

Записки кодревьюера:

мыслим выше,
чем пробелы и табуляция



бизнес
ВКонтакте



Александра
Качина

*ВКонтакте для бизнеса –
ваш гид в мир бизнеса
в крупнейшей российской
социальной сети*

Стек

- Java
- Kotlin
- Swift
- Selenide
- Espresso
- Gradle
- TeamCity
- Allure
- TestOps
- Xcode
- XCUI Test
- Android Studio
- Charles Postman
- Swagger
- TestNG
- AssertJ



План доклада

Зачем моему коду нужен код-ревью, и почему я должен писать комментарии в чужом коде? 4

Что мы можем делать и автоматизировать до отправки кода на merge request, чтобы минимизировать замечания на код-ревью? 5

Что мы можем автоматизировать в процессе код-ревью, чтобы облегчить друг другу рабочую жизнь? 18

Общие рекомендации по проведению ревью: как проводить, как отвечать, типовые ситуации 42



Зачем нужен код-ревью для моего кода, и почему я должен писать комментарии в чужом коде?

- 1** Улучшить наше решение
- 2** Найти ошибки в коде
- 3** Поделиться собственным опытом
- 4** Научиться новым подходам

Дельное замечание...
какой же ревьюер
умный



*подсмотрел решение
у другого ревьюера*

*Что можно
делать до
отправки кода на
merge request?*



Используем возможности среды разработки

- Реформатор кода
- Пресеты
- Внешние утилиты
- Хуки
- Ошибки и предупреждения

Проблема: ОДИНАКОВЫЕ КОММЕНТАРИИ О СТИЛЕ КОДА

```
@Test(description = "Проверка открытия формы логина с главной страницы при нажатии на 'Войти'", testName = "Открытие формы логина с главной")
```

В Code Style написано, что после и до скобок должен быть перенос строки

```
@Test(  
    description = "Проверка открытия формы логина с главной страницы  
при нажатии на 'Войти'",  
    testName = "Открытие формы логина с главной"  
)
```

Исправь отступы, у нас в проекте используют 4 пробела

Используем Reformat Code

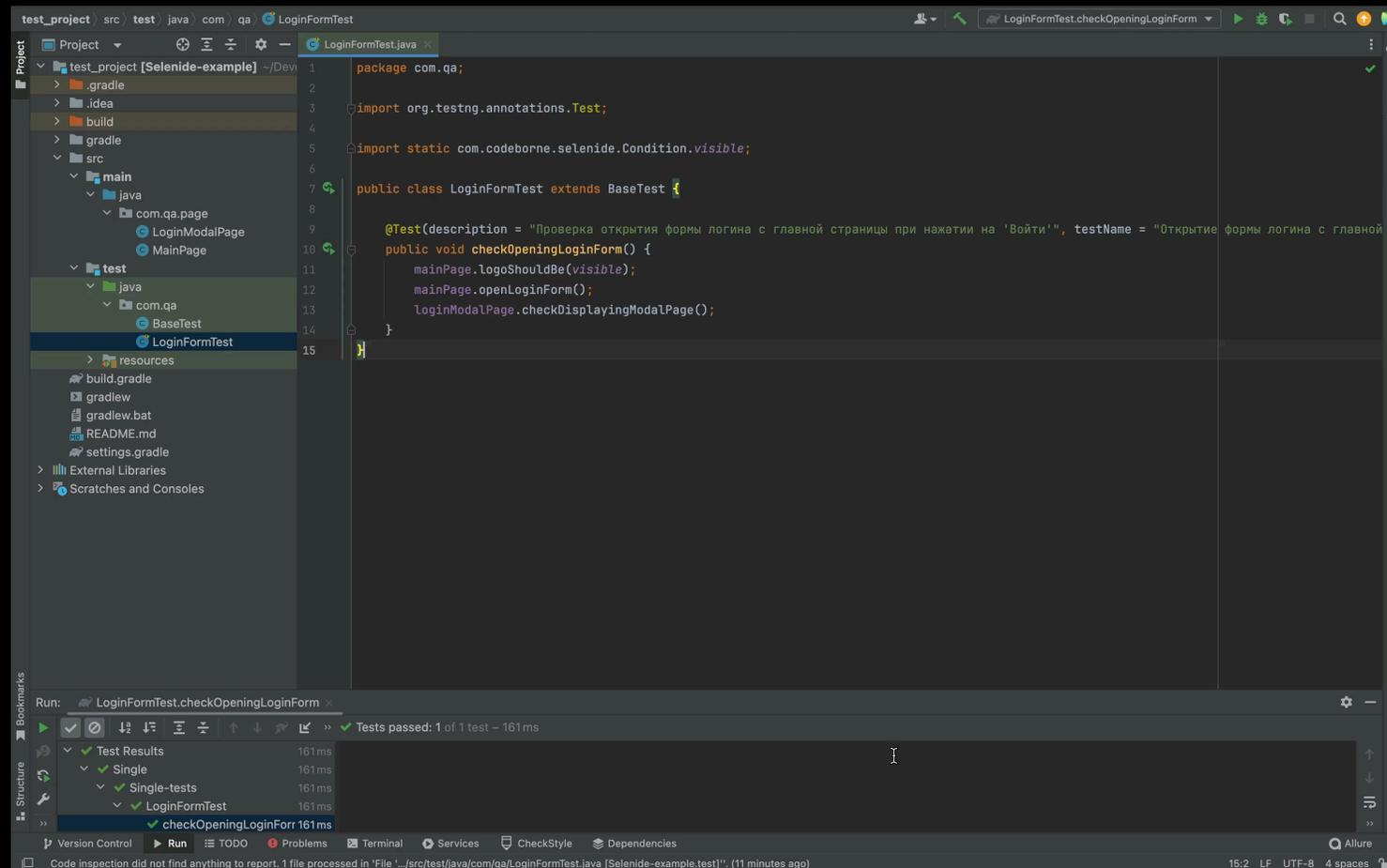
1. Code Style Settings: создаём
кастомные настройки и
импортируем

Использование:

Code → Reformat code

Настройка:

Settings → Editor → Code Style



The screenshot shows an IDE window with a Java file named `LoginFormTest.java`. The code is as follows:

```
1 package com.qa;
2
3 import org.testng.annotations.Test;
4
5 import static com.codeborne.selenide.Condition.visible;
6
7 public class LoginFormTest extends BaseTest {
8
9     @Test(description = "Проверка открытия формы логина с главной страницы при нажатии на 'Войти'", testName = "Открытие формы логина с главной
10     public void checkOpeningLoginForm() {
11         mainPage.logoShouldBe(visible);
12         mainPage.openLoginForm();
13         loginModalPage.checkDisplayingModalPage();
14     }
15 }
```

Below the editor, the Run window shows the test results:

```
Run: LoginFormTest.checkOpeningLoginForm
Tests passed: 1 of 1 test - 161ms
Test Results
  Single - 161ms
  Single-tests - 161ms
  LoginFormTest - 161ms
    checkOpeningLoginForr - 161ms
```

The status bar at the bottom indicates: 15:2 L F UTF-8 4 spaces.

