Нативные автотесты кроссплатформенного Flutter

Мария Лещинская

Surf
Обо мне

- native и flutter мобильные приложения
- e2e автотесты на Calabash
- скриншот-тестирование в e2e-автотестах
- flutter-автоматизация

Surf

О нас

- 11 лет создаем мобильные продукты
- 100+ проектов за плечами: native и flutter
- Google Certified Agency
- Топ-10 мобильных разработчиков
- Flutter Dev Podcast

Surf
Автотестирование

- Calabash
- Flutter

Surf
Проекты

Автотестирование

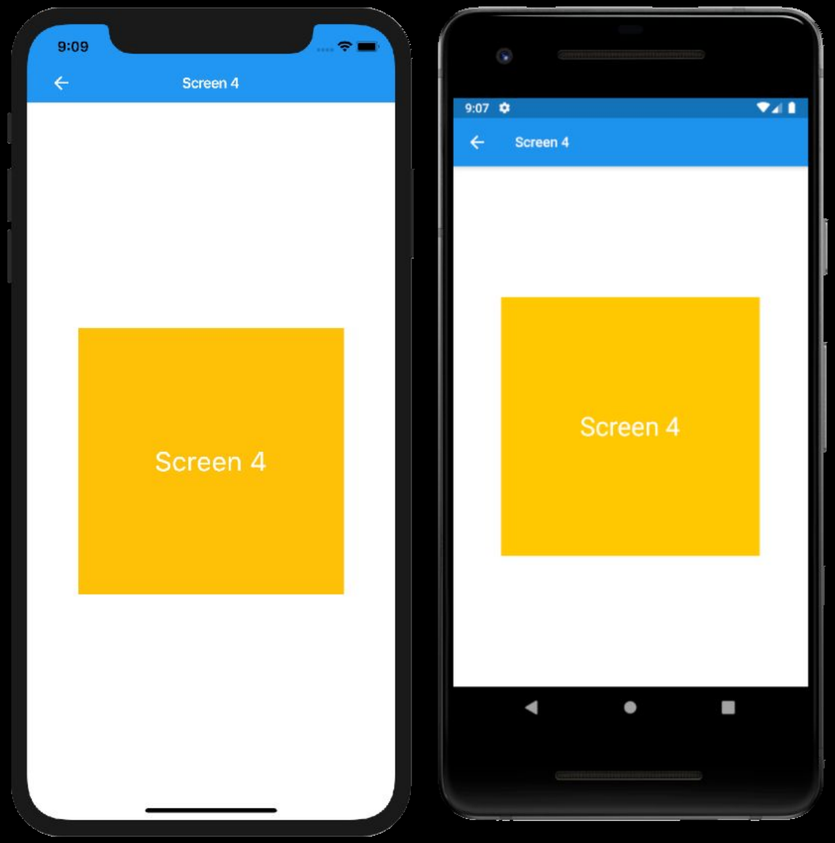
Flutter

Автотестирование

Flutter

vs

Flutter
Кроссплатформенные приложения



Flutter

Единая кодовая база

Flutter

Единая кодовая база

- В 2 раза меньше задач

Flutter

Единая кодовая база

- В 2 раза меньше задач
- Логика на обеих платформах одинакова

Flutter
Единая кодовая база

- В 2 раза меньше задач
- Логика на обеих платформах одинакова
- Экономия времени при тестировании

Flutter

Коммуникация внутри команды

Flutter

Коммуникация внутри команды

- Одна реализация на двух платформах

Flutter

Коммуникация внутри команды

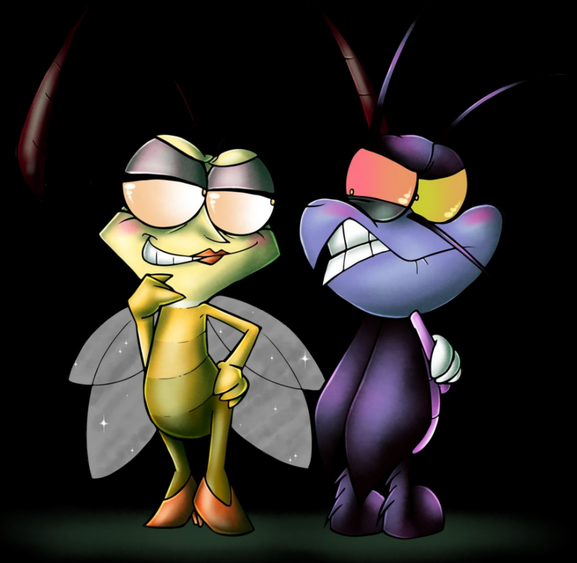
- Одна реализация на двух платформах
- Информацию нужно распространять только на одну команду разработки

Flutter

Минусы

Flutter
Минусы

- Баги фреймворка



Flutter
Минусы

- Баги фреймворка
- Недоработки сторонних библиотек

Flutter
Минусы

- Баги фреймворка
- Недоработки сторонних библиотек
- Ожидаемое нативное поведение

Flutter
Минусы

- Баги фреймворка
- Недоработки сторонних библиотек
- Ожидаемое нативное поведение
  - backswipe на iOS

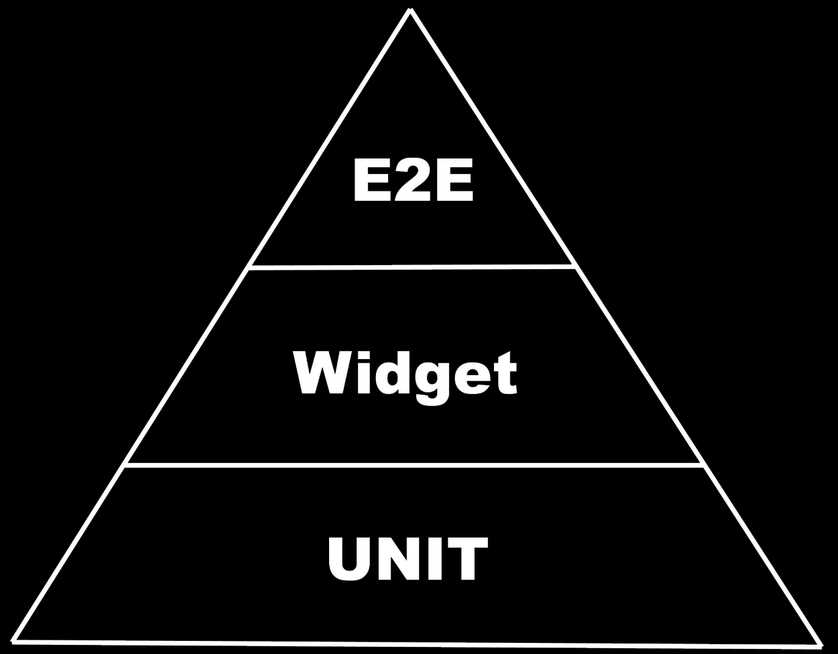
Flutter
Минусы

- Баги фреймворка
- Недоработки сторонних библиотек
- Ожидаемое нативное поведение
  - backswipe на iOS
  - запрос на доступ к уведомлениям на iOS

Flutter

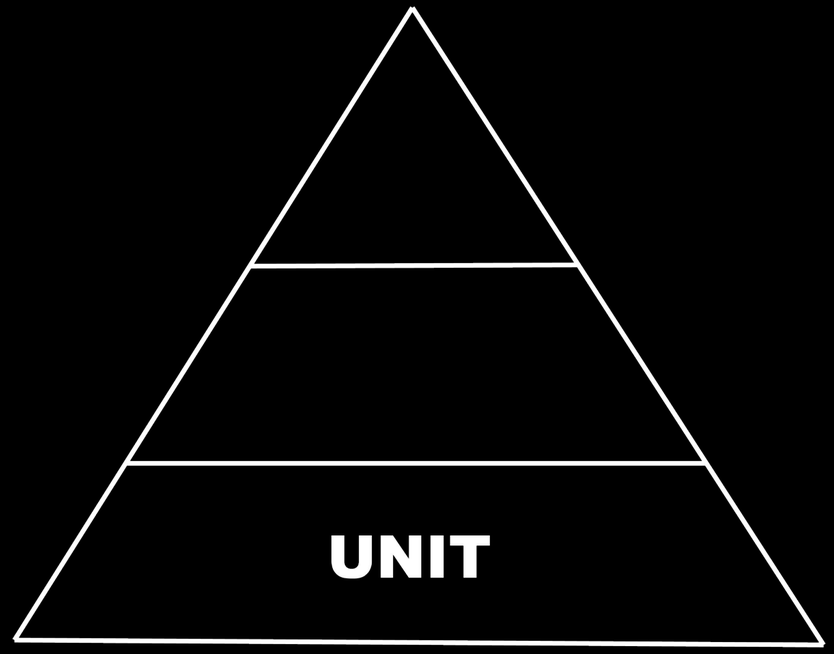
- это просто кроссплатформа

Автотесты
Flutter



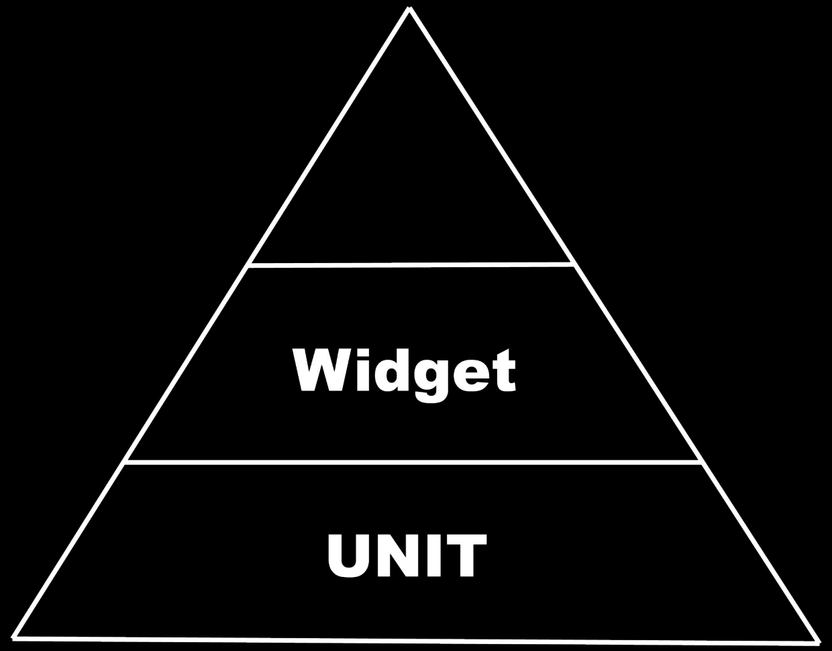
Автотесты
Flutter

- unit -> конкретный модуль системы
  - не эмулируется приложение
  - проверка поведения контроллеров и элементов



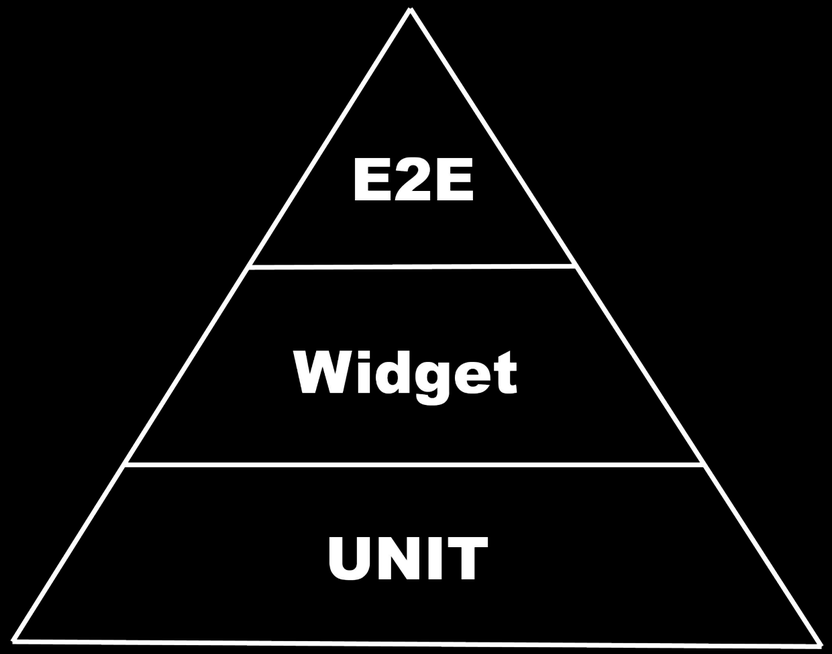
Автотесты
Flutter

- unit -> конкретный модуль системы
  - не эмулируется приложение
  - проверка поведения контроллеров и элементов
- widget -> покрытие виджетов, использующихся в фичах
  - эмулируются виджеты
  - проверка поведения виджета внутри сценария

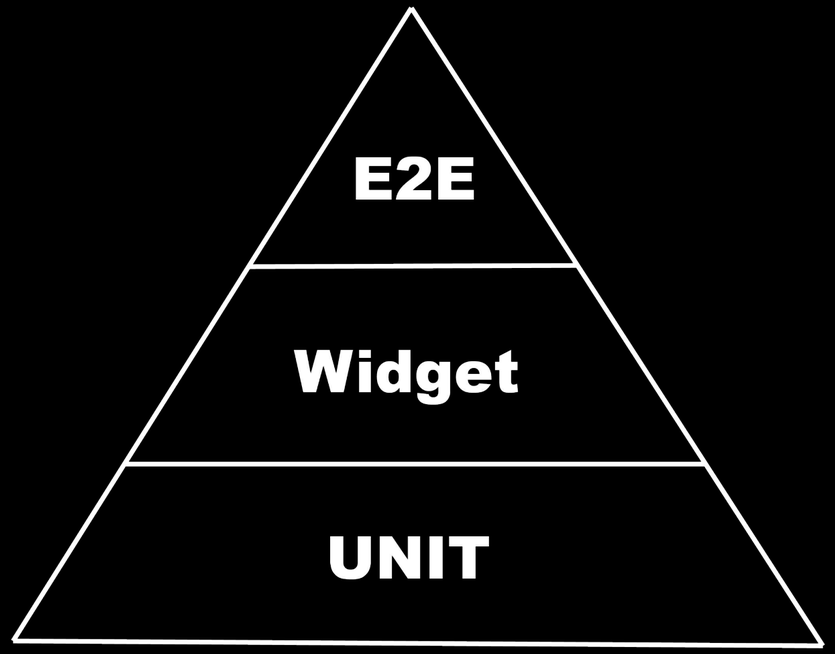


Автотесты
Flutter

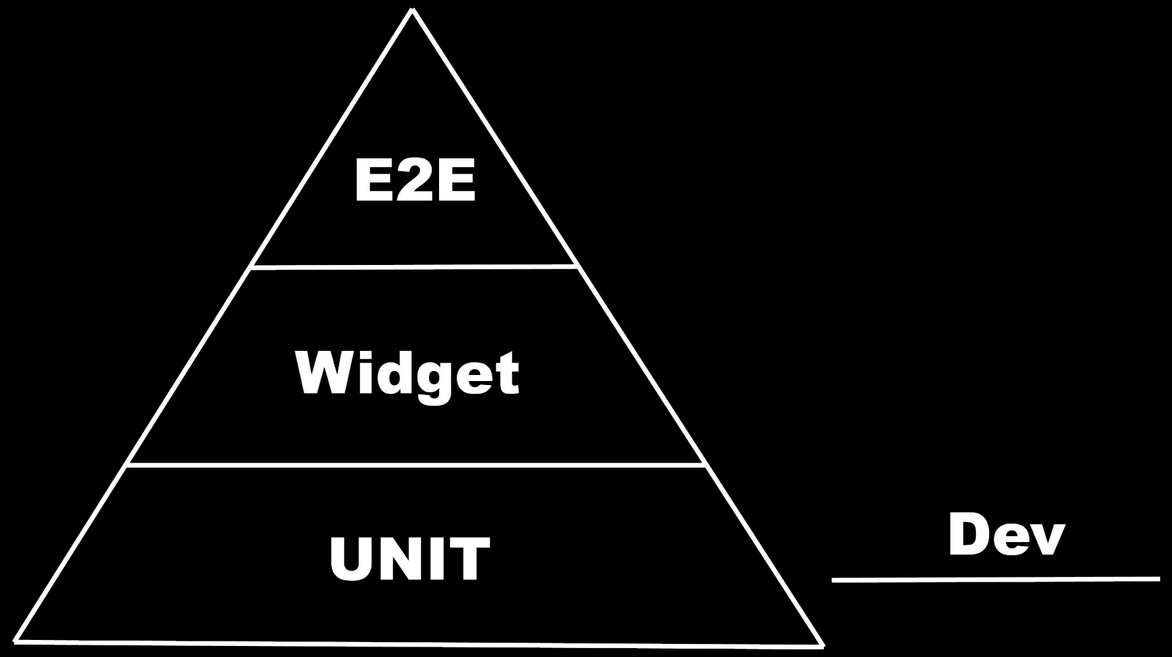
- unit -> конкретный модуль системы
  - не эмулируется приложение
  - проверка поведения контроллеров и элементов
- widget -> покрытие виджетов, использующихся в фичах
  - эмулируются виджеты
  - проверка поведения виджета внутри сценария
- e2e -> покрытие пользовательских сценариев
  - загружается приложение (с реальными сервисами, API)
  - проверка поведения мп внутри сценария



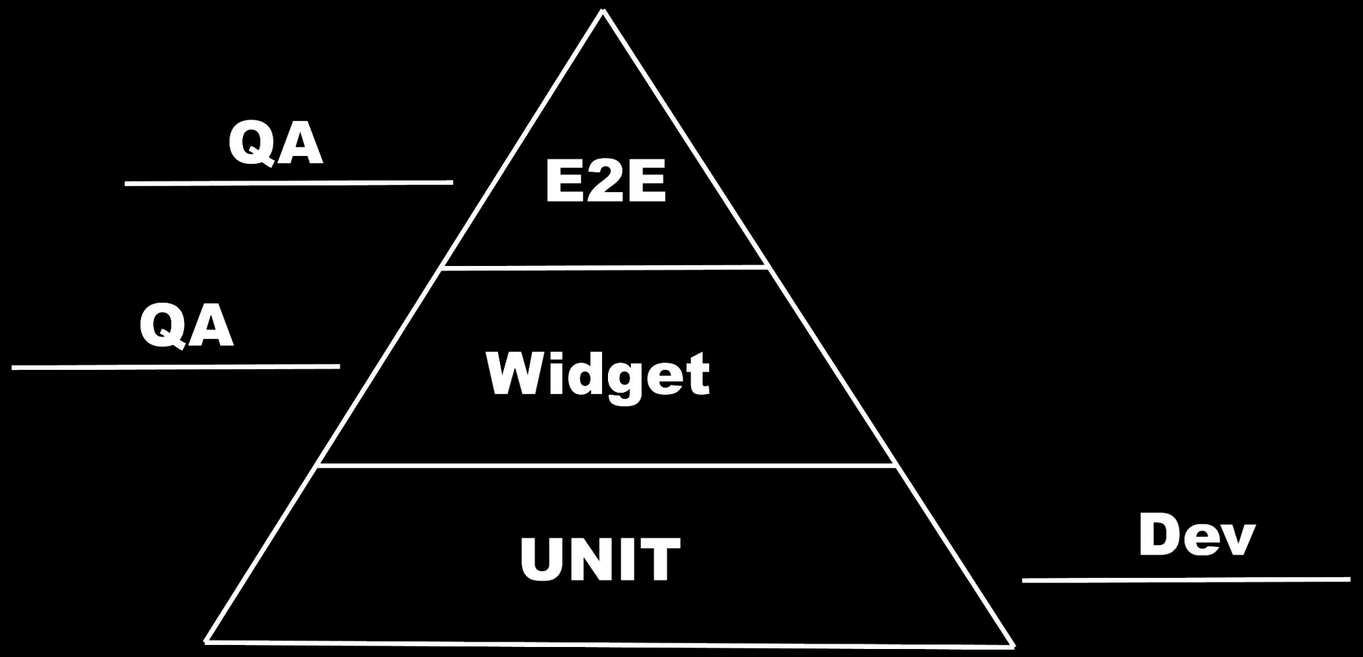
Автотесты в Surf
Flutter



Автотесты в Surf
Flutter



Автотесты в Surf
Flutter



Автотесты в Surf

Flutter: стратегия QA

Автотесты в Surf
Flutter: стратегия QA

- стабильные функциональности

Автотесты в Surf
Flutter: стратегия QA

- стабильные функциональности
- автотесты

Автотесты в Surf
Flutter: стратегия QA

- стабильные функциональности
- автотесты
  - запускать на pull request’ах

Автотесты в Surf
Flutter: стратегия QA

- стабильные функциональности
- автотесты
  - запускать на pull request’ах - widget

Автотесты в Surf
Flutter: стратегия QA

- стабильные функциональности
- автотесты
  - запускать на pull request’ах - widget
  - полные пользовательские сценарии - e2e

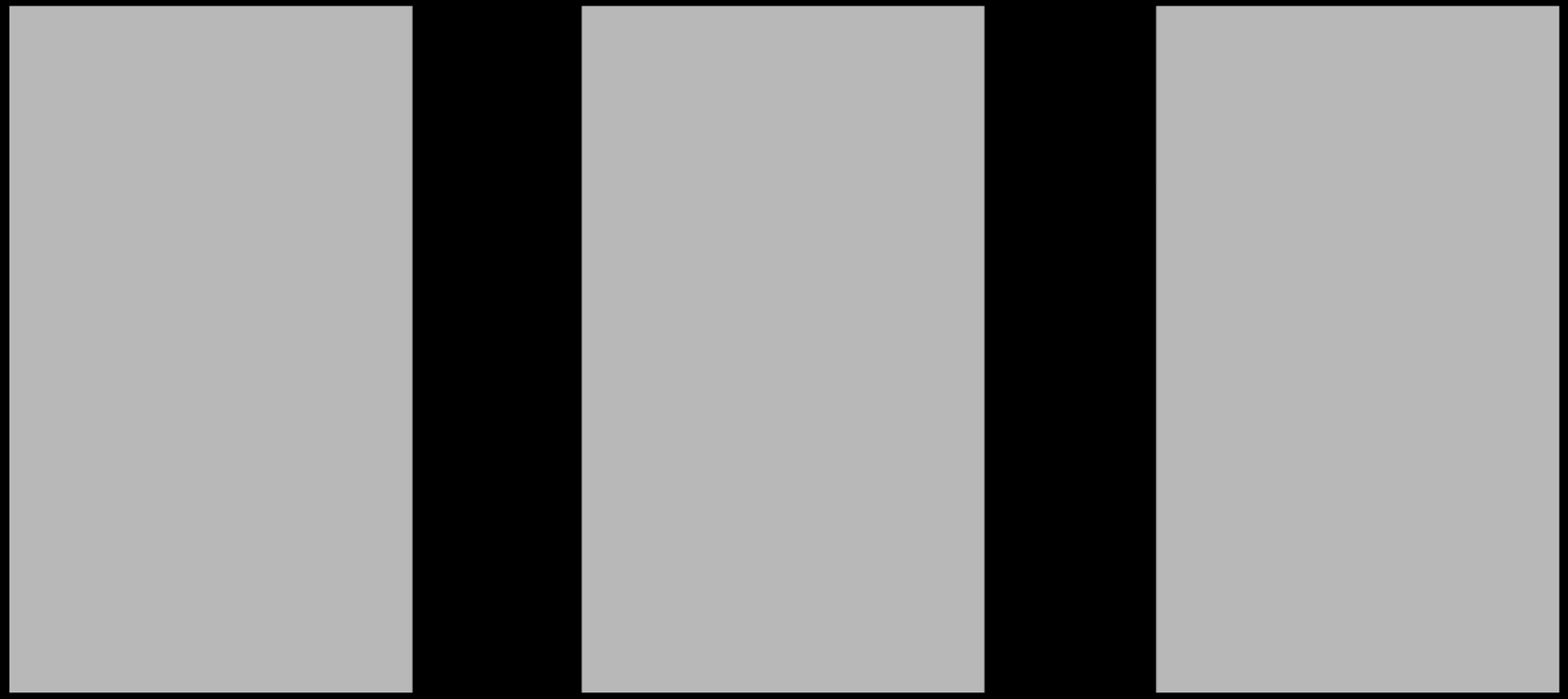
Автотесты в Surf
Flutter: стратегия QA

- стабильные функциональности
- автотесты: widget + e2e

e2e-тесты	**widget-тесты**
запуск ночью на пользовательских сценариях	запуск на pull request’ах
работа тестовым сервером	работа с моками

