**Краткий отчет о проделанной работе и результатах тестирования**

**Цель работы:** разработка и тестирование приложения для управления задачами с использованием Docker, PostgreSQL и Go. Условия проекта включали добавление задач, обновления их статуса и получения информации о задачах.

Для реализации использовались следующие технологии и инструменты:

* Язык программирования Go.
* База данных PostgreSQL:

Была создана база данных task\_queue с таблицей jobs:

|  |
| --- |
| CREATE TABLE jobs (  id SERIAL PRIMARY KEY,  name VARCHAR(255) NOT NULL,  data JSONB NOT NULL,  status VARCHAR(50) DEFAULT 'pending',  priority INTEGER DEFAULT 10,  created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,  updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP  ); |

* Docker и Docker Compose для контейнеризации и управления окружением.

**Структура проекта**

Проект состоит из следующих компонентов:

* main.go — основной файл приложения.
* Dockerfile — файл для сборки Docker-образа.
* docker-compose.yml — файл для запуска контейнеров.

**Реализация функционала**

Были реализованы следующие эндпоинты:

* POST /jobs — для создания новой задачи.
* GET /jobs/{id} — для получения информации о задаче.

**Тестирование**

Для тестирования использовался Postman. Была создана коллекция запросов на добавление задач в базу данных и очередь для проверки метода POST (рисунок 1).

Проверим как выглядит база данных до внесения изменений (рисунок 2).

В терминале проконтролируем выполнение задач (рисунок 3). Важно отметить, что в ходе работы был реализован перезапуск задач при неудачном выполнении. Чтобы сымитировать провал выполнения, была изменена функция PerformJob.

Согласно требованиям, запуск системы происходит в Docker (рисунок 4).

Тестирования метода GET представлены на рисунках 5 – 6. Полученный статус для каждого задания совпадает с соответствующим значением из базы данных (рисунок 7).

Для упрощения отладки и улучшения читаемости кода были добавлены различные сообщения об ошибках. Примеры работы двух из них представлены на рисунках 8 – 9.

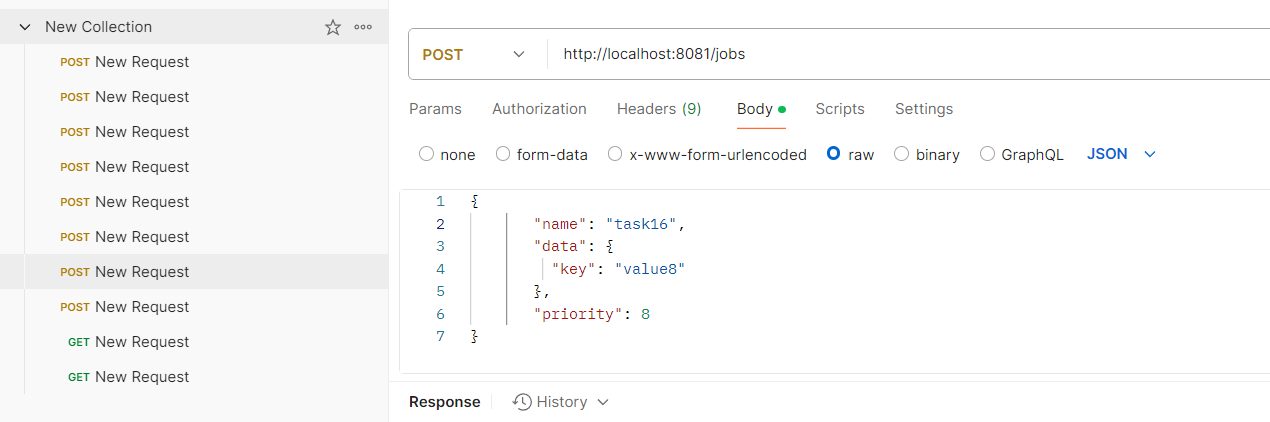


Рисунок 1 – Создание коллекции из запросов для тестирования метода POST в Postman

