Санкт-Петербургский государственный университет

Фомина Виктория Викторовна

Выпускная квалификационная работа

Создание облачного сервиса для автоматической генерации юнит-тестов в языках Java, C, C++ и измерения тестового покрытия

Уровень образования: бакалавриат

Направление *02.03.03 «Математическое обеспечение и администрирование информационных систем»*

Основная образовательная программа *СВ.5006.2018 «Математическое обеспечение и администрирование информационных систем»*

Профиль *Системное программирование*

Научный руководитель:
доцент кафедры системного программирования, к.т.н. Ю. В. Литвинов

Консультант:
руководитель департамента анализа программ ООО «Техкомпания Хуавэй» Д. А. Иванов

Рецензент:
инженер ключевых проектов ООО «Техкомпания Хуавэй» Е. К. Куликов

Санкт-Петербург
2022

Saint Petersburg State University

Viktoriia Fomina

Bachelor's Thesis

Creation of a cloud service for automatic generation of unit tests in Java, C, C++ languages and measurement of test coverage

Education level: bachelor

Speciality *02.03.03 "Software and Administration of Information Systems"*

Programme *CB.5006.2018 "Software and Administration of Information Systems"*

Profile: *Software Engineering*

Scientific supervisor:
C.Sc, docent Y.V. Litvinov

Consultant:
head of program analysis department at LLC Tech Company Huawei D.A. Ivanov

Reviewer:
key project engineer at LLC Tech Company Huawei E.K. Kulikov

Saint Petersburg
2022

Оглавление

Введение

Качество покрытия кода юнит-тестами является критичным для качества конечного программного продукта. Любая современная IDE, например, IntelliJ IDEA [11] или Visual Studio [2] умеет отображать информацию о тестовом покрытии. Даже существуют облачные инструменты вроде Codecov [30], которые умеют анализировать и удобно отображать такую информацию. Трудно покрывать тестами пути исполнения, что ожидаемо, ведь даже программа из 100 строк кода может содержать огромное количество путей [31], с. 44. Поскольку покрывать пути исполнения тестами вручную слишком сложно, существуют символьные исполнители, решающие эту задачу. Они могут статически анализировать покрытие кода, а также генерировать тесты, покрывающие пути исполнения, а не строки. Хотя на программе из книги Сэма Канера хороший результат покрытия тестами с помощью такого подхода получить не удастся, он все же может быть эффективным для практических целей.

Одна из компаний, занимающихся разработкой систем для автоматической генерации тестов – компания «Хуавэй». «Хуавэй» предлагает разработчикам такие продукты как UTBot Java и UTBot C/C++ – движки-генераторы тестов, основанные на символьном исполнении. На основе этих движков разработаны плагины для IDE, а также консольные приложения UTBot C/C++ CLI и UTBot Java CLI. Но эти утилиты не предоставляют возможность мгновенно оценить качество сгенерированных UTBot[1] тестов, потому что требуют предварительной установки. Более того, процесс автоматической генерации тестов занимает большое количество оперативной памяти и очень нагружает процессор, поэтому на недостаточно мощных машинах может занимать много времени.

В связи с этим компания «Хуавэй» начала разработку веб-приложения для автоматической генерации юнит-тестов в языках Java, C, C++ и измерения тестового покрытия. Сейчас сервис развернут внутри ло-

[1]UTBot – UTBot Java и UTBot C/C++

кальной сети «Хуавэй». Он состоит из 2 частей: серверной (UTBot Online) и клиентской (UTBot Site) и предоставляет возможность сгенерировать и запустить тесты на C, сгенерировать тесты на Java. Появилась необходимость в расширении функциональных возможностей веб-приложения, автоматизации многих процессов в нем и развертывании его на Huawei Cloud[2]. Это и стало целью работы.

[2]Huawei Cloud – коммерческое публичное облако, поддерживаемое и развиваемое компанией Huawei

Постановка задачи

Цель работы – создать облачный сервис для автоматической генерации юнит-тестов в языках Java, C, C++ и измерения тестового покрытия.

