Universidad ORT Uruguay

Facultad de Ingeniería

Escuela de Tecnología

OBLIGATORIO PROGRAMACIÓN 2

DOCUMENTO DE ANÁLISIS

Joaquín García - 219732

David de Araujo - 207813

M2B

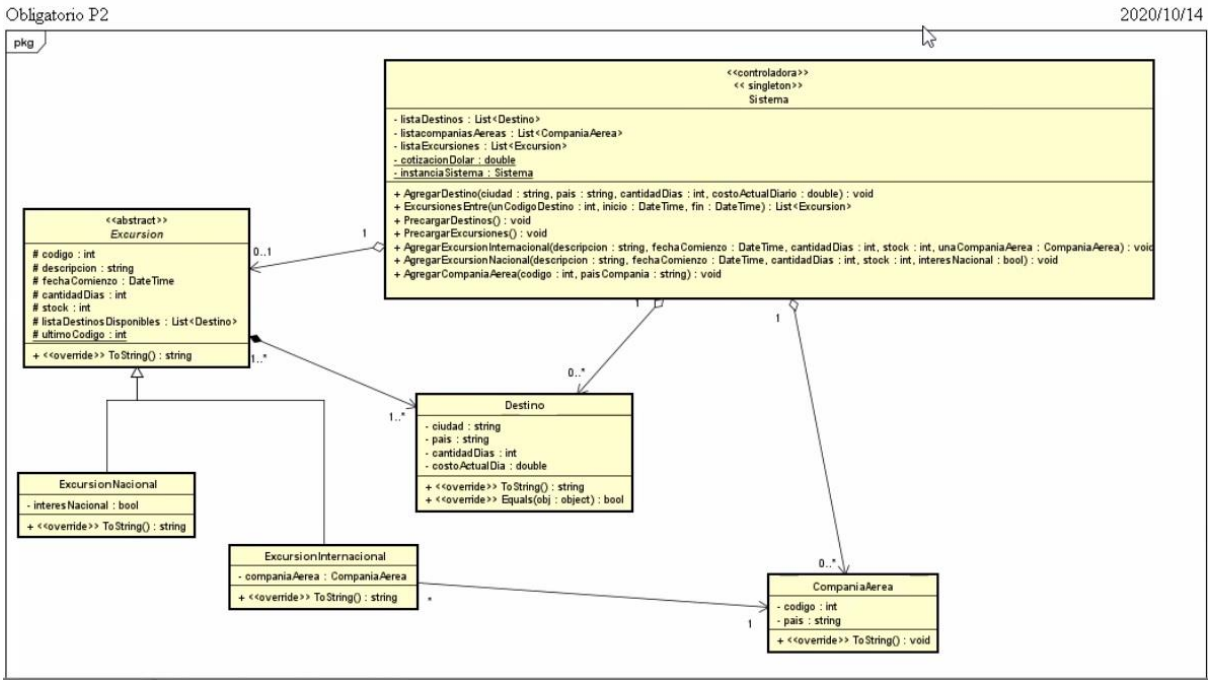
Docente: Francisco Bouza

Analista en TI/P

15/10/2020

Índice

1. Diagrama UML



2. Código Dominio

2.1 Excursion

```
public abstract class Excursion
    {

        protected static int ultimoCodigo = 1000;
        protected int codigo;
        protected string descripcion;
        protected DateTime fehcaComienzo;
        protected int cantidadDias;
        protected int stock;
        protected List<Destino> listaDestinosDisponibles;



        public Excursion(string descripcion, DateTime fehcaComienzo, int cantidadDias,
int stock)
```

```
{
    this.codigo = ultimoCodigo;
    this.descripcion = descripcion;
    this.fehcaComienzo = fehcaComienzo;
    this.cantidadDias = cantidadDias;
    this.stock = stock;
    this.listaDestinosDisponibles = new List<Destino>();
    ultimoCodigo+=100;
}

#region Propiedades


public string Descripcion
{
    get { return descripcion; }
    set { descripcion = value; }
}


public DateTime FechaComienzo
{
    get { return fehcaComienzo; }
    set { fehcaComienzo = value; }
}


public int CantidadDias
{
    get { return cantidadDias; }
    set { cantidadDias = value; }
}



public int Stock
{
    get { return stock; }
    set { stock = value; }
}
public List<Destino> ListaDestinosDisponibles
{
    get { return listaDestinosDisponibles; }
    //set { destinosDisponibles = value; }
}
#endregion

public override string ToString()
{
    //calculo costo total
    double costoTotal=0;
    string destinosDisponibles = "";
    foreach (Destino i in this.ListaDestinosDisponibles)
    {
        costoTotal += (i.CantidadDias * i.CostoActualDiario);
    }
    //Muestro todos los destinos
    foreach (Destino i in this.ListaDestinosDisponibles)
    {
        destinosDisponibles += $"{i.Ciudad}, {i.Pais}; ";

    }
```

```
            return $"Codigo: {codigo}, Desscripcion: {descripcion}, Fecha comienzo:
{fehcaComienzo}, Cantidad de dias: {cantidadDias}, Stock: {stock}, Costo
total(dolares): {costoTotal}, Costo total(pesos): {costoTotal *
Sistema.CotizacionDolar}, Destinos : {destinosDisponibles} ";
        } }
```

2.2 ExcursionInternacional

```
public class ExcursionInternacional : Excursion
    {
        #region Variables
        private CompaniaAerea companiaAerea;
        #endregion


