LICEO LA PAZ
FORMACIÓN PROFESIONAL

PROGRAMACIÓN DE PROCESOS Y SERVIZOS

Documentación del Proyecto: Juego Ahorcado

LAURA LODEIRO CASAS

ANA CARAMÉS CANOSA

CURSO 2024/2025

1. Requisitos del Ejercicio

El proyecto consiste en desarrollar un juego del "Ahorcado" utilizando una arquitectura cliente/servidor con las siguientes tecnologías y requisitos:

-Tecnologías

- Hibernate para la persistencia de datos
- Entorno de ventanas para la interfaz gráfica
- Sockets multihilo para la comunicación cliente/servidor

-Funcionalidades principales

Base de datos:

- Almacenar palabras para el juego (con ID y la palabra)
- Importar palabras desde un archivo JSON (palabras-ahorcado.json)
- Guardar datos de jugadores (ID y nombre)
- Registrar puntuaciones de partidas (fecha, hora, éxito o fracaso)

Servidor:

- Escuchar en el puerto 65000
- Generar números aleatorios para seleccionar palabras
- Verificar y registrar jugadores en la base de datos
- Procesar las jugadas de cada participante
- Controlar los turnos y el flujo del juego
- Gestionar partidas individuales o con dos jugadores
- Mantener el registro de puntuaciones

Cliente:

- Proporcionar interfaz gráfica para el jugador
- Permitir al jugador introducir su nombre
- Permitir elegir el modo de juego (solo o esperar otro jugador)
- Proporcionar interfaz para introducir letras
- Mostrar botones para enviar letra, cancelar partida y consultar puntuación

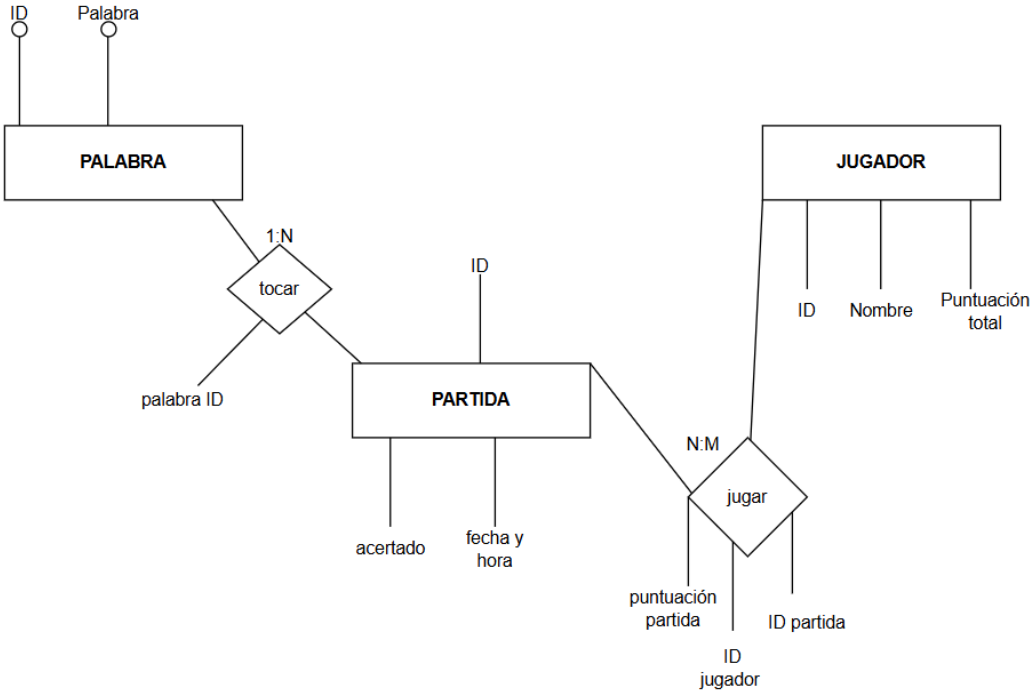
Reglas del juego:

- Número máximo de intentos = mitad de la longitud de la palabra
- Si juega un solo jugador, se siguen las reglas básicas
- Si juegan dos jugadores:
    - Se elige aleatoriamente quién empieza
    - Un jugador mantiene el turno mientras acierta letras
    - Al fallar, el turno pasa al otro jugador
    - Si un jugador cancela, la partida se cancela para ambos

