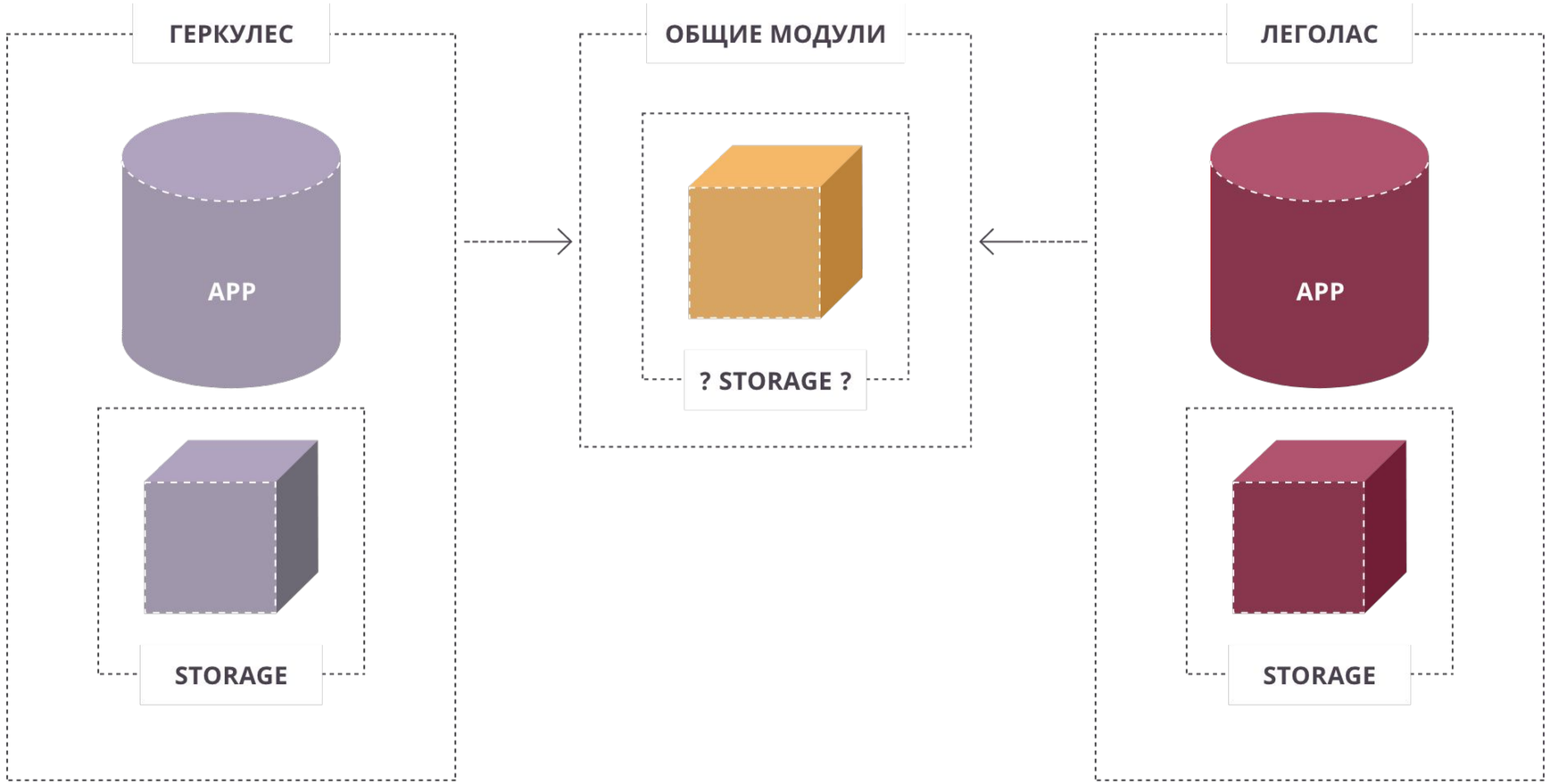
Перейти к единому проекту



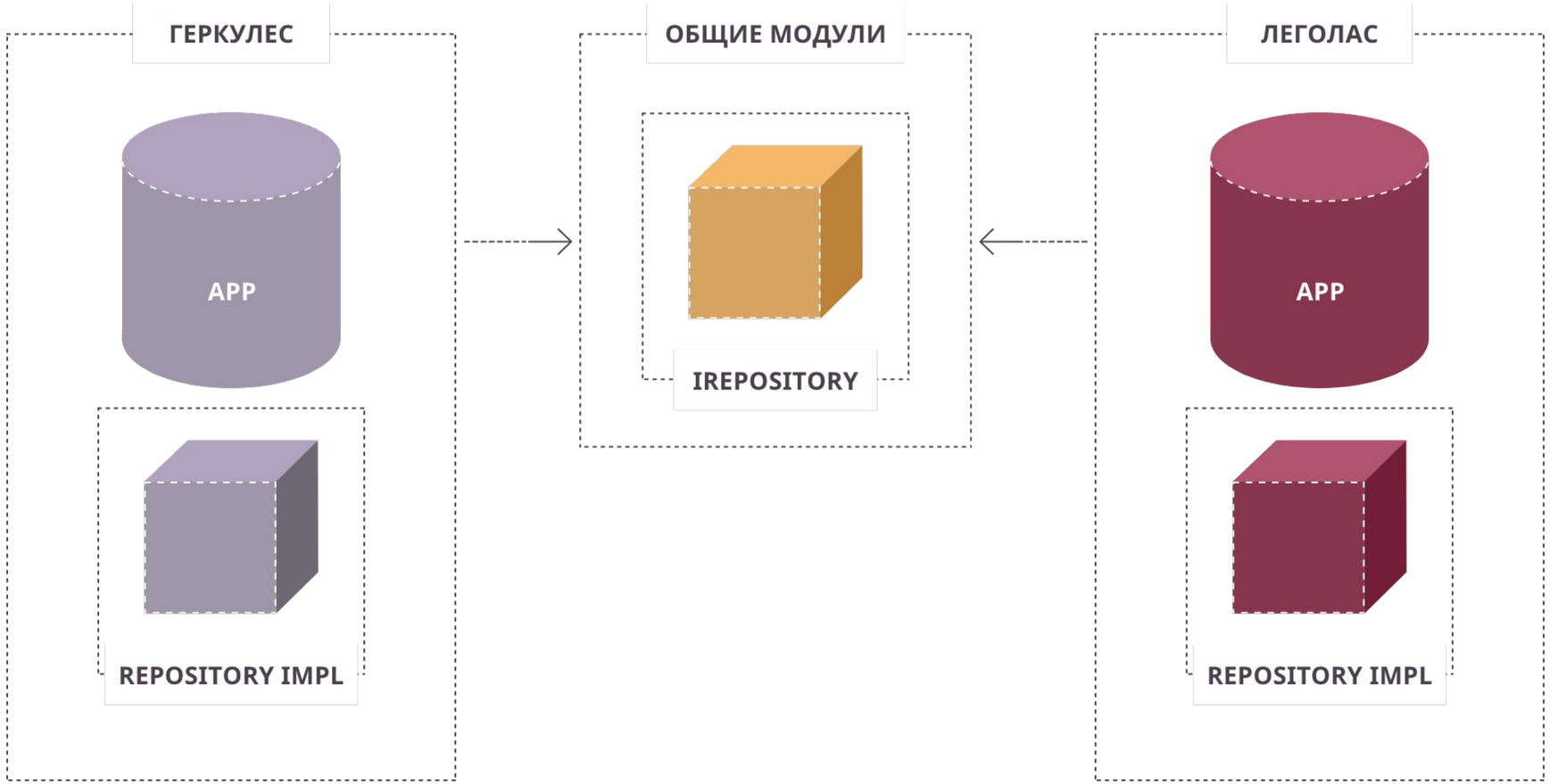
Общий план

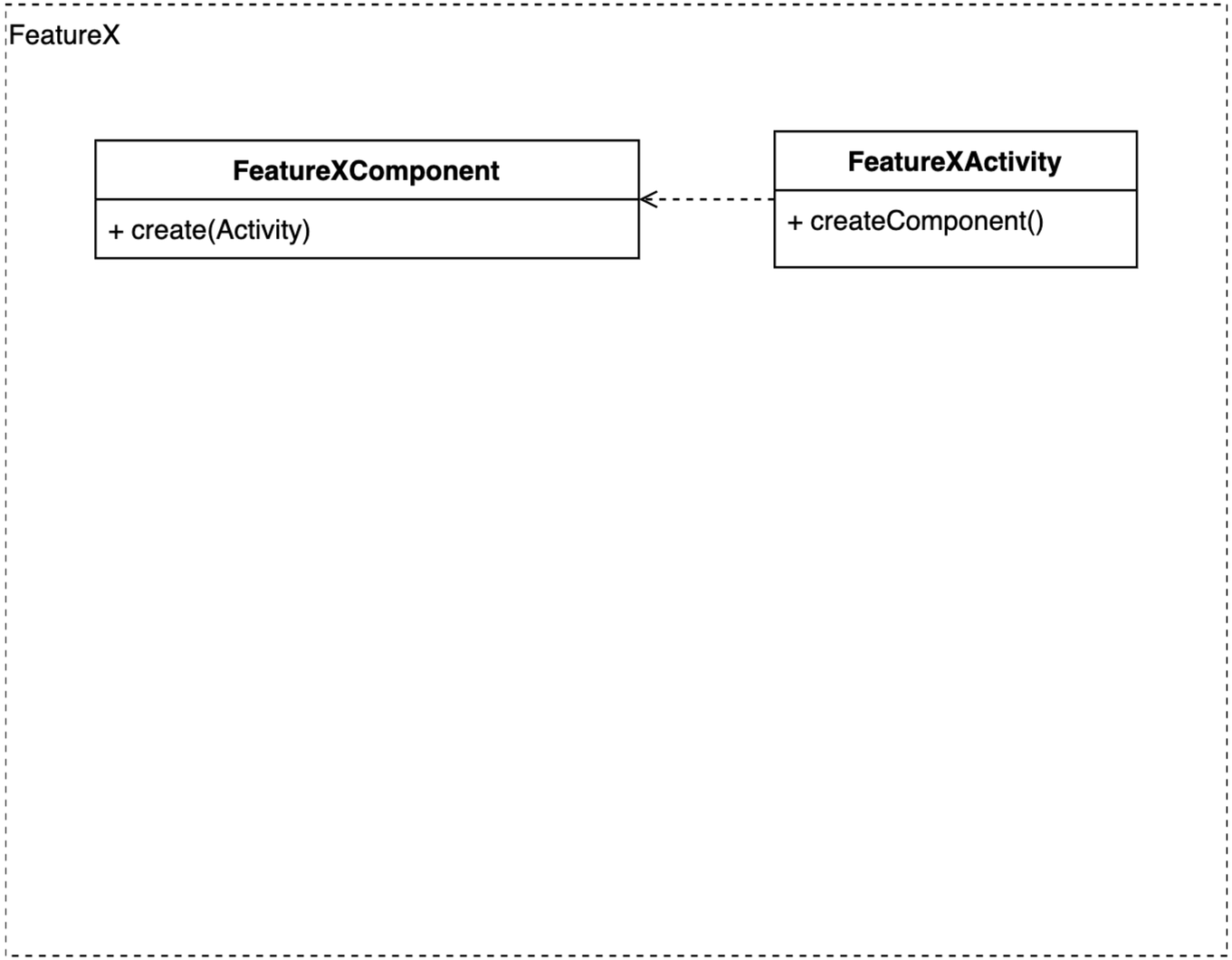
- ✓ С чего все началось
- ✓ Разделение проектов
- ✓ Соединение проектов обратно
- **Шаринг модулей и фич**
- Соединение 2 проектов в одном АПК
- Плавный перевод пользователей между проектами

