**Historia de Usuario 1: Estructura Base del Juego**

**ID:** FE-01  
**Duración estimada:** 10 – 16 octubre (1 semana)

**Descripción:**  
Como desarrollador Front-End, quiero definir las clases principales (Game, Board, Ship, ScoreManager) siguiendo el diagrama UML, para establecer la arquitectura base del juego.

**Criterios de Aceptación:**

* Clases implementadas en archivos modulares.
* Atributos y métodos coinciden con el diagrama UML.
* Código documentado y probado en consola.

**Tareas:**

1. Crear clase Game (atributos gameState, shots; métodos startGame(), shoot(), checkVictory()).
2. Crear clase Board (grid[10][10], render(), updateCell()).
3. Crear clase Ship (name, size, hits, isSunk()).
4. Crear clase ScoreManager (saveScore(), getScores(), renderRanking()).
5. Verificar integración básica (Game instancia correctamente las demás clases).

**Criterios para cumplir:**  
 Clases funcionales sin errores.  
 Métodos probados mediante console.log.  
 Código modular subido al repositorio (feature/front-structure).

**Historia de Usuario 2: Interacción Inicial y Renderizado (DOM)**

**ID:** FE-02  
**Duración estimada:** 17 – 30 octubre (2 semanas)  
*(Dividida en dos bloques: Renderizado y Eventos DOM)*

**Descripción:**  
Como usuario, quiero visualizar el tablero de juego e interactuar con él mediante clics, para poder disparar y seguir el progreso de la partida.

**Criterios de Aceptación:**

* El tablero de 10x10 se genera dinámicamente en el DOM.
* Cada celda responde al clic del usuario y actualiza su estado.
* Se muestran mensajes de aciertos, hundimientos y victoria.
* No hay errores de consola ni visuales.

**Semana 2 (Renderizado del tablero – 17 a 23 oct)**

**Tareas:**

1. Implementar Board.render() para crear el tablero dinámicamente con DOM (document.createElement() o plantillas HTML).
2. Aplicar clases CSS base para celdas (.agua, .tocado, .hundido).
3. Insertar el tablero dentro del contenedor principal del HTML.
4. Verificar correcta visualización y estructura.

**Criterios para cumplir:**  
 Tablero visible en navegador.  
 Celdas diferenciadas por CSS.  
 No hay errores de consola.

**Semana 3 (Interacción DOM – 24 a 30 oct)**

**Tareas:**

1. Asignar eventos de clic (addEventListener('click')) a cada celda.
2. Conectar los clics con Game.shoot(row, col).
3. Actualizar visualmente las celdas con updateCell(row, col, state).
4. Mostrar mensajes en el DOM (por ejemplo, <div id="mensaje">Has acertado</div>).
5. Implementar detección de victoria (checkVictory()) y mensaje final.

**Criterios para cumplir:**  
 El tablero es totalmente interactivo.  
 Los disparos afectan el estado del tablero.  
 Mensajes de juego visibles en pantalla.  
 Código limpio y funcional (feature/board-dom).

**Historia de Usuario 3: Gestión de Puntuaciones y Ranking**

**ID:** FE-03  
**Duración estimada:** 31 octubre – 9 noviembre (última semana)

**Descripción:**  
Como jugador, quiero guardar mis puntuaciones y ver un ranking local con las mejores partidas para comparar resultados.

**Criterios de Aceptación:**

* ScoreManager guarda y recupera datos de localStorage.
* Se muestra un ranking ordenado en pantalla.
* El sistema se actualiza automáticamente al ganar una partida.

**Tareas:**

1. Implementar saveScore(player, shots) con localStorage.
2. Crear getScores() para recuperar los registros.
3. Desarrollar renderRanking() para mostrar los resultados en el DOM.
4. Conectar ScoreManager con checkVictory() en Game.
5. Aplicar estilos simples para la tabla de puntuaciones.

**Criterios para cumplir:**  
 Ranking visible en pantalla.  
 Datos persistentes en localStorage.  
 Integración completa con Game.  
 Commit final (feature/score-manager).