Go and SQL

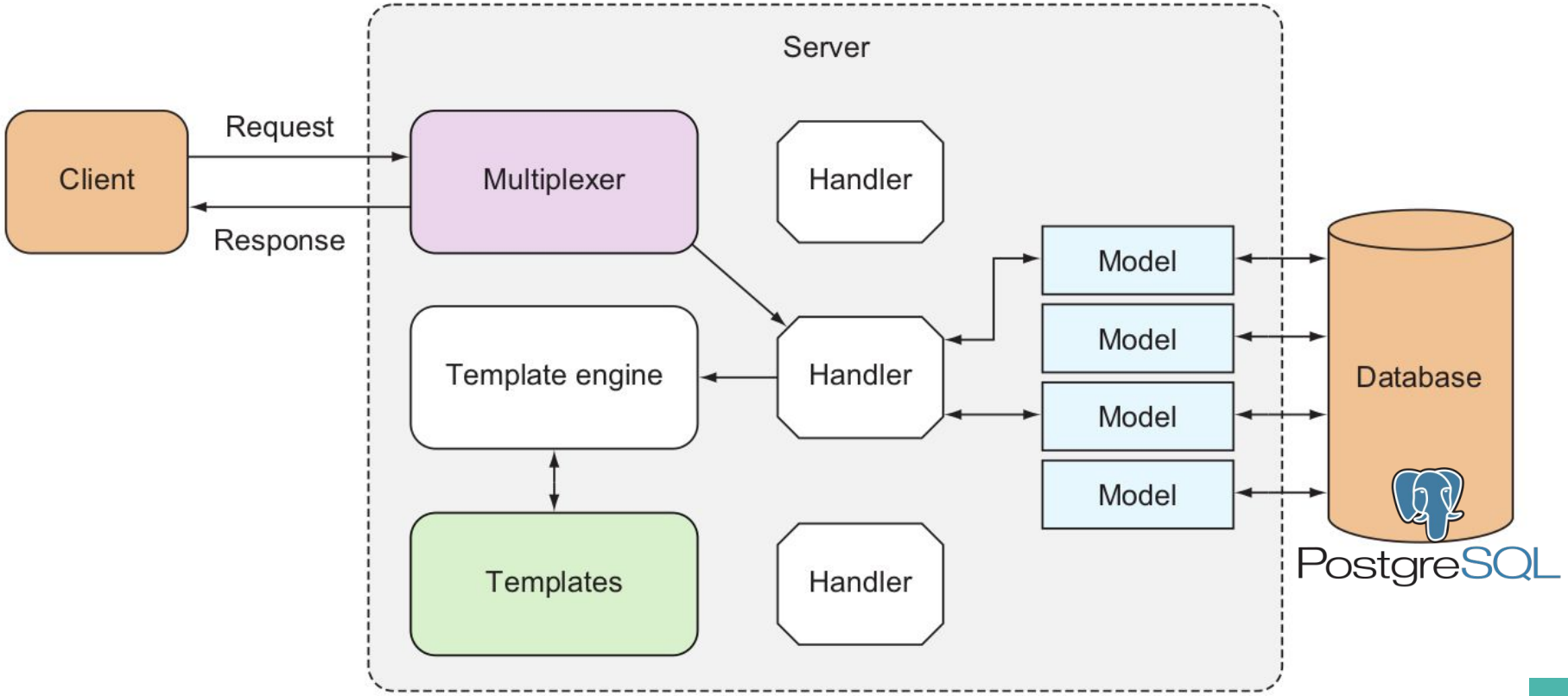
LAB 07

Learning Outcomes

After Completing this lab lesson you should be able to

- Perform CRUD operations on PostgreSQL database
- Access form and URL request values

