МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н.Г.ЧЕРНЫШЕВСКОГО»**

Кафедра Математического и компьютерного моделирования

Разработка фреймворка для автоматизации тестирования веб-приложений и сервисов

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студентки 4 курса 441 группы

направление 09.03.03 — Прикладная информатика

механико-математического факультета

Панфиловой Анастасии Алексеевны

Научный руководитель
доцент, к.ф. – м.н., С. В. Иванов

Зав. кафедрой
зав. каф., д.ф.-м.н., доцент Ю.А. Блинков

Саратов 2022

Введение. Тестирование программного обеспечения — процесс исследования, испытания программного продукта, имеющий своей целью проверку соответствия между реальным поведением программы и её ожидаемым поведением на конечном наборе тестов, выбранных определённым образом.

Контроль качества ПО — «это планомерная и систематичная программа действий, призванная гарантировать, что система обладает желательными характеристиками».

Тестирование ПО является важной частью процесса контроля качества, так как позволяет проверить и оценить соответствие разработанного ПО требованиям к его функциональности.

Цель автоматизации тестирования – уменьшить количество тестовых примеров, которые нужно запускать вручную, а не полностью исключить ручное тестирование. Автоматизация тестирования является актуальной задачей для компаний в сфере разработки программного обеспечения, так как позволяет сократить время на проведение тестирования, а также количество необходимых для проведения тестирования специалистов, что, несомненно, способствует увеличению экономической эффективности IT-компаний.

При этом остается проблема вовлечения в работу тестировщиков-автоматизаторов, так как для разработки автоматизированных тестов необходимо знать языки программирования, чего у ручных тестировщиков в большинстве случаев не хватает, так же имеется высокий порог вхождения в данную область в виду необходимости знаний в проведении ревью кода и разработки «качественного» кода. Также проблемой является трудоемкость ручного тестирования. Для упрощения вхождения и всего процесса разработки могут быть разработаны специальные инструменты – фреймворки автоматизированного тестирования, дающие возможность сосредоточиться на конкретной задаче, без разработки вспомогательного инструментария, без планирования и разработки архитектуры проекта и так же без трат времени на поддержание и обновление разработанной архитектуры.

Фреймворк (англ. framework — «каркас, структура») — готовая модель в IT, заготовка, шаблон для программной платформы, на основе которого можно дописать собственный код.

Объектом исследования является обеспечение качества и надежности программного обеспечения.

Предметом исследования является инструмент – фреймворк для тестирования, служащий для обеспечения удобства, качества и скорости при разработке автоматизированных тестов для широкого круга специалистов тестирования.

Целью работы является разработка фреймворка для автоматизации тестирования веб-приложений и сервисов.

Структура работы. Основная часть состоит из 3 разделов:

- Обзор существующих программных решений и анализ предметной области;
- Выбор необходимого программного обеспечения;
- Этапы разработки программного обеспечения.

В первом разделе собрана информация о фреймворках и об особенностях их использования для выбранной предметной области.

Необходимо изучить процессы в предметной области. Фреймворк – это сложная структура со множеством модулей, некоторые из них: модуль для работы с API, модуль для работы с UI, модуль для разработки тестов, модуль для работы с базами данных. Полное функционирование любого проекта по разработке ПО уже не существует без тестирования, которое помогает выявлять ошибки прямо на стадии разработки и таким образом улучшая конечный продукт. Фреймворк необходим для упрощения развертывания среды тестирования и ускорения работы тестирования продукта, что существенно уменьшает количество человеко-часов на подготовку и помогает быстрее начать работу разработку самих тестов и быстрее начать выявлять проблемы с ПО на самых ранних этапах.

Рассмотренные в первом разделе фреймворки имеют как плюсы, так и минусы, использовать данные решения для работы над проектами в автоматизации тестирования не представляется возможным, в виду отсутствия ключевых модулей, в связи с этим и возникла необходимость в разработке собственного фреймворка для автоматизации тестирования.

Второй раздел относится к выбору необходимого программного обеспечения для разработки фреймворка. В данной работе, основным языком разработки фреймворка стала Java.

Java – это строго типизированный объектно-ориентированный язык программирования общего назначения, разработанный компанией Sun Microsystems (в последующем приобретённой компанией Oracle).

