Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное

учреждение высшего образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра автоматизированных систем управления (АСУ)

РАЗРАБОТКА ВЕБ-ПРИЛОЖЕНИЯ НА PYTHON

Отчет по лабораторной работе № 1 по дисциплине

«Разработка веб-сервисов для научных и прикладных задач»

Вариант 6

|  |  |  |
| --- | --- | --- |
|  |  | Студент гр. з-430П8-1    Н. И. Небиев  (подпись) (И.О. Фамилия)  « 15 » декабря 2024 г.  (дата) |
|  | (оценка) | Руководитель  преподаватель кафедры АСУ    А. Я. Суханов  (подпись)  « » 2024 г.  (дата) |

Томск 2024

# Введение

Целью данной лабораторной работы является изучение возможностей языка Python по обработке данных, создание веб-приложения с использованием фреймворка Flask, системы контроля версий GIT и системы непрерывной интеграции.

# Ход работы

Системы интеграции и доставки кода, находящиеся в основе процессов разработки современных программных решений, представляют собой сложный набор взаимозависимых процедур, нацеленных на обеспечение максимальной эффективности в рамках жизненного цикла разработки программного обеспечения.

Важнейшей составляющей этой системы является непрерывная интеграция — Continuous Integration, предусматривающая автоматизированное слияние исходного кода, внесённого разработчиками, в репозитории и тестирование с целью выявления ошибок на ранних стадиях разработки, что, в свою очередь, сокращает время, затрачиваемое на исправление дефектов и внедрение новых функций.

Второй ключевой компонент — Continuous Delivery, непрерывная доставка, представляющая собой процесс сборки, тестирования и развёртывания программного продукта, что обеспечивает возможность быстрого и надёжного выпуска новых версий программного обеспечения в рамках жизненного цикла разработки. Этот процесс позволяет разработчикам и операционным инженерам совершенствовать рабочие процессы, тем самым удовлетворяя требования заказчиков с минимальными временными затратами и рисками, связанными с внедрением новых функций и устранением ошибок.

Обзор сервисов для организации CI/CD

GitLab CI — сервис от GitLab. Облачный, интегрирован с GitLab, предоставляет лимитированное бесплатное время работы серверов GitLab CI. Не интегрирован с GitHub, нам не подходит.

Bitbucket Pipelines — автоматизация CI/CD от Bitbucket. Облачный, интегрирован в Bitbucket, предоставляет лимитированное бесплатное время работы серверов Bitbucket Pipelines. Не интегрирован с GitHub, нам не подходит.

Azure Pipelines — сервис для CI/CD в различных окружениях. Есть интеграция с GitHub, предоставляет пробный период и лимитированное бесплатное время работы серверов Azure Pipelines.

CircleCI — облачный сервис для создания CI/CD в различных окружениях. Есть интеграция с GitHub, предоставляет лимитированное бесплатное время работы сервера CircleCI.

Travis CI — сервис для создания CI/CD в различных окружениях. Есть интеграция с GitHub, бесплатных тарифов нет.

GitHub Actions — сервис от GitHub. Облачный, интегрирован в GitHub, для приватных репозиториев предоставляется лимитированное бесплатное время работы серверов; бесплатного времени хватит примерно на 200–400 запусков тестирования и сборки.

Остановимся на GitHub Actions: для работы этого сервиса нужна минимальная дополнительная настройка репозитория, а сам сервис предлагает множество готовых решений для CI/CD.

Workflow (воркфлоу, «рабочий процесс») — на GitHub Actions так называют пайплайн, цепочку шагов CI и CD, которую должен выполнить сервер GitHub Actions, когда происходит событие, запускающее процесс.

Событие, которое запускает выполнение воркфлоу, называют «триггер» — «спусковой крючок».

Jobs («задачи», если по-русски) — это шаги, отдельные этапы, которые должны быть выполнены в пределах одного воркфлоу.

Раннер (англ. runner) — это сервер, который будет выполнять все описанные задачи.

Получается, что jobs для проекта — это:

* запустить тесты и убедиться, что код работает;
* собрать обновлённые образы всех контейнеров;
* обновить образы на продакшен-сервере;
* перезапустить Docker Compose.

И весь этот процесс должен стартовать, когда произойдёт событие-триггер. При возникновении события-триггера GitHub Actions читает файл с описанием workflow и для каждой отдельной задачи-job этого workflow выделяет отдельный раннер, который будет выполнять эту задачу.

Мною выбрана следующая структура веб-приложения:

.

