ДОРОГОБЕД А. Н., СОЧКО С. С., ШАРФИНА Е. С. АВТОМАТИЗАЦИЯ ПРОЦЕССОВ НЕПРЕРЫВНОЙ ИНТЕГРАЦИИ И НЕПРЕРЫВНОГО РАЗВЕРТЫВАНИЯ ЛОКАЛЬНЫХ ИНФОРМАЦИОННО-УПРАВЛЯЮЩИХ СИСТЕМ

УДК 681.51, ВАК 2.3.1., ГРНТИ 50.43.17

Автоматизация процессов непрерывной интеграции и непрерывного развёртывания локальных информационно-управляющих систем

А. Н. Дорогобед, С. С. Сочко, Е. С. Шарфина

Ухтинский государственный технический университет, г. Ухта

Статья посвящена проектированию и разработке технологии автоматизации процессов непрерывной интеграции и непрерывного развёртывания локальных информационно-управляющих систем (ЛИУС). Объектом исследования являются процессы организации коллективной разработки, модификации, тестирования и внедрения ЛИУС. В ходе работ была разработана соответствующая технология автоматизации непрерывной интеграции и развёртывания.

Ключевые слова: *CI/CD, Jenkins, Git*

"Automation of the processes of continuous integration and continuous deployment of local information and control systems

A. N. Dorogobed, S. S. Sochko, E. S. Sharfina

Ukhta State Technical University, Ukhta

The article is devoted to the design and development of technology for automating the processes of continuous integration and continuous deployment of local information management systems (LIMS). The object of the research is the processes of organizing collective development, modification, testing and implementation of LIMS. In the course of the work, an appropriate technology for automating continuous integration and deployment was developed.

Keywords: *CI/CD, Jenkins, Git*

Введение

В рамках статьи рассматривается изучение и автоматизация процессов непрерывной интеграции и непрерывного развёртывания локальных информационных-управляющих систем (далее – ЛИУС).

ООО «Газпром трансгаз Ухта» — дочернее общество ПАО «Газпром» и субъект естественной монополии. Основные задачи Общества — транспорт газа по системе магистральных газопроводов, бесперебойная поставка газа промышленным и коммунально-бытовым потребителям.

Разработка и развитие ЛИУС на предприятии выполняется работниками службы информационно-управляющих систем.

В качестве централизованного хранилища исходных кодов ЛИУС используется Git под управлением свободно распространяемого программного обеспечения Gogs. Разработка в системе контроля версий Git ориентирована на обеспечение высокой производительности, безопасности и гибкости распределенной системы, позволяет всем разработчикам хранить историю изменений проекта в полном объеме.

В ходе изучения рабочего процесса была проанализирована работа программистов предприятия. В зависимости от сложности и объема разработки/модификации ЛИУС в процессе разработки участвуют один или несколько программистов.

На основе разработанной инструкции, администратор системно-технической инфраструктуры (далее – администратор СТИ) для WEB-приложений выполняет сборку и публикует приложение на тестовом ландшафте.

Следующим шагом выполняется функциональное тестирование разработанной/модифицированной ЛИУС на тестовом ландшафте. Часто функциональное тестирование выполняется тем-же программистом, который выполнял разработку/модификацию ЛИУС.

Предлагается разработать технологию автоматизации непрерывной интеграции и развёртывания, отвечающие всем функциональным требованиям, предъявляемым заказчиком.

Необходимо выполнить предпроектный анализ, на основе которого сформулировать и описать задачи и предметную область. Для реализации следует изучить и выбрать средство автоматизации, удовлетворяющее условиям и масштабам заказчика.

Предпроектный анализ

Разработка и модификация локальных информационно-управляющих систем на предприятии ведётся работниками службы информационно-управляющих систем.

Между сотрудниками распределены следующие роли:

- Разработчик – сотрудник, который занимается написанием кода для систем. Может как разрабатывать новую систему с нуля, так и дорабатывать имеющиеся;
- Администратор системно-технической инфраструктуры – сотрудник, отвечающий за настройку и функционирование инфраструктуры производственного и тестовых ландшафтов, выполняющий работы по сборке и публикации ЛИУС;
- Руководитель проекта – сотрудник, отвечающий за подготовку готовых проектов к публикации на производственный и тестовый ландшафт. Знает структуру и принцип работы всего ЛИУС, распределяет работу между сотрудниками.