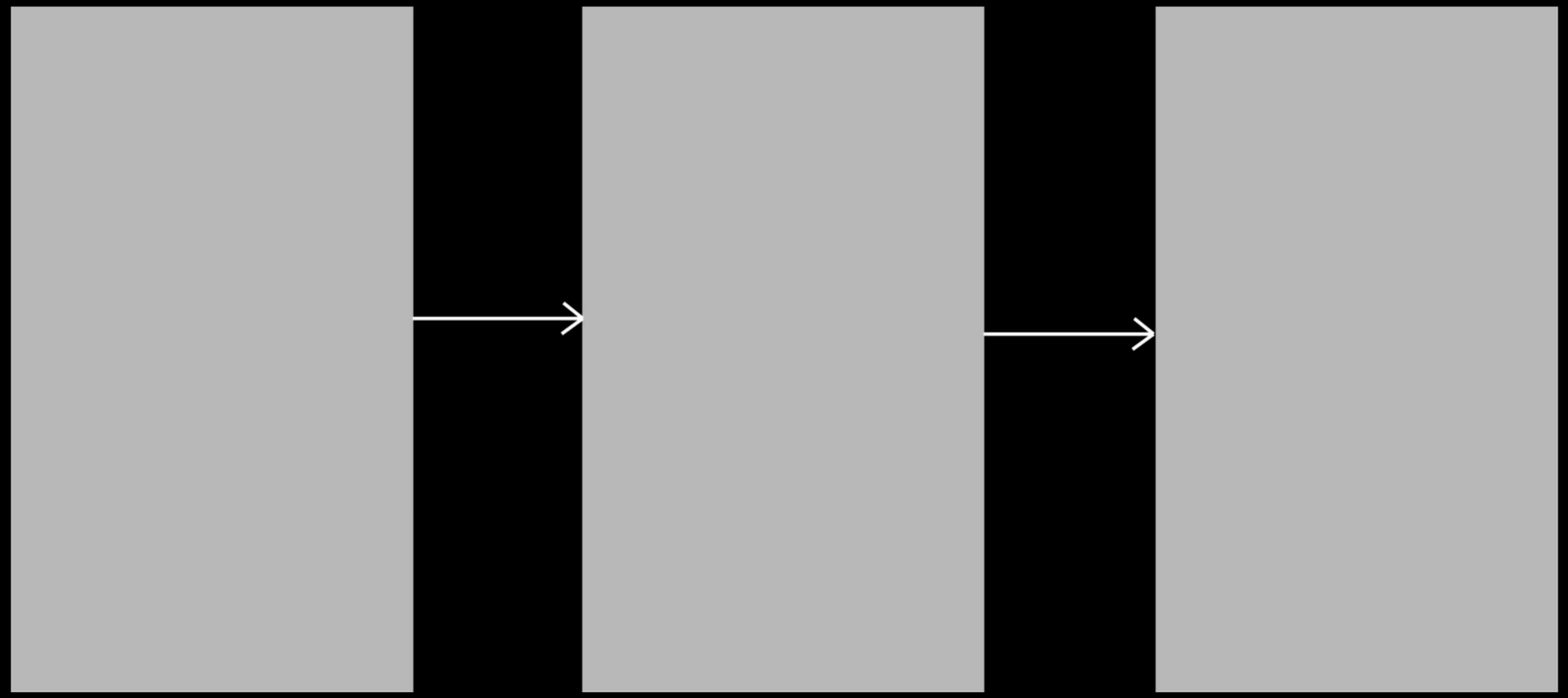
Автотесты в Surf

Flutter: стратегия QA



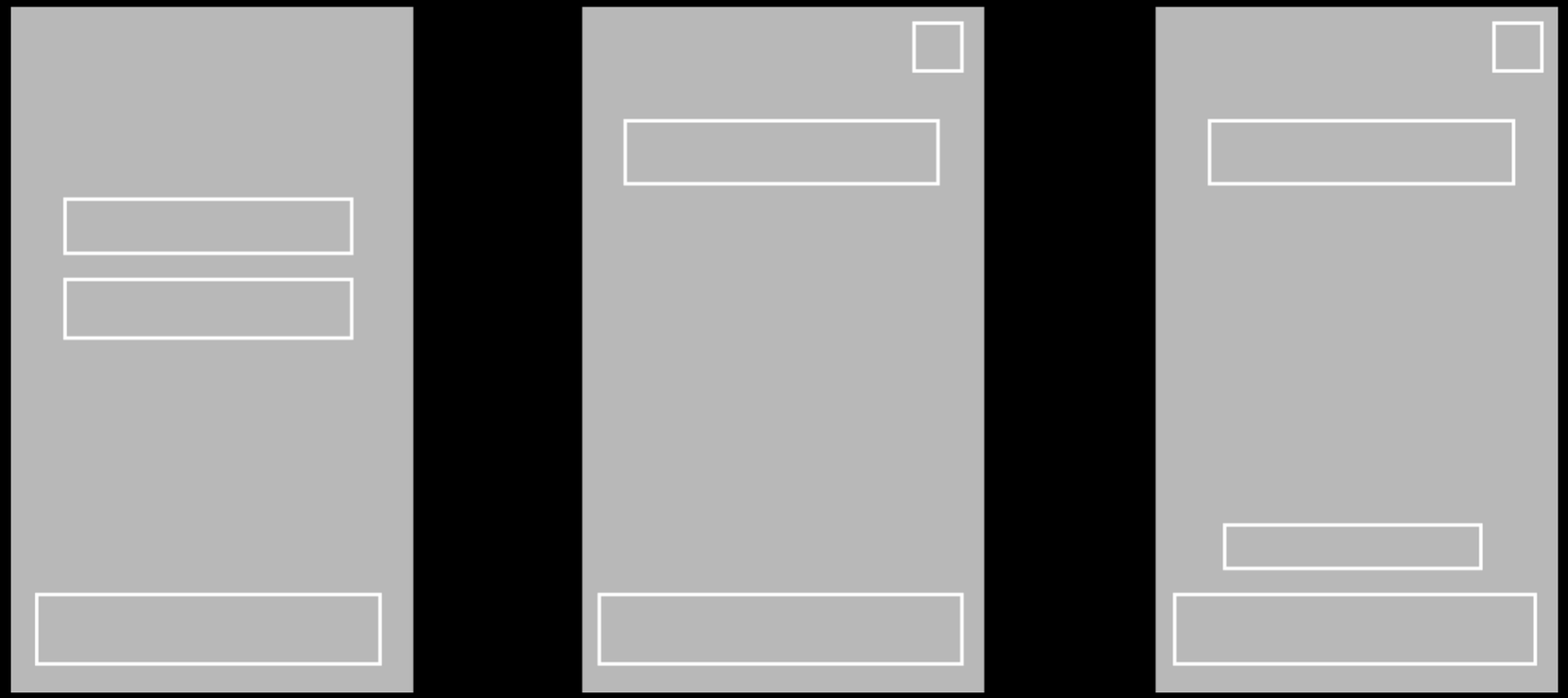
Автотесты в Surf

Flutter: стратегия QA



Автотесты в Surf

Flutter: стратегия QA



Автотесты в Surf

Flutter: начало

Автотесты в Surf
Flutter: начало

- Gherkin
  - понятность
  - аналогия ручному сценарию

Автотесты в Surf
Flutter: начало

- Gherkin
  - понятность
  - аналогия ручному сценарию

gherkin

```gherkin
#language: ru
Функциональность: Авторизация
  @auth
  Сценарий: Авто: Авторизация с корректным ОТП
    Когда Я запускаю приложение
    И Я перехожу на таб Библиотека
    И Я тапаю на кнопку Войти
    И Я ввожу рандомный телефон
    И Я тапаю на кнопку далее
    И Я ввожу ОТП код "12345"
    Тогда Я вижу таб Библиотека авторизанта

< . . . >
```

Автотесты в Surf
Flutter: начало

- Gherkin
  - понятность
  - аналогия ручному сценарию
- Widget+e2e

Автотесты в Surf
Flutter: начало

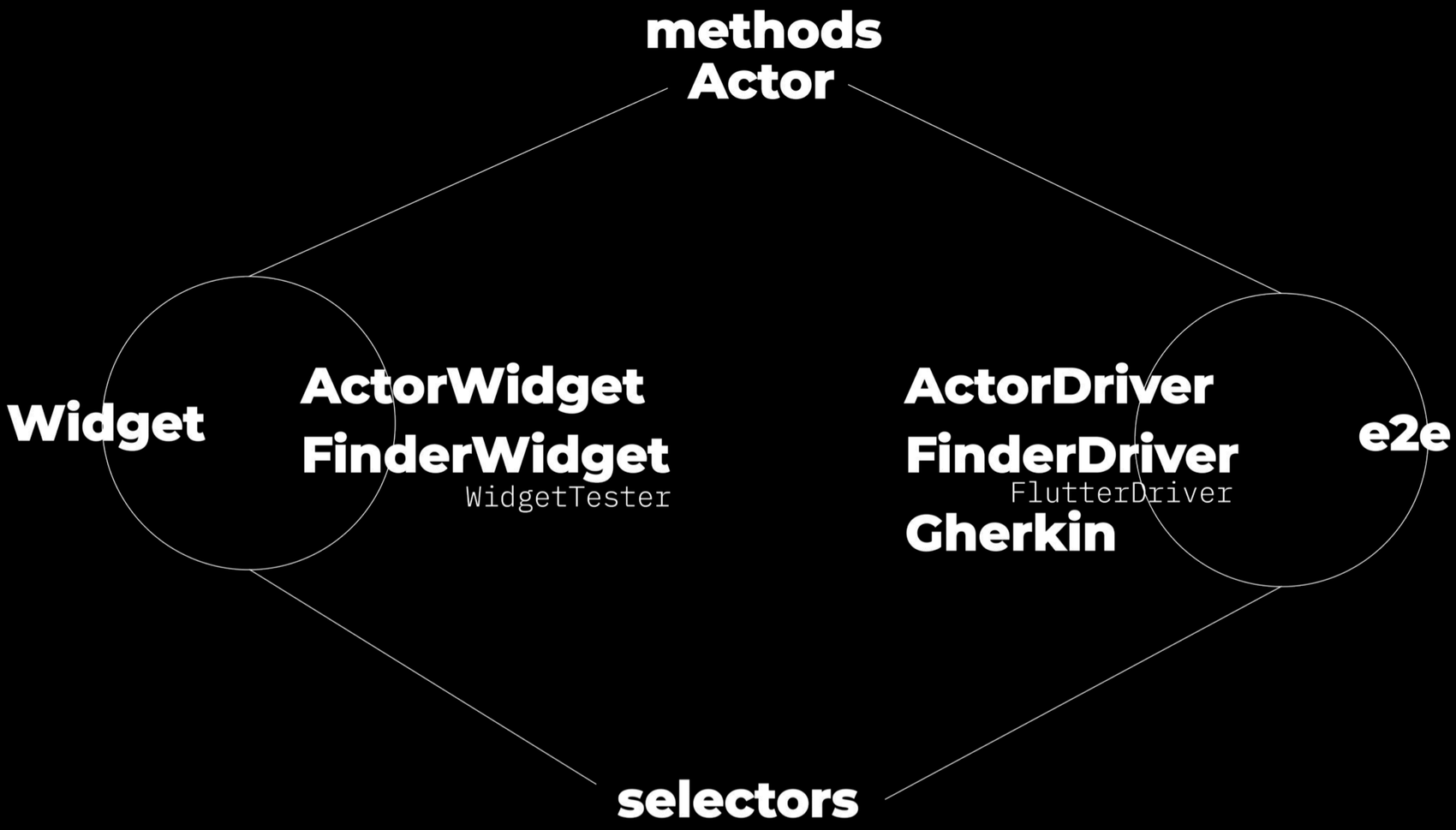
- Gherkin
  - понятность
  - аналогия ручному сценарию
- Widget+e2e
- Переиспользуемые компоненты
  - селекторы
  - функции

Автотесты в Surf
Flutter: начало

- Gherkin
  - понятность
  - аналогия ручному сценарию
- Widget+e2e
- Переиспользуемые компоненты
  - селекторы
  - функции
- Отчеты

Автотесты в Surf

Структура: начало



Структура: начало
Gherkin

- flutter gherkin

- ogurets flutter

Структура: начало
Gherkin

- `flutter_gherkin`

Структура: начало
Gherkin

- `flutter_gherkin`

gherkin

```
#language: ru
Функциональность: Авторизация
  @auth
  Сценарий: Авто: Авторизация с корректным ОТП
    Когда Я запускаю приложение
    И Я перехожу на таб Библиотека
    И Я тапаю на кнопку Войти
    И Я ввожу рандомный телефон
    И Я тапаю на кнопку далее
    И Я ввожу ОТП код "12345"
    Тогда Я вижу таб Библиотека авторизанта

< . . . >
```

Структура: начало
Имплементация шагов e2e

e2e

```
class AuthSteps {
 Iterable<StepDefinitionGeneric> get steps => [
       when<FlutterWorld>(
         'Я тапаю на кнопку Войти',
         (context) async {
           final actor = ActorDriver(context.world.driver);
           await actor.runAsync(actor.tap(auth.loginBtn));
         },
       ),
       when1<String, FlutterWorld>(
         'Я ввожу ОТП код {string}',
         (code, context) async {
           final actor = ActorDriver(context.world.driver);
           await actor.tap(auth.otpFieldNoError);
           await actor.enterText(code);
         },
       ),
```

Структура: начало
Имплементация шагов e2e

e2e

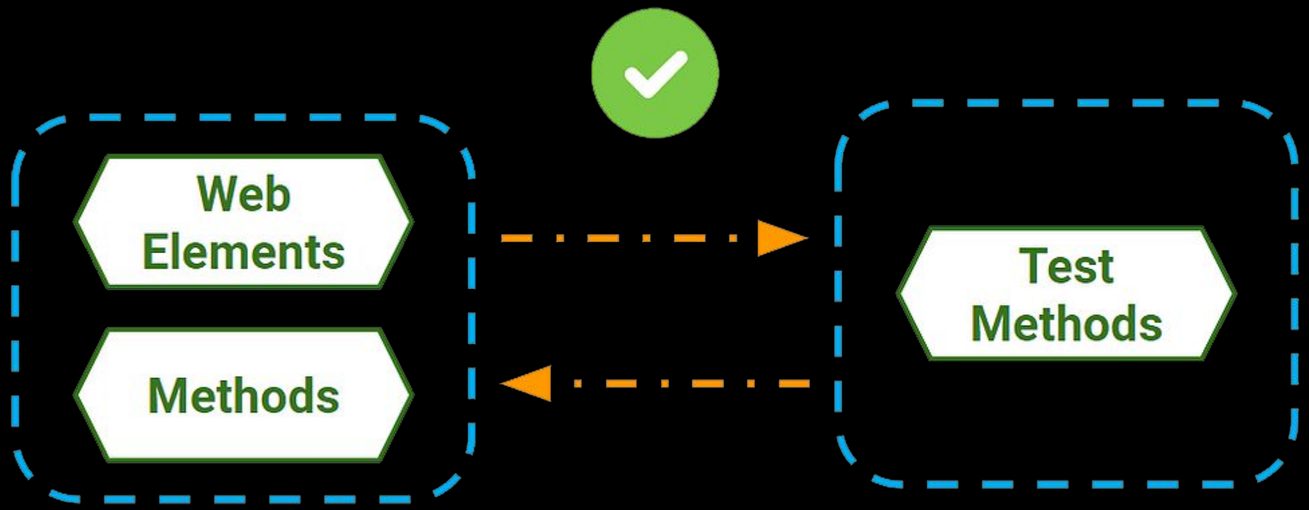
```
class AuthSteps {
 Iterable<StepDefinitionGeneric> get steps => [
      when<FlutterWorld>(
        'Я тапаю на кнопку Войти',
        (context) async {
          final actor = ActorDriver(context.world.driver);
          await actor.runAsync(actor.tap(auth.loginBtn));
        },
      ),
      when1<String, FlutterWorld>(
        'Я ввожу ОТП код {string}',
        (code, context) async {
          final actor = ActorDriver(context.world.driver);
          await actor.tap(auth.otpFieldNoError);
          await actor.enterText(code);
        },
      ),
```

Структура: начало
Имплементация шагов e2e

e2e

```
class AuthSteps {
 Iterable<StepDefinitionGeneric> get steps => [
      when<FlutterWorld>(
        'Я тапаю на кнопку Войти',
        (context) async {
          final actor = ActorDriver(context.world.driver);
          await actor.runAsync(actor.tap(auth.loginBtn));
        },
      ),
      when1<String, FlutterWorld>(
        'Я ввожу ОТП код {string}',
        (code, context) async {
          final actor = ActorDriver(context.world.driver);
          await actor.tap(auth.otpFieldNoError);
          await actor.enterText(code);
        },
      ),
```

Структура: начало
~ **Page Object pattern**



Структура: начало
~ **Page Object pattern**

- Локаторы для всех элементов
- Методы для взаимодействия с элементами

selectors

```
class AuthScreen {
 // кнопка Войти
 KeySelector get loginBtn => KeySelector('login_btn');

 // поле Номер телефона
 KeySelector get phoneField => KeySelector('phone_field');

 // поле Ввода ОТП
 KeySelector get otpField => KeySelector('otp_invisible_input');
< . . . >
}
```

functions

```
Future enterSearchField(actor, text) async {
   await actor.tap(search);
   await actor.enterTextTo(search, text);
 }
< . . . >
```

Структура: начало
~ Page Object pattern

- Локаторы для всех элементов -> идентификатор элемента
- Методы для взаимодействия с элементами

selectors

```
class AuthScreen {
 // кнопка Войти
 KeySelector get loginBtn => KeySelector('login_btn');

 // поле Номер телефона
 KeySelector get phoneField => KeySelector('phone_field');

 // поле Ввода ОТП
 KeySelector get otpField => KeySelector('otp_invisible_input');
< . . . >
}
```

functions

```
Future enterSearchField(actor, text) async {
   await actor.tap(search);
   await actor.enterTextTo(search, text);
 }
< . . . >
```

Структура: начало
~ Page Object pattern

- Локаторы для всех элементов -> идентификатор элемента -> ключ виджета
- Методы для взаимодействия с элементами

selectors

```
class AuthScreen {
 // кнопка Войти
 KeySelector get loginBtn => KeySelector('login_btn');

 // поле Номер телефона
 KeySelector get phoneField => KeySelector('phone_field');

 // поле Ввода ОТП
 KeySelector get otpField => KeySelector('otp_invisible_input');
< . . . >
}
```

functions

```
Future enterSearchField(actor, text) async {
   await actor.tap(search);
   await actor.enterTextTo(search, text);
 }
< . . . >
```

Структура: начало
~ **Page Object pattern**

- `e2e-tests`
- `Widget-tests`

Структура: начало
~ **Page Object pattern**

- `e2e-tests -> Finder`
- `Widget-tests -> SerializeableFinder`

Структура: начало
Переиспользуемые компоненты

- Actor
- Finder

Actor

```
class Actor {
 dynamic actor;
 dynamic finder;

 Future tap(key) async => await actor.tap(finder.locate(key));

< . . . >
}
```

Структура: начало
Переиспользуемые компоненты

- FinderWidget

FinderWidget

```
class FinderWidget {
 FinderWidget() {
   _finder = find;
 }
 CommonFinders _finder;
 Finder locate(key) {
   if (key is KeySelector) {
     return _finder.byKey(Key(key.value.toString()));
   } else if (key is TextSelector) {
     return _finder.text(key.value);
   } else if (key is DescendantSelector) {
     return _finder.descendant(
         of: locate(key.value[0]), matching: locate(key.value[1]));
   } else {
     throw ArgumentError('Selector $key has wrong type ${key.runtimeType}');
   }
 }
< . . . >
}
```

- FinderDriver

FinderDriver

```
class FinderDriver {
 FinderDriver() {
   _finder = find;
 }
 CommonFinders _finder;
 SerializableFinder locate(key) {
   if (key is KeySelector) {
     return _finder.byValueKey(key.value);
   } else if (key is TextSelector) {
     return _finder.text(key.value);
   } else if (key is DescendantSelector) {
     return _finder.descendant(
         of: locate(key.value[0]), matching: locate(key.value[1]));
   } else {
     throw ArgumentError('Selector $key has wrong type ${key.runtimeType}');
   }
 }
< . . . >
}
```

Структура: начало
Переиспользуемые компоненты

- FinderWidget

FinderWidget

```
class FinderWidget {
 FinderWidget() {
   _finder = find;
 }
 CommonFinders _finder;
 Finder locate(key) {
   if (key is KeySelector) {
     return _finder.byKey(Key(key.value.toString()));
   } else if (key is TextSelector) {
     return _finder.text(key.value);
   } else if (key is DescendantSelector) {
     return _finder.descendant(
         of: locate(key.value[0]), matching: locate(key.value[1]));
   } else {
     throw ArgumentError('Selector $key has wrong type ${key.runtimeType}');
   }
 }
< . . . >
}
```

- FinderDriver

FinderDriver

```
class FinderDriver {
 FinderDriver() {
   _finder = find;
 }
 CommonFinders _finder;
 SerializableFinder locate(key) {
   if (key is KeySelector) {
     return _finder.byValueKey(key.value);
   } else if (key is TextSelector) {
     return _finder.text(key.value);
   } else if (key is DescendantSelector) {
     return _finder.descendant(
         of: locate(key.value[0]), matching: locate(key.value[1]));
   } else {
     throw ArgumentError('Selector $key has wrong type ${key.runtimeType}');
   }
 }
< . . . >
}
```

Структура: начало
Переиспользуемые компоненты

- BaseSelector и другие

Selector

```
class BaseSelector {
 dynamic _value;
 dynamic get value => _value;
}

class KeySelector extends BaseSelector {
 KeySelector(key) {
   _value = key;
 }
}

class TextSelector extends BaseSelector {
 TextSelector(String text) {
   _valueText = text;
 }
 String _valueText;
```

Selector

```
 @override
 String get value => _valueText;
}
class DescendantSelector extends BaseSelector {
 DescendantSelector(selector1, selector2) {
   _value1 = selector1;
   _value2 = selector2;
 }

 dynamic _value1;
 dynamic _value2;

 @override
 List get value => [_value1, _value2];
}
< . . . >
```

Структура: начало
Переиспользуемые компоненты

- BaseSelector и другие

Selector

```
class BaseSelector {
 dynamic _value;
 dynamic get value => _value;
}

class KeySelector extends BaseSelector {
 KeySelector(key) {
   _value = key;
 }
}

class TextSelector extends BaseSelector {
 TextSelector(String text) {
   _valueText = text;
 }
 String _valueText;
```

Selector

```
 @override
 String get value => _valueText;
}
class DescendantSelector extends BaseSelector {
 DescendantSelector(selector1, selector2) {
   _value1 = selector1;
   _value2 = selector2;
 }

 dynamic _value1;
 dynamic _value2;

 @override
 List get value => [_value1, _value2];
}
< . . . >
```

Структура: начало
Переиспользуемые компоненты

- BaseSelector и другие

Selector

```
class BaseSelector {
 dynamic _value;
 dynamic get value => _value;
}

class KeySelector extends BaseSelector {
 KeySelector(key) {
   _value = key;
 }
}

class TextSelector extends BaseSelector {
 TextSelector(String text) {
   _valueText = text;
 }
 String _valueText;
```

Selector

```
 @override
 String get value => _valueText;
}
class DescendantSelector extends BaseSelector {
 DescendantSelector(selector1, selector2) {
   _value1 = selector1;
   _value2 = selector2;
 }

 dynamic _value1;
 dynamic _value2;

 @override
 List get value => [_value1, _value2];
}
< . . . >
```

Структура: начало
Переиспользуемые компоненты

- BaseSelector и другие

Selector

```
class BaseSelector {
 dynamic _value;
 dynamic get value => _value;
}

class KeySelector extends BaseSelector {
 KeySelector(key) {
   _value = key;
 }
}

class TextSelector extends BaseSelector {
 TextSelector(String text) {
   _valueText = text;
 }
 String _valueText;
```

Selector

```
 @override
 String get value => _valueText;
}
class DescendantSelector extends BaseSelector {
 DescendantSelector(selector1, selector2) {
   _value1 = selector1;
   _value2 = selector2;
 }

 dynamic _value1;
 dynamic _value2;

 @override
 List get value => [_value1, _value2];
}
< . . . >
```

Структура: начало
Переиспользуемые компоненты

- Actor Widget

ActorWidget

```
class ActorWidget extends Actor {
 ActorWidget(WidgetTester actor) {
   this.actor = actor;
   finder = FinderWidget(); // from flutter_test package
 }

 Future toPump() async => await actor.pump();

 Future enterTextTo(selector, text) async =>
     await actor.enterText(finder.locate(selector), text.toString());

 dynamic getFinder(selector) => finder.locate(selector);
< . . . >
}
```

- Actor Driver

ActorDriver

```
class ActorDriver extends Actor {
 ActorDriver(FlutterDriver actor) {
   this.actor = actor;
   finder = FinderDriver(); // from flutter_driver package
 }

 Future enterText(text) async => await actor.enterText(text.toString());

Future enterTextTo(selector, text) async =>
     await actor.enterText(finder.locate(selector), text.toString());

Future runAsync(Future function, {Duration timeout}) async {
   return actor.runUnsynchronized(() async => function, timeout: timeout);
 }

< . . . >
}
```

Структура: начало
Переиспользуемые компоненты

- Actor Widget

ActorWidget

```
class ActorWidget extends Actor {
 ActorWidget(WidgetTester actor) {
   this.actor = actor;
   finder = FinderWidget(); // from flutter_test package
 }

 Future toPump() async => await actor.pump();

 Future enterTextTo(selector, text) async =>
     await actor.enterText(finder.locate(selector), text.toString());

 dynamic getFinder(selector) => finder.locate(selector);
< . . . >
}
```

- Actor Driver

ActorDriver

```
class ActorDriver extends Actor {
 ActorDriver(FlutterDriver actor) {
   this.actor = actor;
   finder = FinderDriver(); // from flutter_driver package
 }

 Future enterText(text) async => await actor.enterText(text.toString());

Future enterTextTo(selector, text) async =>
     await actor.enterText(finder.locate(selector), text.toString());

Future runAsync(Future function, {Duration timeout}) async {
   return actor.runUnsynchronized(() async => function, timeout: timeout);
 }

< . . . >
}
```

Структура: начало
Имплементация widget

widgets

```
group('Авторизация по телефону', () {
  testWidgets('Зона неАЗ, кнопка Войти присутствует',
    (WidgetTester tester) async {
    // given
    final actor = ActorWidget(tester);
    await AuthWidgets.initUnAuthorizedHeader(tester);
    // then
    expect(actor.getFinder(auth.loginBtn), findsOneWidget);
  });
  testWidgets('Зона неАЗ, работа кнопки Войти',
    (WidgetTester tester) async {
    // given
    final actor = ActorWidget(tester);
    await AuthWidgets.initUnAuthorizedHeader(tester);
    // then
    await actor.tap(auth.loginBtn);
    await actor.toPump();
    expect(actor.getFinder(auth.loginBtn), findsOneWidget);
  });
```

Структура: начало
Имплементация widget

widgets

```
group('Авторизация по телефону', () {
  testWidgets('Зона неАЗ, кнопка Войти присутствует',
    (WidgetTester tester) async {
    // given
    final actor = ActorWidget(tester);
    await AuthWidgets.initUnAuthorizedHeader(tester);
    // then
    expect(actor.getFinder(auth.loginBtn), findsOneWidget);
  });
  testWidgets('Зона неАЗ, работа кнопки Войти',
    (WidgetTester tester) async {
    // given
    final actor = ActorWidget(tester);
    await AuthWidgets.initUnAuthorizedHeader(tester);
    // then
    await actor.tap(auth.loginBtn);
    await actor.toPump();
    expect(actor.getFinder(auth.loginBtn), findsOneWidget);
  });
```

Структура: начало
Имплементация widget

widgets

```
group('Авторизация по телефону', () {
  testWidgets('Зона неАЗ, кнопка Войти присутствует',
    (WidgetTester tester) async {
    // given
    final actor = ActorWidget(tester);
    await AuthWidgets.initUnAuthorizedHeader(tester);
    // then
    expect(actor.getFinder(auth.loginBtn), findsOneWidget);
  });
  testWidgets('Зона неАЗ, работа кнопки Войти',
    (WidgetTester tester) async {
    // given
    final actor = ActorWidget(tester);
    await AuthWidgets.initUnAuthorizedHeader(tester);
    // then
    await actor.tap(auth.loginBtn);
    await actor.toPump();
    expect(actor.getFinder(auth.loginBtn), findsOneWidget);
  });
```

