Proyecto

En este proyecto implementaré la creación de objetos persistentes además de utilidades con los mismos, lo primero a mencionar son los distintos modelos

```
export class Cliente {

    _ID:number
    _Nombre:string
    _Edad:number
    _carrito:Array<any>


    constructor(_ID:number, _Nombre:string, _Edad:number, _carrito:Array<any>)
    {
        this._ID=_ID
        this._Edad=_Edad
        this._Nombre=_Nombre
        this._carrito=_carrito
    }

     get ID(){
        return this._ID
    }
    get edad(){
        return this._Edad
    }
    get nombre(){
        return this._Nombre
    }
    get carrito (){
        return this._carrito
    }
```

```
//import { Discos } from './discos'
export class Tocadiscos {
    protected _ID : number
    protected _stock : number
    protected _precio : number

    //private discos:Array<Discos>

    constructor(_ID: number,  _stock:number, _precio: number
        ){
            this._ID=_ID
            this._stock=_stock
            this._precio=_precio

        }
    get ID(){
        return this._ID
    }
    get precio(){
        return this._precio
    }
    get stock(){
        return this._stock
    }
    set ID(ID:number){
        this._ID=ID
    }
    set stock(stock:number){
        this._stock=stock
    }
```

```
export class Gramofono extends Tocadiscos{
    protected _ID:number
    protected _velocidades='manual'
    protected _corneta: boolean
    protected _stock: number
    protected _precio : number
```

```
import { Tocadiscos } from "./tocadiscos"
export class Typedisco extends Tocadiscos{

    protected _ID : number
    protected _modelo:string
    protected _velocidades:number
    protected _stock: number
    protected _precio:number
```

```
export class Vinilo{
    protected _id: number
    protected _nombre: string
    protected _tamaño: string
    protected _stock: number
    protected _precio:number

    constructor(_id:number, _nombre:string,_tamaño:string, _stock:number, _precio:number)
    {

        this._id=_id
        this._nombre=_nombre
        this._tamaño=_tamaño
        this._stock=_stock
        this._precio=_precio
    }
    get ID (){
        return this._id
    }
    get tamaño(){
        return this._tamaño
    }
    get stock(){
        return this._stock
    }
    get nombre(){
        return this._nombre
    }
    get precio(){
        return this._precio
    }
```

```
export class Pedido {

    protected _ID: number
    protected _objetos: Array<any>
    protected _fecha: Date
    protected _precio_total:number

    constructor(_ID:number, _objetos:Array<any>, _fecha:Date,_precio_total:number){
        this._ID=_ID
        this._objetos=_objetos
        this._fecha=_fecha
        this._precio_total=_precio_total
    }

    get ID (){
        return this._ID
    }

    get objetos(){
        return this._objetos.forEach(Element=>console.log(Element))
    }
    get fecha(){
        return this._fecha
    }
    get precio_total(){
        return this._precio_total
    }
```

Ahora mostraré los distintos Schemas creados para la subida de objetos a la base de datos

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

const clienteSchema = new Schema ({

    _ID :{
        type : Number
    },
    _Edad:{
        type: Number
    },
    _Nombre:{
        type:String
    },
    _carrito:{
        type: []
    }
})
export type iCliente = {
    _ID : number | null
    _Edad: number | null
    _Nombre: string | null
    _carrito: Array<any> | null
}

export const Clientes = model ('Clientes', clienteSchema)
```

```
import {Schema, model } from 'mongoose'
import { Cliente } from '../modelos/clientes'
const maquinaSchema = new Schema({

    _tipoObjeto:{
        type:String
    },
    _ID:{
        type:Number
    },
    _velocidades:{
        type:String
    },
    _corneta:{
        type:Boolean
    },
    _modelo:{
        type:String
    },
    _fecha:{
        type:Date
    },
    _reparacion:{
        type:String
    },
    _stock:{
        type:Number
    },
    _precio:{
        type:Number
    }
})
```

```
export type igramofono = {

    _tipoObjeto: string | null
    _ID: number | null
    _velocidades:string | null
    _corneta:boolean | null
    _stock:number | null
    _precio:number | null
}
```

```
export type iTipotocadiscos = {

    _tipoObjeto: string | null
    _ID: number | null
    _velocidades:number | null
    _modelo:string | null
    _stock:number | null
    _precio:number | null
}
```