Используем Reformat Code

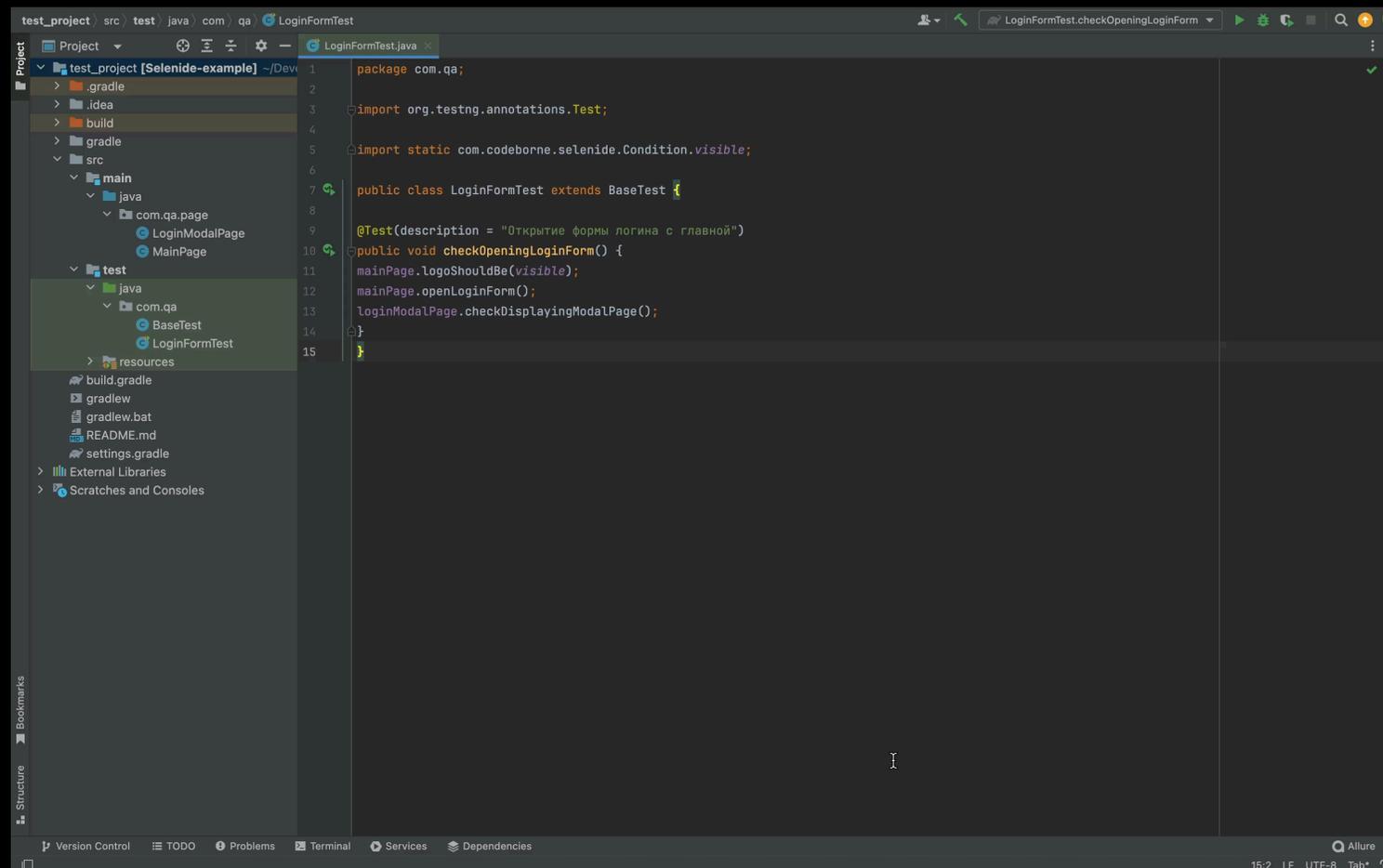
2. Используем в проекте
файл .editorconfig

Настройка:

Settings → Editor → Code Style

Scheme: Default - стандартное
форматирование

Enable EditorConfig support



```
1 package com.qa;
2
3 import org.testng.annotations.Test;
4
5 import static com.codeborne.selenide.Condition.visible;
6
7 public class LoginFormTest extends BaseTest {
8
9     @Test(description = "Открытие формы логина с главной")
10    public void checkOpeningLoginForm() {
11        mainPage.logoShouldBe(visible);
12        mainPage.openLoginForm();
13        loginModalPage.checkDisplayingModalPage();
14    }
15 }
```

Используем Reformat Code

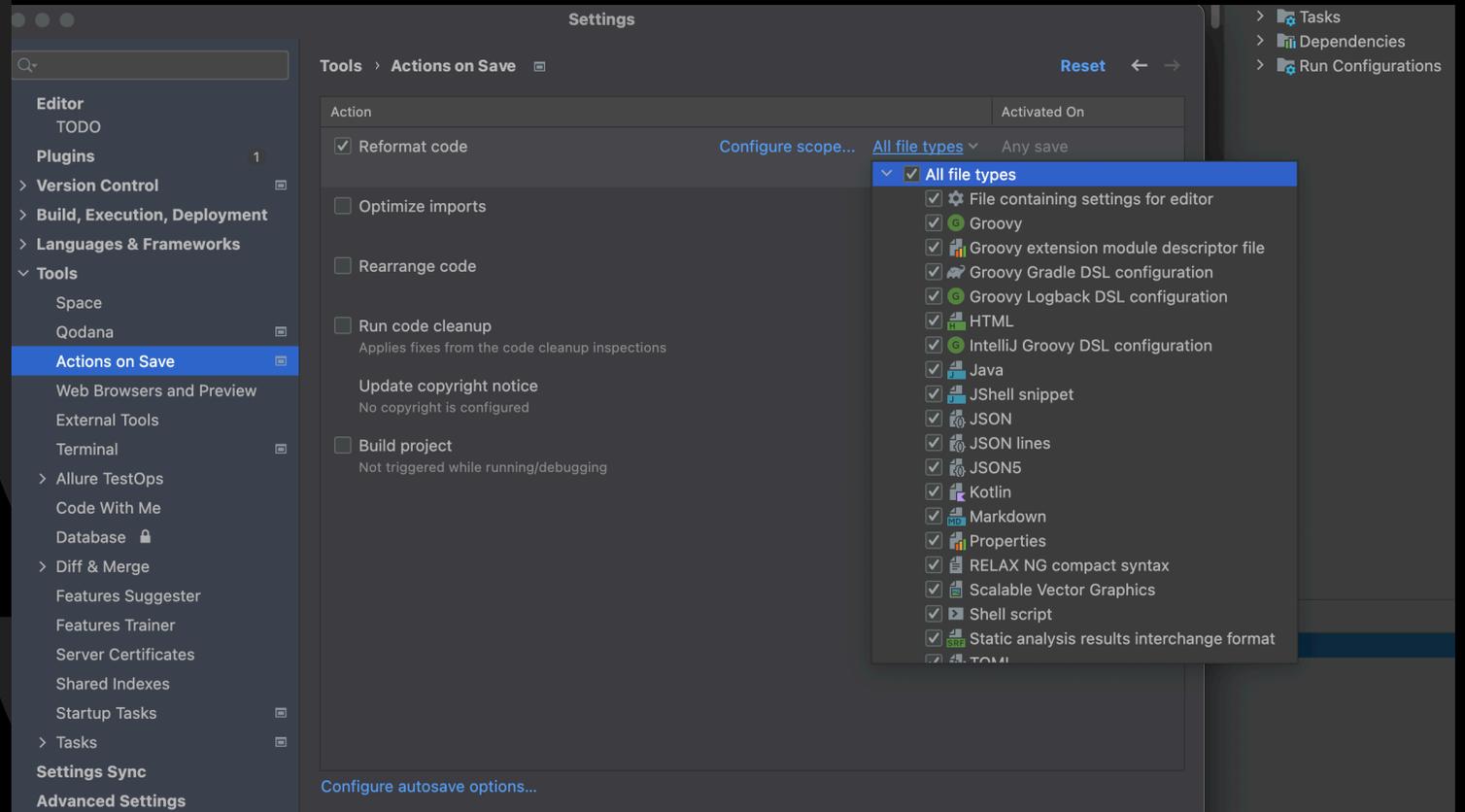
3. Автоматизируем
переформатирование кода при
сохранении изменённых файлов

Настройка:

Settings → Tools → Actions on Save

✓ Reformat code

- Configure scope
- All file types
- Whole file
- Changed lines



Внешние утилиты

Инструмент для изменения физического вида кода: добавить отступы, пробелы, переносы строк и т.д

Примеры:

- Prettier — JavaScript
- Clang-Format — C, C++, Objective-C
- Black — Python
- Gofmt — Go

Настройка

- 1** Установка утилиты с помощью систем управления пакетами
- 2** Настройка файла конфигурации в корне проекта / Интеграция с IDE через плагин / Настройка автоматических проверок / Интеграция с Git

В отличие от линтеров они не проверяют код на ошибки, которые потенциально способны причинить ущерб работоспособности

Git Hooks

Скрипты, которые Git выполняет перед или после того, как определенные важные действия произведены, например, commit, push или pull



Клиентские хуки:

- 1** pre-commit — проверка соответствия кода стандартам, запуск тестов, автоматическое форматирование
- 2** commit-msg — проверка содержания и формата коммит-сообщения
- 3** post-commit — уведомление о коммите

Серверные хуки:

