GoKVServer with PostgreSQL and Cache

Student Name: Viraj Ashar
Course: DECS
Project Title: Key-Value Store with Database and Cache
Repository: Github Link
Language & Tools: GO (GoLang), PostgreSQL, godotenv, net/http

Overview

I implemented a simple key-value server in GO that supports CRUD operations through HTTP routes.

It includes 4 separate packages for:

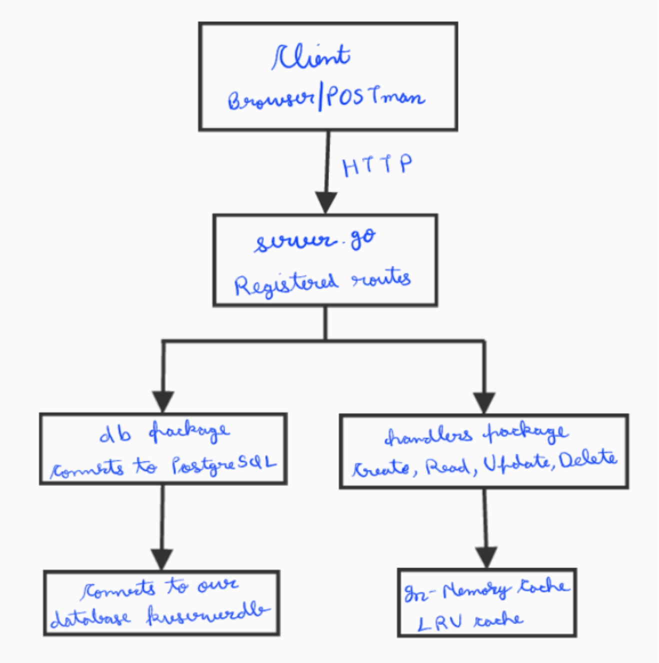
1. Main server setup(main package)
2. Database access (db package)
3. Request handlers (handlers package)
4. Cache access (cache package)

The system supports 2 execution paths:

1. In memory access through cache (Fast to access)
2. Persistent database access through PostgreSQL (Slow to access)

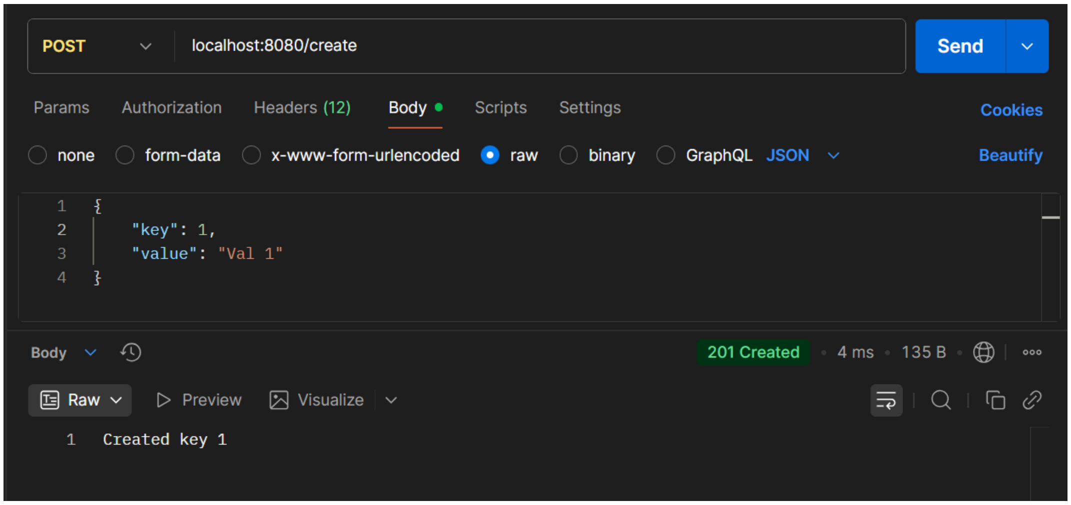
When a key is requested, it first looks in the LRU cache. If not found in it then it looks in the database(disk as of now) and populates to cache and sends the response.

System Architecture



Request Handlers

Route: localhost:8080/create



POST localhost:8080/create

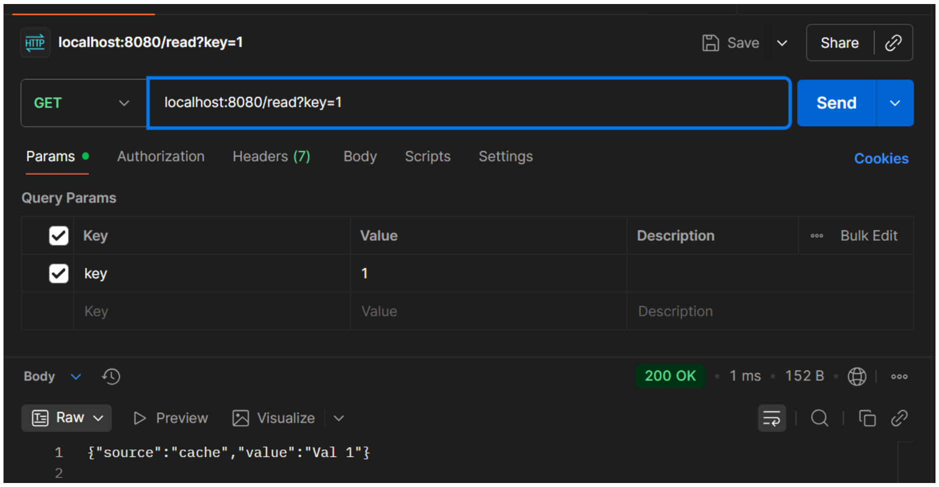
○ none ○ form-data ○ x-www-form-urlencoded ● raw ○ binary ○ GraphQL JSON

```
{
    "key": 1,
    "value": "Val 1"
}
```