```
import {Schema, model} from 'mongoose'
const discoSchema = new Schema({

    _ID:{
        type:Number
    },
    _objetos:{
        type:Array<any>()
    },
    _fecha:{
        type:String
   },
   _id_cliente:{
       type:Number
   },
   _precio_total:{
       type:Number
   }
})

export type iPedido = {

    _ID: number | null
    _objetos: Array<any> | null
    _fecha: Date | null
    _id_cliente:number |null
    _precio_total:number|null
}

export const Pedidos1 = model ('pedidos', discoSchema)
```

```
import {Schema, model} from 'mongoose'
import { Cliente } from '../modelos/clientes'
const discoSchema = new Schema({

    _ID:{
        type:Number
    },
    _nombre:{
        type:String
    },
    _tamaño:{
        type:String
    },
    _stock:{
        type:Number
    },
    id_cliente:{
        type:Number
    },
    _precio:{
        type:Number
    }
})

export type iVinilo = {

    _ID: number | null
    _nombre:string | null
    _tamaño:string | null
    _stock: number | null
    _precio:number | null
}
export const Discos = model ('discos', discoSchema)
```

Una vez introducido explicare las principales funciones del sistema

salvar()

```
export let salvar = async () => {

    let discos: Array<Vinilo> = new Array<Vinilo>();

    discos[0] = new Vinilo(1, 'Led Zeppelin -- Led zeppelin 1', 'LP', 10, 25);
    discos[1] = new Vinilo(2, 'Led Zeppelin -- Led zeppelin 2', 'LP', 13, 30);
    discos[2] = new Vinilo(3, 'Black Sabbath -- Volumen 4', 'EP', 20, 27);
    discos[3] = new Vinilo(4, 'Black Sabbath -- Black Sabbath', 'LP', 12, 55);
    discos[4] = new Vinilo(5, 'Uriah Heep -- Magicians Birthday', 'LP', 44, 34);
    discos[5] = new Vinilo(6, 'Speedometer -- Again and Again', 'Single', 24, 17);

    let tocadiscos: Array<Tocadiscos> = new Array<Tocadiscos>();

    tocadiscos[0] = new Gramofono(1, true, 5, 1000)
    tocadiscos[1] = new Gramofono(2, false, 7, 500)
    tocadiscos[2] = new Typedisco(3, 'MX-300', 2, 10, 300)
    tocadiscos[3] = new Typedisco(4, 'MX-200', 1, 11, 250)
    tocadiscos[4] = new Typedisco(5, 'MX-600', 3, 9, 400)

    let Objeto: any
    let dSchemaVinilo: iVinilo =
    {
        _ID: null,
        _nombre: null,
        _tamaño: null,
        _stock: null,
        _precio: null
    }
```

```
let Objeto2: any
let dShcmeaGramofono: igramofono = {

    _tipoObjeto: null,
    _ID: null,
    _velocidades: null,
    _corneta: null,
    _stock: null,
    _precio: null
}

let dSchemaTocadiscos: iTipotocadiscos = {

    _tipoObjeto: null,
    _ID: null,
    _velocidades: null,
    _modelo: null,
    _stock: null,
    _precio: null
}

await db.conectarBD()

for (let a of discos) {

    dSchemaVinilo._ID = a.ID
    dSchemaVinilo._nombre = a.nombre
    dSchemaVinilo._tamaño = a.tamaño
    dSchemaVinilo._stock = a.stock
    dSchemaVinilo._precio = a.precio
```

```
    Objeto = new Discos(dSchemaVinilo)
    console.log(Objeto)
    await Objeto.save()
}

for (let b of tocadiscos) {

    dShcmeaGramofono._ID = dSchemaTocadiscos._ID = b.ID
    dShcmeaGramofono._stock = dSchemaTocadiscos._stock = b.stock
    dShcmeaGramofono._precio = dSchemaTocadiscos._precio = b.precio

    if (b instanceof Gramofono) {
        dShcmeaGramofono._corneta = b.corneta
        dShcmeaGramofono._velocidades = b.velocidad
        dShcmeaGramofono._tipoObjeto = 'gram'
        console.log(Objeto2)
        Objeto2 = new Tocadisco(dShcmeaGramofono)
    }
    else if (b instanceof Typedisco) {
        dSchemaTocadiscos._modelo = b.modelo
        dSchemaTocadiscos._velocidades = b.velocidad
        dSchemaTocadiscos._tipoObjeto = 'tipo'
        console.log(Objeto2)
        Objeto2 = new Tocadisco(dSchemaTocadiscos)
    }
    await Objeto2.save()

}
```

Cliente_ident(ID)

