Нужен рефакторинг? Есть Idea!

Heisenbug Moscow, 2018


Привет, Артем
Меня зовут Артем
и я автоматизатор...

Автоматизация тестирования

Много разных проектов

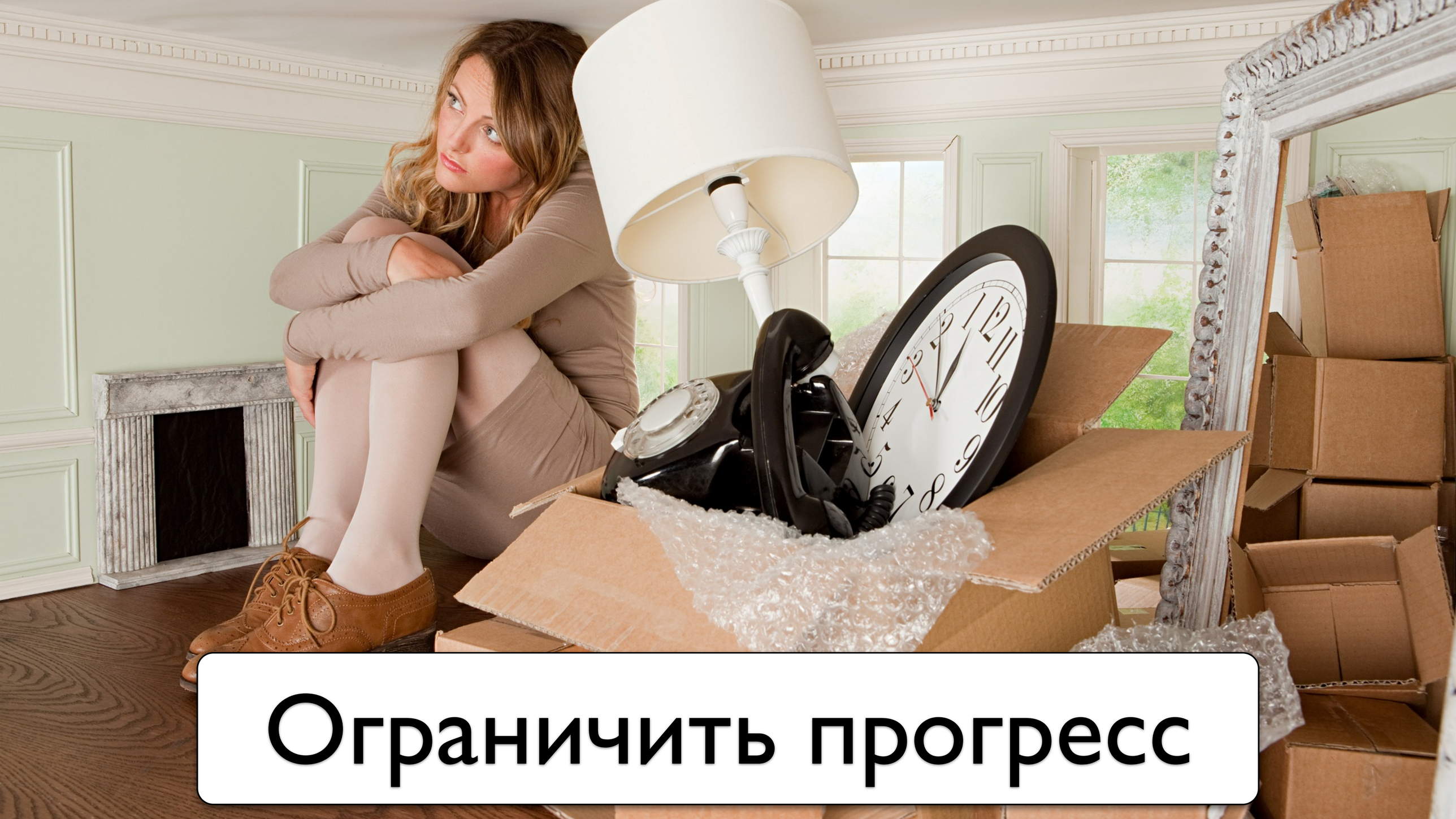

В каждом проекте сюрприз

СУОЖНРІЙ


Монстр

Расставить все по местам

Как обеспечить равномерность?



Ограничить прогресс

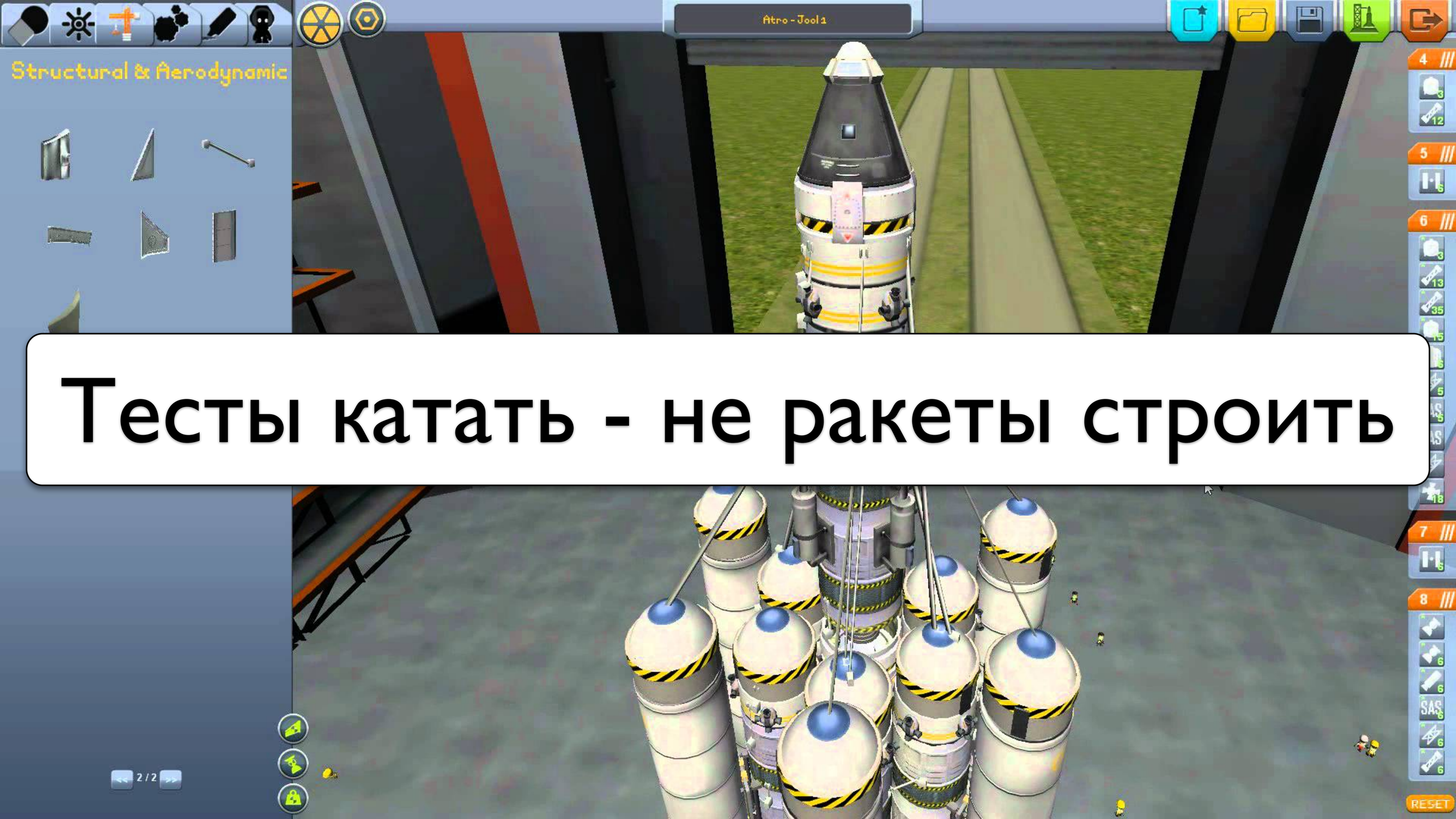

Быстро догонять

Добавить пару рук

Совершенствовать орудие труда

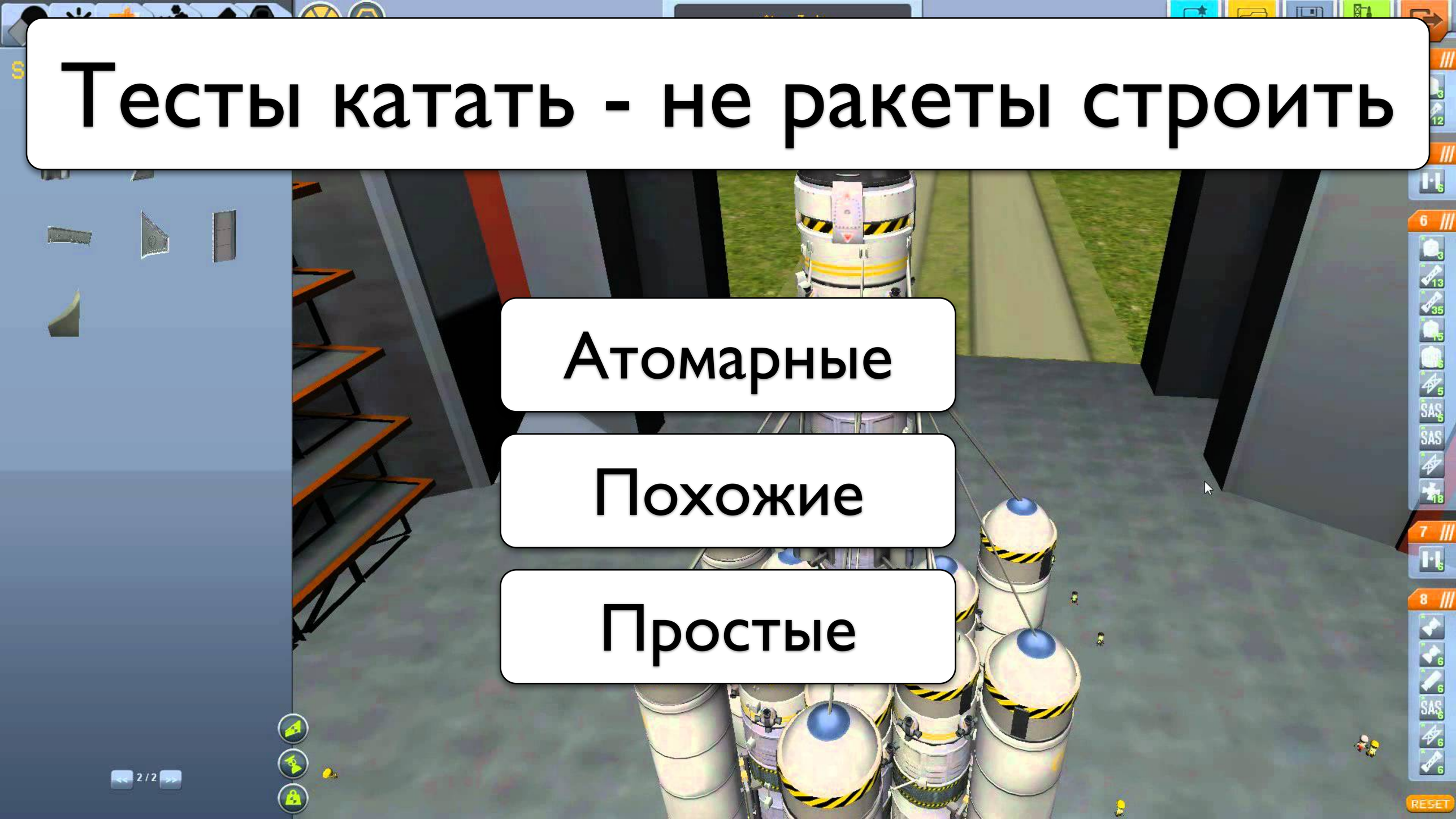

Я видел много инструментов

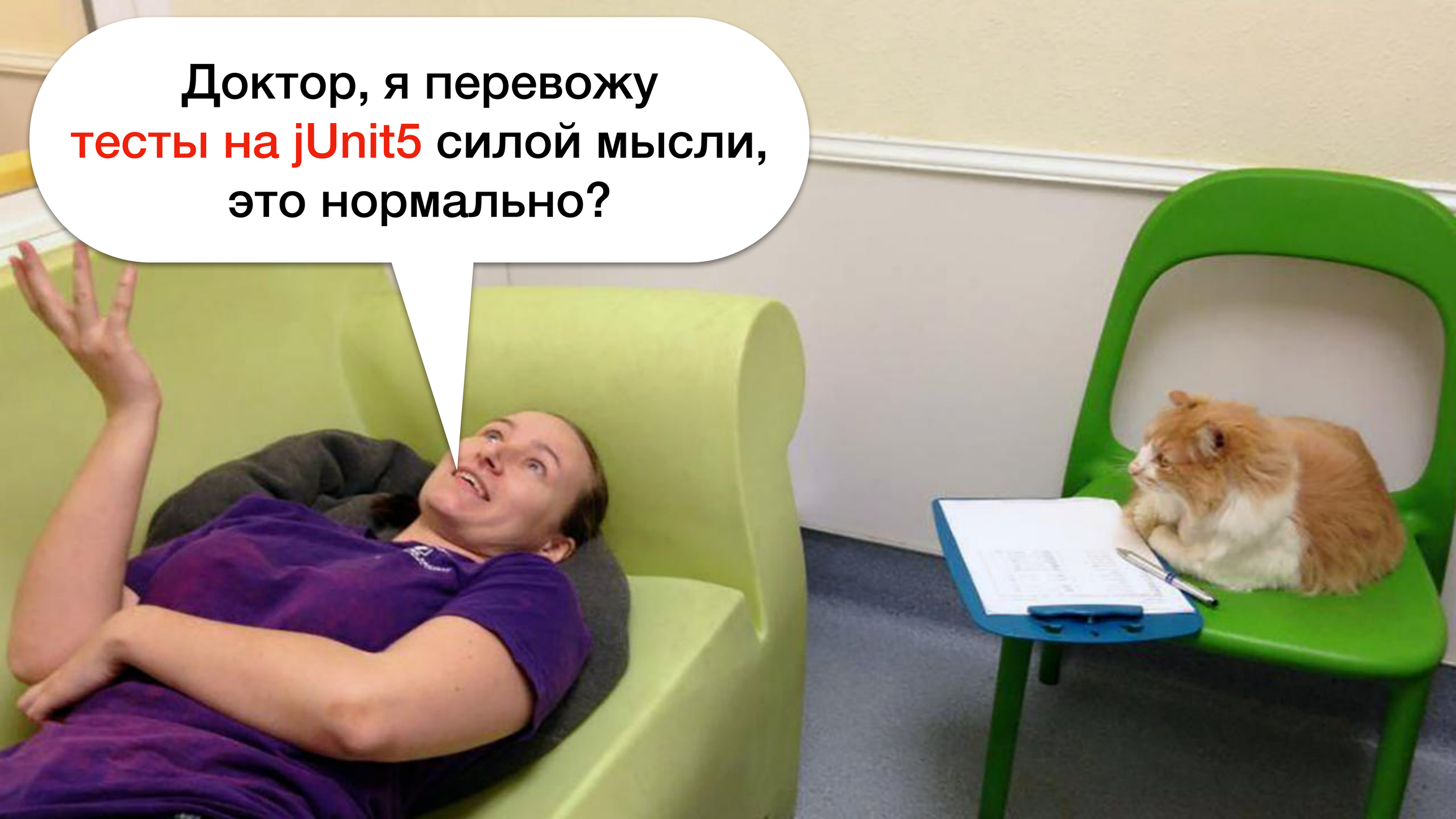
Возможности Idea Plugin

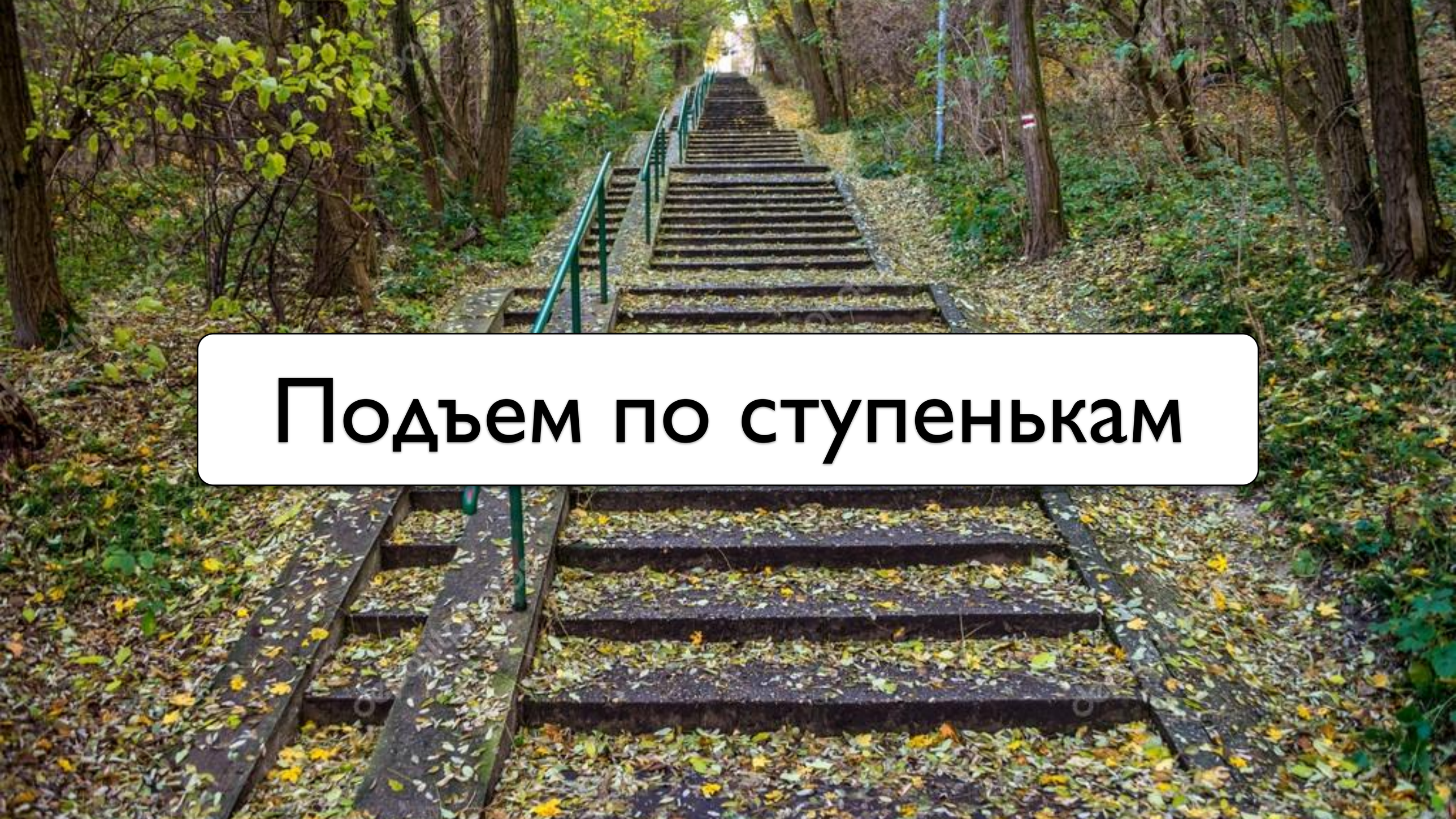


Тесты катать - не ракеты строить

Тесты катать - не ракеты строить

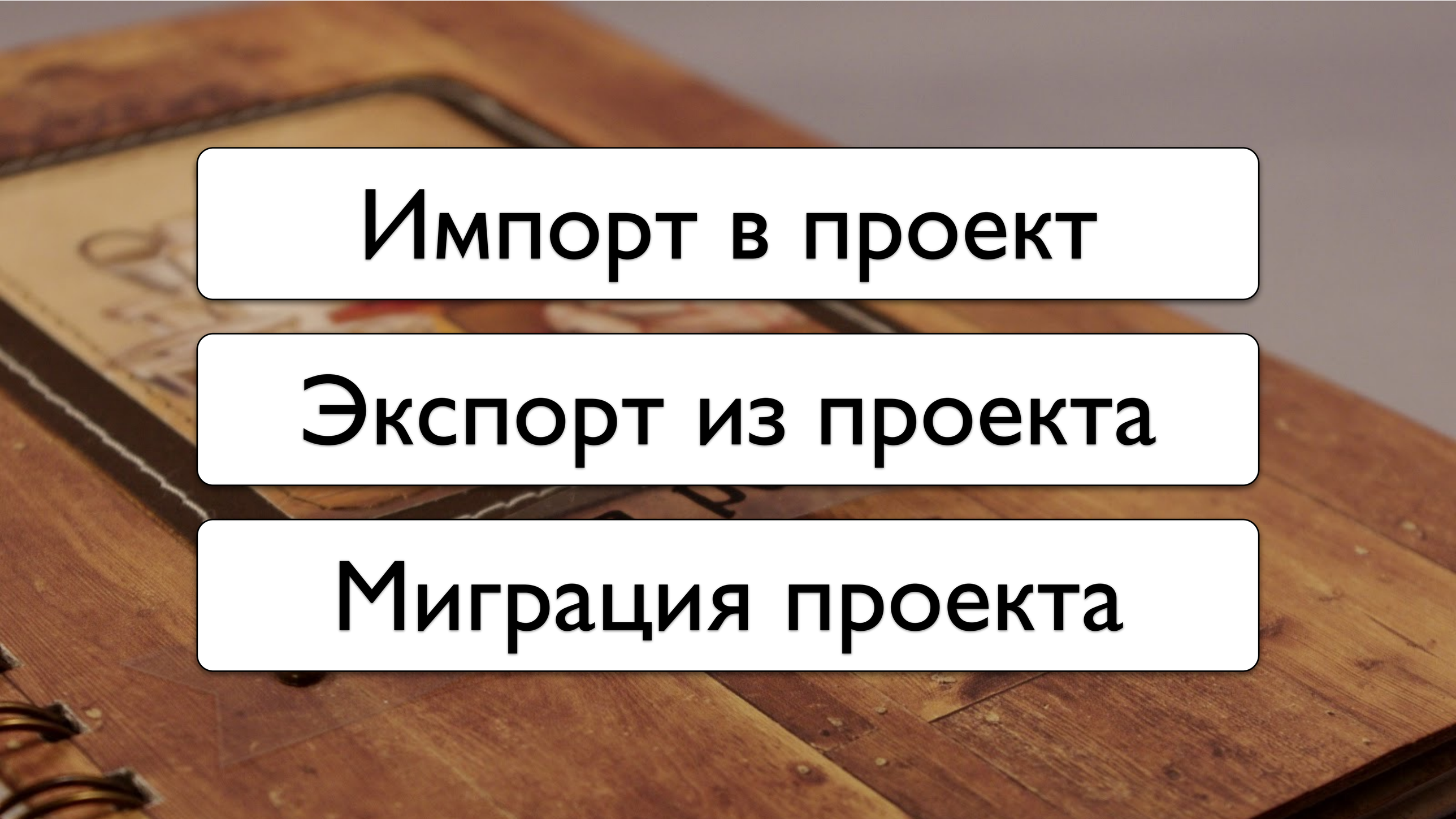


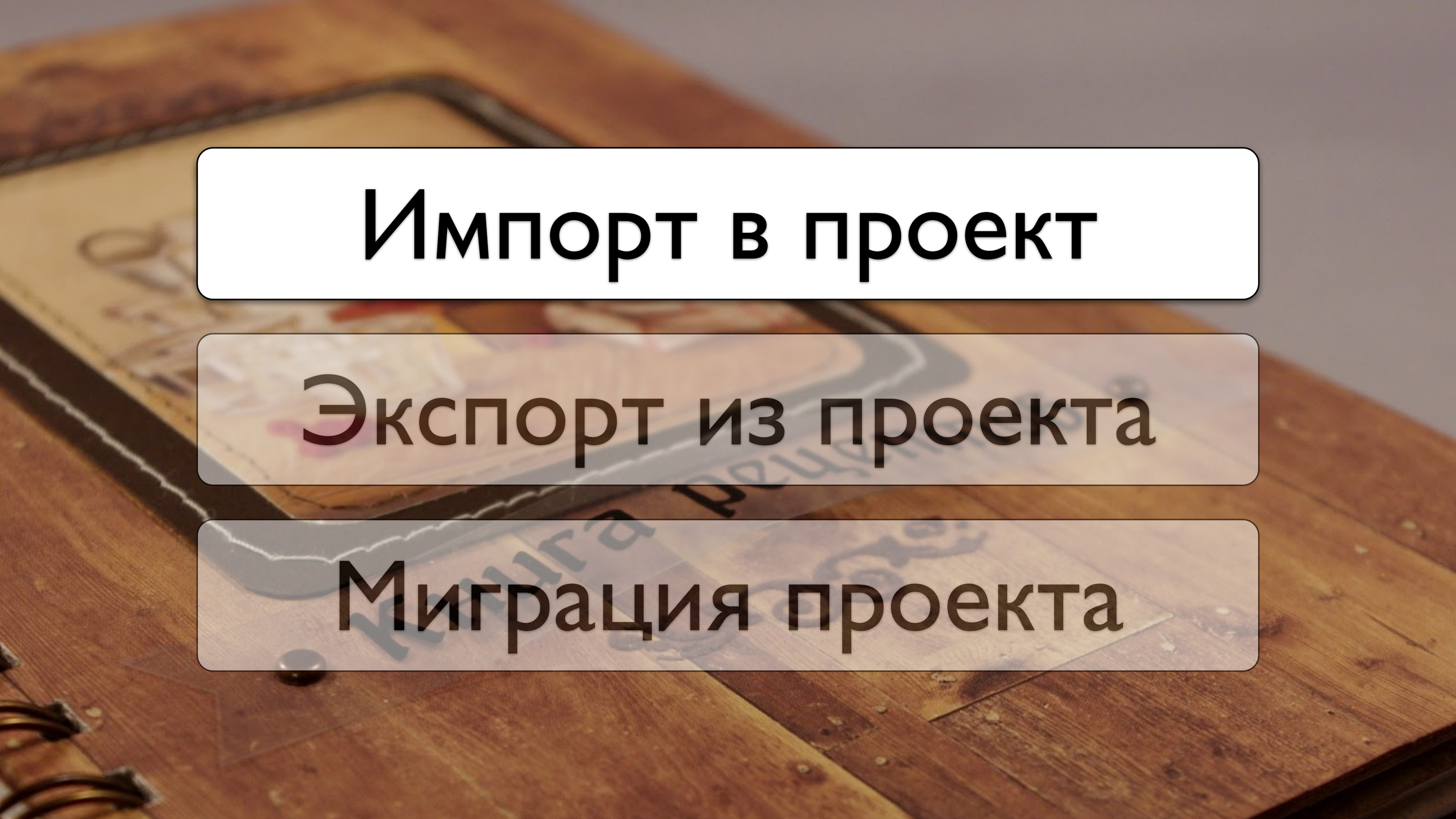

Доктор, я перевожу
тесты на jUnit5 силой мысли,
это нормально?



