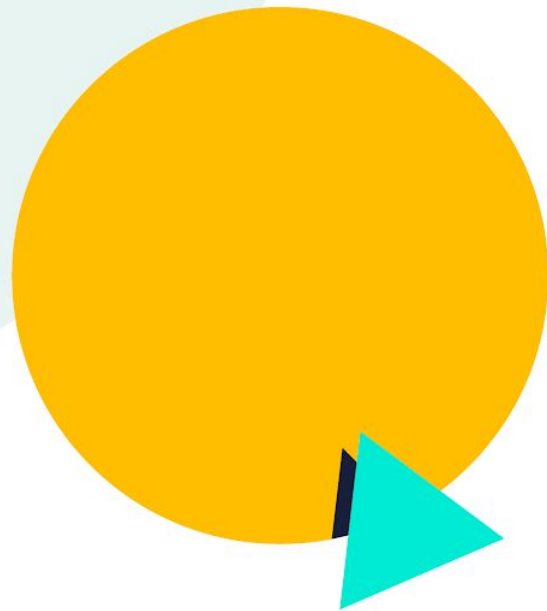




**Прозрачное тестовое  
покрытие – залог  
стабильного деплоя**



# О себе

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



# План

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

**Wrike**



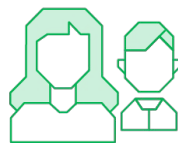
# Wrike – Лидер на рынке Work Management



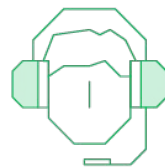
Основан в  
2006



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.**

Beautyfilter | Design

New

Set a date

What needs to be done for what purpose?

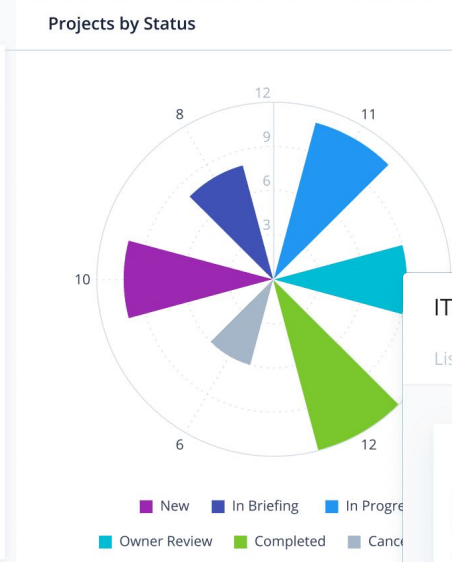
Do you need a Web design?

Search

Collin

Xavier

- Polar Chart**
- 123 Indicator
  - Pie Chart
  - Column Chart
  - Bar Chart
  - Line Chart
  - Area Chart
  - Pivot
  - Polar Chart**
  - Table
  - Tree Map
  - Calendar Heatmap
  - Scatter Map



Version comparison

Image for Summer Sale.jpg

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 not work

Investigate: Integration with new billing system

IT Projects Active

- Implement SSO for Office 365
  - Plan the project
  - Implementation
  - Staff training
- New billing system
  - Plan the new billing system

