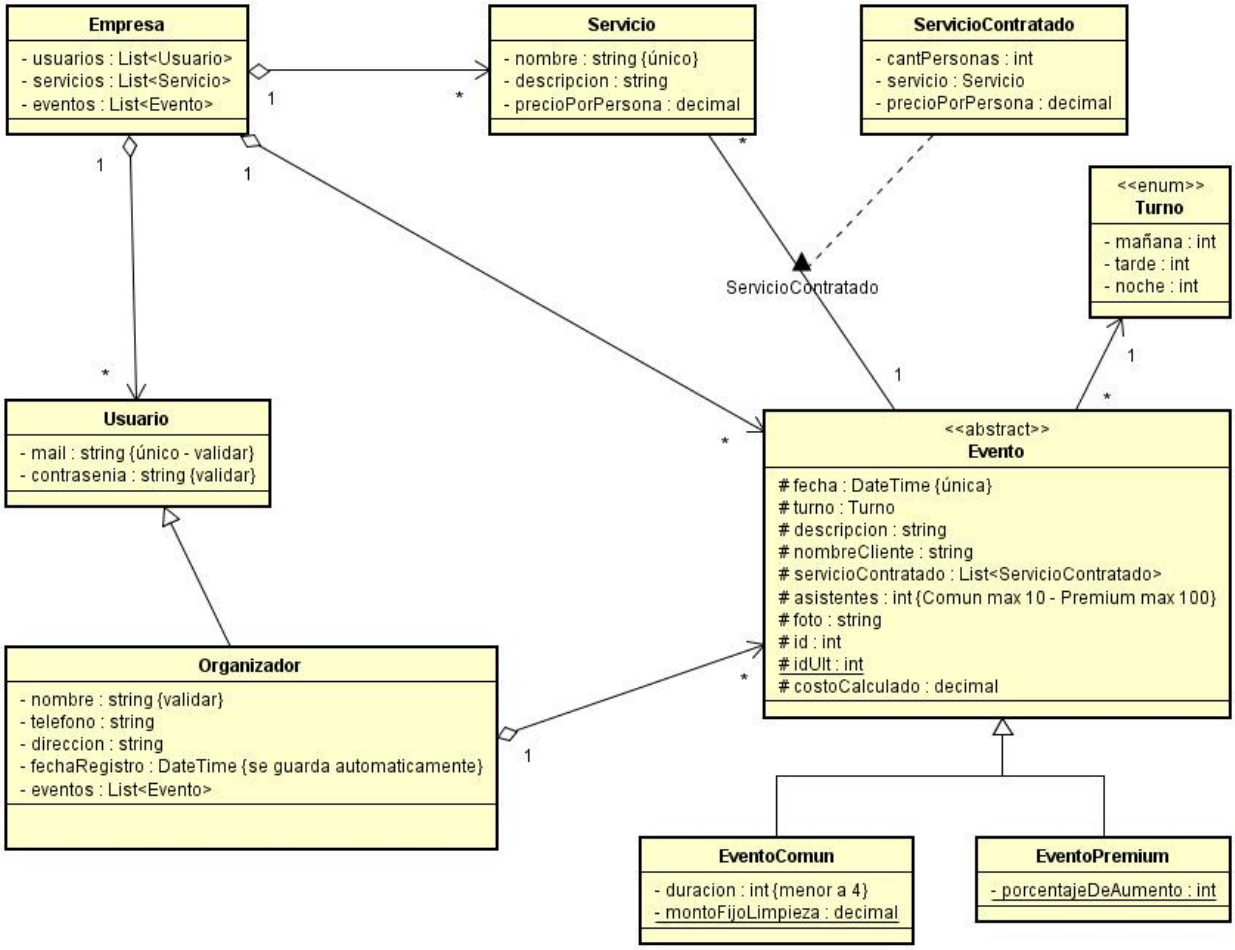
SANTIAGO CAMPS - 219114

Programación 2 - Grupo M2C - Liliana Pino

CONTENIDO

DIAGRAMA 3
DATOS PRECARGADOS 4
CÓDIGO 5
PROGRAM 5
EMPRESA 12
USUARIO 21
ORGANIZADOR 22
EVENTO 24
EVENTO COMÚN 26
EVENTO PREMIUM 27
SERVICIO 28
SERVICIO CONTRATADO 29

DIAGRAMA


Empresa
- usuarios : List<Usuario>
- servicios : List<Servicio>
- eventos : List<Evento>
Servicio
- nombre : string {único}
- descripcion : string
- precioPorPersona : decimal
ServicioContratado
- cantPersonas : int
- servicio : Servicio
- precioPorPersona : decimal
<<enum>>
Turno
- mañana : int
- tarde : int
- noche : int
ServicioContratado
Usuario
- mail : string {único - validar}
- contrasenia : string {validar}
<<abstract>>
Evento
fecha : DateTime {única}
turno : Turno
descripcion : string
nombreCliente : string
servicioContratado : List<ServicioContratado>
asistentes : int {Comun max 10 - Premium max 100}
foto : string
id : int
idUlt : int
costoCalculado : decimal
Organizador
- nombre : string {validar}
- telefono : string
- direccion : string
- fechaRegistro : DateTime {se guarda automaticamente}
- eventos : List<Evento>
EventoComun
- duracion : int {menor a 4}
- montoFijoLimpieza : decimal
EventoPremium
- porcentajeDeAumento : int
1
*

DATOS PRECARGADOS

Servicios:

Cabalgata Comida Parrilla Cotillon

Administrador:

admin@eventos17.com - Admin!99

Organizadores:

org@eventos17.com - Org!9999
lucas@mail.com - Lucas!111

Eventos:

Evento Común - 2018-10-01 - org@eventos17.com
Evento Premium - 2018-03-12 - org@eventos17.com
Evento Común - 2018-05-30 - org@eventos17.com
Evento Común - 2018-07-27 - lucas@mail.com
Evento Premium - 2018-04-08 - lucas@mail.com

```
AltaServicio("Cabalgata", "Un recorrido a caballo por un maravilloso paisaje.", 200);
AltaServicio("Comida", "Exelentes platos de comida muy rica.", 400);
AltaServicio("Parrilla", "Variedad de exquisitas carnes a la parrilla.", 550);
AltaServicio("Cotillon", "Chucherias para una fiesta bien divertida.", 120);

AltaAdministrador("admin@eventos17.com", "Admin!99");
AltaOrganizador("org@eventos17.com", "Org!9999", "Carlos", "099542788", "Calle 123");
AltaOrganizador("lucas@mail.com", "Lucas!111", "Lucas", "0996556545", "Avenida 456");

AltaEventoComun(new DateTime(2018, 10, 01), "tarde", "Cumpleaños de mi tío.", "Eduardo
Gonzalez", "Parrilla", 9, 9, "", 3);
AgregarEventoOrganizador(new DateTime(2018, 10, 01), "org@eventos17.com");

AltaEventoPremium(new DateTime(2018, 03, 12), "noche", "Aniversario del club.", "Alberto
P.", "Comida", 75, 100, "");
AgregarEventoOrganizador(new DateTime(2018, 03, 12), "org@eventos17.com");

AltaEventoComun(new DateTime(2018, 05, 30), "mañana", "Reunión de amigos.", "Eduardo
Gonzalez", "Cabalgata", 7, 9, "", 4);
AgregarEventoOrganizador(new DateTime(2018, 05, 30), "org@eventos17.com");

AltaEventoComun(new DateTime(2018, 07, 27), "tarde", "Reunión de negocios.", "Fernando
P.", "Comida", 7, 7, "", 4);
AgregarEventoOrganizador(new DateTime(2018, 07, 27), "lucas@mail.com");

AltaEventoPremium(new DateTime(2018, 04, 08), "noche", "Cumpleaños de Roberto.", "Alberto
P.", "Parrilla", 75, 100, "");
AgregarServicioEvento("Comida", 25, new DateTime(2018, 07, 08));
AgregarEventoOrganizador(new DateTime(2018, 04, 08), "lucas@mail.com");
```

