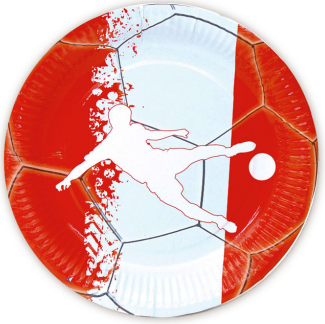
# BD - Proyecto 3ª evaluación



BASE DE DATOS
MIGUEL
BORREGUERO
SORIA

En este proyecto se utiliza una base de datos futbolística. Esta está dividida en 4 colecciones:

- Competiciones
- Equipos
- Plantillas
- Traspasos

## Estructura

3 carpetas son las que componen este proyecto, estas son dist, doc y src.

- dist: en este directorio se encuentran los archivos .js transpilados.
- doc: en este directorio podemos encontrar varios PDF, uno explica todo lo correspondiente con Node y Typescript, otro explica Mongo Atlas y el último es este, que es toda la documentación del proyecto.
- src: directorio en el que podremos encontrar hasta 3 directorios más. Estos son database, datos y model.
    - database: en este se encuentra toda la configuración para conectar con la base de datos de Mongo Atlas.
    - datos: este contiene un archivo .js llamado insert.js donde se encuentran todas las colecciones de la base de datos listas para ser insertadas.
    - model: este directorio almacena los schemas y las interfaces.

  Por último, en esta carpeta se encuentra el archivo intex.ts, en el cual se ve todo el código del programa, rutas,etc.

# Model

En esta carpeta se encuentran los siguientes schemas:

Competiciones:

src/model/competiciones.ts

```ts
import {Schema, model } from 'mongoose'

// Definimos el Schema
const competicionesSchema = new Schema({
    id_competicion: Number,
    nombre_competicion: String
})

// La colección de la BD (Plural siempre)
export const Competiciones = model
('competiciones', competicionesSchema)
```

Equipos:

src/model/equipos.ts

```ts
import {Schema, model } from 'mongoose'

// Definimos el Schema
const equiposSchema = new Schema({
    id_equipo: Number,
    id_competiciones: Array,
    nombre_equipo: String,
    fecha_fundado: Date,
    palmares: [{
      UEFA: Number,
      LaLigaSantander: Number,
      LaLigaSmartBank: Number,
      Champions: Number
    }],
    rey: [{
      confirm: Boolean,
      competicion: Array
    }]
})

// La colección de la BD (Plural siempre)
export const Equipos = model('equipos',
equiposSchema)
```

Plantillas:

src/model/plantillas.ts

```ts
import {Schema, model } from 'mongoose'

// Definimos el Schema
const plantillasSchema = new Schema({
    id_integrante: Number,
    id_equipo: Number,
    tipo: String,
    nombre: String,
    apellido: String,
    fecha_nacimiento: Date,
    valor: Number
})

// La colección de la BD (Plural siempre)
export const Plantillas = model('plantillas', plantillasSchema)
```

Traspasos:

src/model/traspasos.ts

```ts
import {Schema, model } from 'mongoose'

// Definimos el Schema
const traspasosSchema = new Schema({
    id_integrante: Number,
    id_equipo: Number
})

// La colección de la BD (Plural siempre)
export const Traspasos = model('traspasos', traspasosSchema)
```

Por último, en este directorio, encontramos las interfaces:

```ts
export interface Integrante{
  id_integrante: number,
    id_equipo: number,
    tipo: string,
    nombre: string,
    apellido: string,
    fecha_nacimiento: Date,
    valor: number
}
```

```
export interface FondoEquipo{
  id_equipo: number,
  nombre_equipo: string,
  valorTotal: number
}

export interface EquipoPlantilla{
  id_equipo: number,
  nombre_equipo: string,
  id_integrante: number,
  nombre_integrante: string,
  fecha_nacimiento: Date,
  valor: number
}

export interface TraspasosRealizados{
  id_equipo: number,
  nombre_equipo: string,
  id_traspaso: number,
  nombre_integrante: string,
  id_integrante: number
}
```