Xavier

4 | 0 | 0 | 4 | 8 | 8 | 8

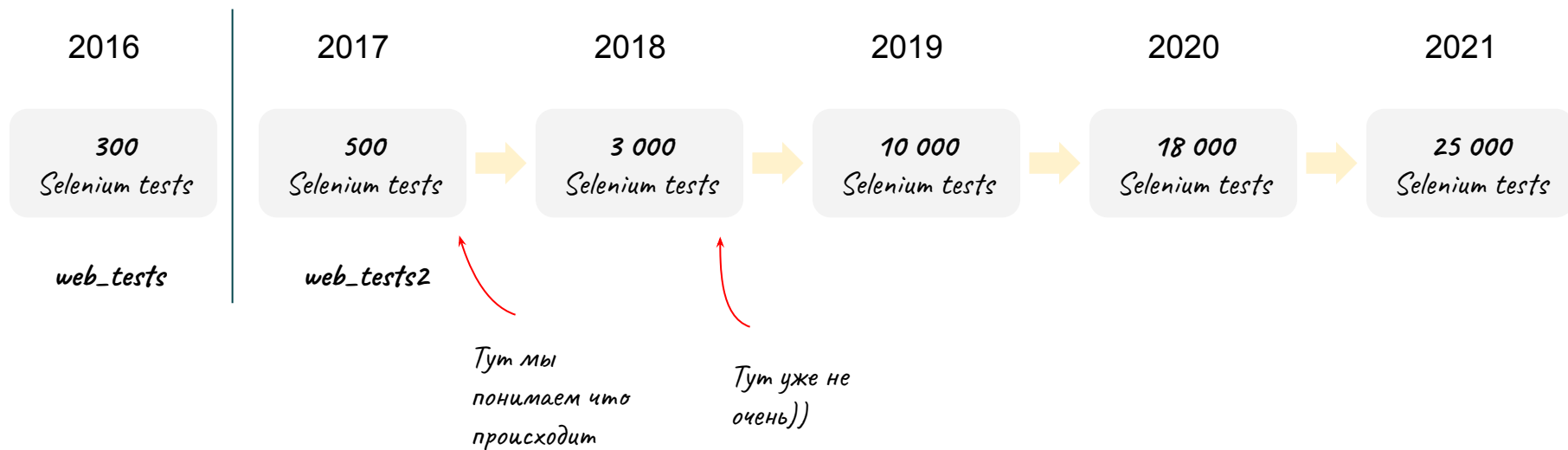
Create new task

# 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

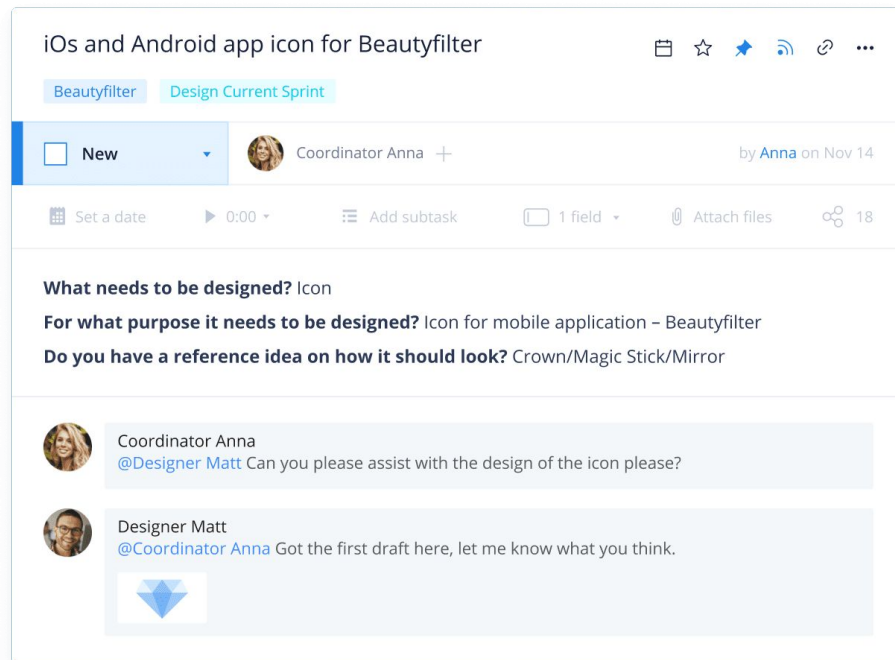
```
@RunWith(RetryRunner.class)
@GuiceModules(WebTestsModule.class)
@Category({AllTests.class, WsRegress.class,
           Tasks.class, FirefoxRegress.class})
@Feature(TASKS)
public class CheckSubFoldersTest {
```

## 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?



The screenshot shows a Jira issue page for the task "iOS and Android app icon for Beautyfilter". The issue is assigned to "Coordinator Anna" and was created on "Nov 14". The issue description includes the following text:

**What needs to be designed?** Icon  
**For what purpose it needs to be designed?** Icon for mobile application – Beautyfilter  
**Do you have a reference idea on how it should look?** Crown/Magic Stick/Mirror

The comment history shows:

- Coordinator Anna** (@Designer Matt) asked: "Can you please assist with the design of the icon please?"
- Designer Matt** (@Coordinator Anna) responded: "Got the first draft here, let me know what you think." and attached a file named "draft.png".

