

Universidad ORT Uruguay
Facultad de Ingeniería
Escuela de Tecnología

OBLIGATORIO 2 PROGRAMACION 2
DOCUMENTO DE ANÁLISIS



Emmanuel Duffaut - 177427



Joaquin Caviglia – 328187

N2E

Docente: Mauricio Quinteros

Analista Programador 20-06-2024

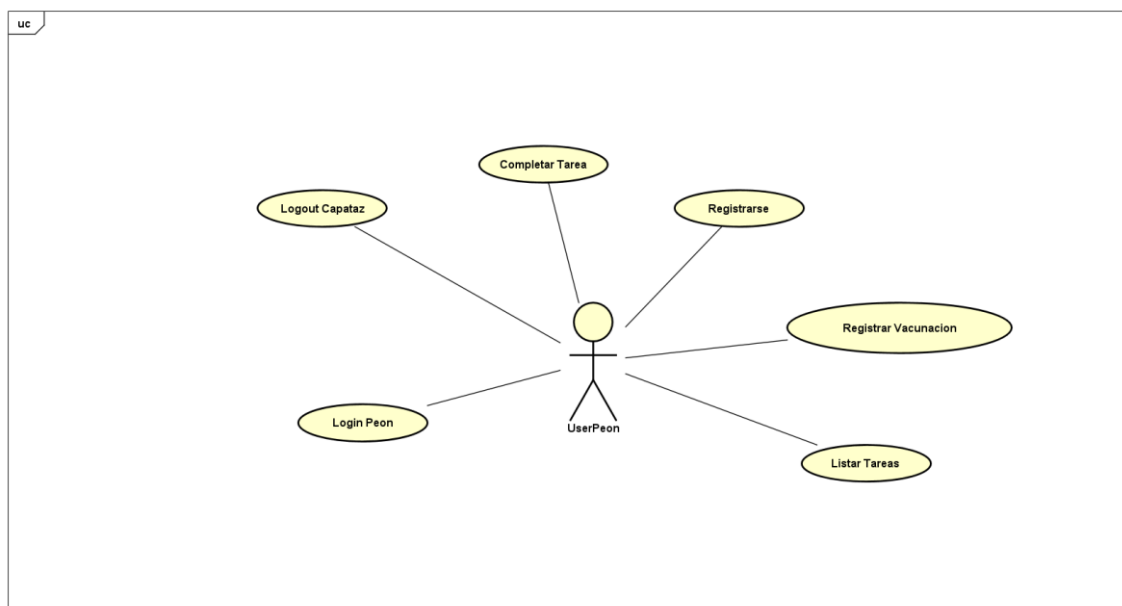
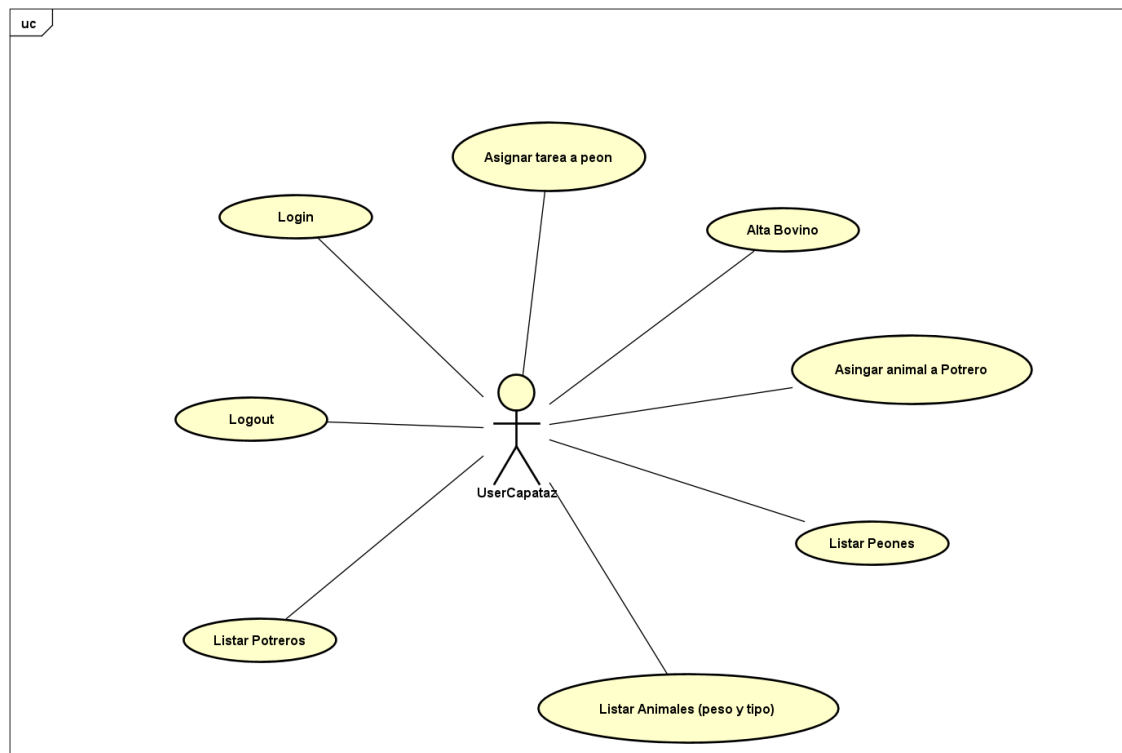
CONTENIDO

1.	Diagrama de clases (Uml) y Casos de Uso	5
2.	Precargas.	6
2.1	Ovinos.....	8
2.2	Bovinos.	9
2.3	Capataces.	11
2.4	Peones.	11
2.5	Tareas.	12
2.6	Potreros.	13
2.7	Inyecciones.	13
2.8	Vacunas.	14
3.	Código fuente.....	15
3.1	Estancia	15
3.1.1	Alimento.cs	15
3.1.2	Bovino.cs.....	15
3.1.3	Capataz.cs.....	17
3.1.4	Empleado.cs.....	17
3.1.5	IComparable.cs	20
3.1.6	Inyeccion.cs.....	20
3.1.7	IValidable.cs	21
3.1.8	Ovino.cs	22
3.1.9	Peon.cs.....	23
3.1.10	Potrero.cs.....	26
3.1.11	Res.cs	30
3.1.12	Sexo.cs	35
3.1.13	Sistema.cs	35
3.1.14	Tarea.cs.....	57
3.1.15	TareAAsignada.....	58
3.1.16	Vacuna.cs	61
3.2	Obligatorio.cs	62
3.2.1	Program.cs.....	62
3.2.2	Utilidades.cs.....	69
4	Interfaz de usuario (MVC)	71
4.1	Empleados	71
4.1.1	ListarParaAsignar.cshtml	71
4.1.2	PerfilPeon.cshtml.....	72

4.1.3	RegistrarPeon.cshtml.....	74
4.1.4	SeleccionarPeon.cshtml.....	75
4.1.5	TareasNoCompletadas.cshtml.....	76
4.1.6	VerPeones.cshtml.....	78
4.1.7	VerTareas.cshtml.....	79
4.2	Ganado.....	80
4.2.1	AltaBovino.cshtml.....	80
4.2.2	AnimalesPorPesoYTipo.cshtml.....	82
4.2.3	Index.cshtml.....	84
4.2.4	ResesLibresParaAsignar.cshtml.....	86
4.2.5	VacunarAnimal.cshtml.....	88
4.3	Home.....	90
4.3.1	Index.cshtml.....	90
4.3.2	Login.cshtml.....	90
4.4	Potreros.....	92
4.4.1	ListarPotreros.cshtml.....	92
4.4.2	ListarTodosLosPotreros.cshtml.....	93
4.5	Tareas.....	95
4.5.1	Alta.cshtml.....	95
4.5.2	Editar.cshtml.....	96
4.5.3	Shared.cshtml.....	97
4.6	Controllers.....	101
4.6.1	EmpleadosController.cs.....	101
4.6.2	GanadoController.cs.....	107
4.6.3	HomeController.cs.....	111
4.6.4	PotrerosControler.cs.....	113
4.6.5	TareasController.cs.....	115
5	Casos de uso.....	118
5.1	Capataz.....	118
5.1.1	Asignar Tarea.....	118
5.1.2	Alta Bovino.....	119
5.1.3	Login.....	120
5.1.4	Asignar Animal a un Potrero.....	121
5.1.5	Listar Peones.....	122
5.1.6	Listar animales por peso y tipo.....	123
5.1.7	Listar Potreros.....	123

5.1.8	Logout.....	124
5.2	Peón.....	125
5.2.1	Completar Tarea.	125
5.2.2	Registrar vacunación.	126
5.2.3	Listar Tareas.....	127
5.2.4	Login.....	127
5.2.5	Logout.....	128
5.2.6	Registrarse	129
6	Casos de Prueba	130
6.1	Peón.....	130
6.2	Capataz.....	131
7	Consultas IA	131
7.1	Vacunas.	132
7.2	Tareas	136
7.3	Peones	140
7.4	Bovinos	142
7.5	Ovinos.....	146
8	Publicación en Somee	149

[illegible]



2. PRECARGAS.

PreLoadVaccine();

PreLoadTask();

PreLoadPeon();

PreLoadTaskToPeon();

PreLoadCapataz();

```
PreLoadBovino();
```

```
PreLoadOvino();
```

```
PreLoadPotrero();
```

2.1 OVINOS

InstanceRes(new Ovino(30, "7654321A", Sexo.MALE, "Corriedale", _fechas[0], 50, 80, true, LoadInjectionTo()));

InstanceRes(new Ovino(2.5, "A2345678", Sexo.FEMALE, "Merino", _fechas[1], 150, 45, true, LoadInjectionTo()));

InstanceRes(new Ovino(3.2, "B9876543", Sexo.MALE, "Corriedale", _fechas[2], 220, 62, true, LoadInjectionTo()));

InstanceRes(new Ovino(1.8, "C1234567", Sexo.FEMALE, "Hampshire Down", _fechas[3], 120, 38, true, LoadInjectionTo()));

InstanceRes(new Ovino(2.7, "D8765432", Sexo.MALE, "Suffolk", _fechas[4], 180, 55, true, LoadInjectionTo()));

InstanceRes(new Ovino(2.4, "E2345678", Sexo.FEMALE, "Texel", _fechas[5], 165, 42, true, LoadInjectionTo()));

InstanceRes(new Ovino(3.1, "F9876543", Sexo.MALE, "Île-de-France", _fechas[6], 210, 60, true, LoadInjectionTo()));

InstanceRes(new Ovino(1.9, "G1234567", Sexo.FEMALE, "Romney Marsh", _fechas[7], 130, 36, true, LoadInjectionTo()));

InstanceRes(new Ovino(2.8, "H8765432", Sexo.MALE, "Lincoln", _fechas[8], 195, 58, true, LoadInjectionTo()));

InstanceRes(new Ovino(2.5, "I2345678", Sexo.FEMALE, "Border Leicester", _fechas[9], 170, 44, true, LoadInjectionTo()));

InstanceRes(new Ovino(3.2, "J9876543", Sexo.MALE, "Cheviot", _fechas[10], 230, 65, true, LoadInjectionTo()));

InstanceRes(new Ovino(1.8, "K1234567", Sexo.FEMALE, "Bluefaced Leicester", _fechas[11], 125, 35, true, LoadInjectionTo()));

InstanceRes(new Ovino(2.7, "L8765432", Sexo.MALE, "Swaledale", _fechas[12], 185, 57, true, LoadInjectionTo()));

InstanceRes(new Ovino(2.4, "M2345678", Sexo.FEMALE, "Dalesbred", _fechas[13], 160, 43, true, LoadInjectionTo()));

InstanceRes(new Ovino(3.1, "N9876543", Sexo.MALE, "Exmoor Horn", _fechas[14], 215, 61, true, LoadInjectionTo()));

InstanceRes(new Ovino(1.9, "O1234567", Sexo.FEMALE, "Scottish Blackface", _fechas[15], 135, 37, true, LoadInjectionTo()));

InstanceRes(new Ovino(2.8, "P8765432", Sexo.MALE, "Clun Forest", _fechas[16], 190, 59, true, LoadInjectionTo()));

InstanceRes(new Ovino(2.5, "Q2345678", Sexo.FEMALE, "Soay", _fechas[17], 175, 45, true, LoadInjectionTo()));

InstanceRes(new Ovino(3.2, "R9876543", Sexo.MALE, "Hebridean", _fechas[18], 235, 66, true, LoadInjectionTo()));

InstanceRes(new Ovino(1.8, "S1234567", Sexo.FEMALE, "Shetland", _fechas[19], 120, 34, true, LoadInjectionTo()));

InstanceRes(new Ovino(2.7, "T8765432", Sexo.MALE, "Arapawa", _fechas[20], 180, 56, true, LoadInjectionTo()));

InstanceRes(new Ovino(2.4, "U2345678", Sexo.FEMALE, "Columbia", _fechas[21], 165, 43, true, LoadInjectionTo()));

InstanceRes(new Ovino(3.1, "V9876543", Sexo.MALE, "Rambouillet", _fechas[22], 220, 64, true, LoadInjectionTo()));

InstanceRes(new Ovino(1.9, "W1234567", Sexo.FEMALE, "Merino Argentino", _fechas[23], 130, 36, true, LoadInjectionTo()));

InstanceRes(new Ovino(2.8, "X8765432", Sexo.MALE, "Corriedale Australiano", _fechas[24], 195, 58, true, LoadInjectionTo()));

InstanceRes(new Ovino(2.5, "Y2345678", Sexo.FEMALE, "Romney Marsh Neozelandés", _fechas[25], 170, 45, true, LoadInjectionTo()));

InstanceRes(new Ovino(3.2, "Z9876543", Sexo.MALE, "Tex", _fechas[26], 175, 40, true, LoadInjectionTo()));

InstanceRes(new Ovino(3.2, "Z9876544", Sexo.MALE, "Merilin", _fechas[27], 155, 40, false, LoadInjectionTo()));

InstanceRes(new Ovino(3.2, "Z9876545", Sexo.FEMALE, "Merino Australiano", _fechas[28], 145, 40, true, LoadInjectionTo()));

InstanceRes(new Ovino(3.2, "Z9876546", Sexo.MALE, "Romney Marsh", _fechas[29], 145, 40, false, LoadInjectionTo()));

2.2 BOVINOS.

InstanceRes(new Bovino(Alimento.GRANO, "1234567A", Sexo.FEMALE, "Angus", _fechas[0], 1500, 150, true, LoadInjectionTo()));

InstanceRes(new Bovino(Alimento.PASTURA, "7890123B", Sexo.FEMALE, "Hereford", _fechas[1], 1200, 180, true, LoadInjectionTo()));

InstanceRes(new Bovino(Alimento.GRANO, "4567890C", Sexo.MALE, "Holstein", _fechas[2], 1800, 120, false, LoadInjectionTo()));

InstanceRes(new Bovino(Alimento.PASTURA, "1234567D", Sexo.FEMALE, "Shorthorn", _fechas[3], 1000, 100, true, LoadInjectionTo()));

InstanceRes(new Bovino(Alimento.PASTURA, "7890123E", Sexo.MALE, "Limousin", _fechas[4], 1600, 130, false, LoadInjectionTo()));

InstanceRes(new Bovino(Alimento.GRANO, "7890123E", Sexo.MALE, "Limousin", _fechas[5], 1600, 130, false, LoadInjectionTo()));

```
InstanceRes(new Bovino(Alimento.PASTURA, "456789oF", Sexo.FEMALE, "Charolais",
_fechas[6], 1300, 160, true, LoadInjectionTo()));

InstanceRes(new Bovino(Alimento.GRANO, "1234567G", Sexo.MALE, "Gelbvieh", _fechas[7],
1700, 180, true, LoadInjectionTo()));

InstanceRes(new Bovino(Alimento.PASTURA, "7890123H", Sexo.FEMALE, "Maine",
_fechas[8], 1100, 120, true, LoadInjectionTo()));

InstanceRes(new Bovino(Alimento.PASTURA, "456789oI", Sexo.MALE, "South Devon",
_fechas[9], 1550, 120, false, LoadInjectionTo()));

InstanceRes(new Bovino(Alimento.GRANO, "1234567J", Sexo.FEMALE, "Red Angus",
_fechas[10], 1250, 150, true, LoadInjectionTo()));

InstanceRes(new Bovino(Alimento.PASTURA, "7890123K", Sexo.MALE, "Wagyu",
_fechas[11], 2000, 155, false, LoadInjectionTo()));

InstanceRes(new Bovino(Alimento.PASTURA, "456789oL", Sexo.FEMALE, "Belted Galloway",
_fechas[12], 1150, 135, true, LoadInjectionTo()));

InstanceRes(new Bovino(Alimento.GRANO, "1234567M", Sexo.MALE, "Texas Longhorn",
_fechas[13], 1650, 170, false, LoadInjectionTo()));

InstanceRes(new Bovino(Alimento.PASTURA, "7890123N", Sexo.FEMALE, "Brahman",
_fechas[14], 1350, 170, true, LoadInjectionTo()));

InstanceRes(new Bovino(Alimento.GRANO, "456789oO", Sexo.MALE, "Murray Grey",
_fechas[15], 1850, 130, false, LoadInjectionTo()));

InstanceRes(new Bovino(Alimento.PASTURA, "1234567P", Sexo.FEMALE, "Highland",
_fechas[16], 1200, 140, false, LoadInjectionTo()));

InstanceRes(new Bovino(Alimento.GRANO, "7890123Q", Sexo.MALE, "Chianina", _fechas[17],
1700, 190, true, LoadInjectionTo()));

InstanceRes(new Bovino(Alimento.PASTURA, "456789oR", Sexo.FEMALE, "Simmental",
_fechas[18], 1400, 190, true, LoadInjectionTo()));

InstanceRes(new Bovino(Alimento.PASTURA, "1234567S", Sexo.FEMALE, "Sussex",
_fechas[19], 1900, 190, true, LoadInjectionTo()));

InstanceRes(new Bovino(Alimento.GRANO, "7890123T", Sexo.MALE, "Devon", _fechas[20],
1050, 190, true, LoadInjectionTo()));

InstanceRes(new Bovino(Alimento.PASTURA, "456789oU", Sexo.FEMALE, "Galloway",
_fechas[21], 1500, 165, true, LoadInjectionTo()));

InstanceRes(new Bovino(Alimento.GRANO, "1234567V", Sexo.MALE, "Brangus", _fechas[22],
1200, 195, true, LoadInjectionTo()));

InstanceRes(new Bovino(Alimento.GRANO, "7890123W", Sexo.FEMALE, "Beefmaster",
_fechas[23], 1300, 185, true, LoadInjectionTo()));

InstanceRes(new Bovino(Alimento.PASTURA, "7890123X", Sexo.MALE, "Angus", _fechas[24],
1300, 140, true, LoadInjectionTo()));
```

```
InstanceRes(new Bovino(Alimento.PASTURA, "7890123Y", Sexo.FEMALE, "Aberdeen angus",  
_fechas[25], 1300, 190, true, LoadInjectionTo()));
```

```
InstanceRes(new Bovino(Alimento.GRANO, "7890123Z", Sexo.MALE, "Charolais",  
_fechas[26], 1300, 1550, true, LoadInjectionTo()));
```

```
InstanceRes(new Bovino(Alimento.PASTURA, "7890123M", Sexo.FEMALE, "Normando",  
_fechas[27], 1300, 166, true, LoadInjectionTo()));
```

```
InstanceRes(new Bovino(Alimento.GRANO, "7890123N", Sexo.MALE, "Holando", _fechas[28],  
1300, 180, true, LoadInjectionTo()));
```

```
InstanceRes(new Bovino(Alimento.PASTURA, "7890123L", Sexo.FEMALE, "Jersey",  
_fechas[29], 1300, 185, true, LoadInjectionTo()));
```

2.3 CAPATACES.

```
InstanceCapataz( new Capataz(LoadPeonACargo(), "jose.perez@estancia.com",  
"CapEstancia123", "Jose Perez", new DateTime(2015, 01, 01)));
```

```
InstanceCapataz( new Capataz(LoadPeonACargo(), "juan.rodriguez@estancia.com",  
"CapEstancia456", "Juan Rodriguez", new DateTime(2016, 06, 01)));
```