## Rutas

En mi proyecto encontramos 6 rutas, estas son las siguientes:

Inicio: una breve descripción e introducción de todas las rutas que comprende el proyecto.

```
const fun0 = async (req: Request, res: Response) => {
  res.send("BASE DE DATOS FUTBOLÍSTICA DE MIGUEL BORREGUERO
  SORIA<br><br>Valor de la suma de todo los equipos registrados en la base
  de datos:<br>/valor_equipos<br><br>Valor del equipo introducido por
  teclado, sea el nombre o el id:
  <br>/valor/:nombre_equipo_entrada/:id_equipo_entrada<br><br>Registro de
  todos los traspasos realizados entre los equipos de la base de
  datos:<br>/traspasos<br><br>Buscador de jugadores según la función que
  tenga el integrante del equipo y su valor.<br>/buscador/:funcion/:valor")
}
```

Función 1: todos los equipos de la base de datos de este proyecto contienen 4 integrantes. Cada uno de ellos tienen un valor. La suma de estos 4 valores que obtendríamos es el valor

total del equipo. Sabiendo esto, lo que esta función tiene como objetivo es mostrar por pantalla el valor total de todos los equipos de la base de datos.

Lo primero que hace esta función es realizar un find de la colección Plantillas. Dicho resultado se guardará en un array y en una variable irán sumándose todos los valores de todos los integrantes debido a que se estará recorriendo el array con un for.

```
let arrayIntegrantes: Array<Integrante>
const query: any = await Plantillas.find(
{}, {_id:0, id_integrante:1, id_equipo:1, tipo:1, nombre:1,
 apellido:1, fecha_nacimiento:1, valor:1})

console.log(query)
arrayIntegrantes = query
console.log(arrayIntegrantes)

let valorTotal: number = 0
let integrante: Integrante

for (integrante of arrayIntegrantes){
  console.log(integrante.valor)
  valorTotal += integrante.valor
}

res.json("ValorTotal: " + valorTotal )
```

Función 2: esta función sirve como buscador de equipos. El objetivo es obtener el nombre y el valor total del equipo que se introduce por teclado. Para ello, la función pide el nombre del equipo o el id. Si se ofrecen los dos, solo se obtendrá el valor correspondiente al de la id. Esto lo podemos conseguir realizando un lookup de Plantillas y Equipos y filtrando la salida según sea el nombre o el id del equipo que se introduzca. Después de esto creamos una variable valorTotal y otra para el nombre del equipo para cada una hacemos un for recorriendo un array. Hecho esto tanto el valor como el nombre se almacenará con una nueva interfaz.

```
let valorTotal: number = 0
let nombre_del_equipo: string = ""
let integrante: Integrante
let equipo: FondoEquipo

interface respuesta{
  nombre_equipo: string,
  valorEquipo:number
}
let resultado: Array<respuesta> = []

for (integrante of arrayIntegrantes){
  console.log(integrante.valor)
  valorTotal += integrante.valor
}
for (equipo of arrayEquipo){
  nombre_del_equipo = equipo.nombre_equipo
}
console.log(`Sueldo total: ${valorTotal}`)
resultado.push({
  nombre_equipo: nombre_del_equipo,
  valorEquipo: valorTotal
})
res.json(resultado)
```

Función 3: el objetivo de esta función es mostrar todos los equipos de la base de datos con sus respectivos traspasos si los hubiera. Para ello haremos dos lookup, uno para unir Equipos con Traspasos y otro para unir Equipos con Plantillas. Para ello, crearé una variable String en la que meteré toda la información demandada. Esto lo consigo recorriendo un array y metiendo en la variable anteriormente nombrada el nombre del equipo al que se va el integrante, el id de todos los integrantes de ese equipo, el nombre de los integrantes de ese equipo y por último id del integrante que es traspasado.

```
let salida: string = ""
let traspasos: TraspasosRealizados

interface respuesta{
  salidaRespuesta: string
}
let resultado: Array<respuesta> = []

for (traspasos of arrayTraspasos){
  console.log(traspasos.id_traspaso)
  salida =`${salida}"
  "${traspasos.nombre_equipo}" | "$
  {traspasos.id_integrante}" | "$
  {traspasos.nombre_integrante}" | "$
  {traspasos.id_traspaso}"`
}
console.log(salida)
resultado.push({
  salidaRespuesta: salida
})
res.json(resultado)
```

