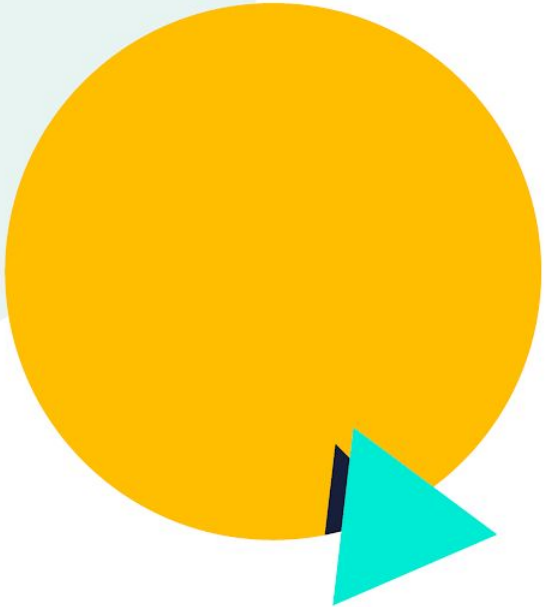
wrike

Прозрачное тестовое покрытие – залог стабильного деплоя



О себе

- В автоматизации тестирования больше 5 лет
- Автоматизировал Web, Desktop, API
- Техлид в Wrike
- Занимаюсь развитием автоматизации в Wrike
- Люблю волейбол и мотоциклы)

varivoda.ivan

ivan-varivoda

vari_voda

План

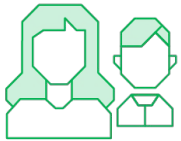
- Wrike и автоматизация тестирования
- Прозрачное тестовое покрытие
- Разметка тестов
- Фильтрация тестов

Wrike

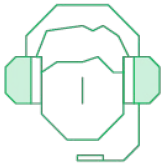
Wrike – Лидер на рынке Work Management

Основан в 2006

9 Офисов



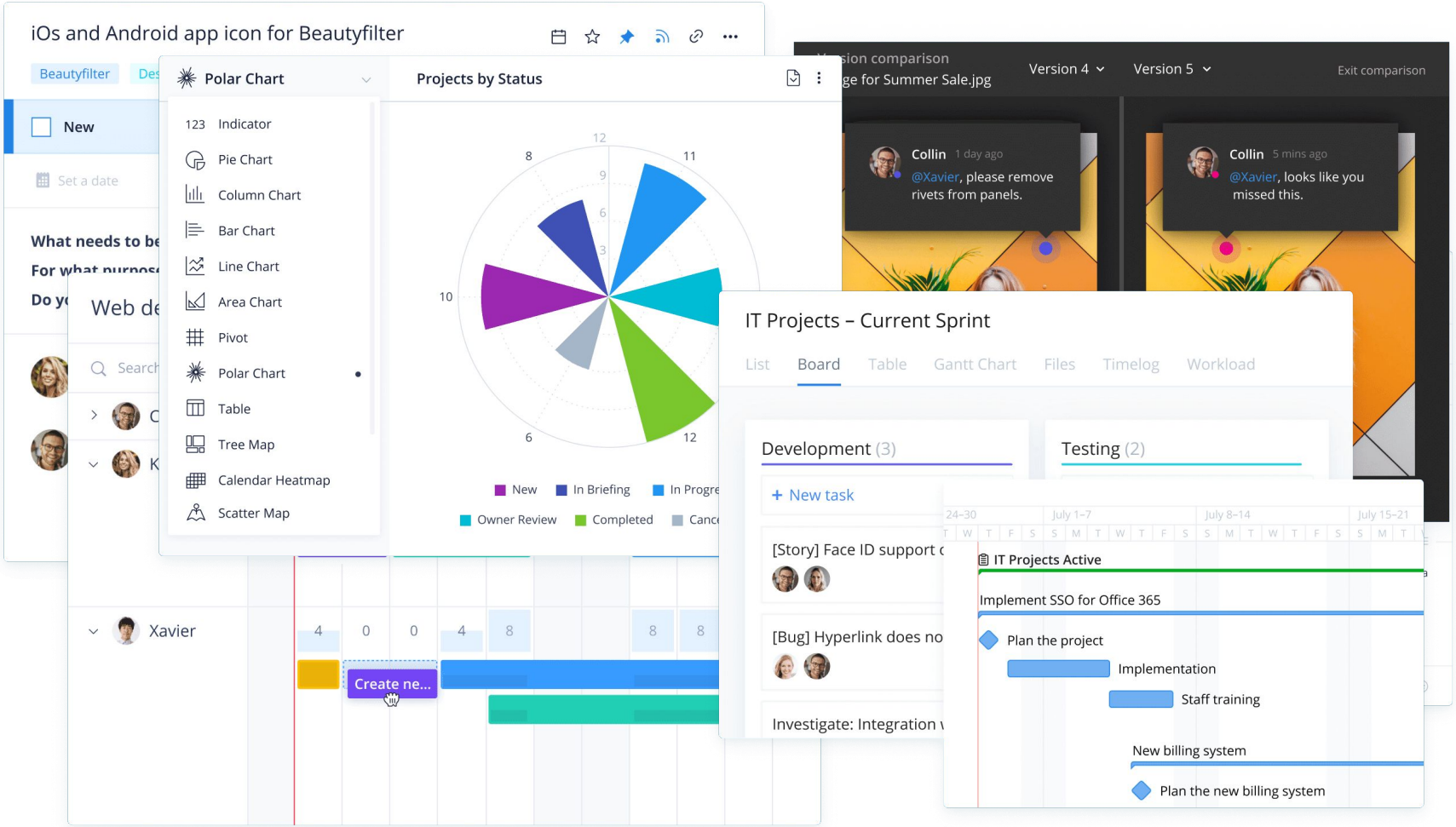
20,000+ Компаний-клиентов



1100+ Сотрудников

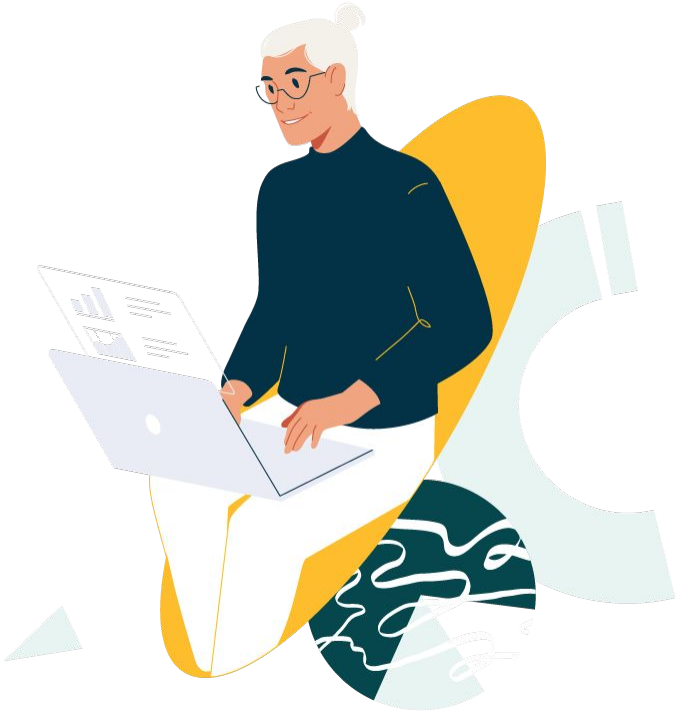
5 лет в Fast 500

Наша платформа для совместного управления рабочими процессами помогает командам добиться прозрачности, упростить планирование, обеспечить совместную работу и оптимизировать свой рабочий процесс.