- Тестировщик – сотрудник, который проверяет работоспособность и корректность работы системы.

Разработка ЛИУС разбита на несколько ключевых этапов, которые помогают контролировать процесс разработки и делать совместную работу более продуктивной для предприятия.

Разработка приложений происходит в несколько этапов:

– Анализ функциональных требований к системам, проектирование. На этом этапе работники обсуждают по каким принципам лучше всего проектировать приложение или новые функции для него. Это помогает команде лучше координировать работу.

– Реализация функциональных требований. В зависимости от объема проекта выполняется одним или несколькими работниками. На этом этапе разработчики работают на своих ветках Git и их работа не пересекается между собой.

– Слияние изменений в основную ветку проекта. Проект выполняется руководителем после визуальной проверки написанного кода.

– Разработка инструкции администратора системно-технической инфраструктуры, содержащую требования к системным пакетам и их версиям и пошаговое описание процесса сборки и конфигурации ЛИУС. Необходимый для дальнейшей сборки и публикации ЛИУС.

– Сборка. Процесс сборки зависит от технологий разработки ЛИУС. Для Web приложений сборка выполняется на сервере администратором системно-технической инфраструктуры. Для Desktop приложений сборка выполняется разработчиком вручную. В текущий момент при разработке отдаётся предпочтение web приложениям.

– Функциональное тестирование. Выполняется вручную на тестовом ландшафте. Тестировщиком может быть как сам разработчик, так и любой человек из команды.

– Подготовка Релиза. По итогам ручного функционального тестирования в ЛИУС при необходимости вносятся изменения. После этого выпускается версия кода, готовая для публикации, ей присваивается номер и начинается следующий этап цикла [1].

– Публикация на производственный ландшафт. Выполняется вручную администратором системно-техническое инфраструктуры, этап на котором нужна особая внимательность. В случае проблем с разработанными функциями приложения, его исправляют в срочном порядке.

– Поддержка и мониторинг. Конечные пользователи начинают работать с ЛИУС. Команда разработки поддерживает его и анализирует пользовательский опыт.

В текущем процессе разработки имеется ряд существенных недостатков, которые приводят к появлению дополнительных ошибок и общему замедлению работы сотрудников [2].

В ходе анализа процесса разработки/модификации ЛИУС и их публикации выявлены следующие проблемы:

- отсутствует автоматическое модульное тестирование;

- слияние модификаций в основную ветку проекта:
  - усложняет процесс отката ошибочных изменений при необходимости [3];
  - не дает ясной картины какая версия кода опубликована на продуктивном и тестовом ландшафтах;
  - приводит к значительному количеству конфликтов слияния при коллективной разработке/модификации ЛИУС.
- при выполнении функционального тестирования программистом, который выполнял разработку/модификацию, снижается качество тестирования и осведомленность других программистах о выполненных модификациях[4];
- необходимость разработки инструкции администратора системно-технической инфраструктуры по сборке и публикации ЛИУС;
- ручная сборка и публикация ЛИУС может приводить к ошибкам человеческого фактора;
- длительность процесса публикации новых версий ЛИУС, вызванная необходимость разработки инструкций и ручным выполнением операций по сборке и публикации ЛИУС [5].

На предприятии остро стоит вопрос с импортозамещением используемых приложений. Следовательно выбор должен удовлетворять этот запрос, система должна быть либо российской разработкой, либо полностью бесплатной.

Сопоставление функций систем представлены ниже

Таблица 3. Сравнение аналогов

Функции системы	Jenkins	TeamCity	Travis	Codeship
Поддержка ОС Linux	+	+	+	-
Хостинг в облаке	+	-	+	+
Поддержка контейнеризации	+	+	+	-
Бесплатность	+	-	-	-
Наличие множества плагинов	+	+	-	+
Обширная документация	+	+	-	-
Удобство использования	+	-	+	+
Простота в обучении	+	-	+	+
Удобство для больших проектов	+	+	-	+

В результате обзора ряда программных продуктов, существующих в данный момент на рынке, были выявлены как достоинства, так и недостатки каждого и было принято решение об использовании Jenkins.