# Разметка тестов



# Как вообще размечают тесты?

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

```
test
@Tag("Task view")
public class StarButtonHeaderTest {
    @Test
    public void checkTaskIsStared() {
    }
    @Test
    @Tag("Screenshot")
    public void checkUiElement() {
    }
}
```

# Аннотации Allure report


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







# На примере Task view

*Epic: Task view*


## iOs and Android app icon for Beautyfilter


Beautyfilter Design Current Sprint


New  Coordinator Anna [+](#) by [Anna](#) on Nov 14

 Set a date  0:00  Add subtask  1 field  Attach files  18

**What needs to be designed?** Icon  
**For what purpose it needs to be designed?** Icon for mobile application – Beautyfilter  
**Do you have a reference idea on how it should look?** Crown/Magic Stick/Mirror

 Coordinator Anna  
[@Designer Matt](#) Can you please assist with the design of the icon please?

 Designer Matt  
[@Coordinator Anna](#) Got the first draft here, let me know what you think.





# На примере Task view

*Feature: Header*

iOs and Android app icon for Beautyfilter 📅 ☆ ★ 📶 🔗 ⋮

Beautyfilter Design Current Sprint


New Coordinator Anna + by Anna on Nov 14


📅 Set a date ▶ 0:00 ⋮ Add subtask 📄 1 field 📎 Attach files 🗨️ 18


**What needs to be designed?** Icon

**For what purpose it needs to be designed?** Icon for mobile application – Beautyfilter

**Do you have a reference idea on how it should look?** Crown/Magic Stick/Mirror

 Coordinator Anna  
[@Designer Matt](#) Can you please assist with the design of the icon please?

 Designer Matt  
[@Coordinator Anna](#) Got the first draft here, let me know what you think.



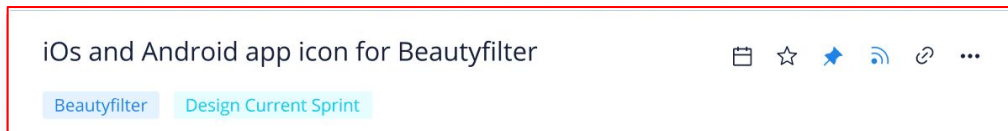
*Feature: Status bar*

*Feature: Live editor*

*Feature: Stream*

# На примере Task view

*Feature: Header*



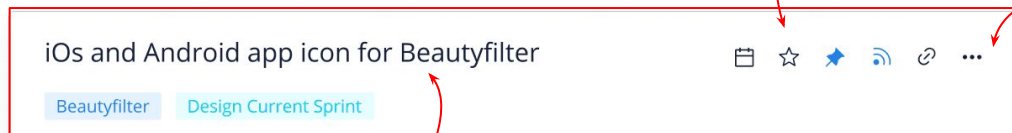
*Feature: Status bar*

*Feature: Live editor*

*Feature: Stream*

# На примере Task view

*Feature: Header*



*Story: Star button*

*Story: Dot menu*

*Story: Task title*

# 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



**Allure**

- Overview
- Categories
- Suites
- Graphs

## Behaviors

order	name	duration	status	Filter by status:
0	0	1	0	0
1	Task view		1	
2	Header		1	
3	Star button		1	
4	#1 Func: Check task is stored via button	14s 870ms		

*Epic: Task view*

*Feature: Header*

*Story: Star button*

# Пример на большом проекте

▼ <u>Task list</u>	471
> Delete	56
> Dnd	122
> Effort	78
> folder tag	45
> Selection	102
> Task creation	68
▼ <u>Task view</u>	10 127
> Custom field	10
> Dependency dialog	2
▼ <u>Header</u>	55
> Blueprint header	3
> Close button	3
> Folder tag	5
> Following button	8



# 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*

## task\_hot\_contrary

Personal +

New + Add ... #461377529  
by Tester-1200155298--38307382--1589779407 T. at 09:30

Set date Approvals 0:00 Add subtask Attach files 0 1

Click to add the description

---

Today

**TT** Tester-1200155298--38307382--1589779407 Testero-  
1200155298--38307382--1589779407 09:30  
Included task into Personal

**TT** Add a comment... @ 😊 Aa

*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*

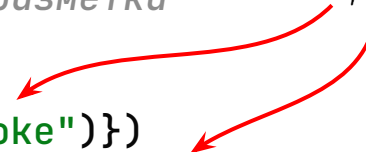
# Техническая разметка

*// 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



# Техническая разметка

*// Можно создавать свои аннотации для Тех 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