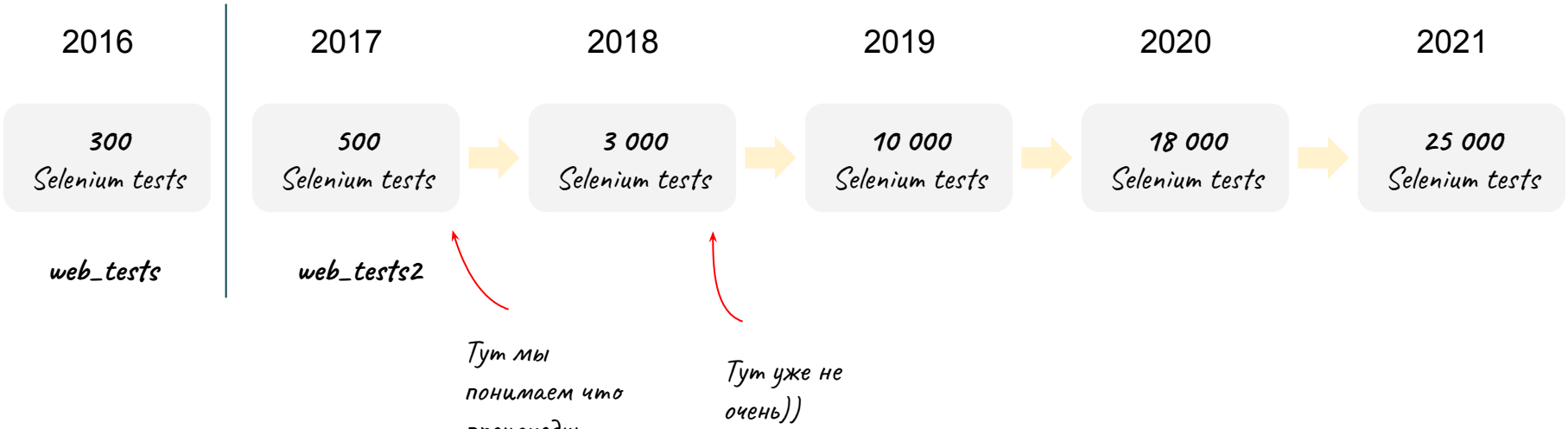

iOs and Android app icon for Beautyfilter
Beautyfilter
New
Set a date
What needs to be
Polar Chart
123 Indicator
Pie Chart
Column Chart
Bar Chart
Line Chart
Area Chart
Pivot
Polar Chart
Table
Tree Map
Calendar Heatmap
Scatter Map
Projects by Status
New
In Briefing
Owner Review
Completed
Version 4
Version 5
Exit comparison
Collin 1 day ago
@Xavier, please remove rivets from panels.
Collin 5 mins ago
@Xavier, looks like you missed this.
IT Projects – Current Sprint
List
Board
Table
Gantt Chart
Files
Timelog
Workload
Development (3)
Testing (2)
+ New task
[Story] Face ID support
[Bug] Hyperlink does no
Investigate: Integration
Xavier
Create ne...
IT Projects Active
Implement SSO for Office 365
Plan the project
Implementation
Staff training
New billing system
Plan the new billing system

Автоматизация тестирования в Wrike

- 25 000 Selenium тестов
- 10 000 API тестов
- 1 000 000 строк кода
- Java, Apache maven, JUnit 5, Retrofit
- Ежедневно запускаем около 250 000
- Выполнение регрессии за 1 час
- Деплой минимум раз в день



История автоматизации



Стали размечать @Category для запуска тестов

```
@RunWith(RetryRunner.class)
@GuiceModules(WebTestsModule.class)
@Category(Folder.class)
@Feature(TASKS)
public class CheckSubFoldersTest {
```

Стали размечать @Category для запуска тестов

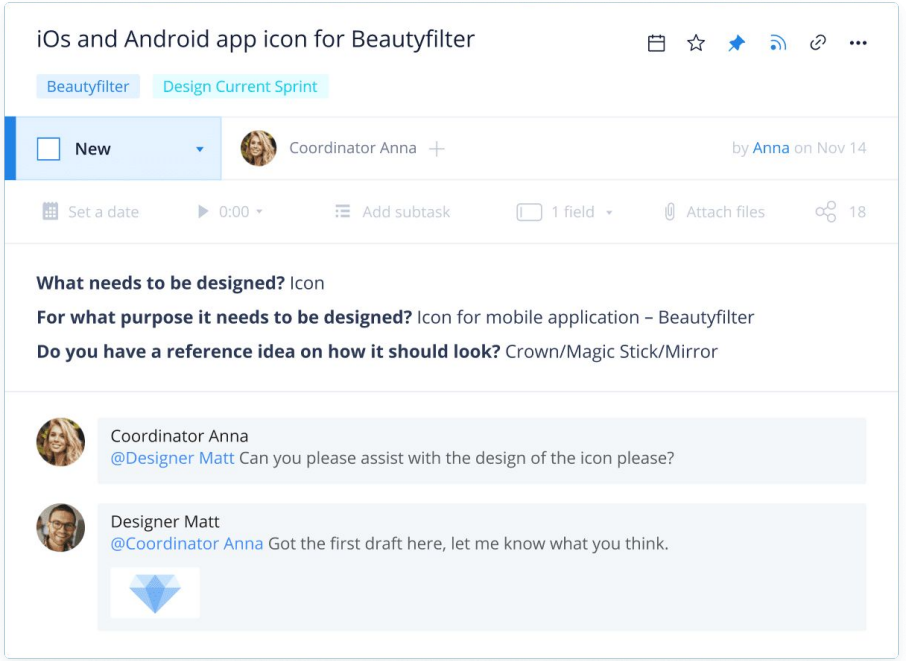
```
@RunWith(RetryRunner.class)
@GuiceModules(WebTestsModule.class)
@Category({AllTests.class, WsRegress.class,
        Tasks.class, FirefoxRegress.class})
@Feature(TASKS)
public class CheckSubFoldersTest {
```

Стали размечать @Category для запуска тестов

```
@RunWith(RetryRunner.class)
@GuiceModules(WebTestsModule.class)
@Category({AllTests.class, WsRegress.class,
           Tasks.class, DragAndDrop.class,
            Smoke.class, ScreenshotTest.class
            FirefoxRegress.class, Folder.class})
@Feature(TASKS)
public class CheckSubFoldersTest {
```

Прозрачное тестовое покрытие

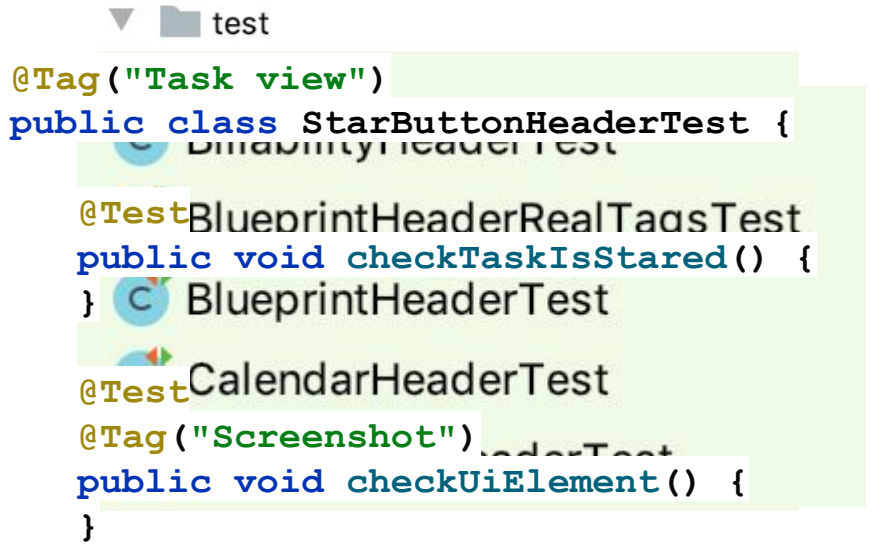
- Сколько тестов API?
- Сколько тестов Selenium?
- Визуальная часть / проверка логики
- Какая часть логики покрыта?
- Много интеграций данного компонента?



Разметка тестов

Как вообще размечают тесты?

- Никак – небольшой репозиторий
- По структуре проекта – пакеты
- По названиям классов
- @Аннотации
  - JUnit 5 @Tag
  - Allure Report

