Министерство цифрового развития, связи и массовых коммуникаций Российской Федерации

Ордена Трудового Красного Знамени

федеральное государственное бюджетное образовательное учреждение высшего образования

«Московский технический университет связи и информатики»

(МТУСИ)

Лабораторная работа № 4

по дисциплине: «Рефакторинг баз данных и приложений»

кафедра: «Программная инженерия»

по теме:

«Рефакторинг приложений с изучением перекрестного код-

ревью»

Выполнил: студент группы БПИ2303

Любченко Никита Тариэлович

Москва, 2025

В рамках перекрестного код-ревью, я взял код Дмитрия Гусака. В качестве кода, который он предоставил мне, был проект по управлению пользователями.

Код требовал рефакторинга.

**Проблемы в коде Дмитрия:**

1. **Нет валидации входных данных**

Любой словарь без гарантии структуры.

1. **Нет типизации**

Отсутствует типизация в коде.

1. **Использование словарей**

Словарь лучше заменить на список.

1. **Отсутствие респонс моделей у эндпоинтов**

Стоит добавить респонс модели для эндпоинтов.

1. **Отсутствие указания ошибок**

Выводится юзер, даже если он не найден, будет выведен пустое значение.

Также я предоставил код Дмитрию, для выявления проблем в моем коде.

В качестве кода был проект Todo List.

**Исходные проблемы в моем коде:**

1. **Дублирование кода работы с БД**
2. **Нет валидации ввода**
3. **Жёсткие URL**
4. **Уязвимость к SQL-инъекциям**
5. **Нет обработки ошибок БД**
6. **Сложность тестирования**

**Рефакторинг:**

1. **Дублирование кода работы с БД**

В каждой функции (index, add, delete) повторяется один и тот же код для:

* Открытия соединения с БД
* Создания курсора
* Закрытия соединения

Исходный код:

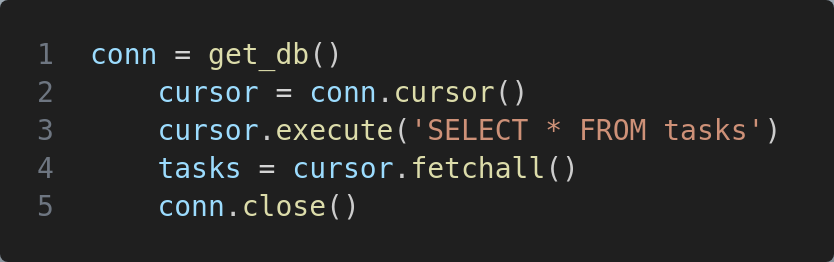


Рисунок 1 - Исходный принцип работы с БД

Изменённый код:

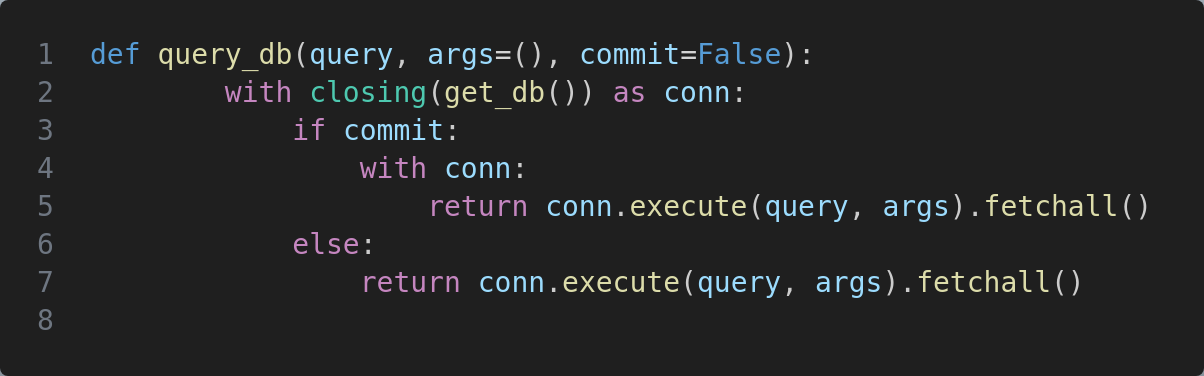


Рисунок 2 - Функция для работы с БД

* Используем контекстные менеджеры (with) для автоматического управления соединениями.
* Выносим работу с БД в отдельную функцию query\_db.
* Добавляем параметр commit для автоматического подтверждения изменений (для INSERT/DELETE).
* Используем with closing(...) для гарантированного закрытия соединения.

1. **Нет валидации ввода**

При добавлении задачи нет проверки на пустую строку.

Исходный код:

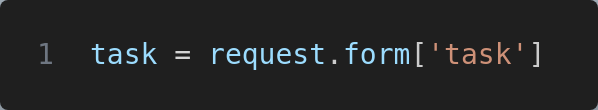


Рисунок 3 - Отсутствие проверки на пустую строку

Изменённый код:

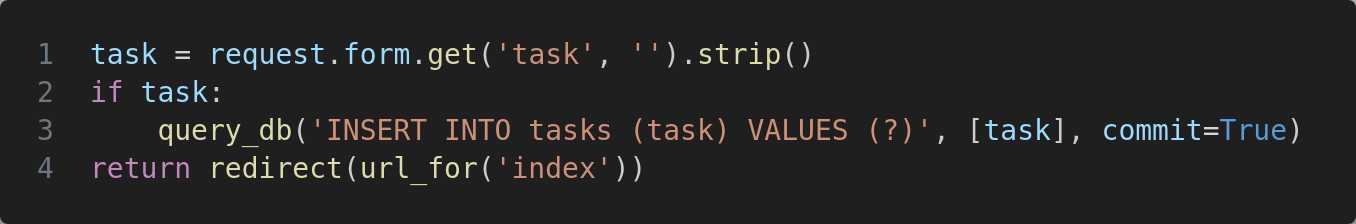


Рисунок 4 - Добавленная проверка ввода

* Добавляем проверку и очистку ввода.
* Заменем request.form['task'] на request.form.get('task', '') для безопасного получения данных.
* Добавляем strip() для удаления лишних пробелов.
* Проверяем, что задача не пустая.

1. **Жёсткие URL**

Редиректы используют жёстко закодированные URL.

Исходный код:

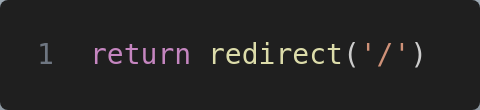


Рисунок 5 - Жёсткий редирект

Изменённый код:

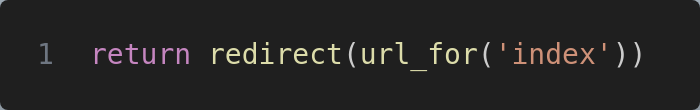


Рисунок 6 - Динамическое формирование URL

* Используем url\_for для динамического формирования URL.
* Заменяем '/' на url\_for('index').
* Теперь при изменении маршрута @app.route('/') не нужно обновлять все редиректы.

1. **Уязвимость к SQL-инъекциям**

Хотя код использует параметризованные запросы (?), сам подход с сырыми SQL-запросами рискован.

Исходный код:

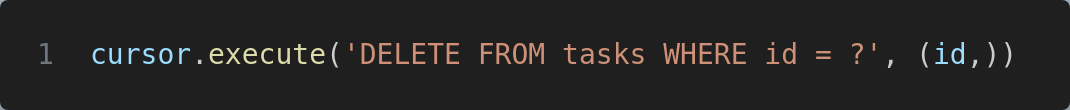


Рисунок 7 - Сырой запрос

Изменённый код:

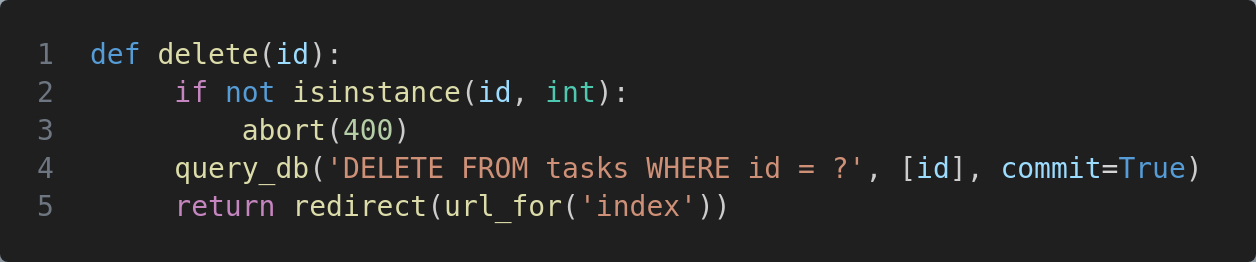


Рисунок 8 - Изменённый вариант запроса

* Добавляем проверку типов для id.
* Добавляем проверку isinstance(id, int).
* Используем abort(400) для некорректных запросов.

