Universidad Veracruzana

El Ahorcado

Proyecto Final de la Experiencia Educativa de Tecnologías para la Construcción de Software

Equipo 5
Luis Ángel Elizalde Arroyo
Daniel Mongeote Tlachy

20/04/2025

Índice

1 Planteamiento del problema

El juego de adivinanzas de palabras, conocido como Ahorcado, involucra a dos jugadores que asumen roles diferentes. El Jugador 1, denominado como “Jugador Retador”, selecciona una palabra de una categoría previamente definida y esta es representada mediante guiones bajos que indican la cantidad de letras que la componen. Por ejemplo, si la palabra elegida es "Conejo", se mostraría como "_ _ _ _ _ _". Además, el sistema proporciona una pista relacionada con la palabra misteriosa para orientar al otro jugador.

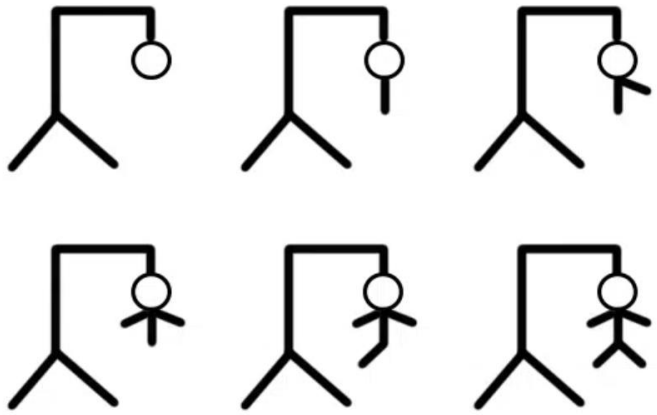


Ilustración 1.- Visualización de oportunidades para el Jugador 1

El segundo jugador, denominado como “Jugador Adivinador”, tiene como objetivo revelar la palabra secreta antes de que se complete la representación gráfica de una figura ahorcada. Para ello, el sistema muestra una colección de letras que podrían ser de utilidad para formar parte de la palabra oculta. Si la letra propuesta por el Jugador Adivinador se encuentra presente, el sistema revela todas las posiciones donde aparece dicha letra en la representación de guiones bajos. Sin embargo, si la letra sugerida no forma parte de la palabra secreta, el sistema procede a dibujar una parte de la figura del ahorcado (cabeza, cuerpo, brazos o piernas). El Jugador adivinador tiene un máximo de 6 oportunidades para proponer letras erróneas antes de perder el juego.

El juego finaliza cuando el Jugador Adivinador logra revelar la palabra secreta antes de que se complete el dibujo del ahorcado, en cuyo caso resulta victorioso, o cuando el Jugador Retador completa la representación gráfica del ahorcado antes de que el Jugador Adivinador complete la palabra misteriosa, resultando el Jugador Retador como ganador.

2 Decisiones de diseño de la solución y tecnologías a utilizar

Para el desarrollo del juego del ahorcado, se ha optado por una arquitectura Cliente-Servidor con servicios web. Las aplicaciones cliente para los jugadores 1 y 2 se implementarán como aplicaciones de escritorio WPF (Windows Presentation Foundation), aprovechando las ventajas de esta tecnología de Microsoft para construir interfaces de usuario atractivas y empleando distintos componentes de diseño.

La implementación del juego se realizará mediante una arquitectura cliente-servidor con servicios web desarrollados en WCF (Windows Communication Foundation) en C#. Esta elección permite mantener el estado y control centralizado de las sesiones de juego en el servidor, sincronizando de manera eficiente y segura el flujo de comunicación entre los clientes (aplicaciones WPF de los jugadores). WCF ofrece mecanismos robustos para el manejo de transacciones, control de flujo, autenticación, autorización y cifrado de datos, garantizando la integridad y consistencia de la información compartida entre las sesiones participantes. Además, su capacidad de interoperabilidad brinda flexibilidad para futuras expansiones o integraciones del sistema.

En cuanto al almacenamiento de datos, se utilizará SQL Server como sistema gestor de base de datos robusto y confiable para albergar toda la información necesaria del juego, como usuarios, partidas, puntuaciones, entre otros.

Además, para una gestión adecuada del código fuente y facilitar la colaboración en equipo, se empleará un sistema de control de versiones como GitHub, el cual permitirá llevar un registro detallado de los cambios realizados y fomentar las mejores prácticas de desarrollo de software.

En cuanto el manejo de la lógica del juego, se utilizará la comunicación por medio de Sockets. Los Sockets permitirán establecer una comunicación bidireccional y en tiempo real entre los clientes (aplicaciones WPF de los jugadores) y el servidor.

A continuación, se detallan los casos de uso principales donde se utilizarán los sockets:

1. **Creación de una nueva partida:**
    a. El Cliente (jugador 1) enviará una solicitud al servidor para crear una nueva partida.
    b. El Servidor asignará un identificador único a la partida y la almacenará en la base de datos.
    c. El Servidor notificará al cliente (Jugador 1) que la partida ha sido creada correctamente.
2. **Unirse a una partida existente:**
    a. El Cliente (Jugador 2) enviará una solicitud al servidor para unirse a una partida disponible.
    b. El Servidor asociará al cliente (Jugador 2) con la partida seleccionada.
    c. El Servidor notificará a ambos clientes que la partida ha comenzado.