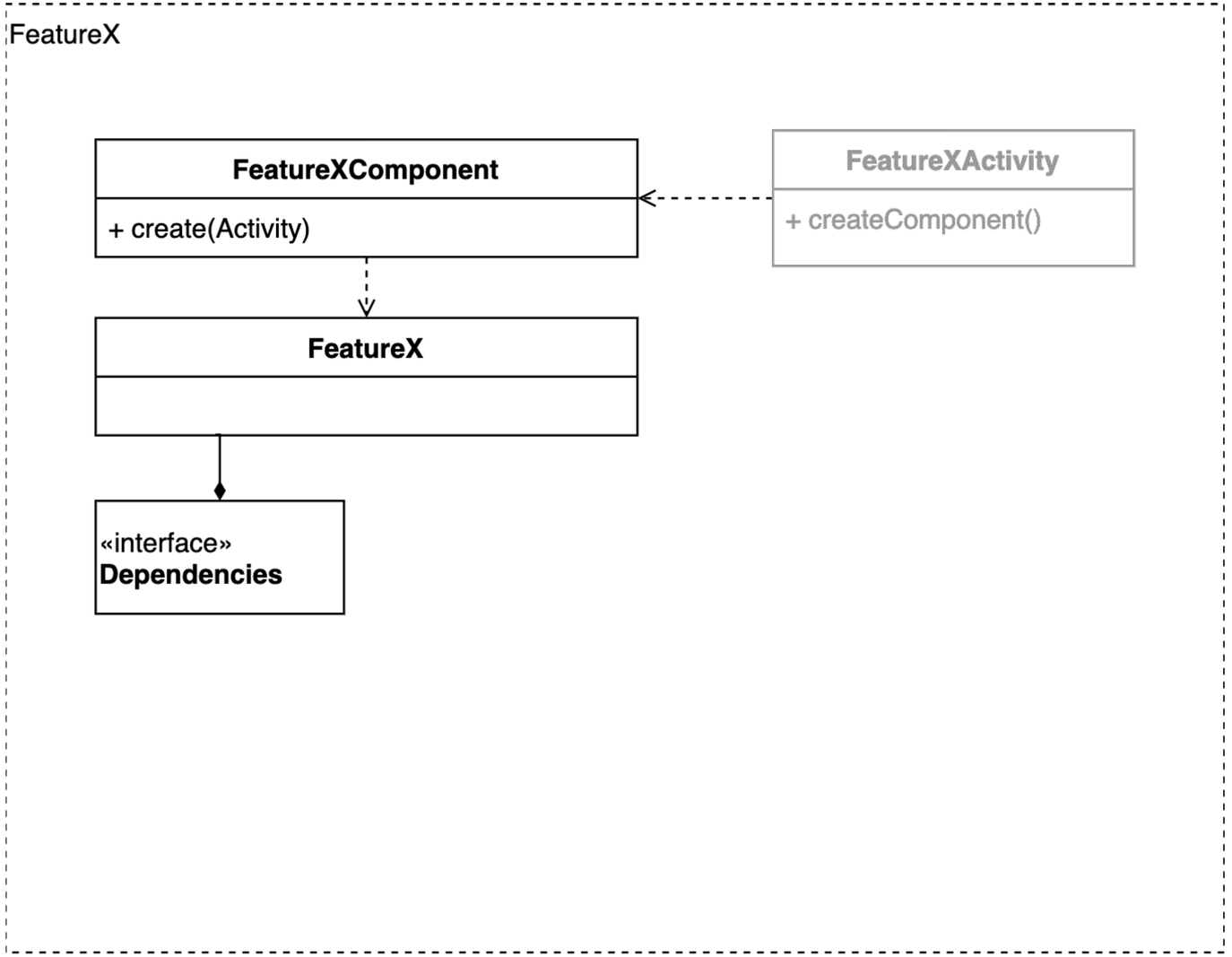
Даггер из Ада



Особенности национальной модуляризации

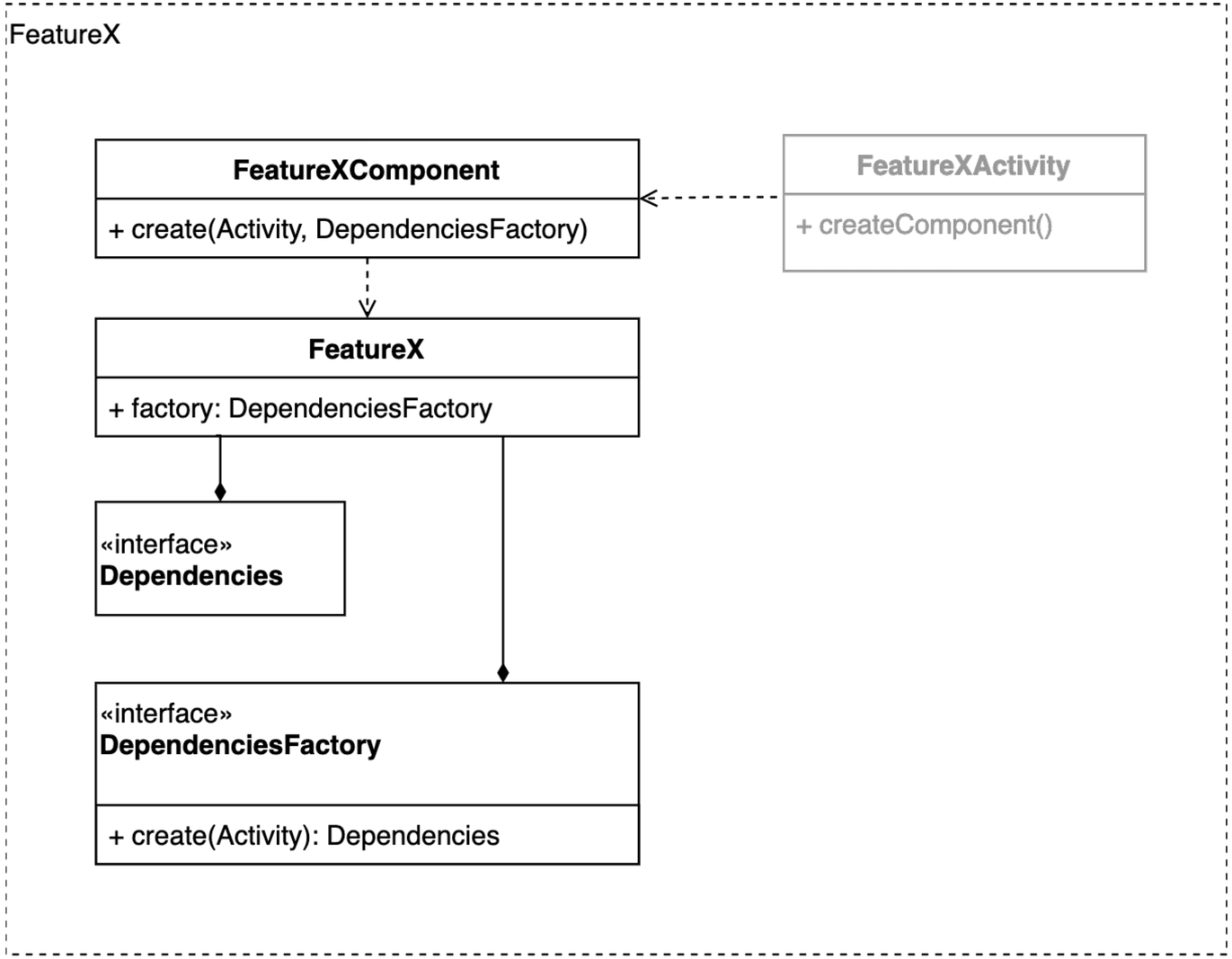


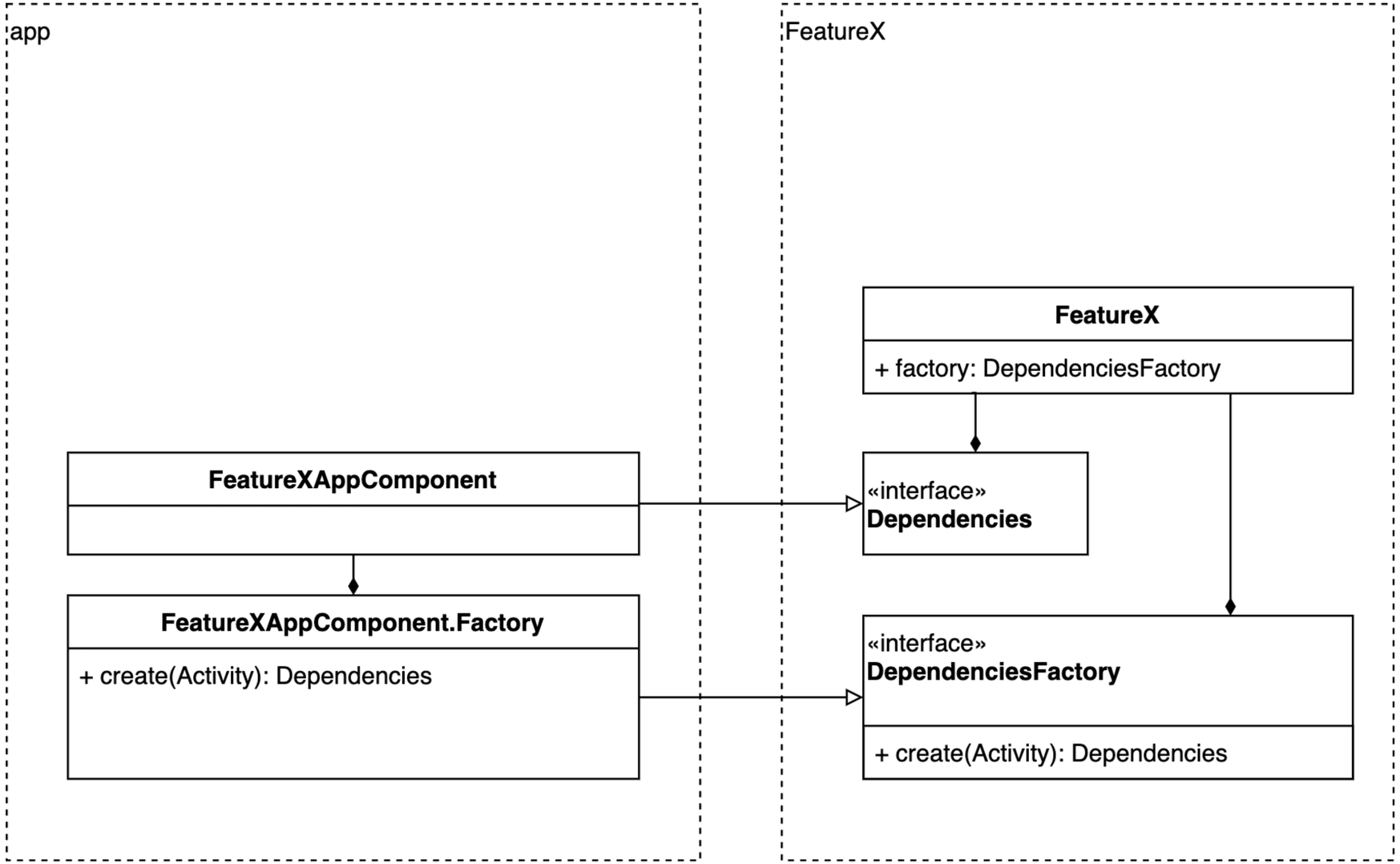

FeatureX
FeatureXComponent