2.4 PEONES.

```
InstanceEmpleados(new Peon(true, false, "john.smith@estancia.com", "password123", "John  
Smith", new DateTime(2020, 10, 15)));
```

```
InstanceEmpleados(new Peon(true, false, "jane.doe@estancia.com", "password456", "Jane  
Doe", new DateTime(2019, 12, 01)));
```

```
InstanceEmpleados(new Peon(true, false, "peter.jones@estancia.com", "password789",  
"Peter Jones", new DateTime(2024, 01, 04)));
```

```
InstanceEmpleados(new Peon(true, false, "mary.williams@estancia.com", "password012",  
"Mary Williams", new DateTime(2023, 03, 22)));
```

```
InstanceEmpleados(new Peon(true, false, "david.miller@estancia.com", "password345",  
"David Miller", new DateTime(2022, 05, 18)));
```

```
InstanceEmpleados(new Peon(true, false, "sarah.taylor@estancia.com", "password678",  
"Sarah Taylor", new DateTime(2024, 02, 11)));
```

```
InstanceEmpleados(new Peon(true, false, "michael.brown@estancia.com", "password901",  
"Michael Brown", new DateTime(2023, 04, 07)));
```

```
InstanceEmpleados(new Peon(true, false, "laura.davis@estancia.com", "password234",  
"Laura Davis", new DateTime(2022, 06, 24)));
```

```
InstanceEmpleados(new Peon(true, false, "richard.thomas@estancia.com", "password567",  
"Richard Thomas", new DateTime(2018, 03, 09)));
```

```
InstanceEmpleados(new Peon(true, false, "elizabeth.anderson@estancia.com",  
"password890", "Elizabeth Anderson", new DateTime(2021, 05, 13)));
```

2.5 TAREAS.

InstanceTask(new Tarea("Inspeccionar el sistema de alarma contra incendios en el edificio principal."));

InstanceTask(new Tarea("Verificar el funcionamiento de las luces de emergencia en todas las salidas."));

InstanceTask(new Tarea("Realizar pruebas de fugas de gas en todas las áreas de almacenamiento. "));

InstanceTask(new Tarea("Calibrar los manómetros de presión en las calderas"));

InstanceTask(new Tarea("Inspeccionar el estado de las barandas y pasamanos en las escaleras"));

InstanceTask(new Tarea("Revisar los extintores de incendios para garantizar su funcionamiento adecuado."));

InstanceTask(new Tarea("Verificar que las señales de seguridad y emergencia estén visibles y en buen estado."));

InstanceTask(new Tarea("Realizar una inspección general de las instalaciones en busca de posibles peligros."));

InstanceTask(new Tarea("Documentar los hallazgos de las inspecciones y tomar las medidas correctivas necesarias."));

InstanceTask(new Tarea("Capacitar al personal sobre los procedimientos de seguridad en caso de emergencias."));

InstanceTask(new Tarea("Realizar simulacros de incendio y evacuación para evaluar la preparación del personal."));

InstanceTask(new Tarea("Mantener registros actualizados de las tareas de mantenimiento y seguridad realizadas."));

InstanceTask(new Tarea("Comunicar los riesgos de seguridad y los procedimientos de emergencia a todo el personal."));

InstanceTask(new Tarea("Investigar los incidentes de seguridad y tomar medidas para prevenir su recurrencia."));

InstanceTask(new Tarea("Implementar programas de mejora continua para garantizar la seguridad en el lugar de trabajo."));

2.6 POTREROS.

```
InstancePotrero(new Potrero("Potrero 1", random.Next(1, 10)));  
InstancePotrero(new Potrero("Potrero 2", random.Next(1, 10)));  
InstancePotrero(new Potrero("Potrero 3", random.Next(1, 10)));  
InstancePotrero(new Potrero("Potrero 4", random.Next(1, 10)));  
InstancePotrero(new Potrero("Potrero 5", random.Next(1, 10)));  
InstancePotrero(new Potrero("Potrero 6", random.Next(1, 10)));  
InstancePotrero(new Potrero("Potrero 7", random.Next(1, 10)));  
InstancePotrero(new Potrero("Potrero 8", random.Next(1, 10)));  
InstancePotrero(new Potrero("Potrero 9", random.Next(1, 10)));  
InstancePotrero(new Potrero("Potrero 10", random.Next(1, 10)));
```

2.7 INYECCIONES.

```
private List<Inyeccion> LoadInjectionTo()  
{  
    List<Inyeccion> injectionTo = new List<Inyeccion>();  
    List<Inyeccion> injection = new List<Inyeccion>();  
    Random random = new Random();  
    DateTime fechaAleatoria;  
  
    for (int i = 0; i < 8; i++)  
    {  
  
        fechaAleatoria = new DateTime(DateTime.Now.Year, random.Next(1, 13),  
random.Next(1, 29));  
  
        Vacuna vacunaGenerada = _vacunas[random.Next(0, _vacunas.Count)];  
  
        Inyeccion inyeccion = new Inyeccion(vacunaGenerada, fechaAleatoria,  
fechaAleatoria.AddYears(1));  
  
        ValidarInyeccion(inyeccion);  
  
        injection.Add(inyeccion);  
    }  
  
    for(int i= 0; i < random.Next(0,9); i++)
```

```

{
    injectionTo.Add(injection[i]);
}

return injectionTo;
}

```

2.8 VACUNAS.

InstanceVaccine(new Vacuna("Fiebre Aftosa", "Enfermedad viral altamente contagiosa que afecta a bovinos, ovinos y porcinos. Se caracteriza por fiebre, vesículas en la boca y patas, y dificultad para caminar.", "Virus de la Fiebre Aftosa"));

InstanceVaccine(new Vacuna("Brucelosis", "Enfermedad bacteriana que afecta el sistema reproductivo de bovinos y ovinos. Provoca abortos, infertilidad y nacimiento de terneros débiles.", "Bacteria Brucella abortus"));

InstanceVaccine(new Vacuna("Carbón", "Enfermedad bacteriana grave que afecta a bovinos, ovinos y otros animales. Se caracteriza por muerte súbita, hinchazón y sangre en la nariz y boca.", "Bacteria Clostridium chauvoei"));

InstanceVaccine(new Vacuna("Manquera", "Enfermedad viral que afecta las patas de bovinos y ovinos. Provoca cojera, úlceras en las patas y disminución de la producción de leche.", "Virus de la Manquera"));

InstanceVaccine(new Vacuna("Leptospirosis", "Enfermedad bacteriana que afecta a bovinos y ovinos. Provoca abortos, infertilidad, problemas renales y hepáticos.", "Bacteria Leptospira spp"));

InstanceVaccine(new Vacuna("Rabia", "Enfermedad viral mortal que afecta a todos los mamíferos, incluyendo bovinos y ovinos. Se caracteriza por síntomas neurológicos como agresividad, parálisis y muerte.", "Virus de la Rabia"));

InstanceVaccine(new Vacuna("Diarrea Viral Bovina", "Enfermedad viral que afecta a bovinos de todas las edades. Provoca diarrea, neumonía, y disminución de la producción de leche.", "Virus de la Diarrea Viral Bovina"));

InstanceVaccine(new Vacuna("Parainfluenza Bovina", "Enfermedad viral que afecta a bovinos de todas las edades. Provoca problemas respiratorios, especialmente en terneros", "Virus Parainfluenza Bovino tipo 3"));

InstanceVaccine(new Vacuna("Rotavirus", "Enfermedad viral que afecta a terneros jóvenes. Provoca diarrea severa, deshidratación y muerte en algunos casos.", "Rotavirus bovino"));

InstanceVaccine(new Vacuna("Rinotraqueítis Infecciosa Bovina", "Enfermedad viral que afecta a bovinos de todas las edades. Provoca problemas respiratorios, abortos y disminución de la producción de leche.", "Virus de la Rinotraqueítis Infecciosa Bovina"));

```

}

```

3. CÓDIGO FUENTE.

3.1 ESTANCIA

3.1.1 ALIMENTO.CS

```
public enum Alimento
```

```
{
```

```
    GRANO = 1,
```

```
    PASTURA
```

```
}
```

3.1.2 BOVINO.CS

```
namespace Estancia
```

```
{
```

```
    public class Bovino : Res
```

```
    {
```

```
        private Alimento _tipoAlimento;
```

```
        private static decimal s_precioPorkgEnPie = 100;
```

```
        private static decimal s_costoAlimentacion = 200;
```

```
        public Alimento Alimento
```

```
        {
```

```
            get { return _tipoAlimento; }
```

```
        }
```

```
        public Bovino(Alimento alimento, string id, Sexo sexo, string raza, DateTime  
fechaNacimiento, decimal costoAdquisicion, double pesoActual, bool esHibrido,  
List<Inyeccion> vacunas) : base(id, sexo, raza, fechaNacimiento, costoAdquisicion,  
pesoActual, esHibrido, vacunas)
```

```
        {
```

```
            _tipoAlimento = alimento;
```

```
        }
```

```
        public Bovino() {}
```

```
public override decimal CostoCrianza()
{
    return s_costoAlimentacion + CostoAdquisicion;
}
```

```
public override string TipoRes()
{
    return "Bovino";
}
```

```
public override bool Equals(object? obj)
{
    if(obj == null || !(obj is Bovino))
        return false;

    Bovino otroBovino = (Bovino)obj;

    return Id == otroBovino.Id;
}
```

```
public override decimal GananciaPotencial()
{
    //Precio base
    decimal ganancia = (s_precioPorkgEnPie * (decimal)PesoActual);

    //Si se alimento a grano aumento 30%
    if (_tipoAlimento == Alimento.GRANO)
        ganancia *= 1.30M;

    //Si además es hembra aumento 10%
    if (Sexo == Sexo.FEMALE)
        ganancia *= 1.10M;
```



```

        return s_precioPorkgEnPie * (decimal)PesoActual;
    }
}

```

3.1.3 CAPATAZ.CS

```
namespace Estancia
```

```

{
    public class Capataz : Empleado
    {
        List<Peon> _personasACargo = new List<Peon>();

        public Capataz() {}

        public Capataz(List<Peon> personasACargo, string email, string contrasenia, string
nombre, DateTime fechaIngreso) : base(email, contrasenia, nombre, fechaIngreso)
        {
            _personasACargo = personasACargo;
        }

        public override string Rol()
        {
            return "capataz";
        }
    }
}

```

3.1.4 EMPLEADO.CS

```
namespace Estancia
```

```

{
    public class Empleado : IValidable
    {
        private int _id;

        private static int s_ultimoid = 1;
    }
}

```

```
private string _email;  
private string _contrasenia;  
private string _nombre;  
private DateTime _fechaIngreso;
```

```
public string Nombre  
{  
    get { return _nombre; }  
    set { _nombre = value; }  
}
```

```
public string Contraseña  
{  
    get { return _contrasenia; }  
    set { _contrasenia = value; }  
}
```

```
public string Email  
{  
    get { return _email; }  
    set { _email = value; }  
}
```

```
public DateTime FechaIngreso  
{  
    get { return _fechaIngreso; }  
}
```

```
public Empleado() {}  
public Empleado(string email, string contraseña, string nombre, DateTime fechaIngreso)  
{
```

```
_id = s_ultimold++;  
_email = email;  
_contrasenia = contrasenia;  
_nombre = nombre;  
_fechaIngreso = fechaIngreso;  
}
```

```
public void Validar()  
{  
    if (string.IsNullOrEmpty(_contrasenia) || _contrasenia.Length < 8)  
        throw new Exception("Contraseña no cumple con los requisitos");  
}
```

```
public override string ToString()  
{  
    return $"Empleado: {_nombre}\n" +  
        $"Gmail {_email} \n" +  
        $"Fecha de Ingreso: {_fechaIngreso}\n";  
}
```

```
public virtual string Rol()  
{  
    return "";  
}
```

```
public override bool Equals(object? obj)  
{  
    return obj is Empleado otroE && _email == otroE._email;  
}
```

```
}  
}
```

3.1.5 ICOMPARABLE.CS

namespace Estancia

```
{  
    public interface IComparable  
    {  
        public void CompareTo(object obj);  
    }  
}
```

3.1.6 INYECCION.CS

public class Inyeccion : IValidable

```
{  
    private Vacuna _tipoVacuna;  
    private DateTime _fechaDeVacunacion;  
    private DateTime _vencimiento; //se vence luego de un año  
    private bool _estaPrecargada;  
  
    public bool EstaPrecargada  
    {  
        get { return _estaPrecargada; }  
        set { _estaPrecargada = value; }  
    }  
  
    public Inyeccion(Vacuna tipoVacuna, DateTime fechaDeVacunacion, DateTime  
vencimiento)  
    {  
        _tipoVacuna = tipoVacuna;  
        _fechaDeVacunacion = fechaDeVacunacion;  
        _vencimiento = vencimiento;  
        _estaPrecargada = false;  
    }  
}
```

```

public Inyeccion()
{
    _tipoVacuna = new Vacuna();
    _fechaDeVacunacion = new DateTime();
    _vencimiento = new DateTime();
    _estaPrecargada = false;
}

// Revisar letra si necesitamos algún otro control
public void Validar()
{
    if (_tipoVacuna == null)
        throw new Exception("No existe vacuna");
}
}

```

3.1.7 IVALIDABLE.CS

namespace Estancia

```

{
    public interface IValidable
    {
        public void Validar();
    }
}

```

3.1.8 OVINO.CS

namespace Estancia

{

public class Ovino : Res

{

private double _pesoEstimadoLana;

private static decimal s_precioPorKgLana = 50;

private static decimal s_precioPorKgEnPie = 30;

private static decimal s_costoAlimentacion = 100;

public decimal PrecioKiloLana

{

get { return s_precioPorKgLana; }

set { s_precioPorKgLana = value; }

}

public Ovino(double pesoEstimadoLana, string id, Sexo sexo, string raza, DateTime fechaNacimiento, decimal costoAdquisicion, double pesoActual, bool esHibrido, List<Inyeccion> vacunas) : base(id, sexo, raza, fechaNacimiento, costoAdquisicion, pesoActual, esHibrido, vacunas)

{

_pesoEstimadoLana = pesoEstimadoLana;

}

public override decimal CostoCrianza()

{

return s_costoAlimentacion + CostoAdquisicion;

}

public static decimal GetPrecioActualKgLana()

{

return s_precioPorKgLana;

}

```

public static void SetNewPrecioKgLana(decimal newPrecio)
{
    s_precioPorKgLana = newPrecio;
}

public override string TipoRes()
{
    return "Ovino";
}

public override decimal GananciaPotencial()
{
    //Precio base
    decimal ganancia = ((decimal)_pesoEstimadoLana * s_precioPorKgLana) +
(s_precioPorKgEnPie * (decimal)PesoActual);

    //Si es hibrido le resto el 5%
    if (EsHibrido)
        ganancia *= 0.95M;

    return ganancia;
}
}
}

```

3.1.9 PEON.CS

```
namespace Estancia
```

```

{
    public class Peon : Empleado
    {
        private bool _esResidente;
        private List<TareaAsignada> _tarefas;
        private bool _tieneCapataz;
    }
}

```

```
public bool TieneCapataz
{
    get { return _tieneCapataz; }
    set { _tieneCapataz = value; }
}
```

```
public bool EsResidente
{
    set { _esResidente = value; }
    get { return _esResidente; }
}
```

```
public List<TareaAsignada> TareasAsignadas
{
    set { _tareas = value; }
    get { return _tareas; }
}
```

```
public Peon() { }
```

```
public Peon(bool esResidente, List<TareaAsignada> tareas, bool tieneCapataz, string
email, string contrasenia, string nombre, DateTime fechaIngreso) : base(email, contrasenia,
nombre, fechaIngreso)
```

```
{
    _esResidente = esResidente;
    _tareas = tareas;
    _tieneCapataz = tieneCapataz;
}
```

```
public Peon(bool esResidente, bool tieneCapataz, string email, string contrasenia, string
nombre, DateTime fechaIngreso) : base(email, contrasenia, nombre, fechaIngreso)
```

```
{
```



```

        _esResidente = esResidente;

        _tieneCapataz = tieneCapataz;

        _tareas = new List<TareaAsignada>();
    }

    internal void AsignarTarea(TareaAsignada tarea)
    {
        _tareas.Add(tarea);
    }

    public override string Rol()
    {
        return "peon";
    }

    public TareaAsignada BuscarTareaAsignada(int id)
    {
        TareaAsignada buscada = null;
        int i = 0;
        while(buscada == null && i < _tareas.Count)
        {
            if (_tareas[i].Tarea.Id == id)
            {
                buscada = _tareas[i];
            }
            i++;
        }
        if (buscada == null)
            throw new Exception("Tarea no encontrada");

        return buscada;
    }

```

```

    }

    public void actualizarTareaAsignada(TareaAsignada tareaA)
    {
        tareaA.Validar();
        TareaAsignada buscada = BuscarTareaAsignada(tareaA.Tarea.Id);
        this.TareasAsignadas.Remove(buscada);
        this.TareasAsignadas.Add(tareaA);
    }
}
}

```

3.1.10 POTRERO.CS

```

namespace Estancia
{
    public class Potrero : IValidable, IComparable<Potrero>
    {
        private int _id;
        private static int s_ultimold = 1;
        private string _descripcion;
        private int _cantHectarias;
        private int _capacidadAnimales;
        private List<Res> _cantPastando = new List<Res>();

        public int Id
        {
            get { return _id; }
        }

        public string Descripcion
        {
            get { return _descripcion; }
        }
    }
}

```

```
}

public int CantidadHectareas
{
    get { return _cantHectarias; }
}

public int CapacidadMaxima
{
    get { return _capacidadAnimales; }
}

public int CapacidadAnimales
{
    get { return _capacidadAnimales; }
}

public int CantidadAnimales
{
    get { return _cantPastando.Count; }
}

public Potrero(string descripcion, int cantHectarias, int capacidadAnimales = 10)
{
    _id = s_ultimold++;
    _descripcion = descripcion;
    _cantHectarias = cantHectarias;
    _capacidadAnimales = capacidadAnimales;
    //_cantPastando = cantPastando;
}

public decimal CostoTotalCrianza()
{
```

```

        decimal costoTotal = 0;

        foreach(Res res in _cantPastando)
        {
            costoTotal += res.GananciaPotencial() - (res.CostoCrianza() +
res.CostoTotalVacunas());
        }

        return costoTotal;
    }

```

```

public void AssignAnimalCapacity()
{
    //Asigno la capacidad de los animales respecto a la cantidad de hectarias
    _capacidadAnimales = _cantHectarias * 2;
}

```

```

public void AddRes(Res unRes)
{
    if (!HayEspacio())
        throw new Exception("Potrero lleno");
    _cantPastando.Add(unRes);
}

```

```

public bool HayEspacio()
{
    return _capacidadAnimales > _cantPastando.Count;
}

```

```

public void Validar()
{
    if (_cantHectarias > 10)
        throw new Exception("Los potreros no pueden tener más de 10 hectareas");
}

```

```
}
```

```
internal void AssignAnimals(List<Res> freeAnimals)
```

```
{
```

```
    int aux = 0;
```

```
    while( (_capacidadAnimales > _cantPastando.Count) && (aux<60) )
```

```
    {
```

```
        if (freeAnimals[aux].Libre == true)
```

```
        {
```

```
            _cantPastando.Add(freeAnimals[aux]);
```

```
            freeAnimals[aux].Libre = false;
```

```
        }
```

```
        aux++;
```

```
    }
```

```
}
```

```
public override string ToString()
```

```
{
```

```
    return $"Identificador: {_id}\n" +
```

```
        $"_Descripción: {_descripcion}\n" +
```

```
        $"_Héctareas: {_cantHectarias}\n" +
```

```
        $"_Capacidad: {_capacidadAnimales}\n";
```

```
}
```

```
internal bool AreaMayorACantHectareasYCapacidadMaxSuperiorNroDado(int hectareas,  
int valor)
```

```
{
```

```
    return this.CantidadHectareas > hectareas && this.CapacidadMaxima > valor;
```

```
}
```

```
public int CompareTo(Potrero otroP)
```

```
{
```

```

        if (_capacidadAnimales.CompareTo(otroP._capacidadAnimales) == 0)
            return _cantPastando.Count.CompareTo(otroP._cantPastando.Count) * -1;
        return _capacidadAnimales.CompareTo(otroP._capacidadAnimales);
    }
}
}