Flutter

flutter_test

Flutter
flutter_test

driver подход -> deprecated

Автотесты в Surf
flutter_test

Автотесты в Surf
flutter_test

- Gherkin

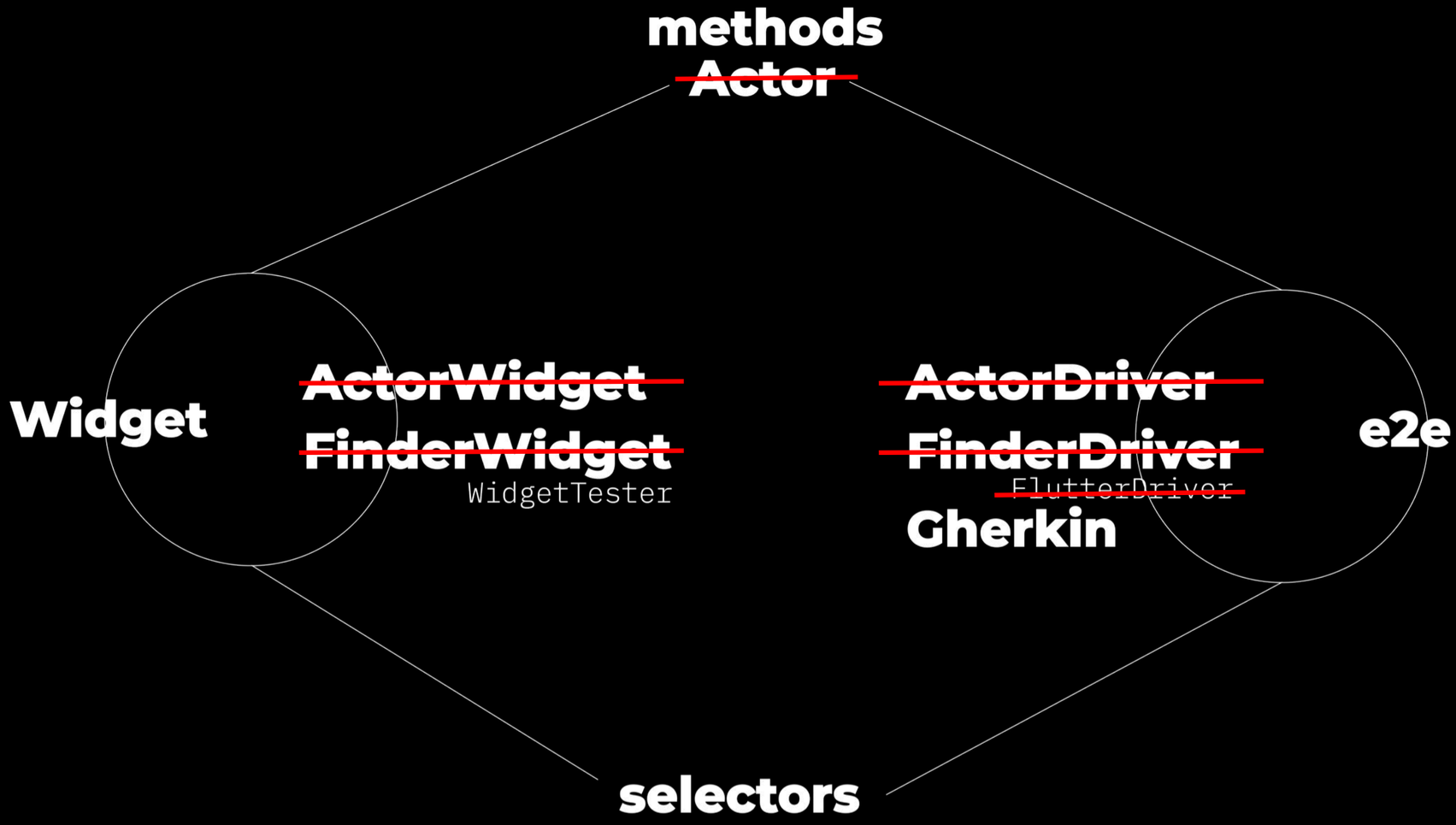
Автотесты в Surf
flutter_test

- Gherkin
- Переиспользуемые компоненты (e2e/widget)
  - селекторы
  - функции
  - жесты

Автотесты в Surf
flutter_test

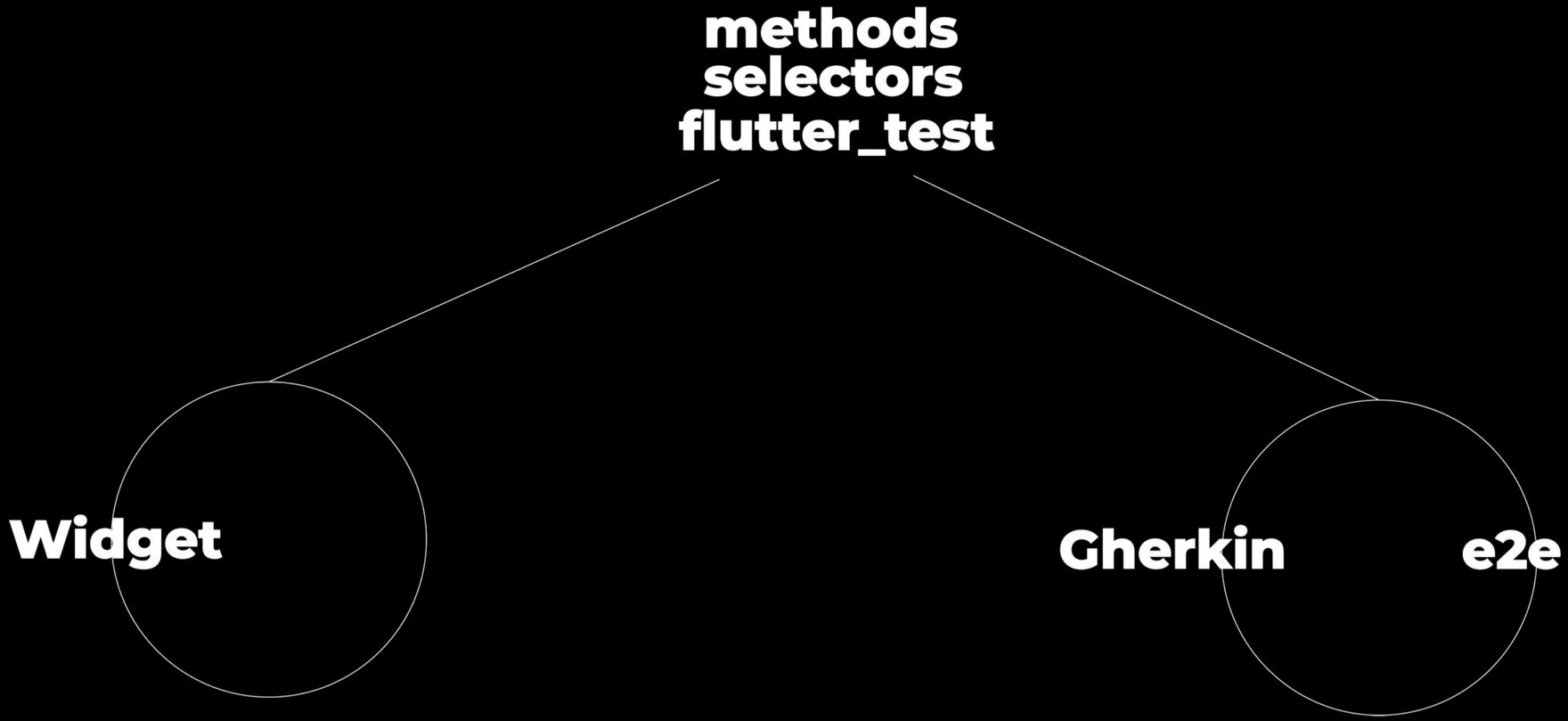
- Gherkin
- Переиспользуемые компоненты (e2e/widget)
  - селекторы
  - функции
  - жесты
- Отчеты

Автотесты в Surf
Структура: теперь



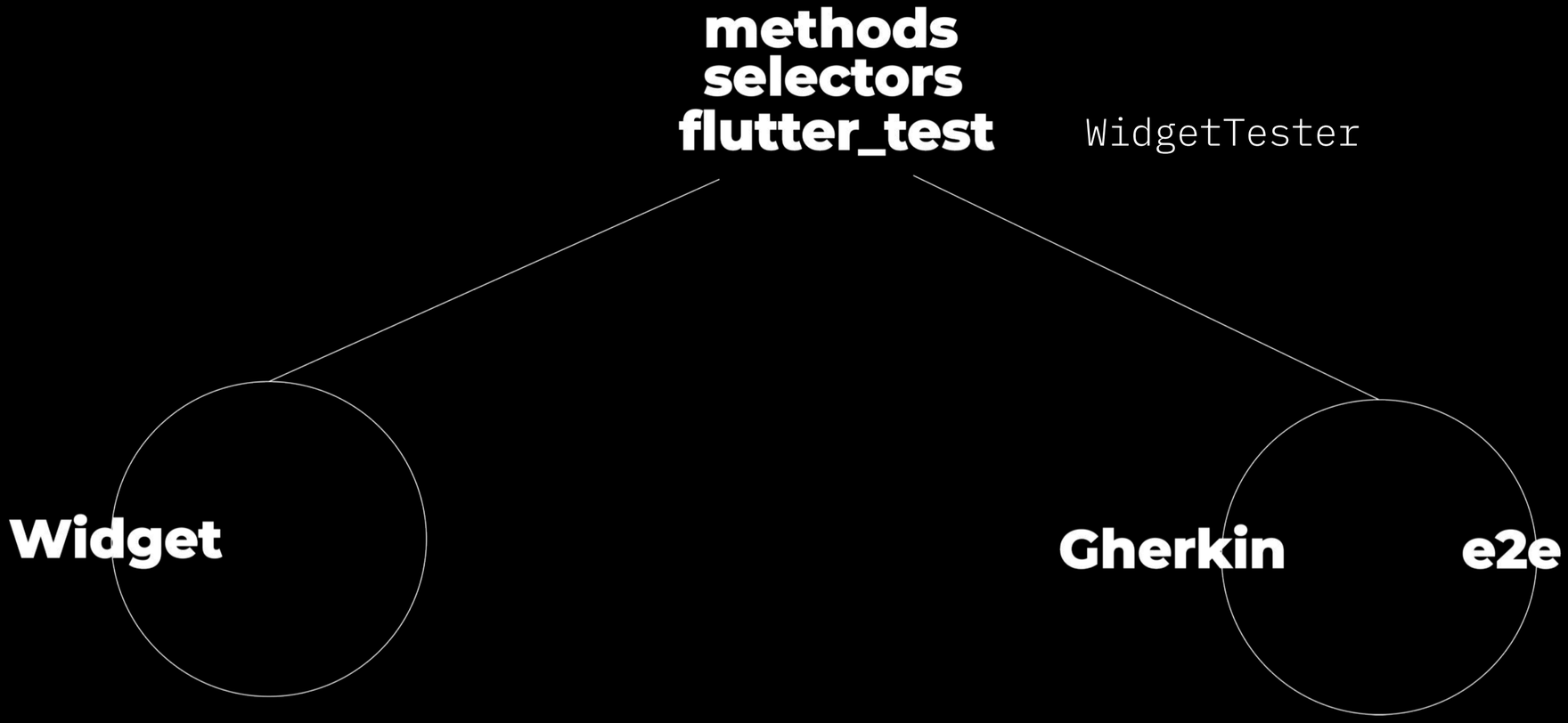
Автотесты в Surf

Структура: теперь



Автотесты в Surf

Структура: теперь



Автотесты в Surf
Структура: теперь

```
.
├── build.yaml
├── dart_test.yaml
├── integration_test
│   ├── credentials
│   │   ├── profiles
│   │   │   └── *_profile.dart
│   │   ├── credentials.dart
│   │   └── texts.dart
│   ├── features
│   │   └── *.feature
│   ├── gherkin
│   │   └── reports
│   │       └── gherkin_reports.json
```

Автотесты в Surf
Структура: теперь

```
.
├── build.yaml                                  - файл который содержит настройки для build_runner
├── dart_test.yaml
├── integration_test
│   ├── credentials
│   │   ├── profiles
│   │   │   └── *_profile.dart
│   │   ├── credentials.dart
│   │   └── texts.dart
│   ├── features
│   │   └── *.feature
│   ├── gherkin
│   │   └── reports
│   │       └── gherkin_reports.json
```

Автотесты в Surf
Структура: теперь

```
.
├── build.yaml                                  - файл который содержит настройки для build_runner
├── dart_test.yaml                              - файл конфигурации для виджет и юнит тестов
├── integration_test
│   ├── credentials
│   │   ├── profiles
│   │   │   └── *_profile.dart
│   │   ├── credentials.dart
│   │   └── texts.dart
│   ├── features
│   │   └── *.feature
│   ├── gherkin
│   │   └── reports
│   │       └── gherkin_reports.json
```

Автотесты в Surf
Структура: теперь

```
.
├── build.yaml                                  - файл который содержит настройки для build_runner
├── dart_test.yaml                              - файл конфигурации для виджет и юнит тестов
├── integration_test                            - папка в которой лежат тесты, сценарии, шаги итд
│   ├── credentials                             - папка с профилями *тестовых* аккаунтов и их данными
│   │   ├── profiles
│   │   │   └── *_profile.dart                  - файл профиля который задает соответствие между аккаунтом в сценарии и в данных
│   │   ├── credentials.dart
│   │   └── texts.dart
│   ├── features
│   │   └── *.feature
│   ├── gherkin
│   │   └── reports
│   │       └── gherkin_reports.json
```

Автотесты в Surf
Структура: теперь

```
.
├── build.yaml                                  - файл который содержит настройки для build_runner
├── dart_test.yaml                              - файл конфигурации для виджет и юнит тестов
├── integration_test                            - папка в которой лежат тесты, сценарии, шаги итд
│   ├── credentials                             - папка с профилями *тестовых* аккаунтов и их данными
│   │   ├── profiles
│   │   │   └── *_profile.dart                  - файл профиля который задает соответствие между аккаунтом в сценарии и в
данных
│   │   ├── credentials.dart                    - файл с тестовыми аккаунтами
│   │   └── texts.dart
│   ├── features
│   │   └── *.feature
│   ├── gherkin
│   │   └── reports
│   │       └── gherkin_reports.json
```

Автотесты в Surf
Структура: теперь

```
.
├── build.yaml                                  - файл который содержит настройки для build_runner
├── dart_test.yaml                              - файл конфигурации для виджет и юнит тестов
├── integration_test                            - папка в которой лежат тесты, сценарии, шаги итд
│   ├── credentials                             - папка с профилями *тестовых* аккаунтов и их данными
│   │   ├── profiles
│   │   │   └── *_profile.dart                  - файл профиля который задает соответствие между аккаунтом в сценарии и в данных

│   │   ├── credentials.dart                    - файл с тестовыми аккаунтами
│   │   └── texts.dart                          - файл с с текстами различных строк, снекбаров итд
│   ├── features
│   │   └── *.feature
│   ├── gherkin
│   │   └── reports
│   │       └── gherkin_reports.json
```

Автотесты в Surf
Структура: теперь

```
.
├── build.yaml                                  - файл который содержит настройки для build_runner
├── dart_test.yaml                              - файл конфигурации для виджет и юнит тестов
├── integration_test                            - папка в которой лежат тесты, сценарии, шаги итд
│   ├── credentials                             - папка с профилями *тестовых* аккаунтов и их данными
│   │   ├── profiles
│   │   │   └── *_profile.dart                  - файл профиля который задает соответствие между аккаунтом в сценарии и в данных
│   │   ├── credentials.dart                    - файл с тестовыми аккаунтами
│   │   └── texts.dart                          - файл с с текстами различных строк, снекбаров итд
│   ├── features
│   │   └── *.feature                           - файл с Gherkin сценариями
│   ├── gherkin
│   │   └── reports
│   │       └── gherkin_reports.json
```

Автотесты в Surf
Структура: теперь

```
.
├── build.yaml                                  - файл который содержит настройки для build_runner
├── dart_test.yaml                              - файл конфигурации для виджет и юнит тестов
├── integration_test                            - папка в которой лежат тесты, сценарии, шаги итд
│   ├── credentials                             - папка с профилями *тестовых* аккаунтов и их данными
│   │   ├── profiles
│   │   │   └── *_profile.dart                  - файл профиля который задает соответствие между аккаунтом в сценарии и в данных
│   │   ├── credentials.dart                    - файл с тестовыми аккаунтами
│   │   └── texts.dart                          - файл с с текстами различных строк, снекбаров итд
│   ├── features
│   │   └── *.feature                           - файл с Gherkin сценариями
│   ├── gherkin
│   │   └── reports
│   │       └── gherkin_reports.json            - файл отчета после прогона
```

Автотесты в Surf
Структура: теперь

```
|   ├── hooks
|   |   └── *_hook.dart                    - файл хука, который выполняется в определенное время в сценариях
|   ├── step_definitions
|   |   └── *_steps.dart
|   ├── worlds
|   |   └── *_world.dart
|   ├── gherkin_suite_test.dart
|   ├── gherkin_suite_test.g.dart
|   └── step_definitions_library.dart
├── lib
|   ├── ui
|   |   └── res
|   |       └── test_keys.dart
|   └── main.dart
```

Автотесты в Surf
Структура: теперь

```
|   ├── hooks
|   |   └── *_hook.dart                    - файл хука, который выполняется в определенное время в сценариях
|   ├── step_definitions
|   |   └── *_steps.dart                   - один из файлов с имплементациями шагов
|   ├── worlds
|   |   └── *_world.dart
|   ├── gherkin_suite_test.dart
|   ├── gherkin_suite_test.g.dart
|   └── step_definitions_library.dart
├── lib
|   ├── ui
|   |   └── res
|   |       └── test_keys.dart
|   └── main.dart
```

Автотесты в Surf
Структура: теперь

```
|   ├── hooks
|   |   └── *_hook.dart                    - файл хука, который выполняется в определенное время в сценариях
|   ├── step_definitions
|   |   └── *_steps.dart                   - один из файлов с имплементациями шагов
|   ├── worlds
|   |   └── *_world.dart                   - файл кастомного world, который является средой для сценария
|   ├── gherkin_suite_test.dart
|   ├── gherkin_suite_test.g.dart
|   └── step_definitions_library.dart
├── lib
|   ├── ui
|   |   └── res
|   |       └── test_keys.dart
|   └── main.dart
```

Автотесты в Surf
Структура: теперь

```
|   ├── hooks
|   |   └── *_hook.dart                    - файл хука, который выполняется в определенное время в сценариях
|   ├── step_definitions
|   |   └── *_steps.dart                   - один из файлов с имплементациями шагов
|   ├── worlds
|   |   └── *_world.dart                   - файл кастомного world, который является средой для сценария
|   ├── gherkin_suite_test.dart            - target для запуска flutter drive
|   ├── gherkin_suite_test.g.dart
|   └── step_definitions_library.dart
├── lib
|   ├── ui
|   |   └── res
|   |       └── test_keys.dart
|   └── main.dart
```

Автотесты в Surf
Структура: теперь

```
|   ├── hooks
|   |   └── *_hook.dart                    - файл хука, который выполняется в определенное время в сценариях
|   ├── step_definitions
|   |   └── *_steps.dart                   - один из файлов с имплементациями шагов
|   ├── worlds
|   |   └── *_world.dart                   - файл кастомного world, который является средой для сценария
|   ├── gherkin_suite_test.dart            - target для запуска flutter drive
|   ├── gherkin_suite_test.g.dart          - генерируемый build-runner файл
|   └── step_definitions_library.dart
├── lib
|   ├── ui
|   |   └── res
|   |       └── test_keys.dart
|   └── main.dart
```

Автотесты в Surf
Структура: теперь

```
|   ├── hooks
|   |   └── *_hook.dart                    - файл хука, который выполняется в определенное время в сценариях
|   ├── step_definitions
|   |   └── *_steps.dart                   - один из файлов с имплементациями шагов
|   ├── worlds
|   |   └── *_world.dart                   - файл кастомного world, который является средой для сценария
|   ├── gherkin_suite_test.dart            - target для запуска flutter drive
|   ├── gherkin_suite_test.g.dart          - генерируемый build-runner файл
|   └── step_definitions_library.dart      - файл для группировки шагов
├── lib
|   ├── ui
|   |   └── res
|   |       └── test_keys.dart
|   └── main.dart
```

Автотесты в Surf
Структура: теперь

```
|   ├── hooks
|   |   └── *_hook.dart                    - файл хука, который выполняется в определенное время в сценариях
|   ├── step_definitions
|   |   └── *_steps.dart                   - один из файлов с имплементациями шагов
|   ├── worlds
|   |   └── *_world.dart                   - файл кастомного world, который является средой для сценария
|   ├── gherkin_suite_test.dart            - target для запуска flutter drive
|   ├── gherkin_suite_test.g.dart          - генерируемый build-runner файл
|   └── step_definitions_library.dart      - файл для группировки шагов
├── lib                                    - папка с кодом приложения
|   ├── ui
|   |   └── res
|   |       └── test_keys.dart             - файл с ключами, которые определяют виджеты в тестах
|   └── main.dart
```

Автотесты в Surf
Структура: теперь

```
|   ├── hooks
|   |   └── *_hook.dart                    - файл хука, который выполняется в определенное время в сценариях
|   ├── step_definitions
|   |   └── *_steps.dart                   - один из файлов с имплементациями шагов
|   ├── worlds
|   |   └── *_world.dart                   - файл кастомного world, который является средой для сценария
|   ├── gherkin_suite_test.dart            - target для запуска flutter drive
|   ├── gherkin_suite_test.g.dart          - генерируемый build-runner файл
|   └── step_definitions_library.dart      - файл для группировки шагов
├── lib                                    - папка с кодом приложения
|   ├── ui
|   |   └── res
|   |       └── test_keys.dart             - файл с ключами, которые определяют виджеты в тестах
|   └── main.dart                          - файл приложение, его запуск запускает приложение
```

Автотесты в Surf
Структура: теперь

```
├── test                                  - папка с юнит и виджет тестами
|   ├── widget_test                        - папка с виджет тестами
|   └── tests                              - папка с юнит тестами
├── test_screens
|   ├── screens
|   |   └── *_screen.dart
|   ├── utils
|   |   └── *_utils.dart
|   ├── screens_library.dart
|   └── test_delays.dart
├── pubspec.yaml
└── test_driver
    └── integration_test.dart
```

Автотесты в Surf
Структура: теперь

```
├── test                                  - папка с юнит и виджет тестами
|   ├── widget_test                       - папка с виджет тестами
|   └── tests                             - папка с юнит тестами
├── test_screens                          - папка с локаторами виджетов и вынесенными методами
|   ├── screens
|   |   └── *_screen.dart                 - файл с локаторами экрана/группы экранов
|   ├── utils
|   |   └── *_utils.dart
|   ├── screens_library.dart
|   └── test_delays.dart
├── pubspec.yaml
└── test_driver
    └── integration_test.dart
```

Автотесты в Surf
Структура: теперь

```
├── test                                  - папка с юнит и виджет тестами
|   ├── widget_test                       - папка с виджет тестами
|   └── tests                             - папка с юнит тестами
├── test_screens                          - папка с локаторами виджетов и вынесенными методами
|   ├── screens
|   |   └── *_screen.dart                 - файл с локаторами экрана/группы экранов
|   ├── utils
|   |   └── *_utils.dart                  - файл с методами экрана/группы экранов
|   ├── screens_library.dart
|   └── test_delays.dart
├── pubspec.yaml
└── test_driver
    └── integration_test.dart
```

Автотесты в Surf
Структура: теперь

```
├── test                                  - папка с юнит и виджет тестами
|   ├── widget_test                       - папка с виджет тестами
|   └── tests                             - папка с юнит тестами
├── test_screens                          - папка с локаторами виджетов и вынесенными методами
|   ├── screens
|   |   └── *_screen.dart                 - файл с локаторами экрана/группы экранов
|   ├── utils
|   |   └── *_utils.dart                  - файл с методами экрана/группы экранов
|   ├── screens_library.dart              - файл для группировки экранов
|   └── test_delays.dart
├── pubspec.yaml
└── test_driver
    └── integration_test.dart
```

Автотесты в Surf
Структура: теперь

```
├── test                                  - папка с юнит и виджет тестами
|   ├── widget_test                       - папка с виджет тестами
|   └── tests                             - папка с юнит тестами
├── test_screens                          - папка с локаторами виджетов и вынесенными методами
|   ├── screens
|   |   └── *_screen.dart                 - файл с локаторами экрана/группы экранов
|   ├── utils
|   |   └── *_utils.dart                  - файл с методами экрана/группы экранов
|   ├── screens_library.dart              - файл для группировки экранов
|   └── test_delays.dart                  - файл с вынесенными Duration для переиспользования
├── pubspec.yaml
└── test_driver
    └── integration_test.dart
```

Автотесты в Surf
Структура: теперь

```
├── test                                  - папка с юнит и виджет тестами
|   ├── widget_test                       - папка с виджет тестами
|   └── tests                             - папка с юнит тестами
├── test_screens                          - папка с локаторами виджетов и вынесенными методами
|   ├── screens
|   |   └── *_screen.dart                 - файл с локаторами экрана/группы экранов
|   ├── utils
|   |   └── *_utils.dart                  - файл с методами экрана/группы экранов
|   ├── screens_library.dart              - файл для группировки экранов
|   └── test_delays.dart                  - файл с вынесенными Duration для переиспользования
├── pubspec.yaml                          - файл с зависимостями проекта
└── test_driver
    └── integration_test.dart
```

Автотесты в Surf
Структура: теперь

```
├── test                                  - папка с юнит и виджет тестами
|   ├── widget_test                       - папка с виджет тестами
|   └── tests                             - папка с юнит тестами
├── test_screens                          - папка с локаторами виджетов и вынесенными методами
|   ├── screens
|   |   └── *_screen.dart                 - файл с локаторами экрана/группы экранов
|   ├── utils
|   |   └── *_utils.dart                  - файл с методами экрана/группы экранов
|   ├── screens_library.dart              - файл для группировки экранов
|   └── test_delays.dart                  - файл с вынесенными Duration для переиспользования
├── pubspec.yaml                          - файл с зависимостями проекта
└── test_driver
    └── integration_test.dart             - driver для запуска flutter drive
```

Структура: теперь **Gherkin**

- `flutter_gherkin`

gherkin

```gherkin
#language: ru
Функциональность: Авторизация
  @auth
  Сценарий: Авто: Авторизация с корректным ОТП
    Когда Я запускаю приложение
    И Я перехожу на таб Библиотека
    И Я тапаю на кнопку Войти
    И Я ввожу рандомный телефон
    И Я тапаю на кнопку далее
    И Я ввожу ОТП код "12345"
    Тогда Я вижу таб Библиотека авторизанта

< . . . >
```

Структура: теперь **Gherkin**

- `flutter_gherkin + integration_test`

Структура: теперь **Gherkin**

- `flutter_gherkin + integration_test`
  - `после правок gherkin-сценариев нужно выполнять code-gen`

Структура: теперь
Имплементация шагов e2e

e2e

```
class AuthSteps {
when<ContextualWorld>(
        RegExp(r'Я тапаю на кнопку Войти'),
        (context) async {
          final tester = context.world.appDriver.rawDriver;
          await tester.implicitTap(AuthScreen.loginBtn);
          await tester.pumpAndSettle();

        },
      ),
when1<String, ContextualWorld>(
        RegExp(r'Я ввожу ОТП код {string}$'),
        (code, context) async {
          final tester = context.world.appDriver.rawDriver;
   await tester.pumpUntilVisible(AuthScreen.otpCreateScreen);
          await tester.enterOtp(code, AuthScreen.otpFieldNoError);
          await tester.pumpUntilVisible(AuthScreen.otpRetryScreen);
        },
      ),
< . . . >
```