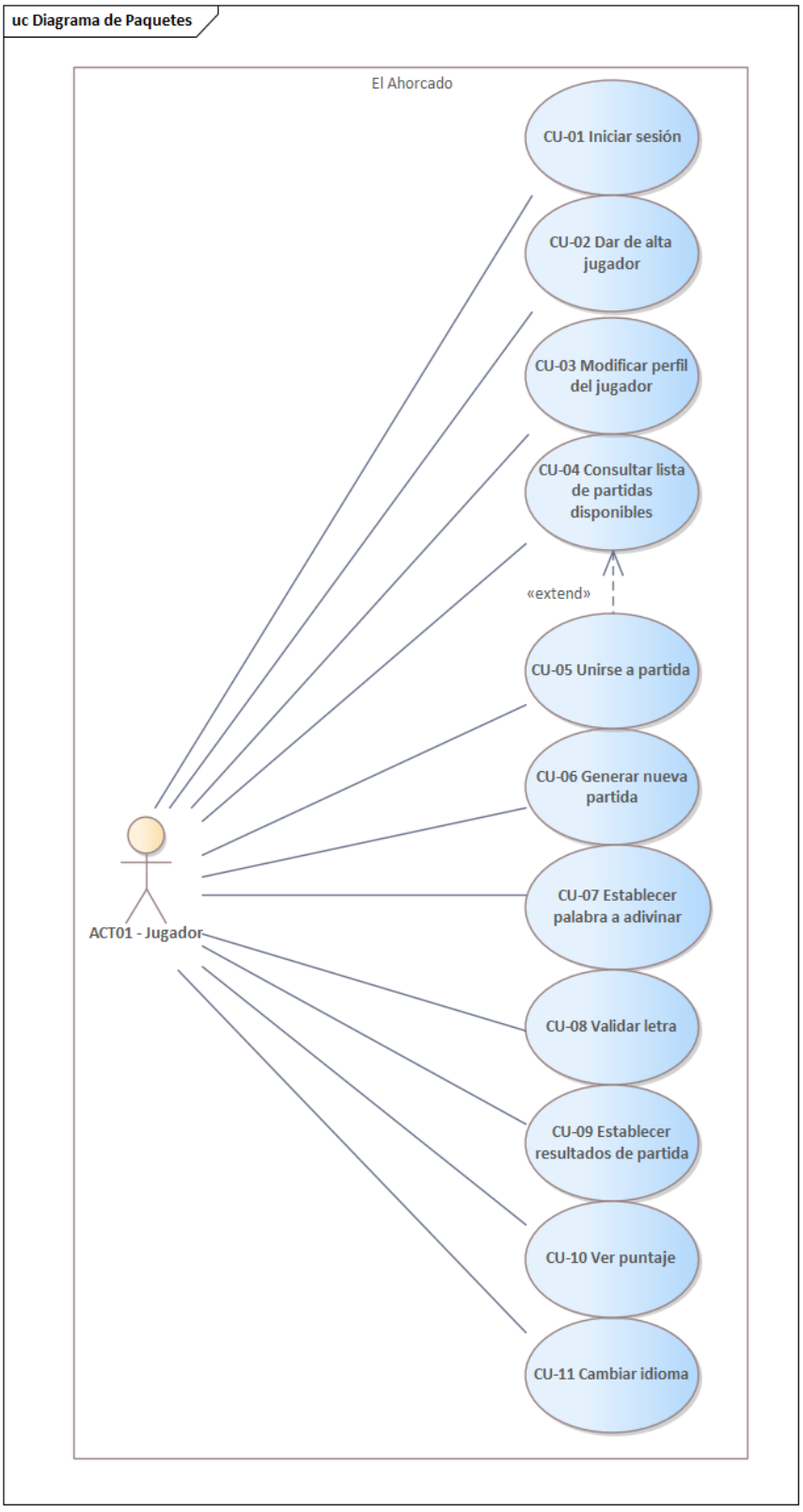
3. **Selección de una letra por parte del Jugador que adivina:**
    a. El Cliente (Jugador 2) enviará la letra seleccionada al servidor.
    b. El Servidor determinará si la letra pertenece a la palabra secreta y actualizará el estado de la partida.
    c. El Servidor notificará a ambos clientes el resultado de la selección (letra correcta o incorrecta) y actualizará las interfaces de los jugadores en consecuencia.
4. **Finalización de la partida:**
    a. Cuando un Jugador adivine la palabra o se agoten los intentos, el servidor determinará el resultado de la partida.
    b. El Servidor notificará a ambos clientes el resultado final de la partida (ganador, puntuación, palabra secreta).
    c. El Servidor almacenará el resultado de la partida en la base de datos.
5. **Desconexión de un Jugador:**
    a. Si un Jugador se desconecta durante una partida, el Servidor detectará la desconexión.
    b. El Servidor notificará al otro Jugador que la partida ha sido cancelada debido a la desconexión.
    c. El Servidor almacenará el resultado de la partida en la base de datos como "desconectada".

Además de los casos de uso principales, los Sockets también se utilizarán para otras funcionalidades secundarias, como la transmisión de información relacionada con el perfil y el puntaje del jugador.

La comunicación mediante Sockets permitirá una interacción en tiempo real y bidireccional entre los Clientes y el Servidor, lo que es esencial para el desarrollo fluido del juego del ahorcado

3 Diagrama de casos de uso

En esta sección se mostrará el diagrama de casos de uso que muestra gráficamente los casos de uso que se determinaron para el correcto funcionamiento del sistema.



4 Descripción de casos de uso

En este apartado se detallas las descripciones de los casos de uso que se diseñaron para el correcto funcionamiento del sistema.

4.1 CU – 01. Iniciar sesión

ID:	CU – 01
Nombre del CU:	Iniciar sesión
Responsable:	Luis Ángel Elizalde Arroyo
Fecha de actualización:	14/05/2025
Descripción:	El JUGADOR inicia sesión en el sistema con sus credenciales para acceder al contenido.
Actor(es):	ACT01 – Jugador
Disparador:	El Jugador ejecuta el programa de escritorio del juego.
Precondiciones:	PRE – 01. El Jugador tiene una cuenta registrada en la base de datos del sistema.
Flujo Normal:	1. El sistema despliega la ventana IniciarSesion con los siguientes campos para iniciar sesión en el sistema: correo y contraseña, junto con el botón Entrar (EX-01) (EX-02). 2. El Jugador ingresa sus credenciales y selecciona el botón Entrar (FA-01) (FA-02). 3. El sistema acepta las credenciales y despliega la alerta emergente InicioSesionExitoso con el mensaje: "¡Bienvenido(a), ¿Estás listo para empezar una partida divertida?" junto con el botón Aceptar (EX-01) (EX-02). 4. El Jugador selecciona el botón Aceptar. 5. Termina caso de uso.
Flujos Alternos:	FA – 01. El Jugador deja campos vacíos. 1. El sistema despliega la alerta emergente CamposVacios con el siguiente mensaje: "¡Error! ¡Hay campos vacíos! Complételos para continuar" y marca en rojo los campos vacíos. 2. El Jugador selecciona el botón Aceptar. 3. El sistema regresa al paso 2 del flujo normal. FA – 02. El Jugador ingresa credenciales no registradas en el sistema. 1. El sistema despliega la alerta emergente CredencialesErroneas con el siguiente mensaje: "¡Error! ¡Las credenciales ingresadas no corresponden a ningún usuario registra! Inténtelo de nuevo o hágalo más tarde" y marca en rojo los campos erróneos. 2. El Jugador selecciona el botón Aceptar. 3. El sistema regresa al paso 2 del flujo normal.
Excepciones:	EX – 01. Base de datos sin conexión. 1. El sistema despliega la alerta emergente BaseDatosSinConexion con el siguiente mensaje: "¡Error! ¡Ocurrió un error al tratar de conectarse a la base de datos! Inténtelo más tarde" junto con el botón Aceptar. 2. El Jugador selecciona el botón Aceptar. 3. Termina el caso de uso. EX – 02. Servidor del sistema sin conexión. 1. El sistema despliega la alerta emergente ServidorSinConexion con el siguiente mensaje: "¡Error! ¡Ocurrió un error al tratar de conectarse al servidor del Sistema! Inténtelo más tarde" junto con el botón Aceptar. 2. El Jugador selecciona el botón Aceptar. 3. Termina el caso de uso.
Postcondiciones:	POST – 01. Se recupera la información del JUGADOR de la base de datos correctamente.

Extiende:	No aplica.
Incluye:	No aplica.

