МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н.Г.ЧЕРНЫШЕВСКОГО»

Кафедра информатики и программирования

РАЗРАБОТКА И РЕАЛИЗАЦИЯ АВТОМАТИЗИРОВАННОГО ТЕСТИРОВАНИЯ ВЕБ-ПРИЛОЖЕНИЙ С ИСПОЛЬЗОВАНИЕМ МЕТОДОЛОГИИ BDD

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

Студентки 4 курса 441 группы
направления 02.03.03 Математическое обеспечение и администрирование информационных систем
факультета компьютерных наук и информационных технологий
Горячкина Виктория Михайловна

Научный руководитель:
Доцент _____ Е.В. Кудрина

подпись, дата

Зав. кафедрой ИиП:
к.ф.-м.н., доцент _____ М.В. Огнева

подпись, дата

Саратов 2022

ВВЕДЕНИЕ

Актуальность темы. В современном мире IT-сфера не стоит на месте, она постоянно развивается. Изо дня в день разрабатываются новые продукты, которые требуют проверки перед тем, как выйти к конечному пользователю, и чтобы он получил высококачественный продукт. При этом нужно немедленно и качественно реагировать на любые изменения. Чаще всего это касается крупных компаний, в частности тех компаний, которое относятся к банковской деятельности. По этой причине, становятся все более популярны многие гибкие модели разработки. Тем не менее выбор между ними определяется для каждого конкретного проекта и может меняться в широком диапазоне, в зависимости от масштаба, новизны и критичности проекта, распределения участников, требований заказчика. Стоит отметить методологию Behavior driven development (сокр. BDD, дословно «разработка через поведение»), она всё чаще вызывает интерес у IT-индустрии как логически верная ступень развития традиционных подходов к тестированию проектов, в том числе подходов к автоматизации тестирования. Применяемо к области автоматизированного тестирования, основной идеей данной методологии является учет пожелания бизнес руководства IT-проектов, заинтересованных в прозрачности и более доступном понимании процессов тестирования, в отрыве от большего числа технических вопросов, что позволяет всем участникам проекта тем самым говорить на понятном друг-другу языке. Первостепенной задачей подхода BDD, центральной точкой, вокруг которой в дальнейшем будут строиться все остальные процессы, является грамотное проектирование структуры и техническая реализация фреймворка автоматизированного тестирования. При правильном использовании методология BDD позволяет сократить время тестирования выпускаемых продуктов, повышает качество проводимого тестирования и делает сам процесс очевидным и доступным для всех.

Цель бакалаврской работы – разработать и реализовать автоматизированное тестирование веб-приложения с использованием методологии BDD на примере сайта интернет-магазина http://it-techno.by/ .

Поставленная цель определила **следующие задачи**:

1. Рассмотреть методологии разработки программного обеспечения через тестирование.
2. Выявить особенности методологии BDD.
3. Изучить инструментальные средства и технологии автоматизированного тестирования.
4. Разработать набор автоматизированных тестов на Java для тестирования пользовательского интерфейса интернет-магазина http://it-techno.by/.
5. Выполнить автоматизированное тестирование пользовательского интерфейса интернет-магазина http://it-techno.by/ проанализировать полученные результаты.

Методологические основы автоматизированного тестирования с помощью методологии BDD представлены в работах Трубанова В. [1], Филькина М.С. и Ручкина В.Н. [2], Куликова С. С. [3], Мартина Р. [4], Дэвида Челимски [5], Себа Роуза, Мэтта Уинна и Аслака Хеллесой [6].

Теоретическая значимость данной исследовательской работы состоит в успешно рассмотренных методологиях разработки программного обеспечения. С упором на изучение методологии BDD, которая помогает всей команде разработчиков и заказчику сформировать пользовательские тесты еще до начала разработки кода.

Практическая значимость данной исследовательской работы состоит в том, что было реализовано автоматизированное тестирование с помощью методологии BDD. Данные тесты могут быть использованы разработчиками сайта для исправления ошибок в функциональной доступности сайта. Также данная разработка может послужить примером для создания тестов для других приложений.

