



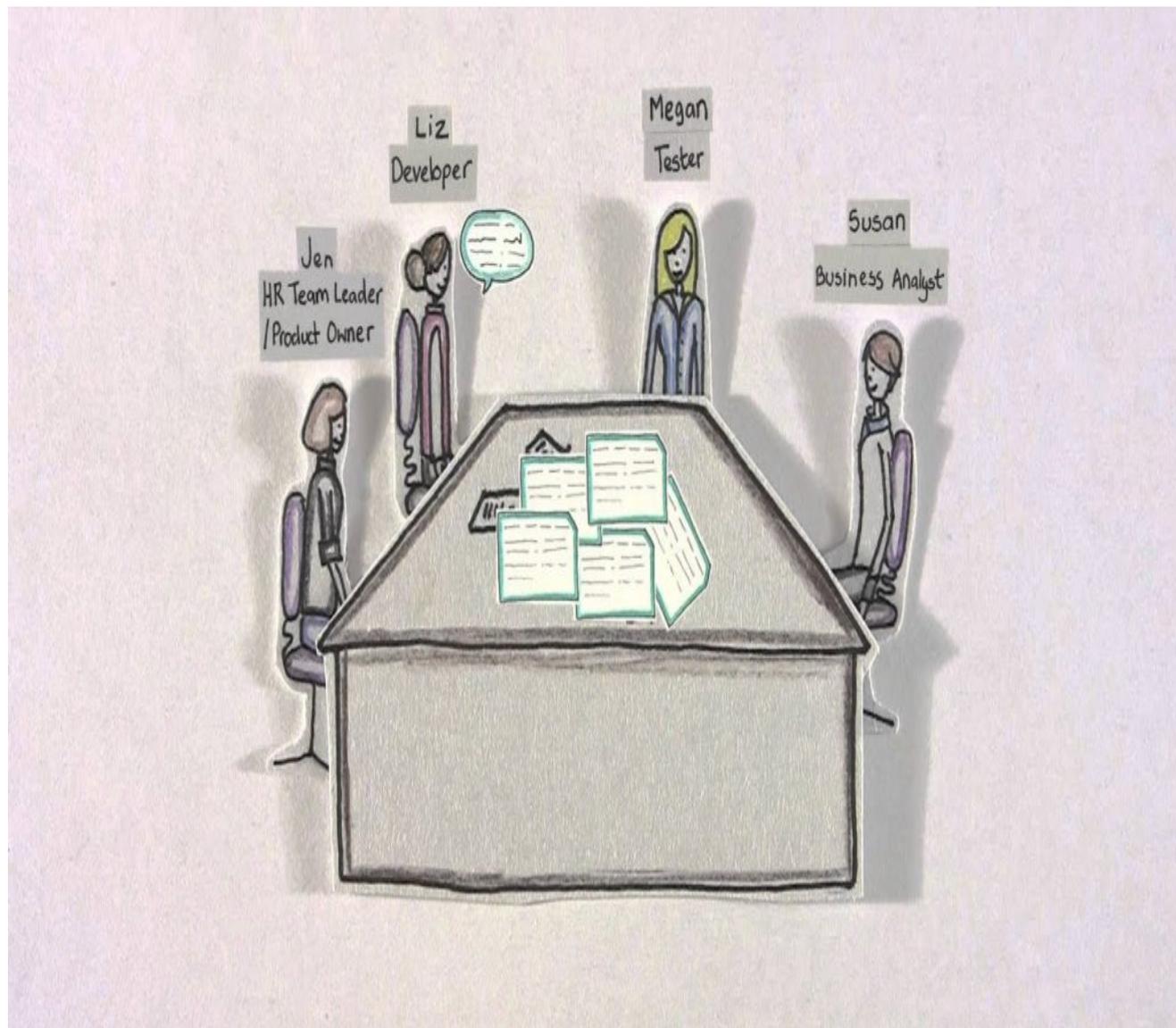
## 12 Лучших инструментов тестовой разработки (TDD) для экстремального программирования

### Описание

Ниже представлен список инструментов для разработки, управляемой тестами (TDD), которые помогут вам разрабатывать более качественное программное обеспечение в Agile DevOps-среде. Время и пространство разработки программного обеспечения стремительно меняются. Стремительно меняются требования к компьютерному программному обеспечению и мобильным приложениям. К тому времени, когда вы разрабатываете минимально жизнеспособный продукт (MVP), тенденции рынка уже могут измениться, и прототип становится не таким уж и трендовым.

Agile-разработка программного обеспечения, основанная на концепции экстремального программирования (XP), позволяет справиться с подобными рисками при разработке программного обеспечения для проектов с фиксированным временем выполнения, связанных с новейшими технологическими тенденциями. DevOps, следуя фреймворку XP, в значительной степени опирается на TDD. Поэтому, если вы находитесь в таком проекте, вам могут помочь следующие инструменты.

## Что такое разработка, управляемая тестами (TDD)?



Test-Driven Development (TDD) – это часть процесса DevOps, в котором особое внимание уделяется написанию тестовых сценариев перед написанием реального кода. Это циклический рабочий процесс, в котором программисты сначала пишут тестовый сценарий, состоящий из функциональности, которая им нужна в программе. При тестировании программа, очевидно, дает сбой. Этот неудачный тест становится предметом дальнейших корректирующих действий. Например, программисты должны написать минимальное количество строк кода, чтобы сценарий тестового случая прошел.

