



# UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

CARRERA DE SOFTWARE

MATERIA: Desarrollo de software para MiPymes

DOCENTE: Ing. Mauricio Rea

ESTUDIANTE: Mario Salazar

FECHA: 11 de enero de 2023

TEMA: **API Golang para una estructura simple de datos (GO)**

## Tabla de Contenidos

DESARROLLO .....	2
Creación de la base de datos .....	2
Instalación de librerías .....	3
Inicializar el proyecto .....	3
Instalar <i>gorilla/mux</i> .....	3
Instalar la librería <i>air</i> .....	3
Inicializar <i>air</i> .....	3
Instalar la librería <i>gorm</i> .....	4
Instalar <i>driver</i> de PostgreSQL .....	4
Conexión a la base de datos PostgreSQL .....	5
Modelo .....	6
Modelo de la tabla empresa .....	6
Controlador .....	6
Importar dependencias para el controlador .....	6
Método insertar empresa .....	6
Método buscar empresa por el id .....	7
Método obtener la lista de las empresas .....	7
Método actualizar empresa .....	8
Método eliminar empresa .....	8
Routers .....	9
Importar las dependencias .....	9
Método GET (lista de empresas) .....	9
Método GET (buscar empresa por el id) .....	10
Método PUT (actualizar empresa) .....	10
Método DELETE (eliminar empresa) .....	10
Método POST (insertar empresa) .....	11



# UNIVERSIDAD TÉCNICA DEL NORTE

## FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

### CARRERA DE SOFTWARE

Main .....	11
Métodos HTTP – POST, GET, PUT, DELETE .....	11
Resultados de la API .....	12
Ejecución del método HTTP – POST crear empresa.....	12
Ejecución del método HTTP – GET buscar empresa por id .....	12
Ejecución del método HTTP – GET lista de empresas .....	13
Ejecución del método HTTP – PUT actualizar empresa .....	13
Ejecución del método HTTP – DELETE eliminar empresa .....	14
CONCLUSIÓN .....	14

## DESARROLLO

### Creación de la base de datos

Explicación del código
Código
<pre>CREATE TABLE empresa( id_empresa SERIAL NOT NULL, emp_ruc VARCHAR(13) NOT NULL, emp_nombre_empresa VARCHAR(100) NOT NULL, emp_matriz VARCHAR(100) NOT NULL, emp_sucursal VARCHAR(100), emp_pais VARCHAR(50) NOT NULL, emp_provincia VARCHAR(50) NOT NULL, emp_ciudad VARCHAR(50) NOT NULL, emp_telefono VARCHAR(15) NOT NULL, emp_email VARCHAR(60) NOT NULL UNIQUE, nro_empleados INTEGER NOT NULL, ingresos_anuales NUMERIC(10,2), CONSTRAINT pk_empresa PRIMARY KEY(id_empresa) );</pre>



# UNIVERSIDAD TÉCNICA DEL NORTE

## FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

### CARRERA DE SOFTWARE

## Instalación de librerías

### Inicializar el proyecto

Inicializar el proyecto <i>go</i> con GitHub
Código
PS C:\Users\Mario\OneDrive\Desktop\Projects\Git\mipymes\api_rest_go> go mod init github.com/MarioUTN/api_rest_go go: creating new go.mod: module github.com/MarioUTN/api_rest_go go: to add module requirements and sums: go mod tidy PS C:\Users\Mario\OneDrive\Desktop\Projects\Git\mipymes\api_rest_go>

### Instalar *gorilla/mux*

<i>Gorilla/mux</i>
Código
PS C:\Users\Mario\OneDrive\Desktop\Projects\Git\mipymes\api_rest_go> go get -u github.com/gorilla/mux go: downloading github.com/gorilla/mux v1.8.0 go: added github.com/gorilla/mux v1.8.0 PS C:\Users\Mario\OneDrive\Desktop\Projects\Git\mipymes\api_rest_go>