4.2 CU – 02. Dar de alta jugador

ID:	CU – 02
Nombre del CU:	Dar de alta jugador
Responsable:	Daniel Mongeote Tlachy
Fecha de actualización:	14/05/2025
Descripción:	El Jugador crea un nuevo perfil en el juego.
Actor(es):	ACT01 – Jugador
Disparador:	El Jugador selecciona el botón Registrarse dentro de la ventana InicioSesión.
Precondiciones:	PRE – 01. No debe estar registrado un usuario previamente en el sistema con los mismos datos que planea tener dicho inmueble.
Flujo Normal:	1. El sistema despliega la ventana FormularioUsuario con los siguientes campos para crear un nuevo perfil: usuario, nombre completo, fecha de nacimiento, teléfono celular, correo electrónico y contraseña, junto con los botones Guardar y Cancelar (EX-01) (EX-02). 2. El Jugador ingresa la información del USUARIO y selecciona el botón Confirmar (FA-01) (FA-02). 3. El sistema guarda la información del perfil correspondiente al JUGADOR en la base de datos; además, despliega la alerta emergente CreacionUsuarioCorrecto con el siguiente mensaje: "¡Registro exitoso! ¿Estás listo para empezar una partida divertida?" junto con el botón Aceptar (EX-01) (EX-02). 4. El Jugador selecciona el botón Aceptar. 5. Termina el caso de uso.
Flujos Alternos:	FA – 01. El Jugador deja campos vacíos. 1. El sistema despliega la alerta emergente CamposVacios con el siguiente mensaje: "¡Error! ¡Hay campos vacíos! Complételos para continuar" y marca en rojo los campos vacíos. 2. El Jugador selecciona el botón Aceptar. 3. El sistema regresa al paso 2 del flujo normal. FA – 02. El Jugador ingresa un correo electrónico previamente registrado en el sistema. 1. El sistema despliega la alerta emergente CorreoElectronicoErroneo con el siguiente mensaje: "¡Error! ¡El Correo Electrónico ingresado ya está registrado en el sistema! Ingrese uno diferente para continuar o intentélo más tarde" y marca en rojo el campo erróneo. 2. El Jugador selecciona el botón Aceptar. 3. El sistema regresa al paso 2 del flujo normal.
Excepciones:	EX – 01. Base de datos sin conexión. 1. El sistema despliega la alerta emergente BaseDatosSinConexion con el siguiente mensaje: "¡Error! ¡Ocurrió un error al tratar de conectarse a la base de datos! Inténtelo más tarde" junto con el botón Aceptar. 2. El Jugador selecciona el botón Aceptar. 3. Termina el caso de uso. EX – 02. Servidor del sistema sin conexión. 1. El sistema despliega la alerta emergente ServidorSinConexion con el siguiente mensaje: "¡Error! ¡Ocurrió un error al tratar de conectarse al servidor del Sistema! Inténtelo más tarde" junto con el botón Aceptar.

	2. El Jugador selecciona el botón Aceptar. 3. Termina el caso de uso.
Postcondiciones:	POST – 01. Se guarda la información del JUGADOR en la base de datos.
Extiende:	No aplica.
Incluye:	No aplica.

4.3 CU – 03. Modificar perfil del jugador

ID:	CU – 03
Nombre del CU:	Modificar perfil del jugador
Responsable:	Daniel Mongeote Tlachy
Fecha de actualización:	14/05/2025
Descripción:	El JUGADOR modifica su información de perfil registrada dentro de la base de datos del sistema.
Actor(es):	ACT01 – Jugador
Disparador:	El Jugador selecciona el botón Modificar Perfil dentro del Menú Principal.
Precondiciones:	PRE – 01. Debe estar registrado por lo menos un JUGADOR en el sistema, independientemente de su rol.
Flujo Normal:	1. El sistema recupera de la base de datos la información registrada del JUGADOR actual y despliega la ventana FormularioUsuario con los siguientes campos registrados con información y que permiten ser modificados (a excepción del correo electrónico): usuario, nombre completo, fecha de nacimiento, teléfono celular, correo electrónico y contraseña junto con los botones Guardar y Eliminar (EX-01) (EX-02). 2. El Jugador edita la información de los campos y selecciona el botón Confirmar (FA-01). 3. El sistema actualiza la información del perfil del JUGADOR en la base de datos; además, despliega la ventana ModificacionUsuarioExitosa siguiente mensaje: “¡Se han modificado los datos del Usuario correctamente!” junto con el botón Aceptar (EX-01) (EX-02). 4. El Jugador selecciona el botón Aceptar. 5. Termina el caso de uso.
Flujos Alternos:	FA – 01. El Jugador deja campos vacíos. 1. El sistema despliega la alerta emergente CamposVacios con el siguiente mensaje: “¡Error! ¡Hay campos vacíos! Complételos para continuar” y marca en rojo los campos vacíos. 2. El Jugador selecciona el botón Aceptar. 3. El sistema regresa al paso 2 del flujo normal.
Excepciones:	EX – 01. Base de datos sin conexión. 1. El sistema despliega la alerta emergente BaseDatosSinConexion con el siguiente mensaje: “¡Error! ¡Ocurrió un error al tratar de conectarse a la base de datos! Inténtelo más tarde” junto con el botón Aceptar. 2. El Jugador selecciona el botón Aceptar. 3. Termina el caso de uso. EX – 02. Servidor del sistema sin conexión. 1. El sistema despliega la alerta emergente ServidorSinConexion con el siguiente mensaje: “¡Error! ¡Ocurrió un error al tratar de conectarse al servidor del Sistema! Inténtelo más tarde” junto con el botón Aceptar. 2. El Jugador selecciona el botón Aceptar. 3. Termina el caso de uso.

Postcondiciones:	POST – 01. Se modifica la información del JUGADOR en la base de datos
Extiende:	No aplica.
Incluye:	No aplica.

4.4 CU – 04. Consultar lista de partidas disponibles

ID:	CU – 04
Nombre del CU:	Consultar lista de partida disponibles
Responsable:	Luis Ángel Elizalde Arroyo
Fecha de actualización:	14/05/2025
Descripción:	El Jugador visualiza una lista de partidas disponibles que otros Jugadores han creado y que aún no cuentan con un jugador con el rol de Jugador Adivinador.
Actor(es):	ACT01 – Jugador
Disparador:	El Jugador inicia sesión en el sistema correctamente.
Precondiciones:	PRE – 01. Existe por lo menos una partida disponible en el sistema sin un JUGADOR con el rol de Adivinador. PRE – 02. Debe estar registrado por lo menos un JUGADOR en el sistema, independientemente de su rol.
Flujo Normal:	1. El sistema recupera una lista de PARTIDA(s) asociadas a un Jugador con el rol de Retador y los despliega en la ventana ListaPartidasDisponibles donde se muestra los siguientes datos: Usuario, correo del Jugador Retador, fecha de creación y el botón Unirse, junto a los botones Generar partida, Estadísticas y Modificar perfil (FA-01) (EX-01) (EX-02). 2. Termina caso de uso.
Flujos Alternos:	FA – 01. El Jugador selecciona la opción de Unirse a una partida existente. 1. Ir al CU-04.
Excepciones:	EX – 01. Base de datos sin conexión. 1. El sistema despliega la alerta emergente BaseDatosSinConexion con el siguiente mensaje: “¡Error! ¡Ocurrió un error al tratar de conectarse a la base de datos! Inténtelo más tarde” junto con el botón Aceptar. 2. El Jugador selecciona el botón Aceptar. 3. Termina el caso de uso. EX – 02. Servidor del sistema sin conexión. 1. El sistema despliega la alerta emergente ServidorSinConexion con el siguiente mensaje: “¡Error! ¡Ocurrió un error al tratar de conectarse al servidor del Sistema! Inténtelo más tarde” junto con el botón Aceptar. 2. El Jugador selecciona el botón Aceptar. 3. Termina el caso de uso.
Postcondiciones:	POST – 01. Se recupera correctamente la información de la(s) PARTIDA(s).
Extiende:	No aplica.
Incluye:	CU – 05. Unirse a partida

4.5 CU – 05. Unirse a partida

ID:	CU – 05
Nombre del CU:	Unirse a partida extendido de Consultar lista de partidas disponibles
Responsable:	Luis Ángel Elizalde Arroyo
Fecha de actualización:	14/05/2025