Приложения Java транслируются в специальный байт-код, поэтому они могут работать на любой компьютерной архитектуре, для которой существует реализация виртуальной Java-машины, то есть приложения, разработанные на языке Java, могут работать в Windows, Unix, Solaris OS и других.

Java – достаточно простой язык для обучения, и в тоже время очень мощный для разработки огромных систем, по этому он и получил широкое распространение во всем мире. Тем не менее с помощью него можно решать задачи разной сложности.

Средой разработки была выбрана IntelliJ IDEA.

IntelliJ IDEA - интегрированная среда разработки программного обеспечения для многих языков программирования, в частности Java, JavaScript, Python, разработанная компанией JetBrains.

Другими словами, IntelliJ IDEA – это интеллектуальный редактор для Java, Python, JavaScript, CSS и многих других языков с подсветкой кода, автоматической проверкой кода «на лету» и встроенными подсказками для упрощения разработки.

Эта среда разработки предоставляет нам следующие возможности:

- Возможность интеграции с системами управления версиями Git, позволяющая совершать многие действия, например commit, merge и другие, прямо из IDE;
- Поддержка подключения к SQL, тем самым упрощая создание запросов в базу автодополнением кода самой IDE;
- Отслеживание любых изменений в коде (LocalHistory);
- Возможность автодополнения, проверка стандартных тегов, свойств;

- Использование различные сочетаний клавиш для повышения эффективности.

Веб хостингом проекта с фреймворком был выбран GitLab, по причине наличия большого количества удобных систем работы с ним.

GitLab имеет множество возможностей, основные из них представлены ниже.

Планирование: GitLab способен эффективно поддерживать различные модели коллективной работы вне зависимости от выбранной методологии разработки. Гибкие инструменты управления проектами GitLab позволяют делать процесс разработки наглядным, координировать его, отслеживать и назначать приоритеты.

Создание: С Gitlab команда разработчиков или тестировщиков может консолидировать исходный код в общей распределенной среде контроля версий. Веб-сервис позволяет управлять и поддерживать распределенную среду, не нарушая процессы разработки.

GitLab имеет целый арсенал инструментов для управления ветками и доступом к проектам, создавая общую достоверную среду для совместной работы команды.

Тестирование: В GitLab реализованы инструменты ревью кода, его тестирования и оценки качества, что позволяет разработчикам или тестировщикам быстрее находить ошибки и сокращать цикл их исправления.

Можно персонально настраивать модель приемки качества, тестировать код в автоматическим режиме и назначать изменения в среды тестирования для каждой версии кода.

Сборка: Репозиторий контейнеров GitLab дает возможность создавать безопасное хранилище кастомных образов контейнеров Docker. Причем для этого не придется задействовать дополнительные инструменты — возможности скачивания и загрузки образов внедрены в среду управления репозиторием Git по умолчанию.

Релиз: Компоненты поддержки технологий непрерывной доставки и развертывания позволяют эффективно автоматизировать операции, связанные со сборкой, автоматическим тестированием и установкой релизов. Установка

релиза как на один сервер, так и на множество, будет занимать минимум времени.

Конфигурирование: GitLab позволяет автоматизировать весь процесс разработки приложения. Для этого предоставляются готовые шаблоны моделей, с которыми начать работу можно без сложных предварительных настроек — достаточно добавить специфику приложения на каждом этапе сборки и развертывания.

В качестве сервиса с предварительно настроенными шаблонами приложений для разработки можно использовать GitLab CE Virtual Appliance.

Мониторинг: С GitLab можно отслеживать время, затраченное на каждый этап, проверять работоспособность приложения, собирать и просматривать метрики, а также анализировать, как изменения кода влияют на производительность среды.

Было принято решение выбрать именно его, так как в GitLab имеется множество интересных и удобных систем.

Вспомогательными системами для работы фреймворка были выбраны: Apache Maven, Selenium WebDriver, Apache Kafka, TestNG, REST Assured, Log4j, Apache Cassandra, Amazon S3, PostgreSQL, MySQL, Lombok, Allure Report.

Автоматическая сборка приложения особенно важна на этапах разработки, отладки и тестирования — Maven помогает собрать код и ресурсы в исполняемое приложение без IDE (среды разработки). При этом система сборки отличается гибкостью:

- Может использоваться в IDE — Eclipse, IntelliJ IDEA, NetBeans и других;
- Не зависит от операционной системы;
- Не требует установки — архив с программой можно распаковать в любой директории;
- Все необходимые параметры имеют оптимальные настройки по умолчанию;
- Упрощает организацию командной работы и документирование;

- Запускает библиотеки для модульного тестирования;
- Обеспечивает соблюдение стандартов;
- Имеет огромное количество плагинов и расширений.

