25 DE MAYO DE 2021

INVENTARIO EN GO

GRISELDA NAVARRETE DE LA CRUZ

Índice

Introducción

La razón por la que se decidió trabajar con el lenguaje de programación Go fue por lo versátil que llega a ser al poderse implementar en diversas áreas. Es un lenguaje que cuenta con una inmensa variedad de librerías.

Lo quise usar para poder desarrollar un sistema de inventario en el cual se podrían realizar consultas, editar, borrar y agregar datos a una base de datos, la cual estaría en un host, consideramos que sería la mejor forma ya que se tenía en mente que cualquier dispositivo tuviera acceso a dicha aplicación web, en particular los celulares, esto debido a que es más fácil contar con un celular que tenga los componentes necesarios para poder funcionar sin muchas interrupciones, lo cual es algo que siempre se busca en esta clase de trabajos al realizar un inventario, que se pueda llevar a cabo de forma ágil, correcta.

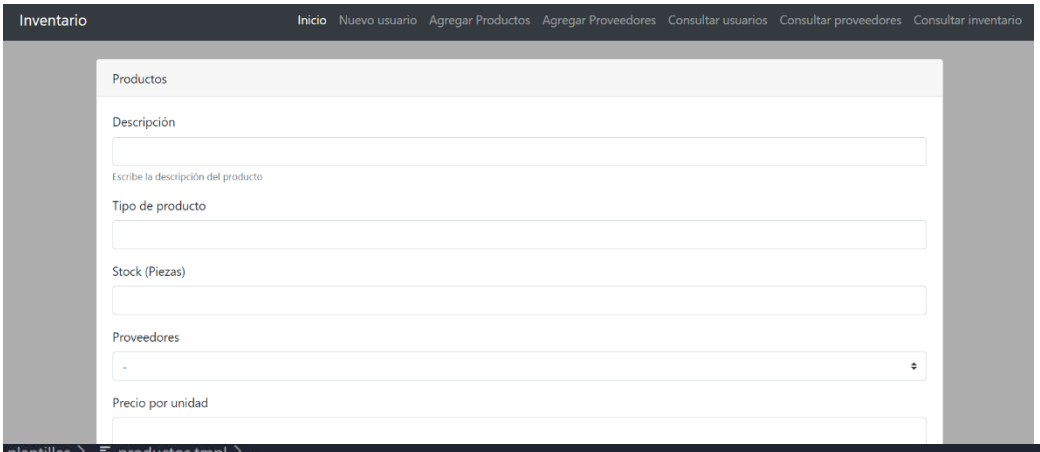
Lo primero para su uso dentro de un restaurante de hamburguesas, ya que el sistema con el cual se contaba era un poco anticuado, hacían uso de hojas de inventario las cuales pueden a veces llegar a presentar complicaciones, como; no tener un correcto cuidado de estas, gastos extras, se estropea si se requiere realizar alguna modificación, no se puede disponer de él en todo momento.

Estructura para guardar la información rescatada de la base de datos

```go
type prod struct {//se hace una estructura en la cual se guardan la información de cada producto
    Id_p            int// su ID
    Descripcion     string//Una breve descripcion
    Tipo_de_producto string//tipo, puede ser lacteos, limpieza
    Stock           int//cantidad existente
    Id_pro          int//id del proveedor
    Precio_unidad   int//precio por cada producto
    Precio_t        int//se estima el valor total de los productos
}
type prove struct {//se hace una estructura en la cual se guardan la información de cada proveedor
    Id_pro    int//id del proveedor
    Proveedor string//nombre del proveedor
}
```

Agregar Productos:

Plantilla de agregar productos en html



plantillas > productos.tmpl > ...

```
<div class="card-body">
    <form  action="/enviar" method="POST">
     <div class="form-group">
       <label for="">Descripción</label>
       <input type="text"
         class="form-control" name="descripcion" id="descripcion" aria-describedby="helpId" placeholder=""
       <small id="helpId" class="form-text text-muted">Escribe la descripción del producto</small>
     </div>

     <div class="form-group">
       <label for="">Tipo de producto</label>
       <input type="text" class="form-control" name="Tipo_de_producto" id="Tipo_de_producto" placeholder="
     </div>
     <div class="form-group">
       <label for="">Stock (Piezas)</label>
       <input type="number" class="form-control" name="stock" id="stock" placeholder="" required>
     </div>
     <div class="form-group">
        <label for="">Proveedores</label>
        <select class="custom-select" name="Id_pro" id="Id_pro" required>
          <option selected>-</option>
          {{range.}}
          <option value="{{.Id_pro}}">{{.Proveedor}}</option>
```

Codigo de GO:

El siguiente codigo muestra en que los primero a realizar al querer ingresar a la pagina de agregar productos es mandar llamar la conexión de la base de datos para poder desplegar en el dropdown de los proveedores registrados en el Sistema, y que el usuario al agregar un nuevo product pueda visualizarlos.

```go
func Productos(w http.ResponseWriter, r *http.Request) { //función que redirige a la pagina de productos
    conexionestablecida := conexionBD() //llama la conexión de la bd
    //hace una consulta para desplegar los proveedores
    mprov, err := conexionestablecida.Query("SELECT * FROM proveedores")
    if err != nil {
        panic(err.Error()) //verifica los errores
    }
    us := prove{} //manda llamar la estructura de proveedores
    arregloprovee := []prove{} //crea un arreglo sobre la estructura
    for mprov.Next() { //se recorre la tabla de proveedores hasta encontrar el id recibido
        var Id_pro int
        var Proveedor string
        err = mprov.Scan(&Id_pro, &Proveedor) //recibe los datos a desplegar
        if err != nil {
            panic(err.Error()) //verifica los errores
        }
        us.Id_pro = Id_pro
        us.Proveedor = Proveedor
        arregloprovee = append(arregloprovee, us) //copia los datos recibido al arreglo

    }
    fmt.Println("Proveedores: ", arregloprovee)
    plantillas.ExecuteTemplate(w, "productos", nil) //manda la consulta a la pagina
```

Función de enviar: recibe los datos ingresados por el usuario y los agrega a la table de productos en la base de datos:

```go
func Enviar(w http.ResponseWriter, r *http.Request) {

    if r.Method == "POST" {
        descripcion := r.FormValue("descripcion")
        Tipo_de_producto := r.FormValue("Tipo_de_producto")
        stock := r.FormValue("stock")
        Id_pro := r.FormValue("Id_pro")
        precio_unidad := r.FormValue("precio_unidad")
        precio_t := r.FormValue("precio_t")

        conexionestablecida := conexionBD()
        insertarproducts, err := conexionestablecida.Prepare("INSERT INTO productos (descripcion, Tipo_d
        if err != nil {
            panic(err.Error())
        }
        insertarproducts.Exec(descripcion, Tipo_de_producto, stock, Id_pro, precio_unidad, precio_t)

        http.Redirect(w, r, "/inventory", 301)
    }
}
```

Consulta de inventario

ID producto	Id proveedor	Descripción	Precio por unidad	Precio total	Stock	Tipo de producto
1	1	sdfs	50	100	8	sdfsd
2	3	crema batida	30	0	2	malteadas
3	2	lechuga	10	30	3	verdura

```go
func Productos(w http.ResponseWriter, r *http.Request) {

    conexionestablecida := conexionBD()//se hace la conexion a la base de datos
    //se establece la clase de consulta a relizar
    mprov, err := conexionestablecida.Query("SELECT * FROM proveedores")
    if err != nil {//en caso de que no se logre hacer la query mostrará un error
        panic(err.Error())
    }
    us := prove{}//variable del tipo de estructura de proveedores
    arregloprovee := []prove{}//arreglo que almacenará los elementos de la estrucutra prove
    for mprov.Next() {//ciclo para guardar los datos obtenidos de la consulta
        var Id_pro int//variables para los distintos datos a extraer
        var Proveedor string

        err = mprov.Scan(&Id_pro, &Proveedor)//obtiene de la base de datos los elementos especifica

        if err != nil {
            panic(err.Error())
        }
        us.Id_pro = Id_pro//se almacenan los datos en la varible del tipo estrucutra de prove
        us.Proveedor = Proveedor

        arregloprovee = append(arregloprovee, us)//se agrega al arreglo los datos de la estrucutra

    }
    fmt.Println("Proveedores: ", arregloprovee)
    plantillas.ExecuteTemplate(w, "productos", arregloprovee)//ejecuta la pagina html

}
```

```
 style="background: #fff; border:1px solid #BDBDBD; border-radius:5px;">

 class="table">
    <thead>
        <tr>
            <th>ID producto</th>//el nombre que llevará cada columna
            <th>Id proveedor</th>
            <th>Descripción</th>
            <th>Precio por unidad</th>
            <th>Precio total</th>
            <th>Stock</th>
            <th>Tipo de producto</th>
            <th>Acciones</th>
        </tr>
    </thead>
    <tbody>
{{ range.}}//se habre función por la cual se extraen los datos que requerimos de la tabla en la base de datos
        <tr>
            <td>  {{.Id_p}} </td>//id del producto
            <td id="celda +'{{.Id_pro}}'"><text id="color"+"'{{.Id_pro}}'"> {{.Id_pro}}</text></td>//id del prov
            <td>   {{.Descripcion}}//descripcion del producto
            <td>   {{.Precio_unidad}}</td>//precio de cada producto
                    <td>   {{.Precio_t}}</td>//el precio total de los productos
            <td>   {{.Stock}}</td>//cantidad disponible
            <td>   {{.Tipo_de_producto}}</td>
            //botones para poder editar o borrar los productos
            <td> <a name="" id="" class="btn btn-warning" href="/editar3?id_p={{.Id_p}}" role="button">Editar</a
        </tr>
{{end}}
    </tbody>
```