Весь процесс публикации версий на производственного и тестового сервера остаётся подконтрольным и осознанным, но рутинные действия выполняются системой автоматизации. [6]

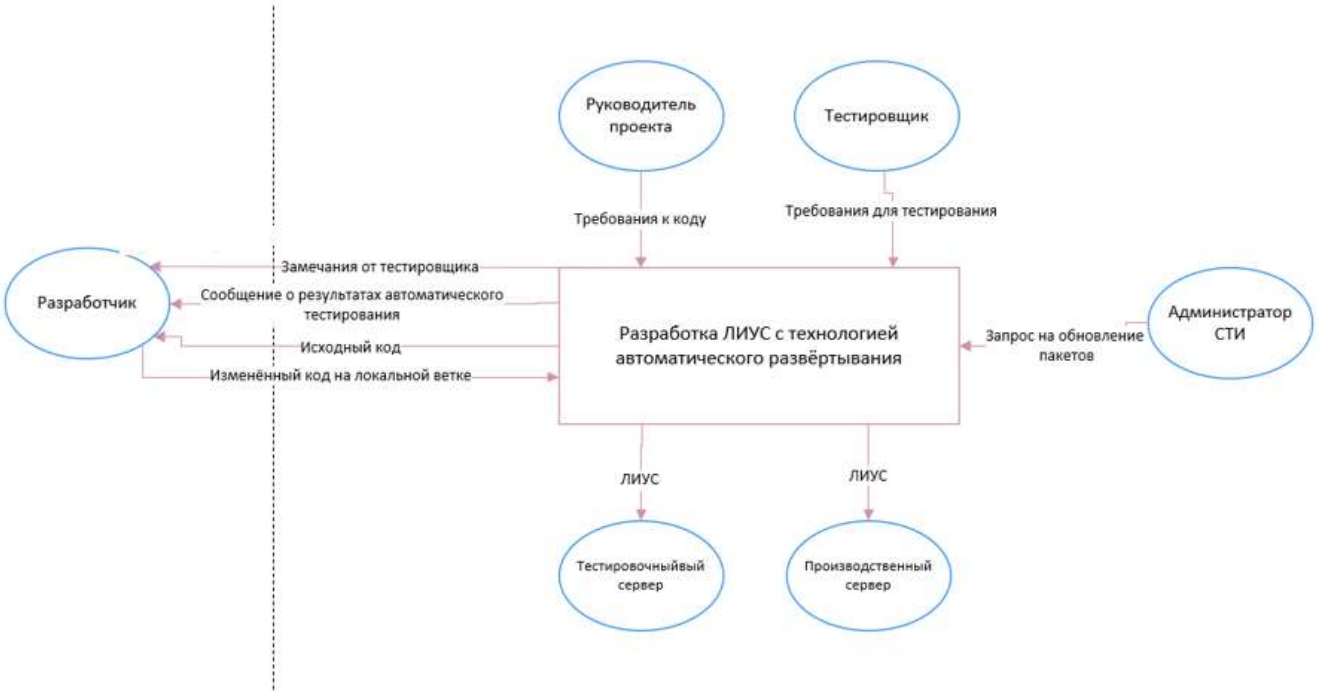

Рисунок 2. Контекстная диаграмма «Как будет»