### Instalar la librería *air*

<i>air</i>
Código
PS C:\Users\Mario\OneDrive\Desktop\Projects\Git\mipymes\api_rest_go> go install github.com/cosmtrek/air@latest go: downloading github.com/cosmtrek/air v1.40.4 go: downloading github.com/fatih/color v1.10.0 go: downloading github.com/fsnotify/fsnotify v1.4.9 go: downloading github.com/imaario/mergo v0.3.12 go: downloading github.com/pelletier/go-toml v1.8.1 go: downloading github.com/mattn/go-colorable v0.1.8 go: downloading github.com/mattn/go-isatty v0.0.12 PS C:\Users\Mario\OneDrive\Desktop\Projects\Git\mipymes\api_rest_go>

### Inicializar *air*

Inicializar <i>air</i>
Código
PS C:\Users\Mario\OneDrive\Desktop\Projects\Git\mipymes\api_rest_go> air init  _ _ _ _ /_--\  _   _  \ , built with Go  .air.toml file created to the current directory with the default settings PS C:\Users\Mario\OneDrive\Desktop\Projects\Git\mipymes\api_rest_go>



# UNIVERSIDAD TÉCNICA DEL NORTE

## FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

### CARRERA DE SOFTWARE

#### Instalar la librería *gorm*

<i>gorm</i>
Código
<pre>PS C:\Users\Mario\OneDrive\Desktop\Projects\Git\mipymes\api_rest_go&gt; go get -u gorm.io/gorm go: downloading gorm.io/gorm v1.24.3 go: downloading github.com/jinzhu/inflection v1.0.0 go: downloading github.com/jinzhu/now v1.1.4 go: downloading github.com/jinzhu/now v1.1.5 go: added github.com/jinzhu/inflection v1.0.0 go: added github.com/jinzhu/now v1.1.5 go: added gorm.io/gorm v1.24.3 PS C:\Users\Mario\OneDrive\Desktop\Projects\Git\mipymes\api_rest_go&gt;  </pre>

#### Instalar *driver* de PostgreSQL

PostgreSQL
Código
<pre>PS C:\Users\Mario\OneDrive\Desktop\Projects\Git\mipymes\api_rest_go&gt; go get -u gorm.io/driver/postgres go: downloading gorm.io/driver/postgres v1.4.6 go: downloading github.com/jackc/pgx/v5 v5.2.0 go: downloading github.com/jackc/pgx v3.6.2+incompatible go: downloading github.com/jackc/pgpassfile v1.0.0 go: downloading github.com/jackc/pgservicefile v0.0.0-20221227161230-091c0ba34f0a go: downloading golang.org/x/crypto v0.4.0 go: downloading golang.org/x/text v0.5.0 go: downloading golang.org/x/crypto v0.5.0 go: downloading golang.org/x/text v0.6.0 go: added github.com/jackc/pgpassfile v1.0.0 go: added github.com/jackc/pgservicefile v0.0.0-20221227161230-091c0ba34f0a go: added github.com/jackc/pgx/v5 v5.2.0 go: added golang.org/x/crypto v0.5.0 go: added golang.org/x/text v0.6.0 go: added gorm.io/driver/postgres v1.4.6 PS C:\Users\Mario\OneDrive\Desktop\Projects\Git\mipymes\api_rest_go&gt;  </pre>



# UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS  
CARRERA DE SOFTWARE

## Conexión a la base de datos PostgreSQL

### Método para la conexión a la base de datos

#### Código

```
package database

import (
    "fmt"
    "gorm.io/driver/postgres"
    "gorm.io/gorm"
    "log"
)

var DB *gorm.DB
var err error

func DatabaseConnection() (*gorm.DB, error) {
    host := "localhost"
    port := "5432"
    dbName := "app_golang"
    dbUser := "postgres"
    password := "password-postgresql"
    dsn := fmt.Sprintf("host=%s port=%s user=%s dbname=%s password=%s sslmode=disable",
        host,
        port,
        dbUser,
        dbName,
        password,
    )

    DB, err = gorm.Open(postgres.Open(dsn), &gorm.Config{})
    if err != nil {
        log.Fatal("Error connecting to the database...", err)
    } else {
        fmt.Println("Database connection successful...")
    }
    return DB, err
}
```