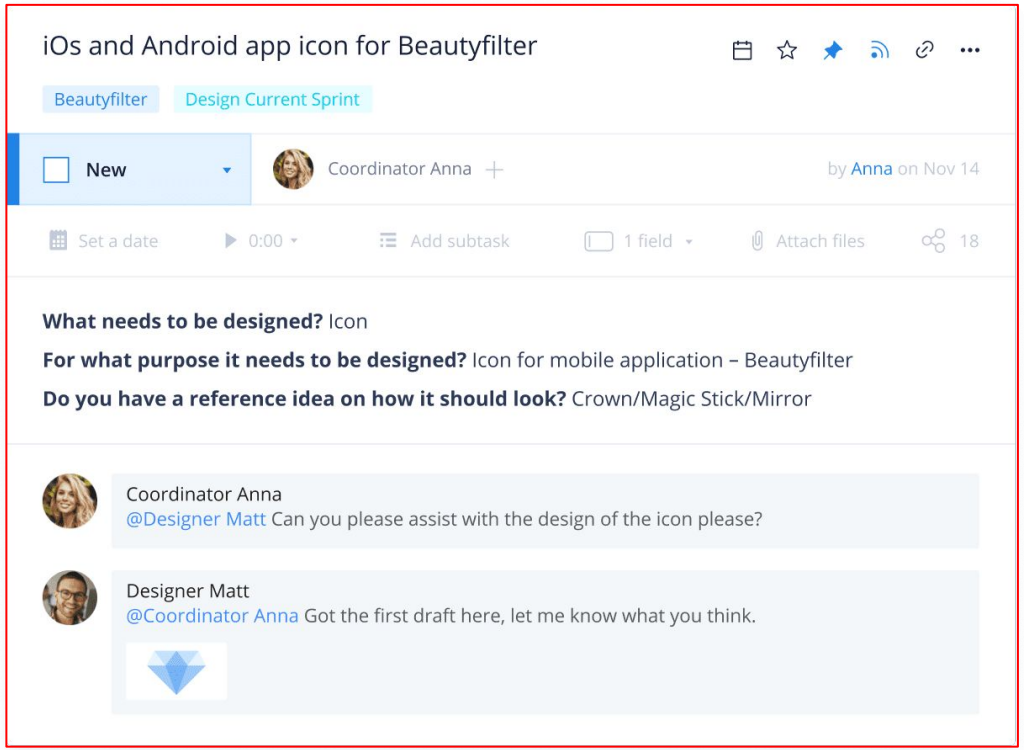


Аннотации Allure report

```
@Epic("Самостоятельное приложение")
@Feature("Конкретная фича эпика")
@Story("Детализация фичи")
public class ТестовыйКласс {
```

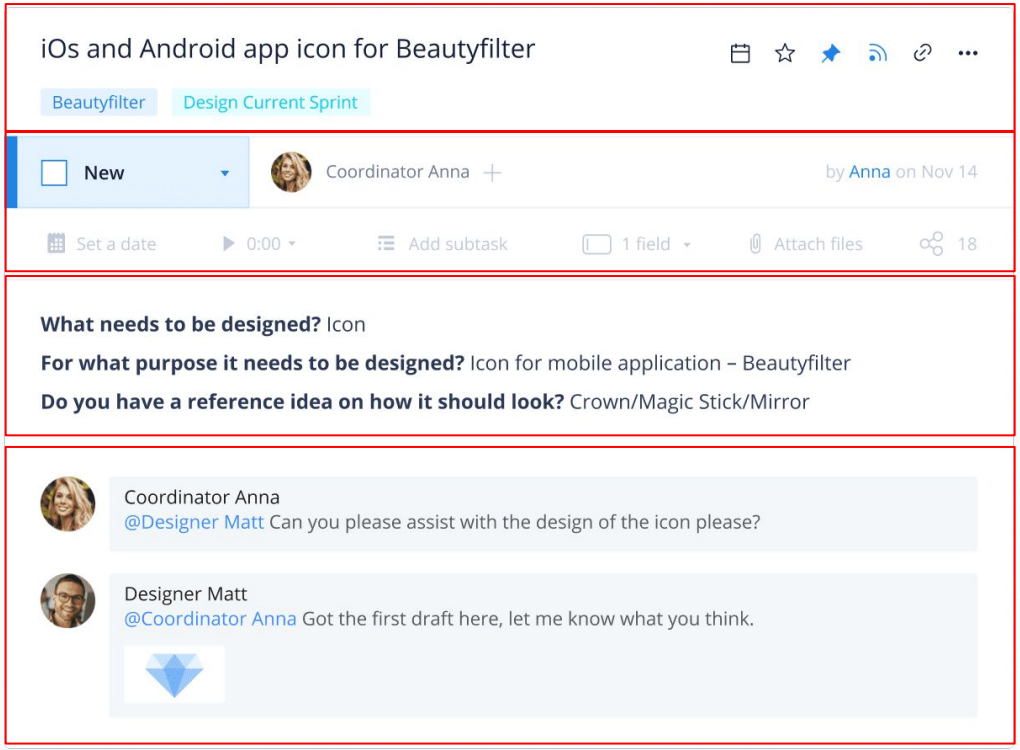
На примере Task view

Epic: Task view



На примере Task view

Feature: Header



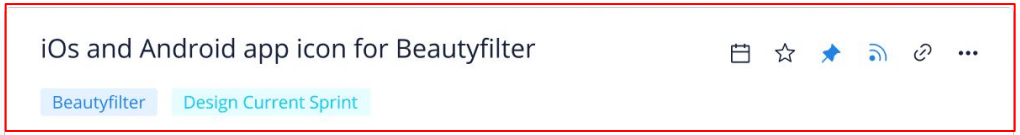
Feature: Status bar

Feature: Live editor

Feature: Stream

На примере Task view

Feature: Header

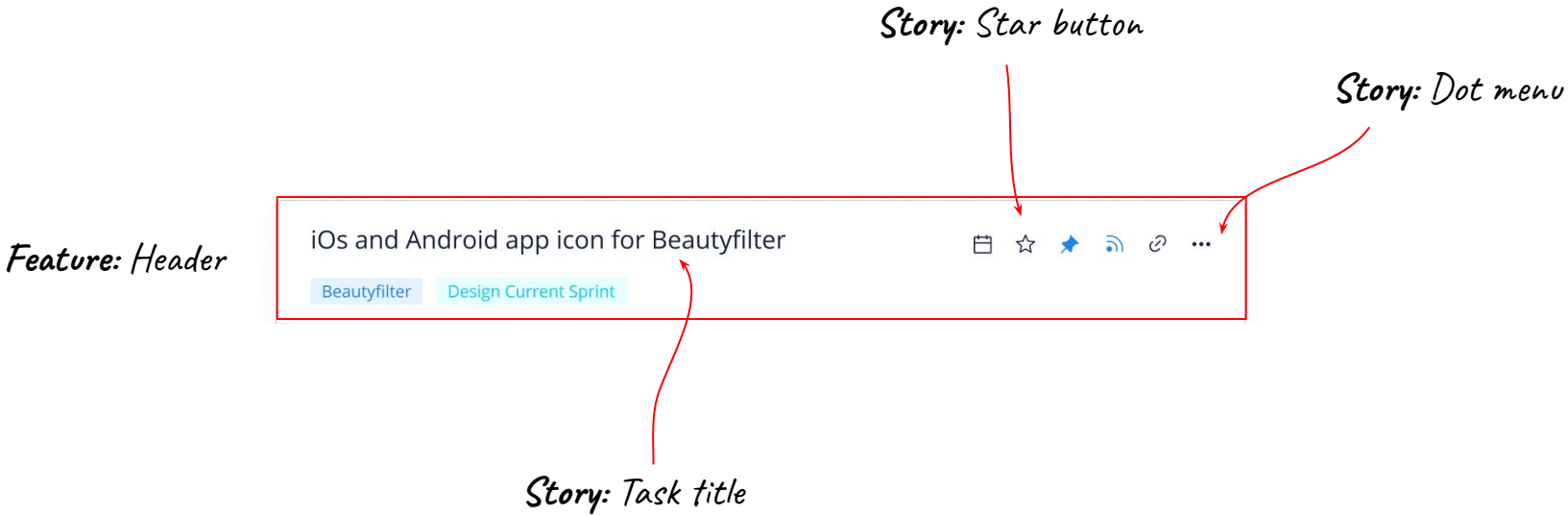


Feature: Status bar

Feature: Live editor

Feature: Stream

На примере Task view



Как выглядит в коде

```
@Epic("Task view")
@Feature("Header")
@Story("Star button")
public class StarButtonHeaderTest {

    @Test
    public void checkTaskIsStaredViaButton() {
      //   Базовая проверка
    }

    @Test
    public void checkSelectedStarButtonScreenshot() {
      //  Проверка интерфейса
    }

}
```