+ create(Activity)
FeatureXActivity
+ createComponent()

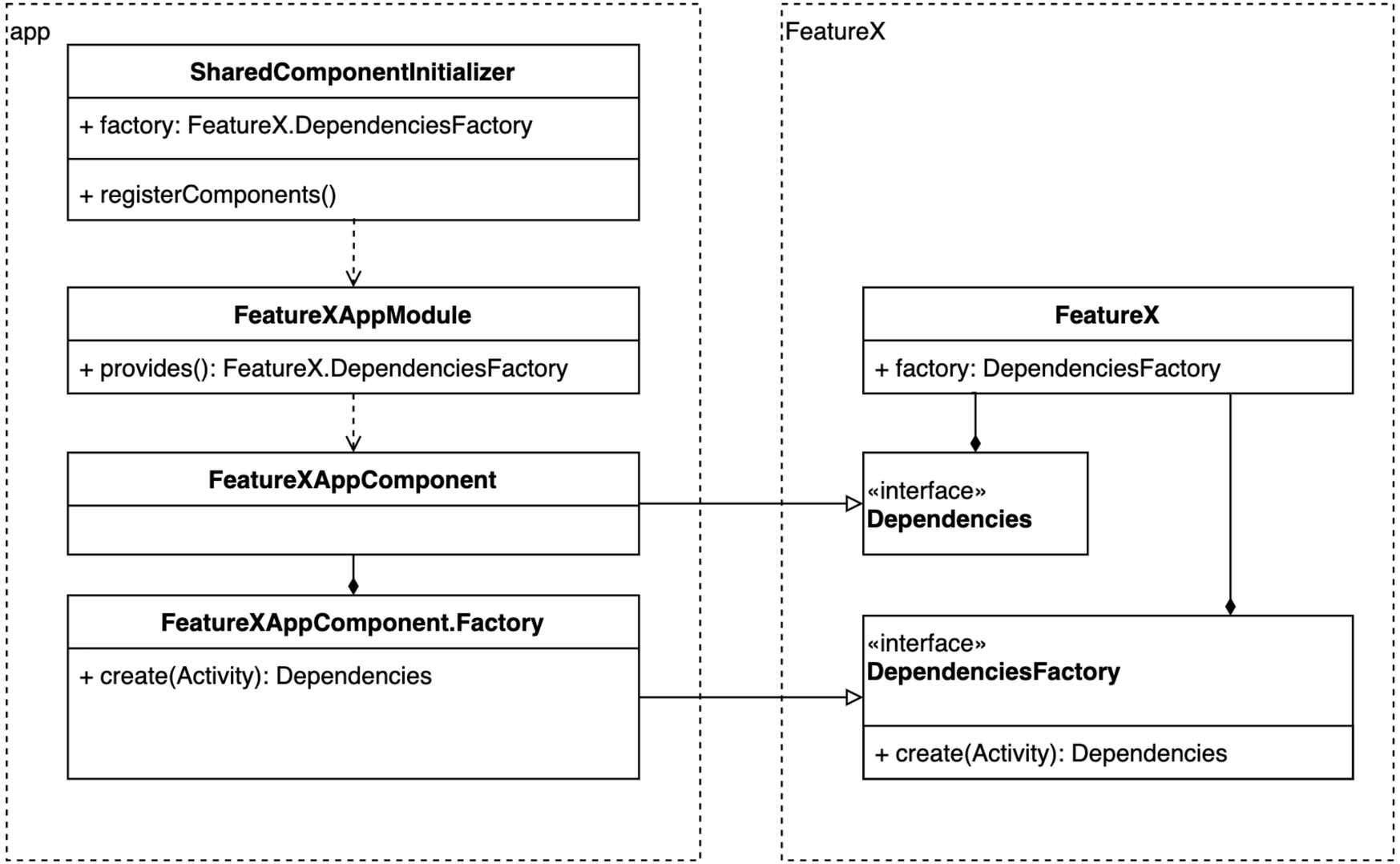

FeatureX
FeatureXComponent
+ create(Activity)
FeatureXActivity
+ createComponent()
FeatureX
«interface»
Dependencies