- 1** pre-receive — проверка и изменение входящих коммитов
- 2** update — для вызова на каждый пуш ветки
- 3** post-receive — для развертывания кода и отправки уведомлений

1 Каталог `.git/hooks` с файлами расширения `.sample`

2 Создать файл без расширения и сделать его исполняемым:

```
touch pre-commit # Создает файл хука pre-commit
chmod +x pre-commit # Делает файл исполняемым
```

3 Создание скрипта

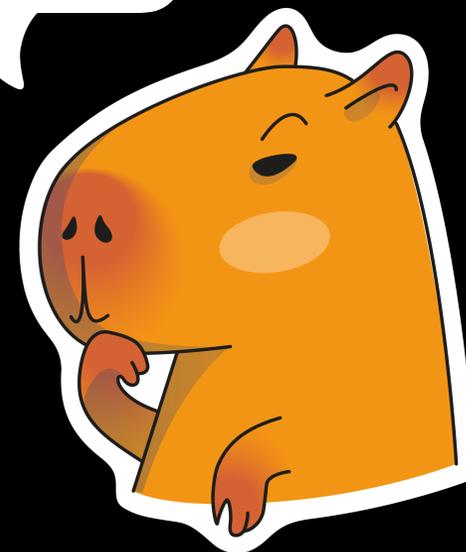
```
#!/bin/sh
# pre-commit
# Запуск линтера или утилиты для проверки стиля кодирования
bad_code_style=$(lint-command)

# Если линтер нашёл ошибки, отменяем коммит
if [ ! -z "$bad_code_style" ]; then
echo "Код не соответствует стилю кодирования. Пожалуйста, исправьте ошибки перед коммитом."
exit 1
fi
```

Проблема: комментарии об отсутствии однотипных параметров

```
@Owner(КАРИБАГА)  
@Severity(NORMAL)  
@AllureId("111111")  
@Test(description = "Проверка открытия формы логина с главной")
```

Кажется, ты забыл указать аннотацию с меткой @SprinterLabels(SMB). Она нужна, чтобы размечать тесты нашего бизнес-юнита

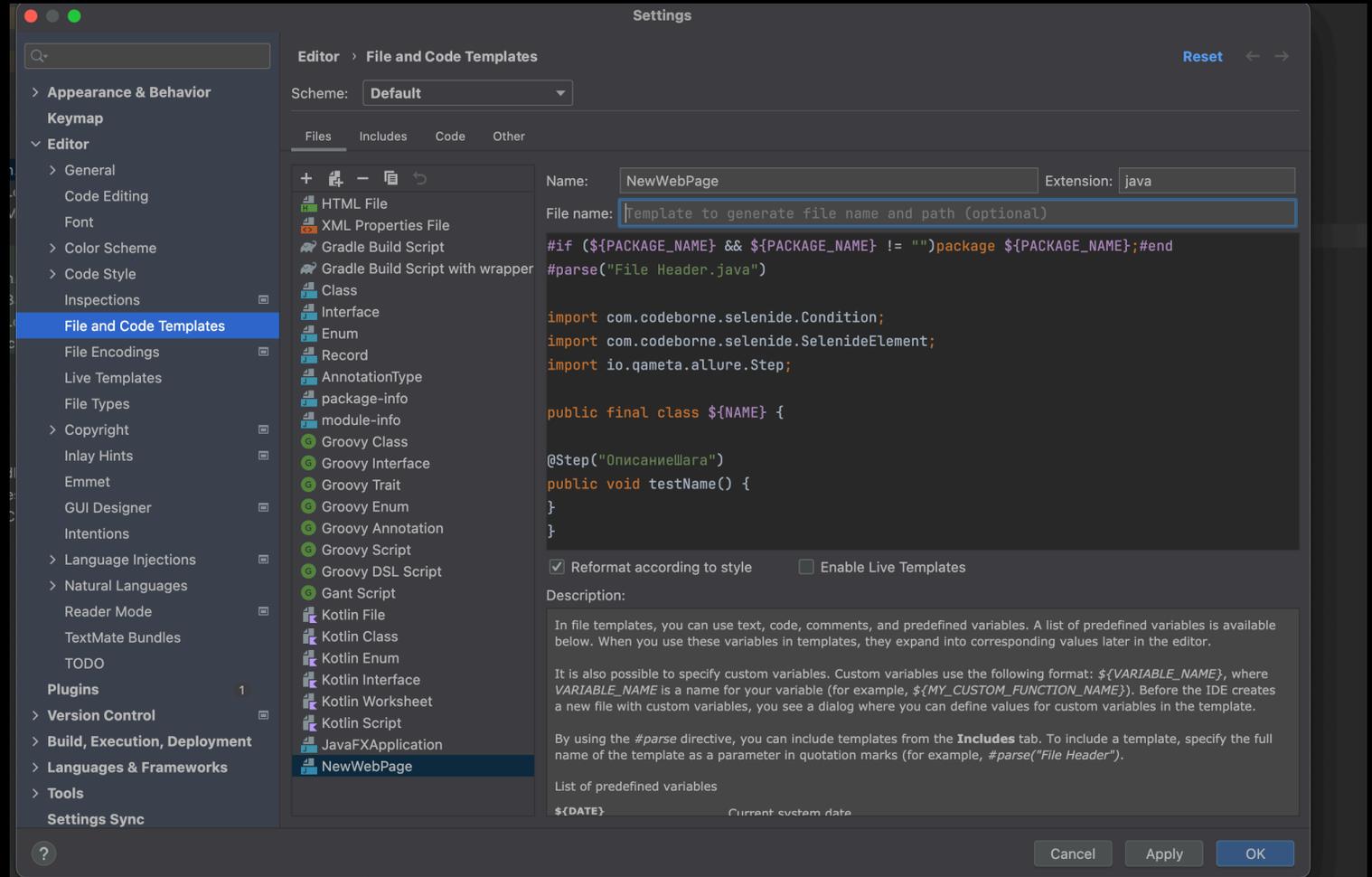


Пресеты

Шаблоны для создания
новых файлов

Настройка:

Settings → Editor →
File and Code Templates

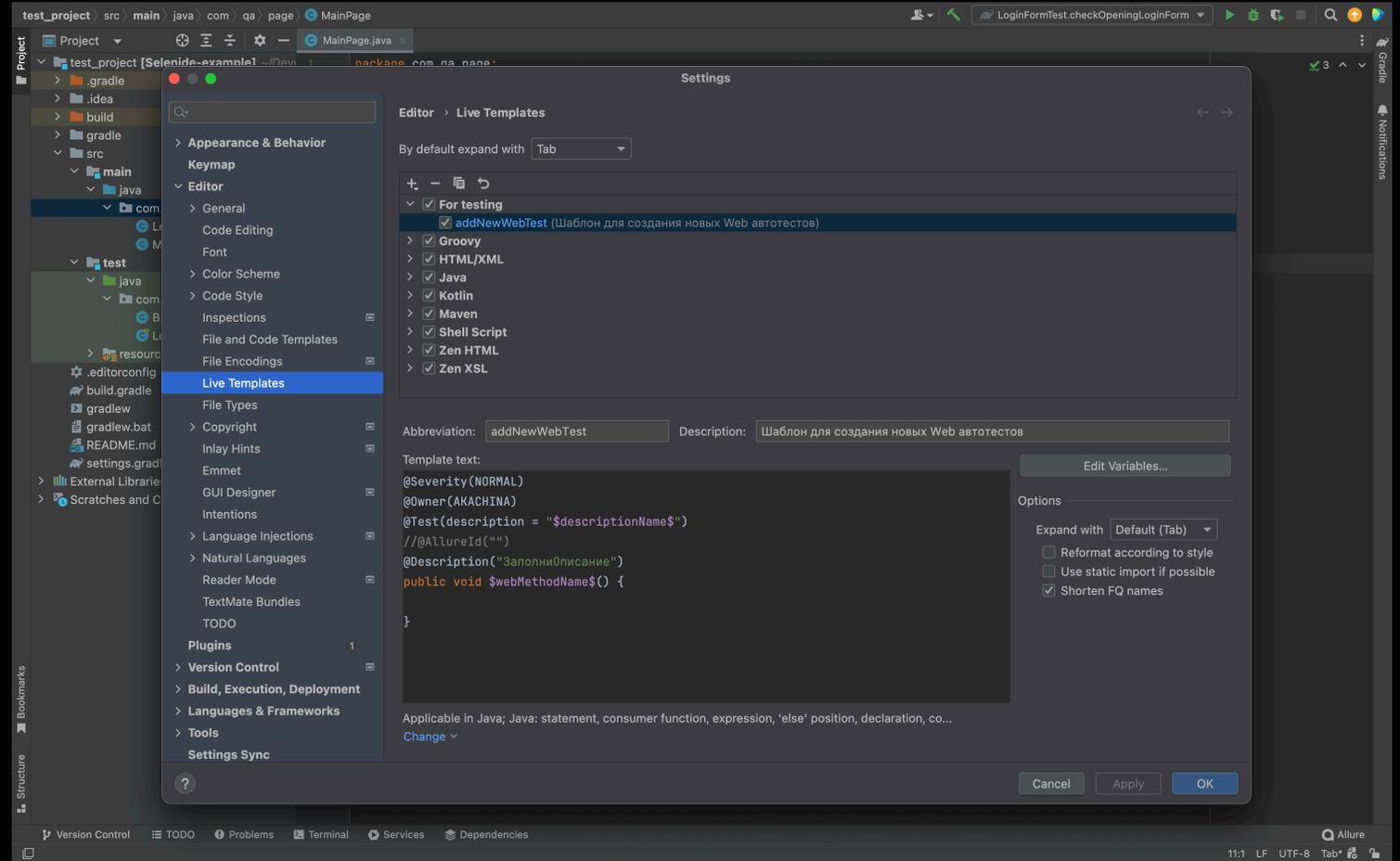


Пресеты

Шаблоны для создания
одинаковых блоков кода

Настройка:

Settings → Editor
→ Live Templates



Проблема:
ревьюеры
повторяют
замечания
среды
разработки

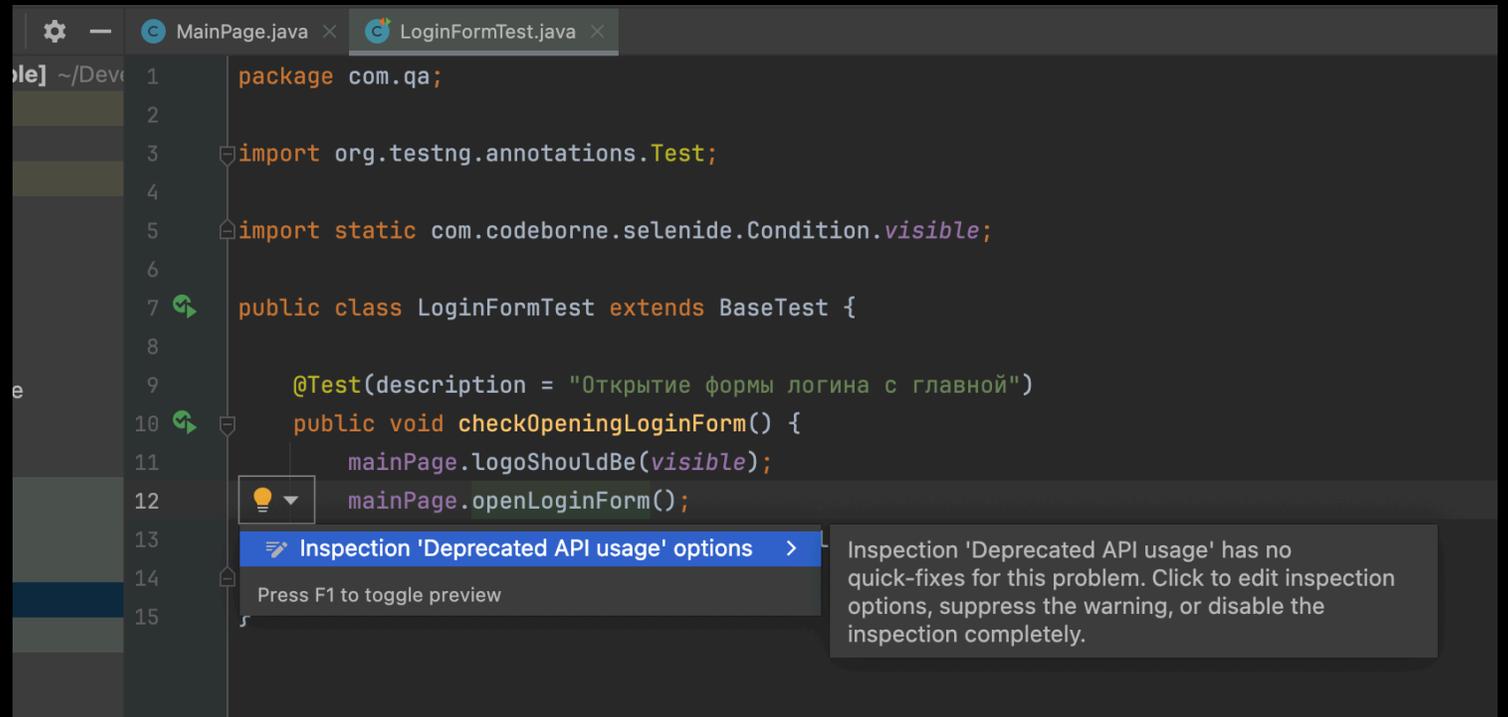
```
public void checkOpeningLoginForm() {  
    mainPage.logoShouldBe(visible);  
    mainPage.openLoginForm();  
}
```

Метод openLoginForm() не нужно использовать,
он устарел. Замени его на showLoginForm()



Errors Warnings

- Использование устаревших конструкций
- Необработанные исключения
- Потенциально неправильное использование логики программы



```
1 package com.qa;
2
3 import org.testng.annotations.Test;
4
5 import static com.codeborne.selenide.Condition.visible;
6
7 public class LoginFormTest extends BaseTest {
8
9     @Test(description = "Открытие формы логина с главной")
10    public void checkOpeningLoginForm() {
11        mainPage.logoShouldBe(visible);
12        mainPage.openLoginForm();
13    }
14
15 }
```

Inspection 'Deprecated API usage' options > Inspection 'Deprecated API usage' has no quick-fixes for this problem. Click to edit inspection options, suppress the warning, or disable the inspection completely.

Press F1 to toggle preview

*Что можно
делать во
время ревью
merge
request'a?*

- 1 Линтеры
- 2 Боты
- 3 Ревью в IDE
- 4 Функции комментирования



**Вроде всё
автоматизировал**

Проблема:
авторы кода
забывают
проверять
код-стайл.
Ревьюеры пишут
одинаковые
комментарии

```
public void checkOpeningLoginForm() {  
    mainPage.logoShouldBe( visible );  
    mainPage.openLoginForm();  
}
```

Здесь снова проблема с отступами :(



Линтеры



Инструмент для анализа кода, который помогает обнаруживать ошибки, несоответствия стилевым стандартам и потенциальные уязвимости для повышения качества и безопасности исходного кода

Примеры:

- ESLint - JavaScript
- RuboCop - Ruby
- Pylint - Python
- Checkstyle - Java

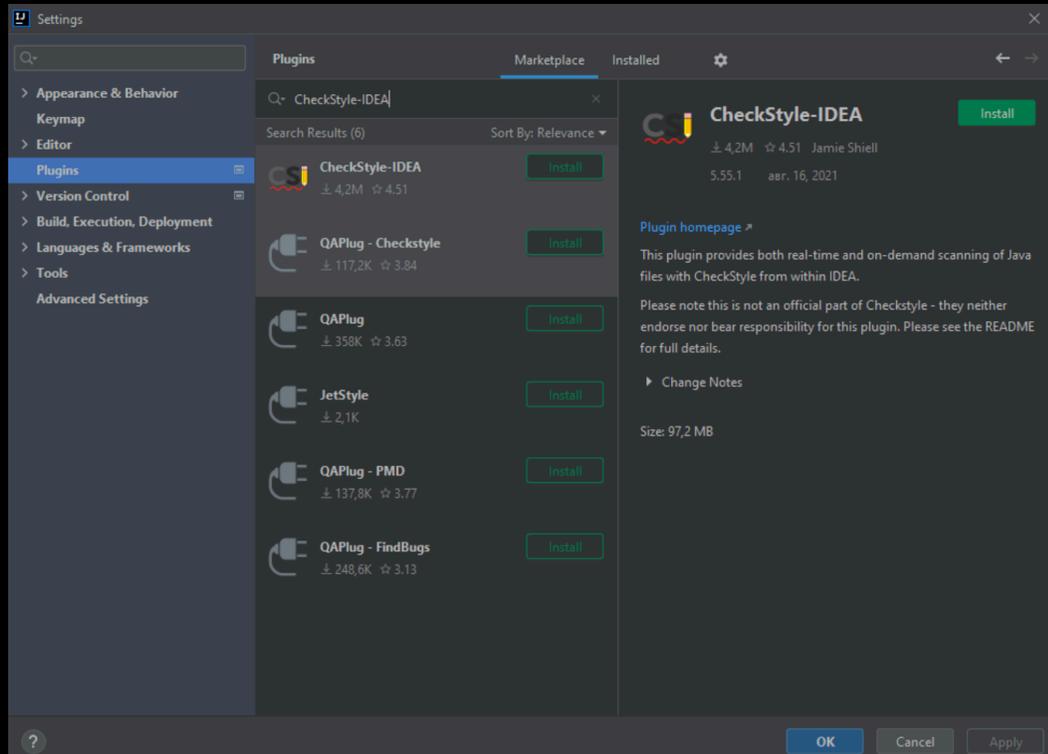
Линтеры

```
<configuration>
  <module name="Checker">
    <module name="TreeWalker">
      <module name="Indentation">
        <property name="basicOffset" value="4" />
        <property name="braceAdjustment" value="0" />
        <property name="caseIndent" value="4" />
      </module>
      <module name="MethodName">
        <property name="format" value="^[a-z][a-zA-Z0-9]*$" />
      </module>
      ...
    </module>
  </module>
</configuration>
```

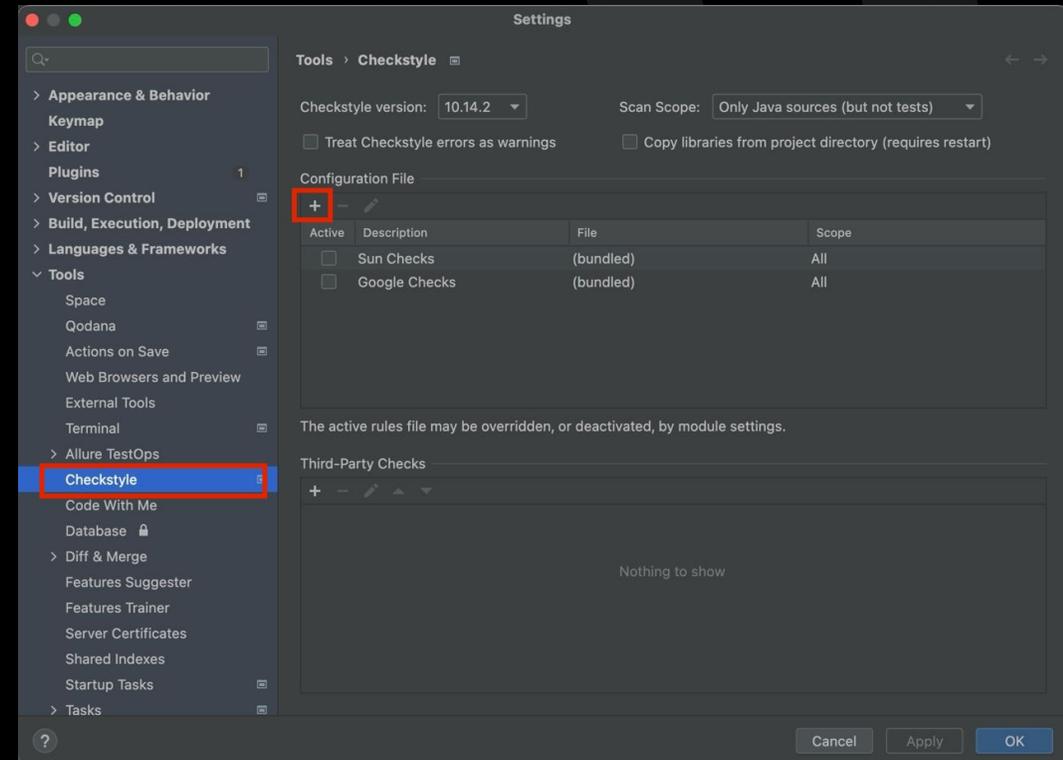
Запуск из командной строки

```
java -jar <path_to_checkstyle_jar> -c <path_to_checkstyle_xml> <path_to_source_files>
```

Линтеры



Настройка плагина:
Settings → Plugins



Конфигурация для своего проекта:
Settings → Other Settings → Checkstyle

1 Подготовить файл конфигурации checkstyle.xml

2 Добавить плагин Checkstyle в build.gradle

```
plugins {  
    // Применить плагин Checkstyle к проекту  
    id 'checkstyle'
```

3 Настроить Checkstyle в build.gradle

```
checkstyle {  
    toolVersion = '8.29' // Указывает версию Checkstyle  
    configFile = file("${project.rootDir}/config/checkstyle/checkstyle.xml") // Путь к файлу  
    конфигурации  
    ignoreFailures = false // Определяет, должен ли сборка продолжаться, если обнаружены ошибка  
    стиля  
    sourceSets = [sourceSets.main] // Директории или файлы для проверки  
}
```

4 Запустить Checkstyle

```
./gradlew checkstyleMain // Проверить основной исходный код в src/main/java  
./gradlew checkstyleTest // Проверить код тестов в src/test/java
```

Отчёты в формате XML и HTML в `${projectDir}/build/reports/checkstyle/`

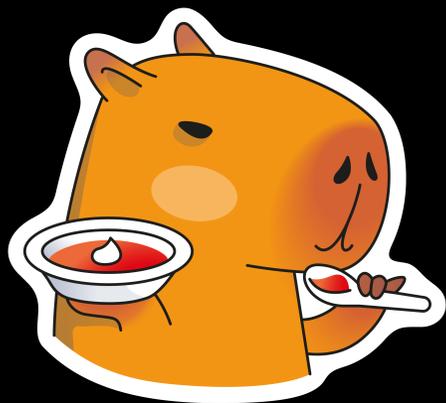
5 Настроить порядок выполнения во время сборки в build.gradle

```
check.shouldRunAfter checkstyleMain, checkstyleTest
```

6 Настроить проверку на CI. Пример файла .gitlab-ci.yml

```
checkstyle:
  stage: checkstyle
  script:
    - ./gradlew checkstyleMain checkstyleTest # Запускаем задачи Checkstyle
  only:
    - merge_requests # Определяем, что задача Checkstyle запускается только для merge
request '0В
  artifacts: # Сохраняем отчеты Checkstyle в качестве артефактов
    paths:
      - build/reports/checkstyle # Путь к отчетам Checkstyle
    expire_in: 1 week # Устанавливаем срок хранения артефактов – 1 неделя
```

