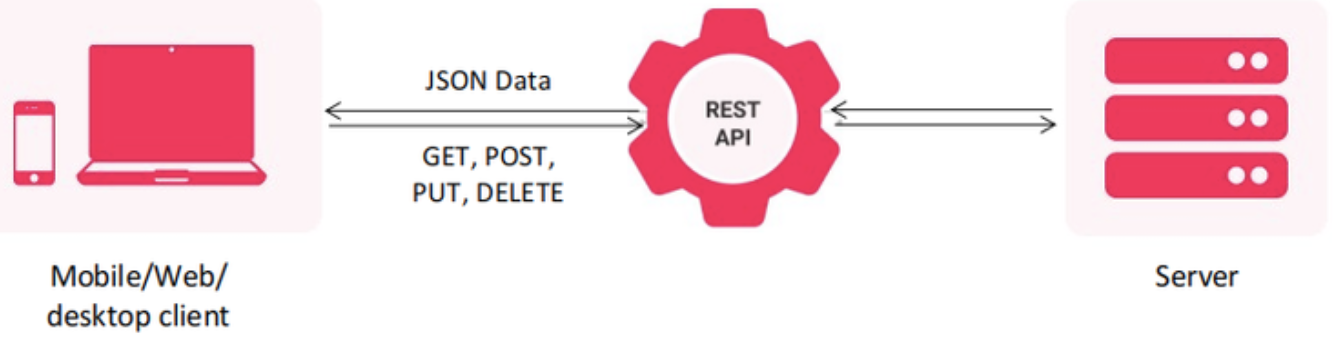
**PRAKTIKUM MCS BAB 6**

**Card bridge**

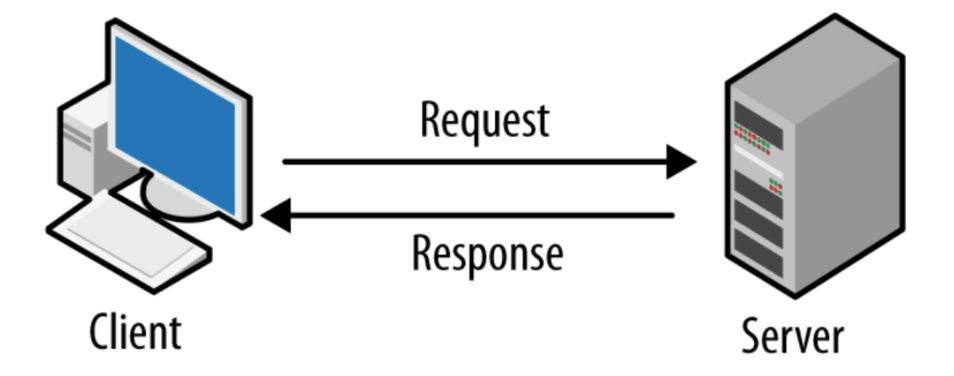
**PENDAHULUAN**

Pada praktikum MCS bab 6 akan belajar bagaimana caranya membangun RESTFUL API sebuah aplikasi back end sebuah layanan yang menjadi penghubung antara end user dengan Internet of Things (IOT). Server menjadi connector antara aplikasi android mobile dan IOT yang dibangun berbasis Uniform Resource Locator (URL). Aplikasi back end yang dibangun menggunakan bahasa pemrograman Go dengan framework yang bernama Gin. Adapun dari sisi IOT menggunakan micro controller esp32 dan sensor Radio Frequency Identification (RFID). Setiap kartu yang dibaca oleh RFID akan masuk ke server kemudian data kartu yang ada di server akan dibaca oleh aplikasi android. Untuk berkomunikasi dengan back end berbasis URL (RESTFUL API) diperlukan beberapa method, beberapa method yang digunakan adalah GET, POST, PUT, DELETE.



Connect server

User mengirim request dengan beberapa method yang digunakan untuk berkomunikasi dengan server kemudian server mengirim response.

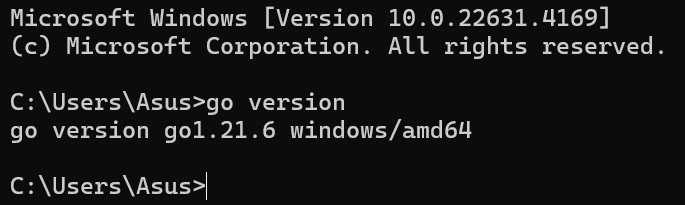


Request and response

Di praktikum ini hanya akan membangun dari sisi server saja dan praktikum akan berlanjut di bab 7 dan 8.

**PRAKTIKUM BAB 6**

Pastikan perangkat yang digunakan untuk membangun REST API sudah terdapat software Go language, Visual Studio Code dengan extensionnya golang dan code runner, Postman, web browser.