# UNIVERSIDAD TÉCNICA DEL NORTE

## FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

### CARRERA DE SOFTWARE

## Modelo

### Modelo de la tabla empresa

Creamos el modelo para representar a la tabla empresa	
Código	
	<pre>package models  import (     "gorm.io/gorm" )  type Empresa struct {     gorm.Model      Id_empresa      uint   `json:"id_empresa"`     Emp_ruc          string `json:"emp_ruc"`     Emp_nombre_empresa string `json:"emp_nombre_empresa"`     Emp_matriz       string `json:"emp_matriz"`     Emp_sucursal     string `json:"emp_sucursal"`     Emp_pais         string `json:"emp_pais"`     Emp_provincia    string `json:"emp_provincia"`     Emp_ciudad       string `json:"emp_ciudad"`     Emp_telefono     string `json:"emp_telefono"`     Emp_email        string `json:"emp_email"`     Nro_empleados    int    `json:"nro_empleados"`     Ingresos_anuales float64 `json:"ingresos_anuales"` }</pre>

## Controlador

### Importar dependencias para el controlador

Instalación de dependencias	
Código	
	<pre>package controller  import (     "fmt"     "github.com/MarioUTN/api_rest_go/database"     "github.com/MarioUTN/api_rest_go/models" )</pre>

### Método insertar empresa

Insertar una nueva empresa en la base de datos
Código



# UNIVERSIDAD TÉCNICA DEL NORTE

## FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

### CARRERA DE SOFTWARE

```
func Create_Company(e models.Empresa) error {
    db, err := database.DatabaseConnection()
    if err != nil {
        fmt.Println("Error: ", err.Error())
    }
    query := "INSERT INTO empresa (emp_ruc, emp_nombre_empresa, emp_matriz, emp_sucursal, emp_pais,
    emp_provincia, emp_ciudad, emp_telefono, emp_email, nro_empleados, ingresos_anuales) VALUES
    ($1, $2, $3, $4, $5, $6, $7, $8, $9, $10, $11)"
    err = db.Exec(query, e.Emp_ruc, e.Emp_nombre_empresa, e.Emp_matriz, e.Emp_sucursal, e.Emp_pais,
    e.Emp_provincia, e.Emp_ciudad, e.Emp_telefono, e.Emp_email, e.Nro_empleados, e.
    Ingresos_anuales).Error
    return err
}
```

#### Método buscar empresa por el id

Buscar la empresa por medio del parámetro id

Código

```
func Get_CompanyById(id int) (models.Empresa, error) {
    db, err := database.DatabaseConnection()
    var e models.Empresa
    if err != nil {
        fmt.Println("Error: ", err.Error())
    }
    query := "SELECT id_empresa, emp_ruc, emp_nombre_empresa, emp_matriz, emp_sucursal, emp_pais,
    emp_provincia, emp_ciudad, emp_telefono, emp_email, nro_empleados, ingresos_anuales FROM
    empresa WHERE id_empresa = $1"
    row := db.Raw(query, id).Row()
    err = row.Scan(&e.Id_empresa, &e.Emp_ruc, &e.Emp_nombre_empresa, &e.Emp_matriz, &e.
    Emp_sucursal, &e.Emp_pais, &e.Emp_provincia, &e.Emp_ciudad, &e.Emp_telefono, &e.Emp_email, &e.
    Nro_empleados, &e.Ingresos_anuales)
    if err != nil {
        return e, err
    }
    return e, nil
}
```

#### Método obtener la lista de las empresas

Obtener todas las empresas de la base de datos

Código



# UNIVERSIDAD TÉCNICA DEL NORTE

## FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

### CARRERA DE SOFTWARE

```
func Get_AllCompanys() ([]models.Empresa, error) {  
    db, err := database.DatabaseConnection()  
    companys := []models.Empresa{}  
    if err != nil {  
        fmt.Println("Error: ", err.Error())  
    }  
    query := "SELECT id_empresa, emp_ruc, emp_nombre_empresa, emp_matriz, emp_sucursal, emp_pais,  
emp_provincia, emp_ciudad, emp_telefono, emp_email, nro_empleados, ingresos_anuales FROM  
empresa"  
    rows, err := db.Raw(query).Rows()  
    if err != nil {  
        return companys, err  
    }  
    for rows.Next() {  
        var e models.Empresa  
        err = rows.Scan(&e.Id_empresa, &e.Emp_ruc, &e.Emp_nombre_empresa, &e.Emp_matriz, &e.  
Emp_sucursal, &e.Emp_pais, &e.Emp_provincia, &e.Emp_ciudad, &e.Emp_telefono, &e.Emp_email, &  
e.Nro_empleados, &e.Ingresos_anuales)  
        if err != nil {  
            return companys, err  
        }  
        // and append it to the array  
        companys = append(companys, e)  
    }  
    return companys, nil  
}
```

