|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ **Информатика и системы управления**

КАФЕДРА **Компьютерные системы и сети (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.01 Компьютерные системы и сети**

**Отчет**

**по лабораторной работе № 8**

**Название:** Организация клиент-серверного взаимодейсвтия между Golang и PostgreSQL

**Дисциплина:** языки интернет-программирования

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ6-32Б |  |  | И.В.Порохницкий |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  | В.Д.Шульман |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

Москва, 2024

**Цель работы** - получение первичных навыков в организации долгосрочного хранения данных с использованием PostgreSQL и Golang

**Задание**

1. Установить и настроить PostgreSQL
2. Ознакомиться с теоретическими сведениями
3. Сделать форк данного репозитория в GitHub, склонировать получившуюся копию локально, создать от мастера ветку dev и переключиться на неё
4. Перекопировать код сервисов, полученный в ходе выполнения 6-й лабораторной работы, в соответствующие поддиректории в директории cmd (кроме кода сервиса hello, т.к. он уже реализован в качестве примера)
5. Доработать сервисы таким образом, чтобы они использовали для хранения данных СУБД PostgreSQL. Каждый сервис должен как добавлять новые данные в БД (insert/update), так и доставать их для предоставления пользователю (select)
6. Проверить свой код линтерами с помощью команды make lint
7. Сделать отчёт и поместить его в директорию docs
8. Зафиксировать изменения, сделать коммит и отправить получившееся состояние ветки дев в личный форк данного репозитория в GitHub
9. Через интерфейс GitHub создать Pull Request dev --> master

**Ход работы**

1. Установил и настроил PostgreSQL
2. Ознакомился с теоретическими сведениями
3. Сделал форк данного репозитория в GitHub, склонировал получившуюся копию локально, создал от мастера ветку дев и переключился на нее
4. Выполнил 3 задания:

**Задание №1** “count”:

Необходимо написать веб сервер, аналогичный данному из 6-ой лабораторной работы:

Напиши веб сервер (**порт :3333**) - счетчик который будет обрабатывать GET (/count) и POST (/count) запросы:

**GET**:  возвращает счетчик  
**POST**: увеличивает ваш счетчик на значение  (с ключом "count") которое вы получаете из формы, но если пришло НЕ число то нужно ответить клиенту: "это не число" со статусом http.StatusBadRequest (400).

В ходе выполнения программы была построена таблица count в БД sandbox (рис. 1):

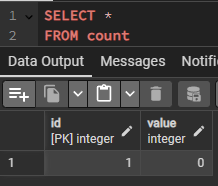
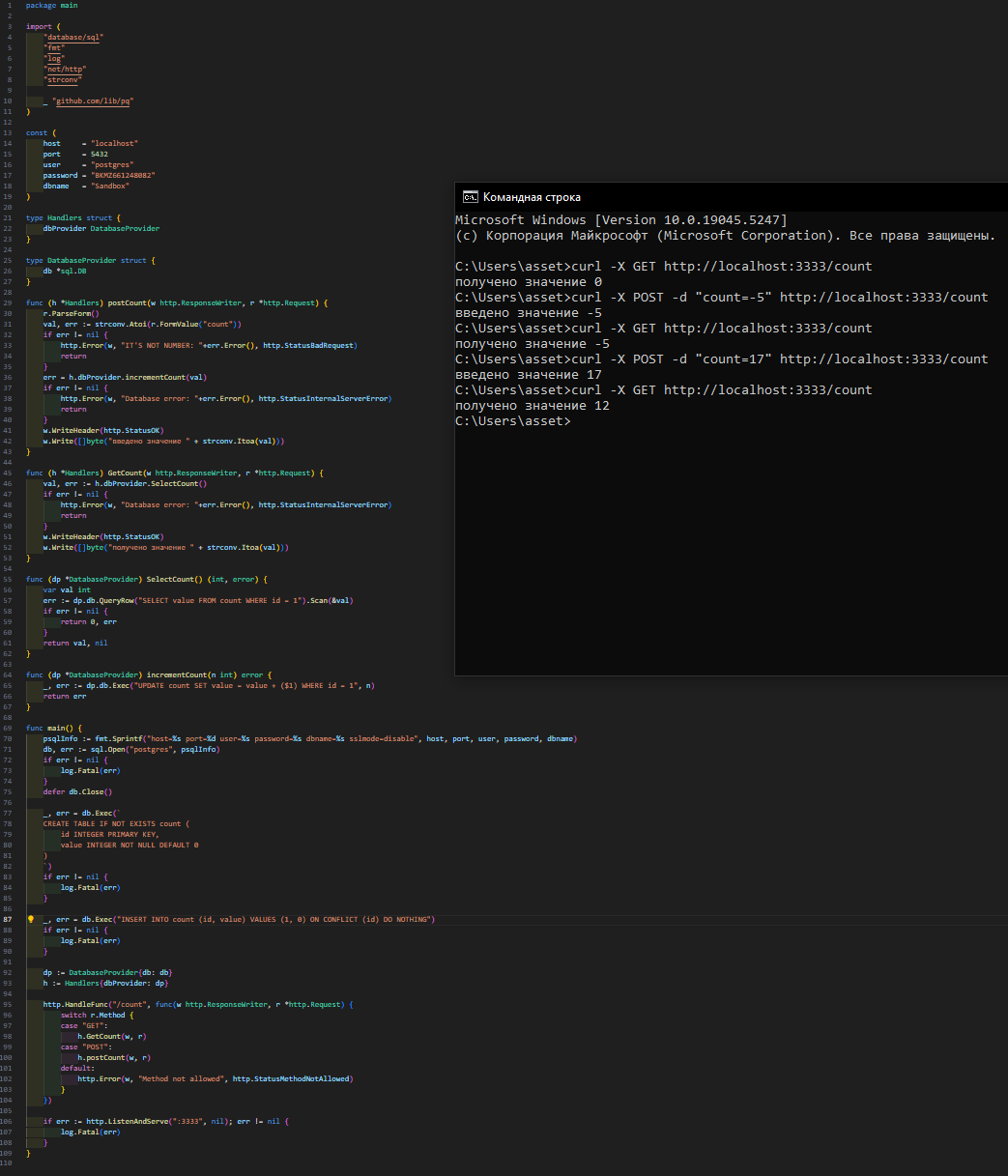