Descripción:	El Jugador se une a una partida existente desde la lista de partidas disponibles.
Actor(es):	ACT01 – Jugador
Disparador:	El Jugador selecciona el botón Unirse del Menú Principal.
Precondiciones:	PRE – 01. Debe existir por lo menos una partida disponible en el sistema sin un JUGADOR con el rol de Adivinador. PRE – 02. Debe estar registrado por lo menos un JUGADOR en el sistema, independientemente de su rol.
Flujo Normal:	1. El sistema muestra la ventana SalaDeEspera en donde se muestra el mensaje: "El código de la partida creada es: X X X X X X", así como el botón Cancelar partida (EX-01) (EX-02). 2. El Jugador con el rol de Adivinador espera a que el Jugador con el rol de Retador comience manualmente la PARTIDA (FA-01) (FA-02). 3. El sistema inicia la PARTIDA entre ambos jugadores (EX-01) (EX-02). 3. Termina el caso de uso.
Flujos Alternos:	FA – 01. El Jugador con el rol de Adivinador selecciona el botón Cancelar partida. 1. El sistema despliega la alerta emergente PartidaCancelada con el siguiente mensaje: "La partida fue cancelada por uno de los jugadores." 2. El Jugador selecciona el botón Aceptar. 3. Termina el caso de uso. FA – 02. El Jugador con el rol de Retador selecciona el botón Cancelar partida. 1. El sistema despliega la alerta emergente PartidaCancelada con el siguiente mensaje: "La partida fue cancelada por uno de los jugadores." 2. El Jugador selecciona el botón Aceptar. 1. Termina el caso de uso.
Excepciones:	EX – 01. Base de datos sin conexión. 1. El sistema despliega la alerta emergente BaseDatosSinConexion con el siguiente mensaje: "¡Error! ¡Ocurrió un error al tratar de conectarse a la base de datos! Inténtelo más tarde" junto con el botón Aceptar. 2. El Jugador selecciona el botón Aceptar. 3. Termina el caso de uso. EX – 02. Servidor del sistema sin conexión. 1. El sistema despliega la alerta emergente ServidorSinConexion con el siguiente mensaje: "¡Error! ¡Ocurrió un error al tratar de conectarse al servidor del Sistema! Inténtelo más tarde" junto con el botón Aceptar. 2. El Jugador selecciona el botón Aceptar. 3. Termina el caso de uso.
Postcondiciones:	POST – 01. Se inicia la PARTIDA entre ambos JUGADOR(es).
Extiende:	No aplica.
Incluye:	No aplica.

4.6 CU – 06. Generar nueva partida

ID:	CU – 06
Nombre del CU:	Generar nueva partida
Responsable:	Luis Ángel Elizalde Arroyo
Fecha de actualización:	14/05/2025
Descripción:	El Jugador crea una nueva partida en la cual fungirá con el rol de Retador para que un Jugador entre a la partida con el rol de Adivinador para tratar de adivinar la palabra del ahorcado.
Actor(es):	ACT01 – Jugador

Disparador:	El Jugador selecciona el botón Generar partida del Menú Principal.
Precondiciones:	PRE – 01. Debe estar registrado por lo menos un JUGADOR en el sistema, independientemente de su rol.
Flujo Normal:	1. El sistema despliega la ventana FormularioPartida en donde se muestran los botones de las distintas categorías de palabras: “Musica”, “Series TV” y “Peliculas”, así como una table que muestra cada una de las PALABRA(s) asociadas a la CATEGORIA para ser seleccionada como PALABRA a adivinar (EX-01) (EX-02). 2. El Jugador selecciona la CATEGORÍA y la PALABRA (FA-01). 3. El sistema guarda la información de la PARTIDA ingresada en la base de datos, genera un código de acceso único y muestra la ventana ListaEspera con el código de la partida (EX-01) (EX-02). 4. El Jugador selecciona el botón Aceptar. 4. Termina caso de uso.
Flujos Alternos:	FA – 01. El Jugador cancela la creación de la PARTIDA. 1. El sistema muestra la ventana “PartidaCancelada” con el mensaje: “La partida fue cancelada por uno de los jugadores.” 2. El Jugador selecciona el botón Aceptar. 2. Termina el caso de uso.
Excepciones:	EX – 01. Base de datos sin conexión. 1. El sistema despliega la alerta emergente BaseDatosSinConexion con el siguiente mensaje: “¡Error! ¡Ocurrió un error al tratar de conectarse a la base de datos! Inténtelo más tarde” junto con el botón Aceptar. 2. El Jugador selecciona el botón Aceptar. 3. Termina el caso de uso. EX – 02. Servidor del sistema sin conexión. 1. El sistema despliega la alerta emergente ServidorSinConexion con el siguiente mensaje: “¡Error! ¡Ocurrió un error al tratar de conectarse al servidor del Sistema! Inténtelo más tarde” junto con el botón Aceptar. 2. El Jugador selecciona el botón Aceptar. 3. Termina el caso de uso.
Postcondiciones:	POST – 01. Se guarda la información de la PARTIDA en la base de datos.
Extiende:	No aplica.
Incluye:	No aplica.

4.7 CU – 07. Establecer palabra a adivinar

ID:	CU – 07
Nombre del CU:	Adivinar partida
Responsable:	Luis Ángel Elizalde Arroyo
Fecha de actualización:	14/05/2025
Descripción:	El Sistema determina la cantidad de espacios “-“ dependiendo de la palabra que seleccionó el Jugador durante el proceso de creación de la partida.
Actor(es):	ACT01 – Jugador
Disparador:	El Jugador selecciona la palabra dentro de la ventana FormularioPartida.
Precondiciones:	PRE – 01. Debe existir por lo menos una partida disponible en el sistema sin un Jugador con el rol de Adivinador. PRE – 02. Debe estar registrado por lo menos un Jugador en el sistema, independientemente de su rol.

Flujo Normal:	1. El sistema recupera la palabra seleccionada por el Jugador dependiendo del idioma del juego (definido por él mismo) y coloca una cantidad de guiones bajos "_" acorde a la longitud de la palabra (EX-01) (EX-02). 2. Termina caso de uso.
Flujos Alternos:	No aplica
Excepciones:	EX – 01. Base de datos sin conexión. 1. El sistema despliega la alerta emergente BaseDatosSinConexion con el siguiente mensaje: "¡Error! ¡Ocurrió un error al tratar de conectarse a la base de datos! Inténtelo más tarde" junto con el botón Aceptar. 2. El Jugador selecciona el botón Aceptar. 3. Termina el caso de uso. EX – 02. Servidor del sistema sin conexión. 1. El sistema despliega la alerta emergente ServidorSinConexion con el siguiente mensaje: "¡Error! ¡Ocurrió un error al tratar de conectarse al servidor del Sistema! Inténtelo más tarde" junto con el botón Aceptar. 2. El Jugador selecciona el botón Aceptar. 3. Termina el caso de uso.
Postcondiciones:	POST – 01. Se muestra la ventana Partida con los guiones "-" acorde a la longitud de la palabra.
Extiende:	No aplica.
Incluye:	No aplica.

4.8 CU – 08. Validar letra

ID:	CU – 08
Nombre del CU:	Validar letra
Responsable:	Daniel Mongeote Tlachy
Fecha de actualización:	14/05/2025
Descripción:	El Jugador con el rol de Adivinador selecciona una letra del teclado del juego y el sistema valida si la palabra a adivinar contiene dicha letra.
Actor(es):	ACT01 – Jugador
Disparador:	El Jugador selecciona el botón de la letra de su preferencia dentro de la ventana Partida.
Precondiciones:	PRE – 01. Debe existir por lo menos una partida disponible en el sistema sin un Jugador con el rol de Adivinador. PRE – 02. Debe estar registrado por lo menos un Jugador en el sistema, independientemente de su rol.
Flujo Normal:	1. El Jugador con rol de Adivinador selecciona uno de los botones que contiene una letra (EX-02) 2. El sistema valida la letra seleccionada por el JUGADOR con rol de Adivinador (FA-01) (FA-02) (EX-02). 3. Termina caso de uso.
Flujos Alternos:	FA – 01. El carácter ingresado está dentro de la palabra. 1. El sistema pone el carácter de color verde y lo escribe en los espacios correspondientes de la palabra. FA – 02. El carácter ingresado no está dentro de la palabra. 1. El sistema pone el carácter de color rojo, disminuye en 1 el número restante de intentos para adivinar la palabra (de un total de 5) y dibuja una parte del ahorcado dependiendo de la cantidad de intentos usados.
Excepciones:	EX – 01. Base de datos sin conexión.