├── Dockerfile

├── app.py

├── constants.py

├── docker-compose.yml

├── docker-compose.production.yml

├── gateway

│ ├── Dockerfile

│ └── nginx.conf

├── requirements.txt

├── service.py

├── setup.cfg

└── templates

├── index.html

└── result.html

Запуск приложений реализовано в Docker-контейнерах. Docker image (докер-образ) — это шаблон, из которого создаются Docker-контейнеры. Образ хранит в себе всё необходимое для запуска приложения: код приложения, среду выполнения, нужные библиотеки, переменные окружения, конфигурационные файлы.

Для передачи медиа файлов и использовал Nginx, так как gunicorn не предназначен для передачи статики и медиа файлов. Его конфиг и образ находится в директории gateway. Так же nginx будет отвечать за проксирование запросов.

Сборка проекта происходит в файле docker-compose.yml в нём задаются описания нужных контейнеров и томов (volume). По умолчанию файл конфигурации должен называться docker-compose.yml. Сеть контейнеров запускается командой docker compose up;

Что внутри docker-compose.yml

version — версия спецификации файла docker-compose.yml. Обязательный параметр. От версии к версии набор доступных команд меняется, и какие-то команды из новых версий могут не поддерживаться старыми версиями Docker Compose. Узнать, какая версия Docker Compose установлена на компьютере, можно с помощью команды docker compose version.

В документации описано соответствие версий Docker Compose версиям файла docker-compose.yml. Версии Docker Compose 1.10 и выше вполне подойдут для дальнейшей работы.

volumes — перечень volumes для докера, необязательный параметр. Для каждого имени volume через двоеточие можно указать его подробные настройки. Их можно и не указывать — докер применит настройки по умолчанию.

services — названия и описания контейнеров, которые должны быть запущены. В листинге описаны два контейнера: app и gateway.

Ключи в конфигурации можно указывать в любом порядке. В примере сначала указаны volumes, а потом services: при описании контейнеров удобнее видеть, какие volumes уже созданы.

Описание каждого контейнера — это YAML-словарь, значения в этом словаре похожи на параметры запуска, которые вы применяли при ручном старте контейнеров.

Для простой проверки достаточно ввести команду docker compose up (Рис. 1).



Рисунок 1 – Результат выполнения команды.

В результате перейдя по адресу <http://localhost:9000/> пользователь увидит веб-интерфейс приложения (Рис. 2)