Структура: теперь
Имплементация шагов e2e

e2e

```
class AuthSteps {
when<ContextualWorld>(
        RegExp(r'Я тапаю на кнопку Войти'),
        (context) async {
          final tester = context.world.appDriver.rawDriver;
          await tester.implicitTap(AuthScreen.loginBtn);
          await tester.pumpAndSettle();

        },
      ),
when1<String, ContextualWorld>(
        RegExp(r'Я ввожу ОТП код {string}$'),
        (code, context) async {
          final tester = context.world.appDriver.rawDriver;
   await tester.pumpUntilVisible(AuthScreen.otpCreateScreen);
          await tester.enterOtp(code, AuthScreen.otpFieldNoError);
          await tester.pumpUntilVisible(AuthScreen.otpRetryScreen);
        },
      ),
< . . . >
```

Структура: теперь
Имплементация шагов e2e

e2e

```
class AuthSteps {
when<ContextualWorld>(
        RegExp(r'Я тапаю на кнопку Войти'),
        (context) async {
          final tester = context.world.appDriver.rawDriver;
          await tester.implicitTap(AuthScreen.loginBtn);
          await tester.pumpAndSettle();

        },
      ),
when1<String, ContextualWorld>(
        RegExp(r'Я ввожу ОТП код {string}$'),
        (code, context) async {
          final tester = context.world.appDriver.rawDriver;
  await tester.pumpUntilVisible(AuthScreen.otpCreateScreen);
          await tester.enterOtp(code, AuthScreen.otpFieldNoError);
          await tester.pumpUntilVisible(AuthScreen.otpRetryScreen);
        },
      ),
< . . . >
```

Структура: теперь
Имплементация шагов e2e

e2e

```
class AuthSteps {
when<ContextualWorld>(
        RegExp(r'Я тапаю на кнопку Войти'),
        (context) async {
          final tester = context.world.appDriver.rawDriver;
          await tester.implicitTap(AuthScreen.loginBtn);
          await tester.pumpAndSettle();

        },
      ),
when1<String, ContextualWorld>(
        RegExp(r'Я ввожу ОТП код {string}$'),
        (code, context) async {
          final tester = context.world.appDriver.rawDriver;
   await tester.pumpUntilVisible(AuthScreen.otpCreateScreen);
          await tester.enterOtp(code, AuthScreen.otpFieldNoError);
          await tester.pumpUntilVisible(AuthScreen.otpRetryScreen);
        },
      ),
< . . . >
```

Структура: теперь
Имплементация шагов e2e

e2e

```
class AuthSteps {
when<ContextualWorld>(
        RegExp(r'Я тапаю на кнопку Войти'),
        (context) async {
          final tester = context.world.appDriver.rawDriver;
          await tester.implicitTap(AuthScreen.loginBtn);
          await tester.pumpAndSettle();

        },
      ),
when1<String, ContextualWorld>(
        RegExp(r'Я ввожу ОТП код {string}$'),
        (code, context) async {
          final tester = context.world.appDriver.rawDriver;
   await tester.pumpUntilVisible(AuthScreen.otpCreateScreen);
          await tester.enterOtp(code, AuthScreen.otpFieldNoError);
          await tester.pumpUntilVisible(AuthScreen.otpRetryScreen);
        },
      ),
< . . . >
```

Структура: теперь
Имплементация шагов e2e

e2e

```
class AuthSteps {
when<ContextualWorld>(
        RegExp(r'Я тапаю на кнопку Войти'),
        (context) async {
          final tester = context.world.appDriver.rawDriver;
          await tester.implicitTap(AuthScreen.loginBtn);
          await tester.pumpAndSettle();

        },
      ),
when1<String, ContextualWorld>(
        RegExp(r'Я ввожу ОТП код {string}$'),
        (code, context) async {
          final tester = context.world.appDriver.rawDriver;
   await tester.pumpUntilVisible(AuthScreen.otpCreateScreen);
          await tester.enterOtp(code, AuthScreen.otpFieldNoError);
          await tester.pumpUntilVisible(AuthScreen.otpRetryScreen);
        },
      ),
< . . . >
```

регулярные выражения

Структура: теперь
Имплементация шагов e2e

e2e

```
class AuthSteps {
when<ContextualWorld>(
        RegExp(r'Я тапаю на кнопку Войти'),
        (context) async {
          final tester = context.world.appDriver.rawDriver;
          await tester.implicitTap(AuthScreen.loginBtn);
          await tester.pumpAndSettle();

        },
      ),
when1<String, ContextualWorld>(
        RegExp(r'Я ввожу ОТП код {string}$'),
        (code, context) async {
          final tester = context.world.appDriver.rawDriver;
   await tester.pumpUntilVisible(AuthScreen.otpCreateScreen);
          await tester.enterOtp(code, AuthScreen.otpFieldNoError);
          await tester.pumpUntilVisible(AuthScreen.otpRetryScreen);
        },
      ),
< . . . >
```

регулярные выражения

$ - говорит, что это конец строки,
без $ есть шанс,
что шаги с одинаковым началом
перепутаются

Структура: теперь
Имплементация widget

widgets

```
group('Авторизация по телефону', () {
  testWidgets('Зона неАЗ, кнопка Войти присутствует',
    (WidgetTester tester) async {
    // given
    await tester.pumpWidget(GeneralWidgetInit.defaultWrapper(
       const AuthLoginScreen(),
     ));
    // then
    expect(AuthScreen.loginBtn, findsOneWidget);
  });
  testWidgets('Зона неАЗ, работа кнопки Войти',
    (WidgetTester tester) async {
    // given
    await tester.pumpWidget(GeneralWidgetInit.defaultWrapper(
       const AuthLoginScreen(),
     ));
    // then
    await tester.tap(AuthScreen.loginBtn);
    await tester.pumpAndSettle();
    expect(AuthScreen.loginBtn, findsOneWidget);
  });
```

widgets

```
  testWidgets('Максимальная длина поля 11 символов', (WidgetTester tester) async {
// given
    await tester.pumpWidget(GeneralWidgetInit.defaultWrapper(
      const AuthLoginScreen(),
    ));
    await tester.pumpAndSettle();
    await tester.doEnterText(AuthScreen.phoneFld, randomNumber(12));
// then
    final field = tester.widget<TextField>(AuthScreen.phoneFld);
    expect(field.controller.text.length, 11);
  });
< . . . >
 }); // group('Авторизация по телефону')
}
< . . . >
```

Структура: теперь
Имплементация widget

widgets

```
group('Авторизация по телефону', () {
  testWidgets('Зона неАЗ, кнопка Войти присутствует',
    (WidgetTester tester) async {
    // given
    await tester.pumpWidget(GeneralWidgetInit.defaultWrapper(
       const AuthLoginScreen(),
     ));
    // then
    expect(AuthScreen.loginBtn, findsOneWidget);
  });
  testWidgets('Зона неАЗ, работа кнопки Войти',
    (WidgetTester tester) async {
    // given
    await tester.pumpWidget(GeneralWidgetInit.defaultWrapper(
       const AuthLoginScreen(),
     ));
    // then
    await tester.tap(AuthScreen.loginBtn);
    await tester.pumpAndSettle();
    expect(AuthScreen.loginBtn, findsOneWidget);
  });
```

widgets

```
  testWidgets('Максимальная длина поля 11 символов', (WidgetTester tester) async {
// given
    await tester.pumpWidget(GeneralWidgetInit.defaultWrapper(
      const AuthLoginScreen(),
    ));
    await tester.pumpAndSettle();
    await tester.doEnterText(AuthScreen.phoneFld, randomNumber(12));
// then
    final field = tester.widget<TextField>(AuthScreen.phoneFld);
    expect(field.controller.text.length, 11);
  });
< . . . >
 }); // group('Авторизация по телефону')
}
< . . . >
```

Структура: теперь
Имплементация widget

widgets

```
group('Авторизация по телефону', () {
  testWidgets('Зона неАЗ, кнопка Войти присутствует',
    (WidgetTester tester) async {
    // given
    await tester.pumpWidget(GeneralWidgetInit.defaultWrapper(
       const AuthLoginScreen(),
     ));
    // then
    expect(AuthScreen.loginBtn, findsOneWidget);
  });
  testWidgets('Зона неАЗ, работа кнопки Войти',
    (WidgetTester tester) async {
    // given
    await tester.pumpWidget(GeneralWidgetInit.defaultWrapper(
       const AuthLoginScreen(),
     ));
    // then
    await tester.tap(AuthScreen.loginBtn);
    await tester.pumpAndSettle();
    expect(AuthScreen.loginBtn, findsOneWidget);
  });
```

widgets

```
  testWidgets('Максимальная длина поля 11 символов', (WidgetTester tester) async {
// given
    await tester.pumpWidget(GeneralWidgetInit.defaultWrapper(
      const AuthLoginScreen(),
    ));
    await tester.pumpAndSettle();
    await tester.doEnterText(AuthScreen.phoneFld, randomNumber(12));
// then
    final field = tester.widget<TextField>(AuthScreen.phoneFld);
    expect(field.controller.text.length, 11);
  });
< . . . >
 }); // group('Авторизация по телефону')
}
< . . . >
```

Структура: теперь
Переиспользуемые компоненты

- Селекторы, общие функции, жесты

selectors

```
class AuthScreen {
 // Экран ввода пин-кода при авторизации
 static Finder pinScreen = find.byKey(AuthTestKeys.pinScreen);

 // Экран задания пин-кода
 static Finder pinCreateScreen = find.byKey(AuthTestKeys.pinCreateScreen);

 // Элемент пин-клавиатуры при вводе пин-кода при авторизации
 static Finder pinNumberLogin(String number) =>
     find.descendant(of: pinScreen, matching:
find.byKey(AuthTestKeys.pinKeyboardBtn(number)));

 // Поле Логин на экране авторизации
 static Finder loginField = find.byKey(AuthTestKeys.loginField);

< . . . >
}
```

Структура: теперь
Переиспользуемые компоненты

- Ключи

keys

```
abstract class AuthTestKeys
{
  /// ключ поля логина на экране авторизации
static const Key loginField = Key('fld_auth_login');
}

class AuthScreen {
  /// поле Логин на экране авторизации
  static Finder loginField = find.byKey(AuthTestKeys.loginField);
}
```

Структура: теперь
Переиспользуемые компоненты

- Функции

functions

```
extension AuthExtendedWidgetTester on WidgetTester {
  /// вводим пин в переданый Finder, при этом Finder
  // пин-кода это функция которая в зависимости от
  // экрана и цифры пина возвращает Finder с ключом,
  // поэтому и передаем не Finder а функцию которая
  // позволяет получить Finder

  Future<void> enterPin(String pin, Finder Function(String)
    pinNumber) async {
    final streamPin = Stream.fromIterable(pin.split(''));
    await pumpForDuration(const Duration(milliseconds: 500));
    await for (final String ch in streamPin) {
      await implicitTap(pinNumber(ch));
    }
  }
}
```

Структура: теперь
Переиспользуемые компоненты

- Функции

functions

```
extension AuthExtendedWidgetTester on WidgetTester {
  /// вводим пин в переданый Finder, при этом Finder
  // пин-кода это функция которая в зависимости от
  // экрана и цифры пина возвращает Finder с ключом,
  // поэтому и передаем не Finder а функцию которая
  // позволяет получить Finder

  Future<void> enterPin(String pin, Finder Function(String)
    pinNumber) async {
    final streamPin = Stream.fromIterable(pin.split(''));
    await pumpForDuration(const Duration(milliseconds: 500));
    await for (final String ch in streamPin) {
      await implicitTap(pinNumber(ch));
    }
  }
}
```

Структура: теперь
Переиспользуемые компоненты

- Жесты

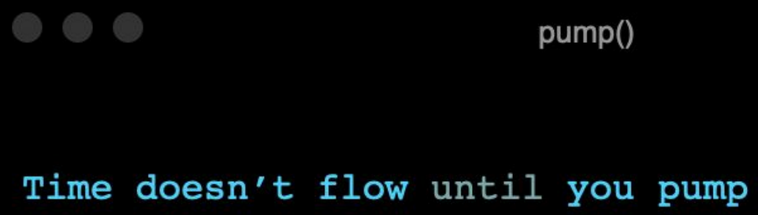
gestures

```
Offset scrollDown = Offset(0, -120)

abstract class GeneralGestures {
  /// Направление для scroll - направление движения списка
  /// Направление для flick/swipe etc - направление движения пальца
  // жесты для скролла экранов вниз
  static const Offset scrollDown = Offset(0, -120);
  static const Offset flickUp = Offset(0, -600);

  // жесты для скролла экранов вверх
  static const Offset scrollUp = Offset(0, 120);
  static const Offset flickDown = Offset(0, 600);
}
```

Структура: теперь
Удобства

Структура: теперь
Удобства: pump

- pump()



Структура: теперь
Удобства: pump

- pump() vs pumpAndSettle()

```
pump — запускает обработку смены состояния виджета и ожидает ее завершения в течении
заданного таймаута;

pumpAndSettle — вызывает pump в цикле для смены состояний в течении заданного таймаута,
это ожидание завершения всех анимаций;
```

Структура: теперь
Удобства: pump

- custom - pumpUnitilCondition()

functions

```dart
/// Метод для того, чтобы делать pump пока не произойдет условие [condition]
Future<bool> pumpUntilCondition(bool Function() condition,
    {Duration timeout = _defaultPumpTimeout}) async {
  final times = (timeout.inMicroseconds / _minimalPumpDelay.inMicroseconds).ceil();
  for (var i = 0; i < times; i++) {
    if (condition()) {
      await pumpForDuration(_minimalInteractionDelay);
      return true;
    }
    await pump(_minimalPumpDelay);
  }
  return false;
}
```

Структура: теперь
Удобства: pump

- custom - pumpUnitilCondition()

functions

```dart
/// Метод для того, чтобы делать pump пока не произойдет условие [condition]
Future<bool> pumpUntilCondition(bool Function() condition,
    {Duration timeout = _defaultPumpTimeout}) async {
  final times = (timeout.inMicroseconds / _minimalPumpDelay.inMicroseconds).ceil();
  for (var i = 0; i < times; i++) {
    if (condition()) {
      await pumpForDuration(_minimalInteractionDelay);
      return true;
    }
    await pump(_minimalPumpDelay);
  }
  return false;
}
```

Структура: теперь
Удобства: pump

- custom - pumpUnitilCondition()

functions

```dart
/// Метод для того, чтобы делать pump пока не произойдет условие [condition]
Future<bool> pumpUntilCondition(bool Function() condition,
    {Duration timeout = _defaultPumpTimeout}) async {
  final times = (timeout.inMicroseconds / _minimalPumpDelay.inMicroseconds).ceil();
  for (var i = 0; i < times; i++) {
    if (condition()) {
      await pumpForDuration(_minimalInteractionDelay);
      return true;
    }
    await pump(_minimalPumpDelay);
  }
  return false;
}
```

Структура: теперь
Удобства: pump

- custom - pumpUnitilCondition()

functions

```dart
/// Метод для того, чтобы делать pump пока не произойдет условие [condition]
Future<bool> pumpUntilCondition(bool Function() condition,
    {Duration timeout = _defaultPumpTimeout}) async {
  final times = (timeout.inMicroseconds / _minimalPumpDelay.inMicroseconds).ceil();
  for (var i = 0; i < times; i++) {
    if (condition()) {
      await pumpForDuration(_minimalInteractionDelay);
      return true;
    }
    await pump(_minimalPumpDelay);
  }
  return false;
}
```

Plan the delay in advance!

Структура: теперь
Удобства: tap

- custom - implicitTap()

functions

```
Future<void> implicitTap(Finder finder, {Duration duration}) async {
    final found = await pumpUntilVisible(finder, duration: duration
                                         ?? _defaultPumpTimeout);
    if (!found) {
      // ignore: only_throw_errors
      throw TestFailure(finder.toString());
    }
```

Структура: теперь **Удобства: tap**

- custom - implicitTap()

functions

```
Future<void> implicitTap(Finder finder, {Duration duration}) async {
    final found = await pumpUntilVisible(finder, duration: duration
                                         ?? _defaultPumpTimeout);
    if (!found) {
      // ignore: only_throw_errors
      throw TestFailure(finder.toString());
    }
```

Структура: теперь
Удобства: context

- getContext() / setContext()

functions

```
class ContextualWorld extends FlutterWidgetTesterWorld {
 Map<String, dynamic> scenarioContext = <String, dynamic>{};

 @override
 void dispose() {
   super.dispose();
   scenarioContext.clear();
 }

 T getContext<T>(String key) {
   return scenarioContext[key] as T;
 }

 void setContext(String key, dynamic value) {
   scenarioContext[key] = value;
 }

 Future<void> attachScreenshot() async {
   final bytes = await appDriver.screenshot();
   attach(base64Encode(bytes), 'image/png');
 }
}
```

Структура: теперь
Удобства: context

- getContext() / setContext()

functions

```
class ContextualWorld extends FlutterWidgetTesterWorld {
 Map<String, dynamic> scenarioContext = <String, dynamic>{};

 @override
 void dispose() {
   super.dispose();
   scenarioContext.clear();
 }

 T getContext<T>(String key) {
   return scenarioContext[key] as T;
 }

 void setContext(String key, dynamic value) {
   scenarioContext[key] = value;
 }

 Future<void> attachScreenshot() async {
   final bytes = await appDriver.screenshot();
   attach(base64Encode(bytes), 'image/png');
 }
}
```

Структура: теперь
Удобства: context

- getContext() / setContext()

functions

```
class ContextualWorld extends FlutterWidgetTesterWorld {
  Map<String, dynamic> scenarioContext = <String, dynamic>{};

  @override
  void dispose() {
    super.dispose();
    scenarioContext.clear();
  }

  T getContext<T>(String key) {
    return scenarioContext[key] as T;
  }

  void setContext(String key, dynamic value) {
    scenarioContext[key] = value;
  }

  Future<void> attachScreenshot() async {
    final bytes = await appDriver.screenshot();
    attach(base64Encode(bytes), 'image/png');
  }
}
```

Структура: теперь
Удобства: context

- getContext() / setContext()

functions

```
class ContextualWorld extends FlutterWidgetTesterWorld {
 Map<String, dynamic> scenarioContext = <String, dynamic>{};

 @override
 void dispose() {
   super.dispose();
   scenarioContext.clear();
 }

 T getContext<T>(String key) {
   return scenarioContext[key] as T;
 }

 void setContext(String key, dynamic value) {
   scenarioContext[key] = value;
 }

 Future<void> attachScreenshot() async {
   final bytes = await appDriver.screenshot();
   attach(base64Encode(bytes), 'image/png');
 }
}
```

Структура: теперь
Удобства: context

- getContext() / setContext()

functions

```
class ContextualWorld extends FlutterWidgetTesterWorld {
 Map<String, dynamic> scenarioContext = <String, dynamic>{};

 @override
 void dispose() {
   super.dispose();
   scenarioContext.clear();
 }

 T getContext<T>(String key) {
   return scenarioContext[key] as T;
 }

 void setContext(String key, dynamic value) {
   scenarioContext[key] = value;
 }

 Future<void> attachScreenshot() async {
   final bytes = await appDriver.screenshot();
   attach(base64Encode(bytes), 'image/png');
 }
}
```

Структура: теперь **Удобства: context**

- `getContext() / setContext()`
  - `Я использую аккаунт “account”`

Структура: теперь
Удобства: context

- `getContext() / setContext()`
  - `Я использую аккаунт “account” <- запоминает user'a`

Структура: теперь
Удобства: context

- getContext() / setContext()
  - Я использую аккаунт “account” <- запоминает user'а
  - ~~Я ввожу пароль юзера “user”, Я ввожу ОТП юзера “user”, и тп~~

Структура: теперь

Удобства: hooks

Структура: теперь
Удобства: hooks

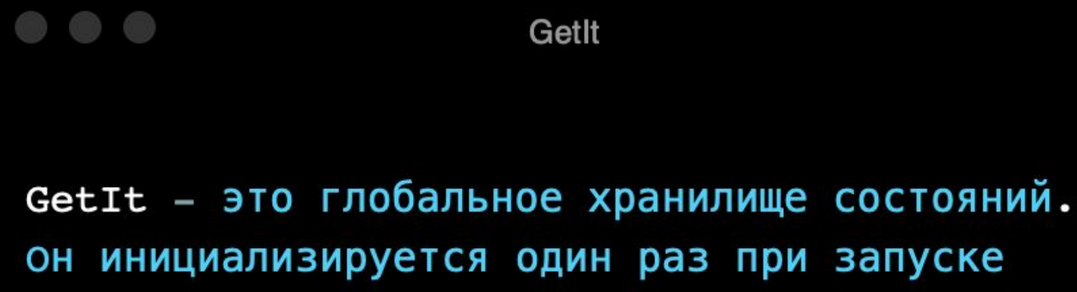
- сброс состояния приложения

Структура: теперь
Удобства: hooks

- сброс состояния приложения
  - onBeforeScenario сбрасывает GetIt перед сценарием

Структура: теперь **Удобства: hooks**

- сброс состояния приложения
  - onBeforeScenario сбрасывает GetIt перед сценарием



Структура: теперь
Удобства: hooks

- сброс состояния приложения
  - onBeforeScenario сбрасывает GetIt перед сценарием
  - onAfterScenarioWorldCreated сбрасывает все хранилища приложения после сценария

Структура: теперь
Удобства: hooks

- сброс состояния приложения
  - onBeforeScenario сбрасывает GetIt перед сценарием
  - onAfterScenarioWorldCreated сбрасывает все хранилища приложения после сценария
- скриншот при падении теста

Структура: теперь
Удобства: hooks

- сброс состояния приложения
  - onBeforeScenario сбрасывает GetIt перед сценарием
  - onAfterScenarioWorldCreated сбрасывает все хранилища приложения после сценария
- скриншот при падении теста
- перезапуск приложения

Структура: теперь
Удобства: hooks

- сброс состояния приложения
  - onBeforeScenario сбрасывает GetIt перед сценарием
  - onAfterScenarioWorldCreated сбрасывает все хранилища приложения после сценария
- скриншот при падении теста
- перезапуск приложения

hooks

```
when<ContextualWorld>(
      RegExp(r'Я перезапускаю приложение$'),
          (context) async {
        final navigator = devGetIt<GlobalKey<NavigatorState>>().currentState;
        unawaited(navigator.pushAndRemoveUntil(SplashScreenRoute(), (_) => false));
        final tester = context.world.appDriver.rawDriver;
        await tester.pumpUntilVisibleAny([AuthScreen.loginField, AuthScreen.pinScreen]);
      },
    ),
```

Структура: теперь
Удобства: hooks

- сброс состояния приложения
  - onBeforeScenario сбрасывает GetIt перед сценарием
  - onAfterScenarioWorldCreated сбрасывает все хранилища приложения после сценария
- скриншот при падении теста
- перезапуск приложения

hooks

```
when<ContextualWorld>(
      RegExp(r'Я перезапускаю приложение$'),
          (context) async {
        final navigator = devGetIt<GlobalKey<NavigatorState>>().currentState;
        unawaited(navigator.pushAndRemoveUntil(SplashScreenRoute(), (_) => false));
        final tester = context.world.appDriver.rawDriver;
        await tester.pumpUntilVisibleAny([AuthScreen.loginField, AuthScreen.pinScreen]);
      },
    ),
```

Структура: теперь
Удобства: hooks

- сброс состояния приложения
  - onBeforeScenario сбрасывает GetIt перед сценарием
  - onAfterScenarioWorldCreated сбрасывает все хранилища приложения после сценария
- скриншот при падении теста
- перезапуск приложения

hooks

```
when<ContextualWorld>(
      RegExp(r'Я перезапускаю приложение$'),
          (context) async {
        final navigator = devGetIt<GlobalKey<NavigatorState>>().currentState;
        unawaited(navigator.pushAndRemoveUntil(SplashScreenRoute(), (_) => false));
        final tester = context.world.appDriver.rawDriver;
        await tester.pumpUntilVisibleAny([AuthScreen.loginField, AuthScreen.pinScreen]);
      },
    ),
```

Структура: теперь
Удобства: hooks

- сброс состояния приложения
  - onBeforeScenario сбрасывает GetIt перед сценарием
  - onAfterScenarioWorldCreated сбрасывает все хранилища приложения после сценария
- скриншот при падении теста
- перезапуск приложения

hooks

```
when<ContextualWorld>(
      RegExp(r'Я перезапускаю приложение$'),
          (context) async {
        final navigator = devGetIt<GlobalKey<NavigatorState>>().currentState;
        unawaited(navigator.pushAndRemoveUntil(SplashScreenRoute(), (_) => false));
        final tester = context.world.appDriver.rawDriver;
        await tester.pumpUntilVisibleAny([AuthScreen.loginField, AuthScreen.pinScreen]);
      },
    ),
```

Структура: теперь

Удобства: steps

Структура: теперь
Удобства: steps

- `StepDefinitionGeneric`

Структура: теперь
Удобства: steps

- `StepDefinitionGeneric -> делим шаги на экраны`

Структура: теперь **Удобства: steps**

- StepDefinitionGeneric -> делим шаги на экраны

steps

```
static Iterable<StepDefinitionGeneric> get steps => [
        ..._SeeProductDetails.steps,
        <otherStepsHere>,
]
---
/// Шаги "Я вижу" детальных экранов продуктов
class _SeeProductDetails {
 static final Iterable<StepDefinitionGeneric> steps = [
   then<ContextualWorld>(
     RegExp(r'Я вижу
```

Структура: теперь **Удобства: steps**

- StepDefinitionGeneric -> делим шаги на экраны

steps

```
static Iterable<StepDefinitionGeneric> get steps => [
        ..._SeeProductDetails.steps,
        <otherStepsHere>,
]
---
/// Шаги "Я вижу" детальных экранов продуктов
class _SeeProductDetails {
 static final Iterable<StepDefinitionGeneric> steps = [
   then<ContextualWorld>(
     RegExp(r'Я вижу
```