	1. El sistema despliega la alerta emergente BaseDatosSinConexion con el siguiente mensaje: "¡Error! ¡Ocurrió un error al tratar de conectarse a la base de datos! Inténtelo más tarde" junto con el botón Aceptar. 2. El Jugador selecciona el botón Aceptar. 3. Termina el caso de uso. EX – 02. Servidor del sistema sin conexión. 1. El sistema despliega la alerta emergente ServidorSinConexion con el siguiente mensaje: "¡Error! ¡Ocurrió un error al tratar de conectarse al servidor del Sistema! Inténtelo más tarde" junto con el botón Aceptar. 2. El Jugador selecciona el botón Aceptar. 3. Termina el caso de uso.
Postcondiciones:	POST – 01. El sistema actualiza la ventana Partida dependiendo el resultado de validación.
Extiende:	No aplica.
Incluye:	No aplica.

4.9 CU – 09. Establecer resultados de partida

ID:	CU – 09
Nombre del CU:	Establecer resultados de partida
Responsable:	Daniel Mongeote Tlachy
Fecha de actualización:	14/05/2025
Descripción:	El Sistema determina los resultados finales de la Partida dependiendo si el Jugador con rol de Adivinador logró adivinar o no la Palabra.
Actor(es):	ACT01 – Jugador
Disparador:	El Jugador con rol de Adivinador adivina la Palabra o se queda sin más intentos para adivinarla.
Precondiciones:	PRE – 01. Debe existir por lo menos una partida disponible en el sistema sin un Jugador con el rol de Adivinador. PRE – 02. Debe estar registrado por lo menos un Jugador en el sistema, independientemente de su rol.
Flujo Normal:	1. El Jugador completa la palabra correcta (FA-01) (EX-02). 2. El sistema finaliza la partida, establece el estado de la partida como "ganada", suma una puntuación global de 10 puntos al perfil del Jugador con rol de Adivinador y muestra la ventana "MensajeGanador" con el mensaje: ¡Felicidades! ¡Has adivinado la palabra correctamente! junto con el botón Aceptar (EX-01) (EX-02). 3. El Jugador selecciona el botón Aceptar. 4. Termina caso de uso.
Flujos Alternos:	FA – 01. El Jugador con el rol de Adivinador consume su cantidad total de intentos disponibles. 1. El sistema establece el estado de la partida como "perdida" y muestra la alerta emergente MensajePerdedor con el mensaje: ¡Oh no! ¡Has sido ahorcado! La palabra era [nombre de la palabra], y junto con el botón Aceptar. 2. El Jugador selecciona el botón Aceptar. 3. Termina caso de uso.
Excepciones:	EX – 01. Base de datos sin conexión. 1. El sistema despliega la alerta emergente BaseDatosSinConexion con el siguiente mensaje: "¡Error! ¡Ocurrió un error al tratar de conectarse a la base de datos! Inténtelo más tarde" junto con el botón Aceptar.

	2. El Jugador selecciona el botón Aceptar. 3. Termina el caso de uso. EX – 02. Servidor del sistema sin conexión. 1. El sistema despliega la alerta emergente ServidorSinConexion con el siguiente mensaje: "¡Error! ¡Ocurrió un error al tratar de conectarse al servidor del Sistema! Inténtelo más tarde" junto con el botón Aceptar. 2. El Jugador selecciona el botón Aceptar. 3. Termina el caso de uso.
Postcondiciones:	POST – 01. Se finaliza el estado de la PARTIDA en la base de datos y se guardan los resultados obtenidos de ambos JUGADOR(es).
Extiende:	No aplica.
Incluye:	No aplica.

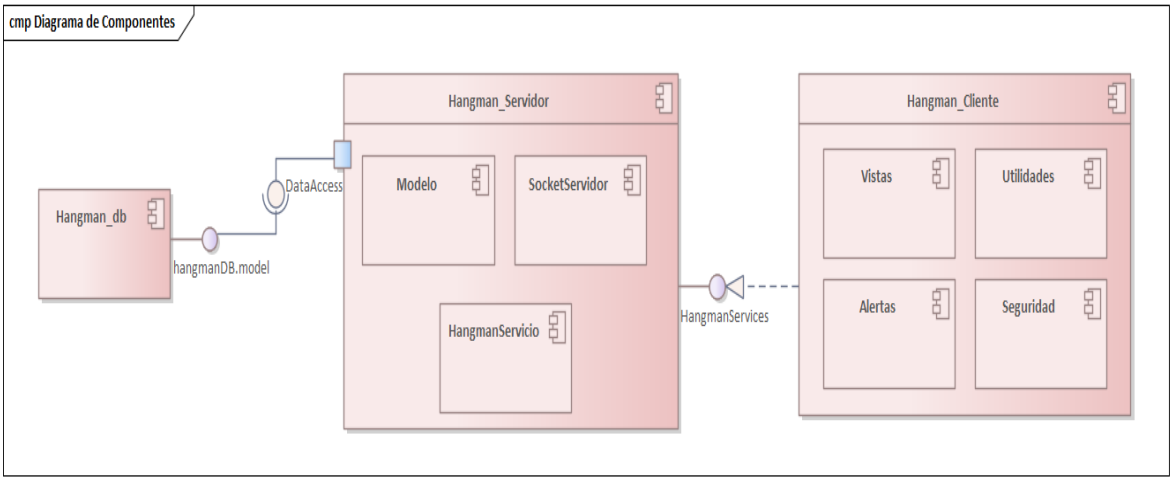
4.10 CU – 10. Ver puntaje

ID:	CU – 10
Nombre del CU:	Ver puntaje
Responsable:	Luis Ángel Elizalde Arroyo
Fecha de actualización:	14/05/2025
Descripción:	El Jugador visualiza su historial de partidas en las que participó.
Actor(es):	ACT01 – Jugador
Disparador:	El Jugador selecciona el botón Ver puntajes del Menú Principal.
Precondiciones:	PRE – 01. Debe existir por lo menos una partida finalizada relacionada al Jugador. PRE – 02. Debe estar registrado por lo menos un Jugador en el sistema, independientemente de su rol.
Flujo Normal:	1. El sistema recupera una lista de PARTIDA(s) y muestra la ventana Estadisticas con la lista recuperada donde se muestra los datos: Usuario, resultado (ganador, vencedor), fecha de juego y el puntaje acumulado total (FA-01). 2. Termina el caso de uso.
Flujos Alternos:	FA – 01. El Jugador regresa al Menú Principal. 1. Termina el caso de uso.
Excepciones:	EX – 01. Base de datos sin conexión. 1. El sistema despliega la alerta emergente BaseDatosSinConexion con el siguiente mensaje: "¡Error! ¡Ocurrió un error al tratar de conectarse a la base de datos! Inténtelo más tarde" junto con el botón Aceptar. 2. El Jugador selecciona el botón Aceptar. 3. Termina el caso de uso. EX – 02. Servidor del sistema sin conexión. 1. El sistema despliega la alerta emergente ServidorSinConexion con el siguiente mensaje: "¡Error! ¡Ocurrió un error al tratar de conectarse al servidor del Sistema! Inténtelo más tarde" junto con el botón Aceptar. 2. El Jugador selecciona el botón Aceptar. 3. Termina el caso de uso.
Postcondiciones:	POST – 01. Se recupera la información de las PARTIDA(s) en las que participó el JUGADOR correctamente.
Extiende:	No aplica.
Incluye:	No aplica.