Sistema de puntuación:

- Victoria con palabra de menos de 10 caracteres: 1 punto
- Victoria con palabra de 10 o más caracteres: 2 puntos

2. Diagrama ER de la Base de Datos



Descripción de las entidades:

Palabra

- id: Identificador único de la palabra (entero, clave primaria)
- palabra: Texto de la palabra para jugar (cadena)

Jugador

- id: Identificador único del jugador (entero, clave primaria)
- nombre: Nombre del jugador (cadena)
- puntuacionTotal: Puntuación acumulada del jugador (entero)

Partida

- id: Identificador único de la partida (entero, clave primaria)

- acertado: Indica si la palabra fue adivinada (booleano)
- fecha_hora: Fecha y hora de la partida (timestamp)
- palabra_id: Referencia a la palabra utilizada (clave foránea)

JugadorPartida

- id: Identificador único de la relación (entero, clave primaria)
- idJugador: Referencia al jugador (clave foránea)
- idPartida: Referencia a la partida (clave foránea)
- puntuacionPartida: Puntos obtenidos por el jugador en esta partida (entero)

3. Estructura del Código

Entidades

- Jugador.java: Representa a un jugador con su nombre y puntuación total
- Palabra.java: Representa las palabras disponibles para el juego
- Partida.java: Representa una partida jugada con su resultado
- JugadorPartida.java: Relaciona jugadores con partidas y sus puntuaciones específicas

Clases DAO (Data Access Objects)

- JugadorDAO.java: Gestiona operaciones de la entidad Jugador
- PalabraDAO.java: Gestiona operaciones de la entidad Palabra, incluida la carga desde JSON
- PartidaDAO.java: Gestiona operaciones de la entidad Partida
- JugadorPartidaDAO.java: Gestiona operaciones de la relación JugadorPartida

Componentes del servidor

- ServidorAhorcado.java: Clase principal del servidor que gestiona conexiones
- PartidaAhorcado.java: Implementa la lógica del juego y gestiona la partida

Componentes del cliente

- AhorcadoApplication.java: Clase principal de la aplicación JavaFX
- AhorcadoController.java: Controlador que gestiona la interfaz gráfica y la comunicación con el servidor
- ahorcado-view.fxml: Definición de la interfaz gráfica en formato FXML
- styles.css: Estilos CSS para la interfaz gráfica

4. Protocolo de comunicación Cliente-Servidor

El servidor y el cliente se comunican a través de mensajes de texto con el siguiente formato:

Mensajes del Servidor al Cliente:

- SOLICITUD_NOMBRE: Solicita el nombre del jugador
- SOLICITUD_MODO_JUEGO: Solicita el modo de juego (solo/esperar)
- PARTIDA_EN_CURSO: Informa que hay una partida en curso
- PARTIDA_SOLO_INICIADA: Informa que inicia partida individual
- PARTIDA_INICIADA;OPONENTE:[nombre]: Informa que inicia partida con otro jugador
- ESTADO;PALABRA:[progreso];INTENTOS_RESTANTES:[n]: Estado actual de la partida
- TURNO;JUGADOR:[nombre]: Indica que es el turno del jugador
- ESPERA_TURNO: Indica que debe esperar su turno
- SOLICITUD_LETRA: Solicita una letra al jugador

- ACIERTO: Indica que la letra acertó
- FALLO: Indica que la letra no estaba en la palabra
- INTENTOS_RESTANTES:[n]: Informa los intentos restantes
- PALABRA;[progreso]: Muestra el progreso actual de la palabra
- FIN_PARTIDA;RESULTADO:[resultado];PALABRA:[palabra]: Fin de partida con resultado
- PARTIDA_CANCELADA: La partida ha sido cancelada
- PUNTUACION_TOTAL;[n]: Muestra la puntuación total del jugador
- MODO_INVALIDO: Indica que el modo seleccionado no es válido

Mensajes del Cliente al Servidor:

- Nombre del jugador
- Modo de juego: "solo", "esperar" o "puntuacion"
- Letra propuesta para jugar
- "cancelar": Para cancelar la partida en curso

