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

на тему

«*Сервис* *сбора отчетов сотрудников компании о проделанных работах на объекте заказчика*»

по дополнительной профессиональной программе –

программы профессиональной переподготовки

«Прикладное программирование на языке Python»

|  |  |
| --- | --- |
| Руководитель: Бутаков Руслан Александрович | подпись |
| Слушатель Шаранов А.А. | подпись |

Екатеринбург

2024

**СОДЕРЖАНИЕ**

[1. ПАСПОРТ ПРОЕКТА 3](#_Toc168263038)

[2. ОПИСАНИЕ ПРОДУКТА 7](#_Toc168263039)

[3. ОСНОВНЫЕ ЭТАПЫ/ИТЕРАЦИИ РЕАЛИЗАЦИИ ПРОЕКТА 9](#_Toc168263040)

[4. ОРГАНИЗАЦИЯ РАБОТ В ПРОЕКТЕ 10](#_Toc168263041)

[5. ЗАКЛЮЧЕНИЕ 11](#_Toc168263042)

[Библиографический список 13](#_Toc168263043)

# 1. ПАСПОРТ ПРОЕКТА

**Описание проекта**

Сервис сбора отчетов сотрудников компании о проделанных работах на объекте заказчика. Пользователям предоставлена возможность загружать свои отчёты или просматривать их.

**Описание функциональности**

**1. Загрузка нового отчёта на сервис.**

1.1. Загрузка нового отчёта:

* заполнение поля с ФИО;
* заполнения поля с текстом отчёта;
* применение одного из заранее заготовленных эффектов;

1.2. Отображение формы для ввода данных:

Пользователь заходит на главную страницу (URL: /) и видит форму для ввода имени (ФИО) и текста отчёта. Эта форма создана с использованием Flask-WTF и отображается через шаблон index.html.

1.3 Ввод и отправка данных формы:

Пользователь заполняет форму и нажимает кнопку "Отправить". Данные формы отправляются методом POST на тот же URL (/).

1.4 Валидация данных и создание нового отчёта:

Сервер получает данные формы, проверяет их валидность (валидируется наличие данных в полях full\_name и report\_text). Если данные валидны, создаётся новый объект отчёта (Report), который затем сохраняется в базу данных.

1.5 Подтверждение успешной отправки:

После успешного сохранения данных в базу данных, пользователю отображается сообщение об успешной отправке отчёта через шаблон success.html.

