Инженерный вестник Дона, №8 (2024)
ivdon.ru/ru/magazine/archive/n8y2024/9406

Автоматизация тестирования систем управления облачными базами данных на примере DBaaS Postgres Pro

А.В. Мересий[1], *И.С. Полевщиков*[1,2,3]

[1]*Пермский национальный исследовательский политехнический университет*
[2]*Московский государственный университет технологий и управления имени К.Г. Разумовского (Первый казачий университет)*
[3]*Российский биотехнологический университет*

Аннотация: Статья посвящена проектированию системы автоматизации тестирования менеджера управления облачными базами данных DBaaS Postgres Pro. Сформулированы новые и актуализированы старые подходы, понятия и определения теории автоматизации тестирования. Проведен анализ современных инструментов, широко используемых в коммерческой разработке программных продуктов. Исследованы особенности тестируемой системы, включающие в себя специфику работы с облачными вычислениями и СУБД Postgres. На основе полученных данных сформирован оптимальный стек технологий, который планируется использовать в разработке, и выработаны функциональные требования к системе автоматизации тестирования. В практическом аспекте применение данной системе на проекте DBaaS позволит снизить трудоемкость и ускорить работы на этапах тестирования и разработки, повысить эффективность тестирования и качество программного продукта.
Ключевые слова: автоматизация тестирования программного обеспечения, DBaaS, облачная база данных, СУБД Postgres, язык программирования GO.

1 Введение

В современном информационном обществе облачные технологии становятся все более популярными и широко используются в различных сферах бизнеса. Одной из ключевых составляющих облачных сервисов являются облачные базы данных, которые предоставляют возможность хранения и управления данными удаленно с использованием облачной инфраструктуры [1].

DBaaS (Database as a Service) – это модель предоставления баз данных как облачного сервиса, которая позволяет пользователям получить доступ к базам данных через интернет, не заботясь о установке, настройке и обслуживании баз данных на своей стороне [1, 2]. Одним из популярных решений в этой области является DBaaS Postgres Pro – менеджер управления

облачными базами данных на основе PostgreSQL, предлагающий высокую производительность, надежность и масштабируемость.

Однако, для обеспечения качества и надежности работы облачных баз данных, необходимо проводить тестирование и контроль качества программного обеспечения (ПО). Тестирование DBaaS является объемным и сложным процессом по ряду причин, связанных с особенностями облачных баз данных и бизнес логикой, связанной с автоматизацией большинства работ по администрирования.

Существует множество систем автоматизации тестирования, которые широко используются в индустрии разработки программного обеспечения. Наиболее популярными являются: Selenium, Selenide, Playwright, Cypress, Puppiter и другие. Каждая система автоматизации имеет свои особенности и недостатки. К примеру, недостатком может является низкая скорость прохождения тестов, отсутствие возможности запуска тестов параллельно, формирование отчетов о тестовых прогонах или же сложная установка и настройка тестовой системы.

Применение систем автоматизации тестирования при разработке позволяет [3-5]: снизить трудоемкость тестирования; ускорить тестирование и разработку; повысить эффективность тестирования; повысить качество программного продукта; снизить риски выхода некачественного продукта.

Актуальной является задача развития систем в сфере автоматизации тестирования с целью снизить трудоемкость и ускорить работы на этапах тестирования и разработки, повысить эффективность процесса тестирования и качество программного продукта. Результаты решения этой задачи, основанные на формировании новых и актуализации устаревших понятий и подходов к автоматизации тестирования, разработке программного обеспечения системы для автоматизации процесса тестирования в сфере информационных технологий, рассмотрены далее.

2 Подходы к автоматизации тестирования

Существует три основных типа подходов к автоматизации тестирования: 1) тестирование на уровне кода; 2) тестирование пользовательского интерфейса; 3) тестирование API интерфейса.

К первому типу относится модульное unit тестирование. Автоматизированные модульные тесты пишутся для тестирования на уровне кода [4]. В них выявляются ошибки в функциях, методах и процедурах, написанных разработчиками.

Ко второму типу относится функциональное тестирование, имитация действий пользователя через графический UI-интерфейс. Этот подход к тестированию максимально приближен к пользовательским действиям, в нем имитируется работа мыши, клавиатуры и нажатий сенсора экрана [6].

К третьему типу относится имитация действий пользователя или сервиса через программный API интерфейс. Также к этому типу относятся тестирование производительности и функциональное тестирование безопасности [4].

