Universidad ORT Uruguay
Facultad de Ingeniería

OBLIGATORIO
PROGRAMACIÓN 2

Pablo De Melo - 330897 N2B

Fabian Fernandez - 210990 N2B

Tutor/es: Santiago Baillo/Alberto Zonca

2024

Índice

iagrama de clases UML dominio

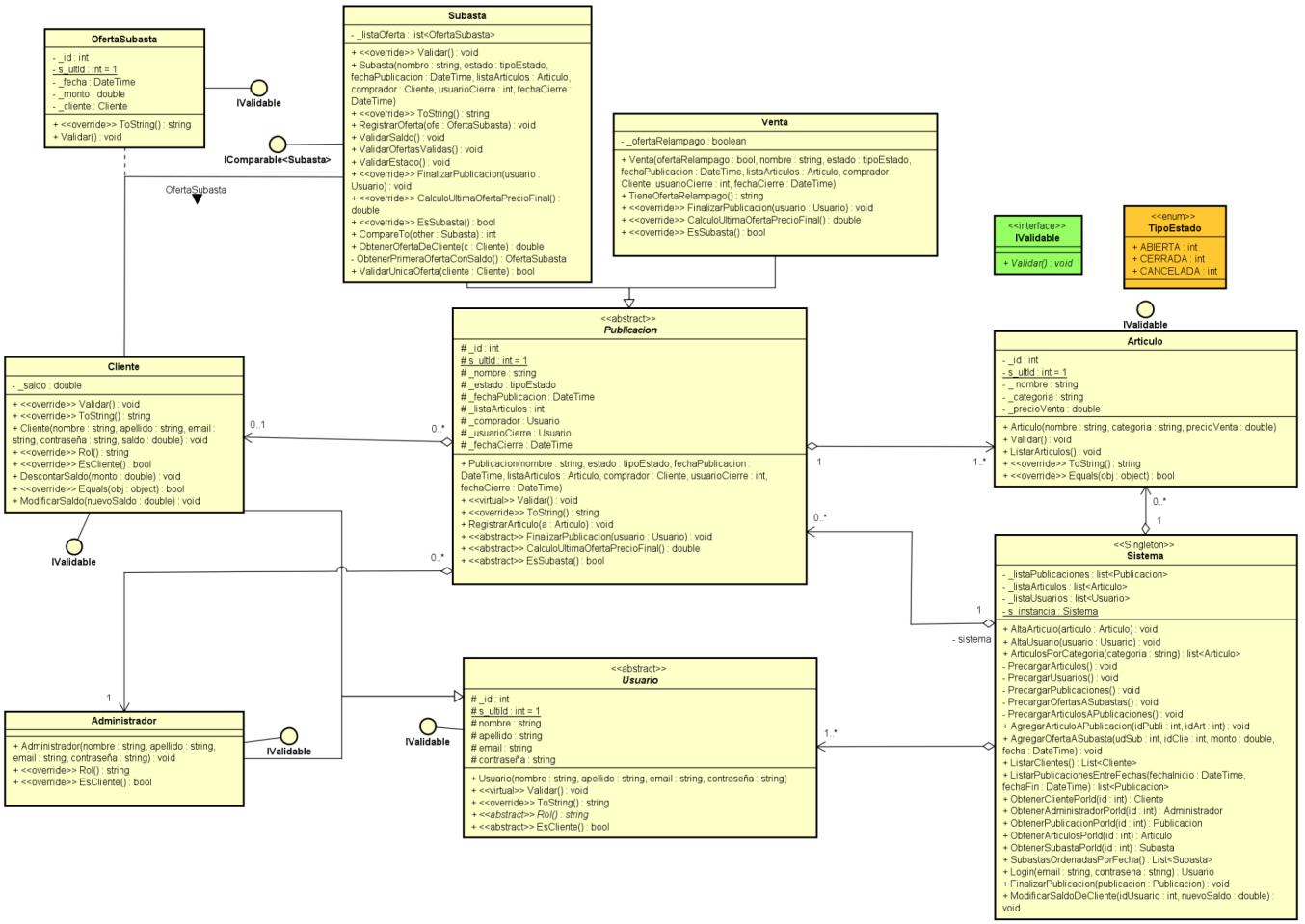
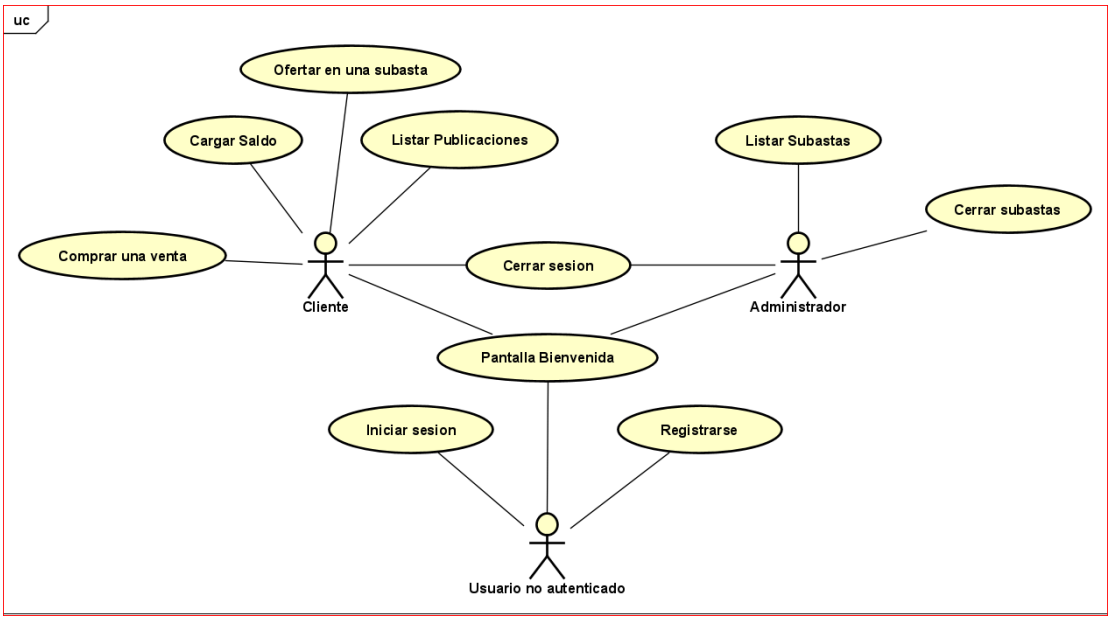


Diagrama de casos de uso



Planilla de casos de prueba

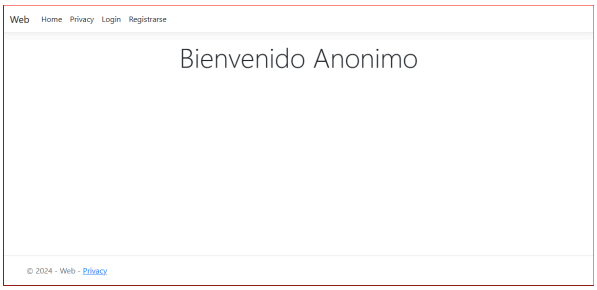
id	Caso de Prueba	Descripción	fecha	Tipo de usuario	Resultado Esperado	Estado
CP01	Pantalla Bienvenida	Para todos los tipos de usuarios debe ser visible una pantalla de bienvenida.	26/11/2024	Anónimo, Cliente, Administrador	Pantalla de bienvenida visible.	OK
CP02	Registrarse	Los usuarios anónimos deben tener la posibilidad de registrarse como cliente.	26/11/2024	Anónimo	Registrarse exitosamente.	OK
CP03	Iniciar Sesión	Los usuarios registrados deben tener la posibilidad de iniciar sesión.	26/11/2024	Anónimo	Login exitoso.	OK
CP04	Cerrar Sesión	Los usuarios logueados deben poder cerrar sesión.	26/11/2024	Cliente, Administrador	Logout exitoso.	OK
CP05	Listar Publicaciones	Los clientes pueden ver la lista de publicaciones.	26/11/2024	Cliente	Lista completa.	OK
CP06	Comprar Una Venta	Los clientes pueden comprar una venta de la lista de publicaciones si	26/11/2024	Cliente	Compra exitosa.	OK

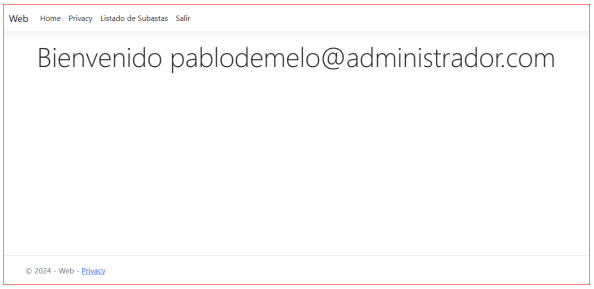
		tiene el saldo necesario.				
CP07	Ofertar una subasta	Los clientes pueden ofertar una subasta de la lista de publicaciones si no hizo una oferta anteriormente en la misma..	26/11/2024	Cliente	Oferta Exitosa.	OK
CP08	Cargar saldo	El cliente puede cargar saldo a su cuenta.	26/11/2024	Cliente	Saldo cargado exitosamente.	OK
CP09	Listar Subastas	El administrador puede ver el listado de subastas.	26/11/2024	Administrador	Lista completa ordenada por fecha.	OK
CP10	Cerrar Subasta.	El administrador puede cerrar una subasta con una oferta válida del listado de subastas.	26/11/2024	Administrador	Subasta cerrada exitosamente.	OK
CP11	Registrarse	Los usuarios anónimos al intentar registrarse como cliente, en el caso de que dejen un campo sin completar, se desplegará un mensaje de advertencia	26/11/2024	Anónimo	El campo contraseña no puede ser vacío. El campo email no puede ser vacío. El campo nombre no puede ser vacío. El campo apellido no puede ser vacío.	OK
CP12	Registrarse	Los usuarios anónimos al intentar ingresar un formato invalido de email al registrarse como clientes, se despliega un mensaje de advertencia.	26/11/2024	Anónimo	El formato del campo email no es válido.	OK
CP13	Registrarse	Los usuarios anónimos al intentar ingresar una contraseña menor a 8 caracteres, se despliega un mensaje de advertencia.	26/11/2024	Anónimo	La contraseña debe tener al menos 8 caracteres.	OK
C14	Registrarse	Los usuarios anonimos al intentar ingresar un correo que ya se encuentra registrado, se despliega un mensaje de advertencia	26/11/2024	Anonimo	El email ya está registrado. Por favor, use uno diferente.	OK
C15	Login	Los usuarios anonimos que ya se registraron o los usuarios clientes y/o administradores que intentan iniciar sesion en la pantalla de login y no completar ningún	26/11/2024	Anonimo, Cliente, Administrador	Debe ingresar un email. Debe ingresar una contraseña. Email o contraseña incorrectas.	OK

		campo o no completar uno de los campos o ingresar los datos incorrectamente, se despliega un mensaje de advertencia.				
C16	Listado de Publicaciones	Los clientes al realizar una compra sin saldo, se despliega un mensaje de error.	26/11/2024	Cliente	Saldo insuficiente para completar la compra.	OK
C17	Listado de Publicaciones	Los clientes al realizar una segunda oferta en la misma subasta, se despliega un mensaje de error.	26/11/2024	Cliente	El cliente ya realizó una oferta en esta publicación.	OK
C18	Modificar Saldo	Los clientes al realizar modificaciones en su saldo, no pueden dejar el campo vacio o ingresar un monto igual a 0 o igual a un número negativo	26/11/2024	Cliente	El monto no puede ser vacío o 0. El monto no puede ser negativo.	OK
C19	Listado de Subastas	Los administradores al cerrar una subasta sin ofertas, se cancela automáticamente. También al cerrar una subasta en la que el cliente no posea saldo suficiente, automáticamente su estado cambia a CANCELAR	26/11/2024	Administrador	No hay ofertas válidas CANCELADA con éxito!	OK
C20	Servidor MICROSOFT AZURE	Al detener el servidor de Microsoft Azure, se detiene el funcionamiento de la aplicación al refrescar la página.	26/11/2024		Error 403 - This web app is stopped.	OK
C21	Listado de Subastas	Cuando un administrador intenta cerrar una subasta cuyo monto supera un umbral definido por el sistema, esta se cancela automáticamente, mostrando el mensaje de error correspondiente.	26/11/2024	Administrador	No hay ofertas válidas CANCELADA con éxito!	OK

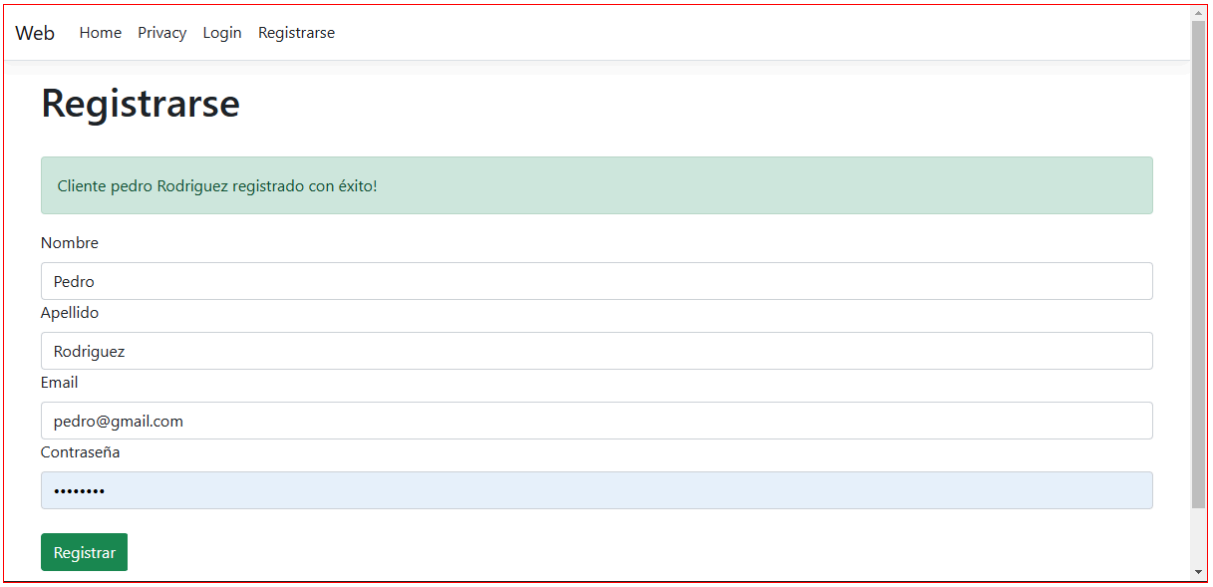
Evidencias casos de prueba

CP01:

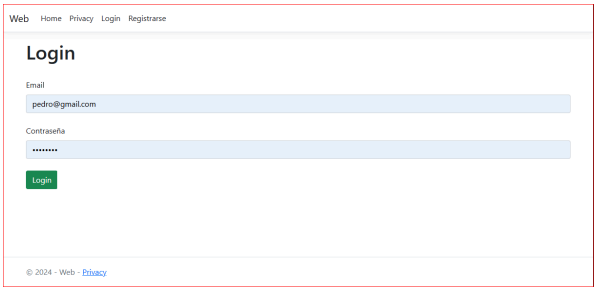




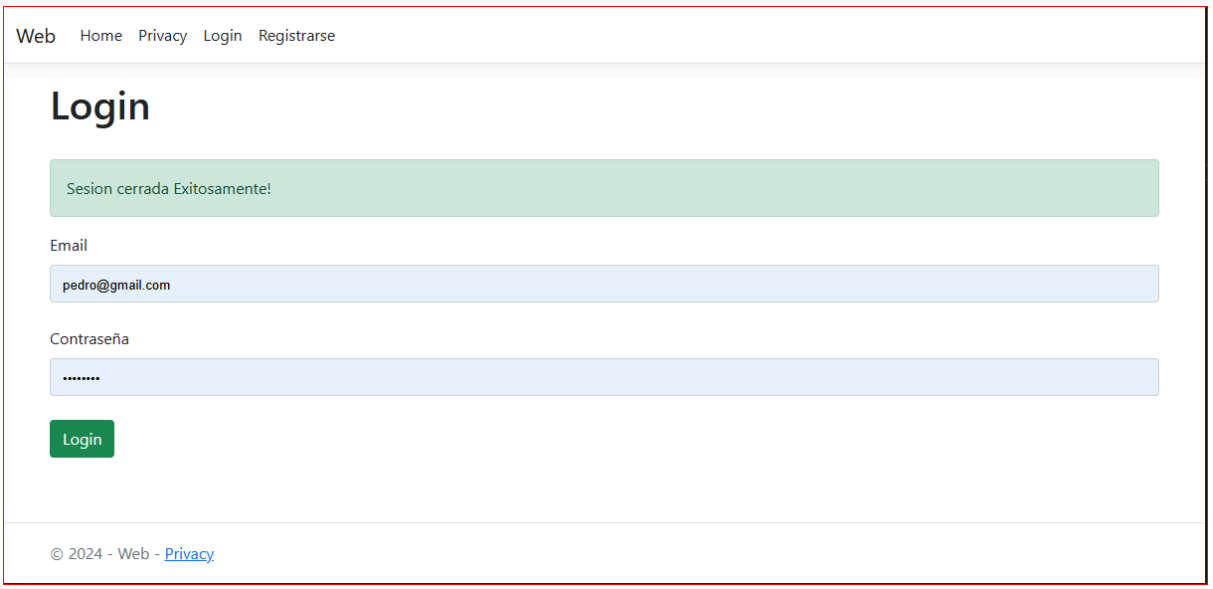
CP02:



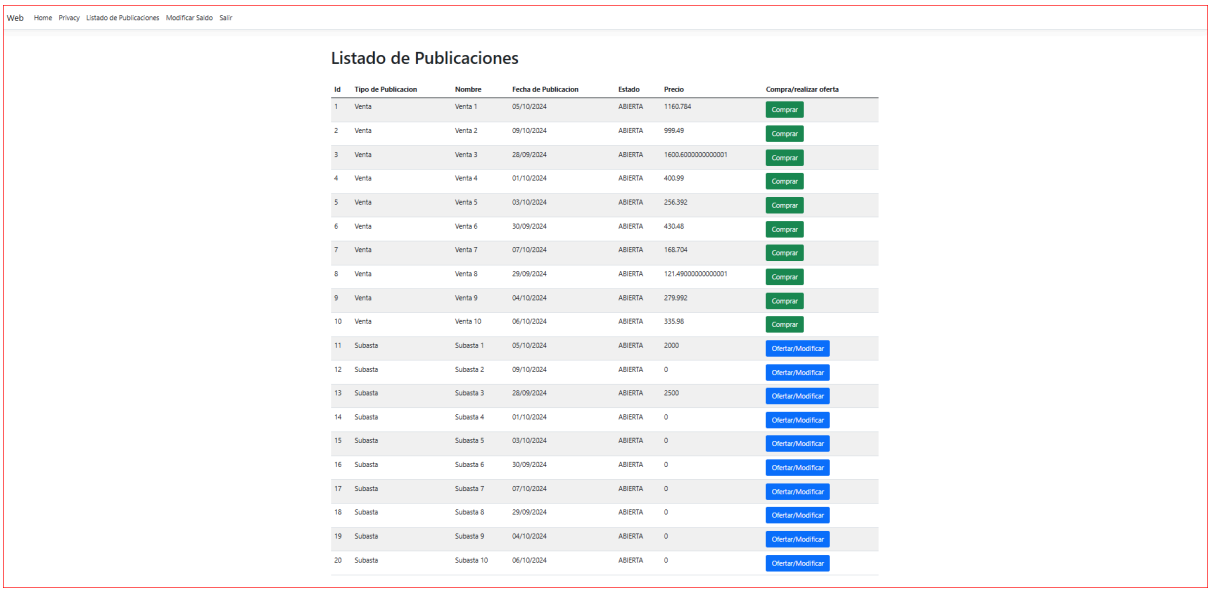
CP03:



CP04:


Web
Home
Privacy
Login
Registrarse
Login
Sesion cerrada Exitosamente!
Email
pedro@gmail.com
Contraseña
Login
© 2024 - Web - Privacy

CP05:


Web
Home
Privacy
Listado de Publicaciones
Modificar Saldo
Salir
Listado de Publicaciones
Id
Tipo de Publicacion
Nombre
Fecha de Publicacion
Estado
Precio
Compra/realizar oferta
1 Venta Venta 1 05/10/2024 ABIERTA 1160.784 Comprar
2 Venta Venta 2 09/10/2024 ABIERTA 999.49 Comprar
3 Venta Venta 3 28/09/2024 ABIERTA 1600.6000000000001 Comprar
4 Venta Venta 4 01/10/2024 ABIERTA 400.99 Comprar
5 Venta Venta 5 03/10/2024 ABIERTA 256.392 Comprar
6 Venta Venta 6 30/09/2024 ABIERTA 430.48 Comprar
7 Venta Venta 7 07/10/2024 ABIERTA 168.704 Comprar
8 Venta Venta 8 29/09/2024 ABIERTA 121.49000000000001 Comprar
9 Venta Venta 9 04/10/2024 ABIERTA 279.992 Comprar
10 Venta Venta 10 06/10/2024 ABIERTA 335.98 Comprar
11 Subasta Subasta 1 05/10/2024 ABIERTA 2000 Ofertar/Modificar
12 Subasta Subasta 2 09/10/2024 ABIERTA 0 Ofertar/Modificar
13 Subasta Subasta 3 28/09/2024 ABIERTA 2500 Ofertar/Modificar
14 Subasta Subasta 4 01/10/2024 ABIERTA 0 Ofertar/Modificar
15 Subasta Subasta 5 03/10/2024 ABIERTA 0 Ofertar/Modificar
16 Subasta Subasta 6 30/09/2024 ABIERTA 0 Ofertar/Modificar
17 Subasta Subasta 7 07/10/2024 ABIERTA 0 Ofertar/Modificar
18 Subasta Subasta 8 29/09/2024 ABIERTA 0 Ofertar/Modificar
19 Subasta Subasta 9 04/10/2024 ABIERTA 0 Ofertar/Modificar
20 Subasta Subasta 10 06/10/2024 ABIERTA 0 Ofertar/Modificar

CP06:


Web Home Privacy Listado de Publicaciones Modificar Saldo Salir
Publicacion comprada con exito!
Listado de Publicaciones
Id Tipo de Publicacion Nombre Fecha de Publicacion Estado Precio Compra/realizar oferta
1 Venta Venta 1 05/10/2024 CERRADA 1160.784 COMPRADO
2 Venta Venta 2 09/10/2024 ABIERTA 999.49 Comprar
3 Venta Venta 3 28/09/2024 ABIERTA 1600.6000000000001 Comprar
4 Venta Venta 4 01/10/2024 ABIERTA 400.99 Comprar

CP07:

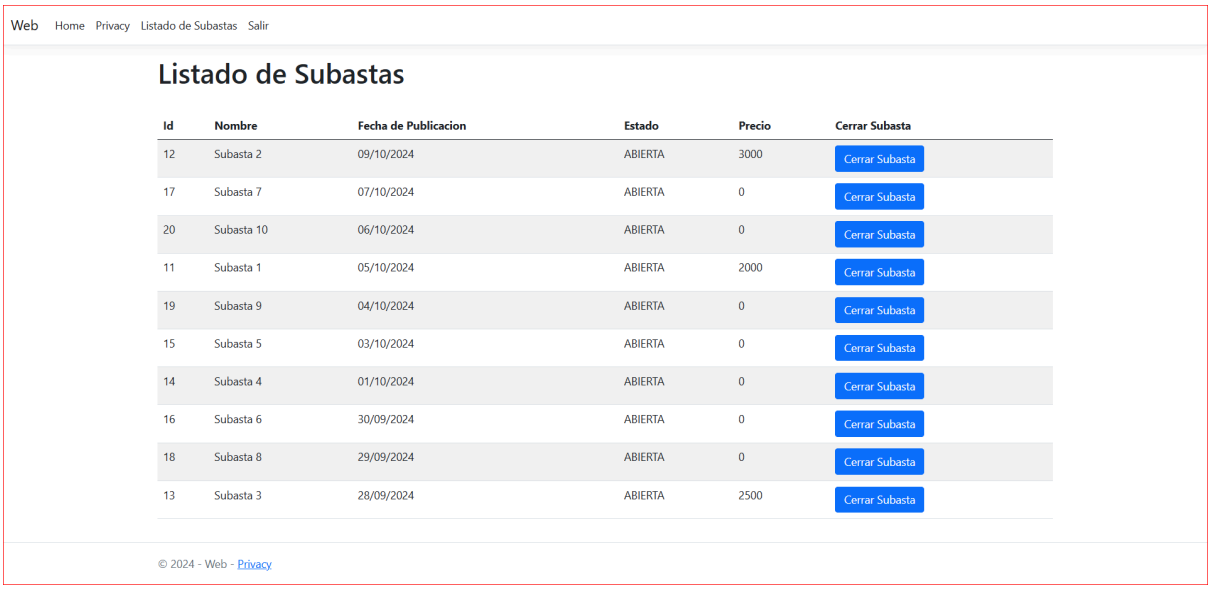

Web Home Privacy Listado de Publicaciones Modificar Saldo Salir
Cambiar Oferta
Oferta 3000
Ofertar
© 2024 - Web - Privacy


Web Home Privacy Listado de Publicaciones Modificar Saldo Salir
Oferta realizada con exito!
Listado de Publicaciones
Id Tipo de Publicacion Nombre Fecha de Publicacion Estado Precio Compra/realizar oferta
1 Venta Venta 1 05/10/2024 CERRADA 1160.784 COMPRADO
2 Venta Venta 2 09/10/2024 ABIERTA 999.49 Comprar
3 Venta Venta 3 28/09/2024 ABIERTA 1600.6000000000001 Comprar
4 Venta Venta 4 01/10/2024 ABIERTA 400.99 Comprar

CP08:


Web Home Privacy Listado de Publicaciones Modificar Saldo Salir
Modificar Saldo
Se modificó el saldo del cliente Cristian - Nuevo saldo: $2009.216
Cargar Saldo
2000
Modificar Saldo
© 2024 - Web - Privacy

CP09:

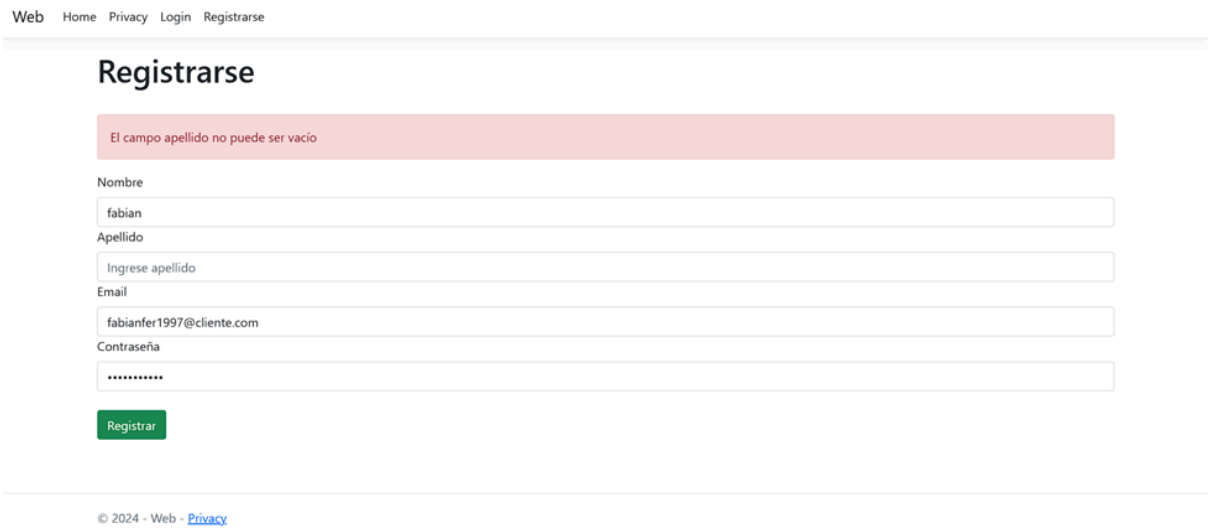


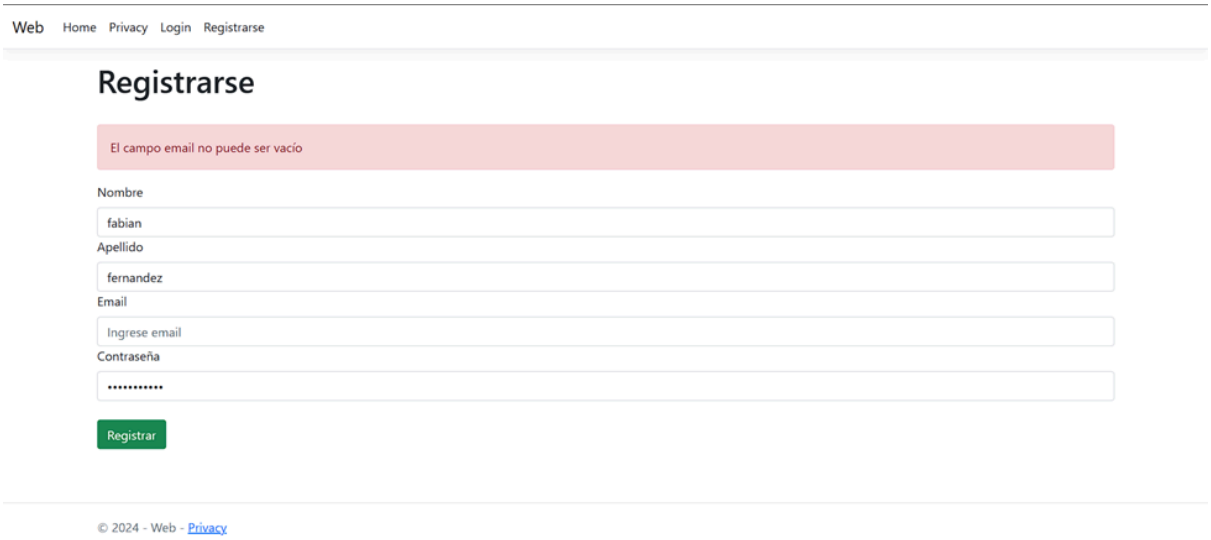
CP10:

CP11:


Web Home Privacy Login Registrarse
Registrarse
El campo nombre no puede ser vacío
Nombre
Ingrese nombre
Apellido
fernandez
Email
fabianfer1997@cliente.com
Contraseña
Registrar
© 2024 - Web - Privacy


