



MANUAL TÉCNICO

RUDY AARÓN GOPAL MARROQUÍN GARCIA

201903872

Universidad de San Carlos de Guatemala

Facultad de Ingeniería

Guatemala

2021

Requerimientos mínimos del sistema

Para el uso correcto de la aplicación es necesario los siguientes componentes y programas:

- Memoria ram de 2gb o superior
- Espacio disponible en disco de 2 gb
- react version 17.0.2
- Nodejs 6.14.7
- GO 13.0.8
- Oracle 18c
- Sistema operativo LINUX UBUNTU 20.04

Paradigma utilizado en el programa.

Para la creación de la aplicación se utilizó un paradigma orientado a objetos debido a que por el tipo de funcionamiento que se requería era el más óptimo para la realización de la aplicación, las clases utilizadas en la aplicación fueron necesarias para crear contenedores de datos específicos para su manejo de una manera más sencilla y óptima, todos los datos de la base de datos poseen su propia estructura dentro de la aplicación para su manipulación en la vía web.

Explicación del API.

Para la programación del api fue utilizado golang el cual monta un servidor en el puerto 7000, las clases utilizadas son las siguientes

insert_CargaMasiva	Esta clase se encarga de de llenar todas las tablas por medio de la carga masiva, devuelve un valor 1 si es incorrecta la operacion y un valor 0 si es correcta
Loguear	Esta clase es utilizada para loguear a los usuarios, devuelve 2 respuestas integer la primera es el valor de retorno del resultado del select del usuario siendo 1 para error y 0 para correcto, y la 2da casilla es si el usuario es un administrador
USUARIO	Estructura que almacena los datos de un Usuario.
RESPUESTA	Estructura utilizada para valores de retorno múltiples
VerificarUsuario	Esta función se encarga de registrar y verificar a los usuarios, devuelve un valor 0 si es correcto y 1 si es incorrecto para el cifrado de las contraseñas fue utilizado un trigger
main	metodo principal el cual posee las rutas que activan el resto de funciones del api.

Estruct de carga masiva

```
type CargaMasiva struct {  
  
    NOMBRE string `json:"NOMBRE"`  
  
    APELLIDO string `json:"APELLIDO"`  
  
    PASSWORD string `json:"PASSWORD"`  
  
    USERNAME string `json:"USERNAME"`  
  
    TEMPORADA string `json:"TEMPORADA"`  
  
    TIER string `json:"TIER"`  
  
    JORNADA string `json:"JORNADA"`  
  
    DEPORTE string `json:"DEPORTE"`  
  
    FECHA string `json:"FECHA"`  
  
    EQUIPO_V string `json:"EQUIPO_V"`  
  
    EQUIPO_L string `json:"EQUIPO_L"`  
  
    PREDICCION_L int `json:"PREDICCION_L"`  
  
    PREDICCION_V int `json:"PREDICCION_V"`  
  
    RESULTADO_V int `json:"RESULTADO_V"`  
  
    RESULTADO_L int `json:"RESULTADO_L"`  
}
```

insert_CargaMasiva

```
func insert_CargaMasiva(w http.ResponseWriter, r *http.Request) {
    fmt.Print(r)

    temps := make([]CargaMasiva, 0)
    reqBody, _ := ioutil.ReadAll(r.Body)
    json.Unmarshal(reqBody, &temps)

    db, err := sql.Open("oci8", "AARON/Marroquin1@localhost:1521/ORCL18")
    if err != nil {
        fmt.Println(err)
        return
    }

    defer db.Close()

    base = db

    for _, valor := range temps {
        s := strings.Split(valor.TEMPORADA, "-Q")
        temp, _ := strconv.ParseInt(s[1], 10, 64)
        temp2, _ := strconv.ParseInt(s[0], 10, 64)

        _, err := db.Exec("BEGIN procedimiento_temporal (:1, :2,
:3,:4,:5,:6,:7,:8,:9,:10,:11,:12,:13,:14,:15,:16,:17);end;",
valor.NOMBRE, valor.APELLIDO, valor.PASSWORD, valor.USERNAME,
valor.TEMPORADA, temp, temp2, valor.TIER, valor.JORNADA, valor.DEPORTE,
valor.FECHA, valor.EQUIPO_V, valor.EQUIPO_L, valor.PREDICCION_L,
valor.PREDICCION_V, valor.RESULTADO_V, valor.RESULTADO_L)

        if err != nil {
            fmt.Println(err)
            return
        }
    }

    println()
    println()
    println("*****CARGA TERMINADA*****")
}
```

Estruct USUARIO

```
type USUARIO struct {  
  
    USERNAME      string `json:"USERNAME"`  
  
    PASS          string `json:"PASS"`  
  
    NOMBRE        string `json:"NOMBRE"`  
  
    APELLIDO       string `json:"APELLIDO"`  
  
    ID_TIER        int     `json:"ID_TIER"`  
  
    FECHA_NACIMIENTO string `json:"FECHA_NACIMIENTO"`  
  
    CORREO         string `json:"CORREO"`  
  
    FOTOPERFIL     string `json:"FOTOPERFIL"`  
  
}
```