Боты для линтеров



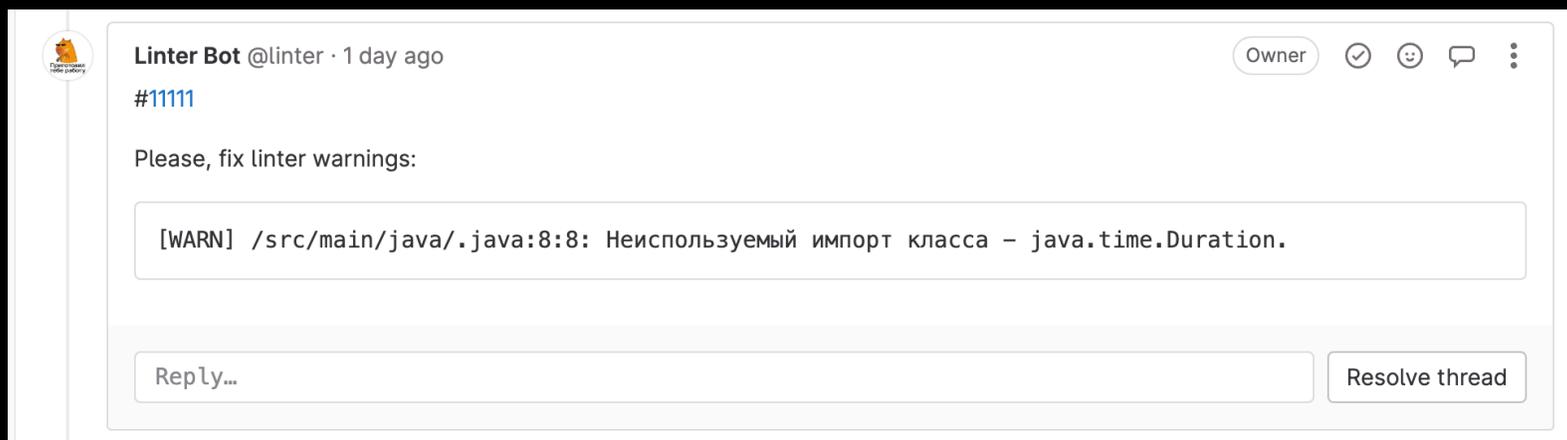
Получаем данные



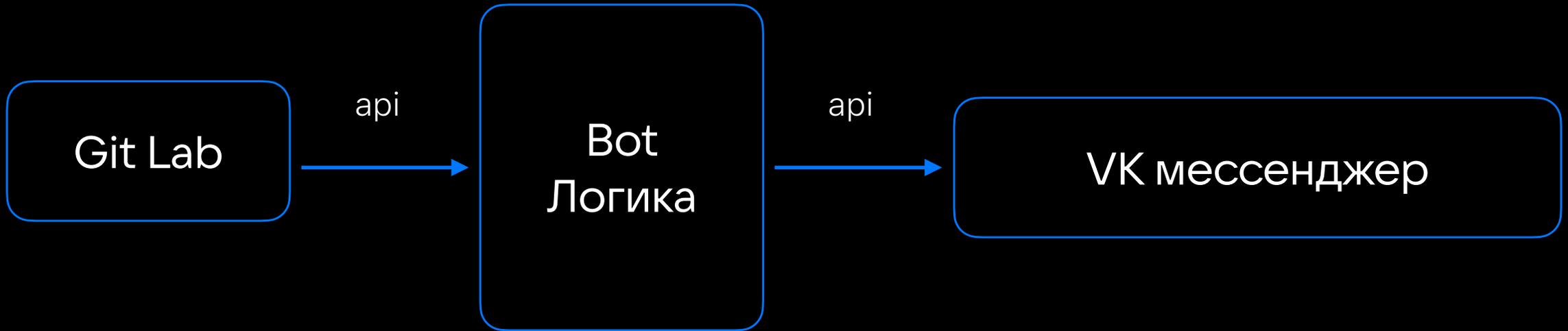
Анализируем данные



Пишем в комментарий merge request'a



Боты для линтеров



GitLab Bot 17:52



В мерж-реквесте

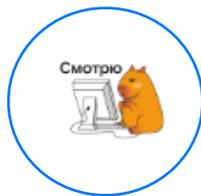
https://gitlab.com/qa/-/merge_requests/1111

были найдены проблемы линтером. Ниже список всех проблемных мест:

Please, fix linter warnings:[WARN]

src/main/java/pageObjects/functional/testClass.java:36:3: Длина идентификатора 'veryVeryLongIdentifier' не должна превышать '20' символов. [VariableNameLength]

Проблема:
*ревьюеры не
подключаются
к MR'ам*

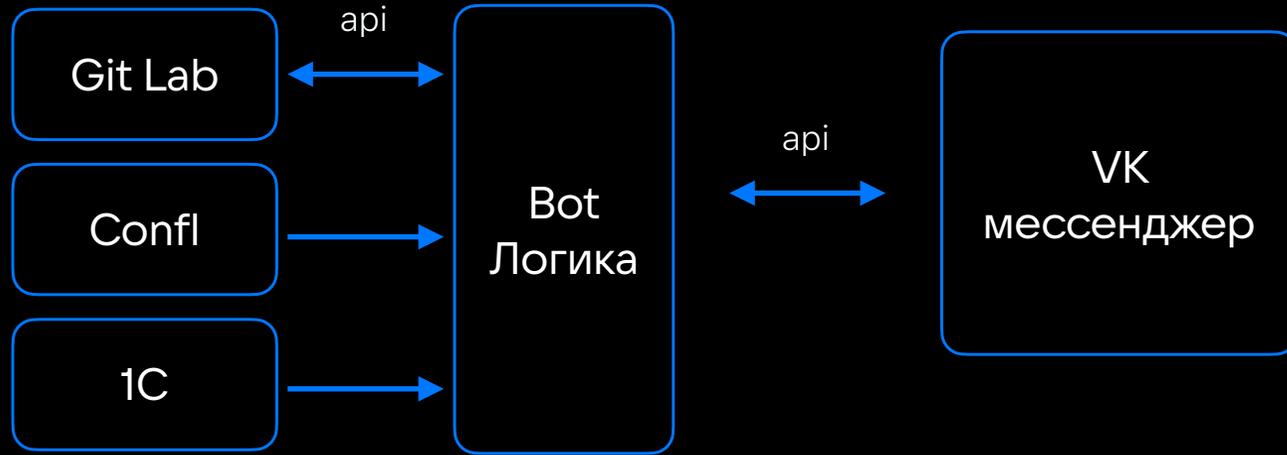


Mr Karibaga 13:57

Здравствуйте! Посмотрите, пожалуйста, мр
https://gitlab.com/qa/-/merge_requests/1111



БОТЫ ДЛЯ ЧАТОВ



| Ссылка на проект | Ссылка на группу ревьюверов | Люди – исключения для назначения | Слать ли уведомления о новых МР в личку? | Сколько ревьюверов назначать? |
|------------------|-----------------------------|---|--|-------------------------------|
| 12121 (ID) | 1221 (ID) | kapibaga_reviewer@vk.team | true | 1 |

Проблема:
не всегда удобно
проводить ревью
в системе
управления
репозиториями

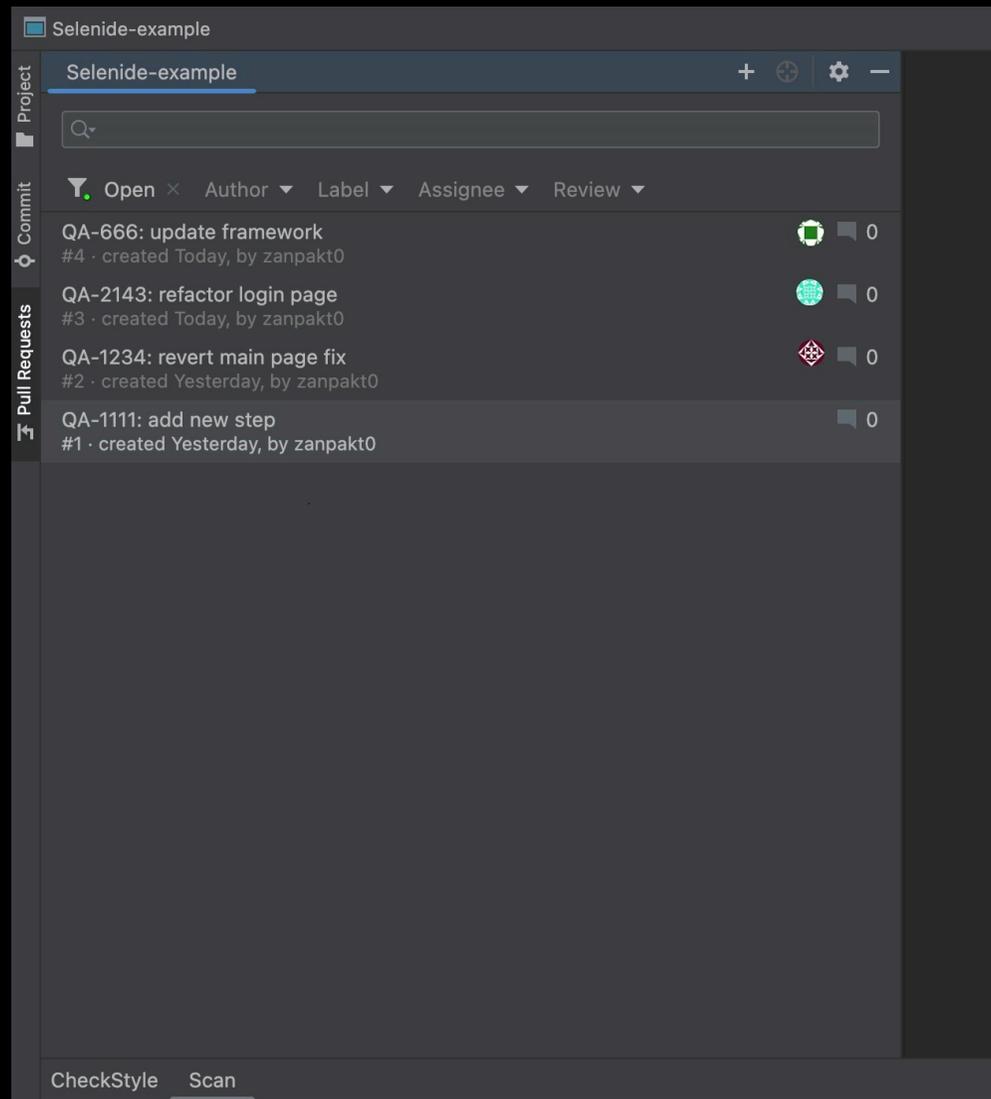
```
public class LoginFormTest extends BaseTest {
```

10 тысяч новых строк?
Может, стоило декомпозировать задачу?



