Technical Test

La prueba se tiene qué realizar en 24 hs, la entrega sólo incluye el proyecto completo de unity y un pdf con la justificación qué corresponda.

Se adjunta en la misma folder de drive qué este archivo, un proyecto de unity (para 2020.3 o superior) con lo necesario para cada ejercicio.

1) Se necesita desarrollar un sistema llamado ScreenNavigator qué controla la navegación entre las screens del juego, por ahora estas son Main Menu, Inventory y Shop, pero es muy probable qué en el juego final haya más.

   Reglas:

   - El ScreenNavigator conoce la Screen actual y puede activarla y desactivarla.
   - Este setea la screen actual y se encarga de hacer lo necesario para mostrarla.
   - Cada vez qué se cambia la screen debe printear cuál es la actual.
   - ScreenNavigator se hereda de Monobehaviour (qué es para poder instanciarlo en la escena) pero **no conoce ninguna clase de Unity.** *Aclaración: esto es solo para el ScreenNavigator.*
   - Para poder resolver el ejercicio se puede añadir cualquier otro script qué le sirva al sistema.

   Se necesita:

   1) Arreglar la UI para qué se adapte al formato 16:9 y qué también sea responsive a cualquier resolución horizontal.
   2) Crear el sistema de navegación de screens.
   3) Añadir Shop Screen y qué se pueda navegar de MainMenu a Shop y viceversa
   4) Añadir Inventory screen, qué se pueda navegar de Mainmenu a inventory y de shop a inventory, pero desde inventory solo se puede volver a la screen anterior.

2) Dado el código de la folder 2 se tiene qué refactorizar aplicando los estándares dados, respetando los principios SOLID y aplicando buenas prácticas de programación. Se puede renombrar, borrar y crear clases e interfaces según sea necesario, el código no va a ser ejecutado por lo tanto no hay qué aplicarlo en ninguna escena, solo es refactorizar.

   Standards
   - Todo debe estar escrito en Inglés.
   - No deben existir variables públicas, en caso de ser necesario usar una property.
   - Las variables privadas llevan el prefijo _ ,ejemplo: _player.
   - Respetar el orden jerárquico:
     1. Variables públicas.
     2. Variables privadas.
     3. Properties.
     4. Métodos.
   - Todas las clases e interfaces están contenidas en un namespace qué identifique y exprese bien de qué conjunto se trata .
   - Los nombres de las clases, namespaces, properties están escritos en CamelCase.
   - Las interfaces comienzan con I.
   - Todas las clases e interfaces deben estar en su propio archivo.
   - Quitar Namespaces innecesarios.

3) Con alguna Clase consumir la Clase LifeLocalRepository.cs y mostrar en pantalla su valor. Después añadir un nuevo LocalRepository<string> qué contenga el nombre del jugador y también mostrarlo en pantalla.

4) Observar el código de la folder 4 (este funciona perfectamente). Proponer una solución mejor y enumerar qué problemas tiene este código en un proyecto a largo plazo, no es necesario codear nada pero si justificar el por qué.

5) Compilar el juego para pc. (no hay qué enviar la build)

Ejercicio Bonus/Opcional

6) Escribir solo dos Unit Test qué pasen correctamente sobre las clases PlayerPresenter y PlayerView. (Se puede usar NSubstitute)