Подъем по ступенькам

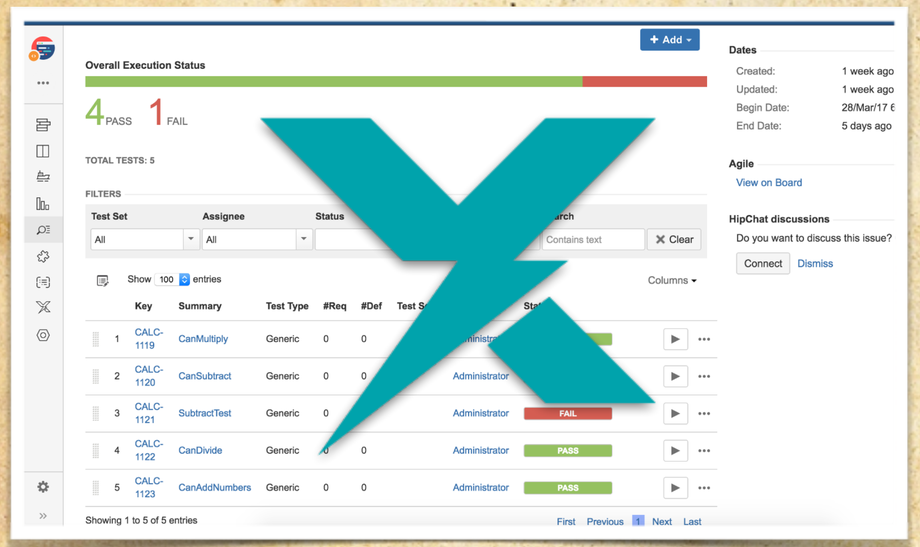

Вперед к приключениям
movepics.ru

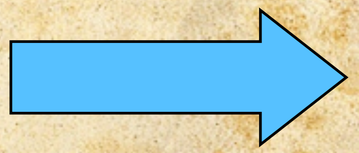

Импорт в проект
Экспорт из проекта
Миграция проекта

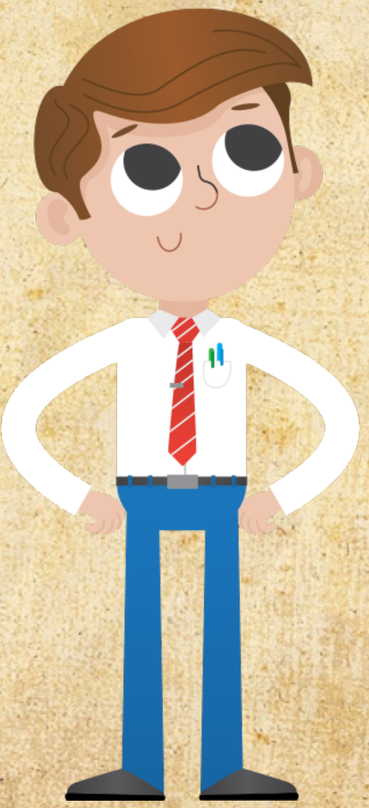

Импорт в проект
Экспорт из проекта
Миграция проекта

Какую проблему решаем?

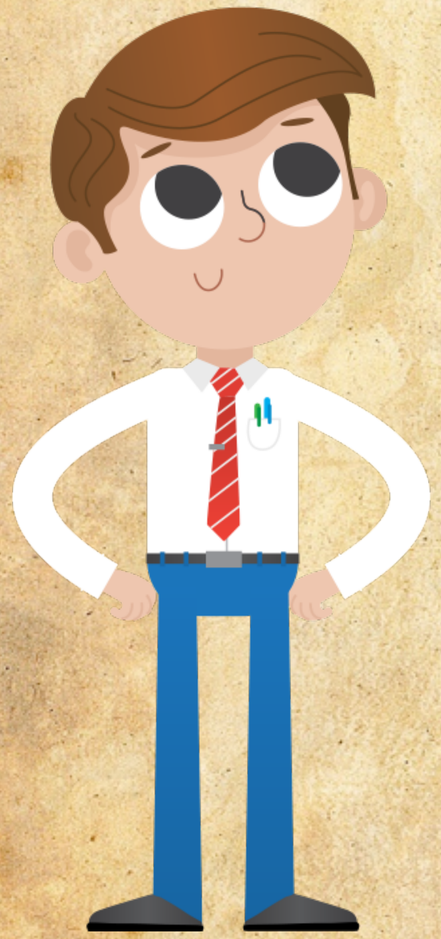
Что автоматизировано?

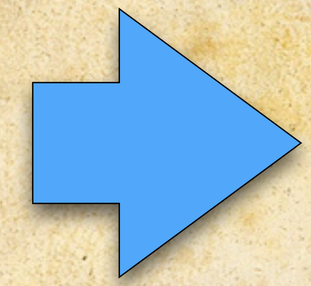


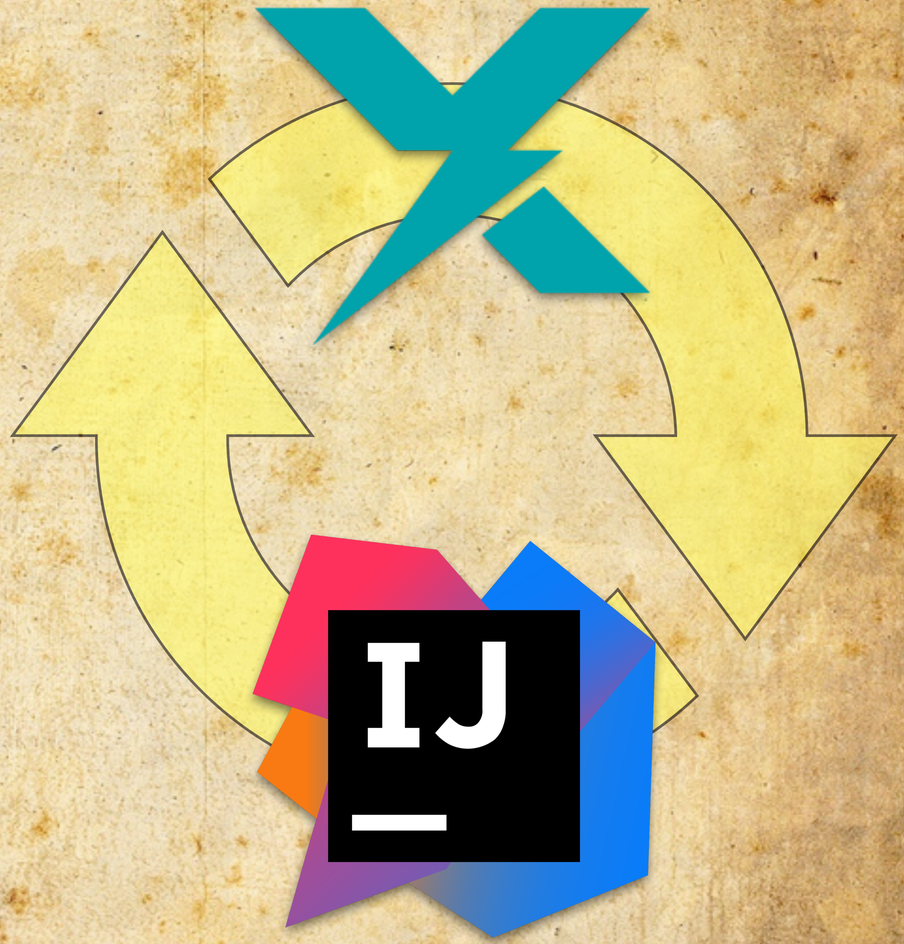




Задача на автоматизацию



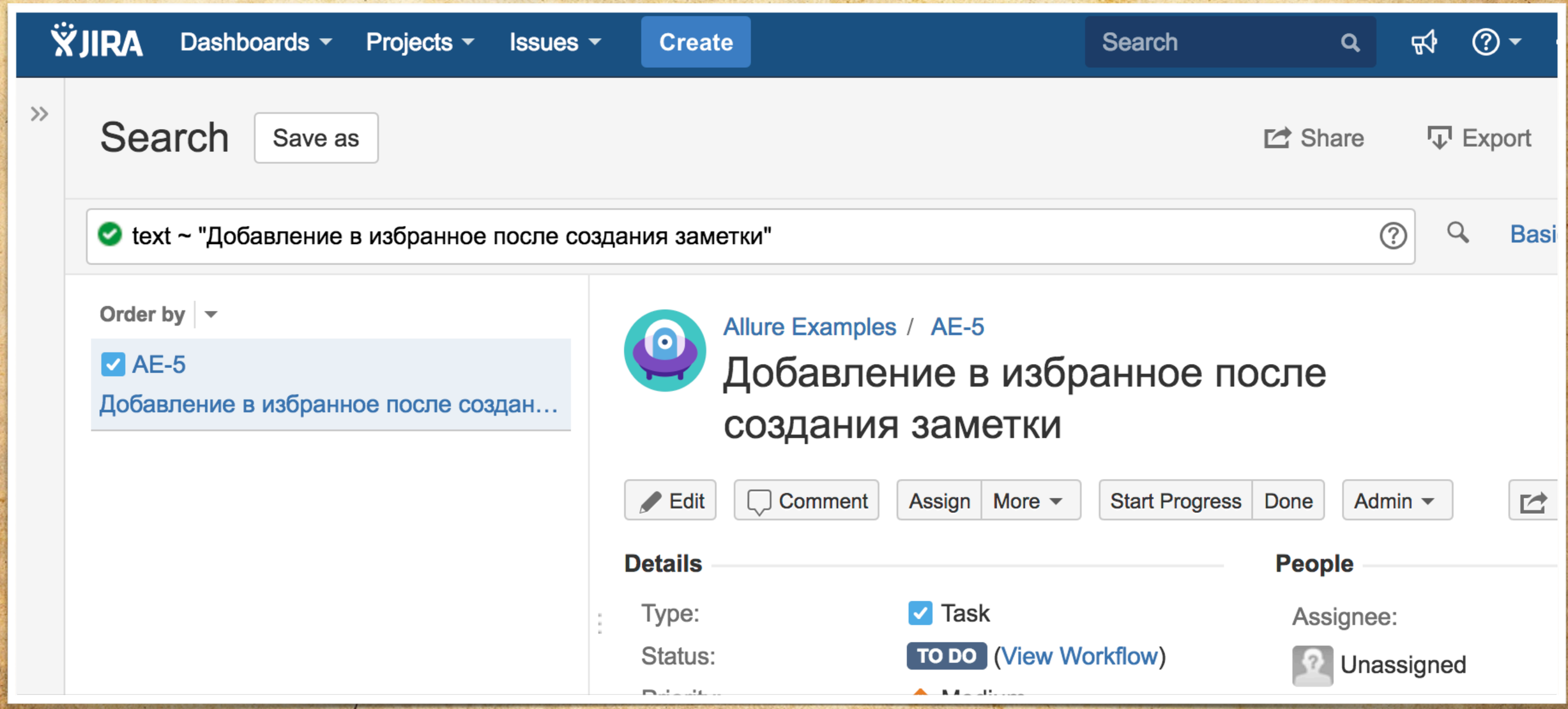




Возьмем название теста

```
public class AddFavoritesAfterNoteTest {

    @Test
    @DisplayName("Добавление в избранное после создания заметки")
    public void shouldAddToFavoriteAfterNodeTest() { ... }

    @Test
    @DisplayName("Удаление из избранного после удаления заметки")
    public void shouldDeleteToFavoriteAfterNodeTest() { ... }

}
```

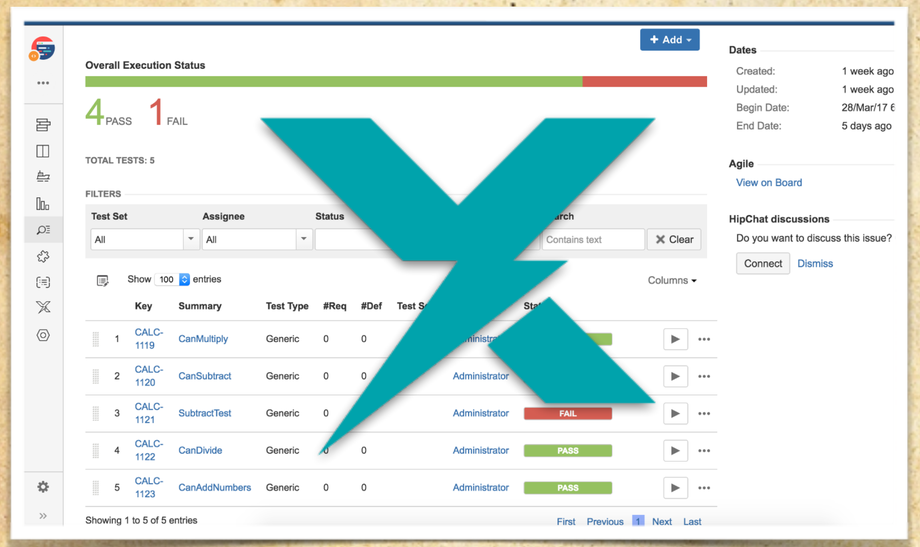
Поиск по тексту в Jira

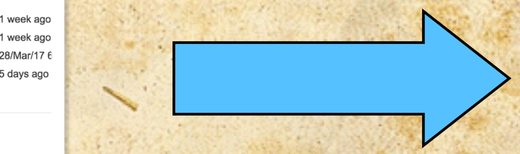


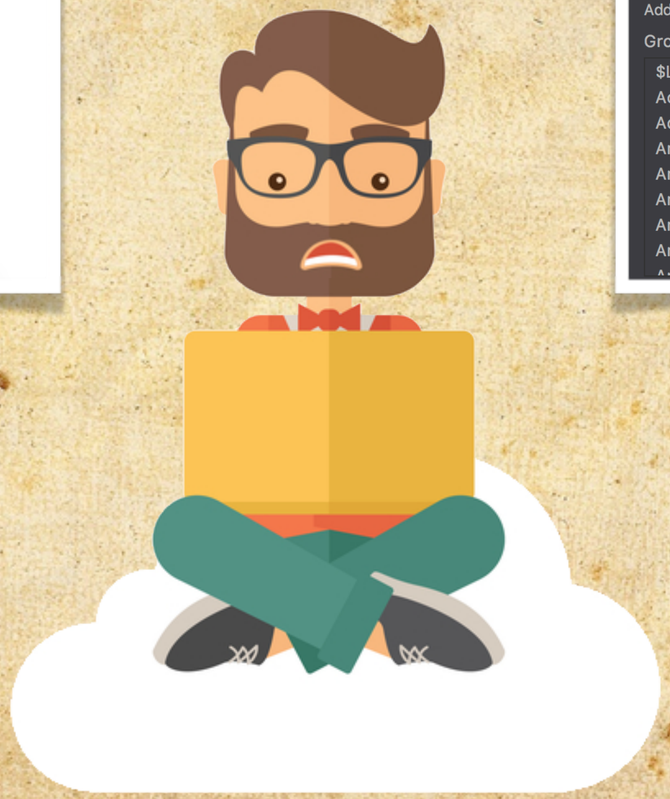
Нужно расставить ссылки

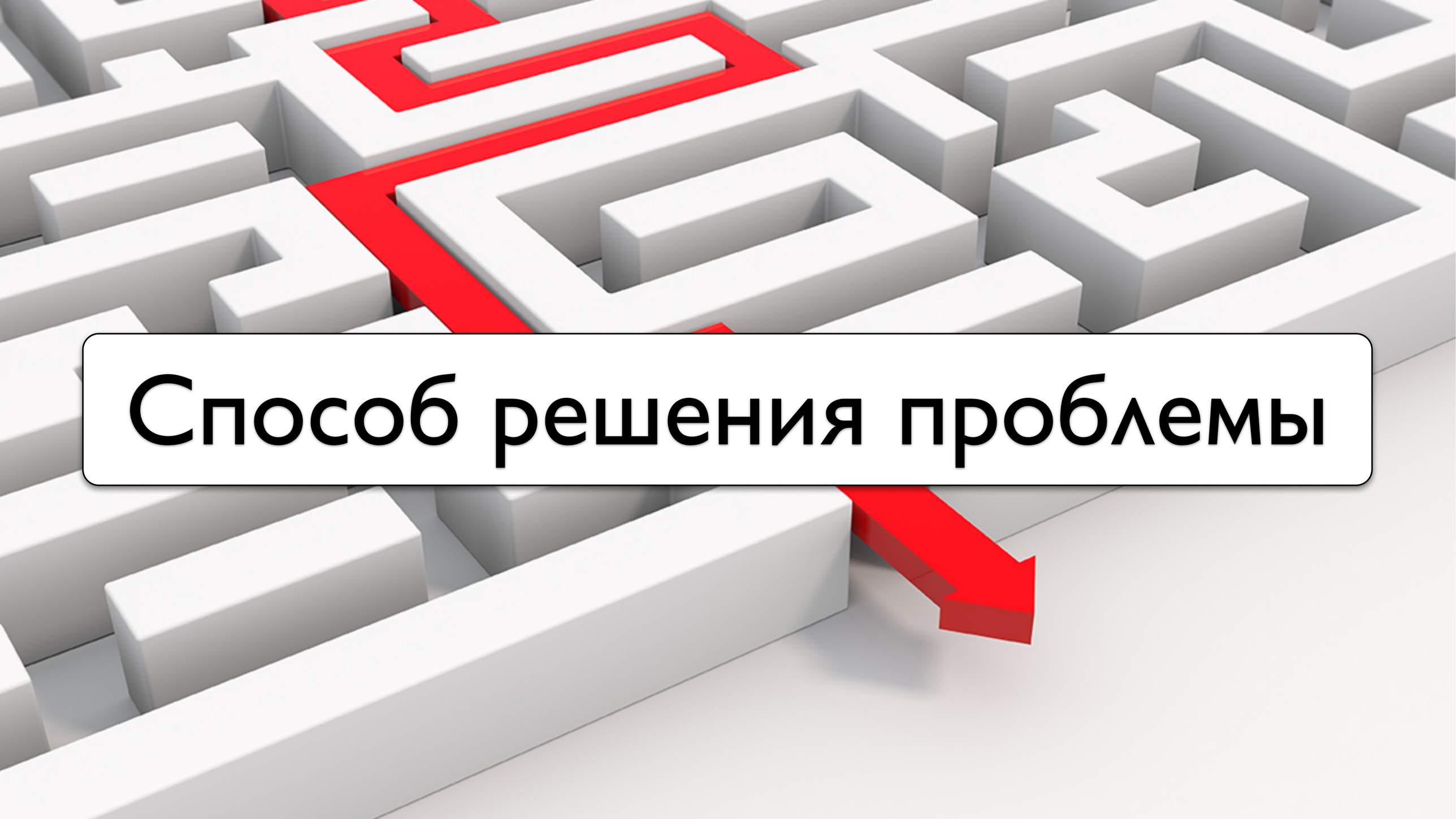
```
public class AddFavoritesAfterNoteTest {

  @Test
  @TmsLink("AE-5")
  @DisplayName("Добавление в избранное после создания заметки")
  public void shouldAddToFavoriteAfterNodeTest() { ... }

  @Test
  @TmsLink("AE-4")
  @DisplayName("Удаление из избранного после удаления заметки")
  public void shouldDeleteToFavoriteAfterNodeTest() { ... }

}
```