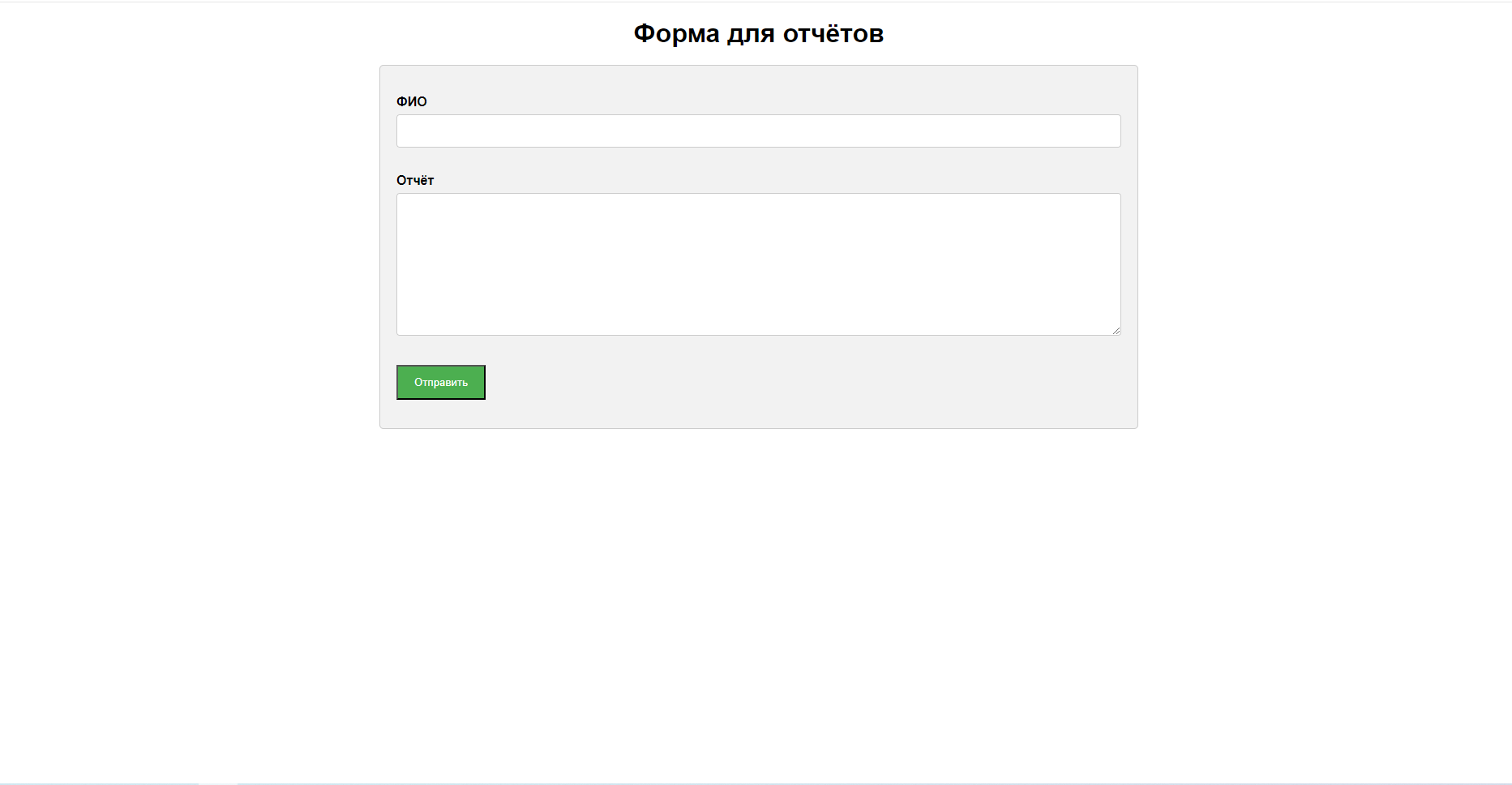


Рисунок 1 – Главная страница отправки отчёта

**2. Отображение всех отчётов в базе**

2.1. Запрос на получение всех отчётов:

Пользователь или клиентское приложение делает GET-запрос к маршруту /reports/all.

2.2. Обработка запроса на сервере:

* Сервер, получив запрос на этот маршрут, обращается к базе данных для извлечения всех записей из таблицы Report.
* Эти записи передаются в шаблон all\_reports.html для рендеринга страницы с таблицей всех отчётов.

2.3. Отображение всех отчётов:

* Шаблон all\_reports.html генерирует HTML-код для отображения всех отчётов в виде таблицы.
* В таблице выводятся все записи с полями "ФИО" и "Отчёт".