3 Основные виды и уровни автоматизированных тестов

Для обеспечения эффективной и качественной автоматизации тесты разделяются на уровни, и изображаются в виде пирамиды тестирования, что показано на рис. 1.

Unit-тесты – это автоматизированные тесты, позволяющие проверить на корректность отдельные модули исходного кода программы [4]. Тестирование данным видом тестов производится на ранних этапах разработки ПО разработчиком или автоматически в системе CI/CD.

Компонентные тесты – это вид автоматизированных тестов, который направлен на проверку отдельных компонентов ПО, таких, как классы,

модули, функции или сервисы.

Рис. 1. – Пирамида автоматизации тестирования

Целью компонентных тестов является проверка корректности работы каждого компонента в изоляции от других частей системы. Могут использоваться на ранних этапах разработки. Для изоляции часто используются моки и заглушки. Автоматизированные тесты этого уровня имеют вид API тестов, реже UI-тестов. Тестирование может проводиться разработчиком, QA-инженером или автоматически через CI/CD при фиксации изменений в коде.

Интеграционные тесты – это вид автоматизированных тестов, который направлен на проверку взаимодействия между различными компонентами или модулями программного продукта. Они проверяют, как компоненты взаимодействуют друг с другом и как правильно обрабатывают данные, передаваемые между ними, и могут включать проверку работы API, UI-интерфейсов, межсервисного взаимодействия и других форм коммуникации между компонентами. На данном уровне тесты используются для проверки полного функционирования системы, используя реальные внешние зависимости. Допускается минимальное использование моков и заглушек. Автоматизированные тесты этого уровня имеют вид API-тестов и UI-тестов.

Тестирование проводиться QA инженером или автоматически, специальным набором тестов, через CI/CD, при фиксации изменений в коде.

End to end тесты – это вид автоматизированных тестов, который направлен на проверку работы всей системы или приложения в целом, в реальных условиях, включая все компоненты и процессы, которые взаимодействуют между собой. Тестирование проводится в готовой системе, на реальных данных и окружениях. Запуск и контроль тестов производится QA инженером.

Также хочется выделить отдельный вид автотестов – диагностические автотесты или Health Check тесты. Данный вид тестов существует вне пирамиды тестирования и зависит от тестируемой системы. Этот вид тестов представляет собой инструмент мониторинга состояния системы в реальном времени, позволяющий моментально реагировать на возникающие неполадки, а в некоторых случая и принимать решение о переключении всей системы на резервный контур. Диагностические тесты запускаются автоматически, согласно расписанию, составленному QA-инженером. Результаты тестов в реальном времени можно увидеть в системе мониторинга, специализированных чат каналах и в системах логирования. Тесты обычно эмулируют действия пользователей или внешних систем и сервисов.

4 Инструменты для тестирования UI

Ниже, в таблице 1 приведен анализ некоторых наиболее популярных инструментов автоматизации тестирования графической части приложений: Cypress [7], Playwright [8], Puppeteer [9], Selenium [10], Selenoid [11].

На основе сравнительного анализа инструментов автоматизации тестирования UI можно сделать вывод, что наиболее универсальным решением является Playwright.