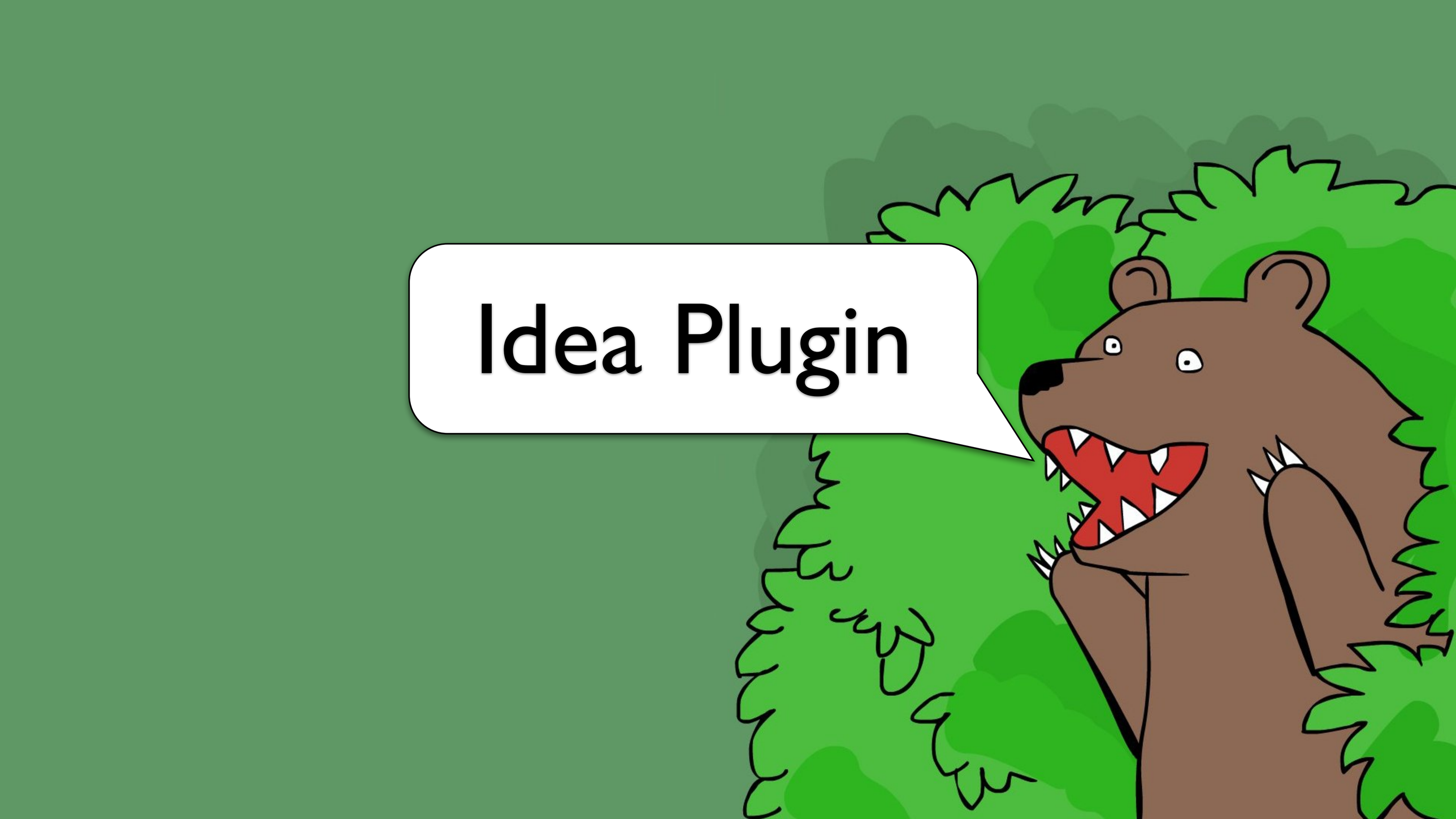
Для всех тестов сложно



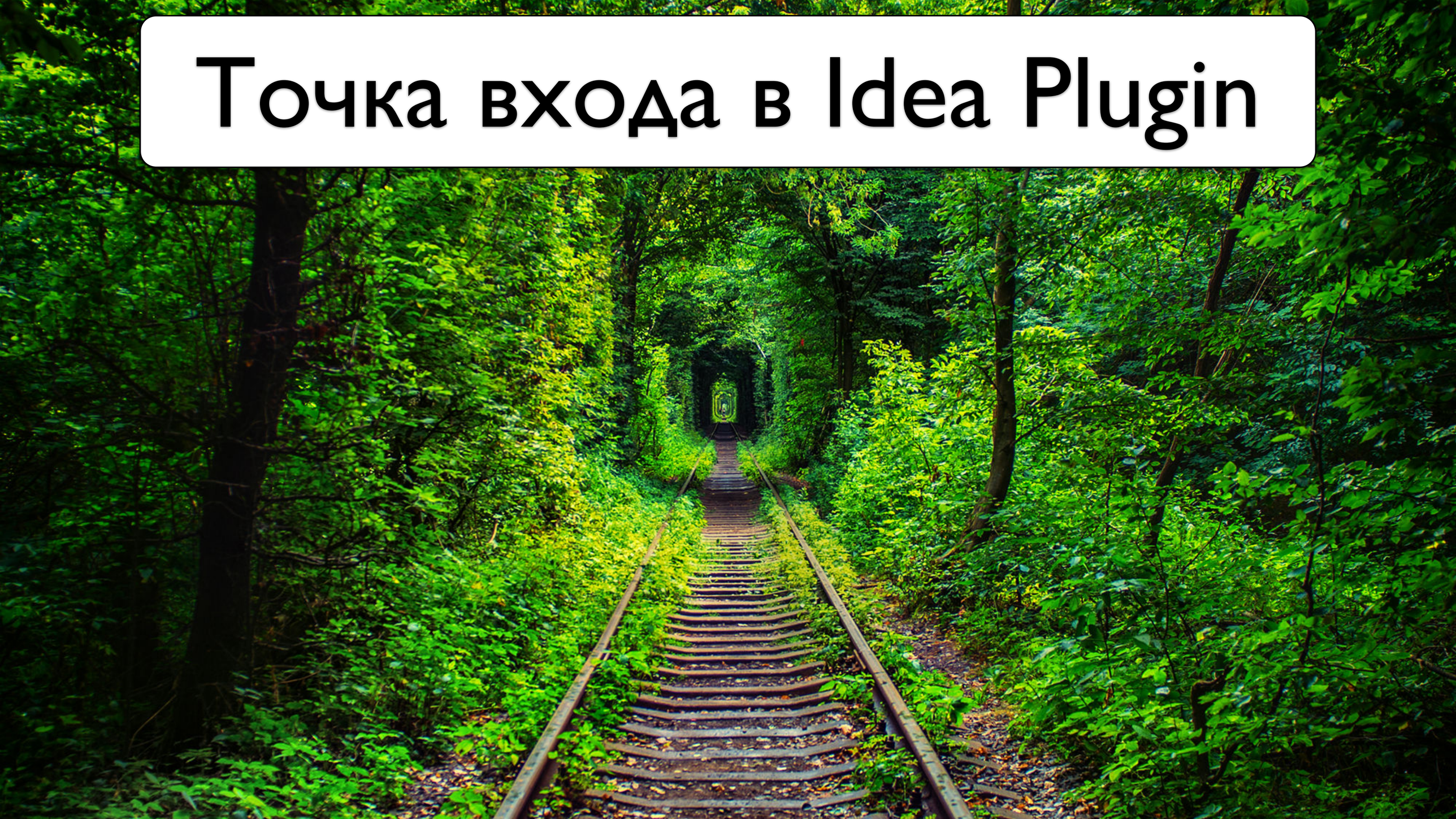





Способ решения проблемы


Idea Plugin

Точка входа в Idea Plugin



Точка входа в Idea Plugin

```
public class AnAction {

   @Override
   public void actionPerformed(AnActionEvent event) {
      //MAGIC here
   }

}
```

Очень похоже на Reflection

PsiClass - работа с классом

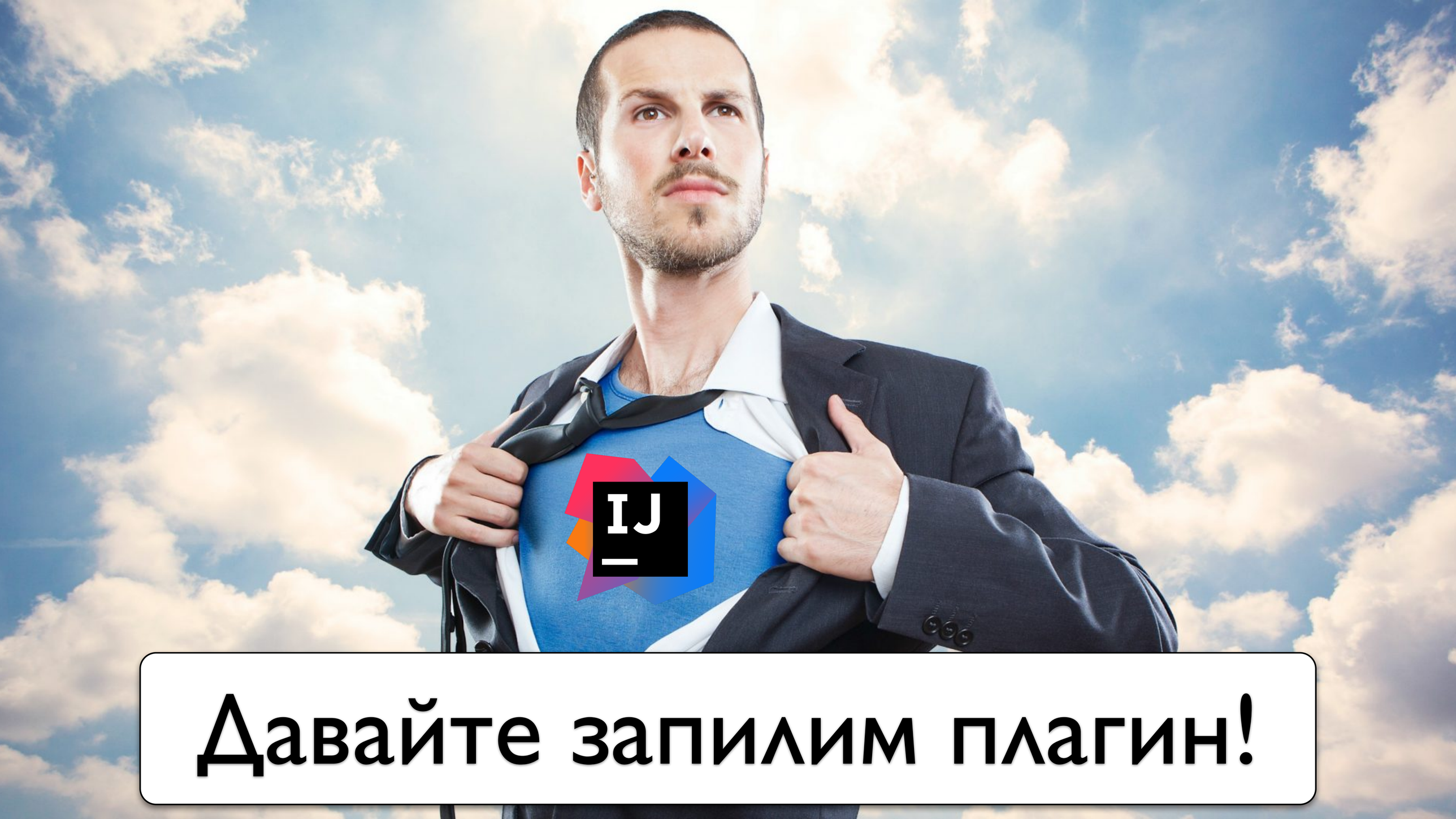
```
public class AddFavoritesAfterNoteTest {

    @Test
    @TmsLink("AE-5")
    @DisplayName("Добавление в избранное после создания заметки")
    public void shouldAddToFavoriteAfterNodeTest() { ... }

    @Test
    @TmsLink("AE-4")
    @DisplayName("Удаление из избранного после удаления заметки")
    public void shouldDeleteToFavoriteAfterNodeTest() { ... }

}
```

PsiMethod - работа с методами

```
public class AddFavoritesAfterNoteTest {

    @Test
    @TmsLink("AE-5")
    @DisplayName("Добавление в избранное после создания заметки")
    public void shouldAddToFavoriteAfterNodeTest() { ... }

    @Test
    @TmsLink("AE-4")
    @DisplayName("Удаление из избранного после удаления заметки")
    public void shouldDeleteToFavoriteAfterNodeTest() { ... }

}
```

PsiAnnotation - аннотации

```
public class AddFavoritesAfterNoteTest {

    @Test
    @TmsLink("AE-5")
    @DisplayName("Добавление в избранное после создания заметки")
    public void shouldAddToFavoriteAfterNodeTest() { ... }

    @Test
    @TmsLink("AE-4")
    @DisplayName("Удаление из избранного после удаления заметки")
    public void shouldDeleteToFavoriteAfterNodeTest() { ... }

}
```



Давайте запилим плагин!

Создаем JiraKeyImportAction

```
public class JiraKeyImportAction extends AnAction {

  @Override
  public void actionPerformed(AnActionEvent event) {
    // Magic here
  }

}
```

AnActionEvent - это контекст

```
public class JiraKeyImportAction extends AnAction {

  @Override
  public void actionPerformed(AnActionEvent event) {

    PsiElement element = event.getData(PSI_ELEMENT)

  }

}
```

Работаем только с классами

```
public class JiraKeyImportAction extends AnAction {

  @Override
  public void actionPerformed(AnActionEvent event) {
    PsiElement element = event.getData(PSI_ELEMENT)

    if (element instance of PsiClass) {
      addTmsLinkToClassMethods((PsiClass)element)
    }

  }
}
```

Выбрали класс для апдейта

```
public class AddFavoritesAfterNoteTest {

  @Test
  @DisplayName("Добавление в избранное после создания заметки")
  public void shouldAddToFavoriteAfterNodeTest() { ... }

  @Test
  @DisplayName("Удаление из избранного после удаления заметки")
  public void shouldDeleteToFavoriteAfterNodeTest() { ... }

  private void utilityMethod() { ... }

}
```

Получаем все методы класса

```
public class JiraKeyImportAction extends AnAction {

  void addTmsLinkToClassMethods(PsiClass testClass) {

    Arrays.stream(testClass.getMethods())
      .filter(m -> m.hasAnnotation("org..Test"))
      .filter(m -> m.hasAnnotation("org..DisplayName"))
      .forEach(this::addTmsLinkToMethod)

  }
}
```

Фильтруем тестовые методы

```
public class JiraKeyImportAction extends AnAction {

  void addTmsLinkToClassMethods(PsiClass testClass) {

    Arrays.stream(testClass.getMethods())
      .filter(m -> m.hasAnnotation("org..Test"))
      .filter(m -> m.hasAnnotation("org..DisplayName"))
      .forEach(this::addTmsLinkToMethod)

  }
}
```

Выбрали методы для апдейта

```
public class AddFavoritesAfterNoteTest {

  @Test
  @DisplayName("Добавление в избранное после создания заметки")
  public void shouldAddToFavoriteAfterNodeTest() { ... }

  @Test
  @DisplayName("Удаление из избранного после удаления заметки")
  public void shouldDeleteToFavoriteAfterNodeTest() { ... }

  private void utilityMethod() { ... }

}
```

Берем аннотацию DisplayName

```
public class JiraKeyImportAction extends AnAction {

void addTmsLinkToMethod(PsiMethod method) {

    PsiAnnotatation nameAnnotation =
                method.getAnnotation("org..DisplayName");
    String text = name
      .findDeclaredAttributeValue("value").getText()
    String key = jiraClient.findByText(text);
    addTmsLinkToMethod(testMethod, key);
  }
}
```

Нашли аннотацию с именем

```
public class AddFavoritesAfterNoteTest {

  @Test
  @DisplayName("Добавление в избранное после создания заметки")
  public void shouldAddToFavoriteAfterNodeTest() { ... }

  @Test
  @DisplayName("Удаление из избранного после удаления заметки")
  public void shouldDeleteToFavoriteAfterNodeTest() { ... }

  private void utilityMethod() { ... }

}
```

Берем текст DisplayName

```
public class JiraKeyImportAction extends AnAction {

  void addTmsLinkToMethod(PsiMethod testMethod) {

    PsiAnnotatation name =
      method.getAnnotation("org..DisplayName");
    String text = name
      .findDeclaredAttributeValue("value").getText()
    String key = jiraClient.findByText(text);
    addTmsLinkToMethod(testMethod, key);
  }
}
```

Тест для поиска в Jira

```
public class AddFavoritesAfterNoteTest {

  @Test
  @DisplayName("Добавление в избранное после создания заметки")
  public void shouldAddToFavoriteAfterNodeTest() { ... }

  @Test
  @DisplayName("Удаление из избранного после удаления заметки")
  public void shouldDeleteToFavoriteAfterNodeTest() { ... }

  private void utilityMethod() { ... }

}
```

Ищем Issue по тексту в Jira

```
public class JiraKeyImportAction extends AnAction {

  void addTmsLinkToMethod(PsiMethod testMethod) {

    PsiAnnotatation name =
      method.getAnnotation("org..DisplayName");
    String text = name
      .findDeclaredAttributeValue("value").getText()
    String key = jiraClient.findByText(text);
    addTmsLinkToMethod(testMethod, key);
  }
}
```

Посмотрим на Jira API

Search for issues using JQL (GET)

GET /rest/api/3/search

Searches for issues using JQL↗ . **Permissions↗ required:** Permission to access Jira.

If the JQL query expression is too large to be encoded as a query parameter, use the POST version of this resource.

App scope required: `READ`

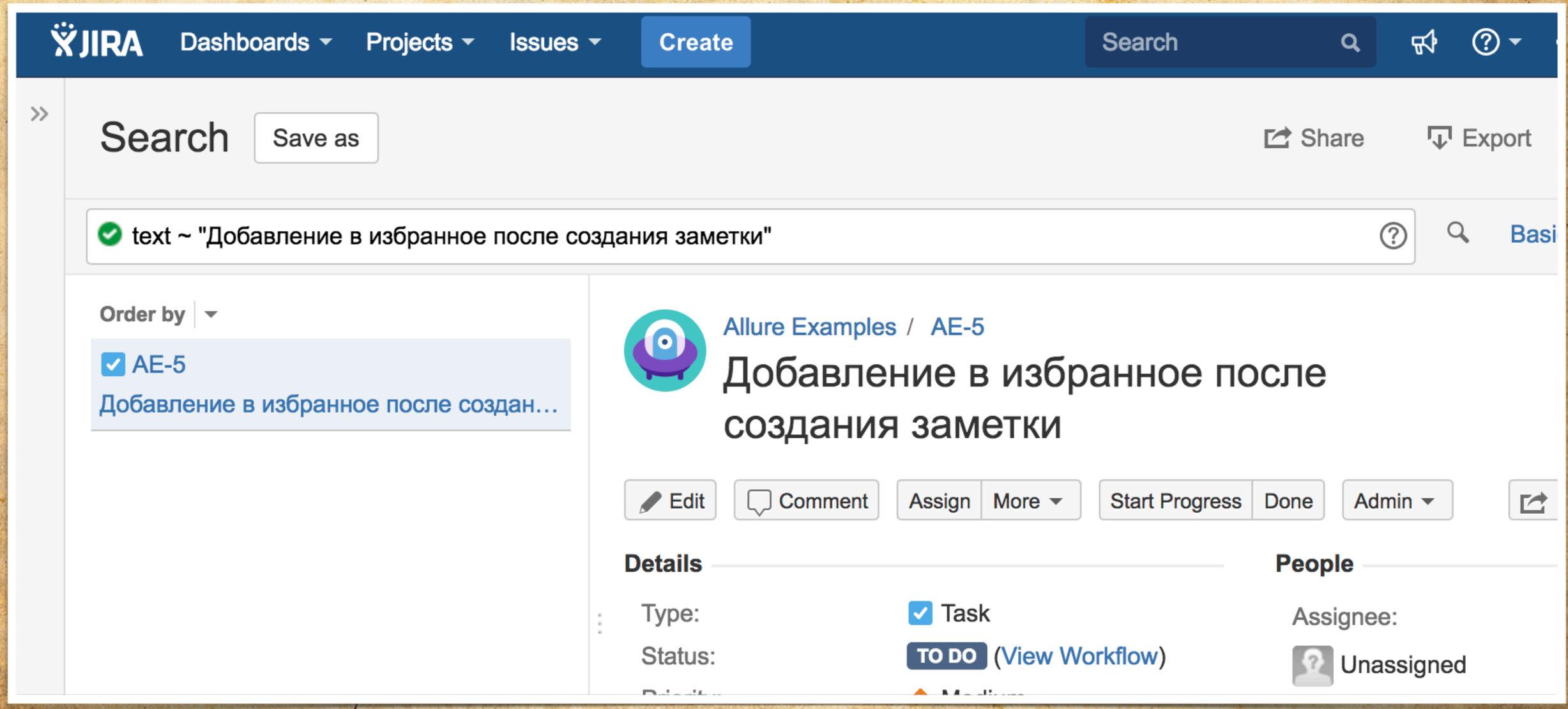
OAuth scopes required: `read:jira-work`

Request

QUERY PARAMETERS

jql
string

Поиск по тексту в Jira



Добавляем аннотацию в код

```
public class JiraKeyImportAction extends AnAction {

void addTmsLinkToMethod(PsiMethod testMethod) {

    PsiAnnotatation name =
      method.getAnnotation("org..DisplayName");
    String text = name
      .findDeclaredAttributeValue("value").getText()
    String key = jiraClient.findByText(text);
    addTmsLinkToMethod(testMethod, key);
  }
}
```

Формируем текст аннотации

```
public class JiraKeyImportAction extends AnAction {

  void addTmsLinkToMethod(PsiMethod method,String key){

    String tmsLinksAnnotationText =
      "@io..TmsLink(\"" + key + "\")";
    PsiAnnotation tmsLink =
      method.addAnnotation(tmsLinkAnnotationText);
    JavaCodeStyleManager.getInstance(project)
        .shortenClassReferences(tmsLink);
  }
}
```

Добавляем аннотацию

```
public class JiraKeyImportAction extends AnAction {

  void addTmsLinkToMethod(PsiMethod method,String key){

    String tmsLinksAnnotationText =
      "@io..TmsLink(\"" + key + "\")";
    PsiAnnotation tmsLink =
      method.addAnnotation(tmsLinkAnnotationText);
    JavaCodeStyleManager.getInstance(project)
        .shortenClassReferences(tmsLink);
  }
}
```

Добавили аннотацию

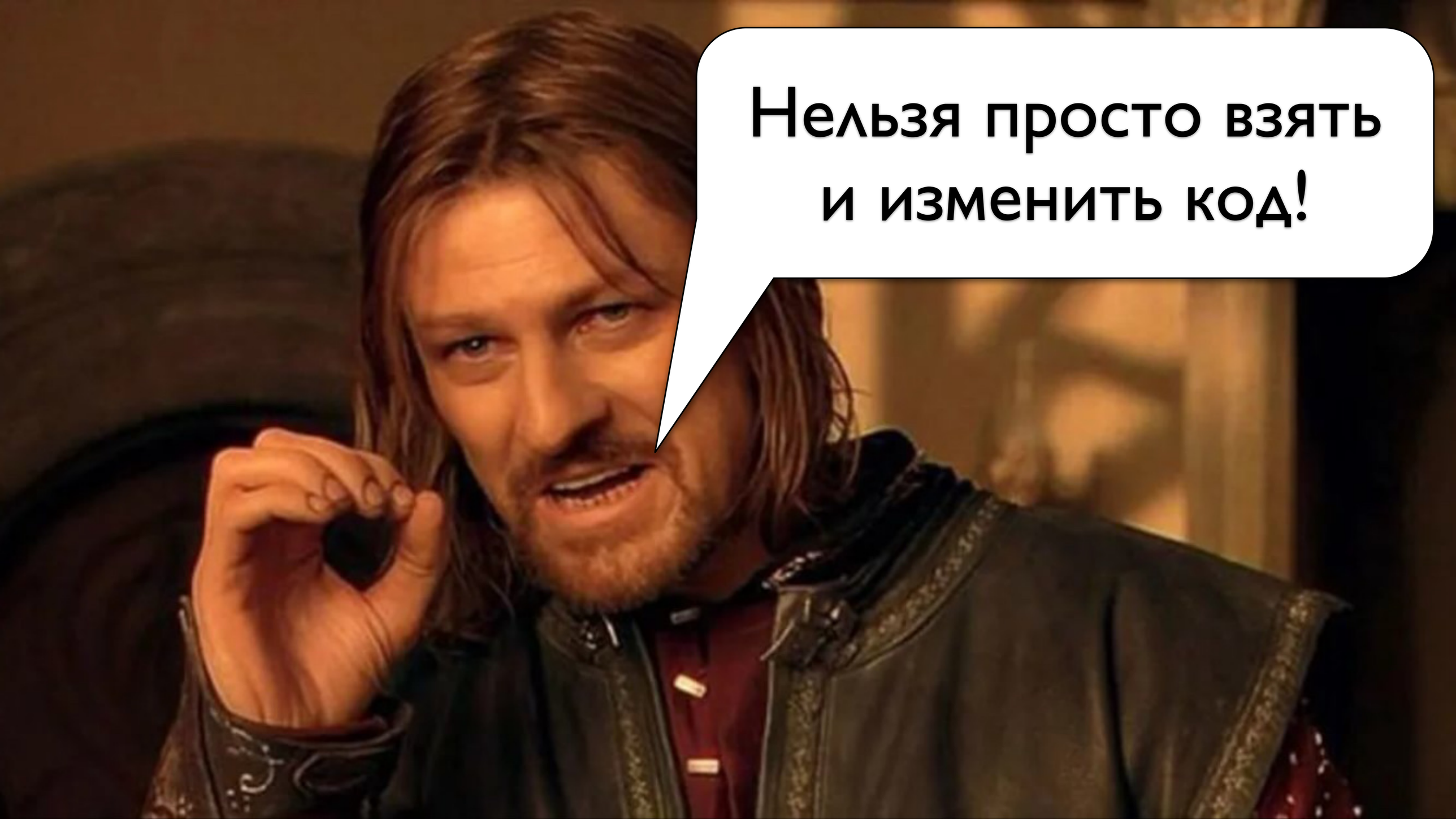
```
public class AddFavoritesAfterNoteTest {

  @Test
  @io.qameta.allure.TmsLink("AE-5")
  @DisplayName("Добавление в избранное после создания заметки")
  public void shouldAddToFavoriteAfterNodeTest() { ... }

  @Test
  @DisplayName("Удаление из избранного после удаления заметки")
  public void shouldDeleteToFavoriteAfterNodeTest() { ... }

}
```

Формируем текст аннотации

```
public class JiraKeyImportAction extends AnAction {

  void addTmsLinkToMethod(PsiMethod method,String key){

    String tmsLinksAnnotationText =
      "@io..TmsLink(\"" + key + "\")";
    PsiAnnotation tmsLink =
      method.addAnnotation(tmsLinkAnnotationText);
    JavaCodeStyleManager.getInstance(project)
        .shortenClassReferences(tmsLink);
  }
}
```

Добавили импорт аннотации

```
public class AddFavoritesAfterNoteTest {

    @Test
    @TmsLink("AE-5")
    @DisplayName("Добавление в избранное после создания заметки")
    public void shouldAddToFavoriteAfterNodeTest() { ... }

    @Test
    @DisplayName("Удаление из избранного после удаления заметки")
    public void shouldDeleteToFavoriteAfterNodeTest() { ... }

}
```


Нельзя просто взять
и изменить код!

