

Universidad ORT Uruguay

Facultad de Ingeniería

Obligatorio II

Programación III

Docente Adriana Cabella
Santiago Corrage (215003)



Junio 2024

Indice

[Programación III](#)

[Indice](#)

[Diagrama de casos de uso](#)

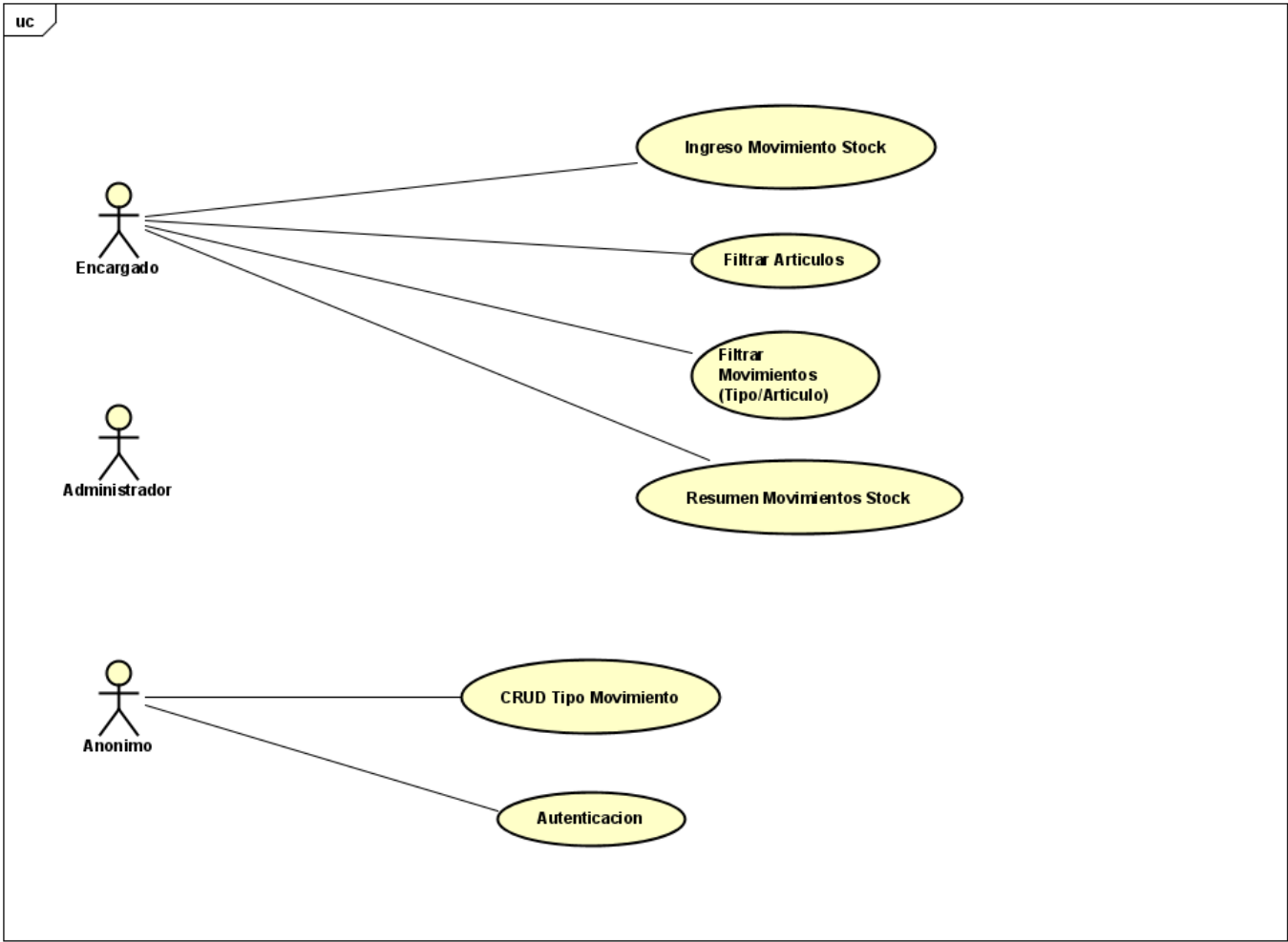
[Datos de Prueba](#)

[Datos de Prueba : Prompt AI](#)

[Codigo Fuente : WEB Api](#)

[Codigo Fuente : HTTP Client](#)

Diagrama de casos de uso



Datos de Prueba

INSERT INTO TiposMovimientos (Nombre, Coeficiente) VALUES

('Venta', -1),
('Compra', 1),
('Devolución de Cliente', 1),
('Devolución a Proveedor', -1),
('Ajuste por Inventario', 1),
('Ajuste por Pérdida', -1),
('Transferencia entre Almacenes', 1),
('Consumo Interno', -1),
('Elementos Rotos', -1),
('Venta Mayor', -1)

GO

INSERT INTO Articulos (Nombre, Descripcion,Codigo, PrecioActual) VALUES

('Lapiz HB', 'Lapiz de madera', '0000000000001', 20.00),
('Boligrafo Azul', 'Boligrafo tinta azul', '0000000000002', 30.00),
('Cuaderno A4', 'Cuaderno espiral A4', '0000000000003', 100.00),
('Goma de Borrar', 'Goma para borrar lapiz', '0000000000004', 12.00),
('Regla 30cm', 'Regla de plastico', '0000000000005', 40.00),
('Tijeras Escolares', 'Tijeras punta redonda', '0000000000006', 48.00),
('Sacapuntas', 'Sacapuntas de metal', '0000000000007', 16.00),
('Marcador Permanente', 'Marcador tinta negra', '0000000000008', 60.00),
('Cartulina Blanca', 'Cartulina tamaño A3', '0000000000009', 32.00),
('Resaltador Amarillo', 'Resaltador fluorescente', '0000000000010', 36.00),
('Carpeta A4', 'Carpeta con anillas', '0000000000011', 120.00),
('Bloc de Notas', 'Bloc de notas adhesivas', '0000000000012', 48.00),
('Cinta Adhesiva', 'Cinta adhesiva transparente', '0000000000013', 28.00),
('Pegamento en Barra', 'Pegamento no toxico', '0000000000014', 40.00),
('Lapices de Colores', 'Caja de 12 lapices de colores', '0000000000015', 180.00),
('Papel Bond A4', 'Resma de 500 hojas', '0000000000016', 200.00),

('Portaminas', 'Portaminas con minas HB', '0000000000017', 80.00),
('Corrector Liquido', 'Corrector liquido blanco', '0000000000018', 60.00),
('Libreta de Bolsillo', 'Libreta de bolsillo con rayas', '0000000000019', 72.00),
('Compás Escolar', 'Compás de metal', '0000000000020', 100.00),
('Marcador de Pizarra', 'Marcador borrrable tinta azul', '0000000000021', 60.00),
('Notas Adhesivas', 'Paquete de 100 notas adhesivas', '0000000000022', 40.00),
('Bolígrafo Negro', 'Bolígrafo tinta negra', '0000000000023', 30.00),
('Resaltador Verde', 'Resaltador fluorescente verde', '0000000000024', 36.00),
('Carpeta Oficio', 'Carpeta con anillas tamaño oficio', '0000000000025', 140.00),
('Separadores de Carpetas', 'Paquete de 10 separadores', '0000000000026', 48.00),
('Archivador A4', 'Archivador de palanca', '0000000000027', 160.00),
('Marcador de Pizarra Rojo', 'Marcador borrrable tinta roja', '0000000000028', 60.00),
('Borrador de Pizarra', 'Borrador para pizarra blanca', '0000000000029', 40.00),
('Papel Fotocopia', 'Resma de 500 hojas A4', '0000000000030', 180.00),
('Libreta de Espiral', 'Libreta de espiral tamaño A5', '0000000000031', 80.00),
('Estuche Escolar', 'Estuche de tela con cremallera', '0000000000032', 120.00),
('Regla Flexible', 'Regla de plastico flexible', '0000000000033', 60.00),
('Perforadora de Papel', 'Perforadora de 2 agujeros', '0000000000034', 100.00),
('Grapadora', 'Grapadora de metal', '0000000000035', 120.00),
('Caja de Grapas', 'Caja de 1000 grapas', '0000000000036', 40.00),
('Rotulador Fino', 'Rotulador punta fina', '0000000000037', 48.00),
('Papel Adhesivo', 'Paquete de 10 hojas adhesivas', '0000000000038', 80.00),
('Rotulador Permanente Rojo', 'Rotulador tinta roja', '0000000000039', 60.00),
('Talonario de Recibos', 'Talonario con 50 hojas', '0000000000040', 80.00),
('Cuaderno de Dibujo', 'Cuaderno de dibujo A4', '0000000000041', 140.00),
('Cinta Correctora', 'Cinta correctora blanca', '0000000000042', 72.00),
('Lápiz de Carpintero', 'Lápiz de carpintero rojo', '0000000000043', 32.00),
('Tijeras Profesionales', 'Tijeras de acero inoxidable', '0000000000044', 200.00),
('Punzón Escolar', 'Punzón de metal con mango', '0000000000045', 60.00),
('Regla Metálica', 'Regla de metal 30 cm', '0000000000046', 80.00),
('Bolígrafo Rojo', 'Bolígrafo tinta roja', '0000000000047', 30.00),

('Cuaderno Tapa Dura', 'Cuaderno tamaño A4 con tapa dura', '0000000000048', 140.00),
('Cinta Adhesiva Doble Cara', 'Cinta adhesiva doble cara', '0000000000049', 48.00),
('Libreta de Apuntes', 'Libreta de apuntes tamaño A5', '0000000000050', 100.00),
('Marcador Verde', 'Marcador tinta verde', '0000000000051', 60.00),
('Boligrafo Verde', 'Boligrafo tinta verde', '0000000000052', 30.00),
('Corrector en Cinta', 'Corrector en cinta blanca', '0000000000053', 60.00),
('Papel Crepe', 'Paquete de 10 hojas de papel crepe', '0000000000054', 80.00),
('Carpeta Plastica', 'Carpeta plastica con cierre', '0000000000055', 120.00),
('Goma de Borrar para Tinta', 'Goma de borrar tinta', '0000000000056', 20.00),
('Boligrafo Multicolor', 'Boligrafo con 4 colores', '0000000000057', 60.00),
('Rotulador Azul', 'Rotulador tinta azul', '0000000000058', 60.00),
('Portafolios', 'Portafolios de piel sintética', '0000000000059', 400.00),
('Plastilina', 'Paquete de 12 barras de plastilina', '0000000000060', 120.00),
('Cuaderno de Matemáticas', 'Cuaderno cuadriculado tamaño A4', '0000000000061', 100.00),
('Papel de Envoltura', 'Rollo de papel de envoltura', '0000000000062', 100.00),
('Cinta Decorativa', 'Cinta decorativa adhesiva', '0000000000063', 60.00),
('Pegamento Liquido', 'Pegamento liquido transparente', '0000000000064', 40.00),
('Papel de Seda', 'Paquete de 10 hojas de papel de seda', '0000000000065', 80.00),
('Cizalla de Papel', 'Cizalla de papel manual', '0000000000066', 480.00),
('Grapadora de Oficina', 'Grapadora de oficina metálica', '0000000000067', 160.00),
('Punzón de Plástico', 'Punzón con mango de plástico', '0000000000068', 40.00),
('Caja de Clips', 'Caja de 100 clips metálicos', '0000000000069', 20.00),
('Caja de Gomas Elásticas', 'Caja de 50 gomas elásticas', '0000000000070', 40.00),
('Cinta de Embalar', 'Cinta adhesiva para embalar', '0000000000071', 80.00),
('Sobres de Papel', 'Paquete de 25 sobres de papel', '0000000000072', 60.00),
('Cuaderno de Bocetos', 'Cuaderno de bocetos tamaño A4', '0000000000073', 160.00),
('Caja de Tizas', 'Caja de 12 tizas blancas', '0000000000074', 48.00),
('Marcador de Textil', 'Marcador de tinta para textil', '0000000000075', 50.00),
('Carpeta de Dibujo', 'Carpeta de dibujo tamaño A3', '0000000000076', 180.00),
('Papel de Calcar', 'Resma de papel de calcar', '0000000000077', 120.00),
('Estuche de Pinturas', 'Estuche de 12 colores de pinturas', '0000000000078', 200.00),

('Calculadora Científica', 'Calculadora científica de 12 dígitos', '0000000000079', 300.00),

('Set de Geometría', 'Set de geometría con regla, escuadra y transportador', '0000000000080', 150.00);

GO

INSERT INTO MovimientosStock (ArticuloId, TipoMovimientoId, Cantidad, FechaMovimiento) VALUES

(1, 1, 20, '2024-01-05'),

(2, 2, 15, '2024-01-10'),

(3, 3, 30, '2024-01-15'),

(4, 4, 25, '2024-01-20'),

(5, 5, 18, '2024-01-25'),

(6, 6, 22, '2024-02-05'),

(7, 7, 17, '2024-02-10'),

(8, 8, 14, '2024-02-15'),

(9, 9, 28, '2024-02-20'),

(10, 10, 24, '2024-02-25'),

(11, 1, 19, '2024-03-05'),

(12, 2, 13, '2024-03-10'),

(13, 3, 29, '2024-03-15'),

(14, 4, 16, '2024-03-20'),

(15, 5, 21, '2024-03-25'),

(16, 6, 26, '2024-04-05'),

(17, 7, 23, '2024-04-10'),

(1, 8, 27, '2024-04-15'),

(2, 9, 12, '2024-04-20'),

(3, 10, 20, '2024-04-25'),

(4, 1, 15, '2024-05-05'),

(5, 2, 18, '2024-05-10'),

(6, 3, 24, '2024-05-15'),

(7, 4, 16, '2024-05-20'),

(8, 5, 21, '2024-05-25'),

(9, 6, 26, '2024-06-05'),

(10, 7, 13, '2024-06-10'),

(11, 8, 29, '2024-06-15'),
(12, 9, 17, '2024-06-20'),
(13, 10, 22, '2024-06-25'),
(14, 1, 20, '2024-07-05'),
(15, 2, 15, '2024-07-10'),
(16, 3, 30, '2024-07-15'),
(17, 4, 25, '2024-07-20'),
(1, 5, 18, '2024-07-25'),
(2, 6, 22, '2024-08-05'),
(3, 7, 17, '2024-08-10'),
(4, 8, 14, '2024-08-15'),
(5, 9, 28, '2024-08-20'),
(6, 10, 24, '2024-08-25'),
(7, 1, 19, '2024-09-05'),
(8, 2, 13, '2024-09-10'),
(9, 3, 29, '2024-09-15'),
(10, 4, 16, '2024-09-20'),
(11, 5, 21, '2024-09-25'),
(12, 6, 26, '2024-10-05'),
(13, 7, 23, '2024-10-10'),
(14, 8, 27, '2024-10-15'),
(15, 9, 12, '2024-10-20'),
(16, 10, 20, '2024-10-25'),
(17, 1, 15, '2024-11-05'),
(1, 2, 18, '2024-11-10'),
(2, 3, 24, '2024-11-15'),
(3, 4, 16, '2024-11-20'),
(4, 5, 21, '2024-11-25'),
(5, 6, 26, '2024-12-05'),
(6, 7, 13, '2024-12-10'),
(7, 8, 29, '2024-12-15'),

(8, 9, 17, '2024-12-20'),

(9, 10, 22, '2024-12-25');

INSERT INTO Parametros(Nombre, Valor) VALUES

('TopeCantidad', 1500),

('TopePaginas', 10)

INSERT INTO Usuarios(Nombre, Apellido, Email, Discriminator, Password_Encriptada, Password_Valor) VALUES

('Santiago', 'Correge', 'scorrege@gmail.com', 'Encargado',
'f4035833cb911a3e8634903e498ff84b05d4b66f73c6155af3f58b22ad3503f4', 'Admin123.'),

('Juan', 'Perez', 'jperez@gmail.com', 'Administrador',
'f4035833cb911a3e8634903e498ff84b05d4b66f73c6155af3f58b22ad3503f4', 'Admin123.')

Datos de Prueba : Prompt AI

Consulta:

Tengo esta clase ; (Comparti la clase Tipo de Movimiento) de C# que es parte de la logica de negocio de una WEBApi. Estoy trabajando con .NET Core 8 y Code First Necesito que crees un script de ingreso de datos para esta entidad. Necesito 10 registros y que respeten las reglas de validacion. Los Nombres de los tipos de movimientos tienen que ser representativos de los tipos de movimientos de stock de un deposito de logistica de una empresa por ejemplo (Venta, Compra, Elementos rotos etc Y necesito el script de sql

Respuesta

```
INSERT INTO TipoMovimientos (Nombre, Coeficiente) VALUES ('Venta', -1); INSERT INTO TipoMovimientos (Nombre, Coeficiente) VALUES ('Compra', 1); INSERT INTO TipoMovimientos (Nombre, Coeficiente) VALUES ('Elementos Rotos', -1); INSERT INTO TipoMovimientos (Nombre, Coeficiente) VALUES ('Devolución de Cliente', 1); INSERT INTO TipoMovimientos (Nombre, Coeficiente) VALUES ('Devolución a Proveedor', -1); INSERT INTO TipoMovimientos (Nombre, Coeficiente) VALUES ('Ajuste de Inventario', 1); INSERT INTO TipoMovimientos (Nombre, Coeficiente) VALUES ('Transferencia a Otro Depósito', -1); INSERT INTO TipoMovimientos (Nombre, Coeficiente) VALUES ('Transferencia desde Otro Depósito', 1); INSERT INTO TipoMovimientos (Nombre, Coeficiente) VALUES ('Donación', -1); INSERT INTO TipoMovimientos (Nombre, Coeficiente) VALUES ('Recepción de Donación', 1);
```

Consulta:

Perfecto, ahora necesito la insercion de 80 articulos distintos cumpliendo las validaciones correspondientes -> (Comparti la clase entera Articulos con las validaciones)

Respuesta:

Para insertar 80 artículos distintos cumpliendo con las validaciones de la clase [Articulo](#), el script SQL debe asegurar que los nombres tengan entre 10 y 200 caracteres, las descripciones tengan al menos 5 caracteres, los códigos tengan exactamente 13 dígitos y los precios sean mayores a 0.

Aquí tienes el script SQL para la inserción de los artículos:

```
INSERT INTO Articulos (Nombre, Descripcion,Codigo, PrecioActual) VALUES ('Articulo 001', 'Descripcion A', '1234567890123', 10.00);
```

```
INSERT INTO Articulos (Nombre, Descripcion,Codigo, PrecioActual) VALUES ('Articulo 002', 'Descripcion B', '1234567890124', 15.50);
```

```
INSERT INTO Articulos (Nombre, Descripcion,Codigo, PrecioActual) VALUES ('Articulo 003', 'Descripcion C', '1234567890125', 20.75);
```

```
INSERT INTO Articulos (Nombre, Descripcion,Codigo, PrecioActual) VALUES ('Articulo
```

Consulta:

Los articulos pertenecen a una papeleria asi que los nombres de los mismos deben corresponderse a nombres de articulos de una papeleria

Respuesta:

Entendido. Aquí tienes un script SQL para insertar 80 artículos de papelería con nombres correspondientes, cumpliendo con las validaciones de tu clase **Articulo**:

```
INSERT INTO Articulos (Nombre, Descripcion,Codigo, PrecioActual) VALUES ('Lapiz HB', 'Lapiz de madera', '00000000000001', 0.50);
```

```
INSERT INTO Articulos (Nombre, Descripcion,Codigo, PrecioActual) VALUES ('Boligrafo Azul', 'Boligrafo tinta azul', '00000000000002', 0.75);
```

```
INSERT INTO Articulos (Nombre, Descripcion,Codigo, PrecioActual) VALUES ('Cuaderno A4', 'Cuaderno espiral A4', '00000000000003', 2.50);
```

```
INSERT INTO Articulos (Nombre, Descripcion,Codigo, PrecioActual) VALUES ('Goma de Borrar', 'Goma para borrar lapiz', '00000000000004', 0.30);
```

```
INSERT INTO Articulos (Nombre, Descripcion,Codigo, PrecioActual) VALUES ('Regla 30cm', 'Regla de plastico', '00000000000005', 1.00);
```

```
INSERT INTO Articulos (Nombre, Descripcion,Codigo, PrecioActual) VALUES ('Tijeras Escolares', 'Tijeras punta redonda', '00000000000006', 1.20);
```

```
INSERT INTO Articulos (Nombre, Descripcion,Codigo, PrecioActual) VALUES ('Sacapuntas', 'Sacapuntas de metal', '00000000000007', 0.40);
```

```
INSERT INTO Articulos (Nombre, Descripcion,Codigo, PrecioActual) VALUES ('Marcador Permanente', 'Marcador tinta negra', '00000000000008', 1.50);
```

```
INSERT INTO Articulos (Nombre, Descripcion,Codigo, PrecioActual) VALUES ('Cartulina Blanca', 'Cartulina tamaño A3', '00000000000009', 0.80);
```

```
INSERT INTO Articulos (Nombre, Descripcion,Codigo, PrecioActual) VALUES ('Resaltador Amarillo', 'Resaltador fluorescente', '00000000000010', 0.90);
```

```
INSERT INTO Articulos (Nombre, Descripcion,Codigo, PrecioActual) VALUES ('Carpeta A4', 'Carpeta con anillas', '00000000000011', 3.00);
```

```
INSERT INTO Articulos (Nombre, Descripcion,Codigo, PrecioActual) VALUES ('Bloc de Notas', 'Bloc de notas adhesivas', '00000000000012', 1.20);
```

```
INSERT INTO Articulos (Nombre, Descripcion,Codigo, PrecioActual) VALUES ('Cinta Adhesiva', 'Cinta adhesiva transparente', '00000000000013', 0.70);
```

INSERT INTO Articulos (Nombre, Descripcion,Codigo, PrecioActual) VALUES ('Pegamento en Barra', 'Pegamento no toxico', '0000000000014', 1.00);

INSERT INTO Articulos (Nombre, Descripcion,Codigo, PrecioActual) VALUES ('Lapices de Colores', 'Caja de 12 lapices de colores', '0000000000015', 4.50);

INSERT INTO Articulos (Nombre, Descripcion,Codigo, PrecioActual) VALUES ('Papel Bond A4', 'Resma de 500 hojas', '0000000000016', 5.00);

INSERT INTO Articulos (Nombre, Descripcion,Codigo, PrecioActual) VALUES ('Portaminas', 'Portaminas con minas HB', '0000000000017', 2.00);