```

3.1.11 RES.CS

```

namespace Estancia
{
    public abstract class Res : IValidable, IComparable<Res>
    {
        private string _numeroCaravana;
        private Sexo _sexo;
        private string _raza;
        private DateTime _fechaNacimiento;
        private decimal _costoAdquisicion;
        private double _pesoActual;
        private bool _esHibrido;
        private List<Inyeccion> _vacunas;
        private bool _esLibre;

        public string Id
        {
            get { return _numeroCaravana; }
            set { _numeroCaravana = value; }
        }

        public string Raza
        {
            get { return _raza; }
            set { _raza = value; }
        }
    }
}

```

```
}

public double PesoActual
{
    get { return _pesoActual; }
    set { _pesoActual = value; }
}

public Sexo Sexo
{
    get { return _sexo; }
    set { _sexo = value; }
}

public bool Libre
{
    get { return _esLibre; }
    set { _esLibre = value; }
}

public DateTime FechaNacimiento
{
    get { return _fechaNacimiento; }
    set { _fechaNacimiento = value; }
}

public decimal CostoAdquisicion
{
    get { return _costoAdquisicion; }
    set { _costoAdquisicion = value; }
}

public bool EsHibrido
{
```

```
    get { return _esHibrido; }  
}
```

```
public List<Inyeccion> Inyecciones  
{  
    get { return _vacunas; }  
    set { _vacunas = value; }  
}
```

```
public Res(string numeroCaravana, Sexo sexo, string raza, DateTime fechaNacimiento,  
decimal costoAdquisicion, double pesoActual, bool esHibrido, List<Inyeccion> vacunas)  
{  
    _numeroCaravana = numeroCaravana;  
    _sexo = sexo;  
    _raza = raza;  
    _fechaNacimiento = fechaNacimiento;  
    _costoAdquisicion = costoAdquisicion;  
    _pesoActual = pesoActual;  
    _esHibrido = esHibrido;  
    _vacunas = vacunas;  
    _esLibre = true;  
}
```

```
public Res() {}
```

```
public decimal CostoTotalVacunas()  
{  
    return _vacunas.Count * 200;  
}
```

```
public virtual int CompareTo(Res animal)
```



```

{
    if(animal.PesoActual == _pesoActual) return 0;
    else if(animal.PesoActual < _pesoActual) return 1;
    return -1;
}

#region validar
public virtual void Validar()
{
    string numeros = "0123456789";
    int contieneNumero = 0;

    if (_numeroCaravana.Length != 8 || string.IsNullOrEmpty(_numeroCaravana))
        throw new Exception("El código identificador no puede ser vacio ni de largo distinto a
8");
    for (int i = 0; i < _numeroCaravana.Length; i++)
    {
        for(int j = 0; j < numeros.Length; j++)

            if (numeros[j] == _numeroCaravana[i])
                contieneNumero++;
    }
    if (contieneNumero == 8 || contieneNumero == 0)
        throw new Exception("La caravana debe ser alfanumerica");

}

#endregion

public bool EsMayorATresMeses()
{
    TimeSpan resta = DateTime.Now - _fechaNacimiento;

```

```

        if (resta.Days < 90)
            throw new Exception("No se puede vacunar, es menor a 3 meses");
        return true;
    }

    public abstract decimal CostoCrianza();

    public abstract string TipoRes();

    public abstract decimal GananciaPotencial();

    public override bool Equals(object? obj)
    {
        if(obj == null && !(obj is Res))
            return true;

        Res unaRes = (Res)obj;

        if (unaRes._numeroCaravana == _numeroCaravana)
            return true;
        return false;
    }

    public override string ToString()
    {
        return $"Número de caravana: {Id}\n" +
            $"_Raza: {Raza}\n" +
            $"_Peso actual: {PesoActual}\n" +
            $"_Sexo: {Sexo}\n";
    }
}

```

3.1.12 SEXO.CS

```
public enum Sexo
```

```
{  
    MALE = 1,  
    FEMALE  
}
```

3.1.13 SISTEMA.CS

```
using static System.Runtime.InteropServices.JavaScript.JSType;
```

```
namespace Estancia
```

```
{  
    public class Sistema  
    {  
        #region singleton  
        // Atributo privado  
        private static Sistema s_instance;  
  
        // Listas  
        private List<Empleado> _empleados;  
        private List<Potrero> _potreros;  
        private List<Res> _reses;  
        private List<Vacuna> _vacunas;  
        private List<Inyeccion> _inyeccionesToPreLoad;  
        private List<Tarea> _tareas;  
        private List<Peon> _registrados;  
        // Propiedad  
        public static Sistema Instance  
        {  
            get  
            {  
                if (s_instance == null)
```

```
        s_instance = new Sistema();  
        return s_instance;  
    }  
}  
#endregion
```

// Constructor

```
private Sistema()  
{  
    _empleados = new List<Empleado>();  
    _potreros = new List<Potrero>();  
    _reses = new List<Res>();  
    _vacunas = new List<Vacuna>();  
    _tareas = new List<Tarea>();  
    PrecargarDatos();  
}
```

private void PrecargarDatos()

```
{  
    PreLoadVaccine();  
    PreLoadTask();  
    PreLoadPeon();  
    PreLoadTaskToPeon();  
    PreLoadCapataz();  
    PreLoadOvino();  
    PreLoadBovino();  
    PreLoadPotrero();  
}
```

private void PreLoadTaskToPeon()

```

{
    foreach (Empleado unEmpleado in _empleados)
        if (unEmpleado is Peon peon)
        {
            for (int i = 0; i < 15; i++)
            {
                Tarea tareaRandom = RandomTaskToPeon();
                DateTime fechaAleatoria = FechaRandom();
                peon.AsignarTarea(tareaRandom, fechaAleatoria);
            }
        }
}

```

```

private DateTime FechaRandom()
{
    DateTime fechaAleatoria;
    Random random = new Random();
    //Creo fechas aleatorias, excepción para el mes 2(febrero).
    int mes = random.Next(1, 13);
    if (mes == 2)
    {
        fechaAleatoria = new DateTime(2024, mes, random.Next(1, 29));
    }
    else
    {
        fechaAleatoria = new DateTime(2024, mes, random.Next(1, 31));
    }
    return fechaAleatoria;
}

```

```

private List<Inyeccion> LoadInjectionTo()
{
    List<Inyeccion> injectionTo = new List<Inyeccion>();
    List<Inyeccion> injection = new List<Inyeccion>();
    Random random = new Random();
    DateTime fechaAleatoria;

    for (int i = 0; i < 8; i++)
    {

        fechaAleatoria = new DateTime(DateTime.Now.Year, random.Next(1, 13),
random.Next(1, 29));

        Vacuna vacunaGenerada = _vacunas[random.Next(0, _vacunas.Count)];
        Inyeccion inyeccion = new Inyeccion(vacunaGenerada, fechaAleatoria,
fechaAleatoria.AddYears(1));
        ValidarInyeccion(inyeccion);
        injection.Add(inyeccion);
    }

    for (int i = 0; i < random.Next(0, 9); i++)
    {
        injectionTo.Add(injection[i]);
    }
    return injectionTo;
}

private void ValidarInyeccion(Inyeccion inyeccion)
{
    inyeccion.Validar();
}

private void PreLoadCapataz()
{

```

```

        InstanceCapataz(new Capataz(LoadPeonACargo(), "jose.perez@estancia.com",
"CapEstancia123", "Jose Perez", new DateTime(2015, 01, 01)));

        InstanceCapataz(new Capataz(LoadPeonACargo(), "juan.rodriguez@estancia.com",
"CapEstancia456", "Juan Rodriguez", new DateTime(2016, 06, 01)));
    }

    private void InstanceCapataz(Capataz capataz)
    {
        ValidarEmpleado(capataz);
        _empleados.Add(capataz);
    }

    private void PreLoadOvino()
    {
        // Creo fechas randoms en una lista _fechas para utilizar en la precarga
        List<DateTime> _fechas = new List<DateTime>();

        DateTime fechaNacimiento;
        Random random = new Random();

        for (int i = 0; i < 30; i++)
        {
            fechaNacimiento = new DateTime(2024, random.Next(1, 12), random.Next(1, 28));
            if (fechaNacimiento.Month > DateTime.Now.Month)
            {
                i--;
            }
            else
            {
                _fechas.Add(fechaNacimiento);
            }
        }
    }

    InstanceRes(new Ovino(30, "7654321A", Sexo.MALE, "Corriedale", _fechas[0], 50, 80,
true, LoadInjectionTo()));

    InstanceRes(new Ovino(2.5, "A2345678", Sexo.FEMALE, "Merino", _fechas[1], 150, 45,
true, LoadInjectionTo()));

    InstanceRes(new Ovino(3.2, "B9876543", Sexo.MALE, "Corriedale", _fechas[2], 220, 62,
true, LoadInjectionTo()));

```

InstanceRes(new Ovino(1.8, "C1234567", Sexo.FEMALE, "Hampshire Down", _fechas[3], 120, 38, true, LoadInjectionTo()));

InstanceRes(new Ovino(2.7, "D8765432", Sexo.MALE, "Suffolk", _fechas[4], 180, 55, true, LoadInjectionTo()));

InstanceRes(new Ovino(2.4, "E2345678", Sexo.FEMALE, "Texel", _fechas[5], 165, 42, true, LoadInjectionTo()));

InstanceRes(new Ovino(3.1, "F9876543", Sexo.MALE, "Île-de-France", _fechas[6], 210, 60, true, LoadInjectionTo()));

InstanceRes(new Ovino(1.9, "G1234567", Sexo.FEMALE, "Romney Marsh", _fechas[7], 130, 36, true, LoadInjectionTo()));

InstanceRes(new Ovino(2.8, "H8765432", Sexo.MALE, "Lincoln", _fechas[8], 195, 58, true, LoadInjectionTo()));

InstanceRes(new Ovino(2.5, "I2345678", Sexo.FEMALE, "Border Leicester", _fechas[9], 170, 44, true, LoadInjectionTo()));

InstanceRes(new Ovino(3.2, "J9876543", Sexo.MALE, "Cheviot", _fechas[10], 230, 65, true, LoadInjectionTo()));

InstanceRes(new Ovino(1.8, "K1234567", Sexo.FEMALE, "Bluefaced Leicester", _fechas[11], 125, 35, true, LoadInjectionTo()));

InstanceRes(new Ovino(2.7, "L8765432", Sexo.MALE, "Swaledale", _fechas[12], 185, 57, true, LoadInjectionTo()));

InstanceRes(new Ovino(2.4, "M2345678", Sexo.FEMALE, "Dalesbred", _fechas[13], 160, 43, true, LoadInjectionTo()));

InstanceRes(new Ovino(3.1, "N9876543", Sexo.MALE, "Exmoor Horn", _fechas[14], 215, 61, true, LoadInjectionTo()));

InstanceRes(new Ovino(1.9, "O1234567", Sexo.FEMALE, "Scottish Blackface", _fechas[15], 135, 37, true, LoadInjectionTo()));

InstanceRes(new Ovino(2.8, "P8765432", Sexo.MALE, "Clun Forest", _fechas[16], 190, 59, true, LoadInjectionTo()));

InstanceRes(new Ovino(2.5, "Q2345678", Sexo.FEMALE, "Soay", _fechas[17], 175, 45, true, LoadInjectionTo()));

InstanceRes(new Ovino(3.2, "R9876543", Sexo.MALE, "Hebridean", _fechas[18], 235, 66, true, LoadInjectionTo()));

InstanceRes(new Ovino(1.8, "S1234567", Sexo.FEMALE, "Shetland", _fechas[19], 120, 34, true, LoadInjectionTo()));

InstanceRes(new Ovino(2.7, "T8765432", Sexo.MALE, "Arapawa", _fechas[20], 180, 56, true, LoadInjectionTo()));

InstanceRes(new Ovino(2.4, "U2345678", Sexo.FEMALE, "Columbia", _fechas[21], 165, 43, true, LoadInjectionTo()));


```

        InstanceRes(new Ovino(3.1, "V9876543", Sexo.MALE, "Rambouillet", _fechas[22], 220,
64, true, LoadInjectionTo()));

        InstanceRes(new Ovino(1.9, "W1234567", Sexo.FEMALE, "Merino Argentino",
_fechas[23], 130, 36, true, LoadInjectionTo()));

        InstanceRes(new Ovino(2.8, "X8765432", Sexo.MALE, "Corriedale Australiano",
_fechas[24], 195, 58, true, LoadInjectionTo()));

        InstanceRes(new Ovino(2.5, "Y2345678", Sexo.FEMALE, "Romney Marsh Neozelandés",
_fechas[25], 170, 45, true, LoadInjectionTo()));

        InstanceRes(new Ovino(3.2, "Z9876543", Sexo.MALE, "Tex", _fechas[26], 175, 40, true,
LoadInjectionTo()));

        InstanceRes(new Ovino(3.2, "Z9876544", Sexo.MALE, "Merilin", _fechas[27], 155, 40,
false, LoadInjectionTo()));

        InstanceRes(new Ovino(3.2, "Z9876545", Sexo.FEMALE, "Merino Australiano",
_fechas[28], 145, 40, true, LoadInjectionTo()));

        InstanceRes(new Ovino(3.2, "Z9876546", Sexo.MALE, "Romney Marsh", _fechas[29],
145, 40, false, LoadInjectionTo()));

    }

    public void InstanceRes(Res unRes)
    {
        //Controlo que no exista otro animal con misma caravana (id)
        if (!_reses.Contains(unRes))
        {
            ValidarRes(unRes);
            _reses.Add(unRes);
        }
        else
        {
            throw new Exception("La caravana ingresada ya existe en el sistema");
        }
    }

    private void ValidarRes(Res unRes)
    {

```

```

        unRes.Validar();
    }

    private void PreLoadBovino()
    {
        // Creo fechas randoms en una lista _fechas para utilizar en la precarga
        List<DateTime> _fechas = new List<DateTime>();

        DateTime fechaNacimiento;
        Random random = new Random();

        for (int i = 0; i < 30; i++)
        {
            fechaNacimiento = new DateTime(2024, random.Next(1, 12), random.Next(1, 28));
            if(fechaNacimiento.Month > DateTime.Now.Month)
            {
                i--;
            }
            else
            {
                _fechas.Add(fechaNacimiento);
            }
        }

        InstanceRes(new Bovino(Alimento.GRANO, "1234567A", Sexo.FEMALE, "Angus",
            _fechas[0], 1500, 150, true, LoadInjectionTo()));

        InstanceRes(new Bovino(Alimento.PASTURA, "7890123B", Sexo.FEMALE, "Hereford",
            _fechas[1], 1200, 180, true, LoadInjectionTo()));

        InstanceRes(new Bovino(Alimento.GRANO, "4567890C", Sexo.MALE, "Holstein",
            _fechas[2], 1800, 120, false, LoadInjectionTo()));

        InstanceRes(new Bovino(Alimento.PASTURA, "1234567D", Sexo.FEMALE, "Shorthorn",
            _fechas[3], 1000, 100, true, LoadInjectionTo()));

        InstanceRes(new Bovino(Alimento.PASTURA, "7890123A", Sexo.MALE, "Limousin",
            _fechas[4], 1600, 130, false, LoadInjectionTo()));

        InstanceRes(new Bovino(Alimento.GRANO, "7890123E", Sexo.MALE, "Limousin",
            _fechas[5], 1600, 130, false, LoadInjectionTo()));
    }

```

```
InstanceRes(new Bovino(Alimento.PASTURA, "456789oF", Sexo.FEMALE, "Charolais",
_fechas[6], 1300, 160, true, LoadInjectionTo()));

InstanceRes(new Bovino(Alimento.GRANO, "1234567G", Sexo.MALE, "Gelbvieh",
_fechas[7], 1700, 180, true, LoadInjectionTo()));

InstanceRes(new Bovino(Alimento.PASTURA, "789o123H", Sexo.FEMALE, "Maine",
_fechas[8], 1100, 120, true, LoadInjectionTo()));

InstanceRes(new Bovino(Alimento.PASTURA, "456789oI", Sexo.MALE, "South Devon",
_fechas[9], 1550, 120, false, LoadInjectionTo()));

InstanceRes(new Bovino(Alimento.GRANO, "1234567J", Sexo.FEMALE, "Red Angus",
_fechas[10], 1250, 150, true, LoadInjectionTo()));

InstanceRes(new Bovino(Alimento.PASTURA, "789o123K", Sexo.MALE, "Wagyu",
_fechas[11], 2000, 155, false, LoadInjectionTo()));

InstanceRes(new Bovino(Alimento.PASTURA, "456789oL", Sexo.FEMALE, "Belted
Galloway", _fechas[12], 1150, 135, true, LoadInjectionTo()));

InstanceRes(new Bovino(Alimento.GRANO, "1234567M", Sexo.MALE, "Texas
Longhorn", _fechas[13], 1650, 170, false, LoadInjectionTo()));

InstanceRes(new Bovino(Alimento.PASTURA, "789o123J", Sexo.FEMALE, "Brahman",
_fechas[14], 1350, 170, true, LoadInjectionTo()));

InstanceRes(new Bovino(Alimento.GRANO, "456789oO", Sexo.MALE, "Murray Grey",
_fechas[15], 1850, 130, false, LoadInjectionTo()));

InstanceRes(new Bovino(Alimento.PASTURA, "1234567P", Sexo.FEMALE, "Highland",
_fechas[16], 1200, 140, false, LoadInjectionTo()));

InstanceRes(new Bovino(Alimento.GRANO, "789o123Q", Sexo.MALE, "Chianina",
_fechas[17], 1700, 190, true, LoadInjectionTo()));

InstanceRes(new Bovino(Alimento.PASTURA, "456789oR", Sexo.FEMALE,
"Simmental", _fechas[18], 1400, 190, true, LoadInjectionTo()));

InstanceRes(new Bovino(Alimento.PASTURA, "1234567S", Sexo.FEMALE, "Sussex",
_fechas[19], 1900, 190, true, LoadInjectionTo()));

InstanceRes(new Bovino(Alimento.GRANO, "789o123T", Sexo.MALE, "Devon",
_fechas[20], 1050, 190, true, LoadInjectionTo()));

InstanceRes(new Bovino(Alimento.PASTURA, "456789oU", Sexo.FEMALE, "Galloway",
_fechas[21], 1500, 165, true, LoadInjectionTo()));

InstanceRes(new Bovino(Alimento.GRANO, "1234567V", Sexo.MALE, "Brangus",
_fechas[22], 1200, 195, true, LoadInjectionTo()));

InstanceRes(new Bovino(Alimento.GRANO, "789o123W", Sexo.FEMALE, "Beefmaster",
_fechas[23], 1300, 185, true, LoadInjectionTo()));

InstanceRes(new Bovino(Alimento.PASTURA, "789o123X", Sexo.MALE, "Angus",
_fechas[24], 1300, 140, true, LoadInjectionTo()));
```

```
InstanceRes(new Bovino(Alimento.PASTURA, "7890123Y", Sexo.FEMALE, "Aberdeen  
angus", _fechas[25], 1300, 190, true, LoadInjectionTo()));
```

```
InstanceRes(new Bovino(Alimento.GRANO, "7890123Z", Sexo.MALE, "Charolais",  
_fechas[26], 1300, 1550, true, LoadInjectionTo()));
```

```
InstanceRes(new Bovino(Alimento.PASTURA, "7890123M", Sexo.FEMALE,  
"Normando", _fechas[27], 1300, 166, true, LoadInjectionTo()));
```

```
InstanceRes(new Bovino(Alimento.GRANO, "7890133J", Sexo.MALE, "Holando",  
_fechas[28], 1300, 180, true, LoadInjectionTo()));
```

```
InstanceRes(new Bovino(Alimento.PASTURA, "7890123L", Sexo.FEMALE, "Jersey",  
_fechas[29], 1300, 185, true, LoadInjectionTo()));
```

```
}
```

```
private List<Peon> LoadPeonACargo()
```

```
{
```

```
List<Peon> _peonesToCapataz = new List<Peon>();
```

```
for (int i = 0; i < 10; i++)
```

```
if (_empleados[i].GetType() == typeof(Peon))
```

```
{
```

```
Peon unPeon = (Peon)_empleados[i];
```

```
if (!(unPeon.TieneCapataz) && _peonesToCapataz.Count < 5)
```

```
{
```

```
_peonesToCapataz.Add(unPeon);
```

```
unPeon.TieneCapataz = true;
```

```
}
```

```
}
```

```
return _peonesToCapataz;
```

```
}
```

```
private void PreLoadPeon()
```

```
{
```

```
InstanceEmpleados(new Peon(true, false, "john.smith@estancia.com", "password123",  
"John Smith", new DateTime(2020, 10, 15)));
```

```

        InstanceEmpleados(new Peon(true, false, "jane.doe@estancia.com", "password456",
"Jane Doe", new DateTime(2019, 12, 01)));

        InstanceEmpleados(new Peon(true, false, "peter.jones@estancia.com", "password789",
"Peter Jones", new DateTime(2024, 01, 04)));

        InstanceEmpleados(new Peon(true, false, "mary.williams@estancia.com",
"password012", "Mary Williams", new DateTime(2023, 03, 22)));

        InstanceEmpleados(new Peon(true, false, "david.miller@estancia.com",
"password345", "David Miller", new DateTime(2022, 05, 18)));

        InstanceEmpleados(new Peon(true, false, "sarah.taylor@estancia.com",
"password678", "Sarah Taylor", new DateTime(2024, 02, 11)));

        InstanceEmpleados(new Peon(true, false, "michael.brown@estancia.com",
"password901", "Michael Brown", new DateTime(2023, 04, 07)));

        InstanceEmpleados(new Peon(true, false, "laura.davis@estancia.com", "password234",
"Laura Davis", new DateTime(2022, 06, 24)));

        InstanceEmpleados(new Peon(true, false, "richard.thomas@estancia.com",
"password567", "Richard Thomas", new DateTime(2018, 03, 09)));

        InstanceEmpleados(new Peon(true, false, "elizabeth.anderson@estancia.com",
"password890", "Elizabeth Anderson", new DateTime(2021, 05, 13)));
    }

