# МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования «Уральский федеральный университет имени первого Президента России Б.Н. Ельцина»

# ИТОГОВАЯ АТТЕСТАЦИОННАЯ РАБОТА

# Разработка сервиса учёта используемых сотрудниками расходников на объекте

по дополнительной профессиональной программе —

программы профессиональной переподготовки

«Прикладное программирование на языке Python»

Руководитель: Бутаков Руслан Александрович

подпись

Студент: Третьяков А. В.

подпись

Екатеринбург

2024

Оглавление

[1. Паспорт проекта 3](#_Toc167015595)

[1.1 Описание проекта 3](#_Toc167015596)

[1.2 Описание функциональности 3](#_Toc167015597)

[1.2.1 Создание таблицы 3](#_Toc167015598)

[1.2.2 Внесение данных о пользователях 4](#_Toc167015599)

[1.2.3 Вывод данных всех пользователей 5](#_Toc167015600)

1.2.4 Вывод данных о конкретном пользователе………..….………6  
 1.2.5 Удаление данных……………..……….……………..…………7  
 1.2.6 Обновление данных………………………………………..8

1.2.7 Использование Docker………………………………………..8

2. Описание продукта…………………………………………………..10

3. Основные этапы Реализации проекта……………………………….11

4. Раздел по специфике программы……………………………………12

Заключение……………………………………………………………..14

Библиографический список……………………………………………15

# 1. паспорт проекта

## 1.1 Описание проекта

Проект представляет собой сервис по информированию об остатках расходников и иных материалов для работников типографии. Он позволяет вносить в базу данные о том, кто из сотрудников и что из материалов взял со склада, виде расходника и о его количестве, просматривать внесённые данные, а также удалять их.

## 1.2 Описание функциональности

### 1.2.1 Создание таблицы

Создание таблицы происходит при запуске следующего блока кода (указан на рисунке 1). Этот блок кода настроен для использования с фреймворком Flask в Python и предназначен для создания базы данных с использованием расширения Flask-SQLAlchemy



1. Команда создания таблицы в базе данных

### 1.2.2 Внесение данных о пользователях

Внесение данных таблицы происходит при помощи блока кода, указанного на рисунке 2. Данный блок кода представляет собой функцию представления в веб-приложении Flask, которая обрабатывает создание нового пользователя.   
Подробная работа данного блока кода указана ниже:

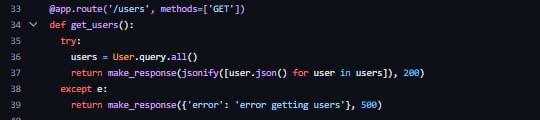
* new\_user = User(username=data['username'], email=data['email']) - создание нового объекта пользователя с именем и электронной почтой (При необходимости можно добавить различные ключи, например, название расходника и т.д)
* db.session.add(new\_user) - добавление нового пользователя в сессию базы данных
* db.session.commit() - фиксация изменений в базе данных
* return make\_response({'message': 'user created'}, 201) - возврат ответа с сообщением об успешном создании
* except e:
* return make\_response({'error': 'error creating user'}, 500) - Возврат ошибки, если что-то пошло не так
* @app.route('/users', methods=['POST']) - Декоратор, который указывает, что функция create\_user будет обрабатывать POST-запросы по URL /users.
* try - Блок try используется для обработки исключений, которые могут возникнуть во время выполнения кода внутри блока.
* data = request.get\_json() - Извлекает данные из POST-запроса, которые отправляются в формате JSON.
* new\_user = User(username=data['username'], email=data['email']) - Создает новый экземпляр модели User, используя данные из запроса.
* db.session.add(new\_user) и db.session.commit() - Добавляет нового пользователя в базу данных и сохраняет изменения.
* return make\_response({'message': 'user created'}, 201) - Возвращает ответ со статусом 201, указывающим на успешное создание ресурса.
* except e - Если во время выполнения кода возникает исключение, блок except перехватывает его и возвращает ответ с ошибкой 500.



1. Команда создания нового пользователя в базе данных

### 1.2.3 Вывод данных о всех пользователях

Вывод данных обо всех пользователях, представленных в базе данных осуществляется посредством GET-запроса – «/users». Блок кода, отвечающий за вывод данных о пользователях, представленных в базе данных указан на рисунке 3.

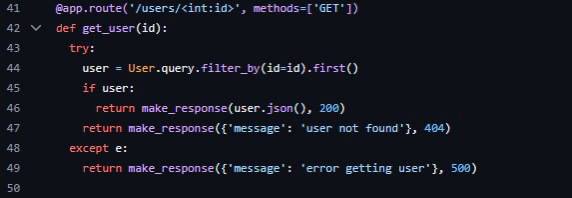


1. Команда вывода данных обо всех пользователях.

