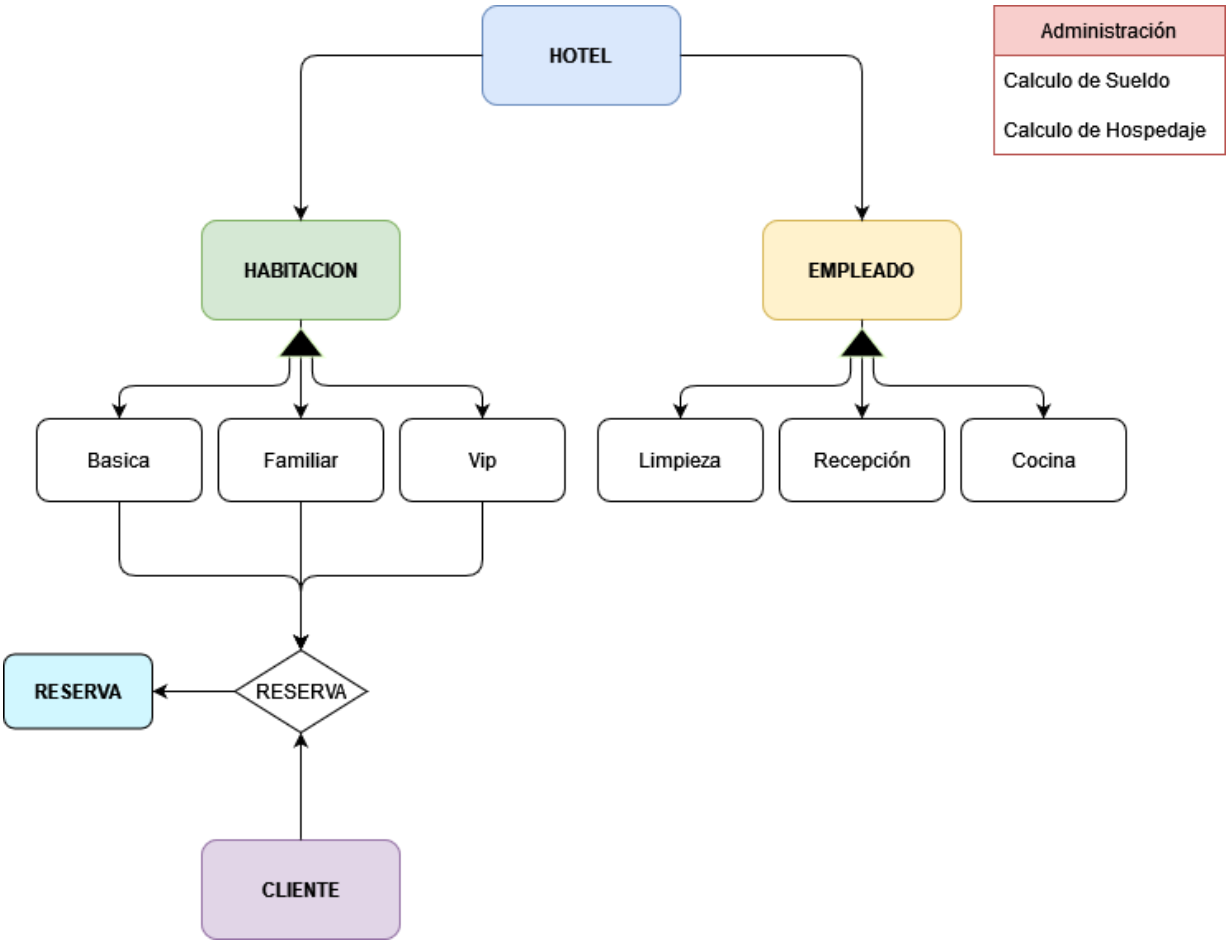
Proyecto de 1ºTrimenste
SGBD

Hotel



Alumno:Alvaro Lucio-Villegas de Cea
Curso:2ºA.S.I.R-A

Índice

Métodos

Métodos de Creación

Metodo de Creacion de Habitaciones

```
//Crear Habitaciones
 let tipo= await leerTeclado('Tipo de Habitación')
 let IDHab=await leerTeclado('Identificador de Habitación')
 let camas=parseInt(await leerTeclado('Numero de Camas'))
 let pNoche=parseInt(await leerTeclado('Precio por noche'))
 if(tipo=="b"){
    let Desayuno=await leerTeclado('Desayuno')
    if (Desayuno=="si"){
        BoolDesayuno=true
    }else{
        BoolDesayuno=false
    }
    Nhabitacion2= new HBasica(tipo,IDHab,camas,pNoche,BoolDesayuno);
    Global.habitaciones.push(Nhabitacion2)
 }else if(tipo=="f"){
     let Supletoria=await leerTeclado('Supletoria')
     if (Supletoria=="si")
     {
         BoolSupletoria=true
        }else{
         BoolSupletoria=false
        }
   Nhabitacion2= new HFamiliar(tipo,IDHab,camas,pNoche,BoolSupletoria);
   Global.habitaciones.push(Nhabitacion2)
 }else if(tipo=="v"){
        let Spa=await leerTeclado('Spa')
        if (Spa=="si")
        {
            BoolSpa=true
        }else{
            BoolSpa=false
        }
    Nhabitacion2= new HVip(tipo,IDHab,camas,pNoche,BoolSpa);
    Global.habitaciones.push(Nhabitacion2)
 }

  let a= await leerTeclado('¿Quieres ver las habitaciones creadas? s o n')
  if (a=="s") {
      Global.habitaciones.forEach(element => {
          console.log(element)
      });
  }
hotel()
 break
```

Metodo de Creacion de Empleados

```
//Crear Empleado
let tipoE= await leerTeclado('Tipo de Empleado')
let DniEmp=await leerTeclado('Dni Empleado')
let SalarioBase=parseInt(await leerTeclado('Salario Base'))

    if(tipoE=="Ec"){