Borrar / editar inventario

ID producto	Id proveedor	Descripción	Precio por unidad	Precio total	Stock	Tipo de producto	Acciones
1	1	sdfs//descripcion del producto	50	100	8	sdfsd	Editar | Borrar
2	3	crema batida//descripcion del producto	30	0	2	malteadas	Editar | Borrar
3	2	lechuga//descripcion del producto	10	30	3	verdura	Editar | Borrar

```
  style="background: #fff; border:1px solid #BDBDBD; border-radius:5px;">

  class="table">
     <thead>
         <tr>
             <th>ID producto</th>//el nombre que llevará cada columna
             <th>Id proveedor</th>
             <th>Descripción</th>
             <th>Precio por unidad</th>
             <th>Precio total</th>
             <th>Stock</th>
             <th>Tipo de producto</th>
             <th>Acciones</th>
         </tr>
     </thead>
     <tbody>
 {{ range.}}//se habre función por la cual se extraen los datos que requerimos de la tabla en la base de datos
         <tr>
             <td>  {{.Id_p}} </td>//id del producto
             <td id="celda +'{{.Id_pro}}'"><text id="color"+"'{{.Id_pro}}'"> {{.Id_pro}}</text></td>//id del prov
             <td>   {{.Descripcion}}//descripcion del producto
             <td>   {{.Precio_unidad}}</td>//precio de cada producto
                    <td>   {{.Precio_t}}</td>//el precio total de los productos
             <td>   {{.Stock}}</td>//cantidad disponible
             <td>   {{.Tipo_de_producto}}</td>
             //botones para poder editar o borrar los productos
             <td> <a name="" id="" class="btn btn-warning" href="/editar3?id_p={{.Id_p}}" role="button">Editar</a
         </tr>
 {{end}}
     </tbody>
 </table>
```

```
func Borrar3(w http.ResponseWriter, r *http.Request) {//funcion para borrar algun producto
    id_p := r.URL.Query().Get("id_p")//del formulario se obtiene el dato que se ingresó en caja de texto id_p
    fmt.Println("id: ", id_p)

    conexionestablecida := conexionBD()//se establece la conexion a la base de datos
    //se realiza la consulta a la base de datos y se especifica la tabla, se ejecuta el comando delete
    borrarregistros, err := conexionestablecida.Prepare("DELETE FROM productos WHERE id_p=?")
    if err != nil {
        panic(err.Error())
    }
    borrarregistros.Exec(id_p)//se indica el usario que se eliminará deacuerdo a su id

    http.Redirect(w, r, "/inventory", 301)//al terminar regresa al menu principal

}
```

Agregar nuevo producto

```go
func Enviar(w http.ResponseWriter, r *http.Request) {

    if r.Method == "POST" {
        descripcion := r.FormValue("descripcion")//recibe lo que se ingresó en la caja de texto descripcion
        //lo mismo para las siguientes lineas
        Tipo_de_producto := r.FormValue("Tipo_de_producto")
        stock := r.FormValue("stock")
        Id_pro := r.FormValue("Id_pro")
        precio_unidad := r.FormValue("precio_unidad")
        precio_t := r.FormValue("precio_t")

        conexionestablecida := conexionBD()//se establece la conexion ala base de datos
        //se indica lo que se va a insertar en la tabla
        insertarproducts, err := conexionestablecida.Prepare("INSERT INTO productos (descripcion, Tipo_de_produc
        if err != nil {//en caso de que haya un error al hacer la edicion
            panic(err.Error())
        }
        //se ingresa el nuevo producto a la tabla
        insertarproducts.Exec(descripcion, Tipo_de_producto, stock, Id_pro, precio_unidad, precio_t)

        http.Redirect(w, r, "/inventory", 301)//devuelve a la pagina principal
    }
}
```

