# Tarea 2 - Refactorización y Calidad de Código

## Contexto ficticio

GlobalSky Logistics es una aerolínea emergente fundada en 2022 con sede en Santiago, especializada en rutas regionales y carga ligera. A finales de 2024, el equipo ejecutivo decidió construir su propio sistema interno de gestión de vuelos y pasajeros para evitar dependencias con software externo. Fue entonces cuando contrataron a un desarrollador freelance con poca supervisión técnica.

El resultado fue un sistema funcional, pero con graves problemas de calidad de código. Como nuevo equipo de desarrollo, su tarea es refactorizar este sistema para hacerlo mantenible, robusto y profesional.

## Parte 1: Refactorización y Limpieza de Código (40%)

### Objetivo

Refactorizar el código existente en `main.py` siguiendo los principios de código limpio vistos en clase. El código actual implementa un sistema de gestión de vuelos con las siguientes funcionalidades:

- Gestión de vuelos (agregar, eliminar, ver)
- Gestión de pasajeros (agregar, eliminar, ver)
- Búsqueda de rutas (directas y con escalas)
- Gestión de horarios de vuelos

### Tareas

1. Analizar el código existente y entender su funcionamiento
2. Identificar problemas de calidad de código empleando la guía “Diseño a Nivel de Código” del curso, verificando que se cumpla lo siguiente:

    - Integridad conceptual (nomenclatura y estilo coherentes, PEP 8).
    - Ocultamiento de información (encapsulamiento adecuado).
    - Aumentar la cohesión y aplicar el Principio de Responsabilidad Única.
    - Reducir el acoplamiento y cumplir el Principio de Demeter.
    - Aplicar composición sobre herencia cuando corresponda.
    - Implementar los principios SOLID.
    - Detectar y eliminar *code smells* comunes (duplicated code, long method/class, feature envy, long parameter list, global variables, primitive obsession, etc.).

3. Diseñar y aplicar refactorizaciones que corrijan cada uno de los problemas diagnosticados, manteniendo la funcionalidad existente.
4. Registrar cada refactorización en un documento indicando el problema detectado, la sección afectada de código, la solución aplicada y el principio o *code smells* abordado.

### Entregables

- Código refactorizado (puede ser en el mismo archivo o en uno nuevo)
- Documento explicativo de los cambios realizados y por qué
- Instrucciones paso a paso para ejecutar el código

### Rúbrica de Evaluación

- Utilización de clases.
- Aplicar al menos 3 principios correctamente.
- Identificar y corregir al menos 3 *code smells*.

## Parte 2: Testing (30%)

### Objetivo

Crear un conjunto de tests unitarios robustos que verifiquen el correcto funcionamiento del código refactorizado.

### Tareas

5. Analizar los tests existentes
6. Modificar los tests para que funcionen con el código refactorizado
7. Asegurar que los tests cubran todas las funcionalidades relevantes
8. Implementar tests unitarios (pueden basarse en la ayudantía 3)
9. Verificar que los tests sean independientes y repetibles

### Entregables

- Archivo de tests actualizado
- Documento explicativo de los tests implementados
- Instrucciones para ejecutar los tests

### Rúbrica de Evaluación

- Contar con al menos 10 tests que cubran distintos escenarios.
- Todos los tests deben ejecutarse y pasar correctamente

## Parte 3: Linters (30%)

### Objetivo

Implementar y configurar un linter para verificar la calidad del código.

### Tareas

10. Instalar y configurar un linter (p.ej., pylint)
11. Configurar el linter para verificar al menos 10 tipos de errores de estilo
12. Crear un script que procese la salida del linter
13. Documentar los tipos de errores que se están verificando

### Entregables

- Configuración del linter
- Script de procesamiento de salida
- Documento explicativo de los errores verificados
- Instrucciones para ejecutar el linter

### Rúbrica de Evaluación

- Instalar, configurar y generar un reporte del linter correctamente.
- El linter debe analizar al menos 10 errores típicos de formato.
- Configurar el linter para excluir el análisis de al menos 1 error elegido y asegurar que la exclusión funciona.

## Formato de Entrega

La entrega debe incluir:

14. Código fuente refactorizado
15. Tests actualizados
16. Configuración del linter
17. Documentos explicativos:

    - `README.md o .pdf` con instrucciones generales
    - `REFACTOR.md o .pdf ` explicando los cambios realizados
    - `TESTING.md o .pdf ` explicando los tests
    - `LINTING.md o .pdf ` explicando la configuración del linter

Todos los archivos de la entrega, incluyendo el código y los documentos explicativos deben estar contenidos dentro de un único **archivo .zip** de entrega con **el nombre completo del estudiante**.

• Se aceptarán dudas sobre esta tarea hasta dos (2) días antes de la fecha de entrega mediante tickets al correo del MCD. Después de ese plazo no se responderán nuevas consultas.

• Se realizó un ejercicio similar a esta tarea en la Ayudantía 3 del curso. La sesión está grabada y el código está disponible en la plataforma Coursera en el módulo de la semana respectiva.