Informe Programa Taxonomía de Bloom

Asignatura: Paradigmas de Programación
Alumno: Lucas Ernesto Maulén Riquelme
Carrera: Ingeniería Civil en Informática
Fecha: 11 de abril de 2025

Introducción

En el contexto del ramo de Paradigmas de Programación, se solicitó el desarrollo de un programa orientado a objetos en C++ que permita gestionar preguntas académicas clasificadas bajo la **Taxonomía de Bloom**. El sistema debía incluir funcionalidades como creación, modificación, eliminación y evaluación de preguntas, todo esto mediante una interfaz por consola.

Este informe presenta el análisis, diseño e implementación de la solución propuesta, junto con el diagrama de clases correspondiente. Se detalla también cómo se abordaron los requerimientos del enunciado y cómo se cumplieron los criterios exigidos por la pauta de corrección.

Objetivo general:
Desarrollar un sistema de evaluación modular, reutilizable y extensible, capaz de manejar preguntas según niveles taxonómicos, validando restricciones como años consecutivos y permitiendo generar evaluaciones realistas y automatizadas.

Objetivos específicos:

- Aplicar herencia y polimorfismo para representar distintos tipos de preguntas.
- Manejar niveles taxonómicos mediante enumeraciones.
- Diseñar clases coherentes con el problema y sus relaciones.
- Implementar funciones de búsqueda, filtrado y evaluación.
- Validar correctamente entradas del usuario.

Diseño del sistema

La solución fue modelada utilizando un enfoque orientado a objetos. Se definieron clases específicas para representar preguntas, bancos de preguntas y evaluaciones. Además, se aplicó herencia para distinguir entre tipos de preguntas, manteniendo una arquitectura limpia y flexible.

A continuación, se describe el diagrama de clases desarrollado:

Clases principales:

1. <<abstract>> Pregunta:
Clase base que representa cualquier pregunta. No puede instanciarse directamente.
Contiene atributos como id, enunciado, nivel (asociado a NivelTaxonomico), anio y tiempoEstimado.
Define los métodos mostrar() y getTipo() como virtuales puros.

2. PreguntaOpcionMultiple:
Subclase de Pregunta que permite definir preguntas con alternativas.
Agrega los atributos opciones (vector de strings) y respuestaCorrecta.

3. PreguntaVerdaderoFalso:
Subclase de Pregunta para preguntas de verdadero/falso.
Contiene esVerdadero y justificacion como atributos adicionales.

4. NivelTaxonomico (<>):
Enumera los niveles de la Taxonomía de Bloom: RECORDAR, ENTENDER, APLICAR, ANALIZAR, EVALUAR, CREAR.
Se utiliza como atributo en la clase Pregunta para clasificar cognitivamente cada ítem.

5. BancoPreguntas:
Clase que almacena y gestiona un conjunto de preguntas (vector<Pregunta*>).
Permite agregar, eliminar, buscar por ID y filtrar por nivel.

6. Evaluacion:
Representa una evaluación académica.

Contiene un conjunto de preguntas, un título, el año en que se aplica y su ponderación.
Permite agregar preguntas bajo ciertas restricciones: no repetir preguntas del año anterior, mantener nivel taxonómico, evitar duplicados y calcular el tiempo total estimado de resolución.

Relaciones entre clases:

- PreguntaOpcionMultiple y PreguntaVerdaderoFalso heredan de Pregunta.
- BancoPreguntas y Evaluacion contienen vector<Pregunta*> usando polimorfismo.
- Pregunta utiliza NivelTaxonomico como atributo de clasificación.
- Todas las clases interactúan respetando el principio de responsabilidad única.