```

```

public void InstanceEmpleados(Empleado unEmpleado)
{
    if (_empleados.Contains(unEmpleado))
    {
        throw new Exception("El empleado ya se encuentra registrado en el sistema");
    }
    else
    {
        ValidarEmpleado(unEmpleado);
        _empleados.Add(unEmpleado);
    }
}

```

```

private void ValidarEmpleado(Empleado empleado)
{
    empleado.Validar();
}

private Tarea RandomTaskToPeon()
{
    Random random = new Random();
    Tarea tarea = _tareas[random.Next(0, 14)];
    return tarea;
}

private void PreLoadTask()
{
    InstanceTask(new Tarea("Inspeccionar el sistema de alarma contra incendios en el edificio principal."));
    InstanceTask(new Tarea("Verificar el funcionamiento de las luces de emergencia en todas las salidas."));
    InstanceTask(new Tarea("Realizar pruebas de fugas de gas en todas las áreas de almacenamiento. "));
    InstanceTask(new Tarea("Calibrar los manómetros de presión en las calderas"));
    InstanceTask(new Tarea("Inspeccionar el estado de las barandas y pasamanos en las escaleras"));
    InstanceTask(new Tarea("Revisar los extintores de incendios para garantizar su funcionamiento adecuado."));
    InstanceTask(new Tarea("Verificar que las señales de seguridad y emergencia estén visibles y en buen estado."));
    InstanceTask(new Tarea("Realizar una inspección general de las instalaciones en busca de posibles peligros."));
    InstanceTask(new Tarea("Documentar los hallazgos de las inspecciones y tomar las medidas correctivas necesarias."));
    InstanceTask(new Tarea("Capacitar al personal sobre los procedimientos de seguridad en caso de emergencias."));
    InstanceTask(new Tarea("Realizar simulacros de incendio y evacuación para evaluar la preparación del personal."));
}

```

```
InstanceTask(new Tarea("Mantener registros actualizados de las tareas de mantenimiento y seguridad realizadas."));
```

```
InstanceTask(new Tarea("Comunicar los riesgos de seguridad y los procedimientos de emergencia a todo el personal."));
```

```
InstanceTask(new Tarea("Investigar los incidentes de seguridad y tomar medidas para prevenir su recurrencia."));
```

```
InstanceTask(new Tarea("Implementar programas de mejora continua para garantizar la seguridad en el lugar de trabajo."));
```

```
}
```

```
public void InstanceTask(Tarea unaTarea)
```

```
{
```

```
    ValidarTarea(unaTarea);
```

```
    _tareas.Add(unaTarea);
```

```
}
```

```
private void ValidarTarea(Tarea tarea)
```

```
{
```

```
    tarea.Validar();
```

```
}
```

```
private void PreLoadPotrero()
```

```
{
```

```
    Random random = new Random();
```

```
    InstancePotrero(new Potrero("Potrero 1", random.Next(1, 10)));
```

```
    InstancePotrero(new Potrero("Potrero 2", random.Next(1, 10)));
```

```
    InstancePotrero(new Potrero("Potrero 3", random.Next(1, 10)));
```

```
    InstancePotrero(new Potrero("Potrero 4", random.Next(1, 10)));
```

```
    InstancePotrero(new Potrero("Potrero 5", random.Next(1, 10)));
```

```
    InstancePotrero(new Potrero("Potrero 6", random.Next(1, 10)));
```

```
    InstancePotrero(new Potrero("Potrero 7", random.Next(1, 10)));
```

```
    InstancePotrero(new Potrero("Potrero 8", random.Next(1, 10)));
```

```

InstancePotrero(new Potrero("Potrero 9", random.Next(1, 10)));
InstancePotrero(new Potrero("Potrero 10", random.Next(1, 10)));

AsignarAnimalesEnPoterros();
}

private void AsignarAnimalesEnPoterros()
{
    // Armo lista de animales libres
    List<Res> freeAnimals = new List<Res>();
    foreach (Res unaRes in _reses)
    {
        if (unaRes.Libres)
            freeAnimals.Add(unaRes);
    }

    // Asigno animales libres a cada potrero
    foreach (Potrero unPotrero in _potreros)
    {
        unPotrero.AssignAnimals(freeAnimals);
    }
}

private void InstancePotrero(Potrero potrero)
{
    ValidarPotrero(potrero);
    potrero.AssignAnimalCapacity();
    _potreros.Add(potrero);
}

private void ValidarPotrero(Potrero potrero)
{
    potrero.Validar();
}

```



```
}
```

```
private void PreLoadVaccine()
```

```
{
```

```
    InstanceVaccine(new Vacuna("Fiebre Aftosa", "Enfermedad viral altamente contagiosa que afecta a bovinos, ovinos y porcinos. Se caracteriza por fiebre, vesículas en la boca y patas, y dificultad para caminar.", "Virus de la Fiebre Aftosa"));
```

```
    InstanceVaccine(new Vacuna("Brucelosis", "Enfermedad bacteriana que afecta el sistema reproductivo de bovinos y ovinos. Provoca abortos, infertilidad y nacimiento de terneros débiles.", "Bacteria Brucella abortus"));
```

```
    InstanceVaccine(new Vacuna("Carbón", "Enfermedad bacteriana grave que afecta a bovinos, ovinos y otros animales. Se caracteriza por muerte súbita, hinchazón y sangre en la nariz y boca.", "Bacteria Clostridium chauvoei"));
```

```
    InstanceVaccine(new Vacuna("Manquera", "Enfermedad viral que afecta las patas de bovinos y ovinos. Provoca cojera, úlceras en las patas y disminución de la producción de leche.", "Virus de la Manquera"));
```

```
    InstanceVaccine(new Vacuna("Leptospirosis", "Enfermedad bacteriana que afecta a bovinos y ovinos. Provoca abortos, infertilidad, problemas renales y hepáticos.", "Bacteria Leptospira spp"));
```

```
    InstanceVaccine(new Vacuna("Rabia", "Enfermedad viral mortal que afecta a todos los mamíferos, incluyendo bovinos y ovinos. Se caracteriza por síntomas neurológicos como agresividad, parálisis y muerte.", "Virus de la Rabia"));
```

```
    InstanceVaccine(new Vacuna("Diarrea Viral Bovina", "Enfermedad viral que afecta a bovinos de todas las edades. Provoca diarrea, neumonía, y disminución de la producción de leche.", "Virus de la Diarrea Viral Bovina"));
```

```
    InstanceVaccine(new Vacuna("Parainfluenza Bovina", "Enfermedad viral que afecta a bovinos de todas las edades. Provoca problemas respiratorios, especialmente en terneros", "Virus Parainfluenza Bovino tipo 3"));
```

```
    InstanceVaccine(new Vacuna("Rotavirus", "Enfermedad viral que afecta a terneros jóvenes. Provoca diarrea severa, deshidratación y muerte en algunos casos.", "Rotavirus bovino"));
```

```
    InstanceVaccine(new Vacuna("Rinotraqueítis Infecciosa Bovina", "Enfermedad viral que afecta a bovinos de todas las edades. Provoca problemas respiratorios, abortos y disminución de la producción de leche.", "Virus de la Rinotraqueítis Infecciosa Bovina"));
```

```
}
```

```
private void InstanceVaccine(Vacuna vaccine)
```

```
{
```

```
ValidateVaccine(vaccine);  
_vacunas.Add(vaccine);  
}
```

```
private void ValidateVaccine(Vacuna oneVaccine)  
{  
    oneVaccine.Validar();  
}
```

```
public List<Res> GetAnimales()  
{  
    List<Res> copiaAnimales = new List<Res>();  
    foreach (Res unaRes in _reses)  
        copiaAnimales.Add(unaRes);  
    return copiaAnimales;  
}
```

```
public List<Potrero> GetPotreros()  
{  
    List<Potrero> copiaPotreros = new List<Potrero>();  
    foreach (Potrero unP in _potreros)  
        copiaPotreros.Add(unP);  
    return copiaPotreros;  
}
```

```
public Potrero BuscarPotrero(int id)  
{  
    Potrero potrero = null;  
    int i = 0;  
    while (potrero == null || i < _potreros.Count)  
    {
```

```

        if (_potreros[i].Id == id)
        {
            potrero = _potreros[i];
        }
        i++;
    }
    if (potrero == null)
        throw new Exception("Potrero no encontrado");
    return potrero;
}

public List<Vacuna> GetVacunas()
{
    List<Vacuna> copiaVacunas = new List<Vacuna>();
    foreach (Vacuna unaV in _vacunas)
    {
        copiaVacunas.Add(unaV);
    }
    return copiaVacunas;
}

public void PotrerosLibres(int hectareas, int valor)
{
    int cont = 0;
    foreach (Potrero unPotrero in _potreros)
    {
        if
(unPotrero.AreaMayorACantHectareasYCapacidadMaxSuperiorNroDado(hectareas, valor))
        {
            Console.WriteLine(unPotrero);
            cont++;
        }
    }
}

```

```
}  
if (cont == 0)  
    Console.WriteLine("No existen potreros disponibles para los valores ingresados.");  
}
```

```
public void SetPrecioLana(decimal precioLana)  
{  
    Ovino.SetNewPrecioKgLana(precioLana);  
}
```

```
public void AltaBovino(Bovino nuevoBovino)  
{  
    ValidarRes(nuevoBovino);  
  
    // Contain para ver si ya no existe  
    if (_reses.Contains(nuevoBovino))  
        throw new Exception("El bovino ya existe en el sistema.");  
    _reses.Add(nuevoBovino);  
}
```

```
public decimal GetPrecioKgLana()  
{  
    decimal precioKgLana = 0;  
    precioKgLana = Ovino.GetPrecioActualKgLana();  
    return precioKgLana;  
}
```

```
public Empleado AutenticarUsuario(string email, string password)  
{  
    Empleado usuario = null;  
    int i = 0;
```

```

while (usuario == null && i < _empleados.Count)
{
    if (_empleados[i].Email == email)
    {
        usuario = _empleados[i];
    }
    i++;
}

if (usuario == null || usuario.Contrasenia != password)
    throw new Exception("Usuario o Contraseña incorrectas");

return usuario;
}

```

```

public Peon BuscarPeon(string email)
{
    Peon usuario = null;
    int i = 0;
    while (usuario == null && i < _empleados.Count)
    {
        if (_empleados[i].Email == email && _empleados[i] is Peon)
        {
            usuario = (Peon)_empleados[i];
        }
        i++;
    }
}

```

```

if (usuario == null)
    throw new Exception("Peón no encontrado");

```

```

return usuario;

```

```
}
```

```
public Tarea BuscarTarea(int id)
```

```
{
```

```
    Tarea tarea = null;
```

```
    int i = 0;
```

```
    while (tarea == null && i < _tareas.Count)
```

```
    {
```

```
        if (_tareas[i].Id == id)
```

```
        {
```

```
            tarea = _tareas[i];
```

```
        }
```

```
        i++;
```

```
    }
```

```
    if (tarea == null)
```

```
        throw new Exception("Tarea no encontrada");
```

```
    return tarea;
```

```
}
```

```
public Res GetUnAnimal(string id)
```

```
{
```

```
    Res buscado = null;
```

```
    int i = 0;
```

```
    while (buscado == null && i < _reses.Count)
```

```
    {
```

```
        if (_reses[i].Id == id)
```

```
        {
```

```
            buscado = _reses[i];
```

```
        }
```

```
        i++;  
    }  
    if (buscado == null)  
        throw new Exception("Animal no encontrado");  
    return buscado;  
}  
  
public List<Peon> GetPeones()  
{  
    List<Peon> copiaPeones = new List<Peon>();  
    foreach (Empleado unE in _empleados)  
    {  
        if (unE is Peon)  
            copiaPeones.Add((Peon)unE);  
    }  
    if (copiaPeones.Count == 0)  
        throw new Exception("No hay peones disponibles");  
    return copiaPeones;  
}
```

```
public List<Tarea> GetTareas()  
{  
    List<Tarea> copiaTareas = new List<Tarea>();  
    foreach (Tarea unE in _tareas)  
    {  
        copiaTareas.Add(unE);  
    }  
    if (copiaTareas.Count == 0)  
        throw new Exception("No hay tareas creadas");  
    return copiaTareas;  
}
```

```

public List<Res> GetResesPorTipo(string tipo)
{
    List<Res> listado = new List<Res>();

    if (tipo == "Ovino")
    {
        foreach (Res unaRes in _reses)
        {
            if (unaRes is Ovino)
            {
                listado.Add(unaRes);
            }
        }
    }
    else if (tipo == "Bovino")
    {
        foreach (Res unaRes in _reses)
        {
            if (unaRes is Bovino)
            {
                listado.Add(unaRes);
            }
        }
    }

    return listado;
}

public Vacuna GetUnaVacuna(string nombreVacuna)
{
    Vacuna vacuna = null;
    int i = 0;
    while (vacuna == null && i < _vacunas.Count)
    {
        if (_vacunas[i].Nombre == nombreVacuna)
        {
            vacuna = _vacunas[i];
        }
    }
}

```



```

        i++;
    }

    if (vacuna == null)
        throw new Exception("No se encontró la vacuna");
    return vacuna;
}
}
}

```

3.1.14 TAREA.CS

namespace Estancia

```

{
    public class Tarea : IValidable
    {
        private int _id;
        private static int s_ultimold = 1;
        private string _descripcion;

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

        public int Id
        {
            get { return _id; }
            set { _id = value; }
        }

        public Tarea(string descripcion)

```

```

{
    _id = s_ultimold++;
    _descripcion = descripcion;
}

public Tarea() {}

public void Validar()
{
    if(string.IsNullOrEmpty(_descripcion))
    {
        throw new Exception("La tarea debe contener una descripción");
    }
}
}
}

```

3.1.15 TAREAASIGNADA

namespace Estancia

```

{
    public class TareaAsignada : IValidable, IComparable<TareaAsignada>
    {
        private string _comentario;
        private DateTime _fechaPactada;
        private DateTime _fechaCierre;
        private Tarea _tarea;

        public string Comentario
        {
            get { return _comentario; }
            set { _comentario = value; }
        }
    }
}

```

```
public DateTime FechaCierre
{
    get { return _fechaCierre; }
    set { _fechaCierre = value; }
}
```

```
public DateTime FechaPactada
{
    get { return _fechaPactada; }
    set { _fechaPactada = value; }
}
```

```
public Tarea Tarea
{
    get { return _tarea; }
    set { _tarea = value; }
}
```

```
public TareaAsignada(string comentario, DateTime fechaPactada, DateTime fechaCierre,
Tarea tarea)
```

```
{
    _comentario = comentario;
    _fechaPactada = fechaPactada;
    _fechaCierre = fechaCierre;
    _tarea = tarea;
}
```

```
public TareaAsignada(DateTime fechaPactada, Tarea tarea)
```

```
{
    _fechaPactada = fechaPactada;
```

```

        _tarea = tarea;
    }

    public TareaAsignada() {}

    public void Validar()
    {
        //Valida casos INCORRECTOS a la hora de asignar una tarea. (No puede existir
        comentario sin fecha de cierre y viceversa.)
        if (_fechaCierre == DateTime.MinValue && _comentario != null)
        {
            throw new Exception("No puede existir una asignación de tarea con un comentario y
            sin fecha de cierre.");
        }
        else if (_fechaCierre != DateTime.MinValue && _comentario == null)
        {
            throw new Exception("La tarea finalizada debe tener un comentario.");
        }
    }

    public int CompareTo(TareaAsignada otraTa)
    {
        return _fechaPactada.CompareTo(otraTa._fechaPactada);
    }
}

```

3.1.16 VACUNA.CS

```
namespace Estancia
{
    public class Vacuna : IValidable
    {
        private string _nombre;
        private string _descripcion;
        private string _patogeno;

        public string Nombre
        {
            get { return _nombre; }
        }

        public Vacuna(string nombre, string descripcion, string patogeno)
        {
            _nombre = nombre;
            _descripcion = descripcion;
            _patogeno = patogeno;
        }

        public Vacuna()
        {
            _nombre = "SN";
            _descripcion = "SD";
            _patogeno = "SP";
        }
    }
}
```

```

public void Validar()
{
    if(string.IsNullOrEmpty(_nombre))
    {
        throw new Exception("El nombre no puede ser vacio");
    }
}
}
}

```

3.2 OBLIGATORIO.CS

3.2.1 PROGRAM.CS

```
using Estancia;
```

```
using System;
```

```
namespace Obligatorio
```

```
{
```

```
    internal class Program
```

```
{
```

```
    static void Main(string[] args)
```

```
{
```

```
        Sistema miSistema = Sistema.Instance;
```

```
        bool sigue = true;
```

```
        Utilidades.MostrarTitulo("OBLIGATORIO 1");
```

```
        Console.WriteLine("Participantes:");
```

```
        Console.ForegroundColor = ConsoleColor.DarkYellow;
```

```
        Console.WriteLine("___ Joaquin Caviglia (328187) : Emmanuel Duffaut (177427)");
```

```
        Console.ResetColor();
```

```
        Console.WriteLine("Materia:");
```

```
        Console.ForegroundColor = ConsoleColor.DarkYellow;
```

```

Console.WriteLine("___ Programación 2");
Console.ResetColor();
Console.WriteLine("Profesor: ");
Console.ForegroundColor = ConsoleColor.DarkYellow;
Console.WriteLine("___ Mauricio Quintero\n");
Utilidades.PresionarCualquierTeclaParaBorrar();
do
{

    bool exito;
    int opcionElegida;
    do
    {
        Utilidades.MostrarMenu();
        Console.ForegroundColor = ConsoleColor.Blue;
        Console.Write("\nElige una opción: ");
        Console.ResetColor();
        exito = int.TryParse(Console.ReadLine(), out opcionElegida);
        if (!exitito || opcionElegida < 0 || opcionElegida > 4)
        {
            Utilidades.MostrarError("\nDebes ingresar una opcion valida");
        }
    } while (!exitito);

    switch( opcionElegida)
    {
        case 0:
            sigue = false;
            break;
        case 1:

```

```

        ListReses();

        break;

    case 2:

        ListFreePotreros();

        break;

    case 3:

        DefinirPrecioLana();

        break;

    case 4:

        AltaBovino();

        break;

    }
} while (sigue);
}

```

```

private static void DefinirPrecioLana()
{
    Console.ForegroundColor = ConsoleColor.Blue;
    Utilidades.MostrarTitulo("Definir precio por kg de Lana");
    Console.ResetColor();

    // Muestro precio actual
    decimal precioKgLanaActual = Sistema.Instance.GetPrecioKgLana();
    Utilidades.MostrarExito($"El precio actual por kg de lana es {precioKgLanaActual}\n");

    // Pido nuevo precio

    bool exito;

    do

    {

        Console.WriteLine("Ingrese precio por kilo de lana:");

        exito = decimal.TryParse(Console.ReadLine(), out precioKgLanaActual);

    } while (!exito);
}