5. Flujo de Juego

Inicio

1. El cliente se conecta al servidor
2. El servidor solicita el nombre del jugador
3. El cliente envía el nombre
4. El servidor verifica/registra al jugador en la base de datos
5. El servidor solicita el modo de juego
6. El cliente selecciona el modo

Partida Individual

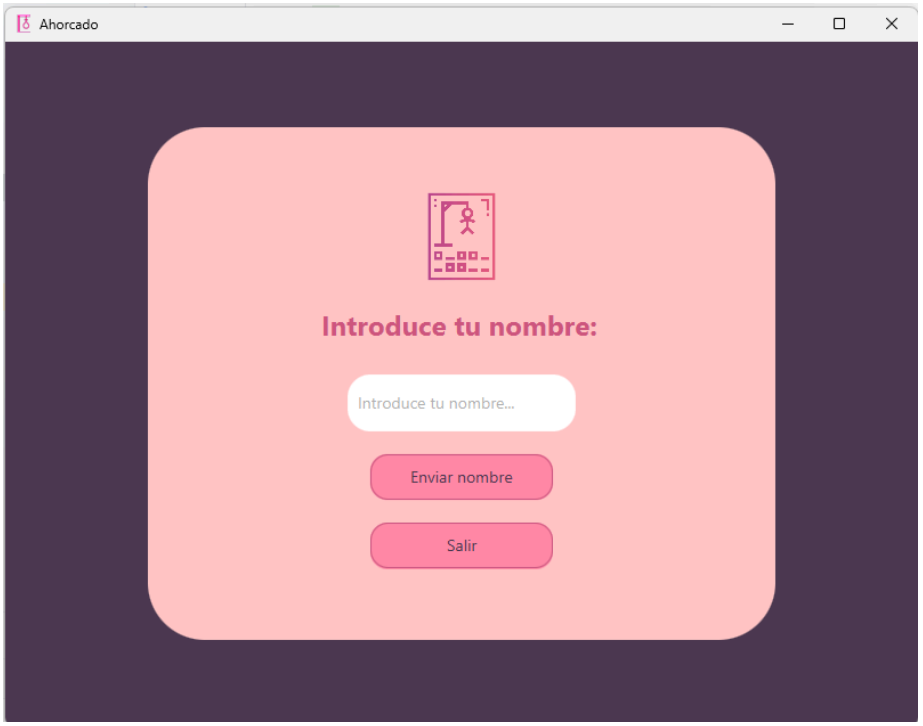
1. El servidor inicia una partida solo para este jugador
2. El servidor selecciona una palabra aleatoria
3. El servidor muestra el estado inicial de la palabra (guiones)
4. El servidor solicita letra al jugador
5. El jugador envía una letra
6. El servidor verifica si la letra está en la palabra
7. El proceso 4-6 se repite hasta:
    - El jugador adivina la palabra (victoria)
    - El jugador agota los intentos (derrota)
    - El jugador cancela la partida