Таблица № 1

Результаты анализа инструментов UI

п/п	Параметр		Playwright	Cypress	Puppeter	Selenium	Selenoid
1	Разработчик		Microsoft / open-source	open-source	Google / open-source	open-source	open-source
2	Язык программирования		JS/TS, Java, Python, C#, Go	JS/TS	JS/TS	Java, Python, Scala, Ruby, C#, Perl, Groovy, PHP, Go	Java, Python, Scala, Ruby, C#, Perl, Groovy, PHP, Go
3	Тестирование web UI		Есть	Есть	Есть	Есть	Есть
4	Тестирование API		Есть	Есть	Есть с оговорками	Нет	Нет
5	Скорость (по шкале от 1 до 10, где 10 это		10	5	10	3	4
6	Возможность интеграции с другими инструментами		Есть	Есть с оговорками	Есть	Есть с оговорками	Есть с оговорками
7	Удобство и простота использования		Высокая сложность	Низкая сложность	Средняя сложность	Средняя сложность	Средняя сложность
8	Простота установки и настройки		Низкая сложность	Высокая сложность	Низкая сложность	Высокая сложность	Средняя сложность
9	Сообщество и поддержка (по шкале от 1 до 10, где 10 это максимум)		5	7	5	10	10
10	Документация и ее полнота (по шкале от 1 до 10, где 10 это максимум)		6	8	4	6	6
11	Тип управления браузером		Нативное	Нативное с ограничениями	Нативное	web server	web server/docker
12	Функционал:	Запись и воспроизведение прохождения тестов (codegen/inspector)	Есть	Нет	Есть с ограничениями	Нет	Нет
		Генерация отчетов	Есть	Есть		Нет	Нет
		Параллельное выполнение	Есть	Ограничено	Есть	Есть с оговорками	Есть
		Тестовая разметка	Есть	Есть	Нет	Нет	Нет
		Headless mode	Есть	Есть	Есть	Есть	Есть
		Управление данными	Есть с оговорками	Есть с оговорками	Есть с оговорками	Есть с оговорками	Есть с оговорками
		Управление вкладками браузера	Есть	Нет	Есть	Есть с ограничениями	Есть с ограничениями
		Управление событиями браузера	Есть	Есть	Есть	Нет	Нет
		Умные ожидания	Есть	Есть	Есть	Нет	Есть с ограничениями
		Возможность создания mock	Есть	Есть	Есть	Нет	Нет
		Геолокация	Есть	Есть	Есть	Нет	Нет
		Локализация	Есть	Есть	Есть	Нет	Нет
		Работа с файлами	Есть	Есть с ограничениями	Есть	Нет	Нет
		Запись видео и скриншотов	Есть	Есть	Есть	Нет	Есть с оговорками
13	Поддерживаемые браузеры и платформы		Chrome, Firefox, WebKit, Mobile	Chrome, Firefox, WebKit, Edge	Chrome	Chrome, Firefox, Edge, EE, Safari, Opera	Chrome, Firefox, Edge, EE, Safari, Opera
14	Стоимость		Бесплатно	Бесплатно	Бесплатно	Бесплатно	Бесплатно

Из важных особенностей можно выделить, что инструмент имеет открытый исходный код, бесплатный, официально поддерживается на всех современных языках программирования, прост в установке и использовании,

даже в режиме параллельного запуска тестов и обладает очень внушительным функционалом «из коробки».

5 Инструменты для тестирования Backend

Тестирование серверной части приложения, а именно API, может производиться с помощью некоторых инструментов тестирования UI.

Для более глубокого и разностороннего тестирования серверной части приложения в автоматизированных тестах обычно используется связка доменной и тестовой библиотек выбранного языка программирования. Например, для языка Go такими связками является: для тестирования Rest API – valyala/fasthttp и stretchr/testify; для тестирования RPC API – grpc/grpc-go и stretchr/testify; для баз данных Postgres – jackc/pgx и stretchr/testify; для баз данных MySQL – jmoiron/sqlx и smartystreets/goconvey; для брокера очередей – confluentinc/confluent-kafka-go и go-check/check; для SSH – gliderlabs/ssh и smartystreets/goconvey; для SFTP – drakkan/sftpgo и go-check/check.

6 Инструменты для формирования, хранения и анализа тестовой документации

Для анализа были выбраны наиболее популярные инструменты Allure Report [12], Allure TestOps [13], Azure DevOps [14], Qase [15], TestIT [16], TestRail [17], которые позволяют управлять тестовой документацией и автоматизацией тестирования. Результаты представлены в таблице 2.

На основе сравнительного анализа инструментов тест-менеджмента можно сделать вывод, что наиболее оптимальным инструментов является Allure TestOps. Система предоставляет широкие возможности для управления ручным и автоматизированным тестированием на проекте.

Таблица № 2

Результаты анализа менеджмент- систем тестирования

п/п	Параметр	Allure reports	Allure TestOps	TestRail	Azure DevOps	Test IT	Qase
1	Стоимость	Бесплатно	Платно	Платно	Платно	Платно	Платно
2	Интеграция с системами CI/CD	Не требуется	Да	Да	Да	Да	Да
3	Интеграция с другими системами разработки	Не требуется	Да	Да	Да	Да	Да
4	Разметка сценариев	Да	Да	Да	Да	Да	Да
5	Формирование отчетов	Да	Да	Да	Да	Да	Да
6	Создание тест-кейсов автоматизации	Нет	Да	Да	Да	Да	Да
7	Анализ и сравнение результатов	Да	Да	Да	Да	Да	Да
8	Доступен в РФ	Да	Да	Нет	Ограниченно	Да	Нет

Также у инструмента имеется упрощенная бесплатная версия – Allure reports, использование которой не требует дополнительных настроек и действий. Это делает инструмент стабильным и устойчивым к современным реалиям.