INSERT INTO Articulos (Nombre, Descripcion,Codigo, PrecioActual) VALUES ('Corrector Liquido', 'Corrector liquido blanco', '0000000000018', 1.50);

INSERT INTO Articulos (Nombre, Descripcion,Codigo, PrecioActual) VALUES ('Libreta de Bolsillo', 'Libreta de bolsillo con rayas', '0000000000019', 1.80);

INSERT INTO Articulos (Nombre, Descripcion,Codigo, PrecioActual) VALUES ('Compás Escolar', 'Compás de metal', '0000000000020', 2.50);

INSERT INTO Articulos (Nombre, Descripcion,Codigo, PrecioActual) VALUES ('Marcador de Pizarra', 'Marcador borrable tinta azul', '0000000000021', 1.50);

INSERT INTO Articulos (Nombre, Descripcion,Codigo, PrecioActual) VALUES ('Notas Adhesivas', 'Paquete de 100 notas adhesivas', '0000000000022', 1.00);

INSERT INTO Articulos (Nombre, Descripcion,Codigo, PrecioActual) VALUES ('Boligrafo Negro', 'Boligrafo tinta negra', '0000000000023', 0.75);

INSERT INTO Articulos (Nombre, Descripcion,Codigo, PrecioActual) VALUES ('Resaltador Verde', 'Resaltador fluorescente verde', '0000000000024', 0.90);

INSERT INTO Articulos (Nombre, Descripcion,Codigo, PrecioActual) VALUES ('Carpeta Oficio', 'Carpeta con anillas tamaño oficio', '0000000000025', 3.50);

INSERT INTO Articulos (Nombre, Descripcion,Codigo, PrecioActual) VALUES ('Separadores de Carpetas', 'Paquete de 10 separadores', '0000000000026', 1.20);

INSERT INTO Articulos (Nombre, Descripcion,Codigo, PrecioActual) VALUES ('Archivador A4', 'Archivador de palanca', '0000000000027', 4.00);

INSERT INTO Articulos (Nombre, Descripcion,Codigo, PrecioActual) VALUES ('Marcador de Pizarra Rojo', 'Marcador borrable tinta roja', '0000000000028', 1.50);

INSERT INTO Articulos (Nombre, Descripcion,Codigo, PrecioActual) VALUES ('Borrador de Pizarra', 'Borrador para pizarra blanca', '0000000000029', 1.00);

INSERT INTO Articulos (Nombre, Descripcion,Codigo, PrecioActual) VALUES ('Papel Fotocopia', 'Resma de 500 hojas A4', '0000000000030', 4.50);

INSERT INTO Articulos (Nombre, Descripcion,Codigo, PrecioActual) VALUES ('Libreta de Espiral', 'Libreta de espiral tamaño A5', '0000000000031', 2.00);

INSERT INTO Articulos (Nombre, Descripcion,Codigo, PrecioActual) VALUES ('Estuche Escolar', 'Estuche de tela con cremallera', '0000000000032', 3.00);

INSERT INTO Articulos (Nombre, Descripcion,Codigo, PrecioActual) VALUES ('Regla Flexible', 'Regla de plastico flexible', '0000000000033', 1.50);

INSERT INTO Articulos (Nombre, Descripcion,Codigo, PrecioActual) VALUES ('Perforadora de Papel', 'Perforadora de 2 agujeros', '0000000000034', 2.50);

INSERT INTO Articulos (Nombre, Descripcion,Codigo, PrecioActual) VALUES ('Grapadora', 'Grapadora de metal', '0000000000035', 3.00);

INSERT INTO Articulos (Nombre, Descripcion,Codigo, PrecioActual) VALUES ('Caja de Grapas', 'Caja de 1000 grapas', '0000000000036', 1.00);

INSERT INTO Articulos (Nombre, Descripcion,Codigo, PrecioActual) VALUES ('Rotulador Fino', 'Rotulador punta fina', '0000000000037', 1.20);

INSERT INTO Articulos (Nombre, Descripcion,Codigo, PrecioActual) VALUES ('Papel Adhesivo', 'Paquete de 10 hojas adhesivas', '0000000000038', 2.00);

INSERT INTO Articulos (Nombre, Descripcion,Codigo, PrecioActual) VALUES ('Rotulador Permanente Rojo', 'Rotulador tinta roja', '0000000000039', 1.50);

INSERT INTO Articulos (Nombre, Descripcion,Codigo, PrecioActual) VALUES ('Talonario de Recibos', 'Talonario con 50 hojas', '0000000000040', 2.00);

INSERT INTO Articulos (Nombre, Descripcion,Codigo, PrecioActual) VALUES ('Cuaderno de Dibujo', 'Cuaderno de dibujo A4', '0000000000041', 3.50);

INSERT INTO Articulos (Nombre, Descripcion,Codigo, PrecioActual) VALUES ('Cinta Correctora', 'Cinta correctora blanca', '0000000000042', 1.80);

INSERT INTO Articulos (Nombre, Descripcion,Codigo, PrecioActual) VALUES ('Lápiz de Carpintero', 'Lápiz de carpintero rojo', '0000000000043', 0.80);

INSERT INTO Articulos (Nombre, Descripcion,Codigo, PrecioActual) VALUES ('Tijeras Profesionales', 'Tijeras de acero inoxidable', '0000000000044', 5.00);

INSERT INTO Articulos (Nombre, Descripcion,Codigo, PrecioActual) VALUES ('Punzón Escolar', 'Punzón de metal con mango', '0000000000045', 1.50);

INSERT INTO Articulos (Nombre, Descripcion,Codigo, PrecioActual) VALUES ('Regla Metálica', 'Regla de metal 30 cm', '0000000000046', 2.00);

INSERT INTO Articulos (Nombre, Descripcion,Codigo, PrecioActual) VALUES ('Boligrafo Rojo', 'Boligrafo tinta roja', '0000000000047', 0.75);

Consulta:

Perfecto, te animas a pasarme los mismos datos pero en un unico insert multiple? -> Y ajustar los precios a que en vez de sean en dolares sean en pesos uruguayos (1 dolar - 40 pesos uruguayos)

Respuesta:

```
INSERT INTO Articulos (Nombre, Descripcion, Codigo, PrecioActual) VALUES
('Lapiz HB', 'Lapiz de madera', '00000000000001', 20.00),
('Boligrafo Azul', 'Boligrafo tinta azul', '00000000000002', 30.00),
('Cuaderno A4', 'Cuaderno espiral A4', '00000000000003', 100.00),
('Goma de Borrar', 'Goma para borrar lapiz', '00000000000004', 12.00),
('Regla 30cm', 'Regla de plastico', '00000000000005', 40.00),
('Tijeras Escolares', 'Tijeras punta redonda', '00000000000006', 48.00),
('Sacapuntas', 'Sacapuntas de metal', '00000000000007', 16.00),
('Marcador Permanente', 'Marcador tinta negra', '00000000000008', 60.00),
('Cartulina Blanca', 'Cartulina tamaño A3', '00000000000009', 32.00),
('Resaltador Amarillo', 'Resaltador fluorescente', '00000000000010', 36.00),
('Carpeta A4', 'Carpeta con anillas', '00000000000011', 120.00),
('Bloc de Notas', 'Bloc de notas adhesivas', '00000000000012', 48.00),
('Cinta Adhesiva', 'Cinta adhesiva transparente', '00000000000013', 28.00),
('Pegamento en Barra', 'Pegamento no toxico', '00000000000014', 40.00),
('Lapices de Colores', 'Caja de 12 lapices de colores', '00000000000015', 180.00),
('Papel Bond A4', 'Resma de 500 hojas', '00000000000016', 200.00),
('Portaminas', 'Portaminas con minas HB', '00000000000017', 80.00),
('Corrector Liquido', 'Corrector liquido blanco', '00000000000018', 60.00),
('Libreta de Bolsillo', 'Libreta de bolsillo con rayas', '00000000000019', 72.00),
('Compás Escolar', 'Compás de metal', '00000000000020', 100.00),
('Marcador de Pizarra', 'Marcador borrrable tinta azul', '00000000000021', 60.00),
('Notas Adhesivas', 'Paquete de 100 notas adhesivas', '00000000000022', 40.00),
('Boligrafo Negro', 'Boligrafo tinta negra', '00000000000023', 30.00),
('Resaltador Verde', 'Resaltador fluorescente verde', '00000000000024', 36.00),
('Carpeta Oficio', 'Carpeta con anillas tamaño oficio', '00000000000025', 140.00),
('Separadores de Carpetas', 'Paquete de 10 separadores', '00000000000026', 48.00),
('Archivador A4', 'Archivador de palanca', '00000000000027', 160.00),
('Marcador de Pizarra Rojo', 'Marcador borrrable tinta roja', '00000000000028', 60.00),
('Borrador de Pizarra', 'Borrador para pizarra blanca', '00000000000029', 40.00),
('Papel Fotocopia', 'Resma de 500 hojas A4', '00000000000030', 180.00),
('Libreta de Espiral', 'Libreta de espiral tamaño A5', '00000000000031', 80.00),
('Estuche Escolar', 'Estuche de tela con cremallera', '00000000000032', 120.00),
('Regla Flexible', 'Regla de plastico flexible', '00000000000033', 60.00),
('Perforadora de Papel', 'Perforadora de 2 agujeros', '00000000000034', 100.00),
('Grapadora', 'Grapadora de metal', '00000000000035', 120.00),
('Caja de Grapas', 'Caja de 1000 grapas', '00000000000036', 40.00),
('Rotulador Fino', 'Rotulador punta fina', '00000000000037', 48.00),
('Papel Adhesivo', 'Paquete de 10 hojas adhesivas', '00000000000038', 80.00),
```

('Rotulador Permanente Rojo', 'Rotulador tinta roja', '0000000000039', 60.00),
('Talonario de Recibos', 'Talonario con 50 hojas', '0000000000040', 80.00),
('Cuaderno de Dibujo', 'Cuaderno de dibujo A4', '0000000000041', 140.00),
('Cinta Correctora', 'Cinta correctora blanca', '0000000000042', 72.00),
('Lápiz de Carpintero', 'Lápiz de carpintero rojo', '0000000000043', 32.00),
('Tijeras Profesionales', 'Tijeras de acero inoxidable', '0000000000044', 200.00),
('Punzón Escolar', 'Punzón de metal con mango', '00000000000645', 60.00), ('Regla
Metálica', 'Regla de metal 30 cm', '0000000000046', 80.00), ('Boligrafo Rojo', 'Boligrafo tinta
roja', '0000000000047', 30.00), ('Cuaderno Tapa Dura', 'Cuaderno tamaño A4 con tapa
dura', '0000000000048', 140.00), ('Cinta Adhesiva Doble Cara', 'Cinta adhesiva doble cara',
'0000000000049', 48.00), ('Libreta de Apuntes', 'Libreta de apuntes tamaño A5',
'0000000000050', 100.00), ('Marcador Verde', 'Marcador tinta verde', '0000000000051',
60.00), ('Boligrafo Verde', 'Boligrafo tinta verde', '0000000000052', 30.00), ('Corrector en
Cinta', 'Corrector en cinta blanca', '0000000000053', 60.00), ('Papel Crepe', 'Paquete de 10
hojas de papel crepe', '0000000000054', 80.00), ('Carpeta Plastica', 'Carpeta plastica con
cierres', '0000000000055', 120.00), ('Goma de Borrarr para Tinta', 'Goma de borrar tinta',
'0000000000056', 20.00), ('Boligrafo Multicolor', 'Boligrafo con 4 colores', '0000000000057',
60.00), ('Rotulador Azul', 'Rotulador tinta azul', '0000000000058', 60.00), ('Portafolios',
'Portafolios de piel sintética', '0000000000059', 400.00), ('Plastilina', 'Paquete de 12 barras
de plastilina', '0000000000060', 120.00), ('Cuaderno de Matemáticas', 'Cuaderno
cuadrado tamaño A4', '0000000000061', 100.00), ('Papel de Envoltura', 'Rollo de papel
de envoltura', '0000000000062', 100.00), ('Cinta Decorativa', 'Cinta decorativa adhesiva',
'0000000000063', 60.00), ('Pegamento Liquido', 'Pegamento liquido transparente',
'0000000000064', 40.00), ('Papel de Seda', 'Paquete de 10 hojas de papel de seda',
'0000000000065', 80.00), ('Cizalla de Papel', 'Cizalla de papel manual', '0000000000066',
480.00), ('Grapadora de Oficina', 'Grapadora de oficina metálica', '0000000000067', 160.00),
('Punzón de Plástico', 'Punzón con mango de plástico', '0000000000068', 40.00), ('Caja de
Clips', 'Caja de 100 clips metálicos', '0000000000069', 20.00), ('Caja de Gomas Elásticas',
'Caja de 50 gomas elásticas', '0000000000070', 40.00), ('Cinta de Embalar', 'Cinta adhesiva
para embalar', '0000000000071', 80.00), ('Sobres de Papel', 'Paquete de 25 sobres de
papel', '0000000000072', 60.00), ('Cuaderno de Bocetos', 'Cuaderno de bocetos tamaño A4',
'0000000000073', 160.00), ('Caja de Tizas', 'Caja de 12 tizas blancas', '0000000000074',
48.00), ('Marcador de Textil', 'Marcador de tinta para textil', '0000000000075', 50.00),
('Carpeta de Dibujo', 'Carpeta de dibujo tamaño A3', '0000000000076', 180.00), ('Papel de
Calcar', 'Resma de papel de calcar', '0000000000077', 120.00), ('Estuche de Pinturas',
'Estuche de 12 colores de pinturas', '0000000000078', 200.00), ('Calculadora Científica',
'Calculadora científica de 12 dígitos', '0000000000079', 300.00), ('Set de Geometría', 'Set de
geometría con regla, escuadra y transportador', '0000000000080', 150.00);

Consulta:

Teniendo en cuenta los registros que me creaste de Articulos (80 registros) y de tipos de movimientos (10 registros) y que el id de ambas clases es identity necesito que me crees 30 movimientos de stock en diferentes fechas , algunos acumulados en enero otros en marzo etc , que la cantidad de movimientos de stock por mes varie(no que me ingreses proporcionalmente por mes las cantidades) y tambien necesito que tampoco sea proporcional el ingreso segun los tipo de movimientos (que algunos tipos de movimientos tengan mas movimientos de stock asignados) -> (Comparti clase Movimientos de Stock). Necesito que me des los datos hardcodeados

Respuesta:

Entendido, aquí te dejo 30 movimientos de stock hardcodeados con diferentes fechas, cantidades y tipos de movimientos:

-- Movimientos de stock hardcodeados

INSERT INTO MovimientoStock (Articuloid, TipoMovimientoid, Cantidad, FechaMovimiento)
VALUES

(1, 1, 20, '2023-01-05'),
(2, 2, 15, '2023-01-10'),
(3, 3, 30, '2023-01-15'),
(4, 4, 25, '2023-02-05'),
(5, 5, 18, '2023-02-10'),
(6, 6, 22, '2023-02-15'),
(7, 7, 17, '2023-03-05'),
(8, 8, 14, '2023-03-10'),
(9, 9, 28, '2023-03-15'),
(10, 10, 24, '2023-04-05'),
(11, 1, 19, '2023-04-10'),
(12, 2, 13, '2023-04-15'),
(13, 3, 29, '2023-05-05'),
(14, 4, 16, '2023-05-10'),
(15, 5, 21, '2023-05-15'),
(16, 6, 26, '2023-06-05'),
(17, 7, 23, '2023-06-10'),
(18, 8, 27, '2023-06-15'),
(19, 9, 12, '2023-07-05'),
(20, 10, 20, '2023-07-10'),
(21, 1, 15, '2023-07-15'),
(22, 2, 18, '2023-08-05'),
(23, 3, 24, '2023-08-10'),
(24, 4, 16, '2023-08-15'),
(25, 5, 21, '2023-09-05'),
(26, 6, 26, '2023-09-10'),
(27, 7, 13, '2023-09-15'),
(28, 8, 29, '2023-10-05'),

(29, 9, 17, '2023-10-10'),
(30, 10, 22, '2023-10-15');

Consulta:

```
INSERT INTO MovimientosStock (ArticuloId, TipoMovimientoId, Cantidad,
FechaMovimiento) VALUES (1, 1, 20, '2023-01-05'), (2, 2, 15, '2023-01-10'), (3, 3, 30,
'2023-01-15'), (4, 4, 25, '2023-01-20'), (5, 5, 18, '2023-01-25'), (6, 6, 22, '2023-02-05'), (7, 7,
17, '2023-02-10'), (8, 8, 14, '2023-02-15'), (9, 9, 28, '2023-02-20'), (10, 10, 24, '2023-02-25'),
(1, 1, 19, '2023-03-05'), (2, 2, 13, '2023-03-10'), (3, 3, 29, '2023-03-15'), (4, 4, 16,
'2023-03-20'), (5, 5, 21, '2023-03-25'), (6, 6, 26, '2023-04-05'), (7, 7, 23, '2023-04-10'), (8, 8,
27, '2023-04-15'), (9, 9, 12, '2023-04-20'), (10, 10, 20, '2023-04-25'), (1, 1, 15, '2023-05-05'),
(2, 2, 18, '2023-05-10'), (3, 3, 24, '2023-05-15'), (4, 4, 16, '2023-05-20'), (5, 5, 21,
'2023-05-25'), (6, 6, 26, '2023-06-05'), (7, 7, 13, '2023-06-10'), (8, 8, 29, '2023-06-15'), (9, 9,
17, '2023-06-20'), (10, 10, 22, '2023-06-25'), (1, 1, 20, '2023-07-05'), (2, 2, 15, '2023-07-10'),
(3, 3, 30, '2023-07-15'), (4, 4, 25, '2023-07-20'), (5, 5, 18, '2023-07-25'), (6, 6, 22,
'2023-08-05'), (7, 7, 17, '2023-08-10'), (8, 8, 14, '2023-08-15'), (9, 9, 28, '2023-08-20'), (10,
10, 24, '2023-08-25'), (1, 1, 19, '2023-09-05'), (2, 2, 13, '2023-09-10'), (3, 3, 29,
'2023-09-15'), (4, 4, 16, '2023-09-20'), (5, 5, 21, '2023-09-25'), (6, 6, 26, '2023-10-05'), (7, 7,
23, '2023-10-10'), (8, 8, 27, '2023-10-15'), (9, 9, 12, '2023-10-20'), (10, 10, 20, '2023-10-25');
```

-> Me puedes dar el mismo insert pero para el año 2024?

Respuesta:

-- Movimientos de stock hardcodeados para el año 2024

```
INSERT INTO MovimientosStock (ArticuloId, TipoMovimientoId, Cantidad,
FechaMovimiento) VALUES
(1, 1, 20, '2024-01-05'),
(2, 2, 15, '2024-01-10'),
(3, 3, 30, '2024-01-15'),
(4, 4, 25, '2024-01-20'),
(5, 5, 18, '2024-01-25'),
(6, 6, 22, '2024-02-05'),
(7, 7, 17, '2024-02-10'),
(8, 8, 14, '2024-02-15'),
(9, 9, 28, '2024-02-20'),
(10, 10, 24, '2024-02-25'),
(1, 1, 19, '2024-03-05'),
(2, 2, 13, '2024-03-10'),
(3, 3, 29, '2024-03-15'),
(4, 4, 16, '2024-03-20'),
(5, 5, 21, '2024-03-25'),
(6, 6, 26, '2024-04-05'),
(7, 7, 23, '2024-04-10'),
(8, 8, 27, '2024-04-15'),
(9, 9, 12, '2024-04-20'),
```

(10, 10, 20, '2024-04-25'),
(1, 1, 15, '2024-05-05'),
(2, 2, 18, '2024-05-10'),
(3, 3, 24, '2024-05-15'),
(4, 4, 16, '2024-05-20'),
(5, 5, 21, '2024-05-25'),
(6, 6, 26, '2024-06-05'),
(7, 7, 13, '2024-06-10'),
(8, 8, 29, '2024-06-15'),
(9, 9, 17, '2024-06-20'),
(10, 10, 22, '2024-06-25'),
(1, 1, 20, '2024-07-05'),
(2, 2, 15, '2024-07-10'),
(3, 3, 30, '2024-07-15'),
(4, 4, 25, '2024-07-20'),
(5, 5, 18, '2024-07-25'),
(6, 6, 22, '2024-08-05'),
(7, 7, 17, '2024-08-10'),
(8, 8, 14, '2024-08-15'),
(9, 9, 28, '2024-08-20'),
(10, 10, 24, '2024-08-25'),
(1, 1, 19, '2024-09-05'),
(2, 2, 13, '2024-09-10'),
(3, 3, 29, '2024-09-15'),
(4, 4, 16, '2024-09-20'),
(5, 5, 21, '2024-09-25'),
(6, 6, 26, '2024-10-05'),
(7, 7, 23, '2024-10-10'),
(8, 8, 27, '2024-10-15'),
(9, 9, 12, '2024-10-20'),
(10, 10, 20, '2024-10-25');

Consulta:

```
INSERT INTO MovimientosStock (Articulold, TipoMovimientold, Cantidad,
FechaMovimiento) VALUES (1, 1, 20, '2024-01-05'), (2, 2, 15, '2024-01-10'), (3, 3, 30,
'2024-01-15'), (4, 4, 25, '2024-01-20'), (5, 5, 18, '2024-01-25'), (6, 6, 22, '2024-02-05'), (7, 7,
17, '2024-02-10'), (8, 8, 14, '2024-02-15'), (9, 9, 28, '2024-02-20'), (10, 10, 24, '2024-02-25'),
(1, 1, 19, '2024-03-05'), (2, 2, 13, '2024-03-10'), (3, 3, 29, '2024-03-15'), (4, 4, 16,
'2024-03-20'), (5, 5, 21, '2024-03-25'), (6, 6, 26, '2024-04-05'), (7, 7, 23, '2024-04-10'), (8, 8,
27, '2024-04-15'), (9, 9, 12, '2024-04-20'), (10, 10, 20, '2024-04-25'), (1, 1, 15, '2024-05-05'),
(2, 2, 18, '2024-05-10'), (3, 3, 24, '2024-05-15'), (4, 4, 16, '2024-05-20'), (5, 5, 21,
'2024-05-25'), (6, 6, 26, '2024-06-05'), (7, 7, 13, '2024-06-10'), (8, 8, 29, '2024-06-15'), (9, 9,
17, '2024-06-20'), (10, 10, 22, '2024-06-25'), (1, 1, 20, '2024-07-05'), (2, 2, 15, '2024-07-10'),
(3, 3, 30, '2024-07-15'), (4, 4, 25, '2024-07-20'), (5, 5, 18, '2024-07-25'), (6, 6, 22,
'2024-08-05'), (7, 7, 17, '2024-08-10'), (8, 8, 14, '2024-08-15'), (9, 9, 28, '2024-08-20'), (10,
10, 24, '2024-08-25'), (1, 1, 19, '2024-09-05'), (2, 2, 13, '2024-09-10'), (3, 3, 29,
'2024-09-15'), (4, 4, 16, '2024-09-20'), (5, 5, 21, '2024-09-25'), (6, 6, 26, '2024-10-05'), (7, 7,
23, '2024-10-10'), (8, 8, 27, '2024-10-15'), (9, 9, 12, '2024-10-20'), (10, 10, 20, '2024-10-25');
-> Modifica este insert para que se ingresen articulos con id del 1 al 17 y necesito 10 inserts
mas
```