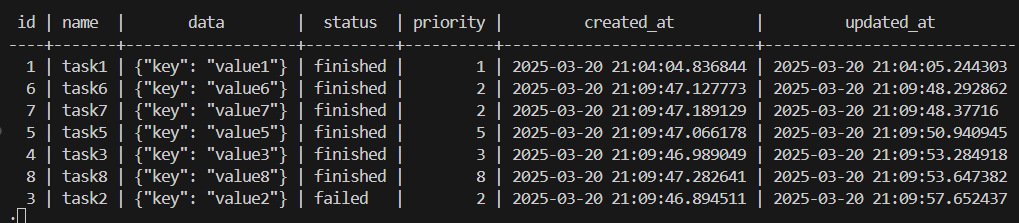
****

Рисунок 2 – Вид базы данных до внесения изменений (начала тестирования)

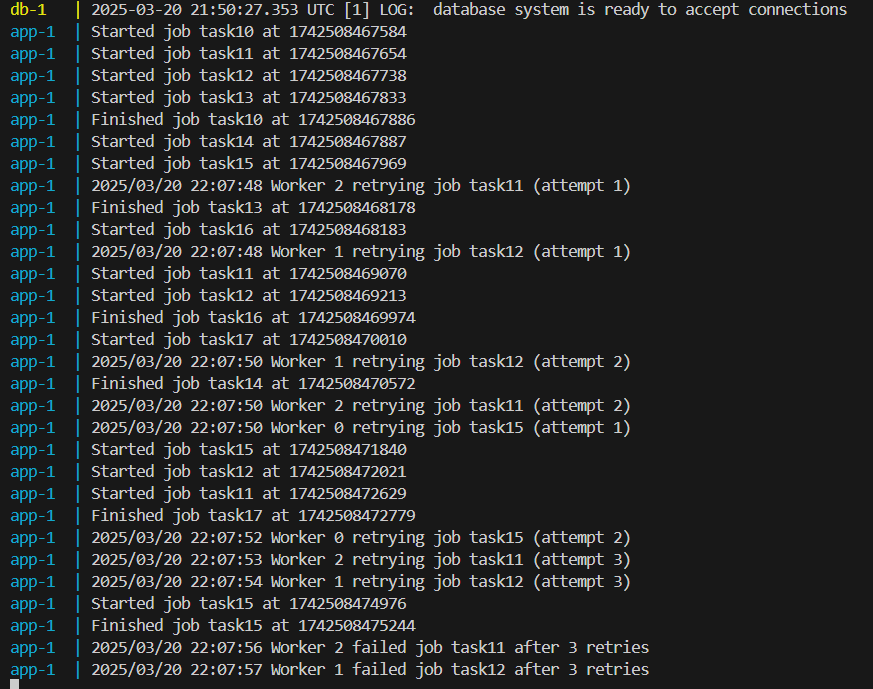
****

Рисунок 3 – Выполнение задач в очереди

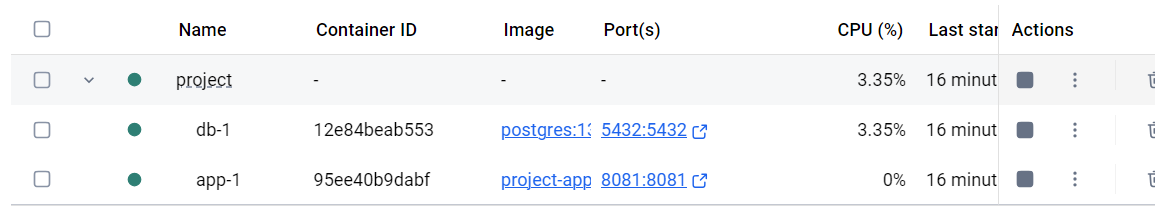
****

Рисунок 4 – Containers в Docker Desktop

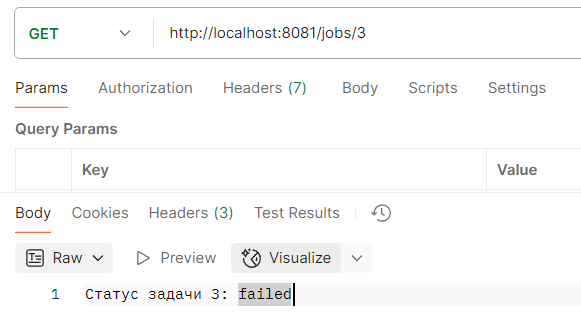
****

Рисунок 5 – Тестирование метода GET для задачи с id 3

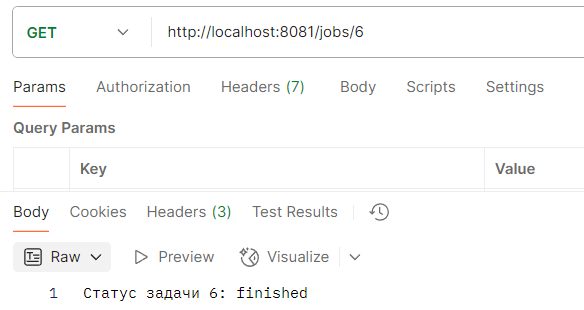
****

Рисунок 6 – Тестирование метода GET для задачи с id 6

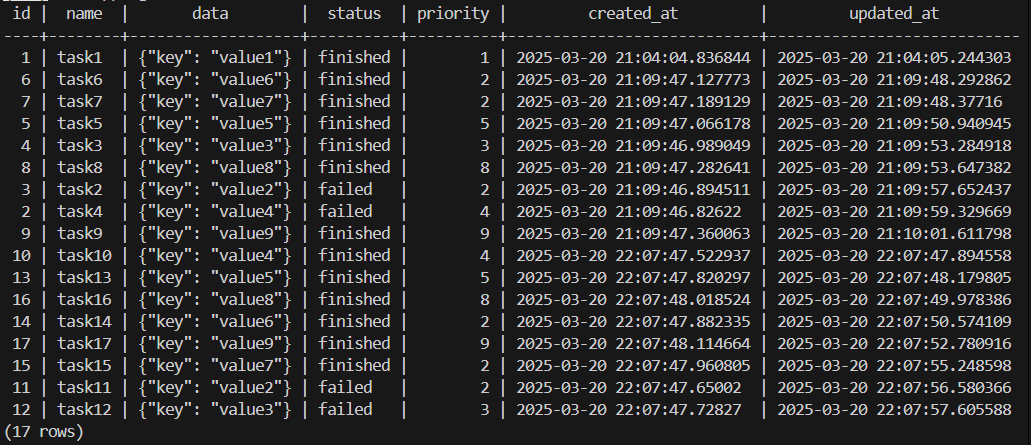
****

Рисунок 7 – Вид базы данных после внесения изменений

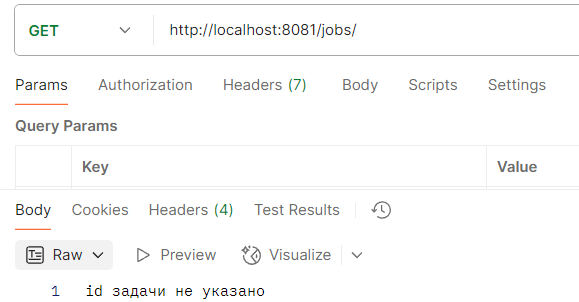
****

Рисунок 8 – Тестирование сообщения об ошибке при отсутствии id

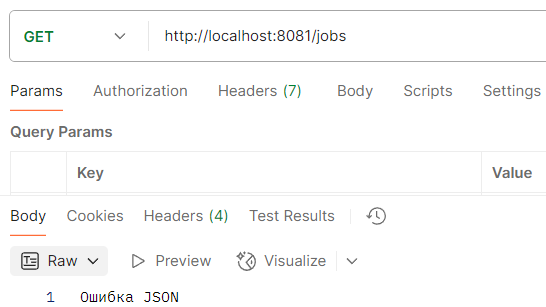
****

Рисунок 9 – Сообщение об ошибке при неверном формате ввода JSON

**Вывод:** было разработано приложение на Go для управления задачами. Реализованы основные функции: создание задачи (POST /jobs), получение информации о задаче (GET /jobs/{id}), обновление статуса задачи. Приложение взаимодействует с базой данных PostgreSQL для хранения данных. Создан Dockerfile для сборки образа приложения. Настроен docker-compose.yml для запуска контейнеров с приложением и базой данных. Успешно запущены контейнеры с помощью команды docker-compose up -d. Проведено тестирование программы.