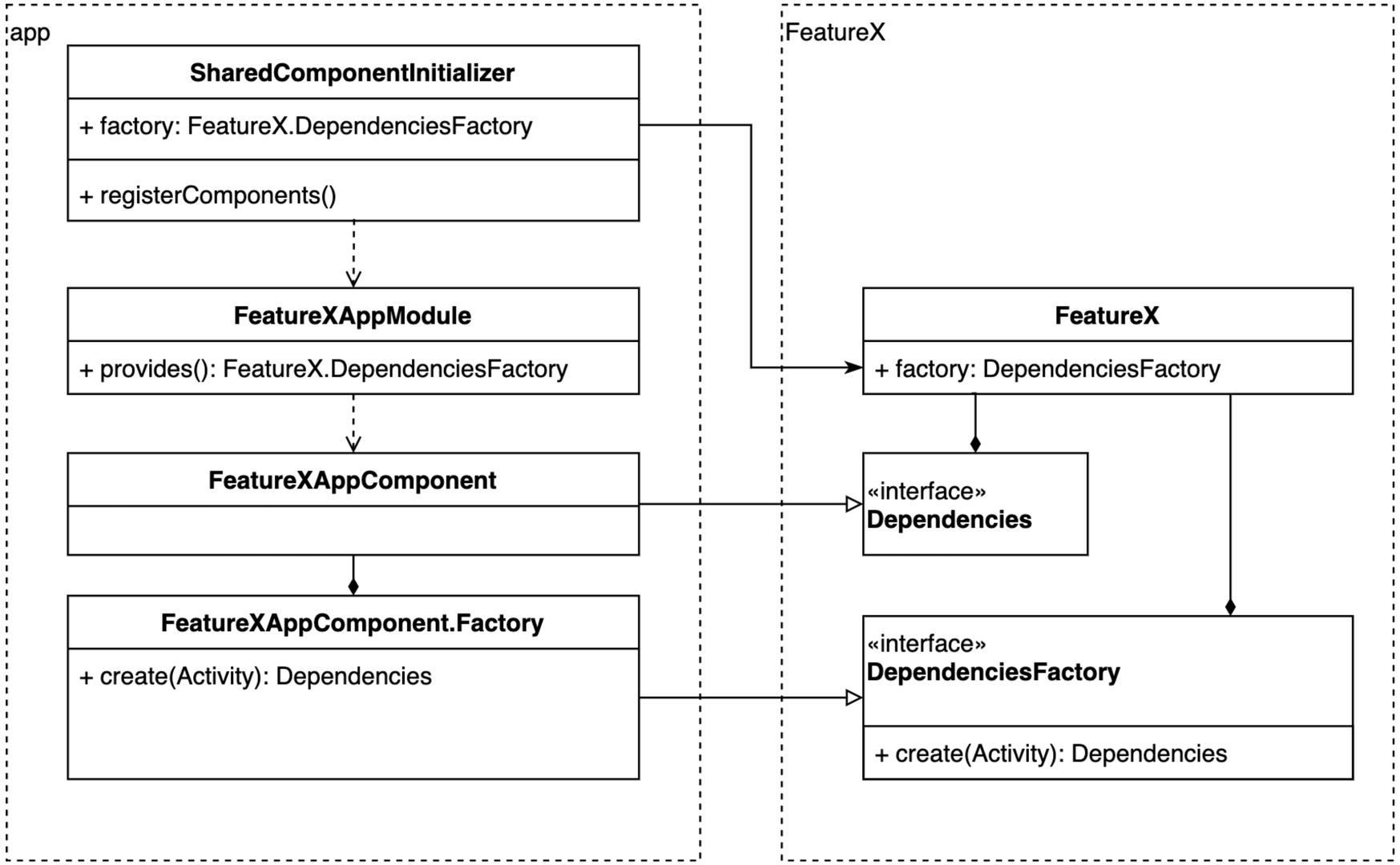
Особенности национальной модуляризации

```
return object : FeatureY.Dependencies {
    override val repository: FeatureYRepository get() = FeatureYRepositoryImpl()
    override val errorNavigator: ErrorNavigator get() = ErrorNavigatorImpl(activity)
    override val something: Something get() = SomethingImpl(context, mapper, holder)
    override val another: Another get() = AnotherImpl(activity, dependency)
    override val somebody: Somebody get() = SomebodyImpl(activity)
    override val anybody: Anybody get() = AnybodyImpl(repository, mapper)
    override val yolo: Yolo get() = YoloImpl(activity)
    override val puke: Puke get() = PukeImpl(activity)
}
```


FeatureX
FeatureXComponent
+ create(Activity, DependenciesFactory)
FeatureXActivity
+ createComponent()
FeatureX
+ factory: DependenciesFactory
«interface»
Dependencies
«interface»
DependenciesFactory
+ create(Activity): Dependencies


app
FeatureX
FeatureX
+ factory: DependenciesFactory
FeatureXAppComponent
«interface»
Dependencies
FeatureXAppComponent.Factory
+ create(Activity): Dependencies
«interface»
DependenciesFactory
+ create(Activity): Dependencies


app
SharedComponentInitializer
+ factory: FeatureX.DependenciesFactory
+ registerComponents()
FeatureXAppModule
+ provides(): FeatureX.DependenciesFactory
FeatureXAppComponent
FeatureXAppComponent.Factory
+ create(Activity): Dependencies
FeatureX
FeatureX
+ factory: DependenciesFactory
«interface»
Dependencies
«interface»
DependenciesFactory
+ create(Activity): Dependencies


app
SharedComponentInitializer
+ factory: FeatureX.DependenciesFactory
+ registerComponents()
FeatureXAppModule
+ provides(): FeatureX.DependenciesFactory
FeatureXAppComponent
FeatureXAppComponent.Factory
+ create(Activity): Dependencies
FeatureX
FeatureX
+ factory: DependenciesFactory
«interface»
Dependencies
«interface»
DependenciesFactory
+ create(Activity): Dependencies

Особенности национальной модуляризации

```
@Component(dependencies = [FeatureX.Dependencies::class])
interface FeatureXComponent {


    fun create(activity: Activity): FeatureXComponent {
        DaggerFeatureXComponent.factory()
            .create(activity, FeatureX.dependenciesFactory.create(activity))
    }
}
```

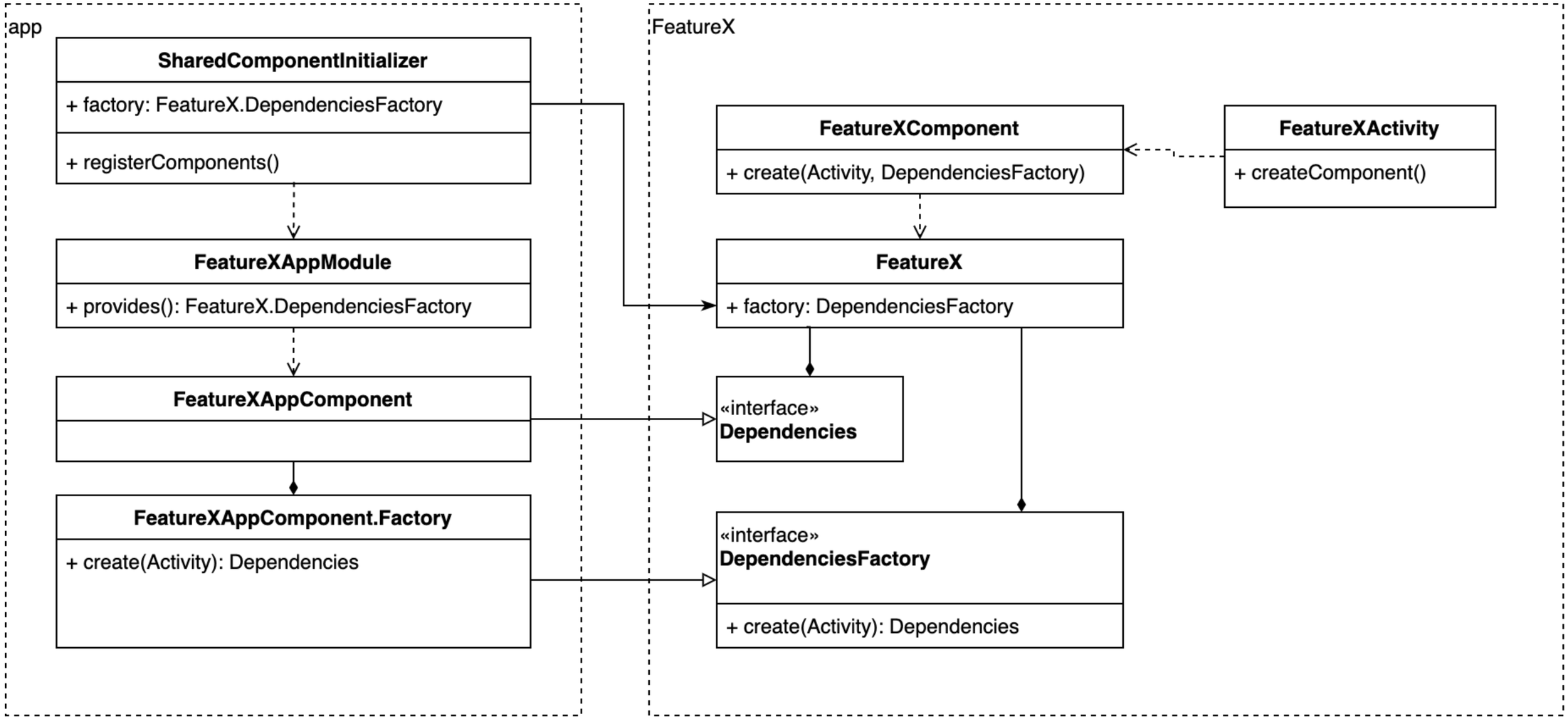
Особенности национальной модуляризации

```
@Component(modules = [.....])
interface AppFeatureXComponent : FeatureX.Dependencies {


    @Component.Factory
    interface Factory : FeatureX.DependenciesFactory {
        override fun create(
            @BindsInstance activity: Activity
        ): FeatureX.Dependencies
    }
}
```

Особенности национальной модуляризации

```
@Module
object FeatureXAppModule {
    @Provides
    fun provideFactory(): FeatureX.DependenciesFactory {
        return DaggerAppFeatureXComponent.factory()
    }
}
```


app
SharedComponentInitializer
+ factory: FeatureX.DependenciesFactory
+ registerComponents()
FeatureXAppModule
+ provides(): FeatureX.DependenciesFactory
FeatureXAppComponent
FeatureXAppComponent.Factory
+ create(Activity): Dependencies
FeatureX
FeatureXComponent
+ create(Activity, DependenciesFactory)
FeatureXActivity
+ createComponent()
FeatureX
+ factory: DependenciesFactory
«interface»
Dependencies
«interface»
DependenciesFactory
+ create(Activity): Dependencies

Перерыв 3 минуты

Чай, печеньки, вопросы?