Структура: теперь **Удобства: steps**

- StepDefinitionGeneric -> делим шаги на экраны -> step_definitions_library.dart

steps

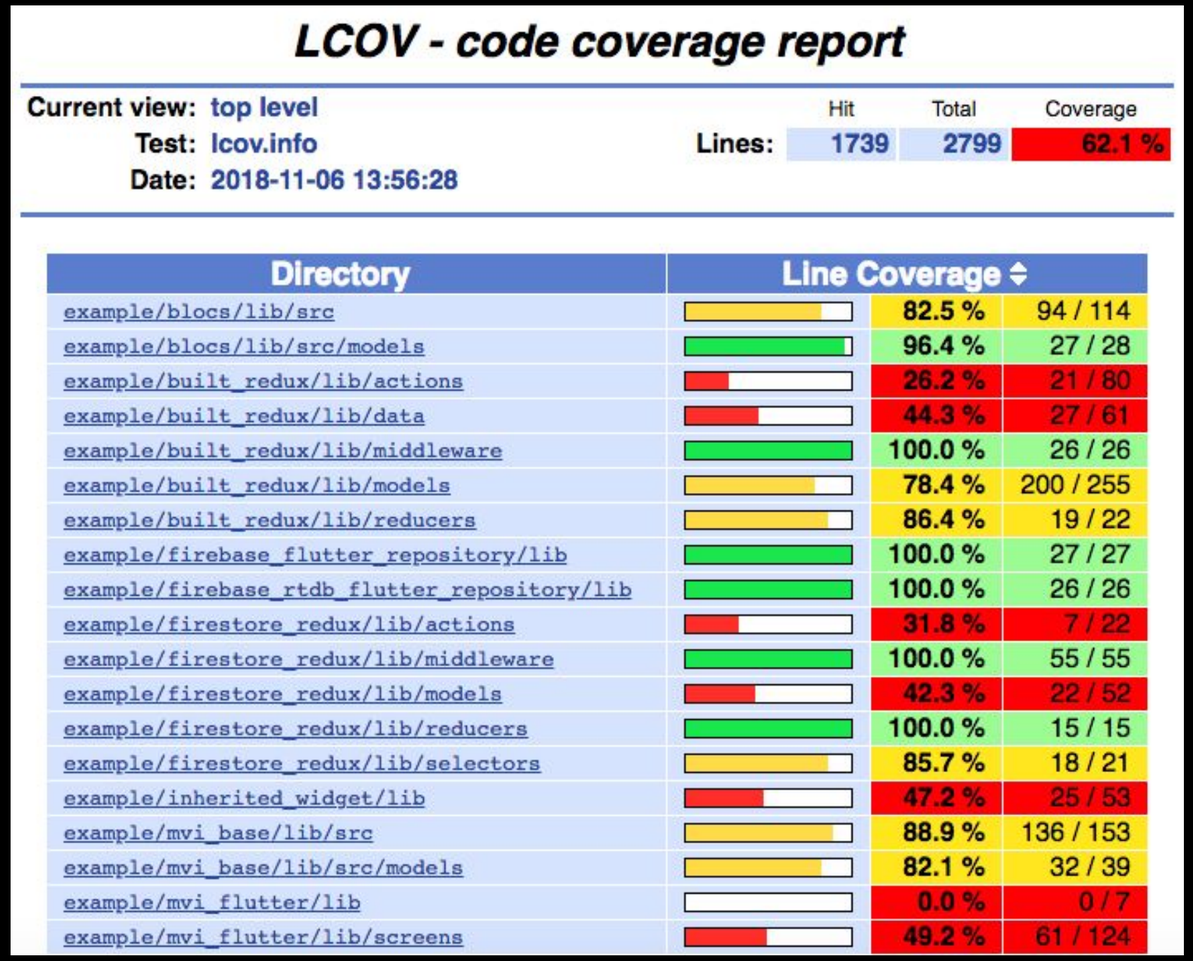
```
static Iterable<StepDefinitionGeneric> get steps => [
        ..._SeeProductDetails.steps,
        <otherStepsHere>,
]
---
/// Шаги "Я вижу" детальных экранов продуктов
class _SeeProductDetails {
 static final Iterable<StepDefinitionGeneric> steps = [
   then<ContextualWorld>(
     RegExp(r'Я вижу
```

Структура: теперь

Покрытие

Структура: теперь **Покрытие**

- Coverage



LCOV - code coverage report

		Hit	Total	Coverage
Current view:	top level			
Test:	lcov.info	Lines:	1739	2799	62.1 %
Date:	2018-11-06 13:56:28			

Directory	Line Coverage	
example/blocs/lib/src	82.5 %	94 / 114
example/blocs/lib/src/models	96.4 %	27 / 28
example/built_redux/lib/actions	26.2 %	21 / 80
example/built_redux/lib/data	44.3 %	27 / 61
example/built_redux/lib/middleware	100.0 %	26 / 26
example/built_redux/lib/models	78.4 %	200 / 255
example/built_redux/lib/reducers	86.4 %	19 / 22
example/firebase_flutter_repository/lib	100.0 %	27 / 27
example/firebase_rtdb_flutter_repository/lib	100.0 %	26 / 26
example/firestore_redux/lib/actions	31.8 %	7 / 22
example/firestore_redux/lib/middleware	100.0 %	55 / 55
example/firestore_redux/lib/models	42.3 %	22 / 52
example/firestore_redux/lib/reducers	100.0 %	15 / 15
example/firestore_redux/lib/selectors	85.7 %	18 / 21
example/inherited_widget/lib	47.2 %	25 / 53
example/mvi_base/lib/src	88.9 %	136 / 153
example/mvi_base/lib/src/models	82.1 %	32 / 39
example/mvi_flutter/lib	0.0 %	0 / 7
example/mvi_flutter/lib/screens	49.2 %	61 / 124

Структура: теперь **Покрытие**

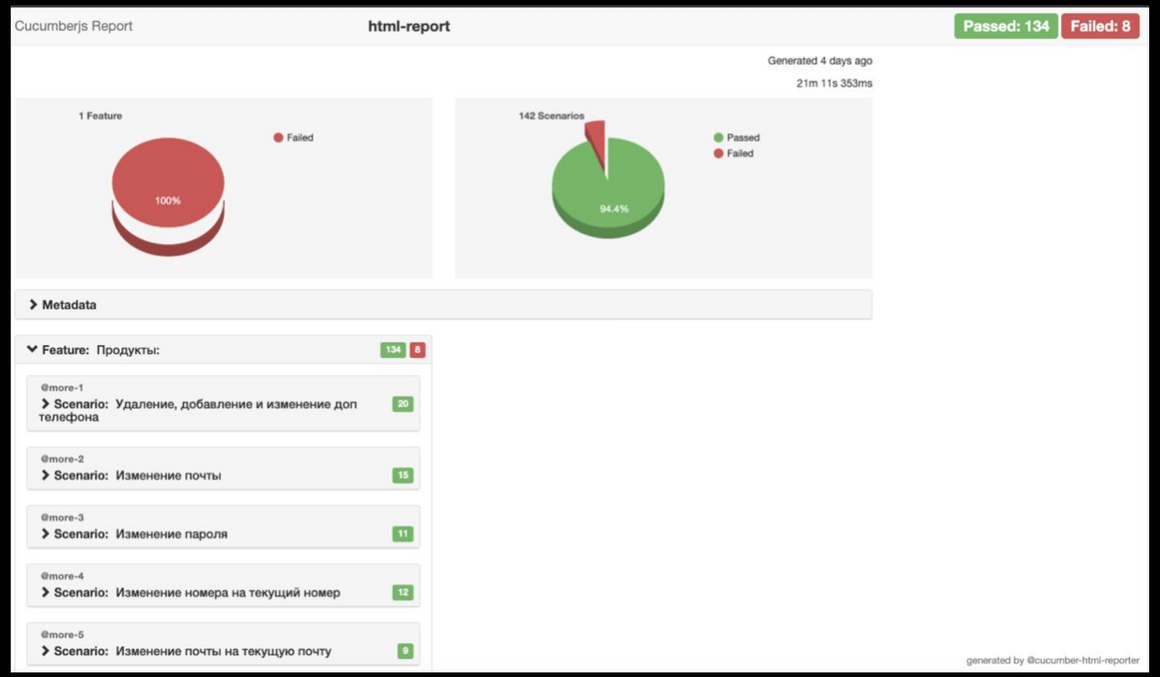
- Coverage
- Компонентные сценарии
- Бизнес-сценарии

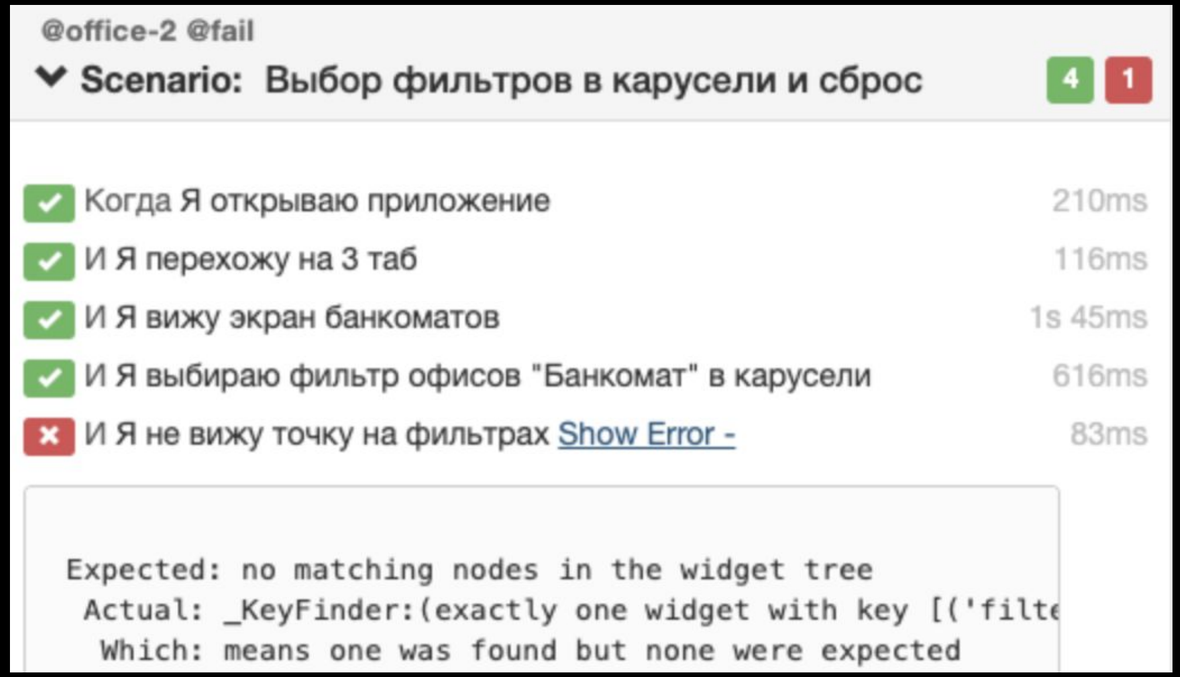
Структура: теперь **Отчеты**

- Cucumber-HTML-Reporter

Структура: теперь **Отчеты**

- Cucumber-HTML-Reporter
  - визуальное представление
  - ошибки и скриншоты





Структура: теперь

Сценарии

Структура: теперь **Сценарии**

- Бизнес-сценарии

Структура: теперь **Сценарии**

- Бизнес-сценарии
  - Запустить приложение -> Оказываемся на экране Авторизации -> Аккаунт имеется, Верно ввести логин и пароль -> Авторизоваться успешно -> Попадаем на экран каталога -> Открыть детали любой книги -> Добавить книгу в корзину -> Значок “в корзину” меняется на кнопку “перейти в корзину” -> Перейти в корзину -> В корзине лежит добавленная книга -> Перейти к оформлению заказа -> Выбрать оплату при получении, заполнить данные по адресу -> Оформить успешно -> Заказ создан, книга из корзины пропала, но имеется запись на экране Мои заказы

Структура: теперь **Сценарии**

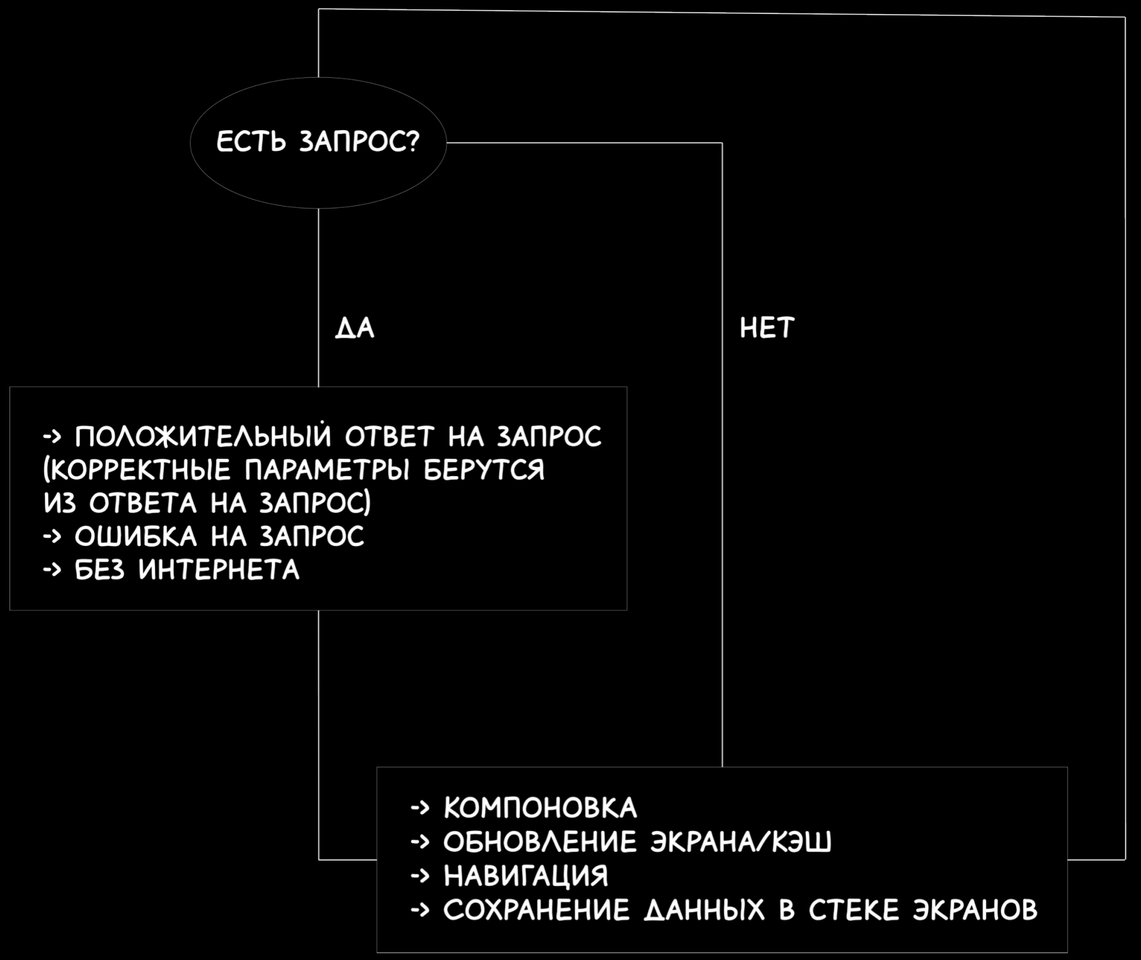
- Компонентные сценарии

Структура: теперь **Сценарии**

- Компонентные сценарии
  - Экран (в т.ч. шторка/popup)
  - Элемент (поле, карусель, чек-бокс, радиобаттон и тп)

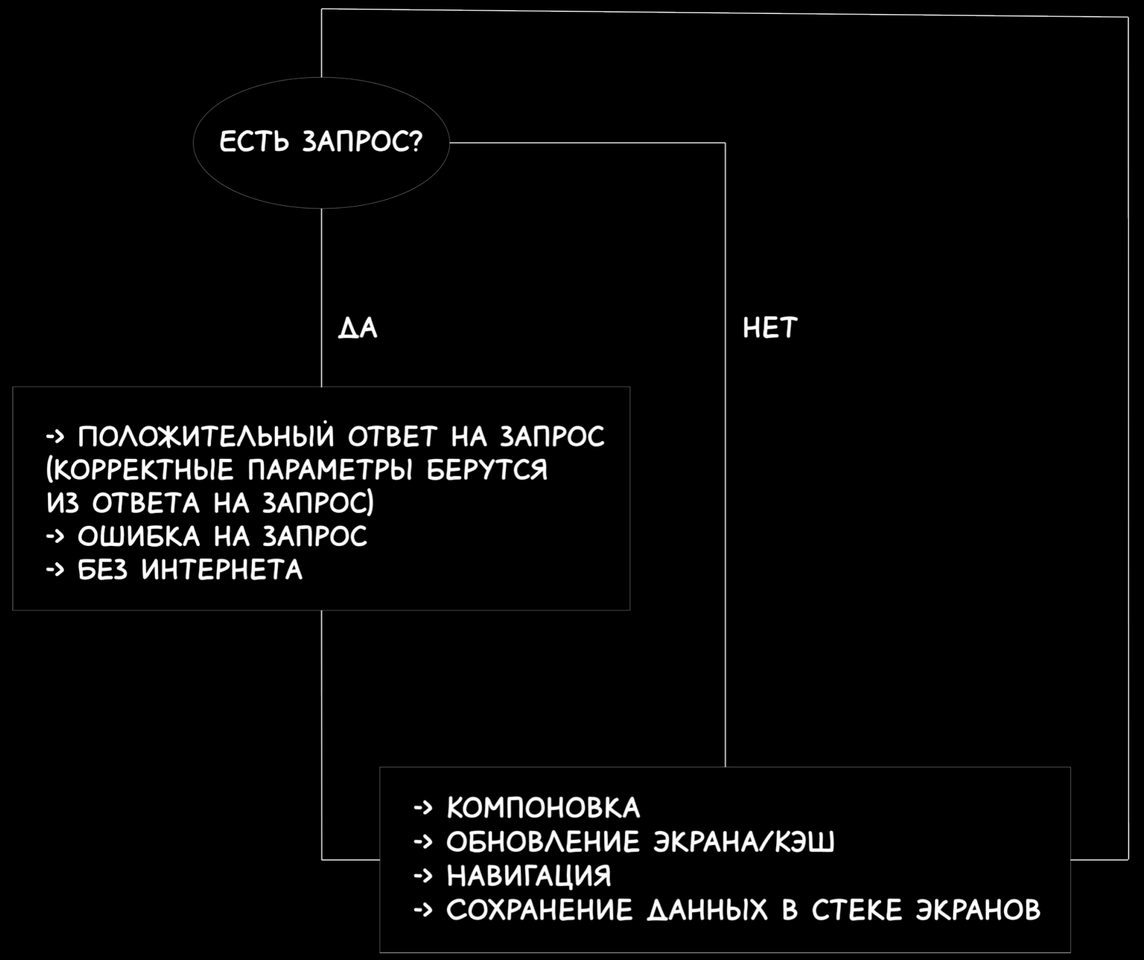
Структура: теперь **Сценарии**

- Экран (в т.ч. шторка/popup)



Структура: теперь **Сценарии**

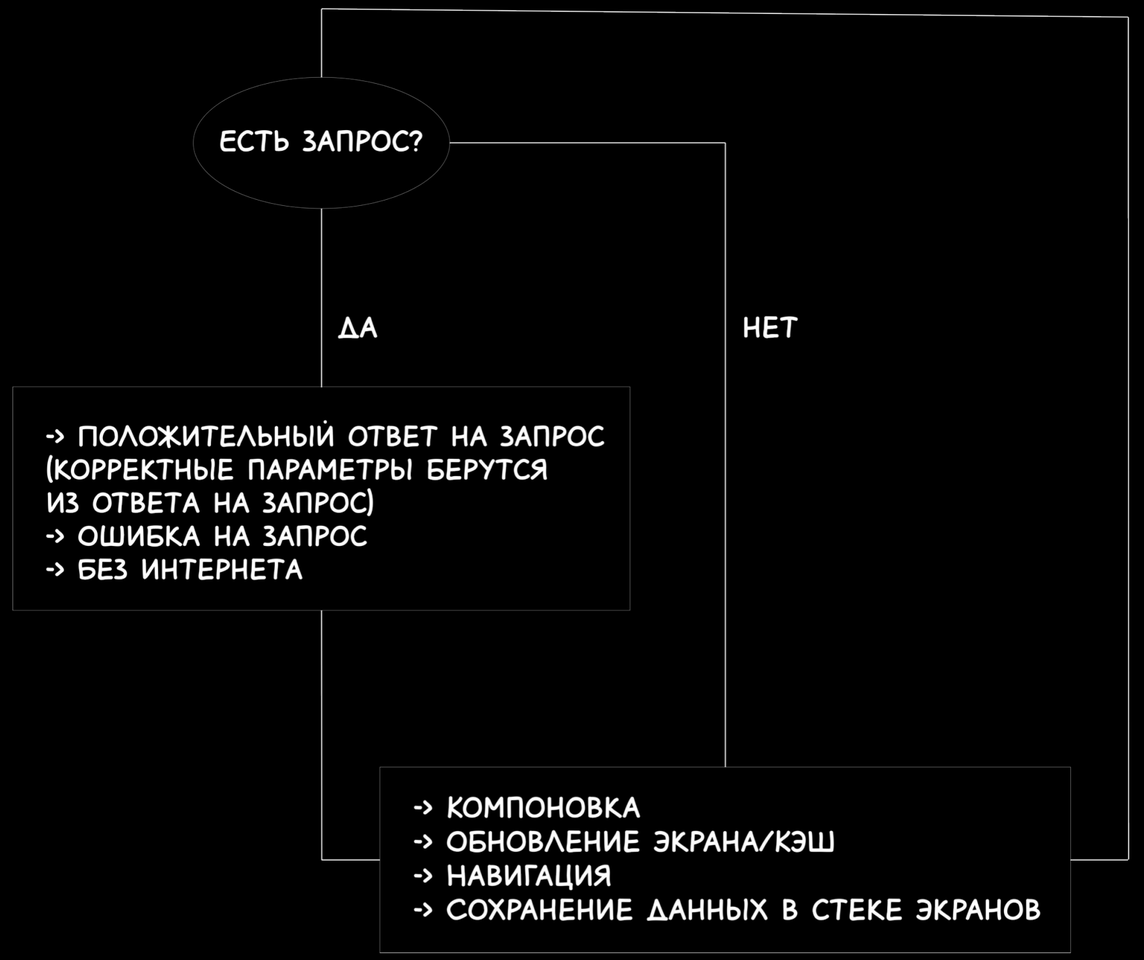
- Экран (в т.ч. шторка/popup)
  - → Инициализация