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

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



# Пример технической разметки

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

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

}
```

Бизнес  
разметка

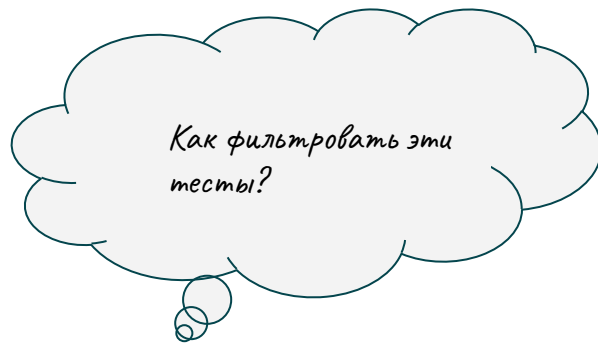
```
{ @Epic("Task view")  
  @Feature("Header")  
  @Story("Star button")  
  public class StarButtonHeaderTest {
```

Техническая  
разметка

```
    @Test  
    @Service(TASK_SERVICE)  
    public void checkTaskIsStaredViaButton() {  
        // Проверка UI для BE task сервиса  
    }  
  
    @Test  
    @Type("Smoke")  
    public void checkSelectedStarButtonScreenshot() {  
        // Тест на критический функционал  
    }  
  
}
```

# 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

*Разметка для запуска*



```
// 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

Стандартный раннер для  
запуска по категориям

Тестовые селекторы

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

  }
```

# Фильтрация тестов JUnit 5

*Разметка для запуска*



```
// 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

Раннер



Тестовые  
селекторы

```
// JUnit 5 - "JUnit 4 like" Suite для запуска
@RunWith(JUnitPlatform.class)
@SelectClasses(StarButtonHeaderTest.class)
@SelectPackages("com.example.tests")
@IncludeTags("Task view")
@ExcludeTags("Screenshot")
public class TaskViewSuiteJUnit5 {
}
```

# 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
    }

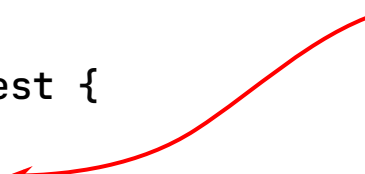
    @Tag("Screenshot")
    @Test
    public void checkSelectedStar() {
        // Не такая важная проверка
    }
}
```

# Фильтрация тестов TestNG