CÓDIGO

PROGRAM

```
private static Empresa empresa = new Empresa();
static void Main(string[] args)
{
    int opcion = 0;
    while (opcion != 10)
    {
        MostrarOpciones();
        int.TryParse(Console.ReadLine(), out opcion);
        VerificarOpcion(opcion);
    }
    Console.ReadKey();
}

// Mostrar Opciones
static void MostrarOpciones()
{
    Console.WriteLine("------------------------------------------------------------");
    Console.WriteLine(" 1 - Registro de Administrador");
    Console.WriteLine(" 2 - Registro de Organizador");
    Console.WriteLine(" 3 - Registro de Evento");
    Console.WriteLine(" 4 - Agregar Servicios a Evento");
    Console.WriteLine(" 5 - Listar todos los Usuarios");
    Console.WriteLine(" 6 - Listar el catálogo de Servicios");
    Console.WriteLine(" 7 - Listar todos los Eventos de un Organizador");
    Console.WriteLine(" 8 - Cambiar Porcentaje de Aumento para Eventos Premium");
    Console.WriteLine(" 9 - Cambiar Monto Fijo de Limpieza para Eventos Comunes");
    Console.WriteLine(" 10 - Salir");
    Console.WriteLine("------------------------------------------------------------");
}

// Verificar Opcion
static void VerificarOpcion(int opcion)
{
    switch (opcion)
    {
        case 1:
            AltaAdministrador();
            break;
        case 2:
            AltaOrganizador();
            break;
        case 3:
            AltaEvento();
            break;
        case 4:
            AgregarServiciosEvento();
            break;
        case 5:
            ListarUsuariosTodos();
            break;
```

```
            case 6:
                ListarServicios();
                break;
            case 7:
                EventosDeOrganizador();
                break;
            case 8:
                CambiarPorcentajeDeAumento();
                break;
            case 9:
                CambiarMontoFijoLimpieza();
                break;
            default:
                Console.Clear();
                Console.WriteLine("Presione cualquier tecla para salir.");
                break;

        }
    }

    // Alta Administrador
    static void AltaAdministrador()
    {

        Console.WriteLine("Ingrese mail");
        string mail = Console.ReadLine();
        Console.WriteLine("Ingrese contraseña");
        string contrasenia = Console.ReadLine();

        if (!empresa.ValidarMail(mail))
        {
            Console.WriteLine("El mail ingresado no es válido");
        }
        else if (!empresa.ValidarContrasenia(contrasenia))
        {
            Console.WriteLine("La contraseña ingresada no es válida");
        }
        else if (empresa.BuscarUsuario(mail) != null)
        {
            Console.WriteLine("Ese mail ya está registrado");
        }
        else
        {
            if (empresa.AltaAdministrador(mail, contrasenia))
            {
                Console.WriteLine("Administrador registrado exitosamente");
            }
            else
            {
                Console.WriteLine("No se pudo registrar, intentelo nuevamente");
            }
        }
    }

    // Alta Organizador
    static void AltaOrganizador()
    {
        Console.WriteLine("Ingrese mail");
```

```
    string mail = Console.ReadLine();
    Console.WriteLine("Ingrese contraseña");
    string contrasenia = Console.ReadLine();
    Console.WriteLine("Ingrese su nombre");
    string nombre = Console.ReadLine();
    Console.WriteLine("Ingrese su telefono");
    string telefono = Console.ReadLine();
    Console.WriteLine("Ingrese su direccion");
    string direccion = Console.ReadLine();

    if (!empresa.ValidarMail(mail))
    {
        Console.WriteLine("El mail ingresado no es válido");
    }
    else if (!empresa.ValidarContrasenia(contrasenia))
    {
        Console.WriteLine("La contraseña ingresada no es válida");
    }
    else if (empresa.BuscarUsuario(mail) != null)
    {
        Console.WriteLine("Ese mail ya está registrado");
    }
    else if (!empresa.ValidarNombre(nombre))
    {
        Console.WriteLine("El nombre ingresado no es válido");
    }
    else
    {
        if (empresa.AltaOrganizador(mail, contrasenia, nombre, telefono, direccion))
        {
            Console.WriteLine("Organizador registrado exitosamente");
        }
        else
        {
            Console.WriteLine("No se pudo registrar, intentelo nuevamente");
        }
    }
}

// Alta Evento
static void AltaEvento()
{
    Console.WriteLine("Ingrese mail");
    string mail = Console.ReadLine();
    Console.WriteLine("Ingrese contraseña");
    string contrasenia = Console.ReadLine();
    if (!empresa.ValidarMail(mail))
    {
        Console.WriteLine("El mail ingresado no es válido");
    }
    else if (!empresa.ValidarContrasenia(contrasenia))
    {
        Console.WriteLine("La contraseña ingresada no es válida");
    }
    else if (!empresa.UsuarioEsOrganizador(mail))
    {
        Console.WriteLine("---Ud no es un Organizador---");
    }
```

```
    else {
        Console.WriteLine("---Sus Datos---");
        Console.WriteLine(empresa.DatosDeOrganizador(mail));
        Console.WriteLine("---------------");
        Console.WriteLine("Ingrese fecha con el formato AAAA-MM-DD");
        DateTime fecha = DateTime.Now;
        string fechaString = Console.ReadLine();
        DateTime.TryParse(fechaString, out fecha);
        if (empresa.BuscarEvento(fecha) != null)
        {
            Console.WriteLine("Ya existe un evento ese día");
        }
        else
        {
            Console.WriteLine("Ingrese turno del evento");
            string[] turnos = Enum.GetNames(typeof(Turno));
            foreach (string tur in turnos)
            {
                Console.WriteLine(tur);
            }
            string turnoString = Console.ReadLine();
            Console.WriteLine("Ingrese descripcion");
            string descripcion = Console.ReadLine();
            Console.WriteLine("Ingrese nombre del cliente");
            string nombreCliente = Console.ReadLine();
            Console.WriteLine("Escriba el nombre de un servicio");
            Console.WriteLine(empresa.ListarServicios());
            string nombreServicio = Console.ReadLine();
            Console.WriteLine("Ingrese cantidad de personas para ese servicio");
            int cantPersonasServicio;
            int.TryParse(Console.ReadLine(), out cantPersonasServicio);
            Console.WriteLine("Ingrese cantidad de asistentes al evento");
            int asistentes;
            int.TryParse(Console.ReadLine(), out asistentes);
            string foto = "";
            Console.WriteLine("Tipo de evento 1-Comun 2-Premium");
            string tipoEvento = Console.ReadLine();
            if (tipoEvento == "1")
            {
                Console.WriteLine("Ingrese duración");
                int duracion;
                int.TryParse(Console.ReadLine(), out duracion);

                if (empresa.AltaEventoComun(fecha, turnoString, descripcion,
nombreCliente, nombreServicio, cantPersonasServicio, asistentes, foto, duracion) &&
                    empresa.AgregarEventoOrganizador(fecha, mail))
                {
                    Console.WriteLine("El evento se creó exitosamente");
                    Console.WriteLine(empresa.MostrarDatosDeEvento(fecha));
                }
                else
                {
                    Console.WriteLine("El evento no pudo crearse, intente nuevamente");
                }
            }
            else if (tipoEvento == "2")
            {
```

```
                if (empresa.AltaEventoPremium(fecha, turnoString, descripcion,
nombreCliente, nombreServicio, cantPersonasServicio, asistentes, foto) &&
                    empresa.AgregarEventoOrganizador(fecha, mail))
                {
                    Console.WriteLine("El evento se creó exitosamente");
                    Console.WriteLine(empresa.MostrarDatosDeEvento(fecha));
                }
                else
                {
                    Console.WriteLine("El evento no pudo crearse, intente nuevamente");
                }
            }
            else
            {
                Console.WriteLine("Algo salió mal");
            }



        }
    }
}

// Agregar Servicios a Evento
static void AgregarServiciosEvento()
{
    Console.WriteLine("Ingrese mail");
    string mail = Console.ReadLine();
    Console.WriteLine("Ingrese contraseña");
    string contrasenia = Console.ReadLine();
    if (!empresa.ValidarMail(mail))
    {
        Console.WriteLine("El mail ingresado no es válido");
    }
    else if (!empresa.ValidarContrasenia(contrasenia))
    {
        Console.WriteLine("La contraseña ingresada no es válida");
    }
    else if (!empresa.UsuarioEsOrganizador(mail))
    {
        Console.WriteLine("---Ud no es un Organizador---");
    }
    else {
        Console.WriteLine("---Mis Datos---");
        Console.WriteLine(empresa.DatosDeOrganizador(mail));
        Console.WriteLine("---------------");
        Console.WriteLine(empresa.ListarEventosOrganizador(mail, contrasenia));
        Console.WriteLine("Ingrese fecha del evento con el formato AAAA-MM-DD");
        DateTime fecha = DateTime.Now;
        string fechaString = Console.ReadLine();
        DateTime.TryParse(fechaString, out fecha);
        if (empresa.BuscarEvento(fecha) == null)
        {
            Console.WriteLine("No existe un evento ese día");
        }
        else
        {
            string masServicios = "1";
```

```
            while (masServicios == "1")
            {
                Console.WriteLine("Escriba el nombre de un servicio");
                Console.WriteLine(empresa.ListarServicios());
                string nombreServicio = Console.ReadLine();
                Console.WriteLine("Ingrese cantidad de personas para ese servicio");
                int cantPersonasServicio;
                int.TryParse(Console.ReadLine(), out cantPersonasServicio);

                if (empresa.AgregarServicioEvento(nombreServicio, cantPersonasServicio,
fecha))
                {
                    Console.WriteLine("El Servicio se agregó correctamente");
                }
                else
                {
                    Console.WriteLine("El Servicio no pudo agregarse");
                }
                Console.WriteLine(empresa.MostrarDatosDeEvento(fecha));
                Console.WriteLine("Desea agregar otro servicio? 1-Si 2-No");
                masServicios = Console.ReadLine();
            }
        }
    }
}

// Listar Usuarios
static void ListarUsuariosTodos()
{
    Console.WriteLine("---Todos los usuarios---");
    Console.WriteLine(empresa.ListarUsuariosTodos());
}

// Listar Servicios
static void ListarServicios()
{
    Console.WriteLine("---Todos los servicios---");
    Console.WriteLine(empresa.ListarServicios());
}

// Eventos de Organizador
static void EventosDeOrganizador()
{
    Console.WriteLine("Ingrese mail");
    string mail = Console.ReadLine();
    Console.WriteLine("Ingrese contraseña");
    string contrasenia = Console.ReadLine();
    Console.WriteLine(empresa.ListarEventosOrganizador(mail, contrasenia));
}

// Cambiar Porcentaje de Aumento para Eventos Premium
static void CambiarPorcentajeDeAumento()
{
    Console.WriteLine("El Porcentaje de Aumento para Eventos Premium es " +
empresa.VerPorcentajeDeAumento());
    Console.WriteLine("Desea cambiarlo? 1-Si 2-No");
    string siNo = Console.ReadLine();
    if (siNo == "1")
```

```
    {
        Console.WriteLine("Ingrese el nuevo valor de Porcentaje de Aumento para Eventos
Premium");
        int nuevoValor;
        int.TryParse(Console.ReadLine(), out nuevoValor);
        if (nuevoValor > 0 &&
            empresa.CambiarPorcentajeDeAumento(nuevoValor))
        {
            Console.WriteLine("Valor cambiado correctamente");
        }
        else
        {
            Console.WriteLine("El valor no pudo cambiarse");
        }
    }
}

// Cambiar Monto Fijo de Limpieza para Eventos Comunes
static void CambiarMontoFijoLimpieza()
{
    Console.WriteLine("El Monto Fijo de Limpieza para Eventos Comunes es " +
empresa.VerMontoFijoLimpieza());
    Console.WriteLine("Desea cambiarlo? 1-Si 2-No");
    string siNo = Console.ReadLine();
    if (siNo == "1")
    {
        Console.WriteLine("Ingrese el nuevo valor de Monto Fijo de Limpieza para Eventos
Comunes");
        int nuevoValor;
        int.TryParse(Console.ReadLine(), out nuevoValor);
        if (nuevoValor > 0 &&
            empresa.CambiarMontoFijoLimpieza(nuevoValor))
        {
            Console.WriteLine("Valor cambiado correctamente");
        }
        else
        {
            Console.WriteLine("El valor no pudo cambiarse");
        }
    }
}
```

EMPRESA

```
private List<Usuario> usuarios = new List<Usuario>();
private List<Servicio> servicios = new List<Servicio>();
private List<Evento> eventos = new List<Evento>();

// Constructor
public Empresa()
{
    PrecargarDatos();
}

// Precargar Datos
private void PrecargarDatos()
{
    AltaServicio("Cabalgata", "Un recorrido a caballo por un maravilloso paisaje.", 200);
    AltaServicio("Comida", "Exelentes platos de comida muy rica.", 400);
    AltaServicio("Parrilla", "Variedad de exquisitas carnes a la parrilla.", 550);
    AltaServicio("Cotillon", "Chucherias para una fiesta bien divertida.", 120);

    AltaAdministrador("admin@eventos17.com", "Admin!99");

    AltaOrganizador("org@eventos17.com", "Org!9999", "Carlos", "099542788", "Calle 123");
    AltaOrganizador("lucas@mail.com", "Lucas!111", "Lucas", "0996556545", "Avenida 456");

    AltaEventoComun(new DateTime(2018, 10, 01), "tarde", "Cumpleaños de mi tío.",
"Eduardo Gonzalez", "Parrilla", 9, 9, "", 3);
    AgregarEventoOrganizador(new DateTime(2018, 10, 01), "org@eventos17.com");

    AltaEventoPremium(new DateTime(2018, 03, 12), "noche", "Aniversario del club.",
"Alberto P.", "Comida", 75, 100, "");
    AgregarEventoOrganizador(new DateTime(2018, 03, 12), "org@eventos17.com");

    AltaEventoComun(new DateTime(2018, 05, 30), "mañana", "Reunión de amigos.", "Eduardo
Gonzalez", "Cabalgata", 7, 9, "", 4);
    AgregarEventoOrganizador(new DateTime(2018, 05, 30), "org@eventos17.com");

    AltaEventoComun(new DateTime(2018, 07, 27), "tarde", "Reunión de negocios.",
"Fernando P.", "Comida", 7, 7, "", 4);
    AgregarEventoOrganizador(new DateTime(2018, 07, 27), "lucas@mail.com");

    AltaEventoPremium(new DateTime(2018, 04, 08), "noche", "Cumpleaños de Roberto.",
"Alberto P.", "Parrilla", 75, 100, "");
    AgregarServicioEvento("Comida", 25, new DateTime(2018, 07, 08));
    AgregarEventoOrganizador(new DateTime(2018, 04, 08), "lucas@mail.com");
}



// Buscar Servicio
public Servicio BuscarServicio(string nombre)
{
    // Busca un servicio según su nombre y lo devuelve, si no existe devuelve null
    Servicio servicio = null;
    int i = 0;
    while (i < servicios.Count && servicio == null)
    {
```

```
        if (servicios[i].Nombre == nombre)
        {
            servicio = servicios[i];
        }
        i++;
    }
    return servicio;
}

// Alta Servicio
public bool AltaServicio(string nombre, string descripcion, decimal precioPorPersona)
{
    bool alta = false;
    Servicio servicio = BuscarServicio(nombre);
    int cantServicios = servicios.Count;
    if (servicio == null)
    {
        servicios.Add(new Servicio(nombre, descripcion, precioPorPersona));
    }
    if (cantServicios < servicios.Count)
    {
        alta = true;
    }
    return alta;
}

// Listar Servicios
public string ListarServicios()
{
    // Lista todos los servicios existentes
    string lista = "";
    foreach (Servicio servicio in servicios)
    {
        lista += servicio.ToString() + "\n";
    }
    return lista;
}



// Buscar Usuario
public Usuario BuscarUsuario(string mail)
{
    // Busca un usuario según su mail y lo devuelve, null si no está registrado
    Usuario usuario = null;
    int i = 0;
    while (i<usuarios.Count && usuario == null)
    {
        if (usuarios[i].Mail == mail)
        {
            usuario = usuarios[i];
        }
        i++;
    }
    return usuario;
}
```

```
// Alta Administrador
public bool AltaAdministrador(string mail, string contrasenia)
{
    bool alta = false;
    Usuario usuario = BuscarUsuario(mail);
    int cantUsuarios = usuarios.Count;
    bool datosValidados = ValidarMail(mail) && ValidarContrasenia(contrasenia);
    if (usuario == null && datosValidados)
    {
        usuarios.Add(new Usuario(mail, contrasenia));
    }
    if (cantUsuarios < usuarios.Count)
    {
        alta = true;
    }
    return alta;
}

// Alta Organizador
public bool AltaOrganizador(string mail, string contrasenia, string nombre, string
telefono, string direccion)
{
    bool alta = false;
    Usuario usuario = BuscarUsuario(mail);
    int cantUsuarios = usuarios.Count;
    bool datosValidados = ValidarMail(mail) && ValidarContrasenia(contrasenia) &&
ValidarNombre(nombre);
    if (usuario == null && datosValidados)
    {
        usuarios.Add(new Organizador(mail, contrasenia, nombre, telefono, direccion));
    }
    if (cantUsuarios < usuarios.Count)
    {
        alta = true;
    }
    return alta;
}

// Validar si Usuario es Organizador
public bool UsuarioEsOrganizador(string mail)
{
    // Devuelve true si el usuario es de la clase Organizador
    bool validado = false;
    Usuario usuario = BuscarUsuario(mail);
    if (usuario != null && usuario.MiRol() == "Organizador")
    {
        validado = true;
    }
    return validado;
}

// Datos de Organizador
public string DatosDeOrganizador(string mail)
{
    // Devuelve los datos del Organizador
    string datos = "";
    Usuario usuario = BuscarUsuario(mail);
    Organizador organizador = null;
```

```
    if (usuario != null && UsuarioEsOrganizador(mail))
    {
        organizador = (Organizador)usuario;
        datos = organizador.MisDatos();
    }
    return datos;
}

// Agregar Evento a Organizador
public bool AgregarEventoOrganizador(DateTime fechaEvento, string mailOrganizador)
{
    // Agrega un Evento a la lista de Eventos de un Organizador y devuelve true, si el
usuario es organizador y si el evento existe
    bool agregado = false;
    Evento evento = BuscarEvento(fechaEvento);
    Usuario usuario = BuscarUsuario(mailOrganizador);
    Organizador organizador = null;
    if (usuario != null && UsuarioEsOrganizador(mailOrganizador))
    {
        organizador = (Organizador)usuario;
    }
    if (evento != null && organizador != null)
    {
        agregado = organizador.AgregarEvento(evento);
    }
    return agregado;
}

// Listar Usuarios Todos
public string ListarUsuariosTodos()
{
    // Devuelve una lista de todos los usuario con sus datos y su rol
    string lista = "";
    foreach ( Usuario usuario in usuarios)
    {
        lista += "ROL: " + usuario.MiRol() + " " + usuario.ToString() + "\n";
    }
    return lista;
}




// Buscar Evento
public Evento BuscarEvento(DateTime fecha)
{
    // Busca si un evento existe y lo devuelve, de lo contrario devuelve null
    Evento evento = null;
    int i = 0;
    while (i < eventos.Count && evento == null)
    {
        if (eventos[i].Fecha.Year == fecha.Year &&
            eventos[i].Fecha.Month == fecha.Month &&
            eventos[i].Fecha.Day == fecha.Day)
        {
            evento = eventos[i];
        }
        i++;
```

```
    }
    return evento;
}

// Alta Evento Comun
public bool AltaEventoComun(DateTime fecha, string turnoString, string descripcion,
string nombreCliente, string nombreServicio, int cantPersonasServicio, int asistentes,
string foto, int duracion)
{
    bool alta = false;
    bool datosValidados = ValidarEventoComun(fecha, turnoString, descripcion,
nombreCliente, nombreServicio, cantPersonasServicio, asistentes, foto, duracion);
    Turno turno = BuscarTurno(turnoString);
    Servicio servicio = BuscarServicio(nombreServicio);
    Evento evento = BuscarEvento(fecha);
    int cantEventos = eventos.Count;
