La ejecución comienza con el objeto en escena *NumbersGameView*. Éste instancia un controlador que genera los datos de cada ronda sucesiva y lleva el control de las respuestas.

Todas las vistas (Monobehaviours) heredan de *ComponentView*. Esto añade una capa de abstracción que permite añadir funcionalidad adicional a todos los componentes de vista. En este caso la capacidad de alterar el Alpha de un *CanvasGroup* heredando de *FadeableComponentView*, para provocar efectos de fadeIn/fadeOut.

Se decide utilizar animación por código para poder editar el comportamiento de las mismas sin necesidad de referencias en el editor. Al no permitirse assets o código externo, se simula una animación de FadeIn/FadeOut a través de corutinas.

Para las configuraciones de juego y referencias de prefabs se utiliza el ScriptableObject *InnovamatTestConfig*. Esto permite tener un lugar central desde donde acceder a las mismas, sin necesidad de añadir referencias de editor en cada vista o script que las necesite. Así mismo, se accede a ellas a través del *Singleton* *NumbersGameDataModel*.

Se utiliza un sistema de mensajería que permite a las vistas suscribirse a eventos y responder a ellos a través de delegates, desacoplando y modularizando enormemente todos los componentes ya que se comunican a través de un hub central sin necesidad de conocerse entre sí.

El método RoundController.OnCheckAnswer contiene condicionales por simplicidad. Si se busca que los comportamientos estén más compartimentalizados, podría implementarse una cadena de responsabilidad o un patrón estrategia.

La vista *NumbersGameView* contiene instrucciones para comenzar una nueva ronda como consecuencia de que Invoke solo puede ejecutarse sobre un Monobehaviour, limitándose éste a llamar al controller principal tras unos segundos de retardo.