Для достижения цели были сформулированы следующие задачи.

1. Провести обзор существующих решений.
2. Настроить CI/CD в UTBot Cloud.
3. Продумать и реализовать архитектуру, позволяющую запускать в UTBot Online сгенерированные тесты в изоляции.
4. Реализовать в UTBot Java CLI и интегрировать в UTBot Cloud функциональность, позволяющую запускать тесты на Java.
5. Реализовать в UTBot Cloud генерацию и запуск тестов для кода на языке C++.
6. Развернуть UTBot Online на Huawei Cloud.
7. Провести апробацию и тестирование.

1. Обзор

Для того, чтобы провести сравнение UTBot Cloud с аналогами, необходимо было понять, что представляет собой UTBot Cloud.

1.1. UTBot Cloud на момент начала работы

UTBot Cloud – веб-приложение, развернутое внутри локальной сети компании «Хуавэй» и позволяющее:

- генерировать тесты для кода на языках C и Java;
- получать информацию о запуске тестов для кода на языке C;
- измерять количество покрытых тестами строк для кода на языке C.

UTBot Cloud состоит из двух частей:

- UTBot Site – клиентская часть;
- UTBot Online – серверная часть.

UTBot Cloud для генерации тестов, получении информации о запуске тестов, генерации информации о количестве покрытых тестами строк кода на языке C использует консольное приложение UTBot C/C++ CLI. Это приложение создает абстракцию для работы с движком UTBot C/C++, генерирующим тесты.

Для генерации тестов на языке Java используется консольное приложение UTBot Java CLI. Оно создает абстракцию для работы с генерирующим тесты движком UTBot Java.

1.2. Обзор существующих решений

UTBot Cloud значительно отличается от других систем, позволяющих автоматически генерировать тесты (Рис. 1). Он позволяет генерировать тесты только по коду из одного файла. Если тестируемый

код на языке Java зависит от классов, определенных в других файлах, то UTBot Cloud не сможет сгенерировать тесты по такому коду. Это делает его не таким функциональным, как плагины для IDE вроде Diffblue Cover[3] [3], отдельные экосистемы со своими IDE вроде Parasoft C/C++test [19], Parasoft JTest [20], консольные приложения вроде Diffblue Cover CLI или EvoSuite CLI [28].

UTBot Cloud небходимо рассматривать в связке с UTBot C/C++ [7] и UTBot Java [29], которые предоставляют в том числе и плагины для IDE, более функциональные, нежели UTBot Cloud. Тем не менее UTBot Cloud дает возможность разработчику зайти на сайт, вставить кусок кода, сгенерировать к нему тесты, получить результат запуска этих тестов и отчет о тестовом покрытии, ничего при этом не скачивая и не устанавливая. Для независимых кусков кода или небольших задач, логика которых может быть помещена в один файл, UTBot Cloud может быть использован, и результат от такого использования может быть получен в один клик. Если разработчик будет удовлетворен качеством сгенерированных UTBot Cloud тестов, но захочет иметь возможность генерировать тесты для больших проектов, он сможет поставить себе соответствующие плагины для IntelliJ IDEA[4] или VS Code[5].

Стоит отметить тот факт, что UTBot Cloud является бесплатным и не имеет ограничений на количество генерируемых тестов, в то время как Diffblue Cover бесплатно позволяет генерировать не более 200 тестов в день. А все продукты Parasoft являются платными и имеют такую высокую стоимость, что доступны только крупным IT-компаниям.

Существует бесплатный продукт EvoSuite. EvoSuite предлагает консольное приложение, плагин для InetelliJ Idea, а также Maven плагин. Но все эти утилиты требуют установки в отличие от UTBot Cloud.