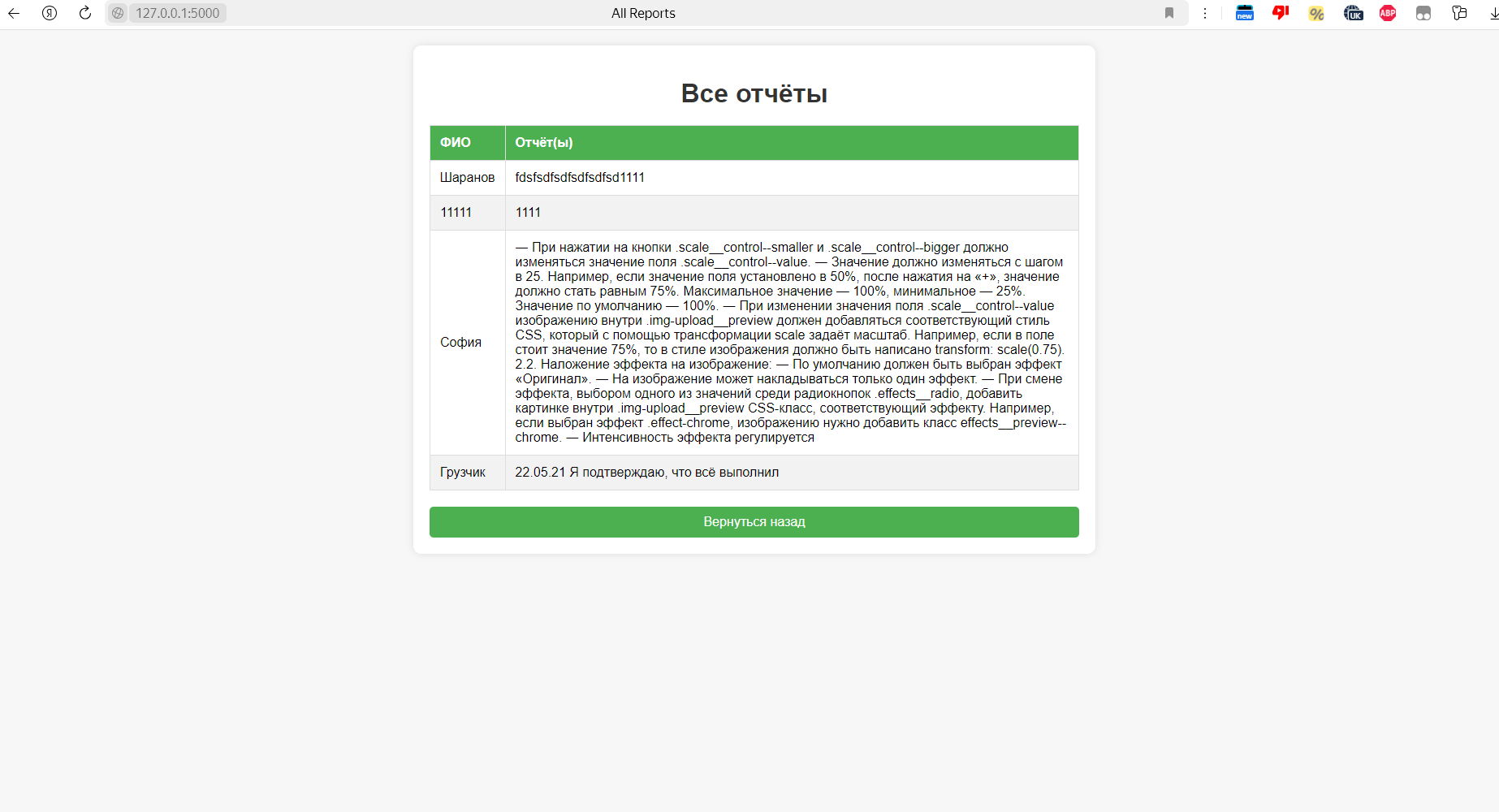


Рисунок 2 – Отображение всех отчётов в базе

**3. Отображение отчётов по ключу «name»**

3.1. Запрос на получение отчётов по имени:

Пользователь или клиентское приложение делает GET-запрос к маршруту /reports/by\_name с параметром name, например: `/reports/by\_name?name=Petrov`.

3.2. Обработка запроса на сервере:

* Сервер получает запрос и извлекает значение параметра name.
* Далее он обращается к базе данных для поиска всех записей в таблице Report, где поле full\_name совпадает с переданным значением.
* Найденные записи передаются в шаблон reports\_by\_name.html для рендеринга страницы с таблицей отчётов по заданному имени.

3.3 Отображение отчётов по имени:

* Шаблон reports\_by\_name.html генерирует HTML-код для отображения отчётов, соответствующих заданному имени, в виде таблицы.
* В таблице выводятся все записи с полями "ФИО" и "Отчёт".