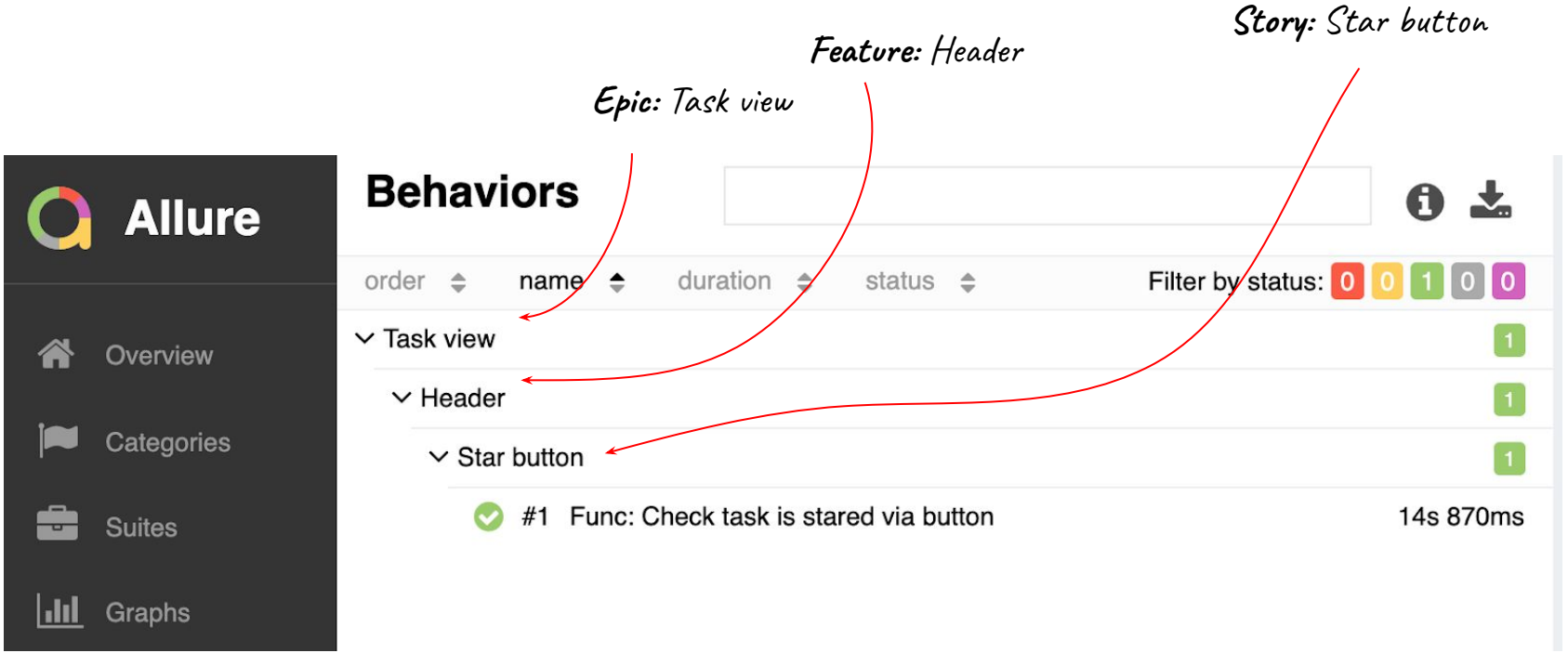
Чем хороша разметка Allure?

- Простота и понятность всем членам команды
- Достаточность
- Красота в Allure отчете
- Удобство разметки интеграционных тестов

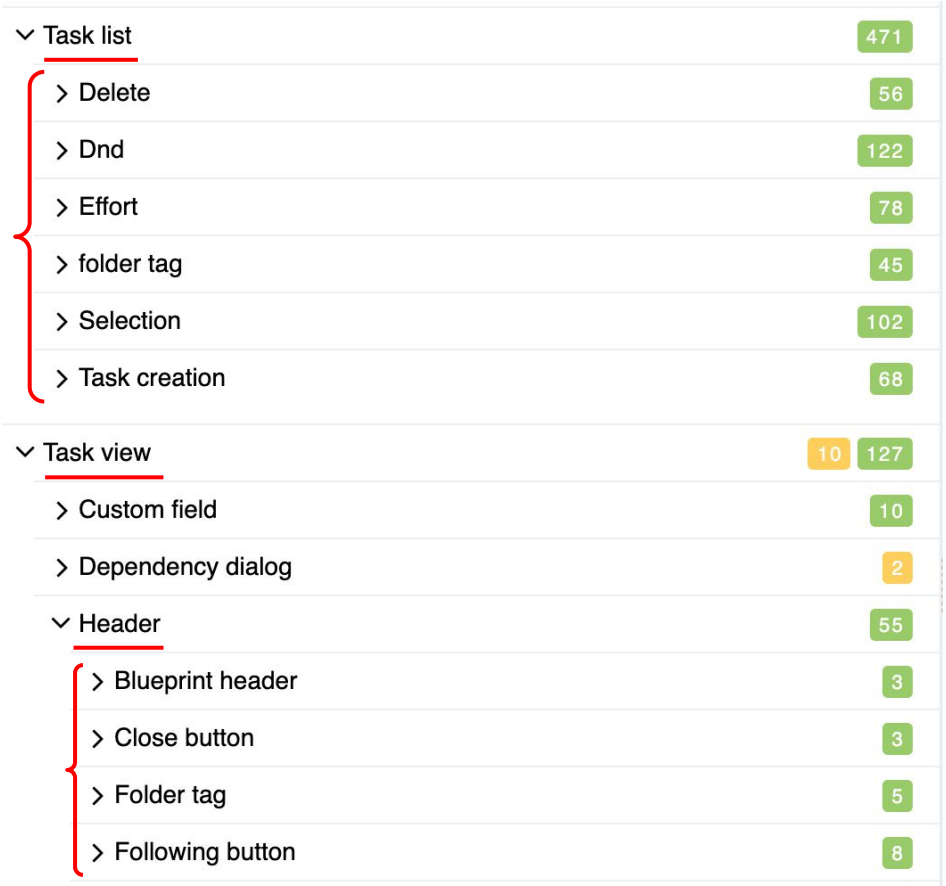
Чем хороша разметка Allure?

- Простота и понятность всем членам команды
- Достаточность
- **Красота в Allure отчете**
- Удобство разметки интеграционных тестов

Визуализация в Allure report



Пример на большом проекте

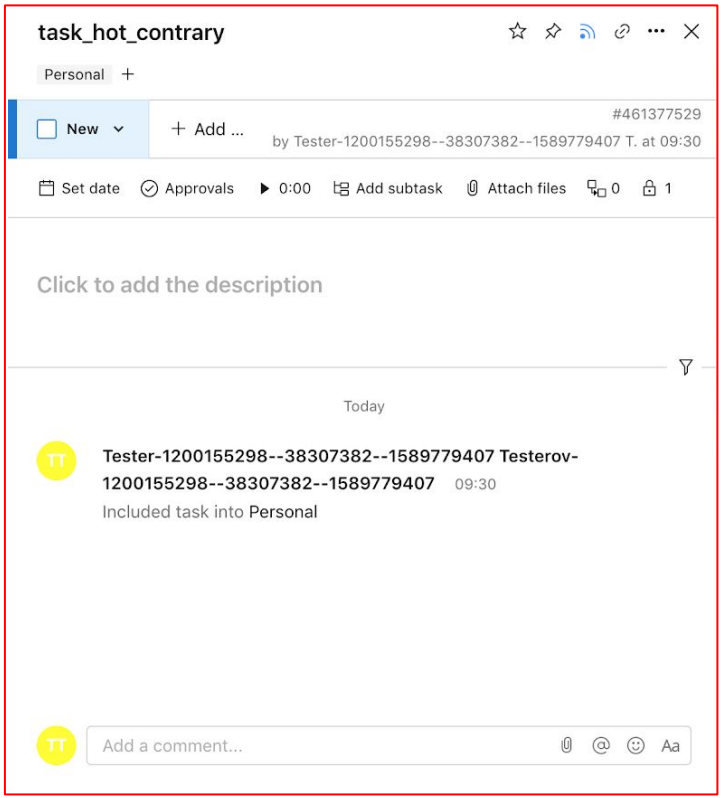


Чем хороша разметка Allure?

