UI тесты за 21 день

Докладчики:

Алексей Хайминов
Алексей Червяков

юла
проект

Лёша Хайминов
из Платформы

Лёша Червяков
из Монетизации

юла
проект

Кому пригодится доклад?

Новым молодым стартапам/проектам

Проектам «с историей», но без ui-тестов

юла
проект

Disclaimer

Подойдёт, если непонятно:

- Профит от внедрения
- Какая инфраструктура нужна для первой версии
- Что именно придётся переделать в проекте
- Как найти время в команде
- Как договориться с бизнесом о ресурсах

Не подойдёт, если нет:

Разработчиков / Денег / Лишних часов в сутках / ...

Результаты

- 97% avg success rate
- MVP Инфраструктуры
- +40 тестовых сценариев при сохранении графика фич
- Прогон тестов каждый Merge Request

юла
проект

Цель

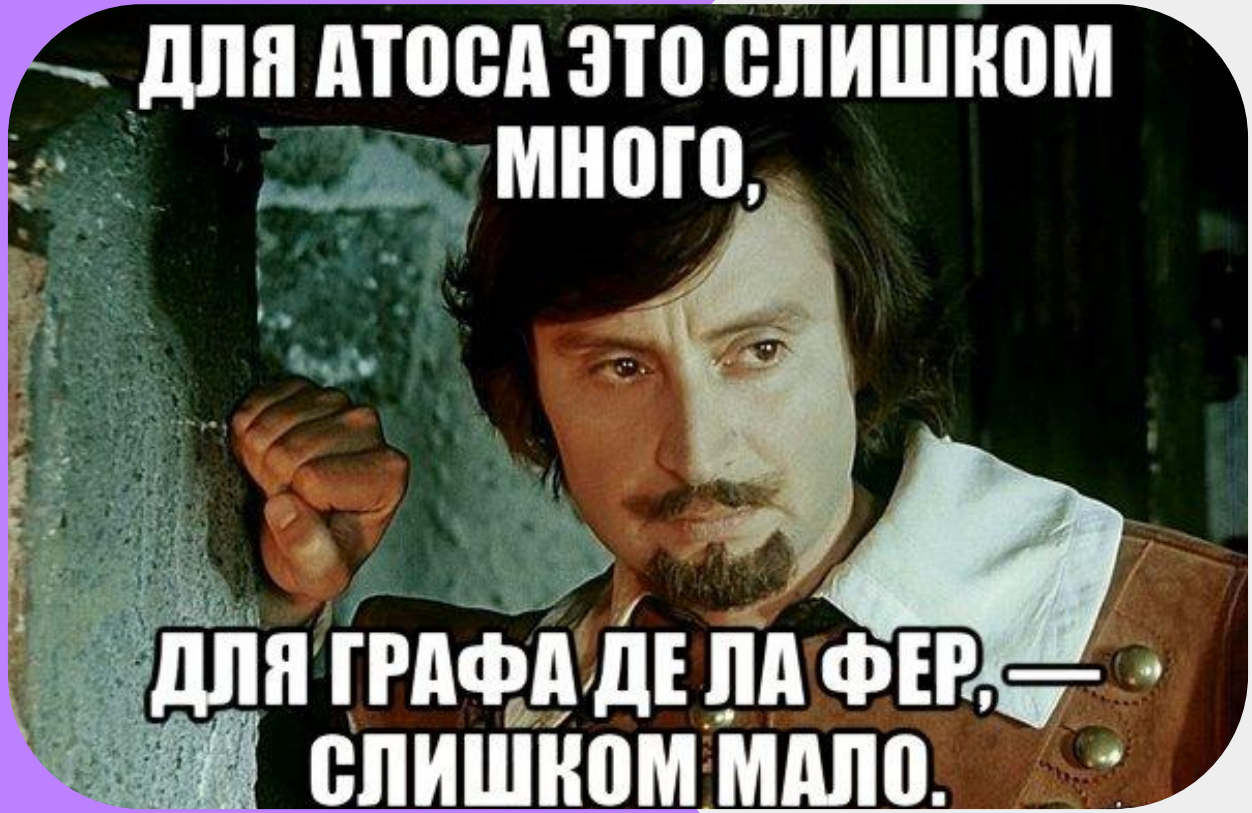
Гайд, после которого будет понятно!

юла
проект

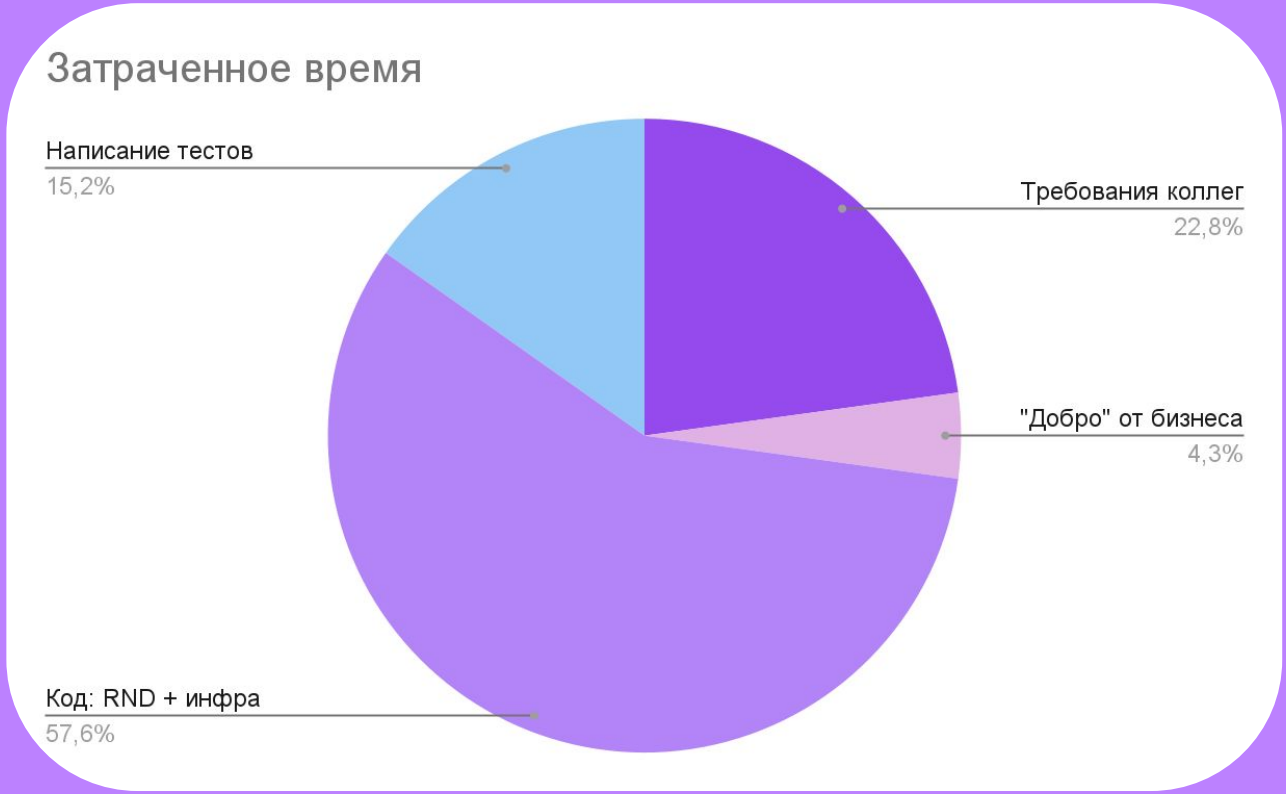
Фокус на идеи

Реализации – пример

«21 день» – это много или мало?



Почему «21 день»?



Цель

Гайд, после которого будет понятно!

Чтобы у вас получилось намного быстрее

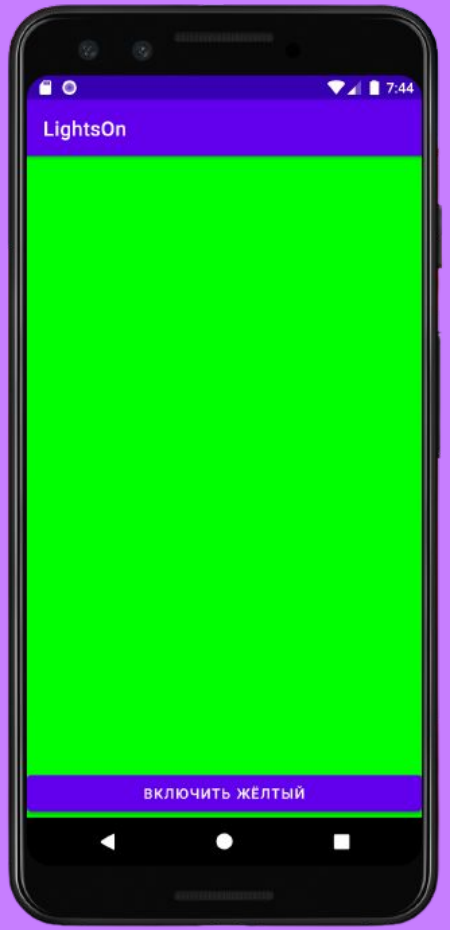
юла
проект

Как правильно готовить тесты?

