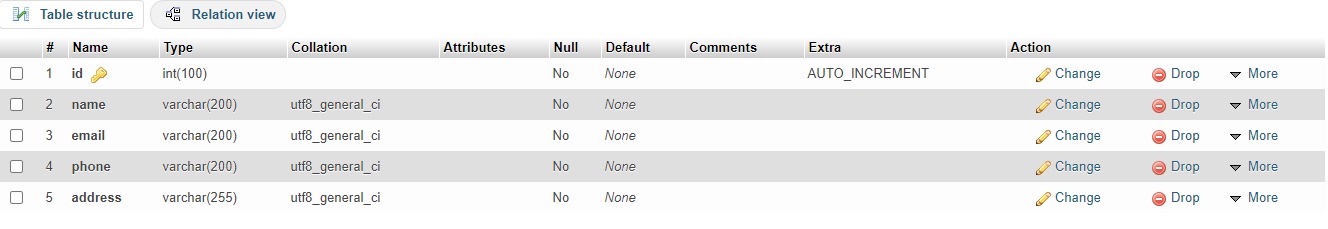
**Requirements**

* First of all, please download and install the Golang package first [**here**](https://golang.org/dl/).
* After the golang is successfully installed, make sure you have installed mysql. I am using XAMPP application which already have mysql.
* Run the apache and mysql service on XAMPP.
* If you are running on localhost, turn off the proxy and SSL checks. I have had experienced some glitches( User accounts🡪 localhost🡪uncheck proxy and SSl).
* If everything is installed, the next step is to create a database to store data from our API. Please create a database with the name **first\_go**, then create a table with the name **user**with a structure like the below:

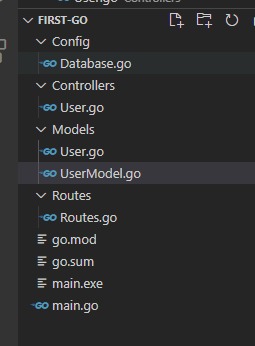
.

* Final POSTMAN for testing the API

**Create New Project**

Create a new folder for the project inside src where go is located. In my case, go is located in Home folder and I created first-api folder inside the src folder of go. Now lets open it in your favorite editor, in my case it is VSCode. So, to initialize the new module, we have to enter following command in our vscode terminal inside the project.

***go mod init***



**Connection Parameters:**

***MyPort: // localhost: 8080***

***connectdb function***

***root => username used on mysql  
1234 => password used on mysql.***

### if you don’t use a password it can be immediately emptied. localhost => host that we use. if we run it on our local PC then it can be written localhost or 127.0.0.1. if you use a server then the server can be written ip / hostname.

***3306 => mysql port  
first\_go => name of the database that we are using***

**Installation**

Here I used the gin-gonic framework. please type this in cmd or terminal if using linux.

***go get github.com/gin-gonic/gin***

***go get github.com/go-sql-driver/mysql  
go get github.com/jinzhu/gorm***

***go get gopkg.in/go-playground/validator.v9***

**ENDPOINTS:**

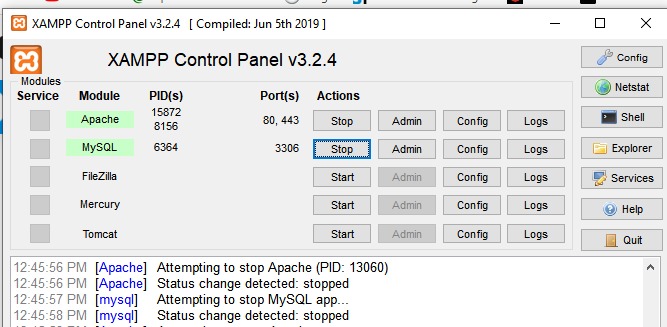
* ***GET user-api/user → Retrieves all the user data***
* ***POST user-api/user → Add new user data***
* ***GET user-api/user/{id} → Retrieve the single user data***
* ***PUT user-api/user/{id} → Update the user data***
* ***DELETE user-api/user → Delete the user data***