        let NEstrella=parseInt( await leerTeclado('Numero de Estrellas Michelin'))
        let Titulacion=await leerTeclado('Titulo de competencia')

        Empleado2= new ECocina(DniEmp,SalarioBase,NEstrella,Titulacion);
        Global.trabajadores.push(Empleado2)
    }else if(tipoE=="Er"){

        let Snocturnidad=await leerTeclado('Nocturnidad')
        if (Snocturnidad=="si")
        {
            BoolNoctur=true
        }else{
            BoolNoctur=false
        }


        Empleado2= new ERecepcion(DniEmp,SalarioBase,BoolNoctur);
        Global.trabajadores.push(Empleado2)

    }else if(tipoE=="El"){
       Global.habitaciones.forEach(element => {
          console.log(element)
       });
       let habitacionesLimp=await leerTeclado('Numero de habitaciones limpiadas separados por ","')
       let NHLimp:Array<Habitacion>=new Array<Habitacion>()

       habitacionesLimp.toLowerCase().trim().split(",").forEach(element => {
            Global.habitaciones.forEach(h => {
                if(h.IDHab==element){
                    NHLimp.push(h)
                    }
                });
        })
        Empleado2= new ELimpieza(DniEmp,SalarioBase,NHLimp);

        Global.trabajadores.push(Empleado2)


        let c= await leerTeclado('¿Quieres ver las habitaciones asignadas? s o n')
        if (c=="s") {
            Global.trabajadores.forEach(element => {···
            });
```

Metodo de Creacion de Clientes

```
case 3:
    //Crear Cliente
    let AltaCliente:Cliente
    let dniCl= await leerTeclado('Dni del Cliente')
    let nombreCl= await leerTeclado('Nombre del Cliente')
    let nTarheta=parseInt(await leerTeclado('Numero de Tarjeta de Credito'))
    AltaCliente=new Cliente(dniCl,nombreCl,nTarheta)
    Global.clientes.push(AltaCliente)
    hotel()
    break
```