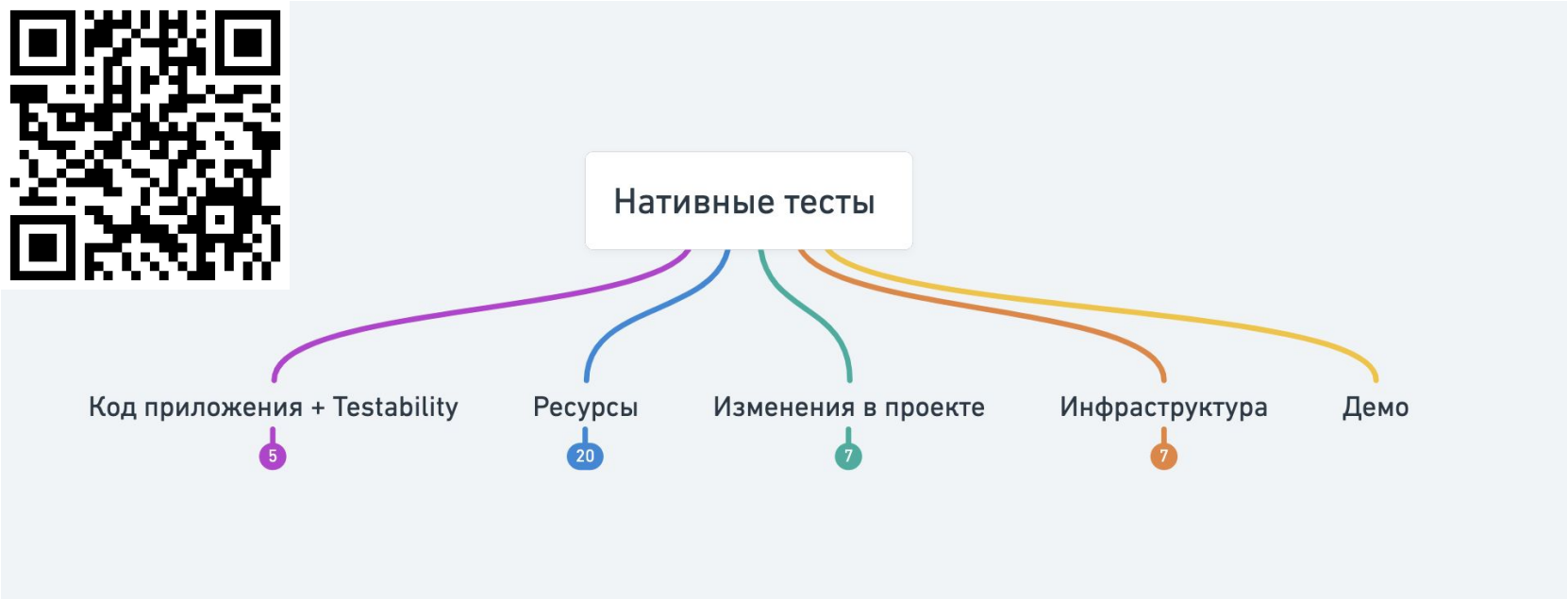
Расскажет
Алексей Тюрин

Демо-проект

Демо-проект



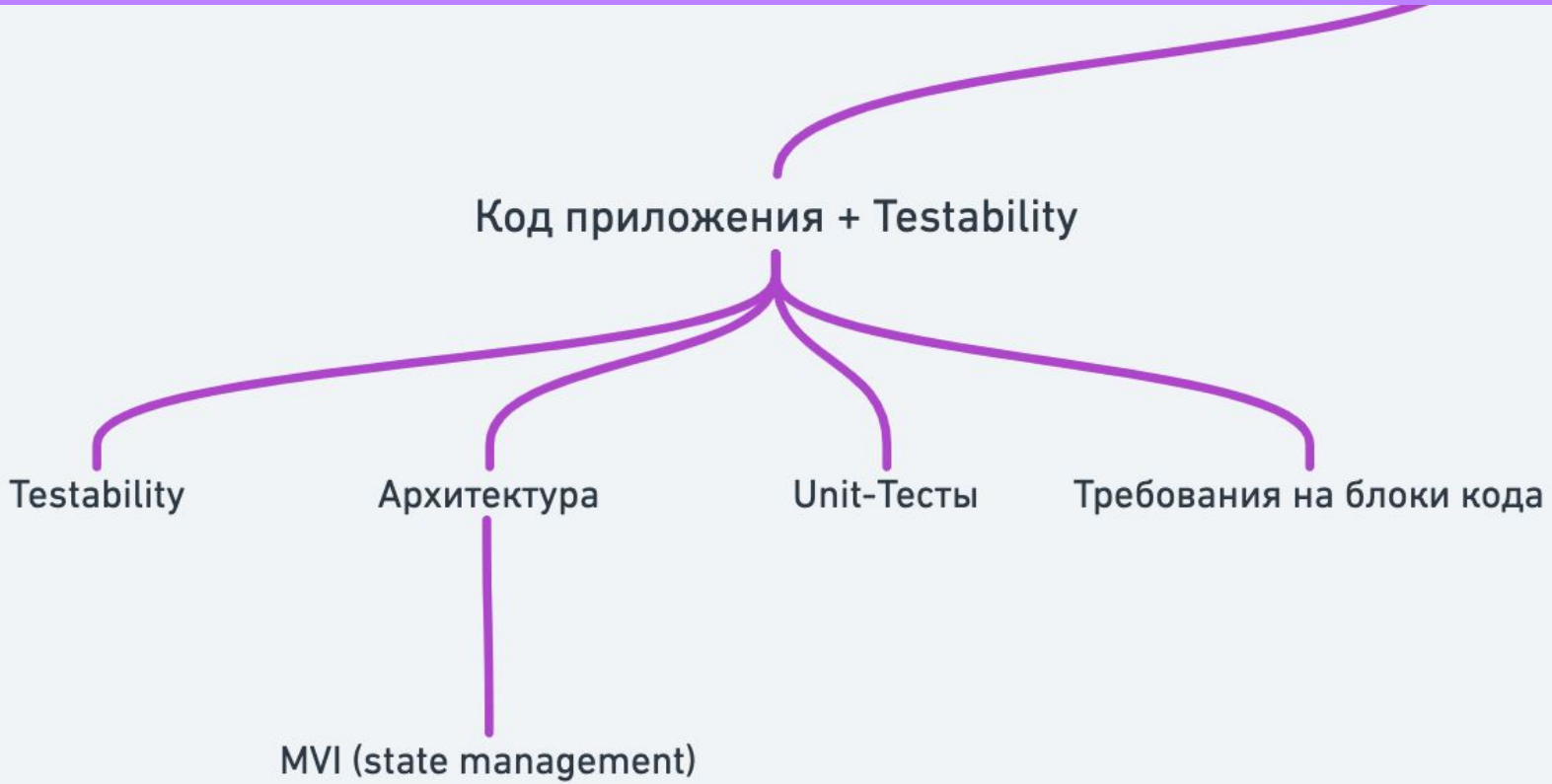
Расскажем, как подготовиться



юла
проект

Код приложения / Testability

юла
проект


Код приложения + Testability
Testability
Архитектура
Unit-Тесты
Требования на блоки кода
MVI (state management)

юла
проект

Вопросы

- Можно ли обойтись без ui-тестов?
- Зачем нужны тесты?

юла
проект

Testability

Software testability is the degree to which a software artifact <..> supports testing in a given test context.

If the testability of the software artifact is high, then finding faults in the system (if it has any) by means of testing is easier.

юла
проект

Нет тестов - пишем код, влияем на:

1. **Controllability:** The degree to which it is possible to control the state of the component under test (CUT) as required for testing.
2. **Isolateability:** The degree to which the component under test (CUT) can be tested in isolation.
3. **Separation of concerns:** The degree to which the component under test has a single, well defined responsibility.
4. **Understandability:** The degree to which the component under test is documented or self-explaining.

юла
проект

Непосредственно с написанием тестов:

5. **Observability:** The degree to which it is possible to observe (intermediate and final) test results.

6. **Automatability:** The degree to which it is possible to automate testing of the component under test.

7. **Heterogeneity:** The degree to which the use of diverse technologies requires to use diverse test methods and tools in parallel.

юла
проект

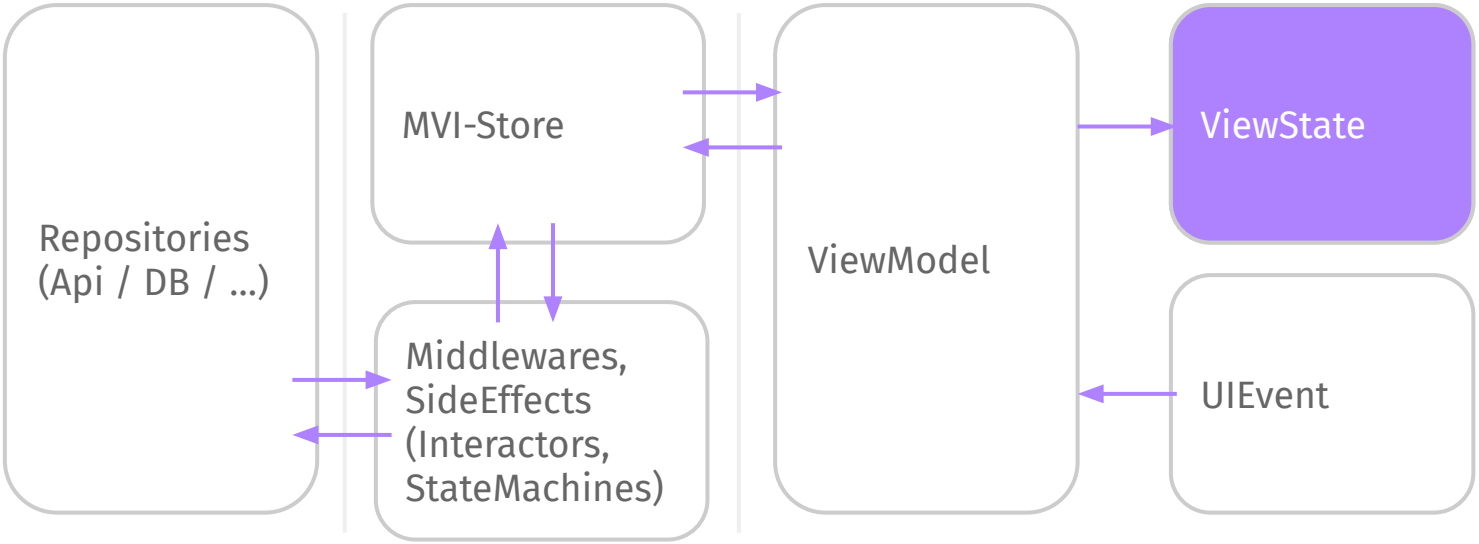
На что влияет архитектура?