Cek versi golang

Buat folder project misalnya “mcs\_bab\_6” kemudian buka command prompt masuk ke path di dalam folder tersebut dan ketik command berikut :

go mod init mcs\_bab\_6

Setelah muncul file go.mod masukkan perintah berikut :

go get -u "github.com/gin-gonic/gin"

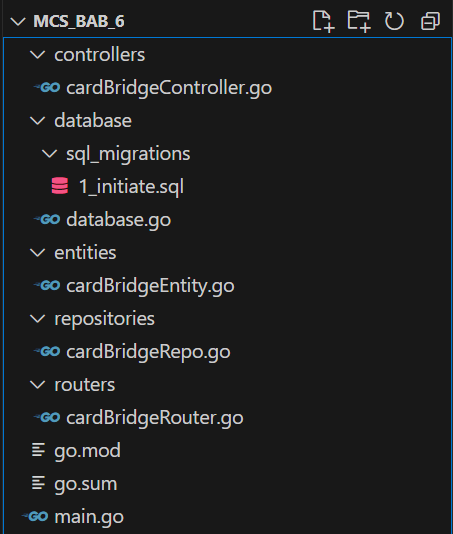
go get -u "github.com/lib/pq"

go get -u "github.com/rubenv/sql-migrate"

go get -u "github.com/joho/godotenv"

Perintah go get -u "github.com/gin-gonic/gin" untuk package Gin framework, Gin framework ,e,udahkan pengembangan API. Package ini menyediakan berbagai fitur seperti routing, middleware dan handling JSON. Perintah go get -u "github.com/lib/pq" adalah instalasi driver untuk PostgreSQL, di mana aplikasi yang dibangun menggunakan Go dapat berkomunikasi dengan PostgreSQL dan mengirim query. Perintah go get -u "github.com/rubenv/sql-migrate" adalah mengunduh / instalasi migrasi sql, dengan migrasi sql pengembang dapat mengelola konfigurasi database. Perintah go get -u "github.com/joho/godotenv" adalah untuk mengunduh / instalasi godotenv yang digunakan untuk membaca file .env. file .env adalah file yang berisi konfigurasi. Semua perintah yang dimasukkan menggunakan -u untuk mendapatkan package terbaru jika ditemukan versi terbaru.

Buat root project dengan folder dan file sebagai berikut :



Root project

Buka file cardBridgeEntity.go dan isi dengan code berikut

package entities

type Card struct {

    ID string `json:"id"`

}

Struct yang ada pada file ini digunakan sebagai model yang disesuaikan dengan table di database. Karena hanya menyimpan id dari kartu saja maka Card dengan tipe struct memiliki 1 property saja yaitu id.

Buka file 1\_initiate.sql dan isi dengan coide berikut :

-- +migrate Up

-- +migrate StatementBegin

CREATE TABLE card(

    id varchar(20)

);

-- +migrate StatementEnd

Code ini adalah untuk membuat table database. Table bernama card dan di dalam table ini memiliki 1 field bernama id dengan tipe varchar yang bisa diisi 20 karakter. Untuk penulisan migrate up, statement begin dan statement end merupakan instruksi. Migrate up adalah instruksi utuk menerapkan semua query SQL ke yang lebih baru. Statement begin dan statemnet end merupakan perintah yang harus dieksekusi sebagai satu kesatuan yang tidak bisa dipisahkan, menjadikan hal tersebut atomisitas sebagai pernyataan yang kompleks.

Buka file database.go dan isi dengan code berikut :

package database

import (

    "database/sql"

    "embed"

    "fmt"

    migrate "github.com/rubenv/sql-migrate"

)

//go:embed sql\_migrations/\*.sql

var dbMigrations embed.FS

var DbCoonnection \*sql.DB

func DBMigrate(dbParam \*sql.DB) {

    migrations := &migrate.EmbedFileSystemMigrationSource{

        FileSystem: dbMigrations,

        Root:       "sql\_migrations",

    }

    n, errs := migrate.Exec(dbParam, "postgres", migrations, migrate.Up)

    if errs != nil {

        panic(errs)

    }

    DbCoonnection = dbParam

    fmt.Println("Migrations success applied", n, migrations)

}

Code pada file ini bertujuan untuk melakukan migrasi database. Melalui parameter yang ada pada funtion DBMigrate() menerima koneksi database dan menjalankan migrasi berdasarkan file yang ada di dalam sql\_migrations. //go:embed sql\_migrations/\*.sql digunakan menyematkan semua file .sql yang ada di dalam folder sql\_migrations ke dalam variabel dbMigrations.

Buka file cardBridgeRepo.dart dan isi dengan code berikut :

package repositories

import (

    "database/sql"

    "mcs\_bab\_6/entities"

)

func GetCards(db \*sql.DB) (result []entities.Card, err error) {

    sql := "SELECT \* FROM card"

    rows, err := db.Query(sql)

    if err != nil {

        return

    }

    defer rows.Close()

    for rows.Next() {

        var data entities.Card

        err = rows.Scan(&data.ID)

        if err != nil {

            return

        }

        result = append(result, data)

    }

    return

}

func InsertCard(db \*sql.DB, card entities.Card) (err error) {

    sql := "INSERT INTO card(id) values($1)"

    // errs := db.QueryRow(sql, 1)

    // return errs.Err()

    \_, err = db.Exec(sql, card.ID)

    return err

}

func DeleteCard(db \*sql.DB, card entities.Card) (err error) {

    sql := "DELETE FROM card WHERE id = $1"

    // errs := db.QueryRow(sql, 1)

    // return errs.Err()

    \_, err = db.Exec(sql, card.ID)

    return err

}

**LAPORAN PENDAHULUAN (LP)**

sdeweqwd

**LAPORAN AKHIR (LA)**

wedwedwe