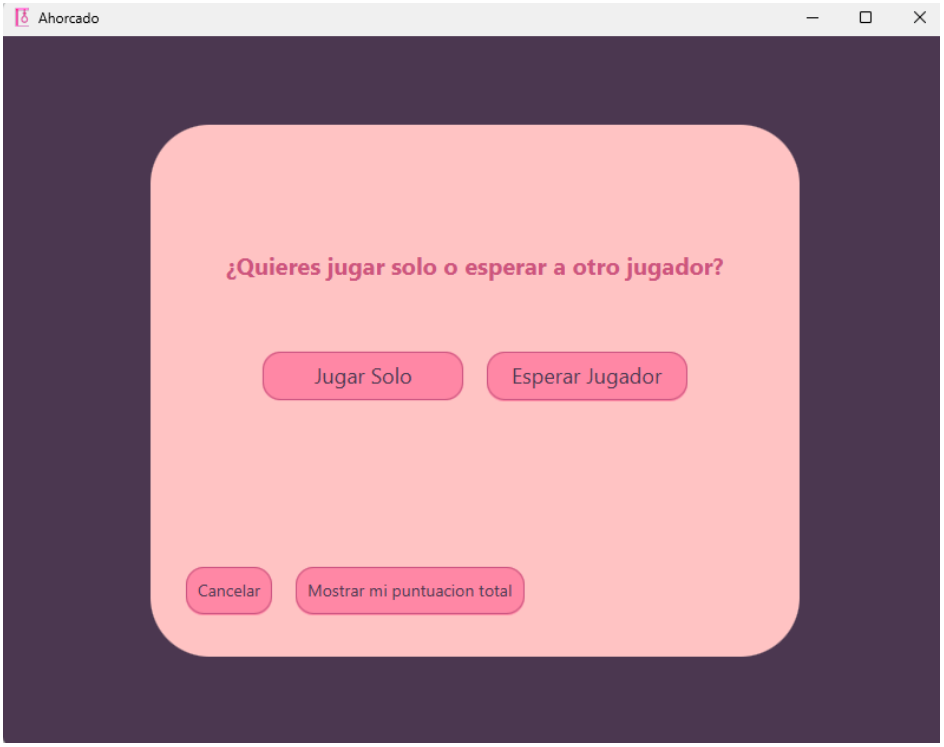
Partida con dos jugadores

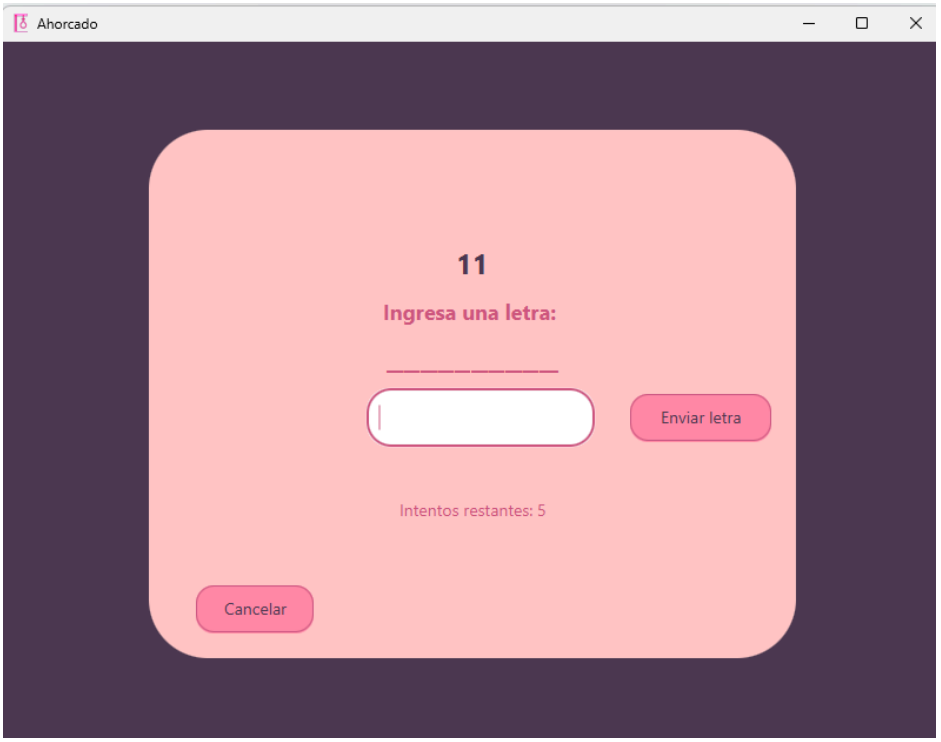
1. El servidor empareja a dos jugadores que eligieron "esperar"
2. El servidor selecciona una palabra aleatoria
3. El servidor decide aleatoriamente quién comienza
4. El servidor muestra el estado inicial de la palabra (guiones)
5. El servidor solicita letra al jugador actual
6. El jugador envía una letra
7. El servidor verifica si la letra está en la palabra
8. Si acierta, continúa su turno
9. Si falla, el turno pasa al otro jugador
10. El proceso 5-9 se repite hasta:
    - Un jugador adivina la palabra (victoria para él, derrota para el otro)
    - Ambos jugadores agotan intentos (empate)