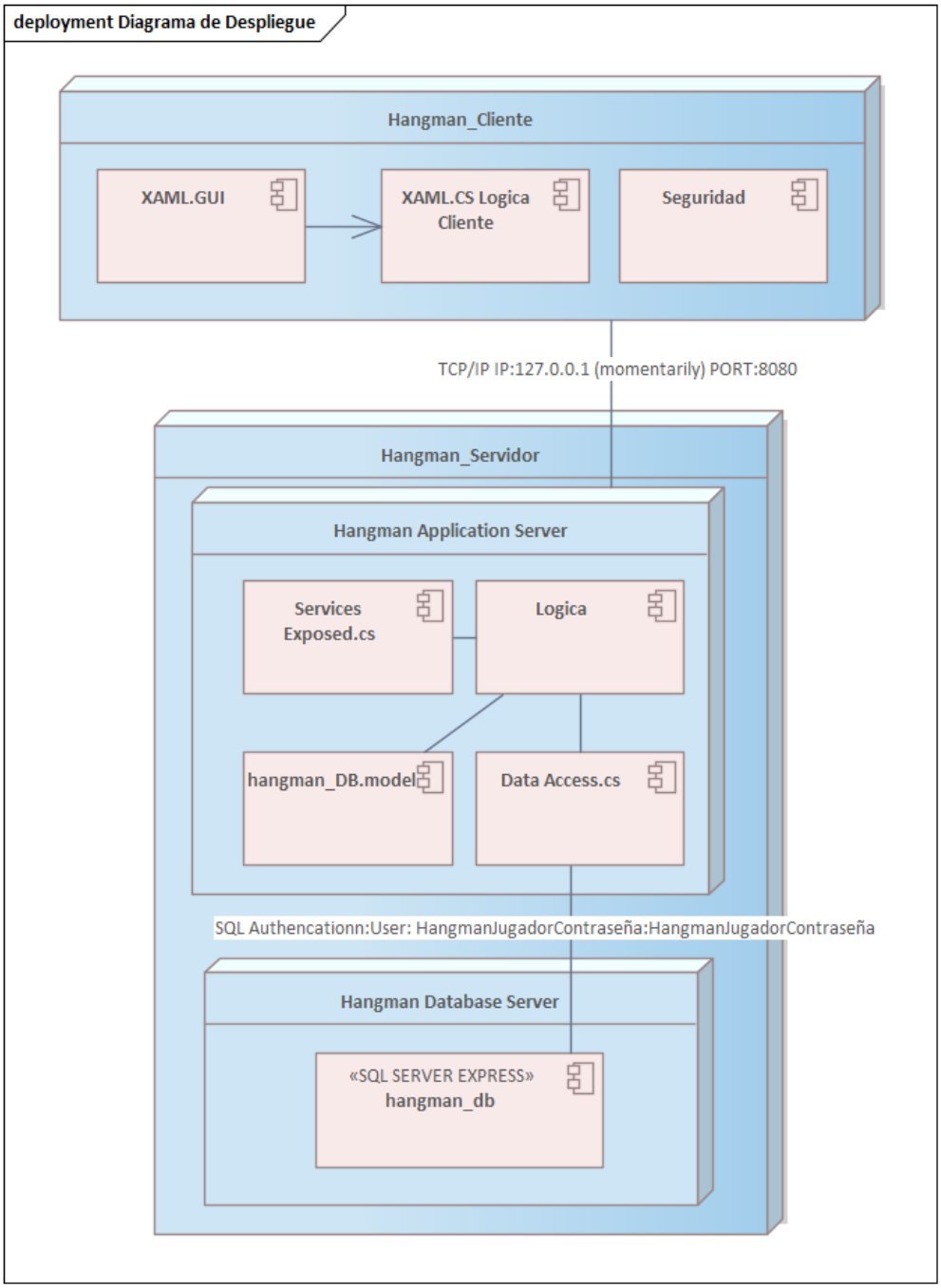
4.11 CU – 11. Cambiar idioma

ID:	CU – 11
Nombre del CU:	Cambiar idioma
Responsable:	Daniel Mongeote Tlachy
Fecha de actualización:	14/05/2025
Descripción:	El Jugador cambia el idioma del juego de Español a Inglés o viceversa.
Actor(es):	ACT01 – Jugador
Disparador:	El Jugador selecciona el botón Cambiar idioma de la ventana IniciarSesion.
Precondiciones:	PRE – 01. Debe estar registrado por lo menos un Jugador en el sistema, independientemente de su rol.
Flujo Normal:	1. El sistema muestra los nombres correspondientes al idioma Español e Inglés. 2. El Jugador selecciona el idioma de su preferencia (FA-01). 3. El sistema cambia la internacionalización de todo el juego al idioma seleccionado. 4. Termina caso de uso.
Flujos Alternos:	No Aplica.
Excepciones:	EX – 01. Base de datos sin conexión. 1. El sistema despliega la alerta emergente BaseDatosSinConexion con el siguiente mensaje: “¡Error! ¡Ocurrió un error al tratar de conectarse a la base de datos! Inténtelo más tarde” junto con el botón Aceptar. 2. El Jugador selecciona el botón Aceptar. 3. Termina el caso de uso. EX – 02. Servidor del sistema sin conexión. 1. El sistema despliega la alerta emergente ServidorSinConexion con el siguiente mensaje: “¡Error! ¡Ocurrió un error al tratar de conectarse al servidor del Sistema! Inténtelo más tarde” junto con el botón Aceptar. 2. El Jugador selecciona el botón Aceptar. 3. Termina el caso de uso.
Postcondiciones:	POST – 01. Se cambia correctamente la internacionalización del sistema al idioma seleccionado.
Extiende:	No aplica.
Incluye:	No aplica.