Затем инженеры-программисты рефакторят код, чтобы улучшить его пользовательский интерфейс, UX и дизайн и избавиться от неэффективного или дублирующего кода. В двух словах, TDD – это дисциплинированный процесс

тестирования программного обеспечения перед созданием прототипа для проверки кода на соответствие различным неудачным сценариям. Таким образом, прототип становится менее глючным и более пригодным для бета-тестирования тестировщиками или несколькими фокус-группами из числа конечных пользователей. Коды, созданные в процессе TDD, более удобны для сопровождения и надежны.

Например, ниже приведен процесс TDD для сайта электронной коммерции:

- Напишите тестовые примеры для списков товаров, описаний товаров, карусели товаров, корзины и процесса оформления заказа.
- Затем создайте случайные маршруты клиентов для выбора и добавления товаров в корзину и оформления заказа.
- Отметьте все сценарии, в которых эти случайные действия клиентов оказываются неудачными.
- Переделайте исходные коды так, чтобы они прошли тесты.
- Теперь команда разработчиков может работать над такими элементами, как дизайн, UI, UX и т.д.

## **Как TDD вписывается в Agile и DevOps?**

TDD является важной частью экстремального программирования, которое, в свою очередь, является неотъемлемой частью Agile и DevOps. В Agile-разработке TDD способствует инкрементальному и итеративному процессу разработки путем создания неудачных тестовых примеров и написания минимального кода для прохождения теста. Затем команды, расположенные ниже по конвейеру, могут предоставить обратную связь, и команда TDD начнет разрабатывать больше неудачных тестовых сценариев и модифицировать код для прохождения всех этих тестовых случаев. Такая итерация неудачных и пройденных тестов обеспечивает обратную связь между командами, участвующими в Agile-разработке. В DevOps TDD поддерживает общую цель – быстрое создание высококачественного программного обеспечения.

Внедряя автоматизацию тестирования, TDD помогает создать конкретную основу для конвейеров непрерывной интеграции и непрерывной доставки (CI/CD). Поскольку на ранних этапах разработки вы устраняете часто встречающиеся неудачные тестовые случаи, вам нужно беспокоиться только о функциональности основных функций ПО, которые будут использовать конечные пользователи, и вы

можете заниматься процессом бета-тестирования. Синхронизируйте результаты бета-тестирования с CI/CD, и вы сможете наладить циклический рабочий процесс разработки высококачественного ПО в более короткие сроки. Сотрудничество и коммуникация – важный аспект Agile и DevOps. Тестовая разработка также способствует этому в межфункциональных командах. Разработчики, дизайнеры и операторы могут согласовать свое понимание функциональных возможностей конечного продукта, предварительно обсудив сценарии тестовых примеров.

## **Преимущества разработки, управляемой тестами (TDD)**

Ниже перечислены преимущества TDD по сравнению с традиционным тестированием:

- TDD стимулирует оптимизацию кода.
- Он помогает разработчикам более эффективно анализировать и понимать требования заказчика, при необходимости обращаясь за разъяснениями.
- TDD упрощает процесс добавления и тестирования новых функциональных возможностей на более поздних этапах разработки.
- Разработка, управляемая тестами, обеспечивает более высокое тестовое покрытие по сравнению с традиционными моделями разработки.
- При этом особое внимание уделяется созданию тестов для каждой функциональности с самого начала.
- TDD также повышает производительность труда разработчиков.
- Кодовая база, созданная в рамках TDD-проекта, более гибкая и удобная для сопровождения, чем коды, полученные в результате традиционного тестирования.

## **Обязательные функции TDD-инструмента для DevOps**

Ниже перечислены функции, которые вы должны искать в TDD-инструменте:

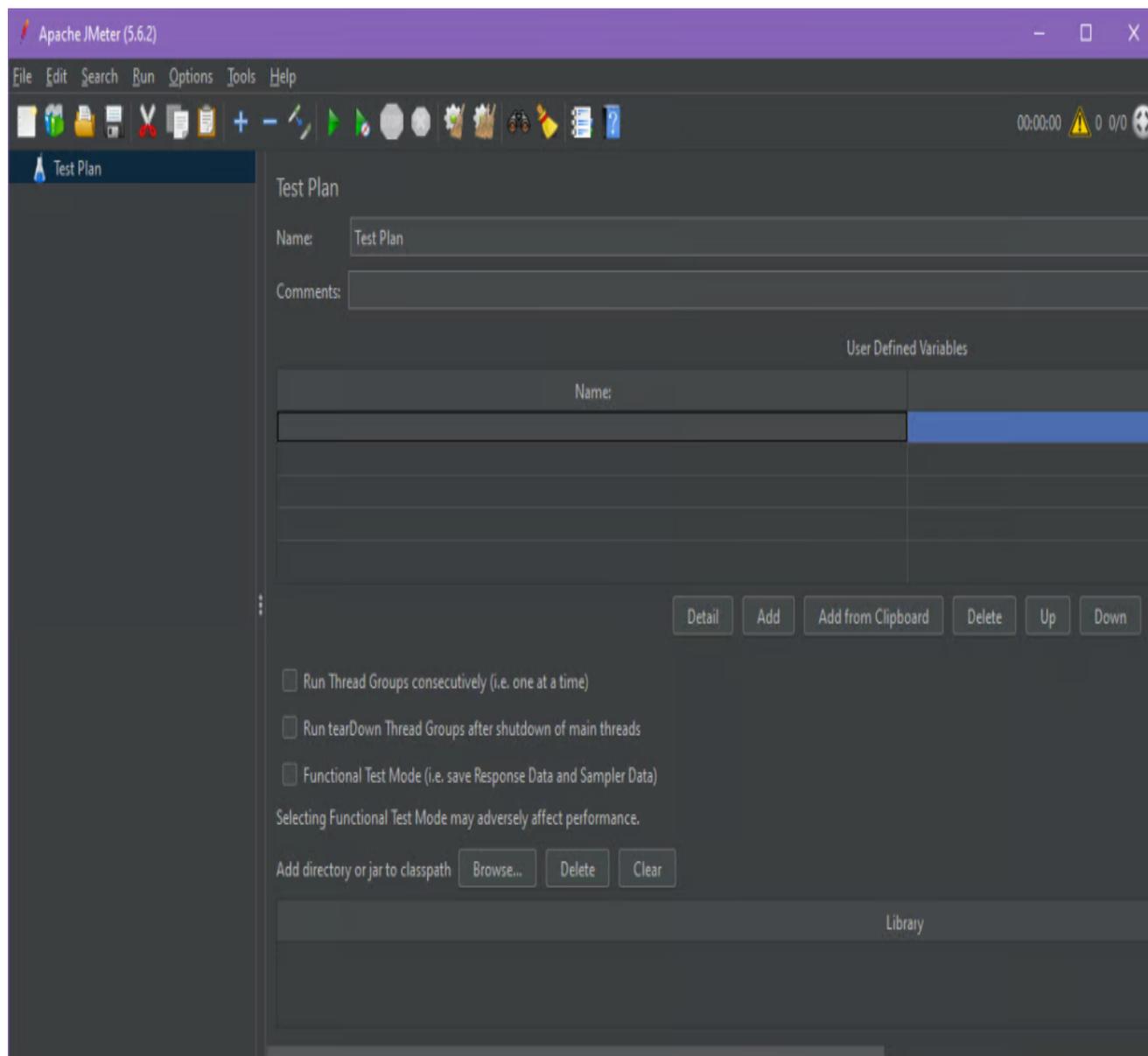
- Приборная панель, напоминающая о функциональных возможностях, необходимых в конечном программном продукте.
- Возможность написания небольших тестовых кодов для конкретных функций.

- Должны быть доступны функции рефакторинга кода.
- Тестовая среда для выполнения тестового кода и немедленного получения обратной связи.
- Функции автоматизации для выполнения тестовых примеров без постоянного контроля.
- Возможность реализации цикла Red-Green-Refactor фреймворка TDD.
- Функциональные возможности, позволяющие сбалансировать потребность в приемочных, интеграционных и модульных тестах.
- Интеграция с CI/CD, чтобы инструмент мог запускать автоматические тесты при изменении кода.

Давайте рассмотрим лучшие инструменты TDD, которые вы можете использовать в своих DevOps-проектах:

## **12 Лучших инструментов тестовой разработки (TDD) для экстремального программирования**

### **Apache JMeter**



Apache JMeter – это Java-приложение, позволяющее проводить нагрузочное тестирование функционального поведения приложений и измерять производительность тестов. С его помощью можно тестировать производительность как динамических, так и статических приложений, а также веб-приложений. Ниже перечислены его основные возможности:

- Нагрузочное тестирование и тестирование производительности серверов, приложений и интернет-протоколов.
- Поддерживаются такие протоколы, как LDAP, базы данных через JDBC, FTP, веб-сервисы SOAP / REST и др.
- Полнофункциональная тестовая среда IDE, позволяющая записывать,

---

отлаживать и создавать тестовые планы из нативных приложений и веб-браузеров.