[3]Diffblue Cover – плагин для IntelliJ IDEA для генерации юнит-тестов к коду на языке Java

[4]плагин на основе движка UTBot Java

[5]плагин на основе движка UTBot C/C++

	UTBot Cloud	Parasoft C/C++test	Parasoft Jtest	Diffblue Cover	EvoSuite
Бесплатность	✔	✘	✘	✔	✔
Крупные проекты	✘	✔	✔	✔	✔
Тестовое покрытие	✔	✔	✔	✔	✔
Использование в один клик	✔	✘	✘	✘	✘
Поддерживаемые языки	C/C++/Java	C/C++	Java	Java	Java

Рис. 1: Сравнение с аналогами

2. Используемые технологии

В начале разработки UTBot Cloud было принято решение использовать современные технологии. Это обусловлено нежеланием в будущем сталкиваться с проблемами, связанными с поиском специалистов, владеющих технологиями.

2.1. Технологии, используемые в UTBot Online

- Java [14]

  Java – язык разработки серверной части приложения, занимает 3-е место среди самых популярных языков программирования по версии индекса TIOBE[6] [26] на момент марта 2022.

- Spring Framework [23]

  Spring Framework представляет собой контейнер внедрения зависимостей (IoC[7]), что позволяет быстро и удобно создавать Java-приложения.

  Учитывая тот факт, что серверная часть UTBot Cloud бы-

[6]Индекс TIOBE оценивает популярность языков программирования на основе поисковых запросов в различных поисковых системах.

[7]Inversion of Control (IoC) – процесс, в соответствии с которым объекты определяют свои зависимости (объекты, с которыми они работают) через аргументы конструктора/фабричного метода или свойства, установленные или возвращенные фабричным методом.

ла запланирована небольшой по размеру, то для ее реализации вполне мог был быть выбран другой фреймворк и другой язык программирования. Одной из причин выбора Java и Spring послужило наличие в команде Cloud BU[8] отдельной Java-команды, знакомой с этими технологиями. В процессе разработки есть возможность консультироваться Java-командой, что ускоряет разработку.

- Docker [4]

  Docker – контейнеризатор приложений, позволяющий автоматизировать процесс развертывания и управления приложениями в средах с поддержкой контейнеризации.

- CI (GitHub Actions) [9]

  GitHub Actions – CI/CD платформа, внедренная в GitHub[9] и позволяющая автоматизировать процесс сборки, тестирования и развертывания. Удобна для использования в проекте, так как исходные коды UTBot Site и UTBot Online хранятся на GitHub.

2.2. Технологии, используемые в UTBot Site

- Gatsby [8]

  Gatsby – React-фреймворк с открытым исходным кодом [22]. За счет того, что Gatsby основан на React, разработчик имеет возможность использовать пакеты, поставляемый nmp – менеджером пакетов, входящим в состав Node.js[10].
  Gatsby генерирует статические сайты. Это предоставляет несколько возможностей:

  * позволяет не создавать страницу по каждому запросу, а возвращать уже однажды скомпилированную;

[8]Команда внутри компании «Хуавей», занимающаяся разработкой продуктов UTBot.
[9]GitHub – основанный на Git веб-сервис для хостинга и совместной разработки IT-проектов.
[10]Node.js – среда выполнения кода на Javascript с открытым исходным кодом.

* позволяет разместить сайт на GitHub Pages[1] или похожих ресурсах, предоставляющих возможность хранить статические сайты. Сервисы такого рода позволяют подключить собственный домен и опубликовать сайт примерно за минуту.

В Gatsby нет необходимости создавать Route для компонентов в папке страниц – страница генерируется автоматически. Также Gatsby предоставляет API для написания плагинов и имеет много уже написанных плагинов, удовлетворяющих разным целям.

- Javascript [15]

  Javascript – один из двух языков программирования, доступных внутри фреймворка Gatsby. Второй доступный язык – Typescript.

