Министерство образования Республики Беларусь
Учреждение образования
Белорусский государственный университет
информатики и радиоэлектроники

УДК 004.415

Корбовский
Денис Олегович

Система непрерывной интеграции с применением облачных технологий

АВТОРЕФЕРАТ
диссертации на соискание степени
магистра информатики и вычислительной техники
по специальности 1-40 81 01 – Информатика и технологии разработки
программного обеспечения

Минск 2020

Работа выполнена на кафедре информатики учреждения образования «Белорусский государственный университет информатики и радиоэлектроники»

Научный руководитель: **БАХТИЗИН Вячеслав Вениаминович,** кандидат технических наук, профессор кафедры программного обеспечения информационных технологий Учреждения образования «Белорусский государственный университет информатики и радиоэлектроники»

Рецензент: **ЧУБАРОВ Сергей Ильич,** кандидат физико-математических наук, доцент кафедры информационных технологий в образовании Учреждения образования «Белорусский государственный педагогический университет»

Защита диссертации состоится «25» июня 2020 г. года в $15^{\underline{00}}$ часов на заседании Государственной экзаменационной комиссии по защите магистерских диссертаций в учреждении образования «Белорусский государственный университет информатики и радиоэлектроники» по адресу: 220013, Минск, ул. Гикало, 9, копр. 4, ауд. 111, тел. 293-85-91, e-mail: inform@bsuir.by

С диссертацией можно ознакомиться в библиотеке учреждения образования «Белорусский государственный университет информатики и радиоэлектроники».

ВВЕДЕНИЕ

Современные реалии рынка требуют от любого продукта быстрого появления, высокого качества и постоянной поддержки. Любое промедление может стоить бизнесу большого числа потенциальных клиентов. В этих условиях скорость и качество разработки, тестирования и поддержки являются критическими. Любая ошибка в программном коде, лишающая возможности пользователя сделать заказ, долгая реализация необходимой функциональности или даже низкая скорость загрузки страницы могут снизить прибыль и обанкротить фирму.

В связи с этим помимо развития непосредственно технологий для разработки становится критически важным развивать процессы и технологии, отвечающие за инфраструктуру, этапы интеграции, доставки, тестирования, масштабируемости и отказоустойчивости. В последнее время невозможно представить проект, разрабатывающий коммерческий продукт, который не будет иметь системы непрерывной интеграции.

Многие крупные продукты, находящиеся на рынке очень долгое время, имеют устаревшие технологии, устоявшиеся процессы и зачастую проблемы с адаптированием под современный рынок. Некоторые из них пытаются избавиться от этих проблем и сталкиваются с разнообразием существующих архитектур, технологий и подходов. Не существует единого правильного решения для любого продукта, однако есть общепринятые стандарты, практики и опыт огромного числа проектов на рынке. К примеру, крупные монолитные системы разбиваются на микросервисы и легко развиваются в соответствии с нуждами потребителей.

В связи с этим становится очень важным следить за трендами в технологиях и применять их в соответствии с особенностями своего продукта. Многие крупные компании разрабатывают целые системы, помогающие разрабатывать и доставлять свои разработки до конечного пользователя.

ОБЩАЯ ХАРАКТЕРИСТИКА РАБОТЫ

Актуальность темы исследования

Тема данной диссертации является крайне актуальной в современных реалиях, когда автоматизация технологических процессов является одним из решающих факторов улучшения качества продукции, повышения производительности и улучшения условий труда. Все существующие сферы в той или иной степени оснащаются средствами автоматизации. Непрерывная интеграция программного обеспечения является одним из основных средств автоматизации процессов в сфере информационных технологий.

Степень разработанности проблемы

Имеется большое количество технологий и подходов, позволяющих решить данную проблему, однако из-за их большого количества легко потеряться в разнообразии и выбрать не подходящее решение.

В рамках исследования затронуты различные современные практики и технологии, проведено их сравнение и построено универсальное решение по непрерывной интеграции с применением облачных технологий, на основе которого можно базировать большинство проектов, использующих наиболее популярные инструменты.

В открытых источниках большинство информации по данной проблеме является верхнеуровневой и не предоставляет понятных и конкретных шагов для настройки шаблонного проекта.

Цель и задачи исследования

Основной целью диссертации является построение автоматизированной и отказоустойчивой системы непрерывной интеграции исходного кода.

Можно выделить главные задачи, которые выполнены в рамках исследования:

1. Организация отказоустойчивости для инфраструктуры непрерывной интеграции
2. Проектирование непосредственно самой системы непрерывной интеграции.

Главная задача моделируемой системы – организация быстрого и эффективного процесса по анализу, сборке и тестированию ПО.

К основным требованиям можно отнести:

- Отказоустойчивость;
- Высокая скорость сборки, тестирования и развертывания ПО;
- Мониторинг;
- Логгирование.

Теоретическая и методологическая основа исследования

В основу диссертации легли современные практики и технологии организации автоматизации процессов на проектах по разработке программного обеспечения. Информационная база исследования сформирована на основе литературы, открытой информации, а также сведений из электронных ресурсов

Научная новизна

Теоретическая значимость работы заключается в детальном анализе современных подходов, сравнения их с предшествующими аналогами, описание преимуществ и недостатков современных решений.

Практическая значимость диссертации состоит в спроектированной системе непрервыной интеграции, размещенной в современном облачном сервисе, который позволит упрощать разработку продукта и расширять масштабы производства.

Основные положения, выносимые на защиту

1. Использование микросервисной архитектуры, позволяющей легко горизонтально масштабировать систему, строя сервисы вокруг бизнес-потребителей и разворачивая независимо с использованием полностью автоматизированной среды.
2. Реализация полностью автоматизированной среды на основе непрерывной интеграции, позволяющей гибко и просто настраивать все процессы и этапы разработки, тестирования и поддержки программного продукта.
3. Использование современных облачных решений в практике непрерывной интеграции, позволяющее упростить процесс настройки инфраструктуры, обеспечивает дополнительную безопасность, позволяет

экономить ресурсы, предоставляя мощности и абстракции для применения решений.

