1. Roles y Responsabilidades del Equipo (Pygame Focus)

Rol	Nombre (Asignar)	Responsabilidades Clave en Pygame
Product Owner (PO)	[Nombre del PO]	Define el producto.
Scrum Master (SM)	[Nombre del SM]	Facilita el proceso.
Desarrollador A (Core Logic)	[Nombre del Dev A]	**Estructura Lógica:** Diseño del grafo del tablero, validaciones matemáticas de movimiento y reglas de finalización (independiente de gráficos).
Desarrollador B (Flujo y Estado)	[Nombre del Dev B]	**Motor del Juego:** Implementación de la clase `Juego` (gestor de estado), ejecución de movimientos, gestión del turno y lógica de victoria/reinicio.
Desarrollador C (Gráficos y Eventos)	[Nombre del Dev C]	**Renderizado:** Implementación de la ventana de Pygame, dibujo del tablero, manejo del bucle principal (`pygame.event.get()`) y *sprites* (fichas).

2. Planificación de Sprints con Tareas de Ingeniería (Pygame)

📌 Sprint 1: Configuración Gráfica y Modelo de Datos (Total: 8 Puntos)

- **Meta:** Ventana de Pygame funcional, tablero dibujado y fichas en su lugar.

HU	Tareas de Ingeniería (Implementación Python/Pygame)	Responsable
HU1	**Python:** Diseñar el objeto `TABLERO_DATA` (Grafo/Diccionario de 121 casillas) para el modelo lógico.	**Dev A**
HU1	**Pygame:** Inicializar la ventana, el *display* y manejar el **bucle principal de Pygame** (`while running:`).	**Dev C**
HU1	**Pygame:** Implementar la **Clase** `BoardRenderer`; dibujar la estrella usando polígonos y círculos en la ventana.	**Dev C**
HU2	**Python:** Implementar la **Clase** `Ficha`. Definir y ejecutar el método `Tablero.colocar_fichas_iniciales()`.	**Dev B**
HU2	**Pygame:** Crear la **Clase** `FichaSprite` y asociar coordenadas lógicas a coordenadas de píxeles (`x, y` de Pygame).	**Dev B**
HU3	**Pygame:** Implementar la función `render_turno()` usando `pygame.font` en una esquina de la pantalla.	**Dev C**
HU4	**Pygame:** Manejar el evento `MOUSEBUTTONDOWN`; implementar la función de **detección de colisión** de clic con el *sprite* de la ficha (`sprite.rect.collidepoint`).	**Dev C**

📌 **Sprint 2: Movimiento y Reglas Core (Total: 8 Puntos)**

- **Meta:** Implementar la lógica del movimiento de paso simple y todas las validaciones.

HU	Tareas de Ingeniería (Implementación Python)	Responsable
HU5	Implementar la lógica de validación de **`Validar_Movimiento_Adyacente(origen, destino)`** (verifica vecinos).	**Dev A**
HU6	Integrar las validaciones de **casilla ocupada** (HU6a) en la función `Validar_Movimiento`.	**Dev A**
HU6	Integrar la validación de **turno correcto** (`ficha.jugador == self.turno_actual`) en la función de movimiento (HU6c).	**Dev B**
HU6	**Pygame:** Implementar la función de **mensaje de advertencia** (e.g., texto temporal) para movimientos inválidos.	**Dev C**
HU7	Implementar el método **`Tablero.ejecutar_movimiento(origen, destino)`** (actualiza el modelo de datos).	**Dev B**
HU7	Actualizar la **posición de píxeles del *sprite*** de la ficha (`sprite.rect.x, sprite.rect.y`) para reflejar la nueva posición lógica.	**Dev B**
HU7	Implementar **`Juego.pasar_turno()`** y actualizar el *display* gráfico del turno (HU3b).	**Dev B**

📌 **Sprint 3: Finalización y Utilidades (Total: 13 Puntos)**

- **Meta:** Implementar las condiciones de victoria, empate y la funcionalidad de reinicio.