7 Инструменты CI/CD

Continuous Integration (CI) и Continuous Delivery/Continuous Deployment (CD) – это подходы к разработке программного обеспечения, которые позволяют автоматизировать процессы сборки, тестирования и развертывания приложений.

В таблице 3 произведен анализ наиболее популярных инструментов: Bamboo [18], Gitlab [19], Jenkins [20], TeamCity [21].

На основе сравнительного анализа инструментов CI/CD можно сделать вывод о том, что Gitlab является наиболее подходящим вариантом.

Таблица № 3

Результаты анализа CI/CD инструментов для управления запуском автоматизированных тестов

п/п	Параметр	Jenkins	Gitlab	Teamcity	Bamboo
1	Разработчик	open-source	open-source	Jetbrains	Atlassian
2	Стоимость	Бесплатно/ Платно	Бесплатно/ Платно	Платно	Платно
3	Функциональность стандартной версии (по шкале от 1 до 10, где 10 это максимум)	4	8	7	7
4	Наличие расширений	Да	Да	Да	Да
5	Порог входа	Средний	Средний	Средний	Средний
6	Сообщество и поддержка (по шкале от 1 до 10, где 10 это максимум)	10	10	5	5
7	Параллельные вичисления	Да	Да	Да	Да
8	Язык скрипта	Groovy/Java	Bash/Python	Bash, C#, C++, Go, Java, Kotlin, JS, Swift, PHP, Python, Ruby	Bash/Python
9	Стабильность системы (по шкале от 1 до 10, где 10 это максимум)	7	9	7	7
10	Доступен в РФ	Доступен	Доступен	Условно доступен	Условно доступен

Инструмент является open-source, имеет обширный функционал в стандартной версии, легко интегрируется с любыми сторонними системами и имеет различные варианты инсталляции. Коммерческая версия системы разворачивается на серверах заказчика, что позволяет выстроить дополнительную систему защиты информации.

8 Анализ особенностей тестируемой системы

Устройство DBaaS (рис. 2) представляет виртуальную машину, где

работает сам DBaaS-сервис, к которому подключаются клиенты по API или через веб-интерфейс. Далее DBaaS создает виртуальные машины, устанавливает на них своего управляющего агента и разворачивает базу данных согласно запросу клиента.

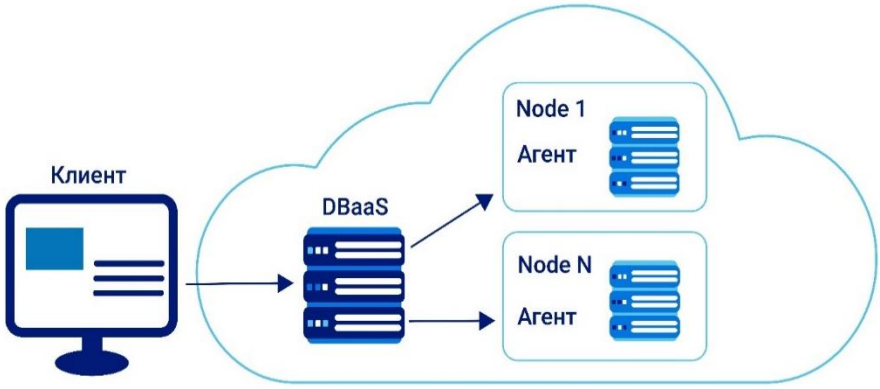

Рис. 2. – Схема устройства DBaaS Postgres Pro

Для наилучшего обеспечения качества тестируемой системы необходимо также учитывать достоинства и недостатки технологий, используемых в основе этой системы. Для этого в системе автоматизации тестирования необходимо предусмотреть возможность проверки функциональных и не функциональных требований облачных вычислений и баз данных.

Достоинства использования облачных баз данных: гибкость и масштабируемость; доступность и отказоустойчивость; быстрота развертывания; управление затратами; безопасность и соответствие стандартам.

Недостатки использования облачных баз данных: зависимость от Интернет-соединения; проблемы с конфиденциальностью данных; ограничения в управлении инфраструктурой; зависимость от провайдера; сложности с миграцией данных.