Método de Creación de Reservas

```
//Crear Hacer Reserva


        let Reserva2:Reserva
        Global.clientes.forEach(element => {
            console.log("Dni Clientes :"+element.dni+"\n");

        });


        Global.habitaciones.forEach(element => {
            if(element.estado==false){
                console.log("Id.Haitaciones Libres :"+element.IDHab);
            }


        })
        let dniCl= await leerTeclado('Dni del Cliente')
        let IdHab= await leerTeclado('Identificador de la Habitacion')
        let NDias= parseInt(await leerTeclado('Numero de Dias Hospedados'))
        let nPersonas= parseInt(await leerTeclado("Numero de Personas"))

        Reserva2= new Reserva (dniCl,NDias,IdHab,nPersonas);

        Global.habitaciones.forEach((element2,index)=>{
            if(element2.IDHab==IdHab){
                Global.habitaciones[index].estado=true

            }

        })

        Global.reservas.push(Reserva2)
        hotel()
break
```

Cálculos que realiza el programa

Sueldo del Empleado

```
//Calcular Sueldo
//Maximo
let max=0
let max_id=""
let min_id=""

let min=Global.trabajadores[0].calcularSueldo()
let suma=0,n=0,media

Global.trabajadores.forEach(element => {
    if(element.calcularSueldo()>max) {
        max=element.calcularSueldo()
        max_id=element.dni
    }
    if(element.calcularSueldo()<min) {
        min=element.calcularSueldo()
        min_id=element.dni
    }

})

Global.trabajadores.forEach(element => {
   suma+=element.calcularSueldo()
   n++

});

media=suma/n

let informe:string=("INFORME DE LOS SUELDOS\n##################################\n\n\tSUELDO MAXIMO:\n\t\t DNI: "+max_id+
"\n\t\tSUELDO: "+max+"\n\n################################"+"\n\n\tSUELDO MINIMO:\n\t\t DNI: "+min_id+
"\n\t\tSUELDO: "+min+"\n\n################################"+"\n\n\tSUELDO MEDIO:\n\t\t : "+media)

console.log(informe)
hotel()
```

Precio de la reserva

```
//Calcular Precio Reserva

    await db.conectarBD()

    //let dReserva:iReservas
    let DniCl= await leerTeclado(' Dni del Cliente\n ')
    let tmpdHabitaciones:Habitacion
    let tmpReservas:Reserva
    let dHabitacion:tHabitaciones
    let dReserva:tReservas
    let query:any = await Reservas.find({_cliente:DniCl})
    let query2:any = await Habitaciones.find({})
    let PrecioFinal:number =0,PrecioBasica:number =0,PrecioFamiliares:number =0,PrecioVip:number =0

    for ( dHabitacion of query2){

        //El precio varia en base al tipo de Habitacion que sea

        if(dHabitacion._tipoObjeto==="b"){
            tmpdHabitaciones=new HBasica (dHabitacion._tipoObjeto, dHabitacion._IdHab,dHabitacion._Camas,dHabitacion._PNoche,dHabitacion._desayuno)

            for(dReserva of query){

                tmpReservas=new Reserva(dReserva._cliente,dReserva._nDias,dReserva._habitacion,dReserva._nPersonas)
                if(DniCl==tmpReservas.cliente){
                    if(tmpdHabitaciones.IDHab== tmpReservas.habitacion)
                    {
                        PrecioBasica=tmpdHabitaciones.pNoche*tmpReservas.nDias
                        if(dHabitacion._desayuno==true){
                            PrecioBasica=PrecioBasica*1.25
                        }
                        console.log("###################### PRECIO FINAL DE HOSPEDAJE BASICA ######################")
                        console.log("Precio de Habitacion :"+PrecioBasica+"€\n")
                    }

                }
            }

        }else if(dHabitacion._tipoObjeto==="f"){···
        }else if(dHabitacion._tipoObjeto=="v"){···
        }

    }
    PrecioFinal=PrecioBasica+PrecioFamiliares+PrecioVip
    console.log("Precio Total de la Reserva :"+PrecioFinal+"€")

    await db.desconectarBD()
```

