

UNIVERSIDAD VERACRUZANA

Table of Contents

TECNOLOGÍAS PARA LA CONSTRUCCIÓN DE SOFTWARE	2
ACTIVIDAD: Entrega semanal de avances	2
Primer reporte de Avance del proyecto final: Juego Damas Chinas.....	2
Recursos implementados	2
Ventanas desarrolladas.....	6
Mapeo de Base de Datos con Entity Framework	10
Resumen de contribución por integrante	10
Uso de Inteligencia Artificial (IA)	10
Segundo reporte de Avance del proyecto final: Juego Damas Chinas	11
Server	11
AccountManager.cs	12
DataContract	12
Servicios	12
Operaciones / Funcionalidad	13
LoginService.cs	13
DataContract	13
Servicios	14
Operaciones / Funcionalidad	14
SingInService.cs	14
DataContract	14
Servicios	15
Operaciones / Funcionalidad	15
Logica	15
RepositorioUsuarios.cs	15
IMPLEMENTACION	16
SingIn	16
MenuRegisteredPlayer	18
ProfilePlayer	18
ChangeData	18
Evidencia posterior al cambio	20
DOCUMENTACION	20
Avance del Proyecto Final: Juego Damas Chinas (Entrega 2)	20
SETH	20
IVAN	21
Tercer reporte de entrega	21
Modificaciones a la BD	21

Cliente	22
CHAT	22
Friends	24
Servidor	26
AmistadService	26
MessageService	29
LOBBY	31
Avance del Proyecto Final: Juego Damas Chinas (Entrega 3)	34
SETH	34
IVAN	34

TECNOLOGÍAS PARA LA CONSTRUCCIÓN DE SOFTWARE

ACTIVIDAD: Entrega semanal de avances

Realizado por: Rodrigo Iván Ahumada Rodríguez (S21013886) Marquez Rodríguez Seth (S23014042)

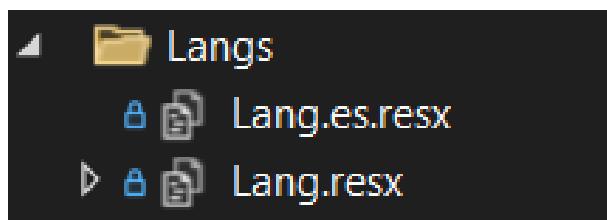
Docente: Perez Arriaga Juan Carlos

Fecha de entrega: Xalapa, Ver., 21 de Octubre de 2025

Primer reporte de Avance del proyecto final: Juego Damas Chinas

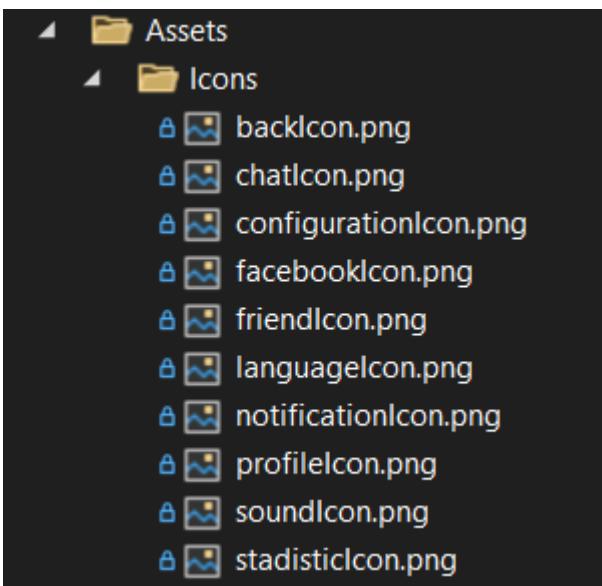
Recursos implementados

- 1. Recurso de idiomas (Lang)** Se implementó internacionalización estática para inglés y español, lo que permite mostrar textos de la interfaz en ambos idiomas.

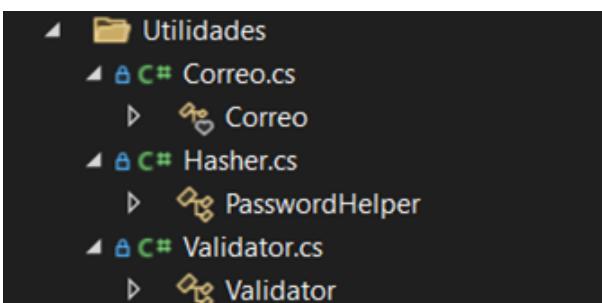


Nombre	Valor neutro	Comentario neutro	es (Español)
back	Back		Regresar
CHAT	CHAT		CHAT
ChatWindow	ChatWindow		Ventana de Chat
Chinese Checkers	Chinese Checkers		Damas Chinas
CREATE GAME	CREATE GAME		CREAR PARTIDA
databaseError	Connection error. Please try again.		Error de conexión con la base de...
email	Email		Correo Electronico
emailError	The email address must contain an...		El correo electrónico debe llevar u...
emailExists	Email is already registered.		El correo ya está registrado
emptyCredentials	Fields can't be empty		Los campos no pueden estar vacíos
english	English		Inglés
exit	Exit		Salir
Friend List	Friend List		Lista de amigos
fullName	Full Name		Nombre
hello	Hello		Hola
HOW TO PLAY	HOW TO PLAY		UNIRSE A PARTIDA
instructionLanguage	Select a language and press OK T...		Selecciona un lenguaje y presiona...
JOIN A PARTY	JOIN A PARTY		UNIRSE A PARTIDA
language	Language		Lenguaje
login	Log in		Iniciar Sesión
loginErrorMessage	Username or password is incorrect		Usuario o contraseña incorrecto
LOSES	LOSES		PERDIDAS
Main Menu	Main Menu		Menu Principal
MATCHED PLAYED	MATCHED PLAYERS		PARTIDAS JUGADAS
ok	OK		OK
password	Password		Contraseña
passwordDontMatchErrorMessage	Password dont match		Las contraseñas no coinciden
passwordMax10Error	Password must be 10 characters or...		La contraseña debe tener 10 caract...
passwordMax10Rule	Password must not exceed 10 char...		La contraseña no debe tener más...

1. **Carpetas Assets/Icons** Contiene los íconos para elementos gráficos de la interfaz, como chat, perfil, notificaciones, sonido, idioma, estadísticas, etc.



1. **Utilidades** Contiene utilidades generales que serán implementadas a lo largo del código facilitando la reutilización y estandarización.



- a. **Correo:** módulo para facilitar el envío de correos.

```

using System;
using System.Net;
using System.Net.Mail;
using System.Threading.Tasks;
using System.Windows;

namespace DCLogin.UTILIDADES
{
    Entorno
    internal static class Correo
    {
        private static readonly string remitente = "domashina94@gmail.com";
        private static readonly string contraseña = "pevd sijn tpc mnt"; // Protección pendiente de implementar previa a eso cambiar la clave

        /// <summary>
        /// Envía un correo usando Gmail
        /// </summary>
        /// <parametros>
        /// </parametros>
        public static async Task<bool> EnviarAsync(string destinatario, string asunto, string cuerpo, bool html = true)
        {
            try
            {
                using (SmtpClient smtp = new SmtpClient("smtp.gmail.com"))
                {
                    port = 587;
                    Credentials = new NetworkCredential(remitente, contraseña);
                    EnableSsl = true;
                }
                using (MailMessage mensaje = new MailMessage())
                {
                    mensaje.From = new MailAddress(remitente);
                    mensaje.To.Add(destinatario);
                    mensaje.Subject = asunto;
                    mensaje.Body = cuerpo;
                    mensaje.IsBodyHtml = html;
                }
                await smtp.SendMailAsync(mensaje);
            }
            return true;
        }
        catch (Exception ex)
        {
            MessageBox.Show("Error al enviar correo: " + ex.Message);
            return false;
        }
    }
}

```

a. Hasher.cs: encriptación de contraseñas con BCrypt.

```

using DCLogin.UTILIDADES;
namespace DCLogin.Hash
{
    2 referencias
    public static class PasswordHasher
    {
        /// <summary>
        /// Genera el hash de una contraseña usando bcrypt.
        /// </summary>
        /// <parametros>
        /// </parametros>
        public static string HashPassword(string plainPassword, int workFactor = 12)
        {
            return BCrypt.Net.BCrypt.HashPassword(plainPassword, workFactor);
        }

        /// <summary>
        /// Verifica si una contraseña coincide con su hash almacenado.
        /// </summary>
        1 referencia
        public static bool VerifyPassword(string plainPassword, string hashedPassword)
        {
            return BCrypt.Net.BCrypt.Verify(plainPassword, hashedPassword);
        }
    }
}

```

a. Validator.cs: utilidades de validación general.

i. Correo

```

 $\text{/// <summary>}$ 
 $\text{/// Valida un correo electrónico.}$ 
 $\text{/// </summary>}$ 
1 referencia
public static bool IsValidEmail(string email, out string error)
{
    error = null;
    if (string.IsNullOrWhiteSpace(email))
    {
        error = "El correo no puede estar vacío.";
        return false;
    }

    try
    {
        var pattern = @"^[\w\.-]+@[^\w\.-]+\.[^\w\.-]+$";
        if (!Regex.IsMatch(email, pattern))
        {
            error = "El correo no tiene un formato válido.";
            return false;
        }
    }
    catch (Exception)
    {
        error = "Error al validar el correo.";
        return false;
    }

    return true;
}

```

i. Password

```

/// <summary>
/// Valida contraseña (máximo 10 caracteres (segura)).
/// </summary>
2 referencias
public static bool IsValidPassword(string password, out string error)
{
    error = null;

    if (string.IsNullOrWhiteSpace(password))
    {
        error = "La contraseña no puede estar vacía.";
        return false;
    }

    if (password.Length > 10)
    {
        error = "La contraseña debe tener como máximo 10 caracteres.";
        return false;
    }

    if (!Regex.IsMatch(password, @"[A-Z]"))
    {
        error = "La contraseña debe contener al menos una letra mayúscula.";
        return false;
    }

    if (!Regex.IsMatch(password, @"\d"))
    {
        error = "La contraseña debe contener al menos un número.";
        return false;
    }

    if (!Regex.IsMatch(password, @"[\W_]+")) // \W = no alfanumérico
    {
        error = "La contraseña debe contener al menos un carácter especial.";
        return false;
    }

    return true;
}

```

i. Usuario

```

/// <summary>
/// Valida un nombre de usuario (máximo 10 caracteres, máximo 2 espacios, sin caracteres especiales).
/// </summary>
1 referencia
public static bool IsValidUsername(string username, out string error)
{
    error = null;

    if (string.IsNullOrWhiteSpace(username))
    {
        error = "El nombre de usuario no puede estar vacío.";
        return false;
    }

    if (username.Length > 10)
    {
        error = "El nombre de usuario debe tener como máximo 10 caracteres.";
        return false;
    }

    // Solo letras, números y espacios
    if (!Regex.IsMatch(username, @"^[A-Za-z0-9 ]+$"))
    {
        error = "El nombre de usuario solo puede contener letras, números y espacios.";
        return false;
    }

    // Máximo 2 espacios
    int spaceCount = username.Split(' ').Length - 1;
    if (spaceCount > 2)
    {
        error = "El nombre de usuario puede tener como máximo 2 espacios.";
        return false;
    }

    return true;
}

```

Ventanas desarrolladas

1. **Login.xaml** Funcionalidad: Permite el acceso al sistema para usuarios registrados. Características: Validación de correo y contraseña con BCrypt. Internacionalización: Inglés y español. Estado: Funcional.