Estruct Respuesta

```
type RESPUESTA struct {  
  
    RESPUESTA  int `json:"RESPUESTA"`  
  
    RESPUESTA1 int `json:"RESPUESTA1"`  
  
    RESPUESTA2 int `json:"RESPUESTA2"`  
  
}
```

Logear

```
func Logear(w http.ResponseWriter, r *http.Request)
    json.NewEncoder(w).Encode(Usuarios)
    reqBody, _ := ioutil.ReadAll(r.Body)
    json.NewEncoder(w).Encode(reqBody)
    var user USUARIO
    var h RESPUESTA
    json.Unmarshal(reqBody, &user)
    db, err := sql.Open("oci8", "AARON/Marroquin1@localhost:1521/ORCL18")
    if err != nil {
        fmt.Println(err)
        return
    }
    defer db.Close()
    base = db
    var respuesta int
    var Admin int
    _, err = db.Exec("BEGIN LOGIN (:1, :2, :3,:4);end;", user.USERNAME,
user.PASS, sql.Out{Dest: respuesta}, sql.Out{Dest: Admin})
    if err != nil {
        fmt.Println(err)
        return
    }
    h.RESPUESTA = respuesta
    h.RESPUESTA2 = Admin
    h.RESPUESTA2 = 0
    fmt.Println(h)
    json.NewEncoder(w).Encode(h)
}
```

VerificarUsuario

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

    reqBody, _ := ioutil.ReadAll(r.Body)

    json.NewEncoder(w).Encode(reqBody)

    var user USUARIO

    json.Unmarshal(reqBody, &user)

    db, err := sql.Open("oci8", "AARON/Marroquin1@localhost:1521/ORCL18")

    if err != nil {

        fmt.Println(err)

        return

    }

    defer db.Close()

    base = db

    fmt.Println(user)

    var respuesta int

    _, err = db.Exec("BEGIN NEWUSER (:1, :2, :3,:4,:5,:6,:7,:8);end;",
user.USERNAME, user.PASS, user.NOMBRE, user.APELLIDO,
user.FECHA_NACIMIENTO, user.CORREO, user.FOTOPERFIL, sql.Out{Dest:
&respuesta})

    if err != nil {

        fmt.Println(err)

        return

    }

    print(respuesta)

    json.NewEncoder(w).Encode(respuesta)}
```


MAIN

```
func main() {  
  
    //Oracle 12c  
  
    router := mux.NewRouter().StrictSlash(true)  
  
    headers := handlers.AllowedHeaders([]string{"X-Request-Headers",  
"Content-Type", "Authorization"})  
  
    methods := handlers.AllowedMethods([]string{"GET", "POST", "PUT",  
"DELETE"})  
  
    origins := handlers.AllowedOrigins([]string{"*"})  
  
  
    router.HandleFunc("/", inicio)  
  
    router.HandleFunc("/Logear", Logear)  
  
    router.HandleFunc("/CargaMasiva", insert_CargaMasiva).Methods("POST")  
  
    router.HandleFunc("/VerificarUsuario",  
VerificarUsuario).Methods("POST")  
  
  
    fmt.Println("servidor sonando en el puerto 7000")  
  
    log.Fatal(http.ListenAndServe(":7000", handlers.CORS(headers, methods,  
origins)(router)))  
  
}
```

EXPLICACIÓN FRONTEND

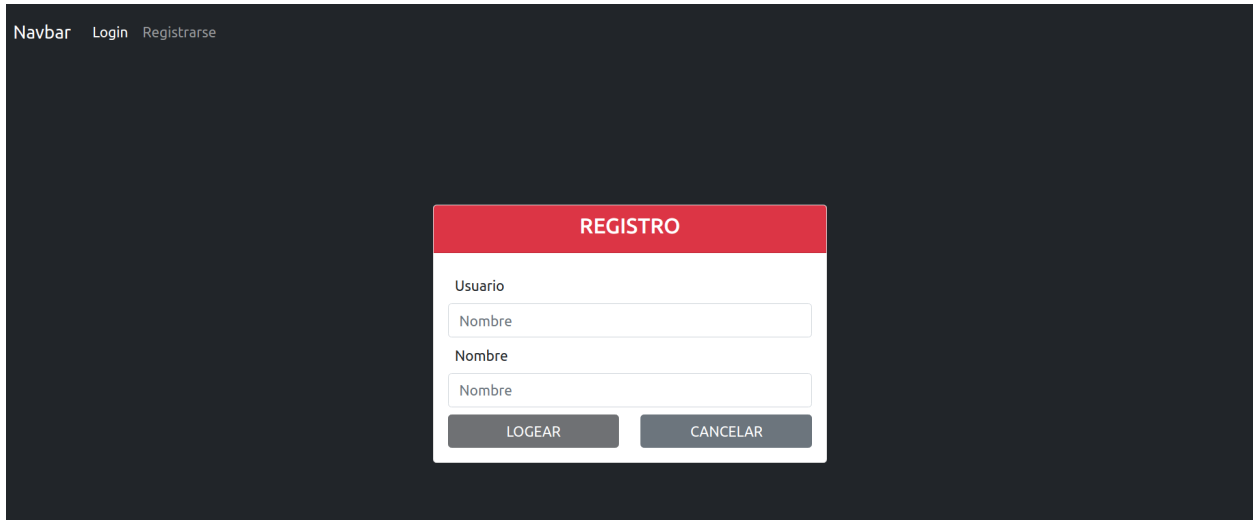
Para esta sección fue utilizado react, apoyandose de bootstrap y Prime react para el diseño de la misma

Creación de routes

se encuentra en el archivo app.js