Обертка над изменением кода

```
public class JiraKeyImportAction extends AnAction {

  void addTmsLinkToMethod(PsiMethod method,String key){

    CommandProcessor.getInstance().executeCommand(()->{
      getApplication().runWriteAction(()->{
        //код с предыдущего слайда
      })
    })
  }
}
```


Web 3.0

Open issues

Switch filter

View all issues and filters

Order by Priority

- AE-5 Добавление в избранное после создания за…
- AE-4 Удаление из избранного после удаления за…
- AE-6 Авторизация пользователя

Allure Examples / AE-4

2 of 3

Удаление из избранного после удаления заметки

Edit | Comment | Assign | More | Start Progress | Done | Admin | Export

Details

Type: Task
Status: TO DO (View Workflow)
Priority: Medium
Resolution: Unresolved
Labels: favorites

People

Assignee: Unassigned
Assign to me

Reporter: admin

Votes: 0

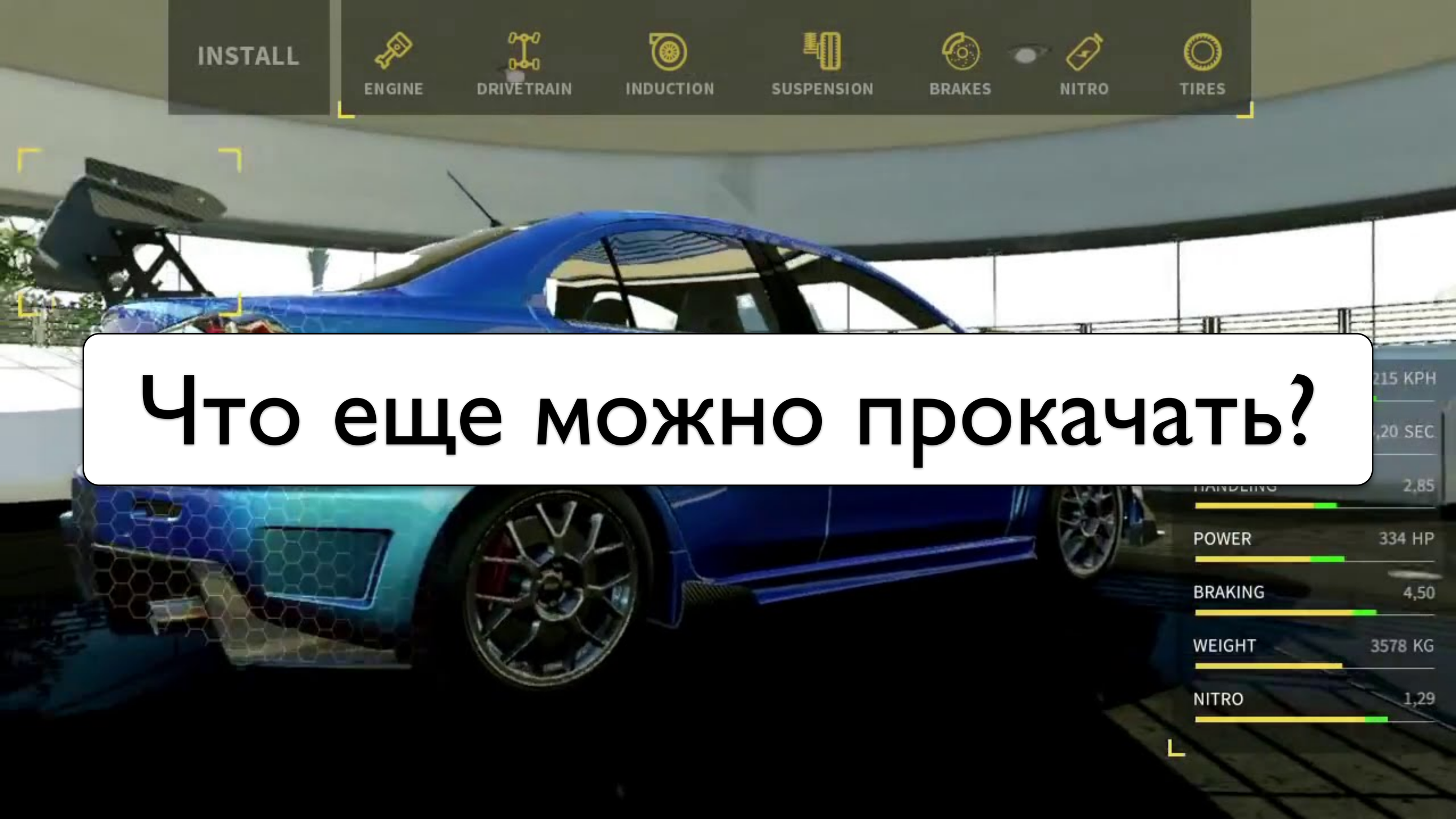
Watchers: 1 Stop watching this issue

Description

Click to add description

Attachments

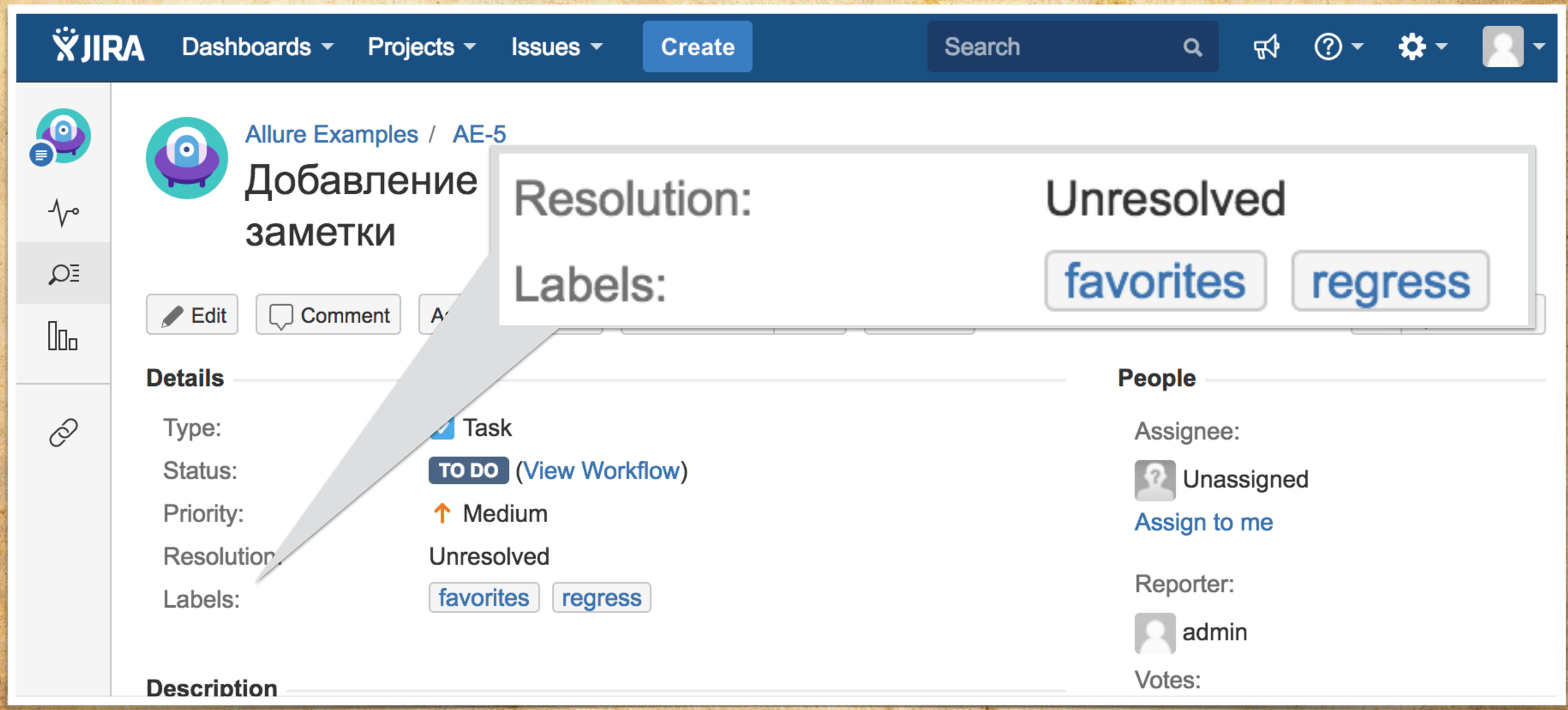
Drop files to attach, or browse.



Что еще можно прокачать?

Теги для запуска тестов

Labels для разметки тестов



Создаем JiraLabelsImportAction

```
public class JiraLabelsImportAction extends AnAction {

  @Override
  public void actionPerformed(AnActionEvent event) {

    PsiElement element = event.getData(PSI_ELEMENT)
    if (element instance of PsiClass) {
      addTagsToClassMethods((PsiClass)element)
    }

  }
}
```

Фильтруем методы с ключом

```
public class JiraLabelsImportAction extends AnAction {

  void addTagsToClassMethods(PsiClass testClass) {

    Arrays.stream(testClass.getMethods())
      .filter(m -> m.hasAnnotation("org..TmsLink"))
      .forEach(this::addTmsLinkToMethod)

  }
}
```

Зачитываем ключ Issue в Jira

```
public class JiraLabelsImportAction extends AnAction {
  void addTagsToMethod(PsiMethod testMethod) {
    PsiAnnotation tmsLink =
      testMethod.getAnnotation("io..TmsLink");
    String key = getTextValue(tmsLink);
    List<String> labels = jiraClient.getLables(key);
    String tagsText = getTagsText(labels);
    PsiAnnotation tags = create(tagsText, testMethod);
    testMethod.getModifiersList()
      .addAfter(tags, tmsLink);
  }
}
```

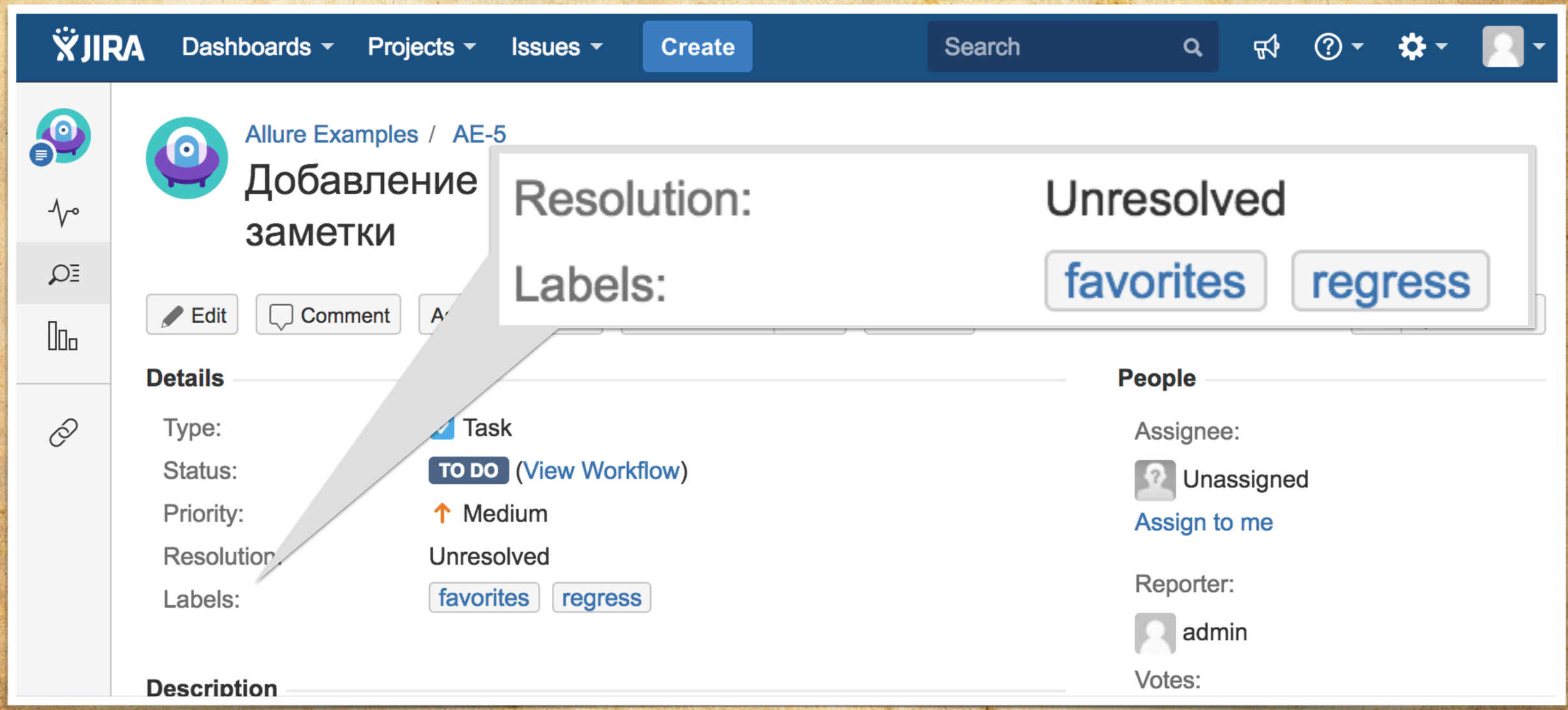
Получили ключи тестов

```
public class AddFavoritesAfterNoteTest {

    @Test
    @TmsLink("AE-5")
    @DisplayName("Добавление в избранное после создания заметки")
    public void shouldAddToFavoriteAfterNodeTest() { ... }

    @Test
    @TmsLink("AE-4")
    @DisplayName("Удаление из избранного после удаления заметки")
    public void shouldDeleteToFavoriteAfterNodeTest() { ... }

}
```

Забираем лейбочки из Jira

```
public class JiraLabelsImportAction extends AnAction {
  void addTagsToMethod(PsiMethod testMethod) {
    PsiAnnotation tmsLink =
      testMethod.getAnnotation("io..TmsLink");
    String key = getTextValue(tmsLink);
    List<String> labels = jiraClient.getLables(key);
    String tagsText = getTagsText(labels);
    PsiAnnotation tags = create(tagsText, testMethod);
    testMethod.getModifiersList()
      .addAfter(tags, tmsLink);
  }
}
```

Labels для разметки тестов



Создаем строку с аннотацией

```
public class JiraLabelsImportAction extends AnAction {
  void addTagsToMethod(PsiMethod testMethod) {
    PsiAnnotation tmsLink =
      testMethod.getAnnotation("io..TmsLink");
    String key = getTextValue(tmsLink);
    List<String> labels = jiraClient.getLables(key);
    String tagsText = getTagsText(labels);
    PsiAnnotation tags = create(tagsText, testMethod);
    testMethod.getModifiersList()
      .addAfter(tags, tmsLink);
  }
}
```

Аннотация в виде строки

```
public String getInnerTagsText(List<String> labels) {
  return features.stream()
    .map(f -> "@org..Tag(\"" + f + "\")")
    .collect(Collectors.joining(","))
}

public String getTagsText(List<String> labels) {
  String inner = getInnerTagsText(labels);
  return "@org..Tag({" + inner + "})";
}
```

@org..Tags({@org..Tag("regress"), ...})

Создаем аннотацию из строки

```
public class JiraLabelsImportAction extends AnAction {
  void addTagsToMethod(PsiMethod testMethod) {
    PsiAnnotation tmsLink =
      testMethod.getAnnotation("io..TmsLink");
    String key = getTextValue(tmsLink);
    List<String> labels = jiraClient.getLables(key);
    String tagsText = getTagsText(labels);
    PsiAnnotation tags = create(tagsText, testMethod);
    testMethod.getModifiersList()
      .addAfter(tags, tmsLink);
  }
}
```

@Tags после @TmsLink

```
public class JiraLabelsImportAction extends AnAction {
  void addTagsToMethod(PsiMethod testMethod) {
    PsiAnnotation tmsLink =
      testMethod.getAnnotation("io..TmsLink");
    String key = getTextValue(tmsLink);
    List<String> labels = jiraClient.getLables(key);
    String tagsText = getTagsText(labels);
    PsiAnnotation tags = create(tagsText, testMethod);
    testMethod.getModifiersList()
      .addAfter(tags, tmsLink);
  }
}
```

Добавили теги в тест

```
public class AddFavoritesAfterNoteTest {

   @Test
   @TmsLink("AE-5")
   @org..Tags({@org..Tag("regress")})
   @DisplayName("Добавление в избранное после создания заметки")
   public void shouldAddToFavoriteAfterNodeTest() { ... }

}
```

Оптимизируем импорты

```
public class JiraLabelsImportAction extends AnAction {

  void addTagsToMethod(PsiMethod testMethod) {

    addImport(testMethod.getContainingFile(), "org..Tag");
    addImport(testMethod.getContainingFile(), "org..Tags");

    // код из предыдущего слайда с write операцией

    optimizeImports(testMethod.getContainingFile());
  }
}
```

Оптимизировали импорты

```
public class AddFavoritesAfterNoteTest {

   @Test
   @TmsLink("AE-5")
   @Tags({@Tag("regress")})
   @DisplayName("Добавление в избранное после создания заметки")
   public void shouldAddToFavoriteAfterNodeTest() { ... }

}
```


Web 3.0

Open issues

Switch filter

View all issues and filters

Order by Priority

- AE-5 Добавление в избранное после создания за…
- AE-4 Удаление из избранного после удаления за…
- AE-6 Авторизация пользователя

Allure Examples / AE-4

2 of 3

Удаление из избранного после удаления заметки

Edit | Comment | Assign | More | Start Progress | Done | Admin | Export

Details

Type: Task
Status: TO DO (View Workflow)
Priority: Medium
Resolution: Unresolved
Labels: favorites

Description

Click to add description

Attachments

Drop files to attach, or browse.

People

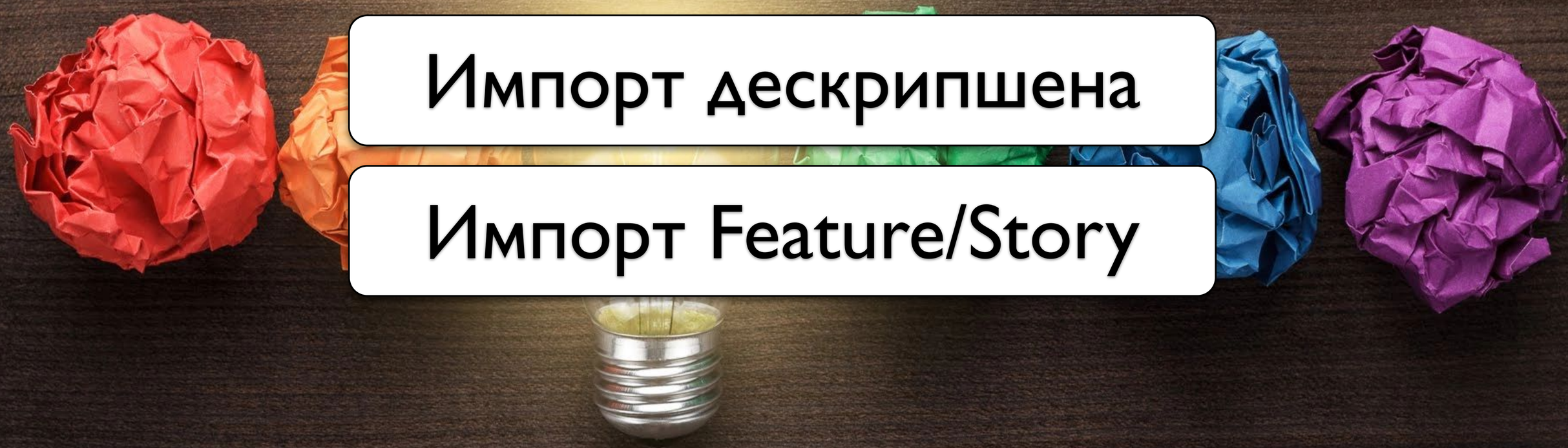
Assignee: Unassigned
Assign to me

Reporter: admin

Votes: 0

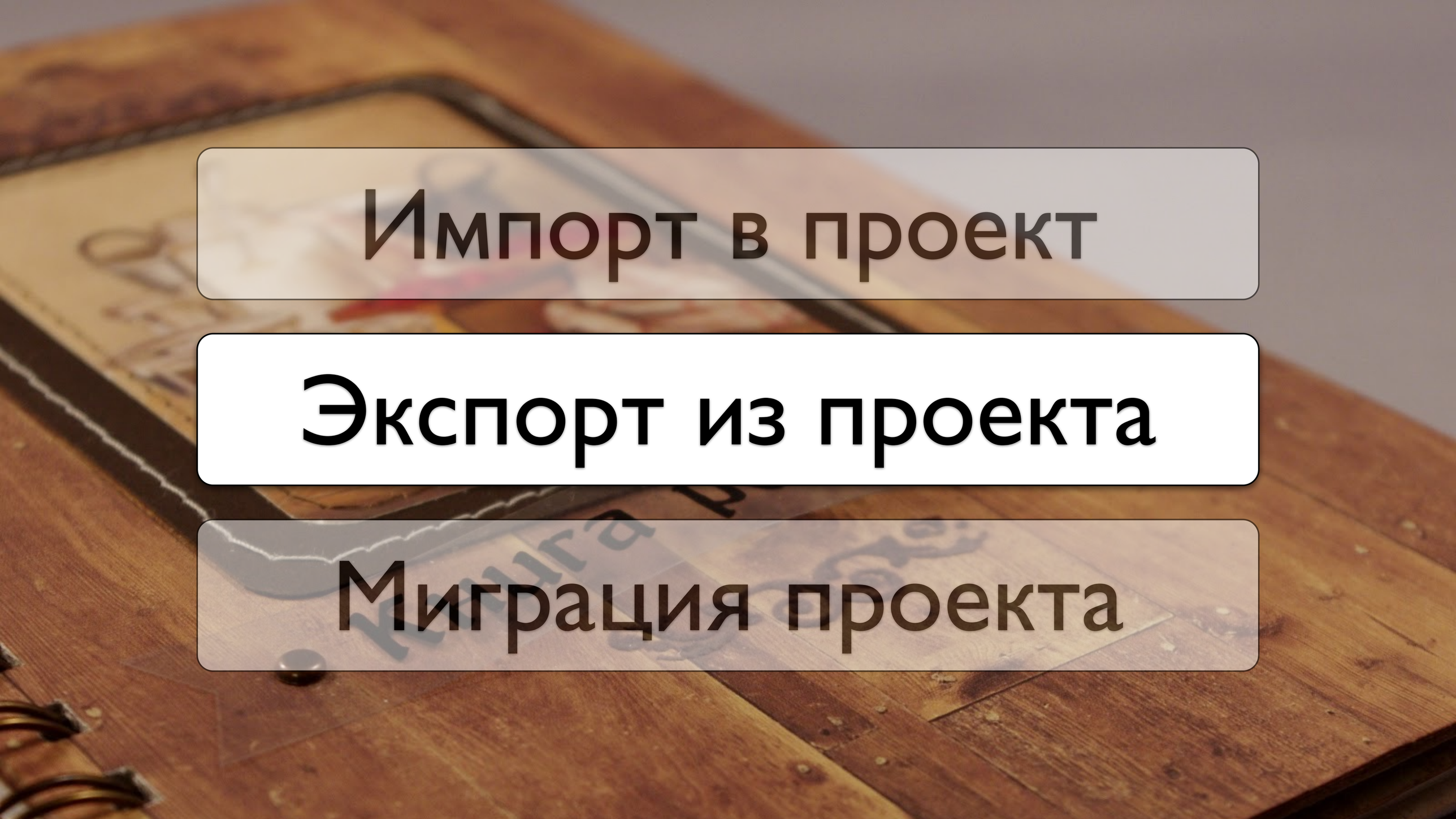
Watchers: 1 Stop watching this issue

Какие еще есть идеи?