Web Home Privacy Login Registrarse
Registrarse
El campo contraseña no puede ser vacío
Nombre
fabian
Apellido
fernandez
Email
fabianfer1997@cliente.com
Contraseña
Ingrese contraseña
Registrar
© 2024 - Web - Privacy


Web Home Privacy Login Registrarse
Registrarse
El campo apellido no puede ser vacío
Nombre
fabian
Apellido
Ingrese apellido
Email
fabianfer1997@cliente.com
Contraseña
Registrar
© 2024 - Web - Privacy


Web Home Privacy Login Registrarse
Registrarse
El campo email no puede ser vacío
Nombre
fabian
Apellido
fernandez
Email
Ingrese email
Contraseña
Registrar
© 2024 - Web - Privacy

CP12:


Web Home Privacy Login Registrarse
Registrarse
El formato del campo email no es válido
Nombre
fabian
Apellido
fernandez
Email
fabianfer1997@gmail
Contraseña
Registrar
© 2024 - Web - Privacy


Web Home Privacy Login Registrarse
Registrarse
El formato del campo email no es válido
Nombre
fabian
Apellido
fernandez
Email
fabianfer1997cliente.com
Contraseña
Incluye un signo "@" en la dirección de correo electrónico. La dirección "fabianfer1997cliente.com" no incluye el signo "@".
Registrar
© 2024 - Web - Privacy

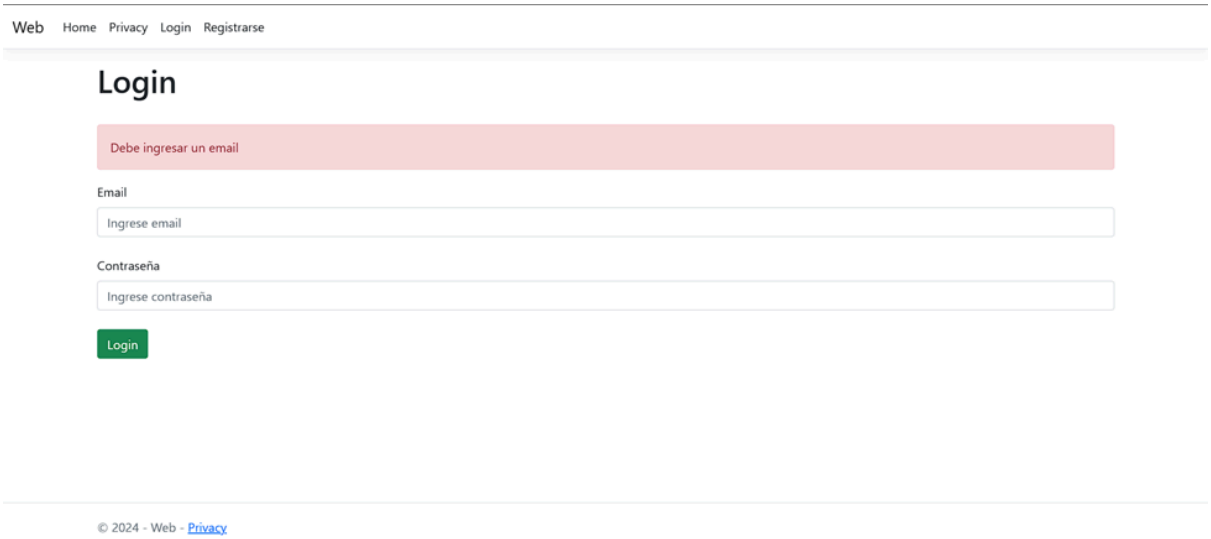
CP13:

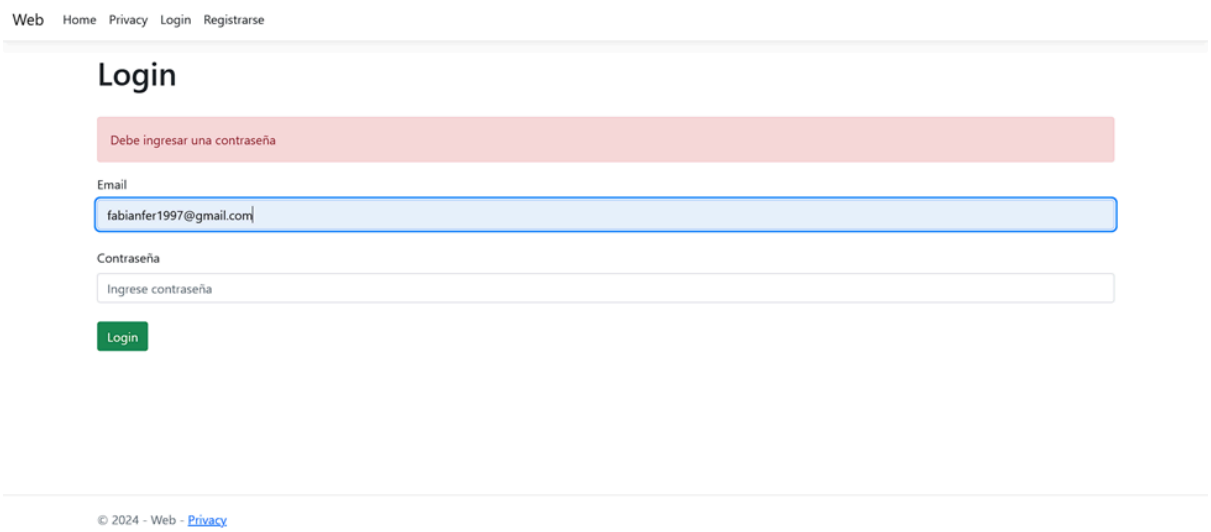

Web Home Privacy Login Registrarse
Registrarse
La contraseña debe tener al menos 8 caracteres
Nombre
fabian
Apellido
fernandez
Email
fabianfer1997@cliente.com
Contraseña
•••••••
Registrar
© 2024 - Web - Privacy

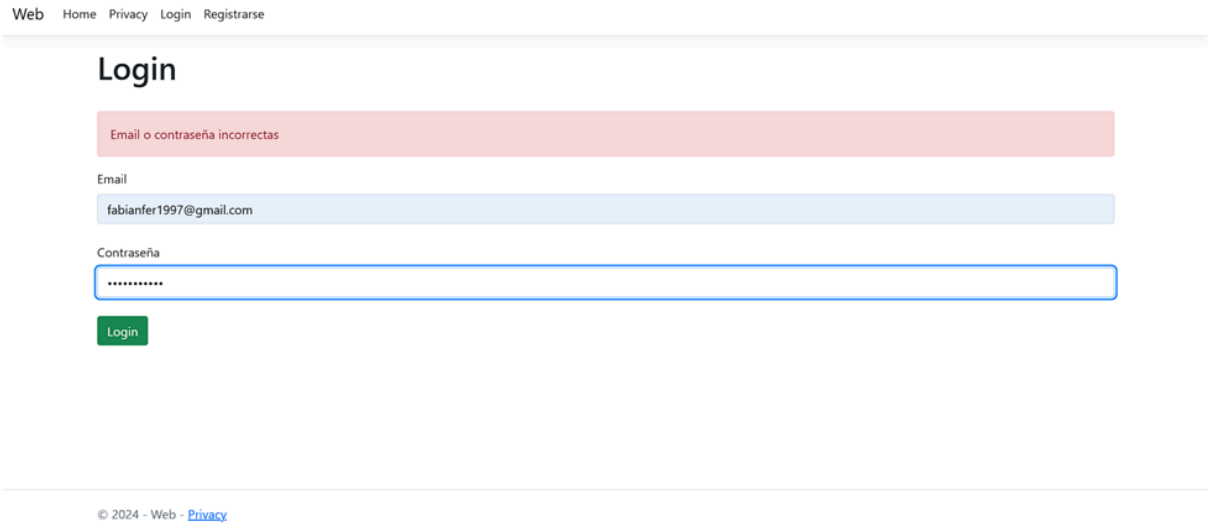
CP14:


Web Home Privacy Login Registrarse
Registrarse
El email ya está registrado. Por favor, use uno diferente.
Nombre
Fabian
Apellido
Fernandez
Email
fabianfer1997@cliente.com
Contraseña
••••••••••
Registrar
© 2024 - Web - Privacy

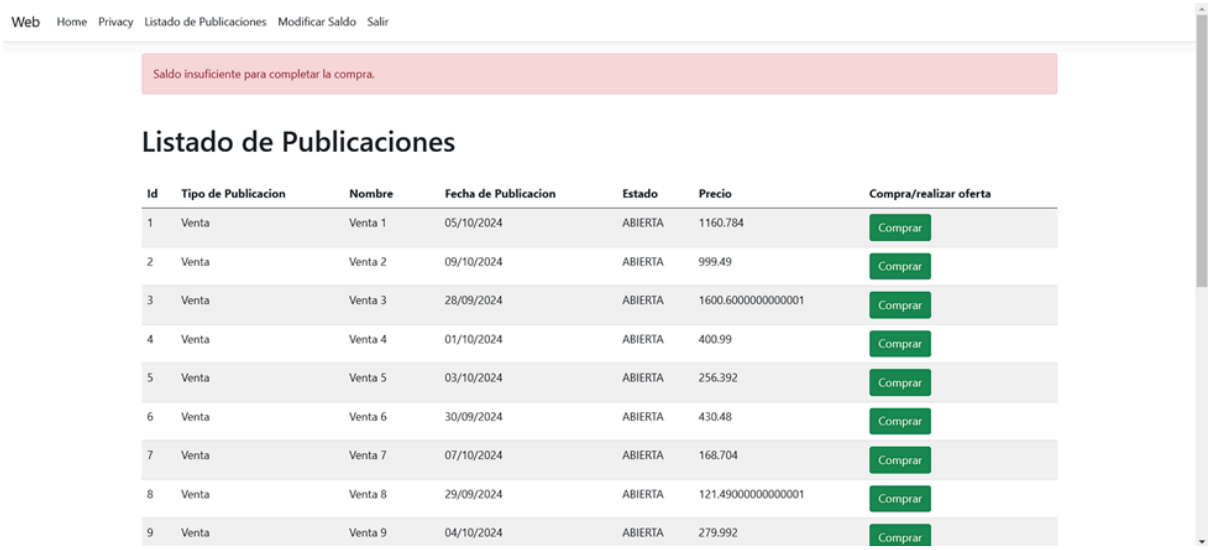
C15:


Web Home Privacy Login Registrarse
Login
Debe ingresar un email
Email
Ingrese email
Contraseña
Ingrese contraseña
Login
© 2024 - Web - Privacy

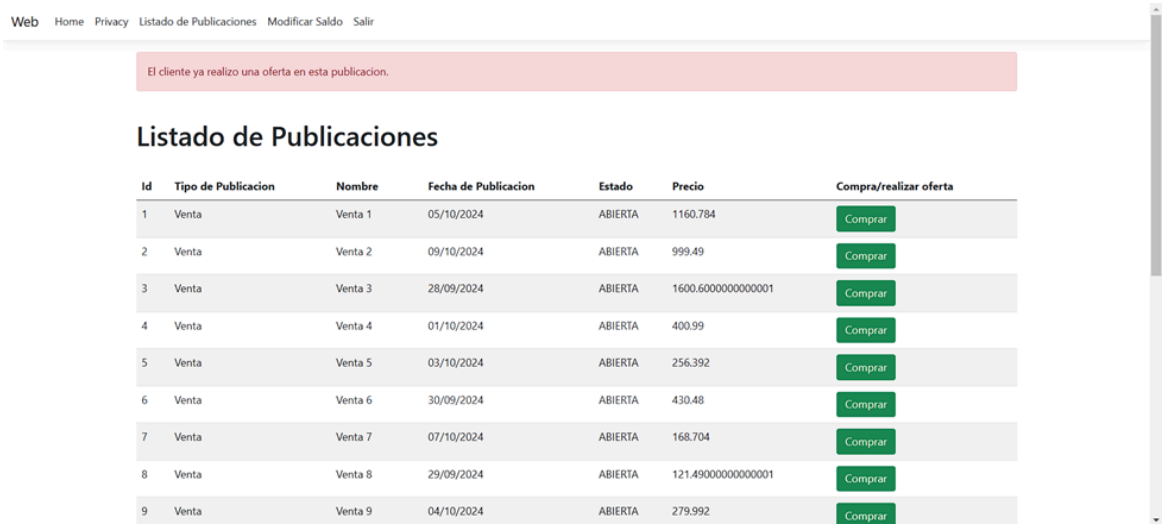

Web Home Privacy Login Registrarse
Login
Debe ingresar una contraseña
Email
fabianfer1997@gmail.com
Contraseña
Ingrese contraseña
Login
© 2024 - Web - Privacy


Web Home Privacy Login Registrarse
Login
Email o contraseña incorrectas
Email
fabianfer1997@gmail.com
Contraseña
Login
© 2024 - Web - Privacy

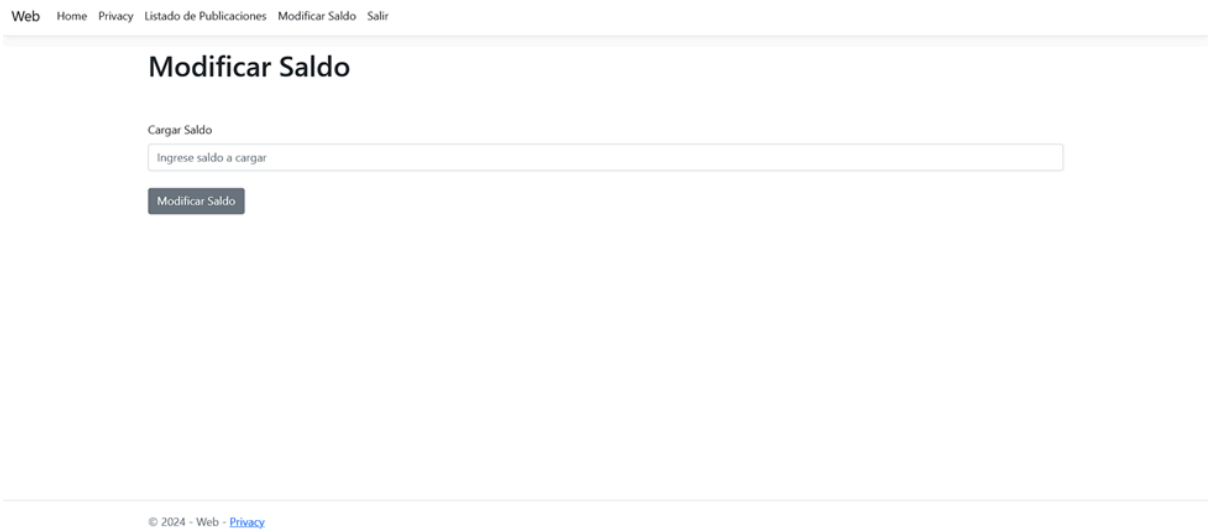
C16:

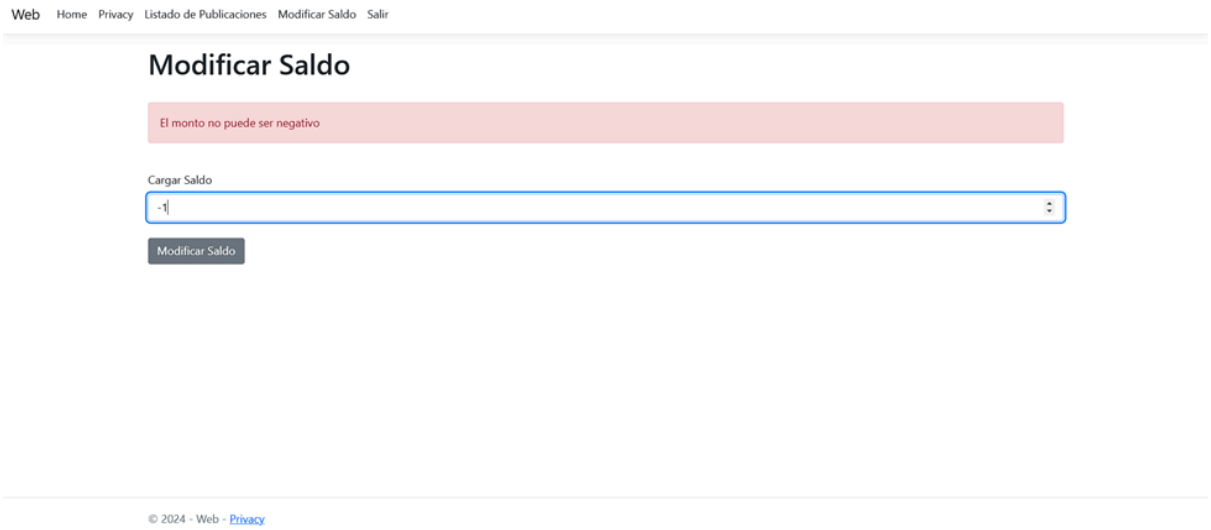

Web Home Privacy Listado de Publicaciones Modificar Saldo Salir
Saldo insuficiente para completar la compra.
Listado de Publicaciones
Id Tipo de Publicacion Nombre Fecha de Publicacion Estado Precio Compra/realizar oferta
1 Venta Venta 1 05/10/2024 ABIERTA 1160.784 Comprar
2 Venta Venta 2 09/10/2024 ABIERTA 999.49 Comprar
3 Venta Venta 3 28/09/2024 ABIERTA 1600.6000000000001 Comprar
4 Venta Venta 4 01/10/2024 ABIERTA 400.99 Comprar
5 Venta Venta 5 03/10/2024 ABIERTA 256.392 Comprar
6 Venta Venta 6 30/09/2024 ABIERTA 430.48 Comprar
7 Venta Venta 7 07/10/2024 ABIERTA 168.704 Comprar
8 Venta Venta 8 29/09/2024 ABIERTA 121.49000000000001 Comprar
9 Venta Venta 9 04/10/2024 ABIERTA 279.992 Comprar

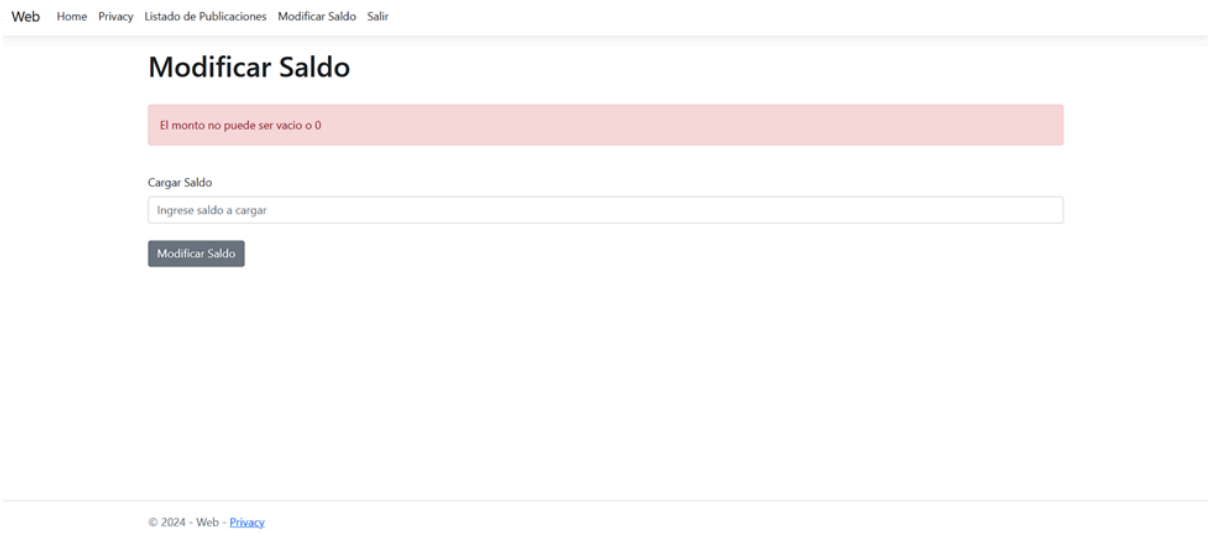
C17:


Web Home Privacy Listado de Publicaciones Modificar Saldo Salir
El cliente ya realizo una oferta en esta publicacion.
Listado de Publicaciones
Id Tipo de Publicacion Nombre Fecha de Publicacion Estado Precio Compra/realizar oferta
1 Venta Venta 1 05/10/2024 ABIERTA 1160.784 Comprar
2 Venta Venta 2 09/10/2024 ABIERTA 999.49 Comprar
3 Venta Venta 3 28/09/2024 ABIERTA 1600.6000000000001 Comprar
4 Venta Venta 4 01/10/2024 ABIERTA 400.99 Comprar
5 Venta Venta 5 03/10/2024 ABIERTA 256.392 Comprar
6 Venta Venta 6 30/09/2024 ABIERTA 430.48 Comprar
7 Venta Venta 7 07/10/2024 ABIERTA 168.704 Comprar
8 Venta Venta 8 29/09/2024 ABIERTA 121.49000000000001 Comprar
9 Venta Venta 9 04/10/2024 ABIERTA 279.992 Comprar

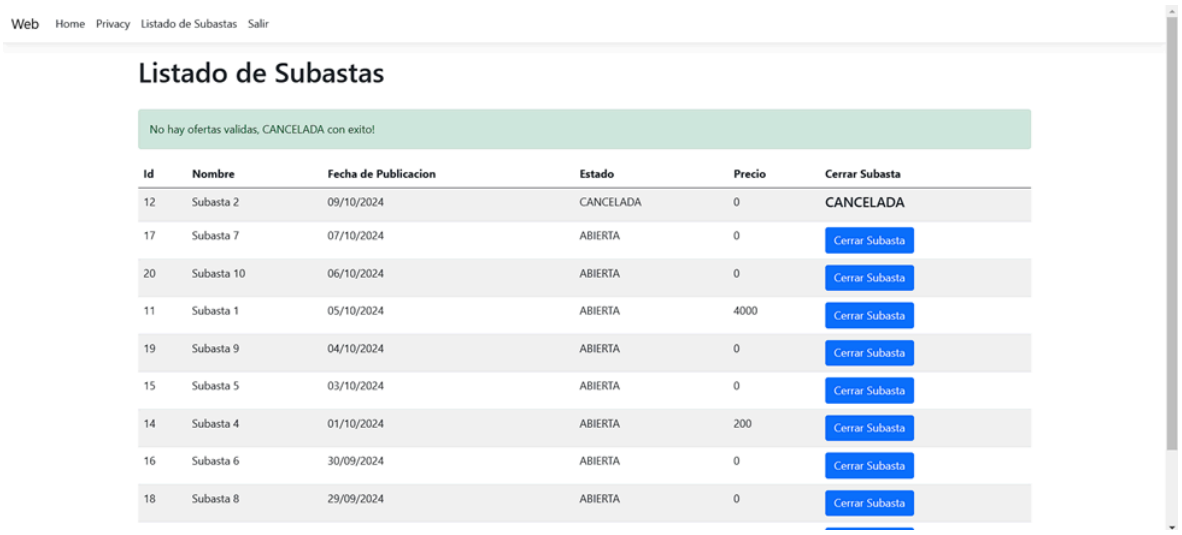
C18:

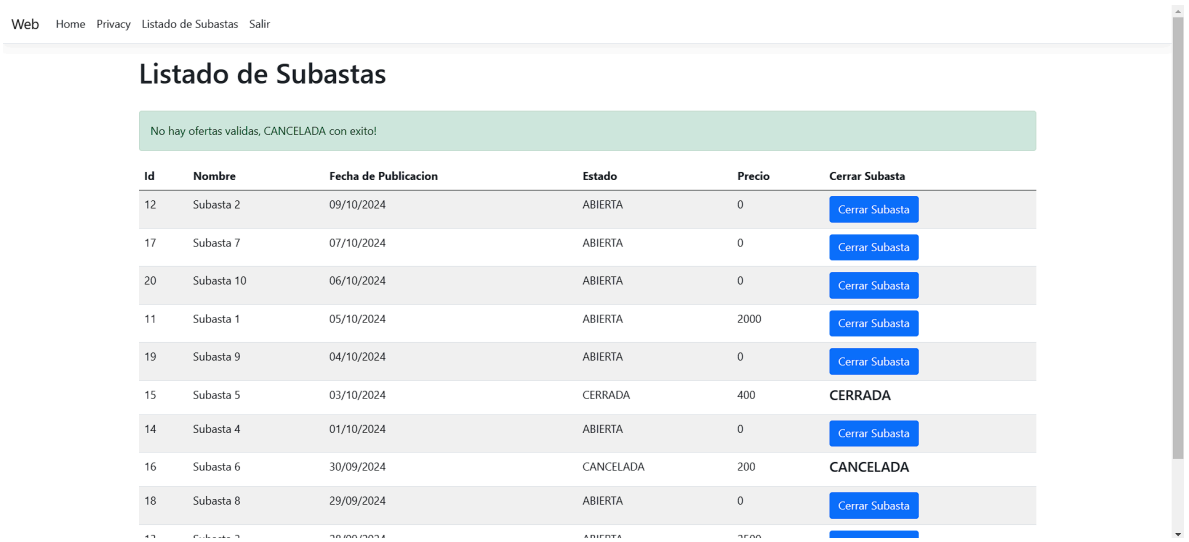

Web Home Privacy Listado de Publicaciones Modificar Saldo Salir
Modificar Saldo
Cargar Saldo
Ingrese saldo a cargar
Modificar Saldo
© 2024 - Web - Privacy


Web Home Privacy Listado de Publicaciones Modificar Saldo Salir
Modificar Saldo
El monto no puede ser negativo
Cargar Saldo
-1
Modificar Saldo
© 2024 - Web - Privacy

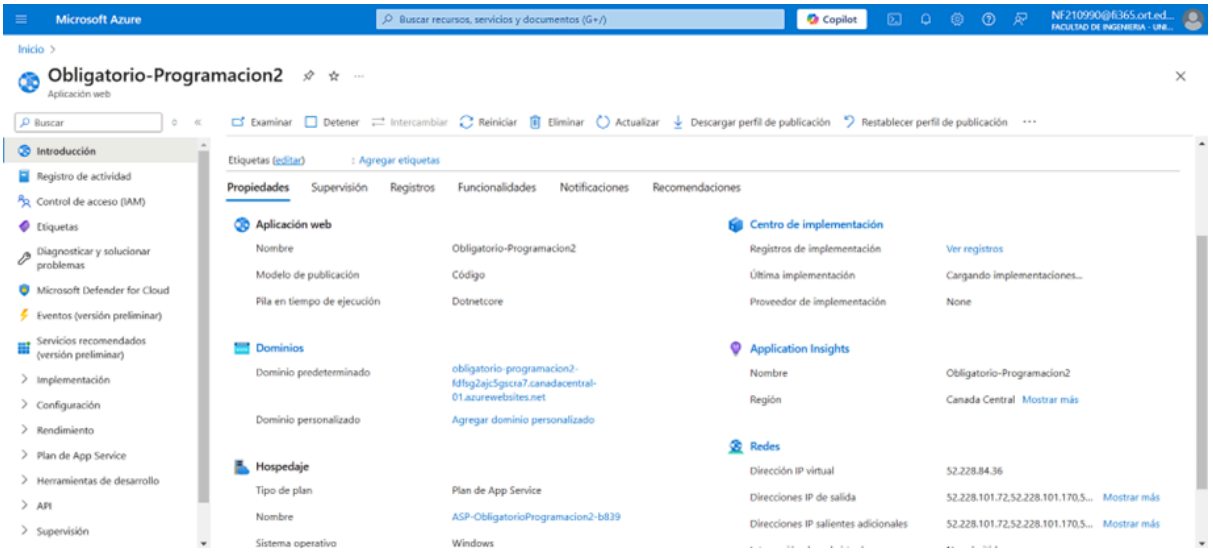

Web Home Privacy Listado de Publicaciones Modificar Saldo Salir
Modificar Saldo
El monto no puede ser vacio o 0
Cargar Saldo
Ingrese saldo a cargar
Modificar Saldo
© 2024 - Web - Privacy

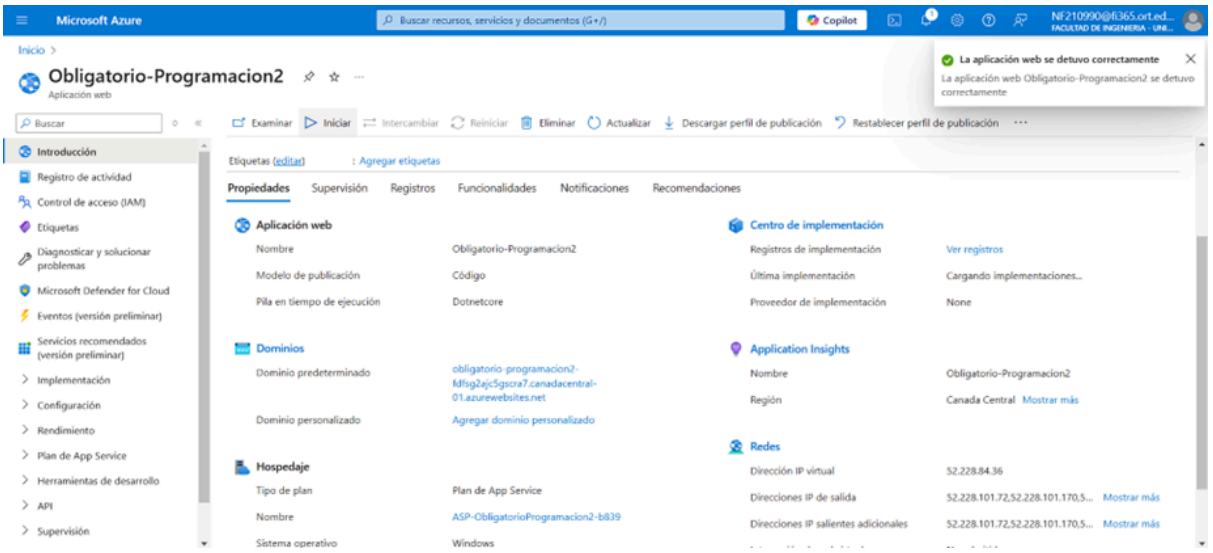
C19:

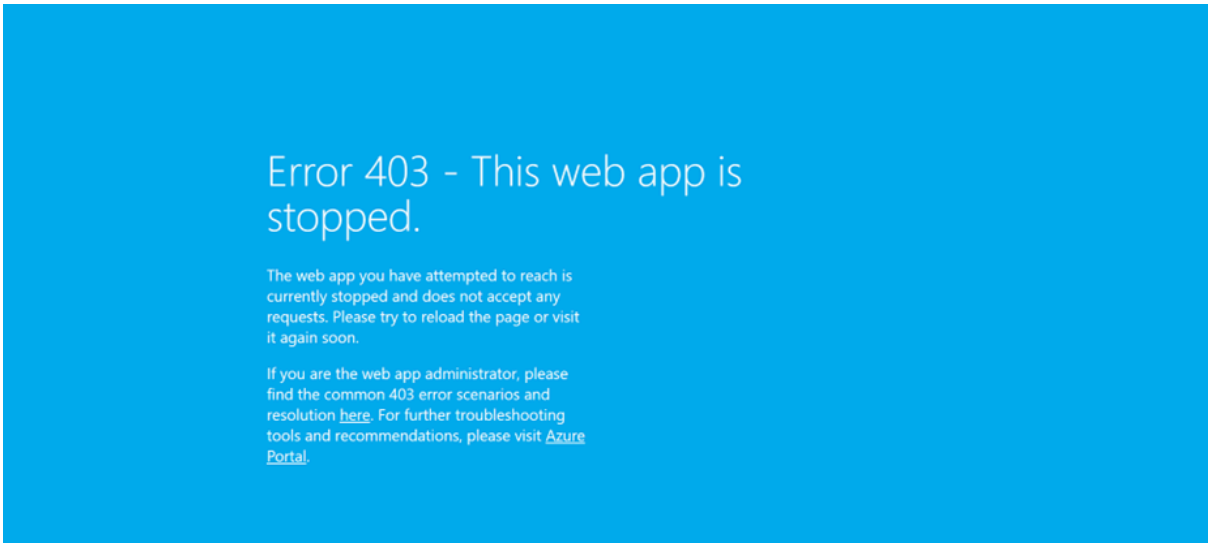

Web Home Privacy Listado de Subastas Salir
Listado de Subastas
No hay ofertas validas, CANCELADA con exito!
Id Nombre Fecha de Publicacion Estado Precio Cerrar Subasta
12 Subasta 2 09/10/2024 CANCELADA 0 CANCELADA
17 Subasta 7 07/10/2024 ABIERTA 0 Cerrar Subasta
20 Subasta 10 06/10/2024 ABIERTA 0 Cerrar Subasta
11 Subasta 1 05/10/2024 ABIERTA 4000 Cerrar Subasta
19 Subasta 9 04/10/2024 ABIERTA 0 Cerrar Subasta
15 Subasta 5 03/10/2024 ABIERTA 0 Cerrar Subasta
14 Subasta 4 01/10/2024 ABIERTA 200 Cerrar Subasta
16 Subasta 6 30/09/2024 ABIERTA 0 Cerrar Subasta
18 Subasta 8 29/09/2024 ABIERTA 0 Cerrar Subasta


Web Home Privacy Listado de Subastas Salir
Listado de Subastas
No hay ofertas validas, CANCELADA con exito!
Id Nombre Fecha de Publicacion Estado Precio Cerrar Subasta
12 Subasta 2 09/10/2024 ABIERTA 0 Cerrar Subasta
17 Subasta 7 07/10/2024 ABIERTA 0 Cerrar Subasta
20 Subasta 10 06/10/2024 ABIERTA 0 Cerrar Subasta
11 Subasta 1 05/10/2024 ABIERTA 2000 Cerrar Subasta
19 Subasta 9 04/10/2024 ABIERTA 0 Cerrar Subasta
15 Subasta 5 03/10/2024 CERRADA 400 CERRADA
14 Subasta 4 01/10/2024 ABIERTA 0 Cerrar Subasta
16 Subasta 6 30/09/2024 CANCELADA 200 CANCELADA
18 Subasta 8 29/09/2024 ABIERTA 0 Cerrar Subasta

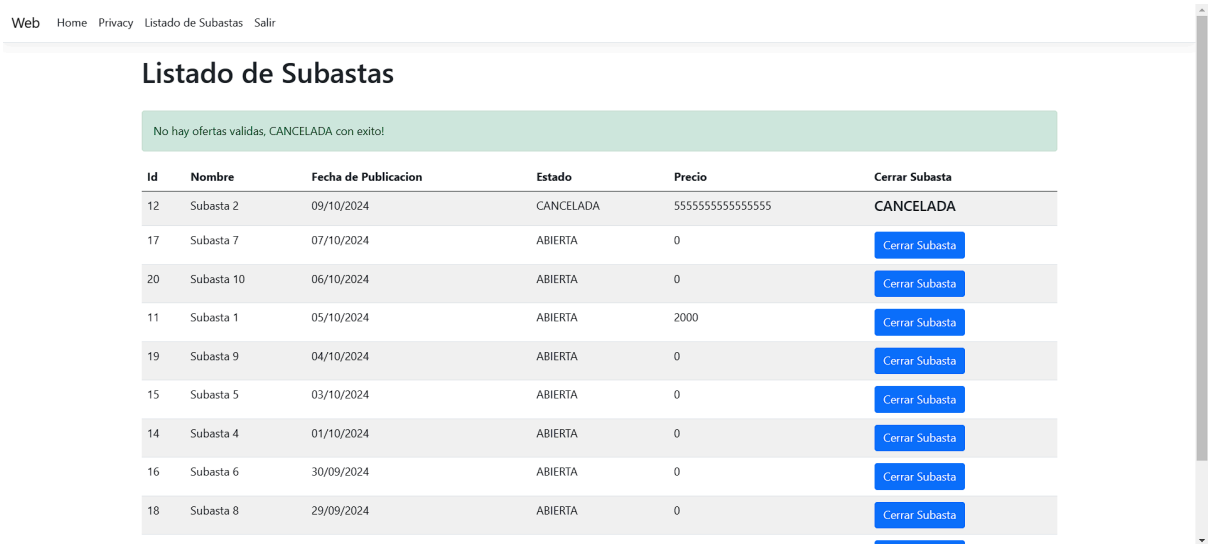
C20:


Microsoft Azure
Buscar recursos, servicios y documentos (G+/)
Copilot
Inicio
Obligatorio-Programacion2
Aplicación web
Examinar
Detener
Intercambiar
Reiniciar
Eliminar
Actualizar
Descargar perfil de publicación
Restablecer perfil de publicación
Introducción
Registro de actividad
Control de acceso (IAM)
Etiquetas
Diagnosticar y solucionar problemas
Microsoft Defender for Cloud
Eventos (versión preliminar)
Servicios recomendados (versión preliminar)
Implementación
Configuración
Rendimiento
Plan de App Service
Herramientas de desarrollo
API
Supervisión
Etiquetas (editar)
Agregar etiquetas
Propiedades
Supervisión
Registros
Funcionalidades
Notificaciones
Recomendaciones
Aplicación web
Nombre
Obligatorio-Programacion2
Modelo de publicación
Código
Pila en tiempo de ejecución
Dotnetcore
Dominios
Dominio predeterminado
obligatorio-programacion2-fdfsg2ajc5gscra7.canadacentral-01.azurewebsites.net
Dominio personalizado
Agregar dominio personalizado
Hospedaje
Tipo de plan
Plan de App Service
Nombre
ASP-ObligatorioProgramacion2-b839
Sistema operativo
Windows
Centro de implementación
Registros de implementación
Ver registros
Última implementación
Cargando implementaciones...
Proveedor de implementación
None
Application Insights
Nombre
Obligatorio-Programacion2
Región
Canada Central
Mostrar más
Redes
Dirección IP virtual
52.228.84.36
Direcciones IP de salida
52.228.101.72,52.228.101.170,5...
Mostrar más
Direcciones IP salientes adicionales
52.228.101.72,52.228.101.170,5...
Mostrar más


Microsoft Azure
Buscar recursos, servicios y documentos (G+/)
Copilot
Inicio
Obligatorio-Programacion2
Aplicación web
La aplicación web se detuvo correctamente
La aplicación web Obligatorio-Programacion2 se detuvo correctamente
Examinar
Iniciar
Intercambiar
Reiniciar
Eliminar
Actualizar
Descargar perfil de publicación
Restablecer perfil de publicación
Introducción
Registro de actividad
Control de acceso (IAM)
Etiquetas
Diagnosticar y solucionar problemas
Microsoft Defender for Cloud
Eventos (versión preliminar)
Servicios recomendados (versión preliminar)
Implementación
Configuración
Rendimiento
Plan de App Service
Herramientas de desarrollo
API
Supervisión
Etiquetas (editar)
Agregar etiquetas
Propiedades
Supervisión
Registros
Funcionalidades
Notificaciones
Recomendaciones
Aplicación web
Nombre
Obligatorio-Programacion2
Modelo de publicación
Código
Pila en tiempo de ejecución
Dotnetcore
Dominios
Dominio predeterminado
obligatorio-programacion2-fdfsg2ajc5gscra7.canadacentral-01.azurewebsites.net
Dominio personalizado
Agregar dominio personalizado
Hospedaje
Tipo de plan
Plan de App Service
Nombre
ASP-ObligatorioProgramacion2-b839
Sistema operativo
Windows
Centro de implementación
Registros de implementación
Ver registros
Última implementación
Cargando implementaciones...
Proveedor de implementación
None
Application Insights
Nombre
Obligatorio-Programacion2
Región
Canada Central
Mostrar más
Redes
Dirección IP virtual
52.228.84.36
Direcciones IP de salida
52.228.101.72,52.228.101.170,5...
Mostrar más
Direcciones IP salientes adicionales
52.228.101.72,52.228.101.170,5...
Mostrar más


Error 403 - This web app is stopped.
The web app you have attempted to reach is currently stopped and does not accept any requests. Please try to reload the page or visit it again soon.
If you are the web app administrator, please find the common 403 error scenarios and resolution here. For further troubleshooting tools and recommendations, please visit Azure Portal.

C21:



Link de URL a aplicación en Azure

https://obligatoriodeprogramacion2-dcaaepgjfhf8amc9.canadacentral-01.azurewebsites.net/

Tabla de Precarga

- PrecargarArticulos()

string nombre	**string categoria**	**double precioVenta**
Televisor	Electronica	450,99
Laptop	Electronica	999,99
Smartphone	Electronica	699,5
Cámara	Fotografia	299,99
Refrigerador	Electrodomesticos	1200
Lavadora	Electrodomesticos	800,75

Microondas	Cocina	150,99
Aspiradora	Hogar	250
Monitor	Computacion	199,99
Impresora	Oficina	120,5
Tablet	Electronica	350,49
Parlante Bluetooth	Electronica	79,99
Auriculares Inalámbricos	Electronica	120,89
Teclado Mecánico	Computacion	89,99
Ratón Inalámbrico	Computacion	45,99
Disco Duro Externo	Computacion	75,5
SSD	Computacion	150
Cámara de Seguridad	Seguridad	199,99
Termómetro Digital	Salud	35,99
Purificador de Aire	Hogar	299,99
Freidora de Aire	Cocina	180,49
Licuadora	Cocina	99,99
Horno Eléctrico	Cocina	249,5
Plancha	Electrodomesticos	65,99
Secadora	Electrodomesticos	950,75
Batidora	Cocina	55,99
Cafetera	Cocina	120
Reloj Inteligente	Electronica	199,99
Proyector	Electronica	499,99
Router Wi-Fi	Electronica	75,99
Silla de Oficina	Oficina	175,5
Escritorio	Oficina	299,99

Lámpara LED	Hogar	29,99
Aire Acondicionado	Electrodomesticos	899,99
Tostadora	Cocina	45,99
Ventilador	Hogar	69,99
Humidificador	Hogar	85,5
Calefactor	Hogar	120
Lava Vajillas	Cocina	600,75
Cepillo Eléctrico	Salud	45,99
Báscula Digital	Salud	50,99
Smartband	Electronica	99,99
Bicicleta Eléctrica	Transporte	1500
Scooter Eléctrico	Transporte	1200
Tijeras Eléctricas	Jardineria	99,99
Robot Aspiradora	Hogar	399,99
Papelera Inteligente	Hogar	45
Cámara Instantánea	Fotografia	120
Trípode	Fotografia	79,99

- PrecargarUsuarios()

Administradores

string nombre	**string apellido**	**string email**	**string contrasena**
Pablo	de Melo	pablodemelo@gmail.com	pablo123
Fabian	Fernandez	fabianfernandez@gmail.com	fabianfer333

Clientes

string nombre	**string apellido**	**string email**	**string contrasena**	**double saldo**

Cristian	Rodriguez	cristian12@gmail.com	abc12	23000
Sofia	Martinez	sofia.martinez@gmail.com	pass123	15000
Lucas	Fernandez	lucas.fernandez@gmail.com	fernandez456	18000
Valentina	Gomez	valentina.gomez@gmail.com	vale789	12000
Mateo	Lopez	mateo.lopez@gmail.com	lopez321	20000
Camila	Garcia	camila.garcia@gmail.com	cami654	25000
Juan	Perez	juan.perez@gmail.com	juan987	17000
Martina	Ruiz	martina.ruiz@gmail.com	ruiz111	22000
Joaquin	Mendez	joaquin.mendez@gmail.com	joaquin222	14000
Renata	Silva	renata.silva@gmail.com	silva333	26000

- PrecargarPublicaciones()

Venta

bool ofertaRelampago	string nombre	TipoEstado estado	DateTime fechaPublicacion	Cliente? comprador	Cliente? usuarioCierre	DateTime? fechaCierre
true	Venta 1	ABIERTA	2024-10-5	null	null	null
false	Venta 2	ABIERTA	2024-10-9	null	null	null
true	Venta 3	ABIERTA	2024-9-28	null	null	null
false	Venta 4	ABIERTA	2024-10-1	null	null	null
true	Venta 5	ABIERTA	2024-10-3	null	null	null
false	Venta 6	ABIERTA	2024-9-30	null	null	null
true	Venta 7	ABIERTA	2024-10-7	null	null	null
false	Venta 8	ABIERTA	2024-9-29	null	null	null
true	Venta 9	ABIERTA	2024-10-4	null	null	null
false	Venta 10	ABIERTA	2024-10-6	null	null	null

Subasta

string nombre	TipoEstado estado	DateTime fechaPublicacion	Cliente? comprador	Cliente? usuarioCierre	DateTime? fechaCierre
Subasta 1	ABIERTA	2024-10-5	null	null	null
Subasta 2	ABIERTA	2024-10-9	null	null	null
Subasta 3	ABIERTA	2024-9-28	null	null	null
Subasta 4	ABIERTA	2024-10-1	null	null	null
Subasta 5	ABIERTA	2024-10-3	null	null	null
Subasta 6	ABIERTA	2024-9-30	null	null	null
Subasta 7	ABIERTA	2024-10-7	null	null	null
Subasta 8	ABIERTA	2024-9-29	null	null	null
Subasta 9	ABIERTA	2024-10-4	null	null	null
Subasta 10	ABIERTA	2024-10-6	null	null	null

- PrecargarOfertasASubastas()

int idSub	int idClie	double monto	DateTime fecha
11	3	1500	2024-10-05
11	7	2000	2024-10-06
13	4	1800	2024-10-03
13	9	2500	2024-10-04

- PrecargarArticulosAPublicaciones()

int idPubli	int idArt
1	1
1	2
2	3
2	4
3	5
3	6
4	7
4	8
5	9
5	10
6	11
6	12
7	13
7	14
8	15
8	16
9	17
9	18
10	19
10	20
11	21
11	22
12	23
12	24
13	25

13	26
14	27
14	28
15	29
15	30
16	31
16	32
17	33
17	34
18	35
18	36
19	37
19	38
20	39
20	40