#### Método actualizar empresa

##### Método para actualizar una empresa por el id

##### Código

```
func Update_Company(id int, e models.Empresa) error {  
    db, err := database.DatabaseConnection()  
    c, err := Get_CompanyById(id)  
    if err != nil || c.Id_empresa == 0 {  
        fmt.Println("Error: ", err.Error())  
    } else {  
        query := "UPDATE empresa SET emp_ruc = $1, emp_nombre_empresa = $2, emp_matriz = $3,  
emp_sucursal = $4, emp_pais = $5, emp_provincia = $6, emp_ciudad = $7, emp_telefono = $8,  
emp_email = $9, nro_empleados = $10, ingresos_anuales = $11 WHERE id_empresa = $12"  
        err = db.Exec(query, e.Emp_ruc, e.Emp_nombre_empresa, e.Emp_matriz, e.Emp_sucursal, e.  
Emp_pais, e.Emp_provincia, e.Emp_ciudad, e.Emp_telefono, e.Emp_email, e.Nro_empleados, e.  
Ingresos_anuales, id).Error  
    }  
    return err  
}
```

#### Método eliminar empresa

##### Método para eliminar empresa por el id

##### Código





# UNIVERSIDAD TÉCNICA DEL NORTE

## FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

### CARRERA DE SOFTWARE

```
func Delete_Company(id int) error {  
    db, err := database.DatabaseConnection()  
    c, err := Get_CompanyById(id)  
    if err != nil {  
        fmt.Println("Error, Company not found!: ", err.Error())  
    } else {  
        query := "DELETE FROM empresa WHERE id_empresa = $1"  
        err = db.Exec(query, c.Id_empresa).Error  
    }  
    return err  
}
```

## Routers

### Importar las dependencias

Insertar datos en la base de datos

Código

```
package routes  
  
import (  
    "encoding/json"  
    "github.com/MarioUTN/api_rest_go/controller"  
    "github.com/MarioUTN/api_rest_go/models"  
    "github.com/gorilla/mux"  
    "io/ioutil"  
    "net/http"  
    "strconv"  
)
```

### Método GET (lista de empresas)

GET: retornar lista de empresas

Código

```
func GetCompanies(w http.ResponseWriter, r *http.Request) {  
    companys, err := controller.Get_AllCompanies()  
    if err != nil {  
        w.Write([]byte("Error to get Companys!"))  
    } else {  
        json.NewEncoder(w).Encode(companys)  
    }  
}
```



# UNIVERSIDAD TÉCNICA DEL NORTE

## FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

### CARRERA DE SOFTWARE

#### Método GET (buscar empresa por el id)

GET: listar empresa por el id	
Código	
<pre>func GetCompany_ById(w http.ResponseWriter, r *http.Request) {     id := mux.Vars(r)["id"]     id_empresa, err := strconv.ParseInt(id, 10, 0)     if err != nil {         w.Write([]byte("Error on Id"))     }     e, err := controller.Get_CompanyById(int(id_empresa))     if err != nil {         w.Write([]byte("Error on Get Company By Id"))     } else {         json.NewEncoder(w).Encode(e)     } }</pre>	

#### Método PUT (actualizar empresa)

PUT: actualizar una empresa por id	
Código	
<pre>func UpdateCompany(w http.ResponseWriter, r *http.Request) {     requestBody, _ := ioutil.ReadAll(r.Body)     var e models.Empresa     json.Unmarshal(requestBody, &amp;e)     id := mux.Vars(r)["id"]     id_empresa, err := strconv.ParseInt(id, 10, 0)     if err != nil {         w.Write([]byte("Error on Id"))     }     err = controller.Update_Company(int(id_empresa), e)     if err != nil {         w.Write([]byte("Update failed!"))     } else {         w.Write([]byte("Update Company successfully!"))     } }</pre>	