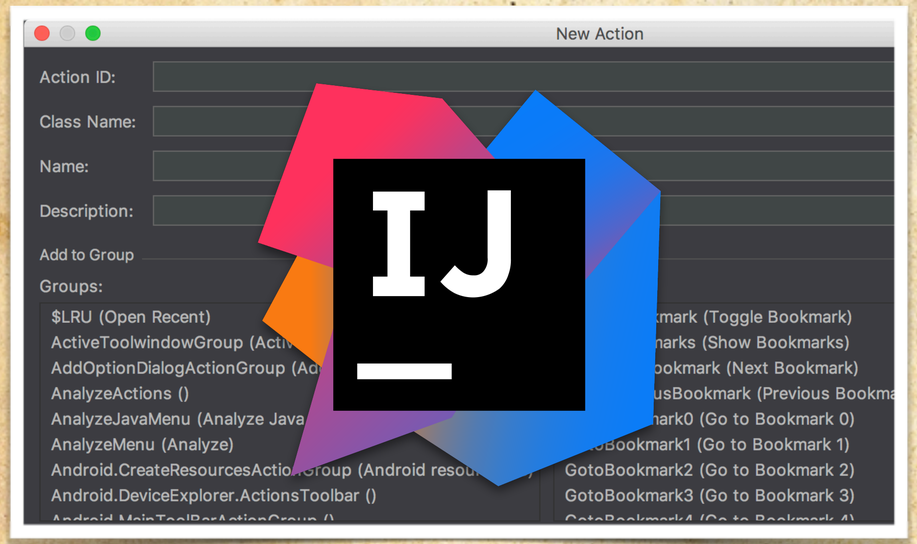
Импорт дескрипшена

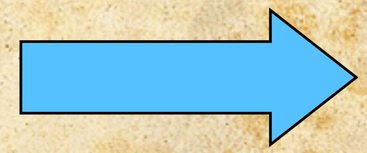
Импорт Feature/Story

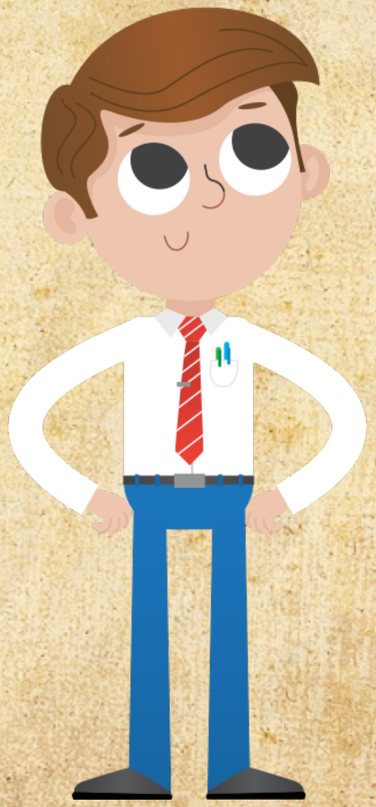

Импорт в проект
Экспорт из проекта
Миграция проекта

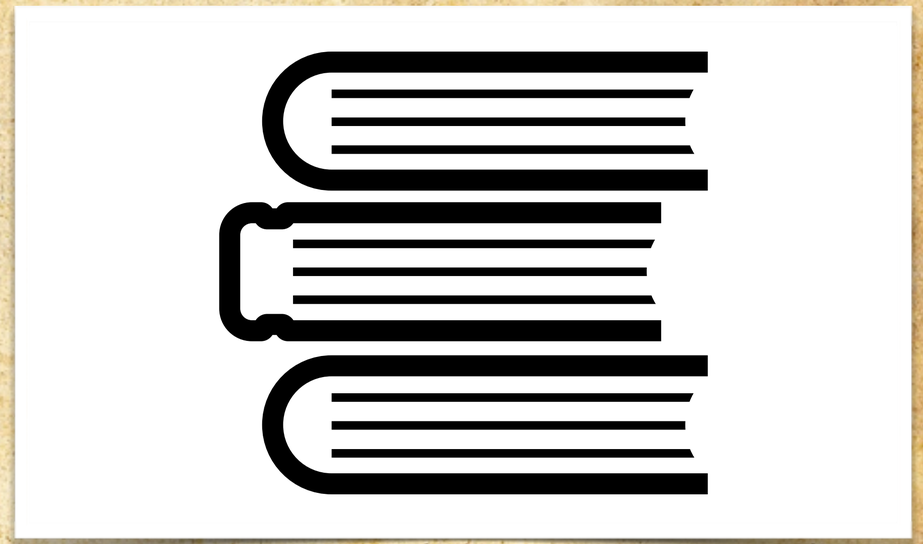
Какую проблему решаем?

А что в тестах проверяется?

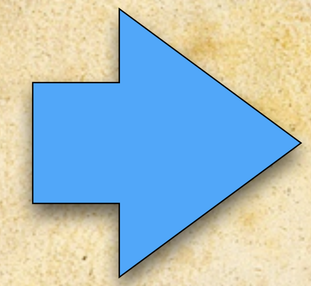


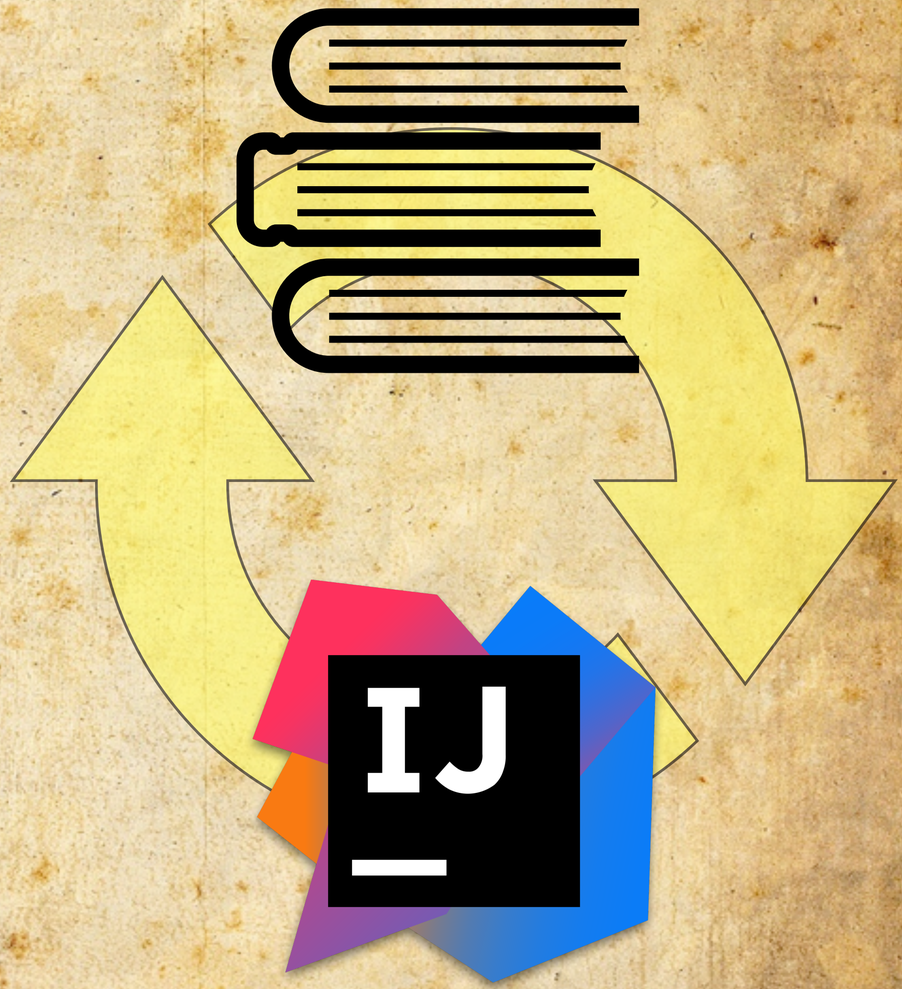


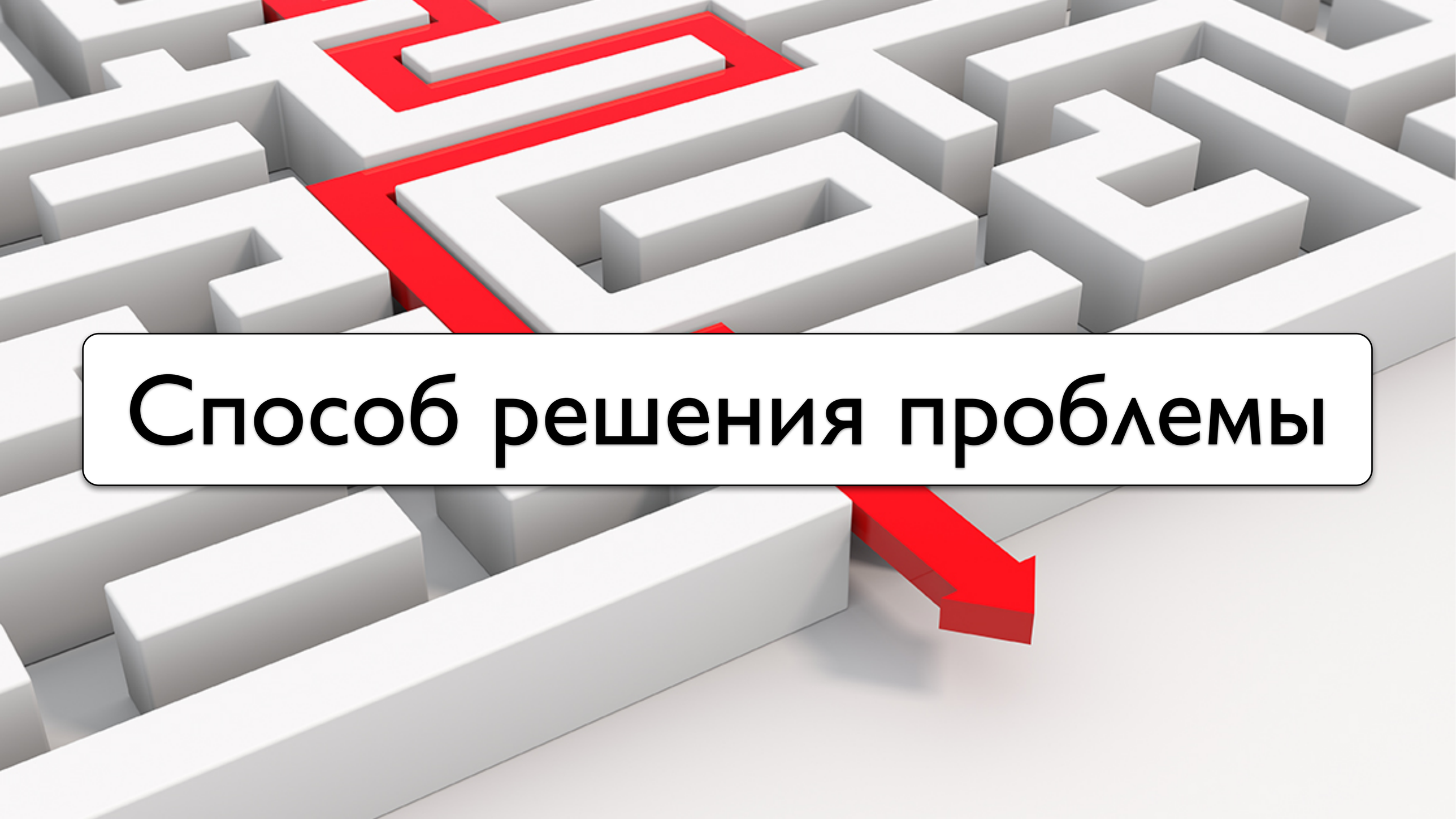




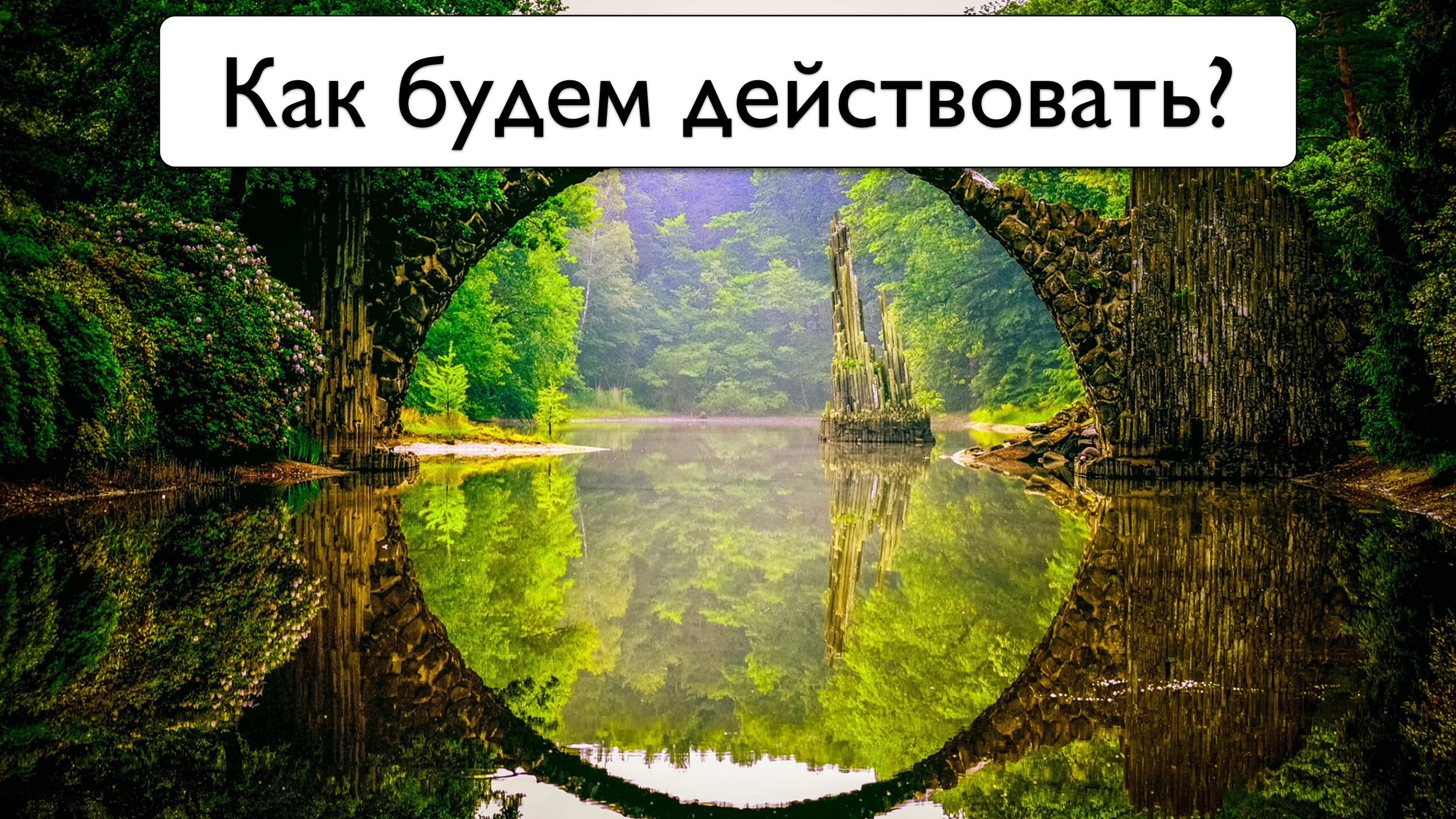
Задача на автоматизацию



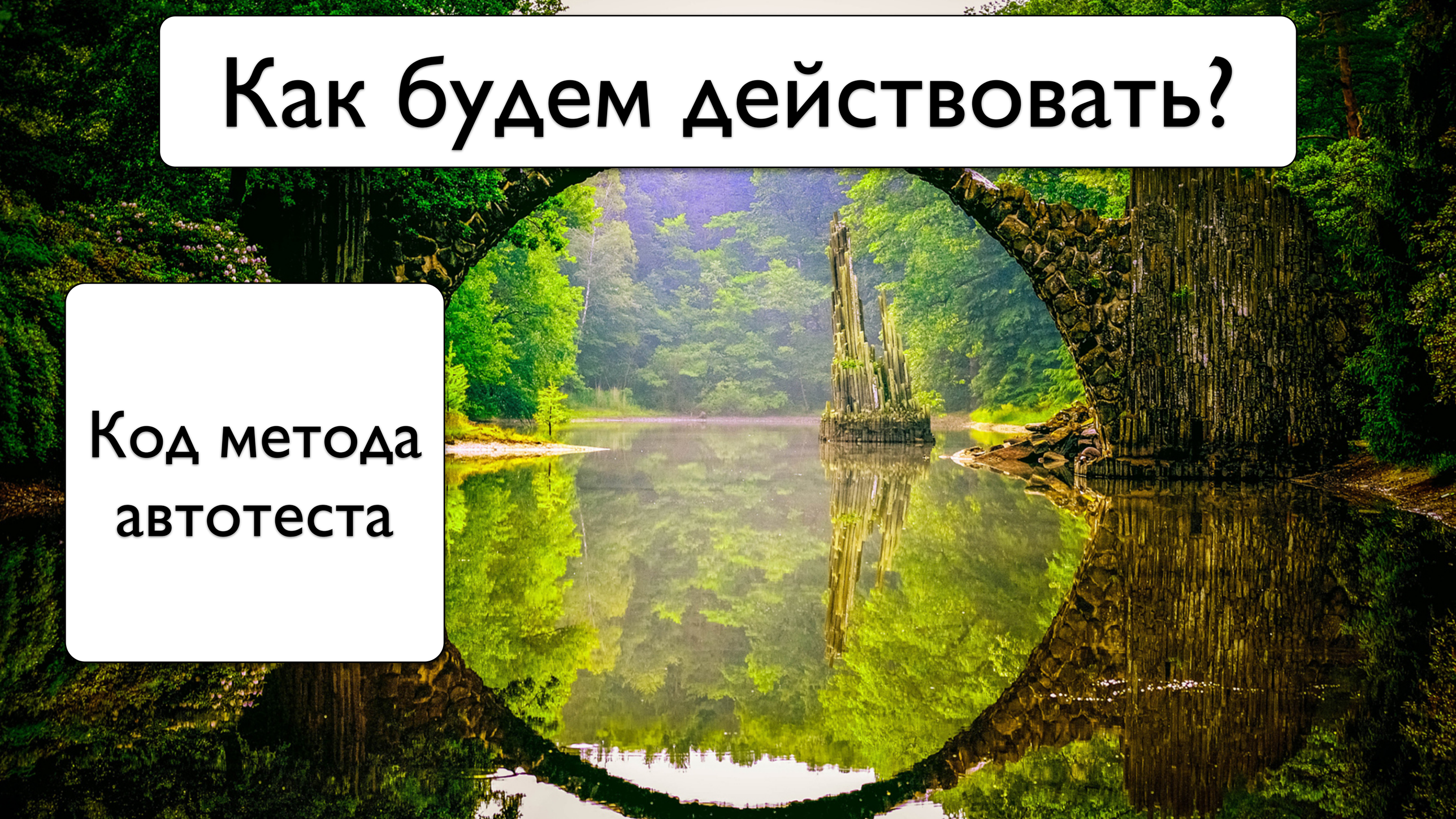



Способ решения проблемы

Как будем действовать?



Как будем действовать?



Код метода автотеста

Мета информация

```
public class AddFavoritesAfterNoteTest {

    @Test
    @TmsLink("AE-5")
    @Features({@Feature("Favorites")})
    @Stories({@Story("Add favorites after note")})
    @DisplayName("Добавление в избранное после создания заметки")
    public void shouldAddToFavoriteAfterNodeTest() {
        steps.openMainPage();
        ...
    }
}
```

Сценарий теста

```
public class AddFavoritesAfterNoteTest {

    @Test
    @TmsLink("AE-5")
    @Features({@Feature("Favorites")})
    @Stories({@Story("Add favorites after note")})
    @DisplayName("Добавление в избранное после создания заметки")
    public void shouldAddToFavoriteAfterNodeTest() {
        steps.openMainPage();
        ...
    }
}
```

Как будем действовать?

Код метода автотеста

Соберем все данные

Информация для экспорта

```
public class TestCase implements Serializable {

   private String id;
   private String name;
   private List<String> features;
   private List<String> stories;
   private List<String> steps;

   ...
}
```

Как будем действовать?

Код метода автотеста

Соберем все данные

Сохраним в удобном формате

HTML-шаблон на Freemarker

<#FREEMARKER> Home | Manual | Java API

WHAT IS APACHE FREEMARKER™?

DOWNLOAD / MAVEN

DOCUMENTATION

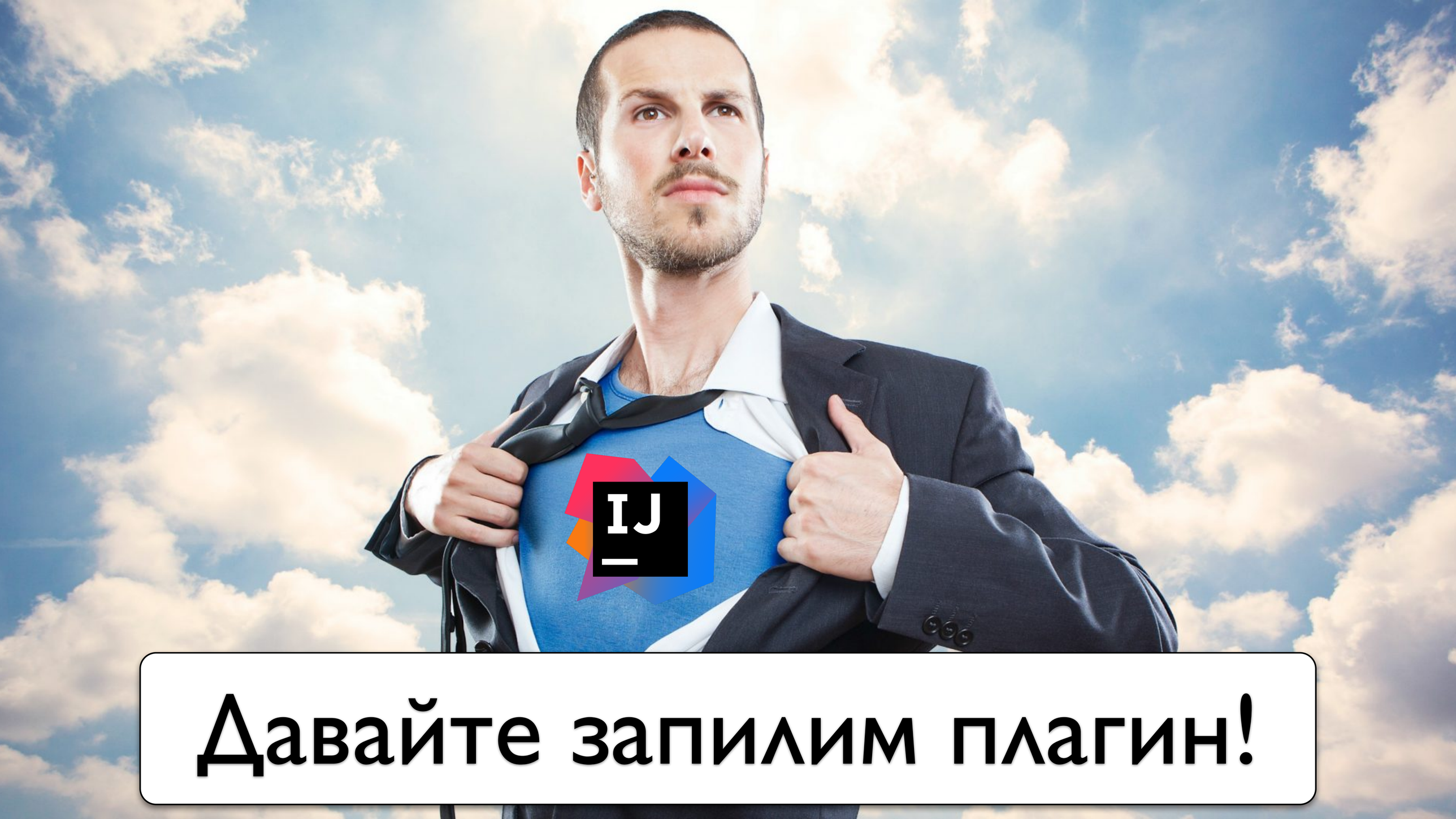
- Manual
- Java API
- Manual Chinese translation

TOOLING

- Editor and IDE plugins
- Online template tester

What is Apache FreeMarker™?

Apache FreeMarker™ is a *template engine*: a Java library to generate text output (HTML web pages, e-mails, configuration files, source code, etc.) based on templates and changing data. Templates are written in the FreeMarker Template Language (FTL), which is a simple, specialized language (not a full-blown programming language like PHP). Usually, a general-purpose programming language (like Java) is used to prepare the data (issue database queries, do business calculations). Then, Apache FreeMarker displays that prepared data using templates. In the template you are focusing on how to present the data, and outside the template you are focusing on what data to



Давайте запилим плагин!