Рис.1

Таблица может как изначально присутствовать, так создаваться автоматически, из поля value по get-запросу берётся значение или изменяется при post-запросе. Результат работы представлен на Рис.2:



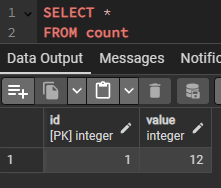


Рис.2

**Задание №2** “query”(Рис. 3):

Напишите веб сервер, который по пути /get отдает текст "Hello, web!".  
Порт должен быть :8080.

В данной реализации сервер выдаёт случайное сообщение из БД, передающееся на сервер JSON-ом

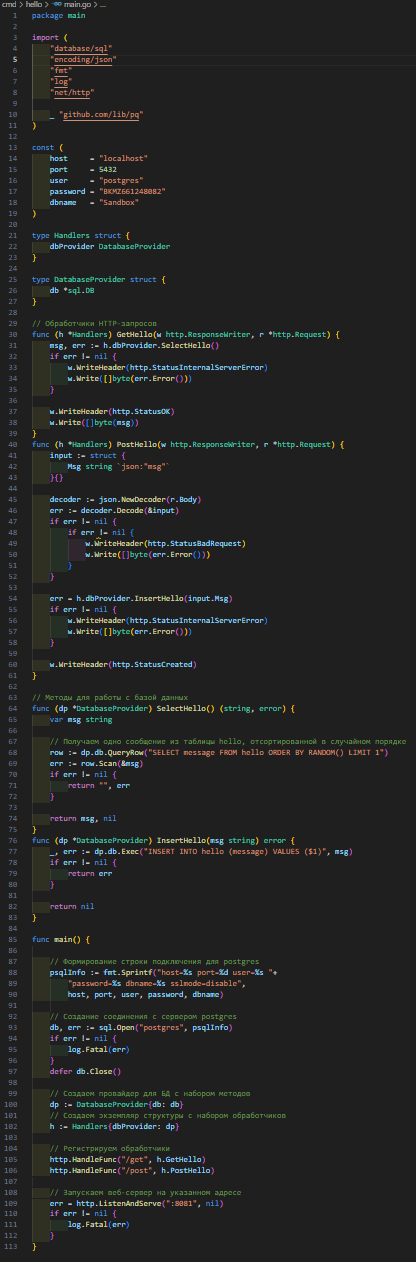


Рис.3

Результат работы (рис. 4):