Precarga de Entidades con ChatGPT

- **https://chatgpt.com/share/67071c61-c318-800a-96ec-81e337a1cd80**

- **https://chatgpt.com/share/66f45260-8e1c-8000-96a9-b600ab75a7fb**

Código comentado

****
Archivo: Administrador.cs
Carpeta: D:\ORT\ObligatorioP2\Dominio\Administrador.cs
****

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

```
using System.Threading.Tasks;

namespace Dominio
{
    public class Administrador : Usuario
    {
        public Administrador(string nombre, string apellido, string email, string
contrasena):base(nombre, apellido, email, contrasena)
        {

        }
        public int Id
        {
            get { return _id; }
        }

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

        public override bool EsCliente()
        {
            return false;
        }
    }
}
```

****
Archivo: Articulo.cs
Carpeta: D:\ORT\ObligatorioP2\Dominio\Articulo.cs
****

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

namespace Dominio
{
    public class Articulo : IValidable
    {
        private int _id;
        private static int s_ultId = 1;
        private string _nombre;
        private string _categoria;
        private double _precioVenta;
```

```
    public Articulo(string nombre, string categoria, double precioVenta)
    {
        _id = s_ultId;
        s_ultId++;
        _nombre = nombre;
        _categoria = categoria;
        _precioVenta = precioVenta;
    }
    public double PrecioVenta
    {
        get { return _precioVenta; }
    }
    public int Id
    {
        get { return _id; }
    }
    public string Nombre
    {
        get { return _nombre; }
    }

    public string Categoria
    {
        get { return _categoria; }
    }

    public void Validar()
    {
        if (string.IsNullOrEmpty(_nombre)) throw new Exception("Por favor! Ingrese el
nombre del articulo!");
        if (string.IsNullOrEmpty(_categoria)) throw new Exception("Por favor! Ingrese una
categoria!");
        if (_precioVenta <= 0) throw new Exception("El precio debe ser mayor a 0");
    }

    public override string ToString()
    {
        return $"Articulo: {_nombre} - Categoria: {_categoria} - Precio: ${_precioVenta}";
    }

    public override bool Equals(object? obj)
    {
        Articulo a = obj as Articulo;
        return a != null && this._id.Equals(a._id);
    }

}
```

```
}

***********************************
Archivo: Cliente.cs
Carpeta: D:\ORT\ObligatorioP2\Dominio\Cliente.cs
***********************************
using Dominio.Interfaces;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Dominio
{
    public class Cliente : Usuario
    {
        private double _saldo;

        public Cliente(string nombre, string apellido, string email, string contrasena, double
saldo):base(nombre, apellido, email, contrasena)
        {
            _saldo = saldo;
        }
        public int Id
        {
            get { return _id; }
        }
        public string Nombre
        {
            get { return _nombre; }
        }
        public string Apellido
        {
            get { return _apellido; }
        }

        public double Saldo
        {
            get { return _saldo; }
            set { _saldo = value; }
        }

        public void Validar()
        {
            if (_saldo <= 0) throw new Exception("El saldo debe ser mayor a 0");
        }
```

```
        //metodo para modificar el saldo del cliente
        public void ModificarSaldo(double nuevoSaldo)
        {
            if (nuevoSaldo <= 0) throw new Exception("El saldo debe ser mayor a 0");
            _saldo = nuevoSaldo;
        }

        public override string ToString()
        {
            return $"id: {_id} - Nombre: {_nombre} {_apellido} - Email: {_email} -  saldo:
${_saldo}";
        }

        public override bool Equals(object? obj)
        {
            Cliente c = obj as Cliente;
            return c != null && this._id.Equals(c._id);
        }

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

        public override bool EsCliente()
        {
            return true;
        }

        public void DescontarSaldo(double monto)
        {
            if (_saldo >= monto)
            {
                Saldo -= monto;
            } else
            {
                throw new Exception("Saldo insuficiente");
            }
        }


    }
}
```

****

Archivo: OfertaSubasta.cs
Carpeta: D:\ORT\ObligatorioP2\Dominio\OfertaSubasta.cs

```
**********
using Dominio.Interfaces;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Dominio
{
    public class OfertaSubasta : IValidable
    {
        private int _id;
        private int s_ultId = 1;
        private Usuario _cliente;
        private double _monto;
        private DateTime _fecha;

        public OfertaSubasta(DateTime fecha, Usuario cliente, double monto)
        {
            _id = s_ultId;
            s_ultId++;
            _cliente = cliente;
            _monto = monto;
            _fecha = fecha;
        }

        public Usuario Cliente
        {
            get { return _cliente; }
            set { _cliente = value; }
        }

        public double Monto
        {
            get { return _monto; }
        }

        public void Validar()
        {
            if (_monto <= 0) throw new Exception("El monto no puede ser negativo o cero");
            if (_cliente == null) throw new Exception("El cliente no puede ser nulo.");
        }

        public override string ToString()
        {
            return $"id: {_id} cliente: {_cliente} - monto: {_monto} - fecha {_fecha}";
        }
```

```
    }
}

**********
Archivo: Publicacion.cs
Carpeta: D:\ORT\ObligatorioP2\Dominio\Publicacion.cs
**********
using Dominio.Interfaces;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Text.RegularExpressions;
using System.Threading.Tasks;

namespace Dominio
{
    public abstract class Publicacion : IValidable
    {
        protected int _id;
        protected static int s_ultId = 1;
        protected string _nombre;
        protected TipoEstado _estado;
        protected DateTime _fechaPublicacion;
        protected List<Articulo> _listaArticulos = new List<Articulo>();
        protected Usuario _comprador;
        protected Usuario _usuarioCierre;
        protected DateTime? _fechaCierre;

        public Publicacion(string nombre, TipoEstado estado, DateTime fechaPublicacion,
Usuario comprador, Usuario usuarioCierre, DateTime? fechaCierre)
        {
            _id = s_ultId;
            s_ultId++;
            _nombre = nombre;
            _estado = estado;
            _fechaPublicacion = fechaPublicacion;
            _comprador = comprador;
            _usuarioCierre = usuarioCierre;
            _fechaCierre = fechaCierre;
        }
        public abstract void FinalizarPublicacion(Usuario usuario);
        public abstract double CalculoUltimaOfertaPrecioFinal();
        public abstract bool EsSubasta();

        public TipoEstado Estado
        {
```

```
            get { return _estado; }
        }
        public DateTime FechaPublicacion
        {
            get { return _fechaPublicacion; }
        }
        public int Id
        {
            get { return _id; }
        }
        public string Nombre
        {
            get { return _nombre; }
        }

        public virtual void Validar()
        {
            if (string.IsNullOrEmpty(_nombre)) throw new Exception("El nombre no puede ser
vacio");
            if (_estado != TipoEstado.ABIERTA) throw new Exception("La publicación no se
encuentra en estado ABIERTA.");
            if (_estado == TipoEstado.CERRADA && _usuarioCierre == null) throw new
Exception("Debe de existir un usuario de cierre si la publicación está cerrada.");
            //if (_comprador == null) throw new Exception("El comprador no puede ser nulo");
//Esta mal porque si puede ser nulo
            if (_listaArticulos == null) throw new Exception("La lista de articulos no puede ser
nula");
        }



        public override string ToString()
        {
            return $"ID: {_id} - Nombre: {_nombre} - Estado: {_estado} - Fecha de Publicación:
{_fechaPublicacion}";

        }

        public void RegistrarArticulo(Articulo a)
        {
            if (a == null) throw new Exception("El articulo no puede ser nulo");
            a.Validar();
            _listaArticulos.Add(a);
        }

    }

}
```

```
**********************************
Archivo: Sistema.cs
Carpeta: D:\ORT\ObligatorioP2\Dominio\Sistema.cs
**********************************
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Dominio
{
   public class Sistema
   {

      private static Sistema s_instancia;

      private List<Publicacion> _listaPublicaciones = new List<Publicacion>();
      private List<Articulo> _listaArticulos = new List<Articulo>();
      private List<Usuario> _listaUsuarios = new List<Usuario>();

      #region properties
      public List<Usuario> Usuario
      {
         get
         {
            return _listaUsuarios;
         }
      }
      public List<Articulo> Articulos
      {
         get
         {
            return _listaArticulos;
         }
      }
      public List<Publicacion> Publicacion
      {
         get
         {
            return _listaPublicaciones;
         }
      }
      #endregion


      private Sistema()
      {
```

```
        // precargas de Publicaciones, articulos a publicaciones y ofertas con chat gpt:
https://chatgpt.com/share/67071c61-c318-800a-96ec-81e337a1cd80 /
        // https://chatgpt.com/share/672f9173-67c0-8013-9a48-9cc3d757ea7d
        PrecargarArticulos();
        PrecargarUsuarios();
        PrecargarPublicaciones();
        PrecargarArticulosAPublicaciones();
        PrecargarOfertasASubastas();
    }

    public static Sistema Instancia
    {
        get
        {
            if(s_instancia == null) s_instancia = new Sistema();
            return s_instancia;
        }
    }


    #region PRECARGAS

    // Precargados 50 articulos con la ayuda de ChatGPT
    private void PrecargarArticulos()
    {
        AltaArticulo(new Articulo("Televisor", "Electronica", 450.99));
        AltaArticulo(new Articulo("Laptop", "Electronica", 999.99));
        AltaArticulo(new Articulo("Smartphone", "Electronica", 699.50));
        AltaArticulo(new Articulo("Cámara", "Fotografia", 299.99));
        AltaArticulo(new Articulo("Refrigerador", "Electrodomesticos", 1200.00));
        AltaArticulo(new Articulo("Lavadora", "Electrodomesticos", 800.75));
        AltaArticulo(new Articulo("Microondas", "Cocina", 150.99));
        AltaArticulo(new Articulo("Aspiradora", "Hogar", 250.00));
        AltaArticulo(new Articulo("Monitor", "Computacion", 199.99));
        AltaArticulo(new Articulo("Impresora", "Oficina", 120.50));
        AltaArticulo(new Articulo("Tablet", "Electronica", 350.49));
        AltaArticulo(new Articulo("Parlante Bluetooth", "Electronica", 79.99));
        AltaArticulo(new Articulo("Auriculares Inalámbricos", "Electronica", 120.89));
        AltaArticulo(new Articulo("Teclado Mecánico", "Computacion", 89.99));
        AltaArticulo(new Articulo("Ratón Inalámbrico", "Computacion", 45.99));
        AltaArticulo(new Articulo("Disco Duro Externo", "Computacion", 75.50));
        AltaArticulo(new Articulo("SSD", "Computacion", 150.00));
        AltaArticulo(new Articulo("Cámara de Seguridad", "Seguridad", 199.99));
        AltaArticulo(new Articulo("Termómetro Digital", "Salud", 35.99));
        AltaArticulo(new Articulo("Purificador de Aire", "Hogar", 299.99));
        AltaArticulo(new Articulo("Freidora de Aire", "Cocina", 180.49));
        AltaArticulo(new Articulo("Licuadora", "Cocina", 99.99));
        AltaArticulo(new Articulo("Horno Eléctrico", "Cocina", 249.50));
```

```
            AltaArticulo(new Articulo("Plancha", "Electrodomesticos", 65.99));
            AltaArticulo(new Articulo("Secadora", "Electrodomesticos", 950.75));
            AltaArticulo(new Articulo("Batidora", "Cocina", 55.99));
            AltaArticulo(new Articulo("Cafetera", "Cocina", 120.00));
            AltaArticulo(new Articulo("Reloj Inteligente", "Electronica", 199.99));
            AltaArticulo(new Articulo("Proyector", "Electronica", 499.99));
            AltaArticulo(new Articulo("Router Wi-Fi", "Electronica", 75.99));
            AltaArticulo(new Articulo("Silla de Oficina", "Oficina", 175.50));
            AltaArticulo(new Articulo("Escritorio", "Oficina", 299.99));
            AltaArticulo(new Articulo("Lámpara LED", "Hogar", 29.99));
            AltaArticulo(new Articulo("Aire Acondicionado", "Electrodomesticos", 899.99));
            AltaArticulo(new Articulo("Tostadora", "Cocina", 45.99));
            AltaArticulo(new Articulo("Ventilador", "Hogar", 69.99));
            AltaArticulo(new Articulo("Humidificador", "Hogar", 85.50));
            AltaArticulo(new Articulo("Calefactor", "Hogar", 120.00));
            AltaArticulo(new Articulo("Lava Vajillas", "Cocina", 600.75));
            AltaArticulo(new Articulo("Cepillo Eléctrico", "Salud", 45.99));
            AltaArticulo(new Articulo("Báscula Digital", "Salud", 50.99));
            AltaArticulo(new Articulo("Smartband", "Electronica", 99.99));
            AltaArticulo(new Articulo("Bicicleta Eléctrica", "Transporte", 1500.00));
            AltaArticulo(new Articulo("Scooter Eléctrico", "Transporte", 1200.00));
            AltaArticulo(new Articulo("Tijeras Eléctricas", "Jardineria", 99.99));
            AltaArticulo(new Articulo("Robot Aspiradora", "Hogar", 399.99));
            AltaArticulo(new Articulo("Papelera Inteligente", "Hogar", 45.00));
            AltaArticulo(new Articulo("Cámara Instantánea", "Fotografia", 120.00));
            AltaArticulo(new Articulo("Trípode", "Fotografia", 79.99));
        }
        private void PrecargarUsuarios()
        {

            AltaUsuario(new Administrador("Pablo", "de Melo",
"pablodemelo@administrador.com", "pablo123"));
            AltaUsuario(new Administrador("Fabian", "Fernandez",
"fabianfernandez@administrador.com", "fabianfer333"));

            // Alta de clientes con ayuda de ChatGPT
            AltaUsuario(new Cliente("Cristian", "Rodriguez", "cristian12@cliente.com",
"abcdefgh12", 1170.00));
            AltaUsuario(new Cliente("Sofia", "Martinez", "sofia.martinez@cliente.com",
"password123", 15000.00));
            AltaUsuario(new Cliente("Lucas", "Fernandez", "lucas.fernandez@cliente.com",
"fernandez456", 18000.00));
            AltaUsuario(new Cliente("Valentina", "Gomez", "valentina.gomez@cliente.com",
"valentina789", 12000.00));
            AltaUsuario(new Cliente("Mateo", "Lopez", "mateo.lopez@cliente.com", "lopez3210",
20000.00));
            AltaUsuario(new Cliente("Camila", "Garcia", "camila.garcia@cliente.com",
"camila654", 25000.00));
```

```
            AltaUsuario(new Cliente("Juan", "Perez", "juan.perez@cliente.com", "juanperez987",
17000.00));
            AltaUsuario(new Cliente("Martina", "Ruiz", "martina.ruiz@cliente.com",
"ruizmartina111", 22000.00));
            AltaUsuario(new Cliente("Joaquin", "Mendez", "joaquin.mendez@cliente.com",
"joaquin222", 14000.00));
            AltaUsuario(new Cliente("Renata", "Silva", "renata.silva@cliente.com",
"silva333666", 26000.00));
        }
        private void PrecargarPublicaciones()
        {
            //Ventas
            AltaPublicacion(new Venta(true, "Venta 1", TipoEstado.ABIERTA, new
DateTime(2024, 10, 5), null, null, null));
            AltaPublicacion(new Venta(false, "Venta 2", TipoEstado.ABIERTA, new
DateTime(2024, 10, 9), null, null, null));
            AltaPublicacion(new Venta(true, "Venta 3", TipoEstado.ABIERTA, new
DateTime(2024, 9, 28), null, null, null));
            AltaPublicacion(new Venta(false, "Venta 4", TipoEstado.ABIERTA, new
DateTime(2024, 10, 1), null, null, null));
            AltaPublicacion(new Venta(true, "Venta 5", TipoEstado.ABIERTA, new
DateTime(2024, 10, 3), null, null, null));
            AltaPublicacion(new Venta(false, "Venta 6", TipoEstado.ABIERTA, new
DateTime(2024, 9, 30), null, null, null));
            AltaPublicacion(new Venta(true, "Venta 7", TipoEstado.ABIERTA, new
DateTime(2024, 10, 7), null, null, null));
            AltaPublicacion(new Venta(false, "Venta 8", TipoEstado.ABIERTA, new
DateTime(2024, 9, 29), null, null, null));
            AltaPublicacion(new Venta(true, "Venta 9", TipoEstado.ABIERTA, new
DateTime(2024, 10, 4), null, null, null));
            AltaPublicacion(new Venta(false, "Venta 10", TipoEstado.ABIERTA, new
DateTime(2024, 10, 6), null, null, null));


