1. Requisitos funcionales

1) El juego debe mostrar un botón en la pantalla inicial para comenzar una nueva partida.

2) Al iniciar la partida, se deben mostrar dos ciudades una al lado de la otra.

3) El juego debe mostrar la población de la primera ciudad únicamente.

4) El jugador debe poder elegir si la segunda ciudad es más grande o pequeña en población.

5) El sistema debe determinar si la elección del jugador es correcta o incorrecta.

6) Si la elección es correcta, la segunda ciudad pasa a ser la primera, y se selecciona una nueva ciudad para compararla.

7) El juego debe mostrar la extensión de la segunda ciudad después de la respuesta.

8) El sistema debe añadir una nueva ciudad aleatoria para seguir jugando.

9) Si la respuesta es correcta, se incrementa el marcador de puntuación en 1 punto.

10) Si la elección es incorrecta, el juego termina y se vuelve a la pantalla inicial.

11) El sistema debe comprobar si la puntuación obtenida es la más alta hasta el momento y guardarla si lo es.

12) El juego debe leer las ciudades desde una base de datos .json.

2. Requisitos no funcionales

1) El sistema debe tener un tiempo de respuesta inferior a 1 segundo al cargar una nueva pregunta.

2) La interfaz debe ser intuitiva y accesible para usuarios sin conocimientos técnicos.

3) La base de datos de ciudades debe poder actualizarse fácilmente.

5) El juego debe ser jugable en navegador web.

6) El desarrollo del proyecto debe incluir buenas prácticas de ingeniería de software, como el diseño previo de diagramas UML (casos de uso, clases y flujo de actividad).

3. Casos de uso

CASO DE USO 1 INICIAR PARTIDA

Elemento	**Descripción**
Nombre	INICIAR PARTIDA
Actor principal	JUGADOR
Precondiciones	Debe de estar en la página inicial del juego.
Flujo principal	El Actor da al botón de jugar El sistema inicia la partida Se cargan las ciudades de la base de datos Se muestra la primera ciudad con su extensión y alado la segunda
Postcondiciones	Se inicia el juego Se pasa a la pantalla de juego

CASO DE USO 2 JUGAR PARTIDA

Elemento	**Descripción**
Nombre	JUGAR PARTIDA
Actor principal	Jugador
Precondiciones	Se ha debido iniciar la partida dando al botón de juagar Debe aparecer en la pantalla las dos ciudades
Flujo principal	El juego debe mostrar la población de la primera ciudad únicamente. El jugador debe poder elegir si la segunda ciudad es más grande o pequeña en población. El sistema debe determinar si la elección del jugador es correcta o incorrecta. Si la elección es correcta, la segunda ciudad pasa a ser la primera, y se selecciona una nueva ciudad para compararla. El juego debe mostrar la población de la segunda ciudad después de la respuesta.

Elemento	Descripción
	El sistema debe añadir una nueva ciudad aleatoria para seguir jugando.
Postcondiciones	Se repite el proceso Cuando el jugador fallé se para este proceso

CASO DE USO 3 TERMINAR PARTIDA

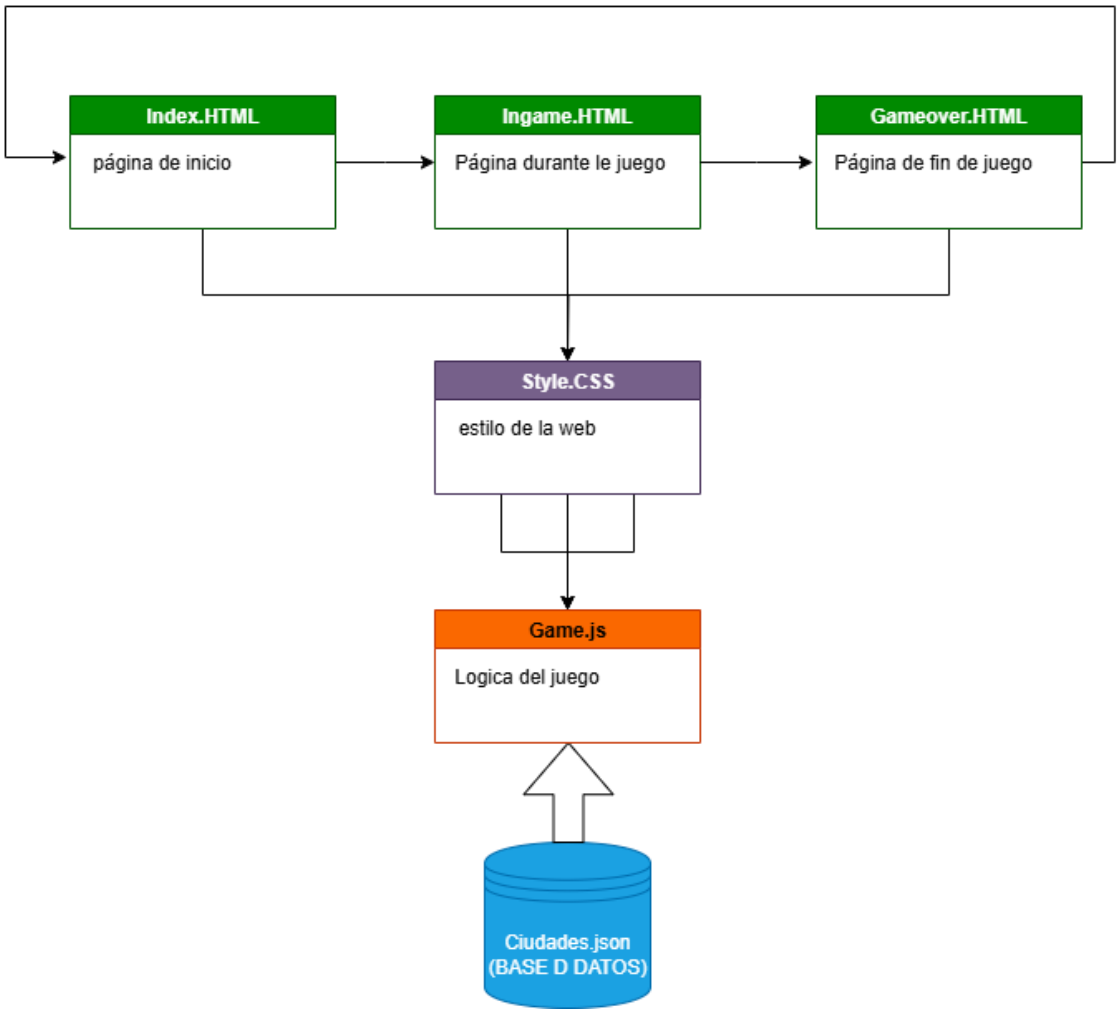
Elemento	Descripción
Nombre	TERMINAR PARTIDA
Actor principal	Jugador
Precondiciones	El jugador ha tenido que elegir la opción incorrecta
Flujo principal	El jugador falla La partida se acaba Se vuelve a la pantalla principal
Postcondiciones	Se vuelve a la pantalla inicial para que el jugador pueda volver a jugar