Структура: теперь **Сценарии**

- Экран (в т.ч. шторка/popup)
  - → Инициализация

-> Данные берутся из нужных параметров

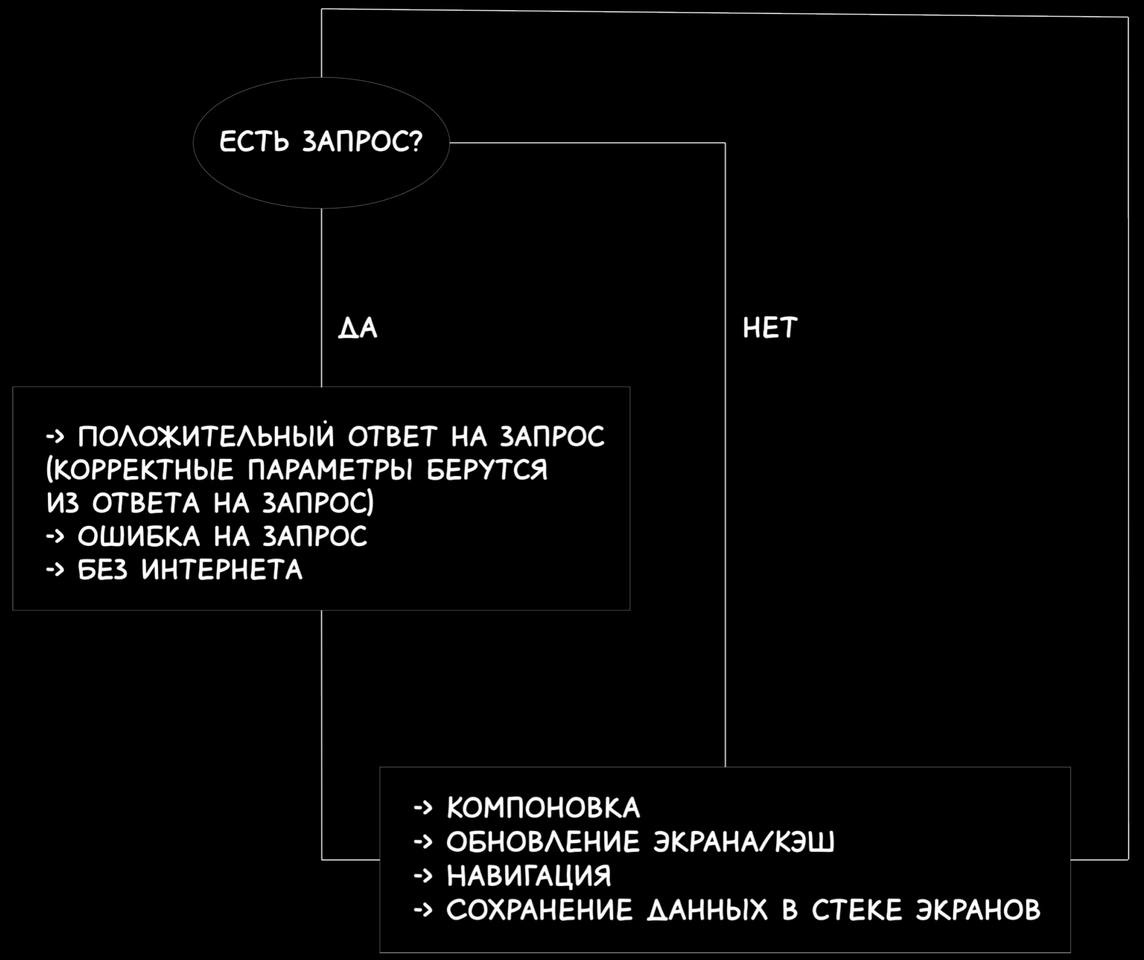


Структура: теперь **Сценарии**

- Экран (в т.ч. шторка/popup)
  - → Инициализация

-> Данные берутся из нужных параметров

-> Empty State



Структура: теперь **Сценарии**

- Экран (в т.ч. шторка/popup)
  - → Инициализация

-> Данные берутся из нужных параметров

-> Empty State

Структура: теперь **Сценарии**

- Экран (в т.ч. шторка/popup)
  - → Инициализация

-> Данные берутся из нужных параметров

-> Empty State

-> Error State

Структура: теперь **Сценарии**

- Экран (в т.ч. шторка/popup)
  - → Инициализация

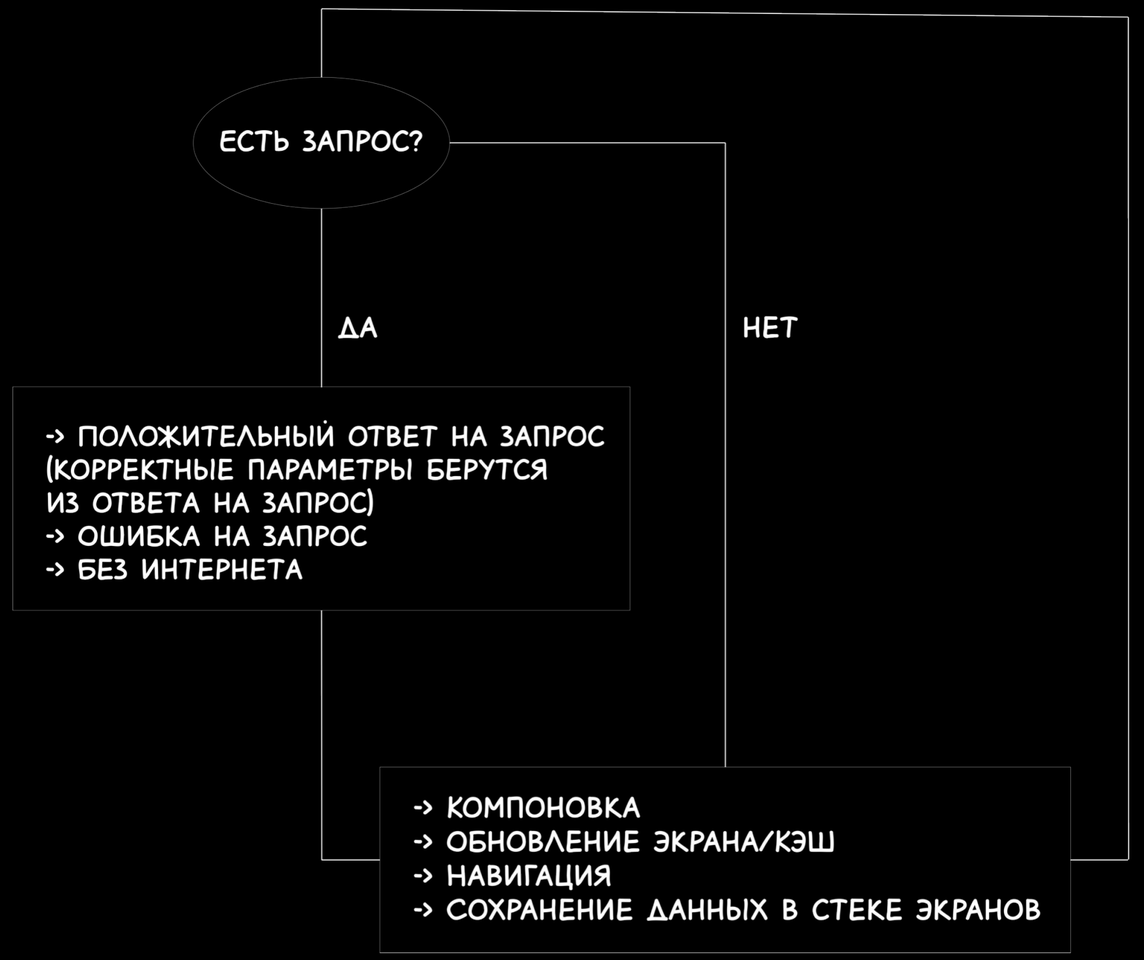
-> Данные берутся из нужных параметров

-> Empty State -> Обновление

-> Error State -> Обновление

Структура: теперь **Сценарии**

- Экран (в т.ч. шторка/popup)
  - → Инициализация
  - → Обновление экрана/КЭШ

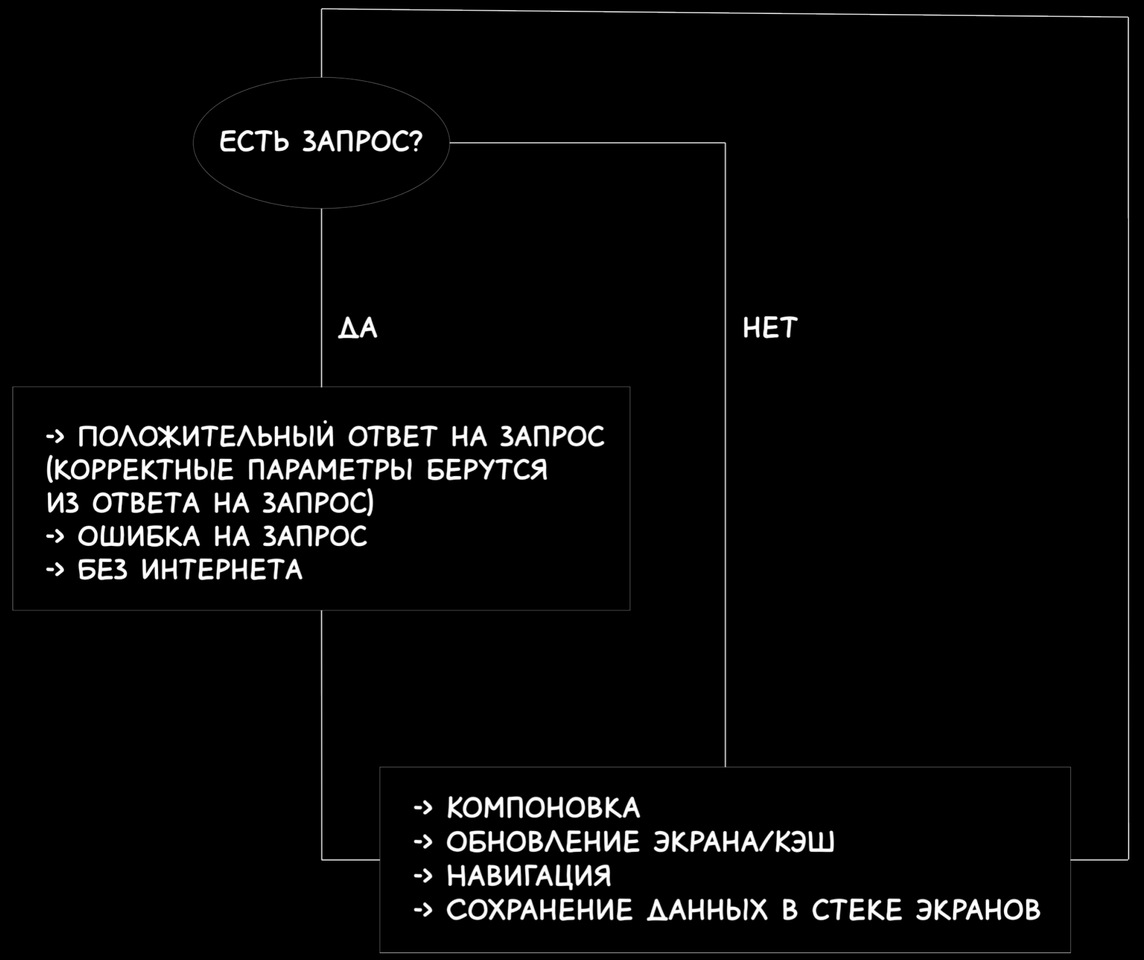


Структура: теперь **Сценарии**

- Экран (в т.ч. шторка/popup)
  - → Инициализация
  - → Обновление экрана/КЭШ

Структура: теперь **Сценарии**

- Экран (в т.ч. шторка/popup)
  - → Инициализация
  - → Обновление экрана/КЭШ
  - → Закрытие/Возврат



Структура: теперь **Сценарии**

- Экран (в т.ч. шторка/popup)
  - → Инициализация
  - → Обновление экрана/КЭШ
  - → Закрытие/Возврат

Структура: теперь **Сценарии**

- Экран (в т.ч. шторка/popup)
  - → Инициализация
  - → Обновление экрана/КЭШ
  - → Закрытие/Возврат

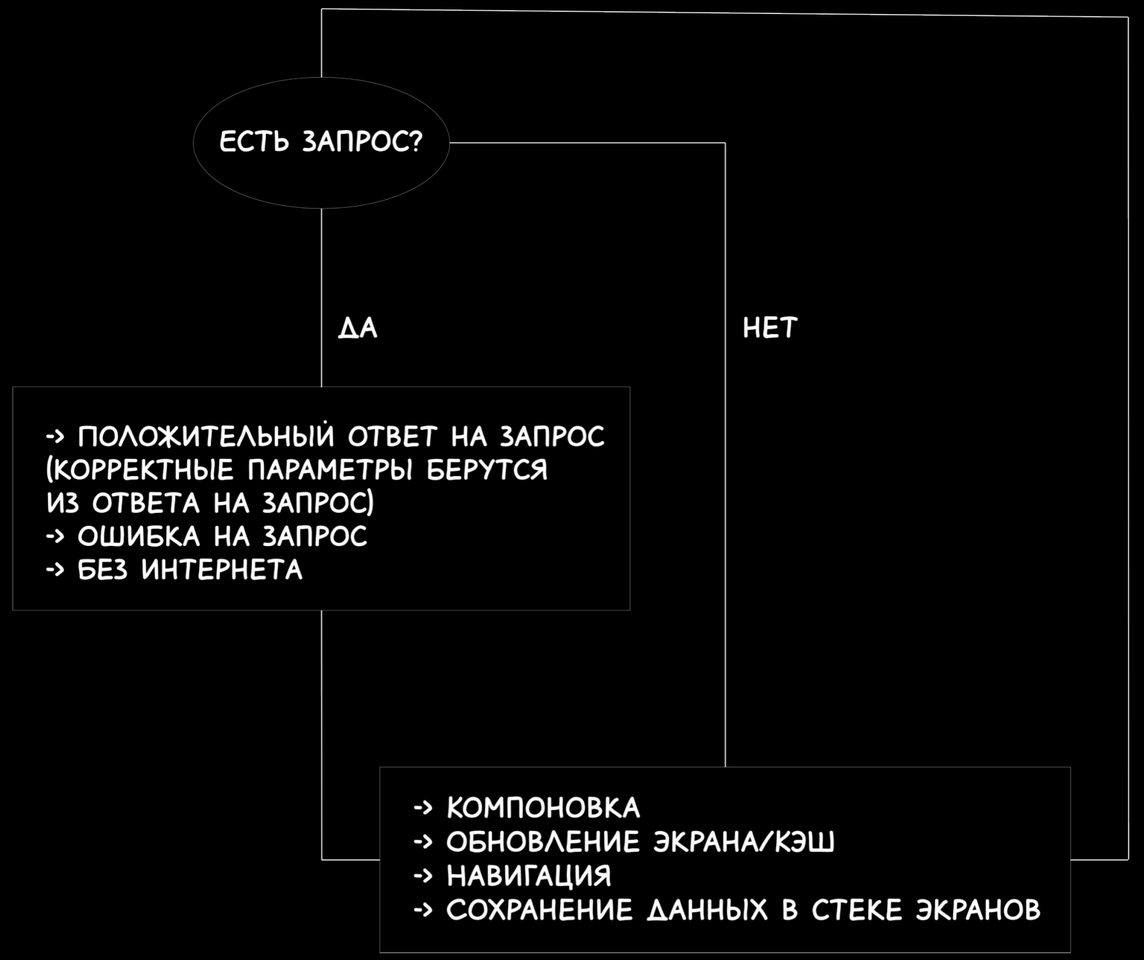
-> Назад по кнопке/Крестик, если есть

-> Физическая кнопка Назад / Жест (Android)

-> Backswipe (iOS)

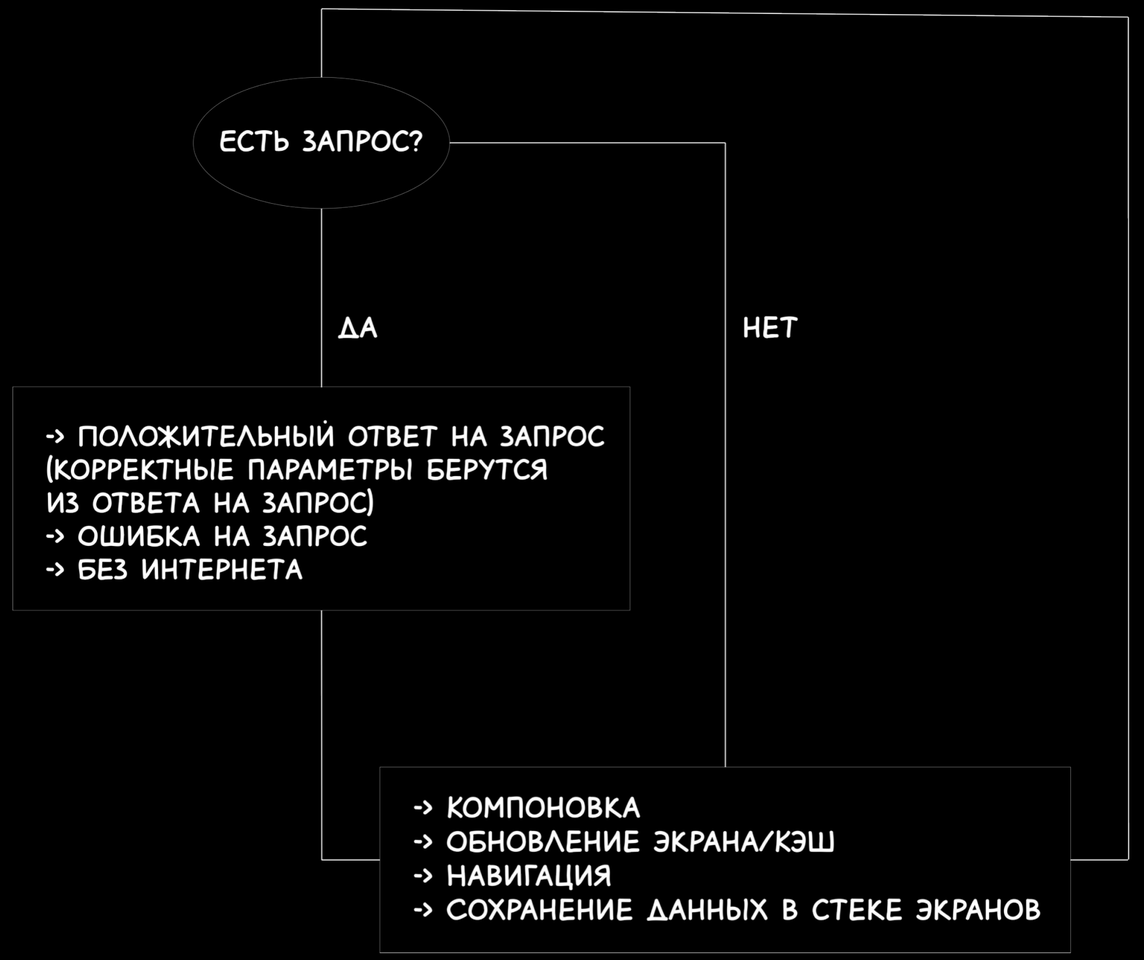
Структура: теперь **Сценарии**

- Экран (в т.ч. шторка/popup)
  - → Инициализация
  - → Обновление экрана/КЭШ
  - → Закрытие/Возврат
  - → Логика взаимодействия между экранами в стеке и мп



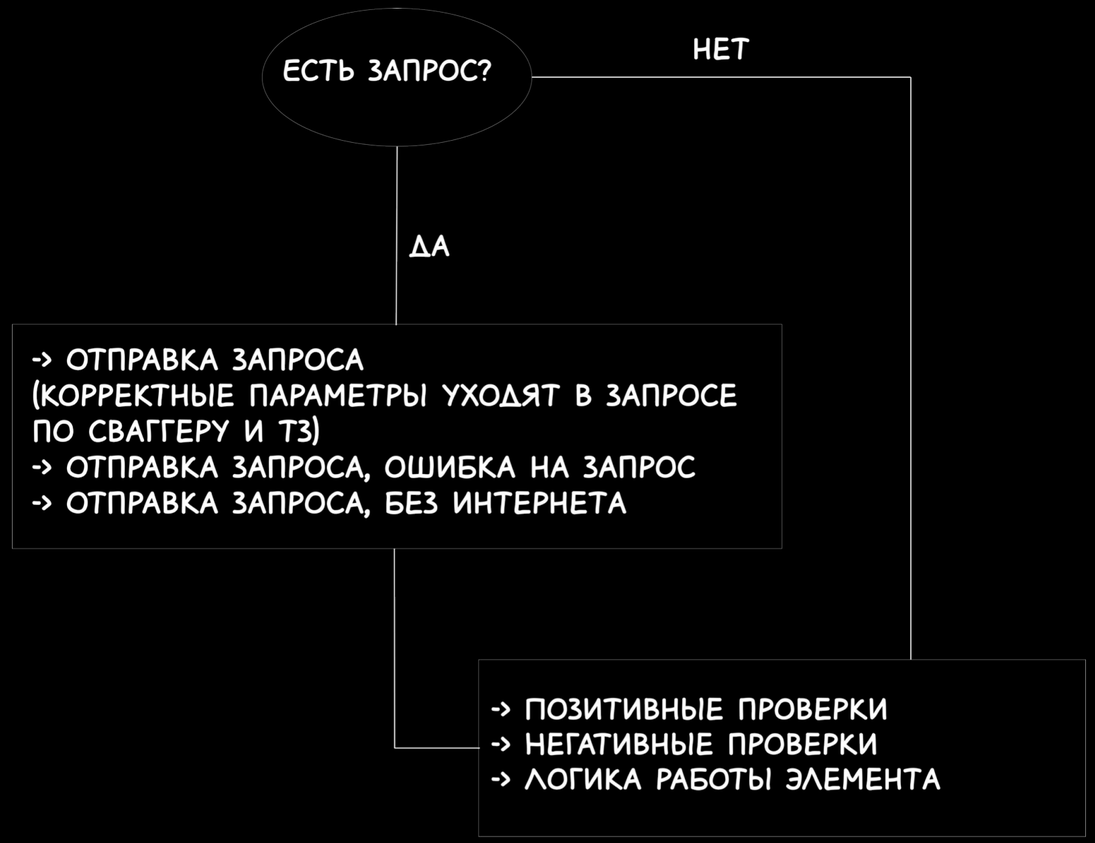
Структура: теперь **Сценарии**

- Экран (в т.ч. шторка/popup)
  - → Инициализация
  - → Обновление экрана/КЭШ
  - → Закрытие/Возврат
  - → Логика взаимодействия между экранами в стеке и мп
  - → Компоновка



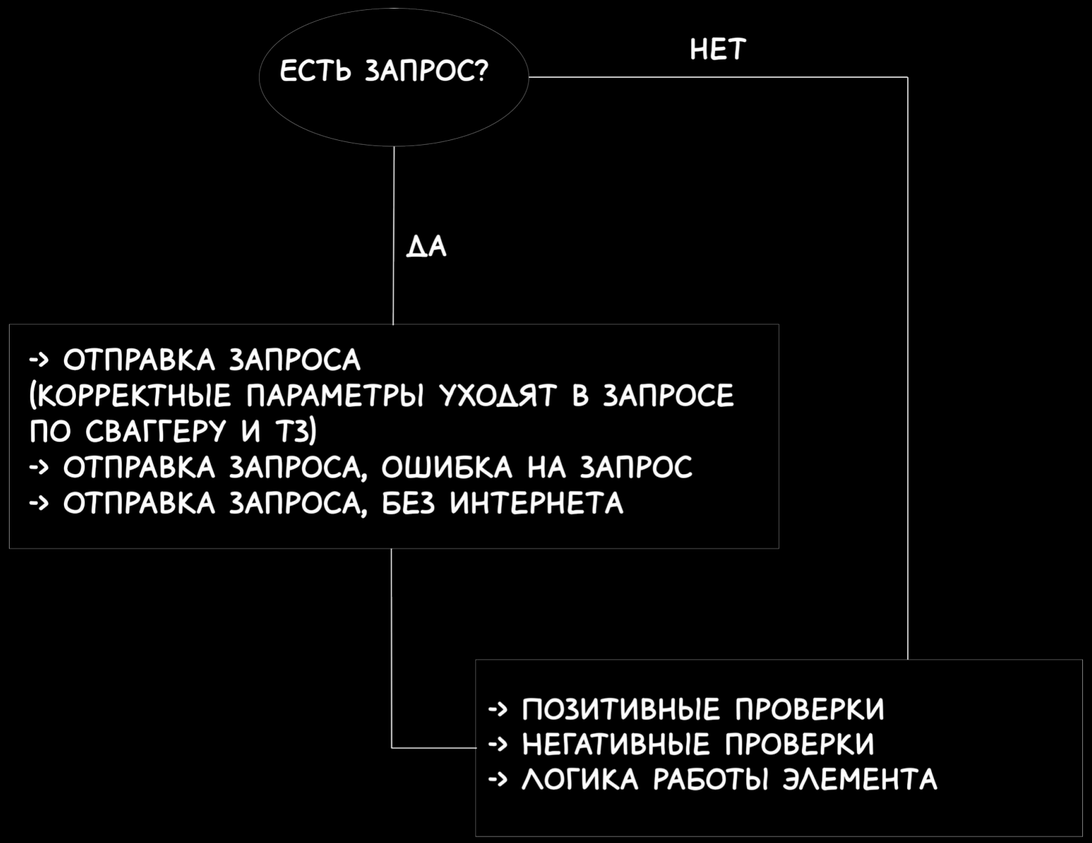
Структура: теперь **Сценарии**

- Элемент (поле, карусель, чек-бокс, радиобаттон и тп)



Структура: теперь **Сценарии**

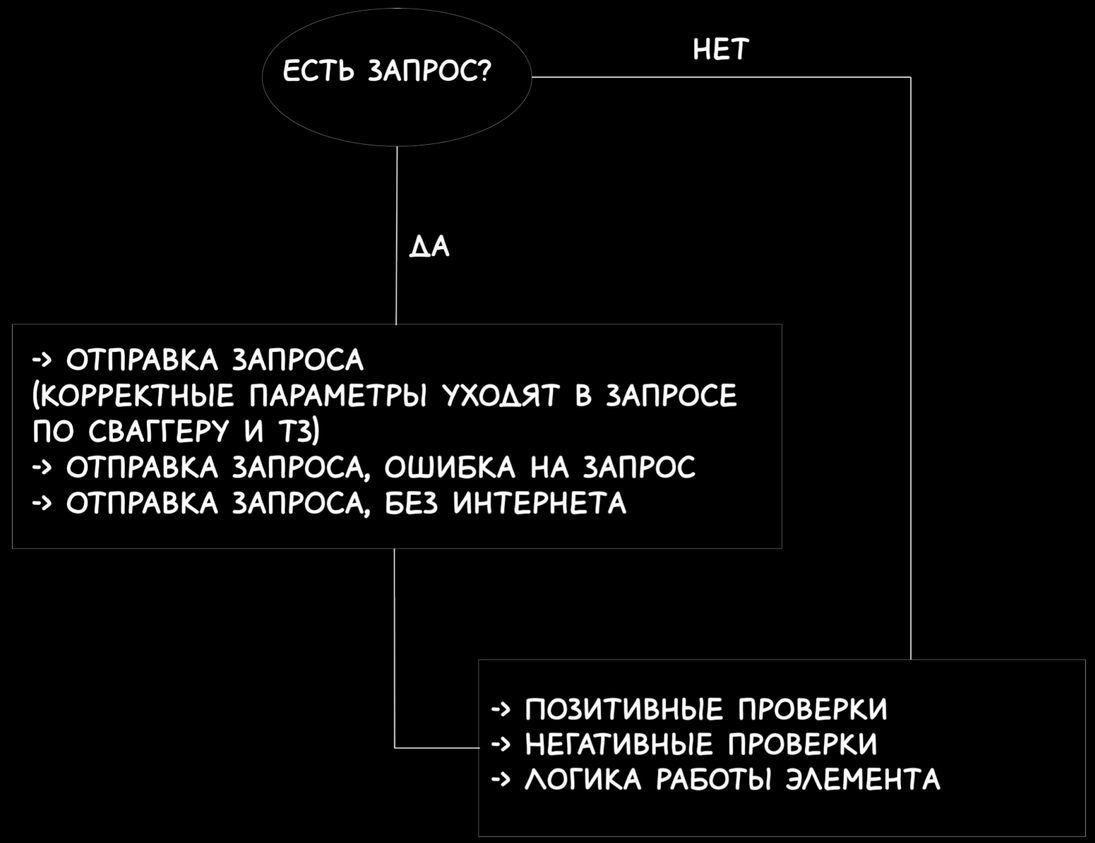
- Элемент (поле, карусель, чек-бокс, радиобаттон и тп)
  - → Позитивные проверки
  - → Негативные проверки
  - → Логика работы элемента