- Controllability
- Isolatability
- Separation of concerns
- Understandability*

Пример:
Clean+ MVI

Архитектура: Clean + MVI + UDF

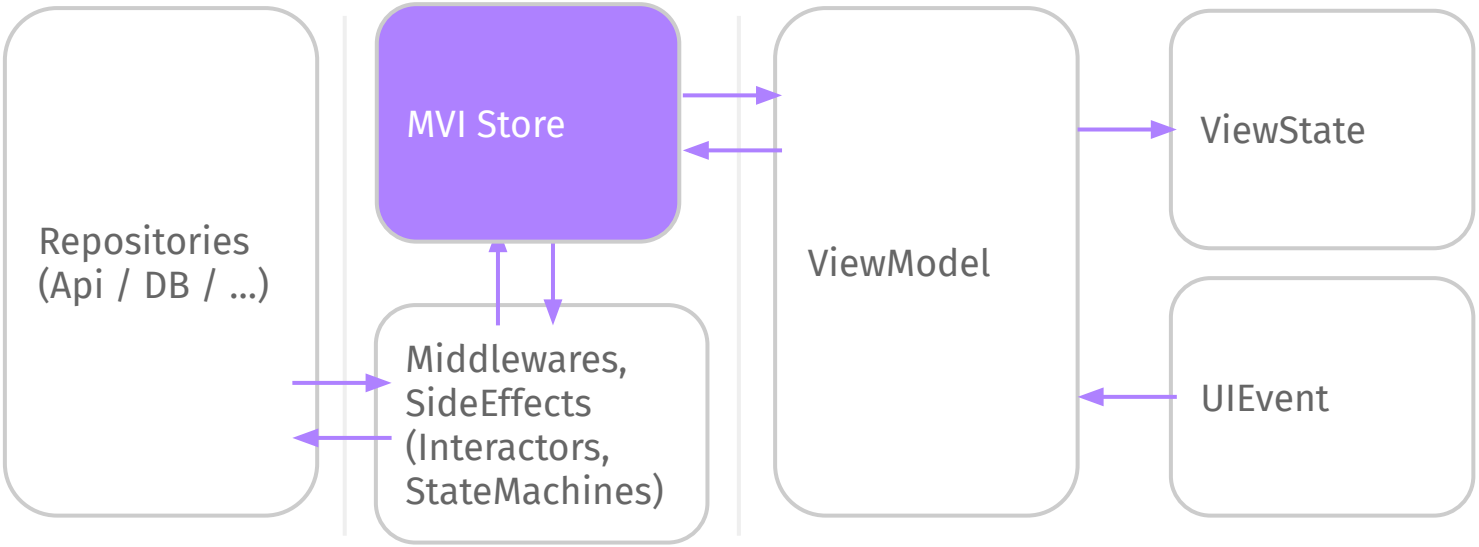
- Controllability
- Isolateability
- Separation of concerns
- Understandability*



юла
проект

Архитектура: Clean + MVI + UDF

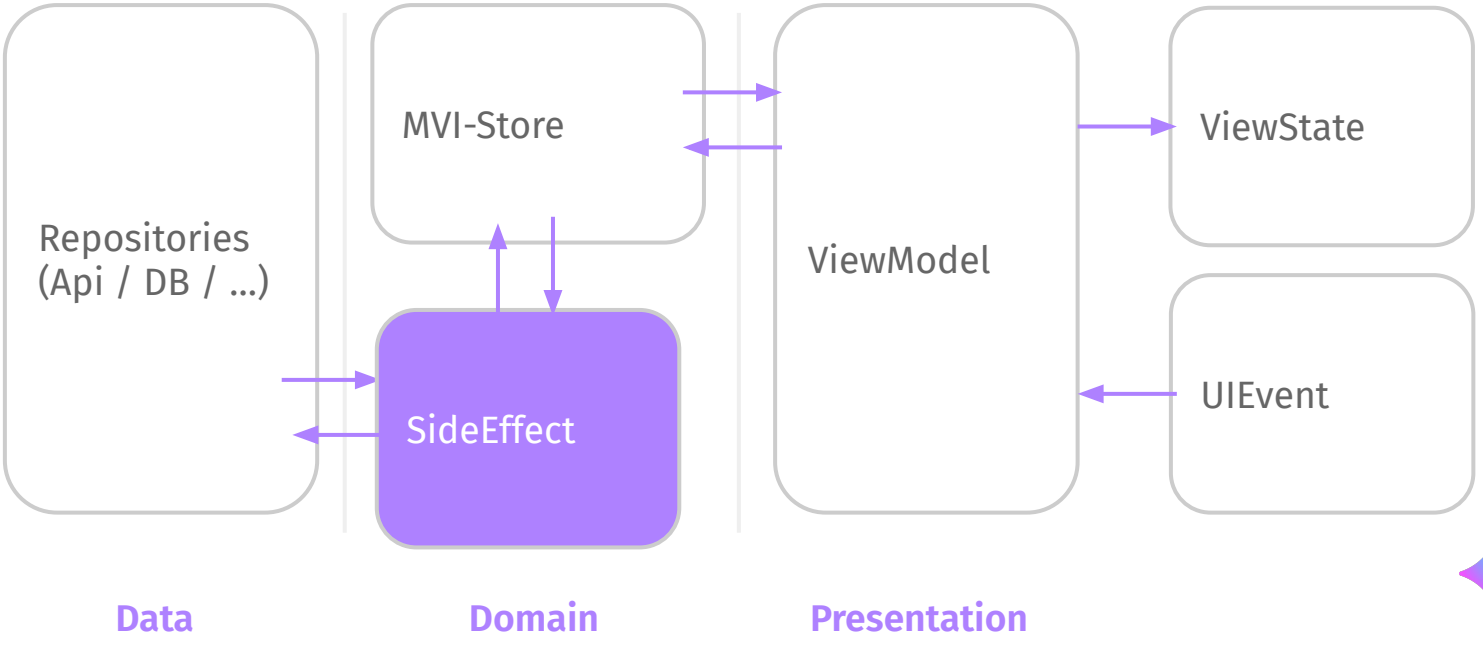
- Controllability
- Isolateability
- Separation of concerns



юла
проект

Архитектура: Clean + MVI + UDF

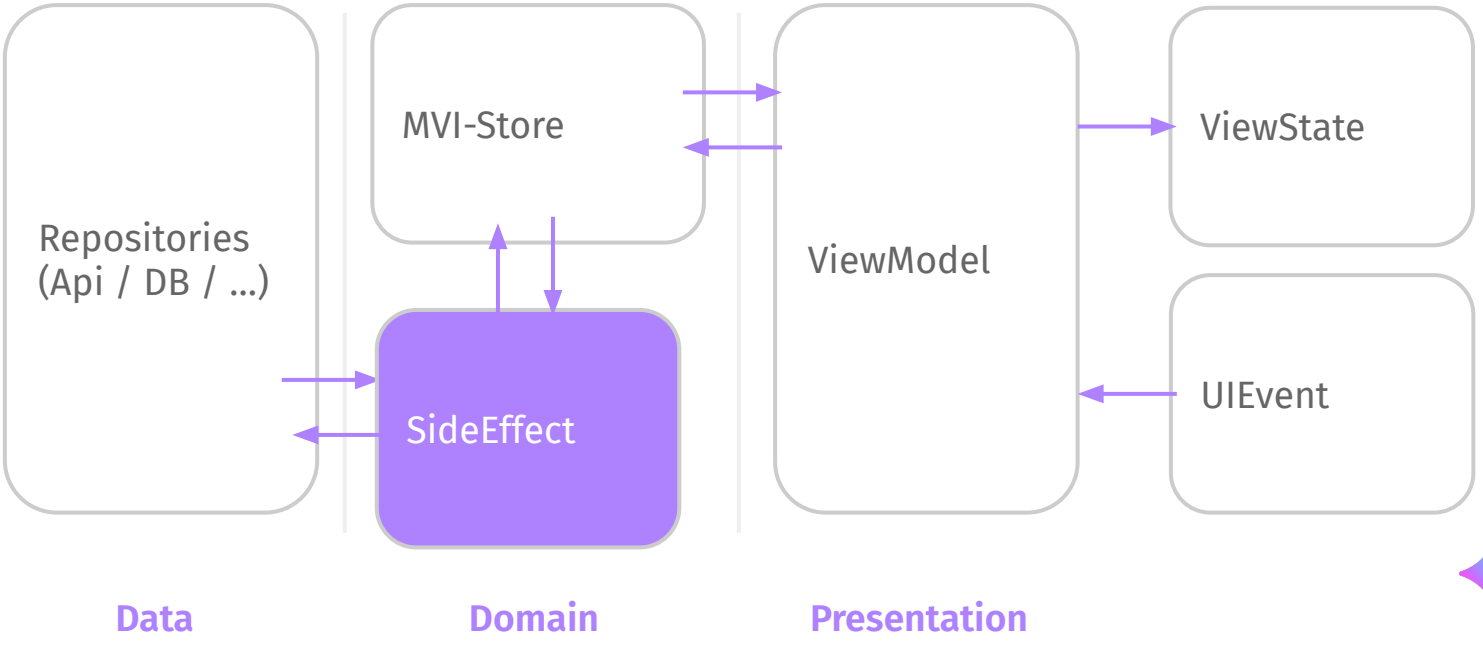
- Isolateability
- Separation of concerns



юла
проект

Архитектура: Clean + MVI + UDF

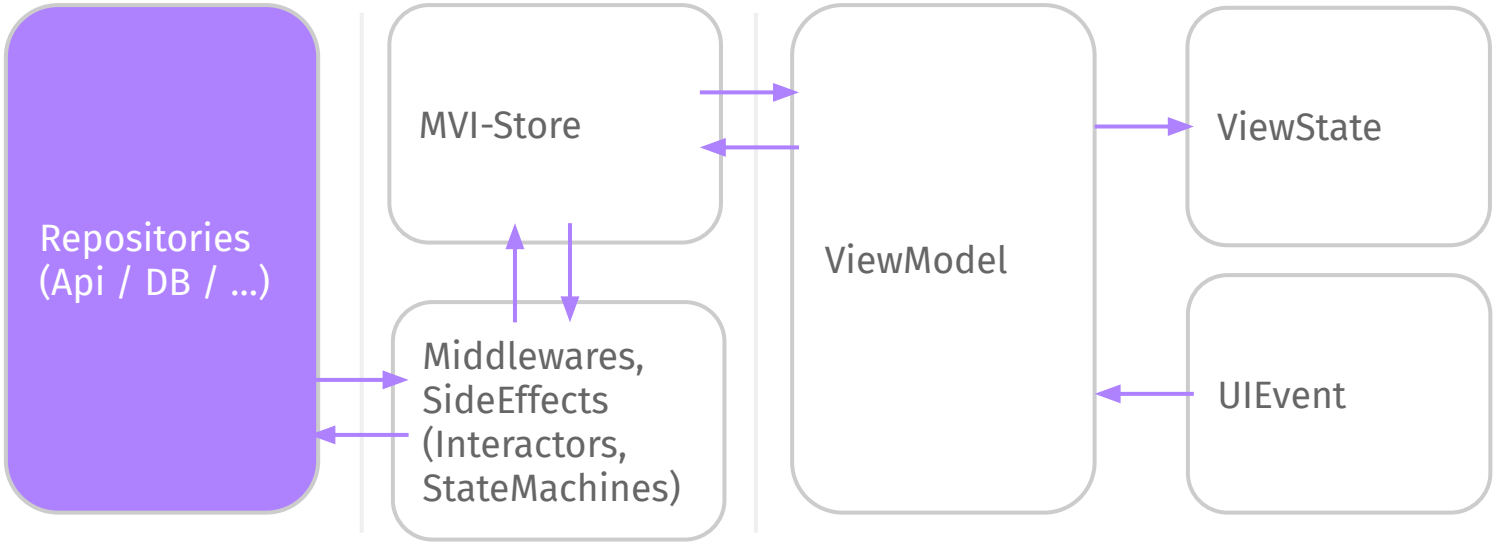
- Isolateability
- Separation of concerns