Еще одной важной технологией, на основе которой построена система DBaaS является СУБД Postgres. Система автоматизации тестирования должна

иметь широкие возможности для взаимодействия с СУБД, для проверки ее критически важного функционала.

Postgres является мощной, популярной и открытой объектно-реляционной системой управления базами данных (СУБД), которая предоставляет широкий спектр возможностей для работы с данными [2].

Основные из них: поддержка SQL; расширяемость; масштабируемость; транзакционность; поддержка триггеров и хранимых процедур; многоязычность; поддержка различных типов данных; расширенная поддержка индексов; полноценная поддержка ACID; системы репликации и резервного копирования; гибкая система безопасности; поддержка различных операционных систем.

9 Проектирование системы автоматизации тестирования

На основе проведенного анализа программных продуктов, используемых для автоматизации тестирования, и системы DBaaS Postgres Pro, с учетом ее конструкции и используемых технологий, были выбраны инструменты, совокупность которых позволит максимально покрыть автоматизированными тестами систему, учитывая различные уровни тестирования, обеспечит стабильность и скорость прохождения тестов, предоставит обширный инструментарий для документирования процесса тестирования и его анализа. При этом управление процессом тестирования является централизованным.

Функциональные требования к системе, посредством которой происходит контроль и обеспечение качества на проекте DBaaS Postgres Pro, в общем виде представлены диаграммой прецедентов UML на рис. 3.

Для системы автоматизации тестирования проекта DBaaS Postgres Pro определены акторы QA инженер (инженер по обеспечения качества или ответственный разработчик), разработчик ПО (разработчик FE/BE или DBA инженер), менеджер (менеджер продукта, направления, департамента,

директорат и тп.), для каждого из которых выявлены и описаны варианты использования.

Рис. 3. – Общие функциональные требования к системе автоматизации тестирования на проекте DBaaS Postgres Pro

Согласно требованиям, QA инженер:

1) учувствует в подготовке и планировании работ по автоматизации тестирования;

2) занимается разработкой и актуализацией автоматизированных тестов;

3) формирует списки для запуска, далее запускает тесты в ручном или автоматизированном режиме, когда это необходимо, используя различные компоненты системы;

4) получает уведомления с необходимой информацией о старте и результатах прогона тестов в мессенджер;

5) может посмотреть отчет с результатами прогона тестов в одном из удобных для него вариантов, а также сравнить его с предыдущими запусками.

Для разработчика ПО, согласно требованиям, доступен функционал по запуску автотестов, выбрав нужный тест-план прогона, получение уведомлений о результатах этого запуска и возможность просмотра результатов прохождения тестов и сравнения их с предыдущими запусками.

Менеджер продукта учувствует в планировании работы отдела QA и расставляет приоритеты для работ с учетом интересов заказчика, используя инструмент на основе метода анализа иерархий, а также имеет возможность просмотра любого отчета с результатами прогона из истории.

Детальные требования для каждого этапа системы представлены диаграммами прецедентов UML. На рис. 4 показаны функциональные требования к подсистеме «Подготовка и планирование» системы автоматизации тестирования.

Рис. 4. – Функциональные требования к подсистеме «Подготовка и планирование» системы автоматизации тестирования

На рис. 5 показаны функциональные требования к подсистеме «Разработка автотестов».