Bienvenido a DAMAS

Correo Electrónico

Contraseña

Iniciar Sesión Regresar

Welcome to DAMAS!

Email

Password

Log in Back

1. **SignIn.xaml** Funcionalidad: Registro de nuevos usuarios. Características: Validación de correo y contraseñas seguras. Notificación por correo en Gmail. Internacionalización: Inglés y español. Estado: Funcional.

Regístrate en DAMAS CHINAS

Nombre de usuario

Nombre

Correo Electrónico

Contraseña
La contraseña no debe tener más de 10 caracteres.

Repite la contraseña

Registrarse Regresar

Register in DAMAS CHINAS

Username

Full Name

Email

Password
Password must not exceed 10 characters.

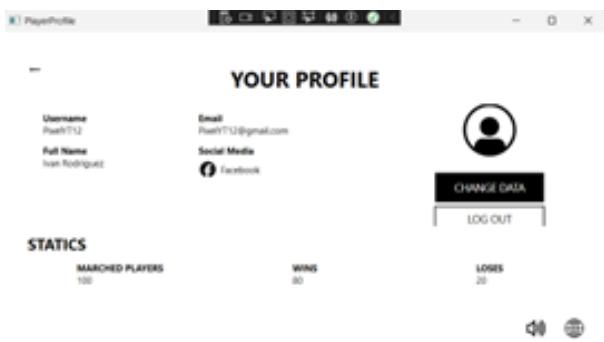
Repite Password

Sign in Back

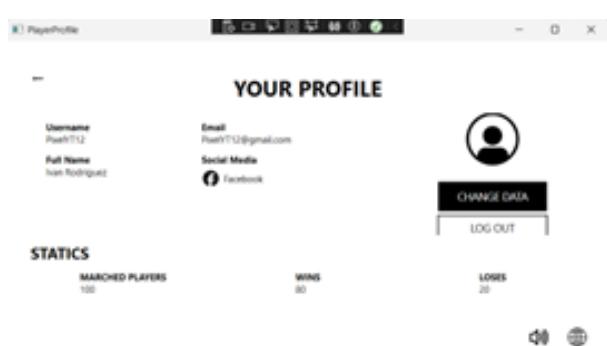
1. **MainMenuRegisteredPlayer.xaml** Menú principal para usuarios registrados (partidas, amigos, perfil, configuración, chat y estadísticas). Internacionalización: Inglés y español.



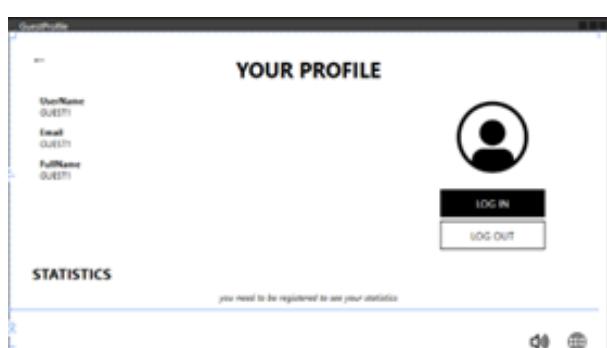
1. **MainMenuGuestPlayer.xaml** Menú principal simplificado para invitados. Acceso limitado a partidas y opciones básicas.



1. **PlayerProfile.xaml** Vista de perfil con estadísticas, logros e información del usuario.



1. GuestProfile.xaml Perfil básico para invitados (nombre temporal, avatar por defecto).



1. FriendsList.xaml Lista de amigos: agregar, eliminar, estados de conexión, mensajes. Estado: En construcción.



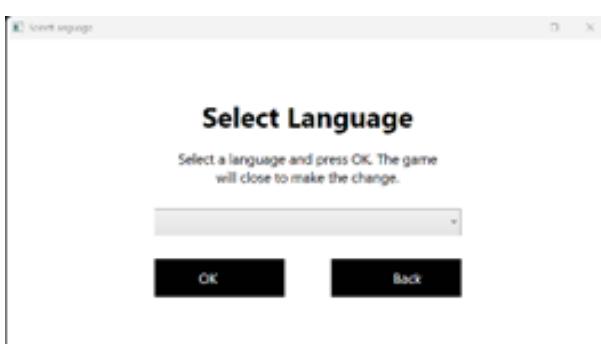


1. **ChatWindow.xaml** Ventana de chat entre jugadores. Estado: En construcción.



[Chat Inglés] | *images/Window_ChatWindow_English.png*

1. **SelectLanguage.xaml** Selección de idioma (inglés/español, carga de diccionarios Lang). Estado: Funcional.



1. **MainWindow.xaml** Ventana base del proyecto en WPF. Punto de arranque de la aplicación.



Mapeo de Base de Datos con Entity Framework

Se creó la base de datos en SQL Server Management Studio y se conectó en Visual Studio usando Entity Framework. Se comprobó la conexión correcta mediante autenticación en SQL Server.

[Entity Mapeo] | *images/xxx.png*

Resumen de contribución por integrante

Integrante 1 – Rodrigo Iván Ahumada Rodríguez - Diseño e implementación de vistas. - Creación e integración de íconos. - Configuración de la conexión a BD. - Internacionalización (50%). Contribución estimada: 50%.

Integrante 2 – Marquez Rodríguez Seth - Desarrollo de la navegabilidad entre ventanas. - Implementación de la BD en SQL Server. - Implementación de utilidades: validación, encriptación, validadores. - Internacionalización (50%). Contribución estimada: 50%.

Nota: El equipo considera que ambas contribuciones son complementarias (uno enfocado en capa visual y BD, el otro en lógica de validación y soporte multilenguaje).

Uso de Inteligencia Artificial (IA)

El equipo definió reglas para un uso responsable:

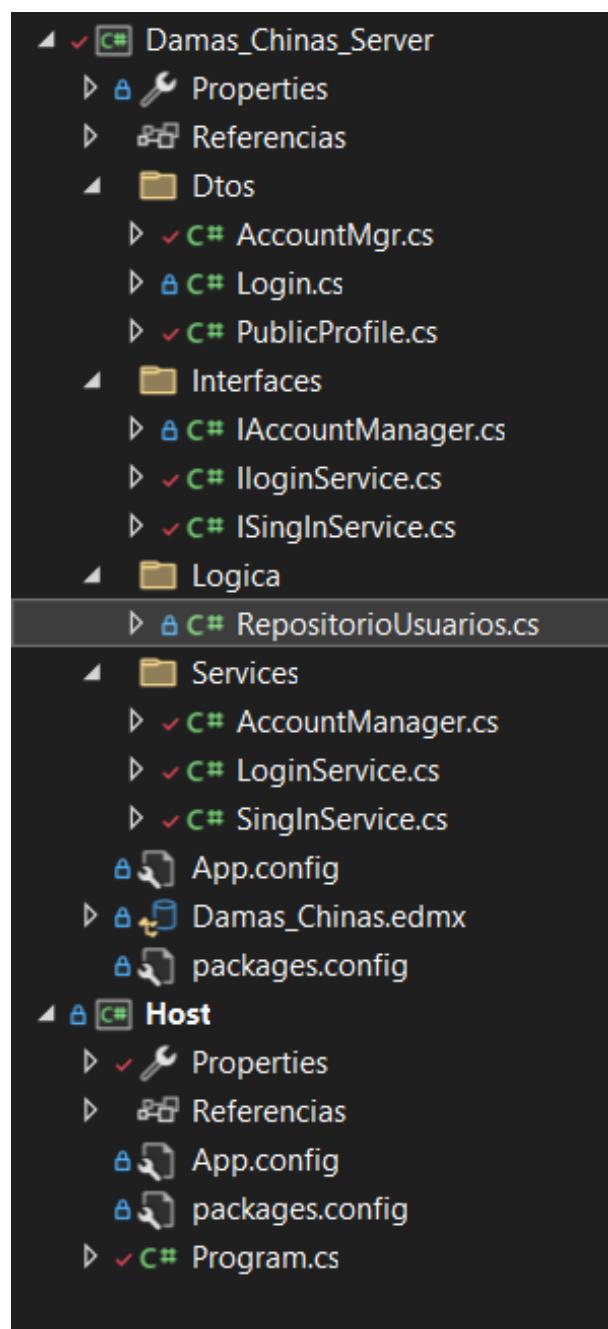
- Permitido: consultar sobre tecnologías y ventajas.
- Permitido: ejemplos generales de implementación.

- No permitido: pedir desarrollo completo de módulos.
- Permitido: compartir código propio para revisión y comentarios.
- Prohibido: usar código generado por IA que no se entienda.

Segundo reporte de Avance del proyecto final: Juego Damas Chinas

Server

```
? LoginService: http://localhost:8739/LoginService/
? SignInService: http://localhost:8736/SignInService/
? AccountManager: http://localhost:8735/AccountManager/
```



AccountManager.cs

localhost:8735/AccountManager/

Servicio de AccountManager

Creó un servicio.

Para probarlo, deberá crear un cliente y usarlo para llamar al servicio. Para ello, puede usar la herramienta svcutil.exe en la línea de comandos con la siguiente sintaxis:

```
svcutil.exe http://localhost:8735/AccountManager/?wsdl
```

También puede tener acceso a la descripción del servicio como un solo archivo:

```
http://localhost:8735/AccountManager/?singleWSDL
```

Esto generará un archivo de configuración y un archivo de código que contiene la clase de cliente. Agregue los dos archivos a la aplicación cliente y use la clase de cliente generada para llamar al servicio. Por ejemplo:

```
C#
```

DataContracts

- **PublicProfile**
- Representa la información pública de un usuario.
- Campos:
 - **Username** (string) – nombre de usuario del perfil.
 - **Nombre** (string) – nombre propio del usuario.
 - **LastName** (string) – apellido del usuario.
 - **Correo** (string) – correo del usuario.
 - **Telefono** (string) – teléfono registrado del usuario.
- **UsuarioInfo**
- Representa la información de un usuario dentro de una operación.
- Campos:
 - **IdUsuario** (int) – identificador del usuario.
 - **Username** (string) – nombre de usuario.
 - **Correo** (string) – correo del usuario.
 - **NombreCompleto** (string) – concatenación de nombre y apellido.
- **ResultadoOperacion**
- Representa el resultado de una operación de modificación de datos.
- Campos:
 - **Exito** (bool) – indica si la operación fue exitosa.
 - **Mensaje** (string) – descripción del resultado.
 - **Usuario** (UsuarioInfo) – información del usuario afectado (opcional, puede ser null).

Servicios

- **IAccountManager** (ServiceContract)
- Define las operaciones expuestas por el servicio WCF **AccountManager**.
- Operaciones (OperationContract):
 - **PublicProfile ObtenerPerfilPublico(int idUsuario)**

- **ResultadoOperacion CambiarUsername(int idUsuario, string nuevoUsername)**
- **ResultadoOperacion CambiarPassword(int idUsuario, string nuevaPassword)**

Operaciones / Funcionalidad

1. ObtenerPerfilPublico(int idUsuario)

- Función: Devuelve la información pública de un usuario. Internamente delega la obtención a **RepositorioUsuarios.ObtenerPerfilPublico**.
- Retorna: **PublicProfile (DataContract)**
- Retorna **null** si el usuario no existe.