  Выбор в качестве языка разработки был сделан в пользу Javascript, так как ранее и на момент марта 2022 по версии индекса TIOBE этот язык являлся более популярным нежели Typescript. Использование современных технологий – одно из требований, которые необходимо соблюдать.

  Стоит учесть тот факт, что UTBot Site – небольшое приложение, где почти вся содержательная логика сосредоточена на одной странице. На таком небольшом приложении отсутствие одного из основных преимуществ Typescript – статической типизации, не будет оказывать заметного влияния на скорость и качество разработки.

- Docz [5]

  Docz – плагин для React-фреймворка Gatsby, позволяющий генерировать документацию, в том числе и по Markdown файлам. Docz позволяет на странице из будущей документации указать специальный заголовок, отвечающий за путь по которому будет отображена страница. Файл doczrc.js предоставляет возможность легко конфигуриро-

вать порядок ссылок на страницы документации в боковом меню.

- Python [21]

  Python – самый популярный язык программирования по версии индекса TIOBE на момент марта 2022. Удобен в качестве скриптового языка, так как в простых сценариях использования позволяет последовательно записывать команды, не требуя дополнительно объявлять функцию main и т.д. При этом существует много библиотек для Python, удобных для разных целей.

- CI (GitHub Actions) [9]

  GitHub Actions – CI/CD платформа, внедренная в GitHub[11] и позволяющая автоматизировать процесс сборки, тестирования и развертывания. Удобна для использования в проекте, так как исходные коды UTBot Site и UTBot Online хранятся на GitHub.

2.3. Технологии, используемые в UTBot Java CLI

- Kotlin [17]

  Kotlin – статически типизированный язык программирования, работающий поверх JVM[12]. За счет большого количества «синтаксического сахара» конструкции на Kotlin получаются достаточно лаконичными. Это позволяет в среднем писать меньше кода, чем на Java.

- Clikt [10]

  Clikt – Kotlin-библиотека, позволяющая удобно писать интерфейсы командной строки.

- JaCoCo [25]

[11]GitHub – основанный на Git веб-сервис для хостинга и совместной разработки IT-проектов.
[12]JVM – Java Virtual Machine

JaCoCo – Java-библиотека для измерения качества покрытия кода тестами.

3. Реализация

В рамках работы над UTBot Cloud встала необходимость в настройке CI/CD, реализации запуска тестов на Java в UTBot Java CLI с последующей интеграцией этой функциональности в UTBot Online, а также в развертывании UTBot Online на Huawei Cloud.

3.1. Настройка CI/CD в UTBot Cloud

Развертывание UTBot Site и UTBot Online требовало выполнения большого количества действий. Необходимо было автоматизировать эти действия для автоматизации процессов развертывания в UTBot Cloud.

3.1.1. Автоматизация процесса развертывания UTBot Site

Документация в UTBot Site – документация с GitHub Wiki[13], написана на языке разметки Markdown. Раньше обновление документации в UTBot Site было трудоемким процессом (Рис. 2). Этот процесс требовал выполнения нескольких шагов.

1. Скачивания документации с Wiki.
2. Помещения ее в директорию внутри проекта UTBot Site.
3. Экранирования некоторых символов и удаления HTML-тегов, не распознаваемых Docz.
4. Добавления специальных заголовков, отвечающих за то по какому пути будет отображена страница.
5. Обновления файла doczrc.js[14] в соответствии с изменениями документации по отношению к предыдущей версии.

Автор работы автоматизировал этот процесс, написав соответствующие Python и Shell-скрипты, вызываемые при сборке или локальном развертывании UTBot Site.