```
return (  
  <Router className>  
    <div className="App content-fluid bg-dark vh-100">  
      {navbar}  
      <Switch>  
        <Route path="/CargaMasiva" component={() => <CargaMasiva  
setTitle={setTitle} />} />  
        <Route path="/REGISTRO" component={() => <REGISTRO  
setTitle={setTitle} />} />  
        <Route path="/" component={() => <LOGIN setTitle={setTitle} />} />  
      </Switch>  
    </div>  
  </Router>  
)
```

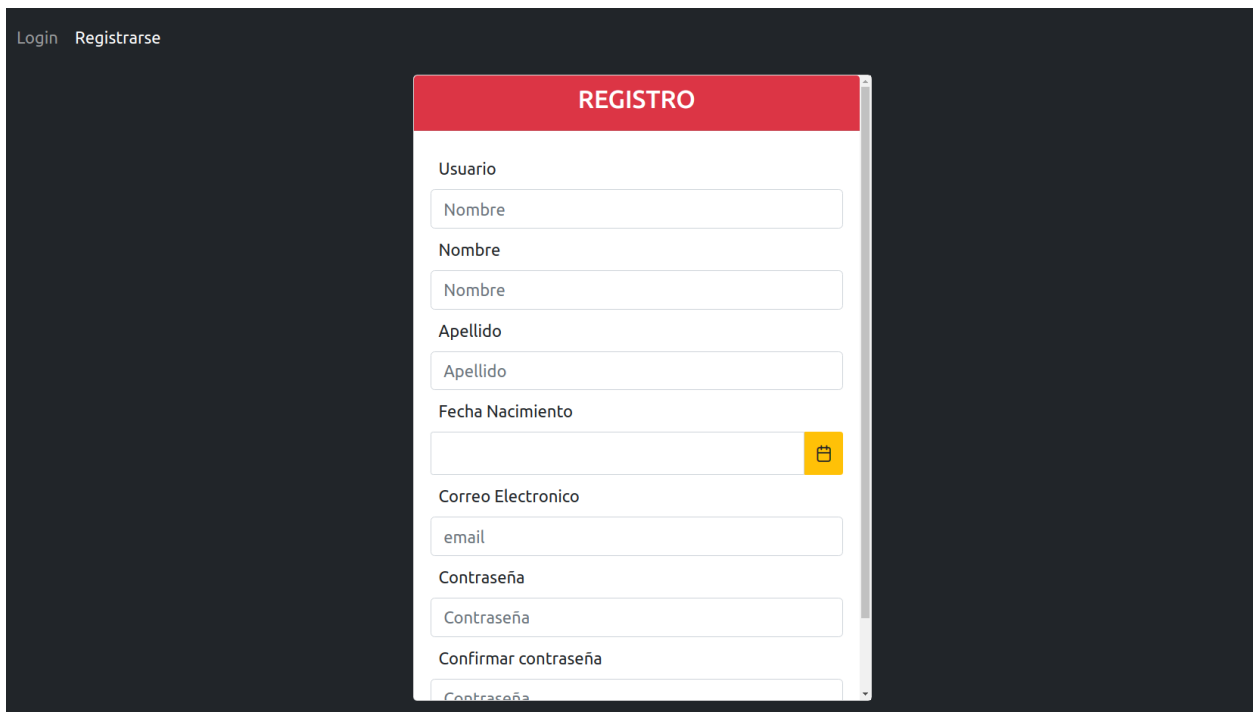
Pestaña de login



The screenshot shows a dark-themed web interface. At the top left, there is a 'Navbar' with links for 'Login' and 'Registrarse'. In the center, a white modal form titled 'REGISTRO' is displayed. The form has two input fields, both labeled 'Nombre', and two buttons at the bottom: 'LOGEAR' and 'CANCELAR'.

La interfaz utilizada para el login los botones permanecen bloqueados hasta que el usuario ingrese los datos. esta interfaz se encuentra en el archivo LoginComponent.js

Pestaña de registro



The screenshot shows a dark-themed web interface. At the top left, there is a 'Login Registrarse' link. In the center, a white modal form titled 'REGISTRO' is displayed. The form has several input fields: 'Nombre' (twice), 'Apellido', 'Fecha Nacimiento' (with a calendar icon), 'Correo Electronico' (with 'email' as a placeholder), 'Contraseña', and 'Confirmar contraseña' (with 'Contraseña' as a placeholder). The form is scrollable, as indicated by a vertical scrollbar on the right side.

Esta interfaz es la elegida para la realización de los registros de usuarios, se encuentra en el archivo RegistroController.js