Также существуют две другие системы сборки Java-приложений — Ant и Gradle, однако Maven пользуется наибольшей популярностью и является стандартом индустрии.

Выберем Apache Maven, так как с помощью него можно легко собирать проект, подключать библиотеки, следить за версиями подключаемых библиотек, а также их связывать.

Selenium WebDriver — это инструмент, который позволяет производить кросс-браузерное тестирование, то есть проверять, как отображается сайт в разных браузерах.

Selenium WebDriver — главный вектор развития Selenium. Вот его основные преимущества:

- Поддерживает разные языки программирования (Java, C, PHP, Ruby, Perl, Python), а значит, его может использовать большое количество разработчиков;
- Облегчает кроссбраузерное тестирование и поддерживает различные браузеры: Firefox, Opera, Chrome, Edge;
- Позволяет проводить параллельное тестирование в нескольких браузерах одновременно;
- Имеет открытый исходный код, он бесплатный для любого разработчика;
- Большое сообщество пользователей — при возникновении трудностей в работе есть у кого попросить помощи.

Selenium WebDriver также имеет и ограничения:

- Для работы с WebDriver нужны продвинутые навыки программирования;
- Это узконаправленный инструмент, с его помощью можно тестировать только веб-приложения;

- Функционал ограничен по сравнению с платными аналогами (такими как Ranorex Studio или UBot Studio).

Выберем Selenium для возможности разработки UI автотестов, так как данный фреймворк является ядром для других фреймворков по работе с UI, например, Selenide, но при этом у данных фреймворков есть еще свои надстройки над Selenium, которые дают дополнительную нагрузку при исполнении кода.

Apache Kafka — распределённый программный брокер сообщений.

Сервисы между собой общаются с помощью Apache Kafka посредством отправки сообщений в топики Apache Kafka, где они складируются, пока другой сервис их не вычитает, для модульного тестирования и тестирования API сервисов необходимо напрямую класть данные сообщения в топики Apache Kafka, для этого добавим возможность работы с ней во фреймворк.

TestNG — система автоматического тестирования с открытым исходным кодом.

Для запуска тестов мало самой Java, необходим еще фреймворк, который будет работать с конфигурациями тестов, их запуском и генерацией результатов теста для отчета, для этого хорошо подойдет TestNG. У TestNG также есть аналог в лице JUnit, но JUnit хуже по таким показателям:

- Невозможность выполнить тестирование зависимостей (в TestNG существуют depenOnMethods и depenOnGroups, они нужны для запуска тестов в определенном порядке);
- Плохое управление конфигурацией (setUp / tearDown);
- Статическая модель программирования (вынуждает перекомпилировать код без необходимости);
- Управление различными наборами тестов в сложных проектах может быть очень сложным;
- Плохая работа с многопоточностью (плохо разделяет тесты на несколько потоков и более низкая скорость работы в многопотоке по сравнению с TestNG).

REST Assured — DSL для тестирования REST API и SOAP API сервисов, который встраивается в тесты на Java.

Log4j — библиотека журналирования (логирования) Java-программ, часть общего проекта «Apache Logging Project» [**?**].

Log4j – это надежная, быстрая и гибкая среда ведения журналов (API), написанная на Java, которая распространяется по лицензии Apache Software License.

Log4j, из-за его плюсов, даже был портирован на языки C, C ++, C , Perl, Python, Ruby и Eiffel.

Log4j состоит из трех основных компонентов:

- Регистраторы - Ответственный за сбор информации журнала;
- Appenders - Отвечает за публикацию информации о регистрации в различных предпочтительных местах назначения;
- Макеты - Отвечает за форматирование информации журнала в разных стилях.