            //Subastas
            AltaPublicacion(new Subasta("Subasta 1", TipoEstado.ABIERTA, new
DateTime(2024, 10, 5), null, null, null));
            AltaPublicacion(new Subasta("Subasta 2", TipoEstado.ABIERTA, new
DateTime(2024, 10, 9), null, null, null));
            AltaPublicacion(new Subasta("Subasta 3", TipoEstado.ABIERTA, new
DateTime(2024, 9, 28), null, null, null));
            AltaPublicacion(new Subasta("Subasta 4", TipoEstado.ABIERTA, new
DateTime(2024, 10, 1), null, null, null));
            AltaPublicacion(new Subasta("Subasta 5", TipoEstado.ABIERTA, new
DateTime(2024, 10, 3), null, null, null));
            AltaPublicacion(new Subasta("Subasta 6", TipoEstado.ABIERTA, new
DateTime(2024, 9, 30), null, null, null));
```

```
        AltaPublicacion(new Subasta("Subasta 7", TipoEstado.ABIERTA, new
DateTime(2024, 10, 7), null, null, null));
        AltaPublicacion(new Subasta("Subasta 8", TipoEstado.ABIERTA, new
DateTime(2024, 9, 29), null, null, null));
        AltaPublicacion(new Subasta("Subasta 9", TipoEstado.ABIERTA, new
DateTime(2024, 10, 4), null, null, null));
        AltaPublicacion(new Subasta("Subasta 10", TipoEstado.ABIERTA, new
DateTime(2024, 10, 6), null, null, null));


    }
    private void PrecargarOfertasASubastas()
    {
        // Ofertas para la subasta 11
        AgregarOfertaASubasta(11, 3, 1500, new DateTime(2024, 10, 05));
        AgregarOfertaASubasta(11, 7, 2000, new DateTime(2024, 10, 06));
        // Ofertas para la subasta 13
        AgregarOfertaASubasta(13, 4, 1800, new DateTime(2024, 10, 03));
        AgregarOfertaASubasta(13, 9, 2500, new DateTime(2024, 10, 04));


    }
    private void PrecargarArticulosAPublicaciones()
    {
        // Artículos para las ventas y subastas
        AgregarArticuloAPublicacion(1, 1);
        AgregarArticuloAPublicacion(1, 2);

        AgregarArticuloAPublicacion(2, 3);
        AgregarArticuloAPublicacion(2, 4);

        AgregarArticuloAPublicacion(3, 5);
        AgregarArticuloAPublicacion(3, 6);

        AgregarArticuloAPublicacion(4, 7);
        AgregarArticuloAPublicacion(4, 8);

        AgregarArticuloAPublicacion(5, 9);
        AgregarArticuloAPublicacion(5, 10);

        AgregarArticuloAPublicacion(6, 11);
        AgregarArticuloAPublicacion(6, 12);

        AgregarArticuloAPublicacion(7, 13);
        AgregarArticuloAPublicacion(7, 14);

        AgregarArticuloAPublicacion(8, 15);
        AgregarArticuloAPublicacion(8, 16);
```

```
            AgregarArticuloAPublicacion(9, 17);
            AgregarArticuloAPublicacion(9, 18);

            AgregarArticuloAPublicacion(10, 19);
            AgregarArticuloAPublicacion(10, 20);

            AgregarArticuloAPublicacion(11, 21);
            AgregarArticuloAPublicacion(11, 22);

            AgregarArticuloAPublicacion(12, 23);
            AgregarArticuloAPublicacion(12, 24);

            AgregarArticuloAPublicacion(13, 25);
            AgregarArticuloAPublicacion(13, 26);

            AgregarArticuloAPublicacion(14, 27);
            AgregarArticuloAPublicacion(14, 28);

            AgregarArticuloAPublicacion(15, 29);
            AgregarArticuloAPublicacion(15, 30);

            AgregarArticuloAPublicacion(16, 31);
            AgregarArticuloAPublicacion(16, 32);

            AgregarArticuloAPublicacion(17, 33);
            AgregarArticuloAPublicacion(17, 34);

            AgregarArticuloAPublicacion(18, 35);
            AgregarArticuloAPublicacion(18, 36);

            AgregarArticuloAPublicacion(19, 37);
            AgregarArticuloAPublicacion(19, 38);

            AgregarArticuloAPublicacion(20, 39);
            AgregarArticuloAPublicacion(20, 40);
        }
        #endregion

        #region AGREGACIONES
        public void AgregarArticuloAPublicacion(int idPubli, int idArt) //Agregamos un articulo a
la lista de articulos de una publicacion
        {
            Articulo a = ObtenerArticulosPorId(idArt); //buscamos el articulo y vaidamos
            if (a == null) throw new Exception("El articulo no puede ser nulo");
            Publicacion p = ObtenerPublicacionPorId(idPubli); //buscamos la publicacion y la
validamos
            if (p == null) throw new Exception("La publicacion no puede ser nula");
```

```
            p.RegistrarArticulo(a);//Añadimos el articulo a la publicacion
        }

        public void AgregarOfertaASubasta(int idSub, int idClie, double monto, DateTime fecha)
        {
            //Agregamos una oferta a una subasta, validando si el cliente nunca
            //realizo una oferta en esa subasta y si el monto ofertado es superior al ultimo.
            Subasta s = ObtenerSubastaPorId(idSub);
            if (s == null) throw new Exception("La subasta no puede ser nula");
            OfertaSubasta ofe = new OfertaSubasta(fecha, ObtenerClientePorId(idClie), monto);
            s.RegistrarOferta(ofe);
        }
        #endregion

        #region ALTAS
        public void AltaArticulo(Articulo articulo)
        {
            if (articulo == null) throw new Exception("El articulo no puede ser nulo");
            articulo.Validar();
            if (_listaArticulos.Contains(articulo)) throw new Exception("El articulo ya existe.");
//agregado con el metodo equals en Articulo
            _listaArticulos.Add(articulo);
        }
        public void AltaUsuario(Usuario usuario)
        {
            if (usuario == null) throw new Exception("El Usuario no puede ser nulo");

            // Verificar que el email sea único antes de agregar el usuario
            foreach (Usuario u in _listaUsuarios)
            {
                if (u.Email == usuario.Email)
                {
                    throw new Exception("El email ya está registrado. Por favor, use uno
diferente.");
                }
            }

            // Validar el usuario
            usuario.Validar();

            // Agregar el usuario a la lista
            _listaUsuarios.Add(usuario);
        }

        public void AltaPublicacion(Publicacion publi)
        {
            if (publi == null) throw new Exception("La publicacion no puede ser nula");
            publi.Validar();
```

```
            _listaPublicaciones.Add(publi);
        }
        #endregion

        public List<Subasta> SubastasOrdenadasPorFecha()
        {
            // Crear una nueva lista para almacenar las subastas
            List<Subasta> subastas = new List<Subasta>();

            // Recorrer la lista de publicaciones y agregar solo aquellas que son subastas
            foreach (Publicacion publicacion in _listaPublicaciones)
            {
                if (publicacion.EsSubasta())
                {
                    subastas.Add((Subasta)publicacion); // (Subasta)publicacion convierte el objeto publicacion de tipo Publicacion (su clase base)
                                                        // al tipo Subasta (una clase derivada).
                }
            }

            // Ordenar las subastas por la fecha de publicación
            subastas.Sort();

            // Retornar la lista de subastas ordenadas
            return subastas;
        }

        public Usuario Login(string email, string contrasena)
        {
            Usuario usuarioBuscado = null;
            int i = 0;

            while(usuarioBuscado == null && i < _listaUsuarios.Count)
            {
                if (_listaUsuarios[i].Email == email && _listaUsuarios[i].Contrasena == contrasena)
                usuarioBuscado = _listaUsuarios [i];
                i++;
            }

            return usuarioBuscado;
        }
        public List<Articulo> ArticulosPorCategoria(string categoria)
        {
            List<Articulo> buscados = new List<Articulo>();
            foreach (Articulo a in _listaArticulos)
            {
                // Pasamos los dos strings a minusculas y los comparamos. si son iguales lo agregamos a los buscados
```

```
            if (a.Categoria.ToLower() == categoria.ToLower()) buscados.Add(a);
        }
        return buscados;

    }

    public List<Publicacion> ListarPublicacionesEntreFechas(DateTime fechaInicio,
DateTime fechaFin)
    {
        List<Publicacion> buscados = new List<Publicacion>();
        foreach (Publicacion p in _listaPublicaciones)
        {
            //comprobamos que la fecha de publicacion de una publicacion este dentro de las
dos fechas solicitadas.
            if (p.FechaPublicacion >= fechaInicio && p.FechaPublicacion <= fechaFin)
buscados.Add(p);
        }
        return buscados;
    }

    public List<Cliente> ListarClientes()
    {
        List<Cliente> buscados = new List<Cliente>();
        foreach (Usuario u in _listaUsuarios)
        {
            // comprueba que Usuario u sea un Cliente y no Administrador y lo agrega a la
lista.
            if (u is Cliente cliente)
            {
                buscados.Add(u as Cliente);
            }

        }
        return buscados;
    }

    public void FinalizarPublicacion(Publicacion publicacion, Usuario usuario)
    {
        publicacion.FinalizarPublicacion(usuario);
    }

    ///pasamanos para obtener al cliente que se quiere modificar el saldo y asignarle el
saldo nuevo
    public void ModificarSaldoDeCliente(int idUsuario, double nuevoSaldo)
    {
        Cliente c = ObtenerClientePorId(idUsuario);

        if (c == null) throw new Exception("El cliente no se encontró");
```

```
    if (nuevoSaldo <= 0) throw new Exception("El monto no puede ser  vacio o 0");
    c.ModificarSaldo(c.Saldo + nuevoSaldo);

}

#region OBTENER POR ID
public Cliente ObtenerClientePorId(int id)
{
    Cliente buscado = null;
    int i = 0;

    while (i < _listaUsuarios.Count && buscado == null)
    {
        // Intentamos convertir el usuario a Cliente usando 'as'
        Cliente cliente = _listaUsuarios[i] as Cliente;

        // Si la conversión fue exitosa (cliente no es null) y el ID coincide
        if (cliente != null && cliente.Id == id)
        {
            buscado = cliente;
        }

        i++;
    }

    return buscado;
}
public Administrador ObtenerAdministradorPorId(int id)
{
    Administrador buscado = null;
    int i = 0;

    while (i < _listaUsuarios.Count && buscado == null)
    {
        // Intentamos convertir el usuario a Administrador usando 'as'
        Administrador administrador = _listaUsuarios[i] as Administrador;

        // Si la conversión fue exitosa (administrador no es null) y el ID coincide
        if (administrador != null && administrador.Id == id)
        {
            buscado = administrador;
        }

        i++;
    }

    return buscado;
}
```

```
public Publicacion ObtenerPublicacionPorId(int id)
{
    Publicacion buscado = null;
    int i = 0;
    while (i < _listaPublicaciones.Count && buscado == null)
    {
        if (_listaPublicaciones[i].Id == id) buscado = _listaPublicaciones[i];
        i++;
    }
    return buscado;
}
public Articulo ObtenerArticulosPorId(int id)
{
    Articulo buscado = null;
    int i = 0;
    while (i < _listaArticulos.Count && buscado == null)
    {
        if (_listaArticulos[i].Id == id) buscado = _listaArticulos[i];
        i++;
    }

    return buscado;
}

public Subasta ObtenerSubastaPorId(int id)
{
    Subasta buscado = null;
    int i = 0;
    while (i < _listaPublicaciones.Count && buscado == null)
    {
        Subasta sub = _listaPublicaciones[i] as Subasta;

        if (_listaPublicaciones[i].Id == id) buscado = sub;
        i++;
    }

    return buscado;
}
public Usuario ObtenerUsuarioPorEmail(string email)
{
    Usuario buscado = null;
    int i = 0;
    while (i < _listaUsuarios.Count && buscado == null)
    {
        Usuario u = _listaUsuarios[i];
        if (_listaUsuarios[i].Email == email) buscado = u;
        i++;
    }
```

```
            return buscado;
        }


    }

}
#endregion
**********
Archivo: Subasta.cs
Carpeta: D:\ORT\ObligatorioP2\Dominio\Subasta.cs
**********
using Dominio.Interfaces;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Dominio
{
    public class Subasta : Publicacion, IComparable<Subasta>
    {
        private List<OfertaSubasta>? _listaOferta = new List<OfertaSubasta>();

        public Subasta(string nombre, TipoEstado estado, DateTime fechaPublicacion, Usuario
comprador, Usuario usuarioCierre, DateTime? fechaCierre):base(nombre, estado,
fechaPublicacion, comprador, usuarioCierre, fechaCierre)
        {

        }
        public List<OfertaSubasta> Ofertas
        {
            get { return _listaOferta; }
        }

        public override void Validar(){
            if (_listaOferta == null) throw new Exception("La lista de ofertas no puede ser nula");

        }

        public override double CalculoUltimaOfertaPrecioFinal()
        {
            if (_listaOferta.Count >= 1)
            {
                return _listaOferta[(_listaOferta.Count) - 1].Monto;
            }
            else return 0;
```

```
    }
    public bool ValidarUnicaOferta(Cliente cliente)
    {
        foreach (OfertaSubasta o in _listaOferta)
        {
            if (o.Cliente.Equals(cliente)) return true;
        }
        return false;
    }
    public void RegistrarOferta(OfertaSubasta ofe)

    {
        foreach (OfertaSubasta o in _listaOferta) //comprobamos que un cliente realize
unicamente una oferta
        {
            if (o.Cliente.Equals(ofe.Cliente)) throw new Exception("El cliente ya realizo una
oferta en esta publicacion.");
        }

        //comprobamos que un cliente realize una oferta con el monto mas alto al anterior
        double ultMonto = 0;
        if (_listaOferta.Count > 0) ultMonto = _listaOferta[(_listaOferta.Count - 1)].Monto;
        if (ofe == null) throw new Exception("La oferta no puede ser nula");
        if (ofe.Monto <= ultMonto) throw new Exception("La oferta no puede ser menor o
igual a la oferta anterior");
        ofe.Validar();
        _listaOferta.Add(ofe);
    }


    public override void FinalizarPublicacion(Usuario usuario)
    {
        // Lanzar excepción si el usuario no es un Administrador
        if (usuario.EsCliente())
        {
            throw new Exception("Solo un administrador puede cerrar la subasta.");
        }