Respuesta:

Claro, aquí tienes el script modificado para que los Articulold vayan del 1 al 17 y te añadí 10 inserts más:

```
INSERT INTO MovimientosStock (Articulold, TipoMovimientold, Cantidad,
FechaMovimiento) VALUES
(1, 1, 20, '2024-01-05'),
(2, 2, 15, '2024-01-10'),
(3, 3, 30, '2024-01-15'),
(4, 4, 25, '2024-01-20'),
(5, 5, 18, '2024-01-25'),
(6, 6, 22, '2024-02-05'),
(7, 7, 17, '2024-02-10'),
(8, 8, 14, '2024-02-15'),
(9, 9, 28, '2024-02-20'),
(10, 10, 24, '2024-02-25'),
(11, 1, 19, '2024-03-05'),
(12, 2, 13, '2024-03-10'),
(13, 3, 29, '2024-03-15'),
(14, 4, 16, '2024-03-20'),
(15, 5, 21, '2024-03-25'),
(16, 6, 26, '2024-04-05'),
(17, 7, 23, '2024-04-10'),
(1, 8, 27, '2024-04-15'),
(2, 9, 12, '2024-04-20'),
(3, 10, 20, '2024-04-25'),
```

(4, 1, 15, '2024-05-05'),
(5, 2, 18, '2024-05-10'),
(6, 3, 24, '2024-05-15'),
(7, 4, 16, '2024-05-20'),
(8, 5, 21, '2024-05-25'),
(9, 6, 26, '2024-06-05'),
(10, 7, 13, '2024-06-10'),
(11, 8, 29, '2024-06-15'),
(12, 9, 17, '2024-06-20'),
(13, 10, 22, '2024-06-25'),
(14, 1, 20, '2024-07-05'),
(15, 2, 15, '2024-07-10'),
(16, 3, 30, '2024-07-15'),
(17, 4, 25, '2024-07-20'),
(1, 5, 18, '2024-07-25'),
(2, 6, 22, '2024-08-05'),
(3, 7, 17, '2024-08-10'),
(4, 8, 14, '2024-08-15'),
(5, 9, 28, '2024-08-20'),
(6, 10, 24, '2024-08-25'),
(7, 1, 19, '2024-09-05'),
(8, 2, 13, '2024-09-10'),
(9, 3, 29, '2024-09-15'),
(10, 4, 16, '2024-09-20'),
(11, 5, 21, '2024-09-25'),
(12, 6, 26, '2024-10-05'),
(13, 7, 23, '2024-10-10'),
(14, 8, 27, '2024-10-15'),
(15, 9, 12, '2024-10-20'),
(16, 10, 20, '2024-10-25'),
(17, 1, 15, '2024-11-05'),
(1, 2, 18, '2024-11-10'),
(2, 3, 24, '2024-11-15'),
(3, 4, 16, '2024-11-20'),
(4, 5, 21, '2024-11-25'),
(5, 6, 26, '2024-12-05'),
(6, 7, 13, '2024-12-10'),
(7, 8, 29, '2024-12-15'),
(8, 9, 17, '2024-12-20'),
(9, 10, 22, '2024-12-25');

Codigo Fuente : WEB Api

```
using Microsoft.EntityFrameworkCore.Metadata.Builders;
using Microsoft.EntityFrameworkCore.Storage.ValueConversion;
using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using SistemaAutenticacion.Entidades;
using SistemaAutenticacion.ValueObject;

namespace Gestor.AccesoDatos.Configuraciones
{
    public class UsuarioConfiguracion : IEntityTypeConfiguration<Usuario>
    {
        public void Configure(EntityTypeBuilder<Usuario> builder)
        {
            var emailConverter = new ValueConverter<Email, string>
            (
                e => e.Valor,
                e => new Email(e)
            );
            builder.Property(u => u.Email).HasConversion(emailConverter);
        }
    }
}

using Gestor.AccesoDatos.Configuraciones;
using Gestor.LogicaNegocio.Articulos;
using Gestor.LogicaNegocio.Entidades;
using Microsoft.EntityFrameworkCore;
using SistemaAutenticacion.Entidades;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Gestor.AccesoDatos.EF
{
    public class GestorContext : DbContext
    {
        public DbSet<Articulo> Articulos { get; set; }
    }
}
```

```

        public DbSet<MovimientoStock> MovimientosStock { get; set; }

        public DbSet<TipoMovimiento> TiposMovimientos { get; set; }

        public DbSet<Usuario> Usuarios { get; set; }

        public DbSet<Administrador> Administradores { get; set; }

        public DbSet<Encargado> Miembros { get; set; }

        public DbSet<Parametro> Parametros { get; set; }

        public GestorContext(DbContextOptions options) : base(options)
        {
        }

        protected override void OnModelCreating(ModelBuilder modelBuilder)
        {
            new UsuarioConfiguracion().Configure(modelBuilder.Entity<Usuario>());
        }
    }

    using Gestor.LogicaNegocio.Articulos;
    using Gestor.LogicaNegocio.Excepciones;
    using Gestor.LogicaNegocio.InterfacesRepositorios;
    using Microsoft.Data.SqlClient;
    using Microsoft.EntityFrameworkCore;
    using System;
    using System.Collections.Generic;
    using System.Linq;
    using System.Text;
    using System.Threading.Tasks;

    namespace Gestor.AccesoDatos.EF
    {
        public class RepositorioArticuloEF : IRepositoryArticulo
        {
            private GestorContext _db;

            public RepositorioArticuloEF(GestorContext db)
            {
                _db = db;
            }

            public void Add(Articulo articulo)
            {

```

```
        try
        {
            if (articulo == null) throw new ArticuloException("El articulo no puede ser
nulo");
            _db.Articulos.Add(articulo);
            _db.SaveChanges();
        }
        catch (DbUpdateException ex)
        {
            if (ex.InnerException != null)
            {
                {
                    SqlException exSql = ex.InnerException as SqlException;
                    if (exSql.Number == 2601)
                        throw new Exception("Ya existe un articulo con ese nombre o codigo");
                }
                throw;
            }
        }
        catch (Exception ex)
        {
            throw new ArticuloException($"El articulo no se pudo agregar. Más info:
{ex.Message}");
        }
    }

    public IEnumerable<Articulo> GetAll()
    {
        try
        {
            return _db.Articulos.ToList();
        }
        catch (Exception ex)
        {
            throw new ArticuloException($"No se pudo devolver la lista de articulos. Más
info: {ex.Message}");
        }
    }

    public Articulo GetById(int id)
    {
        try
        {
            var articulo = _db.Articulos.Find(id);
            if (articulo == null)
            {
                throw new ArticuloException("El articulo con ese id no existe");
            }
            return articulo;
        }
    }
}
```

```
    }
    catch (Exception e)
    {
        throw new ArtículoException($"No se ha podido encontrar el articulo con id
{id}. Mas info: {e.Message}");
    }
}

public IEnumerable<Articulo> GetArticulosOrdenadosAlfabeticamente()
{
    try
    {
        var articulos = _db.Articulos.OrderBy(a => a.Nombre).ToList();
        if (articulos == null || articulos.Count == 0)
        {
            throw new Exception("No hay articulos");
        }
        return articulos;
    }
    catch (Exception ex)
    {
        throw new ArtículoException($"Error: {ex.Message}");
    }
}

public void Remove(int id)
{
    try
    {
        Articulo articulo = _db.Articulos.Find(id);
        if (articulo == null)
        {
            throw new ArtículoException($"No existe un articulo con id:{id}.");
        }
        _db.Articulos.Remove(articulo);
        _db.SaveChanges();
    }
    catch (Exception e)
    {
        throw new ArtículoException($"Error. Mas info: {e.Message}");
    }
}

public void Remove(Articulo articulo)
{
    try
    {
```



```
        if (articulo == null)
        {
            throw new ArgumentNullException("El articulo no puede ser nulo");
        }
        _db.Articulos.Remove(articulo);
        _db.SaveChanges();
    }
    catch (Exception ex)
    {
        throw new ArticuloException($"No eliminar el articulo {articulo.Nombre} la
lista de articulos. Más info: {ex.Message}");
    }
}

public void Update(int id, Articulo articuloModificado)
{
    try
    {
        var articulo = _db.Articulos.Find(id);
        if (articulo == null)
        {
            throw new ArticuloException($"No existe el autor con el id {id}");
        }
        articulo.Modificar(articuloModificado);
        _db.SaveChanges();
    }
    catch (Exception ex)
    {
        throw new ArticuloException($"No se ha podido modificar el articulo con id:
{id}. Mas info: {ex.Message}");
    }
}
}
```

```
using Gestor.LogicaNegocio.Articulos;
using Gestor.LogicaNegocio.Entidades;
using Gestor.LogicaNegocio.Excepciones;
using Gestor.LogicaNegocio.InterfacesRepositorios;
using Microsoft.Data.SqlClient;
using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
using System.Data.Common;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

```

namespace Gestor.AccesoDatos.EF
{
    public class RepositorioMovimientoStockEF : IRepositoryMovimientoStock
    {
        private GestorContext _db;
        private static int _cantMovimientos = 0;

        public RepositorioMovimientoStockEF(GestorContext db)
        {
            _db = db;
        }

        public void Add(MovimientoStock movimientoStock)
        {
            try
            {
                if(movimientoStock == null) { throw new MovimientoStockException("El
movimiento que desea agregar no puede ser nulo"); }
                _db.Entry(movimientoStock.Articulo).State = EntityState.Unchanged;
                _db.Entry(movimientoStock.TipoMovimiento).State = EntityState.Unchanged;
                _db.MovimientosStock.Add(movimientoStock);
                _db.SaveChanges();
            }
            catch (DbUpdateException ex)
            {
                if (ex.InnerException != null)
                {
                    SqlException exSql = ex.InnerException as SqlException;
                    if (exSql.Number == 2601)
                        throw new Exception("Ya existe un movimiento stock con ese nombre o
codigo");
                }
                throw;
            }
            catch (Exception ex)
            {
                throw new ArticuloException($"El articulo no se pudo agregar. Más info:
{ex.Message}");
            }
        }

        public IEnumerable<MovimientoStock> GetAll()
        {
            try
            {
                var movimientosStock = _db.MovimientosStock

```

```
        .Include(m => m.Articulo)
        .Include(m => m.TipoMovimiento)
        .ToList();
        if (movimientosStock == null || movimientosStock.Count == 0)
        {
            throw new MovimientoStockException("No existen movimientos stock
registrados");
        }
        return movimientosStock;
    }
    catch (Exception ex)
    {
        throw;
    }
}

public MovimientoStock GetById(int id)
{
    try
    {
        if (id == null || id == 0) throw new MovimientoStockException("El id del
movimientos stock que desea buscar no es valido");
        var movimientoStock = _db.MovimientosStock.FirstOrDefault(m => m.Id ==
id);
        if (movimientoStock == null) throw new MovimientoStockException("El
movimientos stock que esta buscando no existe");
        return movimientoStock;
    }
    catch (Exception ex)
    {
        throw new MovimientoStockException("No existe un movimientos stock con
ese id");
    }
}

public IEnumerable<MovimientoStock> GetMovimientosStockByArticuloAndTipo(int
idArticulo, int idTipoMovimiento, int numPagina, int cantidadRegistros)
{
    try
    {
        if (idArticulo == null || idArticulo == 0) { throw new
MovimientoStockException("El id del articulo no puede ser nulo"); }
        if (idTipoMovimiento == null || idArticulo == 0) { throw new
MovimientoStockException("El id del tipo de movimiento no puede ser nulo"); }
        var movimientosStock = _db.MovimientosStock
        .Include(m => m.Articulo)
        .Include(m => m.TipoMovimiento)
```

```

        .Where(m => m.Articuloid == idArticulo && m.TipoMovimientoid ==
idTipoMovimiento)
        .OrderByDescending(m => m.FechaMovimiento)
        .ThenBy(m => m.Cantidad)
        .ToList();

        if (numPagina <= 1)
        {
            numPagina = 1;
            _cantMovimientos = movimientosStock.Count();
        }

        int numRegistrosAnteriores = cantidadRegistros * (numPagina - 1);
        movimientosStock = movimientosStock
        .Skip(numRegistrosAnteriores)
        .Take(cantidadRegistros)
        .ToList();

        if (!movimientosStock.Any())
            _cantMovimientos = 0;

        if (movimientosStock == null || movimientosStock.Count() == 0)
        {
            throw new MovimientoStockException("No existen movimientos de stock para
ese articulo con ese tipo de movimiento");
        }
        return movimientosStock;
    }
    catch(Exception ex)
    {
        throw;
    }
}

public IEnumerable<Articulo> GetArticulosConMovimientosRangoFechas(DateTime
fechalnicio, DateTime fechaFin, int numPagina, int cantidadRegistros)
{
    try
    {
        if (fechalnicio == null || fechaFin == null) { throw new
MovimientoStockException("Las fechas no pueden ser nulas"); }

        var movimientosStock = _db.MovimientosStock
        .Where(m => m.FechaMovimiento >= fechalnicio && m.FechaMovimiento <=
fechaFin)
        .Select(m => m.Articulo)
        .Distinct()

```

```

        .ToList();
        if (numPagina <= 1)
        {
            _cantMovimientos = movimientosStock.Count();
        }

        int numRegistrosAnteriores = cantidadRegistros * (numPagina - 1);
        if(numRegistrosAnteriores < 0)
        {
            numRegistrosAnteriores = 0;
        }
        movimientosStock =
            movimientosStock
            .Skip(numRegistrosAnteriores)
            .Take(cantidadRegistros)
            .ToList();

        if (!movimientosStock.Any())
            _cantMovimientos = 0;

        if (movimientosStock == null || movimientosStock.Count() == 0)
        {
            throw new MovimientoStockException("No existen articulos que posean
movimientos de stock entre el rango de fechas");
        }
        return movimientosStock;
    }
    catch (Exception ex)
    {
        throw;
    }
}

```

//Obtener la información de resumen de las cantidades movidas agrupadas por año, y dentro de año por tipo de movimiento.

//Es la cantidad de movidos no el stock actual

```

public IEnumerable<dynamic> GetMovimientosCantidadesByTipoAndAno()
{
    try
    {

```

//agrupa las recetas por tiempo de elaboración

//.GroupBy(r => r.ElaboracionReceta.TiempoElaboracion)

var movimientosStock = _db.MovimientosStock

.Include(m => m.TipoMovimiento)

.AsEnumerable() // Cambia a evaluación en memoria

```

        .GroupBy(m => new { Year = m.FechaMovimiento.Year, TipoMovimiento =
m.TipoMovimiento })
        .Select(g => new
        {
            Ano = g.Key.Year,
            TipoMovimiento = g.Key.TipoMovimiento.Nombre,
            CantidadTotal = g.Sum(m => m.Cantidad)
        })
        .ToList();

        if (movimientosStock == null || movimientosStock.Count() == 0)
        {
            throw new MovimientoStockException("No existen movimientos de stock para
ese articulo con ese tipo de movimiento");
        }
        return movimientosStock;
    }
    catch (Exception ex)
    {
        throw;
    }
}
}
}
}

```

```

using Gestor.LogicaNegocio.Entidades;
using Gestor.LogicaNegocio.InterfacesRepositorios;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

```

```

namespace Gestor.AccesoDatos.EF
{
    public class RepositorioParametroEF : IRepositoryParametro
    {
        private GestorContext _db;

        public RepositorioParametroEF(GestorContext db)
        {
            _db = db;
        }
        public Parametro GetByName(string nombre)
        {
            try
            {

```

```
        Parametro parametro = _db.Parametros
        .SingleOrDefault(parametro => parametro.Nombre.Equals(nombre));
        return parametro;
    }
    catch (Exception ex)
    {
        throw;
    }
}

public int GetValorInt(string nombre)
{
    try
    {
        Parametro param = GetByName(nombre);
        if (param == null)
            throw new Exception("No hay un parámetro con ese valor");
        int num = 0;
        var ok = false;
        ok = int.TryParse(param.Valor, out num);
        return num;
    }
    catch (Exception ex)
    {
        throw;
    }
}

public string GetValorString(string nombre)
{
    try
    {
        Parametro param = GetByName(nombre);
        if (param == null)
            throw new Exception("No hay un parámetro con ese valor");

        return param.Valor;
    }
    catch (Exception ex)
    {
        throw;
    }
}

public DateTime GetValorDateTime(string nombre)
{
    try
    {
        Parametro param = GetByName(nombre);
```

```
        if (param == null)
            throw new Exception("No hay un parámetro con ese valor");

        return DateTime.Parse(param.Valor);
    }
    catch (Exception ex)
    {
        throw;
    }
}

public decimal GetValorDecimal(string nombre)
{
    try
    {
        Parametro param = GetByName(nombre);
        if (param == null)
            throw new Exception("No hay un parámetro con ese valor");

        return Decimal.Parse(param.Valor);
    }
    catch (Exception ex)
    {
        throw;
    }
}

public void Add(Parametro obj)
{
    if (obj == null) throw new ArgumentNullException("El parámetro no puede ser nulo");
    try
    {
        _db.Parametros.Add(obj);
        _db.SaveChanges();
    }
    catch (Exception ex)
    {
        throw;
    }
}

public IEnumerable<Parametro> GetAll()
{
    try
    {
        return _db.Parametros.ToList();
    }
}
```



```
}
catch (Exception ex)
{

    throw;

};
}

public Parametro GetById(int id)
{
    try
    {
        return _db.Parametros.Find(id);
    }
    catch (Exception ex)
    {

        throw;

    };
}

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

public void Remove(Parametro obj)
{
    if (obj == null) throw new ArgumentNullException("El parámetro no puede ser nulo");
    try
    {
        _db.Parametros.Remove(obj);
        _db.SaveChanges();
    }
    catch (Exception ex)
    {

        throw;

    }
}

public void Update(String nombreParametro, string valor)
{
    try
    {
        Parametro param = _db.Parametros.Find(nombreParametro);
        if (param == null)
            throw new NullReferenceException($"No existe un parámetro
{nombreParametro}");
    }
}
```

```
        param.Valor = valor;
        _db.Parametros.Update(param);
        _db.SaveChanges();

    }
    catch (Exception ex)
    {

        throw;
    }
}

}

using Gestor.LogicaNegocio.Articulos;
using Gestor.LogicaNegocio.Entidades;
using Gestor.LogicaNegocio.Excepciones;
using Gestor.LogicaNegocio.InterfacesRepositorios;
using Microsoft.Data.SqlClient;
using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Gestor.AccesoDatos.EF
{
    public class RepositorioTipoMovimientoEF : IRepositoryTipoMovimiento
    {
        private GestorContext _db;

        public RepositorioTipoMovimientoEF(GestorContext db)
        {
            _db = db;
        }

        public void Add(TipoMovimiento tipoMovimiento)
        {
            try
            {
                _db.TiposMovimientos.Add(tipoMovimiento);
                _db.SaveChanges();
            }
            catch (DbUpdateException ex)
            {
                if (ex.InnerException != null)
```

```
        {
            SqlException exSql = ex.InnerException as SqlException;
            if (exSql.Number == 2601)
                throw new TipoMovimientoException("Ya existe un tipo de movimiento con
ese nombre");
        }
        throw;
    }
    catch (Exception ex)
    {
        throw new TipoMovimientoException($"El tipo de movimiento no se pudo
agregar. Más info: {ex.Message}");
    }
}

public IEnumerable<TipoMovimiento> GetAll()
{
    try
    {
        return _db.TiposMovimientos.ToList();
    }
    catch (Exception ex)
    {
        throw new TipoMovimientoException($"No se pudo devolver la lista de tipos
de movimientos. Más info: {ex.Message}");
    }
}

public TipoMovimiento GetById(int id)
{
    try
    {
        var tipoMovimiento = _db.TiposMovimientos.Find(id);
        if (tipoMovimiento == null)
        {
            throw new TipoMovimientoException("El tipo de movimiento con ese id no
existe");
        }
        return tipoMovimiento;
    }
    catch (Exception e)
    {
        throw new TipoMovimientoException($"No se ha podido encontrar el articulo
con id {id}. Mas info: {e.Message}");
    }
}
```

```

    public void Remove(int id)
    {
        try
        {
            TipoMovimiento tipoMovimiento = _db.TiposMovimientos.Find(id);
            if (tipoMovimiento == null)
            {
                throw new TipoMovimientoException($"No existe un tipo de movimiento con
id:{id}.");
            }
            var movimientosConTipoMovimiento = _db.MovimientosStock.Where(m =>
m.TipoMovimiento.Id == tipoMovimiento.Id).ToList();
            if (movimientosConTipoMovimiento.Any())
            {
                throw new TipoMovimientoException($"Existen movimientos registrados con
ese tipo de movimiento, no se puede eliminar de la base datos.");
            }
            _db.TiposMovimientos.Remove(tipoMovimiento);
            _db.SaveChanges();
        }
        catch (Exception e)
        {
            throw new TipoMovimientoException($"Error. Mas info: {e.Message}");
        }
    }

    public void Remove(TipoMovimiento tipoMovimiento)
    {
        try
        {
            if (tipoMovimiento == null)
            {
                throw new ArgumentNullException("El tipo de movimiento no puede ser
nulo");
            }
            _db.TiposMovimientos.Remove(tipoMovimiento);
            _db.SaveChanges();
        }
        catch (Exception ex)
        {
            throw new TipoMovimientoException($"No eliminar el tipo de movimiento
{tipoMovimiento.Nombre} de la lista. Más info: {ex.Message}");
        }
    }

    public void Update(int id, TipoMovimiento tipoMovimientoModificado)
    {

```

```

        try
        {
            var tipoMovimiento = _db.TiposMovimientos.Find(id);
            if (tipoMovimiento == null)
            {
                throw new TipoMovimientoException($"No existe el tipo de movimiento con el
id {id}");
            }
            tipoMovimiento.Modificar(tipoMovimientoModificado);
            _db.SaveChanges();
        }
        catch (Exception ex)
        {
            throw new TipoMovimientoException($"No se ha podido modificar el tipo de
movimiento con id: {id}. Mas info: {ex.Message}");
        }
    }
}

```