```
// 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
    }
}
```

*Группировка для тестов*



# 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

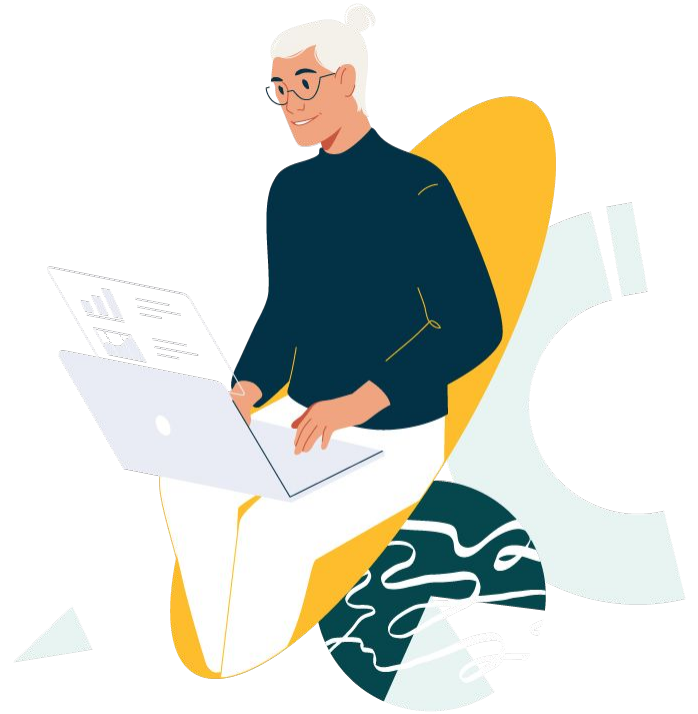
```
@RunWith(RetryRunner.class)
@GuiceModules(WebTestsModule.class)
@Category({AllTests.class, WsRegress.class,
          Tasks.class, DragAndDrop.class,
          Smoke.class, ScreenshotTest.class,
          FirefoxRegress.class, Folder.class})
@Feature(TASKS)
public class CheckSubFoldersTest {
```

# 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

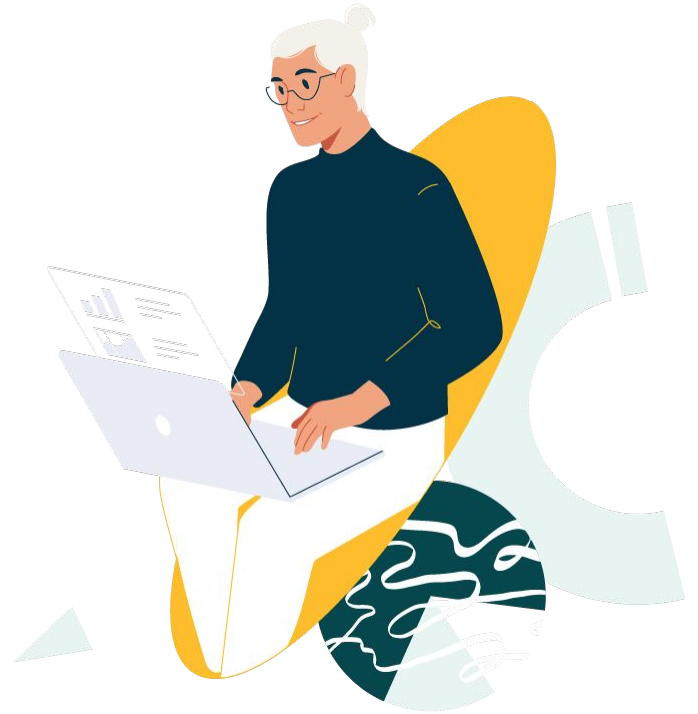
- Тестовый фильтр
- Модуль, вычисляющий выражения тестовых фильтров
- Адаптеры
  - JUnit 4
  - JUnit 5
  - TestNG





# 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}
```

```
@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

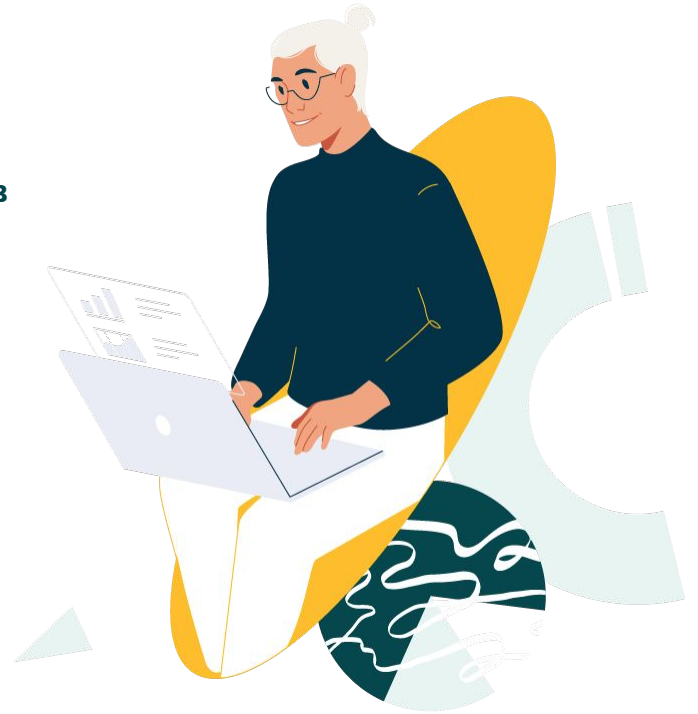
*// Реальные примеры*