Función 4: el objetivo es mostrar el o los integrantes que cumplan con las condiciones introducidas por teclado. Estas condiciones son: la función del integrante (entrenador, médico o jugador) y su valor (€). De esta forma al introducir una función del integrante y un valor, se buscará en la base de datos, en concreto en la colección plantillas, un integrante que cumpla dichas condiciones. Una vez encontrado el o los integrantes mostrará su id, su nombre, el equipo al que pertenece, su valor actual, su fecha de nacimiento y su función.

```
const fun4 = async (req: Request, res: Response) => {
  const funcion : string = req.params.funcion
  const valor : number = parseInt(req.params.valor)
    await db.conectarBD()
    .then(
        async (mensaje) => {
          console.log(mensaje)
          const query: any = await Plantillas.aggregate([
            {
              $match:{
                $and:[
                  {
                    "tipo": funcion,
                  },
                  {
                    "valor": valor
                  }
                ]
              }
            },
            {
              $lookup:{
                from: "equipos",
                localField: "id_equipo",
                foreignField: "id_equipo",
                as: "equipos"
              }
            },
            {
              $project:{
                id_integrante: "$id_integrante",
                nombre_integrante: "$nombre",
                id_equipo: "$equipos.id_equipo",
                nombre_equipo: "$equipos.nombre_equipo",
                valor: "$valor",
                fecha_nacimiento: "$fecha_nacimiento",
                funcion:funcion,
                valorFinal:valor
              }
            },
            {
              $project:{
                _id:0,
                id_integrante: 1,
                nombre_integrante: 1,
                id_traspaso: 1,
                nombre_equipo: 1,
                valor: 1,
                fecha_nacimiento: 1,
                funcion: 1,
                valorFinal: 1
              }
            }
          ])

          console.log(query)
          res.json(query)
})
```

Función 5: esta función tiene como objetivo mostrar por pantalla los equipos que sean reyes de alguna competición o no, dependiendo de lo que se le introduzca por teclado (“si” o “no”) y las competiciones en las que están participando.
Para realizar esta función se requiere un lookup para acceder, además de la colección equipos, a la colección competiciones. Hecho esto crearemos dos interfaces, una para la opción sí y otra para la opción no. Posteriormente se recorrerá un array con la salida del aggregate, donde se decidirá qué mostrar por pantalla.

```
arrayRespuesta = query
let equipo : respuestasi
let equipo2 : respuestano

interface respuestasi{
  nombre_equipo: string,
  confirm: boolean,
  id_competiciones_participa: number,
  participa: string
  rey_competicion: Array<string>
}
interface respuestano{
  nombre_equipo: string,
  confirm: boolean,
}

let resultado: Array<respuestasi> = []
let resultadono: Array<respuestano> = []
let salidaError: string = "ERROR: no se ha introducido correctamente la entrada
por teclado."

for (equipo of arrayRespuesta){
  console.log(equipo.nombre_equipo)
  if(confirmar=="si"){
    if(equipo.confirm){
      resultado.push({
        nombre_equipo: equipo.nombre_equipo,
        confirm: equipo.confirm,
        id_competiciones_participa: equipo.id_competiciones_participa,
        participa: equipo.participa,
        rey_competicion: equipo.rey_competicion,
      })
    }
  } else if (confirmar=="no"){
    if(!equipo.confirm){
      resultadono.push({
        nombre_equipo: equipo.nombre_equipo,
        confirm: equipo.confirm,
      })
    }
  }
}

if (confirmar=="si"){
    console.log(resultado)
    res.json(resultado)
  } else if (confirmar=="no"){
    console.log(resultadono)
  res.json(resultadono)
} else {
    console.log(salidaError)
    res.json(salidaError)
}
```

## Bibliografía


- Replit de Adolfo
- Typescript
- Replit
- Node
- Mongo Atlas