[13]https://github.com/UnitTestBot/UTBotCpp/wiki (дата обращения: 09.04.2022)

[14]doczrc.js – файл, конфигурирующий порядок ссылок на страницы документации в боковом меню

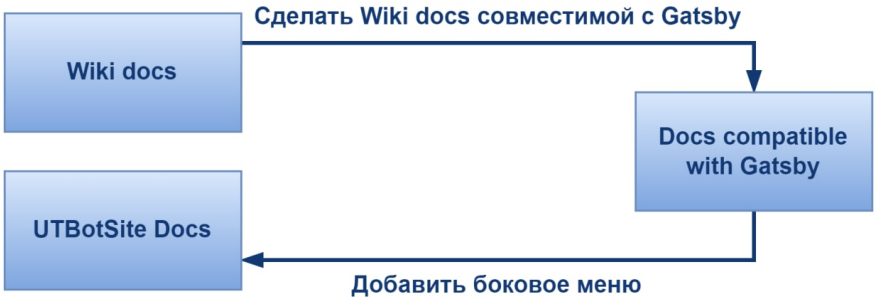

Рис. 2: Процесс преобразования документации с GitHub Wiki в Docz

Также автором был написан скрипт, собирающий UTBot Site и развертывающий его на GitHub Pages в случае успешной сборки.

3.1.2. Автоматизация процесса развертывания UTBot Online

Был написан YML-скрипт, собирающий UTBot Online на GitHub Actions. Результатом сборки UTBot Online служат артефакты, которые могут быть использованы для его развертывания на локальной машине или на Huawei Cloud.

3.2. Запуск в UTBot Online тестов в изоляции

Для простоты повествования будем рассматривать процесс генерации и запуска тестов только для кода на языке C. Для кода на других языках программирования процессы аналогичны.

В тот момент когда автор приступил к проекту, запуск и работа UTBot Online состояли из следующих действий (Рис. 3).

- Запуска docker контейнера, в котором запускалось Spring приложение – UTBot Online.
- Запуска в том же контейнере UTBot C/C++ CLI на любой запрос пользователя на генерацию или запуск тестов.
- Возврата пользователю UTBot'ом Online результатов работы UTBot C/C++ CLI.

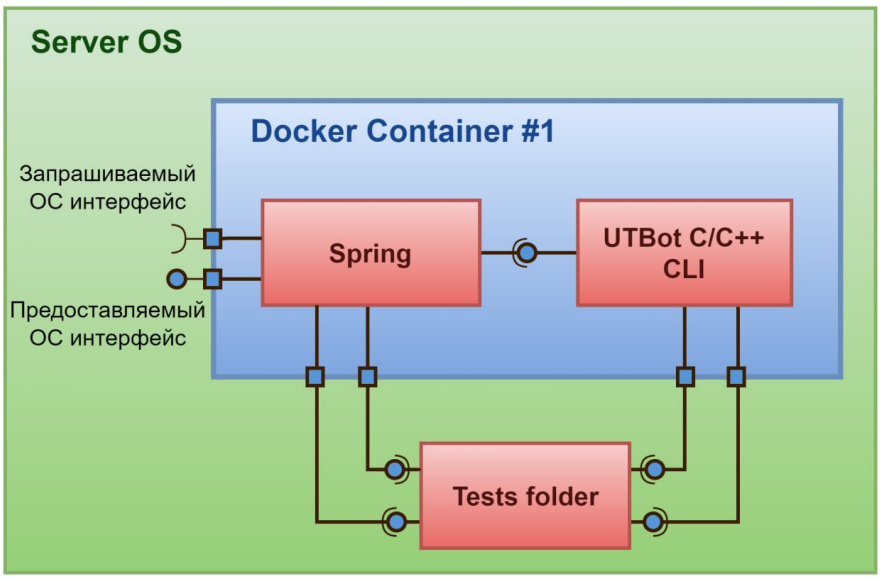

Рис. 3: На момент начала работы

При этом UTBot C/C++ CLI имел доступ ко всем директориям с исходными кодами и тестами к ним.

Это создавало потенциальные проблемы.

- Пользователь мог используя уязвимости UTBot Online получить root доступ к контейнеру и использовать его для того, чтобы «положить» UTBot Online.
- Пользователь мог используя уязвимости UTBot Online подменить тесты другому пользователю.