```
{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

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

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

```
@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*  
*feature:Header*  
*story:Star button*  
*type:Smoke*  
*service:task service*

*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*

{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*

**{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

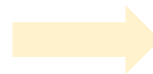
***epic:Task view  
feature:Header  
story:Star button  
type:A11Y***

**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

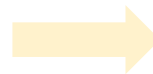
*epic:Task view  
feature:Header  
story:Star button  
type:A11Y*

true&{type:Smoke}

# Вычисление тестовых фильтров

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

true&true



true

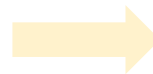
*epic:Task view  
feature:Header  
story:Star button  
type:A11Y*

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*

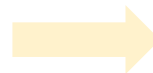
true&false



# 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

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



true

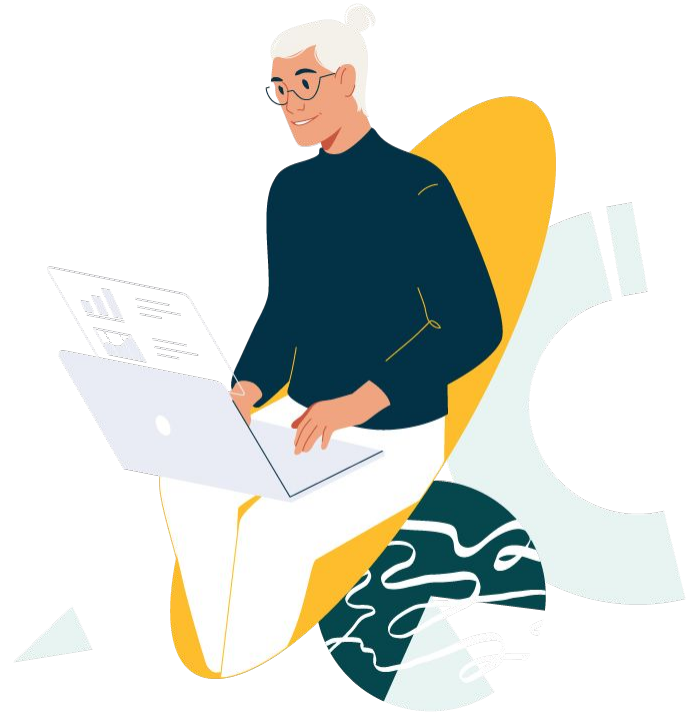
*epic:Task view*  
*feature:Header*  
*story:Star button*  
*type:A11Y*



false

# 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


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

# Адаптер Junit4

MichaelTamm / **junit-toolbox** Notifications Star 132 Fork 23

<> Code Issues 6 Pull requests 1 Actions Projects Security Insights

master Go to file Code

 **MichaelTamm** [maven-release-plugin] prepare for next developm... on 10 Dec 2020 102

src	WildcardPatternSuite collects test files in alphabeti...	3 months ago
.gitignore	inital commit	9 years ago
AUTHORS.md	appended my name to AUTHORS.md	4 years ago
LICENSE.md	Added LICENSE.md -- fixes #13	4 years ago
README.md	Updated README.md	4 years ago
pom.xml	[maven-release-plugin] prepare for next developme...	3 months ago

**About**  
Useful classes for writing automated tests with JUnit 4  
[Readme](#)  
[Apache-2.0 License](#)

**Releases**  
[18 tags](#)

[junit-toolbox](#)

# Адаптер Junit4

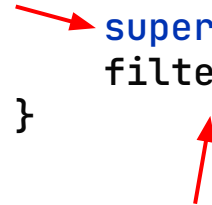
*// Основа: WildcardPatternSuite*