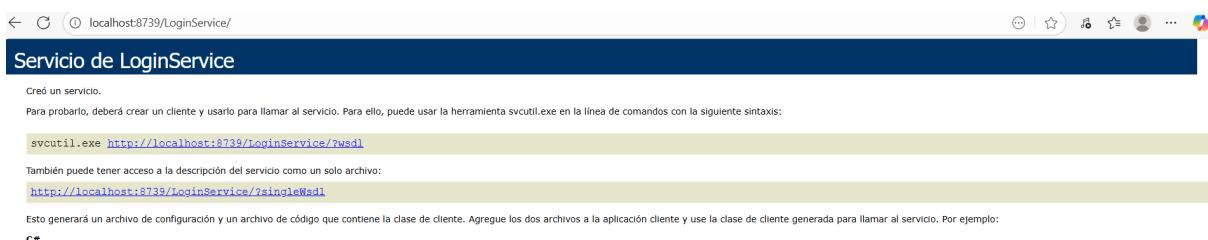
2. CambiarUsername(int idUsuario, string nuevoUsername)

- Función: Modifica el nombre de usuario de un usuario. Delegado a **RepositorioUsuarios.CambiarUsername**, que valida el username.
- Retorna: **ResultadoOperacion (DataContract)**
- **Exito, Mensaje, Usuario** (actualmente null)

3. CambiarPassword(int idUsuario, string nuevaPassword)

- Función: Modifica la contraseña de un usuario. Delegado a **RepositorioUsuarios.CambiarPassword**, que valida la contraseña.
- Retorna: **ResultadoOperacion (DataContract)**
- **Exito, Mensaje, Usuario** (actualmente null)

LoginService.cs



DataContracts

- **LoginResult**
- Representa el resultado de la validación de login de un usuario.
- Campos:
 - **IdUsuario** (int) – identificador del usuario.
 - **Username** (string) – nombre de usuario del perfil.
 - **Success** (bool) – indica si la validación fue exitosa.

Servicios

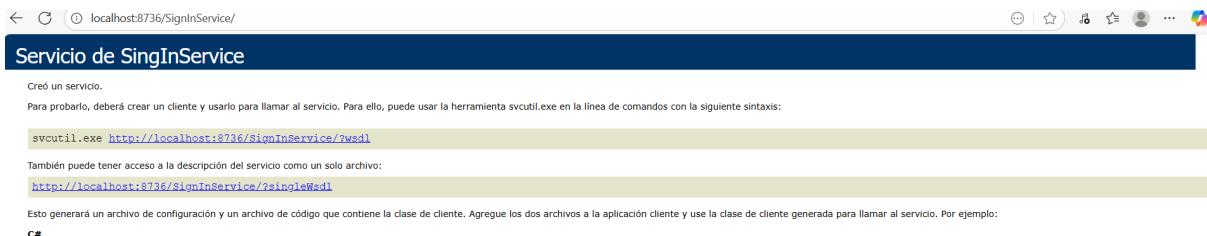
- **ILoginService** (ServiceContract)
- Define las operaciones expuestas por el servicio WCF **LoginService**.
- Operaciones (OperationContract):
- **LoginResult ValidarLogin(string usuarioInput, string password)**

Operaciones / Funcionalidad

1. **ValidarLogin(string usuarioInput, string password)**

- Función: Valida las credenciales de un usuario. Internamente delega la operación a **RepositorioUsuarios.ObtenerLoginResult**.
- Retorna: **LoginResult** (DataContract)
- **IdUsuario, Username, Success**
- Validación mínima: Se asegura de que los parámetros no estén vacíos dentro del repositorio.

SingInService.cs



DataContracts

- **UsuarioInfo**
- Representa la información de un usuario dentro de una operación.
- Campos:
 - **IdUsuario** (int) – identificador del usuario.
 - **Username** (string) – nombre de usuario.
 - **Correo** (string) – correo del usuario.
 - **NombreCompleto** (string) – concatenación de nombre y apellido.
- **ResultadoOperacion**
- Representa el resultado de una operación sobre datos de usuario.
- Campos:
 - **Exito** (bool) – indica si la operación fue exitosa.
 - **Mensaje** (string) – descripción del resultado o error.
 - **Usuario** (UsuarioInfo) – información del usuario afectado (opcional, puede ser null).

Servicios

- **ISingInService** (ServiceContract)
- Define las operaciones expuestas por el servicio WCF **SingInService**.
- Operaciones (OperationContract):
- **ResultadoOperacion CrearUsuario(string nombre, string apellido, string correo, string password, string username)**

Operaciones / Funcionalidad

1. **CrearUsuario(string nombre, string apellido, string correo, string password, string username)**
 - Función: Crea un nuevo usuario junto con su perfil asociado. Internamente delega la operación a **RepositorioUsuarios.CrearUsuario**, que realiza todas las validaciones de nombre, apellido, correo, username y contraseña.
 - Retorna: **ResultadoOperacion** (DataContract)
 - **Exito** – true si se creó correctamente.
 - **Mensaje** – mensaje de éxito o error.
 - **Usuario** – **UserInfo** con los datos del usuario creado.
 - Envío de correo de bienvenida: Opcionalmente envía un email en segundo plano tras la creación del usuario.
 - Validación: Todas las validaciones se realizan en **RepositorioUsuarios**, no en el servicio.

Logica

RepositorioUsuarios.cs

1. **CrearUsuario(string nombre, string apellido, string correo, string password, string username)**
 - Función: Crea un nuevo usuario junto con su perfil asociado en la base de datos. Antes de guardar, valida los datos usando la clase **Validator**:
 - Nombre y apellido → **Validator.ValidarNombre**
 - Correo → **Validator.ValidarCorreo**
 - Username → **Validator.ValidarUsername**
 - Contraseña → **Validator.ValidarPassword**
 - Retorna: usuarios (entidad creada con su perfil agregado).
 - Excepciones: Lanza excepción si ya existe el correo o el username, o si algún dato no cumple las reglas de validación.
2. **ObtenerLoginResult(string usuarioInput, string password)**
 - Función: Valida las credenciales de un usuario y retorna información básica para login.

Realiza validación mínima de que los parámetros no estén vacíos.

- Retorna: **LoginResult**
- **IdUsuario, Username, Success**
- Excepciones: Lanza excepción si **usuarioInput** o **password** están vacíos.

3. ObtenerPerfilPublico(int idUsuario)

- Función: Obtiene la información pública de un usuario a partir de su id.
- Retorna: **PublicProfile** (datos como username, nombre, apellido, correo y teléfono).
- Retorna **null** si el usuario no existe.

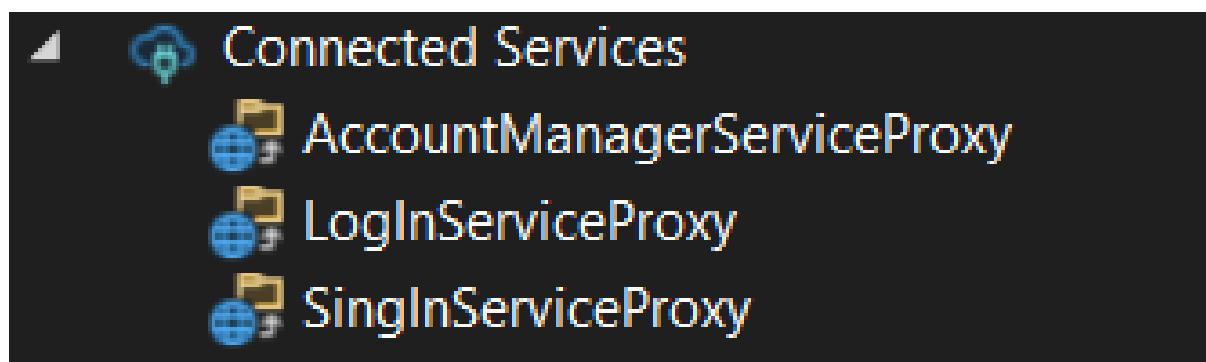
4. CambiarUsername(int idUsuario, string nuevoUsername)

- Función: Actualiza el nombre de usuario de un perfil. Antes de modificarlo, valida el username usando:
 - **Validator.ValidarUsername**
- Retorna: **bool – true** si la operación fue exitosa.
- Excepciones: Lanza excepción si el username ya existe, el perfil no se encuentra, o el username no cumple las reglas de validación.