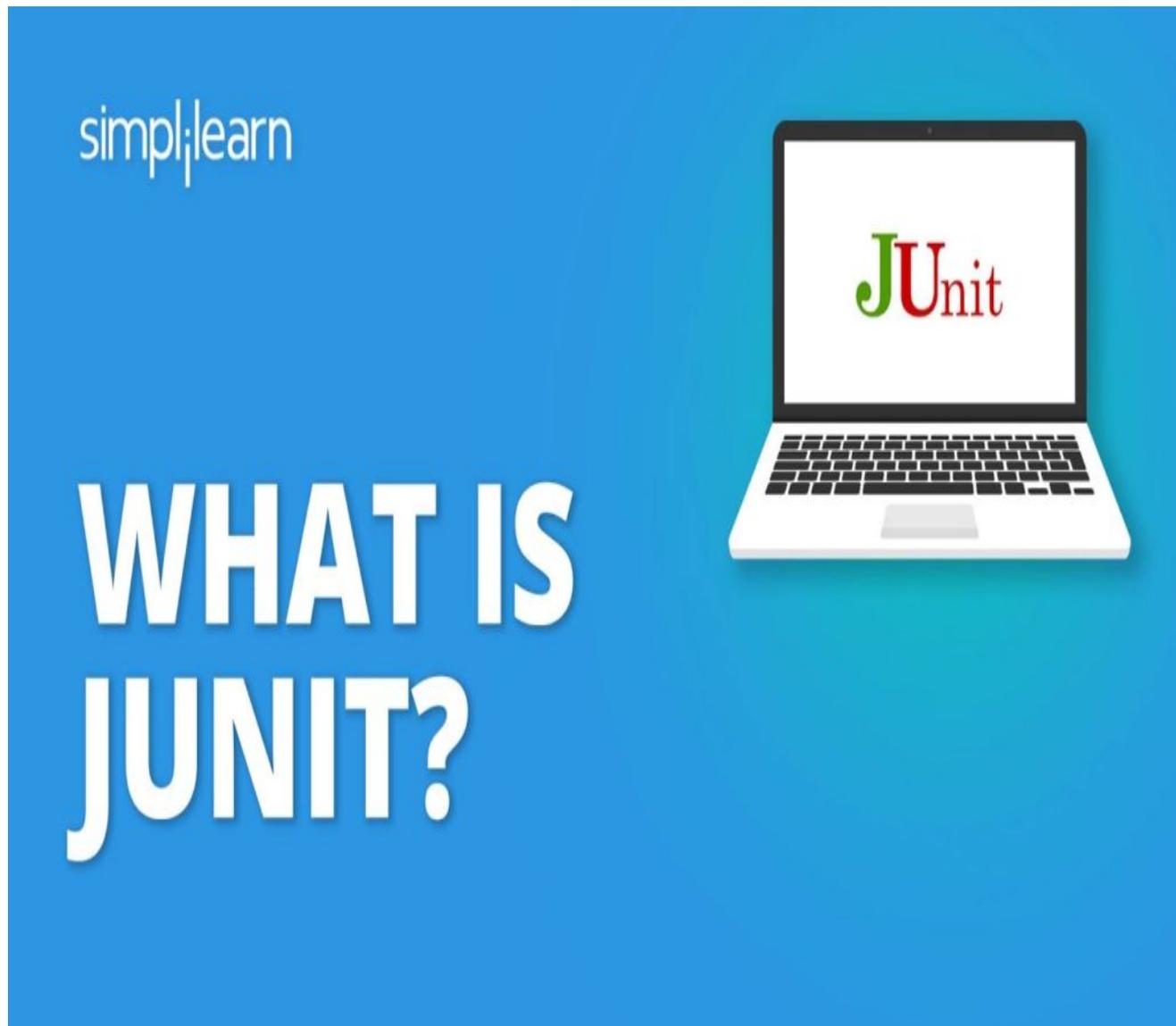
- Создание динамических HTML-отчетов
- Многопоточная обработка, позволяющая выполнять несколько планов тестирования
- Поддерживает непрерывную интеграцию (CI) с помощью Gradle, Maven и Jenkins.

## Mockito



Mockito – это незаменимый фреймворк для TDD-проектов. Он позволяет вам и вашей команде DevOps создавать объекты-макеты для целей тестирования. Этот инструмент для написания тестов упрощает процесс изоляции зависимостей в процессе тестирования кода. Кроме того, можно легко проверить поведение тестового объекта. Mockito позволяет имитировать и внешние зависимости. Например, можно создавать макеты баз данных или веб-сервисов. Затем использовать эти макеты для дальнейшего тестирования макетных объектов небольших тестовых функциональных возможностей вашего программного обеспечения. Различные DevOps-проекты часто используют JUnit 4 вместе с Mockito для облегчения разработки программного обеспечения, ориентированного на тестирование и поведение.

## JUnit



JUnit (последняя версия JUnit 5) – это популярный инструмент TDD для выполнения планов тестирования на виртуальной машине Java (JVM). Он также предлагает TestEngine API, необходимый для разработки фреймворков тестирования на JVM. Кроме того, JUnit 5 включает в себя такие удобные функции, как:

- Консоль для запуска тестов из CLI
- JUnit Platform Suite Engine для запуска настраиваемых тестовых наборов

Более того, в популярных интегрированных средах разработки (IDE), таких как IntelliJ IDEA, Eclipse, NetBeans, Visual Studio Code и т.д., имеется встроенная поддержка этого инструмента. Кроме того, JUnit 5 можно легко интегрировать с

такими инструментами сборки, как Ant, Maven и Gradle.

## pytest

```
$ pytest
===== test session starts =====
platform linux -- Python 3.x.y, pytest-7.x.y, pluggy-1.x.y
rootdir: /home/sweet/project
collected 1 item

test_sample.py F [100%]

===== FAILURES =====
_____ test_answer _____

    def test_answer():
>     assert inc(3) == 5
E       assert 4 == 5
E       + where 4 = inc(3)

test_sample.py:6: AssertionError
===== short test summary info =====
FAILED test_sample.py::test_answer - assert 4 == 5
===== 1 failed in 0.12s =====
```

pytest – это фреймворк для тестирования на основе Python. Разработчики DevOps и Agile-программ используют его для простого написания и масштабирования тестовых кодов на Python CLI. С помощью pytest можно писать простые тестовые примеры для пользовательских интерфейсов (UI), баз данных и интерфейсов прикладного программирования (API). Ниже перечислены некоторые из его лучших возможностей:

- Автоматическое обнаружение тестовых функций и модулей.
- pytest может запускать “носовые” и модульные тесты, используя встроенный функционал
- 1 000+ проектов PyPI или плагинов, которые помогут вам в процессе TDD

Не говоря уже о том, что при увеличении требований можно масштабировать процесс написания и оценки тестовых примеров.

## NUnit

Если вы создаете программное обеспечение на платформе .NET с использованием любого из поддерживаемых языков, таких как F#, C# и Visual Basic, вы можете использовать NUnit для модульного тестирования.

Его лучшие возможности описаны ниже:

- NUnit 3 Test Adapter позволяет запускать тесты NUnit 3 внутри VS Code.
- NUnit Engine позволяет запускать тесты, разработанные в различных тестовых фреймворках
- VS Test Generator помогает создавать IntelliTests и модульные тесты.

Последняя версия NUnit 3 уже доступна в Visual Studio IDE и редакторе кода. Вы можете легко получить к нему доступ через Tools > NuGet Package Manager и доступ к NuGet-пакетам для решения, который открывает браузер для поиска NUnit.Console и пакетов NUnit.

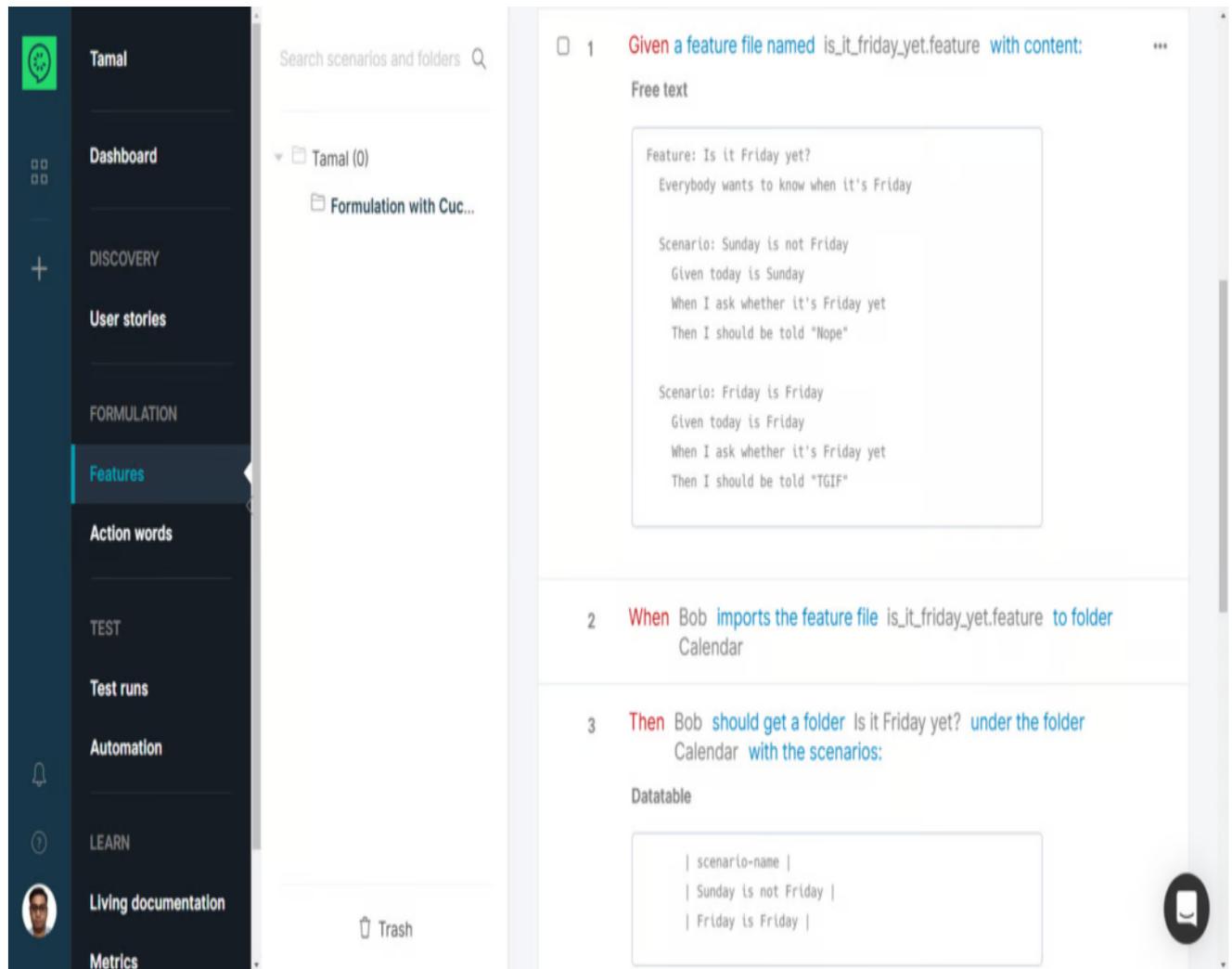
## TestNG

TestNG – это фреймворк для тестирования кода, который упрощает широкий спектр задач тестирования, начиная от юнит-тестирования (тестирования отдельной функциональности в отрыве от всего программного обеспечения) и заканчивая интеграционным тестированием. Богатые возможности TestNG, отличающие его от JUnit и NUnit, таковы:

- Аннотирование случаев модульного тестирования
- Проверка поддержки многопоточности кода
- Возможность тестирования на основе данных.
- Доступны различные плагины и инструменты, такие как IDEA, Eclipse, Selenium, Maven, Ant и т.д.