Ревью в среде разработки

Список мерж реквестов



The screenshot shows a pull request list for a project named "Selenide-example". The interface includes a search bar, filter options (Open, Author, Label, Assignee, Review), and a list of pull requests with their titles, IDs, creation times, authors, and status icons.

| Commit | Author | Label | Assignee | Review |
|--|--------|-------|----------|--------|
| QA-666: update framework #4 · created Today, by zanpakt0 | | | | 0 |
| QA-2143: refactor login page #3 · created Today, by zanpakt0 | | | | 0 |
| QA-1234: revert main page fix #2 · created Yesterday, by zanpakt0 | | | | 0 |
| QA-1111: add new step #1 · created Yesterday, by zanpakt0 | | | | 0 |

CheckStyle Scan

Ревью в среде разработки

Изменённые файлы и блоки кода

The screenshot displays an IDE interface for reviewing a pull request. The left sidebar shows the project structure with the following files:

- QA-1111: add new step #1
 - View Timeline
 - Changes
 - kapibaga/fix/QA-1111
 - Selenide-example.main 1 file src/main
 - java/com/qa/page 1 file
 - LoginModalPage.java
 - Selenide-example.test 1 file src/test
 - java/com/qa 1 file
 - LoginFormTest.java

The main editor area shows a diff for Pull Request #1. The diff compares two versions of the code, with line numbers on the left and right. The changes are highlighted in green (added) and red (removed).

Diff for Pull Request #1

4 differences in 2 files

Review: Submit.....

f490cd4b ↔ dcc33b64 (LoginModalPage.java)

Split Unified

LoginModalPage.java ~/Developer/test_project/src/main/java/com/qa/page

```
package com.qa.page;

public class LoginModalPage {
    // ...
}

package com.qa.page;
import com.codeborne.selenide.SelenideElement;
import io.qameta.allure.Step;
import static com.codeborne.selenide.Selenide.$;

public class LoginModalPage {
    private final SelenideElement closeButton = $("[data-test-id=close-button]");

    @Step("Закрываем форму авторизации")
    public void closeLoginModal() {
        closeButton.click();
    }
}
```

LoginFormTest.java ~/Developer/test_project/src/test/java/com/qa

```
package com.qa;

import org.testng.annotations.Test;

import static com.codeborne.selenide.Condition.visible;

public class LoginFormTest extends BaseTest {
    // ...
}
```

Waiting for review from SashaKachina

Merge...

Ревью в среде разработки

История мерж реквеста и ошибки пайплайна

The screenshot shows the 'Diff for Pull Request #2' window in IntelliJ IDEA. The left sidebar displays the project structure with files like 'LoginModalPage.java' and 'LoginFormTest.java'. The main area shows the commit history for the pull request, including actions by 'zanpakt0' and 'SashaKachina'.

QA-1111: add new step #2

Changes: kapibaga/refactor/QA-1234

- zanpakt0 2 minutes ago: No description provided
- zanpakt0 3 minutes ago: added a commit to this pull request (9afca50 QA-1111: add new step, Deleted user 3 minutes ago)
- zanpakt0 A minute ago: requested a review from SashaKachina, renamed this pull request from 'QA-1111: add new step' to 'QA-1234: add new step'
- SashaKachina Moments ago: approved the changes

This screenshot shows the bottom part of the IDE interface. A tooltip displays build check results: 'Smoke Tests - Custom Branch' (failed), 'Check Code Style' (passed), and 'MR.Check' (passed). Below it, a 'Checks failed' notification is visible. The right sidebar shows the review history, including a review by 'SashaKachina' 4 minutes ago.

Smoke Tests - Custom Branch

Check Code Style

MR.Check

Checks failed Details...

SashaKachina approved review

Merge...

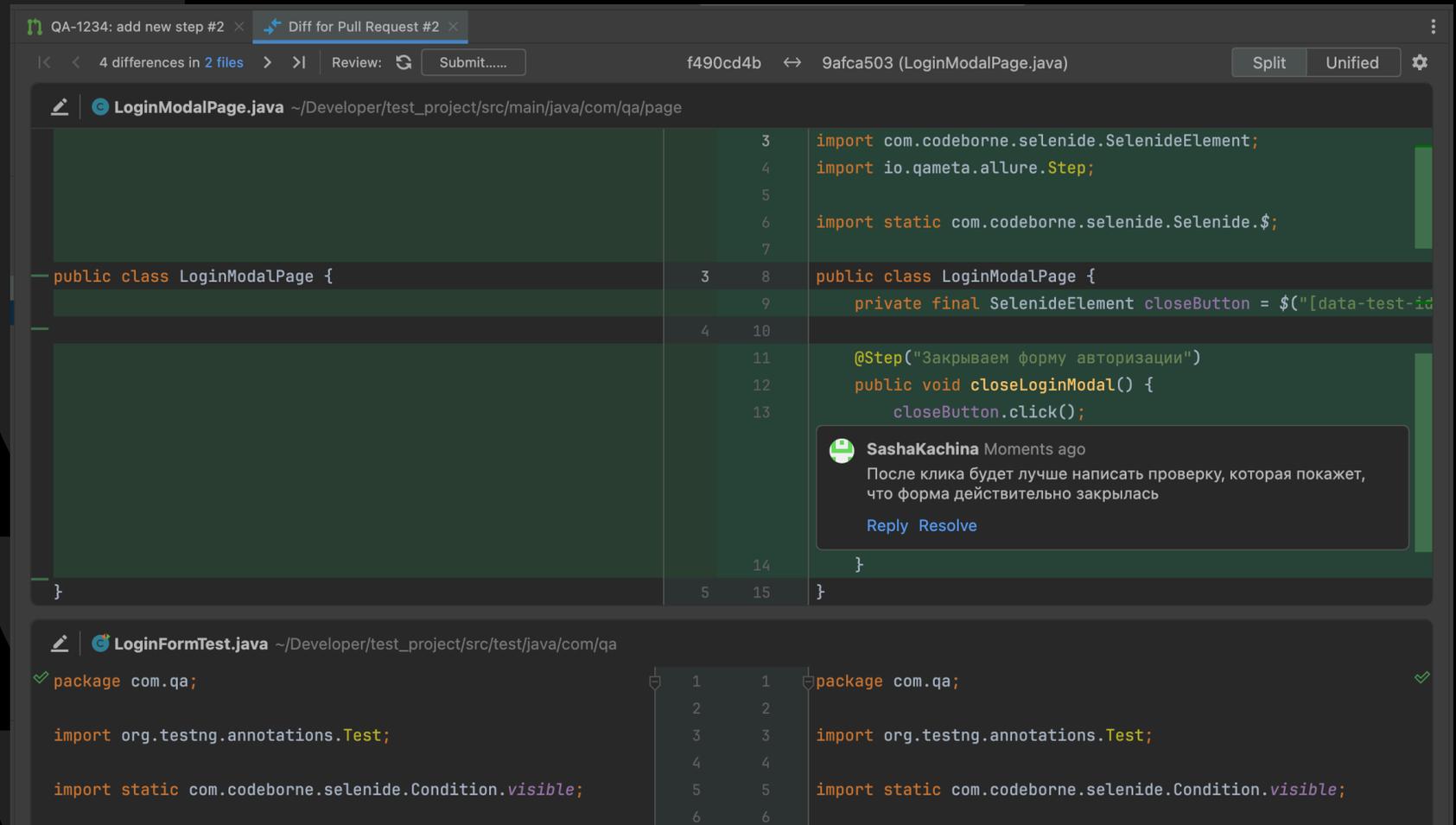
SashaKachina 4 minutes ago: reviewed the changes

SashaKachina Moments ago: approved the changes

Git TODO Problems Terminal CheckStyle Services App Inspection Logcat Build Dependencies

Ревью в среде разработки

Комментирование



Функции комментирования

Саджесты:

- 1** использование специального синтаксиса для комментирования
- 2** принятие или отклонение предложения автором кода

Преимущества подхода:

- 1** ускорение процесса ревью
- 2** точность коммуникации
- 3** образовательный эффект
- 4** улучшение качества кода



Какие могут еще остаться ошибки на ревью?

