Proyecto Final

Curso	Programación Avanzada
Código	SC-601
Profesor	Luis Andrés Rojas Matey

Introducción

En el juego conocido como **Ahorcado** se trata de adivinar una palabra indicando las letras que la componen, con un número máximo de intentos.

Objetivo

Aplicar los conocimientos adquiridos para desarrollar una aplicación web con arquitectura **MVC** (*Model-View-Controller*) del juego **Ahorcado**.

Especificaciones funcionales

Se deben precargar en una base de datos una cantidad de al menos 100 palabras (esto será el "diccionario de palabras"), que cumplan con los siguientes requisitos:

- Cada palabra debe tener una longitud (cantidad de letras) entre 5 y 10.
- Debe haber al menos 3 palabras que empiecen con cada una de las letras.
- Las palabras serán en idioma español.
- Al menos la mitad de las palabras deben tener tilde.
- No se debe distinguir entre mayúsculas y minúsculas (es decir, serán tratadas como *case insensitive*).

Al ejecutar la aplicación web, se le deben proveer cuatro opciones al usuario:

1. Agregar una palabra nueva al diccionario.
2. Crear un jugador nuevo.
3. Crear una partida nueva.
4. Mostrar el escalafón de jugadores.

Agregar una palabra nueva al diccionario

Esta parte le permite al usuario agregar nuevas palabras al diccionario (el cual a su vez está persistente en la base de datos), por lo cual podrán ser escogidas cuando se crea una partida nueva.

Cualquier palabra ingresada debe cumplir con los requerimientos anteriormente mencionados. Así mismo, no se debe permitir agregar una palabra que ya esté guardada con anterioridad.

Para efectos prácticos, las palabras que contengan una letra con tilde, será guardada así (con tilde) en el diccionario. Sin embargo, a la hora de validar si la palabra existe, esta no debe tomar en cuenta la tilde. Por ejemplo: si en el diccionario ya existe la palabra `árbol`, no debe permitir al usuario agregar una nueva palabra si digita `arbol` o `arból`.

Crear un jugador nuevo

Esta sección permite crear un jugador a partir de dos datos que se deben solicitar al usuario:

- `Identificación`. Es un número entero positivo único (por ejemplo, un número de cédula).
- `Nombre`. Es un *string* con el nombre del jugador.

Cada jugador tendrá asociado su propio `Marcador`, el cual será la suma de los puntos según los resultados de las partidas, asignados de la siguiente forma (más adelante se especificará en qué consisten los niveles):

- 1 punto positivo (`+1`) por cada partida ganada en nivel `Fácil`.
- 2 puntos positivos (`+2`) por cada partida ganada en nivel `Normal`.
- 3 puntos positivos (`+3`) por cada partida ganada en nivel `Difícil`.
- 1 punto negativo (`-1`) por cada partida perdida en nivel `Fácil`.
- 2 puntos negativos (`-2`) por cada partida perdida en nivel `Normal`.
- 3 puntos negativos (`-3`) por cada partida perdida en nivel `Difícil`.

Este `Marcador` puede ser negativo en caso que el jugador haya perdido más partidas de las que ha ganado.

Crear una partida nueva

Cuando el usuario selecciona esta opción, se le solicitará que escoja el jugador (de los creados previamente), esto se hará presentado la concatenación de su `Identificación` y `Nombre`. Por ejemplo: `123456789 - Ana`.

Una vez escogido el jugador, se debe mostrar la opción de escoger el nivel. Este nivel está condicionado únicamente al tiempo disponible para terminar la partida:

- `Fácil`: 1.5 minutos (1 minuto con 30 segundos).
- `Normal`: 1 minuto.
- `Difícil`: 0.5 minutos (30 segundos).

Cuando se haya escogido el nivel, la aplicación debe seleccionar aleatoriamente una palabra del diccionario, deshabilitando o eliminando esta palabra para que no pueda ser seleccionada de nuevo. Esta palabra estará "escondida" para el jugador, por lo que únicamente se mostrarán "subrayados" (*underscores*) para representar la cantidad de letras de dicha palabra. Por ejemplo, suponiendo que la palabra seleccionada del diccionario es la palabra `árbol` (de 5 letras), entonces se debe mostrar algo similar a esto (5 subrayados o *underscores*):

_ _ _ _ _

En este momento, se debe mostrar el tiempo disponible según el nivel escogido. Este tiempo será una cuenta regresiva que se actualizará cada segundo.