Consultar, editar y borrar proveedores

Permite visualizar los proveedores existentes en la base de datos y realizar en ellos ya sea un cambio o eliminarlos

```go
//para  visualizar,editar y elimainar los datos de la tabla de proveedores
func proveedores(w http.ResponseWriter, r *http.Request) {
    conexionestablecida := conexionBD()//se hace la conexion a la base de datos
    //se establece la clase de consulta a relizar
    registros, err := conexionestablecida.Query("SELECT * FROM proveedores")
    if err != nil {//en caso de que no se logre hacer la query mostrará un error
        panic(err.Error())
    }
    us := prove{}//variable del tipo de estructura de proveedores
    arregloprovee := []prove{}//arreglo que almacenará los elementos de la estrucutra prove
    for registros.Next() {//ciclo para guardar los datos obtenidos de la consulta
        var Id_pro int//variables para los distintos datos a extraer
        var Proveedor string

        err = registros.Scan(&Id_pro, &Proveedor)//obtiene de la base de datos los elementos especificados

        if err != nil {
            panic(err.Error())
        }
        us.Id_pro = Id_pro//se almacenan los datos en la varible del tipo estrucutra de porve
        us.Proveedor = Proveedor

        arregloprovee = append(arregloprovee, us)//se agrega al arreglo los datos de la estrucutra

    }
    //fmt.Println(arreglousers)
    plantillas.ExecuteTemplate(w, "proveedores", arregloprovee)//ejecuta la pagina html
}
```


```
    <br/>
    <br/><center>
    <div style="background: #fff; border:1px solid #BDBDBD; border-radius:5px;">
        <table class="table" >
            <thead>
                <tr>
                //el nombre que llevará cada columna
                    <th>ID</th>
                    <th>Proveedor</th>
                    <th>Acciones</th>
                </tr>
            </thead>
            <tbody>
            //se habre función por la cual se extraen los datos que requerimos de la tabla en la base de datos
        {{ range.}}
                <tr>
                    <td>   {{.Id_pro }} </td>//id del proveedor
                    <td>  {{.Proveedor}} </td>//nombre del proveedor
                     //botones para poder editar o borrar el proveedor
                     //se envia el id del proveedor a la plantilla editar
                    <td> <a name="" id="" class="btn btn-warning" href="/editar2?id_pro={{.Id_pro}}" role="butto
                     | <a name="" id="" class="btn btn-danger" href="/borrarProveedor?id_pro={{.Id_pro}}" role="b
                </tr>
        {{end}}
            </tbody>
        </table>  </center>
    </div>
{{template "pie"}}
{{end}}
```

Plantilla editar recibe el id del proveedor en un formulario en el cual se ingresan los datos nuevos.

```go
//recibe los datos que se ingresaron en el formulario para agregar nuevo proveedor
func Enviar2(w http.ResponseWriter, r *http.Request) {
    Proveedor := r.FormValue("Proveedor")//recibe lo que se ingresó en la caja caja de texto proveedor

    conexionestablecida := conexionBD()//se establece la conexion ala base de datos
    insertarprovedores, err := conexionestablecida.Prepare("INSERT INTO proveedores (Proveedor) VALUES (?)")
    if err != nil {//en caso de que haya un error al hacer la edicion
        panic(err.Error())
    }
    insertarprovedores.Exec(Proveedor)//se ingresa el nuevo proveedor

    http.Redirect(w, r, "/", 301)//devuelve a la pagina principal
}
func Admin(w http.ResponseWriter, r *http.Request) {
    //fmt.Fprintf(w, "Holi") //mensaje en el navegador
    plantillas.ExecuteTemplate(w, "login", nil)
}
//para  visualizar,editar y elimainar los datos de la tabla de productos
```

```
   Proveedores
</div>
<div class="card-body">
   <form  action="/actualizar2" method="POST">//formulario para modificar los datos del proveedor especificado
   <div class="form-group">
   //se muestra el id del proveedor a editar, no se puede editar el id
      <label for="">Id Proveedor</label>

      <label class="sr-only" for="inputName">ID</label>
      <input type="text" value={{.Id_pro}} class="form-control" name="id_pro" id="id_pro" placeholder="">
    </div>
    <div class="form-group">
      <label for="">Proveedor</label>
      //se inserta el dato a editar en el nombre, en todo caso se mantiene igual
      <input type="text"
        class="form-control" name="proveedor" value={{.Proveedor}} id="proveedor" aria-describedby="helpId" pl
      <small id="helpId" class="form-text text-muted">Escribe proveedor</small>
    </div>


    <button type="submit" class="btn btn-success">Modificar</button>
    <a name="" id="" class="btn btn-primary" href="/" role="button">Cancelar</a>
    </form>
iv>
```