Общий план

- ✓ С чего все началось
- ✓ Разделение проектов
- ✓ Соединение проектов обратно
- ✓ Шаринг модулей и фич

Соединение 2 проектов в одном АПК

Плавный перевод пользователей между проектами

Вижу цель не вижу препятствий

Проект Switch

2 проекта в одном APK

Переключение в рантайме по запросу

Плавный перевод пользователей

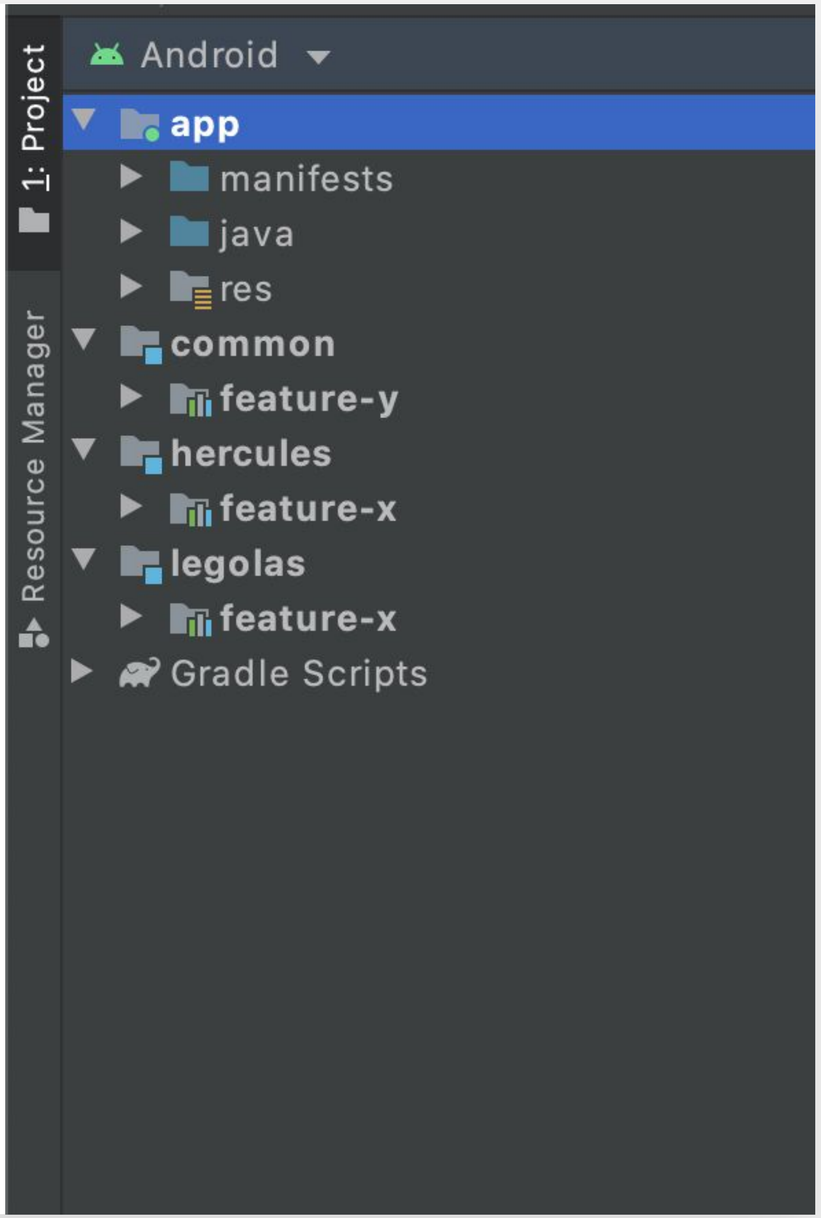
Мега монолит

2 разных проекта

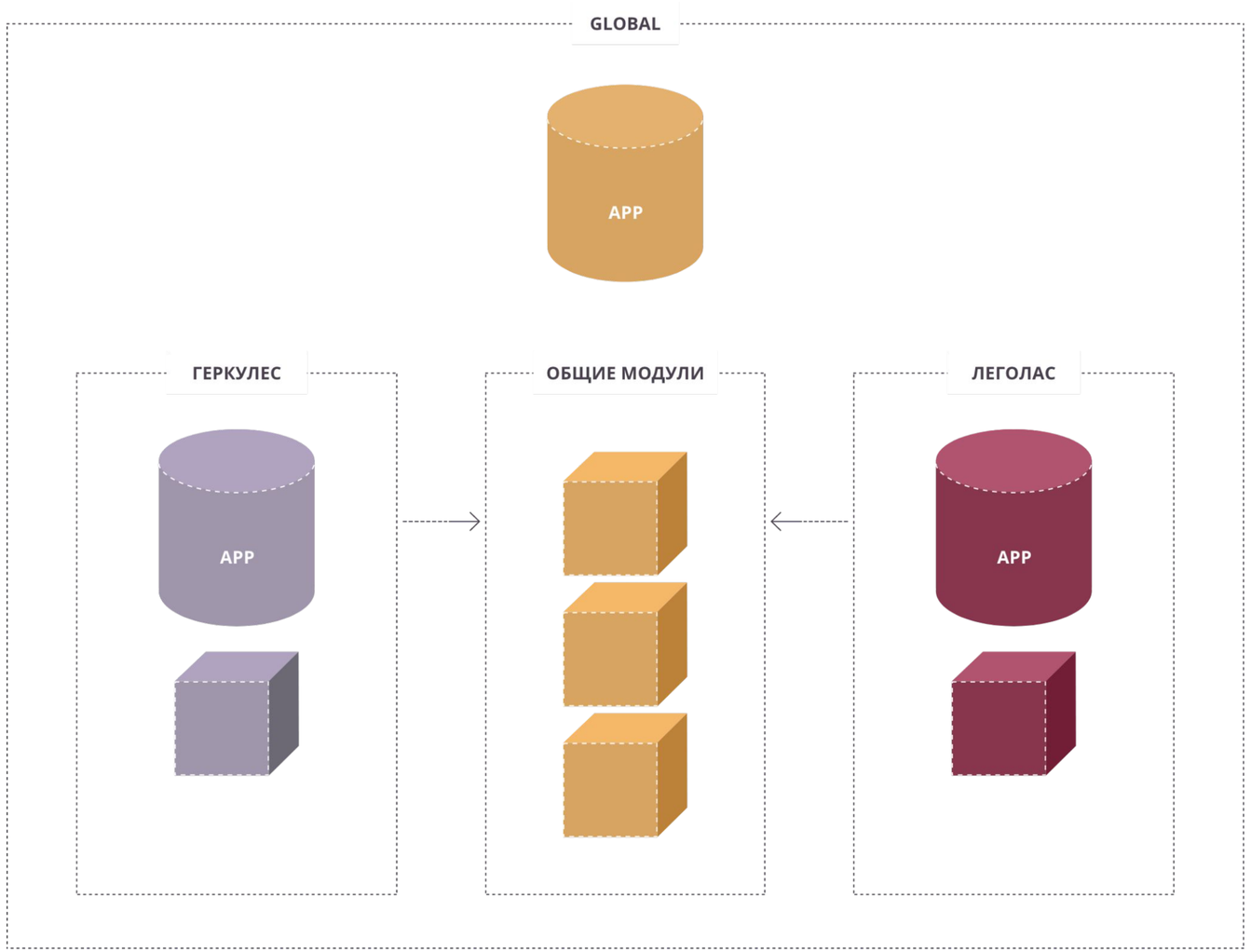
Совпадающие по названиям классы

Общие модули

Разная логика работы с сервером

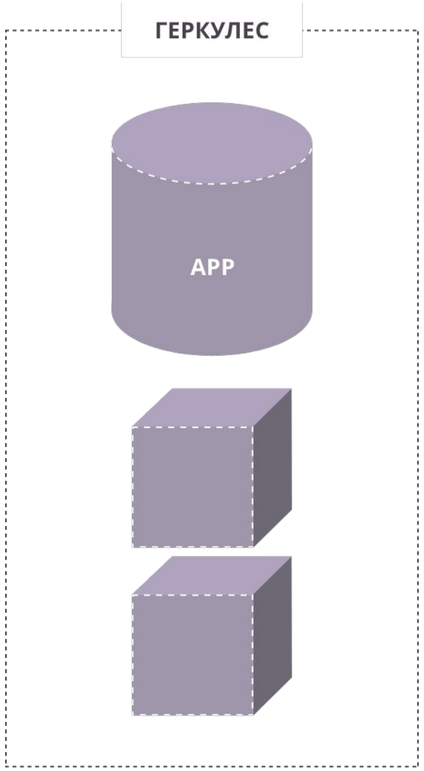


Змей Горыныч



Штаны превращаются в...