```

using Gestor.LogicaNegocio.InterfacesRepositorios;
using SistemaAutenticacion.Entidades;
using SistemaAutenticacion.Exceptions.Usuario;
using SistemaAutenticacion.ValueObject;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

```

```

namespace Gestor.AccesoDatos.EF
{
    public class RepositorioUsuarioEF : IRepositoryUsuario
    {
        private GestorContext _db;

        public RepositorioUsuarioEF(GestorContext db)
        {
            _db = db;
        }

        public Usuario GetById(int id)
        {
            throw new NotImplementedException();
        }

        public Usuario GetByUsuarioLogin(string email, string password)

```

```
{
    try
    {
        if (string.IsNullOrEmpty(email) || string.IsNullOrEmpty(password)) throw new
ArgumentException("El usuario y/o contraseñas no pueden ser nulos");
        string pass = Password.Encrypt(password);
        var consulta = _db.Usuarios.AsQueryable();

        var usuario = consulta
            .AsEnumerable()
            .SingleOrDefault(u => u.Email.Valor.Equals(email) &&
u.Password.Encriptada.Equals(pass));

        if (usuario == null) throw new UsuarioException($"Usuario y/o password
invalidos");
        return usuario;
    }
    catch (Exception ex)
    {
        throw ex;
    }
}

public void Add(Usuario obj)
{
}

public void Update(int id, Usuario obj)
{
    throw new NotImplementedException();
}

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

public void Remove(Usuario obj)
{
    throw new NotImplementedException();
}

public IEnumerable<Usuario> GetAll()
{
    throw new NotImplementedException();
}
```

```

    }
}

using System;
using Microsoft.EntityFrameworkCore.Migrations;

#nullable disable

namespace Gestor.AccesoDatos.Migrations
{
    /// <inheritdoc />
    public partial class prueba1 : Migration
    {
        /// <inheritdoc />
        protected override void Up(MigrationBuilder migrationBuilder)
        {
            migrationBuilder.CreateTable(
                name: "Articulos",
                columns: table => new
                {
                    Id = table.Column<int>(type: "int", nullable: false)
                        .Annotation("SqlServer:Identity", "1, 1"),
                    Nombre = table.Column<string>(type: "nvarchar(450)", nullable: false),
                    Descripcion = table.Column<string>(type: "nvarchar(max)", nullable: false),
                    Codigo = table.Column<string>(type: "nvarchar(450)", nullable: false),
                    PrecioActual = table.Column<decimal>(type: "decimal(10,2)", nullable: false)
                },
                constraints: table =>
                {
                    table.PrimaryKey("PK_Articulos", x => x.Id);
                });

            migrationBuilder.CreateTable(
                name: "TiposMovimientos",
                columns: table => new
                {
                    Id = table.Column<int>(type: "int", nullable: false)
                        .Annotation("SqlServer:Identity", "1, 1"),
                    Nombre = table.Column<string>(type: "nvarchar(450)", nullable: false),
                    Coeficiente = table.Column<int>(type: "int", nullable: false)
                },
                constraints: table =>
                {
                    table.PrimaryKey("PK_TiposMovimientos", x => x.Id);
                });

            migrationBuilder.CreateTable(

```

```

        name: "Usuarios",
        columns: table => new
        {
            Id = table.Column<int>(type: "int", nullable: false)
                .Annotation("SqlServer:Identity", "1, 1"),
            Nombre = table.Column<string>(type: "nvarchar(max)", nullable: false),
            Apellido = table.Column<string>(type: "nvarchar(max)", nullable: false),
            Email = table.Column<string>(type: "nvarchar(450)", nullable: false),
            Discriminator = table.Column<string>(type: "nvarchar(13)", maxLength: 13,
nullable: false),
            Password_Encryptada = table.Column<string>(type: "nvarchar(max)",
nullable: false),
            Password_Valor = table.Column<string>(type: "nvarchar(max)", nullable:
false)
        },
        constraints: table =>
        {
            table.PrimaryKey("PK_Usuarios", x => x.Id);
        });

```

```

migrationBuilder.CreateTable(
    name: "MovimientosStock",
    columns: table => new
    {
        Id = table.Column<int>(type: "int", nullable: false)
            .Annotation("SqlServer:Identity", "1, 1"),
        FechaMovimiento = table.Column<DateTime>(type: "Date", nullable: false),
        ArticuloId = table.Column<int>(type: "int", nullable: false),
        TipoMovimientoId = table.Column<int>(type: "int", nullable: false)
    },
    constraints: table =>
    {
        table.PrimaryKey("PK_MovimientosStock", x => x.Id);
        table.ForeignKey(
            name: "FK_MovimientosStock_Articulos_ArticuloId",
            column: x => x.ArticuloId,
            principalTable: "Articulos",
            principalColumn: "Id",
            onDelete: ReferentialAction.Cascade);
        table.ForeignKey(
            name: "FK_MovimientosStock_TiposMovimientos_TipoMovimientoId",
            column: x => x.TipoMovimientoId,
            principalTable: "TiposMovimientos",
            principalColumn: "Id",
            onDelete: ReferentialAction.Cascade);
    });

```



```
migrationBuilder.CreateIndex(
    name: "IX_Articulos_Codigo",
    table: "Articulos",
    column: "Codigo",
    unique: true);

migrationBuilder.CreateIndex(
    name: "IX_Articulos_Nombre",
    table: "Articulos",
    column: "Nombre",
    unique: true);

migrationBuilder.CreateIndex(
    name: "IX_MovimientosStock_Articuloid",
    table: "MovimientosStock",
    column: "Articuloid");

migrationBuilder.CreateIndex(
    name: "IX_MovimientosStock_TipoMovimientoid",
    table: "MovimientosStock",
    column: "TipoMovimientoid");

migrationBuilder.CreateIndex(
    name: "IX_TiposMovimientos_Nombre",
    table: "TiposMovimientos",
    column: "Nombre",
    unique: true);

migrationBuilder.CreateIndex(
    name: "IX_Usuarios_Email",
    table: "Usuarios",
    column: "Email",
    unique: true);
}

/// <inheritdoc />
protected override void Down(MigrationBuilder migrationBuilder)
{
    migrationBuilder.DropTable(
        name: "MovimientosStock");

    migrationBuilder.DropTable(
        name: "Usuarios");

    migrationBuilder.DropTable(
        name: "Articulos");
}
```

```

        migrationBuilder.DropTable(
            name: "TiposMovimientos");
    }
}

using Microsoft.EntityFrameworkCore.Migrations;

#nullable disable

namespace Gestor.AccesoDatos.Migrations
{
    /// <inheritdoc />
    public partial class updatedb : Migration
    {
        /// <inheritdoc />
        protected override void Up(MigrationBuilder migrationBuilder)
        {
            migrationBuilder.AddColumn<int>(
                name: "Cantidad",
                table: "MovimientosStock",
                type: "int",
                nullable: false,
                defaultValue: 0);

            migrationBuilder.CreateTable(
                name: "Parametros",
                columns: table => new
                {
                    Nombre = table.Column<string>(type: "nvarchar(450)", nullable: false),
                    Valor = table.Column<string>(type: "nvarchar(max)", nullable: false)
                },
                constraints: table =>
                {
                    table.PrimaryKey("PK_Parametros", x => x.Nombre);
                });
        }

        /// <inheritdoc />
        protected override void Down(MigrationBuilder migrationBuilder)
        {
            migrationBuilder.DropTable(
                name: "Parametros");

            migrationBuilder.DropColumn(
                name: "Cantidad",
                table: "MovimientosStock");
        }
    }
}

```

```

    }
    }
}

using Microsoft.EntityFrameworkCore.Migrations;

#nullable disable

namespace Gestor.AccesoDatos.Migrations
{
    /// <inheritdoc />
    public partial class updateNuevo : Migration
    {
        /// <inheritdoc />
        protected override void Up(MigrationBuilder migrationBuilder)
        {

        }

        /// <inheritdoc />
        protected override void Down(MigrationBuilder migrationBuilder)
        {

        }
    }
}

// <auto-generated />
using System;
using System.Collections.Generic;
using Gestor.AccesoDatos.EF;
using Microsoft.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore.Infrastructure;
using Microsoft.EntityFrameworkCore.Metadata;
using Microsoft.EntityFrameworkCore.Storage.ValueConversion;

#nullable disable

namespace Gestor.AccesoDatos.Migrations
{
    [DbContext(typeof(GestorContext))]
    partial class GestorContextModelSnapshot : ModelSnapshot
    {
        protected override void BuildModel(ModelBuilder modelBuilder)
        {

#pragma warning disable 612, 618
            modelBuilder

```

```
.HasAnnotation("ProductVersion", "8.0.6")
.HasAnnotation("Relational:MaxIdentifierLength", 128);

SqlServerModelBuilderExtensions.UseIdentityColumns(modelBuilder);

modelBuilder.Entity("Gestor.LogicaNegocio.Articulos.Articulo", b =>
{
    b.Property<int>("Id")
    .ValueGeneratedOnAdd()
    .HasColumnType("int");

    SqlServerPropertyBuilderExtensions.UseIdentityColumn(b.Property<int>("Id"));

    b.Property<string>("Codigo")
    .IsRequired()
    .HasColumnType("nvarchar(450)");

    b.Property<string>("Descripcion")
    .IsRequired()
    .HasColumnType("nvarchar(max)");

    b.Property<string>("Nombre")
    .IsRequired()
    .HasColumnType("nvarchar(450)");

    b.Property<decimal>("PrecioActual")
    .HasColumnType("decimal(10,2)");

    b.HasKey("Id");

    b.HasIndex("Codigo")
    .IsUnique();

    b.HasIndex("Nombre")
    .IsUnique();

    b.ToTable("Articulos");
});

modelBuilder.Entity("Gestor.LogicaNegocio.Entidades.MovimientoStock", b =>
{
    b.Property<int>("Id")
    .ValueGeneratedOnAdd()
    .HasColumnType("int");
```

```
SqlServerPropertyBuilderExtensions.UseIdentityColumn(b.Property<int>("Id"));
```

```
    b.Property<int>("Articuloid")  
    .HasColumnType("int");
```

```
    b.Property<int>("Cantidad")  
    .HasColumnType("int");
```

```
    b.Property<DateTime>("FechaMovimiento")  
    .HasColumnType("Date");
```

```
    b.Property<int>("TipoMovimientoid")  
    .HasColumnType("int");
```

```
    b.HasKey("Id");
```

```
    b.HasIndex("Articuloid");
```

```
    b.HasIndex("TipoMovimientoid");
```

```
    b.ToTable("MovimientosStock");  
});
```

```
modelBuilder.Entity("Gestor.LogicaNegocio.Entidades.Parametro", b =>
```

```
    {  
    b.Property<string>("Nombre")  
    .HasColumnType("nvarchar(450)");
```

```
    b.Property<string>("Valor")  
    .IsRequired()  
    .HasColumnType("nvarchar(max)");
```

```
    b.HasKey("Nombre");
```

```
    b.ToTable("Parametros");  
});
```

```
modelBuilder.Entity("Gestor.LogicaNegocio.Entidades.TipoMovimiento", b =>
```

```
    {  
    b.Property<int>("Id")  
    .ValueGeneratedOnAdd()  
    .HasColumnType("int");
```

```
SqlServerPropertyBuilderExtensions.UseIdentityColumn(b.Property<int>("Id"));
```

```
b.Property<int>("Coeficiente")
.HasColumnType("int");

b.Property<string>("Nombre")
.IsRequired()
.HasColumnType("nvarchar(450)");

b.HasKey("Id");

b.HasIndex("Nombre")
.IsUnique();

b.ToTable("TiposMovimientos");
});

modelBuilder.Entity("SistemaAutenticacion.Entidades.Usuario", b =>
{
    b.Property<int>("Id")
        .ValueGeneratedOnAdd()
        .HasColumnType("int");

    SqlServerPropertyBuilderExtensions.UseIdentityColumn(b.Property<int>("Id"));

    b.Property<string>("Apellido")
        .IsRequired()
        .HasColumnType("nvarchar(max)");

    b.Property<string>("Discriminator")
        .IsRequired()
        .HasMaxLength(13)
        .HasColumnType("nvarchar(13)");

    b.Property<string>("Email")
        .IsRequired()
        .HasColumnType("nvarchar(450)");

    b.Property<string>("Nombre")
        .IsRequired()
        .HasColumnType("nvarchar(max)");

    b.ComplexProperty<Dictionary<string, object>>("Password",
        "SistemaAutenticacion.Entidades.Usuario.Password#Password", b1 =>
    {
        b1.IsRequired();

        b1.Property<string>("Encriptada")
```

```

        .IsRequired()
        .HasColumnType("nvarchar(max)");

        b1.Property<string>("Valor")
        .IsRequired()
        .HasColumnType("nvarchar(max)");
    });

    b.HasKey("Id");

    b.HasIndex("Email")
    .IsUnique();

    b.ToTable("Usuarios");

    b.HasDiscriminator<string>("Discriminator").HasValue("Usuario");

    b.UseTphMappingStrategy();
    });

modelBuilder.Entity("SistemaAutenticacion.Entidades.Administrador", b =>
{
    b.HasBaseType("SistemaAutenticacion.Entidades.Usuario");

    b.HasDiscriminator().HasValue("Administrador");
});

modelBuilder.Entity("SistemaAutenticacion.Entidades.Encargado", b =>
{
    b.HasBaseType("SistemaAutenticacion.Entidades.Usuario");

    b.HasDiscriminator().HasValue("Encargado");
});

modelBuilder.Entity("Gestor.LogicaNegocio.Entidades.MovimientoStock", b =>
{
    b.HasOne("Gestor.LogicaNegocio.Articulos.Articulo", "Articulo")
    .WithMany("HistorialMovimientos")
    .HasForeignKey("Articuloid")
    .OnDelete(DeleteBehavior.Cascade)
    .IsRequired();

    b.HasOne("Gestor.LogicaNegocio.Entidades.TipoMovimiento",
    "TipoMovimiento")
    .WithMany()
    .HasForeignKey("TipoMovimientoid")
    .OnDelete(DeleteBehavior.Cascade)

```

```

        .IsRequired();

        b.Navigation("Articulo");

        b.Navigation("TipoMovimiento");
    });

    modelBuilder.Entity("Gestor.LogicaNegocio.Articulos.Articulo", b =>
    {
        b.Navigation("HistorialMovimientos");
    });
#pragma warning restore 612, 618
    }
}
}

```

```

INSERT INTO TiposMovimientos (Nombre, Coeficiente) VALUES
('Venta', -1),
('Compra', 1),
('Devolución de Cliente', 1),
('Devolución a Proveedor', -1),
('Ajuste por Inventario', 1),
('Ajuste por Pérdida', -1),
('Transferencia entre Almacenes', 1),
('Consumo Interno', -1),
('Elementos Rotos', -1),
('Venta Mayor', -1)

```

```

--SELECT * FROM INFORMATION_SCHEMA.TABLES
GO

```

```

INSERT INTO Articulos (Nombre, Descripcion, Codigo, PrecioActual) VALUES
('Lapiz HB', 'Lapiz de madera', '00000000000001', 20.00),
('Bolígrafo Azul', 'Bolígrafo tinta azul', '00000000000002', 30.00),
('Cuaderno A4', 'Cuaderno espiral A4', '00000000000003', 100.00),
('Goma de Borrar', 'Goma para borrar lapiz', '00000000000004', 12.00),
('Regla 30cm', 'Regla de plastico', '00000000000005', 40.00),
('Tijeras Escolares', 'Tijeras punta redonda', '00000000000006', 48.00),
('Sacapuntas', 'Sacapuntas de metal', '00000000000007', 16.00),
('Marcador Permanente', 'Marcador tinta negra', '00000000000008', 60.00),
('Cartulina Blanca', 'Cartulina tamaño A3', '00000000000009', 32.00),
('Resaltador Amarillo', 'Resaltador fluorescente', '00000000000010', 36.00),
('Carpeta A4', 'Carpeta con anillas', '00000000000011', 120.00),
('Bloc de Notas', 'Bloc de notas adhesivas', '00000000000012', 48.00),
('Cinta Adhesiva', 'Cinta adhesiva transparente', '00000000000013', 28.00),
('Pegamento en Barra', 'Pegamento no toxico', '00000000000014', 40.00),

```


('Lapices de Colores', 'Caja de 12 lapices de colores', '0000000000015', 180.00),
 ('Papel Bond A4', 'Resma de 500 hojas', '0000000000016', 200.00),
 ('Portaminas', 'Portaminas con minas HB', '0000000000017', 80.00),
 ('Corrector Liquido', 'Corrector liquido blanco', '0000000000018', 60.00),
 ('Libreta de Bolsillo', 'Libreta de bolsillo con rayas', '0000000000019', 72.00),
 ('Compás Escolar', 'Compás de metal', '0000000000020', 100.00),
 ('Marcador de Pizarra', 'Marcador borrable tinta azul', '0000000000021', 60.00),
 ('Notas Adhesivas', 'Paquete de 100 notas adhesivas', '0000000000022', 40.00),
 ('Boligrafo Negro', 'Boligrafo tinta negra', '0000000000023', 30.00),
 ('Resaltador Verde', 'Resaltador fluorescente verde', '0000000000024', 36.00),
 ('Carpeta Oficio', 'Carpeta con anillas tamaño oficio', '0000000000025', 140.00),
 ('Separadores de Carpetas', 'Paquete de 10 separadores', '0000000000026', 48.00),
 ('Archivador A4', 'Archivador de palanca', '0000000000027', 160.00),
 ('Marcador de Pizarra Rojo', 'Marcador borrable tinta roja', '0000000000028', 60.00),
 ('Borrador de Pizarra', 'Borrador para pizarra blanca', '0000000000029', 40.00),
 ('Papel Fotocopia', 'Resma de 500 hojas A4', '0000000000030', 180.00),
 ('Libreta de Espiral', 'Libreta de espiral tamaño A5', '0000000000031', 80.00),
 ('Estuche Escolar', 'Estuche de tela con cremallera', '0000000000032', 120.00),
 ('Regla Flexible', 'Regla de plastico flexible', '0000000000033', 60.00),
 ('Perforadora de Papel', 'Perforadora de 2 agujeros', '0000000000034', 100.00),
 ('Grapadora', 'Grapadora de metal', '0000000000035', 120.00),
 ('Caja de Grapas', 'Caja de 1000 grapas', '0000000000036', 40.00),
 ('Rotulador Fino', 'Rotulador punta fina', '0000000000037', 48.00),
 ('Papel Adhesivo', 'Paquete de 10 hojas adhesivas', '0000000000038', 80.00),
 ('Rotulador Permanente Rojo', 'Rotulador tinta roja', '0000000000039', 60.00),
 ('Talonario de Recibos', 'Talonario con 50 hojas', '0000000000040', 80.00),
 ('Cuaderno de Dibujo', 'Cuaderno de dibujo A4', '0000000000041', 140.00),
 ('Cinta Correctora', 'Cinta correctora blanca', '0000000000042', 72.00),
 ('Lápiz de Carpintero', 'Lápiz de carpintero rojo', '0000000000043', 32.00),
 ('Tijeras Profesionales', 'Tijeras de acero inoxidable', '0000000000044', 200.00),
 ('Punzón Escolar', 'Punzón de metal con mango', '0000000000045', 60.00),
 ('Regla Metálica', 'Regla de metal 30 cm', '0000000000046', 80.00),
 ('Boligrafo Rojo', 'Boligrafo tinta roja', '0000000000047', 30.00),
 ('Cuaderno Tapa Dura', 'Cuaderno tamaño A4 con tapa dura', '0000000000048', 140.00),
 ('Cinta Adhesiva Doble Cara', 'Cinta adhesiva doble cara', '0000000000049', 48.00),
 ('Libreta de Apuntes', 'Libreta de apuntes tamaño A5', '0000000000050', 100.00),
 ('Marcador Verde', 'Marcador tinta verde', '0000000000051', 60.00),
 ('Boligrafo Verde', 'Boligrafo tinta verde', '0000000000052', 30.00),
 ('Corrector en Cinta', 'Corrector en cinta blanca', '0000000000053', 60.00),
 ('Papel Crepe', 'Paquete de 10 hojas de papel crepe', '0000000000054', 80.00),
 ('Carpeta Plastica', 'Carpeta plastica con cierre', '0000000000055', 120.00),
 ('Goma de Borrar para Tinta', 'Goma de borrar tinta', '0000000000056', 20.00),
 ('Boligrafo Multicolor', 'Boligrafo con 4 colores', '0000000000057', 60.00),
 ('Rotulador Azul', 'Rotulador tinta azul', '0000000000058', 60.00),
 ('Portafolios', 'Portafolios de piel sintética', '0000000000059', 400.00),
 ('Plastilina', 'Paquete de 12 barras de plastilina', '0000000000060', 120.00),

('Cuaderno de Matemáticas', 'Cuaderno cuadriculado tamaño A4', '00000000000061', 100.00),
 ('Papel de Envoltura', 'Rollo de papel de envoltura', '00000000000062', 100.00),
 ('Cinta Decorativa', 'Cinta decorativa adhesiva', '00000000000063', 60.00),
 ('Pegamento Liquido', 'Pegamento liquido transparente', '00000000000064', 40.00),
 ('Papel de Seda', 'Paquete de 10 hojas de papel de seda', '00000000000065', 80.00),
 ('Cizalla de Papel', 'Cizalla de papel manual', '00000000000066', 480.00),
 ('Grapadora de Oficina', 'Grapadora de oficina metálica', '00000000000067', 160.00),
 ('Punzón de Plástico', 'Punzón con mango de plástico', '00000000000068', 40.00),
 ('Caja de Clips', 'Caja de 100 clips metálicos', '00000000000069', 20.00),
 ('Caja de Gomas Elásticas', 'Caja de 50 gomas elásticas', '00000000000070', 40.00),
 ('Cinta de Embalar', 'Cinta adhesiva para embalar', '00000000000071', 80.00),
 ('Sobres de Papel', 'Paquete de 25 sobres de papel', '00000000000072', 60.00),
 ('Cuaderno de Bocetos', 'Cuaderno de bocetos tamaño A4', '00000000000073', 160.00),
 ('Caja de Tizas', 'Caja de 12 tizas blancas', '00000000000074', 48.00),
 ('Marcador de Textil', 'Marcador de tinta para textil', '00000000000075', 50.00),
 ('Carpeta de Dibujo', 'Carpeta de dibujo tamaño A3', '00000000000076', 180.00),
 ('Papel de Calcar', 'Resma de papel de calcar', '00000000000077', 120.00),
 ('Estuche de Pinturas', 'Estuche de 12 colores de pinturas', '00000000000078', 200.00),
 ('Calculadora Científica', 'Calculadora científica de 12 dígitos', '00000000000079', 300.00),
 ('Set de Geometría', 'Set de geometría con regla, escuadra y transportador', '00000000000080', 150.00);

GO

-- Movimientos de stock hardcodeados

INSERT INTO MovimientosStock (ArticuloId, TipoMovimientoId, Cantidad, FechaMovimiento) VALUES

(1, 1, 20, '2024-01-05'),
 (2, 2, 15, '2024-01-10'),
 (3, 3, 30, '2024-01-15'),
 (4, 4, 25, '2024-01-20'),
 (5, 5, 18, '2024-01-25'),
 (6, 6, 22, '2024-02-05'),
 (7, 7, 17, '2024-02-10'),
 (8, 8, 14, '2024-02-15'),
 (9, 9, 28, '2024-02-20'),
 (10, 10, 24, '2024-02-25'),
 (11, 1, 19, '2024-03-05'),
 (12, 2, 13, '2024-03-10'),
 (13, 3, 29, '2024-03-15'),
 (14, 4, 16, '2024-03-20'),
 (15, 5, 21, '2024-03-25'),
 (16, 6, 26, '2024-04-05'),
 (17, 7, 23, '2024-04-10'),
 (1, 8, 27, '2024-04-15'),
 (2, 9, 12, '2024-04-20'),