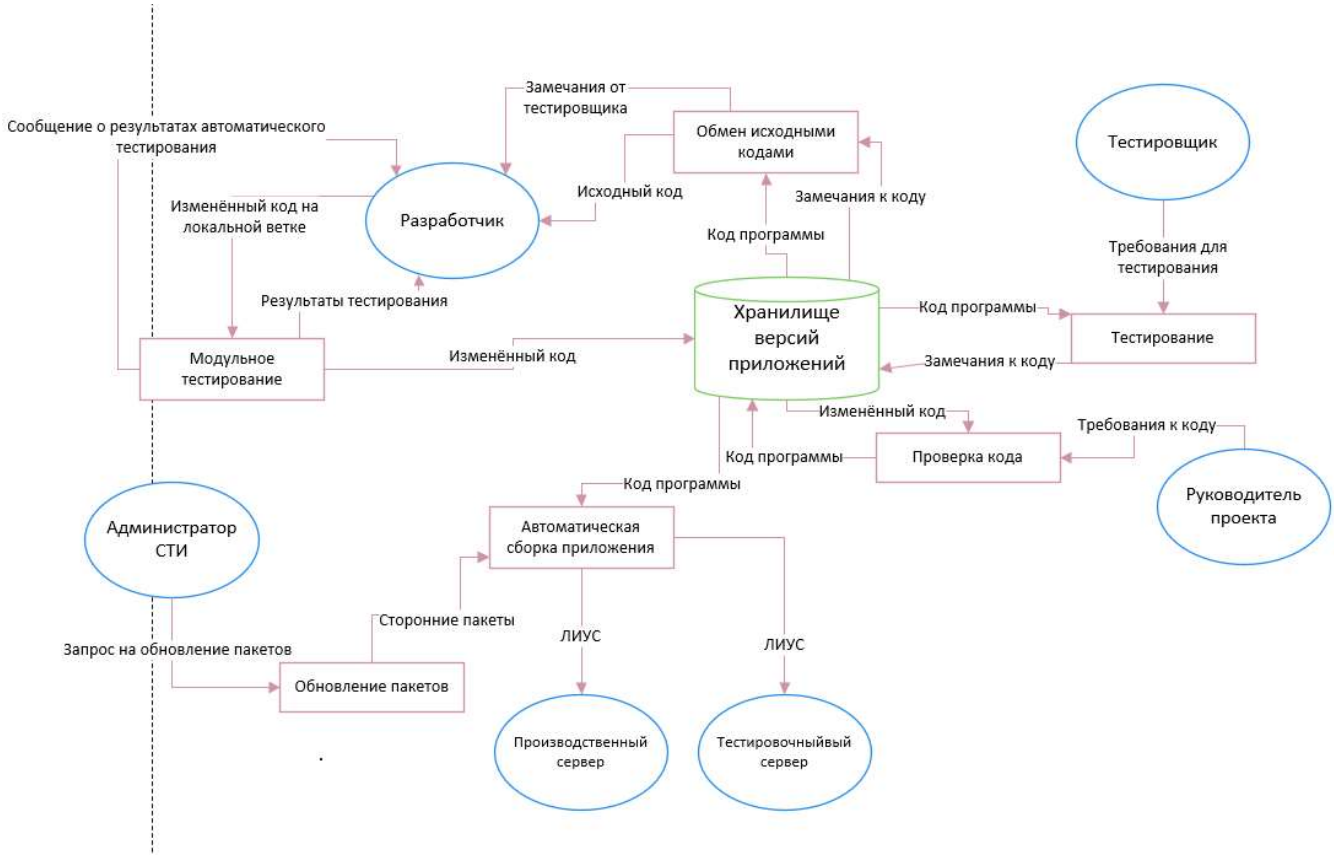

Рисунок 3. Диаграмма потоков данных «Как будет»

Проектирование системы

Проектирование системы непрерывной интеграции и непрерывного развёртывания выполнено с использованием виртуальной машины, созданной в программе VirtualBox на базе Linux, был выбран дистрибутив Ubuntu как один из самых распространённых.

На предприятии используется git-сервис Gogs, к нему в качестве базы данных использована MariaDB. Для размещения веб-сервера было установлено программное обеспечение Apache. Для проверки работоспособности системы и эмулирования модификация ЛИУС использовался язык PHP, менеджер пакетов Compser и фреймворк Laravel.

Система была смоделирована согласно всем требованиям и нововведениям, с использованием описанных в предыдущей главе программных продуктов.

ЛИУС была развёрнута на смоделированной системе. Сперва был создан коммит с новой версией ЛИУС, который был выгружен на удалённый репозиторий. Обновленная ЛИУС была проверена на локальном ландшафте, а затем залита на ветку Develop после проверки другим разработчиком. Затем ветка Develop слита с Stage после проверки руководителем проекта, а Jenkins провел модульное тестирование ЛИУС, после чего отправил её на тестовый ландшафт. После подтверждения тестировщика и слияния с веткой master, был создан новый релиз с тегом 1.1.0, его Дженкинс протестировал и успешно отправил на производственный ландшафт.