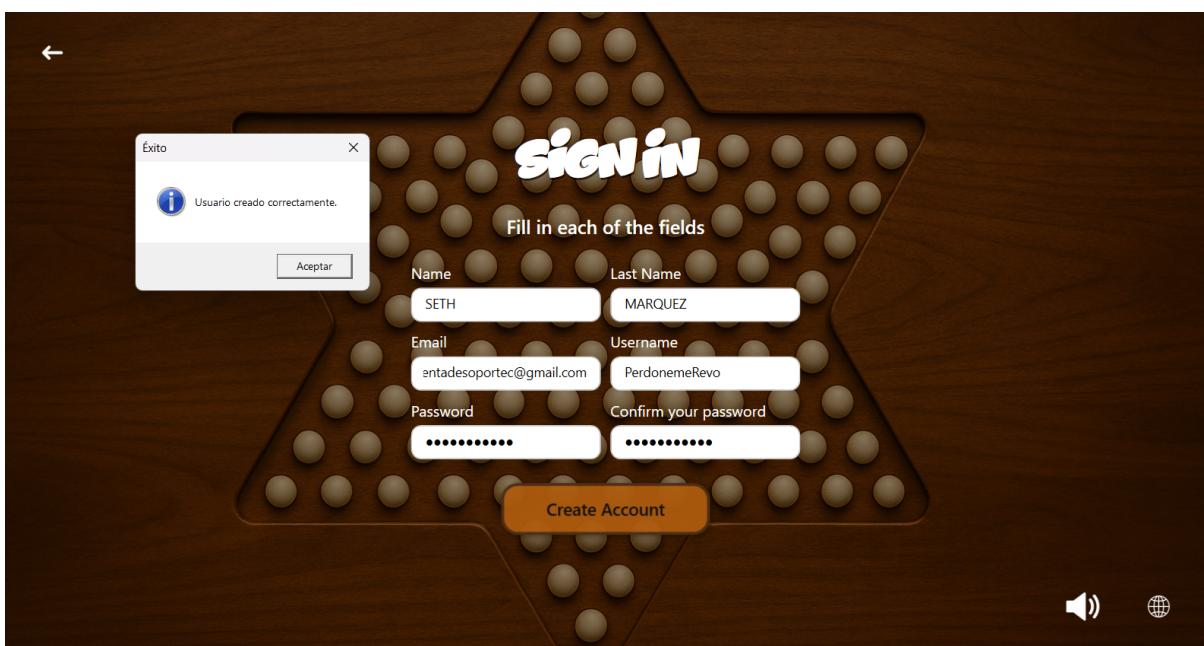
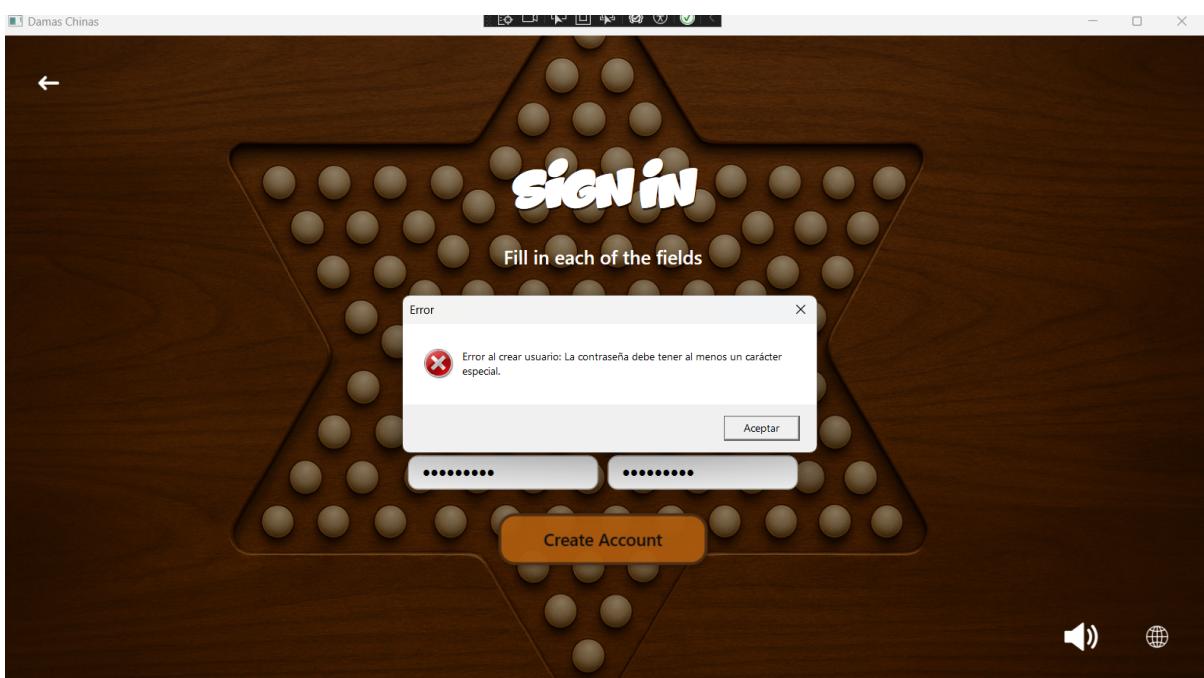
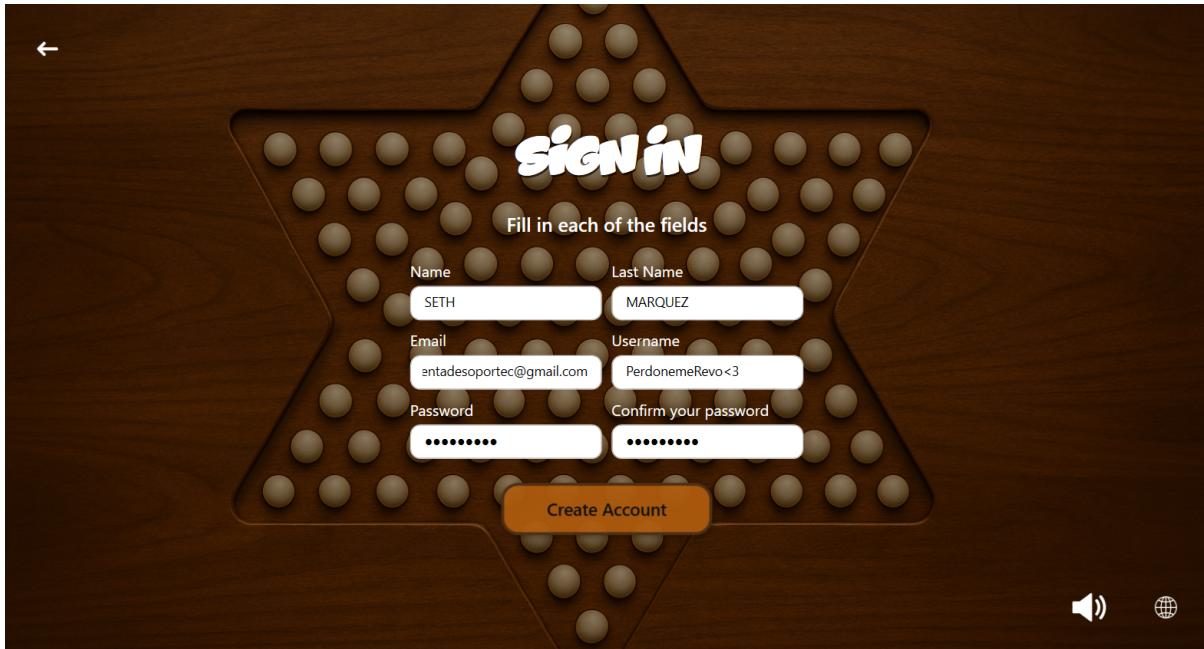
5. CambiarPassword(int idUsuario, string nuevaPassword)

- Función: Actualiza la contraseña de un usuario. Antes de modificarla, valida la contraseña usando:
 - **Validator.ValidarPassword**
- Retorna: **bool – true** si la operación fue exitosa.
- Excepciones: Lanza excepción si el usuario no existe o la contraseña no cumple las reglas de validación.