Структура и объём работы. Бакалаврская работа состоит из введения, двух разделов, заключения, списка использованных источников и девяти приложений. Общий объем работы – 74 страницы, из них 45 страниц – основное содержание, включая 10 рисунков, список использованных источников информации – 33 наименований.

КРАТКОЕ СОДЕРЖАНИЕ РАБОТЫ

Первый раздел «Теоретическая часть» посвящен рассмотрению различных методологий разработки программного обеспечения, основанных на гибкой методологии Agile.

Agile – это набор идей и принципов, которых стоит придерживаться, чтобы оставаться работать с ней. Этот подход к разработке программного обеспечения, где взаимодействие с командой, сотрудничество с клиентами и реагирование на изменения являются ключевыми темами.

Были рассмотрены такие методологии как:

- test Driven Development (сокр. TDD, дословно «Разработка через тестирование»);
- behavior driven development (сокр. BDD, дословно «Разработка через поведение»);
- domain driven design (сокр. DDD, дословно «Предметно - ориентированное проектирование»);
- feature driven development (сокр. FDD, дословно «Разработка, управляемая функциями»);
- model Driven Development (сокр. MDD, дословно «Разработка, управляемая моделями»);
- panic Driven Development (сокр. PDD, дословно «Разработка, через панику»).

Рассмотрим кратко каждую по отдельности:

- TDD — это практика разработки программного обеспечения, которая фокусируется на создании тестовых случаев перед разработкой фактического

кода. Это итеративный подход, который сочетает в себе программирование, создание модульных тестов и рефакторинг.

- BDD является ветвью разработки на основе тестирования (TDD). BDD использует удобно читаемые описания требований пользователей программного обеспечения в качестве основы для тестирования программного обеспечения.
- DDD – это практика программирования, ориентированную на предметную область. DDD предоставляет набор шаблонов, терминов и рекомендаций по созданию программного обеспечения из сложных доменов. Это помогает разработчикам управлять сложностью, с которой они сталкиваются при разработке решения.
- FDD – это управляемый моделью и короткий итерационный процесс, который был разработан на основе лучших практик разработки программного обеспечения, включая объектное моделирование предметной области, разработку по функциям и владение кодом.
- MDD – это методология разработки программного обеспечения, которая позволяет пользователям создавать сложные приложения с помощью упрощенных абстракций готовых компонентов. Вместо того, чтобы рассказывать, эти визуальные строительные блоки показывают потребности бизнеса и способы решения технических проблем. MDD является наиболее важным принципом разработки с низким уровнем кода: это мост, который позволяет экспертам в области ИТ и бизнеса сотрудничать, превращая идеи в приложения, которые приносят пользу.
- PDD – техника, стремительно повышающая скорость работы в команде любого проекта за кратчайшие сроки.

Далее были выведены особенности методологии BDD. Отличительной особенностью данной методологии является то, что сценарии тестирования написаны простым языком, понятным каждому, с подробным описанием:

- Фактический тест;

- Как протестировать приложение;
- Поведение приложения.

Благодаря разработке, основанной на поведении, каждый в команде может читать и писать приемочные тесты. Именно код разработчика автоматизирует тесты, но заинтересованные стороны бизнеса могут его понять. Описание приемочных тестов должно вестись на гибком языке пользовательской истории. В настоящее время в практике BDD существует специальный шаблон для спецификации, который впоследствии стал называться язык Gherkin.

Gherkin - человеко-читаемый язык для описания поведения системы, который использует отступы для задания структуры документа. Каждая строчка начинается с одного из ключевых слов и описывает один из шагов.

Также были рассмотрены технологии и инструментальные средства для создания дальнейшего тестирования. Так, был описан язык программирования Java. И такие инструменты, использующие BDD, как Serenity BDD и Cucumber.