- Простота и понятность всем членам команды
- Достаточность
- Красота в Allure отчете
- **Удобство разметки интеграционных тестов**

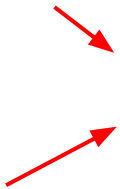
Разметка интеграционных тестов

Epic: Task list



Epic: Task view

Разметка интеграционных тестов



```
//Интеграционные тесты размечены несколькими @Epic

@Epic("Task list")
@Epic("Task view")
@Feature("Open task")
public class TaskViewAndTaskListIntegrationTest {

    @Test
    public void checkTaskOpenFromTaskList() {

    }
```

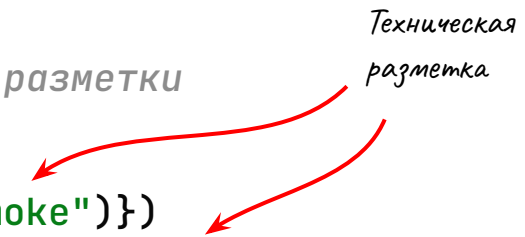
Есть не только бизнес разметка

- Smoke pack
- Screenshot tests
- A11y tests
- Performance
- API tests
- Что-то еще

Нужна кастомизация

Техническая разметка

```
// Не стоит смешивать бизнес и техническую разметки

@Epic("Task view")
@Features({@Feature("Header"), @Feature("Smoke")})
@Stories({@Story("Star button"), @Story("Performance")})
public class StarButtonHeaderTest {

    @Test
    public void checkTaskIsStaredViaButton() {
    }
```



Техническая разметка

```
// Можно создавать свои аннотации для Tex разметки
```

```
@Retention(RUNTIME)
@Target({METHOD, TYPE})
public @interface Type {
   String value();
}
```

Пример технической разметки

```
@Epic("Task view")
@Feature("Header")
@Story("Star button")
public class StarButtonHeaderTest {


    @Test
    public void checkTaskIsStaredViaButton() {
        // Проверка критического функционала
    }

    @Test
    public void checkSelectedStarButtonScreenshot() {
        // Не такая важная проверка
    }
```

Пример технической разметки

```java
@Epic("Task view")
@Feature("Opening")
public class OpenTaskViewTest {


    @Test
    public void checkTaskViewOpenCorrectly() {
        // Тест проверяет сервис на BE неявно
    }
```

Пример технической разметки

```java
@Epic("Task view")
public class TaskViewGetInitDataTest {

    @Test
    public void checkTaskTitleFromResponse() {
        // API тест проверяет основной Handler
    }

}
```

Бизнес разметка

```java
@Epic("Task view")
@Feature("Header")
@Story("Star button")
public class StarButtonHeaderTest {

    @Test
    @Service(TASK_SERVICE)
    public void checkTaskIsStaredViaButton() {
        // Проверка UI для BE task сервиса
    }

    @Test
    @Type("Smoke")
    public void checkSelectedStarButtonScreenshot() {
        // Тест на критический функционал
    }

}
```

Техническая разметка

Чем хороша

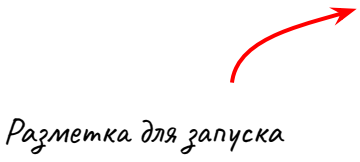
- Разметка на человеческом языке
- Визуализация покрытия
- Полная информация о тесте в коде
- Расширяемая система тестовых координат

Фильтрация тестов

Фильтрация тестов из коробки

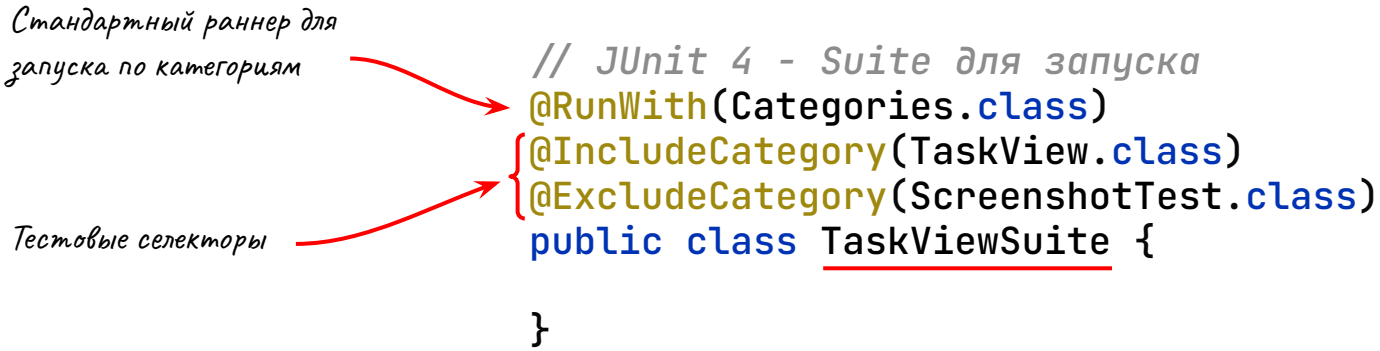
- JUnit 4
- JUnit 5
- TestNG

Фильтрация тестов JUnit 4

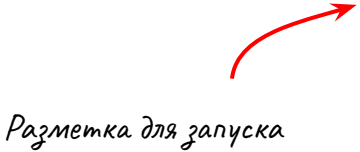



```
// JUnit 4 - @Category для разметки
@Category(TaskView.class)
@Epic("Task view")
@Feature("Header")
@Story("Star button")
public class StarButtonHeaderTest {

    @Test
    public void checkTaskIsStaredViaButton() {
        // Проверка критического функционала
    }

}
```

Фильтрация тестов JUnit 4

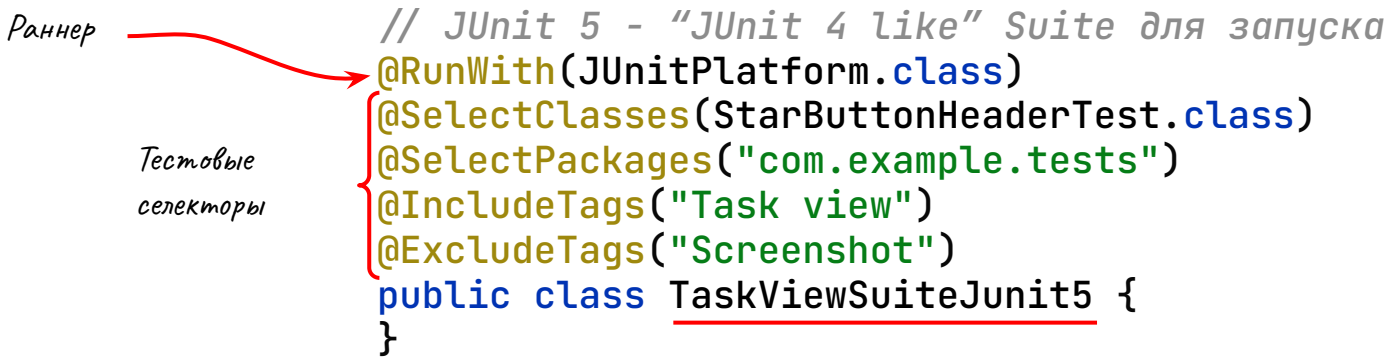


Фильтрация тестов JUnit 5




```java
// JUnit 5 - @Tag для разметки
@Tag("Task view")
@Epic("Task view")
@Feature("Header")
@Story("Star button")
public class StarButtonHeaderTest {

    @Test
    public void checkTaskIsStaredViaButton() {
        // Проверка критического функционала
    }

}
```

Фильтрация тестов JUnit 5



Фильтрация тестов JUnit 5

Операции с множествами **!, &, |**

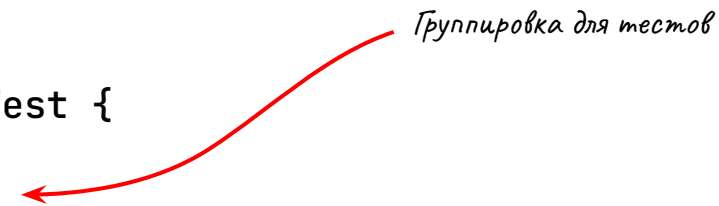
mvn test -Dgroup=**(Task view) & !(Screenshot)**

gradle test -Dgroup= **!(Screenshot)**

```
// JUnit 5 - Tag Expressions для запуска
@Tag("Task view")
public class StarButtonHeaderTest {

    @Tag("Smoke")
    @Test
    public void checkTaskIsStared() {
        // Проверка критического функционала
    }

    @Tag("Screenshot")
    @Test
    public void checkSelectedStar() {
        // Не такая важная проверка
    }
```

Фильтрация тестов TestNG

```java
// TestNG - Groups в @Test
@Epic("Task view")
@Feature("Header")
@Story("Star button")
public class StarButtonHeaderTest {

    @Test(groups="Star button")
    public void checkTaskIsStaredViaButton() {
        // Проверка критического функционала
    }

}
```



Фильтрация тестов TestNG

```
<!--В testng.xml фильтруем нужные наборы-->
<suite name="Suite" verbose="1">
   <test name="Test">
       <classes>
           <class name="com.wrike.webtests.taskview.StarButtonHeaderTest"/>
       </classes>
       <packages>
           <package name="com.wrike.webtests.taskview"/>
       </packages>
       <groups>
           <run>
               <include name="Task view"/>
               <exclude name="brokenTests"/>
           </run>
       </groups>
   </test>
</suite>
```

Берем фильтрацию из коробки и
поехали?