#### Método DELETE (eliminar empresa)

DELETE: eliminar empresa por id	
Código	



# UNIVERSIDAD TÉCNICA DEL NORTE

## FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

### CARRERA DE SOFTWARE

```
func DeleteCompany(w http.ResponseWriter, r *http.Request) {  
    id := mux.Vars(r)["id"]  
    id_empresa, err := strconv.ParseInt(id, 10, 0)  
    if err != nil {  
        w.Write([]byte("Error on Id"))  
        // We return, so we stop the function flow  
    }  
    err = controller.Delete_Company(int(id_empresa))  
    if err != nil {  
        w.Write([]byte("Delete failed!"))  
    } else {  
        w.Write([]byte("Delete Company successfully!"))  
    }  
}
```

#### Método POST (insertar empresa)

Insertar datos en la base de datos	
Código	
<pre>func CreateCompany(w http.ResponseWriter, r *http.Request) {     requestBody, _ := ioutil.ReadAll(r.Body)     var e models.Empresa     json.Unmarshal(requestBody, &amp;e)     err := controller.Create_Company(e)     if err != nil {         w.Write([]byte("Insert failed!"))     } else {         json.NewEncoder(w).Encode(&amp;e)     } }</pre>	

#### Main

#### Métodos HTTP – POST, GET, PUT, DELETE

Creación de los métodos HTTP – POST, GET, PUT, DELETE
Código



# UNIVERSIDAD TÉCNICA DEL NORTE

## FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

### CARRERA DE SOFTWARE

```
package main

import (
    "github.com/MarioUTN/api_rest_go/database"
    "github.com/MarioUTN/api_rest_go/routes"
    "github.com/gorilla/mux"
    "net/http"
)

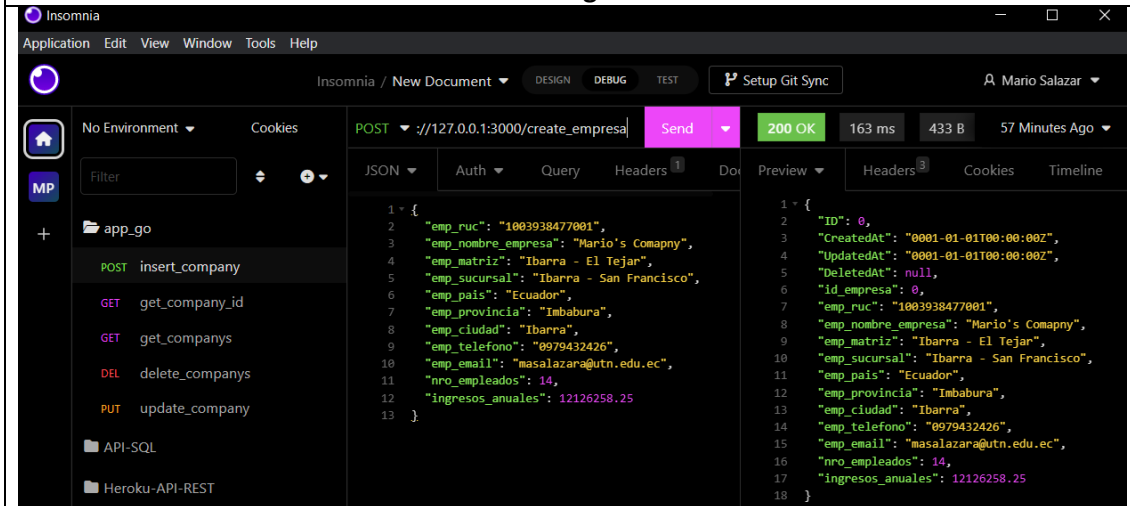
func main() {
    database.DatabaseConnection()
    r := mux.NewRouter()

    r.HandleFunc("/", routes.HolaFunction)
    r.HandleFunc("/get_empresas", routes.GetCompanies).Methods("GET")
    r.HandleFunc("/get_empresabyid/{id}", routes.GetCompany_ById).Methods("GET")
    r.HandleFunc("/update_empresa/{id}", routes.UpdateCompany).Methods("PUT")
    r.HandleFunc("/delete_empresa/{id}", routes.DeleteCompany).Methods("DELETE")
    r.HandleFunc("/create_empresa", routes.CreateCompany).Methods("POST")

    http.ListenAndServe(":3000", r)
}
```