Если существует хотя бы один пользователь, программа выводит на экран соответствующую строку данных в формате json. Если же пользователи отсутствуют, программа выводит сообщение: «error getting users» с присвоением соответствующего номера 500 об ошибке.

### 1.2.4 Вывод данных о конкретном пользователе

Иногда требуется вывести информацию об одном конкретном пользователе, для этого используется метод GET, для поиска конкретного пользователя будет достаточно указать id пользователя. Блок кода, отвечающий за данную функциональность сервиса представлен на рисунке 4.



1. Вывод данных о конкретном пользователе

### 1.2.5 Удаление данных

Удаление данных о пользователе представлено в блоке кода ниже (см. рисунок 5). Этот блок кода определяет функцию delete\_user, которая используется для удаления пользователя из базы данных в веб-приложении Flask. Функция принимает id пользователя как параметр в URL и пытается найти соответствующего пользователя в базе данных. Если пользователь найден, он удаляется, и операция подтверждается. В ответ клиенту отправляется JSON с данными удалённого пользователя и статусом 200. Если пользователь не найден, возвращается сообщение об ошибке с кодом 404. В случае возникновения исключения при удалении, клиенту отправляется сообщение об ошибке с кодом 500. Эта функция важна для управления учетными записями пользователей, позволяя администраторам удалять записи о пользователях.



1. Команда удаления данных

### 1.2.6 Обновление данных

Обновление данных осуществляется посредством использования запроса PATCH (см. рисунок 6).



1. Команда обновление данных о пользователе

Этот блок кода относится к функции update\_user, которая обновляет информацию о пользователе в базе данных веб-приложения Flask. Она принимает id пользователя в качестве переменной в URL и позволяет обновить такие данные, как имя пользователя и электронная почта. Если пользователь с указанным id существует, его данные обновляются согласно полученному JSON-запросу. После успешного обновления возвращается сообщение с кодом 200. Если пользователь не найден, возвращается ошибка 404. В случае возникновения ошибки при обновлении данных пользователя, возвращается ошибка 500. Эта функция важна для поддержания актуальности данных пользователей и обеспечения возможности их изменения администраторами или самими пользователями.

### 1.2.7 Использование Docker

Docker — это платформа для разработки, доставки и запуска приложений в контейнерах. Контейнеры позволяют упаковать приложение со всеми его зависимостями в стандартизированную единицу для программного обеспечения. Это обеспечивает несколько преимуществ:

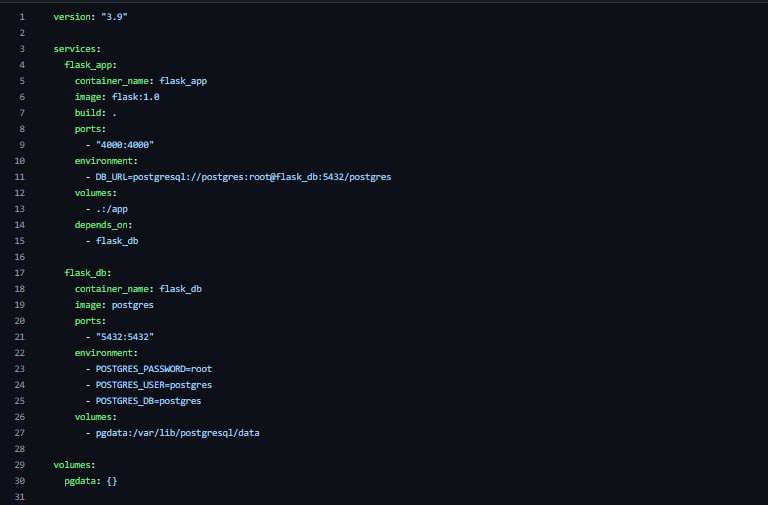
* **Портативность**: Поскольку контейнеры включают все необходимое для работы приложения, они могут запускаться на любой системе, поддерживающей Docker, независимо от настроек и установленных программ.
* **Быстрота развертывания**: Контейнеры могут быть созданы, запущены и остановлены за секунды, в отличие от традиционных виртуальных машин.
* **Эффективность**: Docker использует ресурсы системы более эффективно, что позволяет запускать больше приложений на тех же физических машинах.
* **Изоляция**: Приложения работают в изолированной среде, что повышает безопасность и упрощает управление версиями и зависимостями.
* **Масштабируемость и управление**: Docker облегчает масштабирование приложений и их управление, особенно при использовании оркестраторов контейнеров, таких как Kubernetes.

Использование Docker представлено на рисунке 7.



1. Настройка Docker для приложения.