Публикации

Изложенные в диссертации основные положения и выводы опубликованы в 1 печатной работе, являющейся статьей в международном научном журнале «Научные вести».

Структура и объем работы

Диссертация состоит из введения, общей характеристики работы, трех глав с краткими выводами по каждой главе, заключения, библиографического списка и приложений.

В первой главе приведен обзор методологий разработки ПО, история процесса разработки, современные представления и способы решения проблем, возникающих при проектировании систем и разработке.

Во второй главе спроектирована система, решающая проблемы, приведенные в первой главе, а именно автоматизирующая процессы разработки, значительно уменьшая затрачиваемое время и снижая риск ошибок из-за человеческого фактора.

В третьей главе описан процесс реализации спроектированной системы с обоснованием выбранных технологий и интеграцией системы в облачное решение.

В приложении представлены публикации автора и фрагменты программного кода системы.

Общий объем диссертационной работы составляет 80 страниц. Из них 53 страниц основного текста, 27 иллюстраций на 8 страницах, 5 таблицы на 3 страницах, список собственных публикаций соискателя из 11 наименований на 1 странице, 1 приложение на 8 страницах.

ОСНОВНОЕ СОДЕРЖАНИЕ РАБОТЫ

Во **введении** рассмотрено современное состояние проблемы автоматизации процессов в сфере информационных технологий, сформулировано направление и задача исследования и разработки решения проблемы.

В **общей характеристике работы** показана актуальность темы непрерывной интеграции и возможности применения современных решений на базе облачных технологий, сформулированы объект, предмет и основная гипотеза, положенные в основу диссертационной работы, обозначена связь работы с запросами реального сектора, указана опубликованность и структура диссертации.

В **первой главе** проведен обзор методологий разработки ПО, описаны возникающие проблемы и их история, сформулирована идея подхода непрерывной интеграции. В качестве графического представления приведен процесс непрерывной интеграции в общем виде на рисунке 1.

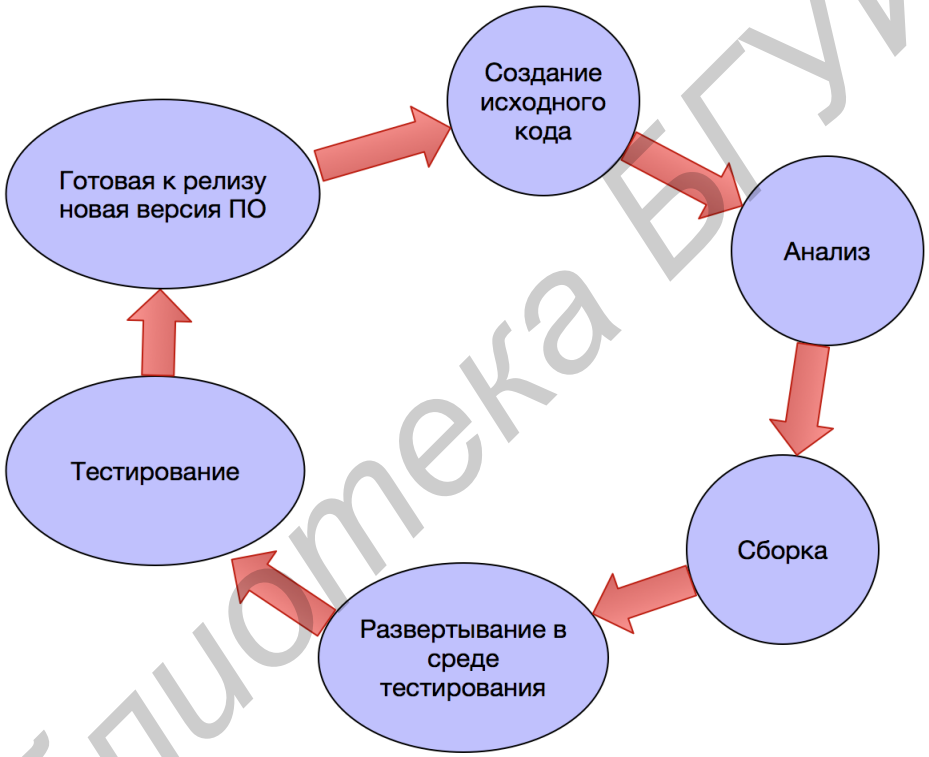

Рисунок 1 – Процесс непрерывной интеграции

Главным принципом непрерывной интеграции установлено значительное увеличение количества выпусков новых версий ПО. Связано это с тем, что выпуск новой версии это самая сложная и трудозатратная операция при разработке ПО.

Помимо процесса непрерывной интеграции описаны архитектуры систем и какие принципы применяются для их расширения. Выделены два основных направления: вертикальное масштабирование и горизонтальное масштабирование. В случае первого установлено, что его эффективность

является очень ограниченной вследствие отсутствия необходимого прироста производительности информационной системы с увеличением мощностией.

Выделены основные модели горизонтального масштабирования, такие как масштабирование по оси X, являющееся классической моделью, использующей балансирование нагрузки, масштабирование по оси Y, представляющее собой декомпозицию системы на различные сервисы и масштабирование по оси Z, основанное на клонировании серверов приложений. Выделены основные различия между данными подходами.

В качестве решения, основанного на горизонтальном масштабировании, предложена микросервисная архитектура, набирающая популярность в последнее время. Для обоснования решения проведено исследование ее архитектуры, описаны процессы сборки и развертывания, определено использования непрерывной интеграции в микросервисной архитектуре.

Во **второй главе** выполнялось проектирование системы непрерывной интеграции с внедрением облачных технологий. Построена структура системы, которая приведена на рисунке 2.