```
(3, 10, 20, '2024-04-25'),  
(4, 1, 15, '2024-05-05'),  
(5, 2, 18, '2024-05-10'),  
(6, 3, 24, '2024-05-15'),  
(7, 4, 16, '2024-05-20'),  
(8, 5, 21, '2024-05-25'),  
(9, 6, 26, '2024-06-05'),  
(10, 7, 13, '2024-06-10'),  
(11, 8, 29, '2024-06-15'),  
(12, 9, 17, '2024-06-20'),  
(13, 10, 22, '2024-06-25'),  
(14, 1, 20, '2024-07-05'),  
(15, 2, 15, '2024-07-10'),  
(16, 3, 30, '2024-07-15'),  
(17, 4, 25, '2024-07-20'),  
(1, 5, 18, '2024-07-25'),  
(2, 6, 22, '2024-08-05'),  
(3, 7, 17, '2024-08-10'),  
(4, 8, 14, '2024-08-15'),  
(5, 9, 28, '2024-08-20'),  
(6, 10, 24, '2024-08-25'),  
(7, 1, 19, '2024-09-05'),  
(8, 2, 13, '2024-09-10'),  
(9, 3, 29, '2024-09-15'),  
(10, 4, 16, '2024-09-20'),  
(11, 5, 21, '2024-09-25'),  
(12, 6, 26, '2024-10-05'),  
(13, 7, 23, '2024-10-10'),  
(14, 8, 27, '2024-10-15'),  
(15, 9, 12, '2024-10-20'),  
(16, 10, 20, '2024-10-25'),  
(17, 1, 15, '2024-11-05'),  
(1, 2, 18, '2024-11-10'),  
(2, 3, 24, '2024-11-15'),  
(3, 4, 16, '2024-11-20'),  
(4, 5, 21, '2024-11-25'),  
(5, 6, 26, '2024-12-05'),  
(6, 7, 13, '2024-12-10'),  
(7, 8, 29, '2024-12-15'),  
(8, 9, 17, '2024-12-20'),  
(9, 10, 22, '2024-12-25');
```

```
INSERT INTO Parametros(Nombre, Valor) VALUES  
('TopeCantidad', 1500),  
('TopePaginas', 10)
```

```
SELECT * FROM Parametros
```

```
--/DBCC CHECKIDENT (MovimientosStock, RESEED, 0);
```

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations.Schema;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

```
namespace Gestor.LogicaAplicacion.DTO.Articulo
```

```
{
    public class DtoArticuloCompleto
    {
        [Required]
        public int Id { get; set; }
        [Required]
        public string Nombre { get; set; }
        [Required]
        public string Descripcion { get; set; }
        [Required]
        public stringCodigo { get; set; }

        [Required]
        [Column(TypeName = "decimal(10,2)")]
        public decimal PrecioActual { get; set; }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

```
namespace Gestor.LogicaAplicacion.DTO.MovimientoStock
```

```
{
    public class DtoArticulosFiltradosFechas
    {
        public DateTime FechaInicio { get; set; }
    }
}
```

```
        public DateTime FechaFin { get; set; }

        public int Pagina { get; set; }
    }
}

using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Gestor.LogicaAplicacion.DTO.MovimientoStock
{
    public class DtoMovimientoIndex
    {
        public string NombreArticulo { get; set; }

        [Required]
        public int Articulold { get; init; }

        public string NombreTipoMovimiento { get; set; }

        [Required]
        public int TipoMovimientold { get; init; }

        [Required]
        public int Cantidad { get; init; }
    }
}

using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Gestor.LogicaAplicacion.DTO.MovimientoStock
{
    public class DtoMovimientoStockAdd
    {
        [Required]
        public int Articulold { get; init; }
```

```
[Required]
public int TipoMovimientoId { get; init; }

[Required]
public int Cantidad { get; init; }
}

}

using Gestor.LogicaNegocio.Articulos;
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations.Schema;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Gestor.LogicaAplicacion.DTO.MovimientoStock
{
    public class DtoMovimientoStockConArticuloTipoMovimiento
    {
        public int Id { get; set; }

        public int Cantidad { get; set; }

        [DataType(DataType.Date)]
        [Column(TableName = "Date")]
        public DateTime FechaMovimiento { get; set; }

        //Articulo
        public int IdArticulo { get; set; }
        public string NombreArticulo { get; set; }

        [Column(TableName = "decimal(10,2)")]
        public decimal PrecioActualArticulo { get; set; }

        //TipoMovimiento

        public int IdTipoMovimiento { get; set; }
        public string NombreTipoMovimiento { get; set; }

        public int CoeficienteTipoMovimiento { get; set; }

    }
}
```

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Gestor.LogicaAplicacion.DTO.TipoMovimiento
{
    public class DtoTipoMovimientoAdd
    {

        [Required]
        public string Nombre { get; set; }

        [Required]
        public int Coeficiente { get; set; }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Gestor.LogicaAplicacion.DTO.TipoMovimiento
{
    public class DtoTipoMovimientoCompleto
    {
        public int Id { get; set; }
        public string Nombre { get; set; }

        public int Coeficiente { get; set; }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Gestor.LogicaAplicacion.DTO.Usuario
{
    public class DtoUsuarioLogin
    {
    }
}
```

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

        public string Password { get; set; }
        public string? Role { get; set; }
    }
}

using Gestor.LogicaAplicacion.DTO.Articulo;
using Gestor.LogicaAplicacion.InterfacesCU.Articulo;
using Gestor.LogicaAplicacion.Mappers;
using Gestor.LogicaNegocio.InterfacesRepositorios;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Gestor.LogicaAplicacion.ImplementacionCU.Articulo
{
    public class GetArticulosAlfabeticamente : IGetArticulosAlfabeticamente
    {
        private IRepositoryArticulo _repoArticulo;

        public GetArticulosAlfabeticamente(IRepositoryArticulo repoArticulo)
        {
            _repoArticulo = repoArticulo;
        }

        public IEnumerable<DtoArticuloCompleto> Ejecutar()
        {
            var articulos = _repoArticulo.GetArticulosOrdenadosAlfabeticamente();
            if (articulos == null) throw new ArgumentNullException("No existen articulos registrados");

            return MapperArticulo.FromLista(articulos);
        }
    }
}

using Gestor.LogicaAplicacion.DTO.MovimientoStock;
using Gestor.LogicaAplicacion.InterfacesCU.MovimientoStock;
using Gestor.LogicaAplicacion.Mappers;
using Gestor.LogicaNegocio.Excepciones;
using Gestor.LogicaNegocio.InterfacesRepositorios;
using System;
using System.Collections.Generic;
```



```

using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Gestor.LogicaAplicacion.ImplementacionCU.MovimientoStock
{
    public class AddMovimientoStock : IAddMovimientoStock
    {
        private IRepositoryMovimientoStock _repoMovimientoStock;
        private IRepositoryArticulo _repoArticulo;
        private IRepositoryTipoMovimiento _repoTipoMovimiento;
        private IRepositoryParametro _repoParametros;

        public AddMovimientoStock(IRepositoryMovimientoStock repoMovimientoStock,
        IRepositoryTipoMovimiento repoTipoMovimiento, IRepositoryArticulo repoArticulo,
        IRepositoryParametro repoParametros)
        {
            _repoMovimientoStock = repoMovimientoStock;
            _repoTipoMovimiento = repoTipoMovimiento;
            _repoArticulo = repoArticulo;
            _repoParametros = repoParametros;
        }

        public int Ejecutar(DtoMovimientoStockAdd dto)
        {
            try
            {
                //Doble comprobacion de cantidad tambien existe en la validacion IValidate
                de Movimiento Stock
                if (dto == null || dto.Cantidad <= 0)
                {
                    throw new MovimientoStockException("El movimiento de stock no puede ser
                    nulo ni tampoco la cantidad menor o igual a 0");
                }
                LogicaNegocio.Entidades.MovimientoStock movimientoStock =
                MapperMovimientoStock.FromDtoAdd(dto);
                if(movimientoStock == null)
                {
                    throw new MovimientoStockException("El movimiento de stock no puede ser
                    nulo, problema mapeo");
                }
                var articulo = _repoArticulo.GetById(movimientoStock.Articuloid);
                if(articulo == null)
                {
                    throw new MovimientoStockException("El articulo del movimiento de stock no
                    puede ser nulo");
                }
            }
        }
    }
}

```

```

        var tipoMovimiento =
        _repoTipoMovimiento.GetById(movimientoStock.TipoMovimientoId);
        if (tipoMovimiento == null)
        {
            throw new MovimientoStockException("El tipo de movimiento del movimiento
de stock no puede ser nulo");
        }
        LogicaNegocio.Entidades.MovimientoStock.s_TopeCantidad =
        _repoParametros.GetValorInt("TopeCantidad");
        //Verifica que la cantidad no supere el tope
        movimientoStock.VerificarTope();
        movimientoStock.TipoMovimiento = tipoMovimiento;
        movimientoStock.Articulo = articulo;
        _repoMovimientoStock.Add(movimientoStock);
        return movimientoStock.Id;
    }
    catch (Exception ex)
    {
        throw;
    }
}
}
}

```

```

using Gestor.LogicaAplicacion.DTO.Articulo;
using Gestor.LogicaAplicacion.DTO.MovimientoStock;
using Gestor.LogicaAplicacion.InterfacesCU.MovimientoStock;
using Gestor.LogicaAplicacion.Mappers;
using Gestor.LogicaNegocio.Excepciones;
using Gestor.LogicaNegocio.InterfacesRepositorios;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

```

```

namespace Gestor.LogicaAplicacion.ImplementacionCU.MovimientoStock
{
    public class GetArticulosByMovimientosPorFecha :
    IGetArticulosByMovimientosPorFecha
    {
        private IRepositoryMovimientoStock _repoMovimientos;

        public GetArticulosByMovimientosPorFecha(IRepositoryMovimientoStock
repoMovimientos)
        {

```

```

        _repoMovimientos = repoMovimientos;
    }

    public IEnumerable<DtoArticuloCompleto> Ejecutar(DateTime fechaInicio, DateTime
fechaFin, int numPagina, int cantidadRegistros)
    {
        if(fechaFin == DateTime.MinValue || fechaFin == DateTime.MinValue)
        {
            throw new MovimientoStockException("Las fechas no pueden ser nulas");
        }
        var articulos =
        _repoMovimientos.GetArticulosConMovimientosRangoFechas(fechaInicio, fechaFin,
numPagina, cantidadRegistros);
        if(articulos == null)
        {
            throw new MovimientoStockException("No existen articulos que poseean
movimientos en el rango de fechas seleccionado");
        }
        return MapperArticulo.FromLista(articulos);
    }
}

```

```

using Gestor.LogicaAplicacion.DTO.MovimientoStock;
using Gestor.LogicaAplicacion.InterfacesCU.MovimientoStock;
using Gestor.LogicaAplicacion.Mappers;
using Gestor.LogicaNegocio.Excepciones;
using Gestor.LogicaNegocio.InterfacesRepositorios;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

```

```

namespace Gestor.LogicaAplicacion.ImplementacionCU.MovimientoStock
{
    public class GetMovimientosStockByArticuloAndTipo :
IGetMovimientosStockByArticuloAndTipo
    {
        private IRepositoryMovimientoStock _repoMovimientos;
        private IRepositoryArticulo _repoArticulos;
        private IRepositoryTipoMovimiento _repoTipos;
        public GetMovimientosStockByArticuloAndTipo(IRepositoryMovimientoStock
repoMovimientos, IRepositoryArticulo repoArticulos, IRepositoryTipoMovimiento repoTipos)
        {
            _repoMovimientos = repoMovimientos;
            _repoArticulos = repoArticulos;
        }
    }
}

```

```

        _repoTipos = repoTipos;
    }

    public IEnumerable<DtoMovimientoStockConArticuloTipoMovimiento> Ejecutar(int
idArticulo, int idTipoMovimiento, int numPagina, int cantidadRegistros)
    {
        if(idArticulo == null || idArticulo == 0)
        {
            throw new MovimientoStockException("El id del articulo no puede ser nulo");
        }
        var articulo = _repoArticulos.GetById(idArticulo);
        if(articulo == null)
        {
            throw new MovimientoStockException("No existe un articulo registrado con
ese id");
        }

        if (idTipoMovimiento == null || idTipoMovimiento == 0)
        {
            throw new MovimientoStockException("El id del tipo de movimiento no puede
ser nulo");
        }
        var tipoMovimiento = _repoTipos.GetById(idTipoMovimiento);
        if (tipoMovimiento == null)
        {
            throw new MovimientoStockException("No existe un tipo de movimiento
registrado con ese id");
        }
        var movimientos =
_repoMovimientos.GetMovimientosStockByArticuloAndTipo(idArticulo, idTipoMovimiento,
numPagina, cantidadRegistros);
        if(movimientos == null || movimientos.Count() == 0)
        {
            throw new MovimientoStockException("No existen movimientos de stock para
ese articulo con ese tipo de movimiento");
        }
        return MapperMovimientoStock.FromListConArticuloYTipo(movimientos);
    }
}

using Gestor.LogicaAplicacion.DTO.MovimientoStock;
using Gestor.LogicaAplicacion.InterfacesCU.MovimientoStock;
using Gestor.LogicaAplicacion.Mappers;
using Gestor.LogicaNegocio.Excepciones;

```

```
using Gestor.LogicaNegocio.InterfacesRepositorios;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Gestor.LogicaAplicacion.ImplementacionCU.MovimientoStock
{
    public class GetMovimientoStock : IGetMovimientoStock
    {
        private IRepositoryMovimientoStock _repoMovimientoStock;
        private IRepositoryParametro _repoParametro;

        public GetMovimientoStock(IRepositoryMovimientoStock repoMovimientoStock,
        IRepositoryParametro repoParametros)
        {
            _repoMovimientoStock = repoMovimientoStock;
            _repoParametro = repoParametros;
        }

        public int ParametroTopePaginas()
        {
            int tope = _repoParametro.GetValorInt("TopePaginas");
            return tope;
        }

        public IEnumerable<DtoMovimientoIndex> GetAll()
        {
            var movimientos = _repoMovimientoStock.GetAll();
            if(movimientos == null || movimientos.Count() == 0)
            {
                throw new MovimientoStockException("No existen movimientos registrados");
            }
            return MapperMovimientoStock.FromListIndex(movimientos);
        }

        public DtoMovimientoStockAdd GetById(int id)
        {
            if (id == null || id == 0)
            {
                throw new MovimientoStockException("El id del movimientos que desea
                buscar no puede ser nulo");
            }
            var movimiento = _repoMovimientoStock.GetById(id);
        }
    }
}
```

```

        if (movimiento == null) throw new MovimientoStockException("No existe un
movimientos con ese id");
        return MapperMovimientoStock.ToDtoAdd(movimiento);
    }

    public IEnumerable<dynamic> GetMovimientosSumarry()
    {
        var movimientos =
_repoMovimientoStock.GetMovimientosCantidadesByTipoAndAno();
        if (!movimientos.Any())
        {
            throw new MovimientoStockException("No existen movimientos registrados");
        }
        return movimientos;
    }
}

```

```

using Gestor.LogicaAplicacion.DTO.TipoMovimiento;
using Gestor.LogicaAplicacion.InterfacesCU.TipoMovimiento;
using Gestor.LogicaAplicacion.Mappers;
using Gestor.LogicaNegocio.Articulos;
using Gestor.LogicaNegocio.InterfacesRepositorios;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

```

```

namespace Gestor.LogicaAplicacion.ImplementacionCU.TipoMovimiento
{
    public class AddTipoMovimiento : IAddTipoMovimiento
    {
        private IRepositoryTipoMovimiento _repoTipoMovimiento;

        public AddTipoMovimiento(IRepositoryTipoMovimiento repo)
        {
            _repoTipoMovimiento = repo;
        }

        public int Ejecutar(DtoTipoMovimientoAdd dto)
        {
            if(dto == null)
            {
                throw new Exception("El tipo de movimiento no puede ser nulo");
            }
        }
    }
}

```

```
        Gestor.LogicaNegocio.Entidades.TipoMovimiento tMovimiento =  
MapperTipoMovimiento.FromDtoAdd(dto);  
        _repoTipoMovimiento.Add(tMovimiento);  
        return tMovimiento.Id;  
    }  
}
```

```
using Gestor.LogicaAplicacion.InterfacesCU.TipoMovimiento;  
using Gestor.LogicaNegocio.Excepciones;  
using Gestor.LogicaNegocio.InterfacesRepositorios;  
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
using System.Threading.Tasks;
```

```
namespace Gestor.LogicaAplicacion.ImplementacionCU.TipoMovimiento  
{  
    public class DeleteTipoMovimiento : IDeleteTipoMovimiento  
    {  
        private IRepositoryioTipoMovimiento _repoTipoMovimiento;  
  
        public DeleteTipoMovimiento(IRepositoryioTipoMovimiento repoTipoMovimiento)  
        {  
            _repoTipoMovimiento = repoTipoMovimiento;  
        }  
  
        public void Ejecutar(int id)  
        {  
            if (id == 0 || id == null) throw new TipoMovimientoException("El id no es valido");  
            _repoTipoMovimiento.Remove(id);  
        }  
    }  
}
```

```
using Gestor.LogicaAplicacion.DTO.TipoMovimiento;  
using Gestor.LogicaAplicacion.InterfacesCU.TipoMovimiento;  
using Gestor.LogicaAplicacion.Mappers;  
using Gestor.LogicaNegocio.Excepciones;  
using Gestor.LogicaNegocio.InterfacesRepositorios;  
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
using System.Threading.Tasks;
```

```

namespace Gestor.LogicaAplicacion.ImplementacionCU.TipoMovimiento
{
    public class GetTipoMovimiento : IGetTipoMovimiento
    {
        private IRepositoryTipoMovimiento _repoTipoMovimiento;

        public GetTipoMovimiento(IRepositoryTipoMovimiento repoTipoMovimiento)
        {
            _repoTipoMovimiento = repoTipoMovimiento;
        }

        public IEnumerable<DtoTipoMovimientoCompleto> GetAll()
        {
            var tiposMovimientos = _repoTipoMovimiento.GetAll();
            if(!tiposMovimientos.Any())
            {
                throw new TipoMovimientoException("No hay tipos de movimientos
registrados");
            }
            return MapperTipoMovimiento.FromLista(tiposMovimientos);
        }

        public DtoTipoMovimientoCompleto GetById(int id)
        {
            if(id == 0) { throw new TipoMovimientoException("El id no puede ser nulo"); }
            var tipoMovimiento = _repoTipoMovimiento.GetById(id);
            if(tipoMovimiento == null) { throw new TipoMovimientoException("No existe un tipo
de movimiento registrado con ese id"); }

            return MapperTipoMovimiento.ToDtoCompleto(tipoMovimiento);
        }
    }
}

```

```

using Gestor.LogicaAplicacion.DTO.TipoMovimiento;
using Gestor.LogicaAplicacion.InterfacesCU.TipoMovimiento;
using Gestor.LogicaAplicacion.Mappers;
using Gestor.LogicaNegocio.Excepciones;
using Gestor.LogicaNegocio.InterfacesRepositorios;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

```

```

namespace Gestor.LogicaAplicacion.ImplementacionCU.TipoMovimiento

```



```

{
    public class UpdateTipoMovimiento : IUpdateTipoMovimiento
    {
        private IRepositoryTipoMovimiento _repoTipoMovimiento;

        public UpdateTipoMovimiento(IRepositoryTipoMovimiento repoTipoMovimiento)
        {
            _repoTipoMovimiento = repoTipoMovimiento;
        }

        public void Ejecutar(int id, DtoTipoMovimientoCompleto dto)
        {
            if (id == null)
            {
                throw new TipoMovimientoException("El id del tipo de movimiento no es
valido");
            }
            if(dto == null)
            {
                throw new TipoMovimientoException("Faltan datos para actualizar el tipo de
movimiento");
            }
            Gestor.LogicaNegocio.Entidades.TipoMovimiento tipoMovimiento =
MapperTipoMovimiento.FromDtoCompleto(dto);
            _repoTipoMovimiento.Update(id, tipoMovimiento);
        }
    }
}

```

```

using Gestor.LogicaAplicacion.DTO.Usuario;
using Gestor.LogicaAplicacion.InterfacesCU.Usuario;
using Gestor.LogicaAplicacion.Mappers;
using Gestor.LogicaNegocio.InterfacesRepositorios;
using SistemaAutenticacion.Exceptions.Usuario;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

```

```

namespace Gestor.LogicaAplicacion.ImplementacionCU.Usuario
{
    public class UsuarioLogin : IUsuarioLogin
    {
        private IRepositoryUsuario _repoUsuario;

        public UsuarioLogin(IRepositoryUsuario repoUsuario)
    }
}

