**Informe Tarea 1**

Luciano Gómez

Nicolas Meneses

<https://github.com/NicolasM2003/PTask1>

**1. Introducción**:

Este informe tiene como objetivo exponer las decisiones de diseño tomadas durante el desarrollo de una aplicación en C++ que permite gestionar evaluaciones educativas basadas en la Taxonomía de Bloom. Esto surge como respuesta a la Tarea 1 de la asignatura Paradigmas de Programación, en la cual se solicita la creación de un programa que genere evaluaciones categorizadas según distintos niveles de la Taxonomía de Bloom.

**2. Descripción de la Solución:**

**2.1 Modelo General**

El sistema se basa en el paradigma de programación orientado a objetos. Se diseñaron clases que representan tanto preguntas (ítems) como la evaluación en su conjunto. Las preguntas pueden ser de dos tipos: opción múltiple y verdadero/falso, cada una con estructura y lógica propia.

**2.2 Clases y Estructura**

* **Clase Item (clase base no instanciable directamente):**

Representa una pregunta general. Incluye atributos comunes como **id**, **texto**, **nivel**, **tipo**, **tiempo**, y **respuesta**. Contiene métodos para mostrar, actualizar y acceder a sus atributos. Es una clase base para otros tipos de ítems.

* **Clase ItemOpcionMultiple (hereda de Item)**:

Especializa la clase base para incluir alternativas **(a**, **b**, **c**, **d**) y sobrescribe el método **mostrar()** y **actualizar()** para adecuarse a las preguntas de opción múltiple.

* **Función tiempoPorNivel(int):**

Determina el tiempo estimado según el nivel taxonómico de Bloom. Por ejemplo, **Recordar** toma 1 minuto, **Crear** toma 6 minutos. Esta lógica es fundamental para la funcionalidad del sistema y se basa en la complejidad de cada nivel.

| **Nivel Taxonómico** | **Tiempo Estimado** |
| --- | --- |
| Recordar | 1 minuto |
| Entender | 2 minutos |
| Aplicar | 3 minutos |
| Analizar | 4 minutos |
| Evaluar | 5 minutos |
| Crear | 6 minutos |

* **Clase Evaluacion:** Administra una colección de ítems. Permite:
* Crear preguntas nuevas mediante entrada desde consola.
* Actualizar preguntas por ID.
* Eliminar ítems con confirmación.
* Consultar información de preguntas específicas.
* Filtrar preguntas por nivel taxonómico.
* Mostrar la evaluación completa junto al tiempo estimado total.
* **Función menu():**

Ofrece una interfaz por consola con todas las opciones requeridas en el enunciado: crear, actualizar, borrar, consultar, buscar por nivel y mostrar evaluación completa.

**2.3 Decisiones de Diseño:**

* **Uso de herencia y polimorfismo:**

Se optó por utilizar una clase base (**Item**) y especializaciones para distintos tipos de preguntas. Esto permite una extensión futura (por ejemplo, agregar tipos como analogías o gráficos) sin modificar la lógica principal del sistema.

* **Gestión dinámica de memoria:**

Se usaron punteros y **new** para crear ítems, asegurando que puedan convivir múltiples tipos en una misma colección (**vector<Item\*>**). Se incluye liberación de memoria en el destructor de **Evaluacion**.

* **Tiempo como métrica cuantitativa:**

La incorporación del cálculo del tiempo de evaluación apoya al profesor en la planificación y cumple con un requerimiento específico del enunciado.

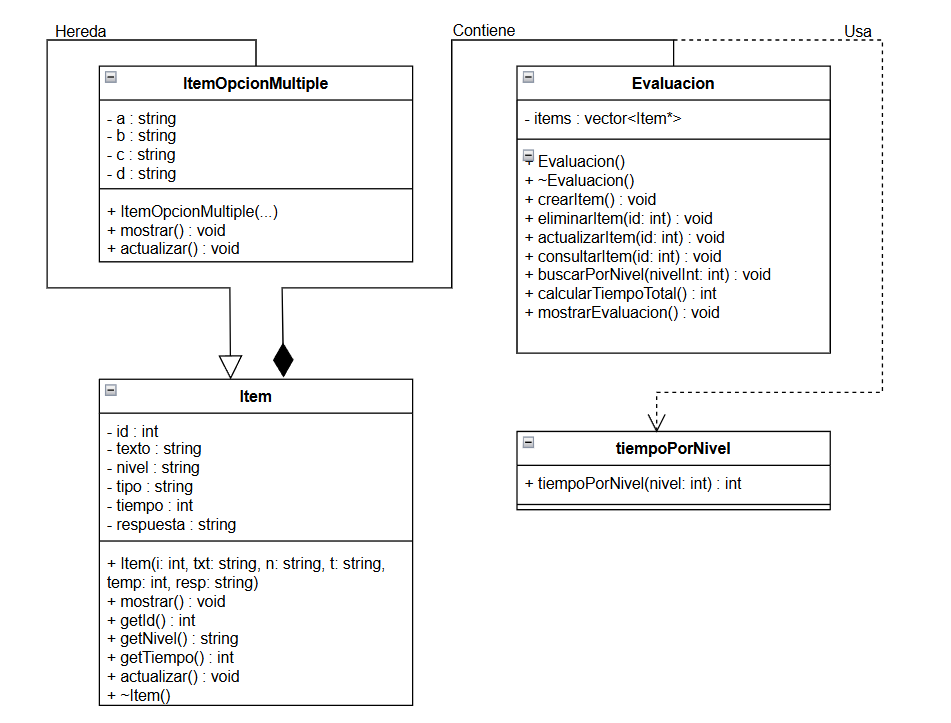
* **Menú amigable y modular:**

Se diseñó un menú claro, con navegación por opciones numéricas y confirmaciones para acciones críticas como eliminación.

* **Separación de preocupaciones:**

Cada clase y función tiene una única responsabilidad clara, lo que mejora la mantenibilidad y escalabilidad del programa.

**2.4 Diagrama de clases UML**



**Conclusión**:

El desarrollo de este sistema permitió consolidar conocimientos clave sobre programación orientada a objetos y diseño modular en C++. Se cumplieron todos los objetivos planteados en el enunciado: crear un sistema capaz de generar evaluaciones con ítems categorizados según la Taxonomía de Bloom, manipular dichos ítems desde un menú interactivo, y calcular el tiempo estimado según el nivel cognitivo asociado.

Además, se tomaron decisiones de diseño pensadas en la extensibilidad del sistema, como el uso de clases abstractas y punteros polimórficos. El código fue estructurado de forma legible, comentada y clara para facilitar su uso y corrección.

Reflexionando sobre el trabajo, no solo fortaleció habilidades técnicas, sino que también permitió valorar la importancia de una planificación cuidadosa y un diseño bien pensado al abordar problemas reales con soluciones eficientes y sostenibles.