Una vez llenos los datos en el formulario la función actualizar2

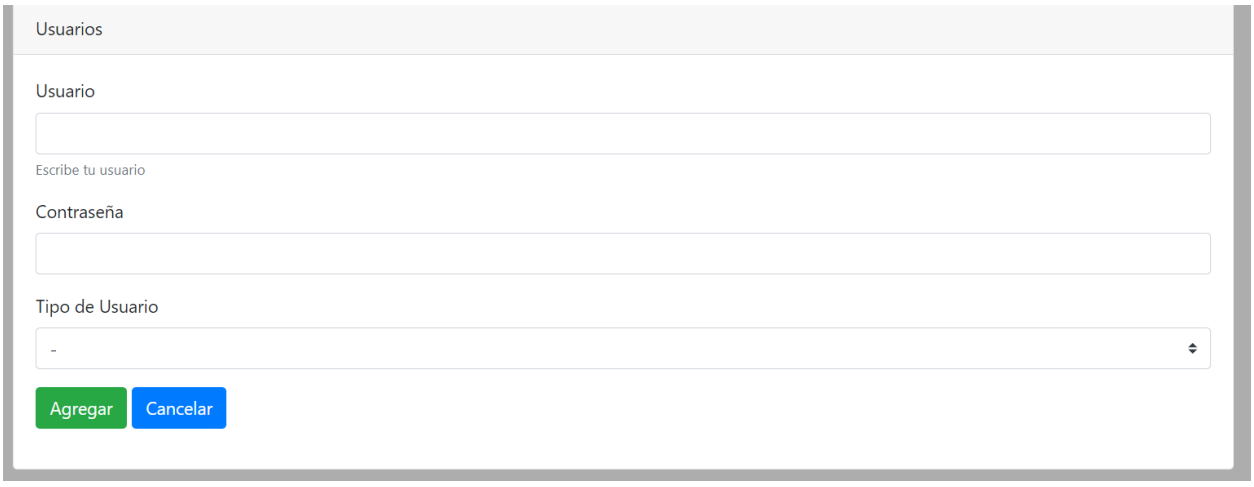
```go
Actualizar2(w http.ResponseWriter, r *http.Request) {
if r.Method == "POST" {//en caso de haber enviado un formulario se ejecuta
    id_pro := r.FormValue("id_pro")//recibe el dato en id_pro
    proveedor := r.FormValue("proveedor")
    fmt.Print(id_pro)
    conexionestablecida := conexionBD()//establece la conexion en la base de datos
    //se define la variable con el tipo de consulta a realizar
    modificarregistros, err := conexionestablecida.Prepare("UPDATE proveedores SET Proveedor=? WHERE id_pro=? "
    if err != nil {
        panic(err.Error())
    }
    modificarregistros.Exec(proveedor, id_pro)//se ejecuta la consulta y las variables que se van a acutalizar

    http.Redirect(w, r, "/", 301)//devueve a la pagina principal
}
```

Al elegir borrar algún proveedor se ejecuta la función borrar Proveedor

```go
func borrarProveedor(w http.ResponseWriter, r *http.Request) { //funcion que permite borrar proveedor
    id_pro := r.URL.Query().Get("id_pro") //del formulario se obtiene el dato que se ingresó en caja de texto id
    fmt.Println("kkk", id_pro)

    conexionestablecida := conexionBD() //se establece la conexion a la base de datos
    borrarregistros, err := conexionestablecida.Prepare("DELETE FROM proveedores WHERE id_pro=?")
    //se realiza la consulta a la base de datos y se especifica la tabla, se ejecuta el comando delete
    if err != nil {
        panic(err.Error())
    }
    borrarregistros.Exec(id_pro) //se indica el usuario que se eliminará deacuerdo a su id

    http.Redirect(w, r, "/", 301) //al terminar regresa al menu principal

}
```

Crear usuarios:

Plantilla de html



```
plantillas > crear.tmpl > ...
    </div>
    <div class="card-body">
        <form  action="/insertar" method="POST">
        <div class="form-group">
            <label for="">Usuario</label>
            <input type="text"
            class="form-control" name="usuario" id="usuario" aria-describedby="helpId" placeholder="">
            <small id="helpId" class="form-text text-muted">Escribe tu usuario</small>
        </div>
        <div class="form-group">
            <label for="">Contraseña</label>
            <input type="password" class="form-control" name="pass" id="pass" placeholder="">
        </div>
        <div class="form-group">
            <label for="">Tipo de Usuario</label>
            <select class="custom-select" name="tipo_u" id="tipo_u">
                <option selected>-</option>
                <option value="admin">admin</option>
                <option value="empleado">empleado</option>
            </select>
        </div>
        <button type="submit" class="btn btn-success">Agregar</button>
        <a name="" id="" class="btn btn-primary" href="/" role="button">Cancelar</a>
```

Codigo en GO:

Se usan tres funciones para completer la acción de agregar un usuario.