Интеграционное тестирование в TestNG может включать такие сценарии тестирования, как тестирование внешних фреймворков, например серверов приложений, тестирование нескольких пакетов, а также тестирование отдельного программного обеспечения, состоящего из различных небольших функциональных блоков.

## Cucumber



Для разработки, ориентированной на поведение, можно использовать Cucumber в качестве основного технологического стека, который будет подтверждать, что тестовый пример или готовое программное обеспечение обладает требуемыми заказчику функциональными возможностями. Cucumber сканирует спецификации, написанные в текстовом формате. Эти спецификации в основном представляют собой функциональные возможности, которые должно обеспечивать разрабатываемое программное обеспечение. В различных сценариях может быть несколько спецификаций. Инструмент просматривает все эти спецификации и проверяет соответствие кода спецификациям. Он генерирует отчет со сценариями неудач и успехов. Поддерживается 20+ языков разработки программного обеспечения, таких как Java, Ruby, C++, Lua, Kotlin, Scala, Python и др.

## TestRail



TestRail – это инструмент, напоминающий рабочую область тестирования для всех ваших DevOps-проектов. Он создает экосистему централизованного тестирования кода и QA-платформу с использованием TestRail Quality OS. С помощью функции Build можно создавать множество автоматизированных тестов для различных проектов разработки ПО и организовывать их в рамках системы управления тестами. Модуль Connect позволяет объединить средства автоматизации тестирования, программы отслеживания проблем, такие как Jira, и конвейеры DevOps с данными тестирования из TestRail. Наконец, модуль Optimize позволяет определить приоритеты QA-процессов для проведения необходимых тестов путем мгновенного выявления рисков.

## RSpec

```
Bowling#score
  with no strikes or spares
    sums the pin count for each roll (FAILED - 1)

Failures:

  1) Bowling#score with no strikes or spares sums the pin count for each roll
     Failure/Error: expect(bowling.score).to eq 80

       expected: 80
        got: nil

       (compared using ==)
       # ./spec/bowling_spec.rb:8:in `block (3 levels) in <top (required)>'

Finished in 0.00142 seconds (files took 0.13793 seconds to load)
1 example, 1 failure

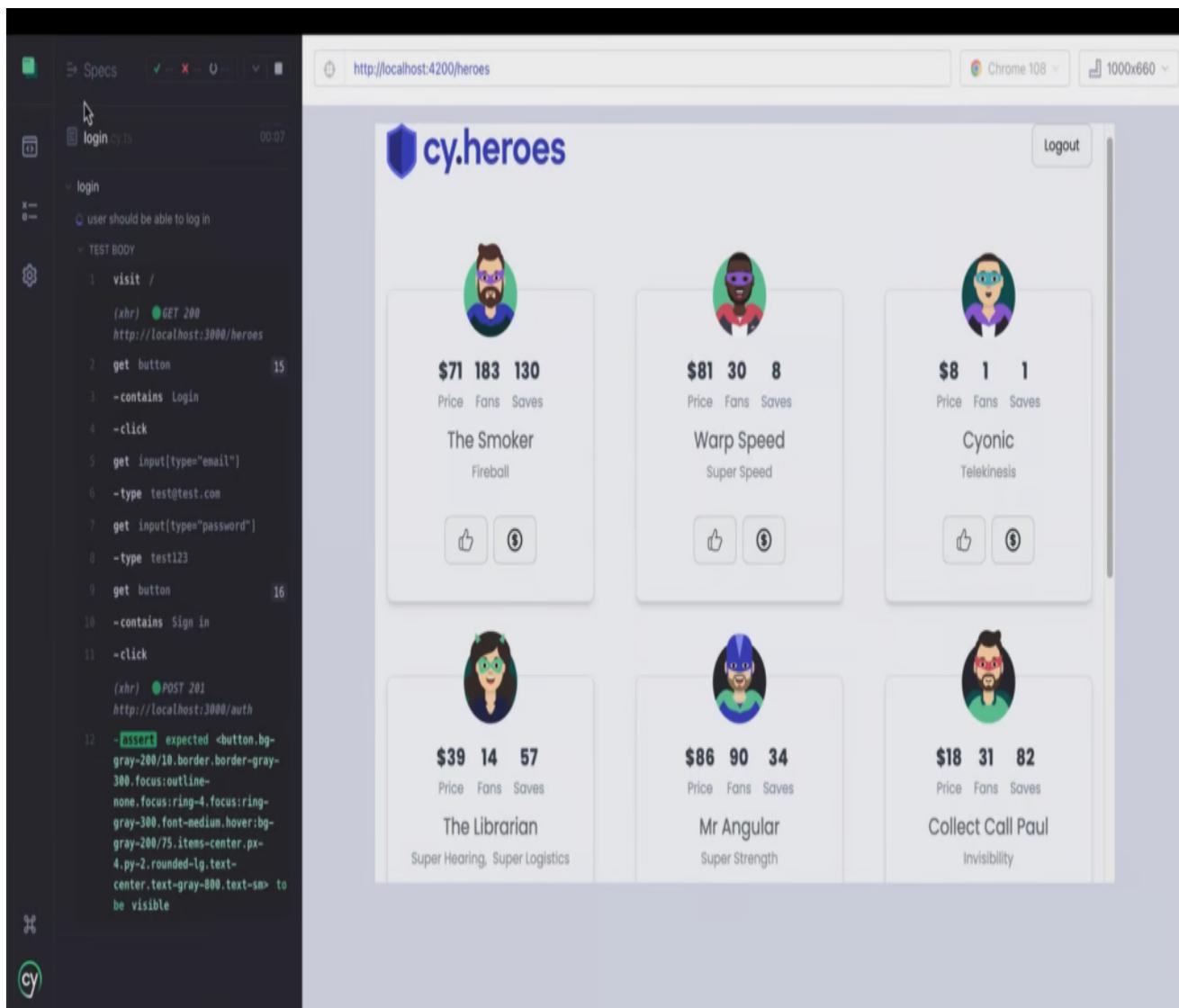
Failed examples:

rspec ./spec/bowling_spec.rb:5 # Bowling#score with no strikes or spares sums the pin count for each roll
→ bowling git:(master)
```