Para que el jugador pueda seleccionar una letra, se deben presentar como botones de forma individual y en mayúsculas, por lo que se tendrá que presionar la letra que se desea escoger. Es decir, algo similar a esto, donde cada letra es un botón (pero no necesariamente en este orden o división):

A	-	B	-	C	-	D	-	E
F	-	G	-	H	-	I	-	J
K	-	L	-	M	-	N	-	Ñ
O	-	P	-	Q	-	R	-	S
T	-	U	-	V	-	W	-	X
Y	-	Z						

Cada vez que el jugador presiona una letra, sucederá lo siguiente:

- El botón de la letra presionada se deshabilitará o esconderá, ya que no se podrá volver a presionar.
- Si la letra escogida no es parte de la palabra, entonces el jugador perderá uno de sus intentos (más adelante se explica cómo funcionan estos intentos).
- Si la letra es parte de la palabra, entonces la letra aparecerá en vez del subrayado (*underscore*) en el sitio o sitios donde dicha letra se halle en la palabra. Ejemplo: si la palabra es `árbol` y el jugador escoge la letra `A`, entonces se debe mostrar así (notar la tilde):

Á _ _ _ _

El jugador tendrá únicamente 5 intentos fallidos, es decir, si selecciona 5 letras que no son parte de la palabra, ya no podrá seleccionar más palabras. La cantidad de intentos disponibles debe estar siempre visible (similar comportamiento que el contador del tiempo regresivo explicado anteriormente).

El juego finaliza cuando se da alguno de los tres escenarios siguientes:

1. Una victoria del jugador cuando logra seleccionar todas las letras correspondientes a la palabra.
2. Una pérdida cuando el jugador utiliza sus 5 intentos.
3. Una pérdida cuando el tiempo (el contador regresivo) llegue a cero (0).

En cualquier caso (cuando se finaliza la partida), la página debe indicar el resultado (victoria o pérdida) y mostrar la palabra completa (que estaba parcial o totalmente escondida).

En todo momento también se debe tener la opción de "nuevo intento", el cual será un botón que permite crear una nueva partida con el mismo jugador y el mismo nivel de juego previamente seleccionados. Eso sí, si la partida actual no ha finalizado y se presiona este botón, entonces se da por terminada (la partida) y se establece como una pérdida para el jugador.

Mostrar escalafón de jugadores

Esta es una tabla que muestra la lista de jugadores (`Identificación` y `Nombre`), ordenados por su `Marcador` de forma descendente. Además debe incluir la cantidad de puntos ganados y perdidos. Por ejemplo:

Identificación	Nombre	Marcador		Ganadas	Perdidas
123456789	Ana	10		15	5
987654321	Braulio	4		10	6
304560765	Carlos	-2		6	8

Especificaciones técnicas

Esta aplicación debe ser creada utilizando `ASP.NET Core MVC` con el lenguaje `C#` del `.NET Framework 4.8.1`.

Para la persistencia de los datos, se debe utilizar una base de datos **SQL** (como `SQLite`, `MariaDB`, `MySQL`, `PostgreSQL`, `SQL Server`, etc.). Así mismo, se debe utilizar la herramienta `Entity Framework`, ya sea en modo "*Code First*", o bien, "*Database First*".

Cualquier otro aspecto (por ejemplo, estilos, paquetes de `NuGet`, uso de un API propio, etc.), queda a criterio de cada grupo.

Entregables

Debido a que este proyecto se debe hacer según los grupos establecidos previamente, el único entregable (es decir, lo único que se debe subir al **Campus Virtual**) es el vínculo (*link*) al repositorio en línea de **Git** (como por ejemplo, a `GitHub` o `GitLab`). Este vínculo debe ser subido por solo uno de los miembros del grupo. Este repositorio puede ser privado pero deberá ser público al momento que les llegue el turno de la exposición del proyecto (ya sea en la semana 14 o 15), para que el profesor pueda tener acceso al mismo.

En el repositorio debe estar lo siguiente:

- Todo el código fuente del proyecto, excepto los archivos compilados (es decir, sin las carpetas `bin` ni `obj`).
- Un archivo `README.md` (escrito en **Markdown**) en la raíz del proyecto, que contenga lo siguiente:
    - Los nombres y carnés de los integrantes del grupo. Estos serán los únicos que serán tomados en cuenta para la calificación.
    - El nombre de usuario y correo de **Git** de cada integrante.
    - Un diagrama (puede utilizar, por ejemplo, **Mermaid**) de la definición de la base de datos con sus tablas, columnas, tipos de campos, llaves, referencias, etc.
    - Todas las referencias de sitios webs con *snippets* de código utilizados, así como los *prompts* (tanto de entrada como salida) de los agentes de AI, que se hayan utilizado.

Evaluación

El proyecto será calificado según la rúbrica que se presenta en el programa del curso.