Go and SQL

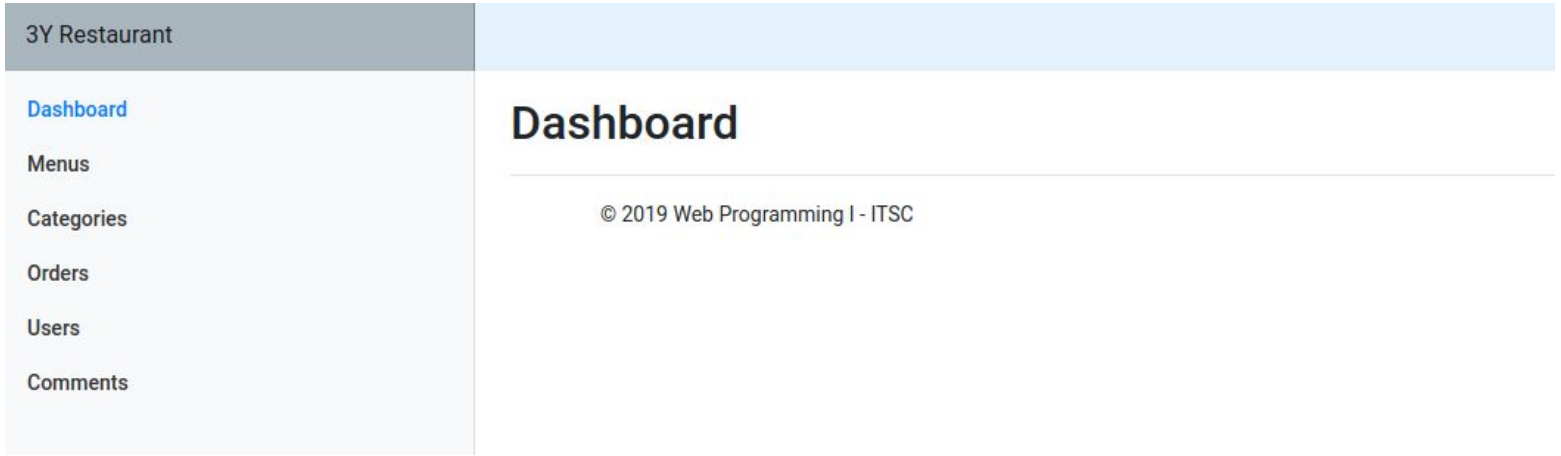

Server
Client
Request
Response
Multiplexer
Handler
Handler
Template engine
Templates
Handler
Model
Model
Model
Model
Database
PostgreSQL

Lab Activities

Download/clone the following code from github

`https://github.com/betsegawlemma/web-prog-go-sample`

Add `User Roles` link on the dashboard in the `admin.navbar.html`



Database Connection

```
dbconn, err := sql.Open("postgres",
"postgres://app_admin:P@$$w0rdD2@localhost/restaurantdb?sslm
ode=disable")

if err != nil {
    panic(err)
}
defer dbconn.Close()

if err := dbconn.Ping(); err != nil {
    panic(err)
}
```

Request paths

The following are request routes that we will implement

- **`/admin/roles`** lists all user roles in the database
- **`/admin/roles/new`** creates new user role
- **`/admin/roles/update?id=1`** updates existing user role with id of 1
- **`/admin/roles/delete?id=4`** deletes existing user role having id of 4

Role entity

Create a file **role.go** inside **entity** directory

Define a Role entity as shown below inside the role.go file

```
1   package entity
2
3   // Role repesents application user roles
4   type Role struct {
5       ID   int
6       Name string
7   }
```

RoleRepository Interface

Inside `menu/repository.go` file, create the `RoleRepository` interface and define the `CRUD` operations

You can use the `CategoryRepository` interface CRUD operations as an example

RoleService Interface

Inside `menu/service.go` file, create the `RoleService` interface and define services similar to `CategoryService` interface

Implement RoleRespository interface

Inside `menu/repository` directory create a file named `psql_role.go`

In this file

- Create a struct, you can name it `RoleRepositoryImpl`, with one field of type `*sql.DB`
- Create a constructor that initialize the struct field and returns a pointer to `RoleRepositoryImpl` struct
- Attach methods to the `RoleRepositoryImpl` struct that implement the operations defined in the `RoleRepository` interface

Implement RoleService interface

Inside `menu/service` directory create a file named `role_service.go`

In this file

- Create a struct, you can name it `RoleServiceImpl`, with one field of type `menu.RoleRepository`
- Create a constructor that initialize the struct field and returns a pointer to `RoleServiceImpl` struct
- Attach methods to the `RoleServiceImpl` struct that implement the operations defined in the `RoleService` interface

Implementation Steps

For the CRUD operations you can start your implementation in the following order

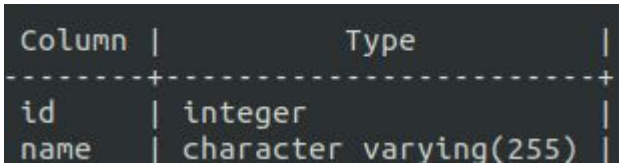
1. Implement the **`store/create`** functionality
2. Implement the **`retrieve all`** functionality
3. Implement the **`retrieve by id`** functionality
4. Implement the **`update`** functionality
5. Implement the **`delete`** functionality

Forms

You can create a form called **`admin.roles.new.html`** inside **`ui/templates`** directory for creating a new role

You can use **`admin.categories.new.html`** as an example

Note that the roles table in the database has two fields as shown below. You only need to have the name field as the id is automatically generated field



```
 Column |          Type          |
--------+------------------------+
 id     | integer                |
 name   | character varying(255) |
```

Forms

You can create a form called **`admin.roles.html`** inside **`ui/templates`** directory for displaying all roles

You can use **`admin.categ.html`** as an example

Handlers

Inside `delivery/http/handler` directory create a file named `admin_role_handler.go`

In this file create a struct, you can name it `AdminRoleHandler` with two fields of type `*template.Template` and `menu.RoleService`

Create a constructor that initialize the two fields and return a pointer to a new `AdminRoleHandler`

As an example you can use `admin_category_handler.go` file

Handlers

Attach methods to the `AdminRoleHandler` struct which handle requests at the following routes

```
/admin/roles

/admin/roles/new

/admin/roles/update?id=1

/admin/roles/delete?id=9
```

As an example you can use the handlers defined inside `admin_role_handler.go` file

Main function

Inside the main function do the following

- Create new instances of the `AdminRoleHandler` using its constructor
- Register the routes to the handler methods defined in the `AdminRoleHandler` struct
- Example can be found inside `main.go` file