La primera es la de ingresar a la pagina para agregar un nuevo usuario

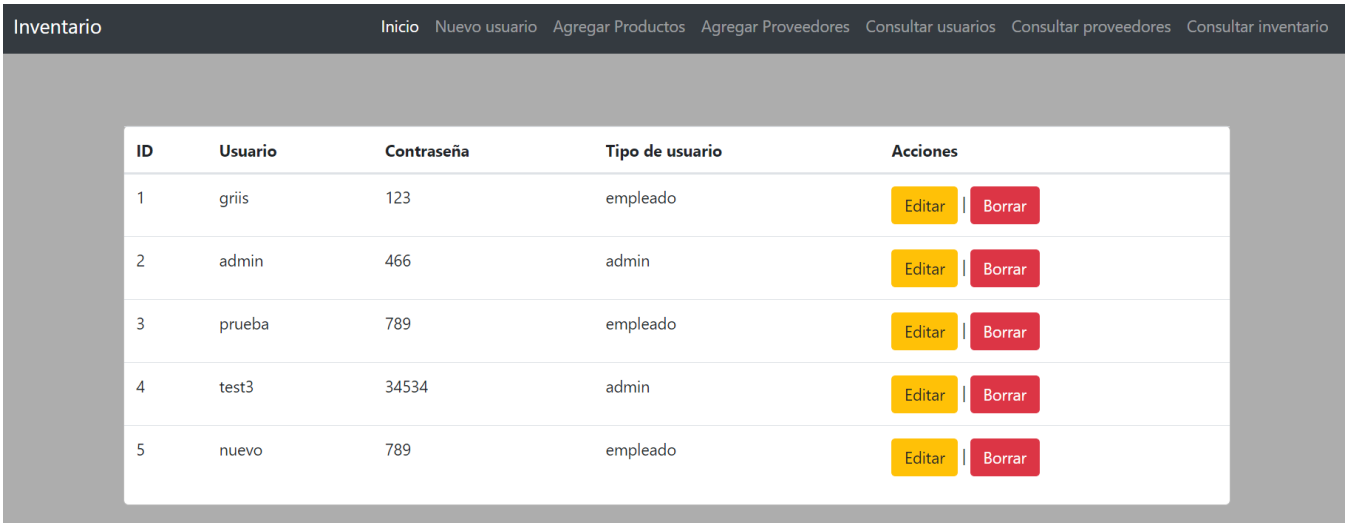
```go
func Crear(w http.ResponseWriter, r *http.Request) { //función que abre la pagina para crear un usuario
    plantillas.ExecuteTemplate(w, "crear", nil)
}
```

Despues de recibir los datos se manda llamar a otra función que recibe los datos para ingresarlos a la base de datos en la tabla de usuarios

```go
func Insertar(w http.ResponseWriter, r *http.Request) { //función que recibe los datos del formulario de nuevo us
    if r.Method == "POST" { //se verifica si los datos se mandaron por el metodo POST
        //recibe los datos
        usuario := r.FormValue("usuario")
        pass := r.FormValue("pass")
        tipo_u := r.FormValue("tipo_u")

        conexionestablecida := conexionBD() //se manda llamar la conexion
        // se hace la consulta para agregar al usuario
        insertarregistros, err := conexionestablecida.Prepare("INSERT INTO usuarios(usuario,pass,tipo_u) VALUES (
        if err != nil { //se verifica que no haya errores
            panic(err.Error())
        }
        insertarregistros.Exec(usuario, pass, tipo_u) //se mandan los datos recibidos por parametros a la bd

        http.Redirect(w, r, "/", 301) //despues de la consulta se redireciona a la pagina principal
    }
}
```

Consultar, editar y borrar usuarios:

Permite visualizar los usuarios registrados, y hacer modificaciones y borrar registros.



Se utiliza la propieda range para definer un rango en la tabla en la plantilla de html