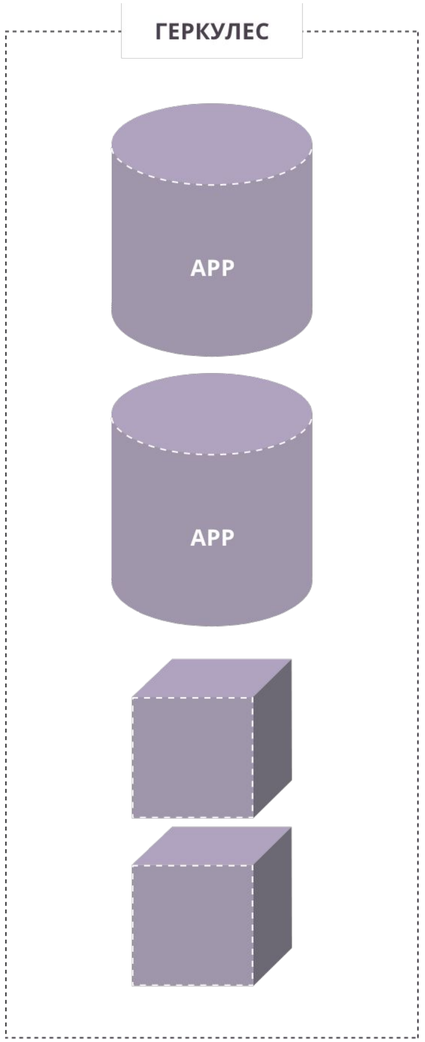
Создать новый app



Штаны превращаются в...

Создать новый app

Вынести логику инициализации



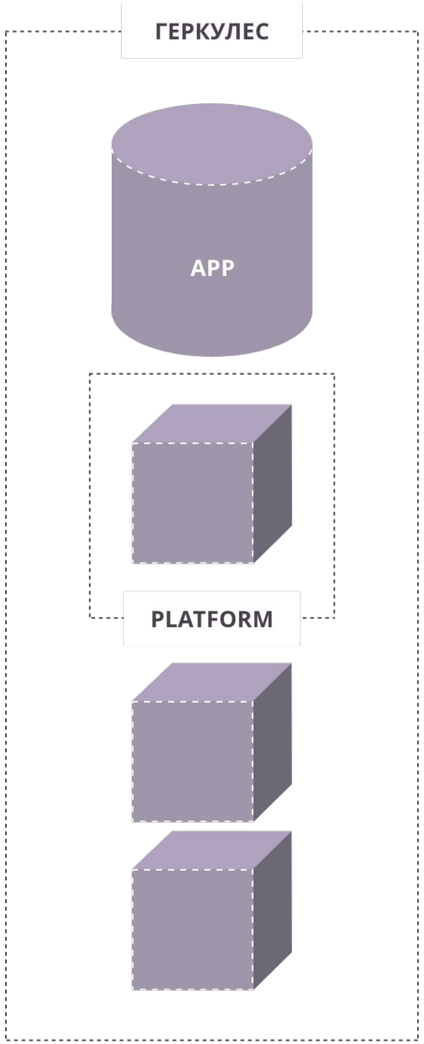
Штаны превращаются в...

Создать новый app

Вынести логику инициализации

Переименовать старый app

Сделать hercules-platform модуль библиотекой



Пакуем вещи

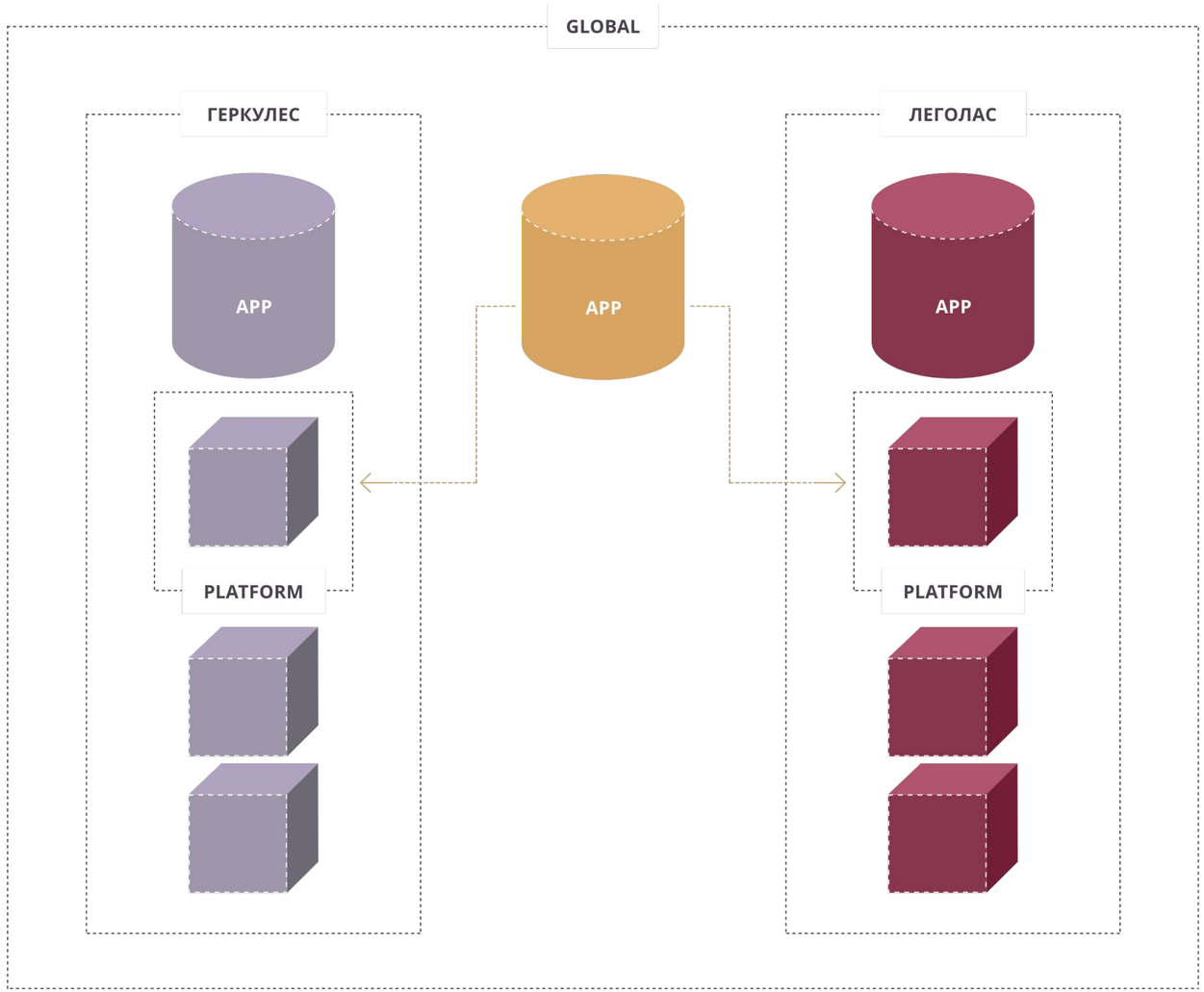
build.gradle

custom gradle конфиги

интеграционные тесты

конфигурации инструментов

google-services.json



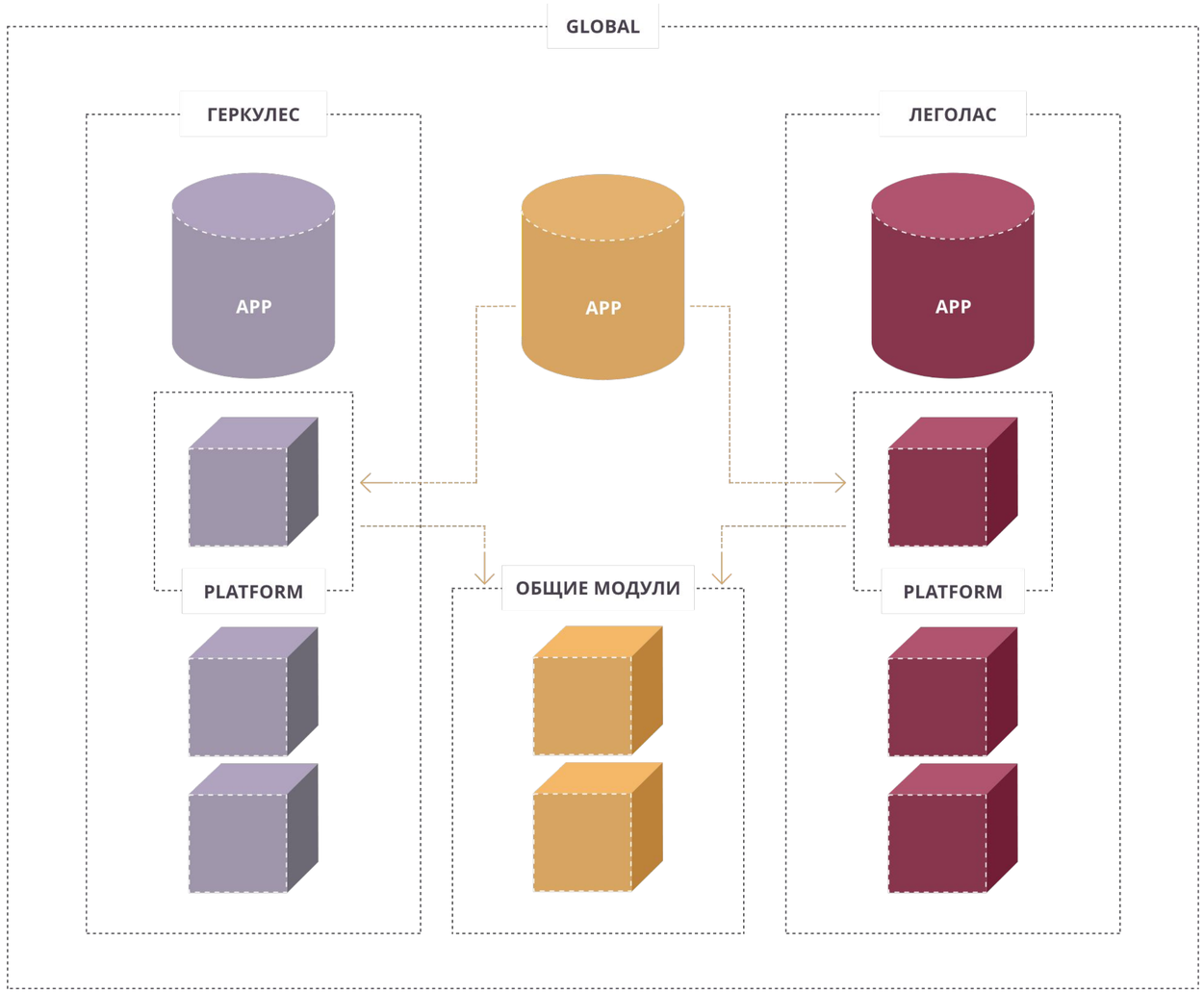
Со вкусом каши

```
flavorDimensions "distribution", "environment"
productFlavors {
    "pure-hercules" { dimension "distribution" }
    "xcombo" { dimension "distribution" }
    prod { dimension "environment" }
    qa { ... }
}
dependencies {
    implementation project(':hercules:hercules-platform')
    xcomboImplementation project(':legolas:legolas-platform')
}
```

