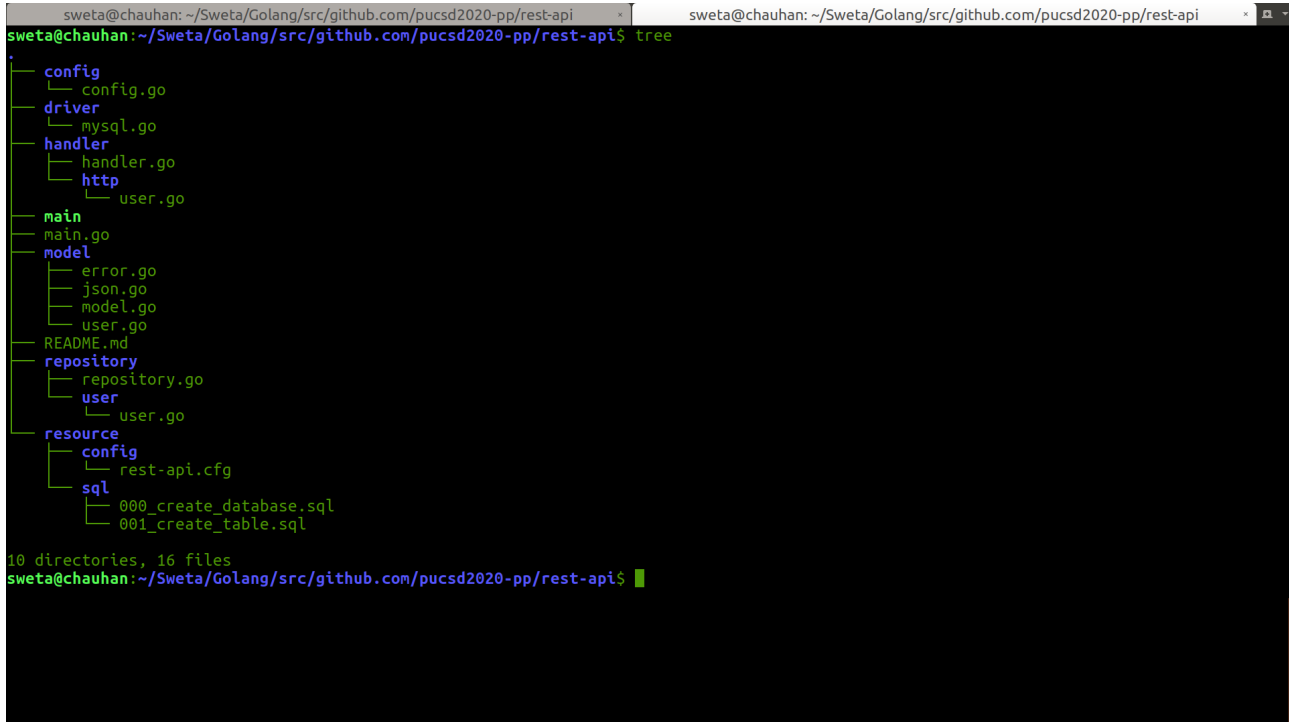


Rest-api

Directory Structure



```
sweta@chauhan: ~/Sweta/Golang/src/github.com/pucsd2020-pp/rest-api$ tree
.
├── config
│   └── config.go
├── driver
│   └── mysql.go
├── handler
│   ├── handler.go
│   ├── http
│   └── user.go
├── main
│   └── main.go
├── model
│   ├── error.go
│   ├── json.go
│   ├── model.go
│   ├── user.go
│   └── README.md
├── repository
│   ├── repository.go
│   └── user
│       └── user.go
└── resource
    ├── config
    │   └── rest-api.cfg
    └── sql
        ├── 000_create_database.sql
        └── 001_create_table.sql

10 directories, 16 files
sweta@chauhan: ~/Sweta/Golang/src/github.com/pucsd2020-pp/rest-api$
```

Module descriptions

1. **config** :- It's an independent module(user's defined). There are one file **config.go**.
2. **driver** :- It's a dependent module(user's defined). It's depend on **model,config**.
3. **handler** :-
 - handler.go** is independent file.
 - Http** :- It's dependent module(user's defined).It depends on **handler, model, repository, repository/user** .
4. **model** :- It's independent module (user's defined). It contains structrue of basic entity with interface.
5. **repository** :-
 - user** :- It's depends on **driver and model**.
 - repository.go** :- Contains struct and interface.
6. **resource** :- **It contains two folder:-**
 1. **sql** :- Sql scripts to create database and table.
 2. **config**:- All basic configuration variables with are mentioned.It is configurable.