IMPLEMENTACION



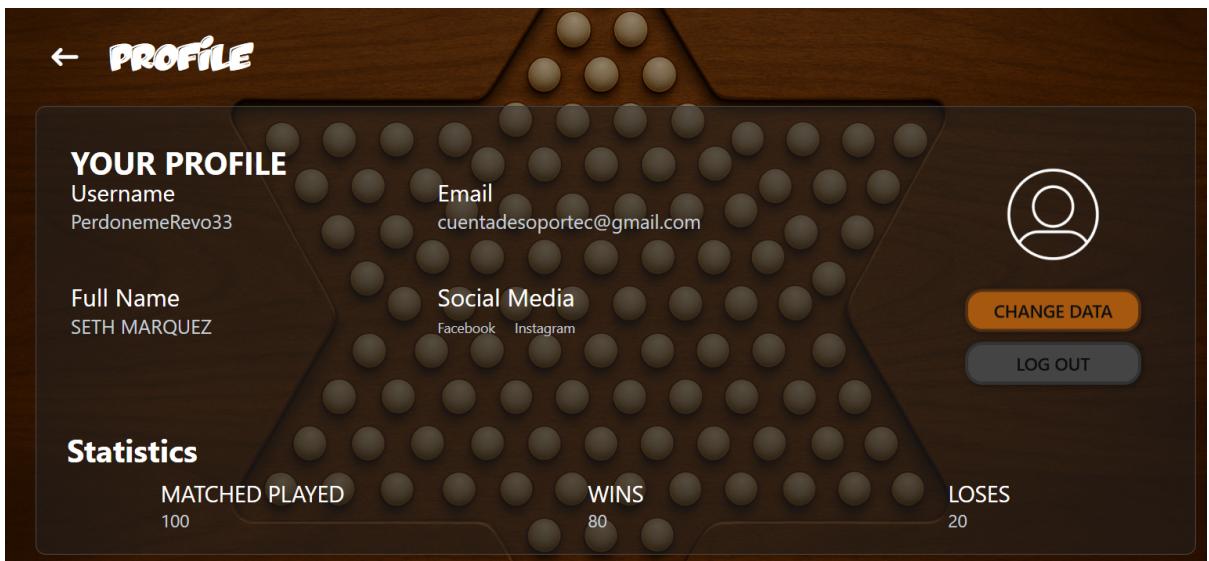
SingIn



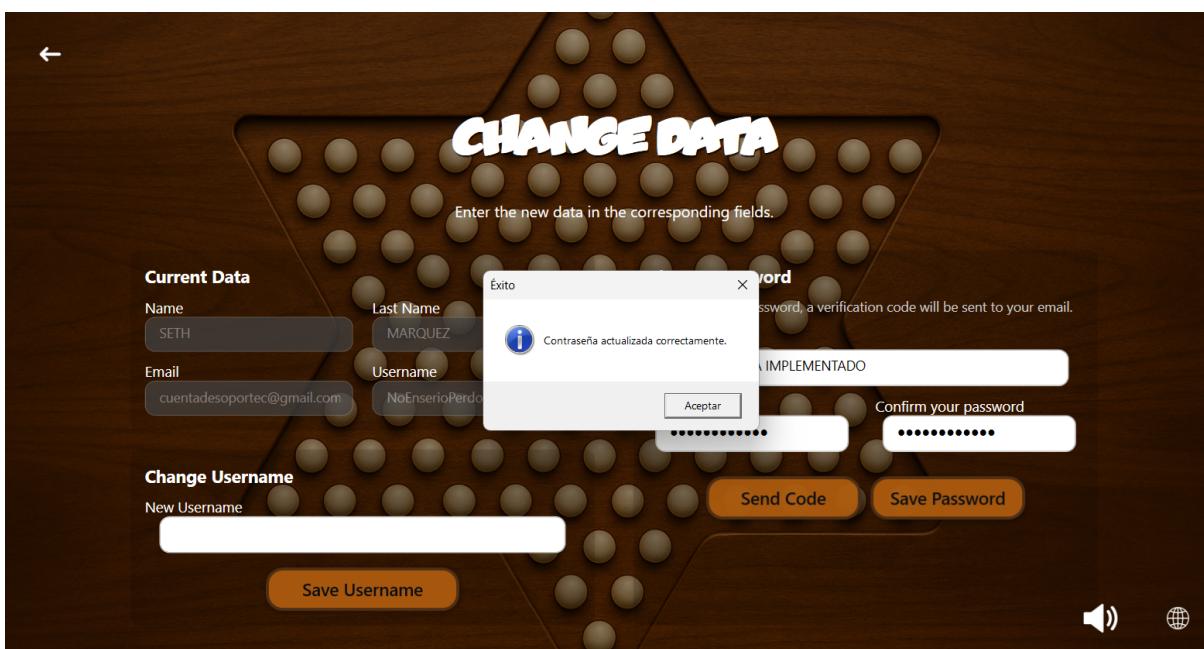
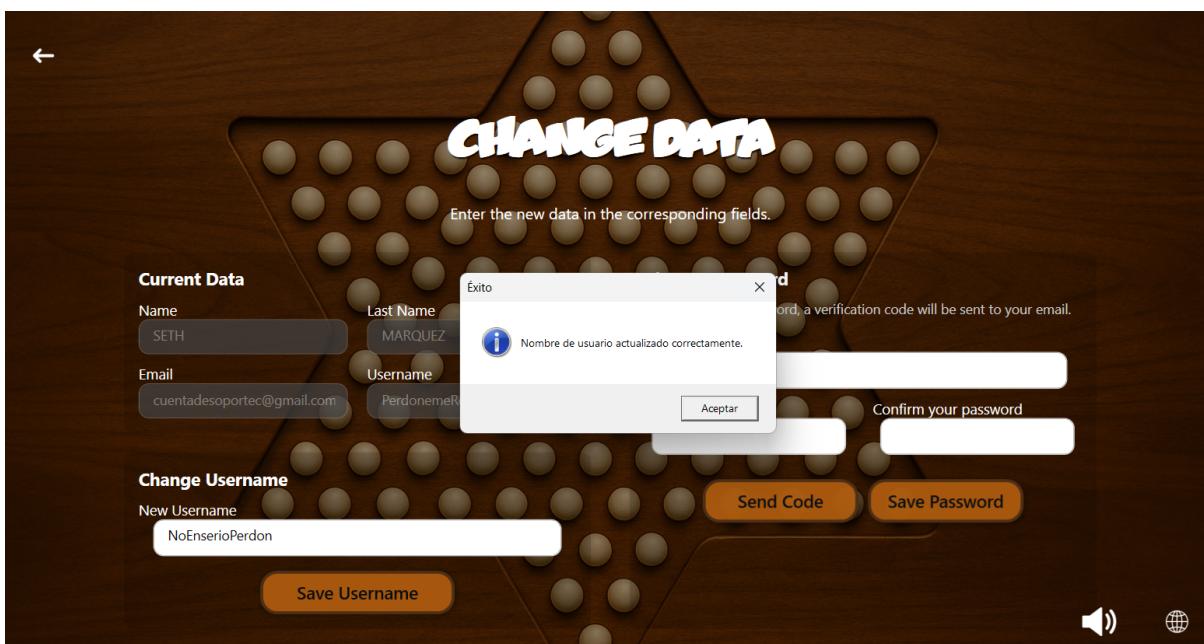
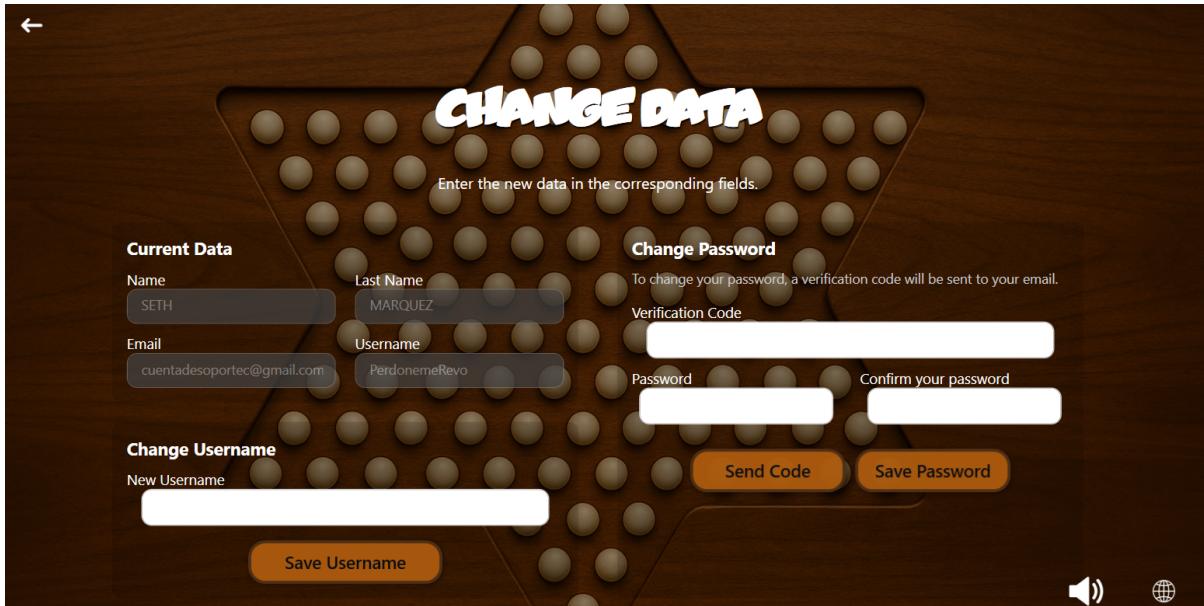
MenuRegisteredPlayer



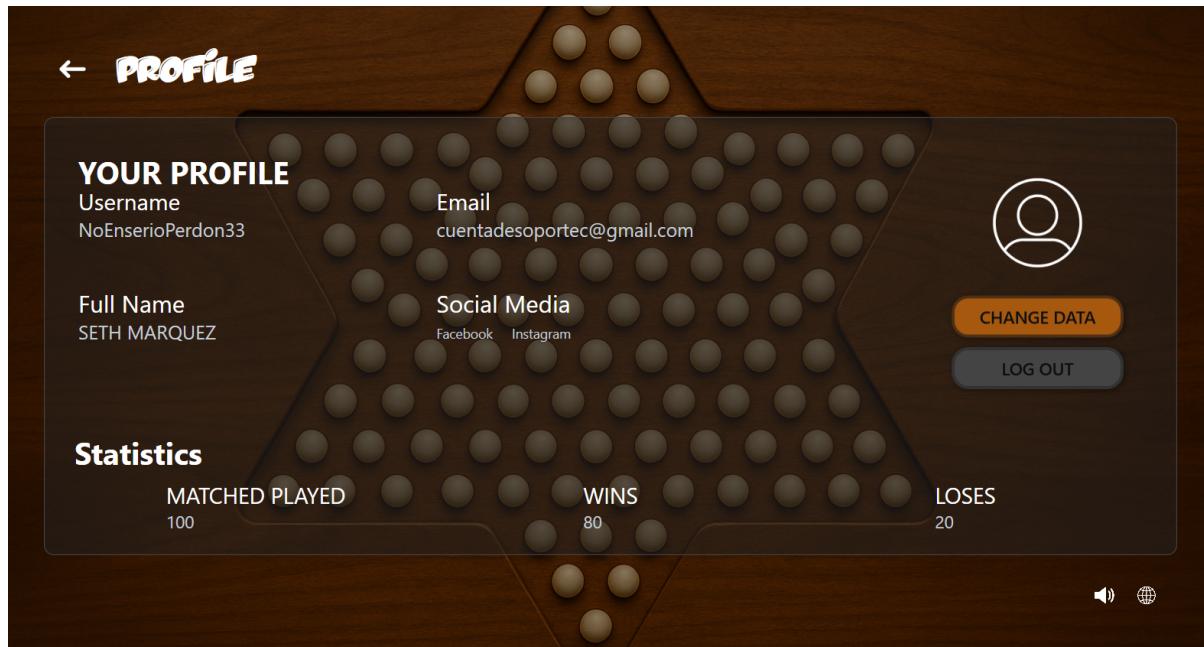
ProfilePlayer



ChangeData



Evidencia posterior al cambio



DOCUMENTACION

Elaboración de un documento de **AsciiDoc** para facilitar el control y registro de los cambios realizados con cada entrega.

```
C:\> Users > cuent > OneDrive > Desktop > Proyecto Damas Chinas > DamasChinas > ReporteAvance > ReporteAvance.ad
1 = UNIVERSIDAD VERACRUZANA
2 LTC. INGENIERÍA DE SOFTWARE
3 FACULTAD DE ESTADÍSTICA E INFORMÁTICA
4 :doctype: report
5 :toc: left
6 :toclevels: 3
7
8 == TECNOLOGÍAS PARA LA CONSTRUCCIÓN DE SOFTWARE
9
10 === ACTIVIDAD: Entrega semanal de avances
11
12 *Realizado por:*
13 Rodrigo Iván Ahumada Rodríguez (S21013886)
14 Marquez Rodriguez Seth (S23014042)
15
16 *Docente:*
17 Perez Arriaga Juan Carlos
18
19 *Fecha de entrega:*
20 Xalapa, Ver., 26 de septiembre de 2025
21
22
23 == Primer reporte de AVANCE del proyecto final: Juego Damas Chinas
24
25
26
27 == Recursos implementados
28
29 . **Recurso de idiomas (Lang)**
30 Se implementó internacionalización estática para inglés y español, lo que
permite mostrar textos de la interfaz en ambos idiomas.
31
32 image::images\Lang_Rute.png[Ruta Lang , width=300]
33
34 image::images\Lang_Resource.png[Recurso Lang , width=300]
35
36
```

UNIVERSIDAD VERACRUZANA
LIC. INGENIERÍA DE SOFTWARE – FACULTAD DE ESTADÍSTICA E INFORMÁTICA

Table of Contents

TECNOLOGÍAS PARA LA CONSTRUCCIÓN DE SOFTWARE

ACTIVIDAD: Entrega semanal de avances

Primer reporte de Avance del proyecto final: Juego Damas Chinas

Recursos implementados

Ventanas desarrolladas

Mapeo de Base de Datos con Entity Framework

Resumen de contribución por integrante

Uso de Inteligencia Artificial (IA)

Segundo reporte de Avance del proyecto final: Juego Damas Chinas

Server

AccountManager.cs

DataContracts

Servicios

Operaciones / Funcionalidad

LoginService.cs

DataContracts

Servicios

Operaciones / Funcionalidad

SingInService.cs

DataContracts

Servicios

Operaciones / Funcionalidad

Logica

RepositorioUsuarios.cs

DOCUMENTACION

Avance del Proyecto Final: Juego Damas Chinas (Entrega 2)

SETH

- Creación del server: 100%

- **Implementación de servicios en el cliente:** 100%
- **Creación de documento ASCII-DOC:** 60%
- **Conexión de Cliente con Host:** 100%

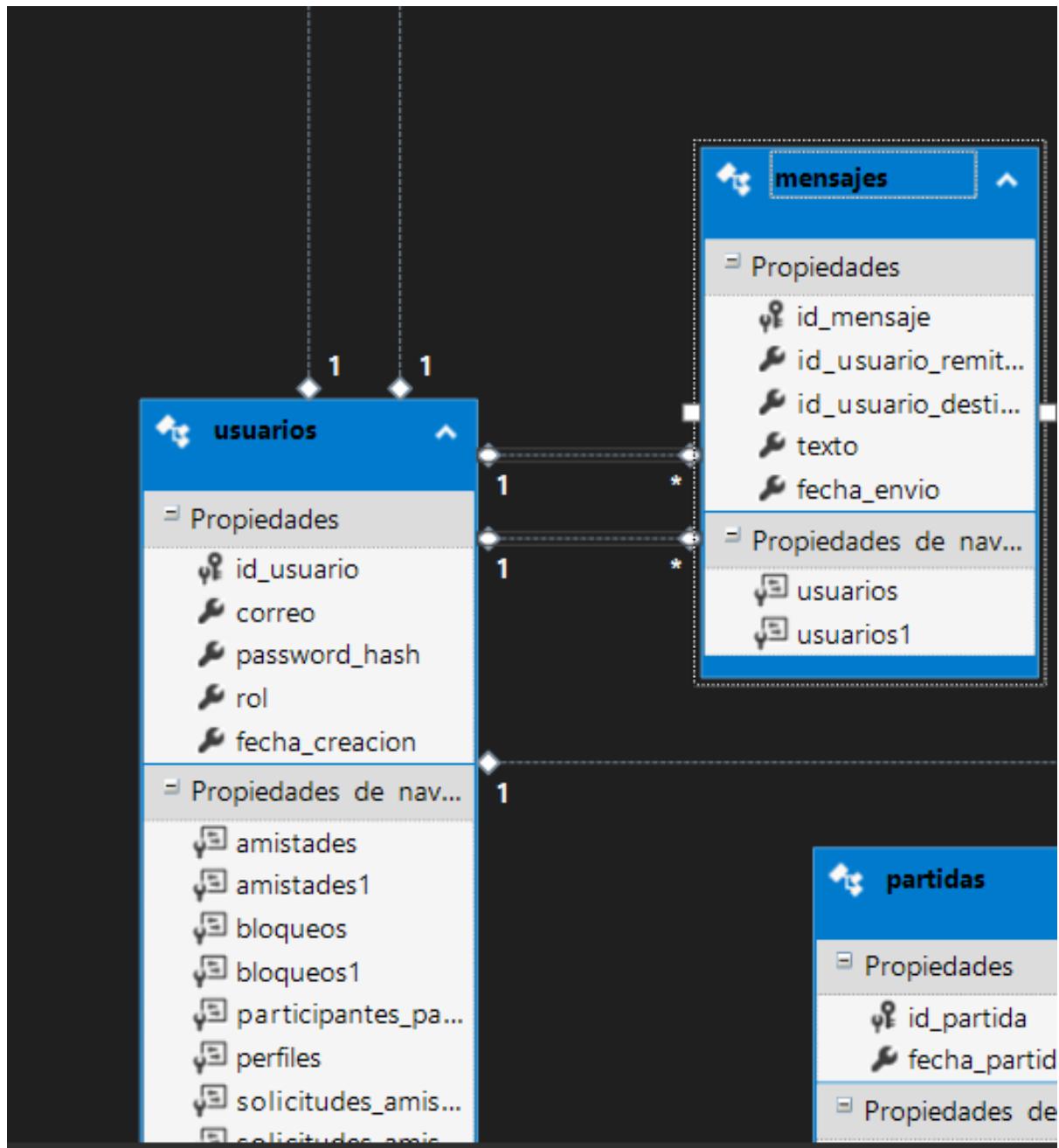
IVAN

- **GUI:** 100%

Tercer reporte de entrega

Modificaciones a la BD

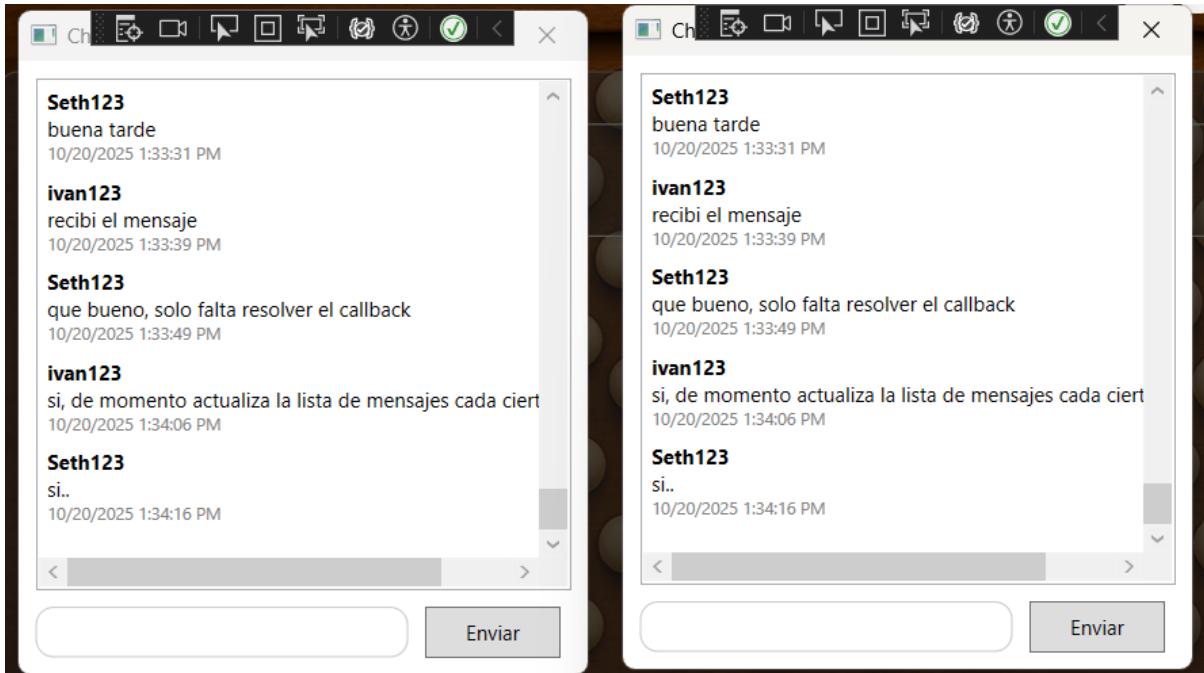
En busqueda de mantener la persistencia de los mensajes existentes en un chat de amigos se implemento la siguiente tabla en la base de datos, con la finalidad de no saturar la memoria del servidor para acceder a las platicas.



Cliente

CHAT

Funcionalidad: Guardar Mensajes entre amigos. Características: Muestra de forma dinamica la recepcion de mensajes manteniendo chats individuales entre amigos. Internacionalización: Inglés y español. Estado: En proceso, la mensajería funciona pero el callback que actualiza al llegar un nuevo mensaje aun no esta del todo funcional



Fucionalidad

```
<ListBox x:Name="MessagesList" ItemsSource="{Binding Messages}">
    <ListBox.ItemTemplate>
        <DataTemplate>
            <StackPanel Margin="0,2">
                <TextBlock Text="{Binding UsernameDestino}" FontWeight="Bold"/>
                <TextBlock Text="{Binding Texto}" TextWrapping="Wrap"/>
                <TextBlock Text="{Binding FechaEnvio}" FontSize="10" Foreground="Gray"/>
            </StackPanel>
        </DataTemplate>
    </ListBox.ItemTemplate>
</ListBox>
```

AL iniciar la ventana de chat, se recurre a la interfaz **MesageService** misma que se encarga de consultar los mensajes existentes entre 2 usuarios y agregarlos a una colección de mensajes misma que sera mostrada automaticamente por la page.

```

1 referencia
public ChatWindow(int miId, string friendUsername)
{
    InitializeComponent();

    _miIdUsuario = miId;
    _friendUsername = friendUsername;
    _myUsername = ObtenerMiUsername();

    DataContext = this;

    var callback = new MensajeriaCallback(this);
    var context = new InstanceContext(callback);

    var binding = new NetTcpBinding
    {
        Security = { Mode = SecurityMode.None },
        ReceiveTimeout = TimeSpan.MaxValue
    };

    var endpoint = new EndpointAddress("net.tcp://localhost:8755/MensajeriaService/");
    var factory = new DuplexChannelFactory<IMensajeriaService>(context, binding, endpoint);
    _client = factory.CreateChannel();

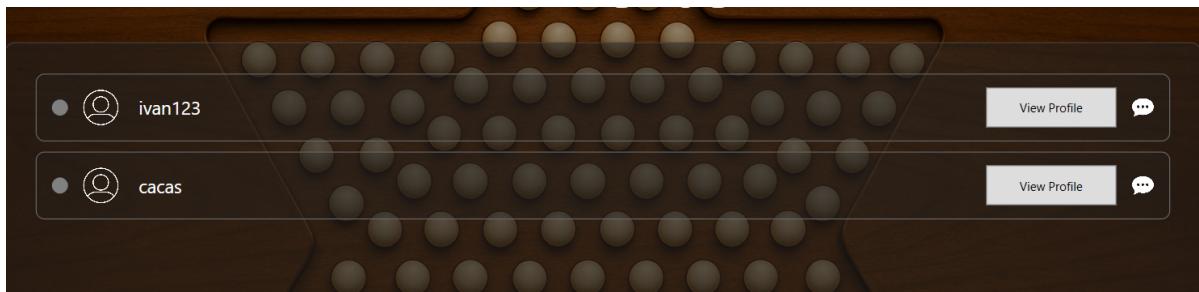
    _client.RegistrarCliente(_myUsername);

    CargarHistorial();

    _refreshTimer = new DispatcherTimer();
    _refreshTimer.Interval = TimeSpan.FromSeconds(5);
    _refreshTimer.Tick += (s, e) => CargarHistorial();
    _refreshTimer.Start();
}

```

Friends



Funcionalidad:

- Mostrar la lista de amigos
- Permite la apertura de un chat con amigos
- Nos permite consultar el perfil publico de nuestros amigos

Características: Al entrar en la ventana se recupera la lista de amigos del usuario y los despliega por medio de una lista de amigos provista por el servidor

```
namespace Damas_Chinas_Server.Dtos
{
    [DataContract]
    5 referencias
    public class AmigoDto
    {
        1 referencia
        public int IdAmigo { get; set; }

        [DataMember]
        1 referencia
        public string Username { get; set; }

        [DataMember]
        1 referencia
        public bool EnLinea { get; set; }

        [DataMember]
        1 referencia
        public string Avatar { get; set; }
    }
}
```

Y lo muestra por medio de un ItemControl Para mantener la estetica de nuestras ventanas y permiten la creacion de botones dinamicos para cada uno de los amigos.

```

<!-- ItemsControl -->
<ItemsControl x:Name="FriendsList" ItemsSource="{Binding AmigosList}" Margin="10">
    <ItemsControl.ItemTemplate>
        <DataTemplate>
            <Border Background="#44222222"
                    BorderBrush="#666"
                    BorderThickness="1"
                    CornerRadius="8"
                    Padding="10"
                    Margin="0,0,0,10">
                <Grid>
                    <Grid.ColumnDefinitions>
                        <ColumnDefinition Width="Auto"/>
                        <ColumnDefinition Width="Auto"/>
                        <ColumnDefinition Width="*"/>
                        <ColumnDefinition Width="Auto"/>
                        <ColumnDefinition Width="Auto"/>
                    </Grid.ColumnDefinitions>

                    <!-- Falta Implementar -->
                    <Ellipse Width="16" Height="16"
                            Fill="Gray"
                            VerticalAlignment="Center"
                            Margin="5,0,10,0"
                            IsEnabled="False"/>

                    <!-- Avatar -->
                    <Image Grid.Column="1"
                           Source="{Binding Avatar}"
                           Width="45" Height="45"
                           Stretch="Uniform"/>

                    <!-- Username -->
                    <TextBlock Grid.Column="2"
                               Text="{Binding Username}"
                               Foreground="White"
                               FontSize="18"
                               VerticalAlignment="Center"
                               Margin="15,0,0,0"/>

                    <!-- Ver perfil (Boton) -->
                    <Button Grid.Column="3"
                           Content="View Profile"
                           Width="130" Height="40"
                           Margin="10,0,10,0"
                           Click="OnViewProfileClick"/>

                    <!-- Chat (Boton) -->
                    <Button Grid.Column="4"
                           Background="Transparent"
                           BorderThickness="0"
                           Cursor="Hand"
                           Click="OnChatClick">
                        <Image Source="pack://application:///DamasChinas_Client.UI;component/Assets/Icons/chatIcon.png"
                               Width="30" Height="30"
                               Stretch="Uniform"/>
                    </Button>
                </Grid>
            </Border>
        </DataTemplate>
    </ItemsControl.ItemTemplate>
</ItemsControl>

```

Servidor

Evidencia Servicios Corriendo

```

? LoginService: http://localhost:8739/LoginService/
? SignInService: http://localhost:8736/SignInService/
? AccountManager: http://localhost:8735/AccountManager/
? SaludoService (NetTcp): net.tcp://localhost:8755/MensajeriaService/
? AmistadService: http://localhost:8741/AmistadService/

```

AmistadService

Funcionalidades y estado.