```



```
// Seteo nuevo precio
Sistema.Instance.SetPrecioLana(precioKgLanaActual);
Console.Clear();
Utilidades.MostrarTitulo("Definir precio por kg de Lana");
Utilidades.MostrarExito($"El precio de lana se a cambiado correctamente, Valor:
{precioKgLanaActual}\n");
```

```
Utilidades.PresionarCualquierTeclaParaBorrar();
}
```

```
private static void ListFreePotreros()
{
    Console.Clear();
    // Pido al usuario que ingrese cant de hectareas y un valor
    Utilidades.MostrarTitulo("Potreros disponibles");
    int hectareas = Utilidades.AskInformation("Ingrese cantidad de hectáreas");
    int valor = Utilidades.AskInformation("Ingrese un valor");
    Console.Clear();
    Utilidades.MostrarTitulo("Potreros disponibles");

    // Busco potreros disponibles.
    Sistema.Instance.PotrerosLibres(hectareas, valor);

    Utilidades.PresionarCualquierTeclaParaBorrar();
}
```

```
private static void ListReses()
{
    Utilidades.MostrarTitulo("Reses");
```

```

// Me traigo una copia de todas las reses
List<Res> listaReses = Sistema.Instance.GetAnimales();
listaReses.Sort();

if(listaReses == null || listaReses.Count == 0)
{
    Utilidades.MostrarError("No existen animales ingresados en el sistema");
}
else
{
    foreach (Res _unaRes in listaReses)
    {
        Console.WriteLine(_unaRes.ToString());
    }
}
Utilidades.PresionarCualquierTeclaParaBorrar();
}

private static void AltaBovino()
{
    Utilidades.MostrarTitulo("Alta Bovino");
    ConsoleColor userInput = ConsoleColor.Cyan;

    string raza;
    do
    {
        Console.WriteLine("Ingrese la Raza del bovino: ");
        Console.ForegroundColor = userInput;
        raza = Console.ReadLine();
        Console.ResetColor();
    }
}

```

```
} while (raza == null);
```

```
string idCaravana;
```

```
do
```

```
{
```

```
    Console.WriteLine("Ingrese número de caravana(8 caracteres): ");
```

```
    Console.ForegroundColor = userInput;
```

```
    idCaravana = Console.ReadLine();
```

```
    Console.ResetColor();
```

```
} while (idCaravana == null);
```

```
int peso = Utilidades.AskInformation("Ingrese el peso actual: ");
```

```
int dia;
```

```
int mes;
```

```
int anio;
```

```
//tomar fecha
```

```
Console.WriteLine("Ingrese Fecha de Nacimiento del Bovino: ");
```

```
do
```

```
{
```

```
    dia = Utilidades.AskInformation("Ingrese el dia");
```

```
} while (dia<=0 || dia > 31);
```

```
do
```

```
{
```

```
    mes = Utilidades.AskInformation("Ingrese el mes");
```

```
}while(mes<=0 || mes >= 13);
```

```
do
```

```
{
```

```

        anio = Utilidades.AskInformation("Ingrese el año");
    } while (anio > DateTime.Now.Year);

    DateTime fechaNacimiento = new DateTime(anio, mes, dia);

    int numSexo = Utilidades.AskInformationTwoValues("Que sexo es?(1-Macho 2-
Hembra): ");

    int numAlimento = Utilidades.AskInformationTwoValues("Cual es su alimentación? (1-
Grano 2-Pastura) : ");

    int costoAdquisicion = Utilidades.AskInformation("Cual es el costo de adquisición?");

    int hibrido = Utilidades.AskInformationTwoValues("Es hibrido? (1-SI 2-NO) : ");

    bool esHibrido = hibrido == 1 ? true : false;

    List<Inyeccion> inyecciones = new List<Inyeccion>();

    Bovino nuevoBovino = new Bovino((Alimento)numAlimento, idCaravana,
(Sexo)numSexo, raza, fechaNacimiento, costoAdquisicion, peso, esHibrido, inyecciones);

    try
    {
        Sistema.Instance.AltaBovino(nuevoBovino);

        Console.Clear();

        Utilidades.MostrarExito($"El bovino se ingresó correctamente\n");

        Console.WriteLine(nuevoBovino.ToString());
    }

    catch (Exception ex)
    {
        Utilidades.MostrarError(ex.Message);
    }

    Utilidades.PresionarCualquierTeclaParaBorrar();
}
}

```

```
}
```

3.2.2 UTILIDADES.CS

```
public class Utilidades
```

```
{
```

```
    private static ConsoleColor s_userInput = ConsoleColor.Cyan;
```

```
    public static void MostrarError(string error)
```

```
    {
```

```
        Console.ForegroundColor = ConsoleColor.Red;
```

```
        Console.WriteLine(error);
```

```
        Console.ResetColor();
```

```
    }
```

```
    public static void MostrarExito(string exito)
```

```
    {
```

```
        Console.ForegroundColor = ConsoleColor.Green;
```

```
        Console.WriteLine(exito);
```

```
        Console.ResetColor();
```

```
    }
```

```
    public static void MostrarMenu()
```

```
    {
```

```
        // Console.Clear();
```

```
        Console.WriteLine("*** OBLIGATORIO 1 ***\n");
```

```
        // Opciones del menú
```

```
        string[] opciones = { "Salir", "Listar animales", "Listar potreros disponibles", "Precio de Lana", "Alta de Bovino" };
```

```
        // Muestro opciones del menú
```

```
        Console.WriteLine("***** Menú *****");
```

```
        for (int i = 0; i < opciones.Length; i++)
```

```
        {
```

```
            Console.WriteLine($"{i}- {opciones[i]}");
```

```
    }  
}
```

```
internal static int AskInformation(string message)
```

```
{
```

```
    bool exito;
```

```
    int hectareas;
```

```
    do
```

```
    {
```

```
        Console.WriteLine(message + ":");
```

```
        Console.ForegroundColor = s_userInput;
```

```
        exito = int.TryParse(Console.ReadLine(), out hectareas);
```

```
        Console.ResetColor();
```

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

```
    return hectareas;
```

```
}
```

```
internal static int AskInformationTwoValues(string message)
```

```
{
```

```
    bool exito;
```

```
    int value;
```

```
    do
```

```
    {
```

```
        Console.WriteLine(message);
```

```
        exito = int.TryParse(Console.ReadLine(), out value);
```

```
    } while (!exito || (value!=1 && value!=2));
```

```
    return value;
```

```
}
```

```
public static void MostrarTitulo(string title)
```

```
{
```

```

    Console.Clear();

    Console.ForegroundColor = ConsoleColor.Yellow;

    Console.WriteLine($"*** {title} ***\n");

    Console.ResetColor();

    //Console.WriteLine("Ingrese o para salir\n");
}

public static void PresionarCualquierTeclaParaBorrar()
{
    Console.WriteLine("Presione cualquier tecla para continuar");

    Console.ReadKey();

    Console.Clear();
}
}

```

4 INTERFAZ DE USUARIO (MVC)

4.1 EMPLEADOS

4.1.1 LISTARPARAASIGNAR.CSHTML

@*

For more information on enabling MVC for empty projects, visit
<https://go.microsoft.com/fwlink/?LinkID=397860>

*@

@{

 ViewData["Title"] = "Tareas";

}

@using Estancia

@model List<Tarea>

<h1>Asignar Tareas</h1>

Nueva Tarea

```

@if (ViewBag.Exito != null)
{
    <p class="text-success">@ViewBag.Exito</p>
}

```

```

<table class="table table-hover table-striped align-items-center table-bordered">
    <thead class="text-center">
        <th>ID</th>
        <th>Descripción</th>
        <th>Seleccionar Peon</th>
    </thead>
    <tbody>
        @foreach(Tarea unaT in Model)
        {
            <tr class="text-center">
                <td>@unaT.Id</td>
                <td>@unaT.Descripcion</td>
                <td><a href="/Empleados/Asignar?id=@unaT.Id&email=@ViewBag.Email" class="btn btn-primary">Seleccionar</a></td>
            </tr>
        }

    </tbody>
</table>

```

4.1.2 PERFILPEON.CSHTML

```
@*
```

For more information on enabling MVC for empty projects, visit
<https://go.microsoft.com/fwlink/?LinkID=397860>

```
*@
```

```
@{
```

```
}
```

```
@using Estancia
```



```
@model Peon
```

```
@if (Model != null)
```

```
{
```

```
<table class="table table-hover table-striped align-items-center table-bordered">
```

```
<thead class="text-center">
```

```
<th>Nombre</th>
```

```
<th>Email</th>
```

```
<th>Fecha de Ingreso</th>
```

```
<th>Residente</th>
```

```
</thead>
```

```
<tbody>
```

```
<tr class="text-center">
```

```
<td>@Model.Nombre</td>
```

```
<td>@Model.Email</td>
```

```
<td>@Model.FechaIngreso.ToShortDateString()</td>
```

```
<td>@(Model.EsResidente ? "Si" : "No")</td>
```

```
</tr>
```

```
</tbody>
```

```
</table>
```

```
}
```

```
@if (ViewBag.Error != null)
```

```
{
```

```
<p class="text-danger">@ViewBag.Error</p>
```

```
}
```

4.1.3 REGISTRARPEON.CSHTML

@*

For more information on enabling MVC for empty projects, visit
<https://go.microsoft.com/fwlink/?LinkID=397860>

*@

@{

}

@using Estancia;

<h1>Registrarse</h1>

<form method="post">

<label for="email">Email:</label>

<input type="email" id="email" name="Email" placeholder="Email" class="form-control"/>

<label for="contraseña">Contraseña:</label>

<input type="password" id="contraseña" name="Contrasenía" placeholder="Contraseña"
class="form-control" />

<label for="nombre">Nombre:</label>

<input type="text" id="nombre" name="Nombre" placeholder="Nombre" class="form-
control" />

<label>Es residente?</label>

<select name="EsResidente" class="form-select">

<option value="@true"> Si</option>

<option value="@false"> No</option>

</select>

<input type="submit" value="Registrar" class="btn btn-primary mt-3" />

</form>

@if(ViewBag.Error != null)

```
{  
    <p>@ViewBag.Error</p>  
}
```

4.1.4 SELECCIONARPEON.CSHTML

```
@*  
    For more information on enabling MVC for empty projects, visit  
    https://go.microsoft.com/fwlink/?LinkID=397860
```

```
*@
```

```
@{  
    ViewData["Title"] = "Peones";  
}
```

```
@using Estancia
```

```
@model List<Peon>
```

```
<h1>Listado de Peones</h1>
```

```
@if (ViewBag.Error != null)
```

```
{  
    <p class="text-danger">@ViewBag.Error</p>  
}
```

```
@if (ViewBag.Exito != null)
```

```
{  
    <p class="text-success">@ViewBag.Exito</p>  
}
```

```
@if (Model != null)
```

```
{  
    <table class="table table-hover table-striped align-items-center table-bordered">  
        <thead class="text-center">  
            <th>Nombre</th>  
            <th>Residente</th>  
            <th>Fecha de Ingreso</th>  
            <th>Tareas</th>  
        </thead>  
        <tbody>  
            @foreach (var peon in Model)  
            {  
                <tr>  
                    <td>@peon.Nombre</td>  
                    <td>@peon.Residente</td>  
                    <td>@peon.FechaIngreso</td>  
                    <td>@peon.Tareas</td>  
                </tr>  
            }  
        </tbody>  
    </table>  
}
```

```

</thead>

<tbody>
    @foreach (Peon unP in Model)
    {
        <tr class="text-center">
            <td>@unP.Nombre</td>
            <td>@(unP.EsResidente?"SI":"NO")</td>
            <td>@unP.FechaIngreso.ToShortDateString()</td>
            <td><a href="/Empleados/ListarParaAsignar?email=@unP.Email" class="btn btn-
primary">Asignar</a></td>
        </tr>
    }

</tbody>
</table>
}

```

4.1.5 TAREASNoCOMPLETADAS.CSHTML

@*

For more information on enabling MVC for empty projects, visit
<https://go.microsoft.com/fwlink/?LinkID=397860>

*@

@{

}

@{

ViewData["Title"] = "Tareas no completadas";

}

@using Estancia

@model List<TareaAsignada>

<h1>Tareas no completadas</h1>

```

@if (Model != null)
{
    @if (Model.Count > 0)
    {
        <table class="table table-hover table-striped align-items-center table-bordered">
            <thead class="text-center">
                <th>Descripción</th>
                <th>Fecha Pactada</th>
                <th>Comentario</th>
                <th>Modificar</th>
            </thead>
            <tbody>
                @foreach (TareaAsignada unaTa in Model)
                {
                    <tr class="text-center">
                        <td>@unaTa.Tarea.Descripcion</td>
                        <td>@unaTa.FechaPactada.ToShortDateString()</td>
                        <td>@unaTa.Comentario</td>
                        <td><a class="btn btn-primary"
href="/Tareas/Editar/@unaTa.Tarea.Id">Editar</a></td>
                    </tr>
                }

            </tbody>
        </table>
    }
    else
    {
        <p class="text-info">No tiene tareas pendientes.</p>
    }
}

```

```
}
```

```
@if(ViewBag.Mensaje != null)
```

```
{
```

```
<p class="text-success">@ViewBag.Mensaje</p>
```

```
}
```

```
@if (ViewBag.Error != null)
```

```
{
```

```
<p class="text-danger">@ViewBag.Error</p>
```

```
}
```

4.1.6 VERPEONES.CSHTML

```
@*
```

For more information on enabling MVC for empty projects, visit
<https://go.microsoft.com/fwlink/?LinkID=397860>

```
*@
```

```
@{
```

```
}
```

```
@{
```

```
    ViewData["Title"] = "Peones";
```

```
}
```

```
@using Estancia
```

```
@model List<Peon>
```

```
<h1>Listado de Peones</h1>
```

```
@if (Model != null)
```

```
{
```

```
<table class="table table-hover table-striped align-items-center table-bordered">
```

```
<thead class="text-center">
```

```
<th>Nombre</th>
```

```

<th>Residente</th>
<th>Fecha de Ingreso</th>
<th>Tareas</th>
</thead>
<tbody>
    @foreach (Peon unP in Model)
    {
        <tr class="text-center">
            <td>@unP.Nombre</td>
            <td>@unP.EsResidente</td>
            <td>@unP.FechaIngreso.ToShortDateString()</td>
            <td><a href="/Empleados/VerTareas?email=@unP.Email" class="btn btn-
primary">Ver</a></td>
        </tr>
    }

</tbody>
</table>
}

```

4.1.7 VERTAREAS.CSHTML

@*

For more information on enabling MVC for empty projects, visit
<https://go.microsoft.com/fwlink/?LinkID=397860>

*@

@{

 ViewData["Title"] = "Tareas";

}

@using Estancia

@model List<TareaAsignada>

<h1>Tareas</h1>

```

@if (Model.Count > 0)
{
<table class="table table-hover table-striped align-items-center table-bordered">
  <thead class="text-center">
    <th>Fecha Pactada</th>
    <th>Fecha Cierre</th>
    <th>Estado</th>
    <th>Descripción</th>
  </thead>
  <tbody>
    @foreach (TareaAsignada unaTa in Model)
    {
      <tr class="text-center">
        <td class="text-center">@unaTa.FechaPactada.ToShortDateString()</td>
        <td>@unaTa.FechaCierre.ToShortDateString()</td>
        <td>@(unaTa.Comentario == null ? "NO CERRADA" : "CERRADA")</td>
        <td>@unaTa.Tarea.Descripcion</td>
      </tr>
    }

  </tbody>

</table>
}

```

4.2 GANADO

4.2.1 ALTABOVINO.CSHTML

@*

For more information on enabling MVC for empty projects, visit
<https://go.microsoft.com/fwlink/?LinkID=397860>

*@

@{

}

@using Estancia

@model Bovino

<h1>Alta Bovino</h1>

<form method="post">

@* ASK: Mostramos el enum tipo lista ? *@

<label for="Alimento">Tipo de Alimento:</label>

<input type="text" id="Alimento" name="Alimento" value="@Model?.Alimento"
placeholder="GRANO o PASTURA" class="form-control" />

<label for="caravana">Caravana(*):</label>

<input type="text" id="caravana" name="Id" value="@Model?.Id" placeholder="Caravana"
class="form-control" Required />

@* ASK: Mostramos el enum tipo lista ? *@

<label for="sexo">Sexo:</label>

<input type="text" id="sexo" name="Sexo" value="@Model?.Sexo" placeholder="Sexo"
class="form-control" />

<label for="raza">Raza:</label>

<input type="text" id="raza" name="Raza" value="@Model?.Raza" placeholder="Raza"
class="form-control" />

<label for="fnac">Fecha de Nacimiento:</label>

<input type="date" id="fnac" name="FechaNacimiento"
value="@Model?.FechaNacimiento" class="form-control" />

<label for="costoAdq">Costo de Adquisición:</label>

<input type="number" id="costoAdq" name="CostoAdquisicion"
value="@Model?.CostoAdquisicion" placeholder="Costo adquisición" class="form-control" />

```

<label for="pesoActual">Peso:</label>

<input type="number" id="pesoActual" name="PesoActual" value="@Model?.PesoActual"
placeholder="Peso" class="form-control" />

```

```

<label for="esHibrido">Hibrido?</label>

<select name="EsHibrido" class="form-select form-control">

    <option>Elegir</option>

    <!option value="true">Si</!option>

    <!option value="false">No</!option>

</select>

<input type="text" name="Libre" value="true" class="form-control" hidden />

```

```

<div class="form-group">

    <input type="submit" name="Agregar" value="Agregar" class="btn btn-primary mt-3" />

</div>

```

```

</form>

@if (ViewBag.Error != null)
{
    <p class="text-danger">@ViewBag.Error</p>
}

```

4.2.2 ANIMALES POR PESO Y TIPO.CSHTML

```

@*

For more information on enabling MVC for empty projects, visit
https://go.microsoft.com/fwlink/?LinkID=397860

```

```

*@

```

```

@{

}

```

```

@{

    ViewData["Title"] = "Animales por peso y tipo";

}

```

```
@using Estancia
```

```
@model List<Res>
```

```
<h1>Animales por Peso y Tipo</h1>
```

```
<form method="post" action="/Ganado/AnimalesPorPesoYTipo">
```

```
  <label for="peso">Peso:</label>
```

```
  <input type="number" id="peso" name="Peso" value="peso" class="form-control" />
```

```
  <select name="tipo" class="form-select">
```

```
    <option>Elegir tipo:</option>
```

```
    <!--option value="Bovino">Bovino</option-->
```

```
    <!--option value="Ovino">Ovino</option-->
```

```
  </select>
```

```
  <div class="form-group">
```

```
    <input type="submit" name="Filtrar" value="Filtrar" class="btn btn-primary mt-3" />
```

```
  </div>
```

```
</form>
```

```
@if (Model != null)
```

```
{
```

```
  @if (Model.Count > 0)
```

```
  {
```

```
    <table class="table table-striped align-items-center table-bordered">
```

```
      <thead class="text-center">
```

```
        <th>Caravana</th>
```

```
        <th>Tipo</th>
```

```
        <th>Peso</th>
```

```
        <th>Sexo</th>
```

```
        <th>Potencial Ganancia</th>
```

```

        <th>Vacunar</th>

    </thead>

    <tbody>

        @foreach (Res unaR in Model)
        {
            <tr class="text-center">
                <td>@unaR.Id</td>
                <td>@unaR.TipoRes()</td>
                <td>@unaR.PesoActual</td>
                <td>@unaR.Sexo</td>
                <td>@unaR.GananciaPotencial()</td>
                <td><a class="btn btn-primary"
href="/Ganado/VacunarAnimal/@unaR.Id">Ir</a></td>
            </tr>
        }
    </table>
}
}

```

4.2.3 INDEX.CSHTML

@*

For more information on enabling MVC for empty projects, visit
<https://go.microsoft.com/fwlink/?LinkID=397860>

*@

@{

 ViewData["title"] = "Animales";

}

@using Estancia

<h1>Listado de Animales</h1>

@if (ViewBag.Mensaje != null)