Особенности Log4j:

- Потокобезопасен;
- Оптимизировано для обеспечения высокой скорости работы;
- Основан на именованной иерархии логгеров;
- Поддерживает несколько выходных приложений для каждого регистратора;
- Поддерживает интернационализацию;
- Не ограничено предопределенным набором средств;
- Поведение ведения журнала может быть установлено во время выполнения с помощью конфигурации;
- Предназначен для обработки исключений Java;
- Использует несколько уровней, а именно ALL, TRACE, DEBUG, INFO, WARN, ERROR и FATAL;
- Формат вывода журнала можно легко изменить, расширив класс Layout;

- Цель вывода журнала, а также стратегия записи могут быть изменены реализациями интерфейса Appender.

Для любого проекта необходимо логирование для более быстрого и простого выявления проблем, библиотека Log4j отлично справляется с данной задачей, выберем её.

Apache Cassandra — распределённая система управления базами данных, относящаяся к классу NoSQL-систем и рассчитанная на создание высокомасштабируемых и надёжных хранилищ огромных массивов данных, представленных в виде хэша.

Apache Cassandra, в отличие от других БД, требует другого вида подключения и работы с ней, поэтому для нее необходим отдельный модуль в фреймворке.

Возьмем драйвер Apache Cassandra с общего репозитория Maven и примеры работы с сайта компании Datastax.

Amazon Simple Storage Service (Amazon S3) – это объектное хранилище, рассчитанное на хранение и извлечение любых объемов данных из любой точки сети. Это простой сервис хранилища, который отличается самой высокой надежностью, доступностью, производительностью и безопасностью в отрасли, а также практически неограниченной масштабируемостью при очень низких затратах.

PostgreSQL и MySQL — это свободные объектно-реляционные системы управления базами данных.

Для них используется разные драйвера подключения, но сам вид и тип подключения одинаков, точно так же в будущем можно будет легко добавить MongoDB и другие базы данных путем подключения только одного драйвера.

Lombok — это плагин компилятора, который добавляет в Java новые «ключевые слова» и превращает аннотации в Java-код, уменьшая усилия на разработку и обеспечивая некоторую дополнительную функциональность.

Lombok преобразует аннотации в исходном коде в Java-операторы до того, как компилятор их обработает: зависимость lombok отсутствует в рантайме, поэтому использование плагина не увеличит размер сборки.

Lombok помогает использовать POJO более гибким и структурированным образом без дополнительного кода. Например: @Data — просто удобная аннотация, которая применяет сразу несколько аннотаций Lombok.

- @ToString - генерирует реализацию для метода toString(), которая состоит из аккуратного представления объекта: имя класса, все поля и их значения;
- @EqualsAndHashCode - генерирует реализации equals и hashCode, которые по умолчанию используют нестатические и нестационарные поля, но настраиваются;
- @Getter/@Setter - генерирует геттеры и сеттеры для частных полей;
- @RequiredArgsConstructor - создаёт конструктор с требуемыми аргументами, где обязательными являются окончательные поля и поля с аннотацией @NonNull.

Используя Lombok, можно существенно сократить количество строк кода и увеличить скорость разработки, так как вместо нескольких строк кода можно добавить всего одну Java аннотацию и Lombok сгенерирует код во время компиляции.

Allure Report — фреймворк от компании Яндекс для создания простых и понятных отчётов автотестов.

Allure Report генерирует подробный отчет о прохождении тестов, сопровождая его графиками, описаниями ошибок, возможностью фильтрации результатов, приложенными к тесту скриншотами, входными и выходными данными.

В третьем разделе описаны этапы разработки фреймворка средствами UML и подробное описание функционала каждого модуля системы.

Начало третьего раздела разделяется на теоретическую часть, в которой описываются особенности языка UML и предоставляются данные необходимые для работы с предметной областью и на часть, в которой раздел по-

священ особенности построения диаграммы прецедентов (use case diagram) и диаграммы классов (class diagram). Остальная часть раздела описывает структуру фреймворка, о каждой функциональной его части, а также в ней приведены примеры структур проектов без использования данного фреймворка и с его использованием, в которых наглядно видна разница в количестве кода, написанного тестировщиком для начала работы над проектом и поддержке данного кода в будущем.

В приложениях представлены исходные программные коды реализации.

Заключение. К одному из распространенных и необходимых для бизнеса сегодня типов информационных систем относится фреймворк для автоматизации тестирования веб-приложений и сервисов. С помощью разработанного фреймворка, тестировщики смогут быстро развертывать архитектуру нового проекта, начать разрабатывать автотесты в кратчайшие сроки, сильно уменьшить время разработки самих автотестов, а также упростить их поддержку.