```
@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());  
    }  
}
```

A diagram with two red arrows. One arrow points from the left towards the word 'super' in the constructor call 'super(klass, builder);'. The other arrow points from the bottom towards the word 'new' in the filter call '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](https://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

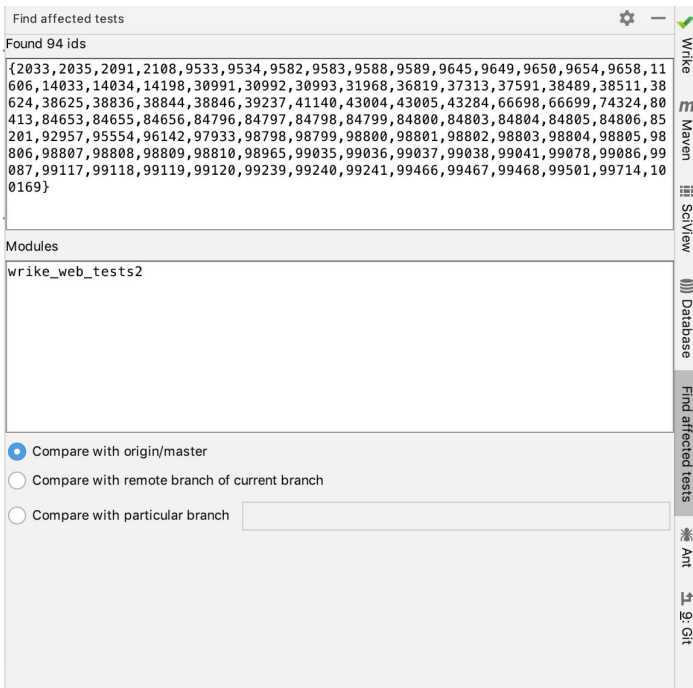
The screenshot shows a test runner interface with a list of test categories. Each category has a name, a count of failed tests, a warning icon, and two buttons: 'Rerun' and 'Run'. A red arrow points to the '3 / 478' status for 'Feature tests (new)', and another red arrow points to the '3' status for 'WS Regression'.

Test Category	Failed Tests	Total Tests	Status	Buttons
Feature tests (new)	3	478	Warning	Rerun, Run
WS Regression	3	-	Warning	Rerun, Run
Smoke Tests	871	-	Warning	Rerun, Run

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

# Дополнительная фильтрация тестов

// Поиск затронутых тестов рефакторингом



Find affected tests

Found 94 ids

{2033,2035,2091,2108,9533,9534,9582,9583,9588,9589,9645,9649,9650,9654,9658,11606,14033,14034,14198,30991,30992,30993,31968,36819,37313,37591,38489,38511,38624,38625,38836,38844,38846,39237,41140,43004,43005,43284,66698,66699,74324,80413,84653,84655,84656,84796,84797,84798,84799,84800,84803,84804,84805,84806,85201,92957,95554,96142,97933,98798,98799,98800,98801,98802,98803,98804,98805,98806,98807,98808,98809,98810,98965,99035,99036,99037,99038,99041,99078,99086,99087,99117,99118,99119,99120,99239,99240,99241,99466,99467,99468,99501,99714,100169}

Modules

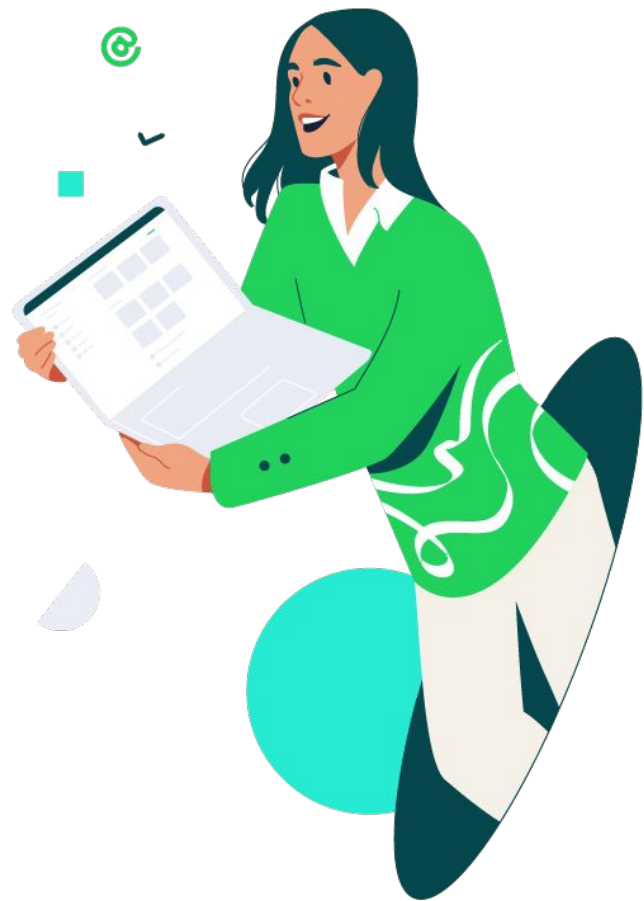
wrike\_web\_tests2

Compare with origin/master  
 Compare with remote branch of current branch  
 Compare with particular branch

Wrike  
Maven  
SciView  
Database  
Find affected tests  
Art  
Git

# 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





**Спасибо за внимание!**