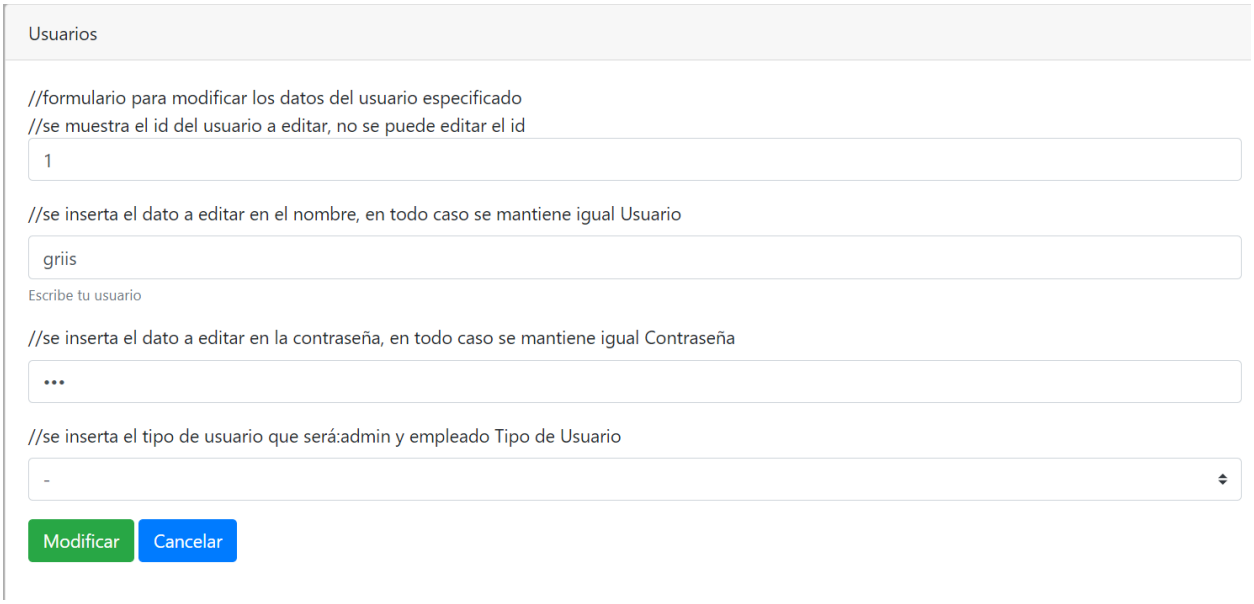
```
{{define "ConsulUsuarios"}}
{{template "cabecera"}}
    <br/>
    <br/>
        <table class="table">
            <thead>
                <tr>
                    <th>ID</th>
                    <th>Usuario</th>
                    <th>Contraseña</th>
                    <th>Tipo de usuario</th>
                </tr>
            </thead>
            <tbody>
            {{ range.}}
                <tr>
                    <td>   {{.Id_u }} </td>
                    <td>  {{.Usuario}} </td>
                    <td>  {{.Pass}}</td>
                    <td>   {{.Tipo_u}}</td>
                </tr>
        {{end}}
            </tbody>
        </table>
```

Codigo de GO:

```go
func ConsulUsuarios(w http.ResponseWriter, r *http.Request) {
    conexionestablecida := conexionBD()
    registros, err := conexionestablecida.Query("SELECT * FROM usuarios")
    if err != nil {
        panic(err.Error())
    }
    us := users{}
    arreglousers := []users{}
    for registros.Next() {
        var Id_u int
        var Usuario, Pass, Tipo_u string
        err = registros.Scan(&Id_u, &Usuario, &Pass, &Tipo_u)
        if err != nil {
            panic(err.Error())
        }
        us.Id_u = Id_u
        us.Usuario = Usuario
        us.Pass = Pass
        us.Tipo_u = Tipo_u
        arreglousers = append(arreglousers, us)
    }
    //fmt.Println(arreglousers)
    plantillas.ExecuteTemplate(w, "ConsulUsuarios", arreglousers)
}
```

Botones – Editar

Muestra una Ventana con los datos del usuario a editar



```
        </div>
        <div class="card-body">
            <form  action="/actualizar" method="POST">//formulario para modificar los datos del usuario especifica
            <div class="form-group">
                <label class="sr-only" for="inputName">ID</label>
                //se muestra el id del usuario a editar, no se puede editar el id
                <input type="text" value={{.Id_u}} class="form-control" name="id_u" id="id_u" placeholder="">
             </div>
             <div class="form-group">
             //se inserta el dato a editar en el nombre, en todo caso se mantiene igual
              <label for="">Usuario</label>
              <input type="text"
                class="form-control" name="usuario" value={{.Usuario}} id="usuario" aria-describedby="helpId" pla
              <small id="helpId" class="form-text text-muted">Escribe tu usuario</small>
             </div>
             <div class="form-group">
             //se inserta el dato a editar en la contraseña, en todo caso se mantiene igual
              <label for="">Contraseña</label>
              <input type="password" class="form-control" name="pass" value={{.Pass}}   id="pass" placeholder="">
             </div>
             <div class="form-group">
             //se inserta el tipo de usuario que será:admin y empleado
                <label for="">Tipo de Usuario</label>
                <select class="custom-select" name="tipo_u" value={{.Tipo_u}}  id="tipo_u">
```

Codigo en GO:

```go
func ConsulUsuarios(w http.ResponseWriter, r *http.Request) {
    conexionestablecida := conexionBD()
    registros, err := conexionestablecida.Query("SELECT * FROM usuarios")
    if err != nil {
        panic(err.Error())
    }
    us := users{}
    arreglousers := []users{}
    for registros.Next() {
        var Id_u int
        var Usuario, Pass, Tipo_u string
        err = registros.Scan(&Id_u, &Usuario, &Pass, &Tipo_u)
        if err != nil {
            panic(err.Error())
        }
        us.Id_u = Id_u
        us.Usuario = Usuario
        us.Pass = Pass
        us.Tipo_u = Tipo_u
        arreglousers = append(arreglousers, us)
    }
    //fmt.Println(arreglousers)
    plantillas.ExecuteTemplate(w, "ConsulUsuarios", arreglousers)
}
```