юла
проект

Архитектура: Clean + MVI + UDF

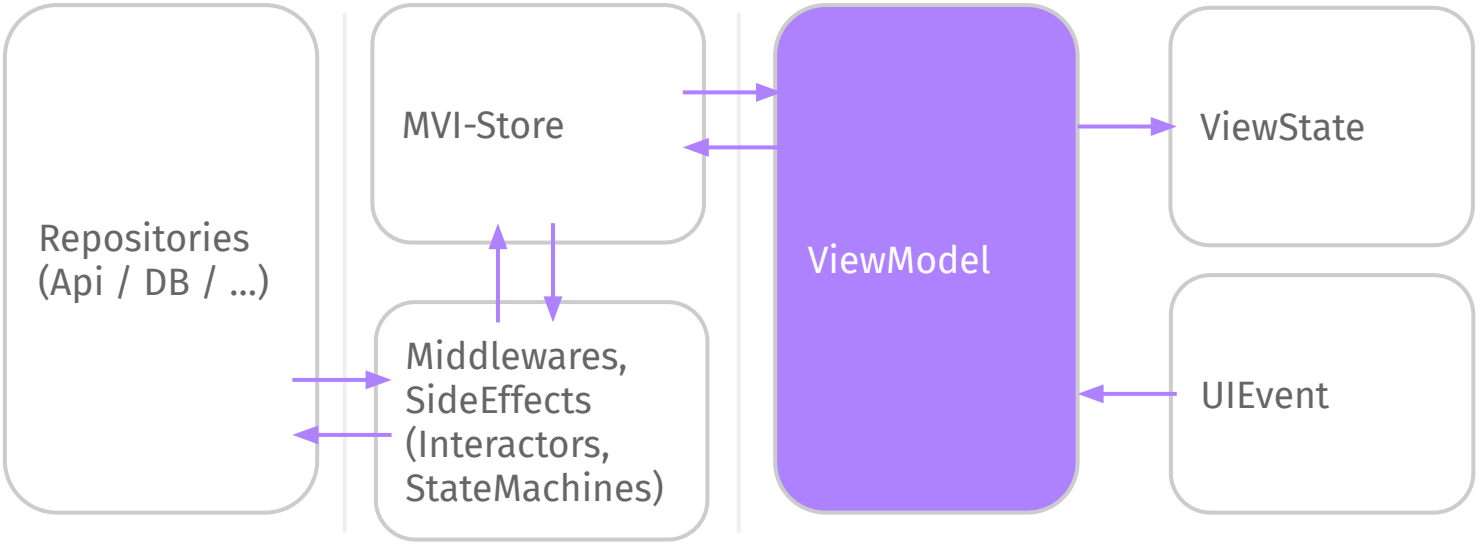
- Controllability*
- Isolateability
- Separation of concerns



юла
проект

Архитектура: Clean + MVI + UDF

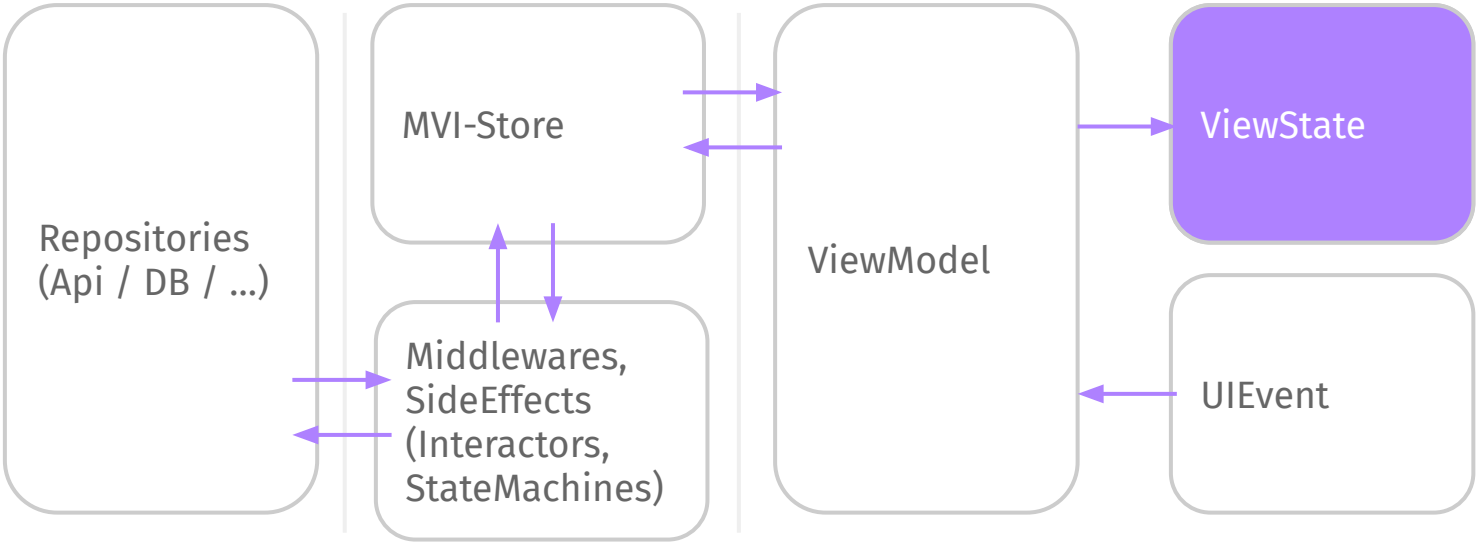
- Controllability
- Isolateability
- Separation of concerns



юла
проект

Архитектура: Clean + MVI + UDF

- Controllability
- Isolateability
- Separation of concerns
- Understandability*



юла
проект

Как писать код?

Явные контракты

square(x: Int) = x * x

- Определены типы входных параметров и результата
- Легко сделать Unit-тест
- Нет побочных эффектов

юла
проект

MVP (Presenter)

```
@Test
fun `accept create event, get user return error EXPECT error state`() {
  // ...  init ... //
  presenter.create()
  verify { paymentSettingsInteractor.getUser() }
  // ... ... //
}
```

юла
проект

MVP (Presenter)

- Controllability
- Isolateability

```
@Test
fun `accept create event, get user return error EXPECT error state`() {
  // ...  init ... //
  presenter.create()
  verify { paymentSettingsInteractor.getUser() }
  // ... ... //
}
```

юла
проект

MVI (ViewModel)

- Controllability
- Isolateability

```
@Test
fun `accept create event, get user return error EXPECT error state`() {
  // ...  init ... //
  val states : TestSubscriber<ViewState> = viewModel.viewStates.test()
  viewModel.consume(UiEvent.Create())
  verify { states.assertLast(expectedErrorState) }
}
```

юла
проект

Стейт-машины и тесты

Алексей Червяков

State Machine

```
@Test
fun `given state PaidPromotion, on Success should result in RequestPopupData`() {
  assertTransition(
    initial = State.PaidPromotion(paymentMethod = PaymentMethod.PHONE),
    event = Event.Success.PromotionPaid,
    expected = State.RequestPopupData(ID_WAITING_PAYMENT_POPUP)
  )
}
```

юла
проект

Как писать код?

Пассивный ui

(отрисовка/простановка state’a)

Как писать код?

- Separation of concerns

```
disposables += viewModel.viewStates
  .observeOn(schedulersFactory.main)
  .subscribe { state ->
      state.doIfForm { showForm(it) }
  }


private fun showForm(items: List<AdapterItem>) {
  adapter.items = items
}
```

юла
проект

Unit VS UI?

Когда не хватает ui-тестов?

MVI (ViewModel)

```
@Test
fun `view model search screen`() {
  // ...   event ... //
  val states : TestSubscriber<ViewState> = viewModel.viewStates.test()
  val items : List<AdapterItem> = states.last().items
 //????
   if (items[0] instanceof Advert) {
        (items[0] as Advert).name().assertEquals(expectedName)
   }
}
```

юла
проект

MVI (ViewModel)

```
@Test
fun `kasspresso Search Screen`() {
    ///..... ..///
    run{
      SearchScreen {
          searchRecycler {
             itemWithText(categoryName).isVisible()
          }
      }
    }
}
```

юла
проект

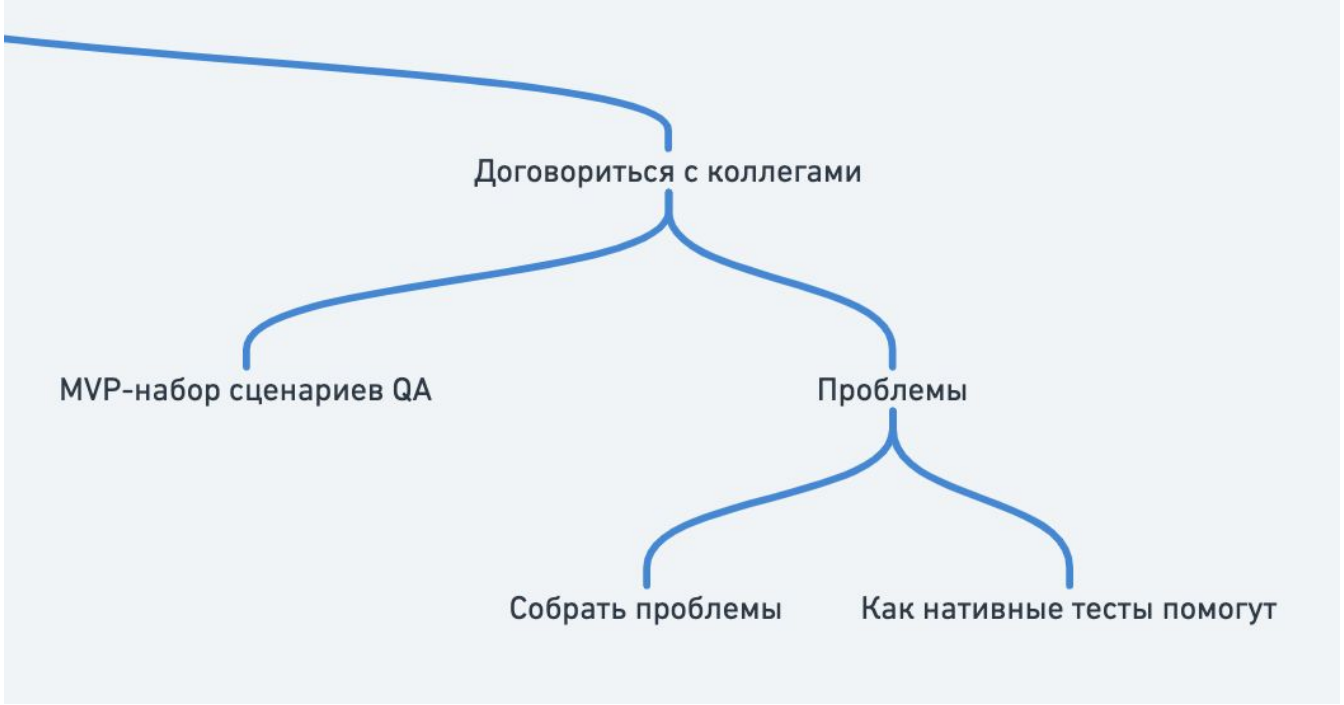
Код приложения, итоги:

- Тесты - средство, которое помогает **ПРОЩЕ** и **БЫСТРЕЕ** находить, исправлять ошибки
- Насколько проще - определяется **Testability**
- Нет времени - пишем testable код, design for testing
- Архитектура **очень** важна, больше всего влияет на Testability
- Какой вариант написания кода выбрать? - Самый «Testable»!
- Testable код - проще писать тесты
- Unit-тесты не заменяют UI

Ресурсы

юла
проект

Договориться с коллегами



юла
проект

Appium - Нюансы

- Разработка не будет поддерживать Appium.
  «Почему медленно ищет элемент?» ¯_(ツ)_/¯
- blackbox e2e-тесты (время)
- blackbox e2e-тесты (pass rate)
- Интеграции внутри ВК - идут вразрез с blackbox
- Целевое использование - сокращение регресса

юла
проект ВК

Appium не отменяет Натив!

- Решение проблем с интеграциями
- Решение проблем с «медленными» appium-тестами
- Ценность QA-специалиста, который может писать нативные тесты выше на рынке
- Разработчики пишут по TDD натив, QA приносят сценарии

Итого, у нас:

e2e - Appium, мелкие интеграционные тесты - натив

юла
проект

Что решаем нативными тестами

Раннее обнаружение проблем

- Тесты на merge-request
- Тесты по расписанию

юла
проект

Что решаем нативными тестами

Раннее обнаружение проблем

- Тесты на merge-request
- Тесты по расписанию

GrayBox

- TDD - сокращение времени на разработку
- Проверки в рамках 1-го экрана
- Интеграции больше не проблема (моки / подмена / дебаг-режимы и т.п.)

юла
проект VK

Пример теста из Smoke - Авторизация

- Нажать на поле ввода номера телефона
- Ввести номер телефона созданного пользователя
- Нажать «Подтвердить Номер»
- Нажать «Отклонить» во всплывшем попапе "Разрешить приложению Юла совершать звонки и ...»
- Проверить, что показан номер телефона, который был введен
- Нажать на поле ввода «Кода из СМС»
- Ввести 4 случайных числа
- Нажать «Готово»
- Открылся экран: «Главная»

юла
проект

Пример теста из Smoke - Просмотр карточки товара

- Назад/Фото/Лайки/шаринг присутствуют на странице
- Тайтл объявления присутствует на странице
- Позвонить/Написать присутствует на странице
- Блок гео присутствует на странице
- Атрибуты объявления присутствуют на странице
- Размещено/просмотры/в избранном присутствует на странице
- Блок инфы о юзере (+ кнопка подписки) присутствует на странице
- Блок шаринга присутствует на странице
- Кнопка «пожаловаться/обратиться в поддержку» присутствует на странице
- Блок похожих объявлений присутствует на странице

юла
проект

Как договориться с бизнесом

юла
проект

Как договориться с бизнесом

Собрать боли - эффект на бизнес

- Лучше всего - графики
- Ожидания команды
- Оценки производительности (команды)

юла
проект

Как договориться с бизнесом

Собрать боли - эффект на бизнес

- Лучше всего - графики
- Ожидания команды
- Оценки производительности (команды)

MVP - Minimum Viable Product

- Что входит в MVP
- Гипотеза - что проверяем
- Результат - что считаем успешной реализацией

юла
проект

Как договориться с бизнесом

Собрать боли - эффект на бизнес

- Лучше всего - графики
- Ожидания команды
- Оценки производительности (команды)

MVP - Minimum Viable Product

- Что входит в MVP
- Гипотеза - что проверяем
- Результат - что считаем успешной реализацией

Оценить затраты на MVP

- Ресурс соседних команд (devops) – минимум
- Железо – будет в следующем квартале
- Настройки CI (предподготовка) – собственными силами
- Ресурс команд Android

юла
проект

MVP - Minimum Viable Product

Что входит в MVP:

- Минимальная инфраструктура
- Smoke (несколько десятков сценариев) - ценность для QA
- Нужные изменения внутри приложения

юла
проект

MVP - Minimum Viable Product

Гипотезы - проверяем, что

- В идеальных условиях Pass Rate Native > Pass Rate Black Box
- Можно ли преодолеть проблемы с интеграциями внутри группы?
- Будут ли «бонусы» по скорости (vs Appium)
- Не появилось проблем в проекте из-за тестов
- Смогут ли разработчики с приоритетом техдолга реализовать Epic за квартал?

юла
проект

MVP - Minimum Viable Product

Результат -

smoke-набор реализован
и регулярно прогоняется
(спойлер - на каждый pr для mvp)

Кто?
- Инициативные группы

Самостоятельная группа людей, сложившаяся внутри одной или нескольких организаций, выполняющих родственную работу, имеющих одинаковые интересы и преследующих одни и те же цели.

- Разработчики находят время сами - **20% времени**
- Разнообразие в задачах (для разработчика) - новое, интересное
- Важен результат (улучшение проекта «для себя»)
- «Престиж» проекта

юла
проект

Что? - Требования к тестам

Можно проверить любой экран приложения
Не требуется проходить «пользовательский путь» до экрана

Отсюда:

- Установка тестового пользователя
- Переход внутри приложения куда угодно
- Установка ab-тестов в нужное состояние
- Мок запросов к бэкенду

юла
проект VK

Ресурсы, итоги:

Хочешь внедрить новое - придётся договориться!

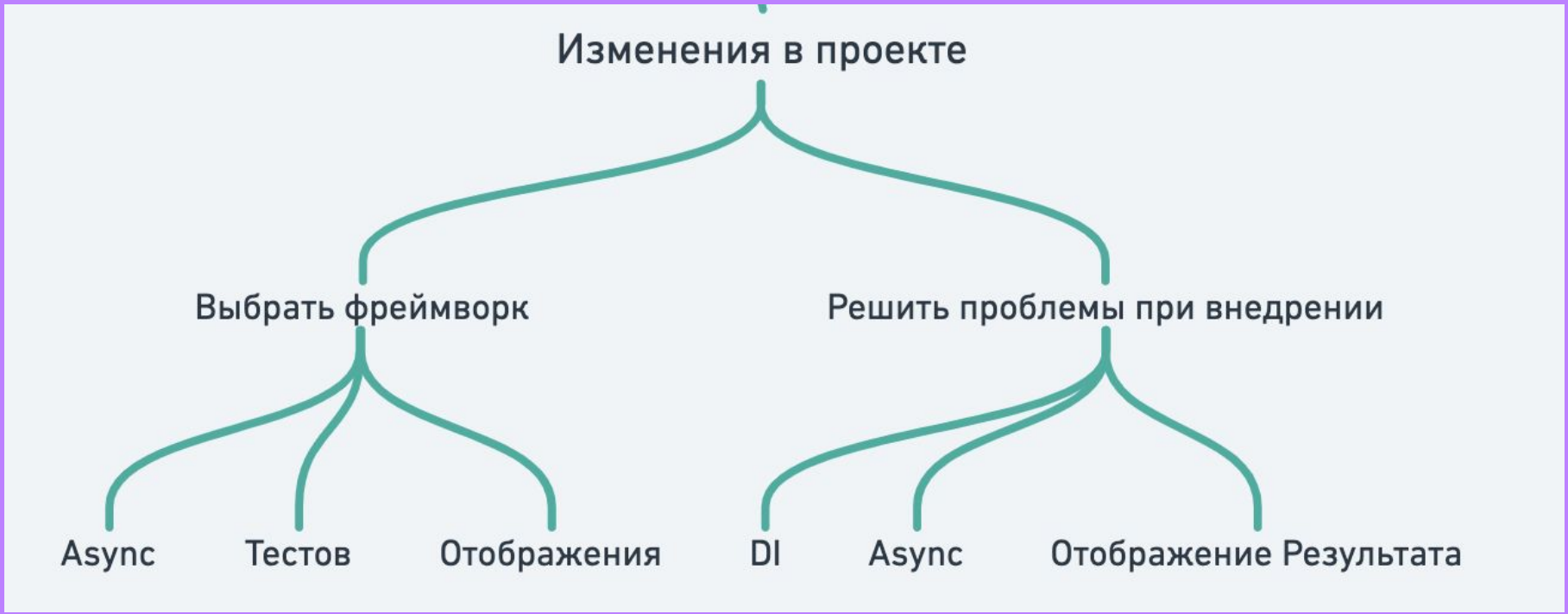
- Нужен MVP:
  a. Образ результата
  b. Какие проблемы мы решаем
  c. Чёткие сроки
  d. Поддержка коллег
- MVP == полномочия + минимальное железо

- Инициативные группы = Время + Рабочие руки
- Фасилитатор - контроль за процессом (и соседними отделами)

Изменения в проекте

юла
проект

Изменения в проекте



Setup тестов в проекте

- Завести TestApplication
- Не забыть прописать свой TestRunner
- Установить DI-компонент
- Решить проблемы своего кода =)

Setup

TestRunner

Кастомизируйте TestRunner

```
override fun newApplication(cl: ClassLoader, className: String, context: Context): Application {
//не забывайте DexOpener
  if (Build.VERSION.SDK_INT < Build.VERSION_CODES.P) {
    DexOpener.install(this)
  }
//вот так отключить компонент, который не нужен в тестах
    val componentName = "com.allgoritm.youla.service.MessagingService")
    PackageManager.setComponentEnabledSetting(
        componentName,
        PackageManager.COMPONENT_ENABLED_STATE_DISABLED,
        PackageManager.DONT_KILL_APP
    )

  return super.newApplication(cl, TestApplication::class.java.name, context)
}
```

юла
проект

Тестовый framework

Whatever you like - взяли kaspresso

Топ 3:

- flakySafely
- генерация allure-отчётов
- scenario

Async

Идея:

тестовый фреймворк не знает
о вашей асинхронной операции

Решение:

IdlingResource’ы

Из практических соображений:

- Rx2Idler + собственный scheduler
- CountingIdlingDispatcher - для Flow / Coroutines (issue)

юла
проект

DI

Controllability

DI

Требуются два графа зависимостей:

- В основном приложении
- В тестовом приложении (моки, аккаунт,авторизация, ab-тесты...)