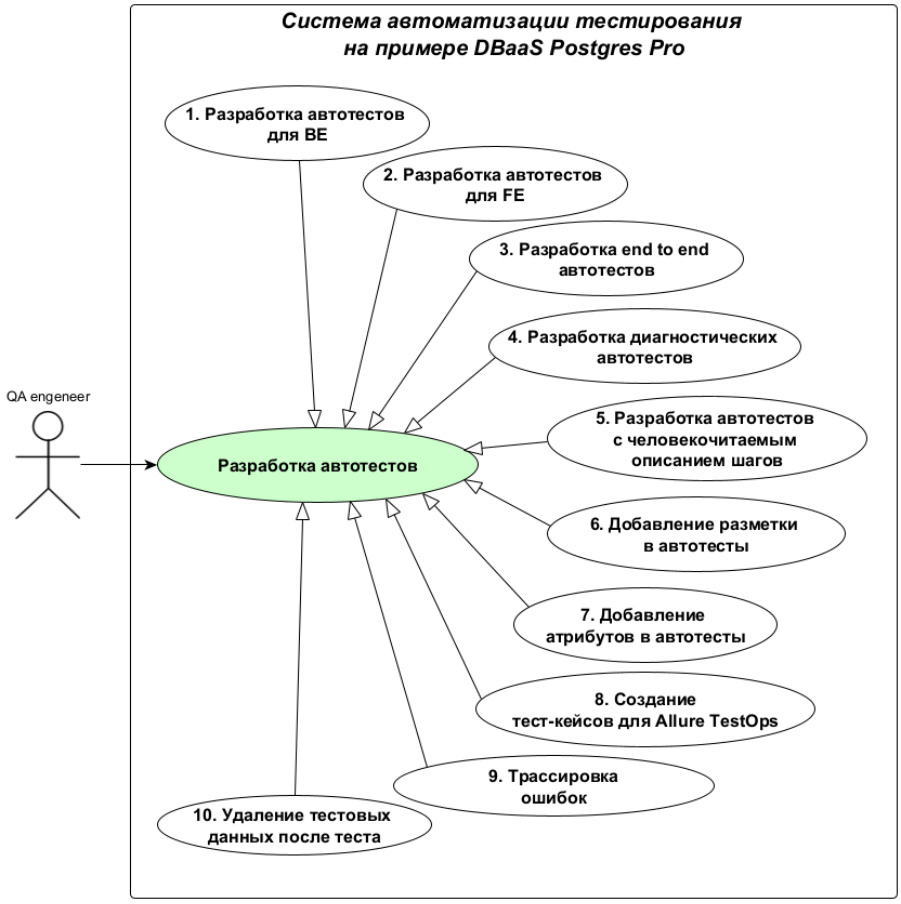

Рис. 5. – Функциональные требования к подсистеме «Разработка автотестов»

На рис. 6 показаны функциональные требования к подсистеме «Формирование списка прогона».

Рис. 6. – Функциональные требования к подсистеме «Формирование списка прогона»

На рис. 7 показаны функциональные требования к подсистеме «Запуск автотестов».

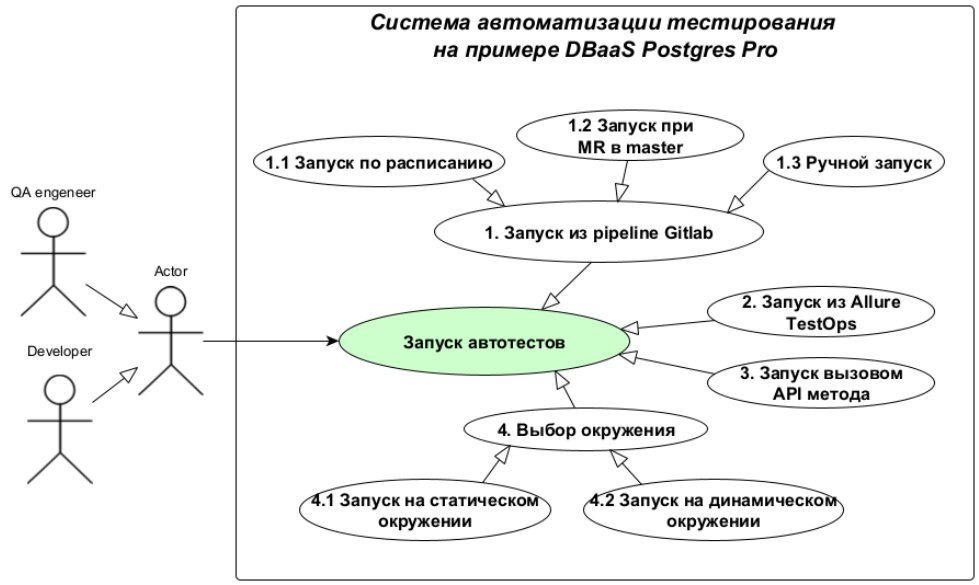

Рис. 7. – Функциональные требования к подсистеме «Запуск автотестов»

На рис. 8 показаны функциональные требования к подсистеме «Получение уведомлений».

Рис. 8. – Функциональные требования к подсистеме «Получение уведомлений»

На рис. 9 показаны функциональные требования к подсистеме «Просмотр отчета».

Рис. 9. – Функциональные требования к подсистеме «Просмотр отчета»

На рис. 10 показаны функциональные требования к подсистеме «Просмотр истории».

Рис. 10. – Функциональные требования к разделу «Просмотр истории»

Функциональные требования к системе автоматизации, описанные в виде диаграмм, позволят автоматизировать процессы тестирования на

проекте, увеличить их эффективность и прозрачность. В целом, это позволит увеличить качество выпускаемого продукта, скорость разработки и выпуска, а также оптимизирует человеческий инженерный ресурс.