Структура: теперь **Сценарии**

- Элемент (поле, карусель, чек-бокс, радиобаттон и тп)
  - → Позитивные проверки

-> Заполнение поля / Вставка в поле корректных значений

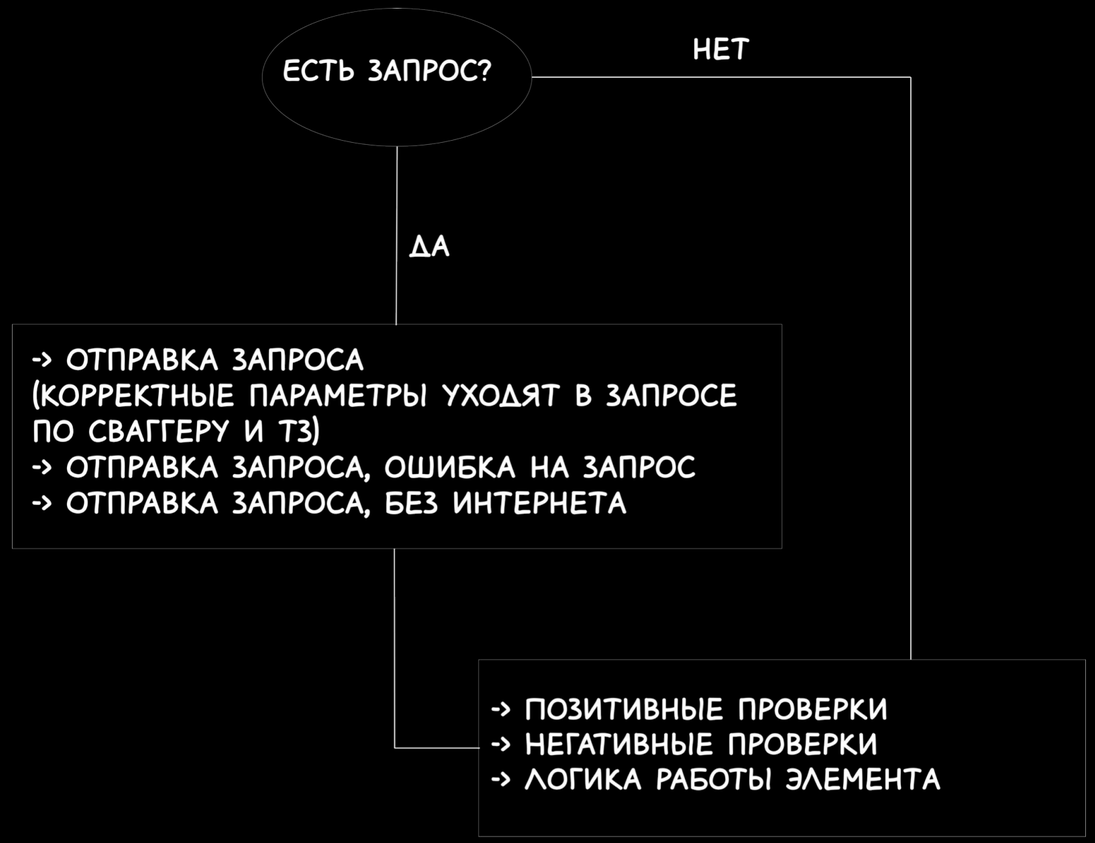


Структура: теперь **Сценарии**

- Элемент (поле, карусель, чек-бокс, радиобаттон и тп)
  - → Позитивные проверки

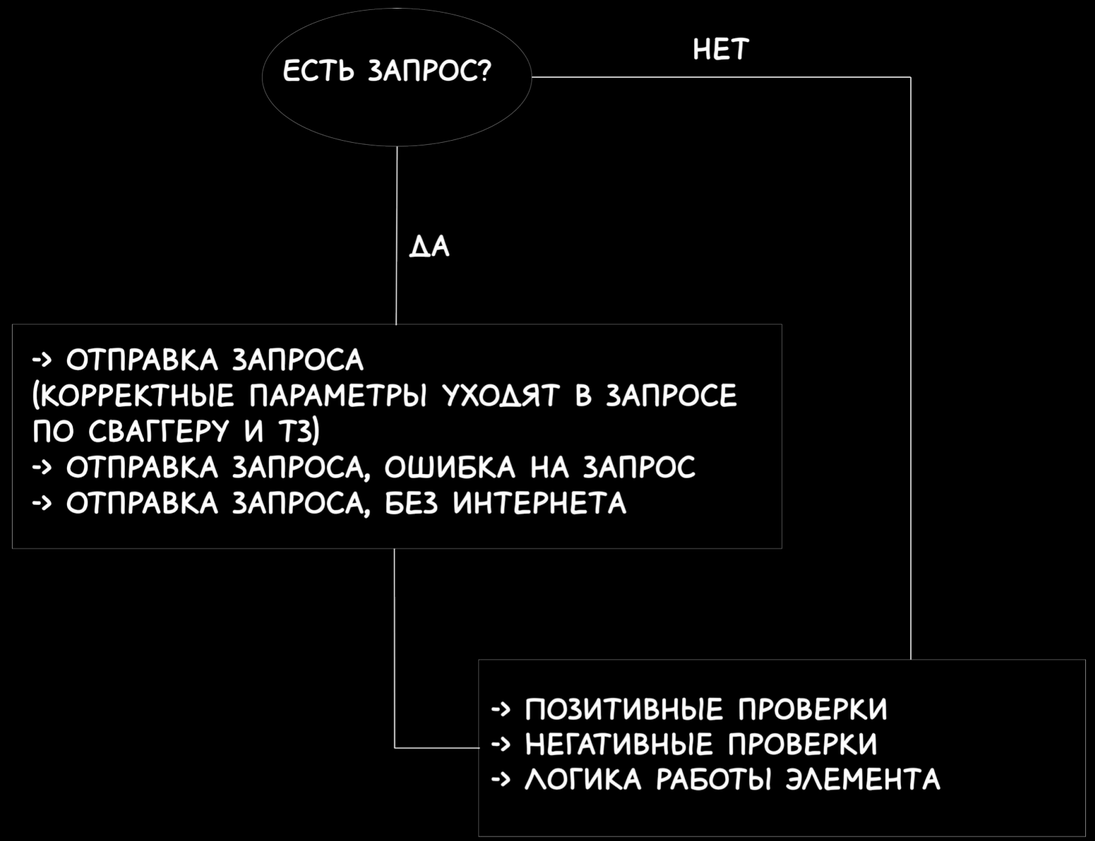
-> Заполнение поля / Вставка в поле корректных значений

-> Ограничение поля



Структура: теперь **Сценарии**

- Элемент (поле, карусель, чек-бокс, радиобаттон и тп)
  - → Позитивные проверки



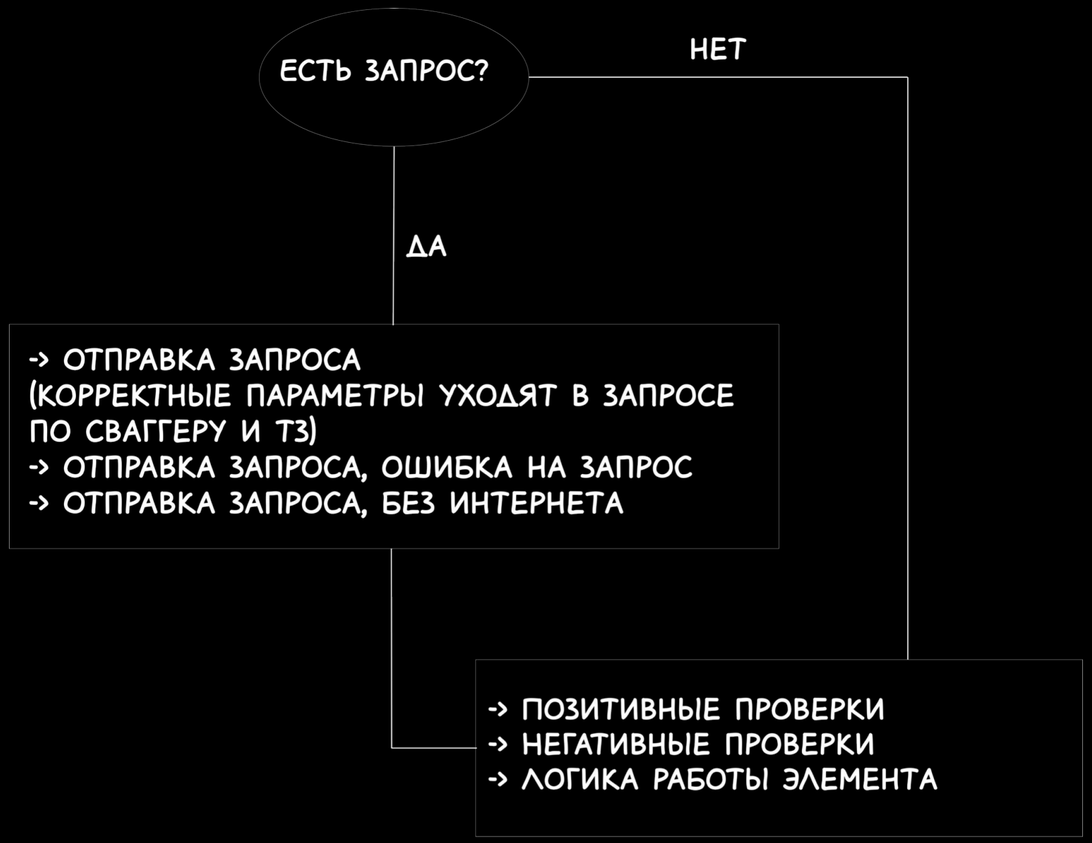
-> Заполнение поля / Вставка в поле корректных значений

-> Ограничение поля

-> Заполнение поля / Вставка максимальных значений

Структура: теперь **Сценарии**

- Элемент (поле, карусель, чек-бокс, радиобаттон и тп)
  - → Позитивные проверки



-> Заполнение поля / Вставка в поле корректных значений

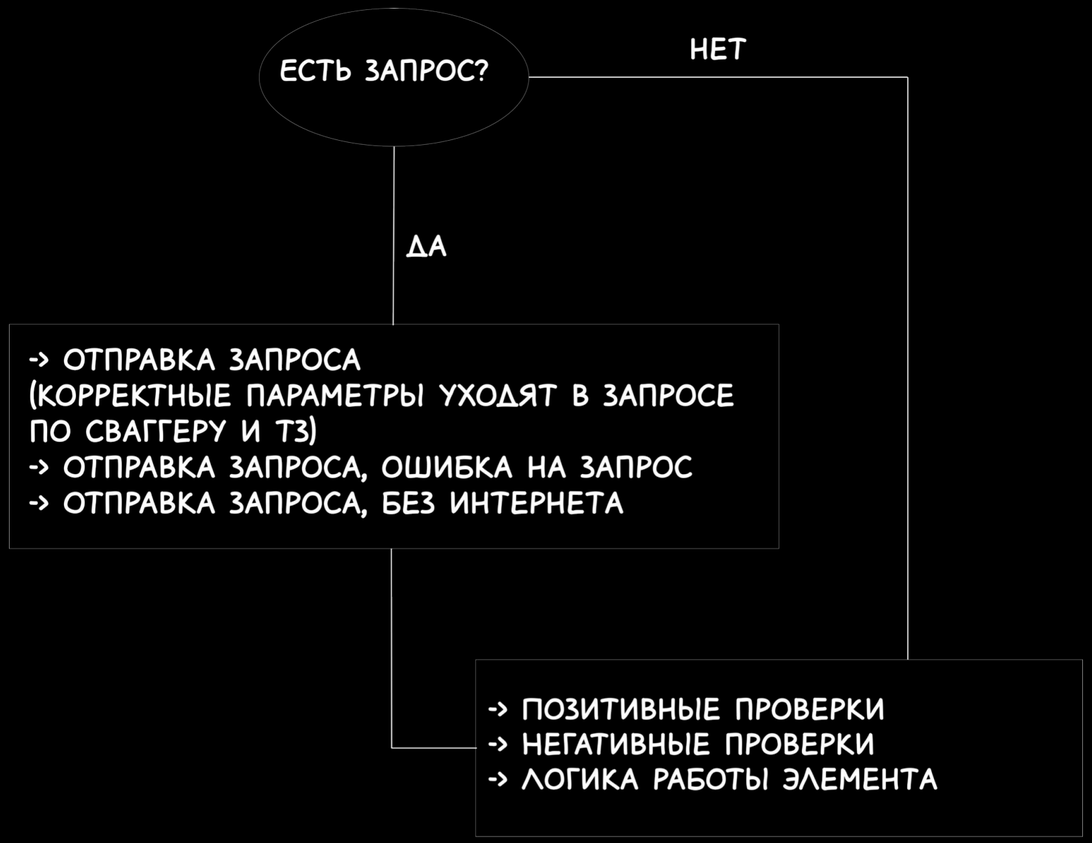
-> Ограничение поля

-> Заполнение поля / Вставка максимальных значений

-> Пустое поле (если заполнение необязательно)

Структура: теперь **Сценарии**

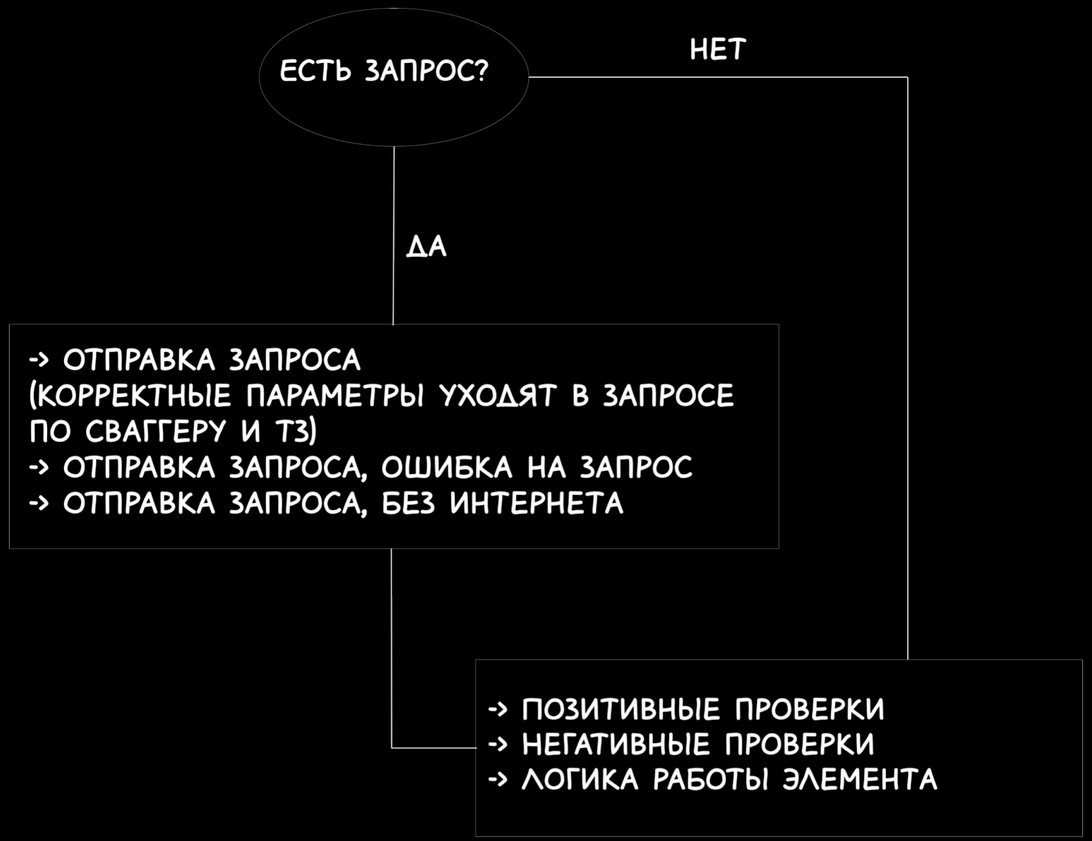
- Элемент (поле, карусель, чек-бокс, радиобаттон и тп)
  - → Позитивные проверки
  - → Негативные проверки



Структура: теперь **Сценарии**

- Элемент (поле, карусель, чек-бокс, радиобаттон и тп)
  - → Позитивные проверки
  - → Негативные проверки

-> Заполнение поля / Вставка в поле корректных значений

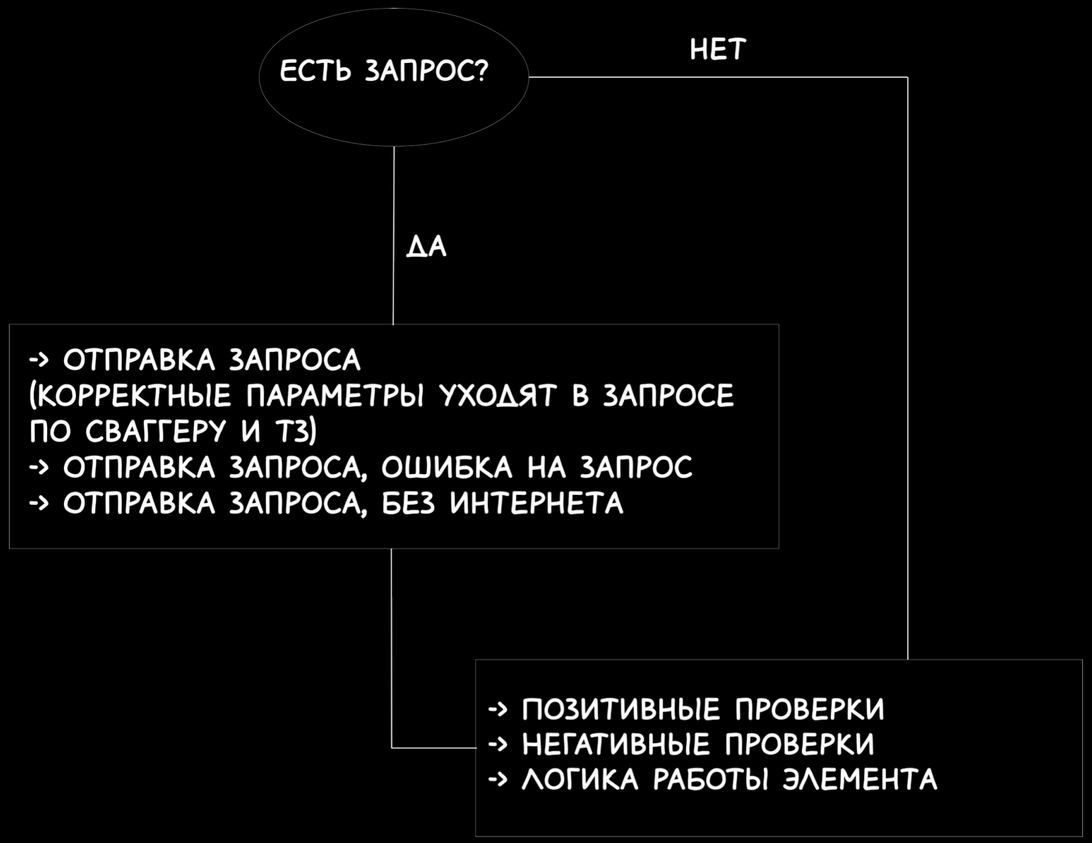


Структура: теперь **Сценарии**

- Элемент (поле, карусель, чек-бокс, радиобаттон и тп)
  - → Позитивные проверки
  - → Негативные проверки

-> Заполнение поля / Вставка в поле корректных значений

-> Ограничение поля



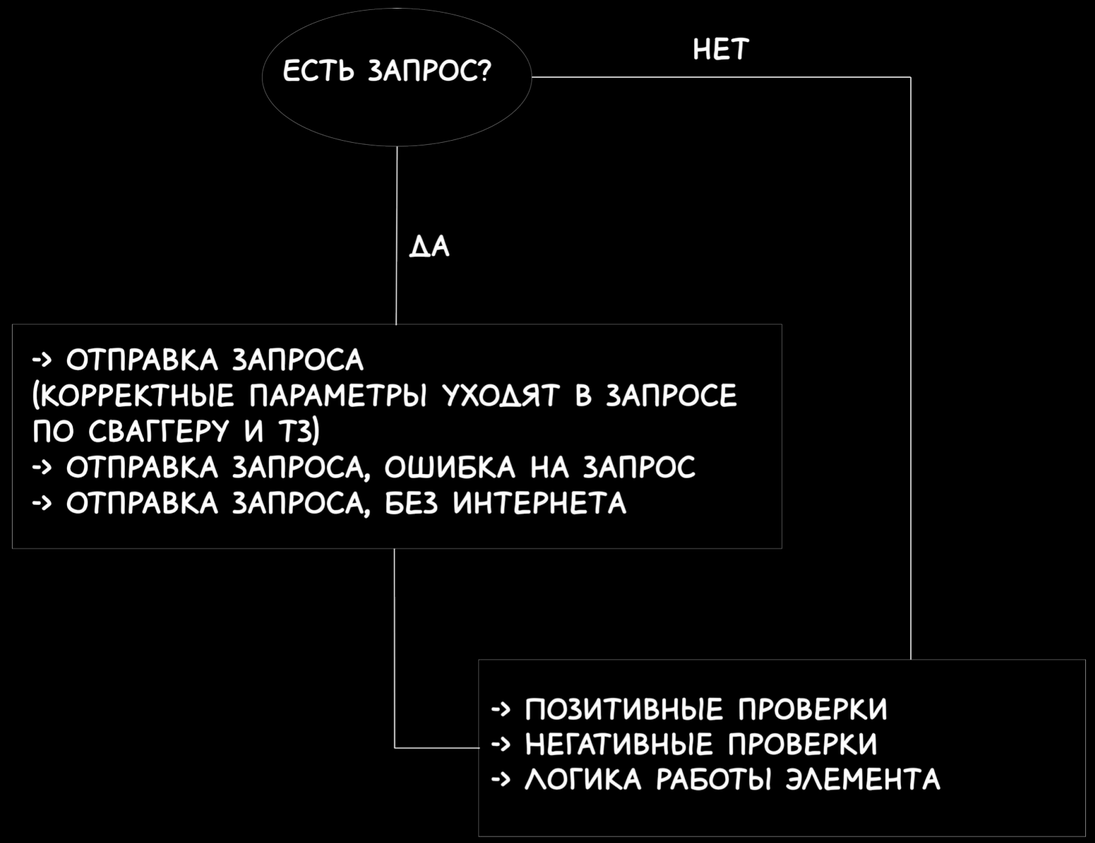
Структура: теперь **Сценарии**

- Элемент (поле, карусель, чек-бокс, радиобаттон и тп)
  - → Позитивные проверки
  - → Негативные проверки

-> Заполнение поля / Вставка в поле корректных значений

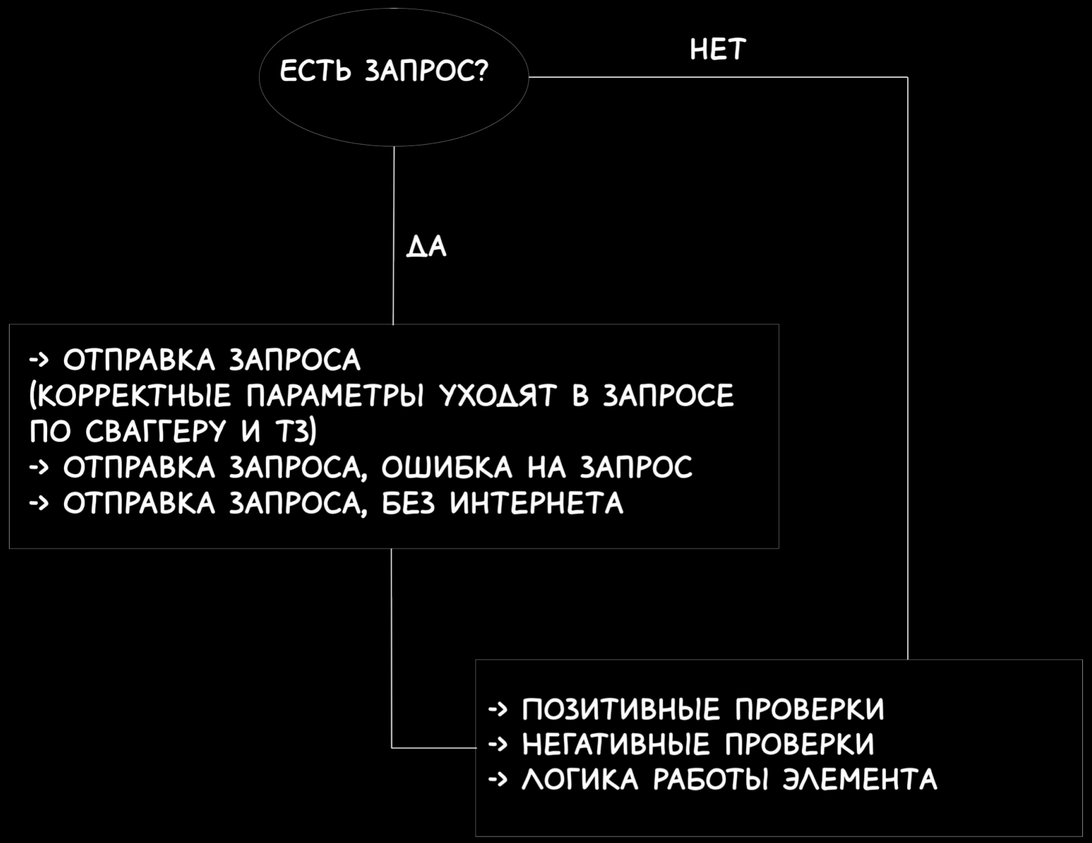
-> Ограничение поля

-> Пустое поле (если заполнение обязательно)



Структура: теперь **Сценарии**

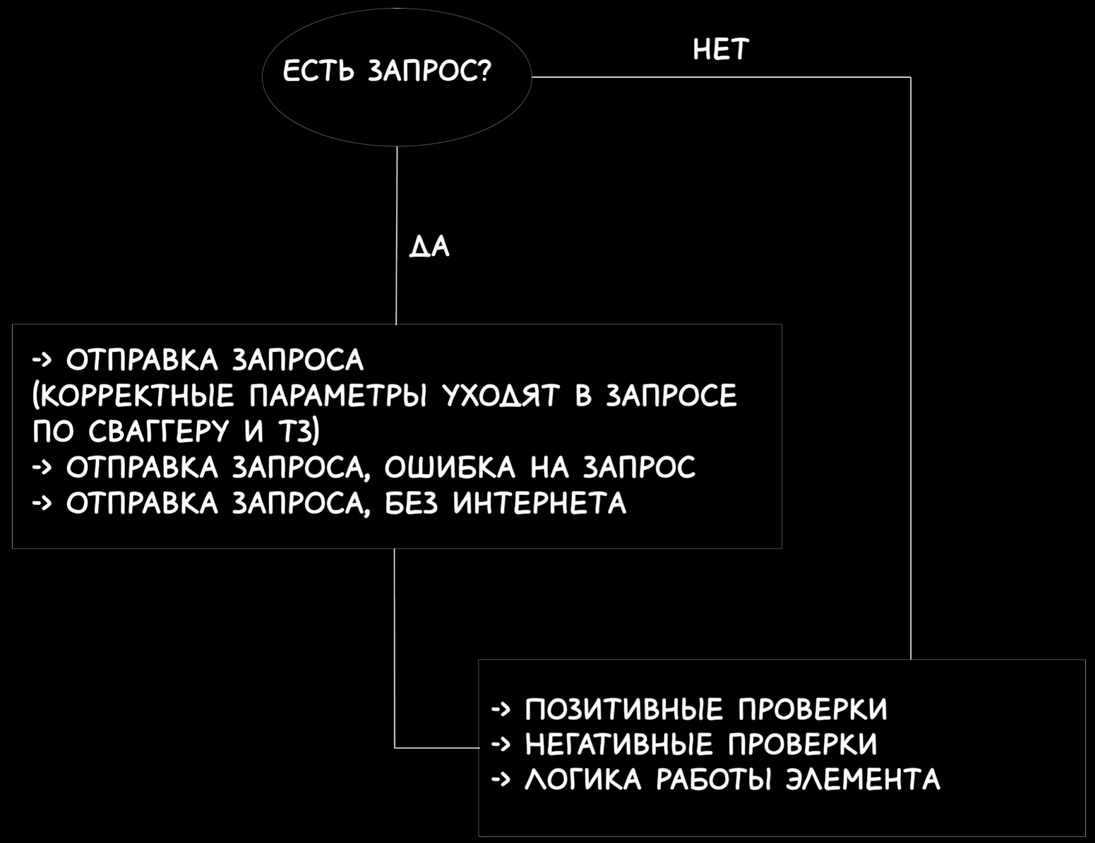
- Элемент (поле, карусель, чек-бокс, радиобаттон и тп)
  - → Позитивные проверки
  - → Негативные проверки
  - → Логика работы элемента



Структура: теперь **Сценарии**

- Элемент (поле, карусель, чек-бокс, радиобаттон и тп)
  - → Позитивные проверки
  - → Негативные проверки
  - → Логика работы элемента

-> Вид клавиатуры

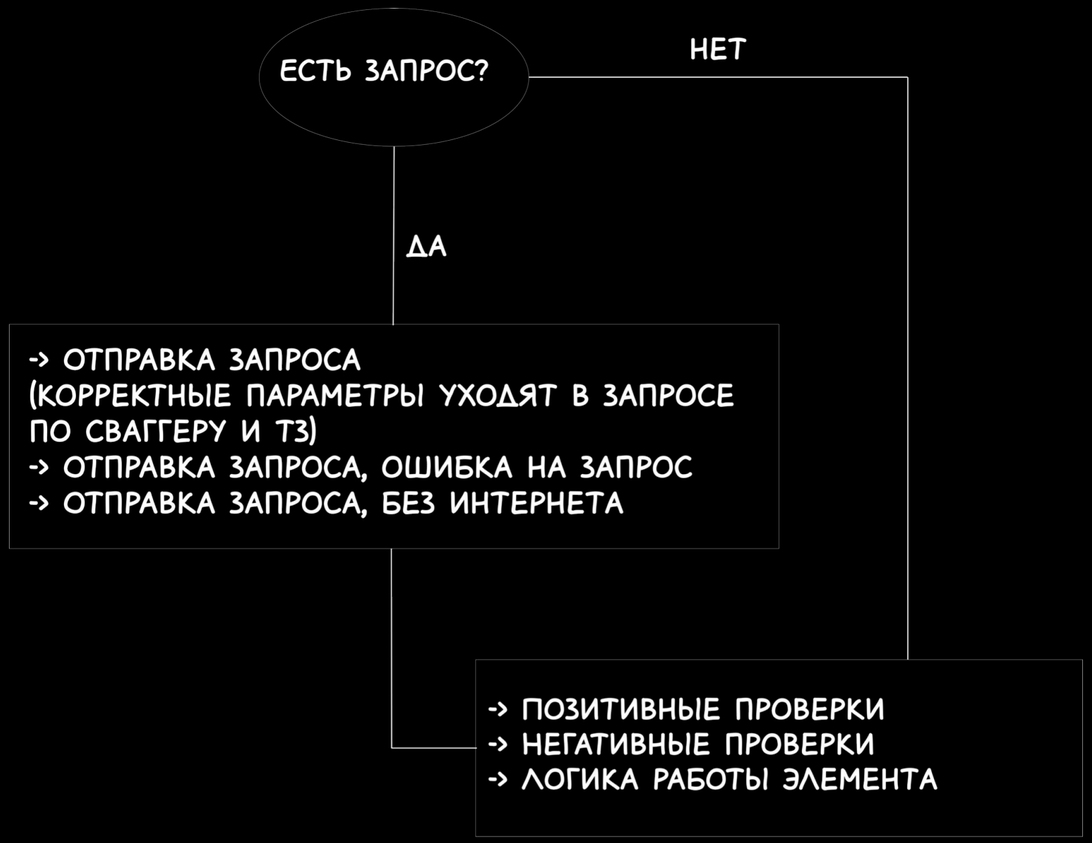


Структура: теперь **Сценарии**

- Элемент (поле, карусель, чек-бокс, радиобаттон и тп)
  - → Позитивные проверки
  - → Негативные проверки
  - → Логика работы элемента

-> Вид клавиатуры

-> Выставление курсора



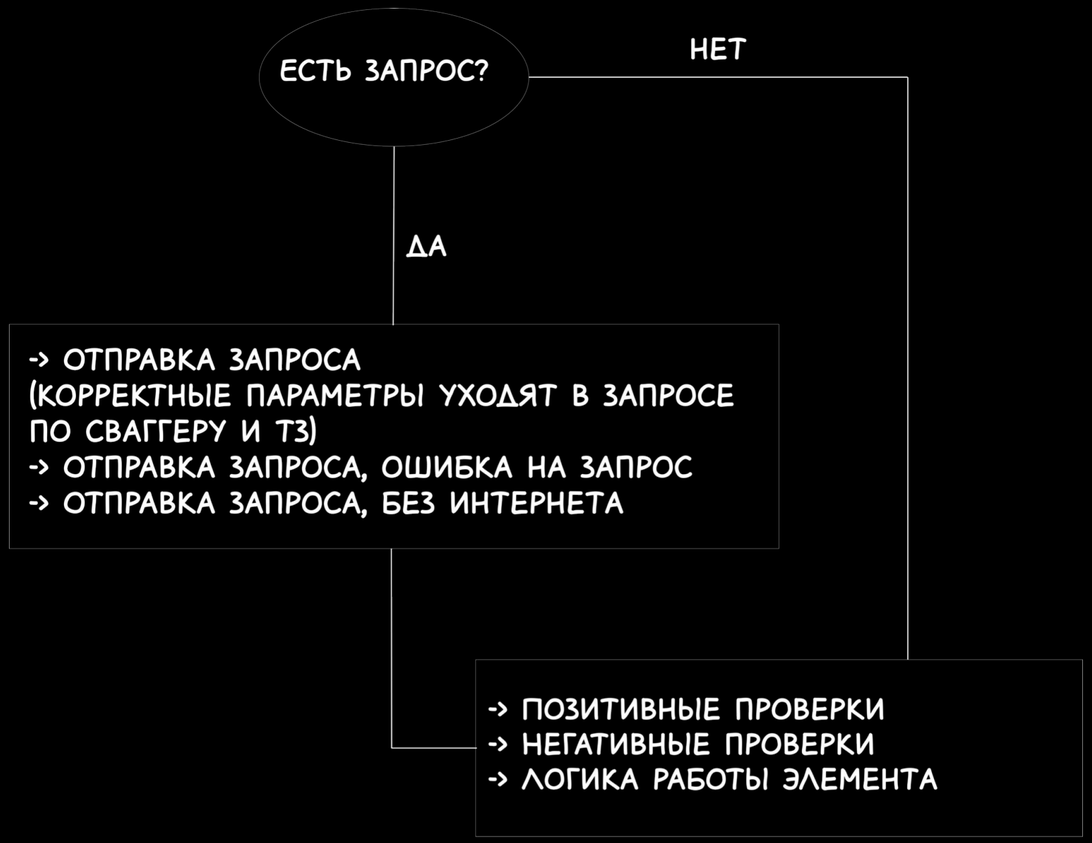
Структура: теперь **Сценарии**

- Элемент (поле, карусель, чек-бокс, радиобаттон и тп)
  - → Позитивные проверки
  - → Негативные проверки
  - → Логика работы элемента

-> Вид клавиатуры

-> Выставление курсора

-> Валидация при снятии фокуса



Структура: теперь **Сценарии**

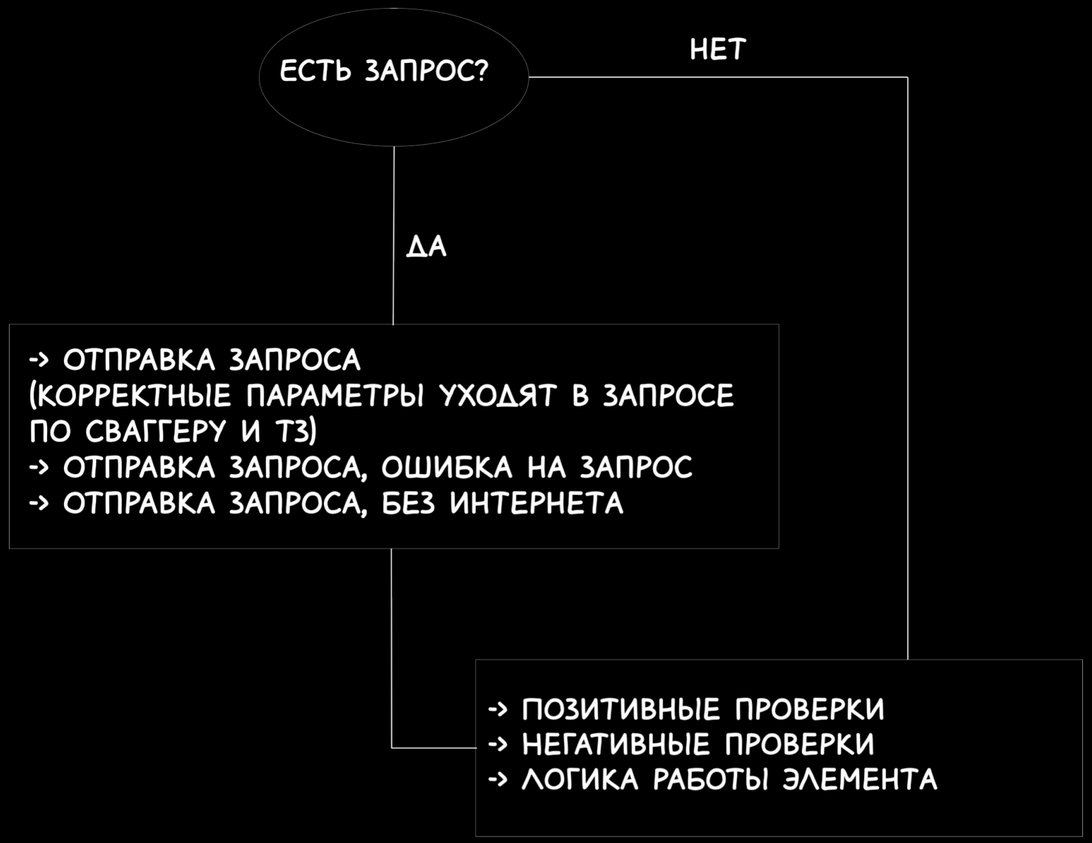
- Элемент (поле, карусель, чек-бокс, радиобаттон и тп)
  - → Позитивные проверки
  - → Негативные проверки
  - → Логика работы элемента

-> Вид клавиатуры

-> Выставление курсора

-> Валидация при снятии фокуса

-> Маска (для номера телефона)



Структура: теперь **Сценарии**

- Элемент (поле, карусель, чек-бокс, радиобаттон и тп)
  - → Позитивные проверки
  - → Негативные проверки
  - → Логика работы элемента

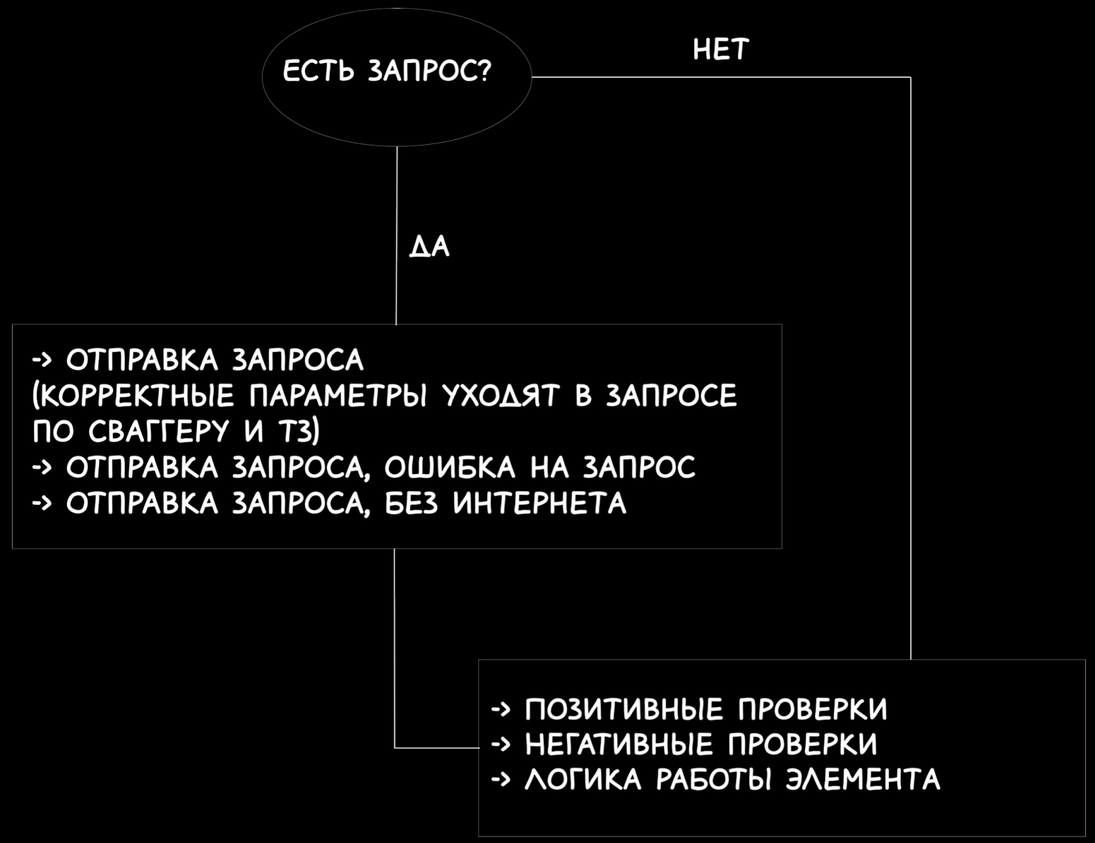
-> Вид клавиатуры

-> Выставление курсора

-> Валидация при снятии фокуса

-> Маска (для номера телефона)

-> Активация/деактивация чек-бокса



Структура: теперь **Сценарии**

- Элемент (поле, карусель, чек-бокс, радиобаттон и тп)
  - → Позитивные проверки
  - → Негативные проверки
  - → Логика работы элемента

Структура: теперь

Сценарии

Структура: теперь **Сценарии**

- Компонентные сценарии -> Widget-tests

Структура: теперь **Сценарии**

- Компонентные сценарии -> Widget-tests
- Бизнес-сценарии -> e2e-tests

Итоги

Итоги

- Стабильные тесты
  - запуск на pull request’ах
  - запуск ночью

Итоги

- Стабильные тесты
  - запуск на pull request'ах
  - запуск ночью
- Предотвращение багов на этапе разработки

Итоги

- Стабильные тесты
  - запуск на pull request’ах
  - запуск ночью
- Предотвращение багов на этапе разработки
- Простая актуализация автотестов

Итоги

- Стабильные тесты
  - запуск на pull request’ах
  - запуск ночью
- Предотвращение багов на этапе разработки
- Простая актуализация автотестов
- Компонентное и интеграционное тестирование

Итоги

- Стабильные тесты
  - запуск на pull request’ах
  - запуск ночью
- Предотвращение багов на этапе разработки
- Простая актуализация автотестов
- Компонентное и интеграционного тестирование
- Моки и тестовый сервер

Итоги

- Стабильные тесты
  - запуск на pull request'ах
  - запуск ночью
- Предотвращение багов на этапе разработки
- Простая актуализация автотестов
- Компонентное и интеграционного тестирование
- Моки и тестовый сервер
- Измерение покрытия

Автотесты в Surf

Flutter: особенности

Плюсы

Автотесты в Surf
Flutter: особенности

Плюсы

- нативные и кроссплатформенные

Автотесты в Surf
Flutter: особенности

Плюсы

- нативные и кроссплатформенные
- активная поддержка фреймворка

Автотесты в Surf
Flutter: особенности

Плюсы

- нативные и кроссплатформенные
- активная поддержка фреймворка
- открытый доступ к архитектуре и сущностям

Автотесты в Surf
Flutter: особенности

Плюсы

- нативные и кроссплатформенные
- активная поддержка фреймворка
- открытый доступ к архитектуре и сущностям
- независимость от разработчиков

Автотесты в Surf
Flutter: особенности

Плюсы

- нативные и кроссплатформенные
- активная поддержка фреймворка
- открытый доступ к архитектуре и сущностям
- независимость от разработчиков
- прозрачность

Автотесты в Surf
Flutter: особенности

Плюсы

- нативные и кроссплатформенные
- активная поддержка фреймворка
- открытый доступ к архитектуре и сущностям
- независимость от разработчиков
- прозрачность
- поиск багов на этапе разработки (запуск на pull request-ах)

Автотесты в Surf

Flutter: особенности

Минусы

Автотесты в Surf
Flutter: особенности

Минусы

- новый фреймворк

Автотесты в Surf
Flutter: особенности

Минусы

- новый фреймворк
- тесты внутри мобильного приложения (не отделимо с ним)

Автотесты в Surf
Flutter: особенности

Минусы

- новый фреймворк
- тесты внутри мобильного приложения (не отделимо с ним)
- изменения в коде вынуждают делать билд (даже внутри тестов)

Автотесты в Surf

Flutter: особенности

Автотесты в Surf
Flutter: особенности

Костыли

- при падении одного теста, падают все без спец.обработки, написанной вручную (в ожидании, когда это исправят)

Автотесты в Surf
Flutter: особенности

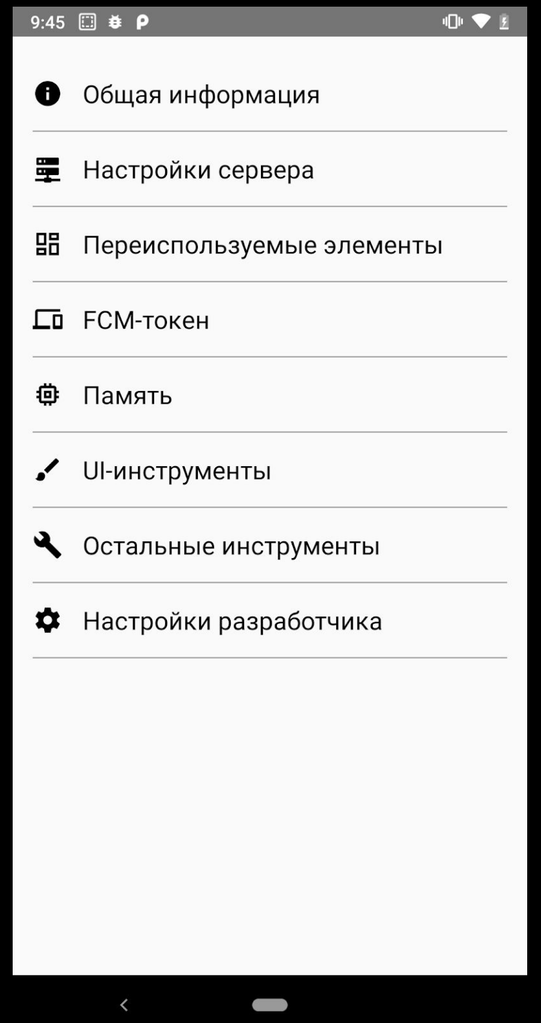
Костыли

- при падении одного теста, падают все без спец.обработки, написанной вручную (в ожидании, когда это исправят)
- прокси на flutter сложнее нативного, в тестах особенно

Автотесты в Surf
Flutter: особенности

Костыли

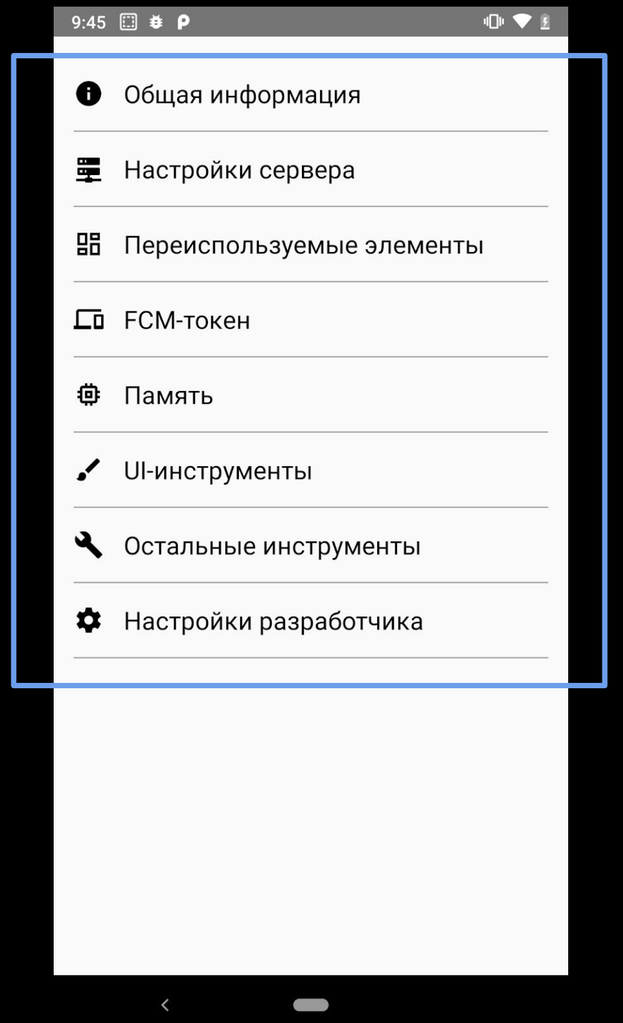
- при падении одного теста, падают все без спец.обработки, написанной вручную (в ожидании, когда это исправят)
- прокси на flutter сложнее нативного, в тестах особенно



Автотесты в Surf
Flutter: особенности

Костыли

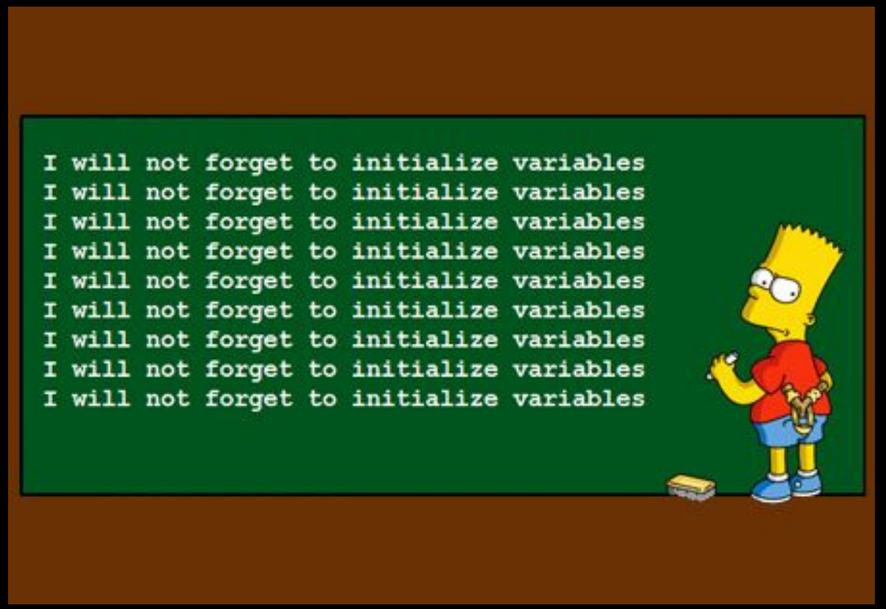
- при падении одного теста, падают все без спец.обработки, написанной вручную (в ожидании, когда это исправят)
- прокси на flutter сложнее нативного, в тестах особенно



Автотесты в Surf
Flutter: особенности

Костыли

- при падении одного теста, падают все без спец.обработки, написанной вручную (в ожидании, когда это исправят)
- прокси на flutter сложнее нативного, в тестах особенно
- иногда приходится сбрасывать GetIt между сценариями



Автотесты в Surf
Flutter: особенности

Костыли

- при падении одного теста, падают все без спец.обработки, написанной вручную (в ожидании, когда это исправят)
- прокси на flutter сложнее нативного, в тестах особенно
- иногда приходится сбрасывать GetIt между сценариями
- нативные алерты

Автотесты в Surf
Flutter: особенности

Костыли

- при падении одного теста, падают все без спец.обработки, написанной вручную (в ожидании, когда это исправят)
- прокси на flutter сложнее нативного, в тестах особенно
- иногда приходится сбрасывать GetIt между сценариями
- нативные алерты

permission

```
Future<void> main() async {
  integration_test_driver.testOutputsDirectory = 'integration_test/gherkin/reports';

  /// Выдаем права на геолокацию чтобы не было проблем
  await Process.run('xcrun', ['simctl', 'privacy', 'booted', 'grant', 'location-always',
<app_packet>]);
  [
    'ACCESS_COARSE_LOCATION',
    'ACCESS_FINE_LOCATION',
  ].forEach((permission) async => Process.run('adb',
      ['shell', 'pm', 'grant', <app_packet>, 'android.permission.$permission']));
  return integration_test_driver.integrationDriver(
    timeout: const Duration(minutes: 120),
    responseDataCallback: writeGherkinReports,
  );
}
```

Автотесты в Surf

Flutter: особенности

Автотесты в Surf
Flutter: особенности

Flutter	Calabash/Appium
Обращение к элементу ○ работа с информацией из виджета (key, значение параметра объекта) + состояние	Обращение к элементу ○ UI распознавание
Скорость ○ быстрое взаимодействие с приложением: раньше известно о состоянии объекта	Скорость ○ более медленное взаимодействие (сторонний драйвер добавляет оверхед)

Автотесты в Surf
Flutter: особенности

Flutter	Calabash/Appium
Расположение тестов ◦ внутри проекта, помощь на pull request’ах ◦ возможность быстро указать локатор для элемента	Расположение тестов ◦ отдельно от проекта ◦ отсутствие возможности быстро указать локатор для элемента

Автотесты в Surf
Flutter: особенности

Flutter	Calabash/Appium
Знание программирования ○ необходимо знать Dart и архитектуру приложения, созданного на flutter, и понимать, что такое виджет-тесты и как с ними работать	Знание программирования ○ достаточно знать специализацию фреймворка, на котором пишутся тесты + нативно обеспечить нужные элементы локаторами

Автотесты в Surf
Flutter: особенности

Flutter	Calabash/Appium
Знание программирования ○ написать специализированные сценарии для widget-тестов и e2e-тестов отдельно (выше покрытие)	Знание программирования ○ написать сценарии только под e2e-тесты

Автотесты в Surf
Flutter: особенности

Flutter	Calabash/Appium
Знание программирования ○ нужно самостоятельно добавлять удобства при необходимости (нет автоматических ожиданий)	Знание программирования ○ существуют уже давно реализованные удобства

Автотесты в Surf
Flutter: особенности

Flutter	Calabash/Appium
Виды автотестов ◦ unit, widget внутри Flutter-приложения ◦ e2e-тесты внутри Flutter-приложения	Виды автотестов ◦ unit нативно внутри приложения ◦ e2e-тесты с помощью фреймворка Calabash/Appium

Автотесты в Surf
Flutter: особенности

Flutter	Calabash/Appium
CI/CD ○ один репозиторий для приложения и тестов	CI/CD ○ минимум два репозитория: для приложения и тестов, соответственно

Автотесты в Surf
Flutter: особенности

Flutter	Calabash/Appium
Скорость ○ тесты относительно быстрые	Скорость ○ тесты относительно быстрые

Резюме

Flutter?
Резюме

- Более близкая интеграция с приложением

Flutter?
Резюме

- Более близкая интеграция с приложением
- Знание программирования, архитектуры приложения, языка Dart, видов тестов во Flutter

Flutter?
Резюме

- Более близкая интеграция с приложением
- Знание программирования, архитектуры приложения, языка Dart, видов тестов во Flutter
- Независимость от разработки, хоть и больше времени на изучение фреймворка

Flutter?
Резюме

- Более близкая интеграция с приложением
- Знание программирования, архитектуры приложения, языка Dart, видов тестов во Flutter
- Независимость от разработки, хоть и больше времени на изучение фреймворка
- Помощь разработке (в поиске багов на pull request’ах)

Flutter?
Резюме

- Более близкая интеграция с приложением
- Знание программирования, архитектуры приложения, языка Dart, видов тестов во Flutter
- Независимость от разработки, хоть и больше времени на изучение фреймворка
- Помощь разработке (в поиске багов на pull request'ах)
- Контроль за большим покрытием
  - компонентные сценарии
  - бизнес-сценарии

Flutter!

Резюме

Время ответить на вопросы

Нативные автотесты кроссплатформенного Flutter

Habr	Telegram	Тестовый проект
habr.com/leshchinskaya	t.me/m_leshchinskaya	github.com/sample_app