юла
проект

DI - Don’t

```
@Override
override fun onCreate(savedInstanceState: Bundle?) {
  super.onCreate(savedInstanceState)
  val mainActivityComponent = DaggerMainActivityComponent.builder()
      .mainActivityModule(MainActivityModule(this))

.randomUserComponent(RandomUserApplication.get(this).getRandomUserApplicationComponent()
)
      .build();
  randomUsersApi = mainActivityComponent.getRandomUserApi();
// some other stuff
}
```

юла
проект

DI - do

- Собственные wrapper’ы: `fun getMainComponent() : MainComponent`
- JSR-330
- <Yours Favorite DI>, Hilt, Dagger-Android

юла
проект

Dagger Android

Идея:

Компоненты не знают, откуда inject’ируются зависимости

Dagger Android

Идея:

Компоненты не знают, откуда inject’ируются зависимости

Критика:

Долго генерируется AppComponent

DI: Dagger-Android замеры

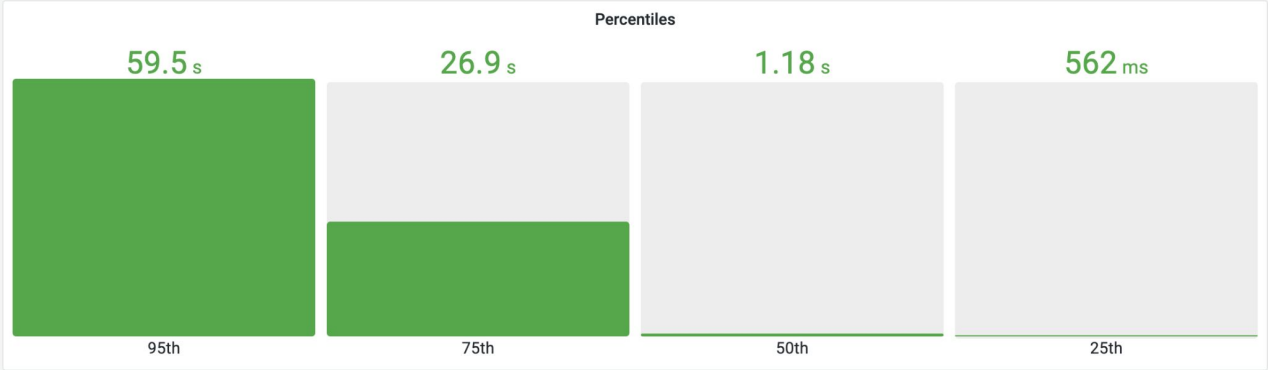
Приложение	Чистый build
~ 7 минут	70 модулей (1 монолит)

! Сколько занимает kapt-генерация DaggerAndroid AppComponent’a? !

16 Гб, Core i9, 2.3 Hgz, 8-мь ядер

юла
проект

Замеры Grafana via Talaiot



Query 1 Transform 0

A (external_metrics)

```
        EXTRACT (EPOCH FROM percentile_cont(0.75) within group (order by "end" - "start")) as "75th",
        EXTRACT (EPOCH FROM percentile_cont(0.5) within group (order by "end" - "start")) as "50th",
        EXTRACT (EPOCH FROM percentile_cont(0.25) within group (order by "end" - "start")) as "25th"
        -- ,'value' as "v"
from public.build
where platform='android'
and component_name=':app'
and component_type='module'
and device_type='local'
and task_name like '%kaptGoogleDebugKotlin%'
-- where task_name = ${task_name:sqlstring}
```

Format as Table Query Builder Show Help › Generated SQL ›

юла
проект

Проблема с приходом тестов?

```
//Продакшн SchedulersFactoryModule.class
@Module
abstract class SchedulersFactoryModule {
  @Binds
  abstract fun schedulersFactory(schedulersFactory: SchedulersFactoryImpl): SchedulersFactory
}

//Тест TestSchedulersFactoryModule.class
@Module
abstract class TestSchedulersFactoryModule {
  @Binds
  abstract fun schedulersFactory(schedulersFactory: TestIdleSchedulersFactory): SchedulersFactory
}
```

Дополнительный codegen?

юла
проект

Tip: сокращение времени kapt’ а (модуль)

```
DaggerTestAppComponent
  .builder()
  .application(this)
  .utilModule(UtilModule(
     accountCacheOverride = TestAccountCache,
//...//
  ))
  .build()
```

юла
проект

Tip: сокращение времени kapt’а (модуль)

```
@Module
class UtilModule(
  private val accountCacheOverride: AccountCache? = null,
  //...//
)
```

юла
проект

Tip: сокращение времени kapt’а (модуль)

```
fun provide() : AccountCache {
  if (accountCacheOverride != null)
      return accountCacheOverride
  // create implementation
}
```

юла
проект

Изменения в проекте

- Setup, Framework - есть туториалы
- Проблемы - Async, DI
- DI - **важен**, влияет на Controllability
- Инженерный подход - никому не верь, измерь =)

TestConfiguration - со звёздочкой

```
///before {
testConfiguration {
 api { rest.face.status returns RestStatusJson.statusOk }
 testAccount = TestAccount(
    userId = "$userId",
    localUser = gson.fromResponseFile("rest/user/$userId/user.json")
 )
 abTests = ABTests(showUnpublishButtonInProduct = true)
}
```

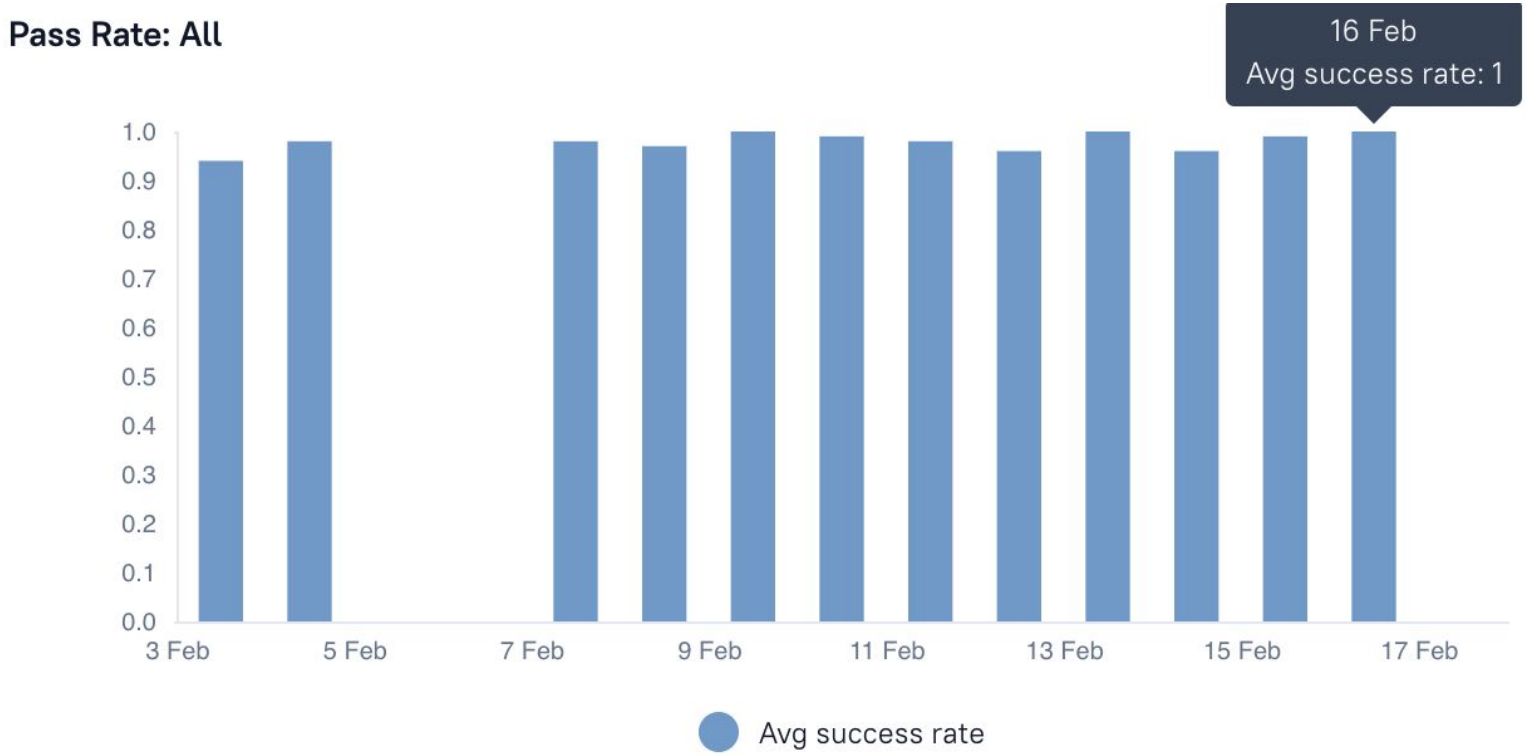
юла
проект

Бонус

Выпил legacy-кода =)

юла
проект

Как представить результат



Резюме

- Посмотрели на полезные практики для проекта (повысили testability)
- Поняли, как договориться с бизнесом и коллегами
- Предположили, какие изменения в проекте потребуются
- Поняли, как может выглядеть Minimum Viable Product
- Обрисовали пользу, применение и направления развития ui-тестов в проекте
- Переходим к инфраструктуре =)

юла
проект

Инфраструктура

юла
проект

Disclaimer

Подойдёт, если:

- нет опыта работы с эмулятором
- нет фермы девайсов
- слышали, что можно запускать тесты в docker

Не подойдёт, если:

- Уже имеется ферма девайсов
- Уже разворачивали ферму
- ...