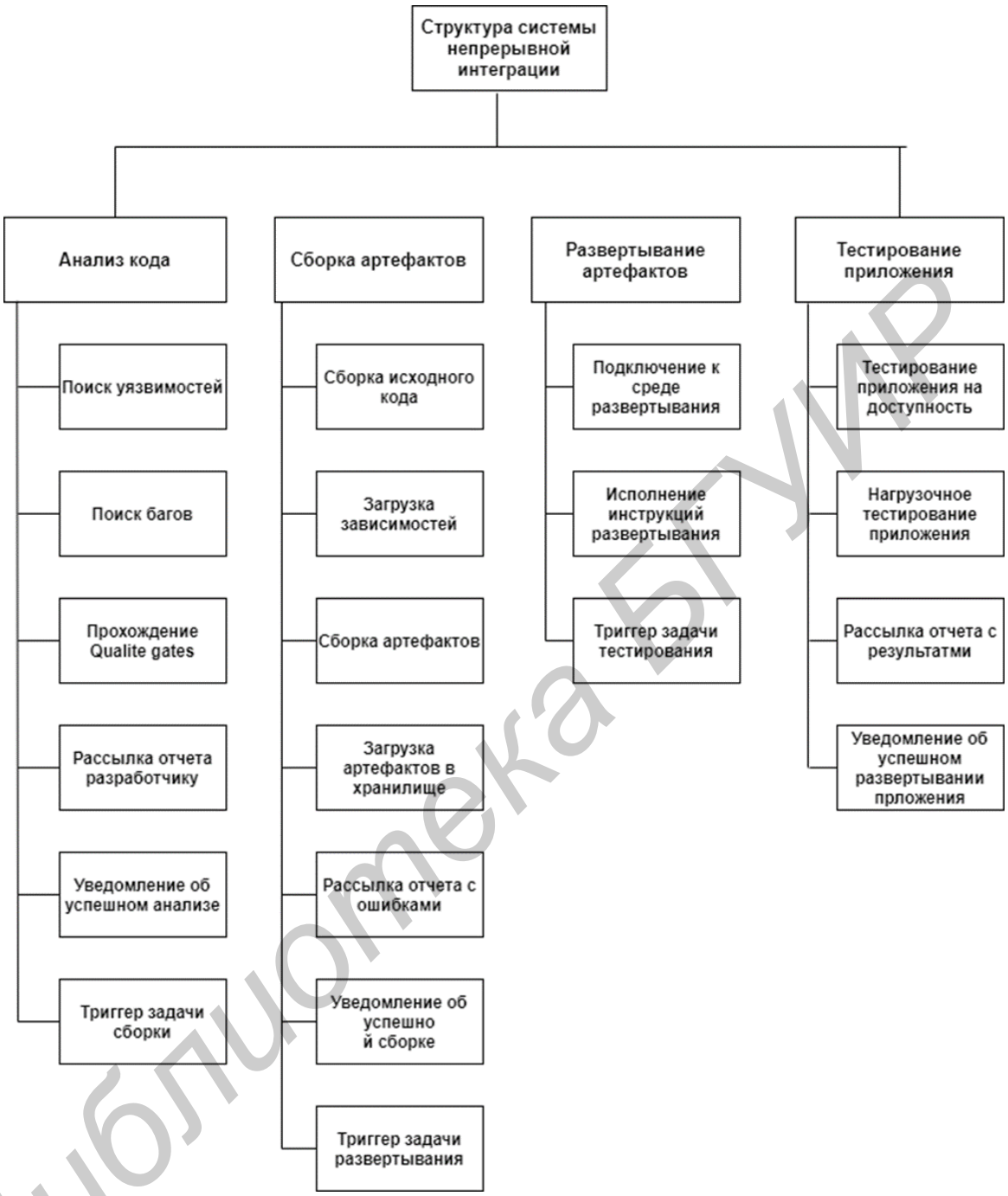

Рисунок 2 – Схема бизнес-процесса по разработки ПО

На основе структуры системы спроектирована ее архитектура. Архитектура системы непрерывной интеграции приведена на рисунке 3.

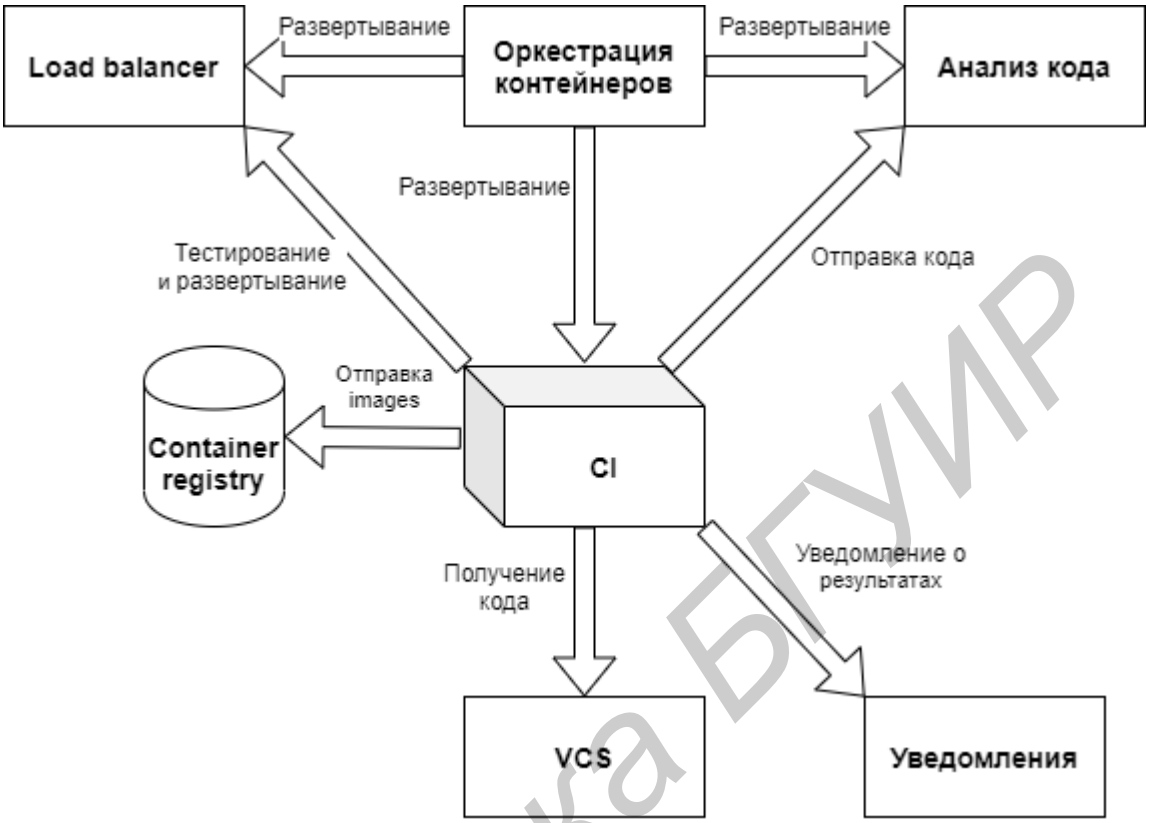

Рисунок 3 - Архитектура системы непрерывной интеграции

Выполнен выбор и обоснование средств реализации, произведено сравнение сервисов и технологий, выведены основные критерии для системы с микросервисной архитектурой:

- Отказоустойчивость и высокая доступность системы;
- Высокая скорость горизонтального масштабирования сервисов;
- Обеспечение механизма автоматического развертывания приложения (система непрерывной доставки);
- Релиз новых версий приложения с минимальным временем простоя
- Гибкость системы (быстрая адаптация системы для любых нужд проекта)
- Политики безопасности (настройка и управление ролями)

Выбран наиболее подходящий поставщик облачных ресурсов для интеграции реализуемой системы, система контроля версий, сервер непрерывной интеграции и хранилище артефактов.

Для выбора хранилища артефактов учтены такие критерии, как:

- Высокая скорость загрузки и выгрузки артефактов;

– Отказоустойчивость и высокая доступность;
– Интеграция со сторонними сервисами.

Архитектура спроектированной системы приведена на рисунке 4.

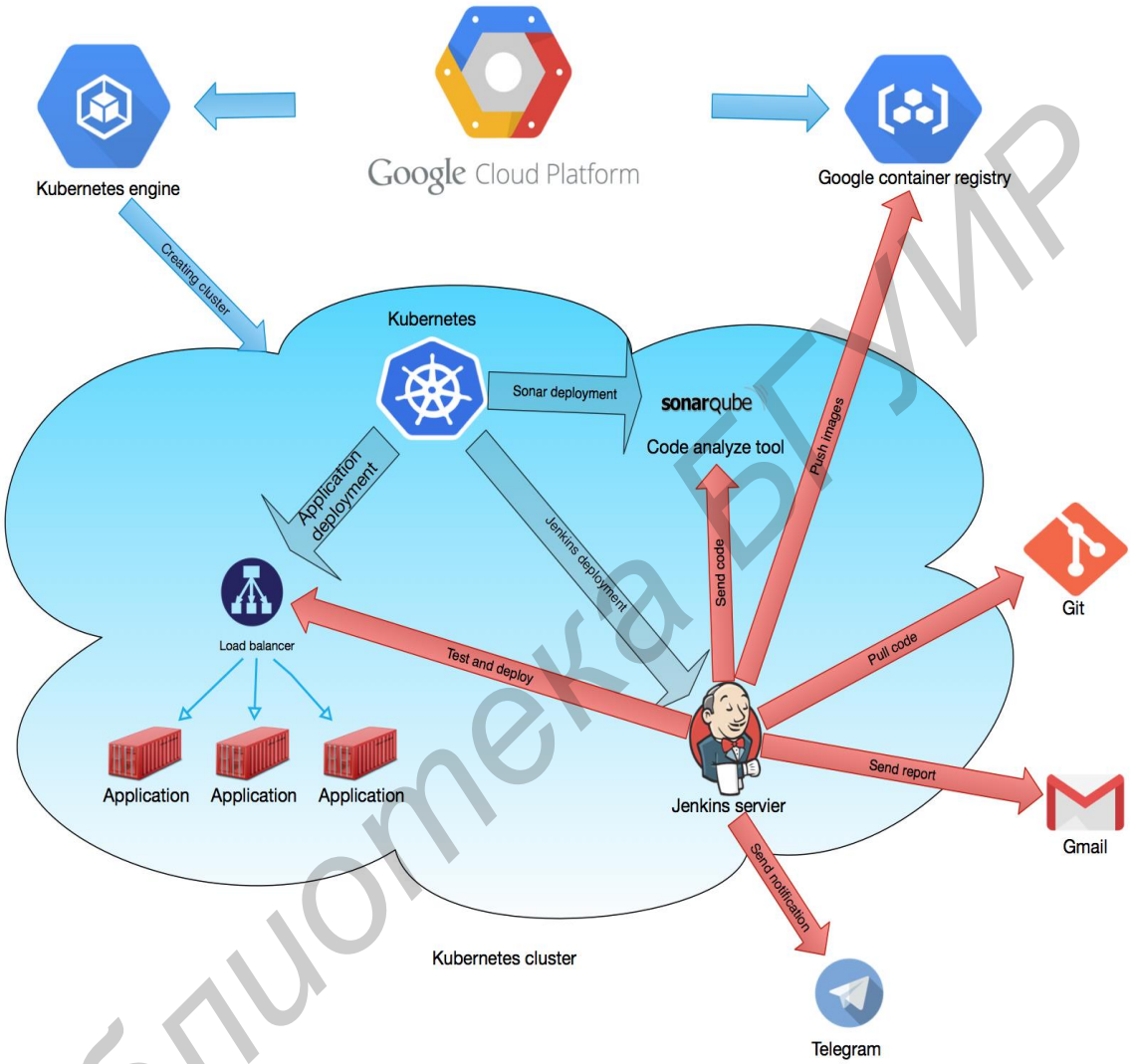

Рисунок 4 – Архитектура системы непрерывной интеграции.

Проектируемая система проанализирована относительно применимости в бизнес-процессах, выделены основные функций системы и построена схема бизнес-процесса для одного из главных процессов – разработки новых требований.

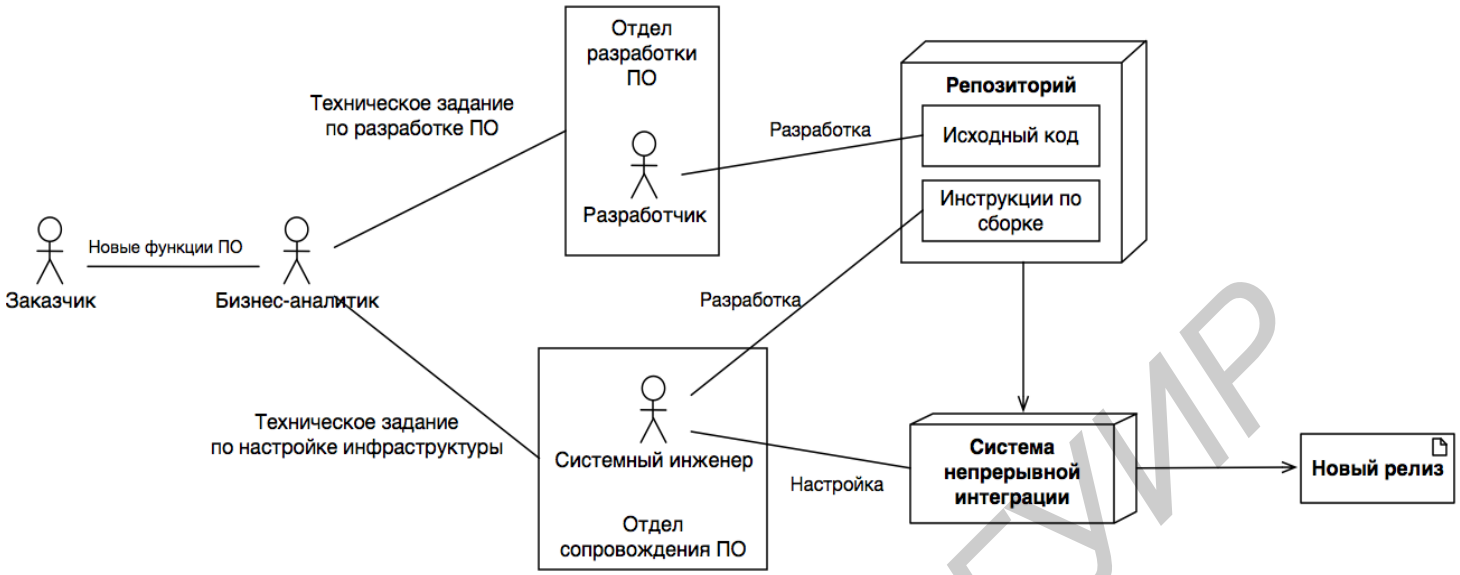

Рисунок 5 – Схема бизнес-процесса по разработки ПО

Приведена общая характеристика реализуемой системы, в том числе построен график информационной системы непрерывной интеграции и выделены основные компоненты системы.

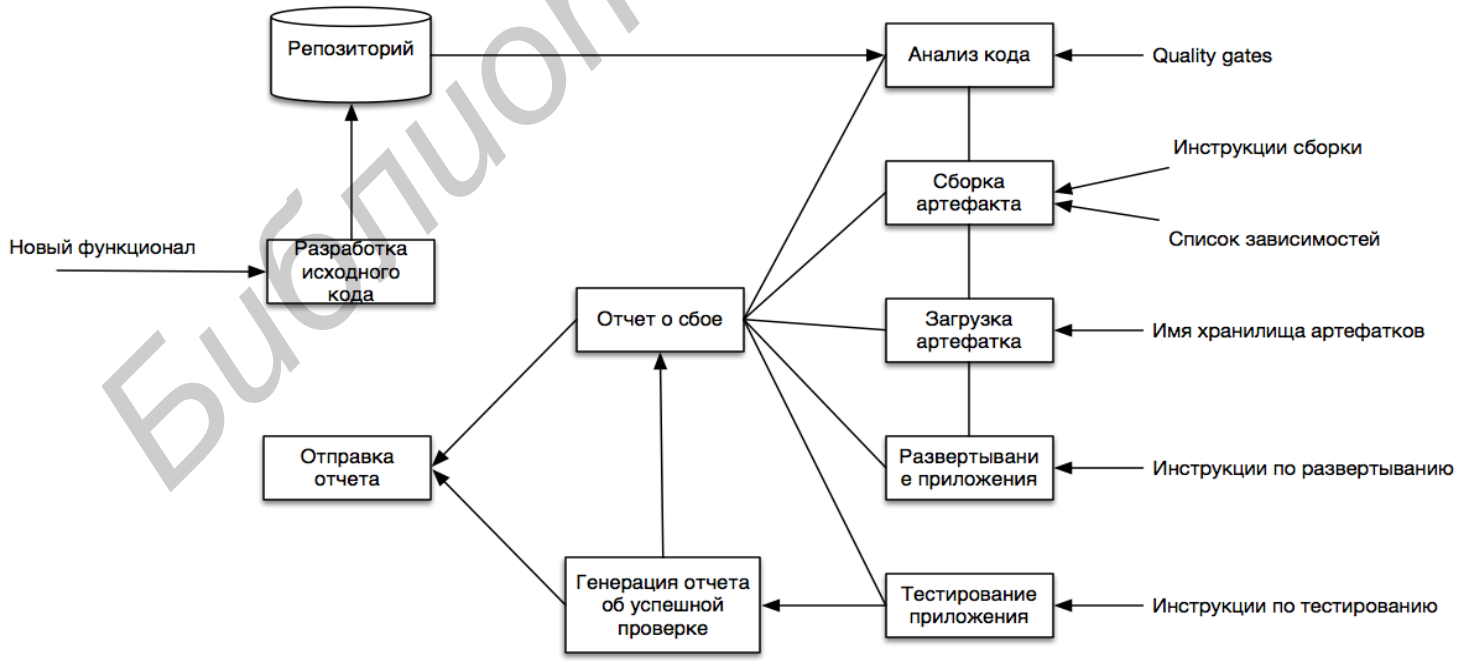

Рисунок 6 – Информационное обеспечение системы непрерывной интеграции

В качестве входных данных в системе выступают элементы, поступающие из репозитория в систему непрерывной интеграции. Входными данными являются весь код и инструкции, поступающие в систему непрерывной интеграции. Они сведены в таблицу ниже:

Таблица 1 – Входные данные системы непрерывной интеграции

Наименование	**Источник**	**Получатель**	**Периодичность**	**Описание**
Исходный код	Отдел разработки	Разработчик	Новый спринт	Файлы с расширением *.py*
Quality gates	Отдел тестирования	Программный модуль	Запуск модуля	Файл в формате *XML*
Инструкция сборки	Отдел разработки	Программный модуль	Новый спринт	Файл с именем *Dockerfile*
Список зависимостей	Отдел разработки	Программный модуль	Новый спринт	Список библиотек и модулей
Имя хранилища артефактов	Отдел сопровождения ПО	Программный модуль	Смена разрабатываемого приложения	*URL* адрес
Инструкции по развертыванию	Отдел сопровождения ПО	Программный модуль	Смена разрабатываемого приложения	Файл в формате *YAML*

В качестве выходных данных программного модуля выступают результаты анализа, сборки и тестирования, а также отчеты о сбоях системы, представленные в различных форматах и сопутствующие элементы, такие как письма, уведомления и прочее.

Таблица 2 – Выходные данные системы непрерывной интеграции

Наименование	**Источник**	**Получатель**	**Периодичность**	**Описание**
Письмо с информацией о результатах	Программный модуль	Администратор системы	Окончание работы модуля	Письмо в системе Outlook 365 Office
Отчеты в различных форматах	Программный модуль	Любой участник разработки	Окончание работы модуля	Файл с логами ошибок

Реализован безопасный подход к работе с системой непрерывной интеграцией, определены роли и должности каждого участника в проекте и распределены права доступа, основываясь на потребностях этих участников.

Выделены два вида пользователей:

- администратор;
- разработчик.

Администратор – является проектировщиком системы, который имеет доступ от всех внутренних компонентов системы непрерывной интеграции.

Задачами администратора являются:

- настройка системы непрерывной интеграции;
- поддержка;
- мониторинг.

Разработчик – является пользователем системы, его основными задачами являются:

- разработка ПО;
- исправление багов;
- просмотр кода.

Так же для обеспечения полной автоматизации выполнения задач сборки и развертывания приложений, используются сервисные аккаунты. На рисунке 7 проиллюстрированы области прав всех пользователей в системе непрерывной интеграции.

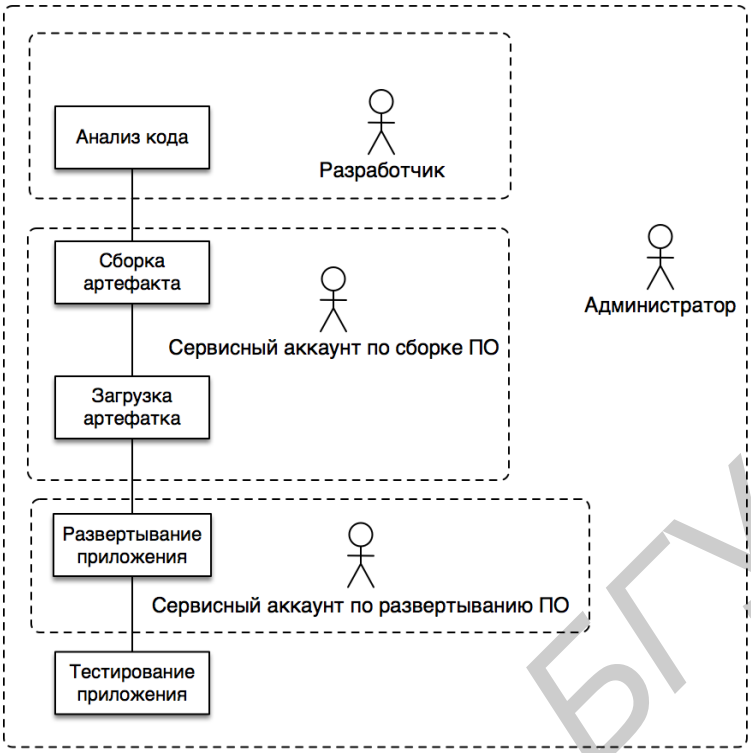

Рисунок 7 – Области прав всех пользователей в системе непрерывной интеграции

В **третьей главе** выполнена реализация спроектированной системы. Реализация начиналась с базовых настроек в облачной платформе, таких как создание проекта и подключение сервисов, настройки способов оплаты за услуги, следующей задачей стояло реализация первой функционально-важной части системы, отвечающей за анализ исходного кода. Для этого создана *Jenkins* задача, которая выполняет следующие операции:

- Загрузка исходного кода из удаленного *Git* репозитория
- Подключение к *SonarQube* и передача параметров для анализа исходного кода
- Нотификация в Telegram при успешном выполнении задачи *Jenkins*
- Отправление логов на почту, при сбое выполнения задачи *Jenkins*
- Триггер *Jenkins* задачи *Build-Push-Deploy*

Следующей не менее важной частью является непосредственно реализация сборки и развертывания приложения. Данный процесс состоит из следующих шагов:

- Авторизация *Google Service Account;*
- Настройка окружения *Docker;*

- Сборка проекта;
- Загрузка образа в *Google Cloud Registry;*
- Настройка доступа к кластеру *Kubernetes;*
- Развертывание приложения в кластер;
- Нотификация в *Telegram* при успешном выполнении задачи *Jenkins;*
- Отправление логов на почту, при сбое выполнения задачи *Jenkins*;
- Триггер *Jenkins* задачи *Automated-Tests.*

Для доступа к облачным сервисам Google из системы непрерывной интеграции Jenkins сконфигурирован отдельный сервисный аккаунт с правами администратора к кластеру и хранилищу артефактов. Непосредственно сборка артефактов производится по следующим шагам:

- Сборка исходного кода;
- Загрузка зависимостей;
- Сборка *Docker*-образа.

В зависимости от исходного кода меняется операция сборки. Для интерпретируемых языков сразу начинается загрузка зависимостей, а для компилируемых используется инструмент по сборке кода, который скачивает нужные библиотеки и компилирует код.

Сборка артефактов идет следующим шагом. Ее конфигурация происходит за счет созданных *Dockerfile*, описывающих создание микросервисов. Данные артефакты загружаются в облачное хранилище *Google Cloud Registry* и выполняется непосредственный деплой приложения. За развертывание приложения отвечает кластер Kubernetes, конфигурация которого происходит в файле с расширением *.yaml*. Этот файл создает сервисы для приложения, запросы к которым распределяются за счет балансировщика нагрузки. Для него выделен внешний IP-адрес и к нему может получить доступ любой пользователь с доступом в Интернет. Схема балансировщика нагрузки изображена на рисунке 5.

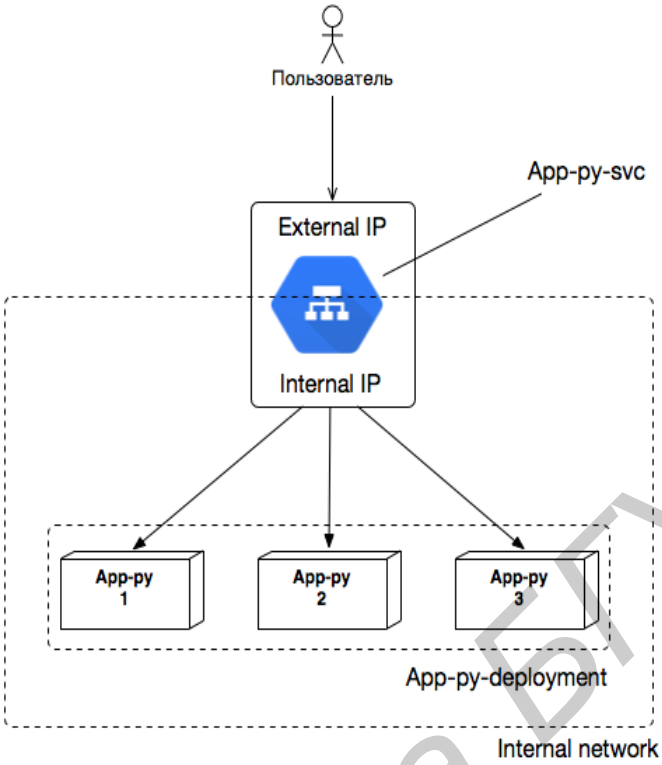

Рисунок 5 – Балансировщик нагрузки

Финальным этапом является тестирование работоспособности приложения, выполняемое за счет создания отдельной задачи в системе непрерывной интеграции, выполняющей следующие операции:

- Нахождение *IP* адреса приложения;
- Проведение нагрузочных тестов;
- Генерация отчета.

В случае ошибки отправляется письмо на почту с логом действий и возникшего исключения. В случае успешного тестирования сервисов, выполняемого инструментом *Apache HTTP server benchmarking tool*, отправляется уведомление в Telegram о работоспособности приложения.