{

```
<p class="text-success">@ViewBag.Mensaje</p>
}
```

```
@if (ViewBag.Vacunado != null)
{
    <p class="text-success">@ViewBag.Vacunado</p>
}
```

```
@if (ViewBag.Error != null)
{
    <p class="text-danger">@ViewBag.Error</p>
}
```

```
@if (ViewBag.Ganado.Count > 0)
{
    <table class="table table-hover table-striped align-items-center table-bordered">
        <thead class="text-center">
            <th>Caravana</th>
            <th>Raza</th>
            <th>Sexo</th>
            <th>Peso</th>
            <th>Tipo</th>
            <th>Vacunar</th>
        </thead>
        <tbody>
            @foreach (Res unaR in ViewBag.Ganado)
            {
                <tr class="text-center">
                    <td>@unaR.Id</td>
                    <td>@unaR.Raza</td>
```

```

        <td>@(unaR.Sexo == Sexo.FEMALE ? "Hembra" : "Macho")</td>

        <td>@unaR.PesoActual</td>

        <td>@unaR.TipoRes()</td>

        <td><a class="btn btn-primary"
href="/Ganado/VacunarAnimal/@unaR.Id">Ir</a></td>

    </tr>

}

</tbody>
</table>
}
else
{
    <p class="text-danger">No hay animales ingresados</p>
}

```

4.2.4 RESESLIBRESPARAASIGNAR.CSHTML

@*

For more information on enabling MVC for empty projects, visit
<https://go.microsoft.com/fwlink/?LinkID=397860>

*@

@{

}

@{

ViewData["Title"] = "Reses libres para asignar";

}

@using Estancia

@model List<Res>

@if (TempData["MensajeError"] != null)

```

{
    <p class="text-danger">@TempData["MensajeError"]</p>
}
@if (TempData["MensajeExito"] != null)
{
    <p class="text-success">@TempData["MensajeExito"]</p>
}

```

```

<h1>Reses libres para asignar</h1>

```

```

@if (Model.Count > 0)
{
    <table class="table table-hover table-striped align-items-center table-bordered">
        <thead class="text-center">
            <th> Caravana </th>
            <th> Fecha Nacimiento</th>
            <th> Sexo</th>
            <th> Es Libre</th>
            <th> Asignar</th>
        </thead>
        <tbody>
            @foreach (Res unR in Model)
            {
                <tr class="text-center">
                    <td>@unR.Id</td>
                    <td>@unR.FechaNacimiento.ToShortDateString()</td>
                    <td>@unR.Sexo</td>
                    <td>@unR.Libre</td>
                    <td><a href="/Potreros/ListarPotreros/@unR.Id" class="btn btn-
primary">Agregar</a></td>
                </tr>
            }
        }
    }

```

```

    }
    </tbody>
</table>
}
else
{
    <p class="text-info">No hay animales libres</p>
}

```

4.2.5 VACUNARANIMAL.CSHTML

@*

For more information on enabling MVC for empty projects, visit
<https://go.microsoft.com/fwlink/?LinkID=397860>

*@

@{

}

@using Estancia

@model Res

<h1>Vacunar</h1>

<form method="post">

<label>Caravana:</label>

@Model.Id

<input type="text" id="caranva" name="idAnimal" value="@Model?.Id" class="form-control" hidden />

<label>Sexo:</label>

@Model?.Sexo

<label>Raza:</label>

@Model?.Raza


```
<label>Fecha de Nacimiento:</label>
```

```
<span class="form-control">@Model?.FechaNacimiento.ToShortDateString()</span>
```

```
<label for="">Vacuna</label>
```

```
<select class="form-select" name="nombreVacuna">
```

```
    <option>Elegir vacuna</option>
```

```
    @foreach(Vacuna unaV in ViewBag.Vacunas)
```

```
    {
```

```
        <option value="@unaV.Nombre">@unaV.Nombre</option>
```

```
    }
```

```
</select>
```

```
<label for="fechalny">Fecha de Inyección:</label>
```

```
<input type="date" id="fechalny" name="fechalny" value="" class="form-control" />
```

```
<div class="form-group">
```

```
    <input type="submit" name="Vacunar" value="Vacunar" class="btn btn-primary mt-3" />
```

```
</div>
```

```
<div class="form-group">
```

```
    <a href="/Ganado/Index" class="btn btn-primary">Volver</a>
```

```
</div>
```

```
</form>
```

```
@if (ViewBag.Error != null)
```

```
{
```

```
    <p class="text-danger">@ViewBag.Error</p>
```

```
}
```

4.3 HOME

4.3.1 INDEX.CSHTML

@{

ViewData["Title"] = "Inicio";

}

<div class="text-center">

<h1 class="display-4">Obligatorio 2</h1>

<p>Participantes: Joaquin Caviglia , Emmanuel Duffaut</p>

<p>Profesor: Mauricio Quintero</p>

<p>Materia: Programacion 2</p>

</div>

4.3.2 LOGIN.CSHTML

@*

For more information on enabling MVC for empty projects, visit
<https://go.microsoft.com/fwlink/?LinkID=397860>

*@

@{

}

<form method="post">

<label for="email">Email</label>

<input type="text" id="email" name="email" class="form-control" />

<label for="contrasenia">Contraseña</label>

<input type="password" id="contrasenia" name="password" class="form-control" />

<div class="form-group">

<input type="submit" value="Enviar" class="btn btn-primary mt-3" />

</div>

</form>

```
@if (ViewBag.Error != null)
{
    <p>@ViewBag.Error</p>
}
```

4.4 POTREROS

4.4.1 LISTARPOTREROS.CSHTML

@*

For more information on enabling MVC for empty projects, visit
<https://go.microsoft.com/fwlink/?LinkID=397860>

*@

@{

}

@{

 ViewData["Title"] = "Potreros";

}

@using Estancia

@model List<Potrero>

<table class="table table-hover table-striped align-items-center table-bordered">

 <thead class="text-center">

 <th> Cantidad Hectareas </th>

 <th> Capacidad Animales</th>

 <th> Cantidad Animales</th>

 <th> Potrero</th>

 </th>

 <tbody>

 @foreach (Potrero unP in Model)

 {

 <tr class="text-center">

 <td>@unP.CantidadHectareas</td>

 <td>@unP.CapacidadAnimales</td>

 <td>@unP.CantidadAnimales</td>

 <td>Seleccionar</td>

```

        </tr>
    }
</tbody>
</table>

```

4.4.2 LISTARTODOSLOSPOTREROS.CSHTML

```
@*
```

For more information on enabling MVC for empty projects, visit
<https://go.microsoft.com/fwlink/?LinkID=397860>

```
*@
```

```
@{
```

```
}
```

```
@{
```

```
    ViewData["Title"] = "Potreros";
```

```
}
```

```
@using Estancia
```

```
@model List<Potrero>
```

```
<h1>Potreros</h1>
```

```
@if (ViewBag.Error != null)
```

```
{
```

```
    <p class="text-danger">@ViewBag.Error</p>
```

```
}
```

```
@if (Model != null)
```

```
{
```

```
    @if (Model.Count > 0)
```

```
{
```

```
    <table class="table table-striped align-items-center table-bordered">
```

```
        <thead class="text-center">
```

```

<th>Id</th>
<th>Descripción</th>
<th>Animales Pastando</th>
<th>Cantidad de Hectareas</th>
<th>Capacidad</th>
<th>Precio de Venta</th>
</thead>
<tbody>
    @foreach(Potrero unP in Model)
    {
        <tr class="text-center">
            <td>@unP.Id</td>
            <td>@unP.Descripcion</td>
            <td>@(unP.CantidadAnimales == 0 ? "No Hay" : unP.CantidadAnimales)</td>
            <td>@unP.CantidadHectareas</td>
            <td>@unP.CapacidadMaxima</td>
            <td>@unP.CostoTotalCrianza()</td>
        </tr>
    }

</tbody>
</table>
}
}

```

4.5 TAREAS

4.5.1 ALTA.CSHTML

@*

For more information on enabling MVC for empty projects, visit
<https://go.microsoft.com/fwlink/?LinkID=397860>

*@

@{

}

@{

 ViewData["Title"] = "Alta de Tarea";

}

<h1>Alta de Tarea</h1>

<form method="post">

 <label for="descripcion">Ingrese descripción de la tarea:</label>

 <input type="text" id="descripcion" name="Descripcion" class="form-control" />

 <div class="form-group">

 <input type="submit" value="Crear" class="btn btn-primary mt-3" />

 </div>

</form>

@if (ViewBag.Error != null)

{

 <p class="text-danger">@ViewBag.Error</p>

}

4.5.2 EDITAR.CSHTML

@*

For more information on enabling MVC for empty projects, visit
<https://go.microsoft.com/fwlink/?LinkID=397860>

*@

@{

 ViewData["Title"] = "Cerrar Tarea";

}

@using Estancia

@model TareaAsignada

<h1>Cerrar tarea</h1>

<form method="post">

 <label for="desc">Descripción:</label>

 @Model.Tarea.Descripcion

 <label for="fecPactada">Fecha Pactada:</label>

 @Model.FechaPactada.ToShortDateString()

 <label for="comentario">Comentario:</label>

 <input type="text" id="comentario" name="Comentario" value="" class="form-control" />

 <label for="fecCierre">Fecha de Cierre:</label>

 <input type="date" name="FechaCierre" value="" class="form-control" />

 <div class="form-group">

 <input type="submit" value="Guardar" class="btn btn-primary mt-3" />

 </div>

 <input type="number" name="Tarea.Id" value="@Model?.Tarea.Id" class="form-control" hidden />


```
</form>
```

```
@if (ViewBag.Error != null)
{
    <p class="text-danger">@ViewBag.Error</p>
}
```

4.5.3 SHARED.CSHTML

```
@{
    string nombre = Context.Session.GetString("nombre");
    string tipo = Context.Session.GetString("rol");
    string email = Context.Session.GetString("email");
}

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>@ViewData["Title"] - MVC</title>
    <link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.min.css" />
    <link rel="stylesheet" href="~/css/site.css" asp-append-version="true" />
    <link rel="stylesheet" href="~/MVC.styles.css" asp-append-version="true" />
</head>
<body>
    <header>
        <nav class="navbar navbar-expand-sm navbar-toggleable-sm navbar-light bg-white
border-bottom box-shadow mb-3">
            <div class="container-fluid">
                <a class="navbar-brand" asp-area="" asp-controller="Home" asp-
action="Index">MVC</a>
                <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-
target=".navbar-collapse" aria-controls="navbarSupportedContent"
```

```

        aria-expanded="false" aria-label="Toggle navigation">
        <span class="navbar-toggler-icon"></span>
    </button>
    <div class="navbar-collapse collapse d-sm-inline-flex justify-content-between">
        <ul class="navbar-nav flex-grow-1">
            <li class="nav-item">
                <a class="nav-link text-dark" asp-area="" asp-controller="Home" asp-
action="Index">Home</a>
            </li>

```

```

        @if (tipo == "peon")
        {
            <li class="nav-item">
                <a class="nav-link text-dark" href="/Ganado/">Animales</a>
            </li>

```

```

            <li class="nav-item">
                <a class="nav-link text-dark"
href="/Empleados/TareasNoCompletadas">Tareas</a>
            </li>

```

```

            <li class="nav-item">
                <a class="nav-link text-dark" href="/Ganado/AltaBovino">Alta Bovino</a>
            </li>
            <li class="nav-item">
                <a class="nav-link text-dark" href="/Ganado/ResesLibresParaAsignar">Asignar
Reses A Potreros</a>
            </li>
            <li class="nav-item">

```

```

        <a class="nav-link text-dark" href="/Empleados/VerPeones">Peones</a>
    </li>
    <li class="nav-item">
        <a class="nav-link text-dark" href="/Empleados/SeleccionarPeon">Asignar
Tarea</a>
    </li>
    <li class="nav-item">
        <a class="nav-link text-dark"
href="/Potreros/ListarTodosLosPotreros">Potreros</a>
    </li>
    <li class="nav-item">
        <a class="nav-link text-dark" href="/Ganado/AnimalesPorPesoYTipo">Animales
por Tipo</a>
    </li>
</ul>
<ul class="navbar-nav flex-grow-1">
    @if (nombre == null)
    {
        <li class="nav-item">
            <a class="nav-link text-dark" href="/Home/Login">Login</a>
        </li>
        <li class="nav-item">
            <a class="nav-link text-dark"
href="/Empleados/RegistrarPeon">Registrarse</a>
        </li>
    }
    @if (nombre != null)
    {
        <li class="nav-item">
            <a class="nav-link text-dark" href="/Home/Logout">Logout</a>
        </li>
    }

```

```

        @if(tipo == "peon")
        {
            <li class="nav-item">
                <a class="nav-link text-dark" href="/Empleados/PerfilPeon">Perfil</a>
            </li>
        }
    </ul>
</div>
</div>
</nav>
</header>
<div class="container">
    <main role="main" class="pb-3">
        @RenderBody()
    </main>
</div>

<footer class="border-top footer text-muted">
    <div class="container">
        &copy; 2024 - MVC - <a asp-area="" asp-controller="Home" asp-
action="Privacy">Privacy</a>
    </div>
</footer>
<script src="~/lib/jquery/dist/jquery.min.js"></script>
<script src="~/lib/bootstrap/dist/js/bootstrap.bundle.min.js"></script>
<script src="~/js/site.js" asp-append-version="true"></script>

    @await RenderSectionAsync("Scripts", required: false)
</body>
</html>

```

4.6 CONTROLLERS

4.6.1 EMPLEADOSCONTROLLER.CS

```
using Estancia;
```

```
using Microsoft.AspNetCore.Mvc;
```

```
namespace MVC.Controllers
```

```
{
```

```
    public class EmpleadosController : Controller
```

```
    {
```

```
        Sistema _sistema = Sistema.Instance;
```

```
        public IActionResult Index()
```

```
        {
```

```
            return View();
```

```
        }
```

```
        public IActionResult RegistrarPeon()
```

```
        {
```

```
            return View();
```

```
        }
```

```
        [HttpPost]
```

```
        public IActionResult RegistrarPeon(Peon nuevoPeon)
```

```
        {
```

```
            try
```

```
            {
```

```
                _sistema.InstanceEmpleados(nuevoPeon);
```

```
                TempData["Mensaje"] = $"{nuevoPeon.Nombre} Registrado Correctamente!";
```

```
                return RedirectToAction("Login", "Home");
```

```
            }
```

```
            catch (Exception ex)
```

```
            {
```

```

        ViewBag.Error = ex.Message;
    }
    return View();
}

```

```

public IActionResult PerfilPeon()
{
    if (HttpContext.Session.GetString("rol") == "peon")
    {
        try
        {
            return View(_sistema.BuscarPeon(HttpContext.Session.GetString("email")));
        }
        catch (Exception ex)
        {
            ViewBag.Error = ex.Message;
        }

        return View();
    }
    return RedirectToAction("Index", "home");
}

```

//ASK: tengo alguna manera más facil de acceder al peon para luego filtrar sus tareas asignadas no completadas? En este caso saco el dato de la session

```

public IActionResult TareasNoCompletadas()
{
    if (HttpContext.Session.GetString("rol") == "peon")
    {
        List<TareaAsignada> tareasNoCompletadas = new List<TareaAsignada>();
    }
}

```

```

        ViewBag.Mensaje = TempData["exitoEditarTareaAsignada"];

        try
        {
            Peon peon = _sistema.BuscarPeon(HttpContext.Session.GetString("email"));
            foreach (TareaAsignada unaTa in peon.TareasAsignadas)
            {
                if (unaTa.Comentario == null)
                {
                    tareasNoCompletadas.Add(unaTa);
                }
            }
            tareasNoCompletadas.Sort();
            return View(tareasNoCompletadas);
        }
        catch (Exception ex)
        {
            ViewBag.Error = ex.Message;
        }
        return View();
    }

    return RedirectToAction("Index", "home");
}

public IActionResult VerPeones()
{
    if (HttpContext.Session.GetString("rol") == "capataz")
    {
        List<Peon> peones = new List<Peon>();

        try
        {
            peones = _sistema.GetPeones();
        }
    }
}

```

```

    }
    catch (Exception ex)
    {
        ViewBag.Error = ex.Message;
    }
    return View(peones);
}
return RedirectToAction("Index", "home");
}

public IActionResult ListarParaAsignar(string email)
{
    if (HttpContext.Session.GetString("rol") == "capataz")
    {
        if (TempData["Email"] != null)
            ViewBag.Email = TempData["Email"];
        else
        {
            ViewBag.Email = email;
        }

        ViewBag.Exito = TempData["Exito"];
        return View(_sistema.GetTareas());
    }
    return RedirectToAction("Index", "home");
}

```

```

public IActionResult Asignar(int id, string email)
{
    if (HttpContext.Session.GetString("rol") == "capataz")

```



```

{
    ViewBag.Email = TempData["email"];
    try
    {
        DateTime fechaHoy = new DateTime();
        Peon peon = _sistema.BuscarPeon(email);
        Tarea tarea = _sistema.BuscarTarea(id);
        TareaAsignada tareaAsignada = new TareaAsignada(fechaHoy, tarea);
        peon.TareasAsignadas.Add(tareaAsignada);
        TempData["Exito"] = "Tarea asignada correctamente";
        TempData["Email"] = email;
        return RedirectToAction("ListarParaAsignar");

    }
    catch (Exception ex)
    {
        TempData["Error"] = ex.Message;
    }
    return RedirectToAction("SeleccionarPeon");
}
return RedirectToAction("Index", "home");
}

public IActionResult VerTareas(string email)
{
    if (HttpContext.Session.GetString("rol") == "capataz")
    {
        List<TareaAsignada> tareasPeon = new List<TareaAsignada>();
        try
        {
            Peon peon = _sistema.BuscarPeon(email);
            foreach (TareaAsignada unTa in peon.TareasAsignadas)

```

```

        tareasPeon.Add(unTa);
        return View(tareasPeon);
    }
    catch (Exception ex)
    {
        ViewBag.Error = ex.Message;
    }
    return View();
}
return RedirectToAction("Index", "home");
}

public IActionResult SeleccionarPeon()
{
    if (HttpContext.Session.GetString("rol") == "capataz")
    {
        List<Peon> peones = new List<Peon>();
        ViewBag.Error = TempData["Error"];
        ViewBag.Exito = TempData["Exito"];
        try
        {
            peones = _sistema.GetPeones();
        }
        catch (Exception ex)
        {
            ViewBag.Error = ex.Message;
        }
        return View(peones);
    }
    return RedirectToAction("Index", "home");
}

```

```
}  
}
```

4.6.2 GANADOCONTROLLER.CS

```
using Estancia;
```

```
using Microsoft.AspNetCore.Mvc;
```

```
namespace MVC.Controllers
```

```
{
```

```
    public class GanadoController : Controller
```

```
    {
```

```
        Sistema _miSistema = Sistema.Instance;
```

```
        public IActionResult Index()
```

```
        {
```

```
            if (HttpContext.Session.GetString("rol") == "peon")
```

```
            {
```

```
                ViewBag.Ganado = _miSistema.GetAnimales();
```

```
                ViewBag.Mensaje = TempData["exitoAltaBovino"];
```

```
                ViewBag.Vacunado = TempData["exitoInyeccion"];
```

```
                ViewBag.Error = TempData["errorInyeccion"];
```

```
                return View();
```

```
            }
```

```
            if (HttpContext.Session.GetString("rol") == "capataz")
```

```
            {
```

```
                ViewBag.Ganado = _miSistema.GetAnimales();
```

```
                ViewBag.Mensaje = TempData["exitoAltaBovino"];
```

```
                return View();
```

```
            }
```

```
            return RedirectToAction("Index", "home");
```

```

    }

    public IActionResult VacunarAnimal(string id)
    {
        if (HttpContext.Session.GetString("rol") == "capataz" ||
            HttpContext.Session.GetString("rol") == "peon")
        {
            ViewBag.Vacunas = _miSistema.GetVacunas();
            return View(_miSistema.GetUnAnimal(id));
        }
        return RedirectToAction("Index", "home");
    }

    [HttpPost]
    public IActionResult VacunarAnimal(string idAnimal, string nombreVacuna, DateTime
    fechalny)
    {
        if (HttpContext.Session.GetString("rol") == "peon")
        {
            ViewBag.Vacunas = _miSistema.GetVacunas();

            try
            {
                Res animalAVacunar = _miSistema.GetUnAnimal(idAnimal);
                animalAVacunar.EsMayorATresMeses();
                Vacuna vacuna = _miSistema.GetUnaVacuna(nombreVacuna);
                DateTime hoy = fechalny;
                Inyeccion nuevaInyeccion = new Inyeccion(vacuna, hoy, hoy.AddYears(1));
                animalAVacunar.Inyecciones.Add(nuevaInyeccion);

                TempData["exitoinyeccion"] = $"Se inyectó el animal {animalAVacunar.Id} con la
                vacuna {vacuna.Nombre} correctamente";
            }
        }
    }

```

```

    }
    catch (Exception ex)
    {
        TempData["errorInyeccion"] = ex.Message;
    }
    return RedirectToAction("Index");
}
return RedirectToAction("Index", "home");
}

```

```

public IActionResult AltaBovino()
{
    if (HttpContext.Session.GetString("rol") == "capataz")
    {
        return View();
    }
    return RedirectToAction("Index", "home");
}

```

```

[HttpPost]
public IActionResult AltaBovino(Bovino nuevoBovino)
{
    if (HttpContext.Session.GetString("rol") == "capataz")
    {
        try
        {
            _miSistema.InstanceRes(nuevoBovino);
            TempData["exitoAltaBovino"] = $"Se dio de alta el bovino {nuevoBovino.Id}";
            return RedirectToAction("Index");
        }
        catch (Exception ex)

```

```

    {
        ViewBag.Error = ex.Message;
    }
    return View(nuevoBovino);
}
return RedirectToAction("Index", "home");
}

```

```

public IActionResult ResesLibresParaAsignar()
{
    if (HttpContext.Session.GetString("rol") == "capataz")
    {
        List<Res> reses = _miSistema.GetAnimales();
        List<Res> resesLibres = new List<Res>();
        foreach (Res res in reses)
        {
            if (res.Libre)
                resesLibres.Add(res);
        }
        return View(resesLibres);
    }
    return RedirectToAction("Index", "home");
}

```

```

public IActionResult AnimalesPorPesoYTipo()
{
    return View(_miSistema.GetAnimales());
}

```

```

[HttpPost]
public IActionResult AnimalesPorPesoYTipo(int peso, string tipo)

```

```

{
    if (HttpContext.Session.GetString("rol") == "capataz")
    {
        List<Res> listado = new List<Res>();
        List<Res> resesPorTipo = _miSistema.GetResesPorTipo(tipo);

        foreach (Res unaR in resesPorTipo)
            if (unaR.PesoActual > peso)
                listado.Add(unaR);

        return View(listado);
    }
    return RedirectToAction("Index", "home");
}
}
}

```

4.6.3 HOMECONTROLLER.CS

```

using Microsoft.AspNetCore.Mvc;
using MVC.Models;
using System.Diagnostics;
using Estancia;

namespace MVC.Controllers
{
    public class HomeController : Controller
    {
        private readonly ILogger<HomeController> _logger;
        private Sistema _sistema = Sistema.Instance;
        public HomeController(ILogger<HomeController> logger)
        {
            _logger = logger;

```

```
}
```

```
public IActionResult Index()
```

```
{
```

```
    return View();
```

```
}
```

```
[ResponseCache(Duration = 0, Location = ResponseCacheLocation.None, NoStore = true)]
```

```
public IActionResult Error()
```

```
{
```

```
    return View(new ErrorViewModel { RequestId = Activity.Current?.Id ?? HttpContext.TraceIdentifier });
```

```
}
```

```
public IActionResult Login()
```

```
{
```

```
    return View();
```

```
}
```

```
[HttpPost]
```

```
public IActionResult Login(string email, string password)
```

```
{
```

```
    try
```

```
    {
```

```
        Empleado usuario = _sistema.AutenticarUsuario(email, password);
```

```
        HttpContext.Session.SetString("nombre", usuario.Nombre);
```

```
        HttpContext.Session.SetString("rol", usuario.Rol());
```

```
        HttpContext.Session.SetString("email", usuario.Email);
```

```
        return RedirectToAction("index", "home");
```

```
}
```



```

        catch(Exception ex)
        {
            ViewBag.Error = ex.Message;
        }
        return View();
    }

    public IActionResult Logout()
    {
        HttpContext.Session.Clear();
        return RedirectToAction("Index");
    }
}
}

```

4.6.4 POTREROSCONTROLLER.CS

```

using Estancia;
using Microsoft.AspNetCore.Mvc;

```

```

namespace MVC.Controllers
{
    public class PotrerosController : Controller
    {
        Sistema _miSistema = Sistema.Instance;

        public IActionResult Index()
        {
            return View();
        }

        public IActionResult ListarPotreros(string id)
        {

```

```

if (HttpContext.Session.GetString("rol") == "capataz")
{
    ViewBag.IdRes = id;
    List<Potrero> potreros = _miSistema.GetPotreros();
    List<Potrero> potrerosDisponibles = new List<Potrero>();
    foreach (Potrero unP in potreros)
    {
        if (unP.HayEspacio())
        {
            potrerosDisponibles.Add(unP);
        }
    }
    return View(potrerosDisponibles);
}
return RedirectToAction("Index", "home");
}

public IActionResult AgregarRes(int IdPotrero, string IdRes)
{
    if (HttpContext.Session.GetString("rol") == "capataz")
    {
        try
        {
            Potrero potrero = _miSistema.BuscarPotrero(IdPotrero);
            Res res = _miSistema.GetUnAnimal(IdRes);
            potrero.AddRes(res);
            res.Libre = false;
            TempData["MensajeExito"] = $"El animal {res.Id} fué agregado correctamente";
        }
        catch (Exception ex)
        {
            TempData["MensajeError"] = ex.Message;
        }
    }
}

```

```

    }

    return RedirectToAction("ResesLibresParaAsignar", "Ganado");
}

return RedirectToAction("Index", "home");
}

public IActionResult ListarTodosLosPotreros()
{
    if (HttpContext.Session.GetString("rol") == "capataz")
    {
        List<Potrero> potreros = _miSistema.GetPotreros();

        try
        {
            potreros.Sort();
            return View(potreros);
        }
        catch (Exception ex)
        {
            ViewBag.Error = ex.Message;
        }

        return View();
    }

    return RedirectToAction("Index", "home");
}
}
}
}

```

4.6.5 TAREASCONTROLLER.CS

```

using Estancia;

using Microsoft.AspNetCore.Mvc;

using System.Threading;

```

```
namespace MVC.Controllers
```

```
{
```

```
    public class TareasController : Controller
```

```
    {
```

```
        Sistema _sistema = Sistema.Instance;
```

```
        public IActionResult Editar(int id)
```

```
        {
```

```
            if (HttpContext.Session.GetString("rol") == "peon")
```

```
            {
```

```
                try
```

```
                {
```

```
                    //Busco peon por la session y luego su tarea segun el id.
```

```
                    Peon peon = _sistema.BuscarPeon(HttpContext.Session.GetString("email"));
```

```
                    TareaAsignada tareaA = peon.BuscarTareaAsignada(id);
```

```
                    return View(tareaA);
```

```
                }
```

```
                catch (Exception ex)
```

```
                {
```

```
                    ViewBag.Error = ex.Message;
```

```
                }
```

```
                return View();
```

```
            }
```

```
            return RedirectToAction("Index", "home");
```

```
        }
```

```
        [HttpPost]
```

```
        public IActionResult Editar(TareaAsignada tareaA)
```

```
        {
```

```
            if (HttpContext.Session.GetString("rol") == "peon")
```

```

{
    try
    {
        //Busco peon por la session y luego su tarea segun el id.
        Peon peon = _sistema.BuscarPeon(HttpContext.Session.GetString("email"));
        peon.actualizarTareaAsignada(tareaA);

        TempData["exitoEditarTareaAsignada"] = $"La tarea {tareaA.Comentario} fue
completada y cerrada";

        //redirecciono a tareas no completadas del peon
        return RedirectToAction("TareasNoCompletadas", "Empleados");
    }
    catch (Exception ex)
    {
        ViewBag.Error = ex.Message;
    }
    return View(tareaA);
}
return RedirectToAction("Index", "home");
}

```

```

public IActionResult Alta()
{
    return View();
}

[HttpPost]
public IActionResult Alta(string descripcion)
{
    try

```

```

{
    Tarea tareaNueva = new Tarea(descripcion);
    _sistema.InstanceTask(tareaNueva);
    TempData["Exito"] = "Tarea agregada correctamente.";
    return RedirectToAction("SeleccionarPeon", "Empleados");
}
catch(Exception ex)
{
    ViewBag.Error = ex.Message;
}
return View();
}
}
}

```

5 CASOS DE USO.

5.1 CAPATAZ

5.1.1 ASIGNAR TAREA.

Identificador: CU-01	Nombre: Asignar tarea
Autor:	Emmanuel Duffaut/Joaquin Caviglia
Fecha:	20/06/2024
Descripción: Permite al capataz asignar una tarea.	
Actor/es: Usuario del sistema	
Precondiciones: -.	
Poscondiciones: El peón queda con una tarea asignada para realizar.	

<p>Flujo Normal:</p> <ol style="list-style-type: none"> 1. El actor selecciona la opción "Asignar tarea". 2. El sistema solicita ingresar nombre. 3. El sistema solicita ingresar email. 4. El actor selecciona una tarea. 5. Se asocia tarea y peón. 6. Se muestra cartel de éxito. 7. Vuelve al menú anterior
<p>Flujo/s Alternativo/s:</p> <p>2a. El actor cancela la asignación.</p> <p>2a - 1. La tarea no es asignada.</p> <p>3a. El peón no se encuentra disponible.</p> <p>3a - 1. La tarea no es asignada.</p> <p>4a. No se puede asignar la tarea.</p> <p>4a - 1. Se muestra mensaje de error.</p> <p>4b. Tarea ya asignada.</p> <p>4b - 1. Se muestra mensaje explicando</p>
<p>Flujo/s Excepcionales/s:</p> <p>Se interrumpe la ejecución del programa, la tarea no se asigna.</p>

5.1.2 ALTA BOVINO

Identificador: CU-01	Nombre: Alta Bovino
Autor:	Emmanuel Duffaut/Joaquin Caviglia
Fecha:	20/06/2024
<p>Descripción:</p> <p>Permite al capataz dar de alta un nuevo Bovino en el sistema</p>	
<p>Actor/es:</p> <p>Usuario del sistema</p>	

Precondiciones:
-.
Poscondiciones:
El bovino es agregado a la lista de bovinos en el sistema.
Flujo Normal:
<ol style="list-style-type: none"> 1. El actor selecciona la opción "Alta Bovino". 2. El sistema solicita ingresar los datos correspondientes. 3. Se muestra cartel de éxito. 4. Vuelve al menú anterior
Flujo/s Alternativo/s:
El actor cancela la asignación.
Bovino no creado
El id ya existe en el sistema
Se muestra mensaje explicando
Flujo/s Excepcionales/s:
Se interrumpe la ejecución del programa, el bovino no se ingresa al sistema.

5.1.3 LOGIN

Identificador: CU-01	Nombre: Login
Autor:	Emmanuel Duffaut/Joaquin Caviglia
Fecha:	20/06/2024
Descripción: Permite al Capataz o Peón iniciar sesión en el sistema.	
Actor/es: Usuario del sistema	
Precondiciones:	
-.	
Poscondiciones:	
El usuario accede a su página de inicio en el sistema.	

Flujo Normal: <ol style="list-style-type: none"> 1. El sistema solicita usuario y contraseña. 2. El actor ingresa usuario y contraseña. 3. El sistema corrobora los datos ingresados. 4. Se muestra cartel de éxito. 5. El actor ingresa al sistema.
Flujo/s Alternativo/s: <p>2a. El actor erra en los datos.</p> <p>2a - 1. El sistema muestra error informativo.</p> <p>3a. El actor no está ingresado en el sistema.</p> <p>3a - 1. El sistema le informa y le permite darse de alta.</p>
Flujo/s Excepcionales/s: <p>Se interrumpe la ejecución del programa, el usuario no puede iniciar sesión.</p>

5.1.4 ASIGNAR ANIMAL A UN POTRERO

Identificador: CU-01	Nombre: Asignar Reses a Potreros
Autor:	Emmanuel Duffaut/Joaquin Caviglia
Fecha:	20/06/2024
Descripción: Permite al Capataz elegir un animal libre y asignarlo a un potrero con disponibilidad.	
Actor/es: Usuario del sistema	
Precondiciones: -.	
Poscondiciones: El usuario capataz accede a su página de inicio en el sistema.	
Flujo Normal: <ol style="list-style-type: none"> 1. El sistema solicita usuario y contraseña. 2. El actor ingresa usuario y contraseña. 3. El sistema corrobora los datos ingresados. 4. Se muestra cartel de éxito. 	

5. El capataz selección un animal libre y lo asigna a un potrero con disponibilidad.
Flujo/s Alternativo/s: 2a. El potrero no tiene disponibilidad 2a - 1. El sistema muestra error informativo.
Flujo/s Excepcionales/s: Se interrumpe la ejecución del programa, el animal se mantiene sin potrero asignado.

5.1.5 LISTAR PEONES

Identificador: CU-01	Nombre: Peones
Autor:	Emmanuel Duffaut/Joaquin Caviglia
Fecha:	20/06/2024
Descripción: Permite al Capataz listar sus peones.	
Actor/es: Usuario del sistema	
Precondiciones: -.	
Poscondiciones: El usuario capataz accede a su página de inicio en el sistema.	
Flujo Normal: 1. El sistema solicita usuario y contraseña. 2. El actor ingresa usuario y contraseña. 3. El sistema corrobora los datos ingresados. 4. Se muestra cartel de éxito. 5. El capataz se dirige a la vista "Peones".	
Flujo/s Alternativo/s: 2a. No tiene peones a cargo. 2a - 1. El sistema muestra error informativo.	
Flujo/s Excepcionales/s: Se interrumpe la ejecución del programa, no se listan los peones.	

5.1.6 LISTAR ANIMALES POR PESO Y TIPO

Identificador: CU-01	Nombre: Animales por tipo
Autor:	Emmanuel Duffaut/Joaquin Caviglia
Fecha:	20/06/2024
Descripción: Permite al Capataz listar los animales y filtrarlos por tipo y peso.	
Actor/es: Usuario del sistema	
Precondiciones: -.	
Poscondiciones: El usuario capataz accede a su página de inicio en el sistema.	
Flujo Normal: <ol style="list-style-type: none">1. El sistema solicita usuario y contraseña.2. El actor ingresa usuario y contraseña.3. El sistema corrobora los datos ingresados.4. Se muestra cartel de éxito.5. El capataz se dirige a la vista "Animales por Tipo".	
Flujo/s Alternativo/s:	
Flujo/s Excepcionales/s: Se interrumpe la ejecución del programa, no se listan los animales.	

5.1.7 LISTAR POTREROS

Identificador: CU-01	Nombre: Potreros
Autor:	Emmanuel Duffaut/Joaquin Caviglia
Fecha:	20/06/2024
Descripción: Permite al Capataz listar los potreros.	
Actor/es: Usuario del sistema	

Precondiciones:
-.
Poscondiciones:
El usuario capataz accede a su página de inicio en el sistema.
Flujo Normal:
<ol style="list-style-type: none"> 1. El sistema solicita usuario y contraseña. 2. El actor ingresa usuario y contraseña. 3. El sistema corrobora los datos ingresados. 4. Se muestra cartel de éxito. 5. El capataz se dirige a la vista "Potreros".
Flujo/s Alternativo/s:
Flujo/s Excepcionales/s:
Se interrumpe la ejecución del programa, no se listan los animales.

5.1.8 LOGOUT

Identificador: CU-01	Nombre: Logout
Autor:	Emmanuel Duffaut/Joaquin Caviglia
Fecha:	20/06/2024
Descripción: Permite al Capataz o Peón cerrar sesión en el sistema.	
Actor/es: Usuario del sistema	
Precondiciones:	
-	
Poscondiciones:	
El usuario es redirigido a la página de inicio del sistema.	
Flujo Normal:	
<ol style="list-style-type: none"> 1. El usuario selecciona "Logout" 	

Flujo/s Alternativo/s:
Flujo/s Excepcionales/s: Se interrumpe la ejecución del programa, el usuario no puede cerrar sesión.

5.2 PEÓN

5.2.1 COMPLETAR TAREA.

Identificador: CU-01	Nombre: Tareas no completadas.
Autor:	Emmanuel Duffaut/Joaquin Caviglia
Fecha:	20/06/2024
Descripción: Permite al peón agregar un comentario y fecha en la tarea para cerrarla.	
Actor/es: Usuario del sistema	
Precondiciones: -.	
Poscondiciones: Se finaliza tarea con comentario al respecto y fecha.	
Flujo Normal: <ol style="list-style-type: none"> 1. El actor selecciona de sus tareas la que desea comentar. 2. El actor selecciona la opción "Comentar tarea". 3. El sistema solicita ingresar un comentario. 4. El actor selecciona guardar comentario. 5. Se muestra cartel de éxito. 6. Vuelve al menú anterior 	
Flujo/s Alternativo/s: <ol style="list-style-type: none"> 2a. El actor cancela el comentario. <ol style="list-style-type: none"> 2a - 1. La tarea queda sin comentario. 3a. La tarea no permite comentar. <ol style="list-style-type: none"> 3a - 1. La tarea queda sin comentario. 	

Flujo/s Excepcionales/s:

Se interrumpe la ejecución del programa, la tarea no se puede comentar.

5.2.2 REGISTRAR VACUNACIÓN.

Identificador: CU-01	Nombre: Vacunar.
Autor:	Emmanuel Duffaut/Joaquin Caviglia
Fecha:	20/06/2024
Descripción: Permite al Peón agregar una inyección en un ovino o bovino.	
Actor/es: Usuario del sistema	
Precondiciones: -.	
Poscondiciones: La inyección queda cargada en el animal.	
Flujo Normal: <ol style="list-style-type: none">1. El actor selecciona el animal a ser inyectado.2. El actor selecciona el botón "Agregar Inyección".3. El actor elige una vacuna de un listado.4. El sistema chequea que el animal sea mayor a 3 meses.5. Se muestra cartel de éxito.6. Vuelve al menú anterior	
Flujo/s Alternativo/s: 2a. El actor cancela la inyección. 2a - 1. El animal no tiene cargada la inyección. 3a. La vacuna no se encuentra en la lista. 3a - 1. El sistema me permite dar de alta la vacuna.	
Flujo/s Excepcionales/s: Se interrumpe la ejecución del programa, la inyección no se puede cargar.	

5.2.3 LISTAR TAREAS

Identificador: CU-01	Nombre: Tareas no completas.
Autor:	Emmanuel Duffaut/Joaquin Caviglia
Fecha:	20/06/2024
Descripción: Permite al Peón visualizar sus tareas aún no completas	
Actor/es: Usuario del sistema	
Precondiciones: -.	
Poscondiciones: Las tareas se listan.	
Flujo Normal: 1. El actor selecciona en el menú "Tareas"	
Flujo/s Alternativo/s: 2a. No tiene tareas sin completar.	
Flujo/s Excepcionales/s: Se interrumpe la ejecución del programa, no se listan las tareas.	

5.2.4 LOGIN

Identificador: CU-01	Nombre: Login
Autor:	Emmanuel Duffaut/Joaquin Caviglia
Fecha:	20/06/2024
Descripción: Permite al Capataz o Peón iniciar sesión en el sistema.	
Actor/es: Usuario del sistema	
Precondiciones: -.	

Poscondiciones:
El usuario accede a su página de inicio en el sistema.
Flujo Normal:
6. El sistema solicita usuario y contraseña. 7. El actor ingresa usuario y contraseña. 8. El sistema corrobora los datos ingresados. 9. Se muestra cartel de éxito. 10. El actor ingresa al sistema.
Flujo/s Alternativo/s:
2a. El actor erra en los datos. 2a - 1. El sistema muestra error informativo. 3a. El actor no está ingresado en el sistema. 3a - 1. El sistema le informa y le permite darse de alta.
Flujo/s Excepcionales/s:
Se interrumpe la ejecución del programa, el usuario no puede iniciar sesión.