HU	Tareas de Ingeniería (Implementación Python)	Responsable
HU8	Implementar el método **`Juego.verificar_victoria()`** (detectar las 10 fichas en el área de Meta).	**Dev B**
HU8	**Pygame:** Implementar la función **`Mostrar_Mensaje_Ganador()`** como una superposición de texto grande y deshabilitar la entrada de mouse.	**Dev C**
HU9	Implementar el método **`Juego.reiniciar_partida()`** (reestablece variables de estado y llama a `colocar_fichas_iniciales`).	**Dev B**
HU9	**Pygame:** Implementar la lógica del **botón de reinicio** y su *event listener* de clic.	**Dev C**
HU10	Implementar el método **`Juego.buscar_movimientos_validos(jugador)`** (itera las piezas y verifica posibles movimientos).	**Dev A**
HU10	Implementar el método **`Juego.verificar_empate()`** (revisar si `movimientos_validos` para ambos es cero).	**Dev A**
HU10	**Pygame:** Implementar **`Mostrar_Mensaje_Empate()`** y deshabilitar los movimientos.	**Dev C**

Listado Consolidado de Todas las Tasks (Python/Pygame)

Sprint	HU	Tarea de Ingeniería (Python/Pygame)	Responsable
1	HU1	**Python:** Diseñar el objeto **TABLERO_DATA** (Grafo/Diccionario de 121 casillas) para el modelo lógico.	Dev A
1	HU1	**Pygame:** Inicializar la ventana, el *display* y manejar el **bucle principal de Pygame**.	Dev C
1	HU1	**Pygame:** Implementar la **Clase BoardRenderer**; dibujar la estrella usando polígonos y círculos.	Dev C
1	HU2	**Python:** Implementar la **Clase Ficha**. Definir y ejecutar Tablero.colocar_fichas_iniciales().	Dev B
1	HU2	**Pygame:** Crear la **Clase FichaSprite** y asociar coordenadas lógicas a coordenadas de píxeles.	Dev B
1	HU3	**Pygame:** Implementar la función **render_turno()** usando pygame.font.	Dev C
1	HU4	**Pygame:** Manejar el evento **MOUSEBUTTONDOWN**; implementar la función de **detección de colisión** de clic (sprite.rect.collidepoint).	Dev C
2	HU5	Implementar la lógica de validación de **Validar_Movimiento_Adyacente(origen, destino)**.	Dev A

2	HU6	Integrar las validaciones de **casilla ocupada** (HU6a) en la función `Validar_Movimiento`.	Dev A
2	HU6	Integrar la validación de **turno correcto** (`ficha.jugador == self.turno_actual`) en la función de movimiento (HU6c).	Dev B
2	HU6	**Pygame:** Implementar la función de **mensaje de advertencia** para movimientos inválidos.	Dev C
2	HU7	Implementar el método **`Tablero.ejecutar_movimiento(origen, destino)`** (actualiza el modelo de datos).	Dev B
2	HU7	Actualizar la **posición de píxeles del *sprite*** de la ficha (`sprite.rect.x, sprite.rect.y`).	Dev B
2	HU7	Implementar **`Juego.pasar_turno()`** y actualizar el *display* gráfico.	Dev B
3	HU8	Implementar el método **`Juego.verificar_victoria()`**.	Dev B
3	HU8	**Pygame:** Implementar la función **`Mostrar_Mensaje_Ganador()`** como una superposición de texto y deshabilitar la entrada.	Dev C
3	HU9	Implementar el método **`Juego.reiniciar_partida()`**.	Dev B
3	HU9	**Pygame:** Implementar la lógica del **botón de reinicio** y su *event listener* de clic.	Dev C

3	HU10	Implementar el método **`Juego.buscar_movimientos_validos(jugador)`** para detectar movimientos posibles.	Dev A
3	HU10	Implementar el método **`Juego.verificar_empate()`** (revisar si no hay movimientos válidos para ninguno).	Dev A
3	HU10	**Pygame:** Implementar **`Mostrar_Mensaje_Empate()`** y deshabilitar los movimientos.	Dev C