```

```
        {
            _repoUsuario = repoUsuario;
        }

        public DtoUsuarioLogin Ejecutar(string email, string password)
        {
            if (string.IsNullOrEmpty(email) || string.IsNullOrEmpty(password)) throw new
            UsuarioException("El email y/o contrasena no pueden ser vacios");
            var usuario = _repoUsuario.GetByUsuarioLogin(email, password);
            if (usuario == null) throw new UsuarioException("El usuario y/o contrasena no es
            valido");
            return MapperUsuario.ToDto(usuario);
        }
    }
}
```

```
using Gestor.LogicaAplicacion.DTO.Articulo;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

```
namespace Gestor.LogicaAplicacion.InterfacesCU.Articulo
{
    public interface IGetArticulosAlfabeticamente
    {
        public IEnumerable<DtoArticuloCompleto> Ejecutar();
    }
}
```

```
using Gestor.LogicaAplicacion.DTO.MovimientoStock;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

```
namespace Gestor.LogicaAplicacion.InterfacesCU.MovimientoStock
{
    public interface IAddMovimientoStock
    {
        public int Ejecutar(DtoMovimientoStockAdd movimiento);
    }
}
```

```
using Gestor.LogicaAplicacion.DTO.Articulo;
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Gestor.LogicaAplicacion.InterfacesCU.MovimientoStock
{
    public interface IGetArticulosByMovimientosPorFecha
    {
        public IEnumerable<DtoArticuloCompleto> Ejecutar(DateTime fechaInicio, DateTime
fechaFin, int numPagina, int cantidadRegistros);
    }
}

using Gestor.LogicaAplicacion.DTO.MovimientoStock;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Gestor.LogicaAplicacion.InterfacesCU.MovimientoStock
{
    public interface IGetMovimientosStockByArticuloAndTipo
    {
        IEnumerable<DtoMovimientoStockConArticuloTipoMovimiento> Ejecutar(int
idArticulo, int idTipoMovimiento, int numeroPagina, int cantidadElementos);
    }
}

using Gestor.LogicaAplicacion.DTO.MovimientoStock;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Gestor.LogicaAplicacion.InterfacesCU.MovimientoStock
{
    public interface IGetMovimientoStock
    {
        public IEnumerable<DtoMovimientoIndex> GetAll();

        public DtoMovimientoStockAdd GetById(int id);

        public IEnumerable<dynamic> GetMovimientosSumarry();
    }
}
```

```
        public int ParametroTopePaginas();
    }
}

using Gestor.LogicaAplicacion.DTO.TipoMovimiento;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Gestor.LogicaAplicacion.InterfacesCU.TipoMovimiento
{
    public interface IAddTipoMovimiento
    {
        int Ejecutar(DtoTipoMovimientoAdd dto);
    }
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Gestor.LogicaAplicacion.InterfacesCU.TipoMovimiento
{
    public interface IDeleteTipoMovimiento
    {
        public void Ejecutar(int id);
    }
}

using Gestor.LogicaAplicacion.DTO.TipoMovimiento;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Gestor.LogicaAplicacion.InterfacesCU.TipoMovimiento
{
    public interface IGetTipoMovimiento
    {
        public IEnumerable<DtoTipoMovimientoCompleto> GetAll();
    }
}
```

```
        public DtoTipoMovimientoCompleto GetById(int id);

    }
}

using Gestor.LogicaAplicacion.DTO.TipoMovimiento;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Gestor.LogicaAplicacion.InterfacesCU.TipoMovimiento
{
    public interface IUpdateTipoMovimiento
    {
        public void Ejecutar(int id, DtoTipoMovimientoCompleto dto);
    }
}

using Gestor.LogicaAplicacion.DTO.Usuario;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Gestor.LogicaAplicacion.InterfacesCU.Usuario
{
    public interface IUsuarioLogin
    {
        public DtoUsuarioLogin Ejecutar(string Email, string password);
    }
}

using Gestor.LogicaAplicacion.DTO.Articulo;
using Gestor.LogicaNegocio.Articulos;
using Gestor.LogicaNegocio.Excepciones;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Gestor.LogicaAplicacion.Mappers
{
```

```
public class MapperArticulo
{
    public static DtoArticuloCompleto ToDtoArticuloCompleto(Articulo articulo)
    {
        if (articulo == null) throw new ArgumentNullException("El articulo que se intenta
mappear es nulo");
        return new DtoArticuloCompleto()
        {
            Id = articulo.Id,
            Nombre = articulo.Nombre,
            Descripcion = articulo.Descripcion,
           Codigo = articulo.Codigo,
            PrecioActual = articulo.PrecioActual,
        };
    }
}

public static IEnumerable<DtoArticuloCompleto> FromLista(IEnumerable<Articulo>
articulos)
{
    if (articulos == null) throw new ArticuloException("No se pueden mapear los
articulos");
    return articulos.Select(a => ToDtoArticuloCompleto(a));
}
}

using Gestor.LogicaAplicacion.DTO.MovimientoStock;
using Gestor.LogicaNegocio.Entidades;
using Gestor.LogicaNegocio.Excepciones;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Gestor.LogicaAplicacion.Mappers
{
    internal static class MapperMovimientoStock
    {
        internal static MovimientoStock FromDtoAdd(DtoMovimientoStockAdd dto)
        {
            if (dto == null) throw new MovimientoStockException("El movimiento de stock que
desea agregar no puede ser nulo");
            LogicaNegocio.Entidades.MovimientoStock movimiento = new
MovimientoStock(dto.ArticuloId, dto.TipoMovimientoId, dto.Cantidad);
            if(movimiento == null) throw new MovimientoStockException("El movimiento de stock
que desea agregar no puede ser nulo");
        }
    }
}
```

```
        return movimiento;
    }

    internal static DtoMovimientoStockAdd ToDtoAdd(MovimientoStock movimiento)
    {
        if (movimiento == null) throw new MovimientoStockException("El movimiento de
stock que desea listar no puede ser nulo");
        var dto = new DtoMovimientoStockAdd()
        {
            Cantidad = movimiento.Cantidad,
            ArticuloId = movimiento.ArticuloId,
            TipoMovimientoId = movimiento.TipoMovimientoId
        };
        return dto;
    }

    internal static IEnumerable<DtoMovimientoStockAdd>
    FromList(IEnumerable<MovimientoStock> movimientos)
    {
        if (movimientos == null || movimientos.Count() == 0) throw new
MovimientoStockException("No existen registros para ser mapeados de movimientos
stock");
        return movimientos.Select(m => ToDtoAdd(m));
    }

    internal static DtoMovimientoIndex ToDtoIndex(MovimientoStock movimiento)
    {
        if (movimiento == null) throw new MovimientoStockException("El movimiento de
stock que desea listar no puede ser nulo");
        var dto = new DtoMovimientoIndex()
        {
            Cantidad = movimiento.Cantidad,
            ArticuloId = movimiento.ArticuloId,
            TipoMovimientoId = movimiento.TipoMovimientoId,
            NombreArticulo = movimiento.Articulo.Nombre,
            NombreTipoMovimiento = movimiento.TipoMovimiento.Nombre
        };
        return dto;
    }

    internal static IEnumerable<DtoMovimientoIndex>
    FromListIndex(IEnumerable<MovimientoStock> movimientos)
    {
        if (movimientos == null || movimientos.Count() == 0) throw new
MovimientoStockException("No existen registros para ser mapeados de movimientos
stock");
```

```

        return movimientos.Select(m => ToDtoIndex(m));
    }

    //Para DtoMovimientoStockConArticuloTipoMovimiento
    internal static DtoMovimientoStockConArticuloTipoMovimiento
    ToDtoMovimientoStockConArticuloTipoMovimiento(MovimientoStock movimiento)
    {
        if (movimiento == null) throw new MovimientoStockException("El movimiento de
stock que desea listar no puede ser nulo");
        var dto = new DtoMovimientoStockConArticuloTipoMovimiento()
        {
            Id = movimiento.Id,
            FechaMovimiento = movimiento.FechaMovimiento,
            Cantidad = movimiento.Cantidad,
            IdArticulo = movimiento.Articulo.Id,
            NombreArticulo = movimiento.Articulo.Nombre,
            PrecioActualArticulo = movimiento.Articulo.PrecioActual,
            IdTipoMovimiento = movimiento.TipoMovimiento.Id,
            NombreTipoMovimiento = movimiento.TipoMovimiento.Nombre,
            CoeficienteTipoMovimiento = movimiento.TipoMovimiento.Coeficiente
        };
        return dto;
    }

    internal static IEnumerable<DtoMovimientoStockConArticuloTipoMovimiento>
    FromListConArticuloYTipo(IEnumerable<MovimientoStock> movimientos)
    {
        if (movimientos == null || movimientos.Count() == 0) throw new
MovimientoStockException("No existen registros para ser mapeados de movimientos
stock");
        return movimientos.Select(m =>
ToDtoMovimientoStockConArticuloTipoMovimiento(m));
    }
}

```

```

using Gestor.LogicaAplicacion.DTO.TipoMovimiento;
using Gestor.LogicaNegocio.Articulos;
using Gestor.LogicaNegocio.Entidades;
using Gestor.LogicaNegocio.Excepciones;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

```

```

namespace Gestor.LogicaAplicacion.Mappers
{

```



```

        internal static class MapperTipoMovimiento
        {
            //ADD
            internal static TipoMovimiento FromDtoAdd(DtoTipoMovimientoAdd dto)
            {
                if (dto == null) throw new TipoMovimientoException("El dto a mapper no puede ser
nulo");
                TipoMovimiento tipoMovimiento = new TipoMovimiento(dto.Nombre, dto.Coeficiente);
                return tipoMovimiento;
            }

            //GET
            internal static DtoTipoMovimientoCompleto ToDtoCompleto(TipoMovimiento
tipoMovimiento)
            {
                if (tipoMovimiento == null) throw new TipoMovimientoException("El tipo de
movimiento que se desea mapear no puede ser nulo");
                var dtoTipoMovimiento = new DtoTipoMovimientoCompleto() {
                    Id = tipoMovimiento.Id,
                    Nombre = tipoMovimiento.Nombre,
                    Coeficiente = tipoMovimiento.Coeficiente
                };
                return dtoTipoMovimiento;
            }

            internal static TipoMovimiento FromDtoCompleto(DtoTipoMovimientoCompleto dto)
            {
                if (dto == null) throw new TipoMovimientoException("El tipo de movimiento que se
desea mapear no puede ser nulo");
                TipoMovimiento tipoMovimiento = new TipoMovimiento(dto.Nombre, dto.Coeficiente);
                return tipoMovimiento;
            }

            internal static IEnumerable<DtoTipoMovimientoCompleto> FromLista
(IEnumerable<TipoMovimiento> tipoMovimientos)
            {
                if (tipoMovimientos == null) throw new TipoMovimientoException("No se pueden
mapear los tipo de movimientos");
                return tipoMovimientos.Select(t => ToDtoCompleto(t));
            }
        }
    }

using Gestor.LogicaAplicacion.DTO.Usuario;
using SistemaAutenticacion.Entidades;
using SistemaAutenticacion.Exceptions.Usuario;
using System;

```

```
using System.Collections.Generic;
using System.Linq;
using System.Runtime.InteropServices;
using System.Text;
using System.Threading.Tasks;

namespace Gestor.LogicaAplicacion.Mappers
{
    internal static class MapperUsuario
    {
        internal static DtoUsuarioLogin ToDto(Usuario user)
        {
            if (user == null) throw new UsuarioException("Error el usuario no puede ser nulo");
            return new DtoUsuarioLogin()
            {
                Email = user.Email.Valor,
                Role = user.Rol()
            };
        }
    }
}
```

```
using Gestor.LogicaNegocio.InterfacesEntidades;
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations.Schema;
using Microsoft.EntityFrameworkCore;
using System.Linq;
using System.Security.Principal;
using System.Text;
using System.Threading.Tasks;
using Gestor.LogicaNegocio.Excepciones;
using Gestor.LogicaNegocio.Entidades;
```

```
namespace Gestor.LogicaNegocio.Articulos
{
    [Index(nameof(Nombre), IsUnique = true)]
    [Index(nameof(Codigo), IsUnique = true)]
    public class Artículo : IEntity, IValidate
    {
        #region Properties
        public int Id { get; set; }
        public string Nombre { get; set; }
        public string Descripcion { get; set; }

        public string Codigo { get; set; }
    }
}
```

```

[Column(TypeName = "decimal(10,2)")]
public decimal PrecioActual { get; set; }

public List<MovimientoStock> HistorialMovimientos { get; set; }

public Artículo(string nombre, string descripcion, string codigo, decimal precioActual)
{
    Nombre = nombre;
    Descripcion = descripcion;
    Codigo = codigo;
    PrecioActual = precioActual;
    Validate();
}

protected Artículo() { }
#endregion
#region Methods
public void Modificar(Artículo articuloModificado)
{
    //No hago validacion de articulo porque ese se realiza en el constructor
    Nombre = articuloModificado.Nombre;
    Descripcion = articuloModificado.Descripcion;
    Codigo = articuloModificado.Codigo;
    PrecioActual = articuloModificado.PrecioActual;
    Validate();
}
public void Validate()
{
    if (string.IsNullOrEmpty(Nombre))
    {
        throw new ArtículoException("El nombre no puede ser nulo");
    }
    if (string.IsNullOrEmpty(Descripcion))
    {
        throw new ArtículoException("La descripcion no puede ser menor a 5
caracteres");
    }
    if (Nombre.Length < 10 || Nombre.Length > 200)
    {
        throw new ArtículoException("El nombre debe tener entre 10 y 200
caracteres");
    }
    if (Descripcion.Length < 5)
    {
        throw new ArgumentException("La descripcion debe tener al menos 5
caracteres");
    }
}

```

```

        if (Codigo == null || Codigo.Length != 13)
        {
            throw new ArgumentException("El codigo debe tener 13 digitos");
        }
        if (PrecioActual <= 0)
        {
            throw new ArgumentException("El precio actual no es valido");
        }
    }
}

public override string ToString()
{
    return $"Id: {Id}\nNombre: {Nombre}\nDescripcion: {Descripcion}\nCodigo: {Codigo}\nPrecio Actual: {PrecioActual}";
}
#endregion
}
}

```

```

using Gestor.LogicaNegocio.Articulos;
using Gestor.LogicaNegocio.Excepciones;
using Gestor.LogicaNegocio.InterfacesEntidades;
using SistemaAutenticacion.InterfacesEntidades;
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using IEntity = Gestor.LogicaNegocio.InterfacesEntidades.IEntity;
using IValidate = Gestor.LogicaNegocio.InterfacesEntidades.IValidate;

```

```

namespace Gestor.LogicaNegocio.Entidades
{
    public class MovimientoStock : IEntity, IValidate
    {
        public int Id { get; set; }

        public Artículo Artículo { get; set; }

        [ForeignKey(nameof(Artículo))]
        public int ArtículoId { get; init; }

        public TipoMovimiento TipoMovimiento { get; set; }

        [ForeignKey(nameof(TipoMovimiento))]
    }
}

```

```
public int TipoMovimientold { get; init; }

public int Cantidad { get; init; }

[DataType(DataType.Date)]
[Column(TypeName = "Date")]
public DateTime FechaMovimiento { get; init; }

public static int s_TopeCantidad {get; set;}

protected MovimientoStock() { }

public MovimientoStock(int articuloId, int tipoMovimientold, int cantidad)
{
    FechaMovimiento = DateTime.Now;
    ArticuloId = articuloId;
    TipoMovimientold = tipoMovimientold;
    Cantidad = cantidad;
    Validate();
}

public void Validate()
{
    if (Articulo == null && ArticuloId == 0)
    {
        throw new MovimientoStockException("El cliente no puede ser nulo");
    }

    if (FechaMovimiento < DateTime.MinValue)
    {
        throw new MovimientoStockException("La fecha de entrega no es valida");
    }
    if(Cantidad <= 0)
    {
        throw new MovimientoStockException("La cantidad debe ser superior a 0");
    }
}

public void VerificarTope()
{
    if(Cantidad > s_TopeCantidad)
    {
        throw new MovimientoStockException("La cantidad de unidades del
movimiento de stock no puede superar el tope (1500)");
    }
}
}
```

```
}

using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Gestor.LogicaNegocio.Entidades
{
    public class Parametro
    {
        [Key]
        public string Nombre { get; set; }
        public string Valor { get; set; }
        public Parametro(string nombre, string valor)
        {
            if (string.IsNullOrEmpty(nombre) || string.IsNullOrEmpty(valor))
                throw new ArgumentException("Parámetros vacíos");
            Nombre = nombre;
            Valor = valor;
        }
    }
}
```

```
using Gestor.LogicaNegocio.Excepciones;
using Gestor.LogicaNegocio.InterfacesEntidades;
using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Gestor.LogicaNegocio.Entidades
{
    [Index(nameof(Nombre), IsUnique = true)]
    public class TipoMovimiento : IEntity, IValidate
    {
        public int Id { get; set; }

        public string Nombre { get; set; }

        //Solo puede ser 1 o -1
        public int Coeficiente { get; set; }
    }
}
```

```

protected TipoMovimiento() { }

public TipoMovimiento(string nombre, int coeficiente)
{
    Nombre = nombre;
    Coeficiente = coeficiente;
    Validate();
}
public void Validate()
{
    if (String.IsNullOrEmpty(Nombre))
    {
        throw new TipoMovimientoException("El nombre no puede ser nulo");
    }
    if(Coeficiente != 1 && Coeficiente != -1)
    {
        throw new TipoMovimientoException("Invalido tipo de movimiento el
movimiento puede ser agregando o quitando stock unicamente");
    }
}

public void Modificar (TipoMovimiento tipoMovimientoModificado)
{
    Nombre = tipoMovimientoModificado.Nombre;
    Coeficiente = tipoMovimientoModificado.Coficiente;
}
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Runtime.Serialization;
using System.Text;
using System.Threading.Tasks;

namespace Gestor.LogicaNegocio.Excepciones
{
    public class ArtículoException : Exception
    {
        public ArtículoException()
        {
        }

        public ArtículoException(string? message) : base(message)
        {
        }
    }
}

```

```

        public ArtículoException(string? message, Exception? innerException) :
base(message, innerException)
        {
        }

        protected ArtículoException(SerializationInfo info, StreamingContext context) :
base(info, context)
        {
        }
    }

using System;
using System.Collections.Generic;
using System.Linq;
using System.Runtime.Serialization;
using System.Text;
using System.Threading.Tasks;

namespace Gestor.LogicaNegocio.Excepciones
{
    public class MovimientoStockException : Exception
    {
        public MovimientoStockException()
        {
        }

        public MovimientoStockException(string? message) : base(message)
        {
        }

        public MovimientoStockException(string? message, Exception? innerException) :
base(message, innerException)
        {
        }

        protected MovimientoStockException(SerializationInfo info, StreamingContext
context) : base(info, context)
        {
        }
    }
}

using System;
using System.Collections.Generic;
using System.Linq;

```



```
using System.Runtime.Serialization;
using System.Text;
using System.Threading.Tasks;

namespace Gestor.LogicaNegocio.Excepciones
{
    public class TipoMovimientoException : Exception
    {
        public TipoMovimientoException()
        {
        }

        public TipoMovimientoException(string? message) : base(message)
        {
        }

        public TipoMovimientoException(string? message, Exception? innerException) :
        base(message, innerException)
        {
        }

        protected TipoMovimientoException(SerializationInfo info, StreamingContext context)
        : base(info, context)
        {
        }
    }
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Gestor.LogicaNegocio.InterfacesEntidades
{
    public interface IEntity
    {
        {
        public int Id { get; set; }
        }
    }
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
```

```
using System.Threading.Tasks;

namespace Gestor.LogicaNegocio.InterfacesEntidades
{
    public interface IValidate
    {
        void Validate();
    }
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Gestor.LogicaNegocio.InterfacesRepositorios
{
    public interface IRepository<T> where T : class
    {
        T GetById(int id);
        void Add(T obj);
        void Update(int id, T obj);
        void Remove(int id);
        void Remove(T obj);
        IEnumerable<T> GetAll();
    }
}

using Gestor.LogicaNegocio.Articulos;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Gestor.LogicaNegocio.InterfacesRepositorios
{
    public interface IRepositoryArticulo : IRepository<Articulo>
    {
        IEnumerable<Articulo> GetArticulosOrdenadosAlfabeticamente();
    }
}

using Gestor.LogicaNegocio.Articulos;
using Gestor.LogicaNegocio.Entidades;
using System;
```

```
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Gestor.LogicaNegocio.InterfacesRepositorios
{
    public interface IRepositoryMovimientoStock
    {
        MovimientoStock GetById(int id);
        void Add(MovimientoStock movimientoStock);
        IEnumerable<MovimientoStock> GetAll();

        IEnumerable<MovimientoStock> GetMovimientosStockByArticuloAndTipo(int
idArticulo, int idTipoMovimiento, int numPagina, int cantidadRegistros);

        IEnumerable<Articulo> GetArticulosConMovimientosRangoFechas(DateTime
fechalInicio, DateTime fechaFin, int numPagina, int cantidadRegistros);
        IEnumerable<dynamic> GetMovimientosCantidadesByTipoAndAno();
    }
}

using Gestor.LogicaNegocio.Entidades;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Reflection.Metadata;
using System.Text;
using System.Threading.Tasks;

namespace Gestor.LogicaNegocio.InterfacesRepositorios
{
    public interface IRepositoryParametro
    {
        public string GetValorString(string nombre);
        public int GetValorInt(string nombre);
        public DateTime GetValorDateTime(string nombre);
        public Parametro GetByName(string nombre);

        public decimal GetValorDecimal(string nombre);
        public void Update(String nombreParametro, string valor);
    }
}

using Gestor.LogicaNegocio.Entidades;
using System;
using System.Collections.Generic;
```

```
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Gestor.LogicaNegocio.InterfacesRepositorios
{
    public interface IRepositoryTipoMovimiento : IRepository<TipoMovimiento>
    {
    }
}

using SistemaAutenticacion.Entidades;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Gestor.LogicaNegocio.InterfacesRepositorios
{
    public interface IRepositoryUsuario : IRepository<Usuario>
    {
        public Usuario GetByUsuarioLogin(string email, string password);
    }
}

using Gestor.LogicaAplicacion.DTO.Articulo;
using Gestor.LogicaAplicacion.InterfacesCU.Articulo;
using Gestor.LogicaNegocio.Excepciones;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;

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

namespace Gestor.WebAPI.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class ArticuloController : ControllerBase
    {
        private IGetArticulosAlfabeticamente _getArticulos;
        public ArticuloController(IGetArticulosAlfabeticamente getArticulosAlfabeticamente)
        {
            _getArticulos = getArticulosAlfabeticamente;
        }
    }
}
```

```

[Authorize(Roles = "Encargado")]
[HttpGet("")]
public ActionResult<IEnumerable<DtoArticuloCompleto>> Get()
{
    try
    {
        var articulos = _getArticulos.Ejecutar();
        if (!articulos.Any()) { return NotFound(); }
        return Ok(articulos);
    }
    catch (ArticuloException ex)
    {
        return BadRequest(ex.Message);
    }
    catch (Exception ex)
    {
        return StatusCode(500, ex.Message);
    }
}
}