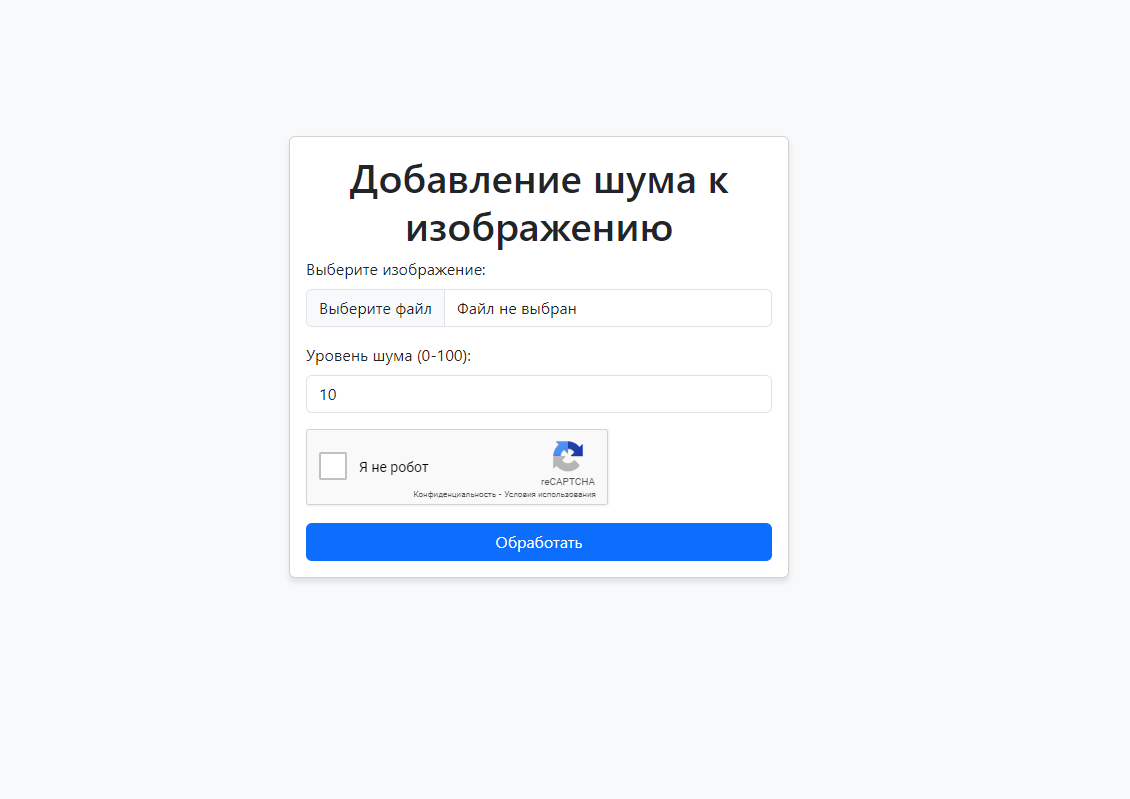


Рисунок 2 – Веб интерфейс приложения

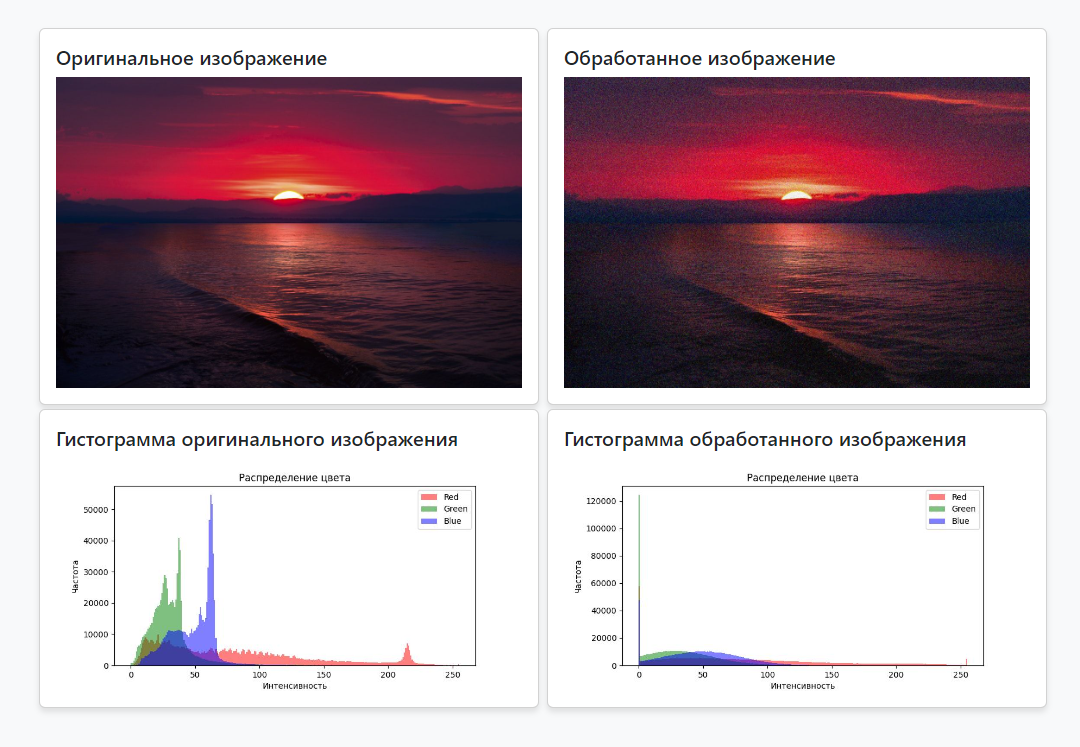
Пользователь может выбрать файл и указать уровень шума. Так же необходимо подтвердить, что вы не робот и обработать изображение. В результате пользователь увидит 4 картинки. 2 фото и 2 гистограммы (Рис.3).  


Рисунок 3 – Выполнение обработки изображения

В проекте было использован tensorflow, его метод random.normal позволял выводить случайные значения из нормального распределения.

GitHub проекта <https://github.com/zaphire12/flaskProject>

Деплой происходило на виртуальную машину. Приложение доступно по адресу <http://noise-app.ddns.net/>. Для получения hostname использовал сервис noip.

# Вывод

В ходе выполнения лабораторной работы были изучены основные возможности языка программирования Python, включая обработку данных. Также были освоены основы разработки веб-приложений с использованием фреймворка Flask, что позволило получить практический опыт создания серверных приложений. Использование системы контроля версий GIT обеспечило эффективное управление проектом и отслеживание изменений в коде. Реализация элементов непрерывной интеграции позволила автоматизировать процесс тестирования и развёртывания приложения, что является важным навыком в современных разработческих практиках.