Стали размечать @Category для запуска тестов

```
@RunWith(RetryRunner.class)
@GuiceModules(WebTestsModule.class)
@Category({AllTests.class, WsRegress.class,
           Tasks.class, DragAndDrop.class,
           Smoke.class, ScreenshotTest.class
           FirefoxRegress.class, Folder.class})
@Feature(TASKS)
public class CheckSubFoldersTest {
```

Стали размечать @Category для запуска тестов

```
@RunWith(RetryRunner.class)
@GuiceModules(WebTestsModule.class)
@Types({@Type(Smoke)}, @Type(Screenshot), @Type(Firefox)})
@Epic(WORK_SPACE)
@Features({@Feature(TASKS)}, @Feature(FOLDER)})
@Story(DragAndDrop)
public class CheckSubFoldersTest {
```

Своя фильтрация тестов

- Тестовый фильтр
- Модуль, вычисляющий выражения тестовых фильтров
- Адаптеры
  - JUnit 4
  - JUnit 5
  - TestNG

Своя фильтрация тестов

- **Тестовый фильтр**
- Модуль, вычисляющий выражения тестовых фильтров
- Адаптеры
  - JUnit 4
  - JUnit 5
  - TestNG

Тестовый фильтр

- Строка определенного формата
- Множество тестов: **{key:value}**
  - Key – аннотация
    - пример: ***epic, feature, story, type***
  - Value **–** значение аннотации
    - пример: ***Task view***
- Операции с множествами: **!**, **|**, **&**
- Доп символы: **(**, **)**

Тестовый фильтр

```
// Примеры тестовых фильтров
```

```
{epic:Task view}

{epic:Task view}&{feature:Header}

{epic:Task view}&{type:Smoke}

{service:Task service}
```

```java
@Epic("Task view")
@Feature("Header")
@Story("Star button")
public class StarButtonHeaderTest {


    @Test
    @Service(TASK_SERVICE)
    public void checkTaskIsStared() {
    }

    @Test
    @Type("Smoke")
    public void checkStarScreenshot() {
    }

}
```

Тестовый фильтр

// Реальные примеры

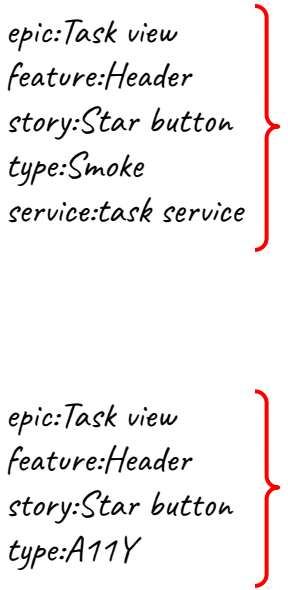
```
{epic:Project progress}&!
({type:Long Tests}|{epic:Site}|{epic:Transcoding}|{feature:Media Valet})
```

```
{epic:Checkbox}|{epic:Form field}|{type:AutoSmoke2Usages}
```

Своя фильтрация тестов

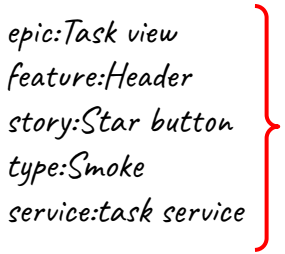
- Тестовый фильтр
- **Модуль, вычисляющий выражения тестовых фильтров**
- Адаптеры
  - JUnit 4
  - JUnit 5
  - TestNG

Вычисление тестовых фильтров

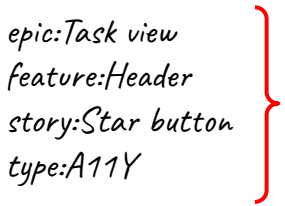



```java
@Epic("Task view")
@Feature("Header")
@Story("Star button")
public class StarButtonHeaderTest {


    @Test
    @Type("Smoke")
    @Service(TASK_SERVICE)
    public void checkTaskIsStared() {
    }

    @Test
    @Type("A11Y")
    public void checkA11YAttr() {
    }

}
```

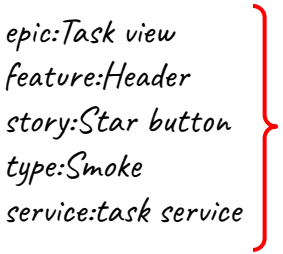
Вычисление тестовых фильтров



{epic:Task view}&{type:Smoke}



Вычисление тестовых фильтров



{epic:Task view}&{type:Smoke}

Вычисление тестовых фильтров

epic:Task view
feature:Header
story:Star button
type:Smoke
service:task service

{epic:Task view}&{type:Smoke}

epic:Task view
feature:Header
story:Star button
type:A11Y

Вычисление тестовых фильтров

epic:Task view
feature:Header
story:Star button
type:Smoke
service:task service

true&{type:Smoke}

epic:Task view
feature:Header
story:Star button
type:A11Y

Вычисление тестовых фильтров

epic:Task view
feature:Header
story:Star button
type:Smoke
service:task service

true&{type:Smoke}

epic:Task view
feature:Header
story:Star button
type:A11Y

Вычисление тестовых фильтров

epic:Task view
feature:Header
story:Star button
type:Smoke
service:task service

true&{type:Smoke}

epic:Task view
feature:Header
story:Star button
type:A11Y

Вычисление тестовых фильтров

epic:Task view
feature:Header
story:Star button
type:Smoke
service:task service

true&**{type:Smoke}**

epic:Task view
feature:Header
story:Star button
type:A11Y

Вычисление тестовых фильтров

epic:Task view
feature:Header
story:Star button
type:Smoke
service:task service

true&**true**

epic:Task view
feature:Header
story:Star button
type:A11Y

Вычисление тестовых фильтров

epic:Task view
feature:Header
story:Star button
type:Smoke
service:task service

true&true

epic:Task view
feature:Header
story:Star button
type:A11Y

Вычисление тестовых фильтров

epic:Task view
feature:Header
story:Star button
type:Smoke
service:task service

true&true

true

epic:Task view
feature:Header
story:Star button
type:A11Y

Вычисление тестовых фильтров

epic:Task view
feature:Header
story:Star button
type:Smoke
service:task service

true&true

true

epic:Task view
feature:Header
story:Star button
type:A11Y

{epic:Task view}&{type:Smoke}

Вычисление тестовых фильтров

epic:Task view
feature:Header
story:Star button
type:Smoke
service:task service

true&true

true

epic:Task view
feature:Header
story:Star button
type:A11Y

{epic:Task view}&{type:Smoke}

Вычисление тестовых фильтров

epic:Task view
feature:Header
story:Star button
type:Smoke
service:task service

true&true

true

epic:Task view
feature:Header
story:Star button
type:A11Y

true&{type:Smoke}

Вычисление тестовых фильтров

epic:Task view
feature:Header
story:Star button
type:Smoke
service:task service

true&true

true

epic:Task view
feature:Header
story:Star button
type:A11Y

true&{type:Smoke}

Вычисление тестовых фильтров

epic:Task view
feature:Header
story:Star button
type:Smoke
service:task service

true&true

true

epic:Task view
feature:Header
story:Star button
type:A11Y

true&{type:Smoke}

Вычисление тестовых фильтров

epic:Task view
feature:Header
story:Star button
type:Smoke
service:task service

true&true

true

epic:Task view
feature:Header
story:Star button
type:A11Y

true&{type:Smoke}

Вычисление тестовых фильтров

epic:Task view
feature:Header
story:Star button
type:Smoke
service:task service

true&true

true

epic:Task view
feature:Header
story:Star button
type:A11Y

true&**{type:Smoke}**

Вычисление тестовых фильтров

epic:Task view
feature:Header
story:Star button
type:Smoke
service:task service

true&true

true

epic:Task view
feature:Header
story:Star button
type:A11Y

true&**false**

Вычисление тестовых фильтров

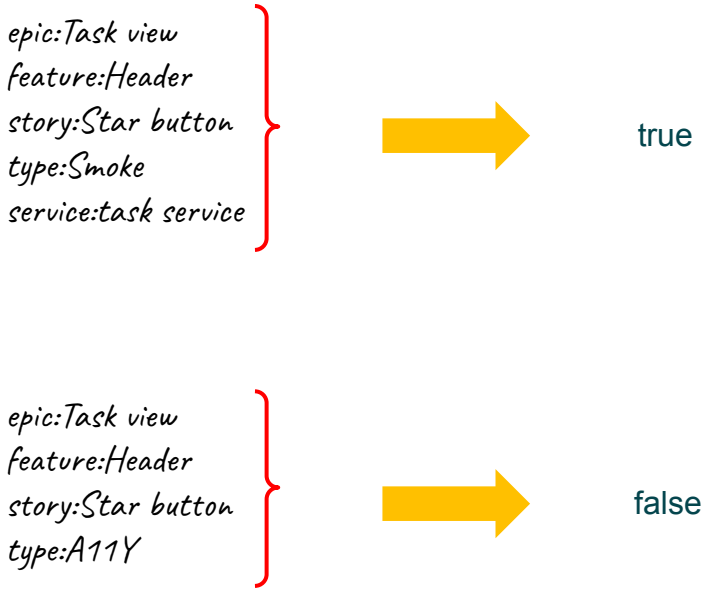
epic:Task view
feature:Header
story:Star button
type:Smoke
service:task service