        //CompaniaAerea Aerea tiene set y get esta ok eso
        #region Constructor
        public ExcursionInternacional(string descripcion, DateTime fehcaComienzo, int
cantidadDias, int stock, CompaniaAerea companiaAerea) : base(descripcion,
fehcaComienzo, cantidadDias, stock)
        {
            this.companiaAerea = companiaAerea;
        }
        #endregion

        #region Propiedades
        public CompaniaAerea CompaniaAerea
        {
            get { return companiaAerea; }
            set { companiaAerea = value; }
        }
        #endregion

        public override string ToString()
        {

            return base.ToString() + $"\nCompania aerea: {CompaniaAerea}";
        }
    }
```

2.3 ExcursionNacional

```
public class ExcursionNacional : Excursion
    {

        #region Variables
        private bool interesNacional;
        #endregion

        #region Constructor
        public ExcursionNacional(string descripcion, DateTime fehcaComienzo, int
cantidadDias, int stock, bool
interesNacional):base(descripcion,fehcaComienzo,cantidadDias,stock)
        {
            this.interesNacional = interesNacional;
```

```
        }
        #endregion

        #region Propiedades
        public bool InteresNacional
        {
            get { return interesNacional; }
            set { interesNacional = value; }
        }
        #endregion

        public override string ToString()
        {
            string esDeInteresNacional = "No";
            if (interesNacional)
            {
                esDeInteresNacional = "Si";
            }
            return base.ToString() + $", {esDeInteresNacional} es de interes
nacional";
        }
    }
```

2.4 CompaniaAerea

```
public class CompaniaAerea
    {

        #region Variables
        private int codigo;
        private string pais;
        #endregion

        #region Constructor
        public CompaniaAerea(int codigo, string pais)
        {
            this.codigo = codigo;
            this.pais = pais;
        }
        #endregion



        #region Propiedades
        public int Codigo
        {
            get { return codigo; }
            set { codigo = value; }
        }

        public string Pais
        {
            get { return pais; }
            set { pais = value; }
        }
        #endregion
```

```
        public override string ToString()
        {
            return $"codigo: {codigo}, Pais origen de la compania: {pais}";
        }
    }
```