1. **Нет обработки ошибок БД**

Нет обработки ошибок при работе с SQLite (например, если БД недоступна).

Исходный код:



Рисунок 9 - Простое подключение БД

Изменённый код:

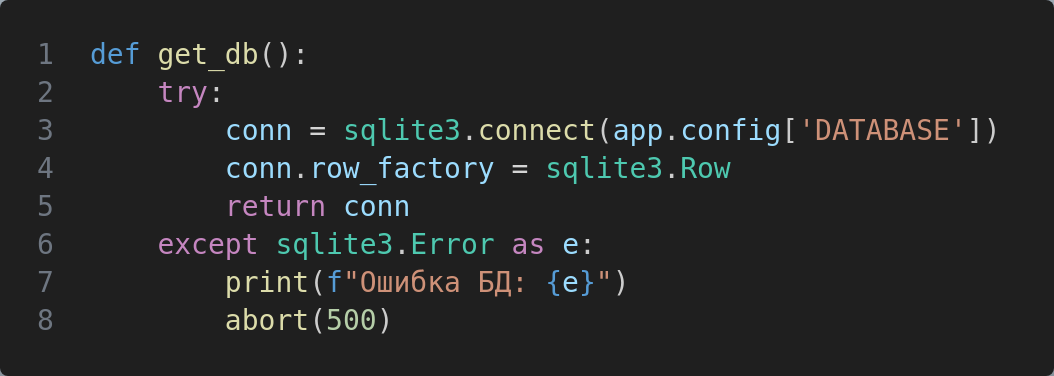


Рисунок 10 - Подключение с возможностью вывода ошибки

* Добавляем try-except:
* Обернули соединение с БД в try-except.
* Добавили row\_factory для удобства (можно обращаться к полям по имени: task['id'] вместо task[0]).

1. **Сложность тестирования**

Глобальная переменная app и жёсткая привязка к SQLite усложняют тесты.

Исходный код:

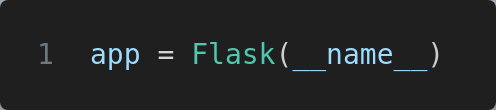


Рисунок 11 - Глобальная переменная app

Изменённый код:

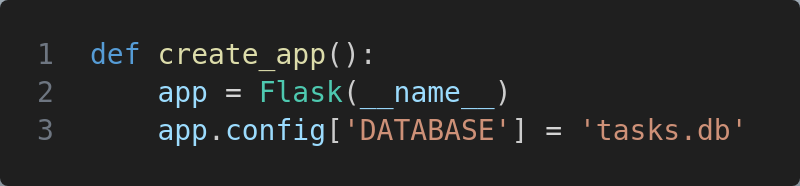


Рисунок 12 - Функция create\_app

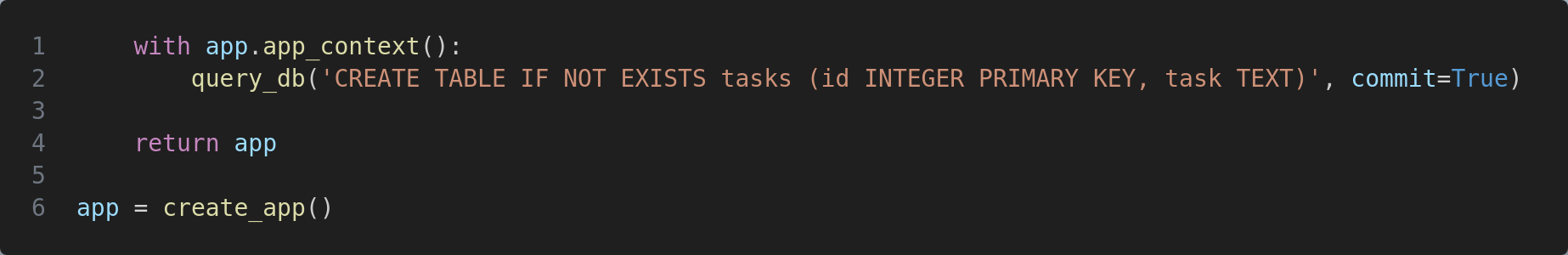


Рисунок 13 - Добавленный app\_context

* Выносим создание приложения в функцию (паттерн Application Factory):
* Перенесли инициализацию в create\_app().
* Добавили app\_context() для работы с БД в контексте приложения.
* Упростили возможность подмены БД в тестах (например, на SQLite в памяти).

Ссылка на Github: https://github.com/dxxdracer/rbdiplab4