Постановка задачи для инфраструктуры

Цель

Бизнесу необходимо четко понимать, что при разработке фичи нет критичных проблем связанных с бизнес логикой и различными интеграциями в уже существующей функциональности

юла
проект

Постановка задачи для инфраструктуры

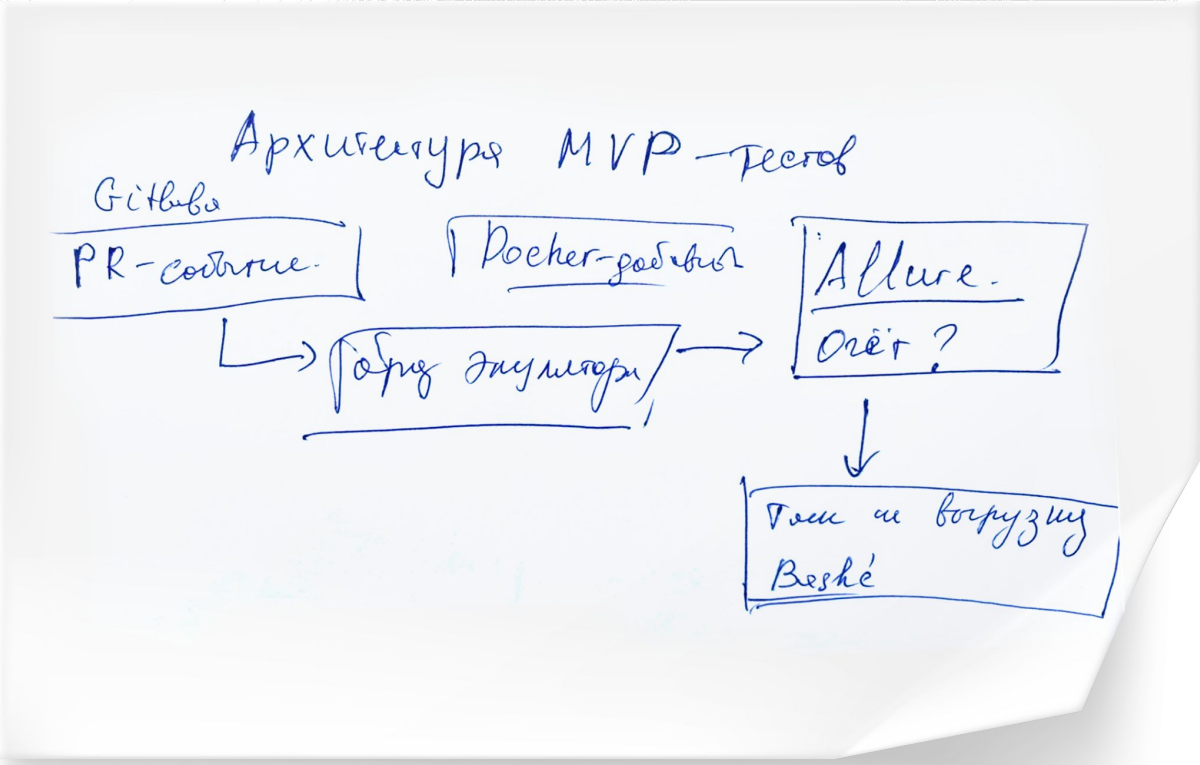
Локальная цель

Нужно было реализовать smoke набор тестов и начать их запускать

юла
проект

С самого начала у нас была какая-то тактика
WINDSOR MOSCOW
LES JOURS DE FRAN
2015
2015
и мы её придерживались

Схема на бумаге



юла
проект

Декомпозиция задачи

- Где будем запускать тесты?

Декомпозиция задачи

- Где будем запускать тесты?
- **В какой момент запускаем тесты?**

юла
проект

Декомпозиция задачи

- Где будем запускать тесты?
- В какой момент запускаем тесты?
- **Как будем смотреть результат?**

юла
проект

Где гонять тесты?

Удаленно на выделенной машине с эмуляторами

Плюсы

- Запускается на выделенной машине
- Можно автоматизировать
- Доступ к результатам появляется у всех

юла
проект

Где гонять тесты?

Удаленно на выделенной машине с эмуляторами

Плюсы

- Запускается на выделенной машине
- Можно автоматизировать
- Доступ к результатам появляется у всех

Минусы

- Нужна выделенная машина
- Необходимо следить за эмуляторами
- Нужно реализовать очередь при прогонах тестов

юла
проект VK

Где гонять тесты?

Удаленно на выделенной машине с эмуляторами

Плюсы

- Эмулятор запускается под прогон тестов
- Можно масштабировать

юла
проект

Где гонять тесты?

Удаленно на выделенной машине с эмуляторами

Плюсы

+ Эмулятор запускается под прогон тестов
+ Можно масштабировать

Минусы

− Нужна выделенная машина с поддержкой kvm
− Нужно собрать или найти docker образ с эмулятором
− Нужны начальные знания docker

юла
проект VK

Построение инфраструктуры

Основные моменты

- Запускаем тесты удаленно на эмуляторе в docker контейнере

юла
проект

Построение инфраструктуры

Основные моменты

- Запускаем тесты удаленно на эмуляторе в docker контейнере
- Запуск прогона тестов на каждый PR

юла
проект

Построение инфраструктуры

Основные моменты

- Запускаем тесты удаленно на эмуляторе в docker контейнере
- Запуск прогона тестов на каждый PR
- Запуск полного набора smoke тестов

юла
проект

Построение инфраструктуры

Основные моменты

- Запускаем тесты удаленно на эмуляторе в docker контейнере
- Запуск прогона тестов на каждый PR
- Запуск полного набора smoke тестов
- Результаты тестов смотрим в отчетах

юла
проект

Создание контейнера с эмулятором

Мы новички в вопросах инфраструктуры для тестов

Выбор решения с контейнером

google-emulator-container
Плюсы: ● Стабильное решение google ● Поддерживается сообществом google ● Есть различные примеры Минусы: ● В сборке dockerfile участвует python

Google

Выбор решения с контейнером

avito-tech
Плюсы: ● Стабильное решение avito ● Поддерживается avito ● Есть примеры Минусы: ● Большой проект

Avito

Выбор решения с контейнером

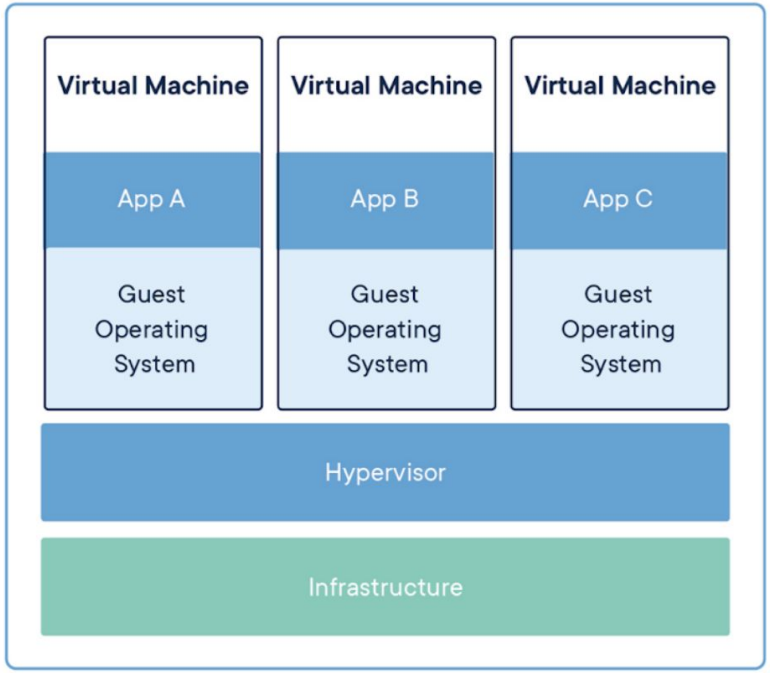
youla-tech
Плюсы: ● Стабильное решение youla ● Поддерживается командой youla ● Есть примеры ● минимальный dockerfile Минусы: ● нет более различных дополнений

Youla

Контейнер
и первые подводные камни

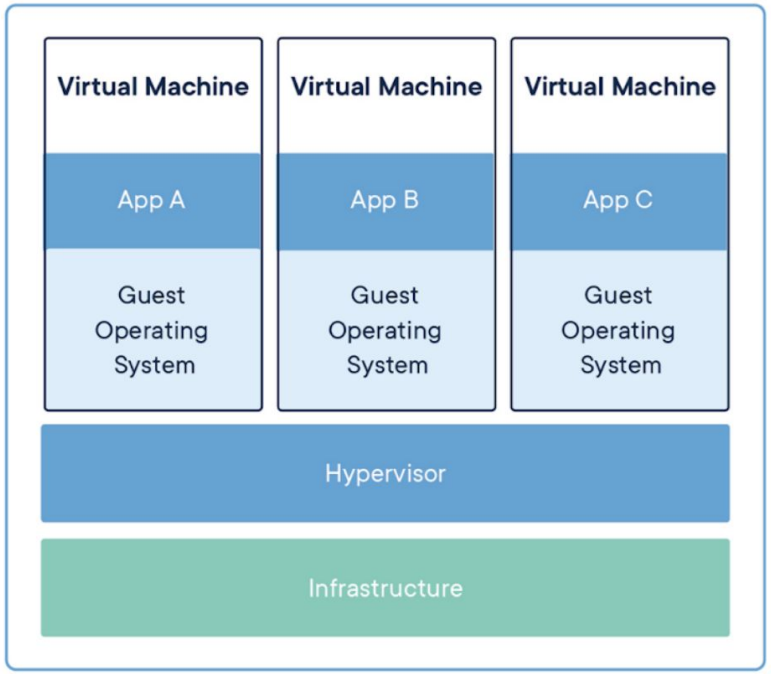
Создание dockerfile и проверка контейнера на локальной машине