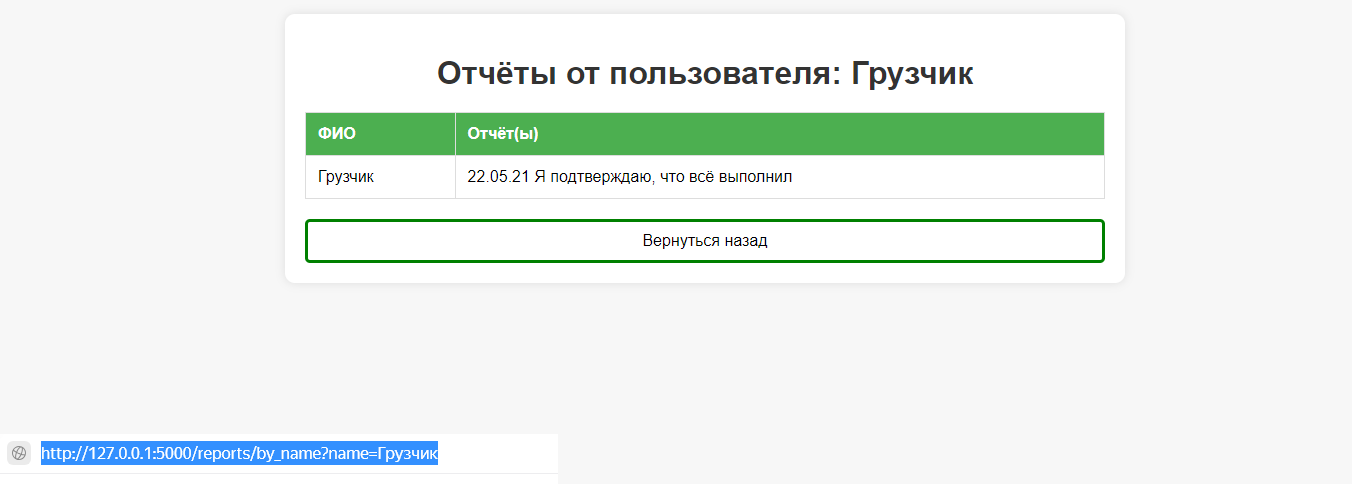


Рисунок 3 - Отображение отчётов по ключу name

**4. Доступ через api**

Можем протестировать через командную строку с помощью curl.

4.1. Получить все отчеты:

curl -X GET <http://127.0.0.1:5000/api/reports>

4.2. Получить отчет по ID:

curl -X GET <http://127.0.0.1:5000/api/reports/1>

4.3. Создать новый отчет:

curl -X POST http://127.0.0.1:5000/api/reports -H "Content-Type: application/json" -d '{"full\_name": "Артём Шаранов", "report\_text": "Это новый отчёт"}'

4.4. Создать новый отчет:

curl -X DELETE http://127.0.0.1:5000/api/reports/1

# 2. ОПИСАНИЕ ПРОДУКТА

Проект представляет собой веб-приложение, предназначенное для сбора, хранения и управления текстовыми отчётами пользователей. Оно предоставляет удобный интерфейс для отправки и просмотра отчётов, а также REST API для интеграции с другими сервисами.

**Основные функции:**

1. **Отправка отчётов:**
   * Пользователи могут отправлять отчёты через веб-форму, указывая своё имя и текст отчёта.
   * После успешной отправки пользователи видят сообщение об успешной отправке и могут вернуться назад.
2. **Просмотр всех отчётов:**
   * Веб-интерфейс позволяет просматривать все отправленные отчёты в виде таблицы.
   * Отчёты содержат имя пользователя и текст отчёта.
3. **Просмотр отчётов по имени:**
   * Пользователи могут искать отчёты по имени через веб-интерфейс.
   * Результаты поиска отображаются в виде таблицы.
4. **RESTful API:**
   * API предоставляет доступ к отчётам в формате JSON.
   * Поддерживаются следующие операции:
     + Получение всех отчётов.
     + Получение отчёта по идентификатору.
     + Создание нового отчёта.
     + Удаление отчёта по идентификатору.

**Backend:**

* Python
* Flask - веб-фреймворк для Python, обеспечивающий маршрутизацию и обработку запросов.
* Flask-WTF - расширение для работы с веб-формами;
* SQLAlchemy - ORM для работы с базой данных SQLite.
* Код проекта написан на языке JavaScript.

**Frontend:**

* HTML и CSS - для создания веб-страниц и стилей.

**База данных:**

* SQLite - база данных для хранения отчётов.