Автор продумал и реализовал решение, которое устранило эти уязвимости. Теперь запуск и работа UTBot Online состоят из следующих этапов (Рис. 4).

- Запуска docker контейнера, в котором запускается Spring приложение – UTBot Online.
- Запуска в отдельном контейнере UTBot C/C++ CLI на любой запрос пользователя на генерацию и запуск тестов (теперь это один запрос).
- Возврата пользователю UTBot'ом Online результата работы UTBot C/C++ CLI.

При этом UTBot C/C++ CLI имеет доступ только к той директории, с файлами которой подразумевается его непосредственная работа. Таким образом, если пользователь посылает запрос на генерацию и запуск тестов, UTBot Online создает директорию и кладет туда тестируемый код. Только к этой директорию UTBot C/C++ CLI имеет доступ. Генерировать отчеты о генерации и запуске тестов он, соответственно, может только в эту директорию.

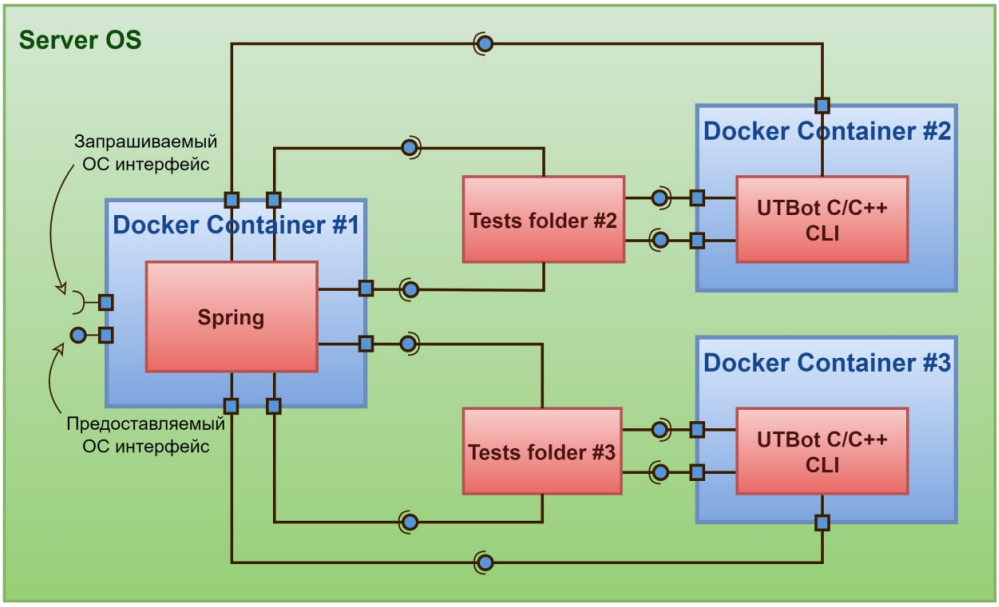

Рис. 4: Решение

3.3. Запуск тестов на Java

На момент начала работы автора над проектом UTBot Online не поддерживал запуск тестов на Java. Необходимо было реализовать эту функциональность.

3.3.1. Реализация в UTBot Java CLI функциональности, позволяющей запускать тесты на Java

Для реализации интерфейса командной строки UTBot Java CLI использует библиотеку Clikt. При помощи этой библиотеки в UTBot Java CLI была добавлена команда *run*, позволяющая запускать тесты на Java

для 3 фреймворков: JUnit 4 [12], JUnit 5 [13], TestNG [27]. Также была добавлена возможность сгенерировать отчет о тестовом покрытии. Для этого необходимо добавить соответствующий ключ к команде *run*.

3.3.2. Интеграция в UTBot Online функциональности, позволяющей запускать тесты на Java

Программный интерфейс, предоставляемый UTBot Java CLI, изменился за последнее время. Текущая версия UTBot Java CLI оказалась несовместимой с UTBot Online. Автор внес изменения в UTBot Online, сделав его совместимым с последней версией UTBot Java CLI.