Создаем TestCaseExportAction

```
public class TestCaseExportAction extends AnAction {
  @Override
  public void actionPerformed(AnActionEvent event) {
    PsiElement element = event.getData(PSI_ELEMENT);
    List<TestCase> testcases =
      Arrays.stream(((PsiClass) element).getMethods())
        .filter(m -> m.hasAnnotation("org..Test"))
        .map(this::getTestCaseFromMethod)
        .collect(Collectors.toList());
    writeTestCases(event.getProject(), testcases);
  }
}
```

Берем все методы тестов

```
public class TestCaseExportAction extends AnAction {
  @Override
  public void actionPerformed(AnActionEvent event) {
    PsiElement element = event.getData(PSI_ELEMENT);
    List<TestCase> testcases =
      Arrays.stream(((PsiClass) element).getMethods())
        .filter(m -> m.hasAnnotation("org..Test"))
        .map(this::getTestCaseFromMethod)
        .collect(Collectors.toList());
    writeTestCases(event.getProject(), testcases);
  }
}
```

Собираем информацию

```
public class TestCaseExportAction extends AnAction {
  @Override
  public void actionPerformed(AnActionEvent event) {
    PsiElement element = event.getData(PSI_ELEMENT);
    List<TestCase> testcases =
      Arrays.stream(((PsiClass) element).getMethods())
        .filter(m -> m.hasAnnotation("org..Test"))
        .map(this::getTestCaseFromMethod)
        .collect(Collectors.toList());
    writeTestCases(event.getProject(), testcases);
  }
}
```

Сохраняем результат

```
public class TestCaseExportAction extends AnAction {
  @Override
  public void actionPerformed(AnActionEvent event) {
    PsiElement element = event.getData(PSI_ELEMENT);
    List<TestCase> testcases =
      Arrays.stream(((PsiClass) element).getMethods())
        .filter(m -> m.hasAnnotation("org..Test"))
        .map(this::getTestCaseFromMethod)
        .collect(Collectors.toList());
    writeTestCases(event.getProject(), testcases);
  }
}
```

Как собираем данные метода?

```
public class TestCaseExportAction extends AnAction {

  public TestCase getTestCaseFromMethod(PsiMethod method) {
    TestCase testCase = new TestCase();
    testCase.setId(getTmsLinkText(method));
    testCase.setName(getDisplayNameText(method));
    testCase.setFeature(getFeaturesText(method));
    testCase.setStories(getStoriesText(method));
    testCase.setSteps(getSteps(method));
    return testCase;
  }
}
```

Уже делали в первом плагине

```
public class TestCaseExportAction extends AnAction {

  public TestCase getTestCaseFromMethod(PsiMethod method) {
    TestCase testCase = new TestCase();
    testCase.setId(getTmsLinkText(method));
    testCase.setName(getDisplayNameText(method));
    testCase.setFeature(getFeaturesText(method));
    testCase.setStories(getStoriesText(method));
    testCase.setSteps(getSteps(method));
    return testCase;
  }
}
```

Уже делали в первом плагине

```
public class AddFavoritesAfterNoteTest {

    @Test
    @TmsLink("AE-5")
    @Features({@Feature("Favorites")})
    @Stories({@Story("Add favorites after note")})
    @DisplayName("Добавление в избранное после создания заметки")
    public void shouldAddToFavoriteAfterNodeTest() {
        steps.openMainPage();
        ...
    }
}
```

Тоже самое, только массивы

```
public class TestCaseExportAction extends AnAction {

  public TestCase getTestCaseFromMethod(PsiMethod method) {
    TestCase testCase = new TestCase();
    testCase.setId(getTmsLinkText(method));
    testCase.setName(getDisplayNameText(method));
    testCase.setFeature(getFeaturesText(method));
    testCase.setStories(getStoriesText(method));
    testCase.setSteps(getSteps(method));
    return testCase;
  }
}
```

Тоже самое, только массивы

```
public class AddFavoritesAfterNoteTest {

    @Test
    @TmsLink("AE-5")
    @Features({@Feature("Favorites")})
    @Stories({@Story("Add favorites after note")})
    @DisplayName("Добавление в избранное после создания заметки")
    public void shouldAddToFavoriteAfterNodeTest() {
        steps.openMainPage();
        ...
    }
}
```

А вот это не тривиально

```
public class TestCaseExportAction extends AnAction {

  public TestCase getTestCaseFromMethod(PsiMethod method) {
    TestCase testCase = new TestCase();
    testCase.setId(getTmsLinkText(method));
    testCase.setName(getDisplayNameText(method));
    testCase.setFeature(getFeaturesText(method));
    testCase.setStories(getStoriesText(method));
    testCase.setSteps(getSteps(method));
    return testCase;
  }
}
```

Сценарий теста

```
public class AddFavoritesAfterNoteTest {

   @Test
   public void shouldAddToFavoriteAfterNodeTest() {
      steps.openMainPage();
      steps.openAutoCardPage("Volvo XC90")
      steps.addNotesToAutoCard("в хорошем состоянии");
      steps.openFavoritesPage("в хорошем состоянии");
      steps.checkFavoritesListContains("Volvo XC90")
   }
}
```

Добываем список Шагов

```
public class ScenarioExportAction extends AnAction {
  public List<String> getSteps(PsiMethod testMethod) {
    Arrays.stream(method.getBody().getStatements())
      .filter(PsiMethodCallExpression.class::isInstance)
      .map(PsiMethodClassExpression.class::cast)
      .map(PsiMethodClassExpression::resolveMethod())
      .filter(m -> m.hasAnnotation("...Step"))
      .map(m -> m.getAnnotation("...Step"))
      .map(a -> a.findDeclaredAttribute("value"))
      .map(PsiAnnotationMemberValue::getText)
      .forEach(Collectors.toList())
  }
}
```

Не метод, а оператор вызова

```
public class AddFavoritesAfterNoteTest {

    @Test
    public void shouldAddToFavoritesAfterNodeTest() {
        steps.openMainPage();
        steps.openAutoCardPage("Volvo XC 90")
        steps.addNotesToAutoCard("в хорошем состоянии");
        steps.openFavoritesPage();
        steps.checkFavoritesListContains("Volvo XC90")
    }
}
```

PsiMethodCallExpression

Добываем список методов

```
public class ScenarioExportAction extends AnAction {
  public List<String> getSteps(PsiMethod testMethod) {
    Arrays.stream(method.getBody().getStatements())
      .filter(PsiMethodCallExpression.class::isInstance)
      .map(PsiMethodClassExpression.class::cast)
      .map(PsiMethodClassExpression::resolveMethod)
      .filter(m -> m.hasAnnotation("...Step"))
      .map(m -> m.getAnnotation("...Step"))
      .map(a -> a.findDeclaredAttribute("value"))
      .map(PsiAnnotationMemberValue::getText)
      .forEach(Collectors.toList())
  }
}
```

Получили реальные методы

```
public class BasicSteps {

  @Step("Открываем главную страни
  public void openMainPage() { ..
  @Step("Открываем страницу маши  марки {mark}")
  public void openAutoCardPage(S ring mark) { ... }
  @Step("Добавляем заметку {tex } к машине")
  public void addNotesToAutoCa d(String text) { ... }

  private void utilityMethod { ... }

}
```

конкретный PsiMethod

Добываем список шагов

```
public class ScenarioExportAction extends AnAction {
  public List<String> getSteps(PsiMethod testMethod) {
    Arrays.stream(method.getBody().getStatements())
      .filter(PsiMethodCallExpression.class::isInstance)
      .map(PsiMethodClassExpression.class::cast)
      .map(PsiMethodClassExpression::resolveMethod())
      .filter(m -> m.hasAnnotation("...Step"))
      .map(m -> m.getAnnotation("...Step"))
      .map(a -> a.findDeclaredAttribute("value"))
      .map(PsiAnnotationMemberValue::getText)
      .forEach(Collectors.toList())
  }
}
```

Получили текст аннотации

```
public class BasicSteps {

  @Step("Открываем главную страницу")
  public void openMainPage() { ... }
  @Step("Открываем страницу машины марки {mark}")
  public void openAutoCardPage(String mark) { ... }
  @Step("Добавляем заметку {text} к машине")
  public void addNotesToAutoCard(String text) { ... }

  private void utilityMethod { ... }

}
```

Добываем список Шагов

```
public class ScenarioExportAction extends AnAction {
  public List<String> getSteps(PsiMethod testMethod) {
    Arrays.stream(method.getBody().getStatements())
      .filter(PsiMethodCallExpression.class::isInstance)
      .map(PsiMethodClassExpression.class::cast)
      .map(PsiMethodClassExpression::resolveMethod())
      .filter(m -> m.hasAnnotation("...Step"))
      .map(m -> m.getAnnotation("...Step"))
      .map(a -> a.findDeclaredAttribute("value"))
      .map(PsiAnnotationMemberValue::getText)
      .forEach(Collectors.toList())
  }
}
```

Как сохранить результат?

```
public class TestCaseExportAction extends AnAction {

  public void writeTestCases(Project project,
                             List<TestCase> testCases) {
     String content = FreemarketUtls
       .processTemplate("testcases.ftl", testCases);
     Path projectDir = Paths.get(project.getBasePath())
     Path indexFile = projectDir.resolve("index.html");
     Files.write(indexFile, content.getBytes());
  }

}
```

Шаблон "testcases.ftl"

```
<#list testcases as testcase>
  <h2>${testcase.id} - ${testcase.name}</h2>
  <div class="meta">
    <div class="features">${testcase.features}</div>
  </div>
  <ul class="list-group">
  <#list testcase.steps as step>
    <li class="list-group-item">${step}</li>
  </#list>
  </ul
</#list>
```

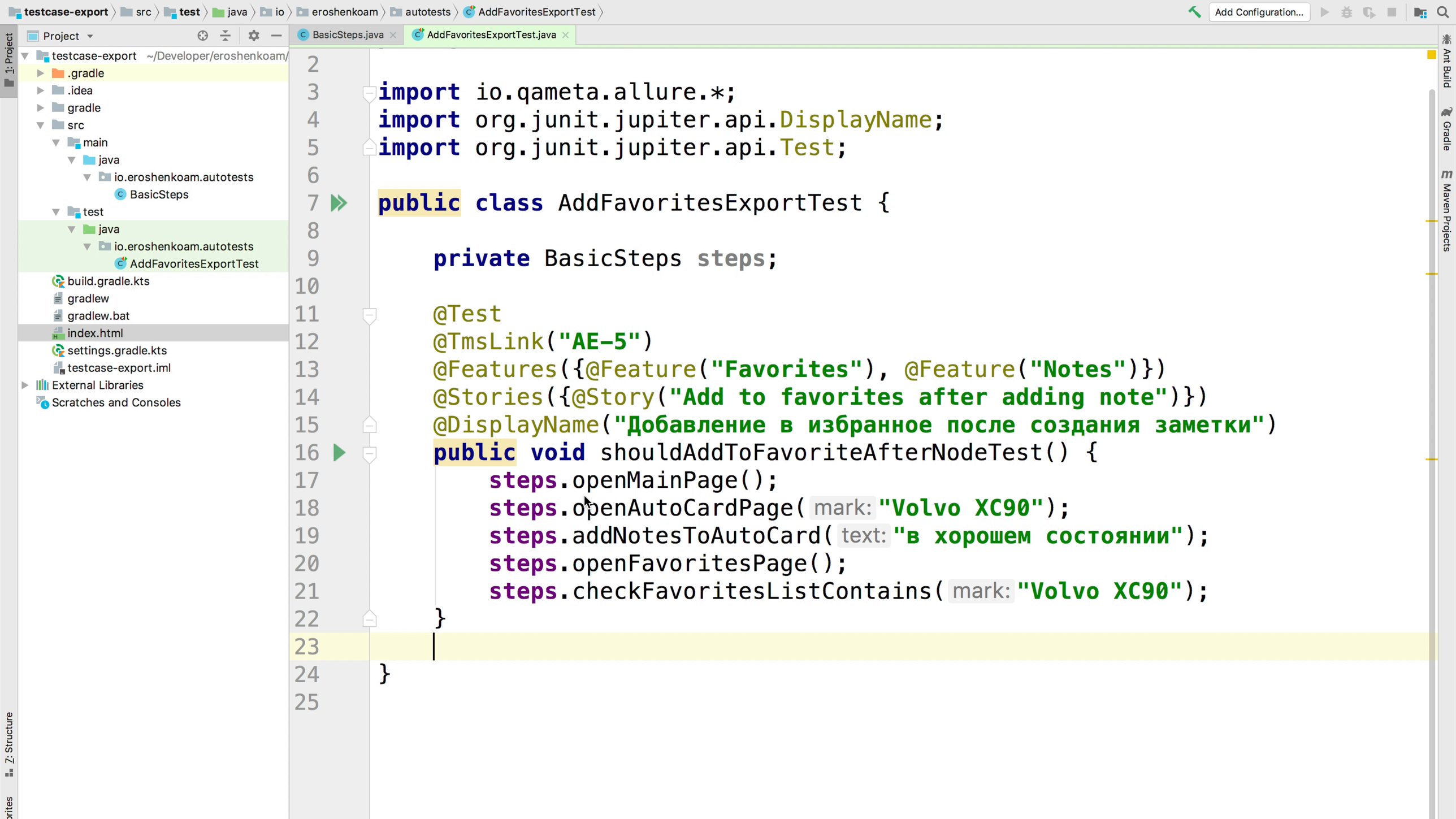
Заполняем шаблон данными

```
public class TestCaseExportAction extends AnAction {

  public void writeTestCases(Project project,
                             List<TestCase> testCases) {
    String content = FreemarketUtls
      .processTemplate("testcases.ftl", testCases);
    Path projectDir = Paths.get(project.getBasePath())
    Path indexFile = projectDir.resolve("index.html");
    Files.write(indexFile, content.getBytes());
  }

}
```

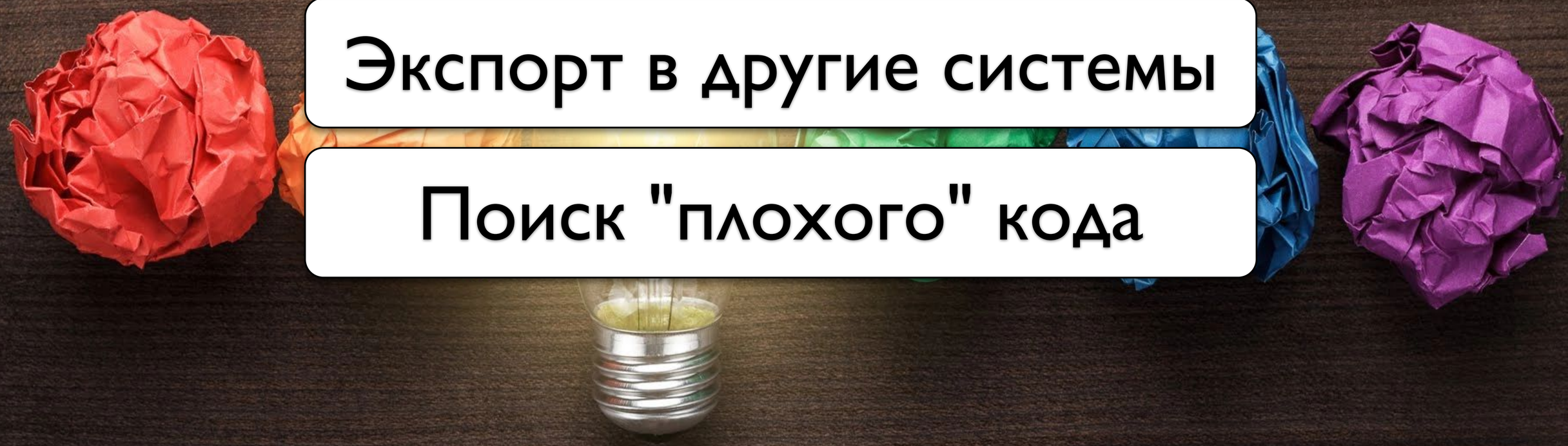
Сохраняем результат

```
public class TestCaseExportAction extends AnAction {

  public void writeTestCases(Project project,
                             List<TestCase> testCases) {
     String content = FreemarketUtls
       .processTemplate("testcases.ftl", testCases);
     Path projectDir = Paths.get(project.getBasePath())
     Path indexFile = projectDir.resolve("index.html");
     Files.write(indexFile, content.getBytes());
  }

}
```


Web 3.0

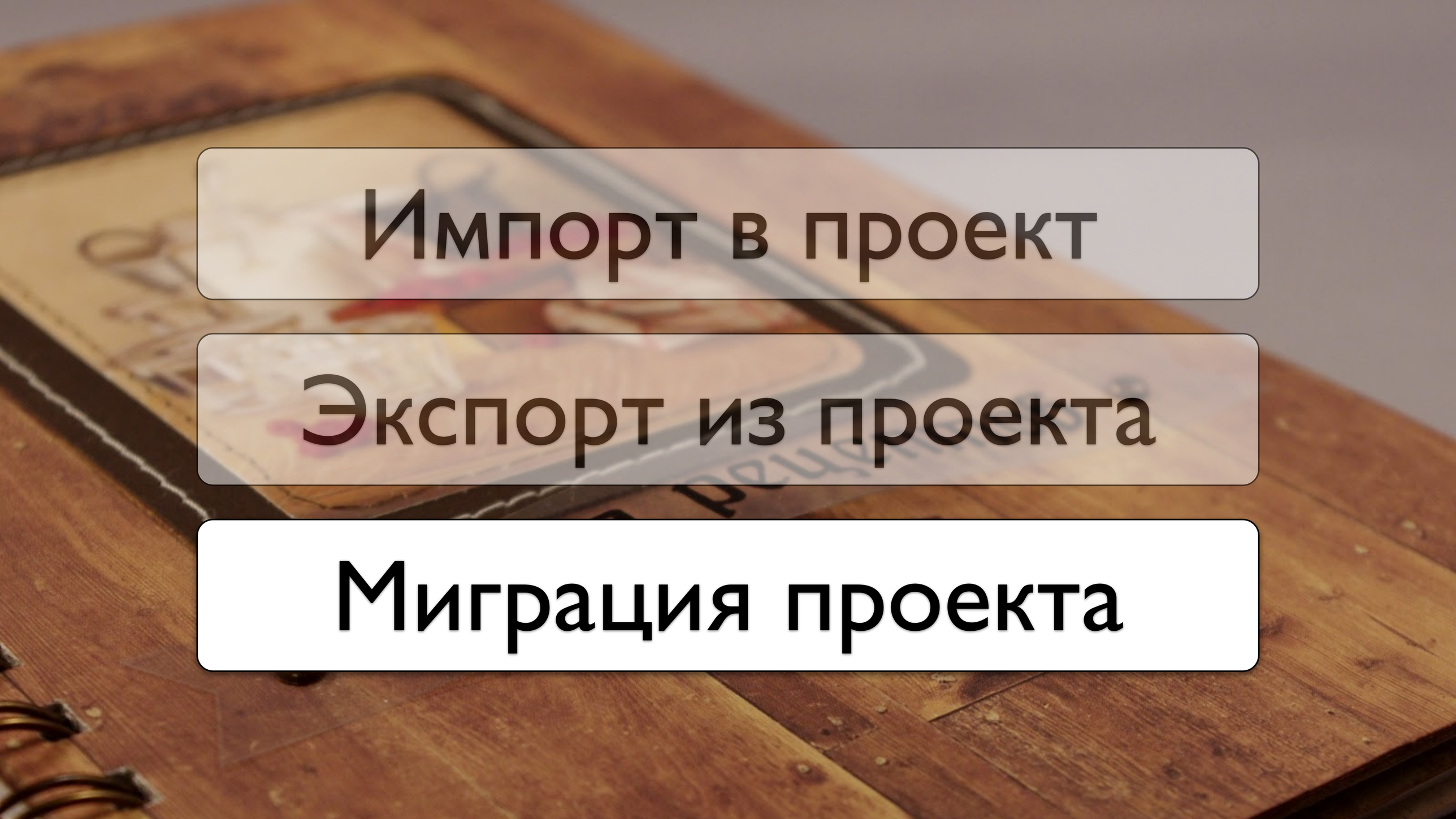

testcase-export
src
test
java
io
eroshenkoam
autotests
AddFavoritesExportTest
Add Configuration...
Project
BasicSteps.java
AddFavoritesExportTest.java
testcase-export ~/Developer/eroshenkoam/
.gradle
.idea
gradle
src
main
java
io.eroshenkoam.autotests
BasicSteps
test
java
io.eroshenkoam.autotests
AddFavoritesExportTest
build.gradle.kts
gradlew
gradlew.bat
index.html
settings.gradle.kts
testcase-export.iml
External Libraries
Scratches and Consoles
import io.qameta.allure.*;
import org.junit.jupiter.api.DisplayName;
import org.junit.jupiter.api.Test;
public class AddFavoritesExportTest {
private BasicSteps steps;
@Test
@TmsLink("AE-5")
@Features({@Feature("Favorites"), @Feature("Notes")})
@Stories({@Story("Add to favorites after adding note")})
@DisplayName("Добавление в избранное после создания заметки")
public void shouldAddToFavoriteAfterNodeTest() {
steps.openMainPage();
steps.openAutoCardPage(mark: "Volvo XC90");
steps.addNotesToAutoCard(text: "в хорошем состоянии");
steps.openFavoritesPage();
steps.checkFavoritesListContains(mark: "Volvo XC90");
}
}
1: Project
Ant Build
Gradle
Maven Projects
Z: Structure

Какие еще есть идеи?



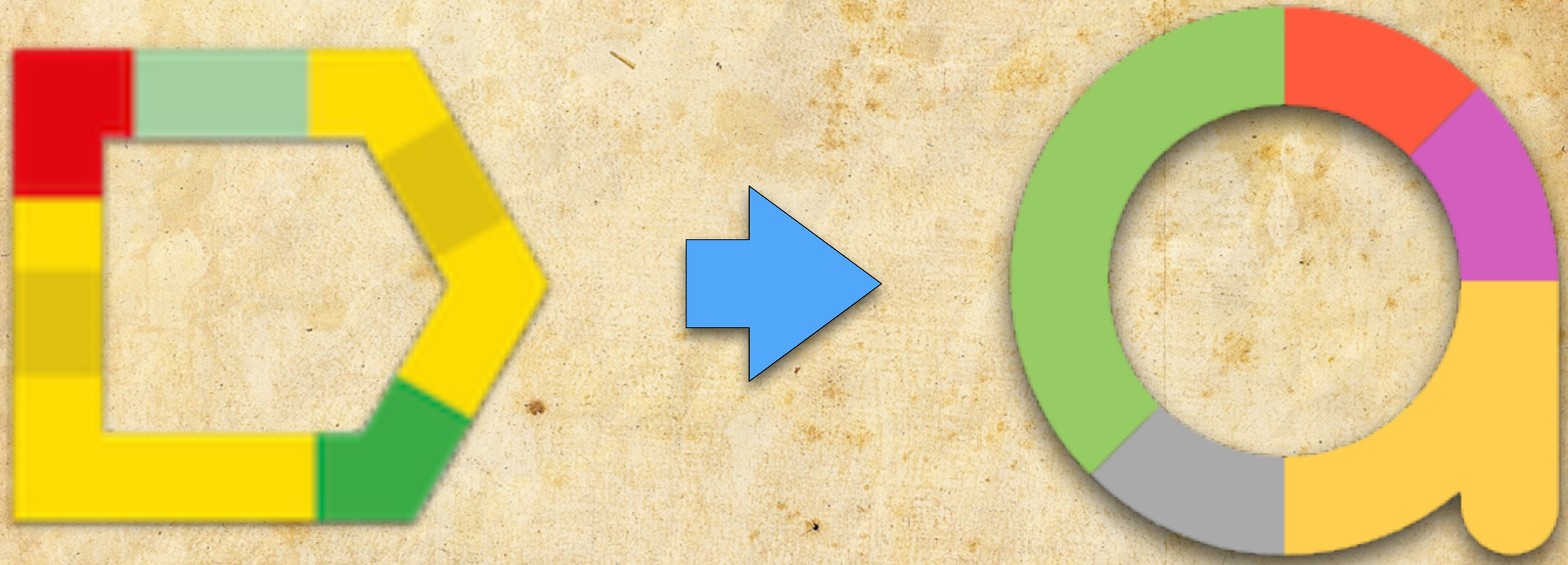
Экспорт в другие системы

Поиск "плохого" кода


Импорт в проект
Экспорт из проекта
Миграция проекта

Какую проблему решаем?