Justificación del diseño

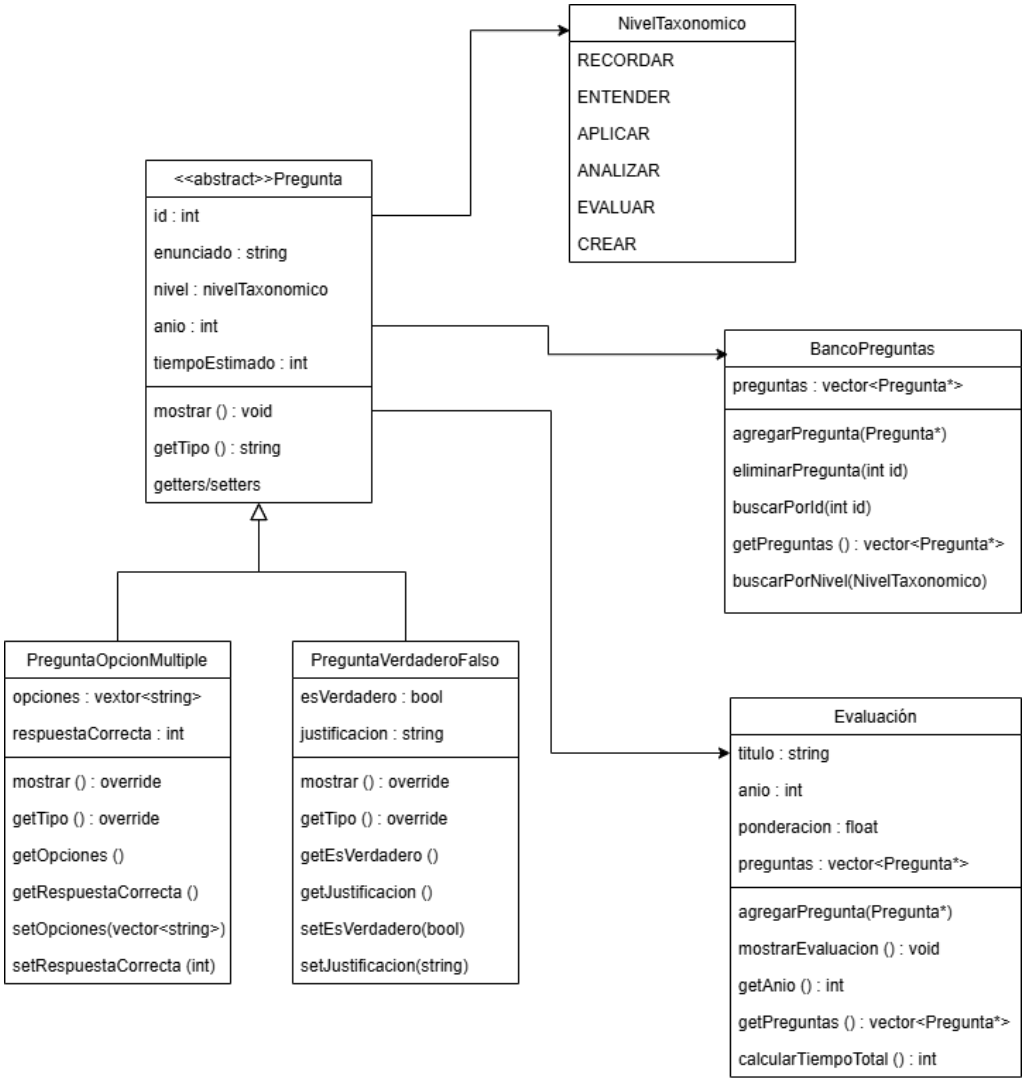
La elección de una clase base abstracta (Pregunta) permitió definir una interfaz común que garantiza la extensibilidad del sistema. Gracias a la herencia, fue posible crear tipos específicos de preguntas con sus propias implementaciones, reutilizando el código general.

Se utilizó un vector de punteros (vector<Pregunta*>) tanto en BancoPreguntas como en Evaluacion, lo que permite almacenar cualquier tipo de pregunta derivada gracias al polimorfismo. Esto simplifica el manejo de preguntas y hace posible trabajar con ellas sin conocer su tipo exacto.

El uso de NivelTaxonomico como enumeración ayuda a evitar errores de tipeo y asegura consistencia al clasificar las preguntas, cumpliendo un requerimiento clave del enunciado.

La generación de evaluaciones incluye controles lógicos que impiden incluir preguntas del año anterior, respetando la restricción académica definida en la pauta.

Diagrama de Clases



Pruebas realizadas

Se realizaron múltiples pruebas para validar el comportamiento del sistema:

- Prueba de creación de preguntas y asociación a nivel taxonómico.
- Verificación de restricciones de año consecutivo en evaluaciones.

- Validación de entradas inválidas (IDs duplicados, nivel inexistente, etc.).
- Prueba de evaluación con tiempo total correctamente calculado.
- Validación del funcionamiento del menú en consola de forma fluida.

Conclusión

El sistema desarrollado cumple con todos los requerimientos funcionales solicitados. Se aplicaron correctamente los conceptos de clases, herencia, encapsulamiento y polimorfismo. El diseño propuesto refleja una solución integral y óptima al problema planteado, con una implementación clara, mantenible y alineada al paradigma de programación orientada a objetos.

El trabajo también incluyó pruebas, validaciones y diseño visual (diagrama de clases) conforme a los criterios de evaluación definidos en la pauta, entregando un sistema sólido y listo para ser extendido o utilizado en nuevos contextos.