Instituto Tecnológico de Costa Rica (ITCR)
Lenguajes de programación
I Semestre 2023
Tema: Programación orientada a objetos con Java

Tarea programada #2

1. Objetivo del presente trabajo es poner en práctica el conocimiento adquirido en el curso sobre **programación orientada a objetos** y diseñar e implementar funcionalidad utilizando el lenguaje de programación **Java y el framework Spring Boot**.

Este objetivo está estrechamente relacionado con el objetivo específico del curso de utilizar un lenguaje ejemplo para cada uno de los cuatro paradigmas principales de programación, en este caso el paradigma de orientación a objetos (OO).

El proyecto permitirá a los estudiantes poner en práctica metodologías de análisis y diseño de aplicaciones orientadas a objetos y su implementación en Java. La aplicación deberá persistir los datos en una base de datos relacional de elección de los estudiantes utilizando **Hibernate** (una herramienta de mapeo objeto-relacional).

2. Descripción de trabajo:

En el presente trabajo deberán diseñar e implementar un sistema utilizando el paradigma de orientación a objetos (OO) y el patrón de diseño **Modelo-Vista-Controlador (MVC)**.

El proyecto consiste en desarrollar una aplicación web que utilice OO y bases de datos como mecanismo de persistencia. La aplicación a desarrollar debe permitir manejar imágenes asociadas a especies.

Descripción de la aplicación:

- El dato principal que maneja la aplicación son imágenes (image).
- Las imágenes poseen los siguientes metadatos asociados:
    - description: (obligatorio) Una descripción asociada a cada imagen en formato de texto con un máximo de 500 caracteres.
    - date: (obligatorio) fecha de creación de la imagen.

- keywords: (obligatorio) Un conjunto de palabras clave estandarizadas, es decir, el usuario debe seleccionar estas de una lista de palabras predefinidas. Un usuario no debe de poder escribir una palabra clave en texto libre.

- author (de tipo Person): (obligatorio) las imágenes deben estar asociadas a un autor. El autor es siempre una persona y posee los siguientes atributos: id, name, lastName, country, phone, e email.

- owner: (obligatorio) el dueño de una imagen puede ser una persona o una institución. Las instituciones (institution) tienen los siguientes atributos: id, name, country, phone, email, y webSite.

- license: (obligatorio) licencia de uso de las imágenes. Para el proyecto se utilizarán solo licencias Creative Commons (CC). Se debe crear una lista con las opciones de licencia que maneja CC. Utilicen un Enum para definir los tipos de licencias.

- taxon: (obligatorio) lista de taxones asociados a la imagen. Cada taxon puede estar en cualquiera de los niveles de la jerarquía taxonómica y una imagen puede estar asociada a cero o más taxones de cualquier nivel en la jerarquía.

  Los niveles obligatorios de la jerarquía taxonómica se definen para la aplicación como: reino, filo, clase, orden, familia, género y/o una especie.

  Todos los taxones tienen los siguientes datos asociados: codigo (id), nombre (scientific_name), autor (author), año de publicación del taxon (publication_year), y ancestro (taxon_ancestor_id). La figura uno muestra un ejemplo de nombres de taxones en varios niveles de la jerarquía.

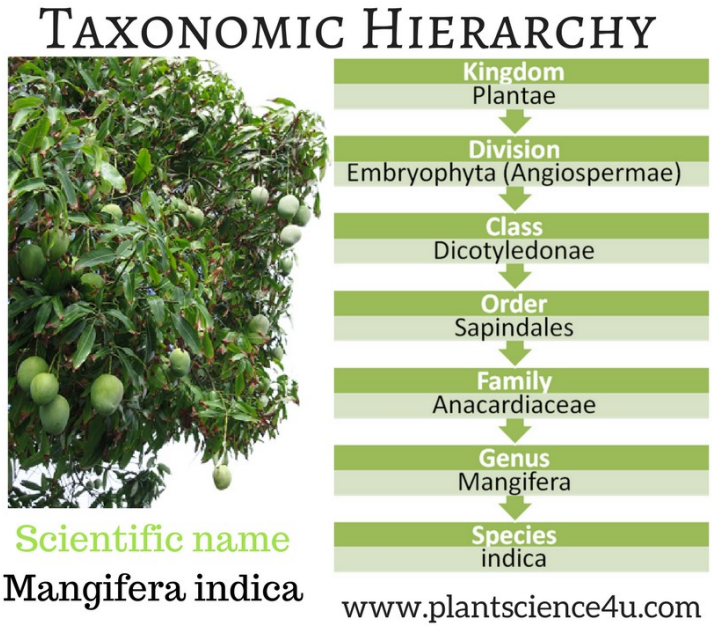

Figura 1. Jerarquía taxonómica del reino Plantae (imagen ASA Academy)