Если ваша команда разработчиков программного обеспечения специализируется на приложениях, разработанных на Ruby, то вам следует обратить внимание на RSpec как на инструмент TDD и BDD для проектов тестирования на Ruby. В RSpec есть несколько библиотек тестирования, которые могут работать как вместе, так и по

отдельности. Это rspec-expectations, rspec-rails, rspec-core и rspec-expectations.

## Cypress



Cypress – это облачный инструмент тестирования, созданный для современных веб-приложений и обладающий функциями нового поколения. Эти возможности описаны ниже:

- Тестирование веб-приложений в веб-браузерах
- Настройка и написание первого тестового случая занимает менее 10 минут
- Отладка неудачных тестов возможна в веб-браузере
- Взаимодействие с приложением осуществляется так же, как и с конечным пользователем, что позволяет устранить типичные ошибки
- Интеграция с такими инструментами непрерывной интеграции, как Circle CI,

GitLab CI, Atlassian Bitbucket и т.д.

Плата подходит для модульных тестов, интеграционных тестов, компонентных тестов и сквозных тестов.

## Jest

```
PASS packages/diff-sequences/src/__tests__/index.test.js
PASS packages/jest-diff/src/__tests__/diff.test.js
PASS packages/jest-mock/src/__tests__/jest_mock.test.js
PASS packages/jest-util/src/__tests__/fakeTimers.test.js
PASS packages/pretty-format/src/__tests__/prettyFormat.test.js

RUNS packages/jest-haste-map/src/__tests__/index.test.js
RUNS packages/pretty-format/src/__tests__/DOMElement.test.js
RUNS packages/jest-config/src/__tests__/normalize.test.js
RUNS packages/expect/src/__tests__/matchers.test.js
RUNS packages/pretty-format/src/__tests__/Immutable.test.js
RUNS packages/expect/src/__tests__/spyMatchers.test.js
RUNS packages/jest-cli/src/__tests__/SearchSource.test.js
RUNS packages/jest-runtime/src/__tests__/script_transformer.test.js
RUNS packages/jest-cli/src/__tests__/watch.test.js
RUNS packages/jest-haste-map/src/crawlers/__tests__/watchman.test.js
RUNS packages/pretty-format/src/__tests__/react.test.js

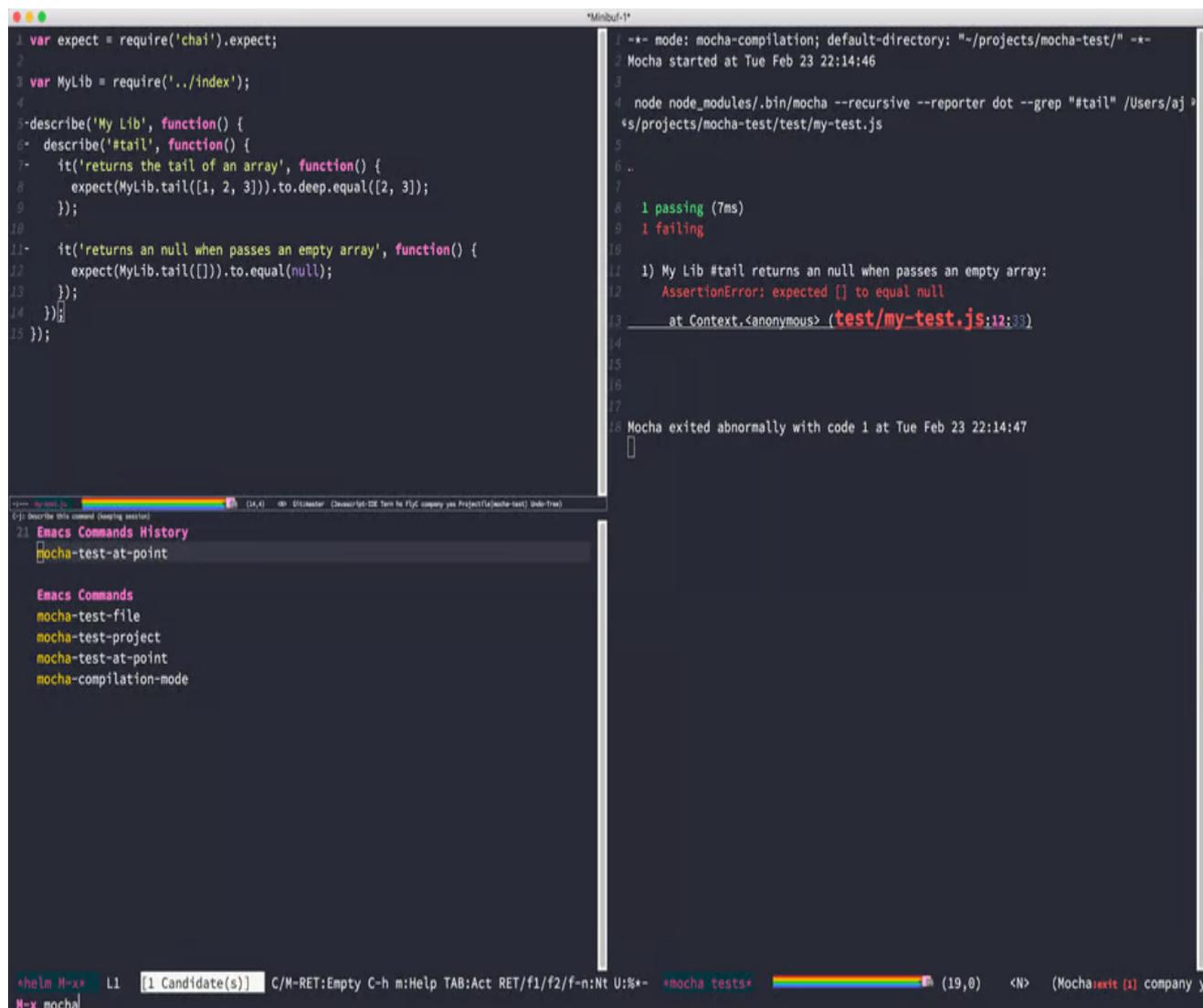
Test Suites: 5 passed, 5 of 303 total
Tests:       332 passed, 332 total
Snapshots:  21 passed, 21 total
Time:        4s
```