Переезд на второй Allure



Аннотация Name

```
@Test
@Name("My simple test here")
public void simpleTest() {
   //test body
}
```

```
@Test
@DisplayName("My simple test here")
public void simpleTest() {
   //test body here
}
```

Аннотация Feature

```
@Test
@Feature(["First", "Second"])
public void simpleTest() { .. }
```

```
@Test
@Features({
   @Feature("First"),
   @Feature("Second")
})
public void simpleTest() { ... }
```

Аннотация Story

```
@Test
@Story(["First", "Second"])
public void simpleTest() { .. }
```

```
@Test
@Stories({
   @Story("First"),
   @Story("Second")
})
public void simpleTest() { ... }
```

Аннотация Issue

```
@Test
@Issue(["AF-1", "AF-2"])
public void simpleTest() { .. }
```

```
@Test
@Issues({
   @Issue("AF-1"),
   @Issue("AF-2")
})
public void simpleTest() { ... }
```

Аннотация TmsLink

```
@Test
@TestCaseId(["AE-4", "AE-5"])
public void simpleTest() { .. }
```

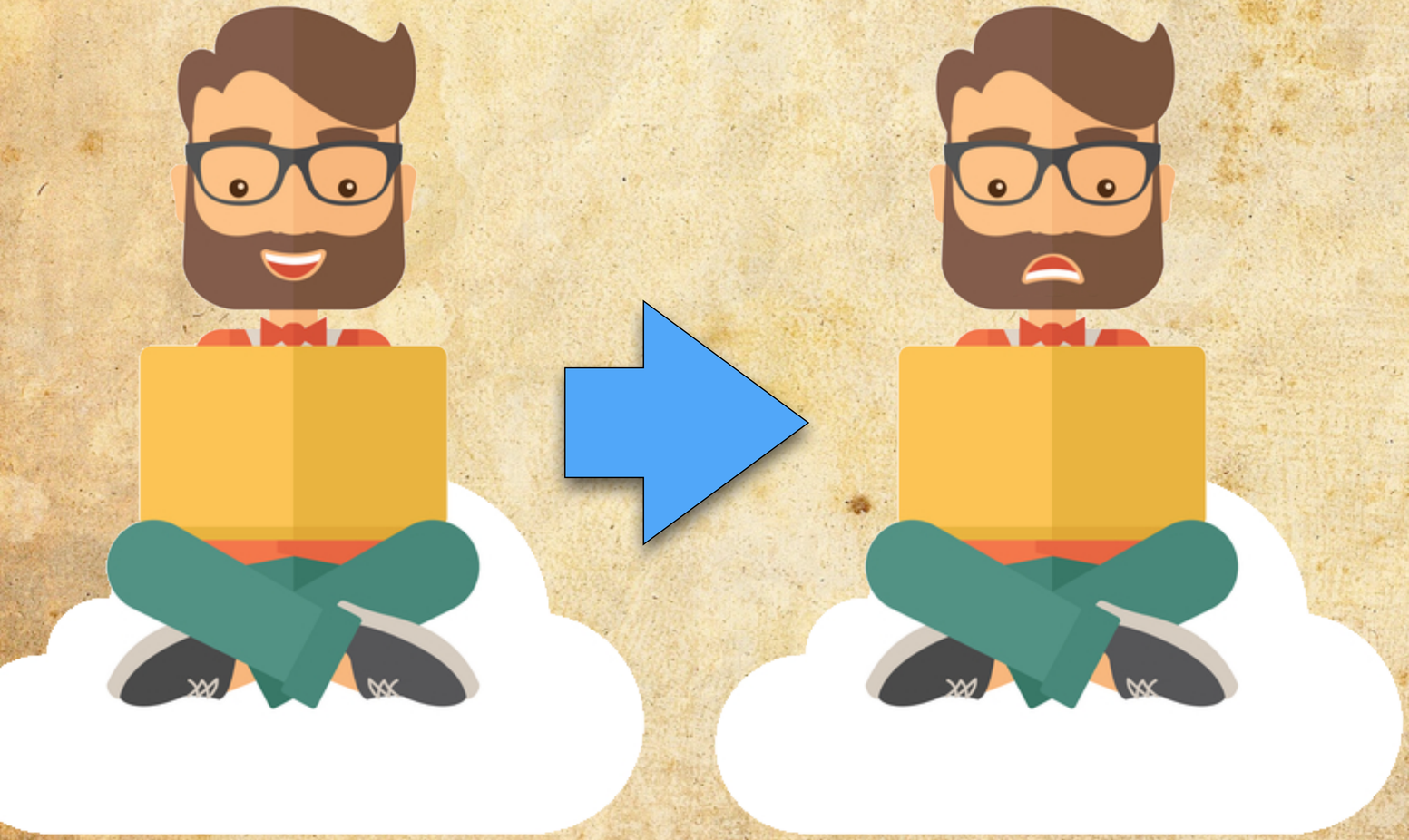
```
@Test
@TmsLinks({
   @TmsLink("AE-4"),
   @TmsLink("AE-5")
})
public void simpleTest() { ... }
```

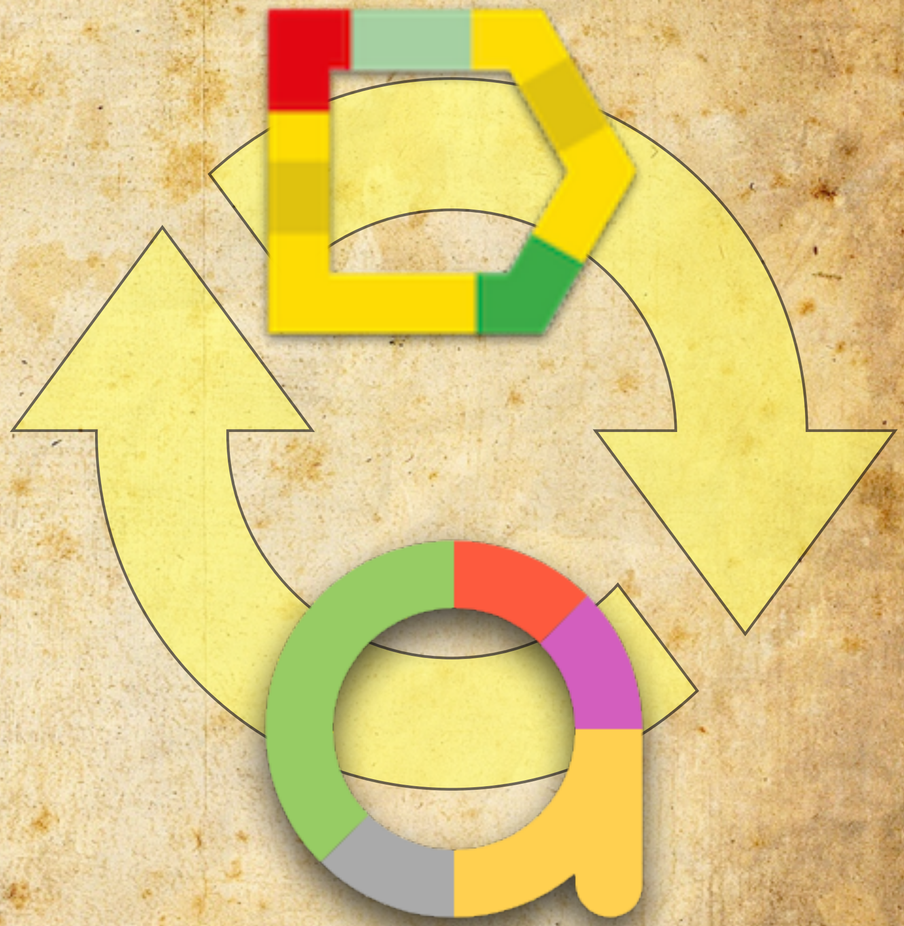
Ультимейт - аннотация Step

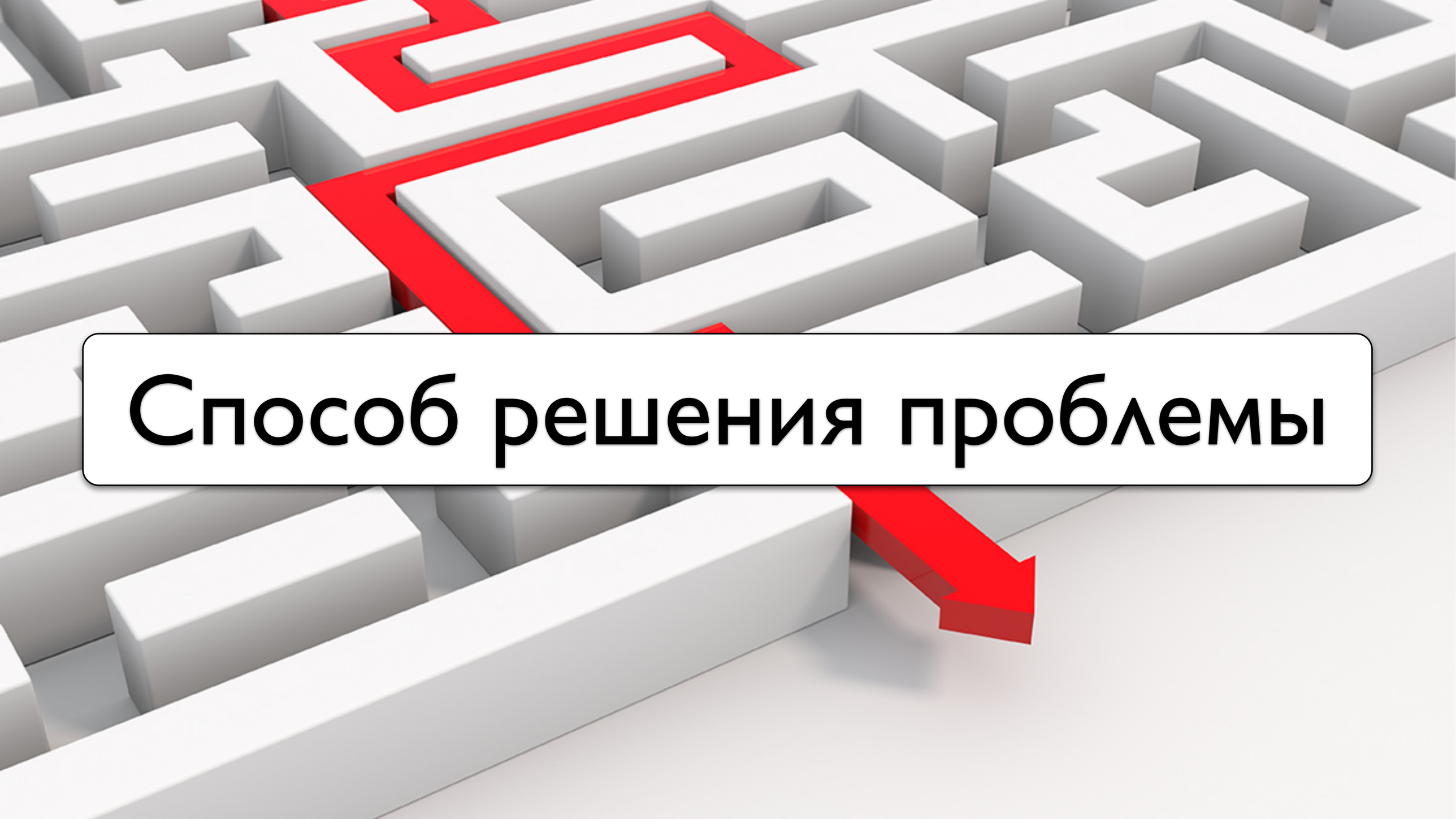
```
@Step("Login as user {0}")
public void loginAsUser(String username) {
   //step body here
}
```

```
@Step("Login as user {username}")
public void simpleTest(String username) {
   //test body here
}
```

Переезд на второй Allure

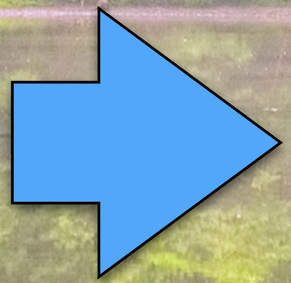





Способ решения проблемы

Будем мигрировать по-одному

@Name	@DisplayName
@Feature	@Features
@Story	@Stories
....	....
@Step	@Step



Создаем AllureMigrationAction

```
public void actionPerformed(AnActionEvent event) {
  PsiElement element = event.getData(PSI_ELEMENT);
  Arrays.stream(((PsiClass) element).getMethods())
    .filter(m -> m.hasAnnotation("org..Test"))
    .forEach(this::migrateTestAnnotations)
  }
  Arrays.stream(((PsiClass) element).getMethods())
    .filter(m -> m.hasAnnotation("org..Step"))
    .forEach(this::migrateStepAnnotation)
  }
}
```

Мигрируем тестовые методы

```
public void actionPerformed(AnActionEvent event) {
  PsiElement element = event.getData(PSI_ELEMENT);
  Arrays.stream(((PsiClass) element).getMethods())
    .filter(m -> m.hasAnnotation("org..Test"))
    .forEach(this::migrateTestAnnotations)
  }
  Arrays.stream(((PsiClass) element).getMethods())
    .filter(m -> m.hasAnnotation("org..Step"))
    .forEach(this::migrateStepAnnotation)
  }
}
```

Мигрируем по одной

```
public void migrateTestAnnotations(PsiMethod method) {

    migrateDisplayName(method);
    migrateDescription(method);
    migrateFeatures(method);
    migrateStories(method);
    migrateTmsIds(method);
    migrateIssues(method);
    migrateLins(method);

}
```

Мигрируем на @Features

```
public void migrateFeatures(PsiMethod method) {

  PsiAnnotation oldFeature =
    method.getAnnotation("ru..Feature");
  List<String> features = getValues(oldFeature, "value");
  String featuresText = getFeaturesText(features);
  PsiAnnotation features = createAnnotation(featureText);
  method.getModifiersList()
    .addAfter(features, oldFeatures);
  oldFeature.delete();
}
```

Тест с аннотацией @Feature

```
public class AddFavoritesAfterNoteTest {

   @Test
   @Feature({"Favorites", "Notes"})
   public void shouldAddToFavoriteAfterNodeTest() {
      ...
   }

}
```

Ищем аннотацию @Feature

```
public void migrateFeatures(PsiMethod method) {

  PsiAnnotation oldFeature =
    method.getAnnotation("ru..Feature");
  List<String> features = getValues(oldFeature, "value");
  String featuresText = getFeaturesText(features);
  PsiAnnotation features = createAnnotation(featureText);
  method.getModifiersList()
    .addAfter(features, oldFeatures);
  oldFeature.delete();
}
```

Взяли значение @Feature

```
public class AddFavoritesAfterNoteTest {

   @Test
   @Feature({"Favorites", "Notes"})
   public void shouldAddToFavoriteAfterNodeTest() {
      ...
   }

}
```

Создаем новую @Features

```
public void migrateFeatures(PsiMethod method) {

  PsiAnnotation oldFeature =
    method.getAnnotation("ru..Feature");
  List<String> features = getValues(oldFeature, "value");
  String featuresText = getFeaturesText(features);
  PsiAnnotation features = createAnnotation(featureText);
  method.getModifiersList()
    .addAfter(features, oldFeatures);
  oldFeature.delete();
}
```

Аннотация в виде строки

```
public String getInnerFeaturesText(List<String> features) {
  return features.stream()
    .map(f -> "@io..Feature(\"" + f + "\")")
    .collect(Collectors.joining(","))
}

public String getFeaturesText(List<String> features) {
  String inner = getInnerFeaturesText(features);
  return "@io..Features({" + inner + "})";
}
```

@io..Features({@io..Feature("Favorites"), ...})

Создаем новую @Features

```
public void migrateFeatures(PsiMethod method) {

  PsiAnnotation oldFeature =
    method.getAnnotation("ru..Feature");
  List<String> features = getValues(oldFeature, "value");
  String featuresText = getFeaturesText(features);
  PsiAnnotation features = createAnnotation(featureText);
  method.getModifiersList()
    .addAfter(features, oldFeatures);
  oldFeature.delete();
}
```

Добавляем ее к методу

```
public void migrateFeatures(PsiMethod method) {

  PsiAnnotation oldFeature =
    method.getAnnotation("ru..Feature");
  List<String> features = getValues(oldFeature, "value");
  String featuresText = getFeaturesText(features);
  PsiAnnotation features = createAnnotation(featureText);
  method.getModifiersList()
    .addAfter(features, oldFeatures);
  oldFeature.delete();
}
```

Добавилась новая аннотация

```
public class AddFavoritesAfterNoteTest {

   @Test
   @Feature({"Favorites", "Notes"})
   @Features({@Feature("Favorites"), @Feature("Notes")})
   public void shouldAddToFavoriteAfterNodeTest() {
      ...
   }

}
```

Удаляем старую @Feature

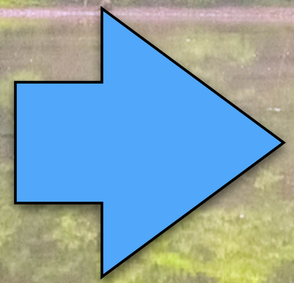
```
public void migrateFeatures(PsiMethod method) {

  PsiAnnotation oldFeature =
    method.getAnnotation("ru..Feature");
  List<String> features = getValues(oldFeature, "value");
  String featuresText = getFeaturesText(features);
  PsiAnnotation features = createAnnotation(featureText);
  method.getModifiersList()
    .addAfter(features, oldFeatures);
  oldFeature.delete();
}
```

Осталась только новая

```
public class AddFavoritesAfterNoteTest {

   @Test
   @Features({@Feature("Favorites"), @Feature("Notes")})
   public void shouldAddToFavoriteAfterNodeTest() {
      ...
   }

}
```

Остальные по аналогии

@Name	@DisplayName
@Feature	@Features
@Story	@Stories
....	....
@Step	@Step



Мигрируем степы тестов

```
public void actionPerformed(AnActionEvent event) {
  PsiElement element = event.getData(PSI_ELEMENT);
  Arrays.stream(((PsiClass) element).getMethods())
    .filter(m -> m.hasAnnotation("org..Test"))
    .forEach(this::migrateTestAnnotations)
  }
  Arrays.stream(((PsiClass) element).getMethods())
    .filter(m -> m.hasAnnotation("org..Step"))
    .forEach(this::migrateStepAnnotation)
  }
}
```

Мигрируем методы @Step

```
public void migrateStepAnnotation(PsiMethod method) {
  PsiAnnotation oldStep =
    method.getAnnotation("ru..Step");
  String oldstepText = getValue(oldStep, "value");
  String[] params = Arrays.stream(method.getParameters())
    .map(JvmParameter::getName).toArray(String[]::new);
  String newStepText = convert(oldStepText, params);
  PsiAnnotation newStep = createStep(newStepText);
  method.getModifiersList()
    .addAfter(newStep, oldStep);
  oldStep.delete();
}
```

Метод с аннотацией @Step

```
public class BasicSteps {

  @Step("Открываем страницу машины марки {0}")
  public void openAutoCardPage(String mark) { ... }

  @Step("Добавляем заметку {0} к машине")
  public void addNotesToAutoCard(String text) { ... }

}
```

Получаем аннотацию @Step

```
public void migrateStepAnnotation(PsiMethod method) {
  PsiAnnotation oldStep =
    method.getAnnotation("ru..Step");
  String oldstepText = getValue(oldStep, "value");
  String[] params = Arrays.stream(method.getParameters())
    .map(JvmParameter::getName).toArray(String[]::new);
  String newStepText = convert(oldStepText, params);
  PsiAnnotation newStep = createStep(newStepText);
  method.getModifiersList()
    .addAfter(newStep, oldStep);
  oldStep.delete();
}
```

Получили текст аннотации

```
public class BasicSteps {

  @Step("Открываем страницу машины марки {0}")
  public void openAutoCardPage(String mark) { ... }

  @Step("Добавляем заметку {0} к машине")
  public void addNotesToAutoCard(String text) { ... }

}
```

Получаем параметры метода

```
public void migrateStepAnnotation(PsiMethod method) {
  PsiAnnotation oldStep =
    method.getAnnotation("ru..Step");
  String oldstepText = getValue(oldStep, "value");
  String[] params = Arrays.stream(method.getParameters())
    .map(JvmParameter::getName).toArray(String[]::new);
  String newStepText = convert(oldStepText, params);
  PsiAnnotation newStep = createStep(newStepText);
  method.getModifiersList()
    .addAfter(newStep, oldStep);
  oldStep.delete();
}
```

Получили параметры метода

```
public class BasicSteps {

  @Step("Открываем страницу машины марки {0}")
  public void openAutoCardPage(String mark) { ... }

  @Step("Добавляем заметку {0} к машине")
  public void addNotesToAutoCard(String text) { ... }

}
```

Конвертируем текст степа

```
public void migrateStepAnnotation(PsiMethod method) {
  PsiAnnotation oldStep =
    method.getAnnotation("ru..Step");
  String oldstepText = getValue(oldStep, "value");
  String[] params = Arrays.stream(method.getParameters())
    .map(JvmParameter::getName).toArray(String[]::new);
  String newStepText = convert(oldStepText, params);
  PsiAnnotation newStep = createStep(newStepText);
  method.getModifiersList()
    .addAfter(newStep, oldStep);
  oldStep.delete();
}
```

Добавляем новый @Step

```
public void migrateStepAnnotation(PsiMethod method) {
  PsiAnnotation oldStep =
    method.getAnnotation("ru..Step");
  String oldstepText = getValue(oldStep, "value");
  String[] params = Arrays.stream(method.getParameters())
    .map(JvmParameter::getName).toArray(String[]::new);
  String newStepText = convert(oldStepText, params);
  PsiAnnotation newStep = createStep(newStepText);
  method.getModifiersList()
    .addAfter(newStep, oldStep);
  oldStep.delete();
}
```

Добавили новый @Step

```
public class BasicSteps {

  @Step("Открываем страницу машины марки {0}")
  @Step("Открываем страницу машины марки {mark}")
  public void openAutoCardPage(String mark) { ... }

  @Step("Добавляем заметку {0} к машине")
  @Step("Добавляем заметку {text} к машине")
  public void addNotesToAutoCard(String text) { ... }

}
```