## Resultados de la API

### Ejecución del método HTTP – POST crear empresa

Insertar empresa desde Insomnia	
Código	
	

### Ejecución del método HTTP – GET buscar empresa por id

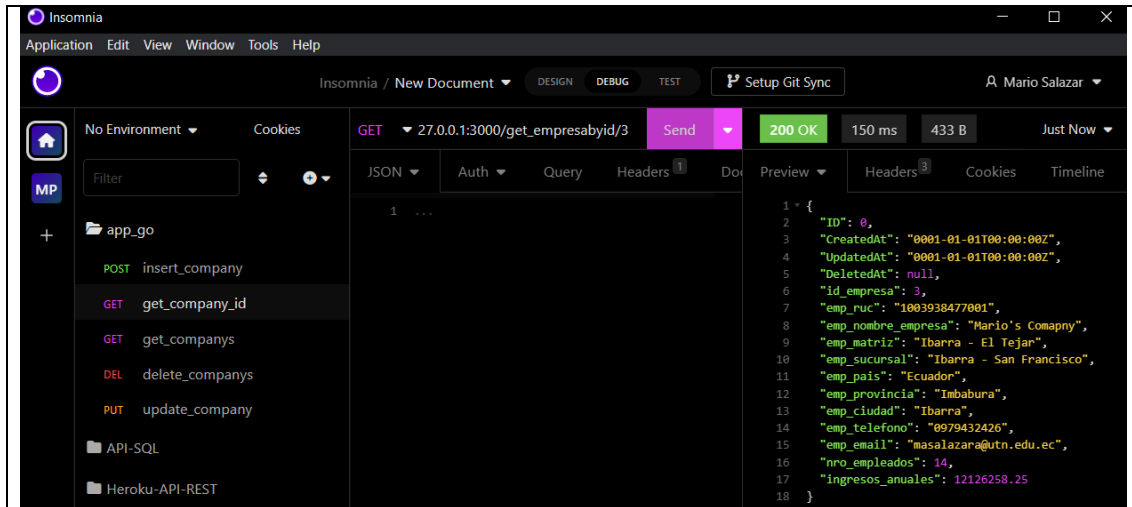
Buscar empresa por el id desde Insomnia
Código