2.5 Destino

```
public class Destino
    {
        #region Variables
        private string ciudad;
        private string pais;
        private int cantidadDias;
        private double costoActualDiario;

        #endregion

        #region Constructor
        public Destino(string ciudad, string pais, int cantidadDias, double
costoActualDiario)
        {
            this.ciudad = ciudad;
            this.pais = pais;
            this.cantidadDias = cantidadDias;
            this.costoActualDiario = costoActualDiario;
        }
        #endregion

        #region Propiedades
        public string Ciudad
        {
            get { return ciudad; }
            set { ciudad = value; }
        }


        public string Pais
        {
            get { return pais; }
            set { pais = value; }
        }


        public int CantidadDias
        {
            get { return cantidadDias; }
            set { cantidadDias = value; }
        }


        public double CostoActualDiario
        {
            get { return costoActualDiario; }
            set { costoActualDiario = value; }
        }
        #endregion
```

```
        #region Equals y ToString
        public override bool Equals(object obj)
        {
            if(obj == null || !(obj is Destino))
            {
                return false;
            }
            else
            {

                return this.ciudad == ((Destino)obj).Ciudad && this.Pais ==
((Destino)obj).Pais;
            }

        }


        public override string ToString()
        {


            //falta agregar la cotizacion del dolar en algun lado y poner la variable
aca al final
            return $"Ciudad: {ciudad}, Pais: {pais}, Cantidad de dias: {cantidadDias},
" +
                $"Costo actual diario: {CostoActualDiario}, Costo por
persona(dolares): {cantidadDias*CostoActualDiario}, " +
                $"Costo por persona(pesos): {cantidadDias * CostoActualDiario *
Sistema.CotizacionDolar}";
        }
        #endregion
    }
```