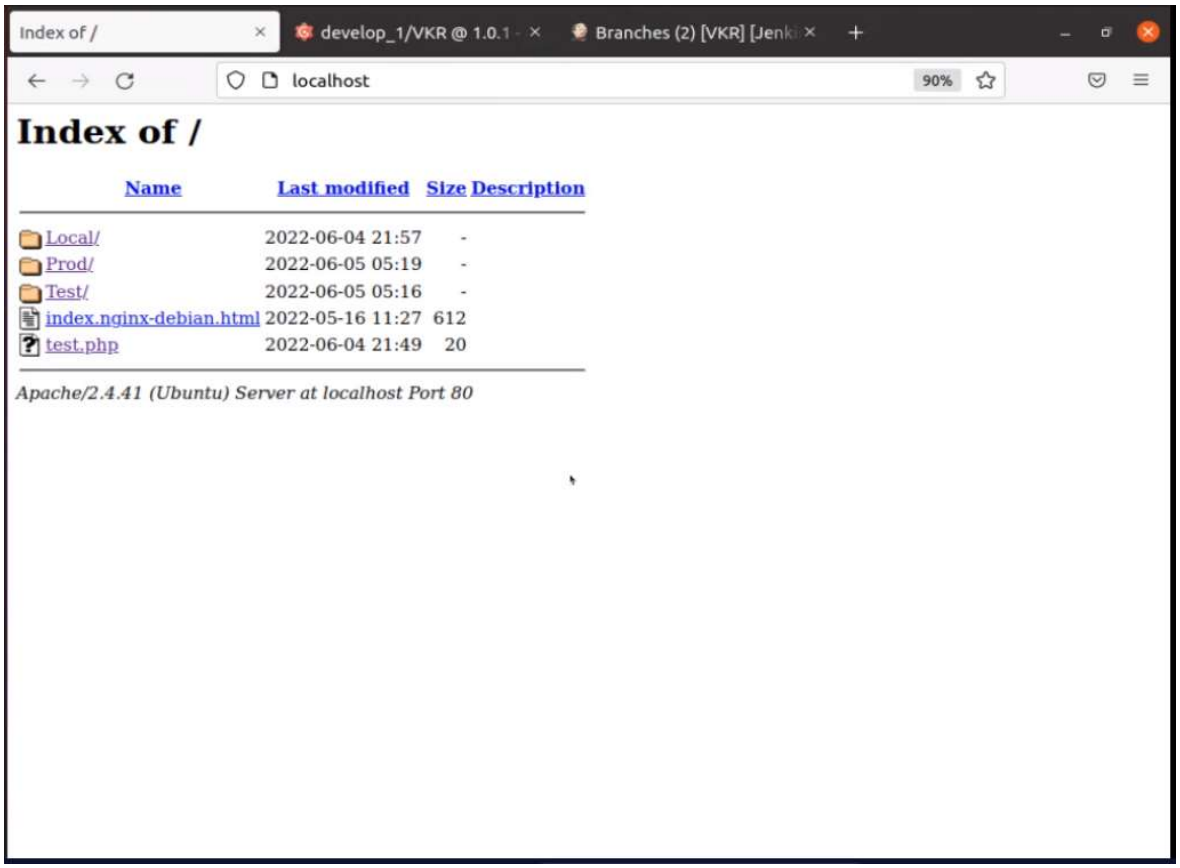

Рисунок 4. Ландшафты, созданные на Apache

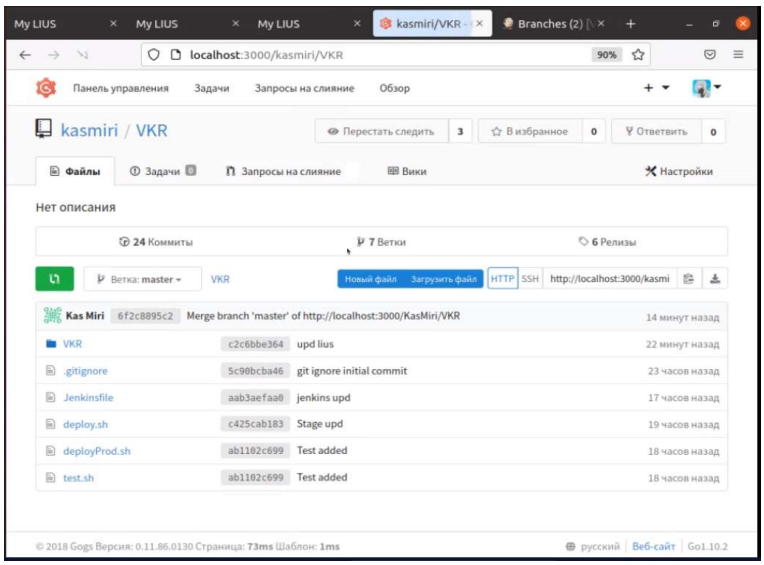

Рисунок 5. Работа с Gogs

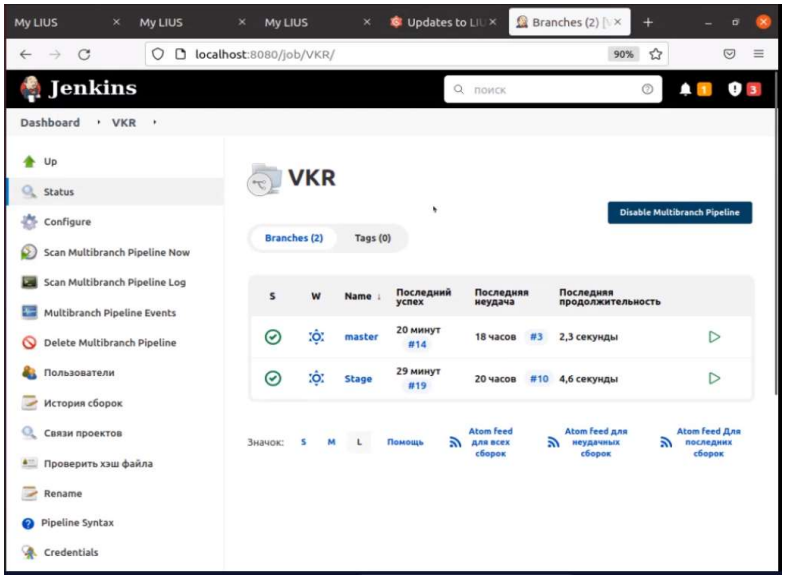

Рисунок 6. Настроенный Jenkins

