# Especificaciones y decisiones de Diseño

Solución de Jessica Peón al Tech Challenge de Jam City

[Especificaciones y decisiones de Diseño 1](#_Toc141106667)

[Diseño de la solución general: 2](#_Toc141106668)

[Uso de Assemblies: 2](#_Toc141106669)

[Visualización de Ejercicios del Challenge: 2](#_Toc141106670)

[Modo de uso: 3](#_Toc141106671)

[Derechos: 3](#_Toc141106672)

## Diseño de la solución general:

Para implementar lo requerido tenía la opción de generar clases con los roles que derivaran de Empleado, o tener una clase Empleado con el atributo Rol. Elegí la segunda opción, priorizando mantener un enfoque simple y directo, para no optimizar prematuramente manteniendo un diseño sencillo con pocas clases, dejando abierto a la cantidad de roles que se quiere manejar. Si los roles tuvieran funciones específicas, hubiera muchos detalles que generaran acoplamiento o hubiera funciones evidentes que implementar que podrían romper el principio de Open/Closed, hubiera ido por la primera opción. Y para un control más directo y organizado de los datos, los mantuve en Diccionarios.

## Uso de Assemblies:

Puse las carpetas Logic y Behaviour en Assemblies separados para que la creación de roles, seniorities y empleados se hiciera a través de la clase ManagerEmployees que ya se encarga de organizar en listas y chequear la existencia de los creados. De esa manera, con la privacidad *internal* en los constructores, los objetos solo pueden ser creados desde dentro de Logic, reduciendo el margen de error de permitir la creación de un objeto que se pueda perder.

## Visualización de Ejercicios del Challenge:

Los ejercicios fueron implementados en las funciones EmployeeSumary(), BaseSalarySumary(), IncrementSumary() y ApplyIncrementToAllBaseSalaries() que luego fueron evaluadas en UnitTest.

Si bien las funciones retornan un string con el texto solicitado, a nivel visual decidí utilizar la implementación para que los datos se mostraran de forma más amena y con cierta interacción.

Carga de datos:

La carga de datos importando un archivo, lo hice por JSON pero dejé el código abierto a la extensión haciendo la llamada a través de una interfaz, para que se pueda agregar la funcionalidad de escritura/lectura en otro formato como podría ser XML.

En la ruta *Assets/Data/*, hay dos archivos json para carga de datos:

* *data.json* con los mismos valores que el botón Default que son los dados en el ejemplo de la letra.
* *2023\_7\_24\_data.json* que es el json que guarda luego de aplicar el incremento a través del programa.

Decisiones de diseño visual:

Si bien los ejercicios eran sobre procesos de gestión de datos y podría haber priorizados cosas como hacer un login, creación de nuevos empleados, roles o seniorities de forma manual, al ser realizado en Unity preferí darle un encare más interactivo y visual pero que aun así fuera práctico para la visualización de los datos. Por eso mi objetivo fue cumplir los ejercicios, realizar pruebas unitarias sobre ellos, mantener un código limpio, escalable y de calidad, y a la vez hacer un estilo visual que se asemejara a un videojuego.

## Modo de uso:

Al entrar se ofrecen 2 opciones de carga: Default que carga un conjunto de datos a través del código, que son los mismos datos usados para los Tests y que se muestran en la descripción del Tech Challenge, y la opción Import que carga un JSON en la ruta que aparece.

Al confirmar, carga los roles mostrándolos como Islas en posiciones aleatorias formando un círculo y con colores random. A nivel visual se limita a 8 posiciones, pero si pudieran existir más roles, se pueden agregar más posiciones en la escena y que se agregarían al vector de posiciones en IslandManager.

Luego, al seleccionar cada isla se puede ver la información de rol y los seniorities de cada rol. Si bien a nivel de código los niveles de seniority son escalables, a nivel visual lo limité para poder mostrarlo como ranking.

Además, en la vista general de todas las islas, se puede presionar el botón Apply Increment para aplicar el incremento a todos los salarios. El guardado en json se da de forma automática y genera un archivo con la fecha del día.

## Derechos:

Las imágenes fueron hechas por mí y los audios son libre de Copyright.