Также автор реализовал запуск тестов на Java в UTBot Online.

Для выполнения задач потребовалось реализовать класс $JavaProcessor$ с методами *generate* и *run*. Вызов этих методов позволяет сгенерировать и запустить тесты на Java посредством вызова метода *run* класса $ExternalProcessRunner$, запускающего внешние процессы, например, создание docker контейнера или запуск команды внутри docker контейнера.

3.4. Генерация и запуск тестов на C++

Так как UTBot Cloud поддерживал запуск тестов на C, то добавить в него поддержку тестов на языке C++ не было сложной задачей поскольку для генерации тестов для C и C++ используется один и тот же движок.

Сложность заключалась в собирании нового докер образа, в контейнерах которого бы генерировались и запускались тесты для C и C++. Последняя версия UTBot C/C++ некорректно работала в окружении, в котором запускалась для UTBot Cloud: тесты и для C, и для C++ не генерировались и не запускались. Необходимо было определить каких зависимостей не хватает UTBot C/C++ и добавить их в сборку докер образа.

3.5. Развертывание UTBot Online на Huawei Cloud

Для развертывания UTBot Online на Huawei Cloud было куплено 3 домена: *utbot.org*, *unittestbot.org*, *unittestbot.com*.

Для домена *utbot.org* был настроен SSL сертификат.

UTBot Online был развернут на Elastic Cloud Server [6] – облачном сервере, предоставляющем виртуальную машину с выделенными ресурсами.

UTBot Online запускается в докер контейнере. В этом же докер контейнере запускается nginx[15] [18]. При обработке запросов он выполняет действия: 1) Запросы, отправляемые пользователями к UTBot Online, отправляются на хост с IP адресом и портом: 127.0.0.1:10000. 2) Запросы, отправляемые пользователями на UTBot Site, перенаправляются на *www.utbot.org* в случае если были отправлены на *utbot.org*, *unittestbot.org*, *unittestbot.com*.

Соответствующая DNS конфигурация и соответствующая конфигурация на GitHub Pages позволили добиться того, что на сайт можно заходить по своему домену.

[15]веб-сервер и почтовый прокси, работающий под Unix-подобными системами

4. Тестирование и апробация

В рамках выполнения работы необходимо было провести тестирование реализованных функциональных возможностей. Также важно было провести апробацию UTBot Cloud, чтобы понять, удобно ли людям им пользоваться.

4.1. Тестирование

Для тестирования функциональных возможностей, реализованных в UTBot Site, UTBot Online и UTBot Java CLI, автором были написаны юнит-тесты.

Для тестирования UTBot Site был использован Jest [16] – фреймворк для тестирования Javascript. Для тестирования UTBot Online и UTBot Java CLI использовался фреймворк JUnit 5.

4.2. Апробация

Были проведены замеры удобства использования UTBot Cloud. Оценка системы была получена в ходе проведения апробации по шкале удобства использования системы (System Usability Scale) [25, 24]. В процессе апробации приняло участие 8 человек, среди которых студенты математико-механического факультета и разработчики. Система была оценена в 88,125 баллов, что может быть интерпретировано как 5 по пятибалльной шкале.

Заключение

В ходе работы были достигнуты результаты.

1. Проведен обзор существующих решений.
2. Настроен CI/CD в UTBot Cloud.
3. Продумана и реализована архитектура, позволяющая запускать в UTBot Online сгенерированные тесты в изоляции.
4. Реализована в UTBot Java CLI и интегрирована в UTBot Cloud функциональность, позволяющая запускать тесты на Java.
5. В UTBot Cloud реализованы генерация и запуск тестов для кода на языке C++.
6. UTBot Online развернут на Huawei Cloud.
7. Проведена апробация и тестирование.