CASO DE USO 4 ACTUALIZAR MARCADOR

Elemento	Descripción
Nombre	ACTUALIZAR MARCADOR
Actor principal	SISTEMA
Precondiciones	El jugador ha tenido que perder la partida
Flujo principal	El jugador falla La partida se acaba Se compara la puntuación actual con la mejor histórica Si es mayor se actualiza y si es menor se desecha
Postcondiciones	Se vuelve a la pantalla inicial para que el jugador pueda volver a jugar y vea el marcador de puntuación bien

4. Diagrama de arquitectura del proyecto

He elegido una **arquitectura basada en HTML, CSS y JavaScript** con una base de datos en formato **JSON**. Esta decisión la he tomado considerando los objetivos principales del proyecto: que el juego pueda ejecutarse en cualquier navegador web de forma sencilla, sin necesidad de instalaciones adicionales y que pueda ser fácilmente compartido y probado por múltiples usuarios desde distintos equipos.



El uso conjunto de estas 3 permite que el juego sea completamente basado en web, eliminando la necesidad de instalar programas adicionales o entornos de desarrollo complejos. Esto también facilita la publicación en internet mediante cualquier hosting estático, haciendo que el juego esté disponible para cualquier usuario con un navegador moderno.

Uso de JSON como base de datos

He decidido utilizar un archivo **JSON** para almacenar la información de las ciudades en lugar de sistemas de gestión de bases de datos como SQL, Oracle o similares, por varias razones:

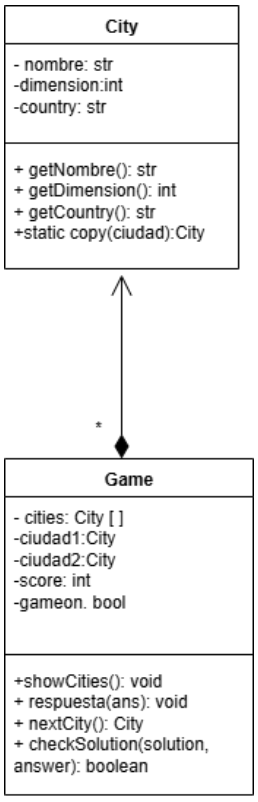
1. **Simplicidad y portabilidad:** Permite cargar los datos directamente en JavaScript mediante fetch, sin necesidad de servidores complejos ni configuraciones adicionales.

2. **Compatibilidad con web:** Al usar JSON y fetch, los datos se pueden servir desde un hosting estático, lo que permite que el juego funcione directamente desde un navegador sin necesidad de backend ni bases de datos remotas.
3. **Escalabilidad controlada:** Aunque JSON puede contener miles de ciudades, es posible implementar técnicas de selección aleatoria y lazy loading si se desea, sin complicaciones adicionales.

Por qué no eligí otras opciones

1. **SQL/Oracle o bases de datos complejas:** Aunque son más potentes y escalables, su uso requeriría un backend y un servidor, lo que complicaría la publicación en la web. Para un juego ligero como este, JSON es suficiente y mucho más práctico.
2. **Java o compilación a web:** Aunque Java permite lógica similar, para que un juego Java funcione en un navegador se necesitan frameworks como WebAssembly o compiladores especiales, lo que complica la distribución. JavaScript permite que el juego se ejecute nativamente en cualquier navegador.

5. Diagrama de clases



6. Diagrama de secuencias: