МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

План-проспект дипломной работы

Тема: Автоматизация процессов поддержки промежуточной аттестации студентов

Финальный объем – 80 стр

Студентка гр. 5304 _____ Орлова В.В.

Преподаватель _____ Заславский М.М.

Санкт-Петербург
2020

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Объем 2-3 стр

Непрерывная интеграция и непрерывное развертывание – это две современные практики разработки программного обеспечения. Непрерывная интеграция (CI) — это процесс автоматизации сборки и тестирования кода каждый раз, когда член команды вносит изменения в управление версиями. Непрерывное развертывание (CD) можно рассматривать как расширение непрерывной интеграции, и оно представляет собой процесс автоматического развертывания приложения после успешного завершения CI. Цель состоит в том, чтобы минимизировать время выполнения заказа - время между написанием одной новой строки кода во время разработки и началом использования этого нового кода живыми пользователями в производстве.

Есть много преимуществ для практики CI / CD. В этой работе не будет подробно рассказываться о каждом преимуществе, но необходимо выделить некоторые из них:

- Преимущества непрерывной интеграции:
    - Меньше ошибок
    - Меньше переключения контекста, так как разработчики получают предупреждение, как только они ломают сборку
    - Затраты на тестирование снижены
    - Команда QA тратит меньше времени на тестирование
- Преимущества непрерывного развертывания:
    - Релизы менее рискованны
    - Легкий выпуск
    - Клиенты видят непрерывный поток улучшений
    - Ускорение разработки, так как нет необходимости приостанавливать разработку для выпусков

Именно поэтому данная работа ставит перед собой задачи исследования процессов непрерывной интеграции и доставки, а также настройки CI/CD для автоматизации процессов тестирования, построения и развертывания приложения.

1. ОБЗОР ПРЕДМЕТНОЙ ОБЛАСТИ

1.1. Используемые термины

Объем 3-4 стр

Непрерывная интеграция (Continuous Integration – CI) — это практика разработки программного обеспечения, которая заключается в слиянии рабочих копий в общую основную ветвь разработки несколько раз в день и выполнении частых автоматизированных сборок проекта для скорейшего выявления потенциальных дефектов и решения интеграционных проблем [5];

Непрерывная доставка (Continuous delivery – CD) – является продолжением непрерывной интеграции, дающая возможность быстро и эффективно выпускать новые изменения для своих клиентов. Это означает, что помимо автоматизации тестирования также возможность автоматизировать процесс выпуска и возможность в любой момент развернуть приложение, нажав кнопку [2];

Непрерывное развертывание (Continuous deployment – CD) – идет на один шаг дальше, чем непрерывная доставка. С этой практикой каждое изменение, которое проходит все стадии производственного процесса, передается клиентам. Вмешательство человека не требуется, и только неудачный тест не позволит внедрить новое изменение в производство [1];

Node.js – это платформа для разработки приложений на языке JavaScript, позволяющая быстро создавать приложения, работающие в сети [7];

Jenkins — программная система с открытым исходным кодом на Java, предназначенная для обеспечения процесса непрерывной интеграции программного обеспечения [1];

Docker — программное обеспечение для автоматизации развёртывания и управления приложениями в средах с поддержкой контейнеризации. Позволяет «упаковать» приложение со всем его окружением и зависимостями в контейнер, который может быть перенесён на Linux-систему, а также предоставляет среду по управлению контейнерами [2];

GitHub — крупнейший веб-сервис для хостинга IT-проектов и их совместной разработки [2];

Github Actions — функциональность, позволяющая автоматизировать рабочий процесс;

DockerHub — открытый репозиторий образов контейнеров Docker;

Amazon Web Services (AWS) — коммерческое публичное облако, поддерживаемое и развиваемое компанией Amazon с 2006 года. Предоставляет подписчикам услуги как по инфраструктурной модели (виртуальные серверы, ресурсы хранения), так и платформенного уровня (облачные базы данных, облачное связующее программное обеспечение, облачные бессерверные вычисления, средства разработки);

Elastic Beanstalk — сервис оркестрации, предлагаемый Amazon Web Services для развертывания приложений, которые управляют различными сервисами AWS, включая S3, CloudWatch и др.

1.2. Актуальность разработки

Объем 1 стр

1.3. Обзор аналогов в предметной области

Объем 2 стр

Принцип отбора аналогов: возможность организации непрерывной интеграции, непрерывной доставки и развертывания приложения, написанного на Node.js.

- Jenkins

Jenkins — это проект с открытым исходным кодом, написанный на Java, который работает на Windows, macOS и других Unix-подобных операционных системах.

В Jenkins доступна обширная библиотека плагинов. Плагины Jenkins охватывают пять областей: платформы, пользовательский интерфейс, администрирование, управление исходным кодом и, наиболее часто, управление сборкой. Хотя другие инструменты CI предоставляют аналогичные функции, им не хватает полной интеграции плагинов, которую имеет Jenkins. Более того, сообщество Jenkins поощряет своих пользователей расширять функциональность новыми функциями, предоставляя учебные ресурсы.

- Travis CI

Как одно из старейших решений CI, Travis завоевал доверие многих пользователей. Он имеет большое и полезное сообщество, которое приветствует новых пользователей и предоставляет множество учебных пособий.

Travis CI может тестировать на Linux и MacOS. Предлагая множество автоматических опций CI, Travis устраняет необходимость в выделенном сервере, так как он размещен в облаке.

Ценовая политика. Первые 100 сборок бесплатны. В противном случае существует четыре тарифных плана для хобби-проектов (69 долларов в месяц)

и для небольших, растущих и более крупных команд (от 129 до 489 долларов в месяц).

- TeamCity

TeamCity от JetBrains - это надежный и качественный CI-сервер. Команды часто выбирают TeamCity для большого количества функций аутентификации, развертывания и тестирования, а также поддержки Docker. TeamCity работает сразу после установки, никаких дополнительных настроек или настроек не требуется. Он обладает рядом уникальных функций, таких как подробные хронологические отчеты, мгновенные отзывы о неудачных тестах и повторное использование настроек, поэтому вам не нужно дублировать код.

Ценовая политика TeamCity предлагает бесплатную версию с полным доступом ко всем функциям продукта, но она ограничена 100 конфигурациями сборки и тремя агентами сборки. Добавление еще одного агента сборки и 10 конфигураций сборки в настоящее время стоит $ 299.

- GitHub Actions

GitHub Actions — это CI/CD система, интегрированная с GitHub. В первом приближении можно думать о ней, как об аналоге TeamCity или Jenkins, предоставляемом в виде сервиса. Сервис бесплатен для открытых проектов, и даже для закрытых, если ваши билды собираются не слишком долго и/или не слишком часто.

Первый и главный плюс GitHub Actions - это простота настройки. Действия GitHub работают в облаке, но у вас также есть возможность запустить его локально.

По этой причине сервер Jenkins требует установки, тогда как GitHub Actions не нуждается в этом. Следовательно, процесс настройки в GitHub

Actions очень удобен. Кроме того, GitHub Actions - это серия запусков докеров. Для этого требуется только сборка докера и запуск докера. Это упрощает запуск и отладку.

- Gitlab CI

Gitlab предоставляет уникальную возможность получить одновременно бесплатный приватный репозиторий и бесплатный CI/CD из коробки в том же месте. Для сохранения баланса он конфигурируется не менее уникальным способом через конфиг-файл.

1.3.1. Критерии сравнения аналогов

- Облачный хостинг

Размещение возможно на физическом сервере или в облаке, аналог должен иметь в возможностях второй вариант.

Обоснование критерия: облачные инструменты размещаются на стороне поставщика, требуют минимальных настроек и могут быть скорректированы по требованию в зависимости от ваших потребностей.

Размещенные решения избавляют от трудностей установки, предлагая большую масштабируемость. Этот вариант более подходит для тестируемого приложения т.к. не требует поддержки физического сервера.

- Актуальные интерфейс и документация

Обоснование критерия: некоторые инструменты могут сделать процесс сборки намного проще, чем другие, благодаря их четкому и понятному графическому интерфейсу и UX. Тщательно продуманный интерфейс и подробная документация могут сэкономить ваше время на этапе адаптации.

- Экономичность

Обоснование критерия: поскольку проект не коммерческий и поддерживается силами исключительно автора работы, цена является одним из главных критериев по понятным причинам. Поскольку упор работы сделан на изучение именно непрерывной интеграции, в дальнейшем предполагается многократный запуск пайплайнов, поэтому оптимальным будет бесплатное решение.

Сравнение аналогов по критериям представлено в таблице 1.3.1.1.

Таблица 1.3.1.1. – Сравнения аналогов по критериям

	Облачный хостинг	Актуальные интерфейс и документация	Экономичность
Jenkins	Физический сервер, -	Устаревшие, -	Бесплатный, +
TeamCity	Физический сервер или облако +	+	Дорогой, – (от 299$)
Travis CI	+	+	Дорогой, – (от 69$/month)
GitHub Actions	+	+	Бесплатный, +
Gitlab CI	+	+	Дешевый, +- (от 15$/месяц)

1.3.2. Выводы по итогам сравнения

Облачный хостинг, как и актуальные интерфейс и документация, являются крайне важными критериями, поэтому решение с использованием Jenkins, несмотря на огромную популярность, не подходит для решения

поставленной задачи. Технология осваивается автором практически с нуля, поэтому документация и туториалы должны содержать актуальную информацию, а интерфейс – быть интуитивно понятным.

Физический сервер был бы приемлем, если бы у студентов была возможность постоянно его поддерживать, но, поскольку аналоги предлагают облачные решения, проще обратиться к ним и избежать, помимо прочего, сложностей настройки сервера.

Экономичность – другой важнейший критерий, т.к. проект не коммерческий и поддерживается силами исключительно автора работы. Большинство решений предлагают учебные версии, но, поскольку планируется продолжительная разработка, пробные периоды, а также ограничение по количеству задач/времени крайне нежелательны. Поэтому дорогие решения TeamCity и Travis CI рассматриваться дальше не будут. Gitlab CI предлагает относительно дешевое использование, но проигрывает в сравнении с бесплатными Jenkins и GitHub Actions.

1.4. Выбор метода решения

Объем 2 стр

Готовым решением будет служить сформированный после анализа исходного пула возможных решений технология непрерывной интеграции. Выбранная технология должна соответствовать следующим требованиям:

- Облачный хостинг – для размещения на стороне поставщика, минимальных затрат сил для начальных и последующих настроек и поддержки.
- Актуальные интерфейс и документация – для быстрого освоения новой технологии, а также последующего удобства работы с системой.

- Экономичность – для возможности поддержания учебного приложения без трудностей и ограничений, связанных с работой с пробными периодами, а также для избегания лишних трат со стороны студента.

В результате сравнения аналогов были показаны преимущества использования Github Actions. Помимо вышесказанного, дополнительные удобства предоставляется тем, что репозиторий учебного проекта уже расположен на Github и не требует никаких дополнительных сил для начала работы с данной технологией.

1.5. Описание метода решения

Объем 2 стр

1.5.1. Выбор фреймворка написания серверной части приложения

Объем 2-3 стр

1.5.2. Выбор технологии реализации автоматизации процессов разработки

Объем 2-3 стр

1.5.3. Выбор хостинга для реализации автоматизированного развертывания приложения

Объем 2-3 стр

2. ОПИСАНИЕ ПРОЦЕССА РАЗРАБОТКИ

2.1. Описание модифицируемого приложения

Объем 2-3 стр

В данном разделе будет подробно описана вся функциональность существующего приложения, которое предстоит автоматизировать.

2.2. Технические требования к решению

Объем 2-3 стр

2.3. Выбор инструментов для реализации

Объем 2-3 стр

2.4. Архитектура предлагаемого решения

Объем 2-3 стр

2.4.1. Клиентская часть

Объем 2-3 стр

2.4.2. База данных

Объем 2-3 стр

2.4.3. Серверная часть

Объем 2-3 стр

2.5. Использование приложения

Объем 2-3 стр

2.6. Тестирование приложения

Объем 2-3 стр

3. СПЕЦИАЛЬНЫЕ ВОПРОСЫ ОБЕСПЕЧЕНИЯ БЕЗОПАСНОСТИ (ДОПОЛНИТЕЛЬНАЯ ГЛАВА)

ЗАКЛЮЧЕНИЕ

Объем 2 стр

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