Исходный код, написанный в рамках работы, открыт и хранится в GitHub репозиториях организации UnitTestBot[16] (аккаунт – victoriafomina).

[16]https://github.com/UnitTestBot – (дата обращения: 27.05.2022)

Список литературы

[1] Clikt. — URL: https://ajalt.github.io/clikt/ (online; accessed: 2022-04-09).

[2] Codecov. — URL: https://about.codecov.io/ (online; accessed: 2022-04-10).

[3] Diffblue Cover. — URL: https://www.diffblue.com/ (online; accessed: 2022-04-08).

[4] Docker. — URL: https://docs.docker.com/ (online; accessed: 2022-04-09).

[5] Docz. — URL: https://www.docz.site/ (online; accessed: 2022-04-09).

[6] Elastic Cloud Server, documentation. — URL: https://support.huaweicloud.com/intl/en-us/productdesc-ecs/en-us_topic_0013771112.html (online; accessed: 2022-04-30).

[7] EvoSuite. — URL: https://www.evosuite.org/ (online; accessed: 2022-04-08).

[8] Gatsby. — URL: https://www.gatsbyjs.com/docs/ (online; accessed: 2022-04-09).

[9] Gihub Actions. — URL: https://docs.github.com/en/actions (online; accessed: 2022-04-09).

[10] GitHub Pages. — URL: https://docs.github.com/en/enterprise-cloud@latest/pages (online; accessed: 2022-04-09).

[11] IntelliJ IDEA. — URL: https://www.jetbrains.com/ru-ru/idea/ (online; accessed: 2022-04-10).

[12] JUnit 4. — URL: https://junit.org/junit4/ (online; accessed: 2022-04-10).

[13] JUnit 5. — URL: https://junit.org/junit5/docs/current/user-guide/ (online; accessed: 2022-04-10).

[14] Java. — URL: https://docs.oracle.com/en/java/ (online; accessed: 2022-04-09).

[15] Javascript. — URL: https://devdocs.io/javascript/ (online; accessed: 2022-04-09).

[16] Jest. — URL: https://jestjs.io/ (online; accessed: 2022-05-27).

[17] Kotlin. — URL: https://kotlinlang.org/docs/home.html (online; accessed: 2022-04-09).

[18] Nginx. — URL: https://nginx.org/ (online; accessed: 2022-05-27).

[19] Parasoft C/C++test. — URL: https://www.parasoft.com/products/parasoft-c-ctest/ (online; accessed: 2022-04-08).

[20] Parasoft Jtest. — URL: https://www.parasoft.com/products/parasoft-jtest/ (online; accessed: 2022-04-08).

[21] Python. — URL: https://docs.python.org/3/ (online; accessed: 2022-04-09).

[22] React. — URL: https://ru.reactjs.org/ (online; accessed: 2022-04-09).

[23] Spring. — URL: https://docs.spring.io/spring-framework/docs/current/reference/html/ (online; accessed: 2022-04-09).

[24] System Usability Scale. — URL: https://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html (online; accessed: 2022-05-27).

[25] System Usability Scale. John Brook. — URL: https://www.researchgate.net/publication/228593520_SUS_A_quick_and_dirty_usability_scale (online; accessed: 2022-05-27).

[26] TIOBE. — URL: https://www.tiobe.com/tiobe-index/ (online; accessed: 2022-04-09).

[27] TestNG. — URL: https://testng.org/doc/ (online; accessed: 2022-04-10).

[28] UTBot C/C++. — URL: https://github.com/UnitTestBot/UTBotCpp (online; accessed: 2022-04-08).

[29] UTBot Java. — URL: https://github.com/UnitTestBot/UTBotJava (online; accessed: 2022-05-27).

[30] Visual Studio. — URL: https://visualstudio.microsoft.com/ru/ (online; accessed: 2022-04-10).

[31] Сэм Канер. Тестирование программного обеспечения. Фундаментальные концепции менеджмента бизнес-приложений. — Киев, 2001. — P. 543.