← C ⓘ localhost:8741/AmistadService/

Servicio de AmistadService

Creó un servicio.

Para probarlo, deberá crear un cliente y usarlo para llamar al servicio. Para ello, puede usar la herramienta svctool.exe en la línea de comandos con la siguiente sintaxis:

```
svctool.exe http://localhost:8741/AmistadService/?wsdl
```

También puede tener acceso a la descripción del servicio como un solo archivo:

```
http://localhost:8741/AmistadService/?singleWSDL
```

Esto generará un archivo de configuración y un archivo de código que contiene la clase de cliente. Agregue los dos archivos a la aplicación cliente y use la clase de cliente generada para llamar al servicio. Por ejemplo:

C#

```
class Test
{
    static void Main()
    {
        AmistadServiceClient client = new AmistadServiceClient();
        // Use la variable 'client' para llamar a operaciones en el servicio.
        // Cierre siempre el cliente.
        client.Close();
    }
}
```

Visual Basic

```
Class Test
    Shared Sub Main()
        Dim client As AmistadServiceClient = New AmistadServiceClient()
        ' Use la variable 'client' para llamar a operaciones en el servicio.
        ' Cierre siempre el cliente.
        client.Close()
    End Sub
End Class
```

Este servicio es el encargado de todo lo relacionado a la gestión de amigos.

- Mostrar Lista de Amigos (Funcional)
- Agregar amigo (pendiente de implementacion)
- Eliminar amigo (pendiente de implementacion)
- Bloquear usuario (pendiente de implementacion)

```
using Damas_Chinas_Server.Dtos;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Runtime.Serialization;
using System.ServiceModel;
using System.Text;

namespace Damas_Chinas_Server
{
    1 referencia
    public class AmistadService : IAmistadService
    {
        private readonly RepositorioAmistades repo = new RepositorioAmistades();

        1 referencia
        public List<AmigoDto> ObtenerAmigos(int idUsuario)
        {
            return repo.ObtenerAmigos(idUsuario);
        }
    }
}
```

```
using Damas_Chinas_Server.Dtos;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Runtime.Serialization;
using System.ServiceModel;
using System.Text;

namespace Damas_Chinas_Server
{
    [ServiceContract]
    2 referencias
    public interface IAmistadService
    {
        [OperationContract]
        1 referencia
        List<AmigoDto> ObtenerAmigos(int idUsuario);
    }
}
```

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

namespace Damas_Chinas_Server.Dtos
{
    [DataContract]
    5 referencias
    public class AmigoDto
    {
        1 referencia
        public int IdAmigo { get; set; }

        [DataMember]
        1 referencia
        public string Username { get; set; }

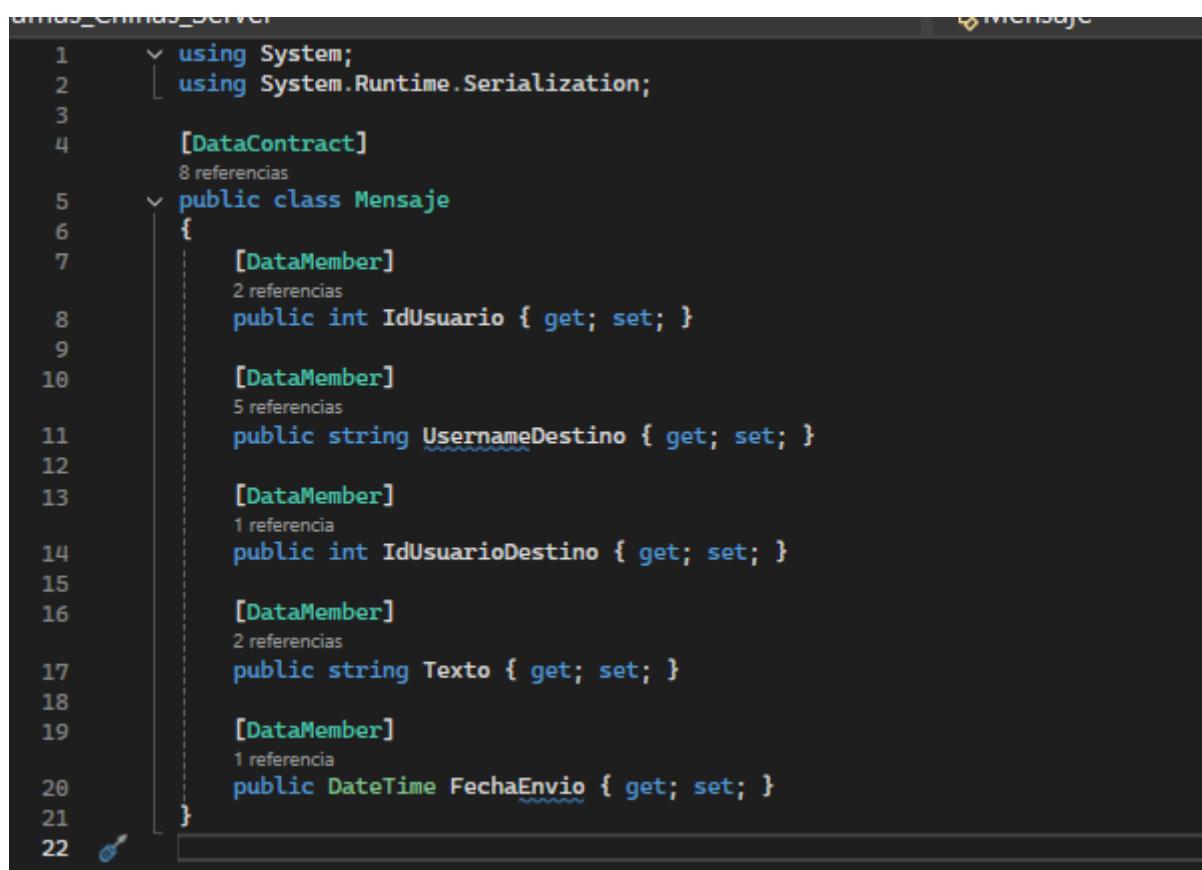
        [DataMember]
        1 referencia
        public bool EnLinea { get; set; }

        [DataMember]
        1 referencia
        public string Avatar { get; set; }
    }
}
```

MessageService

Este servicio es el encargado de todo lo relacionado a la entrega de mensajes entre amigos para el chat

DTO



The screenshot shows a code editor window with the file name 'Mensaje.cs' at the top right. The code defines a class 'Mensaje' with several properties annotated with the [DataMember] attribute:

```
1  using System;
2  using System.Runtime.Serialization;
3
4  [DataContract]
5  public class Mensaje
6  {
7      [DataMember]
8      public int IdUsuario { get; set; }
9
10     [DataMember]
11     public string UsernameDestino { get; set; }
12
13     [DataMember]
14     public int IdUsuarioDestino { get; set; }
15
16     [DataMember]
17     public string Texto { get; set; }
18
19     [DataMember]
20     public DateTime FechaEnvio { get; set; }
21 }
22
```

ServiceContract

```

Damas_Chinas_Server
1     using System;
2     using System.Runtime.Serialization;
3
4     [DataContract]
5     public class Mensaje
6     {
7         [DataMember]
8         public int IdUsuario { get; set; }
9
10        [DataMember]
11        public string UsernameDestino { get; set; }
12
13        [DataMember]
14        public int IdUsuarioDestino { get; set; }
15
16        [DataMember]
17        public string Texto { get; set; }
18
19        [DataMember]
20        public DateTime FechaEnvio { get; set; }
21    }
22

```

Service

```

using Damas_Chinas_Server;
namespace Damas_Chinas_Server
{
    [ServiceBehavior(InstanceContextMode = InstanceContextMode.PerSession)]
    public class MensajeriaService : IMensajeriaService
    {
        private static Dictionary<string, IMensajeriaCallback> _clientes = new Dictionary<string, IMensajeriaCallback>();

        private MensajesRepository _repo = new MensajesRepository();

        public void RegistrarCliente(string username)
        {
            var callback = OperationContext.Current.GetCallbackChannel<IMensajeriaCallback>();
            if (!_clientes.ContainsKey(username))
                _clientes[username] = callback;
        }

        public void EnviarMensaje(Mensaje mensaje)
        {
            int idRemitente = mensaje.IdUsuario;
            int idDestino = _repo.ObtenerIdPorUsername(mensaje.UsernameDestino);

            _repo.GuardarMensaje(idRemitente, idDestino, mensaje.Texto);

            if (_clientes.ContainsKey(mensaje.UsernameDestino))
            {
                try
                {
                    _clientes[mensaje.UsernameDestino].RecibirMensaje(mensaje);
                }
                catch
                {
                    _clientes.Remove(mensaje.UsernameDestino);
                }
            }
        }

        public Mensaje[] ObtenerHistorialMensajes(int idUsuario, string usernameDestino)
        {
            return _repo.ObtenerHistorialPorUsername(idUsuario, usernameDestino).ToArray();
        }
    }
}

```

Utilidades Implementadas

encargadas de mediar la logica entre el servidor y la implementacion de entity framework

```

    using System;
    using System.Collections.Generic;
    using System.Linq;
    using System.ServiceModel;
    using Damas_Chinas_Server.Dtos;
    using Damas_Chinas_Server;

    namespace Damas_Chinas_Server
    {
        [ServiceBehavior(InstanceContextMode = InstanceContextMode.PerSession)]
        public class MensajeriaService : IMensajeriaService
        {

            private static Dictionary<string, IMensajeriaCallback> _clientes = new Dictionary<string, IMensajeriaCallback>();

            private MensajesRepository _repo = new MensajesRepository();

            public void RegistrarCliente(string username)
            {
                var callback = OperationContext.Current.GetCallbackChannel<IMensajeriaCallback>();
                if (_clientes.ContainsKey(username))
                    _clientes[username] = callback;
            }

            public void EnviarMensaje(Mensaje mensaje)
            {
                int idRemitente = mensaje.IdUsuario;
                int idDestino = _repo.ObtenerIdPorUsername(mensaje.UsernameDestino);