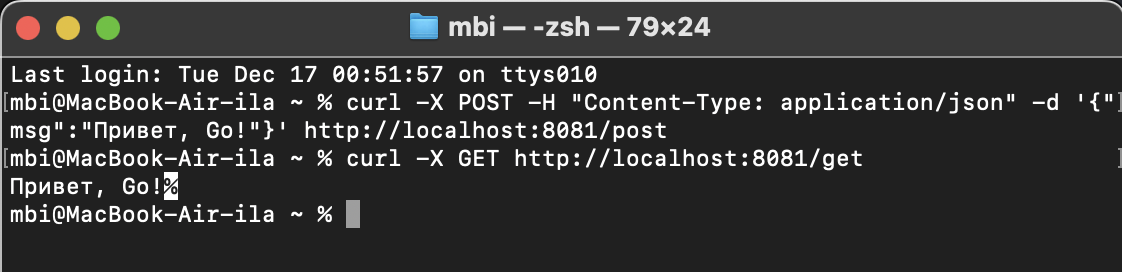


рис. 4

В ходе выполнения программы были внесены изменения в таблицу hello в БД sandbox (рис. 5):

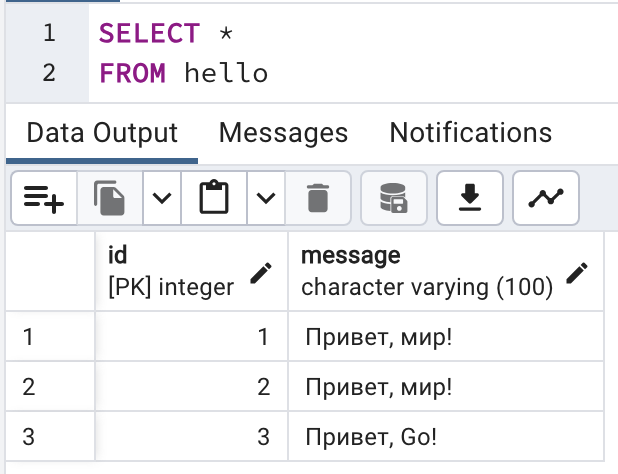


Рис.5

**Задание №2** “query”(Рис. 6):

Напишите веб-сервер который по пути /api/user приветствует пользователя:  
Принимает и парсит параметр name и делает ответ "Hello,<name>!"  
Пример: /api/user?name=Golang  
Ответ: Hello,Golang!

В данной реализации по post-запросу в БД будет поступать name,

По get-запросу будет выводиться сообщение, при наличии на БД данного имени

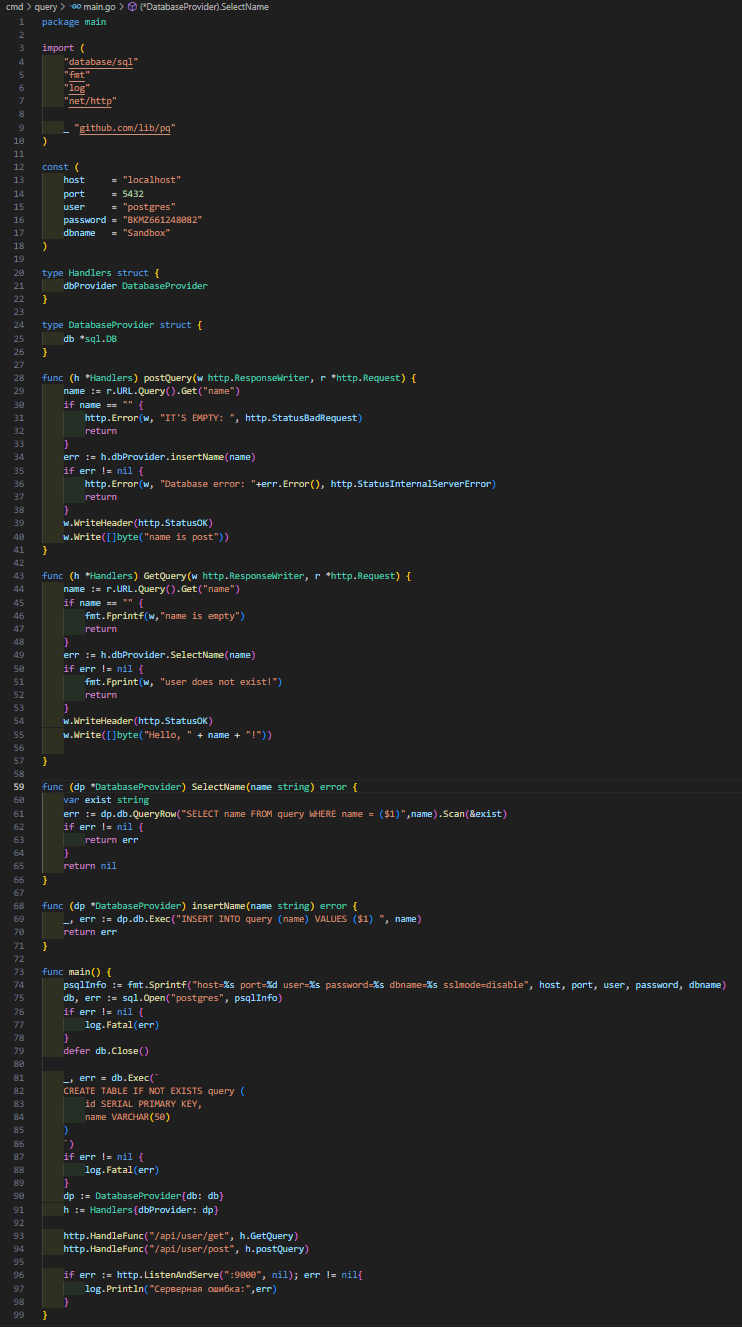


рис. 6

Результат работы (рис. 7):

