Universidad de San Carlos de Guatemala
Manejo e Implementación de Archivos
Aux. Renato Flores
Aux. Carlos Hernandez

# Manual Tecnico

Elmer Gustavo Sanchez García
201801351
Fecha:11 de mayo de 2021

**Get Usuario**

Este es el metodo hecho en Golang , es el encargado de recibir las peticiones que envia la parte del cliente (React) este se encarga de mandar ejecutar el metodo para la base de datos.

```go
// Get usuario by id
func getUserByIDAPI(w http.ResponseWriter, r *http.Request) {
    vars := mux.Vars(r)

    w.Header().Set("Content-Type", "application/json")
    userID, err := strconv.Atoi(vars["id"])
    if err != nil {
        var newError errorRequest
        newError.Error = "Invalid ID"
        w.WriteHeader(http.StatusBadRequest)
        json.NewEncoder(w).Encode(newError)
        return
    }
    getTime("GET to: /api/user/" + strconv.Itoa(userID))
    userNew := getUserByIdDB(userID)
    w.WriteHeader(http.StatusOK)
    json.NewEncoder(w).Encode(userNew)

}
```

**GetUser:**

Este método es el encargado de traer un usuario por id pero hace una petición a la base de datos entonces llena un arreglo  y lo devuleve.

```go
func getUserByIdDB(id int) user {       Find related code in MIA_Proyecto_2
    var userNew user
    userList = allUser{}
    idStr := strconv.FormatInt(int64(id), 10)
    query := "SELECT * FROM Usuario WHERE usuario_ID = " + idStr
    rows, err := database.Query(query)
    if err != nil {
        fmt.Println("Error running query")
        fmt.Println(err)
        return userNew
    }
    for rows.Next() {
        rows.Scan(&userNew.ID, &userNew.UserName, &userNew.Password, &userNew.Nombre,
            &userNew.Apellido, &userNew.FechaNacimiento, &userNew.FechaRegistro, &userNew.Email,
            &userNew.FotoPerfil, &userNew.Tipo, &userNew.Membresia)
        userList = append(userList, userNew)
    }
    defer rows.Close()
    return userNew
}
```

**Rutas:**

Estas son las rutas disponibles que puede consumir el cliente, este se ejecutan por medio de GET,POST, PUT, DELETE.

```
// DEPORTE
router.HandleFunc("/api/deporte", getDeporteApi).Methods("GET")
router.HandleFunc("/api/deporte/{id}", getDeporteByIdApi).Methods("GET")
router.HandleFunc("/api/deporte", createDeporteApi).Methods("POST")
router.HandleFunc("/api/deporte/{id}", updateDeporteApi).Methods("PUT")
router.HandleFunc("/api/deporte/{id}", deleteDeporteApi).Methods("DELETE")

// TEMPORADA
router.HandleFunc("/api/temporada", getTemporadaApi).Methods("GET")

// JORNADA
router.HandleFunc("/api/jornada", getJornadaApi).Methods("GET")

// Prediccion
router.HandleFunc("/api/prediccion", getPrediccionApi).Methods("GET")
```

React:
Estos son hooks que se ejecutan del lado del cliente, cada uno tiene su funcionalidad especial,

```
const [sesonList, setSesonList] = useState<ISeason[]>([]);
const [totalSeson, setTotalSeson] = useState<ITotalSeson>(sesonInitial);
const [userByTierList, setUserByTierList] = useState<IUserByTier[]>([]);
const loadJournal = async () => {
  const res = await getSeasons();
  if (res.data) {
    setSesonList(res.data);
  }
};

const comboBoxChange = async (event: any, newValue: ISeason | null) => {
  if (newValue !== null) {
    await loadResults(newValue.Nombre);
  }
};

const loadResults = async (sesonName: string) => {
  const resTotalSesonTemp: ITotalSeson = await getTotalSeson(sesonName);
  const userByTearTemp: IUserByTier[] = await getUserByTier(sesonName);
  setTotalSeson(resTotalSesonTemp);
  setUserByTierList(userByTearTemp);
};
```