MECALUX CHALLENGE

OBJETO DEL DOCUMENTO

El presente documento pretende describir el proceso de ideación y desarrollo del test propuesto por MECALUX como parte de su proceso de entrevista. En el documento se recogerán algunos datos de interés para el proceso, tales como, el tiempo invertido en la realización, decisiones de diseño tomadas, o cualquier dato que pueda ser de interés para Mecalux.

PREPARACIÓN DE LA PRUEBA

Para la elaboración de la prueba se han realizado la siguientes asunciones:

- **Se ha asumido como precondición *"words separated by a space"***. No se ha implementado ningún algoritmo para el tratamiento del texto de entrada, normalización de la cadena, o para detectar posibles caracteres separadores, como retorno de carro o signos de puntuación o dobles espacios.
- **Se ha asumido que el texto siempre llegaría contenido en un objeto del tipo "String"**. No se ha considerado la posibilidad de que el texto se tuviera que recoger a partir de un "stream".
- **Se ha optado por no usar convenciones "syntactic sugar"**. Tratando en todo momento usar, como convención, siempre tipos finales.
- **Se ha optado por usar nomenclatura camelCase:** para variables, parámetros de entrada y atributos de la clase.
- **Se ha usado para repositorio de la prueba GITHUB.** El repositorio puede convertirse en repositorio privado a petición de Mecalux si lo considera oportuno, una vez que haya finalizado la prueba.
- **Se ha usado Autofac como framework de Dependency Injection.** Es el framework que estoy comenzando a usar ahora en mi proyecto personal.

ELABORACIÓN DE LA PRUEBA

La elaboración de la prueba se ha realizado en tres iteraciones: *diseño y desarrollo* , *incorporación de Autofac* y *desarrollo de pruebas unitarias*.

DISEÑO Y DESARROLLO

Tiempo invertido: [1 hora y 45 minutos aprox.]

- **Diseño de la solución: 15 m. – 20 m. aprox.**
- **Desarrollo de la solución: 1h 30m aprox. (incluye documentación, estilado de código y pruebas de ejecución del código)**

En esta fase se ha realizado un diseño de la arquitectura a desarrollar basándome en la siguientes premisas:

- Nos basaremos en principios SOLID tal y como indica la prueba.
- El diseño debe ofrecer para los tipos de ordenación una solución escalable. Es decir, se deben poder incluir más métodos de ordenación sin "side-effects" sobre el código ya generado.

- El diseño debe ofrecer para los calculadores de estadísticas una solución escalable. Es decir, se deben poder incluir más calculadores de estadísticas sin "side-effects" sobre el código ya generado.
- Intentar máximo desacoplamiento entre clases basando el diseño en interfaces.
- Diseño prescindiendo de herencia.
- Diseño preparado para aplicar Dependency Injection.

INCORPORACIÓN DE AUTOFAC

Tiempo invertido: [45 minutos aprox.]

- **Incorporación de Autofac por NuGet y encapsulamiento en una clase singleton en el proyecto de modelo: 20 min. aprox.**
- **Nuevo componente de entrada TextHandler para crear un caso de uso con Full Dependency Injection desde el punto de entrada a la solución: 15 min. aprox.**
- **Revisión de código y sustitución de construcciones de instancia por llamadas al contenedor: 5 min – 10 min aprox.**

Todo el diseño de la solución se había preparado para trabajar con Dependency Injection de ahí que una vez incorporado se haya implementado otro punto de entrada: TextHandlerDI donde queda bien reflejada la automatización e inferencia de procesos de instanciación que resuelve, en este caso, AutoFac. Así como, el desacoplamiento entre tipos.

DESARROLLO DE PRUEBA UNITARIAS

Tiempo invertido: [45 minutos - 1hora aprox.]

- **Instalación de NUnit y revisión de documentación base : 15 m. – 20 m. aprox.**
- **Desarrollo de pruebas unitarias: 30m aprox.**

Posiblemente esta ha sido la iteración menos ágil. He realizado las pruebas unitarias con NUnit, tal y como indicaba la prueba, y aunque el proyecto de pruebas unitarias que he creado es muy simple. Al no haber usado NUnit hasta ahora, he tenido que instalar tres extensiones de visual studio y he revisado atributos y funcionamiento básico de NUnit.

Se han creado pruebas para los calculadores de estadísticas y para los "ordenadores" de texto.

CONCLUSIONES DE LA PRUEBA

- **El tiempo total invertido en la prueba ha sido de 3 horas aproximadamente.**
- **Tengo la impresión de que en algunas partes de la arquitectura puedo haber caído en "*Over-Engineering*" y quizá habiendo diseñado una capa de servios *IOrdererService* y *IStatisticsService* hubiera sido suficiente para el objetivo de la prueba.**