                _repo.GuardarMensaje(idRemitente, idDestino, mensaje.Texto);

                if (_clientes.ContainsKey(mensaje.UsernameDestino))
                {
                    try
                    {
                        _clientes[mensaje.UsernameDestino].RecibirMensaje(mensaje);
                    }
                    catch
                    {
                        _clientes.Remove(mensaje.UsernameDestino);
                    }
                }
            }

            public Mensaje[] ObtenerHistorialMensajes(int idUsuario, string usernameDestino)
            {
                return _repo.ObtenerHistorialPorUsername(idUsuario, usernameDestino).ToArray();
            }
        }
    }

```

LOBBY

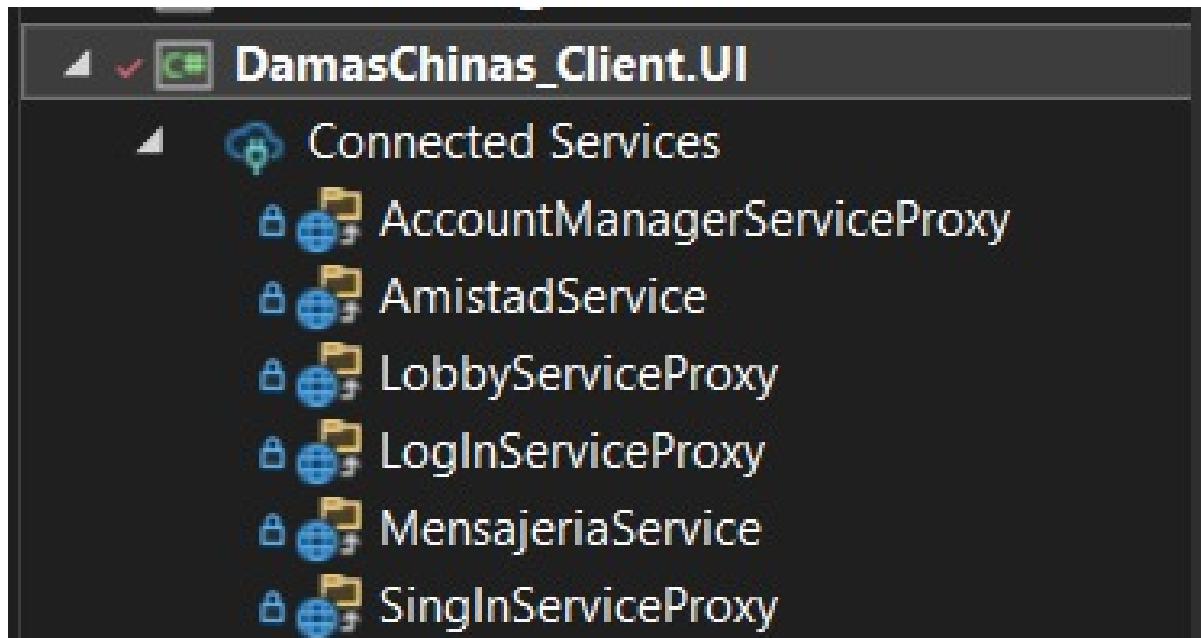
Funcionalidad: Creación, unión y gestión dinámica de lobbies multijugador.

Características: Permite la creación de lobbies privados o públicos por parte del usuario anfitrión, la unión de otros jugadores mediante un código de invitación, y la comunicación en tiempo real entre los miembros de un mismo lobby.

Internacionalización: Inglés y español. Estado: **En funcionamiento**. El cliente y servidor ya se comunican correctamente mediante WCF, el **LobbyManager** del cliente está implementado y los servicios del servidor están en ejecución estable.

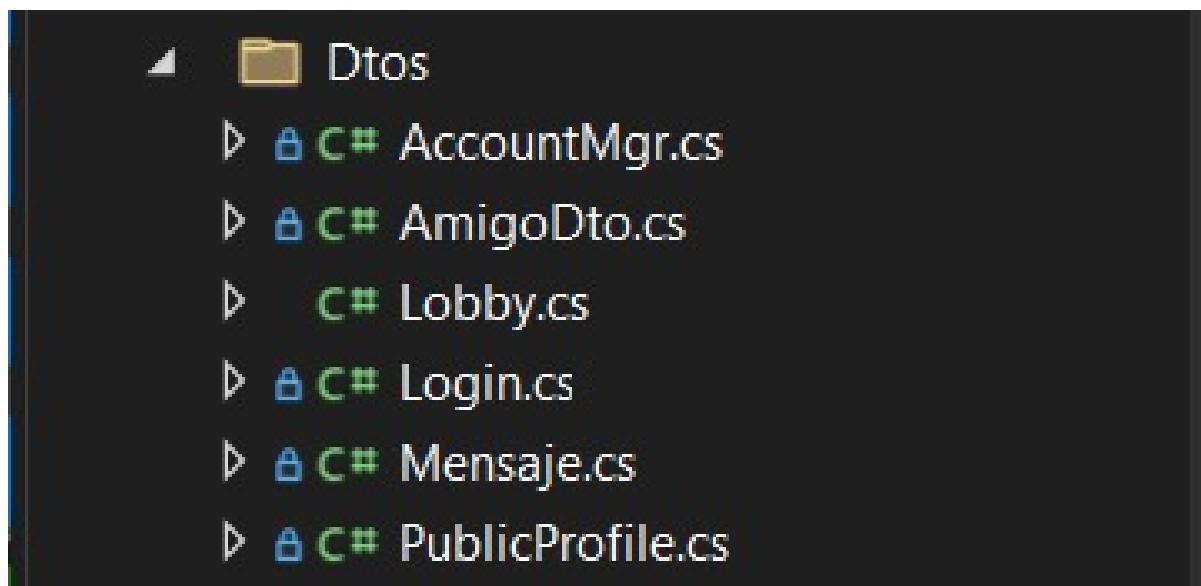
Descripción de la implementación

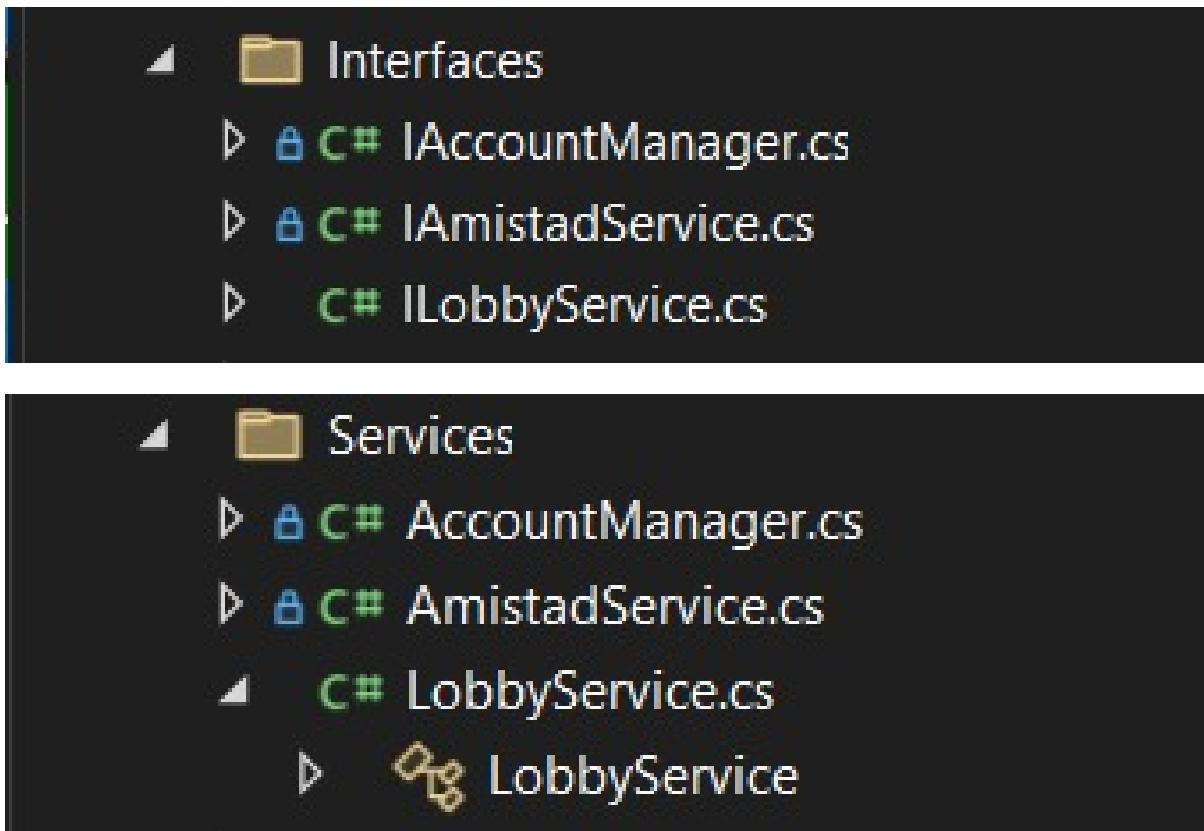
La estructura del sistema se divide en **Cliente (UI)** y **Servidor (WCF Services)**. El **Cliente** utiliza el proxy **LobbyServiceProxy** generado automáticamente desde el servicio del servidor para gestionar la comunicación, mientras que el **Servidor** expone la lógica de negocio a través del contrato **ILobbyService** y el **callback contract ILobbyCallback**.



Estructura del Servidor

- **Carpeta Dtos:** contiene las clases `Lobby`, que representan la sesión multijugador y los jugadores conectados.
- **Carpeta Interfaces:** define `ILobbyService` (contrato principal) e `ILobbyCallback` (canal de retorno que notifica eventos a los clientes).
- **Carpeta Services:** contiene `LobbyService.cs`, que implementa toda la lógica del sistema de lobbies.





ServiceContract y Callback

El servidor implementa las siguientes operaciones mediante WCF:

- `CreateLobby(hostUserId, hostUsername, isPrivate)` → crea un nuevo lobby y registra al host.
- `JoinLobby(code, userId, username)` → permite a un jugador unirse mediante el código.
- `LeaveLobby(code, userId)` → remueve al jugador del lobby.
- `SendLobbyMessage(code, userId, username, message)` → envía mensajes a todos los miembros conectados.
- `StartGame(code)` → notifica a todos los jugadores que la partida ha comenzado.

El `ILobbyCallback` define los métodos que el servidor invoca en los clientes: - `OnMemberJoined(LobbyMember member)` - `OnMemberLeft(int userId)` - `OnMessageReceived(int userId, string username, string message, string utcIso)` - `OnLobbyClosed(string reason)` - `OnGameStarted(string code)`

Estructura del Cliente

El cliente cuenta con una capa de utilidades (`Utilities`) que maneja la conexión con los servicios del servidor y los eventos recibidos por medio de callbacks.

- **LobbyManager.cs**: clase encargada de gestionar la conexión con el servicio `LobbyService`, utilizando el proxy generado por **Connected Services**.
- Implementa la interfaz `ILobbyServiceCallback` para recibir las notificaciones del servidor en

tiempo real.

- Expone eventos públicos (`MemberJoined`, `MemberLeft`, `MessageReceived`, `LobbyClosed`, `GameStarted`) a los que la UI puede suscribirse para actualizar las vistas.
-

Flujo de comunicación

1. El usuario crea o se une a un lobby desde la interfaz WPF.
 2. `LobbyManager` crea una instancia de `LobbyServiceClient` con un `InstanceContext` para manejar los callbacks.
 3. El servicio `LobbyService` en el servidor agrega el usuario al lobby y notifica al resto mediante `OnMemberJoined`.
 4. La UI del cliente actualiza la vista de jugadores conectados automáticamente mediante eventos suscritos.
-

Avance del Proyecto Final: Juego Damas Chinas (Entrega 3)

SETH

- Creacion e implementacion de el servicio de mensajes: 100%
- Creacion del servicio de amigos 100%
- Creacion del servicio de mensajes 100%
- Creación de documento ASCII-DOC: 50%
- Implemtacion de servicio amigos en cliente 100%
- *Implementacion de servicio de mensajes en cliente 100%
- Modificacion de la base de datos 50%

IVAN

- GUI: 100%
- Creacion e implementación del servicio de Lobby de juego 100%
- Creación de documento ASCII-DOC: 50%
- Implementacion de servicio de lobby en cliente 100%
- Modificacion de la base de datos 50%