    if (evento == null && datosValidados && turno != 0)
    {
        eventos.Add(new EventoComun(fecha, turno, descripcion, nombreCliente, servicio,
cantPersonasServicio, asistentes, foto, duracion));
    }
    if (cantEventos < eventos.Count)
    {
        alta = true;
    }
    return alta;
}

// Alta Evento Premium
public bool AltaEventoPremium(DateTime fecha, string turnoString, string descripcion,
string nombreCliente, string nombreServicio, int cantPersonasServicio, int asistentes,
string foto)
{
    bool alta = false;
    bool datosValidados = ValidarEventoPremium(fecha, turnoString, descripcion,
nombreCliente, nombreServicio, cantPersonasServicio, asistentes, foto);
    Turno turno = BuscarTurno(turnoString);
    Servicio servicio = BuscarServicio(nombreServicio);
    Evento evento = BuscarEvento(fecha);
    int cantEventos = eventos.Count;
    if (evento == null && datosValidados && turno != 0)
    {
        eventos.Add(new EventoPremium(fecha, turno, descripcion, nombreCliente, servicio,
cantPersonasServicio, asistentes, foto));
    }
    if (cantEventos < eventos.Count)
    {
        alta = true;
    }
    return alta;
}

// Agregar Servicio a Evento
public bool AgregarServicioEvento(string nombreServicio, int cantPersonas, DateTime
fechaEvento)
{
    bool agregado = false;
    Servicio servicio = BuscarServicio(nombreServicio);
```

```
    Evento evento = BuscarEvento(fechaEvento);
    if (servicio != null &&
        evento != null &&
        cantPersonas <= evento.Asistentes)
    {
        agregado = evento.AgregarServicioContratado(servicio, cantPersonas);
    }
    return agregado;
}

// Ver Porcentaje de Aumento para Eventos Premium
public int VerPorcentajeDeAumento()
{
    // Muestra el ese dato
    return EventoPremium.PorcentajeDeAumento;
}

// Cambiar Porcentaje de Aumento para Eventos Premium
public bool CambiarPorcentajeDeAumento(int nuevoValor)
{
    // Cambia ese dato
    return EventoPremium.CambiarPorcentajeDeAumento(nuevoValor);
}

// Ver Monto Fijo de Limpieza para Eventos Comunes
public decimal VerMontoFijoLimpieza()
{
    // Muestra ese dato
    return EventoComun.MontoFijoLimpieza;
}

// Cambiar Monto Fijo de Limpieza para Eventos Comunes
public bool CambiarMontoFijoLimpieza(decimal nuevoValor)
{
    // Cambia ese dato
    return EventoComun.CambiarMontoFijoLimpieza(nuevoValor);
}

// Listar Eventos de Organizador
public string ListarEventosOrganizador(string mail, string contrasenia)
{
    // Lista todos los Eventos del Organizador si sus datos son correctos, si no es
Organizador o si no existe muestra un aviso
    string lista = "";
    Usuario usuario = BuscarUsuario(mail);
    if (usuario == null || usuario.Contrasenia != contrasenia)
    {
        lista = "El usuario ingresado o la contraseña no son correctos";
    }
    else
    {
        lista = usuario.ListarEventos();
    }
    return lista;
}

// Mostrar datos de Evento
public string MostrarDatosDeEvento(DateTime fecha)
```

```
{
    // Muestra los datos de un Evenot, si es que existe
    string datos = "";
    Evento evento = BuscarEvento(fecha);
    if (evento != null)
    {
        datos += evento.ToString() + "\n" + evento.DatosDeMisServicios();
    }
    return datos;
}



// Validar Mail
public bool ValidarMail(string mail)
{
    // Devuelve true si el mail cumple con los requerimientos especificados
    bool validado = false;
    bool tieneArroba = mail.Contains("@");
    bool arrobaNoAlPrincipio = mail[0] != '@';
    bool arrobaNoAlFinal = mail[mail.Length-1] != '@';
    bool tienePunto = mail.Contains(".");
    bool puntoDespuesDeArroba = mail.IndexOf('.') > mail.IndexOf('@');
    bool dosCharDespuesDePunto = (mail.IndexOf('.') <= mail.Length - 3);
    if (tieneArroba && arrobaNoAlPrincipio && arrobaNoAlFinal && tienePunto &&
puntoDespuesDeArroba && dosCharDespuesDePunto)
    {
        validado = true;
    }
    return validado;
}

// Validar Contraseña
public bool ValidarContrasenia(string contrasenia)
{
    // Devuelve true si la contraseña cumple con los requerimientos especificados
    bool validado = false;
    bool largo = contrasenia.Length >= 8;
    bool caracterEspecial = false;
    if (contrasenia.Contains("!") ||
        contrasenia.Contains(".") ||
        contrasenia.Contains(",") ||
        contrasenia.Contains(";"))
    {
        caracterEspecial = true;
    }
    bool mayuscula = false;
    if (contrasenia.ToLower() != contrasenia)
    {
        mayuscula = true;
    }
    if (largo && caracterEspecial && mayuscula)
    {
        validado = true;
    }
    return validado;
}
```

```
// Validar Nombre (min 3 letras)
public bool ValidarNombre(string nombre)
{
    // Devuelve true si el nombre cumple con los requerimientos especificados
    bool validado = false;
    bool largo = nombre.Length >= 3;
    bool todasLetras = true;
    for (int i = 0; i < nombre.Length; i++)
    {
        if (!Char.IsLetter(nombre[i]))
        {
            todasLetras = false;
            break;
        }
    }
    if (largo && todasLetras)
    {
        validado = true;
    }
    return validado;
}

// Buscar turno
public Turno BuscarTurno(string turnoString)
{
    // Busca un Turno y si existe devuelve ese enum
    Turno turno = 0;
    string[] turnos = Enum.GetNames(typeof(Turno));
    if (turnos.Contains(turnoString))
    {
        turno = (Turno)Enum.Parse(typeof(Turno), turnoString);
    }
    return turno;
}

// Validar Evento Generico
public bool ValidarEventoGenerico(DateTime fecha, string turnoString, string descripcion,
string nombreCliente, string nombreServicio, int cantPersonasServicio, int asistentes,
string foto)
{
    // Hace una validación básica de los datos ingresados para los eventos
    bool validado = false;
    Servicio servicio = BuscarServicio(nombreServicio);
    if (fecha > DateTime.Now &&
        turnoString != "" &&
        descripcion != "" &&
        nombreCliente != "" &&
        servicio != null &&
        cantPersonasServicio > 0 &&
        asistentes > 0 &&
        cantPersonasServicio <= asistentes)
    {
        validado = true;
    }
    return validado;
}
```

```
//Validar Evento Comun
public bool ValidarEventoComun(DateTime fecha, string turnoString, string descripcion,
string nombreCliente, string nombreServicio, int cantPersonasServicio, int asistentes,
string foto, int duracion)
{
    // Hace las validaciones específicas de los Eventos Comunes
    bool validado = false;
    if (ValidarEventoGenerico(fecha, turnoString, descripcion, nombreCliente,
nombreServicio, cantPersonasServicio, asistentes, foto) &&
        duracion > 0 &&
        duracion <= 4 &&
        asistentes <= 10 )
    {
        validado = true;
    }
    return validado;
}

// Validar Evento Premium
public bool ValidarEventoPremium(DateTime fecha, string turnoString, string descripcion,
string nombreCliente, string nombreServicio, int cantPersonasServicio, int asistentes,
string foto)
{
    // Hace la validación específica de los Eventos Premium
    bool validado = false;
    if (ValidarEventoGenerico(fecha, turnoString, descripcion, nombreCliente,
nombreServicio, cantPersonasServicio, asistentes, foto) &&
        asistentes <= 100)
    {
        validado = true;
    }
    return validado;
}
```