true&true → true

epic:Task view
feature:Header
story:Star button
type:A11Y

true&false → false

Вычисление тестовых фильтров



Своя фильтрация тестов

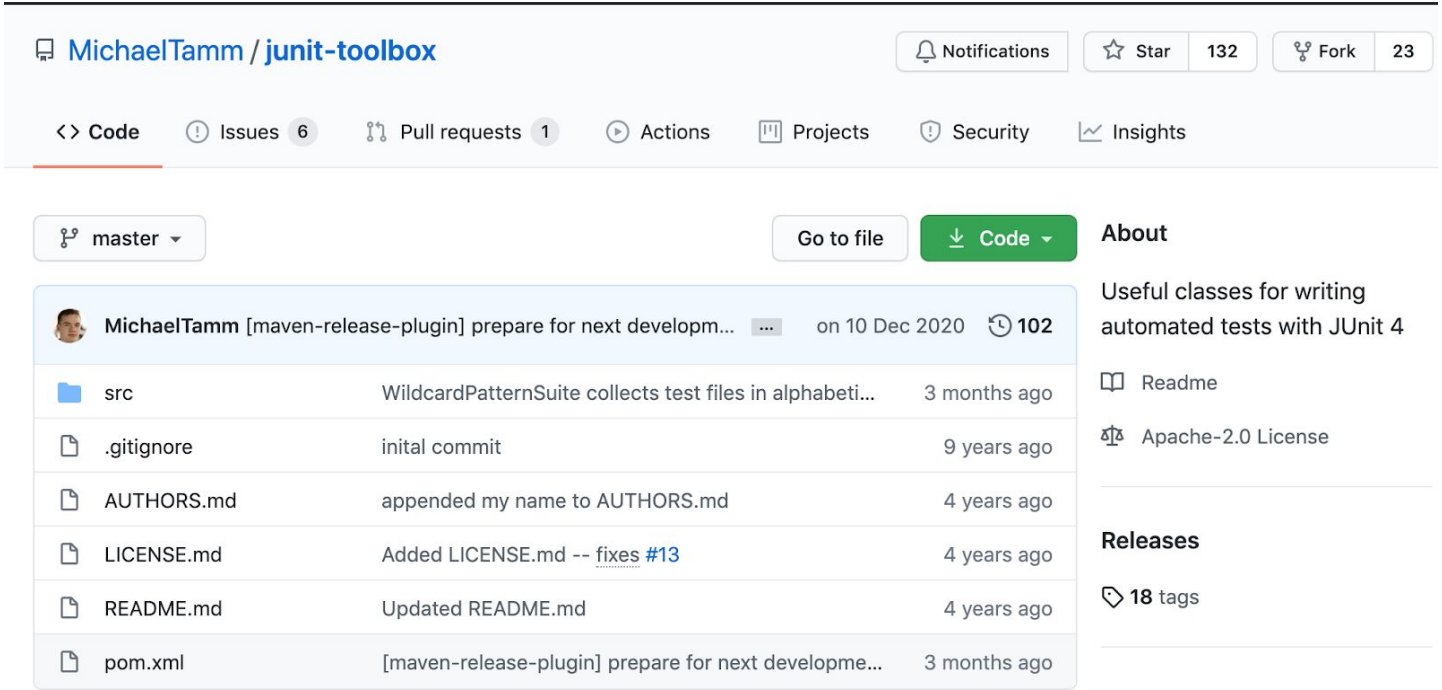
- Тестовый фильтр
- Модуль, вычисляющий выражения тестовых фильтров
- **Адаптеры**
  - JUnit 4
  - JUnit 5
  - TestNG

Адаптер JUnit 4

- Загрузить при запуске все тесты
- Отфильтровать

```java
// JUnit 4 - Suite для запуска
@RunWith(Categories.class)
@IncludeCategory(TaskView.class)
public class TaskViewSuite {

}
```

Адаптер Junit4



[junit-toolbox]

Адаптер Junit4

```
// Основа: WildcardPatternSuite
```

```
@SuiteClasses({"**/*.class",
        "!**/TaskViewSuite.class"})
@RunWith(WildcardPatternSuite.class)
public class TaskViewSuite {

}
```

Адаптер JUnit 4

```
// Наследуемся от WildcardPatternSuite для фильтрации


public class TestFilterSuite extends WildcardPatternSuite {

    public TestFilterSuite(Class<?> klass, RunnerBuilder builder){

        super(klass, builder);
        filter(new TestFilterJunit4());
    }
}
```

Адаптер JUnit 4

```
// Основной раннер ParentRunner реализует Filterable
```

```
public interface Filterable {
    void filter(Filter filter) throws NoTestsRemainException;
}
```

Адаптер JUnit 4

```
// Реализация фильтрации


public class TestFilterJunit4 extends Filter {

    private TestFilter filter = new TestFilter(TestConfig.getTestFilter());

    @Override
    public boolean shouldRun(Description description) {

        if (!description.isTest()) {
            return true;
        }

        Method method = getMethodFromDescription(description);
        Map<String, String> annotationValues = getAllTestCoordinates(method);

        return filter.match(annotationValues);
    }
```

Адаптер JUnit 5

```
// Conditional Test Execution
```

```
@FunctionalInterface
@API(status = STABLE, since = "5.0")
public interface ExecutionCondition extends Extension {

  /**
   * Evaluate this condition for the supplied {@link ExtensionContext}.
   */
  ConditionEvaluationResult evaluateExecutionCondition(ExtensionContext context);

}
```

Адаптер JUnit 5

```
// Пишем свой Condition TestFilterExecutionCondition


public class TestFilterExecutionCondition implements ExecutionCondition {

    private final TestFilter filter = new TestFilter(TestConfig.getTestFilter());

    @Override
    public ConditionEvaluationResult evaluateExecutionCondition(ExtensionContext context) {

      // Код аналогичный из JUnit 4 suite

    }

}
```

Адаптер TestNG

```
// В TestNG фильтрация через IInvokedMethodListener
```

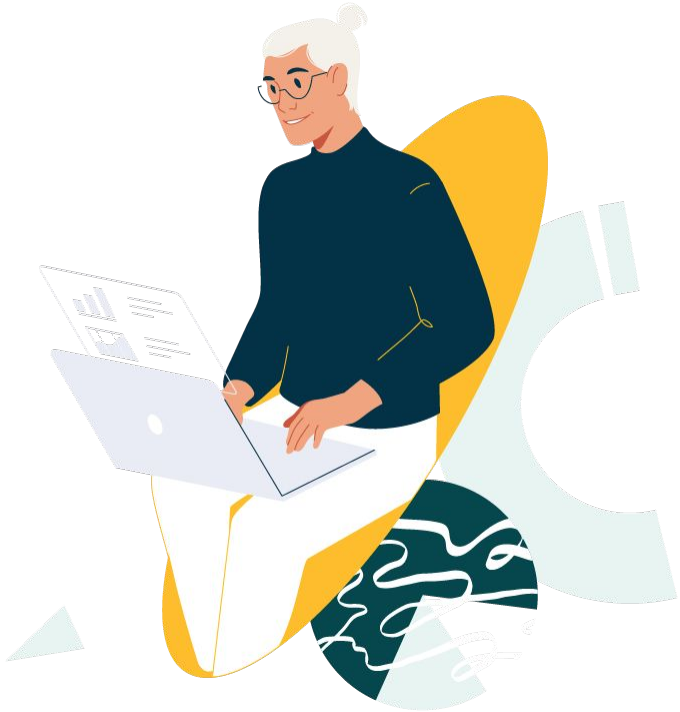
```
public interface IInvokedMethodListener extends ITestNGListener {

 default void beforeInvocation(IInvokedMethod method, ITestResult testResult) {
   // not implemented
 }

 default void afterInvocation(IInvokedMethod method, ITestResult testResult) {
   // not implemented
 }

  ...

}
```

