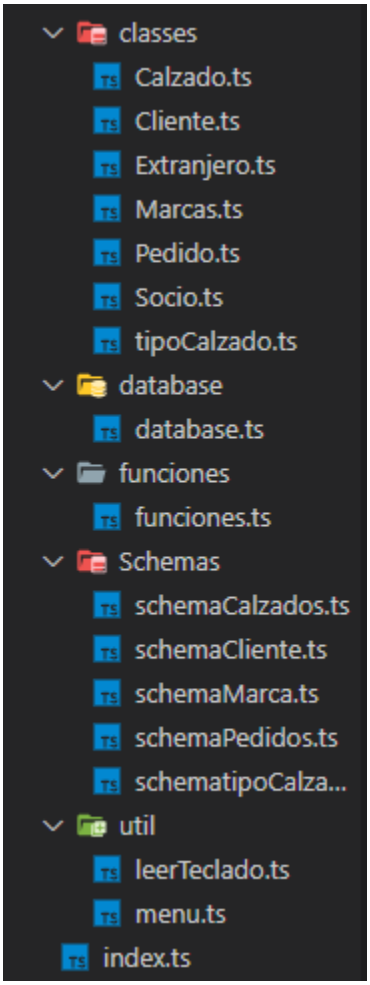
Proyecto Primer Trimestre
Gonzalo Pazos Sardella

Componentes del proyecto:

- Clases
- Conexión base de datos
- Funciones
- Schemas
- Utiles
- Index



Clases principales y subclases:

Clases principales:
Marca
Cliente
Pedido

Subclases:
Marca -> tipoCalzado -> Calzado
Cliente -> Extranjero | Socio

Marca:

```
export class Marca{
    protected _marca: string
    protected _precioExtra: number
    protected _descuento: number

    constructor(_marca: string, _precioExtra: number,_descuento : number){
        this._marca = _marca
        this._precioExtra = _precioExtra
        this._descuento = _descuento
    }

    public get marca(){
        return this._marca
    }

    public get precioExtra(){
        return this._precioExtra
    }

    public get descuento(){
        return this._descuento
    }

}
```

tipoCalzado:

```
import { Marca } from "./Marcas"

export class tiposCalzado extends Marca {
    protected _tipo : string

    constructor (_marca: string, _precioExtra: number, _descuento: number, _tipo: string){
        super(_marca, _precioExtra, _descuento)
        this._tipo = _tipo
    }

    public get tipo(){
        return this._tipo
    }

}
```

Calzado:

```
import { tiposCalzado } from "./tipoCalzado"

export class Calzado extends tiposCalzado {
    protected _talla: number
    protected _color: string
    protected _precio: number
    protected _cBarra: number


    constructor (_marca: string, _precioExtra: number, _descuento: number, _tipo: string, _talla: number, _color: string, _precio: number, _cBarra : number){
        super(_marca, _precioExtra, _descuento, _tipo)
        this._talla = _talla
        this._color = _color
        this._precio = _precio
        this._cBarra = _cBarra
    }

    public get talla(){
        return this._talla
    }

    public get color(){
        return this._color
    }

    public get precio(){
        return this._precio
    }

    get cBarra(){
        return this._cBarra
    }

    public calcularPrecio() {
        this._precio += this._precioExtra
        if(this._descuento == 0) {
            return this._precio;
        }
        return this._precio - Math.trunc(this._precio * this._descuento / 100)
    }

}
```

Cliente:

```
import { tiposCalzado } from "./tipoCalzado"

export class Calzado extends tiposCalzado {
    protected _talla: number
    protected _color: string
    protected _precio: number
    protected _cBarra: number


    constructor (_marca: string, _precioExtra: number, _descuento: number, _tipo: string, _talla: number, _color: string, _precio: number, _cBarra : number){
        super(_marca, _precioExtra, _descuento, _tipo)
        this._talla = _talla
        this._color = _color
        this._precio = _precio
        this._cBarra = _cBarra
    }

    public get talla(){
        return this._talla
    }

    public get color(){
        return this._color
    }

    public get precio(){
        return this._precio
    }

    get cBarra(){
        return this._cBarra
    }

    public calcularPrecio() {
        this._precio += this._precioExtra
        if(this._descuento == 0) {
            return this._precio;
        }
        return this._precio - Math.trunc(this._precio * this._descuento / 100)
    }

}
```