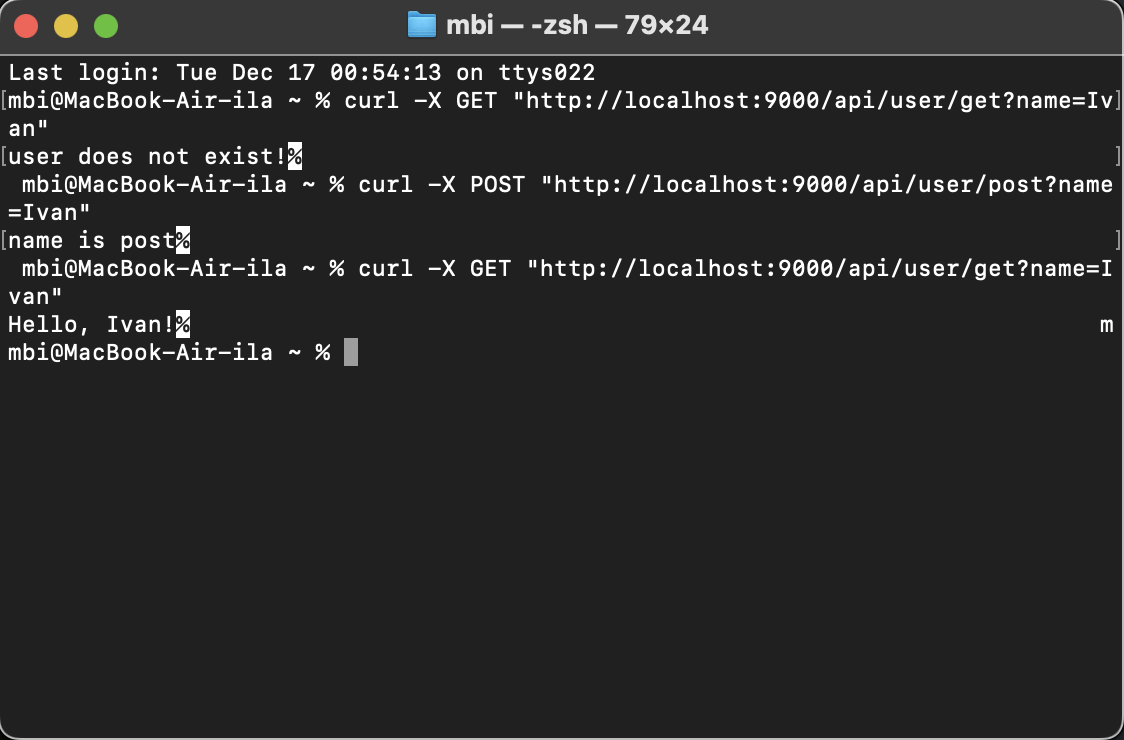


рис. 7

В ходе выполнения программы были внесены изменения в таблицу hello в БД sandbox (рис. 8):

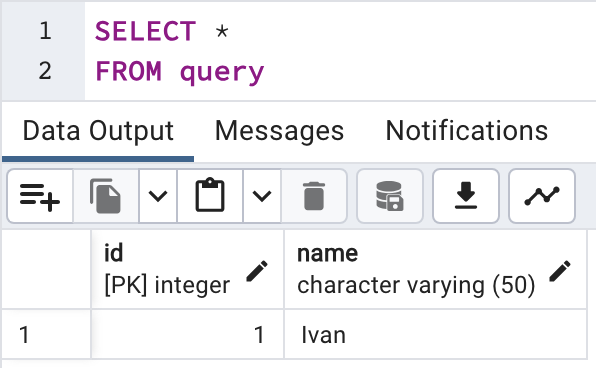


Рис.8

1. Сделал отчёт и поместил его в директорию docs

Зафиксировал изменения, сделал коммит и отправил полученное состояние ветки дев в удаленный репозиторий GitHub

**Вывод**

Я получил первичные навыки в организации долгосрочного хранения данных с использованием PostgreSQL и Golang.