Universidad ORT Uruguay

Facultad de Ingeniería

Escuela de tecnología

Obligatorio 1 Programación 3

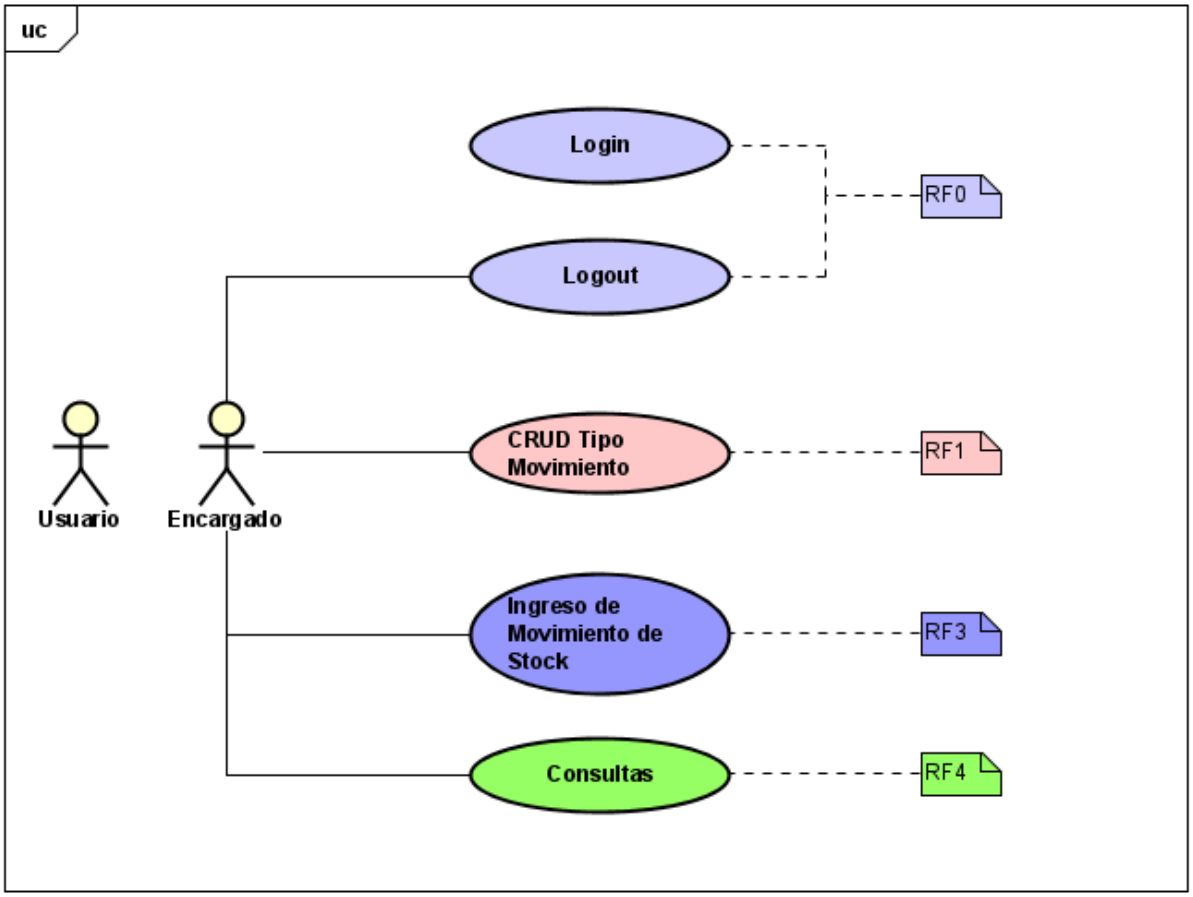
Agustina Sánchez Montoro - **200383**

Jonathan Lima - **318410**

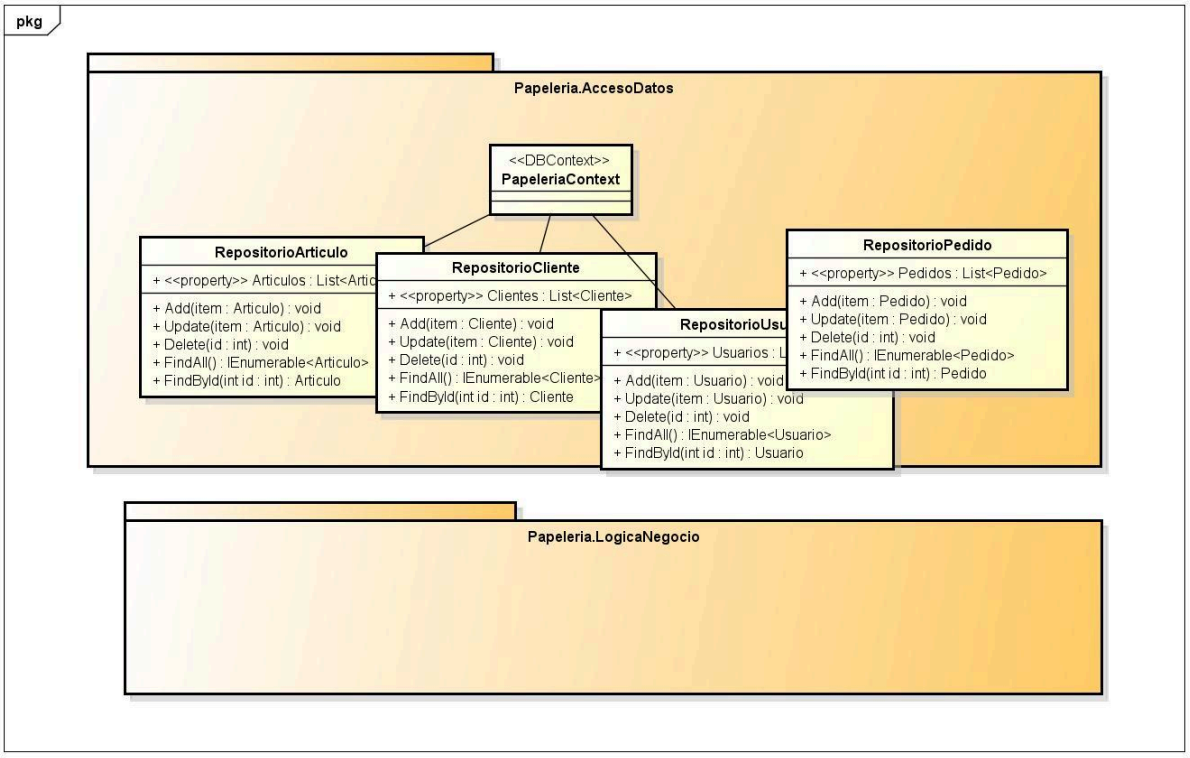
Grupo: N3D
Docente: Lucas López
Mayo 2024
Analista Programador y Tecnologías de la Información

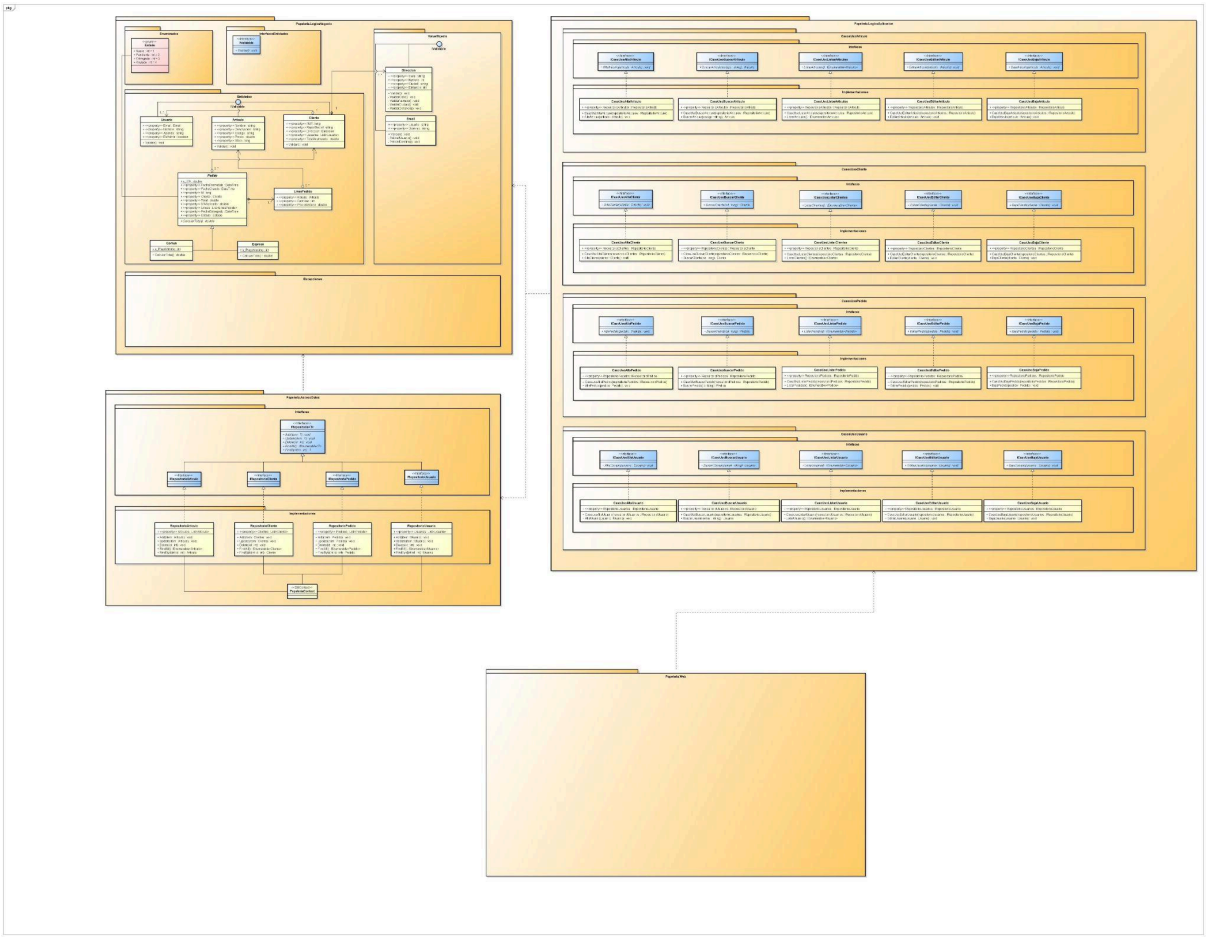
Indice

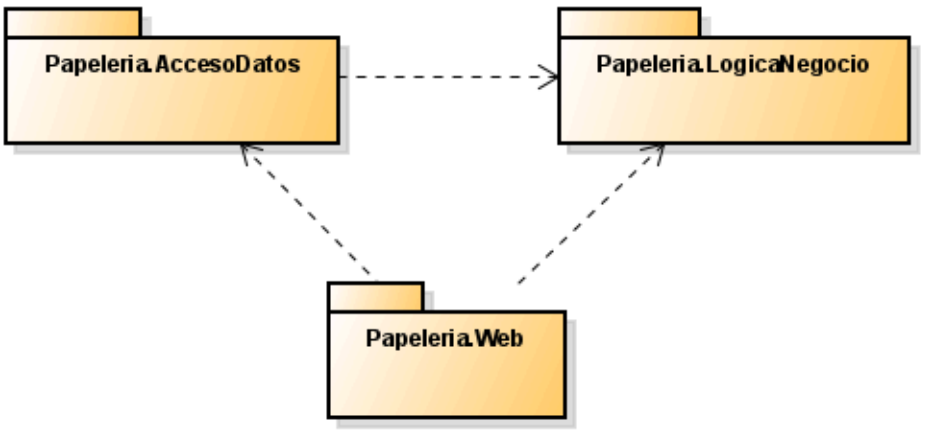
Casos de uso ASTAH:



UML:


pkg
Papeleria.AccesoDatos
<<DBContext>>
PapeleriaContext
RepositorioArticulo
+ <<property>> Articulos : List<Artic
+ Add(item : Articulo) : void
+ Update(item : Articulo) : void
+ Delete(id : int) : void
+ FindAll() : IEnumerable<Articulo>
+ FindById(int id : int) : Articulo
RepositorioCliente
+ <<property>> Clientes : List<Cliente>
+ Add(item : Cliente) : void
+ Update(item : Cliente) : void
+ Delete(id : int) : void
+ FindAll() : IEnumerable<Cliente>
+ FindById(int id : int) : Cliente
RepositorioUsu
+ <<property>> Usuarios : L
+ Add(item : Usuario) : void
+ Update(item : Usuario) : void
+ Delete(id : int) : void
+ FindAll() : IEnumerable<Usuario>
+ FindById(int id : int) : Usuario
RepositorioPedido
+ <<property>> Pedidos : List<Pedido>
+ Add(item : Pedido) : void
+ Update(item : Pedido) : void
+ Delete(id : int) : void
+ FindAll() : IEnumerable<Pedido>
+ FindById(int id : int) : Pedido
Papeleria.LogicaNegocio




Papeleria.AccesoDatos
Papeleria.LogicaNegocio
Papeleria.Web

Código fuente:

Acceso Datos:

Repositorio Artículos:

```
namespace AccesoDatos.Implementaciones.EntityFramework
{
    public class RepositorioArticulos : IRepositorioArticulos
    {
        private PapeleriaContext _papeleriaContext;

        public RepositorioArticulos()
        {
            _papeleriaContext = new PapeleriaContext();
        }

        #region CRUD Operations
        public void Add(Articulo articuloNuevo)
        {
            try
            {
                _papeleriaContext.Articulos.Add(articuloNuevo);
                _papeleriaContext.SaveChanges();
            }
            catch (ArticuloNoEncontradoException)
            {
                throw;
            }
            catch (DbUpdateException)
            {
                throw;
            }
            catch (Exception ex)
            {
                throw new Exception($"Error desconocido: {ex.Message} (Trace:
{ex.StackTrace})");
            }
        }

        public Articulo GetById(int id)
        {
            if (!_papeleriaContext.Articulos.Any())
                throw new DataBaseSetException("La tabla Articulos esta vacia");
```

```
            try
            {
                Articulo? articuloEncontrado =
_papeleriaContext.Articulos.AsNoTracking().FirstOrDefault(articulo => articulo.Id == id);
                return articuloEncontrado ?? throw new ArticuloNoEncontradoException($"No se
encontro el articulo de Id: {id}");
            }
            catch (ArticuloNoEncontradoException)
            {
                throw;
            }
            catch (Exception ex)
            {
                throw new Exception($"Error desconocido: {ex.Message} (Trace:
{ex.StackTrace})");
            }
        }

        public Articulo RetrieveById(int id)
        {
            if (!_papeleriaContext.Articulos.Any())
                throw new DataBaseSetException("La tabla Articulos esta vacia");

            try
            {
                Articulo? articuloEncontrado = _papeleriaContext.Articulos.FirstOrDefault(articulo
=> articulo.Id == id);
                return articuloEncontrado ?? throw new ArticuloNoEncontradoException($"No se
encontro el articulo de Id: {id}");
            }
            catch (ArticuloNoEncontradoException)
            {
                throw;
            }
            catch (Exception ex)
            {
                throw new Exception($"Error desconocido: {ex.Message} (Trace:
{ex.StackTrace})");
            }
        }

        public IEnumerable<Articulo> GetAll()
        {
            return _papeleriaContext.Articulos.ToList();
        }

        public void Update(int id, Articulo articuloEditado)
        {
```

```
            try
            {
                _papeleriaContext.Articulos.Update(articuloEditado);
                _papeleriaContext.SaveChanges();
            }
            catch (ArticuloNoValidoException)
            {
                throw;
            }
            catch (Exception ex)
            {
                throw new Exception($"Error desconocido: {ex.Message} (Trace:
{ex.StackTrace})");
            }
        }

        public void Remove(int id)
        {
            try
            {
                Articulo articuloParaBorrar = GetById(id);
                _papeleriaContext.Articulos.Remove(articuloParaBorrar);
                _papeleriaContext.SaveChanges();
            }
            catch (ArticuloNoEncontradoException)
            {
                throw;
            }
            catch (Exception ex)
            {
                throw new Exception($"Error desconocido: {ex.Message} (Trace:
{ex.StackTrace})");
            }
        }
        #endregion

        #region DML Methods
        public IEnumerable<Articulo> ListadoAscendente()
        {
            return _papeleriaContext.Articulos.OrderBy(articulo => articulo.Nombre).ToList();
        }
        //Consulta B con paginación
        public IEnumerable<Articulo> GetArticulosConMovimientos(DateTime fecha1,
DateTime fecha2, int pageNumber, int pageSize)
        {
            return _papeleriaContext.MovimientoStock.Where(m => m.Fecha >= fecha1 &&
m.Fecha <= fecha2)
                                        .Select(m => m.Articulo)
```

```
                                        .Distinct()
                                        .OrderBy(a => a.Id)
                                        .Skip((pageNumber - 1) * pageSize)
                                        .Take(pageSize)
                                        .ToList();
        }

        #endregion
    }
}
```

Repositorio Movimiento de Stock:

```
namespace AccesoDatos.Implementaciones.EntityFramework
{
    public class RepositorioMovimientoDeStock : IRepositorioMovimientoDeStock
    {
        private readonly PapeleriaContext _papeleriaContext;

        public RepositorioMovimientoDeStock()
        {
            _papeleriaContext = new PapeleriaContext();
        }
        public void Add(MovimientoStock nuevoMovimientoStock)
        {
            try
            {
                nuevoMovimientoStock.EsValido();
                _papeleriaContext.Entry(nuevoMovimientoStock.Articulo).State =
EntityState.Unchanged;
                _papeleriaContext.Entry(nuevoMovimientoStock.Usuario).State =
EntityState.Unchanged;
                _papeleriaContext.Entry(nuevoMovimientoStock.TipoMovimiento).State =
EntityState.Unchanged;
                _papeleriaContext.MovimientoStock.Add(nuevoMovimientoStock);
                _papeleriaContext.SaveChanges();

            }
            catch (Exception ex)
            {
                throw new Exception(ex.Message);
            }
        }

        public IEnumerable<MovimientoStock> GetAll()
        {
```

```
        return _papeleriaContext.MovimientoStock.Include(m => m.TipoMovimiento)
                                        .Include(m => m.Usuario)
                                        .Include(m => m.Articulo)
                                        .ToList();
    }

    ///Consulta A con paginación
    public IEnumerable<MovimientoStock> GetMovimientos(int articuloId, int
tipoMovimientoId, int pageNumber, int pageSize)
    {
        IEnumerable<MovimientoStock> lista = _papeleriaContext.MovimientoStock
                        .Include(m => m.Articulo)
                        .Include(m => m.Usuario)
                        .Include(m => m.TipoMovimiento)
                        .Where(m => m.Articulo.Id == articuloId && m.TipoMovimiento.Id ==
tipoMovimientoId)
                        .OrderByDescending(m => m.Fecha)
                        .ThenBy(m => m.Cantidad)
                        .Skip((pageNumber - 1) * pageSize)
                        .Take(pageSize)
                        .ToList();
        return lista;

    }

    public IEnumerable<Object> GetResumenMovimientos()
    {
        var lista = _papeleriaContext.MovimientoStock
                                    .GroupBy(m => m.Fecha.Year)
                                    .Select(group => new
                                    {
                                        Anio = group.Key,
                                        TipoMovimiento = group.GroupBy(tm =>
tm.TipoMovimiento)
                                        .Select(tm => new
                                        {
                                            NombreTipo = tm.Key,
                                            Cantidad = tm.Sum(movimiento => movimiento.Cantidad)
                                        }),
                                        TotalAnio = group.Sum(m => m.Cantidad)
                                    })
                                    .OrderBy(group => group.Anio)
                                    .ToList();


        return lista;
    }
```

```
        #region Not Needed
        public MovimientoStock GetById(int id)
        {
            return _papeleriaContext.MovimientoStock.Find(id);
        }

        public void Remove(int id)
        {
            throw new NotImplementedException();
        }

        public MovimientoStock RetrieveById(int id)
        {
            throw new NotImplementedException();
        }

        public void Update(int id, MovimientoStock obj)
        {
            throw new NotImplementedException();
        }
        #endregion
    }
}
```

Repositorio Settings:

```
namespace AccesoDatos.Implementaciones.EntityFramework
{
    public class RepositorioSettings : IRepositorioSettings
    {
        private PapeleriaContext _papeleriaContext;

        public RepositorioSettings()
        {
            _papeleriaContext = new PapeleriaContext();
        }
        public double GetValueByName(string name)
        {
            return _papeleriaContext.Settings.Where(setting => setting.Name ==
name).FirstOrDefault().Value;
        }
    }
}
```

Repositorio Tipo de Movimiento:

```
namespace AccesoDatos.Implementaciones.EntityFramework
{
    public class RepositorioTipoMovimiento : IRepositorioTipoMovimiento
    {
        private readonly PapeleriaContext _papeleriaContext;

        public RepositorioTipoMovimiento()
        {
            _papeleriaContext = new PapeleriaContext();
        }

        #region CRUD
        public IEnumerable<TipoMovimiento> GetAll()
        {
            return _papeleriaContext.TipoMovimientos.ToList();
        }

        public TipoMovimiento GetById(int id)
        {
            try
            {
                TipoMovimiento? tipoEncontrado = _papeleriaContext.TipoMovimientos
                    .AsNoTracking()
                    .FirstOrDefault(t => t.Id == id);
                return tipoEncontrado ?? throw new Exception($"No se encontro el tipo
movimiento de ID: {id}");

            }
            catch (Exception ex)
            {
                throw new Exception($"Error desconocido: {ex.Message}");
            }
        }
        public void Add(TipoMovimiento nuevoTipoMovimiento)
        {
            try
            {
                nuevoTipoMovimiento.EsValido();
                _papeleriaContext.TipoMovimientos.Add(nuevoTipoMovimiento);
                _papeleriaContext.SaveChanges();

            }
            catch (Exception ex)
            {
                throw new Exception($"Error desconocido: {ex.Message}");
            }
```

```
    }

    public void Update(int id, TipoMovimiento tipoMovimientoEditado)
    {
        try
        {
            tipoMovimientoEditado.Id = id;
            tipoMovimientoEditado.EsValido();
            _papeleriaContext.TipoMovimientos.Update(tipoMovimientoEditado);
            _papeleriaContext.SaveChanges();
        }
        catch (Exception ex)
        {
            throw new Exception($"Error desconocido: {ex.Message}");
        }

    }

    public void Remove(int id)
    {
        try
        {
            //With AsNoTracking
            TipoMovimiento tipoABorrar = GetById(id);
            _papeleriaContext.TipoMovimientos.Remove(tipoABorrar);
            _papeleriaContext.SaveChanges();
        }
        catch (Exception ex)
        {
            throw new Exception($"Error desconocido: {ex.Message}");
        }
    }
    #endregion

    #region Methods

    public IEnumerable<TipoMovimiento> GetByTipoMovimiento(string tipoMovimiento)
    {
        IEnumerable<TipoMovimiento> tipoMovimientos =
_papeleriaContext.TipoMovimientos
            .Where(t => t.Nombre == tipoMovimiento)
            .ToList();
        return tipoMovimientos;
    }
```

```
        #endregion

        #region Not needed

        public TipoMovimiento RetrieveById(int id)
        {
            throw new NotImplementedException();
        }


        #endregion
    }
}
```

Repositorio Usuarios:

```
namespace AccesoDatos.Implementaciones.EntityFramework
{
    public class RepositorioUsuarios : IRepositorioUsuarios
    {
        private PapeleriaContext _papeleriaContext;

        public RepositorioUsuarios()
        {
            //Inyeccion de dependencia
            _papeleriaContext = new PapeleriaContext();
        }

        #region CRUD Operations
        public Usuario GetById(int id)
        {
            Usuario? usuarioEncontrado =
_papeleriaContext.Usuarios.AsNoTracking().FirstOrDefault(usuario => usuario.Id == id);

            return usuarioEncontrado ?? throw new UsuarioNoEncontradoException($"No se
pudo encontrar el usuario de ID: {id}");
        }

        public Usuario RetrieveById(int id)
        {

            Usuario? usuarioEncontrado =
_papeleriaContext.Usuarios.AsNoTracking().FirstOrDefault(usuario => usuario.Id == id);

            return usuarioEncontrado ?? throw new UsuarioNoEncontradoException($"No se
pudo encontrar el usuario de ID: {id}");
        }
```

```
        public IEnumerable<Usuario> GetAll()
        {
            return _papeleriaContext.Usuarios.ToList();
        }

        public void Add(Usuario usuarioNuevo)
        {
            try
            {
                _papeleriaContext.Usuarios.Add(usuarioNuevo);
                _papeleriaContext.SaveChanges();
            }
            catch (DbUpdateException)
            {
                throw;
            }
            catch (Exception ex)
            {
                throw new Exception($"Error desconocido: {ex.Message} (Trace:
{ex.StackTrace})");
            }
        }

        public void Update(int id, Usuario usuarioEditado)
        {
            try
            {
                _papeleriaContext.Usuarios.Update(usuarioEditado);
                _papeleriaContext.SaveChanges();
            }
            catch (UsuarioNoValidoException)
            {
                throw;
            }
            catch (Exception ex)
            {
                throw new Exception($"Error desconocido: {ex.Message} (Trace:
{ex.StackTrace})");
            }
        }

        public void Remove(int id)
        {
            try
            {
                Usuario aBorrar = GetById(id);
                _papeleriaContext.Usuarios.Remove(aBorrar);
```

```
            _papeleriaContext.SaveChanges();
        }
        catch (UsuarioNoEncontradoException)
        {
            throw;
        }
        catch (Exception)
        {
            throw;
        }
    }

    #endregion

    #region DML Methods
    public Usuario ObtenerUsuarioPorEmail(string email)
    {
        Usuario? usuarioEncontrado = _papeleriaContext.Usuarios.FirstOrDefault(usuario =>
usuario.Email == email);
        return usuarioEncontrado ?? throw new
UsuarioNoEncontradoException("Credenciales incorrectas. Revise el mail y/o la
contraseña");
    }
    #endregion
  }
}
```

IRepositorio CRUD

```
  public interface IRepositorioCRUD<T> where T : class
  {
    void Add(T obj);
    T GetById(int id);
    T RetrieveById(int id);
    IEnumerable<T> GetAll();
    void Update(int id, T obj);
    void Remove(int id);
  }
}
```

IRepositorio Artículos

```
namespace AccesoDatos.Interfaces
{
  public interface IRepositorioArticulos : IRepositorioCRUD<Articulo>
  {
    public IEnumerable<Articulo> ListadoAscendente();
```

```
        public IEnumerable<Articulo> GetArticulosConMovimientos(DateTime fecha1,
DateTime fecha2, int pageNumber, int pageSize);
    }
}
```

IRepositorio Movimiento de Stock

```
namespace AccesoDatos.Interfaces
{
    public interface IRepositorioMovimientoDeStock :IRepositorioCRUD<MovimientoStock>
    {
        public IEnumerable<MovimientoStock> GetMovimientos(int articuloId, int
tipoMovimientoId, int pageNumber, int pageSize);

        public IEnumerable<Object> GetResumenMovimientos();
    }
}
```

IRepositorio Settings

```
namespace AccesoDatos.Interfaces
{
    public interface IRepositorioSettings
    {
        public double GetValueByName(string name);
    }
}
```

IRepositorio Tipo de Movimiento

```
namespace AccesoDatos.Interfaces
{
    public interface IRepositorioTipoMovimiento : IRepositorioCRUD<TipoMovimiento>
    {
        public IEnumerable<TipoMovimiento> GetByTipoMovimiento (string tipoMovimiento);
    }

}
```

IRepositorio Usuarios

```
namespace AccesoDatos.Interfaces
{
    public interface IRepositorioUsuarios : IRepositorioCRUD<Usuario>
    {
        public Usuario ObtenerUsuarioPorEmail(string email);
    }
}
```

Lógica Aplicación:

```
namespace LogicaAplicacion.CasosUso.CasosUsoArticulos.Implementaciones
{
    public class CasoUsoGetArticulosConMovimientos :
ICasoUsoGetArticulosConMovimientos
    {
        private IRepositorioArticulos RepositorioArticulos { get; init; }
        private IRepositorioSettings RepositorioSettings { get; init; }

        public CasoUsoGetArticulosConMovimientos(
            IRepositorioArticulos repositorioArticulos,
            IRepositorioSettings repositorioSettings)
        {
            RepositorioArticulos = repositorioArticulos;
            RepositorioSettings = repositorioSettings;
        }

        public IEnumerable<ArticulosListadoDTO> GetArticulosConMovimientos(DateTime
fecha1, DateTime fecha2, int pageNumber)
        {
            int size = int.Parse(RepositorioSettings.GetValueByName("PageSize") + "");
            return RepositorioArticulos.GetArticulosConMovimientos(fecha1, fecha2,
pageNumber, size)
                                .Select(a => MapperArticulo.ToDTO(a));
        }
    }
}

namespace LogicaAplicacion.CasosUso.CasosUsoArticulos.Interfaces
{
    public interface ICasoUsoGetArticulosConMovimientos
    {
        public IEnumerable<ArticulosListadoDTO> GetArticulosConMovimientos(DateTime
fecha1, DateTime fecha2, int pageNumber);
```

```
    }
}

namespace LogicaAplicacion.CasosUso.CasosUsoLogin.Implementaciones
{
    public class CasoUsoLoginUsuario : ICasoUsoLoginUsuario
    {
        public IRepositorioUsuarios RepositorioUsuarios { get; init; }

        public CasoUsoLoginUsuario(IRepositorioUsuarios repositorioUsuario)
        {
            RepositorioUsuarios = repositorioUsuario;
        }
        public Usuario Ejecutar(string email, string contraseña)
        {
            Usuario buscado = RepositorioUsuarios.ObtenerUsuarioPorEmail(email);

            return buscado.Contrasenia == Usuario.Encriptar(contraseña) ? buscado : null;
        }


    }
}

namespace LogicaAplicacion.CasosUso.CasosUsoLogin.Interfaces
{
    public interface ICasoUsoLoginUsuario
    {
        public Usuario Ejecutar(string email, string contraseña);
    }
}

namespace LogicaAplicacion.CasosUso.CasosUsoMovimientosDeStock.Implementaciones
{
    public class CasoUsoAltaMovimientoStock : ICasoUsoAltaMovimientoStock
    {
        public IRepositorioMovimientoDeStock RepositorioMovimientoStock { get; init; }
        private IRepositorioSettings RepositorioSettings { get; init; }
        public CasoUsoAltaMovimientoStock(
            IRepositorioMovimientoDeStock repositorioMovimientoDeStock,
            IRepositorioSettings repositorioSettings)
        {
            // Inyeccion de dependencia
            RepositorioMovimientoStock = repositorioMovimientoDeStock;
            RepositorioSettings = repositorioSettings;
        }
        public void AltaMovimientoStock(MovimientoDeStockDTO movimientosDeStockDTO)
        {
```

```
            int tope = int.Parse(RepositorioSettings.GetValueByName("Tope") + "");
            if (movimientosDeStockDTO.Cantidad <= tope)
            {

RepositorioMovimientoStock.Add(MapperMovimientoStock.FromDTO(movimientosDeStock
DTO));
            }
            else
            {
                throw new Exception("No se puede superar el tope " + tope);
            }

        }

    }
}

namespace LogicaAplicacion.CasosUso.CasosUsoMovimientosDeStock.Implementaciones
{
    public class CasoUsoFindByIdMovimientoStock : ICasoUsoFindByIdMovimientoStock
    {
        public IRepositorioMovimientoDeStock RepositorioMovimientoDeStock { get; init; }

        public CasoUsoFindByIdMovimientoStock(IRepositorioMovimientoDeStock
repositorioMovimientoDeStock)
        {
            // Inyeccion de dependencia
            RepositorioMovimientoDeStock = repositorioMovimientoDeStock;
        }
        public MovimientoDeStockDTO FindById(int id)
        {
            MovimientoStock movimiento = RepositorioMovimientoDeStock.GetById(id);
            return MapperMovimientoStock.ToDTO(movimiento);
        }
    }
}
namespace LogicaAplicacion.CasosUso.CasosUsoMovimientosDeStock.Implementaciones
{
    public class CasoUsoGetMovimientos : ICasoUsoGetMovimientos
    {
        private IRepositorioMovimientoDeStock _repositorioMovimientoStock { get; init; }
        private IRepositorioSettings _settings {  get; set; }

        public CasoUsoGetMovimientos(
            IRepositorioMovimientoDeStock repositorioMovimientoDeStock,
            IRepositorioSettings repositorioSettings)
        {
            // Inyeccion de dependencia
```

```
            _repositorioMovimientoStock = repositorioMovimientoDeStock;
            _settings = repositorioSettings;
        }
        public IEnumerable<MovimientoDeStockDTO> GetMovimientos(int articuloId, int
tipoMovimientoId, int pageNumber)
        {
            int size = int.Parse(_settings.GetValueByName("PageSize") + "");
            return _repositorioMovimientoStock.GetMovimientos(articuloId, tipoMovimientoId,
pageNumber, size)
                                        .Select(m => MapperMovimientoStock.ToDTO(m));
        }
    }
}

namespace LogicaAplicacion.CasosUso.CasosUsoMovimientosDeStock.Implementaciones
{
    public class CasoUsoGetResumenMovimientos : ICasoUsoGetResumenMovimientos
    {
        public IRepositorioMovimientoDeStock RepositorioMovimientoStock { get; init; }

        public CasoUsoGetResumenMovimientos(
            IRepositorioMovimientoDeStock repositorioMovimientoDeStock
        )

        {
            // Inyeccion de dependencia
            RepositorioMovimientoStock = repositorioMovimientoDeStock;
        }

        public IEnumerable<Object> GetResumenMovimientos()
        {
            return RepositorioMovimientoStock.GetResumenMovimientos();
        }
    }
}

namespace LogicaAplicacion.CasosUso.CasosUsoMovimientosDeStock.Implementaciones
{
    public class CasoUsoListarMovimientoStock : ICasoUsoListarMovimientoStock
    {
        public IRepositorioMovimientoDeStock RepositorioMovimientoDeStock { get; set; }

        public CasoUsoListarMovimientoStock(IRepositorioMovimientoDeStock
repositorioMovimientoDeStock)
        {
            RepositorioMovimientoDeStock = repositorioMovimientoDeStock;
        }
```

```
        public IEnumerable<MovimientoDeStockDTO> ListarMovimientosDeStock()
        {
            return MapperMovimientoStock.ToDTOList(RepositorioMovimientoDeStock.GetAll());
        }
    }
}


namespace LogicaAplicacion.CasosUso.CasosUsoMovimientosDeStock.Interfaces
{
    public interface ICasoUsoAltaMovimientoStock
    {
        public void AltaMovimientoStock(MovimientoDeStockDTO movimientosDeStockDTO);
    }
}

namespace LogicaAplicacion.CasosUso.CasosUsoMovimientosDeStock.Interfaces
{
    public interface ICasoUsoFindByIdMovimientoStock
    {
        public MovimientoDeStockDTO FindById(int id);
    }
}

namespace LogicaAplicacion.CasosUso.CasosUsoMovimientosDeStock.Interfaces
{
    public interface ICasoUsoGetMovimientos
    {
        public IEnumerable<MovimientoDeStockDTO> GetMovimientos(int articuloId, int
tipoMovimientoId, int pageNumber);
    }
}

namespace LogicaAplicacion.CasosUso.CasosUsoMovimientosDeStock.Interfaces
{
    public interface ICasoUsoGetResumenMovimientos
    {
        public IEnumerable<Object> GetResumenMovimientos();
    }
}

namespace LogicaAplicacion.CasosUso.CasosUsoMovimientosDeStock.Interfaces
{
    public interface ICasoUsoListarMovimientoStock
    {
        public IEnumerable<MovimientoDeStockDTO> ListarMovimientosDeStock();
    }
}
```

```
namespace LogicaAplicacion.CasosUso.CasosUsoTipoMovimiento.Implementaciones
{
    public class CasoUsoAltaTipoMovimiento : ICasoUsoAltaTipoMovimiento
    {
        public IRepositorioTipoMovimiento RepositorioTiposMovimiento { get; init; }

        public CasoUsoAltaTipoMovimiento(IRepositorioTipoMovimiento
repositorioTiposMovimiento)
        {
            // Inyeccion de dependencia
            RepositorioTiposMovimiento = repositorioTiposMovimiento;
        }

        public void AltaTipoMovimiento(TipoMovimientoDTO nuevoTipoMovimientoDTO)
        {

RepositorioTiposMovimiento.Add(MapperTipoMovimiento.FromDTO(nuevoTipoMovimientoD
TO));
        }
    }
}

namespace LogicaAplicacion.CasosUso.CasosUsoTipoMovimiento.Implementaciones
{
    public class CasoUsoBajaTipoMovimiento : ICasoUsoBajaTipoMovimiento
    {
        public IRepositorioTipoMovimiento RepositorioTiposMovimiento { get; init; }

        public CasoUsoBajaTipoMovimiento(IRepositorioTipoMovimiento
repositorioTiposMovimiento)
        {
            // Inyeccion de dependencia
            this.RepositorioTiposMovimiento = repositorioTiposMovimiento;
        }

        public void BajaTipoMovimiento(int idTipoMovimiento)
        {
            RepositorioTiposMovimiento.Remove(idTipoMovimiento);
        }
    }
}

namespace LogicaAplicacion.CasosUso.CasosUsoTipoMovimiento.Implementaciones
{
    public class CasoUsoBuscarTipoMovimiento : ICasoUsoBuscarTipoMovimiento
    {
```

```
        public IRepositorioTipoMovimiento RepositorioTiposMovimiento { get; init; }

        public CasoUsoBuscarTipoMovimiento(IRepositorioTipoMovimiento
repositorioTiposMovimiento)
        {
            // Inyeccion de dependencia
            RepositorioTiposMovimiento = repositorioTiposMovimiento;
        }

        public TipoMovimiento BuscarTipoMovimiento(int id)
        {
            return RepositorioTiposMovimiento.GetById(id);
        }
    }
}
namespace LogicaAplicacion.CasosUso.CasosUsoTipoMovimiento.Implementaciones
{
    public class CasoUsoEditTipoMovimiento : ICasoUsoEditTipoMovimiento
    {
        public IRepositorioTipoMovimiento RepositorioTiposMovimiento { get; init; }

        public CasoUsoEditTipoMovimiento(IRepositorioTipoMovimiento
repositorioTiposMovimiento)
        {
            // Inyeccion de dependencia
            RepositorioTiposMovimiento = repositorioTiposMovimiento;
        }

        public void EditTipoMovimiento(int id, TipoMovimientoDTO editTipoMovimientoDTO)
        {
            RepositorioTiposMovimiento.Update(id,
MapperTipoMovimiento.FromDTO(editTipoMovimientoDTO));
        }
    }
}
namespace LogicaAplicacion.CasosUso.CasosUsoTipoMovimiento.Implementaciones
{
    public class CasoUsoListarTipoMovimiento : ICasoUsoListarTipoMovimiento
    {
        public IRepositorioTipoMovimiento RepositorioTiposMovimiento { get; init; }

        public CasoUsoListarTipoMovimiento(IRepositorioTipoMovimiento
repositorioTiposMovimiento)
        {
            // Inyeccion de dependencia
            RepositorioTiposMovimiento = repositorioTiposMovimiento;
        }
        public IEnumerable<TipoMovimientoDTO> GetAll()
```

```
        {
            return MapperTipoMovimiento.ToDTOList(RepositorioTiposMovimiento.GetAll());
        }
    }
}
namespace LogicaAplicacion.CasosUso.CasosUsoTipoMovimiento.Implementaciones
{
    public class CasoUsoObtenerPorTipoMovimiento : ICasoUsoObtenerPorTipoMovimiento
    {
        public IRepositorioTipoMovimiento RepositorioTiposMovimiento { get; init; }

        public CasoUsoObtenerPorTipoMovimiento(IRepositorioTipoMovimiento
repositorioTiposMovimiento)
        {
            // Inyeccion de dependencia
            RepositorioTiposMovimiento = repositorioTiposMovimiento;
        }
        public IEnumerable<TipoMovimientoDTO> ObtenerPorTipoMovimiento(string
tipoMovimiento)
        {
            return
MapperTipoMovimiento.ToDTOList(RepositorioTiposMovimiento.GetByTipoMovimiento(tipo
Movimiento));
        }
    }
}

namespace LogicaAplicacion.CasosUso.CasosUsoTipoMovimiento.Interfaces
{
    public interface ICasoUsoAltaTipoMovimiento
    {
        public void AltaTipoMovimiento(TipoMovimientoDTO nuevoTipoMovimientoDTO);
    }
}

namespace LogicaAplicacion.CasosUso.CasosUsoTipoMovimiento.Interfaces
{
    public interface ICasoUsoBajaTipoMovimiento
    {
        public void BajaTipoMovimiento(int id);
    }
}

namespace LogicaAplicacion.CasosUso.CasosUsoTipoMovimiento.Interfaces
{
    public interface ICasoUsoBuscarTipoMovimiento
    {
        public TipoMovimiento BuscarTipoMovimiento(int id);
```

```
    }
}

namespace LogicaAplicacion.CasosUso.CasosUsoTipoMovimiento.Interfaces
{
    public interface ICasoUsoEditTipoMovimiento
    {
        public void EditTipoMovimiento(int id, TipoMovimientoDTO editTipoMovimientoDTO);
    }
}


namespace LogicaAplicacion.CasosUso.CasosUsoTipoMovimiento.Interfaces
{
    public interface ICasoUsoListarTipoMovimiento
    {
        public IEnumerable<TipoMovimientoDTO> GetAll();
    }
}

namespace LogicaAplicacion.CasosUso.CasosUsoTipoMovimiento.Interfaces
{
    public interface ICasoUsoObtenerPorTipoMovimiento
    {
        public IEnumerable<TipoMovimientoDTO> ObtenerPorTipoMovimiento(string
tipoMovimiento);
    }
}


namespace LogicaAplicacion.CasosUso.CasosUsoUsuarios.Implementaciones
{
    public class CasoUsoGetUsuarioByEmail : ICasoUsoGetUsuarioByEmail
    {
        private IRepositorioUsuarios _repoUsuarios;
        public CasoUsoGetUsuarioByEmail(IRepositorioUsuarios repoUsuarios)
        {
            _repoUsuarios = repoUsuarios;
        }

        public UsuarioDTO FindUserbyEmail(string email)
        {
            return
MapperUsuarioParaJWT.FromDTO(_repoUsuarios.ObtenerUsuarioPorEmail(email));
        }
    }
}
```

```
namespace LogicaAplicacion.CasosUso.CasosUsoUsuarios.Interfaces
{
    public interface ICasoUsoGetUsuarioByEmail
    {
        public UsuarioDTO FindUserbyEmail(string email);
    }
}
```

Mappers:

```
namespace LogicaAplicacion.DataTransferObjects.Mappers
{
    public class MapperArticulo
    {
        public static Articulo FromDTO(ArticulosDTO dto)
        {
            ArgumentNullException.ThrowIfNull(dto);
            return new Articulo(dto.Nombre, dto.Descripcion, dto.Codigo, dto.PrecioVenta,
dto.Stock);
        }

        public static ArticulosListadoDTO ToDTO(Articulo art)
        {
            return new ArticulosListadoDTO()
            {
                Id = art.Id,
                Nombre = art.Nombre,
                Descripcion = art.Descripcion,
                Codigo = art.Codigo,
                PrecioVenta = art.PrecioVenta,
                Stock = art.Stock
            };
        }

        public static IEnumerable<ArticulosListadoDTO> FromList(IEnumerable<Articulo> arts)
        {
            return arts.Select(x => ToDTO(x));
        }
    }
}

namespace LogicaAplicacion.DataTransferObjects.Mappers
{
    public class MapperMovimientoStock
    {
```

```
        public static MovimientoStock FromDTO(MovimientoDeStockDTO
movimientosDeStockDTO)
        {
            return new MovimientoStock
            {
                Id = movimientosDeStockDTO.Id,
                Articulo = movimientosDeStockDTO.Articulo,
                Cantidad = movimientosDeStockDTO.Cantidad,
                Fecha = movimientosDeStockDTO.Fecha,
                TipoMovimiento = movimientosDeStockDTO.TipoMovimiento,
                Usuario = movimientosDeStockDTO.Usuario
            };
        }

        public static MovimientoDeStockDTO ToDTO(MovimientoStock movimientoStock)
        {
            return new MovimientoDeStockDTO
            {
                Id = movimientoStock.Id,
                Articulo = movimientoStock.Articulo,
                ArticuloId = movimientoStock.Articulo.Id,
                ArticuloName = movimientoStock.Articulo.Nombre,
                Cantidad = movimientoStock.Cantidad,
                Fecha = movimientoStock.Fecha,
                TipoMovimiento = movimientoStock.TipoMovimiento,
                TipoMovimientoId = movimientoStock.TipoMovimiento.Id,
                Usuario = movimientoStock.Usuario,
                UsuarioId = movimientoStock.Usuario.Id
            };
        }

        public static IEnumerable<MovimientoDeStockDTO>
ToDTOList(IEnumerable<MovimientoStock> movimientosDeStockDTOs)
        {
            return movimientosDeStockDTOs.Select(movimientosDeStockDTO =>
ToDTO(movimientosDeStockDTO));
        }
    }
}

namespace LogicaAplicacion.DataTransferObjects.Mappers
{
    public class MapperTipoMovimiento
    {
        public static TipoMovimiento FromDTO(TipoMovimientoDTO tipoMovimientoDTO)
        {
            return new TipoMovimiento
            {
```

```
            Id = tipoMovimientoDTO.Id,
            Nombre = tipoMovimientoDTO.Nombre,
            Reduccion = tipoMovimientoDTO.Reduccion
        };
    }

    public static TipoMovimientoDTO ToDTO(TipoMovimiento tipoMovimiento)
    {
        return new TipoMovimientoDTO
        {
            Id = tipoMovimiento.Id,
            Nombre = tipoMovimiento.Nombre,
            Reduccion = tipoMovimiento.Reduccion
        };
    }

    public static IEnumerable<TipoMovimientoDTO>
ToDTOList(IEnumerable<TipoMovimiento> tiposMovimiento)
    {
        return tiposMovimiento.Select(tipoMovimiento => ToDTO(tipoMovimiento));
    }
  }
}

namespace LogicaAplicacion.DataTransferObjects.Mappers
{
  public class MapperUsuario
  {
    public static Usuario FromDTO(UsuarioAltaDTO dto)
    {
        ArgumentNullException.ThrowIfNull(dto);
        return new Usuario(dto.Email, dto.Nombre, dto.Apellido, dto.Password, dto.Rol);
    }
    public static Usuario FromDTO(UsuarioModificacionDTO dto)
    {
        ArgumentNullException.ThrowIfNull(dto);
        var usuario = new Usuario(dto.Email, dto.Nombre, dto.Apellido, dto.Password,
dto.Rol)
        {
            Id = dto.Id
        };
        return usuario;
    }

    public static Usuario FromDTO(UsuarioListadoDTO dto)
    {
        ArgumentNullException.ThrowIfNull(dto);
```

```
            var usuario = new Usuario(dto.Email, dto.Nombre, dto.Apellido, dto.Password,
(ERol)dto.Rol)
            {
                Id = dto.Id
            };
            return usuario;
        }
        public static UsuarioListadoDTO ToDTO(Usuario user)
        {
            return new UsuarioListadoDTO()
            {
                Email = user.Email,
                Password = user.Contrasenia,
                PasswordEncriptada = user.ContraseniaEncriptada,
                Rol = (int)user.Rol,
                Nombre = user.Nombre,
                Apellido = user.Apellido,
                Id = user.Id
            };
        }

        public static IEnumerable<UsuarioListadoDTO> FromList(IEnumerable<Usuario>
users)
        {
            return users.Select(x => ToDTO(x));
        }
    }
}
namespace LogicaAplicacion.DataTransferObjects.Mappers
{
    public class MapperUsuarioParaJWT
    {
        public static UsuarioDTO FromDTO(Usuario user)
        {
            if (user == null) return null;
            return new UsuarioDTO
            {
                Id = user.Id,
                Email = user.Email,
                ContraseniaEncriptada = user.ContraseniaEncriptada,
                Rol = user.Rol
            };
        }
        public static UsuarioDTO ToDTO(Usuario user)
        {
            return new UsuarioDTO
            {
                Id = user.Id,
```

```
            Email = user.Email,
            ContraseniaEncriptada = user.ContraseniaEncriptada,
            Rol = user.Rol
        };
    }

  }
}
```

DTOs

```
namespace LogicaAplicacion.DataTransferObjects.Models.Articulos
{
  public class ArticulosDTO
  {
    public int Id { get; set; }

    [Required(ErrorMessage = "Debe ingresar un nombre para el Articulo")]
    public string Nombre { get; set; }

    [Required(ErrorMessage = "Debe ingresar una descripcion para el Articulo")]
    public string Descripcion { get; set; }

    [Required(ErrorMessage = "Debe ingresar un codigo para el Articulo")]
    public long Codigo { get; set; }

    [Required(ErrorMessage = "Debe ingresar un precio de venta para el Articulo")]
    public decimal PrecioVenta { get; set; }

    [Required(ErrorMessage = "Debe ingresar un precio de venta para el Articulo")]
    public int Stock { get; set; }

    public ArticulosDTO() { }
  }
}

namespace LogicaAplicacion.DataTransferObjects.Models.MovimientosDeStock
{
  public class AltaMovimientoDeStockDTO
  {
    public int ArticuloId { get; set; }
    public int Cantidad { get; set; }
    public int TipoMovimientoId { get; set; }
    public int UsuarioId { get; set; }
  }
```

```
}

namespace LogicaAplicacion.DataTransferObjects.Models.MovimientosDeStock
{
    public class MovimientoDeStockDTO
    {
        public int Id { get; set; }
        public DateTime Fecha { get; set; }
        public Articulo? Articulo { get; set; }
        public int ArticuloId { get; set; }
        public string ArticuloName { get; set; }
        public int Cantidad { get; set; }
        public TipoMovimiento? TipoMovimiento { get; set; }
        public int TipoMovimientoId { get; set; }
        public Usuario? Usuario { get; set; }
        public int UsuarioId { get; set; }
    }
}

namespace LogicaAplicacion.DataTransferObjects.Models.TipoMovimientos
{
    [Index(nameof(Nombre), IsUnique = true)]
    public class TipoMovimientoDTO
    {
        public int Id { get; set; }

        public string Nombre { get; set; }
        public bool Reduccion { get; set; }
    }
}

namespace LogicaAplicacion.DataTransferObjects.Models.Usuarios
{
    public class UsuarioDTO
    {
        public int Id { get; set; }
        public string Email { get; set; }
        public string Contrasenia { get; set; }
        public string? ContraseniaEncriptada { get; set; }
        public ERol Rol { get; set; }

    }
}
```

Lógica Negocio:

```
namespace LogicaNegocio.Entidades
{
   public class Articulo : IValidable<Articulo>
   {
      #region Properties
      public int Id { get; set; }
      public string Nombre { get; set; }
      //[Range(5, int.MinValue, ErrorMessage = "La descripción debe tener minimo 5
caracteres")]
      public string Descripcion { get; set; }
      //[Range(13, int.MinValue, ErrorMessage = "El código debe tener 13 dígitos")]
      public long Codigo { get; set; }
      [Display(Name = "Precio")]
      public decimal PrecioVenta { get; set; }
      public int Stock { get; set; }
      #endregion

      #region Constructor
      public Articulo(string nombre, string descripcion, long codigo, decimal precioVenta, int
stock)
      {
         EsValido();
         Nombre = nombre;
         Descripcion = descripcion;
         Codigo = codigo;
         PrecioVenta = precioVenta;
         Stock = stock;
      }

      public Articulo(int id, string nombre, string descripcion, long codigo, decimal
precioVenta, int stock)
      {
         EsValido();
         Id = id;
         Nombre = nombre;
         Descripcion = descripcion;
         Codigo = codigo;
         PrecioVenta = precioVenta;
         Stock = stock;
      }

      public Articulo() { }
      #endregion

      public void EsValido()
```

```
        {
            ValidarCodigo();
            ValidarNombre();
            ValidarDescripcion();
        }

        private void ValidarCodigo()
        {
            string patronValido = @"\d{13}";

            if (string.IsNullOrEmpty(Codigo.ToString()))
                throw new ArticuloNoValidoException("El código del artículo no puede estar
vacio");
            else if (!Regex.IsMatch(Codigo.ToString(), patronValido))
                throw new ArticuloNoValidoException("El código del artículo debe tener 13
digitos");
        }

        private void ValidarNombre()
        {
            if (string.IsNullOrEmpty(Nombre))
                throw new ArticuloNoValidoException("Debe ingresar un nombre para el articulo");
            else if (Nombre.Length < 10 || Nombre.Length > 200)
                throw new ArticuloNoValidoException("El nombre del artículo debetener entre 10 y
200 carácteres");
        }

        private void ValidarDescripcion()
        {
            if (Descripcion.Length < 5)
                throw new ArticuloNoValidoException("La descripción del artículo debe contener al
menso 5 caracteres");
        }
    }
}

namespace LogicaNegocio.Entidades
{
    public class MovimientoStock
    {
        #region Properties
        public int Id { get; set; }
        public DateTime Fecha { get; set; }
        public Articulo Articulo { get; set; }
        public int Cantidad { get; set; }
        public TipoMovimiento TipoMovimiento { get; set; }
        public Usuario Usuario { get; set; }
        #endregion
```

```
        #region Constructores
        public MovimientoStock() {}

        public MovimientoStock(DateTime fecha, Articulo articulo, int cantidad,
            TipoMovimiento tipoMovimiento, Usuario user)
        {
            Fecha = DateTime.Now;
            Articulo = articulo;
            Cantidad = cantidad;
            TipoMovimiento = tipoMovimiento;
            Usuario = user;
            EsValido();
        }


        #endregion

        #region Methods
        public void EsValido()
        {
            if (Fecha == DateTime.Now)
            {
                throw new Exception("La fecha debe ser el dia de hoy");
            }
            if (Cantidad < 0)
            {
                throw new Exception("La cantidad no puede ser negativa");
            }
            if(Usuario.Rol != Enumerados.ERol.ENCARGADO)
            {
                throw new Exception("El usuario debe ser un encargado");
            }

        }
        #endregion
    }
}

namespace LogicaNegocio.Entidades
{
    public class Settings
    {
        [Key]
        public string Name { get; set; }
        public double Value { get; set; }
    }
}
```

```
namespace LogicaNegocio.Entidades
{
    public class TipoMovimiento
    {
        #region Properties
        public int Id { get; set; }
        public string Nombre { get; set; }
        //En caso que sea reduccion positivo, es un movimiento que reduce el stock
        public bool Reduccion { get; set; }
        #endregion

        #region Constructor
        public TipoMovimiento() {}
        public TipoMovimiento(int id, string nombre, bool reduccion)
        {
            Nombre = nombre;
            Reduccion = reduccion;
            EsValido();
        }


        #endregion

        #region Methods
        public void EsValido()
        {
            if (Nombre.IsNullOrEmpty())
            {
                throw new Exception("El nombre no puede ser nulo o vacio");
            }
        }
        #endregion
    }
}
namespace LogicaNegocio.Entidades
{
    public class Usuario : IValidable<Usuario>
    {
        #region Properties
        public int Id { get; set; }
        public string Email { get; set; }
        public string Nombre { get; set; }
        public string Apellido { get; set; }

        //[Range(6, int.MinValue, ErrorMessage = "La contraseña debe tener minimo 6
caracteres")]
        public string Contrasenia { get; set; }
```

```
        public string ContraseniaEncriptada { get; set; }
        public ERol Rol { get; set; }
        #endregion

        #region Constructor
        public Usuario(string email, string nombre, string apellido, string password, ERol rol)
        {
            Email = email;
            Nombre = nombre;
            Apellido = apellido;
            Contrasenia = password;
            ContraseniaEncriptada = Encriptar(password);
            Rol = rol;
        }

        public Usuario(int id, string email, string nombre, string apellido, string password, ERol
rol)
        {
            Id = id;
            Email = email;
            Nombre = nombre;
            Apellido = apellido;
            Contrasenia = password;
            ContraseniaEncriptada = Encriptar(password);
            Rol = rol;
        }
        public Usuario() { }
        #endregion

        #region Methods
        public static string Encriptar(string contrasenia)
        {
            using (SHA256 sha256Hash = SHA256.Create())
            {
                // Convertimos la contraseña en una secuencia de bytes
                byte[] bytes =
sha256Hash.ComputeHash(Encoding.UTF8.GetBytes(contrasenia));

                // Convertimos los bytes a una cadena hexadecimal
                StringBuilder builder = new StringBuilder();
                for (int i = 0; i < bytes.Length; i++)
                {
                    builder.Append(bytes[i].ToString("x2"));
                }
                return builder.ToString();
            }
        }
        public void ModificarUsuario(Usuario obj)
```

```
        {
            Email = obj.Email;
            Contrasenia = obj.Contrasenia;
            Rol = obj.Rol;
            Nombre = obj.Nombre;
            Apellido = obj.Apellido;
        }

        public void EsValido()
        {
            ValidarDireccionEmail();
            ValidarNombre();
            ValidarApellido();
            ValidarContrasenia();
            ValidarRol();
        }
        private void ValidarDireccionEmail()
        {
            string patronValido = @"^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$";

            if (string.IsNullOrEmpty(Email))
                throw new MailNoValidoException("El email no puede ser vacio.");
            else if (!Regex.IsMatch(Email, patronValido))
                throw new MailNoValidoException("Ingrese un email valido");
        }

        public void ValidarNombre()
        {
            string patronValido = @"^[a-zA-ZáéíóúÁÉÍÓÚüÜñÑ]+(?:['
-][a-zA-ZáéíóúÁÉÍÓÚüÜñÑ]+)*$";

            if (string.IsNullOrEmpty(Nombre))
                throw new NombreNoValidoException("Nombre no puede ser vacío");
            else if (!Regex.IsMatch(Nombre, patronValido))
                throw new NombreNoValidoException("El nombre solo puede contener letras,
espacios, apostrofes o guiones. Los caracteres especiales no pueden estar al principio ni al
final.");
        }

        public void ValidarApellido()
        {
            string patronValido = @"^[a-zA-ZáéíóúÁÉÍÓÚüÜñÑ]+(?:['
-][a-zA-ZáéíóúÁÉÍÓÚüÜñÑ]+)*$";

            if (string.IsNullOrEmpty(Apellido))
                throw new NombreNoValidoException("Apellido no puede ser vacío");
            else if (!Regex.IsMatch(Apellido, patronValido))
```

```
            throw new NombreNoValidoException("El apellido solo puede contener letras, espacios, apostrofes o guiones. Los caracteres especiales no pueden estar al principio ni al final.");
        }

        private void ValidarContrasenia()
        {
            string patronValido = @"^(?=.*[A-Z])(?=.*[a-z])(?=.*\d)(?=.*[.,;!]).{6,}$";
            if (!Regex.IsMatch(Contrasenia, patronValido))
                throw new ContraseñaNoValidaException("La contraseña debe tener al menos largo 6. Largo mínimo de 6 caracteres, al menos una letra mayúscula, una minúscula, un dígito y un carácter de puntuación (.,;!)");
        }
        private void ValidarRol()
        {
            if (Rol != ERol.ADMINISTRADOR && Rol != ERol.USUARIO && Rol != ERol.ENCARGADO)
                throw new UsuarioNoValidoException("Ingrese un rol valido");
        }
        #endregion
    }
}


namespace LogicaNegocio.Enumerados
{
    public enum ERol
    {
        USUARIO = 1,
        ADMINISTRADOR = 2,
        ENCARGADO = 3
    }
}


namespace LogicaNegocio.Interfaces
{
    public interface IValidable<T> where T : class
    {
        public void EsValido();
    }
}
```

WebApi:

```
namespace WebApi.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class LoginController : ControllerBase
    {
        private ICasoUsoGetUsuarioByEmail _getUser;

        public LoginController(ICasoUsoGetUsuarioByEmail getUser)
        {
            _getUser = getUser;
        }

        /// <summary>
        /// Metodo que permite iniciar sesion y obtener un token.
        /// </summary>
        /// <param name="user">Usuario con rol que desea iniciar sesion</param>
        /// <returns>Retorna datos del usuario y token</returns>
        [HttpPost]
        [AllowAnonymous]
        [Route("login")]
        public ActionResult<UsuarioDTO> Login([FromBody] UsuarioDTO user)
        {
            try
            {
                ManejadorJWT handler = new ManejadorJWT(_getUser);
                var usuario = handler.ObtenerUsuario(user.Email);
                if (usuario == null || Usuario.Encriptar(user.Contrasenia) !=
usuario.ContraseniaEncriptada || usuario.Rol != ERol.ENCARGADO)
                {
                    return Unauthorized("Credenciales invalidas");
                }
                var token = ManejadorJWT.GenerarToken(usuario);
                return Ok(new
                {
                    Token = token,
                    Usuario = usuario,
                    Rol = usuario.Rol,
                    Id = usuario.Id
                });
            }
            catch (Exception)
            {
```

```
            return Unauthorized(new { Message = "Se produjo un error, intente nuevamente"
});
        }
    }
  }
}

namespace WebApi.Controllers
{
  [Route("api/[controller]")]
  [ApiController]
  [Authorize]
  public class MovimientoStockController : ControllerBase
  {
    #region Properties
    private ICasoUsoListarMovimientoStock _listarMovimientoStock;
    private ICasoUsoAltaMovimientoStock _altaMovimientoStock;
    private ICasoUsoBuscarArticulo _buscarArticulo;
    private ICasoUsoBuscarUsuario _buscarUsuario;
    private ICasoUsoBuscarTipoMovimiento _buscarTipoMovimiento;
    #endregion

    #region Constructor
    public MovimientoStockController(
      ICasoUsoAltaMovimientoStock altaMovimientoStock,
      ICasoUsoListarMovimientoStock listarMovimientoStock,
      ICasoUsoBuscarArticulo buscarArticulo,
      ICasoUsoBuscarUsuario buscarUsuario,
      ICasoUsoBuscarTipoMovimiento buscarTipoMovimiento
    )
    {
      _listarMovimientoStock = listarMovimientoStock;
      _altaMovimientoStock = altaMovimientoStock;
      _buscarArticulo = buscarArticulo;
      _buscarUsuario = buscarUsuario;
      _buscarTipoMovimiento = buscarTipoMovimiento;
    }
    #endregion

    /// <summary>
    /// Devuelve la lista de todos los movimientos de stock registrados en el sistema.
    /// </summary>
    /// <returns>Lista de movimientos de stock en el sistema</returns>
    [HttpGet(Name = "getAll")]
    [ProducesResponseType(StatusCodes.Status200OK)]
    [ProducesResponseType(StatusCodes.Status204NoContent)]
    [ProducesResponseType(StatusCodes.Status401Unauthorized)]
    public ActionResult<IEnumerable<MovimientoDeStockDTO>> Get()
```

```
        {
            IEnumerable<MovimientoDeStockDTO> lista =
_listarMovimientoStock.ListarMovimientosDeStock();
            if (lista.Count() == 0)
            {
                return NoContent();
            }
            else
            {
                return Ok(lista);
            }
        }


        /// <summary>
        /// Registra un nuevo movimiento de stock en el sistema.
        /// </summary>
        /// <param name="movimientoDeStockDTO">Movimiento de stock que se quiere
registrar</param>
        [HttpPost]
        [ProducesResponseType(StatusCodes.Status201Created)]
        [ProducesResponseType(StatusCodes.Status400BadRequest)]
        [ProducesResponseType(StatusCodes.Status401Unauthorized)]
        public ActionResult<MovimientoDeStockDTO> Post([FromBody]
AltaMovimientoDeStockDTO altaMovimientoDeStockDTO)
        {
            try
            {
                MovimientoDeStockDTO movimientoCompleto = new  MovimientoDeStockDTO()
                {
                    Fecha = DateTime.Now.ToLocalTime(),
                    Cantidad = altaMovimientoDeStockDTO.Cantidad,
                    Articulo =
_buscarArticulo.BuscarArticulo(altaMovimientoDeStockDTO.ArticuloId),
                    Usuario =
MapperUsuario.FromDTO(_buscarUsuario.BuscarUsuario(altaMovimientoDeStockDTO.Usu
arioId)),
                    TipoMovimiento =
_buscarTipoMovimiento.BuscarTipoMovimiento(altaMovimientoDeStockDTO.TipoMovimient
oId)
                };
                _altaMovimientoStock.AltaMovimientoStock(movimientoCompleto);
                return Created("api/TipoMovimiento", movimientoCompleto);
            }
            catch (Exception ex)
            {
                return BadRequest(ex.Message);
            }
```

```
    }
  }
}

namespace WebApi.Controllers
{
  [Route("api/[controller]")]
  [ApiController]
  [Authorize]
  public class ReportesController : ControllerBase
  {
    private ICasoUsoGetArticulosConMovimientos _getArticulosConMovimientos;
    private ICasoUsoGetMovimientos _getMovimientos;
    private ICasoUsoGetResumenMovimientos _getResumenMovimientos;

    public ReportesController(ICasoUsoGetArticulosConMovimientos
getArticulosConMovimientos,
                              ICasoUsoGetMovimientos getMovimientos,
                              ICasoUsoGetResumenMovimientos getResumenMovimientos)
    {
      _getArticulosConMovimientos = getArticulosConMovimientos;
      _getMovimientos = getMovimientos;
      _getResumenMovimientos = getResumenMovimientos;
    }
    /// <summary>
    /// Lista los articulos que contengan movimientos entre dos fechas dadas
    /// </summary>
    /// <param name="fecha1"></param>
    /// <param name="fecha2"></param>
    /// <param name="pageNumber"></param>
    /// <returns>Retorna la lista de articulos con sus respectivos movimientos</returns>
    [HttpGet("GetArticulosConMovimientos/{fecha1}/{fecha2}/{pageNumber}")]
    [ProducesResponseType(StatusCodes.Status200OK)]
    [ProducesResponseType(StatusCodes.Status204NoContent)]
    [ProducesResponseType(StatusCodes.Status400BadRequest)]

    public ActionResult<IEnumerable<ArticulosListadoDTO>>
ArticulosConMovimientos(DateTime fecha1, DateTime fecha2, int pageNumber)
    {
      try
      {
        if (pageNumber < 1) return BadRequest("El numero de pagina debe ser mayor a
0");
        IEnumerable<ArticulosListadoDTO> articulos =
_getArticulosConMovimientos.GetArticulosConMovimientos(fecha1, fecha2, pageNumber);
        if (articulos.Count() == 0)
        {
          return NoContent();
```

```
            }
            else
            {
                return Ok(articulos);
            }
        }
        catch (Exception)
        {

            return BadRequest("Las fechas no son correctas");
        }
    }
    /// <summary>
    /// Lista todos los movimientos de ese tipo Id realizados sobre ese articulo Id
    /// </summary>
    /// <param name="articuloId"></param>
    /// <param name="tipoMovimientoId"></param>
    /// <param name="pageNumber"></param>
    /// <returns>Retorna los movimientos</returns>
    [HttpGet("GetMovimientos/{articuloId}/{tipoMovimientoId}/{pageNumber}")]
    [ProducesResponseType(StatusCodes.Status200OK)]
    [ProducesResponseType(StatusCodes.Status204NoContent)]
    [ProducesResponseType(StatusCodes.Status400BadRequest)]
    //TODO Error con el paginado
    public ActionResult<IEnumerable<MovimientoDeStockDTO>> Movimientos(int
articuloId, int tipoMovimientoId, int pageNumber)
    {
        try
        {
            if (pageNumber < 1) return BadRequest("El numero de pagina debe ser mayor a
0");
            IEnumerable<MovimientoDeStockDTO> movimientos =
_getMovimientos.GetMovimientos(articuloId, tipoMovimientoId, pageNumber);
            if (movimientos.Count() == 0)
            {
                return NoContent();
            }
            else
            {
                return Ok(movimientos);
            }
        }
        catch (Exception)
        {
            return BadRequest("Los id son incorrectos intente de nuevo");
        }
    }
    /// <summary>
```

```
        /// Resumen de movimientos con sus respectivas cantidades movidas agrupadas por
año y por tipo de movimiento
        /// </summary>
        /// <returns>Resumen</returns>
        [HttpGet("GetResumenMovimientos")]
        [ProducesResponseType(StatusCodes.Status200OK)]
        [ProducesResponseType(StatusCodes.Status204NoContent)]
        [ProducesResponseType(StatusCodes.Status400BadRequest)]
        public ActionResult<IEnumerable<Object>> ResumenMovimientos()
        {
            try
            {
                IEnumerable<Object> movimientos =
_getResumenMovimientos.GetResumenMovimientos();
                if (movimientos.Count() == 0)
                {
                    return NoContent();
                }
                else
                {
                    return Ok(movimientos);
                }
            }
            catch (Exception ex)
            {
                return BadRequest(ex.Message);
            }
        }

    }
}

namespace WebApi.Controllers
{
    [Route("api/TipoMovimiento")]
    [ApiController]
    public class TipoMovimientoController : ControllerBase
    {
        #region Properties
        private ICasoUsoListarTipoMovimiento _listarTipoMovimiento;
        private ICasoUsoAltaTipoMovimiento _altaTipoMovimiento;
        private ICasoUsoBajaTipoMovimiento _bajaTipoMovimiento;
        private ICasoUsoEditTipoMovimiento _editTipoMovimiento;
        private ICasoUsoObtenerPorTipoMovimiento _obtenerPorTipoMovimiento;
        #endregion

        #region Constructor
        public TipoMovimientoController(
```

```
            ICasoUsoListarTipoMovimiento listarTipoMovimiento,
            ICasoUsoAltaTipoMovimiento altaTipoMovimiento,
            ICasoUsoBajaTipoMovimiento bajaTipoMovimiento,
            ICasoUsoEditTipoMovimiento editTipoMovimiento,
            ICasoUsoObtenerPorTipoMovimiento obtenerPorTipoMovimiento
        )
        {
            _listarTipoMovimiento = listarTipoMovimiento;
            _altaTipoMovimiento = altaTipoMovimiento;
            _bajaTipoMovimiento = bajaTipoMovimiento;
            _editTipoMovimiento = editTipoMovimiento;
            _obtenerPorTipoMovimiento = obtenerPorTipoMovimiento;
        }
        #endregion

        /// <summary>
        /// Devuelve una lista con todos los tipos de movimiento registrados en el sistema
        /// </summary>
        /// <returns>Lista de tipos de movimiento</returns>
        [HttpGet(Name = "getAllTipoMovimientos")]
        [ProducesResponseType(StatusCodes.Status200OK)]
        [ProducesResponseType(StatusCodes.Status204NoContent)]
        public ActionResult<IEnumerable<TipoMovimientoDTO>> Get()
        {
            IEnumerable<TipoMovimientoDTO> lista = _listarTipoMovimiento.GetAll();
            if (lista.Count() == 0)
            {
                return NoContent();
            }else
            {
                return Ok(lista);
            }
        }

        /// <summary>
        /// Devuelve una lista filtrada por un tipo de movimiento seleccionado
        /// </summary>
        /// <param name="tipoMovimiento">Tipo de movimiento utilizado para filtrar la lista
completa</param>
        /// <returns>Lista de tipos de movimiento</returns>
        [HttpGet("getByTipo/{tipoMovimiento}")]
        [ProducesResponseType(StatusCodes.Status200OK)]
        [ProducesResponseType(StatusCodes.Status204NoContent)]
        [ProducesResponseType(StatusCodes.Status400BadRequest)]
        public ActionResult<IEnumerable<TipoMovimientoDTO>> GetByTipoMovimiento(string
tipoMovimiento)
        {
```

```
            IEnumerable<TipoMovimientoDTO> lista =
_obtenerPorTipoMovimiento.ObtenerPorTipoMovimiento(tipoMovimiento);
            if (lista.Count() == 0)
            {
                return NoContent();
            }
            else
            {
                return Ok(lista);
            }
        }

        /// <summary>
        /// Registra un nuevo tipo de movimiento en el sistema.
        /// </summary>
        /// <param name="tipoDTO">Tipo de movimiento a registrar en el sistema.</param>
        /// <returns>Sin retorno</returns>
        [HttpPost]
        [ProducesResponseType(StatusCodes.Status201Created)]
        [ProducesResponseType(StatusCodes.Status400BadRequest)]
        public ActionResult<TipoMovimientoDTO> Create([FromBody] TipoMovimientoDTO
tipoDTO)
        {
            try
            {
                _altaTipoMovimiento.AltaTipoMovimiento(tipoDTO);
                return Created("api/TipoMovimiento", tipoDTO);
            }
            catch (Exception ex)
            {

                return BadRequest(ex.Message);
            }
        }

        /// <summary>
        /// Modifica la informacion de un tipo de movimiento registrado en el sistema.
        /// </summary>
        /// <param name="tipoMovimientoId">Id del tipo de movimiento que se va a
modificar.</param>
        /// <param name="tipoDTO">Datos utilizados para modificar el tipo de movimiento
seleccionado</param>
        /// <returns>Sin retorno</returns>
        [HttpPut("update/{tipoMovimientoId}")]
        [ProducesResponseType(StatusCodes.Status200OK)]
        [ProducesResponseType(StatusCodes.Status400BadRequest)]
        public ActionResult<TipoMovimientoDTO> Update(int tipoMovimientoId, [FromBody]
TipoMovimientoDTO tipoDTO)
```

```
        {
            try
            {
                _editTipoMovimiento.EditTipoMovimiento(tipoMovimientoId, tipoDTO);
                return Ok("Tipo movimiento " + tipoDTO.Nombre + " editado correctamente.");
            }
            catch (Exception ex)
            {
                return BadRequest(ex.Message);
            }
        }

        /// <summary>
        /// Elimina el registro de un tipo de movimiento en el sistema
        /// </summary>
        /// <param name="id">Id del tipo de movimiento que se desea eliminar</param>
        /// <param name="tipoDTO">Datos del tipo de movimiento que se desea
eliminar</param>
        /// <returns>Sin retorno</returns>
        [HttpDelete("delete/{id}")]
        [ProducesResponseType(StatusCodes.Status200OK)]
        [ProducesResponseType(StatusCodes.Status400BadRequest)]
        public ActionResult<TipoMovimientoDTO> Delete(int id, [FromBody]
TipoMovimientoDTO tipoDTO)
        {
            try
            {
                _bajaTipoMovimiento.BajaTipoMovimiento(id);
                return Ok("Tipo movimiento " + tipoDTO.Id + " dado de baja correctamente.");
            }
            catch (Exception ex)
            {
                return BadRequest(ex.Message);
            }
        }
    }
}
namespace WebApi
{
    public class ManejadorJWT
    {
        private ICasoUsoGetUsuarioByEmail _getUser;

        public ManejadorJWT(ICasoUsoGetUsuarioByEmail getUser)
        {
            _getUser = getUser;
        }
```

```
        public static string GenerarToken(UsuarioDTO usuarioDTO)
        {
            var tokenHandler = new JwtSecurityTokenHandler();

            //Clave secreta, signature:

            var clave =
Encoding.ASCII.GetBytes("CharliXCX!NewAlbum-BRAT-isOUTNOW!STREAM!!!");
            var tokenDescriptor = new SecurityTokenDescriptor
            {
                Subject = new ClaimsIdentity(new Claim[] {
                    new Claim(ClaimTypes.Email, usuarioDTO.Email)
                }),
                Expires = DateTime.UtcNow.AddMonths(1),

                SigningCredentials = new SigningCredentials(new SymmetricSecurityKey(clave),
                SecurityAlgorithms.HmacSha256Signature)
            };

            var token = tokenHandler.CreateToken(tokenDescriptor);

            return tokenHandler.WriteToken(token);
        }

        public UsuarioDTO ObtenerUsuario(string email)
        {
            var usuario = _getUser.FindUserbyEmail(email);
            return usuario;
        }
    }
}
namespace WebApi
{
    public class Program
    {
        public static void Main(string[] args)
        {
            var builder = WebApplication.CreateBuilder(args);

            // Add services to the container.
            builder.Services.AddControllers();

            #region Repositorios
            builder.Services.AddScoped<IRepositorioUsuarios, RepositorioUsuarios>();
            builder.Services.AddScoped<IRepositorioArticulos, RepositorioArticulos>();
            builder.Services.AddScoped<IRepositorioPedidos, RepositorioPedidos>();
            builder.Services.AddScoped<IRepositorioTipoMovimiento,
RepositorioTipoMovimiento>();
```

```
        builder.Services.AddScoped<IRepositorioMovimientoDeStock,
RepositorioMovimientoDeStock>();
        builder.Services.AddScoped<IRepositorioSettings, RepositorioSettings>();
        #endregion

        #region Casos de uso
        //Usuarios
        builder.Services.AddScoped<ICasoUsoGetUsuarioByEmail,
CasoUsoGetUsuarioByEmail>();
        builder.Services.AddScoped<ICasoUsoBuscarUsuario, CasoUsoBuscarUsuario>();
        //Articulos
        builder.Services.AddScoped<ICasoUsoListarArticulos, CasoUsoListarArticulos>();
        builder.Services.AddScoped<ICasoUsoBuscarArticulo, CasoUsoBuscarArticulo>();

builder.Services.AddScoped<ICasoUsoListarOrdenadoAlfabeticamenteAscendenteArticulo,
CasoUsoListarOrdenadoAlfabeticamenteAscendenteArticulo>();

builder.Services.AddScoped<ICasoUsoListarOrdenadoDescendentementePorFechaPedido,
CasoUsoListarOrdenadoDescendentementePorFechaPedido>();
        builder.Services.AddScoped<ICasoUsoGetArticulosConMovimientos,
CasoUsoGetArticulosConMovimientos>();
        //Tipo Movimiento
        builder.Services.AddScoped<ICasoUsoAltaTipoMovimiento,
CasoUsoAltaTipoMovimiento>();
        builder.Services.AddScoped<ICasoUsoBajaTipoMovimiento,
CasoUsoBajaTipoMovimiento>();
        builder.Services.AddScoped<ICasoUsoEditTipoMovimiento,
CasoUsoEditTipoMovimiento>();
        builder.Services.AddScoped<ICasoUsoListarTipoMovimiento,
CasoUsoListarTipoMovimiento>();
        builder.Services.AddScoped<ICasoUsoBuscarTipoMovimiento,
CasoUsoBuscarTipoMovimiento>();
        builder.Services.AddScoped<ICasoUsoObtenerPorTipoMovimiento,
CasoUsoObtenerPorTipoMovimiento>();
        //Movimientos de stock
        builder.Services.AddScoped<ICasoUsoListarMovimientoStock,
CasoUsoListarMovimientoStock>();
        builder.Services.AddScoped<ICasoUsoAltaMovimientoStock,
CasoUsoAltaMovimientoStock>();
        builder.Services.AddScoped<ICasoUsoGetResumenMovimientos,
CasoUsoGetResumenMovimientos>();
        //Reportes
        builder.Services.AddScoped<ICasoUsoGetMovimientos,
CasoUsoGetMovimientos>();

        #endregion
```

```
        // Learn more about configuring Swagger/OpenAPI at
https://aka.ms/aspnetcore/swashbuckle
        builder.Services.AddEndpointsApiExplorer();
        builder.Services.AddSwaggerGen();

        var clave = "CharliXCX!NewAlbum-BRAT-isOUTNOW!STREAM!!!";
        builder.Services.AddAuthentication(
           aut =>
           {
              aut.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;
              aut.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;
           }
        ).AddJwtBearer(opciones =>
           {
              opciones.RequireHttpsMetadata = false;
              opciones.SaveToken = true;
              opciones.TokenValidationParameters = new
Microsoft.IdentityModel.Tokens.TokenValidationParameters
              {
                 ValidateIssuerSigningKey = true,
                 IssuerSigningKey = new
SymmetricSecurityKey(System.Text.Encoding.ASCII.GetBytes(clave)),
                 ValidateIssuer = false,
                 ValidateAudience = false,
              };
           }
        );
        var rutaArchivo =
System.IO.Path.Combine(AppDomain.CurrentDomain.BaseDirectory, "WebApi.xml");
        builder.Services.AddSwaggerGen(
   opciones =>
   {
      opciones.IncludeXmlComments(rutaArchivo);
      opciones.AddSecurityDefinition("oauth2", new OpenApiSecurityScheme()
      {
         Description = "Autorización estándar mediante esquema Bearer",
         In = ParameterLocation.Header,
         Name = "Authorization",
         Type = SecuritySchemeType.ApiKey
      });
      opciones.OperationFilter<SecurityRequirementsOperationFilter>();
      opciones.SwaggerDoc("v1", new Microsoft.OpenApi.Models.OpenApiInfo
      {
         Title = "Papeleria",
         Description = "Proyecto Obligatorio 2 N3D",
         Version = "v1"
      });
   }
```

```
);

        var app = builder.Build();

        // Configure the HTTP request pipeline.
        if (app.Environment.IsDevelopment())
        {
            app.UseSwagger();
            app.UseSwaggerUI();
        }

        app.UseAuthentication();
        app.UseAuthorization();


        app.MapControllers();

        app.Run();
    }
  }
}
```

MVC:

```
namespace WebApplication1.Controllers
{
   public class HomeController : Controller
   {
      private readonly ILogger<HomeController> _logger;

      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 });
```

```
        }
    }
}

namespace WebApplication1.Controllers
{
    public class MovimientosDeStockController : Controller
    {
        // GET: MovimientosDeStockController/Create
        public ActionResult Create()
        {
            if (HttpContext.Session.GetString("token") == "" ||
HttpContext.Session.GetString("rol") != "ENCARGADO")
            {
                TempData["ErrorMessage"] = "Debe estar logueado para ingresar al sistema.";
                return RedirectToAction("Login", "Usuario");
            }
            return View(cargarSelects());
        }

        // POST: MovimientosDeStockController/Create
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Create(MovimientosDeStockModel movimientosDeStockModel)
        {
            if (HttpContext.Session.GetString("rol") != "ENCARGADO")
            {
                TempData["ErrorMessage"] = "Debe estar logueado para ingresar al sistema.";
                return RedirectToAction("Login", "Usuario");
            }
            try
            {
                //Fill missing para,eter with session user id
                movimientosDeStockModel.UsuarioId = HttpContext.Session.GetInt32("id");

                HttpClient cliente = new HttpClient();
                Uri uri = new Uri("http://localhost:5226/api/MovimientoStock");

                string token = HttpContext.Session.GetString("token");

                if (String.IsNullOrEmpty(token))
                {
                    TempData["ErrorMessage"] = "Debe estar logueado para ingresar al sistema.";
                    RedirectToAction("Login", "Usuario");
                }
                cliente.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue(
                    "Bearer",
                    token
```

```
            );

            HttpRequestMessage solicitud = new HttpRequestMessage(HttpMethod.Post, uri);
            string payload = JsonConvert.SerializeObject(movimientosDeStockModel);
            HttpContent contenido = new StringContent(payload, Encoding.UTF8,
"application/json");
            solicitud.Content = contenido;

            Task<HttpResponseMessage> respuesta = cliente.SendAsync(solicitud);
            respuesta.Wait();

            switch (respuesta.Result.StatusCode)
            {
                case HttpStatusCode.Created:
                    TempData["Message"] = "Movimiento de stock ingresado con exito";
                    break;
                case HttpStatusCode.Unauthorized:
                    TempData["ErrorMessage"] = "Debe ser un encargado para ingresar un
movimiento de stock";
                    break;
                case HttpStatusCode.BadRequest:
                    TempData["ErrorMessage"] = "Error en el ingreso de la informacion";
                    break;
                default:
                    TempData["ErrorMessage"] = "Error desconocido contacte a los bebedores
de cafe y energizantes bien facheros.";
                    break;
            }

            return RedirectToAction(nameof(Create));
        }
        catch
        {
            TempData["ErrorMessage"] = "Error desconocido contacte a los bebedores de
cafe y energizantes bien facheros.";
            return View("Index","Home");
        }
    }

    public MovimientosDeStockModel cargarSelects()
    {
        IEnumerable<TipoMovimientoModel> tiposModel = new
List<TipoMovimientoModel>();
        HttpClient httpClient = new HttpClient();
        string tiposUrl = "http://localhost:5226/api/TipoMovimiento";
        HttpResponseMessage response = httpClient.GetAsync(tiposUrl).Result;
```

```
            if (response.IsSuccessStatusCode)
            {
                HttpContent httpContent = response.Content;
                string responseDeMovimiento = httpContent.ReadAsStringAsync().Result;
                tiposModel =
JsonConvert.DeserializeObject<IEnumerable<TipoMovimientoModel>>(responseDeMovimie
nto);
            }
            MovimientosDeStockModel movimientosDeStockModel = new
MovimientosDeStockModel();
            movimientosDeStockModel.TipoMovimientos = tiposModel;

            IEnumerable<ArticuloModel> articuloModel = new List<ArticuloModel>();
            HttpClient httpClient2 = new HttpClient();
            string tiposUrl2 = "http://localhost:5226/api/Articulo";
            HttpResponseMessage response2 = httpClient2.GetAsync(tiposUrl2).Result;


            if (response2.IsSuccessStatusCode)
            {
                HttpContent httpContent2 = response2.Content;
                string responseDeArticulos = httpContent2.ReadAsStringAsync().Result;
                articuloModel =
JsonConvert.DeserializeObject<IEnumerable<ArticuloModel>>(responseDeArticulos);
            }

            movimientosDeStockModel.Articulos = articuloModel;
            return movimientosDeStockModel;
        }

    }
}

namespace WebApplication1.Controllers
{
    public class ReportesController : Controller
    {
        private static int ActualPage { get; set; }
        private static MovimientosEntreFechasModel? MovimientosEntreFechasModelTemp {
get; set; }
        private static MovimientosPorTipoModel? MovimientosPorTipoModelTemp { get; set; }


        public IActionResult Index()
        {
            MovimientosEntreFechasModelTemp = null;
            MovimientosPorTipoModelTemp = null;
            if (ActualPage != 1) { ActualPage = 1; }
```

```
            return View();
        }

        [HttpGet]
        public IActionResult MovimientosEntreFechas()
        {
            if (HttpContext.Session.GetString("token") == "" ||
HttpContext.Session.GetString("rol") != "ENCARGADO")
            {
                TempData["ErrorMessage"] = "Debe estar logueado para ingresar al sistema.";
                return RedirectToAction("Login", "Usuario");
            }
            if (MovimientosEntreFechasModelTemp == null)
            {
                return View();
            }
            else
            {
                try
                {
                    HttpClient cliente = new HttpClient();

                    string? token = HttpContext.Session.GetString("token");

                    if (String.IsNullOrEmpty(token))
                    {
                        TempData["ErrorMessage"] = "Debe estar logueado para ingresar al
sistema.";
                        RedirectToAction("Login", "Usuario");
                    }
                    cliente.DefaultRequestHeaders.Authorization = new
AuthenticationHeaderValue(
                        "Bearer",
                        token
                    );

                    string desde =
MovimientosEntreFechasModelTemp.Desde.ToString("yyyy-MM-dd");
                    string hasta =
MovimientosEntreFechasModelTemp.Hasta.ToString("yyyy-MM-dd");
                    Uri uri = new
Uri("http://localhost:5226/api/Reportes/GetArticulosConMovimientos/" + desde + "/" + hasta
+ "/" + ActualPage);

                    Task<HttpResponseMessage> respuesta = cliente.GetAsync(uri);
                    respuesta.Wait();

                    IEnumerable<MovimientosEntreFechasRespuestaModel>? modeloRespuesta;
```

```
                switch (respuesta.Result.StatusCode)
                {
                    case HttpStatusCode.OK:
                        HttpContent contenidoRespuesta = respuesta.Result.Content;
                        string stringContenido = contenidoRespuesta.ReadAsStringAsync().Result;
                        modeloRespuesta =
JsonConvert.DeserializeObject<List<MovimientosEntreFechasRespuestaModel>>(stringCon
tenido);
                        break;
                    case HttpStatusCode.NoContent:
                        TempData["Information"] = "No hay paginas para mostrar";
                        MovimientosEntreFechasModelTemp = null;
                        ActualPage = 1;
                        throw new Exception();
                    case HttpStatusCode.Unauthorized:
                        TempData["ErrorMessage"] = "Debe ser un encargado para ver el
reporte.";
                        MovimientosEntreFechasModelTemp = null;
                        ActualPage = 1;
                        throw new Exception();
                    case HttpStatusCode.BadRequest:
                        TempData["ErrorMessage"] = "Error en el ingreso de la información.";
                        MovimientosEntreFechasModelTemp = null;
                        ActualPage = 1;
                        throw new Exception();
                    default:
                        TempData["ErrorMessage"] = "Error desconocido. Contacte a los
bebedores de café y energizantes.";
                        MovimientosEntreFechasModelTemp = null;
                        ActualPage = 1;
                        throw new Exception();
                }

                return View("MostrarMovimientosEntreFechas", modeloRespuesta);
            }
            catch (Exception)
            {
                return RedirectToAction("MovimientosEntreFechas");
            }
        }
    }

    [HttpPost]
    public IActionResult MovimientosEntreFechas(MovimientosEntreFechasModel
movimientosEntreFechasModel)
    {
```

```
        if (HttpContext.Session.GetString("token") == "" ||
HttpContext.Session.GetString("rol") != "ENCARGADO")
        {
            TempData["ErrorMessage"] = "Debe estar logueado para ingresar al sistema.";
            return RedirectToAction("Login", "Usuario");
        }

        MovimientosEntreFechasModelTemp ??= movimientosEntreFechasModel;

        return RedirectToAction();
    }

    [HttpGet]
    public IActionResult NextMovimientoEntreFechas()
    {
        try
        {
            ActualPage++;
            if (ActualPage < 1)
            {
                ActualPage = 1;
                TempData["Information"] = "No hay mas paginas para mostrar";
            }

            return RedirectToAction("MovimientosEntreFechas");
        }
        catch (Exception)
        {
            TempData["ErrorMessage"] = "404 Not found";
            return RedirectToAction("Index", "Home");
        }
    }

    [HttpGet]
    public IActionResult PreviuosMovimientoEntreFechas()
    {
        try
        {
            ActualPage--;
            if (ActualPage < 1)
            {
                ActualPage = 1;
                TempData["Information"] = "No hay mas paginas para mostrar";
            }

            return RedirectToAction("MovimientosEntreFechas");
        }
        catch (Exception)
```

```
        {
            TempData["ErrorMessage"] = "404 Not found";
            return RedirectToAction("Index", "Home");
        }
    }

    [HttpGet]
    public IActionResult MostrarMovimientosEntreFechas()
    {
        if (HttpContext.Session.GetString("token") == "" ||
HttpContext.Session.GetString("rol") != "ENCARGADO")
        {
            TempData["ErrorMessage"] = "Debe estar logueado para ingresar al sistema.";
            return RedirectToAction("Login", "Usuario");
        }
        return View();
    }



    [HttpGet]
    public IActionResult MovimientosPorTipo()
    {
        if (HttpContext.Session.GetString("token") == "" ||
HttpContext.Session.GetString("rol") != "ENCARGADO")
        {
            TempData["ErrorMessage"] = "Debe estar logueado para ingresar al sistema.";
            return RedirectToAction("Login", "Usuario");
        }
        if(MovimientosPorTipoModelTemp == null)
        {
            return View(cargarSelects());
        }
        else
        {

            try
            {
                HttpClient cliente = new HttpClient();

                string? token = HttpContext.Session.GetString("token");

                if (String.IsNullOrEmpty(token))
                {
                    TempData["ErrorMessage"] = "Debe estar logueado para ingresar al
sistema.";
                    RedirectToAction("Login", "Usuario");
                }
```

```
            cliente.DefaultRequestHeaders.Authorization = new
AuthenticationHeaderValue(
                "Bearer",
                token
            );

            IEnumerable<MovimientosDeStockModel>? movimientosModelRespuesta = [];
            string url = "http://localhost:5226/api/Reportes/GetMovimientos/" +
MovimientosPorTipoModelTemp.ArticuloId
                + "/" + MovimientosPorTipoModelTemp.TipoMovimientoId + "/" + ActualPage;
            HttpResponseMessage respuesta = cliente.GetAsync(url).Result;

            switch (respuesta.StatusCode)
            {
                case HttpStatusCode.OK:
                    HttpContent contenido = respuesta.Content;
                    string stringContenido = contenido.ReadAsStringAsync().Result;
                    movimientosModelRespuesta =
JsonConvert.DeserializeObject<List<MovimientosDeStockModel>>(stringContenido);
                    break;
                case HttpStatusCode.NoContent:
                    TempData["Information"] = "No hay paginas para mostrar";
                    MovimientosPorTipoModelTemp = null;
                    ActualPage = 1;
                    throw new Exception();
                case HttpStatusCode.Unauthorized:
                    TempData["ErrorMessage"] = "Debe ser un encargado para ver el
reporte.";
                    MovimientosPorTipoModelTemp = null;
                    ActualPage = 1;
                    throw new Exception();
                case HttpStatusCode.BadRequest:
                    TempData["ErrorMessage"] = "Error en el ingreso de la información.";
                    MovimientosPorTipoModelTemp = null;
                    ActualPage = 1;
                    throw new Exception();
                default:
                    TempData["ErrorMessage"] = "Error desconocido. Contacte a los
bebedores de café y energizantes.";
                    MovimientosPorTipoModelTemp = null;
                    ActualPage = 1;
                    throw new Exception();
            }
            return View("MostrarMovimientosPorTipo", movimientosModelRespuesta);
        }
        catch (Exception)
        {
            return RedirectToAction("MovimientosPorTipo");
```

```
            }
        }
    }

    [HttpPost]
    public IActionResult MovimientosPorTipo(MovimientosPorTipoModel
movimientosPorTipoModel)
    {
        if (HttpContext.Session.GetString("token") == "" ||
HttpContext.Session.GetString("rol") != "ENCARGADO")
        {
            TempData["ErrorMessage"] = "Debe estar logueado para ingresar al sistema.";
            return RedirectToAction("Login", "Usuario");
        }

        MovimientosPorTipoModelTemp ??= movimientosPorTipoModel;

        return RedirectToAction();
    }

    [HttpGet]
    public IActionResult NextMovimientosPorTipo()
    {
        try
        {
            ActualPage++;
            if (ActualPage < 1)
            {
                ActualPage = 1;
                TempData["Information"] = "No hay mas paginas para mostrar";
            }

            return RedirectToAction("MovimientosPorTipo");
        }
        catch (Exception)
        {
            TempData["ErrorMessage"] = "404 Not found";
            return RedirectToAction("Index", "Home");
        }
    }

    [HttpGet]
    public IActionResult PreviuosMovimientosPorTipo()
    {
        try
        {
            ActualPage--;
            if (ActualPage < 1)
```

```
            {
                ActualPage = 1;
                TempData["Information"] = "No hay mas paginas para mostrar";
            }

            return RedirectToAction("MovimientosPorTipo");
        }
        catch (Exception)
        {
            TempData["ErrorMessage"] = "404 Not found";
            return RedirectToAction("Index", "Home");
        }
    }

    [HttpGet]
    public IActionResult MostrarMovimientosPorTipo()
    {
        if (HttpContext.Session.GetString("token") == "" ||
HttpContext.Session.GetString("rol") != "ENCARGADO")
        {
            TempData["ErrorMessage"] = "Debe estar logueado para ingresar al sistema.";
            return RedirectToAction("Login", "Usuario");
        }
        return View();
    }



    [HttpGet]
    public IActionResult MovimientosResumen()
    {
        if (HttpContext.Session.GetString("rol") != "ENCARGADO")
        {
            TempData["ErrorMessage"] = "Debe estar logueado para ingresar al sistema.";
            return RedirectToAction("Login", "Usuario");
        }
        try
        {
            IEnumerable<MovimientosResumenRespuestaModel>? resumenModel;

            HttpClient cliente = new HttpClient();
            string? token = HttpContext.Session.GetString("token");
            if (String.IsNullOrEmpty(token))
            {
                TempData["ErrorMessage"] = "Debe estar logueado para ingresar al sistema.";
                RedirectToAction("Login", "Usuario");
            }
            cliente.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue(
```

```
                "Bearer",
                token
            );
            string url = "http://localhost:5226/api/Reportes/GetResumenMovimientos";
            HttpResponseMessage respuesta = cliente.GetAsync(url).Result;

            switch (respuesta.StatusCode)
            {
                case HttpStatusCode.OK:
                    HttpContent contenido = respuesta.Content;
                    string stringContenido = contenido.ReadAsStringAsync().Result;
                    resumenModel =
JsonConvert.DeserializeObject<List<MovimientosResumenRespuestaModel>>(stringConten
ido);
                    break;
                case HttpStatusCode.NoContent:
                    TempData["Information"] = "No hay nada que ver aquí.";
                    throw new Exception();
                case HttpStatusCode.Unauthorized:
                    TempData["ErrorMessage"] = "Debe ser un encargado para ver el reporte.";
                    throw new Exception();
                case HttpStatusCode.BadRequest:
                    TempData["ErrorMessage"] = "Error en el ingreso de la información.";
                    throw new Exception();
                default:
                    TempData["ErrorMessage"] = "Error desconocido. Contacte a los bebedores
de café y energizantes.";
                    throw new Exception();
            }
            return View("MostrarMovimientosResumen", resumenModel);
        }
        catch (Exception)
        {
            return RedirectToAction("MovimientosResumen");
        }
    }

    [HttpGet]
    public IActionResult MostrarMovimientosResumen()
    {
        if (HttpContext.Session.GetString("token") == "" ||
HttpContext.Session.GetString("rol") != "ENCARGADO")
        {
            TempData["ErrorMessage"] = "Debe estar logueado para ingresar al sistema.";
            return RedirectToAction("Login", "Usuario");
        }
        return View();
    }
```

```
        public MovimientosPorTipoModel cargarSelects()
        {
            IEnumerable<TipoMovimientoModel>? tiposModel = [];
            HttpClient httpClient = new HttpClient();
            string tiposUrl = "http://localhost:5226/api/TipoMovimiento";
            HttpResponseMessage response = httpClient.GetAsync(tiposUrl).Result;

            if (response.IsSuccessStatusCode)
            {
                HttpContent httpContent = response.Content;
                string responseDeMovimiento = httpContent.ReadAsStringAsync().Result;
                tiposModel =
JsonConvert.DeserializeObject<IEnumerable<TipoMovimientoModel>>(responseDeMovimie
nto);
            }

            MovimientosPorTipoModel movimientosPorTipoModel = new
MovimientosPorTipoModel();
            movimientosPorTipoModel.TipoMovimientos = tiposModel;

            IEnumerable<ArticuloModel>? articuloModel = [];
            HttpClient httpClient2 = new HttpClient();
            string tiposUrl2 = "http://localhost:5226/api/Articulo";
            HttpResponseMessage response2 = httpClient2.GetAsync(tiposUrl2).Result;


            if (response2.IsSuccessStatusCode)
            {
                HttpContent httpContent2 = response2.Content;
                string responseDeArticulos = httpContent2.ReadAsStringAsync().Result;
                articuloModel =
JsonConvert.DeserializeObject<IEnumerable<ArticuloModel>>(responseDeArticulos);
            }

            movimientosPorTipoModel.Articulos = articuloModel;
            return movimientosPorTipoModel;
        }

    }
}

namespace WebApplication1.Controllers
{
    public class UsuarioController : Controller
    {

        public ActionResult Login()
```

```
        {
            return View();
        }

        //Debo acostumbrarme a bloquear la laptop sino personas pueden hacer cosas
molestas en mi pc.
        [HttpPost]
        public ActionResult Login(string email, string contraseña)
        {

            try
            {
                HttpClient cliente = new HttpClient();
                Uri uri = new Uri("http://localhost:5226/api/Login/login");
                HttpRequestMessage solicitud = new HttpRequestMessage(HttpMethod.Post, uri);
                LoginUserModel model = new LoginUserModel()
                {
                    Email = email,
                    Contrasenia = contraseña
                };
                string json = JsonConvert.SerializeObject(model);
                HttpContent contenido = new StringContent(json, Encoding.UTF8,
"application/json");
                solicitud.Content = contenido;

                Task<HttpResponseMessage> respuesta = cliente.SendAsync(solicitud);
                respuesta.Wait();

                if (respuesta.Result.StatusCode == HttpStatusCode.OK)
                {
                    var objetoComoTexto = respuesta.Result.Content.ReadAsStringAsync().Result;
                    TokenResponseModel? user =
JsonConvert.DeserializeObject<TokenResponseModel>(objetoComoTexto);
                    HttpContext.Session.SetString("email", email);
                    HttpContext.Session.SetString("token", user.Token);
                    HttpContext.Session.SetInt32("id", user.Id);
                    //No llega, cae en unAuthorized
                    switch (user.Rol)
                    {
                        case "1":
                            HttpContext.Session.SetString("rol", "ADMINISTRADOR");
                            break;
                        case "2":
                            HttpContext.Session.SetString("rol", "USUSARIO");
                            break;
                        case "3":
                            HttpContext.Session.SetString("rol", "ENCARGADO");
                            break;
```

```
                default:
                    HttpContext.Session.SetString("rol", "INVALIDO");
                    break;
                }

                if (HttpContext.Session.GetString("rol") != "ENCARGADO")
                {
                    TempData["ErrorMessage"] = "Debe ser un Encargado para ingresar al sitio";
                    return RedirectToAction("Login");
                }
                else
                {
                    return RedirectToAction("Index", "Home");
                }
            }
            else if (respuesta.Result.StatusCode == HttpStatusCode.Unauthorized)
            {
                //TODO: Debe haber otra forma de recuperar el mensaje en la respuesta
                TempData["ErrorMessage"] = "Usuario y/o contraseña incorrectos";
                return RedirectToAction("Login");
            }
            else
            {
                //TODO: Debe haber otra forma de recuperar el mensaje en la respuesta
                TempData["ErrorMessage"] = "Error desconocido contacte a los bebedores de
cafe y energizantes bien facheros";
                return RedirectToAction("Login");
            }
        }
        catch (Exception ex)
        {
            TempData["ErrorMessage"] = ex.Message;
            return RedirectToAction("Login");
        }
    }

    public ActionResult Logout()
    {
        HttpContext.Session.Remove("email");
        HttpContext.Session.Remove("token");
        HttpContext.Session.Remove("rol");
        HttpContext.Session.Remove("id");
        return RedirectToAction("Login");
    }
  }
}
namespace WebApplication1.Models.ModelsArticulos
{
```

```
    public class ArticuloModel
    {
        public int Id { get; set; }
        public string Nombre { get; set; }
        public decimal Precio { get; set; }
    }
}


namespace WebApplication1.Models.ModelsMovimientosDeStock
{
    public class MovimientosDeStockModel
    {
        [DisplayName("Articulo")]
        public int ArticuloId { get; set; }
        public string ArticuloName { get; set; }

        public int Cantidad { get; set; }

        [DisplayName("TipoMovimiento")]
        public int TipoMovimientoId { get; set; }

        public int? UsuarioId { get; set; }

        public IEnumerable<TipoMovimientoModel> TipoMovimientos { get; set; }

        public IEnumerable<ArticuloModel> Articulos { get; set; }

    }
}
namespace WebApplication1.Models.ModelsReportes
{
    public class MovimientosEntreFechasModel
    {
        public DateTime Desde {  get; set; }
        public DateTime Hasta { get; set; }
        public int Page { get; set; }
    }
}

namespace WebApplication1.Models.ModelsReportes
{
    public class MovimientosEntreFechasRespuestaModel
    {
        public string Nombre { get; set; }
        public string Descripcion { get; set; }
        public long Codigo { get; set; }
        public double PrecioVenta { get; set; }
```

```
    }
}


namespace WebApplication1.Models.ModelsReportes
{
    public class MovimientosPorTipoModel
    {
        public int ArticuloId { get; set; }
        public int TipoMovimientoId { get; set; }
        public int UsuarioId { get; set; }
        public int PageNumber { get; set; }
        public IEnumerable<TipoMovimientoModel> TipoMovimientos { get; set; }
        public IEnumerable<ArticuloModel> Articulos { get; set; }
    }
}

namespace WebApplication1.Models.ModelsReportes
{
    public class MovimientosResumenRespuestaModel
    {
            public int Anio { get; set; }
            public List<ResumenTipoMovimientoModel> TipoMovimiento { get; set; }
            public int TotalAnio {  get; set; }
    }
}
namespace WebApplication1.Models.ModelsTipoMovimiento
{
    public class TipoMovimientoModel
    {
        public int Id { get; set; }
        public string Nombre { get; set; }
    }
}

namespace WebApplication1.Models
{
    public class ErrorViewModel
    {
        public string? RequestId { get; set; }

        public bool ShowRequestId => !string.IsNullOrEmpty(RequestId);
    }
}

namespace WebApplication1.Models
{
    public class LoginUserModel
```

```
    {
        public string Email { get; set; }
        public string Contrasenia { get; set; }
    }
}

namespace WebApplication1.Models
{
    public class NombreTipoMovimientoModel
    {
        public int Id { get; set; }
        public string Nombre { get; set; }
        public Boolean Reduccion { get; set; }
    }
}

namespace WebApplication1.Models
{
    public class ResumenTipoMovimientoModel
    {
        public NombreTipoMovimientoModel NombreTipo { get; set; }
        public int Cantidad { get; set; }
    }
}

namespace WebApplication1.Models
{
    public class TokenResponseModel
    {
        public string Token { get; set; }
        public LoginUserModel Usuario { get; set; }
        public string Rol {  get; set; }
        public int Id { get; set; }
    }
}

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.
builder.Services.AddControllersWithViews();

builder.Services.AddSession(options =>
{
    options.IdleTimeout = TimeSpan.FromSeconds(300);
    options.Cookie.HttpOnly = true;
    options.Cookie.IsEssential = true;
});
```

```
var app = builder.Build();

// Configure the HTTP request pipeline.
if (!app.Environment.IsDevelopment())
{
    app.UseExceptionHandler("/Home/Error");
    // The default HSTS value is 30 days. You may want to change this for production
scenarios, see https://aka.ms/aspnetcore-hsts.
    app.UseHsts();
}

app.UseHttpsRedirection();
app.UseStaticFiles();

app.UseRouting();
app.UseSession();

app.UseAuthorization();

app.MapControllerRoute(
    name: "default",
    pattern: "{controller=Home}/{action=Index}/{id?}");

app.Run();
```

Scripts:

https://chatgpt.com/share/aab1ed9a-7408-4a7d-b360-d51539f6e5f3

```
SET IDENTITY_INSERT [dbo].[Articulos] ON
INSERT INTO [dbo].[Articulos] ([Id], [Nombre], [Descripcion], [Codigo], [PrecioVenta],
[Stock]) VALUES (1, N'Hojas A4', N'Hojas A4 para impresora', 8572019346852,
CAST(250.00 AS Decimal(18, 2)), 150)
INSERT INTO [dbo].[Articulos] ([Id], [Nombre], [Descripcion], [Codigo], [PrecioVenta],
[Stock]) VALUES (2, N'Lapices Faber Castell', N'Lapices Colores x24', 1234567891234,
CAST(150.00 AS Decimal(18, 2)), 50)
INSERT INTO [dbo].[Articulos] ([Id], [Nombre], [Descripcion], [Codigo], [PrecioVenta],
[Stock]) VALUES (3, N'Lapicera BIC', N'Lapicera BIC azul', 7932481674023, CAST(99.00
AS Decimal(18, 2)), 100)
INSERT INTO [dbo].[Articulos] ([Id], [Nombre], [Descripcion], [Codigo], [PrecioVenta],
[Stock]) VALUES (4, N'Cuaderno rayado', N'Cuaderno rayado tapa dura', 5127398462058,
CAST(139.00 AS Decimal(18, 2)), 100)
```

INSERT INTO [dbo].[Articulos] ([Id], [Nombre], [Descripcion], [Codigo], [PrecioVenta], [Stock]) VALUES (5, N'Sylvanian Family', N'Sylvanian Family blind box', 9273816405928, CAST(459.00 AS Decimal(18, 2)), 30)
INSERT INTO [dbo].[Articulos] ([Id], [Nombre], [Descripcion], [Codigo], [PrecioVenta], [Stock]) VALUES (6, N'Lápices HB', N'Lápices de grafito estándar', 4927610358274, CAST(50.00 AS Decimal(18, 2)), 200)
INSERT INTO [dbo].[Articulos] ([Id], [Nombre], [Descripcion], [Codigo], [PrecioVenta], [Stock]) VALUES (7, N'Marcadores de colores', N'Juego de marcadores de colores surtidos', 7518296403728, CAST(80.00 AS Decimal(18, 2)), 50)
INSERT INTO [dbo].[Articulos] ([Id], [Nombre], [Descripcion], [Codigo], [PrecioVenta], [Stock]) VALUES (8, N'Cinta adhesiva', N'Rollo de cinta adhesiva transparente', 3654928107365, CAST(30.00 AS Decimal(18, 2)), 300)
INSERT INTO [dbo].[Articulos] ([Id], [Nombre], [Descripcion], [Codigo], [PrecioVenta], [Stock]) VALUES (9, N'Calculadora básica', N'Calculadora de funciones básicas', 9246738102584, CAST(200.00 AS Decimal(18, 2)), 80)
INSERT INTO [dbo].[Articulos] ([Id], [Nombre], [Descripcion], [Codigo], [PrecioVenta], [Stock]) VALUES (10, N'Resaltadores fluorescentes', N'Juego de resaltadores de colores', 8367291047523, CAST(120.00 AS Decimal(18, 2)), 75)
INSERT INTO [dbo].[Articulos] ([Id], [Nombre], [Descripcion], [Codigo], [PrecioVenta], [Stock]) VALUES (11, N'Goma de borrar', N'Goma de borrar blanca pequeña', 1357924680246, CAST(0.99 AS Decimal(18, 2)), 120)
INSERT INTO [dbo].[Articulos] ([Id], [Nombre], [Descripcion], [Codigo], [PrecioVenta], [Stock]) VALUES (12, N'Calculadora científica', N'Calculadora científica de bolsillo', 2468013579246, CAST(19.99 AS Decimal(18, 2)), 30)
INSERT INTO [dbo].[Articulos] ([Id], [Nombre], [Descripcion], [Codigo], [PrecioVenta], [Stock]) VALUES (13, N'Tijeras escolares', N'Tijeras escolares con mango de plástico', 3579246801357, CAST(2.49 AS Decimal(18, 2)), 80)
INSERT INTO [dbo].[Articulos] ([Id], [Nombre], [Descripcion], [Codigo], [PrecioVenta], [Stock]) VALUES (14, N'Agenda semanal', N'Agenda semanal con tapa dura', 4680246913579, CAST(7.99 AS Decimal(18, 2)), 60)
INSERT INTO [dbo].[Articulos] ([Id], [Nombre], [Descripcion], [Codigo], [PrecioVenta], [Stock]) VALUES (15, N'Borradores', N'Paquete de 4 borradores de colores surtidos', 5792468013579, CAST(1.25 AS Decimal(18, 2)), 150)
INSERT INTO [dbo].[Articulos] ([Id], [Nombre], [Descripcion], [Codigo], [PrecioVenta], [Stock]) VALUES (16, N'Carpetas de archivo', N'Juego de 10 carpetas de archivo tamaño carta', 6801357924680, CAST(9.75 AS Decimal(18, 2)), 40)
INSERT INTO [dbo].[Articulos] ([Id], [Nombre], [Descripcion], [Codigo], [PrecioVenta], [Stock]) VALUES (17, N'Lápiz de color', N'Estuche de 12 lápices de colores', 7913579246801, CAST(4.99 AS Decimal(18, 2)), 100)
INSERT INTO [dbo].[Articulos] ([Id], [Nombre], [Descripcion], [Codigo], [PrecioVenta], [Stock]) VALUES (18, N'Corrector líquido', N'Corrector líquido de precisión', 8024691357924, CAST(1.79 AS Decimal(18, 2)), 200)
INSERT INTO [dbo].[Articulos] ([Id], [Nombre], [Descripcion], [Codigo], [PrecioVenta], [Stock]) VALUES (19, N'Libreta de notas', N'Libreta de notas tamaño bolsillo', 9135792468024, CAST(3.49 AS Decimal(18, 2)), 120)
INSERT INTO [dbo].[Articulos] ([Id], [Nombre], [Descripcion], [Codigo], [PrecioVenta], [Stock]) VALUES (20, N'Pegamento en barra', N'Barra de pegamento lavable', 246913579246, CAST(1.50 AS Decimal(18, 2)), 180)

```
INSERT INTO [dbo].[Articulos] ([Id], [Nombre], [Descripcion], [Codigo], [PrecioVenta], [Stock]) VALUES (21, N'Post-it', N'Bloque de notas adhesivas Post-it', 2469135792468, CAST(2.25 AS Decimal(18, 2)), 250)
INSERT INTO [dbo].[Articulos] ([Id], [Nombre], [Descripcion], [Codigo], [PrecioVenta], [Stock]) VALUES (22, N'Rotuladores permanentes', N'Juego de 4 rotuladores permanentes de colores', 5792468024691, CAST(5.99 AS Decimal(18, 2)), 80)
INSERT INTO [dbo].[Articulos] ([Id], [Nombre], [Descripcion], [Codigo], [PrecioVenta], [Stock]) VALUES (23, N'Regla de plástico', N'Regla de plástico transparente de 30 cm', 8024691357924, CAST(1.99 AS Decimal(18, 2)), 150)
INSERT INTO [dbo].[Articulos] ([Id], [Nombre], [Descripcion], [Codigo], [PrecioVenta], [Stock]) VALUES (24, N'Papel de colores', N'Paquete de 100 hojas de papel de colores surtidos', 1357924680246, CAST(6.75 AS Decimal(18, 2)), 100)
INSERT INTO [dbo].[Articulos] ([Id], [Nombre], [Descripcion], [Codigo], [PrecioVenta], [Stock]) VALUES (25, N'Folder manila', N'Folder tamaño oficio color manila', 4680246913579, CAST(1.49 AS Decimal(18, 2)), 200)
INSERT INTO [dbo].[Articulos] ([Id], [Nombre], [Descripcion], [Codigo], [PrecioVenta], [Stock]) VALUES (26, N'Clips metálicos', N'Caja de 100 clips metálicos tamaño pequeño', 6913579246801, CAST(2.29 AS Decimal(18, 2)), 300)
INSERT INTO [dbo].[Articulos] ([Id], [Nombre], [Descripcion], [Codigo], [PrecioVenta], [Stock]) VALUES (27, N'Carpeta escolar', N'Carpeta escolar con bolsillos internos', 8024691357924, CAST(3.99 AS Decimal(18, 2)), 120)
INSERT INTO [dbo].[Articulos] ([Id], [Nombre], [Descripcion], [Codigo], [PrecioVenta], [Stock]) VALUES (28, N'Bolígrafos', N'Paquete de 10 bolígrafos azules', 9135792468024, CAST(4.50 AS Decimal(18, 2)), 150)
INSERT INTO [dbo].[Articulos] ([Id], [Nombre], [Descripcion], [Codigo], [PrecioVenta], [Stock]) VALUES (29, N'Subrayadores', N'Set de 6 subrayadores de colores surtidos', 246913579246, CAST(3.25 AS Decimal(18, 2)), 180)
INSERT INTO [dbo].[Articulos] ([Id], [Nombre], [Descripcion], [Codigo], [PrecioVenta], [Stock]) VALUES (30, N'Agenda diaria', N'Agenda diaria con tapa flexible', 2469135792468, CAST(8.99 AS Decimal(18, 2)), 80)
INSERT INTO [dbo].[Articulos] ([Id], [Nombre], [Descripcion], [Codigo], [PrecioVenta], [Stock]) VALUES (31, N'Lápices de grafito', N'Estuche de 24 lápices de grafito HB', 5792468024691, CAST(7.49 AS Decimal(18, 2)), 100)
INSERT INTO [dbo].[Articulos] ([Id], [Nombre], [Descripcion], [Codigo], [PrecioVenta], [Stock]) VALUES (32, N'Carpeta colgante', N'Carpeta colgante para archivo tamaño carta', 8024691357924, CAST(2.99 AS Decimal(18, 2)), 120)
INSERT INTO [dbo].[Articulos] ([Id], [Nombre], [Descripcion], [Codigo], [PrecioVenta], [Stock]) VALUES (33, N'Pizarra blanca', N'Pizarra blanca magnética de 60x90 cm', 1357924680246, CAST(19.99 AS Decimal(18, 2)), 50)
INSERT INTO [dbo].[Articulos] ([Id], [Nombre], [Descripcion], [Codigo], [PrecioVenta], [Stock]) VALUES (34, N'Tinta para sellos', N'Botella de tinta para sellos de 50 ml', 4680246913579, CAST(3.99 AS Decimal(18, 2)), 100)
INSERT INTO [dbo].[Articulos] ([Id], [Nombre], [Descripcion], [Codigo], [PrecioVenta], [Stock]) VALUES (35, N'Cinta correctora', N'Cinta correctora desechable', 6913579246801, CAST(1.79 AS Decimal(18, 2)), 200)
INSERT INTO [dbo].[Articulos] ([Id], [Nombre], [Descripcion], [Codigo], [PrecioVenta], [Stock]) VALUES (36, N'Folder plastificado', N'Folder plastificado con cierre de velcro', 8024691357924, CAST(2.49 AS Decimal(18, 2)), 150)
```

INSERT INTO [dbo].[Articulos] ([Id], [Nombre], [Descripcion], [Codigo], [PrecioVenta], [Stock]) VALUES (37, N'Plumones para pizarra', N'Juego de 8 plumones para pizarra blanca', 9135792468024, CAST(6.99 AS Decimal(18, 2)), 120)
INSERT INTO [dbo].[Articulos] ([Id], [Nombre], [Descripcion], [Codigo], [PrecioVenta], [Stock]) VALUES (38, N'Carpeta de argollas', N'Carpeta de argollas tamaño carta', 246913579246, CAST(4.25 AS Decimal(18, 2)), 180)
INSERT INTO [dbo].[Articulos] ([Id], [Nombre], [Descripcion], [Codigo], [PrecioVenta], [Stock]) VALUES (39, N'Papel milimetrado', N'Block de papel milimetrado tamaño carta', 2469135792468, CAST(5.50 AS Decimal(18, 2)), 100)
INSERT INTO [dbo].[Articulos] ([Id], [Nombre], [Descripcion], [Codigo], [PrecioVenta], [Stock]) VALUES (40, N'Perforadora', N'Perforadora de 3 agujeros para papel', 5792468024691, CAST(9.99 AS Decimal(18, 2)), 60)
INSERT INTO [dbo].[Articulos] ([Id], [Nombre], [Descripcion], [Codigo], [PrecioVenta], [Stock]) VALUES (41, N'Marcadores permanentes', N'Juego de 6 marcadores permanentes de colores', 8024691357924, CAST(4.99 AS Decimal(18, 2)), 80)
INSERT INTO [dbo].[Articulos] ([Id], [Nombre], [Descripcion], [Codigo], [PrecioVenta], [Stock]) VALUES (42, N'Papel fotográfico', N'Papel fotográfico brillante para impresora', 1357924680246, CAST(12.99 AS Decimal(18, 2)), 40)
INSERT INTO [dbo].[Articulos] ([Id], [Nombre], [Descripcion], [Codigo], [PrecioVenta], [Stock]) VALUES (43, N'Grapadora', N'Grapadora metálica de uso pesado', 4680246913579, CAST(6.49 AS Decimal(18, 2)), 100)
INSERT INTO [dbo].[Articulos] ([Id], [Nombre], [Descripcion], [Codigo], [PrecioVenta], [Stock]) VALUES (44, N'Cartulinas', N'Paquete de 50 cartulinas tamaño carta', 6913579246801, CAST(7.99 AS Decimal(18, 2)), 120)
INSERT INTO [dbo].[Articulos] ([Id], [Nombre], [Descripcion], [Codigo], [PrecioVenta], [Stock]) VALUES (45, N'Etiquetas adhesivas', N'Rollo de 100 etiquetas adhesivas blancas', 8024691357924, CAST(2.99 AS Decimal(18, 2)), 200)
INSERT INTO [dbo].[Articulos] ([Id], [Nombre], [Descripcion], [Codigo], [PrecioVenta], [Stock]) VALUES (46, N'Compás', N'Compás de metal con estuche', 9135792468024, CAST(3.75 AS Decimal(18, 2)), 150)
INSERT INTO [dbo].[Articulos] ([Id], [Nombre], [Descripcion], [Codigo], [PrecioVenta], [Stock]) VALUES (47, N'Portaminas', N'Portaminas automático de 0.7 mm', 246913579246, CAST(1.99 AS Decimal(18, 2)), 250)
INSERT INTO [dbo].[Articulos] ([Id], [Nombre], [Descripcion], [Codigo], [PrecioVenta], [Stock]) VALUES (48, N'Papel vegetal', N'Block de papel vegetal tamaño carta', 2469135792468, CAST(8.25 AS Decimal(18, 2)), 80)
INSERT INTO [dbo].[Articulos] ([Id], [Nombre], [Descripcion], [Codigo], [PrecioVenta], [Stock]) VALUES (49, N'Caja de archivo', N'Caja de archivo de cartón tamaño oficio', 5792468024691, CAST(5.49 AS Decimal(18, 2)), 120)
INSERT INTO [dbo].[Articulos] ([Id], [Nombre], [Descripcion], [Codigo], [PrecioVenta], [Stock]) VALUES (50, N'Borrador de pizarra', N'Borrador de pizarra magnética', 8024691357924, CAST(1.49 AS Decimal(18, 2)), 200)
INSERT INTO [dbo].[Articulos] ([Id], [Nombre], [Descripcion], [Codigo], [PrecioVenta], [Stock]) VALUES (51, N'Carpeta ejecutiva', N'Carpeta ejecutiva con cierre magnético', 1357924680246, CAST(12.99 AS Decimal(18, 2)), 100)
INSERT INTO [dbo].[Articulos] ([Id], [Nombre], [Descripcion], [Codigo], [PrecioVenta], [Stock]) VALUES (52, N'Pegatinas', N'Juego de 200 pegatinas variadas', 4680246913579, CAST(3.25 AS Decimal(18, 2)), 150)

```
INSERT INTO [dbo].[Articulos] ([Id], [Nombre], [Descripcion], [Codigo], [PrecioVenta], [Stock]) VALUES (53, N'Tijeras de precisión', N'Tijeras de precisión para manualidades', 6913579246801, CAST(4.99 AS Decimal(18, 2)), 100)
INSERT INTO [dbo].[Articulos] ([Id], [Nombre], [Descripcion], [Codigo], [PrecioVenta], [Stock]) VALUES (54, N'Notas adhesivas', N'Bloc de notas adhesivas de diferentes colores', 8024691357924, CAST(1.99 AS Decimal(18, 2)), 300)
INSERT INTO [dbo].[Articulos] ([Id], [Nombre], [Descripcion], [Codigo], [PrecioVenta], [Stock]) VALUES (55, N'Plumillas', N'Set de 5 plumillas para caligrafía', 9135792468024, CAST(2.75 AS Decimal(18, 2)), 200)
INSERT INTO [dbo].[Articulos] ([Id], [Nombre], [Descripcion], [Codigo], [PrecioVenta], [Stock]) VALUES (56, N'Pizarra de corcho', N'Pizarra de corcho con marco de madera', 246913579246, CAST(6.49 AS Decimal(18, 2)), 80)
INSERT INTO [dbo].[Articulos] ([Id], [Nombre], [Descripcion], [Codigo], [PrecioVenta], [Stock]) VALUES (57, N'Carpeta expandible', N'Carpeta expandible tamaño carta', 2469135792468, CAST(3.99 AS Decimal(18, 2)), 120)
INSERT INTO [dbo].[Articulos] ([Id], [Nombre], [Descripcion], [Codigo], [PrecioVenta], [Stock]) VALUES (58, N'Plumones de tiza líquida', N'Set de 8 plumones de tiza líquida', 5792468024691, CAST(7.99 AS Decimal(18, 2)), 100)
INSERT INTO [dbo].[Articulos] ([Id], [Nombre], [Descripcion], [Codigo], [PrecioVenta], [Stock]) VALUES (59, N'Carpetas colgantes', N'Juego de 5 carpetas colgantes de plástico', 8024691357924, CAST(5.25 AS Decimal(18, 2)), 150)
INSERT INTO [dbo].[Articulos] ([Id], [Nombre], [Descripcion], [Codigo], [PrecioVenta], [Stock]) VALUES (60, N'Bolígrafos de gel', N'Paquete de 6 bolígrafos de gel color negro', 1357924680246, CAST(4.99 AS Decimal(18, 2)), 180)
INSERT INTO [dbo].[Articulos] ([Id], [Nombre], [Descripcion], [Codigo], [PrecioVenta], [Stock]) VALUES (61, N'Borrador de tinta', N'Borrador de tinta para bolígrafos', 4680246913579, CAST(0.99 AS Decimal(18, 2)), 250)
INSERT INTO [dbo].[Articulos] ([Id], [Nombre], [Descripcion], [Codigo], [PrecioVenta], [Stock]) VALUES (62, N'Portafolios', N'Portafolios de plástico transparente tamaño carta', 6913579246801, CAST(2.49 AS Decimal(18, 2)), 200)
INSERT INTO [dbo].[Articulos] ([Id], [Nombre], [Descripcion], [Codigo], [PrecioVenta], [Stock]) VALUES (63, N'Folios blancos', N'Paquete de 500 folios blancos tamaño carta', 8024691357924, CAST(11.99 AS Decimal(18, 2)), 100)
INSERT INTO [dbo].[Articulos] ([Id], [Nombre], [Descripcion], [Codigo], [PrecioVenta], [Stock]) VALUES (64, N'Cinta de embalar', N'Rollo de cinta de embalar transparente', 9135792468024, CAST(1.75 AS Decimal(18, 2)), 150)
INSERT INTO [dbo].[Articulos] ([Id], [Nombre], [Descripcion], [Codigo], [PrecioVenta], [Stock]) VALUES (65, N'Bloc de notas', N'Bloc de notas de papel reciclado', 246913579246, CAST(2.99 AS Decimal(18, 2)), 200)
INSERT INTO [dbo].[Articulos] ([Id], [Nombre], [Descripcion], [Codigo], [PrecioVenta], [Stock]) VALUES (66, N'Punzones', N'Punzón metálico para perforar papel', 2469135792468, CAST(3.49 AS Decimal(18, 2)), 120)
INSERT INTO [dbo].[Articulos] ([Id], [Nombre], [Descripcion], [Codigo], [PrecioVenta], [Stock]) VALUES (67, N'Cinta adhesiva doble cara', N'Rollo de cinta adhesiva doble cara', 5792468024691, CAST(2.99 AS Decimal(18, 2)), 180)
INSERT INTO [dbo].[Articulos] ([Id], [Nombre], [Descripcion], [Codigo], [PrecioVenta], [Stock]) VALUES (68, N'Gomas para pizarrón', N'Set de 4 gomas para pizarrón', 8024691357924, CAST(4.25 AS Decimal(18, 2)), 100)
```