Extranjero:

```
import { Cliente } from "./Cliente"

export class Extranjero extends Cliente {
    protected _extranjero: boolean

    constructor (_DNI: number, _nombre: string, _apellidos: string, _pais: string, _extranjero: boolean){
        super (_DNI, _nombre, _apellidos, _pais)
        this._extranjero = _extranjero
    }

    public get extranjero(){
        return this._extranjero
    }

}
```

Socio:

```
import { Cliente } from "./Cliente"

export class Socio extends Cliente {
    protected _socio: boolean

    constructor (_DNI: number, _nombre: string, _apellidos: string, _pais: string, _socio: boolean){
        super (_DNI, _nombre, _apellidos, _pais)
        this._socio = _socio
    }

    public get Socio(){
        return this._socio
    }

}
```

Pedido:

```
export class Pedido{
    protected _nPedido: number
    protected _objetos: Array<any>
    protected _fecha: Date
    protected _idCliente: string
    protected _importeTotal: number

    constructor(_nPedido: number, _objetos: Array<any>, _fecha: Date, _idCliente: string, _importeTotal: number){
        this._nPedido = _nPedido
        this._objetos = _objetos
        this._fecha = _fecha
        this._idCliente = _idCliente
        this._importeTotal = _importeTotal
    }

    get nPedido(){
        return this._nPedido
    }

    get objetos(){
        return this._objetos
    }

    get fecha(){
        return this._fecha
    }

    get idCliente(){
        return this._idCliente
    }

    get importeTotal(){
        return this._importeTotal
    }

}
```

Schemas:

Schema Marca
Schema tipoCalzado
Schema Calzado

Schema Cliente

Schema Pedido

Schema Marca:

```
import { model } from 'mongoose'

var mongoose = require('mongoose');
var Schema = mongoose.Schema;
var ObjectIdSchema = Schema.ObjectId;
var ObjectId = mongoose.Types.ObjectId;


const marcaSchema = new Schema({
    _id: {
        type: ObjectIdSchema, default: function () { return new ObjectId()},
        auto: true
    },
    _marca : String,
    _precioExtra : Number,
    _descuento : Number,
    _tipo : String,
    _talla : Number,
    _color : String,
    _precio : Number
})

export type oMarca = {
    _id : null
    _marca : string | null
    _precioExtra : number | null
    _descuento : number | null
}

export const Marcas = model('marcas', marcaSchema)
```

Schema tipoCalzado:

```
import { model } from 'mongoose'

var mongoose = require('mongoose');
var Schema = mongoose.Schema;
var ObjectIdSchema = Schema.ObjectId;
var ObjectId = mongoose.Types.ObjectId;


const tipocalzadoSchema = new Schema({
    _id: {
        type: ObjectIdSchema, default: function () { return new ObjectId()},
        auto: true
    },
    _marca : String,
    _precioExtra: Number,
    _descuento: Number,
    _tipo : String
})

export type otipoCalzado = {
    _id : null
    _marca : string | null
    _precioExtra : number | null
    _descuento : number | null
    _tipo : string | null
}

export const tiposCalzados = model('tiposCalzados', tipocalzadoSchema)
```

Schema Calzado:

```
import { model } from 'mongoose'

var mongoose = require('mongoose');
var Schema = mongoose.Schema;
var ObjectIdSchema = Schema.ObjectId;
var ObjectId = mongoose.Types.ObjectId;


const calzadosSchema = new Schema({
    _id: {
        type: ObjectIdSchema, default: function () { return new ObjectId()},
        auto: true
    },
    _marca: String,
    _tipo: String,
    _talla: Number,
    _color: String,
    _precio: Number,
    _cBarra: Number
})

export type oCalzado = {
    _id: null
    _tipo: string | null
    _marca: string | null
    _precioExtra: number | null
    _descuento: number | null
    _talla: number | null
    _color: string | null
    _precio: number | null
    _cBarra: number | null
}

export const Calzados = model('Calzados', calzadosSchema)
```

Schema Cliente:

```
import { model } from 'mongoose'

var mongoose = require('mongoose');
var Schema = mongoose.Schema;
var ObjectIdSchema = Schema.ObjectId;
var ObjectId = mongoose.Types.ObjectId;

const clienteSchema = new Schema({
    _id: {
        type: ObjectIdSchema, default: function () { return new ObjectId()},
        auto: true
    },
    _DNI : Number,
    _nombre : String,
    _apellidos : String,
    _pais : String,
    _extranjero : Boolean,
    _socio : Boolean
})

export type iCliente = {
    _id : null
    _DNI : number | null
    _nombre : string | null
    _apellidos : string | null
    _pais : string | null
}

export type iExtranjero = {
    _id : null
    _DNI : number | null
    _nombre : string | null
    _apellidos : string | null
    _pais : string | null
    _extranjero : boolean | null
}

export type iSocio = {
    _id : null
    _DNI : number | null
    _nombre : string | null
    _apellidos : string | null
    _pais : string | null
    _socio : boolean | null
}

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

Schema Pedido:

```
import { model } from 'mongoose'

var mongoose = require('mongoose');
var Schema = mongoose.Schema;
var ObjectIdSchema = Schema.ObjectId;
var ObjectId = mongoose.Types.ObjectId;


const pedidoSchema = new Schema({
    _id: {
        type: ObjectIdSchema, default: function () { return new ObjectId()},
        auto: true
    },
    _nPedido: Number,
    _objetos:{
        type: Array
    },
    _fecha: Date,
    _idCliente: String,
    _importeTotal: Number
})

export type iPedido = {
    _ID: number | null
    _nPedido: number | null
    _objetos: Array<any> | null
    _fecha: Date | null
    _idCliente: string | null
    _importeTotal: number | null
}

export const Pedidos = model ('Pedidos', pedidoSchema)
```

Funciones:

- Crear cliente
- Crear marca
- Creación tipo calzado
- Creación calzado
- Calzados disponibles
- Busqueda cliente
- Busqueda calzado
- Borrar cliente
- Borrar calzado
- Crear pedido
- Borrar pedido
- Búsqueda de pedido

Creación Objetos:

Crear cliente:

```
export let creacionCliente = async () => {
  await db.conectarBD()
  let cliente: Cliente | any

  let sSchema: any
  let oSchema: iExtranjero = {
    _id: null,
    _DNI: null,
    _nombre: null,
    _apellidos: null,
    _pais: null,
    _extranjero: null
  }
  let ooSchema: iSocio = {
    _id: null,
    _DNI: null,
    _nombre: null,
    _apellidos: null,
    _pais: null,
    _socio: null
  }

  const DNI = parseInt(await leerTeclado("DNI"))
  const Nombre = await leerTeclado("Nombre")
  const Apellidos = await leerTeclado("Apellido")
  const pais = await leerTeclado("País")
  const extranjero = await leerTeclado("Eres extranjero S/N")
```

```
  if (extranjero == "S") {
    let extranjero = true
    let acliente = new Extranjero(DNI, Nombre, Apellidos, pais, extranjero)
    oSchema._DNI = acliente.DNI
    oSchema._nombre = acliente.Nombre
    oSchema._apellidos = acliente.Apellidos
    oSchema._pais = acliente.Pais
    oSchema._extranjero = acliente.extranjero
    sSchema = new Clientes(oSchema)
    await sSchema.save()
  } else {
    const socio = await leerTeclado("Quieres ser socio S/N")
    if (socio == "S") {
      let socio = true
      let bcliente = new Socio(DNI, Nombre, Apellidos, pais, socio)
      ooSchema._DNI = bcliente.DNI
      ooSchema._nombre = bcliente.Nombre
      ooSchema._apellidos = bcliente.Apellidos
      ooSchema._pais = bcliente.Pais
      ooSchema._socio = bcliente.Socio
      sSchema = new Clientes(ooSchema)
      await sSchema.save()
    } else {
      let socio = false
      let bcliente = new Socio(DNI, Nombre, Apellidos, pais, socio)
      ooSchema._DNI = bcliente.DNI
      ooSchema._nombre = bcliente.Nombre
      ooSchema._apellidos = bcliente.Apellidos
      ooSchema._pais = bcliente.Pais
      ooSchema._socio = bcliente.Socio
      sSchema = new Clientes(ooSchema)
      await sSchema.save()
    }
  } return cliente
}
```

Crear marca:

```
export let creacionMarca = async () => {
  await db.conectarBD()
  let amarca: Marca | any

  let sSchema: any
  let oSchema: oMarca = {
    _id : null,
    _marca : null,
    _precioExtra : null,
    _descuento : null
  }

  console.log("Marcas disponibles: Nike, Adidas, Balenciaga o Puma")
  const marca = await leerTeclado("Marca")
  console.log("Deme un precio Extra")
  const precioExtra = parseFloat(await leerTeclado("Precio extra"))
  console.log("Deme un descuento")
  const descuento = parseInt(await leerTeclado("Descuento"))

  oSchema._marca = marca;
  oSchema._precioExtra = precioExtra;
  oSchema._descuento = descuento;
  sSchema = new Marcas(oSchema)
  await sSchema.save()
  return amarca
}
```

Crear tipoCalzado:

```
export let creaciontipoCalzado = async () => {
  await db.conectarBD()
  let aTipoCalzado: tiposCalzado | any

  let sSchema: any
  let oSchema: otipoCalzado = {
    _id : null,
    _marca : null,
    _precioExtra : null,
    _descuento : null,
    _tipo : null
  }

  const tipo = await leerTeclado("Ingrese un tipo de calzado")
  const marca = await leerTeclado("Ingrese una marca")
  let query: any = await Marcas.findOne({ _marca : marca })
  if (query != null) {
    const precioExtra = query._precioExtra
    const descuento = query._descuento

    let aTipoCalzado = new tiposCalzado(marca,precioExtra,descuento,tipo)
    oSchema._marca = aTipoCalzado.marca
    oSchema._precioExtra = aTipoCalzado.precioExtra
    oSchema._descuento = aTipoCalzado.descuento
    oSchema._tipo = aTipoCalzado.tipo
    sSchema = new tiposCalzados(oSchema)
    await sSchema.save()
  } else {
    console.log("Ingrese una marca valida")
  } return aTipoCalzado
}
```

Crear calzado:

```
export let creacionCalzado = async () => {
  await db.conectarBD()
  let aCalzado: Calzado | any

  let sSchema: any
  let oSchema: oCalzado = {
    _id : null,
    _tipo : null,
    _marca : null,
    _precioExtra : null,
    _descuento : null,
    _talla : null,
    _color : null,
    _precio : null,
    _cBarra : null
  }

  const marca = await leerTeclado("Ingrese marca")
  const tipo = await leerTeclado("Ingrese tipo")
  let query: any = await tiposCalzados.findOne({ "$and":[{ _tipo : tipo }, { _marca : marca}]})
  if (query != null) {
      const precioExtra = query._precioExtra
      const descuento = query._descuento
      const marca = query._marca
      const talla = parseInt(await leerTeclado("Ingrese la talla del calzado"))
      const color = await leerTeclado("Ingrese color para el calzado")
      const precio = parseInt(await leerTeclado("Ingrese un precio para el calzado"))
      const cBarra = parseInt(await leerTeclado("Ingrese el codigo de barras"))
      let aCalzado = new Calzado(marca,precioExtra,descuento,tipo,talla,color,precio,cBarra)
      oSchema._precioExtra = aCalzado.precioExtra
      oSchema._descuento = aCalzado.descuento
      oSchema._marca = aCalzado.marca
      oSchema._talla = aCalzado.talla
      oSchema._color = aCalzado.color
      oSchema._tipo = aCalzado.tipo
      oSchema._precio = aCalzado.calcularPrecio()
      oSchema._cBarra = aCalzado.cBarra
      sSchema = new Calzados(oSchema)
      await sSchema.save()
  } else {
      console.log("Ingrese un tipo valido")
  } return aCalzado
}
```

Busquedas:

Calzado disponible:

```
export let calzadosDisponibles = async () => {
  await db.conectarBD()
  let query: any = await Calzados.find()
  if (query != null) {
    console.log(query)
  } else {
    console.log("No existen productos")
  }
}
```

Busqueda cliente:

```
export let busquedaCliente = async () => {
  await db.conectarBD()
  const DNI = parseInt(await leerTeclado("Ingrese su DNI"))
  let query: any = await Clientes.findOne({ _DNI: DNI })
  if (query != null) {
    console.log(query)
  } else {
    console.log("Este DNI no existe")
  }
}
```

Busqueda calzado:

```
export let busquedaCalzado = async () => {
  await db.conectarBD()
  const cBarra = parseInt(await leerTeclado("Ingrese codigo de barras"))
  let query: any = await Calzados.findOne({ _cBarra: cBarra })
  if (query != null) {
    console.log(query)
  } else {
    console.log("Este codigo de barras no existe")
  }
}
```

Eliminar objetos:

Borrar cliente:

```
export let borrarCliente = async () => {
  await db.conectarBD()
  const DNI = parseInt(await leerTeclado("Ingrese su DNI"))
  let query: any = await Clientes.findOne({ _DNI: DNI })
  console.log(query)
  if (query != null) {
    console.log(query._id)
    // Borro utilizando el campo ID, ya que existe el indice creado por mongoose para _id
    Clientes.deleteOne({ _id: query._id }).then(function () {
      console.log("Objeto borrado correctamente")
    }).catch(function (error) {
      console.log(error)
    })
  } else {
    console.log("Ese DNI no existe")
  }
}
```

Borrar calzado:

```
export let borrarCalzado = async () => {
  await db.conectarBD()
  const cBarra = parseInt(await leerTeclado("Ingrese codigo de barras"))
  let query: any = await Calzados.findOne({ _cBarra: cBarra })
  console.log(query)
  if (query != null) {
    console.log(query._id)
    // Borro utilizando el campo ID, ya que existe el indice creado por mongoose para _id
    Calzados.deleteOne({ _id: query._id }).then(function () {
      console.log("Objeto borrado correctamente")
    }).catch(function (error) {
      console.log(error)
    })
  } else {
    console.log("Ese DNI no existe")
  }
}
```

Crear pedido:

```
export let crearPedido = async () => {
  await db.conectarBD()
  let aPedido : Pedido | any

  let sSchema: any
  let oSchema: iPedido = {
    _ID: null,
    _nPedido: null,
    _objetos: [],
    _fecha: null,
    _idCliente: null,
    _importeTotal: null
  }
```

```
const DNI = parseInt(await leerTeclado("Ingrese su DNI"))
let queryC: any = await Clientes.findOne({ _DNI: DNI })
if (queryC != null){
  let objetos = []
  let importeTotal = 0
  while(true){
    const cBarra = await leerTeclado("Ingrese el codigo de barra del producto (0 para terminar)")
    let query: any = await Calzados.findOne({ _cBarra: cBarra })
    if(cBarra != "0"){
      if (query != null){
        const cantidad = parseInt(await leerTeclado("Ingrese la cantidad de este producto"))
        objetos.push({ "objectID": query._id, "Marca": query._marca , "Tipo": query._tipo, "Talla": query._talla, "Color": query._color ,"Precio": query._precio, "Cantidad": cantidad })
        importeTotal += query._precio * cantidad
      } else {
        console.log("Objeto inexistente, ingrese un válido")
      }
    } else {
      const con = await leerTeclado("¿Quiere confirmar el pedido? S/N")
      if(con == "S"){
        const nPedido = Math.floor(Math.random() * (10000 - 1 + 1)) + 1;
        const fecha = new Date()
        const idCliente = queryC._id
        let aPedido = new Pedido(nPedido,objetos,fecha,idCliente,importeTotal)
        const darnPedido = await leerTeclado("Tu numero de pedido es: " + nPedido)
        oSchema._nPedido = aPedido.nPedido
        oSchema._objetos = aPedido.objetos
        oSchema._fecha = aPedido.fecha
        oSchema._idCliente = aPedido.idCliente
        oSchema._importeTotal = aPedido.importeTotal
        sSchema = new Pedidos(oSchema)
        await sSchema.save()
      } else {
        console.log("Pedido no creado")
      }
      break
    }
  }
}
}
```

Borrar pedido:

```
export let borrarPedido = async () => {
  await db.conectarBD()
  const nPedido = parseInt(await leerTeclado("Ingrese el numero de pedido"))
  let query: any = await Pedidos.findOne({ _nPedido: nPedido })
  console.log(query)
  if (query != null) {
    console.log(query._id)
    // Borro utilizando el campo ID, ya que existe el indice creado por mongoose para _id
    Pedidos.deleteOne({ _id: query._id }).then(function () {
      console.log("Objeto borrado correctamente")
    }).catch(function (error) {
      console.log(error)
    })
  } else {
    console.log("Ese pedido no existe")
  }
}
```

Busqueda pedido:

```
export let busquedaPedido = async () => {
  await db.conectarBD()
  const DNI = parseInt(await leerTeclado("Ingrese su DNI"))
  let query: any = await Clientes.findOne({ _DNI: DNI })
  if (query != null) {
    const tipoCon = await leerTeclado("¿Quiere ver un pedido especifico o todos? E/T")
    if (tipoCon == "T"){
      let query: any = await Pedidos.find()
      console.log(query)
    } else {
      const nPedido = parseInt(await leerTeclado("Ingrese numero de pedido"))
      let queryP: any = await Pedidos.findOne({ _nPedido : nPedido })
      if (queryP != null){
        console.log(queryP)
      }else {
        console.log("El pedido no existe")
      }
    }
  } else {
    console.log("Este DNI no existe")
  }
}
```

DateBase:

```
import mongoose from 'mongoose';

class DataBase {

    private _cadenaConexion: string = 'mongodb+srv://gonzalo:12345@proyecto.mtaer.mongodb.net/Proyecto?retryWrites=true&w=majority'
    constructor(){

    }
    set cadenaConexion(_cadenaConexion: string){
        this._cadenaConexion = _cadenaConexion
    }

    conectarBD = async () => {
        const promise = new Promise<string>( async (resolve, reject) => {
            await mongoose.connect(this._cadenaConexion, {
            })
            .then( () => resolve(`Conectado a ${this._cadenaConexion}`) )
            .catch( (error) => reject(`Error conectando a ${this._cadenaConexion}: ${error}`) )
        })
        return promise
    }

    desconectarBD = async () => {
        const promise = new Promise<string>( async (resolve, reject) => {
            await mongoose.disconnect()
            .then( () => resolve(`Desconectado de ${this._cadenaConexion}`) )
            .catch( (error) => reject(`Error desconectando de ${this._cadenaConexion}: ${error}`) )
        })
        return promise
    }
}

export const db = new DataBase()
```

Utiles:

```
import { leerTeclado } from '../util/leerTeclado'

export const menu = async () => {
console.log ("1- Crear cliente")
console.log ("2- Crear marca")
console.log ("3- Creacion tipo calzado")
console.log ("4- Creacion calzado")
console.log ("5- Calzados disponibles")
console.log ("6- Busqueda cliente")
console.log ("7- Busqueda calzado")
console.log ("8- Borrar cliente")
console.log ("9- Borrar calzado")
console.log ("10- Crear pedido")
console.log ("11- Borrar pedido")
console.log ("12- Busqueda de pedido")
console.log (" ")
let opcion = await leerTeclado(``)
console.log(" ")
return opcion
}
```

Index:

```
let main = async () => {

  while(1) {
  let opcion = parseInt(await menu())

  switch (opcion) {
    case 1:
      await creacionCliente()
    break;
    case 2:
      await creacionMarca()
    break;
    case 3:
      await creaciontipoCalzado()
    break;
    case 4:
      await creacionCalzado()
    break;
    case 5:
      await calzadosDisponibles()
    break;
    case 6:
      await busquedaCliente()
    break;
    case 7:
      await busquedaCalzado()
    break;
    case 8:
      await borrarCliente()
    break;
    case 9:
      await borrarCalzado()
    break;
    case 10:
      await crearPedido()
    break;
    case 11:
      await borrarPedido()
    break;
    case 12:
      await busquedaPedido()
    break;

  }

  console.log("")
  console.log(`Pulse enter para continuar`)
  await leerTeclado ('')
  console.log(` `)
}
}

main()
```