Extras:

Buscar la habitación asignada al cliente

Esta se realiza en un fichero aparte que se llama funciones que se llama en el index.

```
export let BusquedaReserva=async()=>{
  await db.conectarBD()
  let dCliente:Iclientes
  let dShemaReserva:iReservas
  let Cdni= await leerTeclado('Dni Cliente')
  let query: any =  await Clientes.find({_dni:Cdni})
  let query2:any = await Reservas.find({})
  for (dCliente of query){
      for(dShemaReserva of query2){
        if(dCliente._dni==dShemaReserva._cliente)
        {
          console.log("Habitacion Nº"+dShemaReserva)
        }
      }

  }
  await db.desconectarBD()
```

Listar Arrays Globales

Recorremos los Arrays Globales y se muestran por pantalla.

```
case 4:
    console.log("################################")
    console.log("1.-Listar Clientes")
    console.log("2.-Listar Habitaciones")
    console.log("3.-Listar Empleados")
    console.log("4.-Listar Reservas")
    let Selec=parseInt(await leerTeclado("Seleccion"))
    switch(Selec){
        case 1:
        //Listar Clientes

        Global.clientes.forEach(element => {
            console.log(element.nombre,element.dni)
        });
        break
        case 2:
        //Listar Habitaciones
        Global.habitaciones.forEach(element => {
            console.log(element.IDHab,element.estado)
        });
        break
        case 3:
        //Listar Empleados
        Global.trabajadores.forEach(element => {
            console.log(element.dni,element.tipo)
        });
        break
        case 4:
        //Listar Reservas
        Global.reservas.forEach(element => {
            console.log(element)
        })
        break
```

Eliminar Cliente

Buscamos tanto en el array global como en la base de datos para eliminarlo

```
export let EliminarCliente = async ()=>{
  let dniCl= await leerTeclado("Dni del Cliente a eliminar")

  await db.conectarBD()
  let query:any =await Clientes.findOneAndDelete({_dni:dniCl})
    .then(()=> console.log("Elimido Correctamente"))
    .catch((fallo:any)=>console.log("Fallo: "+fallo))
  await db.desconectarBD()

  Global.clientes.forEach((element,index)=>{
    if (element.dni==dniCl){

      Global.clientes.splice (index,1)
    }
  });
```

Check Out

```
//Volver a asignar una Habitacion Libre
Global.habitaciones.forEach((element)=>{
    console.log(element)
})
let IDCko= await leerTeclado('Numero de la habitación que queda libre')
Global.habitaciones.forEach((element,index)=>{
    if(element.IDHab==IDCko)

        Global.habitaciones[index].estado=false
    }

)
```

Cambiar Base de Datos

(Mongodb Local a Mongodb Atlas)

```
case 100:
    let selcbd= await parseInt( await leerTeclado('1.-Para guardar en mongo Atlas --- 2.- Para guardar en local'))

    if(selcbd==1){
        db.cambiarBD(true)
    }else if(selcbd==2){
        db.cambiarBD(false)
    }

    hotel()
```

```
cambiarBD=async(a:boolean)=>{
    if(a==false){
        this._cadenaConexion = 'mongodb://localhost/test'

    }else{

        this._cadenaConexion ='mongodb+srv://alvaro:1234@cluster0.e2jny.mongodb.net/myFirstDatabase?retryWrites=true&w=majority'
    }
}
```

Función Salvar

Esta es muy importante ya que borra lo que hay en la base de datos y agrega los valores que se encuentran el los Arrays Globales.Para convertir los objetos de Typescript a objetos (documentos) en mongodb es necesario realizar una serie de pasos:

1. Crear los Schemas
2. Crear un documento del mismo tipo que el esquema asociado
3. Subir mediante una función a la base de datos