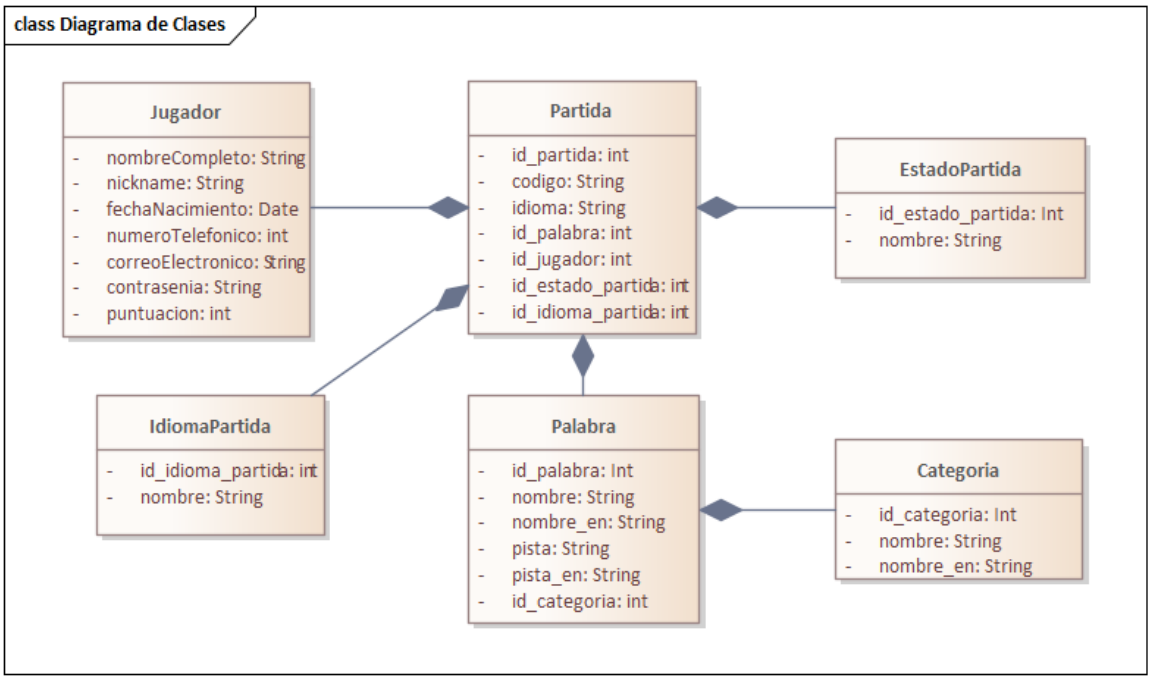
5 Diagrama de Componentes



6 Diagrama de Despliegue



7 Diagrama de Clases



8 Prototipos

8.1 CU – 01. Iniciar sesión

8.2 CU – 02. Dar de alta perfil de usuario

8.3 CU – 03. Modificar perfil de usuario

8.4 CU – 04. Consultar lista de partidas disponibles

8.5 CU – 05. Unirse a partida

8.6 CU – 06. Generar nueva partida


Lista de espera
El código de la partida es:
rbci3415
Cancelar


PartidaCancelada
La partida fue cancelada por uno de los jugadores.
Aceptar


Sin conexión a la base de datos
¡Error! ¡Las credenciales ingresadas no corresponden a ningún usuario registral Inténtelo de nuevo o hágalo más tarde
Aceptar


Sin conexión con el servidor
¡Error! ¡Ocurrió un error al tratar de conectarse al servidor del Sistema! Inténtelo más tarde
Aceptar

8.7 CU – 08. Validar letra


Es el album de música más vendido de la historia
T
Q W E R T Y U I O P
A S D F G H J K L Ñ
Z X C V B N M
Es el album de música más vendido de la historia
T
Q W E R T Y U I O P
A S D F G H J K L Ñ
Z X C V B N M

8.8 CU – 09. Establecer resultados de partida

8.9 CU – 10. Ver puntaje

8.10 CU – 11. Cambiar idioma

9 Estándar de codificación

Propósito	Promover prácticas de programación que resulten en un software seguro, fiable, verificable y de fácil mantenimiento.
Idioma	Se ha decidido usar el idioma Español para la nomenclatura utilizada en este estándar; la decisión fue tomada por todos los miembros del equipo con fines académico y de mejor nuestro aprendizaje en esta tecnología.
Estructura del código fuente	Se ha decidido organizar los archivos de código fuente en carpetas lógicas según la funcionalidad. Se ha decidido dividir el código fuente del lado del **Cliente** en las siguientes capas: **Vistas** (interfaces que el Cliente consume), **Alertas** (interfaces tipo pop-up que el Cliente recibe dependiendo una acción realizada), **Utilidades** (clases que contienen código especifico de ayuda para ciertas funcionalidades), **SocketCliente** (método de comunicación que gestiona funcionalidades como conectar, escuchar y enviar mensaje al SocketServidor) y **Seguridad** (clase que ayuda al proceso de validación del inicio de sesión). Por otro lado, del lado del **Servidor** se decidió dividir el código fuente en las siguientes capas: **Modelo** (definición de la estructura de las clases cargadas desde la base de datos), **HangmanServicio** (definición de las funcionalidades especificadas en Servicios) y **SocketServidor** (método de comunicación que gestiona la lógica de la partida). Adicionalmente, se decidió agregar una división más que trabaja en común con ambas capas para ser consumido, la cual es **Biblioteca** que esta dividido de la siguiente manera: **DTO** (definición de los Data Transfer Objects que son usados en la lógica del proyecto) y **Servicios** (definición de las funcionalidades necesarias para ser consumidas).

Nombrado de clases	Las Clases deben tener como nombre un sustantivo o una frase que haga alusión al mismo (persona, cosa, etc.) y empezar con letra mayúscula. Se debe evitar nombre con verbos.
	a) Se ocupará como formato el **UpperCamelCase** para el nombrado de clases.
	Como ejemplo de nombrado de clases se tiene:
	`private class PalabraDTO{…}`
	`private class JugadorDTO{…}`

Nombrado de Componentes de Interfaz	Los componentes de una interfaz gráfica tales como botones, campos de texto, etiquetas, etc. deben de empezar con una abreviación del componente.
	• Para abreviar un componente de una sola palabra, se deben quitar sus vocales y ocupar las tres primeras consonantes.
	• Para abreviar un componente formado de dos palabras: de la primera palabra, se deben quitar sus vocales y ocupar las tres primeras consonantes, y juntar esto con la segunda palabra, empezando por mayúscula.
	• Se ocupará el formato **lowerCamelCase**.
	Como ejemplos de nombrado se tiene:
	`Label x:Name="txtUsuario"`
	`Label x:Name="txtFechaNacimiento"`
	`Label x:Name="psBContrasenia"`
	`Label x:Name="btnRegresar"`

Nombrado de Variables	Las variables del programa deben tener como nombre un sustantivo que haga alusión a el mismo y empezar con una letra minúscula, se ocupara como formato el **lowerCamelCase**.
	Como ejemplo de nombrado de variables se tiene:
	`// Mal nombrado`
	`var token = “”;`
	`// Buen nombrado`
	`var token = new string[] {};`

Nombrado de Constantes	Las constantes del programa deben tener como nombre un sustantivo que haga alusión a el mismo y se escribirán con mayúsculas, además serán nombradas con el formato **SCREAMING_SNAKE_CASE.**
	Como ejemplo de nombrado de constantes se tiene:
	// Mal nombrado
	const int direccion = 192.168.0.1;
	// Buen nombrado
	const int direccionIP = 192.168.0.1;

Nombrado de namespace	Los namespace permiten declarar que todos los tipos de un archivo están en un único espacio de nombres. El nombre elegido para un namespace debe indicar la funcionalidad disponible por tipos en el espacio de nombres; serán nombrados con el formato **UpperCamelCase**.
	Como ejemplo de nombrado de namespace se tiene:
	// Mal nombrado
	namespace Vistas
	// Buen nombrado
	namespace HangmanGame_Cliente.Cliente.Vistas

Nombrado de Métodos	Los métodos de deben ser verbos o frases que hagan alusiones a verbos y se ocuparan prefijos en caso de ser necesario, además se utilizara el formato **UpperCamelCase**. Como ejemplo de nombrado de métodos se tiene: **public static void ActualizarJugador**(){…} **private async Task NotificarClientes**(string codigoPartida, string mensaje){…} **private void AjustarTextos**(string nickname){…}
Comentarios	Los comentarios en programación se utilizan para poner aclaraciones del código, y así es más fácil de entender lo que hace. a) Explicar el por qué, no el cómo. b) Documentar si existe algo inusual o inesperado dentro del código. c) Evitar palabras altisonantes, chistes, bromas o cualquier comentario innecesario.