Work Flow of Server Setup

- 1) config/config.go/init()
Reading Configuration File
Dumping configuration file values to const _config
- 2) main.go
init()

Logger setup
config.go
 Returning configuration strings
mysql.go
 Come to the database driver connection
config.go
 Now I am getting connection string response
mysql.go
 Checking Idle connection
 Checking max connection
 Setting Connection life line
 return either database driver connection

- 3) main.go
 Passing dbConnection String to Handler
 user.go
 Passing user handler an sql connection to get User struct in
 handler/http/user.User
- 4) main.go
 Setting Up Router by creating object
- 5) main.go
 Now using Recoverer
 Now using Logger
 Setting Up router
 Cases basis http method check
 user.go
 Http Handler Redirecting to object based on path.
 To listen and serve getting configuration values
 config.go
 Returning configuration strings
 config.go
 Returning configuration strings

Work Flow on REST API call

- 6) On GET call
 user.go
 Handler ...
 Creating User object and Parsing
 Getting context and creating model.User object in repository/user module
 mysql.go
 Reading object values from interface type using reflection
 Writing response in JSON
- 7) On GetAll call
 user.go
 Handler ...
 Calling GetAll Function
 Getting context and creating model.User object **repository/user** module
 mysql.go
 Reading object values from interface type using reflection
- 8) On Delete call
 user.go
 Handler ...

Now calling delete with req.context and usr object

user.go

Getting context and creating model.User object **repository/user** module

mysql.go

Soft Deletion String

Delete statement is: %s UPDATE user_detail SET deleted = 1 WHERE id = ?

user.go

Writing response to JSON

9) On Update call

user.go

Handler ...

Update/PUT call

user.go

Creating usr object defined in handler/http

user.go

Creating usr object defined in model

user.go

Decoding request in usr object

user.go

Now calling Update function with req.context and usr object

user.go

Getting context and creating model.User object in repository/user module

driver/mysql.go

Reading object values to update from interface type using reflection

user.go

Now Writing response to json

10) On Create call

user.go

Handler ...

Creating usr object defined in model

user.go

Now calling Create function Create function with req.context and usr object

user.go

Getting context and creating model.User object in repository/user module

mysql.go

Reading object values from interface type using reflection

mysql.go

Creating query string ...

Executing query string ...

Returning result string ...

Structure and Interfaces

model

User

| | |
|---------------|--------|
| Id | int64 |
| FirstName | string |
| LastName | string |
| Email | string |
| Password | string |
| ContactNumber | string |
| UpdatedBy | int64 |

IModel

| | |
|----------|--------|
| Table() | string |
| String() | string |

handler

HTTPHandler

```
Authenticated bool
Method        string
Path          string
Func
func(http.ResponseWriter,
    *http.Request)
```

IHTTPHandler

```
GetHTTPHandler() []*HTTPHandler
GetByID(http.ResponseWriter,
    *http.Request)
Create(http.ResponseWriter,
    *http.Request)
Update(http.ResponseWriter,
    *http.Request)
Delete(http.ResponseWriter,
    *http.Request)
GetAll(http.ResponseWriter,
    *http.Request)
```