5.2.5 LOGOUT

Identificador: CU-01	Nombre: Logout
Autor:	Emmanuel Duffaut/Joaquin Caviglia
Fecha:	20/06/2024
Descripción: Permite al Capataz o Peón cerrar sesión en el sistema.	
Actor/es: Usuario del sistema	
Precondiciones: -.	
Poscondiciones: El usuario es redirigido a la página de inicio del sistema.	
Flujo Normal:	

2. El usuario selecciona "Logout"
Flujo/s Alternativo/s:
Flujo/s Excepcionales/s: Se interrumpe la ejecución del programa, el usuario no puede cerrar sesión.

5.2.6 REGISTRARSE

Identificador: CU-01	Nombre: Registrarse
Autor:	Emmanuel Duffaut/Joaquin Caviglia
Fecha:	20/06/2024
Descripción: Permite al Peón registrarse en el sistema.	
Actor/es: Usuario del sistema	
Precondiciones: -	
Poscondiciones: El usuario es redirigido a la página de inicio del sistema.	
Flujo Normal: 1. El usuario selecciona "Registrarse"	
Flujo/s Alternativo/s:	
Flujo/s Excepcionales/s: Se interrumpe la ejecución del programa, el usuario no puede registrarse.	

6 CASOS DE PRUEBA

6.1 PEÓN

ID	Escenario de test	Pasos	Datos utilizados	Resultado esperado
1	Registrar Peón	Ingresar los siguientes datos: email, contraseña, nombre y residencia	john.smith@estancia.com 1234567AA Juan Si	Registro exitoso, se redirige a la página de inicio.
2	Registro datos incorrectos	Ingresar los siguientes datos: email, contraseña, nombre y residencia	juan.rodriguez@estancia.uy 12345 Juan Si	No valida datos y salta excepción, muestra información de error
3	Login	Ingresar nombre de usuario y contraseña.	Usuario: eduffaut Contraseña: Pass123!	Iniciar sesión, redirige al Home
4	Logout	Seleccionar opción de Logout		Redirido al Home
5	Completar tarea	Elegimos tarea, agregamos comentario y fecha		Se cierra tarea y se muestra cartel de éxito.
6	Registrar Vacunación	Se elige el animal y luego la vacuna		Se muestra cartel de éxito.

6.2 CAPATAZ

ID	Escenario de test	Pasos	Datos utilizados	Resultado esperado
1	Login	Ingresar nombre de usuario y contraseña.	Usuario: jose.perez@estancia.com Contraseña: CapEstancia123	Iniciar sesión, redirige al Home
2	Logout	Seleccionar opción de Logout		Redirido al Home
3	Listar Potreros	Seleccionar opción de Potreros		Muestra lista de potreros
4	Asignar tarea a peón	Listamos peones y luego seleccionamos la tarea.		Tarea asignada, se muestra cartel de éxito.
5	Alta Bovino	Seleccionar del menú la opción "Alta Bovino", ingresar datos solicitados		Alta de bovino, se muestra cartel de éxito.
6	Listar peones	Seleccionar opción de Peones		Listado de peones.
7	Listar Animales(peso y tipo)	Seleccionar opción "Animales por Tipo"	Selecciono opción tipo = bovino	Listado de Bovinos

7 CONSULTAS IA

7.1 VACUNAS.

- E** decime 10 tipos de vacunas de bovinos y 10 de ovinos con su patogeno y descripcion

Mostrar versiones  



Vacunas para Bovinos:

1. Fiebre Aftosa:

- **Patógeno:** Virus de la Fiebre Aftosa (FAV)
- **Descripción:** Enfermedad viral altamente contagiosa que afecta a bovinos, ovinos y porcinos. Se caracteriza por fiebre, vesículas en la boca y patas, y dificultad para caminar.



Vacuna contra la Fiebre Aftosa

2. Brucelosis:

- **Patógeno:** Bacteria Brucella abortus
- **Descripción:** Enfermedad bacteriana que afecta el sistema reproductivo de bovinos y ovinos. Provoca abortos, infertilidad y nacimiento de terneros débiles.



Vacuna contra la Brucelosis

3. Carbón:

- **Patógeno:** Bacteria Clostridium chauvoei
- **Descripción:** Enfermedad bacteriana grave que afecta a bovinos, ovinos y otros animales. Se caracteriza por muerte súbita, hinchazón y sangre en la nariz y boca.



Vacuna contra el Carbón

4. Manquera:

- **Patógeno:** Virus de la Manquera (BVDV)
- **Descripción:** Enfermedad viral que afecta las patas de bovinos y ovinos. Provoca cojera, úlceras en las patas y disminución de la producción de leche.

5. Rabia:

- **Patógeno:** Virus de la Rabia
- **Descripción:** Enfermedad viral mortal que afecta a todos los mamíferos, incluyendo bovinos y ovinos. Se caracteriza por síntomas neurológicos como agresividad, parálisis y muerte.



Vacuna contra la Rabia

6. Leptospirosis:

- **Patógeno:** Bacteria *Leptospira* spp.
- **Descripción:** Enfermedad bacteriana que afecta a bovinos y ovinos. Provoca abortos, infertilidad, problemas renales y hepáticos.

7. Diarrea Viral Bovina (DVB):

- **Patógeno:** Virus de la Diarrea Viral Bovina (BVDV)
- **Descripción:** Enfermedad viral que afecta a bovinos de todas las edades. Provoca diarrea, neumonía, y disminución de la producción de leche.

8. Rinotraqueítis Infecciosa Bovina (IBR):

- **Patógeno:** Virus de la Rinotraqueítis Infecciosa Bovina (IBR)
- **Descripción:** Enfermedad viral que afecta a bovinos de todas las edades. Provoca problemas respiratorios, abortos y disminución de la producción de leche.

9. Parainfluenza Bovina (PI-3):

- **Patógeno:** Virus Parainfluenza Bovino tipo 3 (PI-3)
- **Descripción:** Enfermedad viral que afecta a bovinos de todas las edades. Provoca problemas respiratorios, especialmente en terneros.

10. Rotavirus:

- **Patógeno:** Rotavirus bovino
- **Descripción:** Enfermedad viral que afecta a terneros jóvenes. Provoca diarrea severa, deshidratación y muerte en algunos casos.

Vacunas para Ovinos:

1. Fiebre Aftosa:

- **Patógeno:** Virus de la Fiebre Aftosa (FAV)
- **Descripción:** Enfermedad viral altamente contagiosa que afecta a bovinos, ovinos y porcinos. Se caracteriza por fiebre, vesículas en la boca y patas, y dificultad para caminar.

2. Brucelosis:

- **Patógeno:** Bacteria Brucella ovis
- **Descripción:** Enfermedad bacteriana que afecta el sistema reproductivo de ovinos. Provoca abortos, infertilidad y nacimiento de corderos débiles.

3. Carbón:

- **Patógeno:** Bacteria Clostridium chauvoei
- **Descripción:** Enfermedad bacteriana grave que afecta a bovinos, ovinos y otros animales. Se caracteriza por muerte súbita, hinchazón y sangre en la nariz y boca.

4. Manquera:

- **Patógeno:** Virus de la Manquera (BVDV)
- **Descripción:** Enfermedad viral que afecta las patas de bovinos y ovinos. Provoca cojera, úlceras en las patas y disminución de la producción de lana.

5. Rabia:

- **Patógeno:** Virus de la Rabia
- **Descripción:** Enfermedad viral mortal que afecta a todos los mamíferos, incluyendo bovinos y ovinos. Se caracteriza por síntomas neurológicos como agresividad, parálisis y

7.2 TAREAS

Necesito realizar una precarga de 5 Tareas de campo para un sistema, las tareas tienen los siguientes atributos: descripción, fecha pactada para su realización, si fue completada o no, un comentario en que de que este completada

1. Inspeccionar el sistema de alarma contra incendios en el edificio principal.

(Inspect the fire alarm system in the main building.)

- Fecha programada: 2024-04-19
- Completado: Falso
- Comentario de finalización: Ninguno

2. Verificar el funcionamiento de las luces de emergencia en todas las salidas.

(Verify the operation of emergency lights at all exits.)

- Fecha programada: 2024-04-23
- Completado: Falso
- Comentario de finalización: Ninguno

3. Realizar pruebas de fugas de gas en todas las áreas de almacenamiento.

(Conduct gas leak tests in all storage areas.)

- Fecha programada: 2024-04-26
- Completado: Falso
- Comentario de finalización: Ninguno

4. Calibrar los manómetros de presión en las calderas. (Calibrate pressure gauges on boilers.)

- Fecha programada: 2024-04-29
- Completado: Falso
- Comentario de finalización: Ninguno

5. Inspeccionar el estado de las barandas y pasamanos en las escaleras. (Inspect the condition of handrails and banisters on stairs.)

- Fecha programada: 2024-05-02
- Completado: Falso
- Comentario de finalización: Ninguno

6. Revisar los extintores de incendios para garantizar su funcionamiento adecuado. (Check fire extinguishers to ensure proper operation.)

- Fecha programada: 2024-05-05
- Completado: Falso
- Comentario de finalización: Ninguno

7. **Verificar que las señales de seguridad y emergencia estén visibles y en buen estado.** (Verify that safety and emergency signs are visible and in good condition.)

- Fecha programada: 2024-05-08
- Completado: Falso
- Comentario de finalización: Ninguno

8. **Realizar una inspección general de las instalaciones en busca de posibles peligros.** (Conduct a general inspection of the premises for potential hazards.)

- Fecha programada: 2024-05-11
- Completado: Falso
- Comentario de finalización: Ninguno

9. **Documentar los hallazgos de las inspecciones y tomar las medidas correctivas necesarias.** (Document inspection findings and take necessary corrective actions.)

- Fecha programada: 2024-05-14
- Completado: Falso
- Comentario de finalización: Ninguno

10. Capacitar al personal sobre los procedimientos de seguridad en caso de emergencias. (Train personnel on emergency safety procedures.)

- Fecha programada: 2024-05-17
- Completado: Falso
- Comentario de finalización: La capacitación se realizará durante la próxima reunión de seguridad.

11. Realizar simulacros de incendio y evacuación para evaluar la preparación del personal. (Conduct fire drills and evacuations to assess staff preparedness.)

- Fecha programada: 2024-05-20
- Completado: Falso
- Comentario de finalización: Los simulacros se realizarán de manera periódica.

12. Mantener registros actualizados de las tareas de mantenimiento y seguridad realizadas. (Maintain up-to-date records of maintenance and safety tasks performed.)

- Fecha programada: En curso
- Completado: Verdadero
- Comentario de finalización: Los registros se mantienen a diario.

13. Comunicar los riesgos de seguridad y los procedimientos de emergencia a todo el personal. (Communicate safety risks and emergency procedures to all personnel.)

- Fecha programada: 2024-05-24
- Completado: Falso
- Comentario de finalización: La comunicación se realizará mediante carteles informativos y reuniones de seguridad.

14. Investigar los incidentes de seguridad y tomar medidas para prevenir su recurrencia. (Investigate safety incidents and take steps to prevent their recurrence.)

- Fecha programada: Según sea necesario
- Completado: Falso
- Comentario de finalización: Las investigaciones se realizarán de manera inmediata tras la notificación de incidentes.

15. Implementar programas de mejora continua para garantizar la seguridad en el lugar de trabajo. (Implement continuous

7.3 PEONES

generame los siguientes datos para precargar 15 empleados en un sistema, cada empleado debe tener: un email, una contraseña, nombre, fecha de ingreso a la empresa, si vive en la empresa(bool) y una lista de tareas vacías para comenzar

Email	Password	Name	Hire Date	Lives on Premises
john.smith@example.com	password123	John Smith	2023-10-15	False
jane.doe@example.com	password456	Jane Doe	2022-12-01	True
peter.jones@example.com	password789	Peter Jones	2024-01-04	False
mary.williams@example.com	password012	Mary Williams	2023-03-22	True
david.miller@example.com	password345	David Miller	2022-05-18	False
sarah.taylor@example.com	password678	Sarah Taylor	2024-02-11	True
michael.brown@example.com	password901	Michael Brown	2023-04-07	False

michael.brown@example.com	password901	Michael Brown	2023-04-07	False
laura.davis@example.com	password234	Laura Davis	2022-06-24	True
richard.thomas@example.com	password567	Richard Thomas	2024-03-09	False

7.4 BOVINOS

lista de 30 animales vacunos con los siguientes datos: número de caravana de 8 dígitos alfanumerico y unico, sexo, raza, fecha de nacimiento, costo de adquisicion, peso actual, si es raza pura o combinacion(bool)

Mostrar versiones 

Sure, here is a list of 30 fictional cattle with the following data:

Caravan Number	Sex	Breed	Date of Birth	Acquisition Cost	Current Weight (kg)	Purebred
1234567A	Male	Angus	2022-01-01	\$1,500	450	True
7890123B	Female	Hereford	2021-07-04	\$1,200	380	True
4567890C	Male	Holstein	2023-05-12	\$1,800	520	False
1234567D	Female	Shorthorn	2020-11-23	\$1,000	300	True

7890123E	Male	Limousin	2022-03-09	\$1,600	430	False
4567890F	Female	Charolais	2021-09-16	\$1,300	360	True
1234567G	Male	Gelbvieh	2023-02-08	\$1,700	480	False
7890123H	Female	Maine Anjou	2020-10-15	\$1,100	320	True
4567890I	Male	South Devon	2022-04-11	\$1,550	420	False
1234567J	Female	Red Angus	2021-08-18	\$1,250	350	True
7890123K	Male	Wagyu	2023-03-10	\$2,000	550	False
4567890L	Female	Belted Galloway	2020-11-17	\$1,150	330	True

1234567M	Male	Texas Longhorn	2022-05-13	\$1,650	470	False
7890123N	Female	Brahman	2021-09-20	\$1,350	370	True
4567890O	Male	Murray Grey	2023-04-12	\$1,850	530	False
1234567P	Female	Highland	2020-12-19	\$1,200	340	True
7890123Q	Male	Chianina	2022-06-14	\$1,700	490	False
4567890R	Female	Simmental	2021-10-21	\$1,400	380	True
1234567S	Male	Sussex	2023-05-15	\$1,900	540	False
7890123T	Female	Devon	2020-01-02	\$1,050	310	True

Caravan Number	Sex	Breed	Date of Birth	Acquisition Cost	Current Weight (kg)	Purebred
1234567A	Male	Angus	2022-01-01	\$1,500	450	TRUE
7890123B	Female	Hereford	2021-07-04	\$1,200	380	TRUE
4567890C	Male	Holstein	2023-05-12	\$1,800	520	FALSE
1234567D	Female	Shorthorn	2020-11-23	\$1,000	300	TRUE
7890123E	Male	Limousin	2022-03-09	\$1,600	430	FALSE
4567890F	Female	Charolais	2021-09-16	\$1,300	360	TRUE
1234567G	Male	Gelbvieh	2023-02-08	\$1,700	480	FALSE
7890123H	Female	Maine Anjou	2020-10-15	\$1,100	320	TRUE
4567890I	Male	South Devon	2022-04-11	\$1,550	420	FALSE
1234567J	Female	Red Angus	2021-08-18	\$1,250	350	TRUE
7890123K	Male	Wagyu	2023-03-10	\$2,000	550	FALSE
4567890L	Female	Belted Galloway	2020-11-17	\$1,150	330	TRUE
1234567M	Male	Texas Longhorn	2022-05-13	\$1,650	470	FALSE
7890123N	Female	Brahman	2021-09-20	\$1,350	370	TRUE
4567890O	Male	Murray Grey	2023-04-12	\$1,850	530	FALSE
1234567P	Female	Highland	2020-12-19	\$1,200	340	TRUE
7890123Q	Male	Chianina	2022-06-14	\$1,700	490	FALSE
4567890R	Female	Simmental	2021-10-21	\$1,400	380	TRUE
1234567S	Male	Sussex	2023-05-15	\$1,900	540	FALSE
7890123T	Female	Devon	2020-01-02	\$1,050	310	TRUE
4567890U	Male	Galloway	2022-07-16	\$1,500	410	FALSE
1234567V	Female	Brangus	2021-11-23	\$1,200	340	TRUE
7890123W	Male	Beefmaster	2023-06-17	\$		

7.5 OVINOS

Lista de 30 ovinos que contengan los siguientes datos: peso estimado de lana, precio por kilogramo de lana, numero de carava de 8 digitos alfanumerico y unico, sexo, raza, costo de adquisicion, peso actual, si es de raza puro o no(bool).

Mostrar versiones ▾

Lista de 30 Ovinos

Peso Estimado de Lana (kg)	Precio Lana (Kg)	Número de Caravana	Sexo	Raza	Costo de Adquisición	Peso Actual (kg)
2.5	35	A2345678	Hembra	Merino	150	45
3.2	40	B9876543	Macho	Corriedale	220	62
1.8	32	C1234567	Hembra	Hampshire Down	120	38
2.7	38	D8765432	Macho	Suffolk	180	55
2.4	36	E2345678	Hembra	Texel	165	42

3.1	42	F9876543	Macho	Île-de-France	210	60
1.9	34	G1234567	Hembra	Romney Marsh	130	36
2.8	39	H8765432	Macho	Lincoln	195	58
2.5	37	I2345678	Hembra	Border Leicester	170	44
3.2	43	J9876543	Macho	Cheviot	230	65
1.8	33	K1234567	Hembra	Bluefaced Leicester	125	35
2.7	39	L8765432	Macho	Swaledale	185	57
2.4	37	M2345678	Hembra	Dalesbred	160	43
3.1	42	N9876543	Macho	Exmoor Horn	215	61

1.9	35	O1234567	Hembra	Scottish Blackface	135	37
2.8	40	P8765432	Macho	Clun Forest	190	59
2.5	38	Q2345678	Hembra	Soay	175	45
3.2	44	R9876543	Macho	Hebridean	235	66
1.8	34	S1234567	Hembra	Shetland	120	34
2.7	39	T8765432	Macho	Arapawa	180	56
2.4	37	U2345678	Hembra	Columbia	165	43
3.1	43	V9876543	Macho	Rambouillet	220	64
1.9	35	W1234567	Hembra	Merino Argentino	130	36
2.8	40	X8765432	Macho	Corriedale Australiano	195	58
2.5	38	Y2345678	Hembra	Romney Marsh Neozelandés	170	45
3.2	44	Z9876543	Macho	Tex		

Peso Estimado de Lana (kg)	Precio Lana (Kg)	Número de C	Sexo	Raza	Costo de Ad	Peso Actual	Raza Pura
2.5	35	A2345678	Hembra	Merino	150	45	Si
3.2	40	B9876543	Macho	Corriedale	220	62	Si
1.8	32	C1234567	Hembra	Hampshire Down	120	38	No
2.7	38	D8765432	Macho	Suffolk	180	55	Si
2.4	36	E2345678	Hembra	Texel	165	42	No
3.1	42	F9876543	Macho	Île-de-France	210	60	Si
1.9	34	G1234567	Hembra	Romney Marsh	130	36	No
2.8	39	H8765432	Macho	Lincoln	195	58	Si
2.5	37	I2345678	Hembra	Border Leicester	170	44	No
3.2	43	J9876543	Macho	Cheviot	230	65	Si
1.8	33	K1234567	Hembra	Bluefaced Leicester	125	35	No
2.7	39	L8765432	Macho	Swaledale	185	57	Si
2.4	37	M2345678	Hembra	Dalesbred	160	43	No
3.1	42	N9876543	Macho	Exmoor Horn	215	61	Si
1.9	35	O1234567	Hembra	Scottish Blackface	135	37	No
2.8	40	P8765432	Macho	Clun Forest	190	59	Si
2.5	38	Q2345678	Hembra	Soay	175	45	No
3.2	44	R9876543	Macho	Hebridean	235	66	Si
1.8	34	S1234567	Hembra	Shetland	120	34	No
2.7	39	T8765432	Macho	Arapawa	180	56	Si
2.4	37	U2345678	Hembra	Columbia	165	43	No
3.1	43	V9876543	Macho	Rambouillet	220	64	Si
1.9	35	W1234567	Hembra	Merino Argentino	130	36	Si
2.8	40	X8765432	Macho	Corriedale Australiano	195	58	Si
2.5	38	Y2345678	Hembra	Romney Marsh Neozelandés	170	45	No
3.2	44	Z9876543	Macho	Tex			

8 PUBLICACIÓN EN SOMEE

URL: <http://estanciajye.somee.com/>