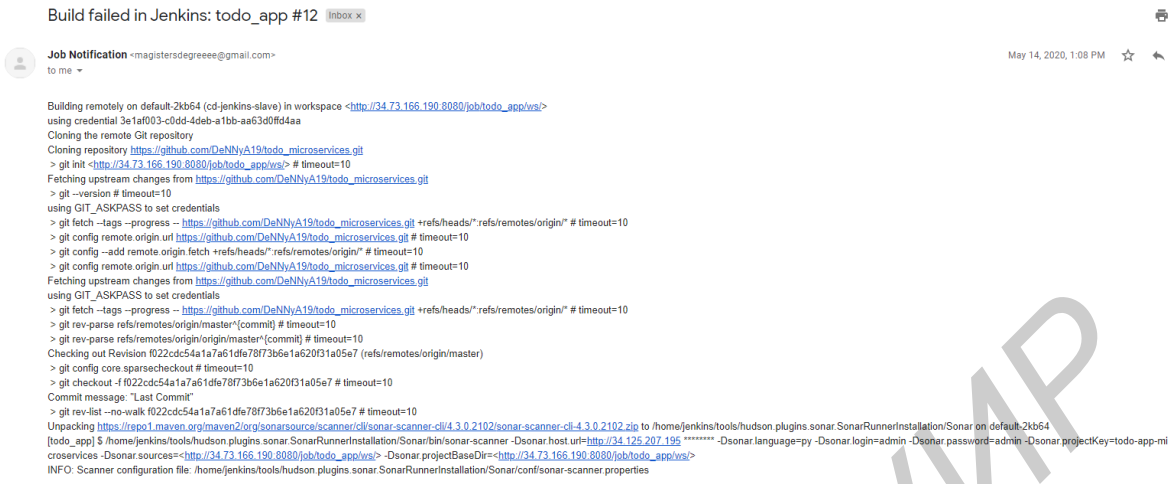

Рисунок 6 – Письмо с ошибкой

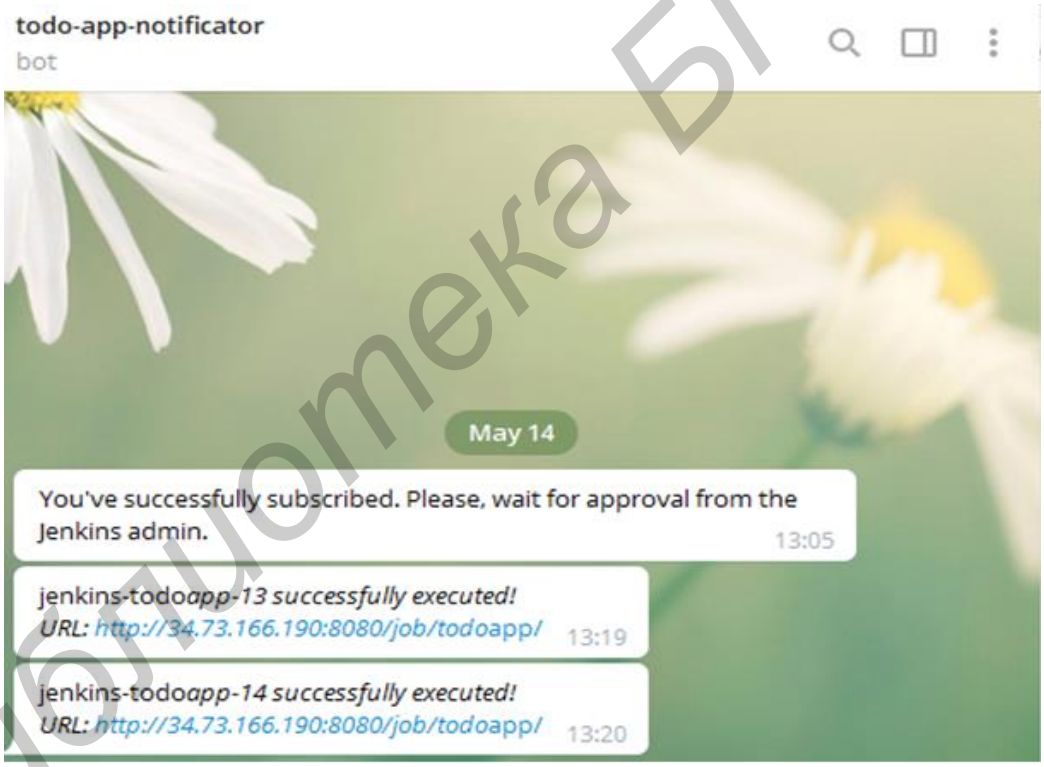

Рисунок 7 – Уведомление об успешном разворачивании приложения

В завершении выполнен анализ реализованной системы. Разработчик не должен проводить никаких ручных операций, и единственным триггером начала сборки является – загрузка кода (коммит) в *master* ветку в системе контроля версий.

Таким образом разработчику доступна только информация о процессе сборки в виде почтовых сообщений с логами ошибок и *Telegram* уведомлений. Задачей администратора системы является поддержка и мониторинг системы, таким образом ему приходят почтовые сообщения с логами ошибок, *Telegram* нотификации и доступна панель мониторинга *Kubernetes, Google Cloud Platform* и *Jenkins*.

За счёт применения облачных технологий упрощен процесс настройки системы непрерывной интеграции, разрабатываемого приложения и вспомогательных сервисов. За счет единой облачной инфраструктуры большинство конфигураций вынесено в скрипты, которые делают процесс повторяемым, контролируемым и отказоустойчивым. Текущая реализация системы позволяет значительно сократить денежные и временные ресурсы, связанные с поддержкой безопасности и масштабирования. Применение облачных технологий позволило выполнить все настройки в рамках одной инфраструктуры, перенесенной на сторону облачного провайдера.

В архитектуру системы заложена возможность горизонтального масштабирования. Использование подхода организации функциональности в виде микросервисов и использования *Google Cloud Load Balancer* позволяет иметь возможность поддерживать мощности от небольшого приложения, разрабатываемого в учебных целях до коммерческого приложения с большим потоком пользователей.

Приведено финансовое сравнение между использованием ресурсов облачного провайдера и локальной установке сервера.

Таблица 3 - Сравнение стоимости локальной инфраструктуры и GCP

Провайдер	-	GCP – South Carolina (East)
Имя сборки	-	Custom
CPU	16 CPUs	16 CPUs
RAM	32 GB	30 GB
Цена	-	357.25$
Хранилище	480GB SSD	667GB SSD
Цена	-	226.39$ + 0$ (20K IOPS)
Итоговая цена	**7669$**	**645.62$/месяц**

Таблица 4 - Сравнение стоимости инфраструктуры AWS и GCP

Провайдер	Azure - East	GCP – South Carolina (East)
Имя сборки	D5 V2	Custom
CPU	16 CPUs	16 CPUs
RAM	30 GB	30 GB
Цена	613.42$	357.25$
Хранилище	500GB SSD	667GB SSD
Цена	62.50$ + 1040$ (16K IOPS)	226.39$ + 0$ (20K IOPS)
Итоговая цена	**1715.92$/месяц - по требованию 1102.50$/месяц - 3 года**	**645.62$/месяц**

Таблица 5 - Сравнение стоимости инфраструктуры Azure и GCP

Провайдер	AWS – Virginia	GCP – South Carolina (East)
Имя сборки	C4.4xlarge	Custom
CPU	16 CPUs	16 CPUs
RAM	56 GB	30 GB
Цена	870.48$	419.23$
Хранилище	512GB SSD	667GB SSD
Цена	73.22$ x 10 (18.4 RAID0 IOPS)	226.39$ + 0$ (20K IOPS)
Итоговая цена	**1602.68$/месяц - по требованию**	**645.62$/месяц**

ЗАКЛЮЧЕНИЕ

Основные результаты диссертации

1. Разработана система непрерывной интеграции, поддерживающая все этапы разработки продукта.

2. Произведена интеграция системы в облачный сервис, реализована ее масштабируемость, безопасность и удобство конфигурации.

3. Спроектировано базовое шаблонное приложение на основе микросервисов, которое внедрено в систему непрерывной интеграции с применением облачных технологий.

Рекомендации по практическому использованию результатов

Полученные результаты применены в рабочих целях на коммерческом проекте по разработке и тестированию программного продукта.

СПИСОК ПУБЛИКАЦИЙ СОИСКАТЕЛЯ

Тезисы конференций

1. Корбовский, Д.О Сервис по аренде автомобилей на основе мобильной платформы Android / Корбовский, Д.О, Глухова Л.А // материалы 55-ой науч. конф. аспирантов, магистрантов и студентов БГУИР, Минск, Респ. Беларусь, 22–26 апреля 2019 г. / УО «БГУИР». – Минск, 2019. – С.204–205.

Статьи в рецензируемых журналах

2. Сервис по аренде автомобилей на основе мобильной платформы Android / Д.О. Корбовский, И.Ю. Фалько // Международный научный журнал «Научные вести». – 2016. – № 12(17) -2019. – С. 305-309.

3. Кроссплатформенное программное средство подбора и управления арендой серверного оборудования / И.Ю. Фалько, Д.О. Корбовский, // Международный научный журнал «Научные вести». – 2016. – № 12(17) -2019. – С. 363-368.

4. Система непрерывной интеграции с применением облачных технологий / Д.О. Корбовский // Международный научный журнал «Научные вести». – 2020. – № 6(23) -2020. – С. 184-189.