1 Бездумная копипаста

```
@Test(  
    description = "Пользователь добавляет сопутствующие  
товары к товару в магазине"  
)
```

Описание скопировано из соседнего теста, выглядит как не то, что должно быть)

```
@Owner(КАПИБАГА)  
@Test(description = "Удаление товара из закладок на новой  
витрине сообщества")  
@SetUp({ "openExtendedStore" })
```

А для чего здесь включается расширенный магазин? Вроде бы дальше работа идет с пейджом базового магазина



Какие могут еще остаться ошибки на ревью?

2 Неинформативные комментарии для отчётов о падении или их отсутствие

```
// Assert  
assertThat(marketItems.getItems()).as("Вернулось кол-во  
товаров, отличное от ожидаемого")
```

Не совсем информативное сообщение об ошибке, лучше написать, например, что количество товаров должно быть равно нулю

```
// Assert  
assertThat(responseId).isPositive();
```

Не хватает комментариев, которые будут отображаться при падении тестов

Какие могут еще остаться ошибки на ревью?

3 Отсутствие проверки после совершения действия

```
elements.photoEditorButton  
  .shouldBe(visible, TIMEOUT)  
  .click();
```

Проблема этого степа в том, что если клик не сработает, то шаг все равно будет зеленым, а ошибка возникнет в другом шаге. Поэтому, если клик — последняя операция в шаге, его нужно проверять

Какие могут еще остаться ошибки на ревью?

4 Оставление в коде сервисных инструментов

```
@Test(description = "Создание услуги", invocationCount = 50)
```

Нужно будет удалить invocationCount, чтобы тест не заддосил инфру (полагаю, это случайно попало в коммит)

Какие могут еще остаться ошибки на ревью?

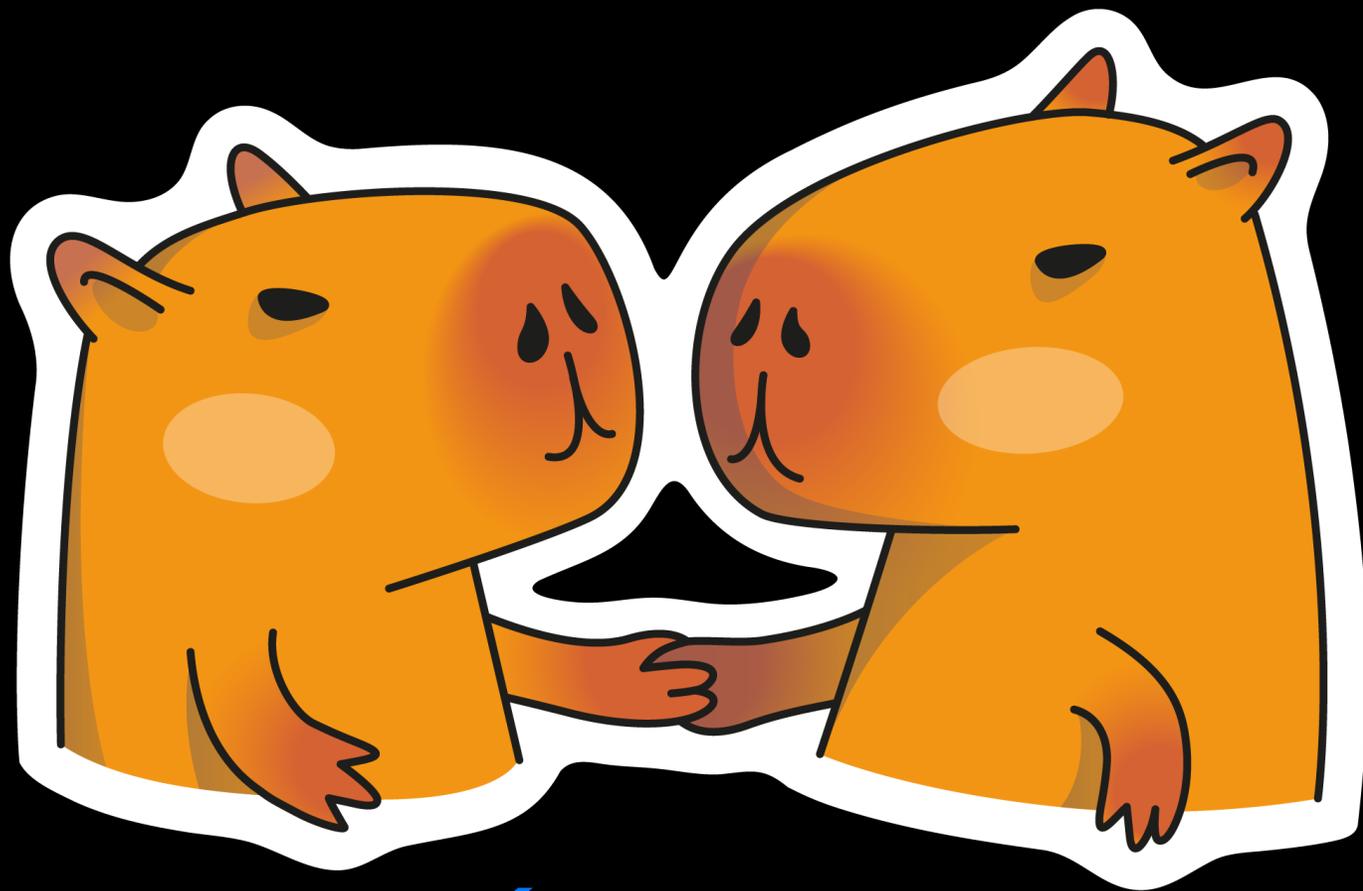
5 Отсутствие выноса кода в общий метод

```
// Assert
assertThat(getResponse.get(0).getId()).as("Поле id не должно
быть пустым").isNotEmpty();

assertThat(getResponse.get(0).getName()).as("Поле name не
должно быть пустым").isNotEmpty();
```

Эти проверки выглядят одинаково, можно общий метод сделать в классе с ассертами

Какие могут еще остаться
ошибки на ревью?



6 Братский апрув

Какие могут еще остаться ошибки на ревью?

Рекомендации

Составьте чеклист для написания автотеста:

1

Проверить аннотации

2

Проверить, как будет отображаться лог в каждом шаге, если тест упадёт

3

Проверить, остались ли в коде сервисные методы для дебага и т.д.

Как проводить код-ревью?

Рекомендации

- ✓ Будьте тактичны
- ✓ Фокусируйтесь
- ✓ Пишите комментарии, подкрепленные аргументами
- ✓ Пишите опциональные комментарии, указывая об этом
- ✓ Запрашивайте уточнения
- ✓ Поставьте себе ограничения по времени
- ✓ Используйте чек-лист для код-ревью

Как проводить код-ревью?

Рекомендации



Не пишите неинформативные комментарии



Не используйте открытую агрессию



Следите за пассивной (скрытой) агрессией и токсичностью



Воздержитесь от шуток и иронии



Не уводите диалог в личные сообщения



Сделаем выводы

- 1** Ревью кода — это полезно
- 2** Использование возможностей среды разработки помогает избегать ошибок
- 3** Автоматизация проверки кода на CI и настройка уведомлений помогает ускорить фиксы типовых ошибок
- 4** Быть душницей и токсиком — не круто

Я не душнила,
я внимательный



Контакты



Telegram стикеры
VKMarketQA



VK Маркет



Александра Качина