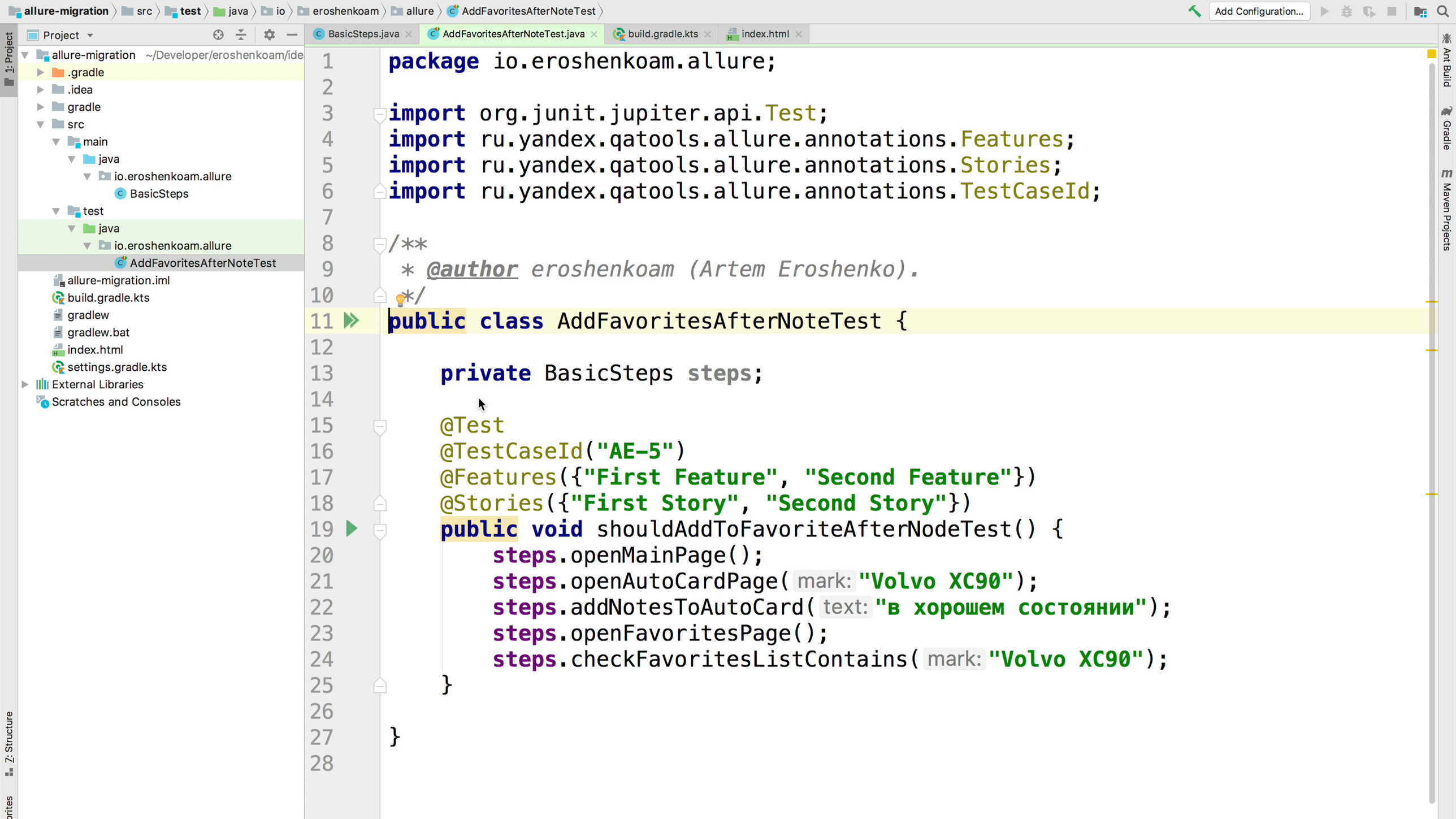
Удаляем старый @Step

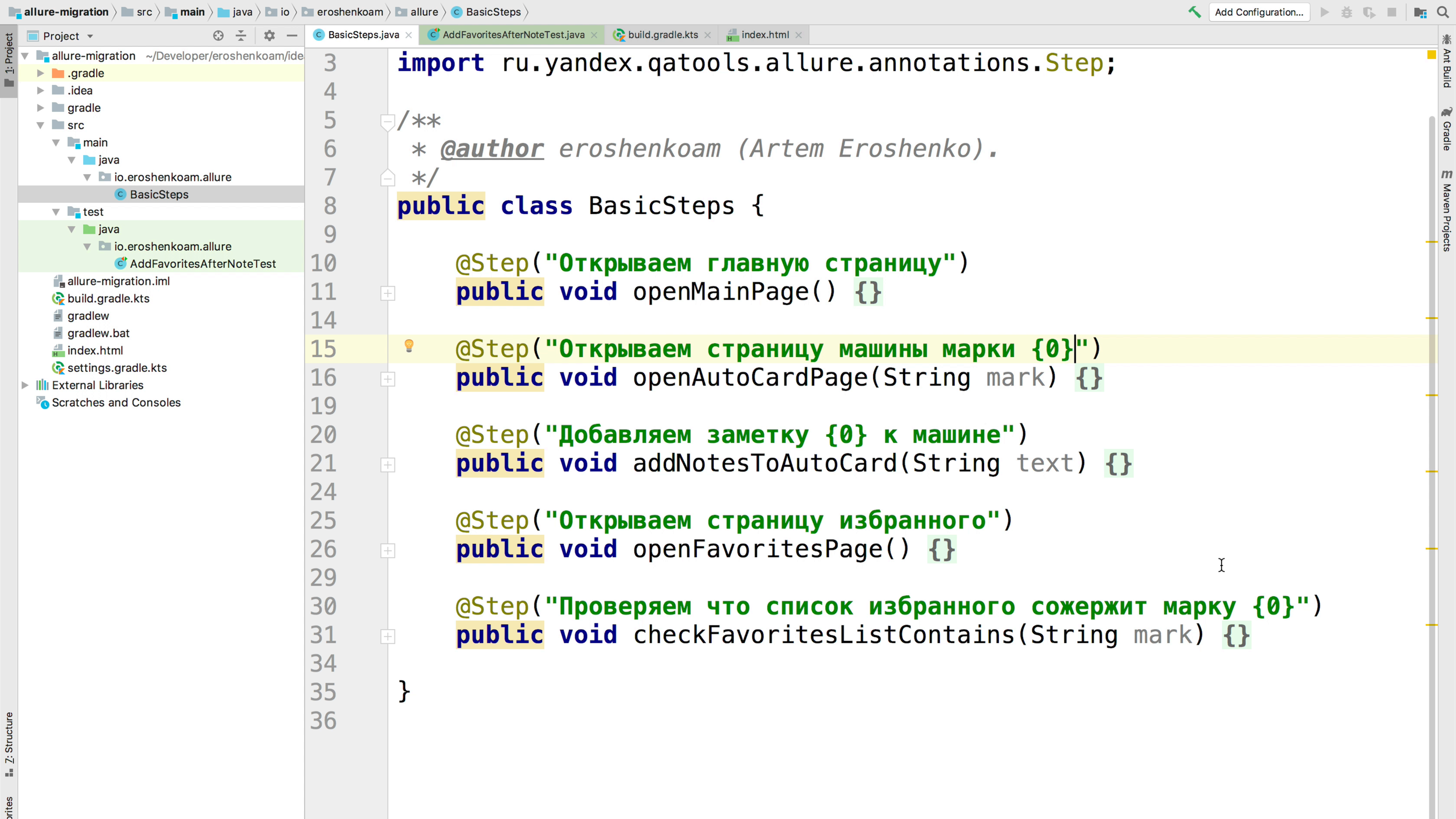
```
public void migrateStepAnnotation(PsiMethod method) {
  PsiAnnotation oldStep =
    method.getAnnotation("ru..Step");
  String oldstepText = getValue(oldStep, "value");
  String[] params = Arrays.stream(method.getParameters())
    .map(JvmParameter::getName).toArray(String[]::new);
  String newStepText = convert(oldStepText, params);
  PsiAnnotation newStep = createStep(newStepText);
  method.getModifiersList()
    .addAfter(newStep, oldStep);
  oldStep.delete();
}
```

Удалили старый @Step

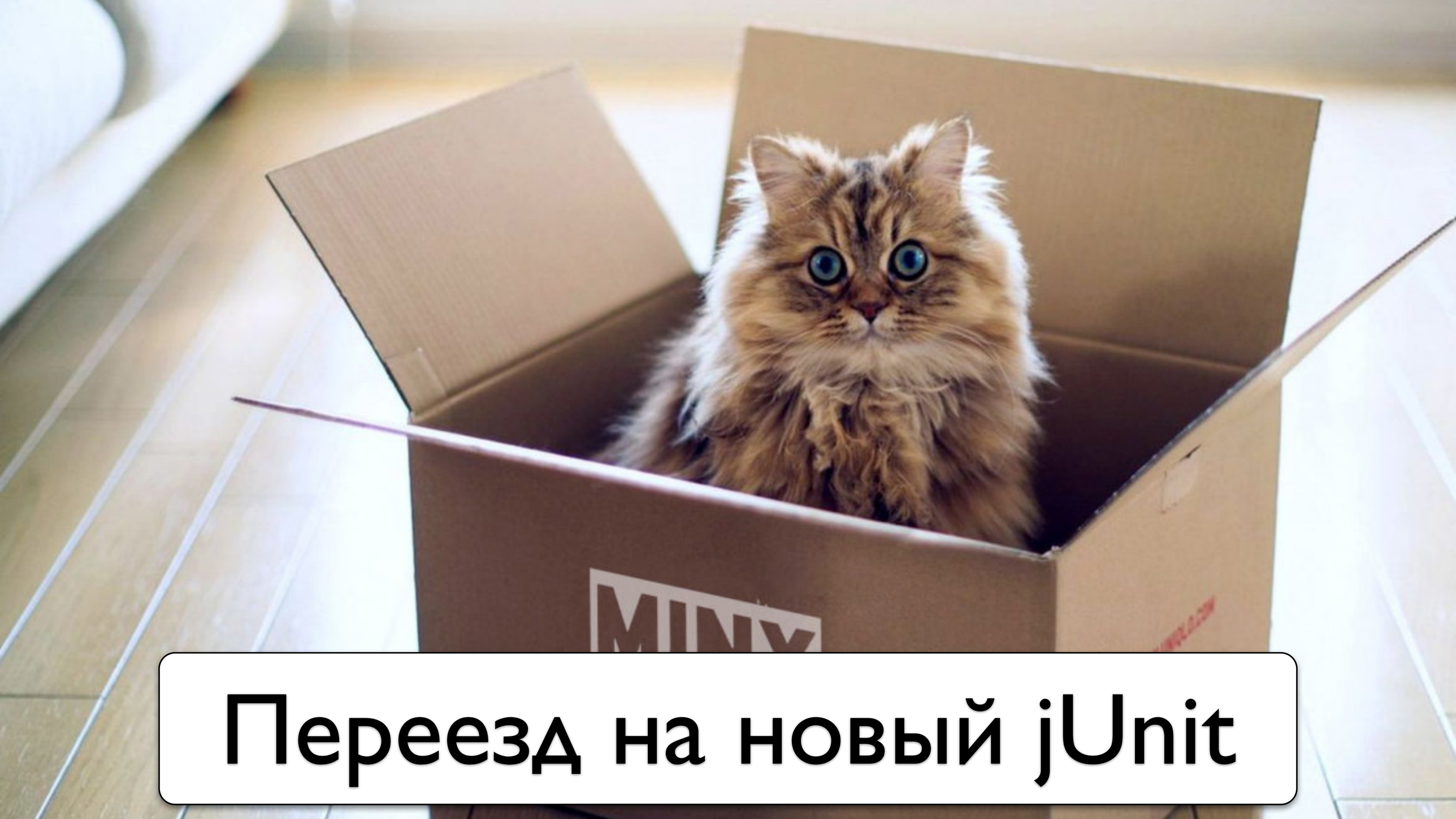
```
public class BasicSteps {

  @Step("Открываем страницу машины марки {mark}")
  public void openAutoCardPage(String mark) { ... }

  @Step("Добавляем заметку {text} к машине")
  public void addNotesToAutoCard(String text) { ... }

}
```


Web 3.0


allure-migration
src
test
java
io
eroshenkoam
allure
AddFavoritesAfterNoteTest
Add Configuration...
Project
BasicSteps.java
AddFavoritesAfterNoteTest.java
build.gradle.kts
index.html
allure-migration ~/Developer/eroshenkoam/ide
.gradle
.idea
gradle
src
main
java
io.eroshenkoam.allure
BasicSteps
test
java
io.eroshenkoam.allure
AddFavoritesAfterNoteTest
allure-migration.iml
build.gradle.kts
gradlew
gradlew.bat
index.html
settings.gradle.kts
External Libraries
Scratches and Consoles
1: Project
Z: Structure
Ant Build
Gradle
Maven Projects
package io.eroshenkoam.allure;
import org.junit.jupiter.api.Test;
import ru.yandex.qatools.allure.annotations.Features;
import ru.yandex.qatools.allure.annotations.Stories;
import ru.yandex.qatools.allure.annotations.TestCaseId;
/**
* @author eroshenkoam (Artem Eroshenko).
*/
public class AddFavoritesAfterNoteTest {
private BasicSteps steps;
@Test
@TestCaseId("AE-5")
@Features({"First Feature", "Second Feature"})
@Stories({"First Story", "Second Story"})
public void shouldAddToFavoriteAfterNodeTest() {
steps.openMainPage();
steps.openAutoCardPage(mark: "Volvo XC90");
steps.addNotesToAutoCard(text: "в хорошем состоянии");
steps.openFavoritesPage();
steps.checkFavoritesListContains(mark: "Volvo XC90");
}
}

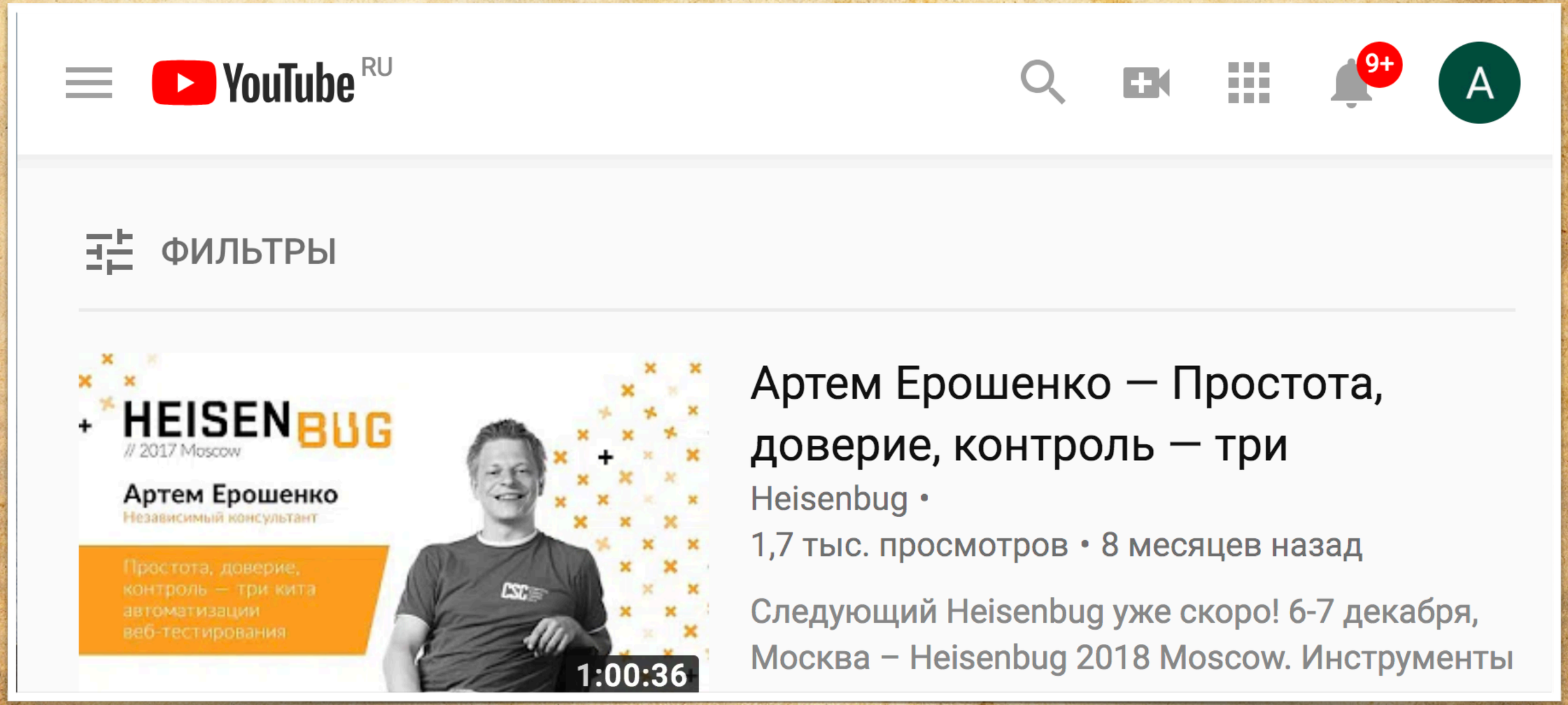

allure-migration
src
main
java
io
eroshenkoam
allure
BasicSteps
Add Configuration...
Project
BasicSteps.java
AddFavoritesAfterNoteTest.java
build.gradle.kts
index.html
allure-migration ~/Developer/eroshenkoam/ide
.gradle
.idea
gradle
src
main
java
io.eroshenkoam.allure
BasicSteps
test
java
io.eroshenkoam.allure
AddFavoritesAfterNoteTest
allure-migration.iml
build.gradle.kts
gradlew
gradlew.bat
index.html
settings.gradle.kts
External Libraries
Scratches and Consoles
import ru.yandex.qatools.allure.annotations.Step;
/**
* @author eroshenkoam (Artem Eroshenko).
*/
public class BasicSteps {
@Step("Открываем главную страницу")
public void openMainPage() {}
@Step("Открываем страницу машины марки {0}")
public void openAutoCardPage(String mark) {}
@Step("Добавляем заметку {0} к машине")
public void addNotesToAutoCard(String text) {}
@Step("Открываем страницу избранного")
public void openFavoritesPage() {}
@Step("Проверяем что список избранного сожержит марку {0}")
public void checkFavoritesListContains(String mark) {}
}
1: Project
Ant Build
Gradle
Maven Projects
Z: Structure

Какие еще есть идеи?



Переезд на новый jUnit

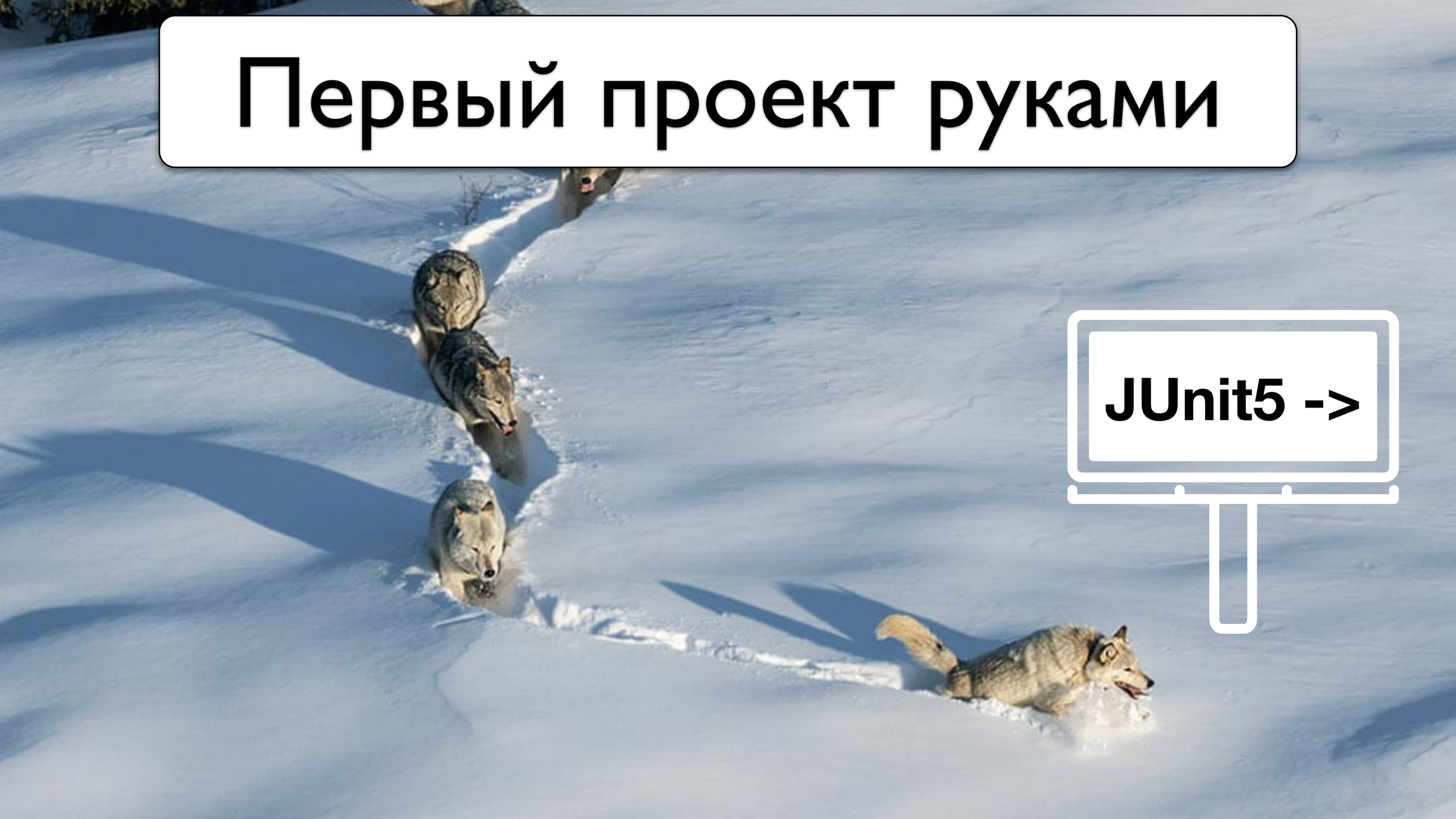
Guice - Dependency Injection



Как выглядит тест на jUnit4

```
@RunWith(GuiceRunner.class)
@GuiceModules(WebModule.class)
public class JUnit4Test {
  @Rule
  @Inject
  private RuleChain rules; //WebDriverRule, ScreenshotOnFail

  @Inject
  private BaseSteps steps;

  @Test
  public void shouldAddToFavoriteAfterNodeTest() { ... }
}
```

Первый проект руками



Как выглядит тест на jUnit5

```
@ExtendWith(GuiceExtension.class)
@IncludeModules(WebModule.class)
@ExtendWith({
  WebDriverExtension.class,
  ScrenshotOnFail.class
})
public class JUnit4Test {
  @Inject
  private BaseSteps steps;

  @Test
  public void shouldAddToFavoriteAfterNodeTest() { ... }
}
```

Заменили Runner на Extension

```
@ExtendWith(GuiceExtension.class)//@RunWith(GuiceRunner.class)
@IncludeModules(WebModule.class)
@ExtendWith({
  WebDriverExtension.class,
  ScrenshotOnFail.class
})
public class JUnit4Test {
  @Inject
  private BaseSteps steps;

  @Test
  public void shouldAddToFavoriteAfterNodeTest() { ... }
}
```

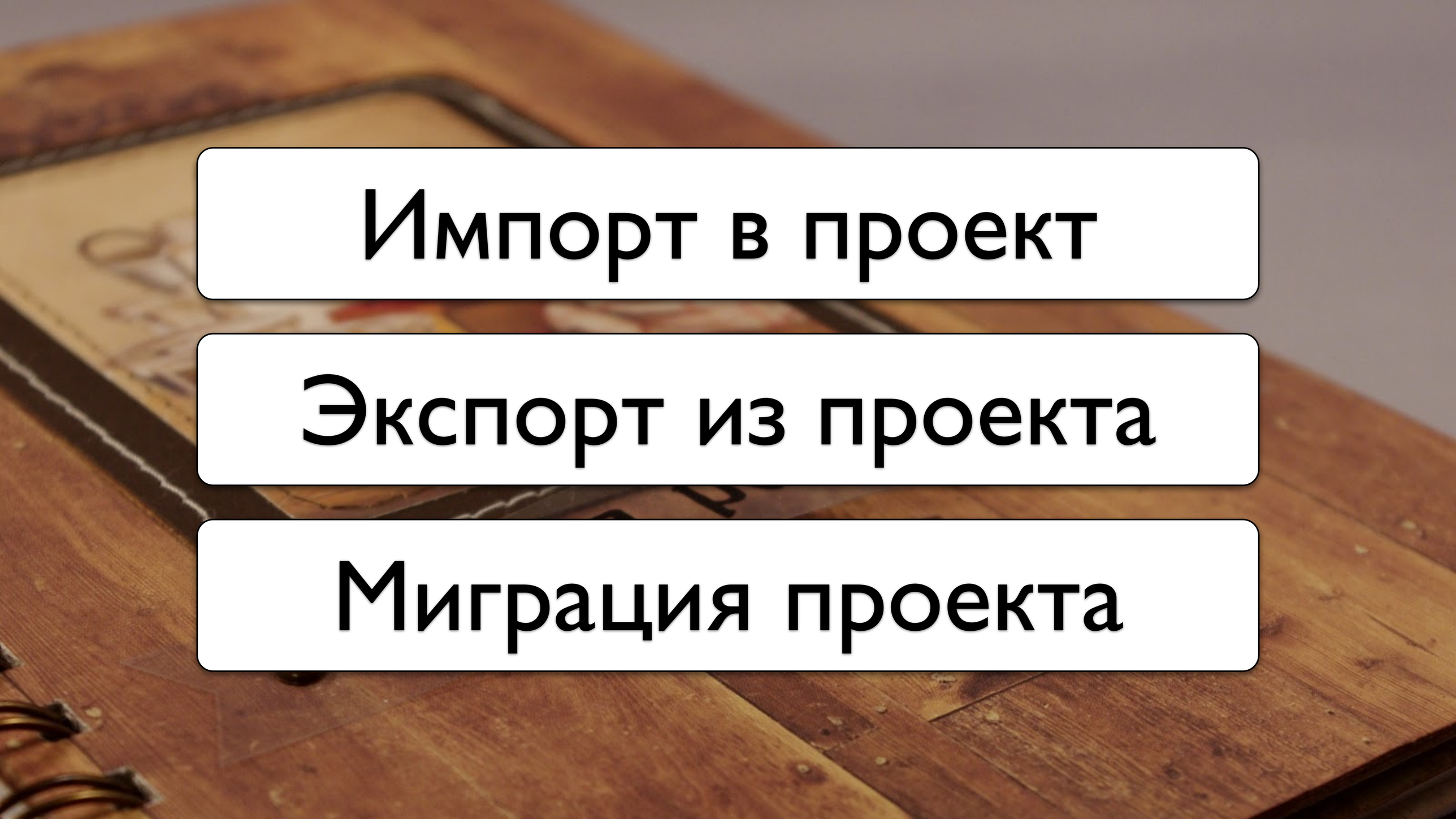
Другая аннотация для модулей

```
@ExtendWith(GuiceExtension.class)
@IncludeModules(WebModule.class)//@GuiceModules(WebModule.class)
@ExtendWith({
  WebDriverExtension.class,
  ScrenshotOnFail.class
})
public class JUnit4Test {
  @Inject
  private BaseSteps steps;

  @Test
  public void shouldAddToFavoriteAfterNodeTest() { ... }
}
```

Заменили Rule на Extension

```
@ExtendWith(GuiceExtension.class)
@IncludeModules(WebModule.class)
@ExtendWith({
  WebDriverExtension.class,
  ScrenshotOnFail.class
})
public class JUnit4Test {
  @Inject
  private BaseSteps steps;

  @Test
  public void shouldAddToFavoriteAfterNodeTest() { ... }
}
```

Домашнее задание


Импорт в проект
Экспорт из проекта
Миграция проекта


Вершина айсберга

Совершенствовать орудие труда


Собери свою коллекцию
deviantvicky.dev


Спасибо, Артем
Вот так я и стал автоматизатором

Артем Ерошенко

@eroshenkoam

https://github.com/eroshenkoam/idea-plugin-sample


Вопросы?