1.Creamos los esquemas basándonos en los valores que queremos almacenar en la base de datos y en la jerarquía de tipos ,subtipos .
Para cada clase o jerarquía es necesario crear un schema éste debe tener todos los componentes de los distinto tipos si cuelgan de él.

```
// Definimos el Schema
const EmpleadosSchema = new Schema({
    //caracteristicas del objeto
    _tipoObjeto: {
        type: String  //Valores "A, "T"...
    },
    _dni: {
        type: Number
    },
    _salariosBase: {
        type: Number
    },
    _Titulacion: {
        type: String
    },
    _NEstrella: {
        type: Number
    },
    _habitaciones:{
        type: Array,
        default:[]
    },
    _Nocturnidad:{
        type:Boolean
    }

})
```

Un ejemplo Clase Empleado tiene 3 Subclases que son Empleados de Limpieza ,Empleados de Recepción y Empleados de Cocina cada uno comparte los valores del padre pero cada uno tiene valores que el resto no tiene

```
export type iCocina={
    //caracteristicas tipo
    _tipoObjeto: string | null,
    _dni: string | null,
    _salariosBase: number | null,
    _Titulacion: string | null,
    _NEstrella: number | null
}
export type iLimpieza={
    _dni: string | null,
    _salariosBase: number | null,
    _tipoObjeto: string | null,
    _habitaciones: Habitacion[] | null

}
export type iRecepción={
    _dni: string | null,
    _salariosBase: number | null,
    _tipoObjeto: string | null,
    _Nocturnidad:boolean | null
}
```

También es muy importante crear un “model” que se encargará de unir (Colección) la base de datos con el Schema.

```
// La colección de la BD (Plural siempre)
export const Empleado = model('Empleados', EmpleadosSchema)
```

2.-Ahora crearemos un documentos con la estructura del tipo que queremos utilizar , en este caso aquí muestro los de Empleados

```
let dShemaECo: iCocina =
{
  _tipoObjeto: null,
  _dni: null,
  _salariosBase: null,
  _Titulacion: null,
  _NEstrella: null
}
let dShemaELi: iLimpieza =
{
  _tipoObjeto: null,
  _dni: null,
  _salariosBase: null,
  _habitaciones: null
}
let dShemaERe: iRecepción =
{
  _tipoObjeto: null,
  _dni: null,
  _salariosBase: null,
  _Nocturnidad: null
}
```

Ahora es el paso más complicado de entender que es unir los valores de los Arrays de Typescript con los documentos .

Para ello en este caso ya que vamos a guardar una jerarquía primero vamos a guardar en todos los documentos de los subtipo los valores comunes ,luego de esto empezamos a discriminar por el tipo de dato ,es recomendable que se guarde en la base de datos el tipo que es para posteriori poder guardarlo en el documento correcto a su tipo.

```
for (let p of Global.trabajadores) {
  dShemaECo._dni = dShemaELi._dni = dShemaERe._dni = p.dni
  dShemaECo._tipoObjeto = dShemaELi._tipoObjeto = dShemaERe._tipoObjeto = p.tipo
  dShemaECo._salariosBase = dShemaELi._salariosBase = dShemaERe._salariosBase = p.SalarioBase


  if (p instanceof ECocina) {
    dShemaECo._tipoObjeto = "Ec"
    dShemaECo._Titulacion = p.Titulacion
    dShemaECo._NEstrella = p.NEstrella

    ESchema = new Empleado(dShemaECo)

  } else if (p instanceof ELimpieza) {
    dShemaELi._tipoObjeto = "El"
    dShemaELi._habitaciones = p.get_Habitaciones

    ESchema = new Empleado(dShemaELi)
  } else if (p instanceof ERecepcion) {
    dShemaERe._tipoObjeto = "Er"
    dShemaERe._Nocturnidad = p.nocturnidad

    ESchema = new Empleado(dShemaERe)
  }
  await ESchema.save()
}
```

3.-Para realizar cualquier operación con la base de datos es importante que se encuentre entre estas funciones:

Abrir la coneccion con la BD `await db.conectarBD()`

Cerrar la coneccion con la BD `await db.desconectarBD()`

Esto se hace para poder realizar algunas operaciones especiales,en este programa se usan las siguientes:

Guardar:

```
await RSchema.save()
```

Consultar:

En este caso no devuelve un array de objetos pero existe la posibilidad de que nos devuelva solo un objeto.