```
export let Cliente_ident = async (ID: number) => {
    await db.conectarBD()
    let cliente: Cliente | any
    let Ocliente: any
    let ClienteSchema: iCliente = {
        _ID: null,
        _Edad: null,
        _Nombre: null,
        _carrito: null
    }
    let query: any = await Clientes.find({ _ID: ID })
    if (query.length == 0) {
        // No existe objeto con ese Id
        const Nombre = await leerTeclado('Escribe tu nombre')
        const Edad = parseInt(await leerTeclado('Dame tu edad'))
        ClienteSchema._ID = ID
        ClienteSchema._Edad = Edad
        ClienteSchema._Nombre = Nombre
        ClienteSchema._carrito = []
        cliente = new Cliente(ID, Nombre, Edad, [])
        Ocliente = new Clientes(ClienteSchema)
        await Ocliente.save()
        console.log('Cliente creado¡¡')
    } else {
        // Si existe objeto con ese Id
        for (let a of query) {
            cliente = new Cliente(a._ID, a._Nombre, a._Edad, ["vacio"])
            console.log(`tu id es: ${cliente.ID}`)
            console.log(`tu nombre es: ${cliente.nombre}`)
            console.log(`tu edad es: ${cliente.edad}`)
            console.log(`tu carrito es: ${cliente.carrito}`)
        }

    } return cliente
```

Cliente_ident(ID)

```
export let Cliente_ident = async (ID: number) => {
    await db.conectarBD()
    let cliente: Cliente | any
    let Ocliente: any
    let ClienteSchema: iCliente = {
        _ID: null,
        _Edad: null,
        _Nombre: null,
        _carrito: null
    }
    let query: any = await Clientes.find({ _ID: ID })
    if (query.length == 0) {
        // No existe objeto con ese Id
        const Nombre = await leerTeclado('Escribe tu nombre')
        const Edad = parseInt(await leerTeclado('Dame tu edad'))
        ClienteSchema._ID = ID
        ClienteSchema._Edad = Edad
        ClienteSchema._Nombre = Nombre
        ClienteSchema._carrito = []
        cliente = new Cliente(ID, Nombre, Edad, [])
        Ocliente = new Clientes(ClienteSchema)
        await Ocliente.save()
        console.log('Cliente creado¡¡')
    } else {
        // Si existe objeto con ese Id
        for (let a of query) {
            cliente = new Cliente(a._ID, a._Nombre, a._Edad, ["vacio"])
            console.log(`tu id es: ${cliente.ID}`)
            console.log(`tu nombre es: ${cliente.nombre}`)
            console.log(`tu edad es: ${cliente.edad}`)
            console.log(`tu carrito es: ${cliente.carrito}`)
        }

    } return cliente
```

bajar() bajar1()

```
export let bajar = async () => {
    await db.conectarBD()
    let query2: any = await Discos.find()
    for (let a of query2) {
        let disco = new Vinilo(a._ID, a._nombre, a._tamaño, a._stock, a._precio)
        disco.mostrar()
    }

}
export let bajar1 = async () => {
    db.conectarBD()
    let gramofono: Gramofono
    let tocadisco: Typedisco
    let query: any = await Tocadisco.find()
    for (let b of query) {
        if (b._tipoObjeto == 'gram') {
            gramofono = new Gramofono(b._ID, b._corneta, b._stock, b._precio)
            console.log(gramofono.mostrar())
        }
        if (b._tipoObjeto == 'tipo') {
            tocadisco = new Typedisco(b._ID, b._modelo, b._velocidades, b._stock, b._precio)
            console.log(tocadisco.mostrar())
        }
    }

}
```

AgregarCarrito(usuario)

```
export let AgregarCarrito = async (usuario: Cliente) => {
    db.conectarBD()
    let aVinilo: Vinilo
    let tocadiscos: Tocadiscos

    let ID: number = parseInt(await leerTeclado('dime el ID del objeto deseado'))
    let pregunt = await leerTeclado('Dime si es disco o maquina')
    switch (pregunt) {
        case 'disco':
            let query: any = await Discos.findOne({ _ID: ID })
            if (query._stock == 0) {
                console.log('Disculpa no nos quedan más')
            } else {
                aVinilo = new Vinilo(query._ID, query._nombre, query._tamaño, query._stock, query._precio)
                usuario.agregarCarrito(aVinilo)
                return query._precio
            }
            break
```