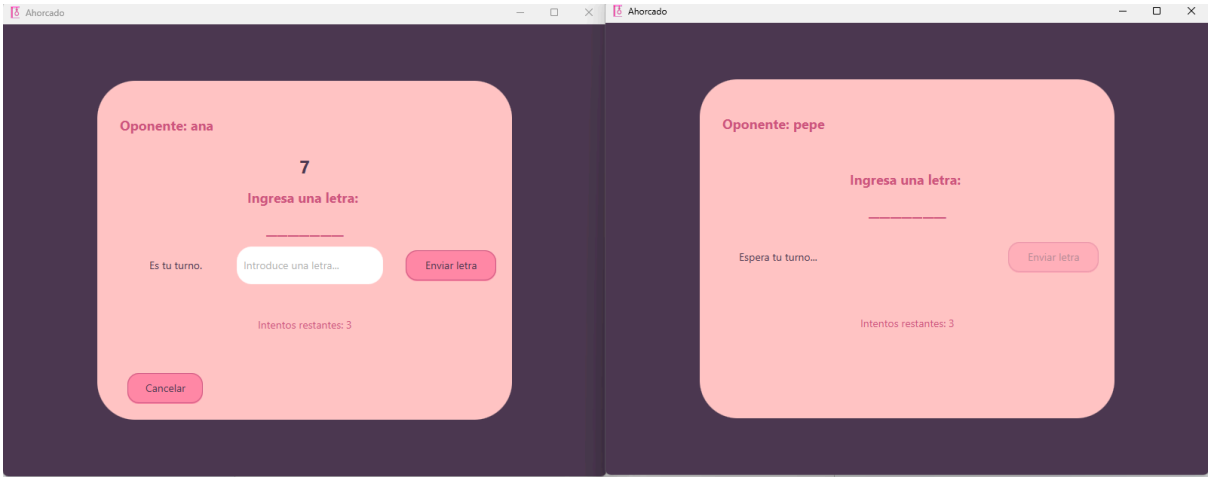
- Un jugador cancela (partida cancelada para ambos)

6. Interfaz gráfica del cliente









La interfaz gráfica del cliente está desarrollada con JavaFX y consta de varias pantallas (VBox) que se muestran u ocultan según el estado del juego:

Pantallas principales

1. Pantalla de nombre (pantallaNombre):
    - Permite al jugador ingresar su nombre
    - Contiene un campo de texto y un botón para enviar
    - Muestra un logo del juego

2. Pantalla de modo (pantallaModo):
    - Permite seleccionar entre jugar solo o esperar a otro jugador
    - Incluye botones para cada opción
    - Contiene un botón para mostrar la puntuación total
3. Pantalla de juego (pantallaJuego):
    - Muestra el estado actual de la palabra (con guiones)
    - Indica los intentos restantes
    - Permite introducir una letra
    - Muestra feedback sobre aciertos/fallos
    - Incluye un contador de tiempo para cada turno
    - Muestra información sobre el oponente (en modo multijugador)
    - Contiene un botón para cancelar la partida
4. Pantalla de finalización (pantallaSaliendo):
    - Muestra el resultado de la partida
    - Incluye un indicador de progreso mientras se reinicia

7. Consideraciones de implementación

Multihilo

- El servidor implementa un sistema multihilo para gestionar múltiples clientes
- Cada partida se ejecuta en su propio hilo
- Se utilizan colas bloqueantes para la comunicación entre hilos
- El cliente también implementa un hilo separado para la comunicación con el servidor, manteniendo la interfaz de usuario responsiva

Persistencia

- Hibernate se utiliza para la persistencia de datos
- Se implementa una capa DAO para cada entidad
- La configuración de Hibernate está en hibernate.cfg.xml (no incluido en este repositorio)

Tolerancia a fallos

- El servidor maneja desconexiones de clientes
- Se implementa un timeout de 15 segundos para las respuestas de los jugadores
- Después de dos timeouts consecutivos, se cancela la partida
- El cliente maneja errores de conexión y reconexiones automáticas
- Se muestran alertas de confirmación antes de cancelar una partida en curso

Colas de jugadores

- El servidor mantiene una cola de jugadores pendientes
- Los jugadores se emparejan según su modo seleccionado
- Los jugadores que eligen "solo" tienen prioridad sobre los que eligen "esperar"

Interfaz de usuario

- La interfaz gráfica está diseñada con JavaFX y FXML
- Se utiliza el patrón MVC (Modelo-Vista-Controlador)
- La actualización de la interfaz se realiza a través de Platform.runLater() para evitar problemas de concurrencia
- Se implementa un contador visual para el tiempo de respuesta