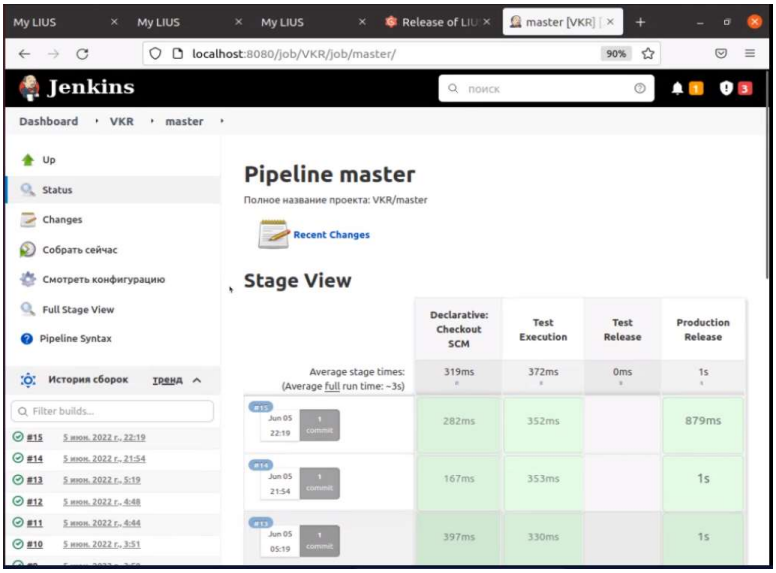

Рисунок 7. Публикация на производственный ландшафт

Теперь разработчикам не нужно писать инструкции для администраторов СТИ, автоматическая система должна сама справляться с задачей развёртывания ЛИУС на ландшафтах.