```

```

using Gestor.LogicaAplicacion.DTO.MovimientoStock;
using Gestor.LogicaAplicacion.DTO.TipoMovimiento;
using Gestor.LogicaAplicacion.InterfacesCU.MovimientoStock;
using Gestor.LogicaNegocio.Excepciones;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;

```

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

```

namespace Gestor.WebAPI.Controllers
{
    [ApiController]
    [Route("api/[controller]")]
    [ApiConventionType(typeof(DefaultApiConventions))]
    public class MovimientoStockController : ControllerBase
    {
        private IAddMovimientoStock _addMovimientoStock;
        private IGetMovimientoStock _getMovimientosStock;
        private IGetMovimientosStockByArticuloAndTipo _getMovimientosByArticuloAndTipo;
        private IGetArticulosByMovimientosPorFecha _getArticulosByMovimientosPorFecha;

        public MovimientoStockController(IAddMovimientoStock addMovimientoStock,
            IGetMovimientoStock getMovimientosStock, IGetMovimientosStockByArticuloAndTipo

```

```

getMovimientosByArticuloAndTipo, IGetArticulosByMovimientosPorFecha
getArticulosByMovimientosPorFecha)
{
    _addMovimientoStock = addMovimientoStock;
    _getMovimientosStock = getMovimientosStock;
    _getMovimientosByArticuloAndTipo = getMovimientosByArticuloAndTipo;
    _getArticulosByMovimientosPorFecha = getArticulosByMovimientosPorFecha;
}

// GET: api/<MovimientoStockController>
[HttpGet]
public IActionResult Get()
{
    try
    {
        var dtoMovimientosStock = _getMovimientosStock.GetAll();
        if (!dtoMovimientosStock.Any()) { return NoContent(); }
        return Ok(dtoMovimientosStock);
    }
    catch (MovimientoStockException ex)
    {
        return BadRequest(ex.Message);
    }
    catch (Exception ex)
    {
        return StatusCode(500, ex.Message);
    }
}

// GET api/<MovimientoStockController>/5
[Authorize(Roles = "Encargado")]
[HttpGet("{id}", Name = "GetMovimientoStockById")]
public IActionResult Get(int id)
{
    try
    {
        var dtoMovimientosStock = _getMovimientosStock.GetById(id);
        if (dtoMovimientosStock == null) { return NoContent(); }
        return Ok(dtoMovimientosStock);
    }
    catch (MovimientoStockException ex)
    {
        return BadRequest(ex.Message);
    }
    catch (Exception ex)
    {
        return StatusCode(500, ex.Message);
    }
}

```

```

    }
    }

    // POST api/<MovimientoStockController>
    [Authorize(Roles = "Encargado")]
    [HttpPost("")]
    public IActionResult Post([FromBody] DtoMovimientoStockAdd dto)
    {
        if (dto == null)
        {
            return BadRequest("Debe ingresar un tipo de movimiento");
        }
        try
        {
            int id = _addMovimientoStock.Ejecutar(dto);

            return CreatedAtRoute("GetMovimientoStockById", new { Id = id }, dto);
        }
        catch (MovimientoStockException ex)
        {
            return BadRequest(ex.Message);
        }
        catch (Exception ex)
        {
            return StatusCode(500, ex.Message);
        }
    }

    [Authorize(Roles = "Encargado")]
    [HttpGet("MovimientosFiltradosXArticuloTipo/{idArticulo}/{idTipo}/{numPagina}")]
    public IActionResult GetFiltradosPorArticuloYTipo(int idArticulo, int idTipo, int
numPagina)
    {
        try
        {
            int itemsPagina = _getMovimientosStock.ParametroTopePaginas();
            var dtoMovimientosStock =
_getMovimientosByArticuloAndTipo.Ejecutar(idArticulo, idTipo, numPagina, itemsPagina);
            if (!dtoMovimientosStock.Any()) { return NoContent(); }
            return Ok(dtoMovimientosStock);
        }
        catch (MovimientoStockException ex)
        {
            return BadRequest(ex.Message);
        }
        catch (Exception ex)
        {

```

```

        return StatusCode(500, ex.Message);
    }
}

[Authorize(Roles = "Encargado")]
[HttpPost("ArticulosConMovimientosRangoFechas/")]
public IActionResult GetArticulosConMovimientosPorFechas ([FromBody]
DtoArticulosFiltradosFechas dto)
{
    try
    {
        int itemsPagina = _getMovimientosStock.ParametroTopePaginas();
        var dtoArticulos =
        _getArticulosByMovimientosPorFecha.Ejecutar(dto.FechaInicio, dto.FechaFin, dto.Pagina,
itemsPagina);
        if (dtoArticulos == null || !dtoArticulos.Any()) { return NoContent(); }
        return Ok(dtoArticulos);
    }
    catch (MovimientoStockException ex)
    {
        return BadRequest(ex.Message);
    }
    catch (Exception ex)
    {
        return StatusCode(500, ex.Message);
    }
}

[Authorize(Roles = "Encargado")]
[HttpGet("MovimientosSummary")]
public IActionResult GetMovimientosSummary ()
{
    try
    {
        var dtoMovimientosStock =
        _getMovimientosStock.GetMovimientosSumarry();
        if (dtoMovimientosStock == null) { return NotFound(); }
        return Ok(dtoMovimientosStock);
    }
    catch (MovimientoStockException ex)
    {
        return BadRequest(ex.Message);
    }
    catch (Exception ex)
    {
        return StatusCode(500, ex.Message);
    }
}

```



```

    }
    }
}

using Gestor.LogicaAplicacion.DTO.TipoMovimiento;
using Gestor.LogicaAplicacion.InterfacesCU.TipoMovimiento;
using Gestor.LogicaNegocio.Entidades;
using Gestor.LogicaNegocio.Excepciones;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Components.Forms;
using Microsoft.AspNetCore.Mvc;
using System;

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

namespace Gestor.WebAPI.Controllers
{
    [ApiConventionType(typeof(DefaultApiConventions))]
    [Route("api/[controller]")]
    [ApiController]
    public class TipoMovimientoController : ControllerBase
    {
        private IGetTipoMovimiento _getTipoMovimientos;
        private IAddTipoMovimiento _addTipoMovimientos;
        private IUpdateTipoMovimiento _updateTipoMovimientos;
        private IDeleteTipoMovimiento _deleteTipoMovimientos;

        public TipoMovimientoController(IGetTipoMovimiento getTipoMovimientos,
        IAddTipoMovimiento addTipoMovimientos, IUpdateTipoMovimiento updateTipoMovimientos,
        IDeleteTipoMovimiento deleteTipoMovimientos)
        {
            _getTipoMovimientos = getTipoMovimientos;
            _addTipoMovimientos = addTipoMovimientos;
            _updateTipoMovimientos = updateTipoMovimientos;
            _deleteTipoMovimientos = deleteTipoMovimientos;
        }

        // GET: api/<TipoMovimientoController>
        [HttpGet("")]
        public IActionResult Get()
        {
            try
            {
                var dtoTipoMovimientos = _getTipoMovimientos.GetAll();
                if (!dtoTipoMovimientos.Any()) { return NotFound(); }
            }
        }
    }
}

```

```
        return Ok(dtoTipoMovimientos);
    }
    catch (TipoMovimientoException ex)
    {
        return BadRequest(ex.Message);
    }
    catch (Exception ex)
    {
        return StatusCode(500, ex.Message);
    }
}

// GET api/<TipoMovimientoController>/5
[HttpGet("{id}", Name = "GetTipoMovimientoById")]
public IActionResult Get(int id)
{
    try
    {
        var dtoTipoMovimiento = _getTipoMovimientos.GetById(id);
        if (dtoTipoMovimiento == null) { return NotFound(); }
        return Ok(dtoTipoMovimiento);
    }
    catch (TipoMovimientoException ex)
    {
        return BadRequest(ex.Message);
    }
    catch (Exception ex)
    {
        return StatusCode(500, ex.Message);
    }
}

// POST api/<TipoMovimientoController>
[HttpPost("")]
public IActionResult Post([FromBody] DtoTipoMovimientoAdd dto)
{
    if (dto == null)
    {
        return BadRequest("Debe ingresar un tipo de movimiento");
    }
    try
    {
        int id = _addTipoMovimientos.Ejecutar(dto);

        return CreatedAtRoute("GetTipoMovimientoById", new { Id = id }, dto);
    }
    catch (TipoMovimientoException ex)
    {
    }
}
```

```
{
    return BadRequest(ex.Message);
}
catch (Exception ex)
{
    return StatusCode(500, ex.Message);
}
}

// PUT api/<TipoMovimientoController>/5
[HttpPut("{id}")]
public IActionResult Put(int id, [FromBody] DtoTipoMovimientoCompleto dto)
{
    if (dto == null)
    {
        return BadRequest("Debe indicar el tipo de movimiento con la nueva
informacion");
    }
    if (id == null || id <= 0)
    {
        return BadRequest("Debe indicar un identificador del tipo de movimiento a
eliminar");
    }
    try
    {
        _updateTipoMovimientos.Ejecutar(id, dto);
        return Ok(dto);
    }
    catch (TipoMovimientoException ex)
    {
        return BadRequest(ex.Message);
    }
    catch (Exception ex)
    {
        return StatusCode(500, ex.Message);
    }
}

// DELETE api/<TipoMovimientoController>/5
[HttpDelete("{id}")]
public IActionResult Delete(int id)
{
    try
    {
        if (id == null || id == 0)
        {

```

```

        return BadRequest("Debe proporcionar el identificador del tipo de movimiento
a eliminar");
    }

    _deleteTipoMovimientos.Ejecutar(id);
    return NoContent();
}
catch(TipoMovimientoException ex)
{
    return BadRequest(ex.Message);
}
catch (Exception ex)
{
    return StatusCode(500, ex.Message);
}
}
}
}

```

```

using Gestor.LogicaAplicacion.DTO.MovimientoStock;
using Gestor.LogicaAplicacion.DTO.Usuario;
using Gestor.LogicaAplicacion.InterfacesCU.Usuario;
using Gestor.WebAPI.JWT;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using System.Diagnostics;

```

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

```

namespace Gestor.WebAPI.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class UsuarioController : ControllerBase
    {
        //CORREGIR, TRABAJAR CON CASO DE USO
        private IUserarioLogin _login;

        public UsuarioController(IUsuarioLogin login)
        {
            _login = login;
        }

        [AllowAnonymous]
        [HttpPost("Login")]
        public IActionResult Login([FromBody] DtoUsuarioLogin dto)

```

```

        {
            try
            {
                if (dto == null || String.IsNullOrEmpty(dto.Email) ||
String.IsNullOrEmpty(dto.Password))
                {
                    return BadRequest("Rellene todos los campos");
                }
                var user = _login.Ejecutar(dto.Email, dto.Password);
                if (user == null)
                {
                    return Unauthorized("Usuario y/o password invalidos");
                }
                string token = JwtHandler.GenerarToken(user.Email, user.Role);
                //El objeto anonimo que envio coincide con el Model de HTTPClient
LoginTokenModel
                return Ok(new { Token = token, Rol = user.Role, Email = user.Email });

            }
            catch (Exception ex)
            {
                return Unauthorized(new { Error = "Usuario y/o password invalidos" });
            }
        }
    }

using Microsoft.IdentityModel.Tokens;
using SistemaAutenticacion.ValueObject;
using System.IdentityModel.Tokens.Jwt;
using System.Security.Claims;
using System.Text;

namespace Gestor.WebAPI.JWT
{
    public class JwtHandler
    {
        /// <summary>
        /// Método para generar el token JWT usando una función estática (no es necesario
tener instancias)
        /// </summary>

        /// <remarks> Creación del "payload" con tiene la información del usuario que se
logueó (subject)

```

```

    /// El usuario tiene "claims", que son pares nombre/valor que se utilizan para guardar
    /// en el cliente. No pueden ser sensibles
    /// Se le debe setear el periodo temporal de validez (Expires)
    ///Se utiliza un algoritmo de clave simétrica para generar el token</remarks>

    public static string GenerarToken(string email, string rol)
    {
        var claveEncriptada = new
SymmetricSecurityKey(Encoding.UTF8.GetBytes("Xyw2T+gs5Gm8Wd3U3fpGnqBkmyUbw
Vx8MLL0zj+zD12+4zwW32gNIEIHEwqP2+KrEmWfX5H7tqvG1BdFhCqS0w==\r\n"));
        //Claim : Es una pieza de informacion sobre le token, Informacion que vamos a
agregar al token y que va a viajar en el mismo (Payroll?)
        List<Claim> claims = [
            new Claim(ClaimTypes.Email, email),
            new Claim(ClaimTypes.Role, rol)
        ];

        var credenciales = new SigningCredentials(claveEncriptada,
SecurityAlgorithms.HmacSha512Signature);

        var token = new JwtSecurityToken(claims: claims, expires:
DateTime.Now.AddDays(1), signingCredentials: credenciales);

        var jwt = new JwtSecurityTokenHandler().WriteToken(token);

        return jwt;
    }
}

{
    "Logging": {
        "LogLevel": {
            "Default": "Information",
            "Microsoft.AspNetCore": "Warning"
        }
    },
    "AllowedHosts": "*",
    "ConnectionStrings": {
        "ConexionGestor":
"SERVER=(localDb)\\MsSqlLocalDb;DATABASE=GestorObligatorio;Integrated
Security=true;Encrypt=false"
    }
}

```

```
using Gestor.AccesoDatos.EF;
using Gestor.LogicaAplicacion.ImplementacionCU.Articulo;
using Gestor.LogicaAplicacion.ImplementacionCU.MovimientoStock;
using Gestor.LogicaAplicacion.ImplementacionCU.TipoMovimiento;
using Gestor.LogicaAplicacion.ImplementacionCU.Usuario;
using Gestor.LogicaAplicacion.InterfacesCU.Articulo;
using Gestor.LogicaAplicacion.InterfacesCU.MovimientoStock;
using Gestor.LogicaAplicacion.InterfacesCU.TipoMovimiento;
using Gestor.LogicaAplicacion.InterfacesCU.Usuario;
using Gestor.LogicaNegocio.InterfacesRepositorios;
using Microsoft.AspNetCore.Authentication.JwtBearer;
using Microsoft.EntityFrameworkCore;
using Microsoft.IdentityModel.Tokens;
using System.Text;

namespace Gestor.WebAPI
{
    public class Program
    {
        public static void Main(string[] args)
        {
            var builder = WebApplication.CreateBuilder(args);

            // Add services to the container.

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

            //Indicar a Swagger que use el archivo generado durante la compilación
            //a partir de los comentarios XML
            var rutaArchivo = Path.Combine(AppContext.BaseDirectory, "DocentesApi.xml");
            builder.Services.AddSwaggerGen();

            //builder.Services.AddControllersWithViews();

            builder.Services.AddDbContext<GestorContext>(options =>
options.UseSqlServer(builder.Configuration.GetConnectionString("ConexionGestor")));

            //builder.Services.AddSession();

            builder.Services.AddScoped<IRepositorioArticulo, RepositorioArticuloEF>();
```

```

        builder.Services.AddScoped<IRepositorioUsuario, RepositorioUsuarioEF>();
        builder.Services.AddScoped<IRepositorioMovimientoStock,
RepositorioMovimientoStockEF>();
        builder.Services.AddScoped<IRepositorioTipoMovimiento,
RepositorioTipoMovimientoEF>();
        builder.Services.AddScoped<IRepositorioParametro, RepositorioParametroEF>();

//Caso de uso
//Tipo Movimientos
builder.Services.AddScoped<IGetTipoMovimiento, GetTipoMovimiento>();
builder.Services.AddScoped<IAddTipoMovimiento, AddTipoMovimiento>();
builder.Services.AddScoped<IUpdateTipoMovimiento, UpdateTipoMovimiento>();
builder.Services.AddScoped<IDeleteTipoMovimiento, DeleteTipoMovimiento>();

//Movimientos Stock
builder.Services.AddScoped<IAddMovimientoStock, AddMovimientoStock>();
builder.Services.AddScoped<IGetMovimientoStock, GetMovimientoStock>();
builder.Services.AddScoped<IGetMovimientosStockByArticuloAndTipo,
GetMovimientosStockByArticuloAndTipo>();
        builder.Services.AddScoped<IGetArticulosByMovimientosPorFecha,
GetArticulosByMovimientosPorFecha>();

//Articulos
        builder.Services.AddScoped<IGetArticulosAlfabeticamente,
GetArticulosAlfabeticamente>();

//User
        builder.Services.AddScoped<IUsuarioLogin, UsuarioLogin>();

//inyectar los servicios necesarios para la autenticación
var claveEncriptada = new
SymmetricSecurityKey(Encoding.UTF8.GetBytes("Xyw2T+gs5Gm8Wd3U3fpGnqBkmyUbw
Vx8MLL0zj+zD12+4zwW32gNIEIHEwqP2+KrEmWfX5H7tqvG1BdFhCqS0w==\r\n"));

#region Registro de servicios JWT
builder.Services.AddAuthentication(JwtBearerDefaults.AuthenticationScheme)
.AddJwtBearer(opt =>
{
    opt.TokenValidationParameters = new TokenValidationParameters
    {
        //Definir las verificaciones a realizar
        ValidateIssuer = false,
        ValidateAudience = false,
        ValidateLifetime = false,
        ValidateIssuerSigningKey = true,

        IssuerSigningKey = claveEncriptada
    }
}

```



```

        };

    });

    #endregion

    var app = builder.Build();

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

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

    app.MapControllers();

    app.Run();
}
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace SistemaAutenticacion.Entidades
{
    public class Administrador : Usuario
    {
        public Administrador(string nombre, string apellido, string email, string password) :
        base(nombre, apellido, email, password)
        {
            Validate();
        }

        protected Administrador() {}
        public override string Rol()
        {
            return "Administrador";
        }
    }
}

```

```

    }
    }
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using SistemaAutenticacion.ValueObject;

namespace SistemaAutenticacion.Entidades
{
    public class Encargado : Usuario
    {
        public Encargado(string nombre, string apellido, string email, string password) :
base(nombre, apellido, email, password)
        {
            Validate();
        }

        protected Encargado() { }

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

```

```

using SistemaAutenticacion.InterfacesEntidades;
using SistemaAutenticacion.ValueObject;
using SistemaAutenticacion.Exceptions.Usuario;
using System.Text.RegularExpressions;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.EntityFrameworkCore;
using System.ComponentModel.DataAnnotations;

namespace SistemaAutenticacion.Entidades
{
    [Index(nameof(Email), IsUnique = true)]
    public abstract class Usuario : IEntity, IValidate
    {

```

```

#region Properties
public int Id { get; set; }
public string Nombre { get; set; }

public string Apellido { get; set; }

public Email Email { get; init; }

public Password Password { get; set; }

public Usuario(string nombre, string apellido, string email, string password)
{
    Nombre = nombre;
    Apellido = apellido;
    Email = new Email(email);
    Password = new Password(password);
    Validate();
}
#endregion

protected Usuario() { }
#region Methods
public void Validate()
{
    //El nombre y el apellido solamente pueden contener caracteres alfabéticos, espacio,
    apóstrofe o guión del medio.
    //Los caracteres no alfabéticos no pueden estar ubicados al principio ni al final de la
    cadena
    string patron = @"^[a-zA-Z]+(?:['-][a-zA-Z]+)*$";
    //Chequeo el caso de que sea nulo ya que el Regex chequearia si fuese vacio.
    if (string.IsNullOrEmpty(Nombre))
    {
        throw new UsuarioException("El nombre no puede ser nulo");
    }
    if (string.IsNullOrEmpty(Apellido))
    {
        throw new UsuarioException("El apellido no puede ser nulo");
    }
    if (!Regex.IsMatch(Nombre, patron))
    {
        throw new UsuarioException("El formato del nombre no es valido ");
    }
    if (!Regex.IsMatch(Apellido, patron))
    {
        throw new UsuarioException("El formato del apellido no es valido ");
    }
}

```

```

    }

    public void Modificar(Usuario usuario)
    {
        Nombre = usuario.Nombre;
        Apellido = usuario.Apellido;
        Password = usuario.Password;
    }

    public abstract string Rol();

    #endregion
    }
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Runtime.Serialization;
using System.Text;
using System.Threading.Tasks;

namespace SistemaAutenticacion.Exceptions.Email;

public class EmailException : Exception
{
    public EmailException()
    {
    }

    public EmailException(string? message) : base(message)
    {
    }

    public EmailException(string? message, Exception? innerException) :
base(message, innerException)
    {
    }

    protected EmailException(SerializationInfo info, StreamingContext context) :
base(info, context)
    {
    }
}

using System;

```

```
using System.Collections.Generic;
using System.Linq;
using System.Runtime.Serialization;
using System.Text;
using System.Threading.Tasks;

namespace SistemaAutenticacion.Exceptions.Password
{
    public class PasswordException : Exception
    {
        public PasswordException()
        {
        }

        public PasswordException(string? message) : base(message)
        {
        }

        public PasswordException(string? message, Exception? innerException) :
base(message, innerException)
        {
        }

        protected PasswordException(SerializationInfo info, StreamingContext context) :
base(info, context)
        {
        }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Runtime.Serialization;
using System.Text;
using System.Threading.Tasks;

namespace SistemaAutenticacion.Exceptions.Usuario
{
    public class UsuarioException : Exception
    {
        public UsuarioException()
        {
        }

        public UsuarioException(string? message) : base(message)
        {
        }
    }
}
```

```
    }

    public UsuarioException(string? message, Exception? innerException) :
base(message, innerException)
    {
    }

    protected UsuarioException(SerializationInfo info, StreamingContext context) :
base(info, context)
    {
    }
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace SistemaAutenticacion.InterfacesEntidades
{
    public interface IEntity
    {
        public int Id { get; set; }
    }
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace SistemaAutenticacion.InterfacesEntidades
{
    public interface IValidate
    {
        void Validate();
    }
}

using SistemaAutenticacion.InterfacesEntidades;
using SistemaAutenticacion.Exceptions.Email;
using System;
using System.Collections.Generic;
using System.Linq;
```

```

using System.Text;
using System.Text.RegularExpressions;
using System.Threading.Tasks;
using System.ComponentModel.DataAnnotations.Schema;

namespace SistemaAutenticacion.ValueObject
{
    [ComplexType]
    public record Email : IValidate
    {
        public string Valor { get; init; }

        public Email(string valor)
        {
            Valor = valor;
        }

        protected Email() { }
        public void Validate()
        {
            if (string.IsNullOrEmpty(Valor))
            {
                throw new EmailException("El email no puede ser nulo");
            }
            if (!Formato())
            {
                throw new EmailException("El formato del email no es valido");
            }
        }

        private bool Formato()
        {
            string patron = @"^([\w\.-]+)@([\w\.-]+)(\.[\w]{2,3})+$";
            return Regex.IsMatch(Valor, patron);
        }
    }
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Security.Cryptography;
using SistemaAutenticacion.InterfacesEntidades;

```

```

using SistemaAutenticacion.Exceptions.Password;
using System.Text.RegularExpressions;
using System.ComponentModel.DataAnnotations.Schema;

namespace SistemaAutenticacion.ValueObject
{
    [ComplexType]
    public record Password
    {
        public string Valor { get; init; }

        public string Encriptada { get; init; }

        public Password (string valor)
        {
            Validate(valor);
            Valor = valor;
            Encriptada = Encrypt(valor);
        }

        protected Password() { }
        public static string Encrypt(string password)
        {
            using (SHA256 sha256 = SHA256.Create())
            {
                // Convertir la contraseña en bytes
                byte[] passwordBytes = Encoding.UTF8.GetBytes(password);

                // Calcular el hash de la contraseña
                byte[] hashBytes = sha256.ComputeHash(passwordBytes);

                // Convertir el hash a una cadena hexadecimal
                StringBuilder sb = new StringBuilder();
                for (int i = 0; i < hashBytes.Length; i++)
                {
                    sb.Append(hashBytes[i].ToString("x2"));
                }

                return sb.ToString();
            }
        }

        public void Validate(string password)
        {
            string patron = @"^(?=.*[A-Z])(?=.*[a-z])(?=.*\d)(?=.*[.,;!]).{6,}$";
            if (string.IsNullOrEmpty(password))
            {

```



```
        throw new PasswordException("La contraseña no puede ser nula");
    }
    if(!Regex.IsMatch(password, patron))
    {
        throw new PasswordException("La contraseña debe tener un largo mínimo
de 6 caracteres, al menos una letra mayúscula, una minúscula, un dígito y un carácter de
puntuación: punto, punto y coma, coma, signo de admiración de cierre");
    }
    }
    }
}
```

Codigo Fuente : HTTP Client

```
using HTTPClientMVC.Models.Usuario;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using Newtonsoft.Json.Linq;
using System.Text;
using System.Text.Json;

namespace HTTPClientMVC.Controllers
{
    public class LoginController : Controller
    {
        private readonly HttpClient _httpClient;
        private readonly string _url = "https://localhost:7223/api/";

        //Configuracion para deserializar el json y evitar errores por mayusculas y
        minusculas
        private readonly JsonSerializerOptions _jsonOptions = new JsonSerializerOptions {
        PropertyNameCaseInsensitive = true };

        public LoginController(HttpClient httpClient)
        {
            _httpClient = httpClient;
            _httpClient.BaseAddress = new Uri(_url);
        }

        // GET: LoginController
        public ActionResult Autorizar()
        {
            return View();
        }

        // POST: LoginController/Create
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Autorizar(LoginModel loginModel)
        {
            try
```

```

        {
            var json = JsonSerializer.Serialize(loginModel);
            var body = new StringContent(json, Encoding.UTF8, "application/json");
            var respuesta = _httpClient.PostAsync("Usuario/Login", body).Result;
            if (respuesta.IsSuccessStatusCode)
            {
                var content = respuesta.Content.ReadAsStringAsync().Result;

                var token = JsonSerializer.Deserialize<LoginTokenModel>(content,
_jsonOptions);
                if (token == null)
                {
                    ViewBag.Error = "No se ha podido deserializar el token";
                    return View(loginModel);
                }
                HttpContext.Session.SetString("Token", token.Token);
                HttpContext.Session.SetString("Rol", token.Rol);
                //HttpContext.Session.SetString("Email", token.Email);
                //Agregar el token a las cabeceras de las peticiones, para que el servidor lo
pueda validar
                //No olvidar que el token debe ser enviado en todas las peticiones
                _httpClient.DefaultRequestHeaders.Add("Authorization", $"Bearer
{token.Token}");
                return RedirectToAction("Index", "MovimientoStock");
            }
            else
            {
                SetError(respuesta);
                return View(loginModel);
            }
        }
    }
    catch (Exception ex)
    {
        ViewBag.Error = ex.Message;
        return View(loginModel);
    }
}

public ActionResult Logout()
{
    if (HttpContext.Session.GetString("Token") == null)
    {
        return RedirectToAction("Login");
    }

    HttpContext.Session.Remove("Token");
}

```

```

        HttpContext.Session.Remove("Rol");
        HttpContext.Session.Remove("Email");
        _httpClient.DefaultRequestHeaders.Remove("Authorization");
        return RedirectToAction("Autorizar");
    }
    private void SetError(HttpResponseMessage respuesta)
    {
        var contenidoError = respuesta.Content.ReadAsStringAsync().Result;
        dynamic mensajeJson = JObject.Parse(@"{"Message":"' + contenidoError + '"}");
        ViewBag.Error = $"Hubo un error. {respuesta.ReasonPhrase} " +
mensajeJson.Message;
    }

}

}

using HTTPClientMVC.Models.Articulo;
using HTTPClientMVC.Models.MovimientoStock;
using HTTPClientMVC.Models.TipoMovimiento;
using HTTPClientMVC.Models.Usuario;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Rendering;
using Newtonsoft.Json.Linq;
using System.Text;
using System.Text.Json;

namespace HTTPClientMVC.Controllers
{
    public class MovimientoStockController : Controller
    {
        private readonly HttpClient _httpClient;
        private readonly string _uri = "https://localhost:7223/api/";
        private readonly JsonSerializerOptions _jsonOptions = new JsonSerializerOptions {
PropertyNameCaseInsensitive = true, WriteIndented = true };

        public MovimientoStockController(HttpClient httpClient)
        {
            _httpClient = httpClient;
            _httpClient.BaseAddress = new Uri(_uri);
            // _httpClient.DefaultRequestHeaders.Accept.Clear();
            // _httpClient.DefaultRequestHeaders.Accept.Add(new
System.Net.Http.Headers.MediaTypeWithQualityHeaderValue("application/json"));
        }
        // GET: MovimientoStockController

```

```

public ActionResult Index()
{
    var token = HttpContext.Session.GetString("Token");
    if (token == null)
    {
        return RedirectToAction("Autorizar", "Usuario");
    }
    HttpResponseMessage respuesta =
_httpClient.GetAsync("MovimientoStock").Result;
    if (respuesta.StatusCode == System.Net.HttpStatusCode.OK)
    {
        string json = respuesta.Content.ReadAsStringAsync().Result;
        var lista = JsonSerializer.Deserialize<List<MovimientoStockModel>>(json,
_jsonOptions);
        return View(lista);
    }
    else
    {
        SetError(respuesta);
        return RedirectToAction("Create", "MovimientoStock");
    }
}

//// GET: MovimientoStockController/Details/5
//public ActionResult Details(int id)
//{
//    return View();
//}

// GET: MovimientoStockController/Create
public ActionResult Create()
{
    try
    {
        var token = HttpContext.Session.GetString("Token");
        if ((token == null))
        {
            return RedirectToAction("Autorizar", "Usuario");
        }
        var articulos = GetArticulos();
        if (articulos == null)
        {
            throw new Exception("No existen articulos registrados o no tiene acceso a los
mismos");
        }
        var tipoMovimientos = GetTiposDeMovimientos();
        if(tipoMovimientos == null)

```

```

        {
            throw new Exception("No existen articulos registrados o no tiene acceso a los
mismos");
        }
        var selectListArticulos = new SelectList(articulos, "Id", "Nombre");
        var selectListTipoMovimiento = new SelectList(tipoMovimientos, "Id",
"Nombre");
        var viewModel = new MovimientoStockCreateModel
        {
            ArticulosSelectList = selectListArticulos,
            TipoMovimientoSelectList = selectListTipoMovimiento,
        };
        return View(viewModel);
    }
    catch(Exception ex)
    {
        TempData["Error"] = ex.Message;
        return RedirectToAction("Index", "MovimientoStock");
    }
}

public List<ArticuloModel> GetArticulos()
{
    var token = HttpContext.Session.GetString("Token");
    _httpClient.DefaultRequestHeaders.Authorization = new
System.Net.Http.Headers.AuthenticationHeaderValue($"Bearer", token);
    var response = _httpClient.GetAsync("Articulo").Result;
    if (response.StatusCode == System.Net.HttpStatusCode.OK)
    {
        string json = response.Content.ReadAsStringAsync().Result;
        var lista = JsonSerializer.Deserialize<List<ArticuloModel>>(json,
_jsonOptions);
        return lista;
    }
    return null;
}

public List<TipoMovimientoModel> GetTiposDeMovimientos()
{
    var token = HttpContext.Session.GetString("Token");
    _httpClient.DefaultRequestHeaders.Authorization = new
System.Net.Http.Headers.AuthenticationHeaderValue($"Bearer", token);
    var response = _httpClient.GetAsync("TipoMovimiento").Result;
    if (response.StatusCode == System.Net.HttpStatusCode.OK)
    {
        string json = response.Content.ReadAsStringAsync().Result;

```

```

        var lista = JsonSerializer.Deserialize<List<TipoMovimientoModel>>(json,
_jsonOptions);
        return lista;
    }
    return null;
}

// POST: MovimientoStockController/Create
[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult Create(MovimientoStockCreateModel model)
{
    var token = HttpContext.Session.GetString("Token");
    if ((token == null))
    {
        return RedirectToAction("Autorizar", "Usuario");
    }
    _httpClient.DefaultRequestHeaders.Authorization = new
System.Net.Http.Headers.AuthenticationHeaderValue($"Bearer", token);
    if (model == null)
    {
        ViewBag.Error = "No se puede agregar un movimiento de stock con datos
nulos";
        return View();
    }
    try
    {
        //Enviar el docente a la api
        var movimientoStockAdd = new MovimientoStockAddModel()
        {
            ArticuloId = model.SelectedArticuloId,
            TipoMovimientoId = model.SelectedTipoMovimientoId,
            Cantidad = model.Cantidad,
        };
        var json = JsonSerializer.Serialize(movimientoStockAdd);
        //Configurar el contenido del request
        var content = new StringContent(json, Encoding.UTF8, "application/json");
        //Enviar el request a la api para crear el docente en la base de datos
        var respuesta = _httpClient.PostAsync("MovimientoStock", content).Result;
        if (respuesta.IsSuccessStatusCode)
        {
            return RedirectToAction(nameof(Index));
        }
        else
        {
            SetError(respuesta);
            return RedirectToAction("Create", "MovimientoStock");
        }
    }
}

```

```

    }

}

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

public ActionResult GetFiltradosPorArticuloYTipo()
{
    try
    {
        var token = HttpContext.Session.GetString("Token");
        if ((token == null))
        {
            return RedirectToAction("Autorizar", "Usuario");
        }
        var articulos = GetArticulos();
        if (articulos == null)
        {
            throw new Exception("No existen articulos registrados o no tiene acceso a los
mismos");
        }
        var tipoMovimientos = GetTiposDeMovimientos();
        if (tipoMovimientos == null)
        {
            throw new Exception("No existen articulos registrados o no tiene acceso a los
mismos");
        }
        var selectListArticulos = new SelectList(articulos, "Id", "Nombre");
        var selectListTipoMovimiento = new SelectList(tipoMovimientos, "Id",
"Nombre");
        var viewModel = new MovimientoStockCreateModel
        {
            ArticulosSelectList = selectListArticulos,
            TipoMovimientoSelectList = selectListTipoMovimiento,
        };
        return View(viewModel);
    }
    catch (Exception ex)
    {
        TempData["Error"] = ex.Message;
        return RedirectToAction("GetFiltradosPorArticuloYTipo", "MovimientoStock");
    }
}

```



```

    }
    }

    [HttpPost]
    [ValidateAntiForgeryToken]
    public ActionResult GetFiltradosPorArticuloYTipo(MovimientoStockCreateModel
model)
    {
        var token = HttpContext.Session.GetString("Token");
        if ((token == null))
        {
            return RedirectToAction("Autorizar", "Usuario");
        }
        _httpClient.DefaultRequestHeaders.Authorization = new
System.Net.Http.Headers.AuthenticationHeaderValue($"Bearer", token);
        if (model == null)
        {
            ViewBag.Error = "No se puede agregar un movimiento de stock con datos
nulos";
            return View();
        }
        try
        {
            if (model.Pagina == null)
                model.Pagina = 1;

            //Enviar el docente a la api
            var movimientoStockAdd = new MovimientoStockAddModel()
            {
                ArticuloId = model.SelectedArticuloId,
                TipoMovimientoId = model.SelectedTipoMovimientoId,
            };
            var json = JsonSerializer.Serialize(movimientoStockAdd);
            //Configurar el contenido del request
            var body = new StringContent(json, Encoding.UTF8, "application/json");
            //Enviar el request a la api para crear el docente en la base de datos
            var respuesta =
            _httpClient.GetAsync($"MovimientoStock/MovimientosFiltradosXArticuloTipo/{model.Selecte
dArticuloId}/{model.SelectedTipoMovimientoId}/{model.Pagina}").Result;
            if (respuesta.IsSuccessStatusCode)
            {
                var content = respuesta.Content.ReadAsStringAsync().Result;
                if (String.IsNullOrEmpty(content))
                {
                    ViewBag.Pagina = -1;
                    return View();
                }
            }
        }
    }
}

```

```

        ViewBag.Pagina = model.Pagina++;
        var respuestaDeserealizada =
JsonSerializer.Deserialize<IEnumerable<MovimientoStockConArticuloTipoMovimientoModel
>>(content, _jsonOptions);
        var articulos = GetArticulos();
        var tipoMovimientos = GetTiposDeMovimientos();
        var selectListArticulos = new SelectList(articulos, "Id", "Nombre");
        var selectListTipoMovimiento = new SelectList(tipoMovimientos, "Id",
"Nombre");
        var viewModel = new MovimientoStockCreateModel
        {
            ArticulosSelectList = selectListArticulos,
            TipoMovimientoSelectList = selectListTipoMovimiento,
            ListaResultadosMovimientos = respuestaDeserealizada
        };
        return View(viewModel);
    }
    else
    {
        SetError(respuesta);
        return RedirectToAction("GetFiltradosPorArticuloYTipo", "MovimientoStock");
    }
}

catch (Exception ex)
{
    ViewBag.Error = ex.Message;
    return RedirectToAction("GetFiltradosPorArticuloYTipo", "MovimientoStock");
}

}

//

[HttpGet]
public ActionResult GetArticulosFiltradosPorMovimientosRangoFechas()
{
    return View();
}

[HttpPost]
public ActionResult
GetArticulosFiltradosPorMovimientosRangoFechas(ArticuloFiltradoModel model)
{
    try
    {

```

```
        if (model.Pagina == null)
            model.Pagina = 1;

        if (model.FechaInicio == DateTime.MinValue || model.FechaFin ==
DateTime.MinValue)
        {
            throw new Exception("No se puede filtrar los movimientos, fechas no
falidas");
        }

        var token = HttpContext.Session.GetString("Token");
        if ((token == null))
        {
            return RedirectToAction("Autorizar", "Usuario");
        }
        _httpClient.DefaultRequestHeaders.Authorization = new
System.Net.Http.Headers.AuthenticationHeaderValue($"Bearer", token);

        var fechas = new
        {
            FechaInicio = model.FechaInicio,
            FechaFin = model.FechaFin
        };
        var json = JsonSerializer.Serialize(fechas);
        var body = new StringContent(json, Encoding.UTF8, "application/json");

        var respuesta =
_httpClient.PostAsync("MovimientoStock/ArticulosConMovimientosRangoFechas",
body).Result;

        if (respuesta.IsSuccessStatusCode)
        {
            var content = respuesta.Content.ReadAsStringAsync().Result;
            if (String.IsNullOrEmpty(content))
            {
                ViewBag.Pagina = -1;
                return View();
            }
            ViewBag.Pagina = model.Pagina++;
            var lista =
JsonSerializer.Deserialize<IEnumerable<ArticuloCompletoModel>>(content, _jsonOptions);

            if (lista == null)
            {
                ViewBag.Error = "No se ha podido deserializar el token";
                return View();
            }
        }
    }
```

```

    }
    model.ListaArticulos = lista;

    //HttpContext.Session.SetString("Email", token.Email);
    return View(model);
}
else
{
    SetError(respuesta);
    return View();
}

}

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

[HttpGet]
public ActionResult GetMovimientoSummary()
{
    try
    {
        var token = HttpContext.Session.GetString("Token");
        if ((token == null))
        {
            return RedirectToAction("Autorizar", "Usuario");
        }
        _httpClient.DefaultRequestHeaders.Authorization = new
System.Net.Http.Headers.AuthenticationHeaderValue($"Bearer", token);
        HttpResponseMessage respuesta =
        _httpClient.GetAsync("MovimientoStock/MovimientosSummary").Result;
        if (respuesta.StatusCode == System.Net.HttpStatusCode.OK)
        {
            string json = respuesta.Content.ReadAsStringAsync().Result;
            var lista = JsonSerializer.Deserialize<List<MovimientoStockSummary>>(json,
_jsonOptions);
            return View(lista);
        }
        else
        {
            SetError(respuesta);
            return RedirectToAction("GetMovimientoSummary", "MovimientoStock");
        }
    }
}

```

```

    }
    catch (Exception ex)
    {
        TempData["Error"] = ex.Message;
        return RedirectToAction("Index", "MovimientoStock");
    }
}
private void SetError(HttpResponseMessage respuesta)
{
    var contenidoError = respuesta.Content.ReadAsStringAsync().Result;
    dynamic mensajeJson = JObject.Parse(@"{"Message":"' + contenidoError + "'}");
    TempData["Error"] = $"Hubo un error. {respuesta.ReasonPhrase} " +
mensajeJson.Message;
}
}
}

```

```
using System.ComponentModel.DataAnnotations.Schema;
```

```
namespace HTTPClientMVC.Models.Articulo
{
    public class ArticuloCompletoModel
    {
        public int Id { get; set; }
        public string Nombre { get; set; }
        public string Descripcion { get; set; }
        public stringCodigo { get; set; }

        public decimal PrecioActual { get; set; }
    }
}

```

```
namespace HTTPClientMVC.Models.Articulo
{
    public class ArticuloFiltradoModel
    {
        public IEnumerable<ArticuloCompletoModel> ListaArticulos { get; set; }

        public DateTime FechaInicio { get; set; }

        public DateTime FechaFin { get; set; }

        public int Pagina { get; set; }
    }
}

```

```
namespace HTTPClientMVC.Models.Articulo
{
    public class ArticuloModel
    {
        public int Id { get; set; }

        public string Nombre { get; set; }
    }
}
```

```
namespace HTTPClientMVC.Models.MovimientoStock
{
    public class MovimientoStockAddModel
    {
        public int ArticuloId { get; init; }

        public int TipoMovimientoId { get; init; }

        public int Cantidad { get; init; }
    }
}
```

```
using System.ComponentModel.DataAnnotations.Schema;
using System.ComponentModel.DataAnnotations;
```

```
namespace HTTPClientMVC.Models.MovimientoStock
{
    public class MovimientoStockConArticuloTipoMovimientoModel
    {
        public int Id { get; set; }

        public int Cantidad { get; set; }

        [DataType(DataType.Date)]
        [Column(TableName = "Date")]
        public DateTime FechaMovimiento { get; set; }

        //Articulo
        public int IdArticulo { get; set; }
        public string NombreArticulo { get; set; }

        [Column(TableName = "decimal(10,2)")]
        public decimal PrecioActualArticulo { get; set; }

        //TipoMovimiento
    }
}
```

```

        public int IdTipoMovimiento { get; set; }
        public string NombreTipoMovimiento { get; set; }

        public int CoeficienteTipoMovimiento { get; set; }
    }
}

using Microsoft.AspNetCore.Mvc.Rendering;
using System.ComponentModel.DataAnnotations;

namespace HTTPClientMVC.Models.MovimientoStock
{
    public class MovimientoStockCreateModel
    {
        [Required(ErrorMessage = "El campo Artículo es requerido.")]
        public int SelectedArticulold { get; set; }

        [Required(ErrorMessage = "El campo Tipo de Movimiento es requerido.")]
        public int SelectedTipoMovimientold { get; set; }

        [Required(ErrorMessage = "El campo Cantidad es requerido.")]
        public int Cantidad { get; set; }
        public SelectList ArticulosSelectList { get; set; }

        public SelectList TipoMovimientoSelectList { get; set; }

        public IEnumerable<MovimientoStockConArticuloTipoMovimientoModel>
        ListaResultadosMovimientos { get; set; }

        public int Pagina { get; set; }
    }
}

using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using System.ComponentModel.DataAnnotations;

namespace HTTPClientMVC.Models.MovimientoStock
{
    public class MovimientoStockModel
    {
        public string NombreArticulo { get; set; }

        [Required]
        public int Articulold { get; init; }
    }
}

```

```
        public string NombreTipoMovimiento { get; set; }

        [Required]
        public int TipoMovimientold { get; init; }

        [Required]
        public int Cantidad { get; init; }

    }
}
```

```
namespace HTTPClientMVC.Models.MovimientoStock
{
    public class MovimientoStockSummary
    {
        public int Ano { get; set; }

        public string TipoMovimiento { get; set; }

        public int CantidadTotal { get; set; }
    }
}
```

```
namespace HTTPClientMVC.Models.TipoMovimiento
{
    public class TipoMovimientoModel
    {
        public int Id { get; set; }

        public string Nombre { get; set; }

    }
}
```

```
namespace HTTPClientMVC.Models.Usuario
{
    public class LoginModel
    {
        public string Email { get; set; }

        public string Password { get; set; }
    }
}
```

```
namespace HTTPClientMVC.Models.Usuario
{
```



```
public class LoginTokenModel
{
    public string Email { get; set; }

    public string Rol { get; set; }

    public string Token { get; set; }
}
}
```