10 Заключение

1) Исследованы основные подходы к автоматизации тестирования, уровни автоматизированных тестов и их виды. На базе данного материала сформулированы новые и актуализированы старые понятия автоматизации тестирования.

2) Проведен анализ преимуществ и недостатков существующих систем автоматизации тестирования и смежных инструментов. Проведен анализ особенностей тестируемого проекта и его основных технологий. Предложены наиболее оптимальные для проекта DBaaS Postgres Pro инструменты тестирования для построения на их основе системы автоматизации тестирования: язык программирования Go; для тестирования UI – инструмент Playwright; для тестирования BE – библиотеки Go, такие как testify, pgx, fasthttp; для хранения тестовой документации и формирования отчетности – Allure TestOps; инструмент CI/CD – Gitlab.

3) Сформулированы требования к системе автоматизации тестирования как в общем виде, так и в детальном исполнении. В практическом аспекте применение системы автоматизации тестирования позволит снизить трудозатраты отдела обеспечения качества, увеличить прозрачность процесса тестирования, а также уменьшить сроки выпуска ПО, улучшив его качество.

4) Представленные формализовано множеством диаграмм UML-требования являются основой детального проектирования, реализации и внедрения как для системы автоматизации тестирования на проекте DBaaS Postgres Pro, так и для других систем автоматизации тестирования с учетом возможных изменений, связанных со спецификой тестируемого ПО.

Литература

1. Букарев А.В. Эффективный метод автоматизированного тестирования программного обеспечения устройств потребительской электроники с использованием облачных устройств // Инженерный вестник Дона. 2023. №9. URL: ivdon.ru/ru/magazine/archive/n9y2023/8697.

2. Новиков Б.А., Горшкова Е.А., Графеева Н. Г. Основы технологий баз данных: учебное пособие. ДМК Пресс, 2020. 582 с.

3. Гребенюк В.М. Оценка целесообразности внедрения автоматизированного тестирования // Интернет-журнал Науковедение. 2013. № 1 (14). С. 13. URL: naukovedenie.ru/PDF/13tvn113.pdf.

4. Плодухин Д.М. Реализация модели автоматизированного тестирования // Огарёв-Online. 2020. № 13 (150). С. 1. URL: journal.mrsu.ru/wp-content/uploads/2020/11/ploduxin.pdf.

5. Полевщиков И.С., Файзрахманов Р.А. Автоматизированное управление тестированием программных систем с применением нейронных сетей // Инженерный вестник Дона. 2018. №4. URL: ivdon.ru/ru/magazine/archive/n4y2018/5283.

6. Кудрявцева Е.Ю. Автоматизированное тестирование веб-интерфейсов // Горный информационно-аналитический бюллетень (научно-технический журнал). 2014. № S. С. 354-356.

7. Cypress. Is a next generation front end testing tool built for the modern web. URL: docs.cypress.io/guides/overview/why-cypress (Дата обращения: 05.06.2024).

8. Playwright: Enables reliable end-to-end testing for modern web apps. URL: playwright.dev/docs/intro (Дата обращения: 05.06.2024).

9. Puppeteer. Is a Node.js library which provides a high-level API to control Chrome/Chromium over the DevTools Protocol. URL: pptr.dev/ (Дата

обращения: 05.06.2024).

10. Selenium: Browser Automation Project. URL: selenium.dev/documentation/ (Дата обращения: 05.06.2024).

11. Selenoid. Is a powerful implementation of Selenium hub using Docker containers to launch browsers. URL: aerokube.com/selenoid/latest/ (Дата обращения: 05.06.2024).

12. Allure Report. Open-source HTML test automation report tool. URL: allurereport.org/ (Дата обращения: 05.06.2024).

13. Allure TestOps. Full-stack Test Management. URL: qameta.io/ (Дата обращения: 05.06.2024).

14. Azure DevOps. Provides integration with popular open source and third-party tools and services-across the entire DevOps workflow. URL: azuredevopslabs.com/ (Дата обращения: 05.06.2024).