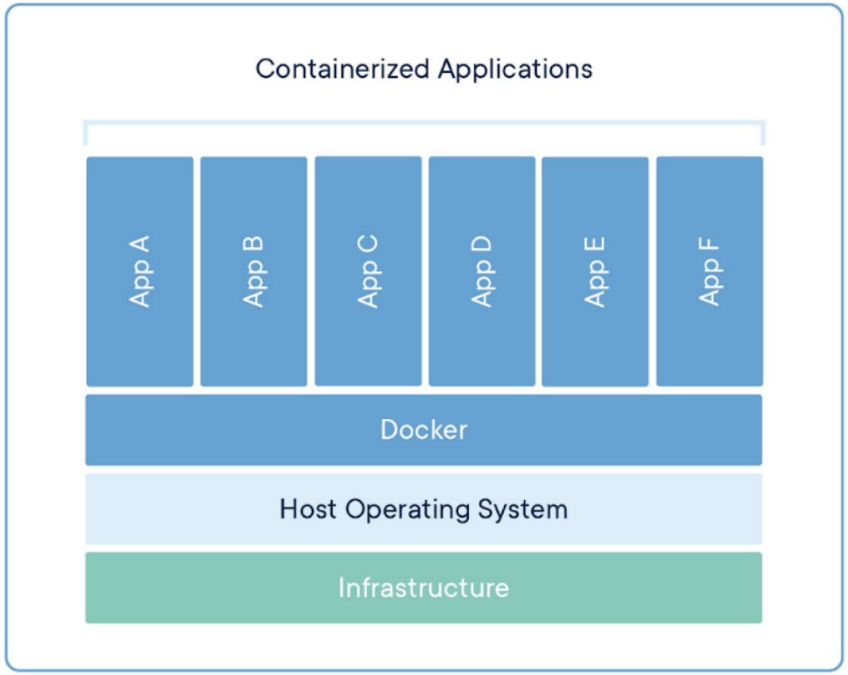
Виртуализация vs Контейнеризация



юла
проект

Виртуализация vs Контейнеризация





Android emulator

- Android эмулятор основан на qemu

юла
проект

Android emulator

- Android эмулятор основан на qemu
- QEMU (Quick Emulator) использует гипервизор KVM

юла
проект

Android emulator

- Android эмулятор основан на qemu
- QEMU (Quick Emulator) использует гипервизор KVM
- KVM (Kernel-based Virtual Machine) - стандартный модуль ядра linux

юла
проект

Android emulator

- Android эмулятор основан на qemu
- QEMU (Quick Emulator) использует гипервизор KVM
- KVM (Kernel-based Virtual Machine) - стандартный модуль ядра linux
- Чтобы запустить эмулятор в контейнере необходим KVM

юла
проект

Запуск контейнера с эмулятором на macOS

Проблема с вложенной виртуализацией на mac и windows

Запуск контейнера с эмулятором на macOS

Проблема с вложенной виртуализацией на mac и windows

Как итог:
Не получилось запустить контейнер с эмулятором на macOS

Запуск контейнера с эмулятором на виртуальной машине с поддержкой KVM

Арендуем виртуальную машину с поддержкой KVM и тестируем запуск эмулятора на ней

юла
проект

Запуск контейнера с эмулятором на виртуальной машине с поддержкой KVM

Арендуем виртуальную машину с поддержкой KVM и тестируем запуск эмулятора на ней

Как итог:

Эмулятор запустился и готов к работе

юла
проект

Запуск тестов на эмуляторе

- Сборка апк и тестового апк

юла
проект

Запуск тестов на эмуляторе

- Сборка апк и тестового апк
- Установка их на эмулятор

юла
проект

Запуск тестов на эмуляторе

- Сборка апк и тестового апк
- Установка их на эмулятор
- Запуск тестов adb shell

юла
проект

Сборка apk для тестирования

./gradlew :app:assembleGoogleDebug - debug apk

./gradlew :app:assembleGoogleDebugAndroidTest - test apk

юла
проект

Запуск тестов на эмуляторе

- Запуск всех тестов

юла
проект

Запуск тестов на эмуляторе

- Запуск всех тестов / выбранных тестов

юла
проект

Запуск тестов на эмуляторе

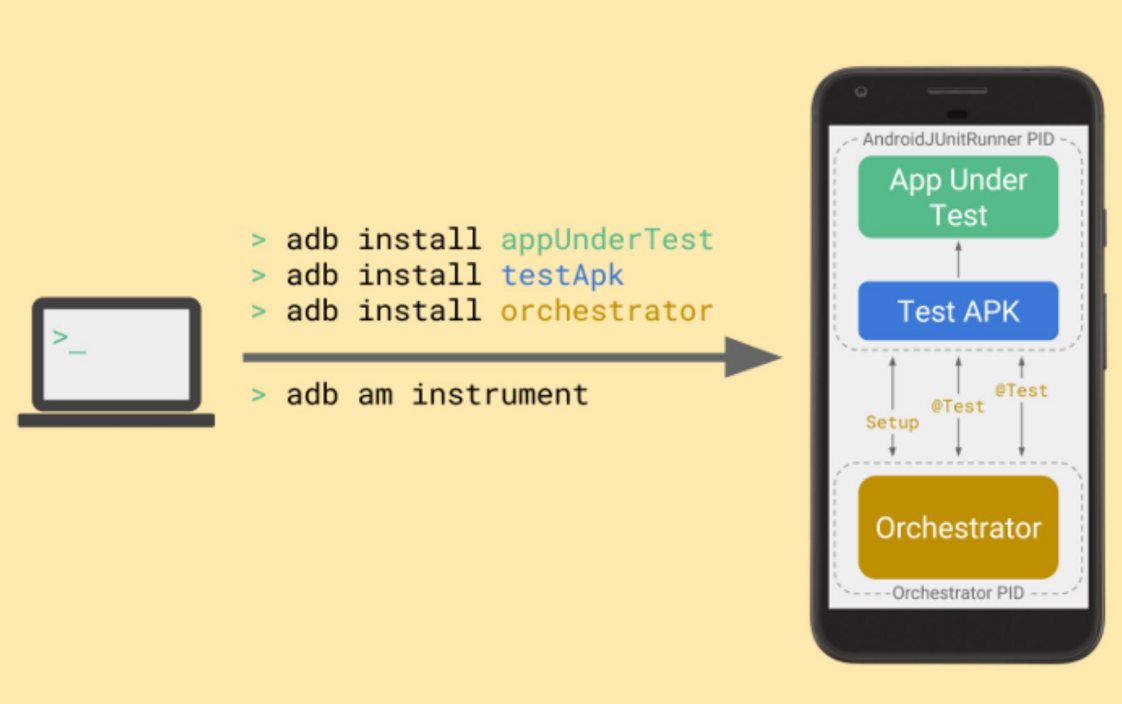
- Запуск всех тестов / выбранных тестов
- Запуск выбранных тестов с использованием оркестратора

юла
проект

Оркестратор

юла
проект

Оркестратор



юла
проект

Запуск тестов на эмуляторе

- Позволяет запустить каждый тест в отдельном процессе
- Повышаем стабильность и убираем зависимость от состояния процесса

юла
проект

Как посмотреть результаты прогона тестов?

Наша команда QA использует **allure**

Поэтому мы решили ничего не придумывать нового

юла
проект

Интеграция allure & gitlab

- Ограничены в хранении артефактов

юла
проект

Интеграция allure & gitlab

- Ограничены в хранении артефактов
- Отчет хранится внутри эмулятора

юла
проект

Allurectl и загрузка отчетов

```
allurectl upload --launch-name "Local 2200-12-31" path/to/allure-results
```

юла
проект

CI для запуска тестов

Gitlab CI

- Из коробки работает с контейнерами

юла
проект

CI для запуска тестов

Gitlab CI

- Из коробки работает с контейнерами
- Можем использовать разные образы на каждом этапе сборки

юла
проект

CI для запуска тестов

Gitlab CI

- Из коробки работает с контейнерами
- Можем использовать разные образы на каждом этапе сборки
- Есть поддержка команды devOps (в нашем случае)

юла
проект

Время собирать все воедино

- **build:** использует контейнер для сборки нашего проекта и тестовой apk

юла
проект

Время собирать все воедино

- **build:** использует контейнер для сборки нашего проекта и тестовой apk
- **ui_test:run:** использует контейнер с эмулятором для прогона тестов

юла
проект

Время собирать все воедино

- **build:** использует контейнер для сборки нашего проекта и тестовой apk
- **ui_test:run:** использует контейнер с эмулятором для прогона тестов
- **ui_test:report:** использует контейнер с allure для отправки отчета на сервер

юла
проект

Время собирать все воедино

- **build:** использует контейнер для сборки нашего проекта и тестовой apk
- **ui_test:run:** использует контейнер с эмулятором для прогона тестов
- **ui_test:report:** использует контейнер с allure для отправки отчета на сервер
- **profit!**

юла
проект

Интерактив

- **Докер-образ** (эмулятор)
- **bash-скрипт** с запуском

юла
проект

Что дальше...

- Добавление фермы дейвайсов
- Расширение базы тестов

юла
проект

Ферма эмуляторов на android

Иван Левиков

Резюме

- Описали схему построения инфраструктуры
- Научились запускать эмулятор в контейнере
- Поняли, как запускать тесты
- Увидели на в демо проекте, как это работает

юла
проект

Вопросы?

Почему не было

Непонятно:

- Какие изменения потребуются в проекте
- Сколько будем тратить времени на написание тестов

Ресурсы:

- Мало разработчиков в команде
- Нет/мало devops

Ограничения:

- Неудобная система сборки (старая версия Bamboo)
- Классное ручное тестирование
- Appium-тесты

На практике

- Понимаем основную цель применения «инструмента» и побочные
- Жёсткий архитектурный шаблон на всех уровнях
- Простые блоки
- Явные контракты («вход» - «выход») - чистые функции
- Пассивный ui (отрисовка/простановка state’a)
- Unit-тесты на сложную бизнес-логику (и на стейт)
- Осознанное ручное тестирование (в нашем случае +Appium)

юла
проект

О чём расскажем?

- **Как жить без ui-тестов → Testability**
- Ресурсы
- Изменения в коде проекта
- Инфра
- Демо

юла
проект VK

О чём расскажем?

- Как жить без ui-тестов → Testability
- **Ресурсы**
- Изменения в коде проекта
- Инфра
- Демо

юла
проект

О чём расскажем?

- Как жить без ui-тестов → Testability
- Ресурсы
- **Изменения в коде проекта**
- Инфра
- Демо

юла
проект

О чём расскажем?

- Как жить без ui-тестов → Testability
- Ресурсы
- Изменения в коде проекта
- **Инфра**
- Демо

юла
проект

О чём расскажем?

- Как жить без ui-тестов → Testability
- Ресурсы
- Изменения в коде проекта
- Инфра
- **Демо**

юла
проект

Полезные практики

Осознанное «ручное» тестирование

(в нашем случае ещё и Appium)

Минутка интерактива

Варианты ответа:

- до 1 минуты
- 1-2 минуты
- более 5 минут

юла
проект