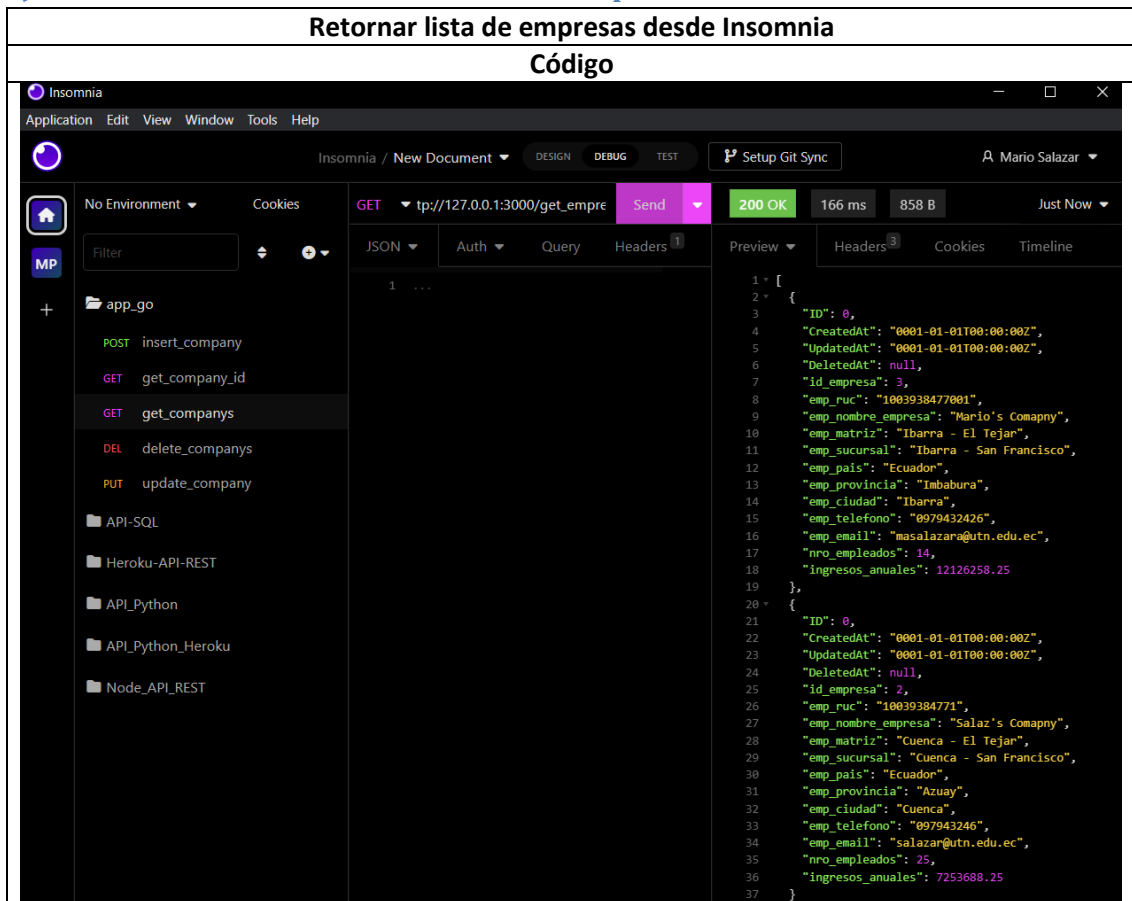
# UNIVERSIDAD TÉCNICA DEL NORTE

## FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

### CARRERA DE SOFTWARE

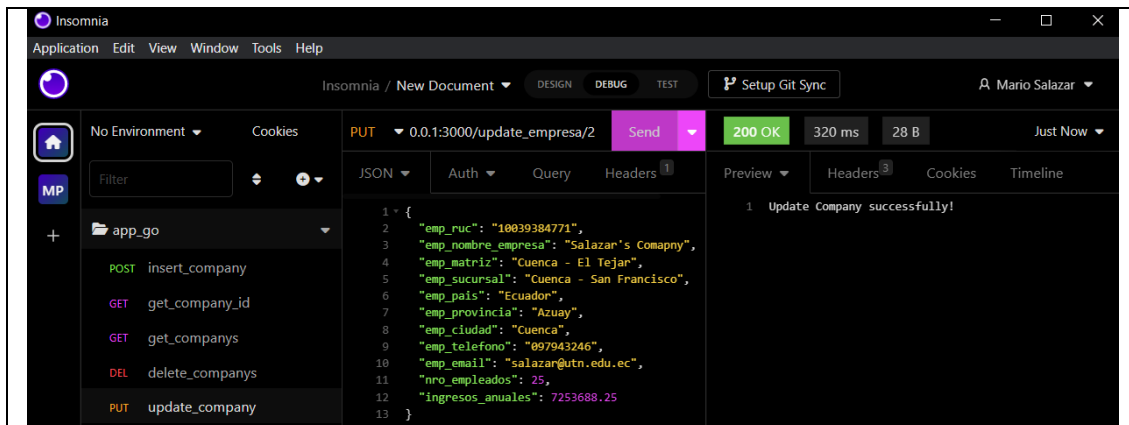


#### Ejecución del método HTTP – GET lista de empresas



#### Ejecución del método HTTP – PUT actualizar empresa

Actualizar una empresa desde Insomnia
Código



### Ejecución del método HTTP – DELETE eliminar empresa



## CONCLUSIÓN

En esta actividad se realizó los métodos HTTP, son el formato de comunicación entre el cliente y servidor web. Maneja varios formatos: POST, GET, PUT, DELETE correspondiente a insertar, leer, actualizar y eliminar respectivamente o comúnmente conocido como CRUD (*Create Read Update Delete*).

Esta práctica permite adquirir los tipos de datos existentes en el lenguaje de *go*, realizar una estructura para la manipulación de una tabla desde la base de datos y su reconocimiento por *json*.

En conclusión, podemos adquirir conocimientos muy básicos y se trabajó con modelo y un controlador y base de datos para la API usando *go*. El enrutamiento se definió mediante funciones, donde dentro de estas funciones recibe el enrutador, de este modo hacemos las cosas más simples y separamos conceptos.

Link del proyecto en GitHub: [https://github.com/MarioUTN/app\\_go\\_salazarmario.git](https://github.com/MarioUTN/app_go_salazarmario.git)