- Funcionalidad
  - La funcionalidad debe utilizar OO y estar implementada de forma fiel al modelo UML, es decir, si el modelo presenta una clase Person la funcionalidad para utilizar datos de personas debe instanciar objetos de esa clase.
  - Las imágenes no se deben almacenar directamente en la base de datos, deben almacenarse en el sistema de archivos y tener una referencia a estas en los metadatos de la imagen.
  - Los taxones forman una jerarquía que se estructura para organizar el conocimiento de los seres vivos del reino Plantae. Hay herencia en esa representación (es un buen ejemplo de representación de conocimiento). La figura 1 muestra un ejemplo de la jerarquía taxonómica de la especie *Manguifera indica*. **La aplicación debe verificar, cuando se crea o modifica un taxon_ancestor_id,** que el taxon padre corresponda a un nivel válido, es decir, los géneros deben tener como ancestro solo taxones del nivel de familia y las familias deben tener como ancestro solo taxones a nivel de orden y así sucesivamente.
  - La interfaz web de usuario debe contar con dos funcionalidades mínimas:

    1. Captura de datos:
       - Los datos asociados a keywords, person, institution y license pueden ser definidos directamente en la base de datos. Pero cuando en la aplicación se usen los datos de una clase, esta debe ser instanciada en un objeto.
       - Los datos de taxones debe ser mantenidos por la aplicación, es decir, se debe poder incluir, consultar, modificar y borrar taxones de cualquier nivel obligatorio de la jerarquía taxonómica.
       - Los datos de las imágenes deben ser mantenidos por la aplicación, es decir, se deben poder incluir, consultar, modificar y borrar imágenes.
    2. Búsqueda y visualización de datos:

        - El sistema despliega un campo de texto en donde el usuario puede digitar el nombre de una persona, institución, un taxón o una palabra clave.
        - El sistema despliega una lista con las imágenes que cumplen con es criterio de búsqueda (se debe paginar la lista).
        - Cuando un usuario hace clic sobre una imagen el sistema debe desplegar esta con todos sus datos asociados. Un ejemplo de funcionalidad lo pueden ver en https://images.ala.org.au/

- Adicionalmente:
    - El diseño UML debe utilizar todas las herramientas de la OO para modelar los requerimientos del sistema.
    - Utilizar la siguiente tecnología: **Java, Spring Boot, Hibernate** (una herramienta de mapeo objeto-relacional) y una **base de datos relacional de su elección**.
    - Deben usar modificadores de visibilidad de los atributos y métodos de las clases.
    - Deben definir al menos una variable y método de clase (static).
    - Deben implementar herencia al menos con:
        - Taxon y muchas sub-clases que heredan todas de Taxon.
        - Owner debe tener dos clases asociadas: Person e Institution.
    - La clase Taxon debe ser abstracta.
    - Se debe definir una interface que aplique a **todas** las clases que implementen.
    - La implementación debe ser fiel al modelo en UML.
    - Deben persistir los datos en una base de datos de su elección.
    - Deben programar una aplicación web.
    - Deben usar el patrón de diseño MVC.
- El grupo debe preparar los datos de prueba necesarios para ejecutar la aplicación.

3. Documentación

El trabajo debe ir acompañado de documentación externa e interna.

Documentación Interna: Para la documentación interna, se deberán incluir comentarios descriptivos para cada clase, atributos y métodos implementados con sus respectivas entradas, salidas y restricciones.

Documentación Externa:

La documentación externa deberá incluir:

- Tabla de contenidos.
- Descripción del problema.
- Descripción de la solución
- Diagrama de casos de uso (utilizando UML)
- Descripción **completa** de cinco casos de uso.
- **Diagrama de clases (utilizando UML).**
- Diagrama Entidad-Relación (utilizando UML)
- Análisis de resultados: objetivos alcanzados, objetivos no alcanzados, y razones por las cuales no se alcanzaron los objetivos (en caso de haberlas).
- Conclusión personal (una de cada estudiante).
- La documentación no debe tener errores de redacción u ortografía.
- Para la presentación deben incluir 20 registros de imágenes ilustrativos para realizar las pruebas de la aplicación.

4. Datos administrativos

- El trabajo se puede realizar de forma individual o en parejas.
- La aplicación y documentación se debe entregar en un archivo zip por medio del TecDigital y se van a programar citas para revisiones.

5. Rúbrica para la evaluación

La rúbrica se compartirá en un archivo aparte en el TecDigital.