**Build & run the API**

***go build main.go***

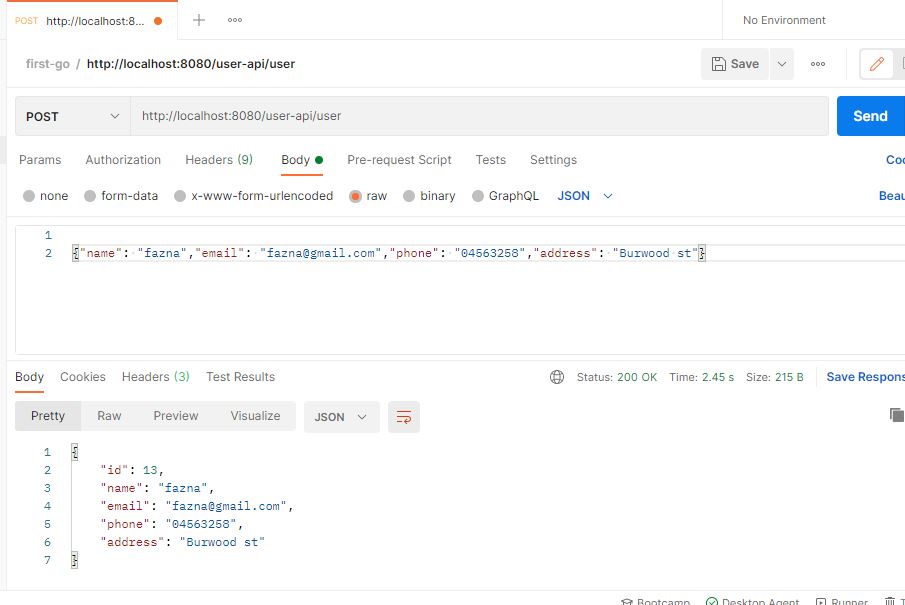
***start main.exe***

**Make sure the Xampp and Postman is working**

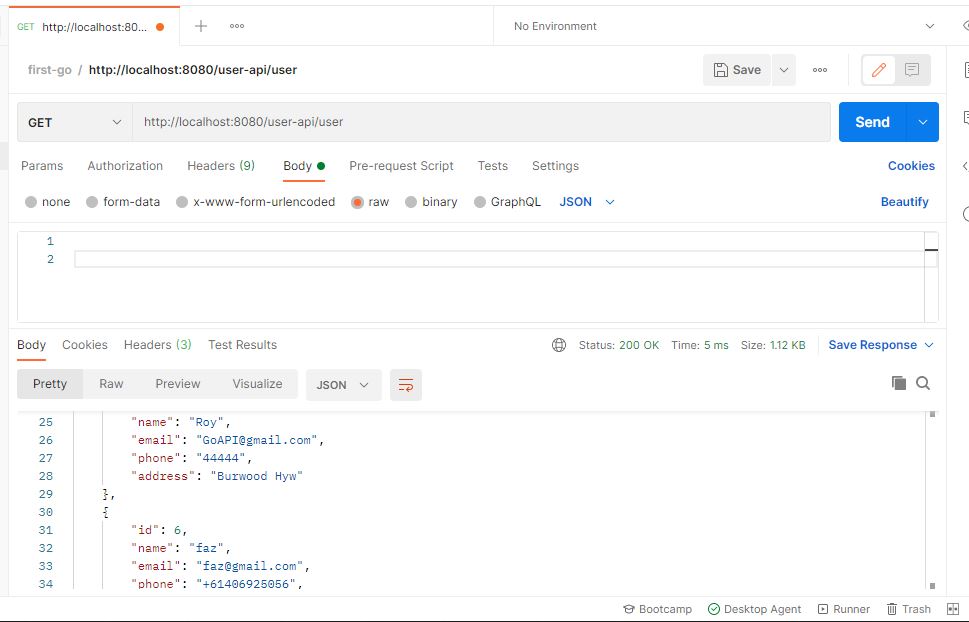
****

Then we try our API using the POSTMAN application by writing the url:

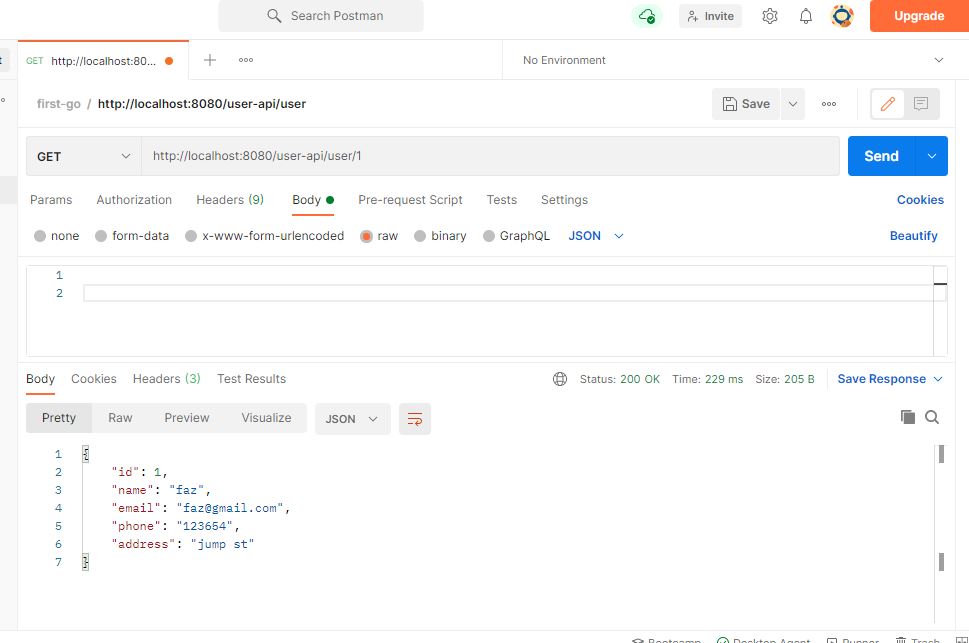
***Add new user data🡪***[***http://localhost:8080/user-api/user***](http://localhost:8080/user-api/user)



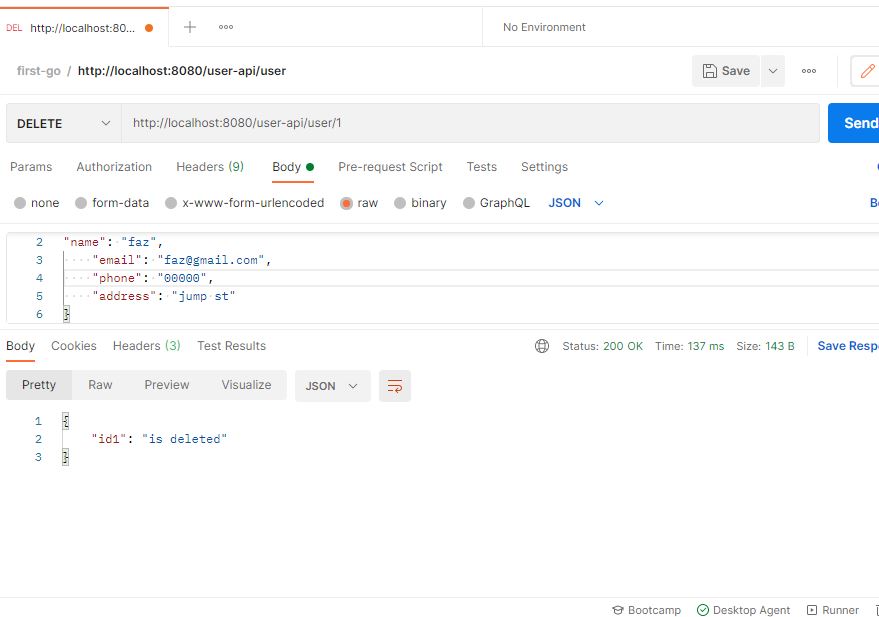
***Get All user data🡪***[***http://localhost:8080/user-api/user***](http://localhost:8080/user-api/user)



***Update specific user data record🡪***<http://localhost:8080/user-api/user>/{id}



***delete user data🡪***<http://localhost:8080/user-api/user>

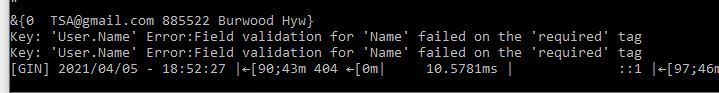


**Errors and validation**

When something goes wrong, our APIs can return more than a single error. They are therefore returned by the client as "error responses" that contain a slice of errors.

It is important to notice that the REST API returns errors with a status code.

*Name validation: Name must required*



*Name must contain only alphabets*



*Email must satisfy basic requirement*****

*Email validation: Email must required*****

**V2 Implementation**

-Phone number validation is not completed.

- Regex for Australian phone number validation (E.164).

**Out of scope:**

**-**Caching to improve performance

-http optimistic locking to prevent inconsistencies in resources.

-E-Tag header

**Documentation**

Please refer test cases and documentation provided with the solution.