Serenity BDD – это библиотека отчетов с открытым исходным кодом, которая позволяет разработчикам писать доступные и структурированные критерии приемлемости для проектов автоматизации тестирования.

Cucumber – это инструмент тестирования, который поддерживает поведенческую разработку.

Подводя итог первого раздела, можно сделать вывод, что при его написании были получены знания о разных методологиях Agile. Также были получены знания о особенностях такой методологии, как BDD. И были рассмотрены такие технологии, как Java, и инструменты Serenity BDD и Cucumber.

Второй раздел «Практическая часть» посвящен реализации автоматизированного тестирования с помощью методологии BDD. Были созданы автоматические тесты для пользовательского интерфейса в связи с тем, что не было доступа к документации и требованиям приложения. Разработка тестов была выполнена на Java, в среде разработки IntelliJ IDEA.

В программе есть файлы с расширением feature. В них прописаны шаги и поведение, которые необходимо автоматизировать.

Например, один из таких файлов выглядит так:

Scenario: Log in to your personal account

Given site_open

When Open form to input

And Input personal data

Then Signed in

Выделенные слова являются ключевые слова. А то, что справа, это выражения, на которые ссылаются в файлах реализации.

Также в программе реализованы step-файлы. В них существуют методы программного кода, которые связываются, по ключевым словам, со сценарием. Первая строчка – это аннотация, с помощью которой Cucumber понимает к какому именно степу относится данное определение. Следующая строка содержит модификатор доступа public, тип возвращаемого значения void и названия метода. Метод, определяющий степ, всегда должен быть public void.

Для вышеуказанного сценария step-файл выглядит так:

```
@Given("^site_open")
public void opening_site() {
    mainpage.open();
}
@When("^Open form to input")
public void site_is_opened() {
    mainpage.EntryField();
}
@And("^Input personal data")
public void input_data(){
    mainpage.login("nvikamail@mail.ru", "Vs458474");
}
@Then("^Signed in")
public void open_site() {
    String  fact_res = "Виктория";
    Assert.assertTrue(mainpage.log_in().equals(fact_res));
```

}

После автоматизированного тестирования были получены следующие результаты:

Рисунок 1 – Графики всех пройденных тестов

Из данного графика можно увидеть, что из 18 тестов удачно прошли 9, а 9 были провалены. Также из правой столбчатой диаграммы видно, что 4 теста пройдены от 1 до 10 секунд, 11 тестов от 10 до 30 секунд и 3 теста от 30 до 60 секунд.

Выбрав вкладку Test Results, можно увидеть какие именно тесты прошли, а какие провалены, которые представлены на рисунке 2.

feature	Scenario	Steps	Started	Duration	Result
Login	Mark a task as completed	3	02:52:43	51s 913ms	✓
Open site	Open website with desired title	3	07:22:32	5s 920ms	✓
Open site	Mark a task as completed	4	07:22:39	16s 611ms	✗
Open site	Registration	3	07:22:56	6s 651ms	✗
Open site	Feedback 01	3	07:23:04	6s 818ms	✓
Open site	Feedback 02	3	07:23:11	7s 088ms	✗
Open site	Site Search	3	23:40:19	11s 413ms	✗
Search no bug	Site Search_01	3	07:22:05	11s 968ms	✓
Search no bug	Site Search_02	3	07:22:18	13s 726ms	✓
Search no bug	Site Search	3	23:38:20	37s 011ms	✗

Showing 1 to 10 of 16 entries

Previous 1 2 Next

Рисунок 2 – Список тестов

Каждый сценарий можно открыть и посмотреть скриншоты каждого шага и причину его «провала». Рассмотрим 2 теста один, который прошел и другой провальный.

Первый тест на проверку открытие страницы с выбранной категорией товар. Результат первого теста на рисунке 3.

Рисунок 3 – Успешно пройденный тест-кейс

В данном тесте используется Scenario Outline, который позволяет запускать сразу несколько тестов с разными значениями. В каждом из них есть возможность раскрыть подробности шагов прохождения и их скриншоты, а также состояние и время прохождение каждого шага. Результаты второго теста на рисунке 4.