# 3. ОСНОВНЫЕ ЭТАПЫ/ИТЕРАЦИИ РЕАЛИЗАЦИИ ПРОЕКТА

1. Определение требований: собрать и определить все функциональные и нефункциональные требования к системе.
2. Проектирование базы данных: спроектировать структуру базы данных (таблицы, связи и т.д.).
3. Разработка макетов пользовательского интерфейса: создать макеты и прототипы основных страниц и интерфейсов.
4. Выбор и настройка среды разработки: установить необходимые инструменты и библиотеки для разработки (Python, Flask, SQLAlchemy и т.д.).
5. Создание и настройка базы данных: инициализировать базу данных и создать необходимые таблицы.
6. Создание основных HTML-страниц: создать шаблоны для основных страниц приложения (главная страница, страница отправки отчёта, страница просмотра отчётов).
7. Создание моделей данных: определить модели данных для работы с базой данных (например, модель Report).
8. Настройка маршрутов: создать маршруты для обработки запросов (отправка отчёта, просмотр всех отчётов, просмотр отчётов по имени).
9. Реализация логики: написать логику для обработки форм, сохранения данных в базу и извлечения данных из базы.
10. Создание RESTful API: определить и реализовать маршруты для API (получение всех отчётов, получение отчёта по ID, создание нового отчёта, удаление отчёта).
11. Тестирование

Ссылка на репозиторий GitHub: <https://github.com/tereradearde/report>

# 4. ОРГАНИЗАЦИЯ РАБОТ В ПРОЕКТЕ

*Работал один Шаранов Артём. На работу ушло порядка часов 10-12.*

# 5. ЗАКЛЮЧЕНИЕ

**За время работы над проектом мы научились:**

1. **Проектирование и реализация веб-приложений**:
   * Научились проектировать и разрабатывать веб-приложения с использованием Flask.
2. **Работа с базами данных**:
   * Получили опыт в проектировании и работе с базами данных с использованием SQLAlchemy.
   * Научились выполнять основные операции с базой данных (создание, чтение, обновление, удаление данных).
3. **Создание и использование форм**:
   * Получили навыки для создания и валидации веб-форм.
   * Научились обеспечивать корректный ввод данных пользователями.
4. **Разработка RESTful API**:
   * Получили опыт в создании REST API с использованием Flask-RESTful.

**К каким выводам пришли:**

1. **Важность планирования и проектирования**:
   * Грамотное планирование и проектирование являются ключевыми для успешной реализации проекта.
2. **Гибкость и модульность**:
   * Модульный подход к разработке позволяет создавать гибкие и легко расширяемые приложения. Разделение на модули упрощает поддержку и развитие проекта.
3. **Значимость пользовательского опыта (UX/UI)**:
   * Удобный и привлекательный интерфейс существенно улучшает взаимодействие пользователей с приложением. Стилизация и внимание к деталям играют важную роль в восприятии продукта.
4. **Необходимость тестирования**:
   * Тестирование на всех этапах разработки помогает выявить и исправить. Это повышает надёжность и стабильность приложения.

**Дальнейшее развитие:**

1. **Расширение функционала**:
   * Добавление новых функций, таких как поддержка нескольких языков, возможность прикрепления файлов к отчётам, уведомления и интеграция с внешними сервисами.
2. **Улучшение производительности и масштабируемости**:
   * Оптимизация запросов к базе данных, использование кэширования и распределённых систем для обеспечения высокой производительности при увеличении нагрузки.
3. **Мобильные платформы**:
   * Разработка мобильной версии приложения или создание нативных мобильных приложений для iOS и Android.
4. **Повышение безопасности**:
   * Внедрение дополнительных мер безопасности, таких как аутентификация и авторизация пользователей, защита от SQL-инъекций и XSS-атак.
5. **Интеграция с аналитическими инструментами**:
   * Внедрение инструментов для аналитики и отчётности, позволяющих пользователям получать статистику и аналитику по отправленным отчётам.

Работа над проектом дала нам ценный опыт и знания в области веб-разработки, баз данных и API. Мы уверены, что полученные навыки помогут нам в будущих проектах, а наш продукт будет полезен многим пользователям.

# Библиографический список

1. Курс «Прикладное программирование на языке Python», URL: https://elearn.urfu.ru/course/view.php?id=6775