201 Created • 4 ms • 135 B

```
Created key 1
```

Route: localhost:8080/read?key=1



GET localhost:8080/read?key=1

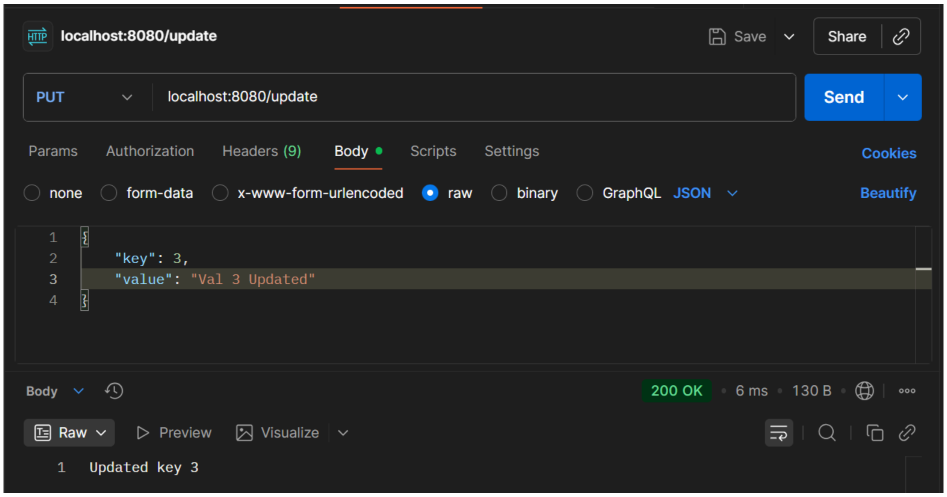
Query Params

	Key	Value	Description
☑	key	1	
	Key	Value	Description

200 OK • 1 ms • 152 B

```
{"source":"cache","value":"Val 1"}
```

Route: localhost:8080/update
Body: { "key": 3, "value": "Val 3 Updated" }



PUT localhost:8080/update

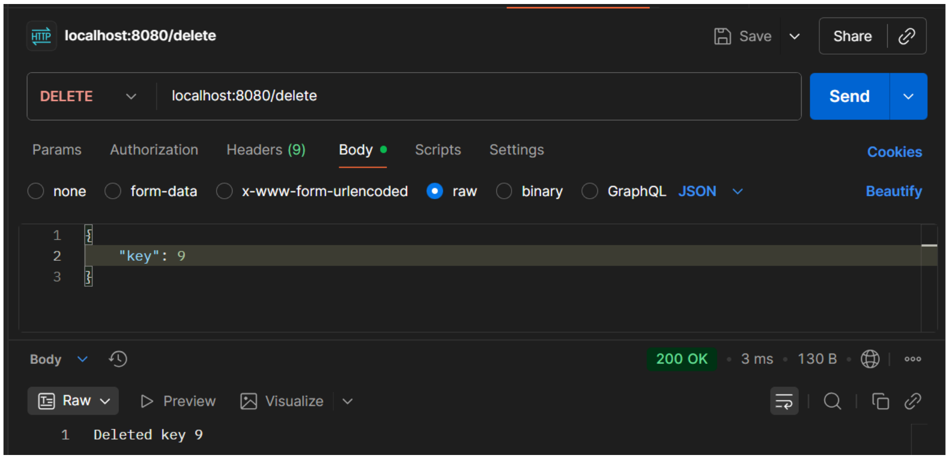
○ none ○ form-data ○ x-www-form-urlencoded ● raw ○ binary ○ GraphQL JSON

```
{
    "key": 3,
    "value": "Val 3 Updated"
}
```

200 OK • 6 ms • 130 B

```
Updated key 3
```

Route: localhost:8080/delete



DELETE localhost:8080/delete

○ none ○ form-data ○ x-www-form-urlencoded ● raw ○ binary ○ GraphQL JSON

```
{
    "key": 9
}
```

200 OK • 3 ms • 130 B

```
Deleted key 9
```

Database Logic

- All PostgresSQL interaction is handled in the `/db` package
- Loads credentials from `.env` using `http://github.com/joho/godotenv`
- Provides functions like
  - InsertKeyValue(key int, value string)
  - GetValue(key int)
  - UpdateValue(key int, value string)
  - DeleteKey(key int)

Caching Layer

- To improve performance, an `LRU cache` is used
- Cache size is handled in `.env` file
- Implemented using combination of HashMap and Doubly Linked List
- For every GET request, if it exists in cache it returns immediately else it reads from DB and add to cache.

How To Run

1. Create `.env` as mentioned below
   ```
   DB_HOST=localhost
   DB_PORT=5432
   DB_USER=postgres
   DB_PASSWORD=13241324
   DB_NAME=kvserverdb
   ```
2. Run `go run server.go`
3. Then you can test the APIs.