Настройка автоматической системы развёртывания и интеграции ЛИУС была успешна реализована.

Заключение

Разработка системы автоматизации процессов непрерывной интеграции и непрерывного развёртывания ЛИУС позволила отказаться от написания разработчиками инструкции по сборке и публикации ЛИУС для администратора системно-технической инфраструктуры. Благодаря этому сборка и публикация ЛИУС не будут больше приводить к ошибкам человеческого фактора. Процессы так же остались контролируемыми и осознанными, они будут выполняться после команды руководителя проекта, что так же ускоряет процесс разработки ЛИУС.

Была решена проблемы отсутствия обязательных автоматических модульных тестов. Теперь модульные тесты запускаются автоматически после сборки ЛИУС. Для тестирования ЛИУС на тестовом ландшафте обязательно привлекается разработчик, который не занимался разработкой тестируемых функций. Таким образом в роли тестировщика он повышает качество и надёжность тестирования, а также больше разработчиков осведомлены о модификациях и работе ЛИУС.

В результате проделанной работы была создана система автоматизации процессов непрерывной интеграции и непрерывного развёртывания ЛИУС.

Список использованных источников и литературы

1. CI/CD concepts // GitLab [Электронный ресурс]. — Режим доступа: https://docs.gitlab.com/ee/ci/introduction/ (дата обращения: 06.04.2022).

2. Технология внедрения непрерывной интеграции в крупных высоконагруженных системах с минимизацией ошибок и временных потерь со стороны разработчиков. – Манаев Р.Г., Инновации и инвестиции. 2020. № 12. С. 127-130.

3. Отражение мировых практик программной инженерии и управления качеством программного обеспечения в отечественной it-отрасли: результаты исследования по регионам России. – Пащенко Д.С., Программная инженерия. 2020. Т. 11. № 2. С. 67-76.

4. Непрерывная интеграция, поставка или развертывание // Atlassian [Электронный ресурс]. — Режим доступа: https://www.atlassian.com/ru/continuous-delivery/principles/continuous-integration-vs-delivery-vs-deployment (дата обращения: 06.04.2022).

5. 5 common pitfalls of CI/CD—and how to avoid them // InfoWorld. [Электронный ресурс]. — Режим доступа: https://www.infoworld.com/article/3113680/5-common-pitfalls-of-cicd-and-how-to-avoid-them.html (дата обращения: 06.04.2022).

6. Автоматизация CI / CD с Jenkins // Zone3000 [Электронный ресурс]. — Режим доступа: https://devops-courses.zone3000.net/jenkins-ci-cd-na-blyudtse/ (дата обращения: 22.05.2022).

List of references

1. CI/CD concepts // GitLab, https://docs.gitlab.com/ee/ci/introduction/ (date of access: 06.04.2022).

2. Technology for the implementation of continuous integration in large, highly loaded systems with minimization of errors and time losses on the part of developers. – Manaev R.G., Innovations and investments. 2020. No. 12. P. 127-130.

3. Reflection of the world practices of software engineering and software quality management in the domestic IT industry: results of a study on the regions of Russia. – Pashchenko D.S., Software engineering. 2020. V. 11. No. 2. P. 67-76.

4. Continuous Integration, Delivery or Deployment // Atlassian — URL: https://www.atlassian.com/ru/continuous-delivery/principles/continuous-integration-vs-delivery-vs-deployment (date of access: 06.04.2022).

5. 5 common pitfalls of CI/CD—and how to avoid them // InfoWorld. — URL: https://www.infoworld.com/article/3113680/5-common-pitfalls-of-cicd-and-how-to-avoid-them.html (date of access: 06.04.2022).

6. Automation CI / CD c Jenkins // Zone3000. — URL: https://devops-courses.zone3000.net/jenkins-ci-cd-na-blyudtse/ (date of access: 22.05.2022).