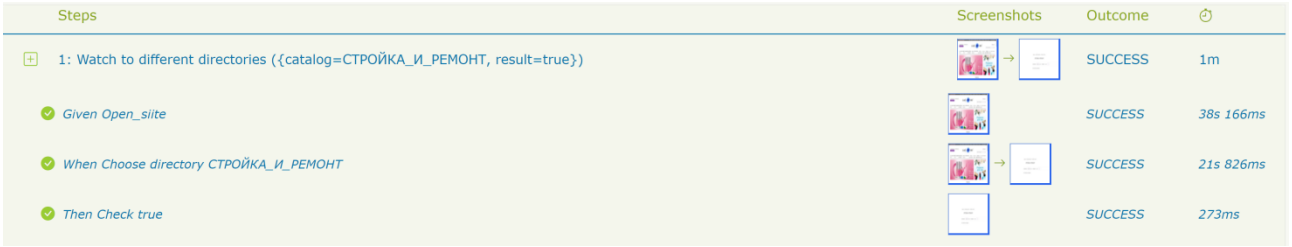

Рисунок 4 – Один из успешно пройденных тестов

В тесте, который оказался провален, получается другая картина. Здесь тот шаг, который завершен неудачно, будет подсвечен красным цветом и выведена причина неудачи. Результат провального теста на рисунке 5.

Рисунок 5 – Провальный тест с указанием ошибки

Данная ошибка означает, что полученный результат не соответствует ожидаемому результату.

Подводя итог второго раздела, можно сделать вывод, что при его написании были разработаны и реализованы автоматизированные тесты с помощью методологии BDD. Также после проверки функциональности пользовательского интерфейса был создан отчет, в котором содержатся детали проверок данных тестов.

ЗАКЛЮЧЕНИЕ

В ходе данной выпускной квалификационной работы были решены все поставленные задачи и цель достигнута. Были изучены теоретические основы методологии BDD, различные инструменты для облегчения работы с методологией BDD. На практике была освоена методология BDD. Приложение было протестировано с помощью фреймворка Serenity и программного инструмента Cucumber для языка программирования Java. После составления автоматических тестов были сформированы отчеты по результатам их прохождения. В ходе практической части было реализовано 18 тестов. По результатам тестирования было пройдено 9 тестов и 9 тестов были провалены. Информация о найденных дефектах были переданы разработчику с рекомендацией и пожеланием исправить найденные дефекты.

Как показала практика, инструмент тестирования Cucumber, который использует Gherkin-нотации довольно несложен в использовании. И может использоваться как для начинающих специалистов, так и для опытных. Также данный инструмент очень часто используется на крупных проектах.

В будущем продолжать работать с автоматизированным тестированием. И создать свой собственный фреймворк, проверяющий пользовательский интерфейс.

Основные источники информации:

1. Трубанов Вадим. Автоматизация по методологии BDD. Наш опыт успешного внедрения. – Блог компании Тинькофф. 2017г.
2. Филькин М.С., Ручкин В.Н. Построение эффективного процесса автоматизации тестирования web-приложений на основе подхода BDD. – Научные исследования XXI века. 2020. № 6 (8). С. 124-128.
3. Куликов, С. С. Тестирование программного обеспечения. Базовый курс / С. С. Куликов. — Минск: Четыре четверти, 2021г. — 300 с.
4. Мартин Р. Чистый код: создание, анализ и рефакторинг. Библиотека программиста. — СПб.: Питер, 2019. — 464 с.
5. David Chelimsky. The RSpec Book Behaviour-Driven Development with RSpec, Cucumber, and Friends. — The United States of America, 2012г, 399с.
6. Seb Rose, Matt Wynne and Aslak Hellesoy. The Cucumber for Java Book — Dallas, Texas • Raleigh, North Carolina, 2015г, 325.