15. Qase. Test management software for quality assurance. URL: qase.io/ (Дата обращения: 05.06.2024).

16. TestIT. Система управления тестированием. URL: testit.software/ (Дата обращения: 05.06.2024).

17. TestRail. Test Case Management & Orchestration Software. URL: testrail.com/ (Дата обращения: 05.06.2024).

18. Bamboo. Is a continuous integration and delivery tool that ties automated builds, tests, and releases into a single workflow. URL: confluence.atlassian.com/bamboo/bamboo-documentation-289276551.html (Дата обращения: 05.06.2024).

19. Gitlab. Is the most comprehensive AI-powered DevSecOps platform.URL: docs.gitlab.com/ (Дата обращения: 05.06.2024).

20. Jenkins. Open source automation server which enables developers around the world to reliably build, test, and deploy their software. URL: jenkins.io/doc/ (Дата обращения: 05.06.2024).

21. TeamCity. CI/CD solution for all sorts of workflows and development practices. URL: jetbrains.com/help/teamcity/teamcity-documentation.html (Дата обращения: 05.06.2024).

References

1. Bukarev A.V. Inzhenernyj vestnik Dona, 2023, №9. URL: ivdon.ru/ru/magazine/archive/n9y2023/8697.

2. Novikov B.A., Gorshkova E.A., Grafeeva N. G. Osnovy tekhnologiy baz dannykh: uchebnoe posobie [Fundamentals of Database Technologies: Tutorial]. DMK Press, 2020. 582 p.

3. Grebenyuk V.M. Internet-zhurnal Naukovedenie. 2013. № 1 (14). P. 13. URL: naukovedenie.ru/PDF/13tvn113.pdf**.**

4. Plodukhin D.M. Ogarev-Online. 2020. № 13 (150). P. 1. URL: journal.mrsu.ru/wp-content/uploads/2020/11/ploduxin.pdf.

5. Polevshchikov I.S., Fayzrakhmanov R.A. Inzhenernyj vestnik Dona, 2018, №4. URL: ivdon.ru/ru/magazine/archive/n4y2018/5283.

6. Kudryavtseva E.Yu. Gornyj informatsionno-analiticheskiy byulleten' (nauchno-tekhnicheskiy zhurnal). 2014. № S. pp. 354-356.

7. Cypress. Is a next generation front end testing tool built for the modern web. URL: docs.cypress.io/guides/overview/why-cypress (Date accessed 05/06/2024).

8. Playwright: Enables reliable end-to-end testing for modern web apps. URL: playwright.dev/docs/intro (Date accessed 05/06/2024).

9. Puppeteer. Is a Node.js library which provides a high-level API to control Chrome/Chromium over the DevTools Protocol. URL: pptr.dev/ (Date accessed 05/06/2024).

10. Selenium: Browser Automation Project. URL: selenium.dev/documentation/ (Date accessed 05/06/2024).

11. Selenoid. Is a powerful implementation of Selenium hub using Docker containers to launch browsers. URL: aerokube.com/selenoid/latest/ (Date accessed 05/06/2024).

12. Allure Report. Open-source HTML test automation report tool. URL: allurereport.org/ (Date accessed 05/06/2024).

13. Allure TestOps. Full-stack Test Management. URL: qameta.io/ (Date accessed 05/06/2024).

14. Azure DevOps. Provides integration with popular open source and third-party tools and services-across the entire DevOps workflow. URL: azuredevopslabs.com/ (Date accessed 05/06/2024).

15. Qase. Test management software for quality assurance. URL: qase.io/ (Date accessed 05/06/2024).

16. TestIT. Sistema upravleniya testirovaniem [TestIT. Test Management System]. URL: testit.software/ (Date accessed 05/06/2024).

17. TestRail. Test Case Management & Orchestration Software. URL: testrail.com/ (Date accessed 05/06/2024).

18. Bamboo. Is a continuous integration and delivery tool that ties automated builds, tests, and releases into a single workflow. URL: confluence.atlassian.com/bamboo/bamboo-documentation-289276551.html (Date accessed 05/06/2024).

19. Gitlab. Is the most comprehensive AI-powered DevSecOps platform.URL: docs.gitlab.com/ (Date accessed 05/06/2024).

20. Jenkins. Open source automation server which enables developers around the world to reliably build, test, and deploy their software. URL: jenkins.io/doc/ (Date accessed 05/06/2024).

21. TeamCity. CI/CD solution for all sorts of workflows and development practices. URL: jetbrains.com/help/teamcity/teamcity-documentation.html (Date accessed 05/06/2024).

Дата поступления: 7.06.2024

Дата публикации: 25.07.2024