2.6 Sistema

```
public class Sistema
    {
        #region Variables y Listas
        private static double cotizacionDolar = 43.2;
        private static Sistema instanciaSistema;
        private List<Destino> listaDestinos;
        private List<CompaniaAerea> listaCompaniasAereas;
        private List<Excursion> listaExcursiones;
        #endregion

        #region Constructor
        private Sistema()
        {
            this.listaDestinos = new List<Destino>();
            this.listaCompaniasAereas = new List<CompaniaAerea>();
            this.listaExcursiones = new List<Excursion>();
        }
        #endregion

        #region Propiedades
        public List<Destino> ListaDestinos
```

```
        {
            get { return listaDestinos; }
            //set { listaDestinos = value; }
        }

        public List<Excursion> ListaExcursiones
        {
            get { return listaExcursiones; }
            //set { listaExcursiones = value; }
        }

        public List<CompaniaAerea> ListaCompaniasAereas
        {
            get { return listaCompaniasAereas; }
            //set { listaCompaniasAereas = value; }
        }



        public static double CotizacionDolar
        {
            get { return cotizacionDolar; }
            set { cotizacionDolar = value; }
        }




        public static Sistema InstanciaSistema
        {

            get
            {
                if (instanciaSistema == null)
                {
                    instanciaSistema = new Sistema();
                }
                return instanciaSistema;
            }

        }

        #endregion


        #region Metodos

        public void AgregarDestino(string ciudad, string pais, int cantidadDias,
double costoActualDiario)
        {
            Destino unDestino = new Destino(ciudad, pais, cantidadDias,
costoActualDiario);
            listaDestinos.Add(unDestino);
        }

        public void PrecargarDestinos()
        {
            AgregarDestino("Montevideo", "Uruguay", 3, 120);
            AgregarDestino("Canelones", "Uruguay", 3, 120);
            AgregarDestino("Rivera", "Uruguay", 3, 120);
            AgregarDestino("Paysandu", "Uruguay", 3, 120);
```

```
            AgregarDestino("Brasilia", "Brasil", 3, 120);
            AgregarDestino("Cuzco", "Peru", 3, 120);
            AgregarDestino("Sucre", "Bolivia", 3, 120);
            AgregarDestino("Santiago de Chile", "Chile", 3, 120);
            AgregarDestino("Hanoi", "Vietnam", 3, 120);
            AgregarDestino("Canberra", "Australia", 3, 120);
        }

        public void AgregarExcursionNacional(string descripcion, DateTime
fechaComienzo, int cantidadDias, int stock, bool interesNacional)
        {
            Excursion unaExursion = new ExcursionNacional(descripcion, fechaComienzo,
cantidadDias, stock, interesNacional);
            listaExcursiones.Add(unaExursion);
        }

        public void AgregarExcursionInternacional(string descripcion, DateTime
fechaComienzo, int cantidadDias, int stock, CompaniaAerea unaCompaniaAerea)
        {
            Excursion unaExursionInternacional = new
ExcursionInternacional(descripcion, fechaComienzo, cantidadDias, stock,
unaCompaniaAerea);
            listaExcursiones.Add(unaExursionInternacional);
        }
        public void AgregarCompaniaAerea(int codigo, string paisCompania)
        {
            CompaniaAerea unaCompania = new CompaniaAerea(codigo, paisCompania);
            listaCompaniasAereas.Add(unaCompania);
        }
        public void PrecargarExcursiones()
        {
            //Agrego una compania aerea
            AgregarCompaniaAerea(22, "Urguay");
            //CompaniaAerea unaCompaniaAerea = new CompaniaAerea(22, "Urguay");
            CompaniaAerea unaCompaniaAerea = listaCompaniasAereas[0];

            //Agrego excursiones nacionales

            AgregarExcursionNacional("una descripcion", new DateTime(2020, 09, 26), 5,
20, true);
            listaExcursiones[0].ListaDestinosDisponibles.Add(listaDestinos[0]);
            listaExcursiones[0].ListaDestinosDisponibles.Add(listaDestinos[1]);

            AgregarExcursionNacional("una descripcion", new DateTime(2020, 09, 26), 5,
20, true);
            listaExcursiones[1].ListaDestinosDisponibles.Add(listaDestinos[1]);
            listaExcursiones[1].ListaDestinosDisponibles.Add(listaDestinos[2]);

            AgregarExcursionNacional("una descripcion", new DateTime(2020, 09, 26), 5,
20, true);
            listaExcursiones[2].ListaDestinosDisponibles.Add(listaDestinos[0]);
            listaExcursiones[2].ListaDestinosDisponibles.Add(listaDestinos[2]);

            AgregarExcursionNacional("una descripcion", new DateTime(2020, 09, 26), 5,
20, true);
            listaExcursiones[3].ListaDestinosDisponibles.Add(listaDestinos[1]);
            listaExcursiones[3].ListaDestinosDisponibles.Add(listaDestinos[3]);

            //Agrego excursiones internacionales

            AgregarExcursionInternacional("una descripcion", new DateTime(2020, 09,
26), 5, 20, unaCompaniaAerea);
```

```
            listaExcursiones[4].ListaDestinosDisponibles.Add(listaDestinos[7]);
            listaExcursiones[4].ListaDestinosDisponibles.Add(listaDestinos[8]);

            AgregarExcursionInternacional("una descripcion", new DateTime(2020, 09,
20), 5, 20, unaCompaniaAerea);
            listaExcursiones[5].ListaDestinosDisponibles.Add(listaDestinos[9]);
            listaExcursiones[5].ListaDestinosDisponibles.Add(listaDestinos[4]);
            AgregarExcursionInternacional("una descripcion", new DateTime(2020, 09,
26), 5, 20, unaCompaniaAerea);
            listaExcursiones[6].ListaDestinosDisponibles.Add(listaDestinos[8]);
            listaExcursiones[6].ListaDestinosDisponibles.Add(listaDestinos[5]);
            AgregarExcursionInternacional("una descripcion", new DateTime(2020, 09,
26), 5, 20, unaCompaniaAerea);
            listaExcursiones[7].ListaDestinosDisponibles.Add(listaDestinos[7]);
            listaExcursiones[7].ListaDestinosDisponibles.Add(listaDestinos[6]);




        }

        //podria hacer un metodo que devuelva un lista de las excursiones que esten
entre esas dos fechas asi no tiene consolewr
        public  List<Excursion> ExcursionesEntre(int unCodigoDestino, DateTime inicio,
DateTime fin)
        {
            List<Excursion> listaBuscada = new List<Excursion>();

            foreach (Excursion i in listaExcursiones)
            {
                if (i.FechaComienzo > inicio && i.FechaComienzo < fin)
                {

                    foreach (Destino j in i.ListaDestinosDisponibles)
                    {

                        if (unCodigoDestino - 1 == listaDestinos.IndexOf(j))
                        {
                            listaBuscada.Add(i);
                        }
                    }

                }

            }

            return listaBuscada;
        }
        #endregion

    }
```