Buenos Comentarios	Un buen comentario describe el que hace cierta parte del código y no el cómo. Todos los comentarios deben ser claros y consistentes, utilizando mismo estilo. Como ejemplo se tiene el siguiente:
	// Limpiar clientes desconectados
	lock (clientesPartida)
	{
	if (!clientesPartida.Any())
	{
	clientes.TryRemove(codigoPartida, out _);
	}
	}

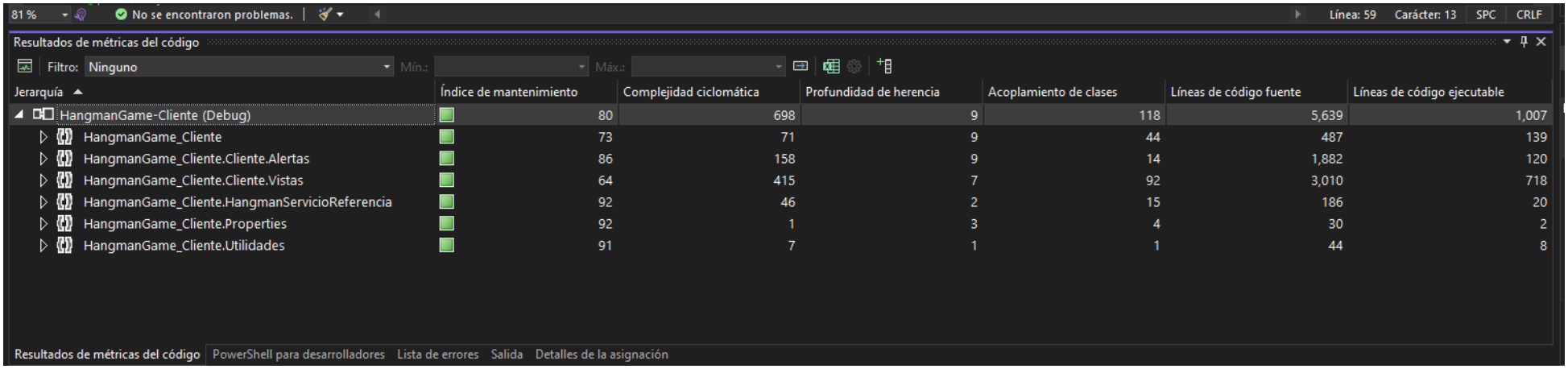
<table>
<tr><td>Malos Comentarios</td><td>En un mal comentario se invierte mucho tiempo en generar cajas de comentarios “bonitas”, esto hace que se pierda mucho tiempo intentando acomodar cada caja cada vez que el comentario sea editado. Como ejemplo se tiene el siguiente:
// <summary>
// User Class
// </summary>
public class User
{
// <summary>
// User Type
// </summary>
public string Type { get; set; }

// <summary>
// Verifies if user is active
// </summary>
public bool IsActive { get; set; }
}</td></tr>
<tr><td>Indentación</td><td>La indentación (o mejor conocido como sangría) es utilizado para una mejor lectura y comprensión del código. Esta debería seguir las siguientes características:
• No hay un salto de línea antes de abrir una llave.
• Hay un salto de línea después de la llave de cierre, solo si esa llave termina una declaración o el cuerpo de un método, constructor o clase con nombre. Por ejemplo, no hay salto de línea después de la llave si va seguido de un else o “;”.
• Poner una sola instrucción por línea.
• Dejar un espacio entre cada uno de los elementos de una expresión.
• Agregar una tabulación en las instrucciones que están dentro de estructuras de control, métodos o bloques específicos de código.
Como ejemplo se tiene el siguiente bloque de código:

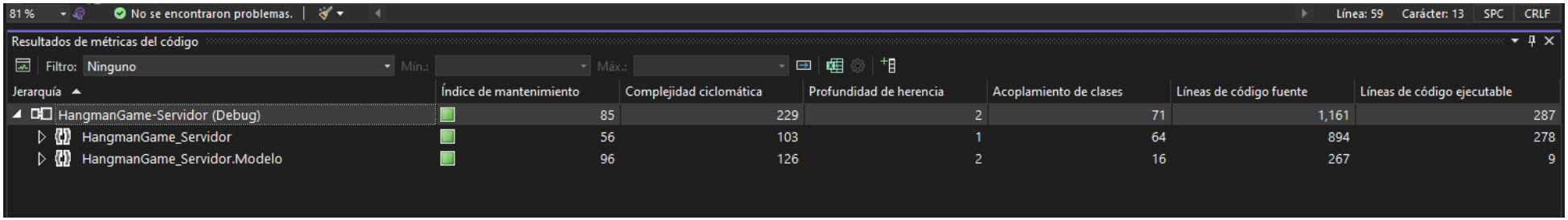
public class example {
public void method() {
if(condition()) {
try{
something();
}catch (ProblemException e) {
recover();
}
}else if (otherCondition()) {
somethingElse();
}else {
lastThing();
}
}
};</td></tr>
</table>

Capitalización	**lowerCamelCase**: La primera letra no está en mayúsculas, pero las siguientes palabras ocupan una letra mayúscula para diferenciar unas palabras de otras.
	
	UpperCamelCase/Pascal Case: La primera letra está en mayúsculas y las primeras letras de las subsecuentes palabras se encuentran también en mayúsculas.
	
	SCREAMING_SNAKE_CASE: Las palabras se escriben en mayúsculas separadas por un "_". Algunos ejemplos son:
	• **lowerCamelCase**:
	`private double workedHours = 0.00;`
	• **UpperCamelCase/Pascal Case:**
	`private class PlayerGuest{...}`
	• **SCREAMING_SNAKE_CASE:**
	`private double MEXICAN_PESO = 18.50;`

10 Resultado de análisis estático de código



Jerarquía	Índice de mantenimiento	Complejidad ciclomática	Profundidad de herencia	Acoplamiento de clases	Líneas de código fuente	Líneas de código ejecutable
HangmanGame-Cliente (Debug)	80	698	9	118	5,639	1,007
HangmanGame_Cliente	73	71	9	44	487	139
HangmanGame_Cliente.Cliente.Alertas	86	158	9	14	1,882	120
HangmanGame_Cliente.Cliente.Vistas	64	415	7	92	3,010	718
HangmanGame_Cliente.HangmanServicioReferencia	92	46	2	15	186	20
HangmanGame_Cliente.Properties	92	1	3	4	30	2
HangmanGame_Cliente.Utilidades	91	7	1	1	44	8



Jerarquía	Índice de mantenimiento	Complejidad ciclomática	Profundidad de herencia	Acoplamiento de clases	Líneas de código fuente	Líneas de código ejecutable
HangmanGame-Servidor (Debug)	85	229	2	71	1,161	287
HangmanGame_Servidor	56	103	1	64	894	278
HangmanGame_Servidor.Modelo	96	126	2	16	267	9

Referencias Bibliográficas

Martin, R. C. (2009) Clean Code: A Handbook of Agile Software Craftsmanship. Prentice Hall.

Carnegie Mellon University (2006): Assignment Kit for Coding Standard, Personal Software Process for Engineers: Part I

BillWagner. (2023). Convenciones de codificación de C# de documentación de .NET - C#. Microsoft.com. Recuperado de: https://learn.microsoft.com/es-es/dotnet/csharp/fundamentals/coding-style/coding-conventions

ByteHide. (2022). C# Clean Code Guide (Tips and Tricks) 2024. ByteHide Blog. Recuperado de: https://www.bytehide.com/blog/important-tips-to-writing-clean-csharp-code