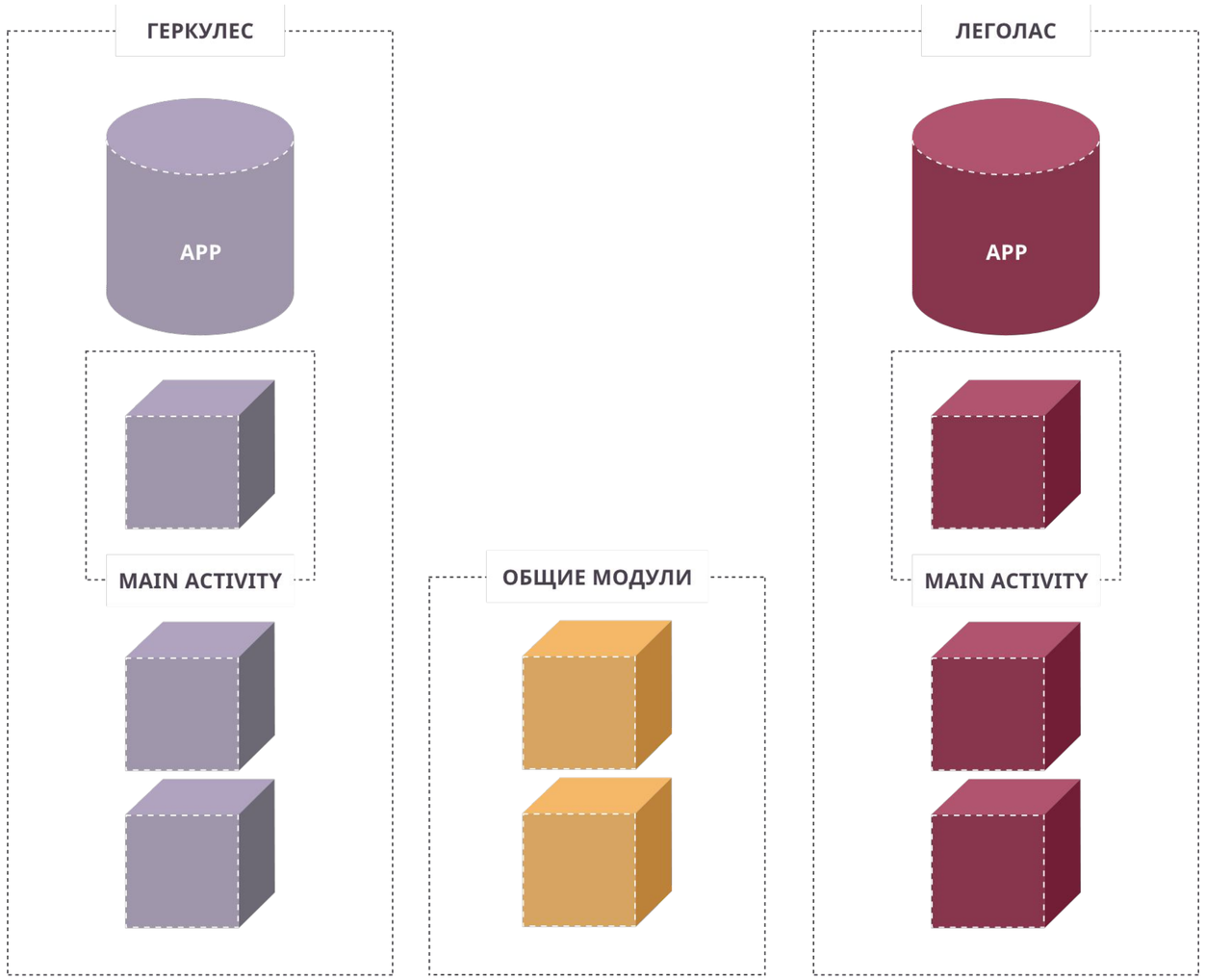
Сделай свой микс

```
flavorDimensions "distribution", "environment"
productFlavors {
    "pure-hercules" { dimension "distribution" }
    "xcombo" { dimension "distribution" }
    prod { dimension "environment" }
    qa { ... }
}
dependencies {
    implementation project(':hercules:hercules-platform')
    xcomboImplementation project(':legolas:legolas-platform')
}
```

Структура


GLOBAL
ГЕРКУЛЕС
APP
PLATFORM
ОБЩИЕ МОДУЛИ
APP
ЛЕГОЛАС
APP
PLATFORM

Структура


ГЕРКУЛЕС
APP
MAIN ACTIVITY
ОБЩИЕ МОДУЛИ
ЛЕГОЛАС
APP
MAIN ACTIVITY

Global app manifest

```
<activity
    android:name="com.company.legolas.MainActivity">
    <intent-filter tools:node="removeAll" />
</activity>

<activity
    android:name="com.company.hercules.MainActivity">
    <intent-filter tools:node="removeAll" />
</activity>
```

Global app manifest

```
<activity
    android:name=".switchover.launch.LauncherActivity"
    />
<activity-alias
    android:name=".Launcher"
    android:targetActivity=".switchover.launch.LauncherActivity"
    >
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
        <category android:name="android.intent.category.DEFAULT" />
    </intent-filter>
</activity-alias>
```

Переключатель

```
class AppInitalizer {
    fun onPreCreate(context: Context) {
        platformConfig = platformConfigFactory(context)
        when (platformConfig.getCurrentPlatform()) {
            Platform.HELCULES -> herculesInitializer.onPreCreate(context)
            Platform.LEGOLAS -> legolasInitializer.onPreCreate(context)
        }
    }
}
```

Переключатель

```
class AppInitalizer {
    fun onPreCreate(context: Context) {
        platformConfig = platformConfigFactory(context)
        when (platformConfig.getCurrentPlatform()) {
            Platform.HELCULES -> herculesInitializer.onPreCreate(context)
            Platform.LEGOLAS -> legolasInitializer.onPreCreate(context)
        }
    }
}
```

Общий план

- ✓ С чего все началось
- ✓ Разделение проектов
- ✓ Соединение проектов обратно
- ✓ Шаринг модулей и фич
- ✓ Соединение 2 проектов в одном АПК

Плавный перевод пользователей между проектами

Общая идея Switch

Хранилище платформы

Состояние платформ

Хлебные крошки

Включение платформы

Смена платформы по запросу

Внутренности переключателя

```
fun getCurrentPlatform(): Platform {
    var platform = platformConfigStorage.getCurrentPlatform()

    if (platform == null) {
        platform = firstLaunchPlatformResolver.getPlatform()
        platformConfigStorage.setCurrentPlatform(platform)
    }

    return platform
}
```

Внутренности переключателя

```
fun getCurrentPlatform(): Platform {
    var platform = platformConfigStorage.getCurrentPlatform()

    if (platform == null) {
        platform = firstLaunchPlatformResolver.getPlatform()
        platformConfigStorage.setCurrentPlatform(platform)
    }

    return platform
}
```

Как мы определяем платформу

```
fun getPlatform(): Platform {
        return resolvePlatformByComponentsState()
                ?: resolvePlatformByLegolasTrace()
                ?: FALLBACK_PLATFORM
    }
}
```

Как мы определяем платформу

```
fun getPlatform(): Platform {
        return resolvePlatformByComponentsState()
                ?: resolvePlatformByLegolasTrace()
                ?: FALLBACK_PLATFORM
    }
}
```

Как мы определяем платформу

```
fun getPlatform(): Platform {
        return resolvePlatformByComponentsState()
                ?: resolvePlatformByLegolasTrace()
                ?: FALLBACK_PLATFORM
    }
}
```

Как мы определяем платформу

```
fun getPlatform(): Platform {
        return resolvePlatformByComponentsState()
                ?: resolvePlatformByLegolasTrace()
                ?: FALLBACK_PLATFORM
    }
}
```

Хлебные крошки

```
private fun resolvePlatformByComponentsState(): Platform? {
        return when {
            isComponentEnabled(com.company.hercules.MainActivity::class.java) -> HERCULES
            isComponentEnabled(com.company.legolas.MainActivity::class.java) -> LEGOLAS
            else -> null
    }
}
private fun isComponentEnabled(clazz: Class<*>): Boolean {
    val state = packageManager.getComponentEnabledSetting(ComponentName(context, clazz))
    return state != PackageManager.COMPONENT_ENABLED_STATE_DISABLED
}
```