Boton de Borrar

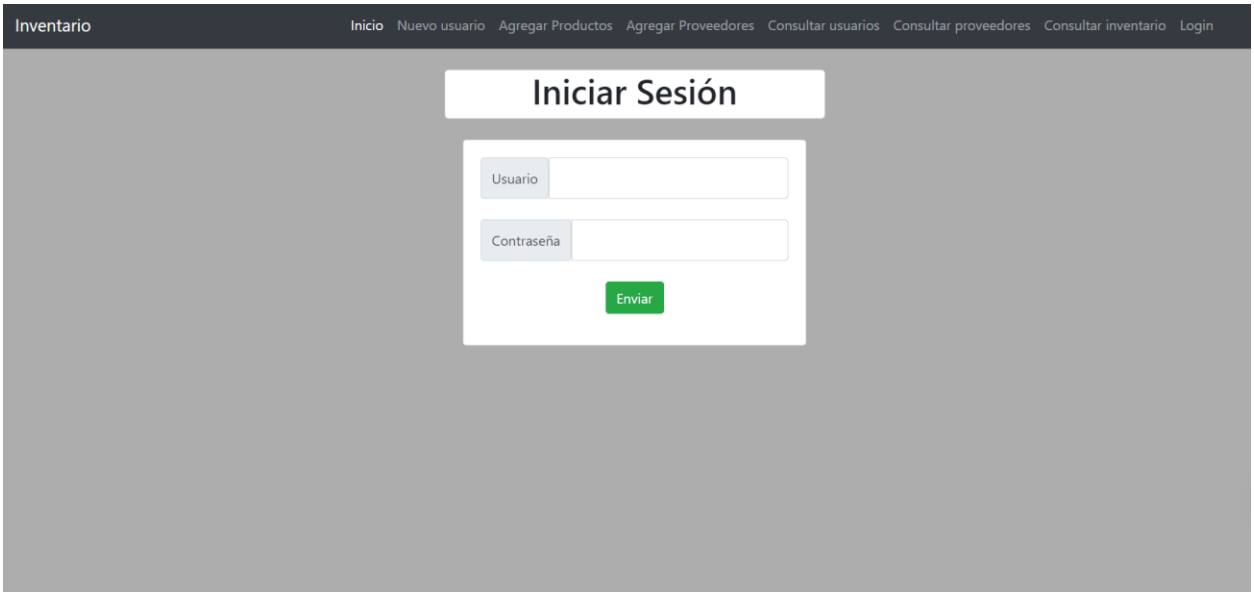
```
| <a name="" id="" class="btn btn-danger" href="/borrar?id_u={{.Id_u}}" role="button">Borrar</a>  </td>
```

Codigo de GO:

Se recibe el id a eliminar por parametro

```go
func Borrar(w http.ResponseWriter, r *http.Request) { //funcion que permite borrar a cierto usaurio
    //fmt.Println(Id_u)
    Id_u := r.URL.Query().Get("id_u")     //del formulario se obtiene el dato que se ingresó en caja de texto id
    conexionestablecida := conexionBD()          // se establece la conexion a la base de datos
    borrarregistros, err := conexionestablecida.Prepare("DELETE FROM usuarios WHERE Id_u=?") //se realiza la con
    if err != nil {
        panic(err.Error())
    }
    borrarregistros.Exec(Id_u) //se indica el usario que se eliminará deacuerdo a su id

    http.Redirect(w, r, "/", 301) //al terminar regresa al menu principal

}
```

Login




```go
func Login(w http.ResponseWriter, r *http.Request) {

    plantillas.ExecuteTemplate(w, "login", nil)
}
```

```
{{ define "login"}}
{{ template "cabecera"}}
<center>
        <div style="background: #fff; border:1px solid #BDBDBD; border-radius:5px; width: 40%;">
         <center><H1>Iniciar Sesión</H1></center>
        </div><br>
        <div class="card" style="background: #fff; width:25rem;">

          <div class="card-body">

           <form action="/enviar3" method="POST">
            <div class="input-group input-group-lg">
              <span class="input-group-text" id="inputGroup-sizing-lg">Usuario</span>
              <input name="usuario" id="usuario"  type="text" class="form-control" aria-label="Sizing example input" aria-describedby="inpu
            </div>
            <br>
            <div class="input-group input-group-lg">
              <span class="input-group-text" id="inputGroup-sizing-lg">Contraseña</span>
              <input name="pass" id="pass" type="password" class="form-control" aria-label="Sizing example input" aria-describedby="inputGr
            </div>
            <br>
            <button type="submit" class="btn btn-success">Enviar</button>

           </form>
          </div>
        </div></center>
{{template "pie"}}
{{end}}
```