Для настройки и «поддержания» приложения в связке с базой данных внутри контейнера Docker используется docker-compose (см. рисунок 8)



1. Docker-compose для правильной работы приложения и базы данных.

# 2. Описание продукта

Проект представляет собой сервис по информированию об остатках расходников и иных материалов для работников типографии. Он позволяет вносить в базу данные о том, кто из сотрудников и что из материалов взял со склада, виде расходника и о его количестве, просматривать внесённые данные, а также удалять их.

При работе с сервисом доступны следующие функции:

1. Внесение id работника, расходника и его количество;
2. Просмотр внесённых ранее данных;
3. Удаление данных.

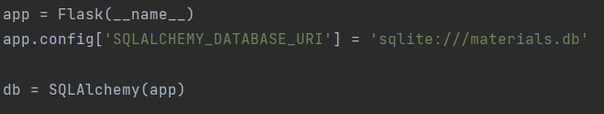
Код проекта написан на языке Python с помощью IDE Microsoft Visual Studio Code. В программе применяется архитектурный стиль для разработки веб-сервисов Rest API. Она определяет запросы для работы с данными в таблице. Использована система управления базами данных mySQL, для поднятия приложения в контейнере используется Docker и его настройка Docker-compose.

# 3. Основные этапы Реализации проекта

1. Настройка рабочего окружения, знакомство с инструментами разработчика.
2. Знакомство с основными возможностями Python. Написание основных функций для проекта.
3. Запуск кода проекта и его отладка в консоли инструментов разработчика.
4. Организация кода — подготовка нужных модулей для проекта, создание точки входа.
5. Протокол HTTP, формат данных json. Написание запросов к серверу из браузера.
6. Rest API. Различные методы работы с данными.
7. Создание и настройка Docker.

# 4. Раздел по специфике программы

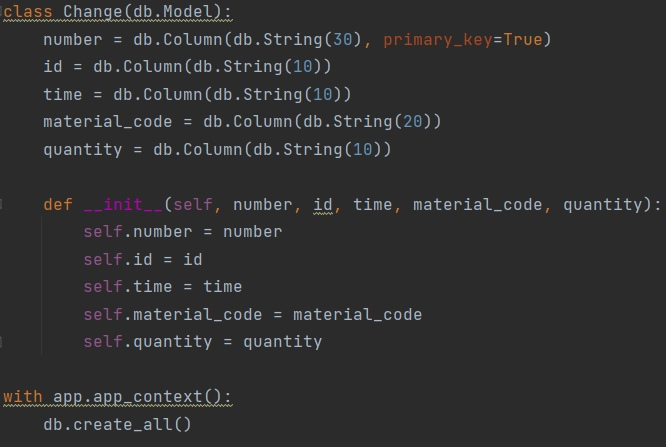
Проект состоит из четырёх модулей: одного модуля создания базы данных и трёх для работы с данными. Перед первым модулем указывается ссылка на базу данных (Рисунок 4).



1. Ссылка на базу данных

При запуске программы создаётся файл «materials.db», в котором будет находиться база данных. Последняя строчка связывает приложение с SQLAlchemy.

Модуль, создающий базу данных, начинается с модели базы данных, которая задаётся как *class* (см. Рисунок 5). В классе указываются все необходимые колонки таблицы данных, указывается тип данных в них, а также primary\_key, по которому можно будет делать поиск.

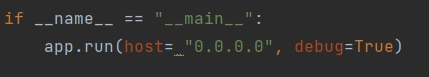


1. Модуль, создающий таблицу данных

Далее задаётся функция *init*, которая вызывается при создании нового объекта класса. Она необходима для установки изначальных значений. Последняя часть модуля непосредственно создаёт базу данных.

Модули для работы с данными задаются командами *@app.route*. В них указываются функции — их название и выполняемая работа. Подробное описание модулей приведено в пункте 1.2.

В конце программы находится функция, с помощью которой происходит запуск (см. Рисунок 6).



1. Функция запуска программы

# Заключение

Изучена работа с базами данных с помощью Python, в частности с базой mySQL. Получены знания о создании баз и работе с данными в них. Приобретены навыки работы с API и форматом данных json, Docker. Изучены различные методы ввода и вывода данных.

В ходе работы над проектом был разработан сервис по информированию об остатках расходников и иных материалов для работников типографии. В дальнейшем возможно изучение других баз данных, способов работы с ними, форматов данных и работы с удалёнными серверами.

# библиографический список

1. Курс: Прикладное программирование на языке Python / [Электронный ресурс] // Moodle: [сайт]. — URL: https://elearn.urfu.ru/course/view.php?id=6775 (дата обращения: 01.05.2024)
2. Flask / [Электронный ресурс] // flask.palletsprojects: [сайт]. — URL: https://flask-restful.readthedocs.io/en/latest/quickstart.html (дата обращения: 08.05.2024).