```
case 'maquina':
    let query1: any = await Tocadisco.findOne({ _ID: ID })
    if (query1._tipoObjeto == 'gram') {
        if (query1._stock == 0) {
            console.log('Disculpa no nos quedan más')
        } else {
            tocadiscos = new Gramofono(query1._ID, query1._corneta, query1._stock, query1._precio)
            usuario.agregarCarrito(tocadiscos)
            return query1._precio
        }
    }
    else if (query1._tipoObjeto == 'tipo') {
        if (query1._stock == 0) {
            console.log('Disculpa no nos quedan más')
        } else {
            tocadiscos = new Typedisco(query1._ID, query1._modelo, query1._stock, query1._velocidades, query1._precio)
            usuario.agregarCarrito(tocadiscos)
            return query1._precio
        }
    }

    break
```

borrarCarrito(usuario)

```
export let borrarCarrito = async (usuario: Cliente) => {
    let b: number = usuario.carrito.length - 1
    let a: number
    let c: number
    if (b == 0) {
        console.log('porfavor añade algo al carrito')
    }
    else {
        usuario.vercarrito()
        a = parseInt(await leerTeclado(`Dime cual quieres borrar desde el 1 al ${b}`))
        usuario.borrarCarrito(a)

    }
}
```

verCarrito(usuario)

```
export let verCarrito = async (usuario: Cliente) => {
    let b = usuario.comprobarcarrito()
    if (b == false) {
        console.log('porfavor añade algo al carrito')
    }
    else {
        usuario.vercarrito()
    }

}
```

realizarPedido(usuario, monedero)

```
export let realizarPedido = async (usuario: Cliente, monedero: number) => {

    let ipedido: iPedido = {
        _ID: null,
        _objetos: null,
        _fecha: null,
        _id_cliente: null,
        _precio_total: null
    }
    let b = usuario.carrito.length
    console.log(`${b}`)
    if (b != 1) {
        await db.conectarBD()
        ipedido._fecha = new Date()
        ipedido._ID = Math.trunc(Math.random() * (9999 - 1111) + 1111 - Math.random() + Math.random());
        ipedido._id_cliente = usuario.ID
        ipedido._objetos = usuario.carrito
        ipedido._precio_total = monedero
        let objet = new Pedidos1(ipedido)
        await objet.save()
        await actualizar(usuario)
        await usuario.eliminarcarrito()

    }
    else {
        console.log('Primero llena el carrito')
    }
    await db.desconectarBD()
}
```

cancelarPedido(usuario)

```
export let cancelarPedido = async (usuario: Cliente) => {
    await db.conectarBD()
    await show_pedidos_cliente(usuario)
    let n: number = parseInt(await (leerTeclado('id')))
    Pedidos1.findOneAndDelete({
        _ID: n,
    }, (err: any, doc: any) => {
        if (err) {

        } else {
            if (!doc) {

            } else {

            }
        }
    });
}
```

elegir0(usuario)

```
export let elegir0 = async (usuario: Cliente) => {
    let n: number | any
    let monedero = 0

    do {
        let n = await menu()
        switch (n) {
            case 1:
                console.log('Los discos')
                await bajar()
                await new Promise(f => setTimeout(f, 1000));
                console.log('Ahora las maquinas')
                await bajar1()
                break
            case 2:
                let c = await AgregarCarrito(usuario)

                monedero = await dinero_carrito(c, monedero)
                break
            case 3:
                await verCarrito(usuario)
                break
            case 4:
                await borrarCarrito(usuario)
                let d: number = parseInt(await leerTeclado('Dame el precio del objeto borrado'))
                monedero = await menos_dinero_carrito(d, monedero)
                break
```

```
            case 5:
                await realizarPedido(usuario, monedero)
                await actualizar(usuario)
                console.log(`Su pedido ha salido por ${monedero}`)
                break
            case 6:
                await cancelarPedido(usuario)
                break
            case 7:
                await show_pedidos_cliente(usuario)
                break
            case 8:
                console.log(monedero)
                break
        }
    } while (n != 0)
}
```

```
import { leerTeclado } from "./leerTeclado"
export const menu = async( ) => {
    let n: number
    console.log(" 1- Ver catálogo")
    console.log(" 2- Agregar carrito")
    console.log(" 3- Ver carrito")
    console.log(" 4- Borrar objeto del carrito")
    console.log(" 5- Realizar pedido")
    console.log(" 6- Cancelar Pedido")
    console.log(" 7- Ver pedidos")
    console.log(" 8- Ver precio")
    console.log(" 0 - exit")
    n = parseInt(await leerTeclado(''))
    return n
}
```

menu()