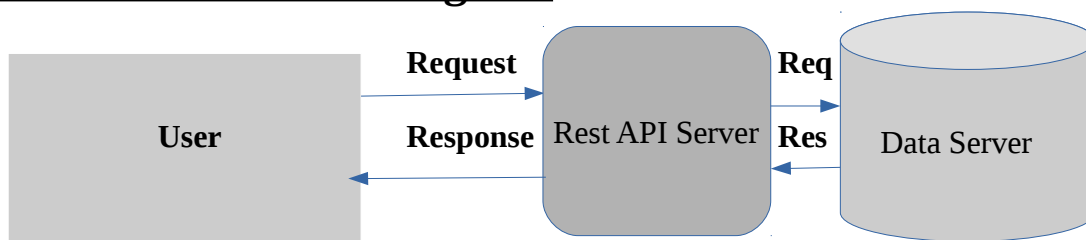
handler/http

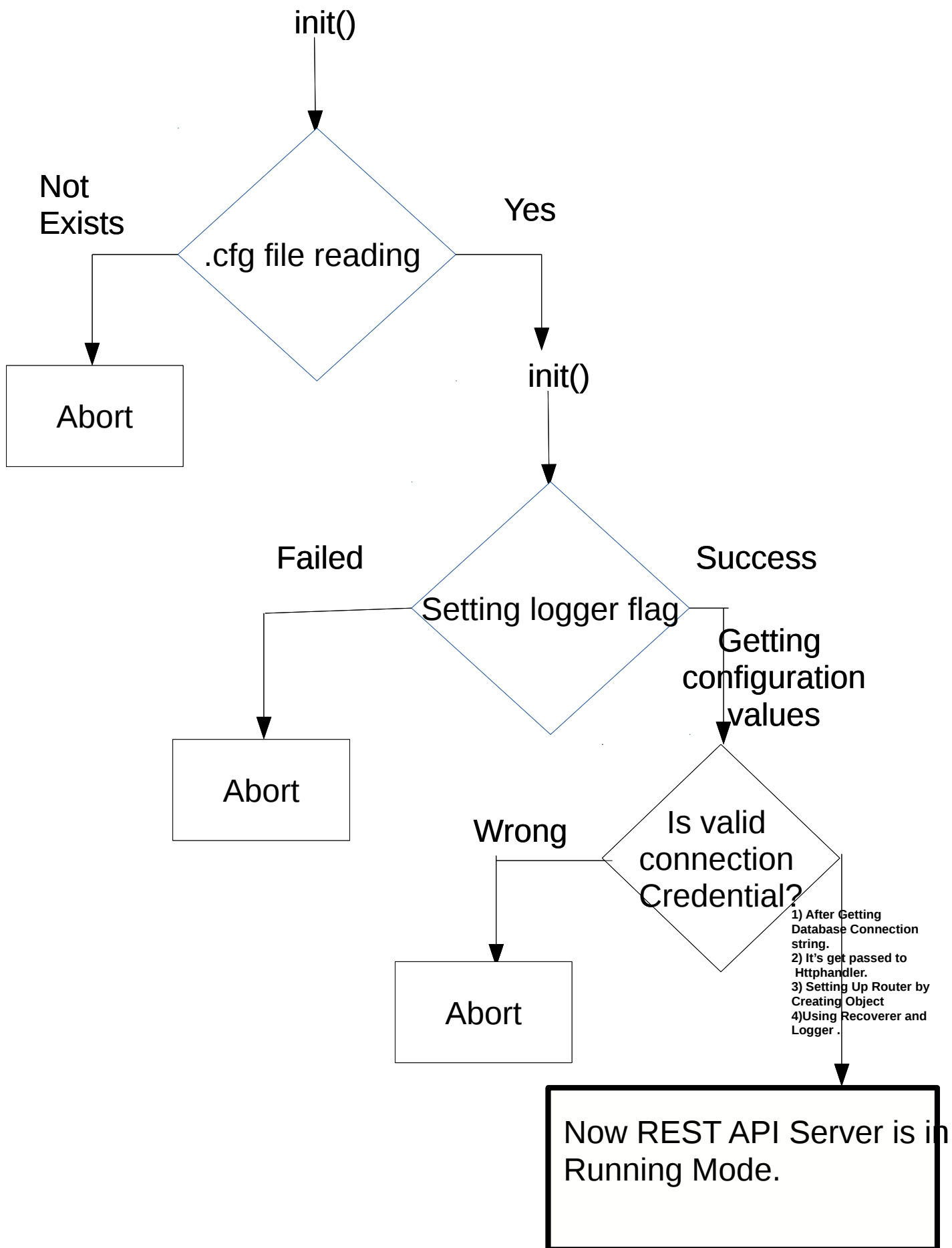
User

```
handler.HTTPHandler
repo repository.IRepository
```

It has implemented IHTTPHandler interface.

Top Level Data Flow Diagram





Data Flow Diagram for end user request processing