INSERT INTO [dbo].[Articulos] ([Id], [Nombre], [Descripcion], [Codigo], [PrecioVenta], [Stock]) VALUES (69, N'Plumón para pizarrón', N'Plumón para pizarrón blanco', 1357924680246, CAST(1.49 AS Decimal(18, 2)), 200)
INSERT INTO [dbo].[Articulos] ([Id], [Nombre], [Descripcion], [Codigo], [PrecioVenta], [Stock]) VALUES (70, N'Separadores', N'Set de 10 separadores de colores para carpeta', 4680246913579, CAST(3.99 AS Decimal(18, 2)), 150)
INSERT INTO [dbo].[Articulos] ([Id], [Nombre], [Descripcion], [Codigo], [PrecioVenta], [Stock]) VALUES (71, N'Borrador de tiza', N'Borrador de tiza para pizarra', 6913579246801, CAST(0.99 AS Decimal(18, 2)), 250)
INSERT INTO [dbo].[Articulos] ([Id], [Nombre], [Descripcion], [Codigo], [PrecioVenta], [Stock]) VALUES (72, N'Pegamento líquido', N'Botella de pegamento líquido para manualidades', 8024691357924, CAST(2.99 AS Decimal(18, 2)), 180)
INSERT INTO [dbo].[Articulos] ([Id], [Nombre], [Descripcion], [Codigo], [PrecioVenta], [Stock]) VALUES (73, N'Clips de papel', N'Caja de 200 clips de papel tamaño pequeño', 9135792468024, CAST(1.25 AS Decimal(18, 2)), 300)
INSERT INTO [dbo].[Articulos] ([Id], [Nombre], [Descripcion], [Codigo], [PrecioVenta], [Stock]) VALUES (74, N'Papel de regalo', N'Rollo de papel de regalo con diseño variado', 246913579246, CAST(5.49 AS Decimal(18, 2)), 120)
INSERT INTO [dbo].[Articulos] ([Id], [Nombre], [Descripcion], [Codigo], [PrecioVenta], [Stock]) VALUES (75, N'Portaminas 0.5 mm', N'Portaminas de 0.5 mm con goma de borrar', 2469135792468, CAST(1.99 AS Decimal(18, 2)), 200)
INSERT INTO [dbo].[Articulos] ([Id], [Nombre], [Descripcion], [Codigo], [PrecioVenta], [Stock]) VALUES (76, N'Carpeta de presentación', N'Carpeta de presentación tamaño oficio', 5792468024691, CAST(2.75 AS Decimal(18, 2)), 180)
INSERT INTO [dbo].[Articulos] ([Id], [Nombre], [Descripcion], [Codigo], [PrecioVenta], [Stock]) VALUES (77, N'Cubiertas plásticas', N'Paquete de 50 cubiertas plásticas transparentes tamaño carta', 8024691357924, CAST(6.99 AS Decimal(18, 2)), 150)
INSERT INTO [dbo].[Articulos] ([Id], [Nombre], [Descripcion], [Codigo], [PrecioVenta], [Stock]) VALUES (78, N'Rotuladores para pizarra', N'Set de 4 rotuladores para pizarra blanca', 1357924680246, CAST(3.49 AS Decimal(18, 2)), 250)
INSERT INTO [dbo].[Articulos] ([Id], [Nombre], [Descripcion], [Codigo], [PrecioVenta], [Stock]) VALUES (79, N'Bloc de dibujo', N'Bloc de dibujo con hojas blancas', 4680246913579, CAST(4.99 AS Decimal(18, 2)), 100)
INSERT INTO [dbo].[Articulos] ([Id], [Nombre], [Descripcion], [Codigo], [PrecioVenta], [Stock]) VALUES (80, N'Carpetas de presentación', N'Juego de 5 carpetas de presentación con bolsillo', 6913579246801, CAST(7.49 AS Decimal(18, 2)), 120)
INSERT INTO [dbo].[Articulos] ([Id], [Nombre], [Descripcion], [Codigo], [PrecioVenta], [Stock]) VALUES (81, N'Papel carbón', N'Paquete de 10 hojas de papel carbón', 8024691357924, CAST(1.99 AS Decimal(18, 2)), 200)
INSERT INTO [dbo].[Articulos] ([Id], [Nombre], [Descripcion], [Codigo], [PrecioVenta], [Stock]) VALUES (82, N'Goma eva', N'Plancha de goma eva de colores', 9135792468024, CAST(3.25 AS Decimal(18, 2)), 150)
INSERT INTO [dbo].[Articulos] ([Id], [Nombre], [Descripcion], [Codigo], [PrecioVenta], [Stock]) VALUES (83, N'Cartulina brillante', N'Paquete de 20 cartulinas brillantes tamaño carta', 246913579246, CAST(8.99 AS Decimal(18, 2)), 80)
INSERT INTO [dbo].[Articulos] ([Id], [Nombre], [Descripcion], [Codigo], [PrecioVenta], [Stock]) VALUES (84, N'Cinta métrica', N'Cinta métrica flexible de 1 metro', 2469135792468, CAST(0.99 AS Decimal(18, 2)), 300)

```
INSERT INTO [dbo].[Articulos] ([Id], [Nombre], [Descripcion], [Codigo], [PrecioVenta], [Stock]) VALUES (85, N'Marcadores para CD/DVD', N'Juego de 4 marcadores para CD/DVD', 5792468024691, CAST(1.49 AS Decimal(18, 2)), 250)
INSERT INTO [dbo].[Articulos] ([Id], [Nombre], [Descripcion], [Codigo], [PrecioVenta], [Stock]) VALUES (86, N'Papel kraft', N'Rollo de papel kraft para embalaje', 8024691357924, CAST(4.99 AS Decimal(18, 2)), 100)
INSERT INTO [dbo].[Articulos] ([Id], [Nombre], [Descripcion], [Codigo], [PrecioVenta], [Stock]) VALUES (87, N'Reglas flexibles', N'Set de 3 reglas flexibles para dibujo', 1357924680246, CAST(3.75 AS Decimal(18, 2)), 150)
INSERT INTO [dbo].[Articulos] ([Id], [Nombre], [Descripcion], [Codigo], [PrecioVenta], [Stock]) VALUES (88, N'Papel bond reciclado', N'Resma de papel bond reciclado tamaño carta', 4680246913579, CAST(10.50 AS Decimal(18, 2)), 120)
INSERT INTO [dbo].[Articulos] ([Id], [Nombre], [Descripcion], [Codigo], [PrecioVenta], [Stock]) VALUES (89, N'Pegamento en spray', N'Bote de pegamento en spray para manualidades', 6913579246801, CAST(7.99 AS Decimal(18, 2)), 80)
INSERT INTO [dbo].[Articulos] ([Id], [Nombre], [Descripcion], [Codigo], [PrecioVenta], [Stock]) VALUES (90, N'Grapas', N'Caja de 1000 grapas para grapadora estándar', 8024691357924, CAST(2.49 AS Decimal(18, 2)), 200)
INSERT INTO [dbo].[Articulos] ([Id], [Nombre], [Descripcion], [Codigo], [PrecioVenta], [Stock]) VALUES (91, N'Bolsas de papel', N'Paquete de 50 bolsas de papel kraft', 9135792468024, CAST(6.99 AS Decimal(18, 2)), 100)
SET IDENTITY_INSERT [dbo].[Articulos] OFF


SET IDENTITY_INSERT [dbo].[Usuarios] ON
INSERT INTO [dbo].[Usuarios] ([Id], [Email], [Nombre], [Apellido], [Contrasenia], [ContraseniaEncriptada], [Rol]) VALUES (1, N'arianagrande@gmail.com', N'Ariana', N'Grande', N'Ariana.1', N'e9266fb12be8a4c55070f079a7f360193efd94fa18927fb2539e1cf537984d63', 3)
INSERT INTO [dbo].[Usuarios] ([Id], [Email], [Nombre], [Apellido], [Contrasenia], [ContraseniaEncriptada], [Rol]) VALUES (2, N'taylorswift@gmail.com', N'Taylor', N'Swift', N'Taylor.1', N'81033beb4effa13f74475f22e4a571b1a9cc17d25d7a51c6655699961a053f64', 2)
INSERT INTO [dbo].[Usuarios] ([Id], [Email], [Nombre], [Apellido], [Contrasenia], [ContraseniaEncriptada], [Rol]) VALUES (3, N'lanadelrey@gmail.com', N'Lana', N'Del Rey', N'Lanita!1', N'c42278f983e92c9ff14c426488b937d2d62740302fac12ee16d8f62eb5078e08', 1)
INSERT INTO [dbo].[Usuarios] ([Id], [Email], [Nombre], [Apellido], [Contrasenia], [ContraseniaEncriptada], [Rol]) VALUES (4, N'charlixcx@gmail.com', N'Charli', N'XCX', N'XCXWorld1!', N'f72c0c5ba91993ddc6a6e1c61599b123c1a2906d9ed70bf16cc8791613a4902e', 1)
INSERT INTO [dbo].[Usuarios] ([Id], [Email], [Nombre], [Apellido], [Contrasenia], [ContraseniaEncriptada], [Rol]) VALUES (5, N'chuu@gmail.com', N'JiWoo', N'Kim', N'underWater!1', N'184676e01731064a395b3038a00e20fa24f8260c771e18fef463d6b900ac4f7a', 1)
INSERT INTO [dbo].[Usuarios] ([Id], [Email], [Nombre], [Apellido], [Contrasenia], [ContraseniaEncriptada], [Rol]) VALUES (6, N'gowon@gmail.com', N'Chaewon', N'Park',
```

N'Gowonnie!2',
N'766f334d6c0ab1e0b97b3d9d5c0e66b73f089bc8f60013c7b88c57f30e758799', 1)
INSERT INTO [dbo].[Usuarios] ([Id], [Email], [Nombre], [Apellido], [Contrasenia], [ContraseniaEncriptada], [Rol]) VALUES (7, N'nayeon@gmail.com', N'Nayeon', N'Im', N'Twice2!', N'12307c7981cfb677c35524c86a0a4520eb0f004be744aced0e6cc84767ce0a81', 1)
INSERT INTO [dbo].[Usuarios] ([Id], [Email], [Nombre], [Apellido], [Contrasenia], [ContraseniaEncriptada], [Rol]) VALUES (8, N'momo@gmail.com', N'Momo', N'Hirai', N'Momo.1',
N'ce19974247e70d17edb6127332b693ee080c3564832359a0dbbcc6044cdd49a1', 1)
INSERT INTO [dbo].[Usuarios] ([Id], [Email], [Nombre], [Apellido], [Contrasenia], [ContraseniaEncriptada], [Rol]) VALUES (9, N'hyeju@gmail.com', N'Hyeju', N'Son', N'Hyeju.1', N'4a5cbe51a4bc1ce2941f66bbf15a58abbb3c543daaa21e804fa9b037e256d446', 1)
INSERT INTO [dbo].[Usuarios] ([Id], [Email], [Nombre], [Apellido], [Contrasenia], [ContraseniaEncriptada], [Rol]) VALUES (10, N'hyunjin@gmail.com', N'Hyunjin', N'Kim', N'Hyunjin.1',
N'730ac593d726450956cb35f90ced4498cf3e03352bdf8b2667109e40341d6337', 1)
INSERT INTO [dbo].[Usuarios] ([Id], [Email], [Nombre], [Apellido], [Contrasenia], [ContraseniaEncriptada], [Rol]) VALUES (11, N'chayanne@gmail.com', N'Chayanne', N'Torero', N'Eltorero123.',
N'192f88f3d534cf59e6fceda8b1f17812de271ed054207aa35c12f1db668085dc', 2)
SET IDENTITY_INSERT [dbo].[Usuarios] OFF

https://chatgpt.com/share/bb54eec5-bc56-493e-b28d-fbb37e580f68

SET IDENTITY_INSERT [dbo].[MovimientoStock] ON
INSERT INTO [dbo].[MovimientoStock] ([Id], [Fecha], [ArticuloId], [Cantidad], [TipoMovimientoId], [UsuarioId]) VALUES (1, N'2024-06-17 02:17:39', 1, 1, 1, 1)
INSERT INTO [dbo].[MovimientoStock] ([Id], [Fecha], [ArticuloId], [Cantidad], [TipoMovimientoId], [UsuarioId]) VALUES (2, N'2024-06-18 02:18:04', 2, 1, 1, 1)
INSERT INTO [dbo].[MovimientoStock] ([Id], [Fecha], [ArticuloId], [Cantidad], [TipoMovimientoId], [UsuarioId]) VALUES (3, N'2024-06-19 02:18:11', 3, 1, 1, 1)
INSERT INTO [dbo].[MovimientoStock] ([Id], [Fecha], [ArticuloId], [Cantidad], [TipoMovimientoId], [UsuarioId]) VALUES (4, N'2024-06-20 04:09:20', 5, 1, 1, 1)
INSERT INTO [dbo].[MovimientoStock] ([Id], [Fecha], [ArticuloId], [Cantidad], [TipoMovimientoId], [UsuarioId]) VALUES (5, N'2024-06-20 11:48:18', 5, 1, 3, 1)
INSERT INTO [dbo].[MovimientoStock] ([Id], [Fecha], [ArticuloId], [Cantidad], [TipoMovimientoId], [UsuarioId]) VALUES (6, N'2024-06-20 12:36:02', 1, 2, 1, 1)
INSERT INTO [dbo].[MovimientoStock] ([Id], [Fecha], [ArticuloId], [Cantidad], [TipoMovimientoId], [UsuarioId]) VALUES (7, N'2024-06-20 12:47:01', 1, 4, 1, 1)
INSERT INTO [dbo].[MovimientoStock] ([Id], [Fecha], [ArticuloId], [Cantidad], [TipoMovimientoId], [UsuarioId]) VALUES (8, N'2024-06-20 12:47:04', 8, 4, 1, 1)
INSERT INTO [dbo].[MovimientoStock] ([Id], [Fecha], [ArticuloId], [Cantidad], [TipoMovimientoId], [UsuarioId]) VALUES (9, N'2024-06-20 12:52:55', 3, 2, 3, 1)
INSERT INTO [dbo].[MovimientoStock] ([Id], [Fecha], [ArticuloId], [Cantidad], [TipoMovimientoId], [UsuarioId]) VALUES (10, N'2024-06-20 12:53:59', 4, 3, 3, 1)

INSERT INTO [dbo].[MovimientoStock] ([Id], [Fecha], [ArticuloId], [Cantidad], [TipoMovimientoId], [UsuarioId]) VALUES (11, N'2024-06-20 12:54:20', 5, 19, 5, 1)
INSERT INTO [dbo].[MovimientoStock] ([Id], [Fecha], [ArticuloId], [Cantidad], [TipoMovimientoId], [UsuarioId]) VALUES (13, N'2024-06-20 13:12:00', 21, 1, 4, 5)
INSERT INTO [dbo].[MovimientoStock] ([Id], [Fecha], [ArticuloId], [Cantidad], [TipoMovimientoId], [UsuarioId]) VALUES (14, N'2024-06-20 13:20:00', 7, 3, 1, 7)
INSERT INTO [dbo].[MovimientoStock] ([Id], [Fecha], [ArticuloId], [Cantidad], [TipoMovimientoId], [UsuarioId]) VALUES (15, N'2024-06-20 13:25:00', 15, 2, 3, 2)
INSERT INTO [dbo].[MovimientoStock] ([Id], [Fecha], [ArticuloId], [Cantidad], [TipoMovimientoId], [UsuarioId]) VALUES (16, N'2024-06-20 13:30:00', 3, 1, 6, 10)
INSERT INTO [dbo].[MovimientoStock] ([Id], [Fecha], [ArticuloId], [Cantidad], [TipoMovimientoId], [UsuarioId]) VALUES (18, N'2024-06-20 13:40:00', 9, 1, 1, 8)
INSERT INTO [dbo].[MovimientoStock] ([Id], [Fecha], [ArticuloId], [Cantidad], [TipoMovimientoId], [UsuarioId]) VALUES (19, N'2024-06-20 13:45:00', 12, 2, 4, 4)
INSERT INTO [dbo].[MovimientoStock] ([Id], [Fecha], [ArticuloId], [Cantidad], [TipoMovimientoId], [UsuarioId]) VALUES (20, N'2024-06-20 13:50:00', 6, 3, 5, 9)
INSERT INTO [dbo].[MovimientoStock] ([Id], [Fecha], [ArticuloId], [Cantidad], [TipoMovimientoId], [UsuarioId]) VALUES (21, N'2024-06-20 14:00:00', 14, 1, 3, 11)
INSERT INTO [dbo].[MovimientoStock] ([Id], [Fecha], [ArticuloId], [Cantidad], [TipoMovimientoId], [UsuarioId]) VALUES (22, N'2024-06-20 14:05:00', 5, 2, 6, 1)
INSERT INTO [dbo].[MovimientoStock] ([Id], [Fecha], [ArticuloId], [Cantidad], [TipoMovimientoId], [UsuarioId]) VALUES (24, N'2024-06-20 14:15:00', 8, 1, 4, 5)
INSERT INTO [dbo].[MovimientoStock] ([Id], [Fecha], [ArticuloId], [Cantidad], [TipoMovimientoId], [UsuarioId]) VALUES (25, N'2024-06-20 14:20:00', 2, 2, 5, 7)
INSERT INTO [dbo].[MovimientoStock] ([Id], [Fecha], [ArticuloId], [Cantidad], [TipoMovimientoId], [UsuarioId]) VALUES (26, N'2024-06-20 14:25:00', 1, 3, 1, 9)
INSERT INTO [dbo].[MovimientoStock] ([Id], [Fecha], [ArticuloId], [Cantidad], [TipoMovimientoId], [UsuarioId]) VALUES (27, N'2024-06-20 14:30:00', 4, 1, 3, 10)
INSERT INTO [dbo].[MovimientoStock] ([Id], [Fecha], [ArticuloId], [Cantidad], [TipoMovimientoId], [UsuarioId]) VALUES (28, N'2024-06-20 14:35:00', 20, 2, 6, 2)
INSERT INTO [dbo].[MovimientoStock] ([Id], [Fecha], [ArticuloId], [Cantidad], [TipoMovimientoId], [UsuarioId]) VALUES (30, N'2024-06-20 14:45:00', 11, 1, 4, 6)
INSERT INTO [dbo].[MovimientoStock] ([Id], [Fecha], [ArticuloId], [Cantidad], [TipoMovimientoId], [UsuarioId]) VALUES (31, N'2024-06-20 14:50:00', 16, 2, 5, 8)
INSERT INTO [dbo].[MovimientoStock] ([Id], [Fecha], [ArticuloId], [Cantidad], [TipoMovimientoId], [UsuarioId]) VALUES (32, N'2024-06-20 15:00:00', 13, 1, 3, 1)
INSERT INTO [dbo].[MovimientoStock] ([Id], [Fecha], [ArticuloId], [Cantidad], [TipoMovimientoId], [UsuarioId]) VALUES (33, N'2024-06-20 15:05:00', 9, 3, 6, 3)
INSERT INTO [dbo].[MovimientoStock] ([Id], [Fecha], [ArticuloId], [Cantidad], [TipoMovimientoId], [UsuarioId]) VALUES (35, N'2024-06-20 15:15:00', 3, 1, 4, 7)
INSERT INTO [dbo].[MovimientoStock] ([Id], [Fecha], [ArticuloId], [Cantidad], [TipoMovimientoId], [UsuarioId]) VALUES (36, N'2024-06-20 15:20:00', 1, 2, 1, 9)
INSERT INTO [dbo].[MovimientoStock] ([Id], [Fecha], [ArticuloId], [Cantidad], [TipoMovimientoId], [UsuarioId]) VALUES (37, N'2024-06-20 15:25:00', 5, 3, 5, 11)
INSERT INTO [dbo].[MovimientoStock] ([Id], [Fecha], [ArticuloId], [Cantidad], [TipoMovimientoId], [UsuarioId]) VALUES (38, N'2024-06-20 15:30:00', 11, 1, 3, 2)
INSERT INTO [dbo].[MovimientoStock] ([Id], [Fecha], [ArticuloId], [Cantidad], [TipoMovimientoId], [UsuarioId]) VALUES (39, N'2024-06-20 15:35:00', 8, 2, 6, 4)

```
INSERT INTO [dbo].[MovimientoStock] ([Id], [Fecha], [ArticuloId], [Cantidad], [TipoMovimientoId], [UsuarioId]) VALUES (41, N'2024-06-20 15:45:00', 4, 1, 1, 8)
SET IDENTITY_INSERT [dbo].[MovimientoStock] OFF

SET IDENTITY_INSERT [dbo].[TipoMovimientos] ON
INSERT INTO [dbo].[TipoMovimientos] ([Id], [Nombre], [Reduccion]) VALUES (1, N'Compra', 0)
INSERT INTO [dbo].[TipoMovimientos] ([Id], [Nombre], [Reduccion]) VALUES (3, N'Venta', 1)
INSERT INTO [dbo].[TipoMovimientos] ([Id], [Nombre], [Reduccion]) VALUES (4, N'Devolucion', 1)
INSERT INTO [dbo].[TipoMovimientos] ([Id], [Nombre], [Reduccion]) VALUES (5, N'Entrada', 0)
INSERT INTO [dbo].[TipoMovimientos] ([Id], [Nombre], [Reduccion]) VALUES (6, N'Salida', 1)
INSERT INTO [dbo].[TipoMovimientos] ([Id], [Nombre], [Reduccion]) VALUES (7, N'Compra internacional', 0)
INSERT INTO [dbo].[TipoMovimientos] ([Id], [Nombre], [Reduccion]) VALUES (8, N'Compra nacional', 0)
INSERT INTO [dbo].[TipoMovimientos] ([Id], [Nombre], [Reduccion]) VALUES (9, N'Venta internacional', 1)
INSERT INTO [dbo].[TipoMovimientos] ([Id], [Nombre], [Reduccion]) VALUES (10, N'Venta nacional', 1)
SET IDENTITY_INSERT [dbo].[TipoMovimientos] OFF
```