Justificaciones

Usar un dictionary<string,string> como estructura de datos principal:

Para guardar los datos de las preguntas, respuestas y su relación entre ellas hemos utilizado un dictionary en GameManager.cs.

Hay varias formas de abordar el tratamiento de una lista de strings que se relacionen con unos números. Pese a que podíamos haber usado un array de strings básico donde cada string estuviera relacionado con su índice por la naturaleza del ejercicio, esto nos habría dado problemas en el momento de cambiar el ejercicio a otro tipo de relación entre objetos, como dos palabras relacionadas. Y, aunque al tratarse de data fija que no varía durante el gameplay una array sería más eficiente, hemos decidido no utilizar 2 arrays y usar un dictionary para mejorar la facilidad para leer el código y la rapidez en las búsquedas.

Instanciar Botones de respuesta:

Se ha decidido crear en cada ronda los botones de las respuestas para facilitar el cambiar la cantidad de estas, pudiendo añadir con facilidad nuevas mecánicas al respecto como variar el número en medio de la ejecución. La creación se da en CreateAnswersButtons() en UIManager.cs.

JSON Data:

Para facilitar el modificar los datos del ejercicio de forma sencilla o con scripts externos hemos decidido guardar nuestra base de datos en un json con una lista de objetos compuestos por respuestas y preguntas.

Controller-Vista:

Hemos intentado modular el código para separar la parte de controlador y tratamiento de datos, de la parte más puramente visual y relacionada con la UI. En un entorno más complejo podríamos haber necesitado un modelo, pero dada la simpleza del gameplay no se ha visto necesario aplicar un patrón más elaborado.

Coroutines:

Podríamos haber utilizado Async & Await para algunas partes del código, pero las coroutines nos parecen más adecuadas y fáciles de sincronizar temporalmente, pese a ser menos eficientes que Async & Await. Nos sentíamos más cómodos con ellas.