Адаптер TestNG

```
// Реализуем свой Listener


public class TestFilterListener implements IInvokedMethodListener {

    private TestFilter filter = new TestFilter(TestConfig.getTestFilter());

    @Override
    public void beforeInvocation(IInvokedMethod method, ITestResult testResult) {

      // Код аналогичный из JUnit 4 suite

    }

}
```

Своя фильтрация тестов

- Тестовый фильтр
- Модуль, вычисляющий выражения тестовых фильтров
- **Адаптеры**
  - **JUnit 4**
  - **JUnit 5**
  - **TestNG**



Демо

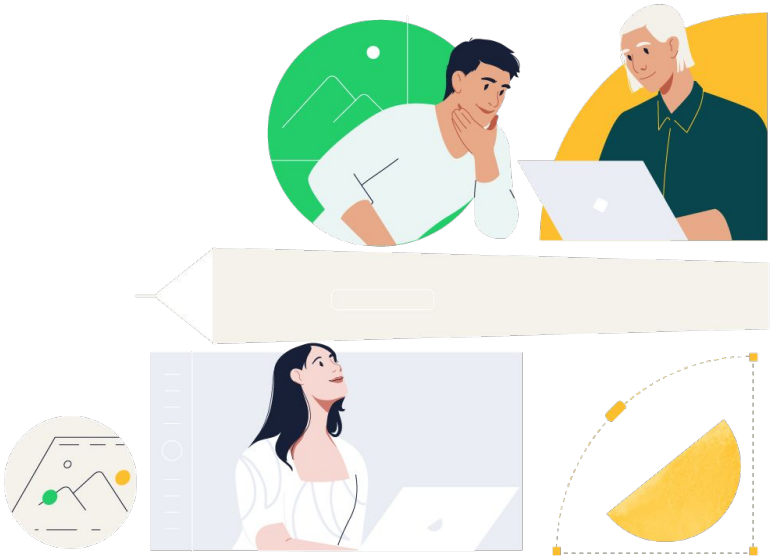
Проект на гитхабе

github.com/varivoda/test_filter

Своя фильтрация тестов

// В чем еще может пригодиться?

- Фильтрация сломанных тестов
- Фильтрация нестабильных тестов
- Фильтрация тестов по своим признакам



Дополнительная фильтрация тестов

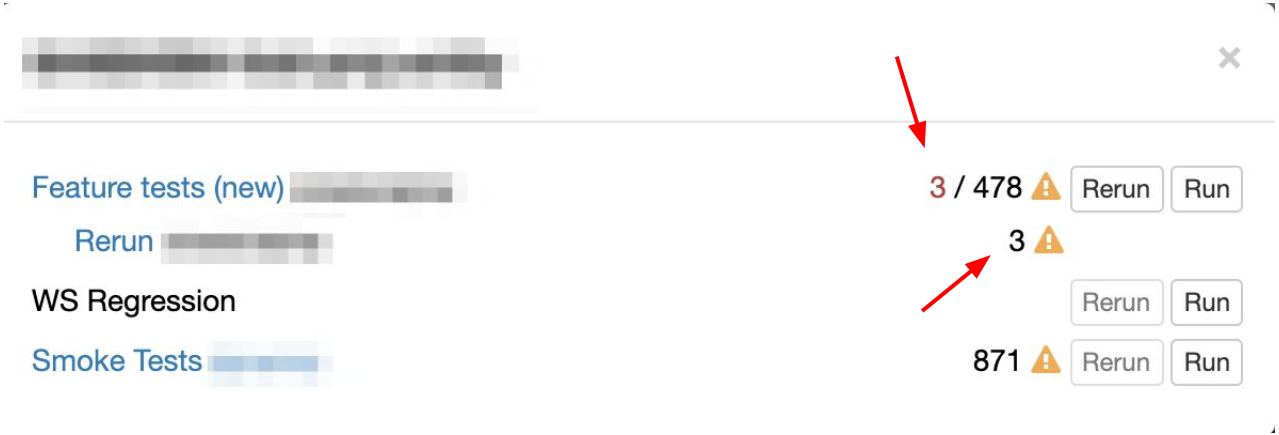
// Фильтрация по тестовым ID

- Новое множество: **{id1,id2,...}**
- Пример тестового фильтра: *{34560,3210}*

```
@TestCaseId(34560)
@Test
public void checkTaskIsUnStared() {
    //   Тут проверка
}
```

Дополнительная фильтрация тестов

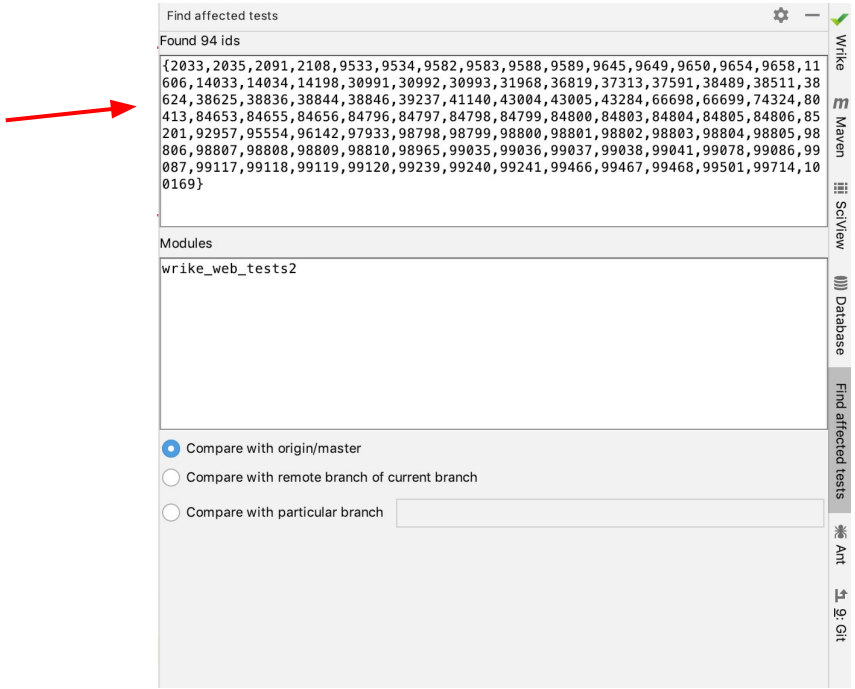
// Перезапуск по тестовому фильтру




```
"failedIds":"{70323,70324,77399}"
```

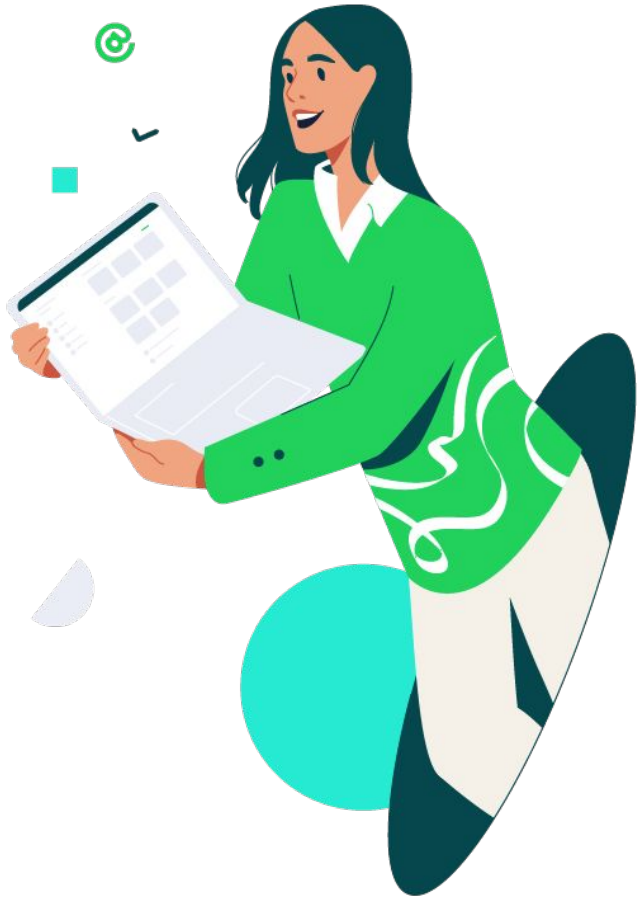
Дополнительная фильтрация тестов

// Поиск затронутых тестов рефакторингом



Выводы

- Используйте Allure аннотации для бизнес разметки тестов
- Для технической разметки тестов используйте свои аннотации
- Эта разметка повысит прозрачность тестового покрытия
- Используйте запуск тестов по фильтру
- Находите баги как можно раньше и повышайте качество деплоя



wrike

Спасибо за внимание!

varivoda.ivan

vari_voda

ivan-varivoda