Jest – это программное обеспечение для тестирования на JavaScript, которое обычно используется разработчиками для разработки, управляемой тестами (TDD). Его основные возможности включают следующее:

- Работает с большинством JavaScript-проектов “из коробки” без какой-либо настройки
- Создание снимков тестов
- Запуск различных тестов в отдельных процессах для повышения производительности тестирования

Простой API для написания тестов со встроенной поддержкой mocking и assertions. В нем также имеются необходимые API-вызовы для создания отчетов о покрытии.

## **Mocha**



```
1 var expect = require('chai').expect;
2
3 var MyLib = require('../index');
4
5 describe('My Lib', function() {
6   describe('#tail', function() {
7     it('returns the tail of an array', function() {
8       expect(MyLib.tail([1, 2, 3])).to.deep.equal([2, 3]);
9     });
10
11    it('returns an null when passes an empty array', function() {
12      expect(MyLib.tail([])).to.equal(null);
13    });
14  });
15 });
```

```
1 -- mode: mocha-compilation; default-directory: "~/projects/mocha-test/" --
2 Mocha started at Tue Feb 23 22:14:46
3
4 node node_modules/.bin/mocha --recursive --reporter dot --grep "#tail" /Users/aj...
5 *s/projects/mocha-test/test/my-test.js
6 ..
7
8 1 passing (7ms)
9 1 failing
10
11 1) My Lib #tail returns an null when passes an empty array:
12     AssertionError: expected [] to equal null
13     at Context.<anonymous> (test/my-test.js:12:38)
14
15
16
17
18 Mocha exited abnormally with code 1 at Tue Feb 23 22:14:47
```

```
21 Emacs Commands History
   mocha-test-at-point

Emacs Commands
mocha-test-file
mocha-test-project
mocha-test-at-point
mocha-compilation-mode
```

Mocha – это гибкий JavaScript-фреймворк для написания тестов, в том числе тестовых примеров, для ваших TDD-проектов. Он предлагает простую и обширную библиотеку синтаксиса, что позволяет разработчикам легко создавать и запускать тесты. Mocha можно запускать непосредственно в браузере с помощью Node.js. В ней реализована широкая поддержка асинхронного тестирования. Таким образом, можно тестировать код, связанный с обратными вызовами, обещаниями или функциями `async/await`. Кроме того, он предлагает различные функции тестирования кода, такие как отчеты о тестировании, покрытие тестов и крючки для управления несколькими проектами тестирования ПО.

## Заключительные слова

Теперь вы знаете, какие средства разработки, основанной на тестировании, необходимо использовать в проектах экстремального программирования,

ориентированных на DevOps. Вы также можете интегрировать процесс тестирования с конвейером CI/CD, чтобы быстро выводить на рынок высококачественное программное обеспечение до того, как тренд на ту или иную технологию сойдет на нет.

**Дата Создания**

21.07.2023