USUARIO

```
private string mail;
private string contrasenia;

public string Mail
{
    get { return mail; }
}
public string Contrasenia
{
    get { return contrasenia; }
}

// Constructor
public Usuario(string mail, string contrasenia)
{
    this.mail = mail;
    this.contrasenia = contrasenia;
}

// Mi Rol
public virtual string MiRol()
{
    return "Administrador";
}

// Listar Eventos
public virtual string ListarEventos()
{
    return "--Soy Administrador, no tengo eventos--";
}

// To String
public override string ToString()
{
    return "MAIL: " + mail + " CONTRASEÑA: " + contrasenia;
}
```

ORGANIZADOR

```
private string nombre;
private string telefono;
private string direccion;
private DateTime fechaRegistro;
private List<Evento> eventos = new List<Evento>();

// Constructor
public Organizador(string mail, string contrasenia, string nombre, string telefono,
string direccion) :base(mail, contrasenia)
{
    this.nombre = nombre;
    this.telefono = telefono;
    this.direccion = direccion;
    fechaRegistro = DateTime.Today;
}

// Mi Rol
public override string MiRol()
{
    return "Organizador";
}

// Agregar Evento
public bool AgregarEvento(Evento evento)
{
    bool agregado = false;
    int cantEventos = eventos.Count;
    if (evento != null)
    {
        eventos.Add(evento);
    }
    if (cantEventos < eventos.Count)
    {
        agregado = true;
    }
    return agregado;
}

// Listar Eventos
public override string ListarEventos()
{
    string lista = "---Mis eventos---" + "\n";
    decimal total = 0;
    foreach (Evento evento in eventos)
    {
        lista += evento.ToString() + "\n";
        total += evento.CostoCalculado;
    }
    lista += "TOTAL GENERADO: " + total;
    if (eventos.Count == 0)
    {
        lista = "---No tengo ningún evento---";
    }
    return lista;
}
```

```
// Mis Datos
public string MisDatos()
{
    return "NOMBRE: " + nombre + " TEL: " + telefono + " DIR: " + direccion + "
REGISTRADO: " + fechaRegistro.ToShortDateString();
}

// To String
public override string ToString()
{
    return base.ToString() + " " + MisDatos();
}
```

EVENTO

```
// Atributos
protected DateTime fecha;
protected Turno turno;
protected string descripcion;
protected string nombreCliente;
protected List<ServicioContratado> serviciosContratados = new List<ServicioContratado>();
protected int asistentes;
protected string foto;
protected int id;
protected static int idUlt = 1;
protected decimal costoCalculado;

// Propiedades
public DateTime Fecha
{
    get { return fecha; }
}
public int Asistentes
{
    get { return asistentes; }
}
public decimal CostoCalculado
{
    get { return costoCalculado; }
}

// Constructor
public Evento(DateTime fecha, Turno turno, string descripcion, string nombreCliente,
Servicio servicio, int cantPersonasServicio, int asistentes, string foto)
{
    this.fecha = fecha;
    this.turno = turno;
    this.descripcion = descripcion;
    this.nombreCliente = nombreCliente;
    this.asistentes = asistentes;
    this.foto = foto;
    serviciosContratados.Add(new ServicioContratado(servicio, cantPersonasServicio));
    this.id = Evento.idUlt;
    Evento.idUlt++;
    this.costoCalculado = CalcularCosto();
}

// Agregar Servicio Contratado
public bool AgregarServicioContratado(Servicio servicio, int cantPersonas)
{
    bool agregado = false;
    int cantServicios = serviciosContratados.Count;
    ServicioContratado servicioNuevo = new ServicioContratado(servicio, cantPersonas);
    serviciosContratados.Add(servicioNuevo);
    if (cantServicios < serviciosContratados.Count)
    {
        ActualizarCosto(servicioNuevo.CalcularCosto());
        agregado = true;
    }
    return agregado;
```

```
}

// Calcular Costos de Servicios Contratados
public decimal CalcularCostosServiciosContratados()
{
    decimal costo = 0;
    foreach (ServicioContratado servicioContratado in serviciosContratados)
    {
        costo += servicioContratado.CalcularCosto();
    }
    return costo;
}

// Calcular Costo
public abstract decimal CalcularCosto();

// Actualizar Costo
public abstract void ActualizarCosto(decimal costoServicio);

// Datos de mis servicios
public string DatosDeMisServicios()
{
    string datos = "";
    foreach (ServicioContratado serv in serviciosContratados)
    {
        datos += serv.ToString()+ "\n";
    }
    return datos;
}
```

EVENTO COMÚN

```
private int duracion;
private static decimal montoFijoLimpieza = 100;

// Propiedades
public static decimal MontoFijoLimpieza
{
    get { return montoFijoLimpieza; }
}

// Constructor
public EventoComun(DateTime fecha, Turno turno, string descripcion, string nombreCliente,
Servicio servicio, int cantPersonasServicio, int asistentes, string foto, int duracion)
:base (fecha, turno, descripcion, nombreCliente, servicio, cantPersonasServicio,
asistentes, foto)
{
    this.duracion = duracion;
}

// Cambiar Monto Fijo Limpieza
public static bool CambiarMontoFijoLimpieza(decimal nuevoValor)
{
    bool cambiado = false;
    decimal valorAnterior = EventoComun.montoFijoLimpieza;
    EventoComun.montoFijoLimpieza = nuevoValor;
    if (valorAnterior != EventoComun.montoFijoLimpieza)
    {
        cambiado = true;
    }
    return cambiado;
}

// Calcular Costo
public override decimal CalcularCosto()
{
    decimal costo = 0;
    costo += base.CalcularCostosServiciosContratados() + montoFijoLimpieza;
    return costo;
}

// Actualizar Costo
public override void ActualizarCosto(decimal costoServicio)
{
    costoCalculado += costoServicio;
}

// ToString
public override string ToString()
{
    return "COD: " + id + " Evento Común  " + " FECHA: " + fecha.ToShortDateString() + "
CLIENTE: " + nombreCliente + " DESC: " + descripcion + " COSTO TOTAL: (inc. limpieza) " +
costoCalculado;
}
```

EVENTO PREMIUM

```
private static int porcentajeDeAumento = 15;

// Propiedades
public static int PorcentajeDeAumento
{
    get { return porcentajeDeAumento; }
}

// Constructor
public EventoPremium(DateTime fecha, Turno turno, string descripcion, string
nombreCliente, Servicio servicio, int cantPersonasServicio, int asistentes, string foto)
: base(fecha, turno, descripcion, nombreCliente, servicio, cantPersonasServicio,
asistentes, foto)
{

}

// Cambiar Porcentaje De Aumento
public static bool CambiarPorcentajeDeAumento(int nuevoValor)
{
    bool cambiado = false;
    int valorAnterior = EventoPremium.porcentajeDeAumento;
    EventoPremium.porcentajeDeAumento = nuevoValor;
    if (valorAnterior != EventoPremium.porcentajeDeAumento)
    {
        cambiado = true;
    }
    return cambiado;
}

// Calcular Costo
public override decimal CalcularCosto()
{
    decimal costo = 0;
    costo += base.CalcularCostosServiciosContratados() * porcentajeDeAumento;
    return costo;
}

// Actualizar Costo
public override void ActualizarCosto(decimal costoServicio)
{
    costoCalculado += costoServicio * porcentajeDeAumento;
}

// ToString
public override string ToString()
{
    return "COD: " + id + " Evento Premium" + " FECHA: " + fecha.ToShortDateString() + "
CLIENTE: " + nombreCliente + " DESC: " + descripcion + " COSTO TOTAL: " + costoCalculado;
}
```

SERVICIO

```
private string nombre;
private string descripcion;
private decimal precioPorPersona;

// Propiedades
public string Nombre
{
    get { return nombre; }
}
public decimal PrecioPorPersona
{
    get { return precioPorPersona; }
}

// Constructor
public Servicio(string nombre, string descripcion, decimal precioPorPersona)
{
    this.nombre = nombre;
    this.descripcion = descripcion;
    this.precioPorPersona = precioPorPersona;
}

// ToString
public override string ToString()
{
    return nombre + " - $" + precioPorPersona + " por persona" + " - " + descripcion;
}
```

SERVICIO CONTRATADO

```
private int cantPersonas;
private Servicio servicio;
private decimal precioPorPersona;

// Constructor
public ServicioContratado(Servicio servicio, int cantPersonas)
{
    this.servicio = servicio;
    this.cantPersonas = cantPersonas;
    this.precioPorPersona = servicio.PrecioPorPersona;
}

// Calcular Costo
public decimal CalcularCosto()
{
    return cantPersonas * precioPorPersona;
}

// ToString
public override string ToString()
{
    return "SERVICIO: " + servicio.Nombre + " PERSONAS: " + cantPersonas + " PRECIO POR
PERSONA: " + precioPorPersona + " COSTO: " + CalcularCosto();
}
```