        Administrador administrador = (Administrador)usuario; // Hacer el cast directo

        // Ciclo para verificar si hay una oferta válida
        bool subastaFinalizada = false; // Control para finalizar el bucle
        while (!subastaFinalizada)
        {
            OfertaSubasta mejorOferta = ObtenerPrimeraOfertaConSaldo();

            if (mejorOferta == null)
```

```
        {
            // Si la oferta es null o sea igual a 0, cambiar estado a CANCELADA
            _estado = TipoEstado.CANCELADA;
            _usuarioCierre = administrador;
            _fechaCierre = DateTime.Now;
            subastaFinalizada = true; // Salir del bucle
        }
        else
        {
            // Procesar la oferta si es válida
            Cliente clienteFinal = (Cliente)mejorOferta.Cliente; // Cast directo
            clienteFinal.DescontarSaldo(mejorOferta.Monto);
            _comprador = mejorOferta.Cliente; // Asignar el comprador
            _estado = TipoEstado.CERRADA;
            _usuarioCierre = administrador;
            _fechaCierre = DateTime.Now;
            subastaFinalizada = true; // Salir del bucle
        }
    }
}

private OfertaSubasta ObtenerPrimeraOfertaConSaldo()
{
    OfertaSubasta buscado = null;
    int i = _listaOferta.Count - 1;
    while (buscado == null && i >= 0)
    {
        Cliente c = _listaOferta[i].Cliente as Cliente;
        if (c.Saldo >= _listaOferta[i].Monto)
        {
            buscado = _listaOferta[i]; // devolvemos la primera oferta válida
        }
        i--;
    }

    return buscado; // Si no hay ninguna oferta válida
}

public double ObtenerOfertaDeCliente(Cliente c)
{
    foreach (OfertaSubasta os in _listaOferta)
    {
        if (os.Cliente.Equals(c)) return os.Monto;
    }
    return 0.0;
}
public override bool EsSubasta()
{
```

```
            return true; // Modifico para confirmar que es una subasta
        }

        public int CompareTo(Subasta? other)
        {
            return other.FechaPublicacion.CompareTo(this.FechaPublicacion);
        }
    }
}
```

****
Archivo: TipoEstado.cs
Carpeta: D:\ORT\ObligatorioP2\Dominio\TipoEstado.cs
****

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

namespace Dominio
{
    public enum TipoEstado
    {
        ABIERTA,
        CERRADA,
        CANCELADA
    }
}
```

****
Archivo: Usuario.cs
Carpeta: D:\ORT\ObligatorioP2\Dominio\Usuario.cs
****

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

namespace Dominio
{
    public abstract class Usuario : IValidable
    {
        protected int _id;
        protected static int s_ultId = 1;
        protected string _nombre;
```

```
    protected string _apellido;
    protected string _email;
    protected string _contrasena;

    public Usuario(string nombre, string apellido, string email, string contrasena)
    {
        _id = s_ultId;
        s_ultId++;
        _nombre = nombre;
        _apellido = apellido;
        _email = email;
        _contrasena = contrasena;
    }
    public string Email
    {
        get { return _email; }
    }

    public string Contrasena
    {
        get { return _contrasena; }
    }
    public int Id
    {
        get { return _id; }
    }

    public double Saldo { get; internal set; }

    public abstract string Rol();
    public abstract bool EsCliente();

    public void Validar()
    {
        if (string.IsNullOrEmpty(_nombre) || string.IsNullOrEmpty(_apellido) ||
            string.IsNullOrEmpty(_email) || string.IsNullOrEmpty(_contrasena))
            throw new Exception("Los campos de nombre, apellido, contraseña e email son
obligatorios.");

        // Validación de formato de email
        if (!_email.Contains("@") || !_email.Contains(".")) throw new Exception("El formato
del campo email no es válido");

        // Validación de la longitud de la contraseña
        if (_contrasena.Length < 8) throw new Exception("La contraseña debe tener al
menos 8 caracteres");

    }
```

```
        public override string ToString()
        {
            return $"nombre: {_nombre} - apellido: {_apellido} email: {_email}";
        }

    }
}
```

****
Archivo: Venta.cs
Carpeta: D:\ORT\ObligatorioP2\Dominio\Venta.cs
****

```
using Dominio.Interfaces;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using static System.Runtime.InteropServices.JavaScript.JSType;

namespace Dominio
{
    public class Venta : Publicacion
    {
        private bool _ofertaRelampago;

        public Venta(bool ofertaRelampago, string nombre, TipoEstado estado, DateTime
fechaPublicacion, Usuario comprador, Usuario usuarioCierre, DateTime? fechaCierre)
:base(nombre, estado, fechaPublicacion, comprador, usuarioCierre, fechaCierre)
        {
            _ofertaRelampago = ofertaRelampago;
        }

        public override bool EsSubasta()
        {
            return false;
        }

        public string TieneOfertaRelampago() //retornar si la venta tiene oferta relampago
        {
            if (_ofertaRelampago) return "Si";
            else return "No";
        }

        public override void FinalizarPublicacion(Usuario usuario)
        {
            if (usuario.EsCliente())
```

```
        {
            Cliente cliente = (Cliente)usuario;  // Hacer el cast directo, ya que sabemos que es
un Cliente
            double precioFinal = CalculoUltimaOfertaPrecioFinal();

            if (cliente.Saldo < precioFinal)
            {
                throw new Exception("Saldo insuficiente para completar la compra.");
            }

            cliente.DescontarSaldo(precioFinal);
            _comprador = cliente;
            _estado = TipoEstado.CERRADA;
            _usuarioCierre = cliente;
            _fechaCierre = DateTime.Now;
        }
        else
        {
            throw new Exception("Solo un cliente puede finalizar esta venta.");
        }
    }

    public override double CalculoUltimaOfertaPrecioFinal()
    {
        double precioTotal = 0;


        foreach (Articulo articulo in _listaArticulos)
        {
            precioTotal += articulo.PrecioVenta;
        }

        // Aplicar descuento del 20% si hay una oferta relámpago
        if (_ofertaRelampago)
        {
            precioTotal *= 0.8;
        }

        return precioTotal;
    }



  }



}
```

****
Archivo: IValidable.cs
Carpeta: D:\ORT\ObligatorioP2\Dominio\Interfaces\IValidable.cs
****

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

namespace Dominio.Interfaces
{
    internal interface IValidable
    {
        public void Validar();
    }
}
```

****
Archivo: HomeController.cs
Carpeta: D:\ORT\ObligatorioP2\Web\Controllers\HomeController.cs
****

```
using Microsoft.AspNetCore.Mvc;
using System.Diagnostics;
using Web.Models;

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

        public HomeController(ILogger<HomeController> logger)
        {
            _logger = logger;
        }

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

        public IActionResult Privacy()
        {
            return View();
        }
```

```
        [ResponseCache(Duration = 0, Location = ResponseCacheLocation.None, NoStore =
true)]
        public IActionResult Error()
        {
            return View(new ErrorViewModel { RequestId = Activity.Current?.Id ??
HttpContext.TraceIdentifier });
        }
    }
}

**********
Archivo: PublicacionesController.cs
Carpeta: D:\ORT\ObligatorioP2\Web\Controllers\PublicacionesController.cs
**********
using Dominio;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using
Microsoft.VisualStudio.Web.CodeGenerators.Mvc.Templates.BlazorIdentity.Pages.Manage;
using System;
using System.Net.Http;
using System.Runtime.Intrinsics.Arm;
using static System.Runtime.InteropServices.JavaScript.JSType;

namespace Web.Controllers
{
    public class PublicacionesController : Controller
    {
        Sistema miSistema = Sistema.Instancia;

        [HttpGet]
        public IActionResult ListadoSubastas()
        {
            if (HttpContext.Session.GetString("rol") == null || HttpContext.Session.GetString("rol")
!= "administrador")
            {
                return View("NoAutorizado");
            }
            // Obtengo las subastas ordenadas. El método SubastasOrdenadasPorFecha() y
devuelve una lista de Subasta.
            // Por lo tanto, no es necesario volver a verificar si un elemento es una Subasta
recorriendo en un bucle foreach.
            List<Subasta> subastas = miSistema.SubastasOrdenadasPorFecha();

            // Pasar las subastas a la vista
            ViewBag.ListadoSubastas = subastas;
            return View();
        }
```

```
        public IActionResult CerrarSubasta(int idSubasta)
        {
            if (HttpContext.Session.GetString("rol") == null || HttpContext.Session.GetString("rol")
!= "administrador")
            {
                return View("NoAutorizado");
            }

            try
            {
                Usuario u =
miSistema.ObtenerUsuarioPorEmail(HttpContext.Session.GetString("email"));
                Subasta s = miSistema.ObtenerPublicacionPorId(idSubasta) as Subasta;
                s.FinalizarPublicacion(u);
                Publicacion p = miSistema.ObtenerPublicacionPorId(idSubasta);
                if (p.Estado == TipoEstado.CANCELADA)
                {
                    TempData["Exito"] = $"No hay ofertas validas, CANCELADA con exito!";
                }
                else TempData["Exito"] = $"Subasta cerrada con exito!";

            }
            catch (Exception ex)
            {
                TempData["Error"] = ex.Message;
            }

            return RedirectToAction("ListadoSubastas");
        }

        [HttpGet]
        public IActionResult ListadoPublicaciones()
        {
            if (HttpContext.Session.GetString("rol") == null || HttpContext.Session.GetString("rol")
!= "cliente")
            {
                return View("NoAutorizado");
            }

            try
            {

            List<Publicacion> publicaciones = new List<Publicacion>();
            publicaciones = miSistema.Publicacion;
            ViewBag.ListadoPublicaciones = publicaciones;
            return View();
```

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


        public IActionResult Comprar(int idP)
        {
            if (HttpContext.Session.GetString("rol") == null || HttpContext.Session.GetString("rol")
!= "cliente")
            {
                return View("NoAutorizado");
            }

            try
            {
                Usuario u =
miSistema.ObtenerUsuarioPorEmail(HttpContext.Session.GetString("email"));
                Publicacion p = miSistema.ObtenerPublicacionPorId(idP);
                if (!p.EsSubasta())
                {
                    p.FinalizarPublicacion(u);
                }
                TempData["Exito"] = $"Publicacion comprada con exito!";
            }
            catch (Exception ex)
            {
                TempData["Error"] = ex.Message;
            }

            return RedirectToAction("ListadoPublicaciones");
        }

        [HttpGet]
        public IActionResult CambiarOfertarSubasta(int idS)
        {
            if (HttpContext.Session.GetString("rol") == null || HttpContext.Session.GetString("rol")
!= "cliente")
            {
                return View("NoAutorizado");
            }
            Cliente c =
miSistema.ObtenerUsuarioPorEmail(HttpContext.Session.GetString("email")) as Cliente;
            Subasta s = miSistema.ObtenerSubastaPorId(idS);
```

```
            if (s.ValidarUnicaOferta(c))
            {
                ViewBag.Valor = s.ObtenerOfertaDeCliente(c);
            }

            ViewBag.idSubasta = idS;
            return View();
        }

        [HttpPost]
        public IActionResult CambiarOfertarSubasta(int idS, double nuevaOferta)
        {
            if (HttpContext.Session.GetString("rol") == null || HttpContext.Session.GetString("rol")
!= "cliente")
            {
                return View("NoAutorizado");
            }

            try
            {
                Subasta s = miSistema.ObtenerSubastaPorId(idS);
                if (nuevaOferta < 0) throw new Exception("el saldo no puede ser negativo");
                OfertaSubasta oferta = new OfertaSubasta(DateTime.Now,
miSistema.ObtenerUsuarioPorEmail(HttpContext.Session.GetString("email")), nuevaOferta);
                s.RegistrarOferta(oferta);
                TempData["Exito"] = $"Oferta realizada con exito!";
            }
            catch (Exception ex)
            {
                TempData["Error"] = ex.Message;
            }
            return RedirectToAction("ListadoPublicaciones");


        }
    }
}
```

****
Archivo: UsuariosController.cs
Carpeta: D:\ORT\ObligatorioP2\Web\Controllers\UsuariosController.cs
****

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

namespace Web.Controllers
{
    public class UsuariosController : Controller
```

```
  {
      Sistema miSistema = Sistema.Instancia;

      [HttpGet]
      public IActionResult Login()
      {
          return View();
      }

      [HttpPost]
      public IActionResult Login(string email, string contrasena)
      {
          try
          {
              if (string.IsNullOrEmpty(email)) throw new Exception("Debe ingresar un email");
              if (string.IsNullOrEmpty(contrasena)) throw new Exception("Debe ingresar una
contraseña");
              Usuario usuario = miSistema.Login(email, contrasena);
              if (usuario == null) throw new Exception("Email o contraseña incorrectas");


              //Variables para el inicio de sesion
              HttpContext.Session.SetString("email", email); //IDENTIFICA EL USUARIO
              HttpContext.Session.SetString("rol", usuario.Rol()); //PERMITE SABER EL ROL
QUE CUMPLE

              return RedirectToAction("Index", "Home");

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

      }

      public IActionResult Logout()
      {
          HttpContext.Session.Clear();
          TempData["logout"] = $"Sesion cerrada Exitosamente!";
          return RedirectToAction("Login");
      }

      [HttpGet]
      public IActionResult RegistrarCliente()
      {
```

```
            return View();
        }

        [HttpPost]
        public IActionResult RegistrarCliente(string nombre, string apellido, string email, string
contrasena)
        {

            try
            {
                if (string.IsNullOrEmpty(nombre)) throw new Exception("El campo nombre no
puede ser vacío");
                if (string.IsNullOrEmpty(apellido)) throw new Exception("El campo apellido no
puede ser vacío");
                if (string.IsNullOrEmpty(email)) throw new Exception("El campo email no puede
ser vacío");
                if (string.IsNullOrEmpty(contrasena)) throw new Exception("El campo contraseña
no puede ser vacío");

                Usuario u = new Cliente(nombre, apellido, email, contrasena, 0);
                miSistema.AltaUsuario(u);

                ViewBag.Exito = $"Cliente {nombre} {apellido} registrado con éxito!";
            }
            catch (Exception ex)
            {
                ViewBag.Error = ex.Message;
                ViewBag.Nombre = nombre;
                ViewBag.Apellido = apellido;
                ViewBag.Email = email;
                ViewBag.Contrasena = contrasena;
            }
            return View();
        }

        [HttpGet]
        public IActionResult ModificarSaldoCliente()
        {
            if (HttpContext.Session.GetString("rol") == null || HttpContext.Session.GetString("rol")
!= "cliente")
            {
                return View("NoAutorizado");
            }

            return View();

        }
```

```
        [HttpPost]
        public IActionResult ModificarSaldoCliente(double nuevoSaldo)
        {
            if (HttpContext.Session.GetString("rol") == null || HttpContext.Session.GetString("rol")
!= "cliente")
            {
                return View("NoAutorizado");
            }

            try
            {

                Cliente c =
miSistema.ObtenerClientePorId(miSistema.ObtenerUsuarioPorEmail(HttpContext.Session.G
etString("email")).Id);
                if (nuevoSaldo < 0) throw new Exception("El monto no puede ser negativo");

                miSistema.ModificarSaldoDeCliente(c.Id, nuevoSaldo);
                ViewBag.Exito = $"Se modificó el saldo del cliente {c.Nombre} - Nuevo saldo:
${c.Saldo}";
            }
            catch (Exception ex)
            {
                ViewBag.Error = ex.Message;
            }
            return View();
        }
    }
}

**********
Archivo: ErrorViewModel.cs
Carpeta: D:\ORT\ObligatorioP2\Web\Models\ErrorViewModel.cs
**********
namespace Web.Models
{
    public class ErrorViewModel
    {
        public string? RequestId { get; set; }

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