По следам компонентов

```
private fun resolvePlatformByComponentsState(): Platform? {
        return when {
            isComponentEnabled(com.company.hercules.MainActivity::class.java) -> HERCULES
            isComponentEnabled(com.company.legolas.MainActivity::class.java) -> LEGOLAS
            else -> null
    }
}
private fun isComponentEnabled(clazz: Class<*>): Boolean {
    val state = packageManager.getComponentEnabledSetting(ComponentName(context, clazz))
    return state != PackageManager.COMPONENT_ENABLED_STATE_DISABLED
}
```

Общая идея Switch

- ✓ Хранилище платформы
- ✓ Состояние платформ
- ✓ Хлебные крошки

Включение платформы

Смена платформы по запросу

Вкл/выкл

```
class FirstLaunchPlatformComponentsStateChecker : SilentContentProvider() {
    override fun onCreate(): Boolean {
        if (isFirstLaunch()) {
            setWasLaunched()
            val currentPlatform = platformConfig.getCurrentPlatform()
            platformComponentsActivator.activatePlatform(currentPlatform)
        }
        disableSelf()
    }
}
```

Вкл/выкл

```
class FirstLaunchPlatformComponentsStateChecker : SilentContentProvider() {
    override fun onCreate(): Boolean {
        if (isFirstLaunch()) {
            setWasLaunched()
            val currentPlatform = platformConfig.getCurrentPlatform()
            platformComponentsActivator.activatePlatform(currentPlatform)
        }
        disableSelf()
    }
}
```

Вкл/выкл

```
class FirstLaunchPlatformComponentsStateChecker : SilentContentProvider() {
    override fun onCreate(): Boolean {
        if (isFirstLaunch()) {
            setWasLaunched()
            val currentPlatform = platformConfig.getCurrentPlatform()
            platformComponentsActivator.activatePlatform(currentPlatform)
        }
        disableSelf()
    }
}
```

Вкл/выкл

```
class FirstLaunchPlatformComponentsStateChecker : SilentContentProvider() {
    override fun onCreate(): Boolean {
        if (isFirstLaunch()) {
            setWasLaunched()
            val currentPlatform = platformConfig.getCurrentPlatform()
            platformComponentsActivator.activatePlatform(currentPlatform)
        }
        disableSelf()
    }
}
```

Включите свет

```
class PlatformComponentsActivator(...) {
    fun activatePlatform(platform: Platform) {
        val appPackageInfo: PackageInfo = currentAppPackageInfoFetcher.getInfo()
        val components: List<Component> = selectAllComponents(appPackageInfo)
        updateComponentsAvailability(platform, components)
    }
}
```

Включите свет

```
class PlatformComponentsActivator(...) {
    fun activatePlatform(platform: Platform) {
        val appPackageInfo: PackageInfo = currentAppPackageInfoFetcher.getInfo()
        val components: List<Component> = selectAllComponents(appPackageInfo)
        updateComponentsAvailability(platform, components)
    }
}
```

Переключение компонента

```
val newState = getComponentState(component, activePackageRoot, inactivePackageRoot)
if (newState != null) {
    packageManager.setComponentEnabledSetting(
        ComponentName(it.packageName, it.name),
        newState,
        PackageManager.DONT_KILL_APP)
}
```

Переключение компонента

```
val newState = getComponentState(component, activePackageRoot, inactivePackageRoot)
if (newState != null) {
    packageManager.setComponentEnabledSetting(
        ComponentName(component.packageName, component.name),
        newState,
        PackageManager.DONT_KILL_APP)
}
```

Общая идея Switch

- ✓ Хранилище платформы
- ✓ Состояние платформ
- ✓ Хлебные крошки
- ✓ Включение платформы
- **Смена платформы по запросу**

Easy peasy lemon squeezy

Перезапустить приложение в новом выделенном процессе

ProcessPhoenix обрабатывает перезапуск

Переключить платформу

Перезапустить приложение в основном процессе

https://github.com/JakeWharton/ProcessPhoenix

Общая идея Switch

- ✓ Хранилище платформы
- ✓ Состояние платформ
- ✓ Хлебные крошки
- ✓ Включение платформы
- ✓ Смена платформы по запросу

Итоги

Вызовы

Проблемы

Текущее состояние

Планы

Выводы

Вызовы: тесты и CI

Тесты в общих модулях гоняются 2 раза

Изменения в общих модулях задевают 2 проекта

Нужно определять где были изменения и что запускать

Вызовы: публикация проекта

Плагин triplet play для автоматизации

Разные конфигурации для разных стран

APK для одной страны 11-13 MB

Мега монолит 16 MB

https://github.com/Triple-T/gradle-play-publisher

Проблемы: релизы

Отдельные релизы в монорепе - боль

Единый релиз в разы удобнее

Проблемы: оценки

Проект стартовал примерно весной 2018

Планировали MVP за несколько месяцев

Полноценный релиз зима 2021

Погрешность в оценке ~1200%

Проблемы: оценки

Стори поинты

Майки

Бизнес говорит на другом языке

Текущее состояние: подушка безопасности

Переход планируется на бекенде

Юзер оповещается о том, что сейчас произойдет

Куча проверок и осторожности

Если что это можно откатить обратно

Текущее состояние: состояние команд

Передали фидбек до высшего руководства

Фидбек собирают со всех

Больше изоляции

Сами себе ромашки

Планы: Что дальше?

Завершить процесс миграции

Полностью выключить Леголас

Выводы: технические

Геркулес содержит много хардкода для конкретной страны

Многие флоу были полностью переделаны в Леголасе

Проще добавить страну в Леголас, чем в Геркулес

Геркулес оправдал себя на бекенде

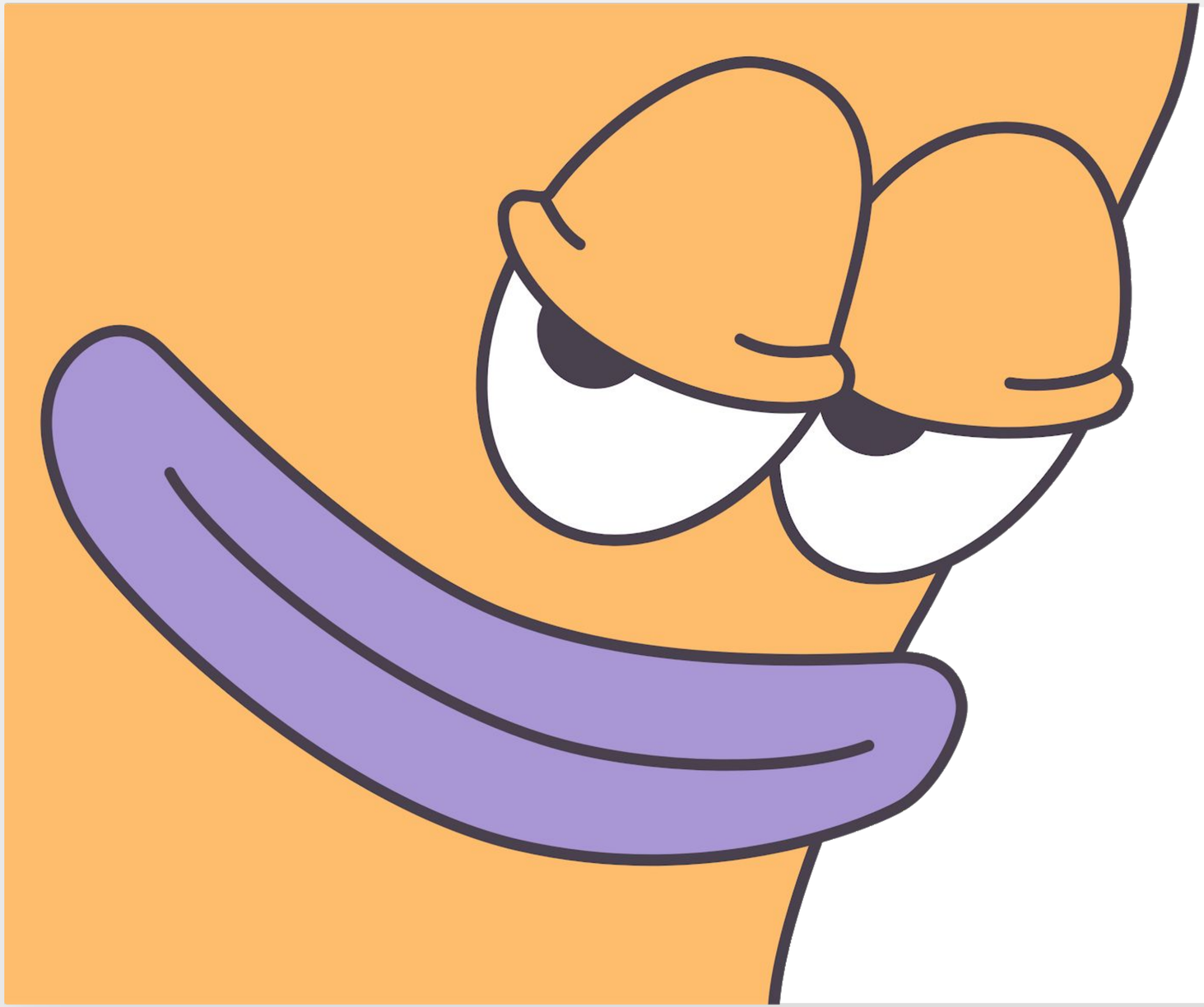
Выводы: командные

Изоляция команд вредит проекту

Нельзя изолировать команды с конфликтами

Новичков нужно онбоардить

Это не всегда очевидно в моменте



Агейченко Александр

Android Teamlead @ Distillery

wackalooon@gmail.com

@Wackaloon