2.6 Funciones Auxiliares

```
public class FunAux
    {

        //Pedir un numero que sea mayor a un minimo
        public static int PedirNumeroMayor(string msg, string errMsg, int min)
```

```
        {
            bool exito;
            int num;
            do
            {
                Console.WriteLine(msg);
                string strNum = Console.ReadLine();
                exito = int.TryParse(strNum, out num) && num > min;
                if (!exito)
                    Console.WriteLine(errMsg);
            } while (!exito);
            return num;

        }

        //Pedir un numero que sea mayor a un minimo con decimales
        public static double PedirNumeroDoubleMayor(string msg, string errMsg, int
min)
        {
            bool exito;
            double num;
            do
            {
                Console.WriteLine(msg);
                string strNum = Console.ReadLine();
                exito = double.TryParse(strNum, out num) && num > min;
                if (!exito)
                    Console.WriteLine(errMsg);
            } while (!exito);
            return num;

        }
        //Pedir un texto que sea mayor a un minimo
        public static string PedirTextoMayor(string msg, string errMsg, int min)
        {
            bool exito;
            string str;
            do
            {
                Console.WriteLine(msg);
                 str = Console.ReadLine();
                exito =  str.Length >= min;
                if (!exito)
                    Console.WriteLine(errMsg);
            } while (!exito);

            return str;

        }

        //Pedir un numero que este entre otros dos
        public static int PedirNumero(string msg, string errMsg, int min, int max)
        {
            bool exito;
            int num;
            do
            {
                Console.WriteLine(msg);
                string strNum = Console.ReadLine();
                exito = int.TryParse(strNum, out num) && num >= min && num <= max;
                if (!exito)
                    Console.WriteLine(errMsg);
```

```
            } while (!exito);
            return num;

        }

        //Pedir una fecha en formato dia/mes/año
        public static DateTime PedirFecha(string msg, string errMsg)
        {
            bool exito;
            DateTime fecha;
            do
            {
                Console.WriteLine(msg);
                string strNum = Console.ReadLine();
                exito = DateTime.TryParseExact(strNum,"dd/MM/yyyy", null,
System.Globalization.DateTimeStyles.None, out fecha);
                if (!exito)
                    Console.WriteLine(errMsg);
            } while (!exito);
            return fecha;

        }
    }
```

3. Datos precargados

3.1 Destinos

Destinos

Destino	Ciudad	País	Cantidad de Días	Costo Diario
1	Montevideo	Uruguay	3	120
2	Canelones	Uruguay	3	120
3	Rivera	Uruguay	3	120
4	Paysandú	Uruguay	3	120
5	Brasilia	Brasil	3	120
6	Cuzo	Perú	3	120
7	Sucre	Bolivia	3	120
8	Santiago de chile	Chile	3	120
9	Hanoi	Vietnam	3	120
10	Canberra	Australia	3	120

3.1 Excursiones nacionales

Código	Descripción	Fecha inicio	cantidad días de traslados	stock	Interés Nacional
1000	una descripción	26/09/2020	5	20	si
1100	una descripción	26/09/2020	5	20	si
1200	una descripción	26/09/2020	5	20	si
1300	una descripción	26/09/2020	5	20	si

3.1 Excursiones internacionales

					Compañía Aérea	
Código	Descripción	Fecha inicio	cantidad días de traslados	stock	código	país
1400	una descripción	26/09/2020	5	20	22	Uruguay
1500	una descripción	20/09/2020	5	20	22	Uruguay
1600	una descripción	26/09/2020	5	20	22	Uruguay
1700	una descripción	26/09/2020	5	20	22	Uruguay