También se puede introducir un valor discriminatorio para no traer todo el contenido de la colección .

```
let query: any =  await Clientes.find({_dni:Cdni})
```

Borrar BD:

Elimina todo el contenido de la colección.

```
await Empleado.deleteMany({})
```

Función Datos de Prueba

Crea en los arrays globales objetos para realizar pruebas de cálculos

```
export let DPrueba = async()=>{
  Global.trabajadores.push(new ECocina("2", 2000, 2, "fpm"))
  Global.trabajadores.push(new ERecepcion("1", 2000, true))
  Global.trabajadores.push(new ELimpieza("3", 2000,[]));



  //let habitaciones: Array<Habitacion> = new Array<Habitacion>();
  Global.habitaciones.push(new HBasica("b", "1", 3,40, false))
  Global.habitaciones.push( new HFamiliar("f", "2", 3, 120, true))
  Global.habitaciones.push( new HVip("v", "3", 3, 200, true))

  //let clientes: Array<Cliente> = new Array<Cliente>();
  Global.clientes.push(new Cliente("1", "alvaro", 1234))

  //let reservas: Array<Reserva> = new Array<Reserva>();
  Global.reservas.push(new Reserva("1",4,"1",4,))
```

Función de Inicio

Esta función carga todos los valores de la base de datos en los arrays globales para ello es necesario hacer el mismo proceso anterior para guardar pero en el orden invertido.

Vamo a ver el Ejemplo de las habitaciones ,Primero se recorre la query y se va almacenado en cada esquema después de esto creamos un objeto temporal del tipo del padre y le asignamos la subclase a la que pertenece para discriminar una de otras usamos el campo tipo Objeto.Después se carga en los arrays globales de cada clase

Este proceso se repite con Empleados ,Clientes y Reservas

```
export let Inicio=async()=>{
  ////HABITACIONES
  await db.conectarBD()
  await Empleado.deleteMany({})
  await Habitaciones.deleteMany({})
  await Clientes.deleteMany({})
  await Reservas.deleteMany({})
  await db.desconectarBD()
  await db.conectarBD()

  let tmpdHabitaciones:Habitacion
  let dHabitacion:tHabitaciones
  let query:any = await Habitaciones.find({})
  for(dHabitacion of query){

    if(dHabitacion._tipoObjeto=="b"){
      tmpdHabitaciones=new HBasica (dHabitacion._tipoObjeto, dHabitacion._IdHab,dHabitacion._Camas,dHabitacion._PNoche,dHabitacion._desayuno)

      Global.habitaciones.push(tmpdHabitaciones)
    }else if(dHabitacion._tipoObjeto=="f"){
      tmpdHabitaciones=new HFamiliar (dHabitacion._tipoObjeto, dHabitacion._IdHab,dHabitacion._Camas,dHabitacion._PNoche,dHabitacion._supletoria)

      Global.habitaciones.push(tmpdHabitaciones)
    }else if(dHabitacion._tipoObjeto=="v"){
      tmpdHabitaciones=new HVip (dHabitacion._tipoObjeto, dHabitacion._IdHab,dHabitacion._Camas,dHabitacion._PNoche,dHabitacion._spa)

      Global.habitaciones.push(tmpdHabitaciones)
    }


  }
  /*Global.habitaciones.forEach(element=>{
    console.log(element)
  })*/
  await db.desconectarBD()
 ///////EMPLEADOS

  await db.conectarBD()
  let tmpEmpleado:Trabajador
  let dEmpleado:tEmpleado
  let query2:any = await Empleado.find({})
  for( dEmpleado of query2){···
  }
  await db.desconectarBD()
  /*Global.trabajadores.forEach(element=>{
      console.log(element)
  })
  */
 ///Clientes
 await db.conectarBD()
  let tmpCliente:Cliente
  let dCliente:tClientes
```