Instituto Tecnológico de Costa Rica

Centro académico de Alajuela

IC-4700. Lenguajes de programación

Semestre I – 2023

Tarea Programada #2

Programación orientada a objetos con Java

Prof. María Mora Cross.

Estudiantes:

Brandon Retana Chacón.

Carné: 2021121141

Correo: sthuar@estudiantec.cr

Kevin Cubillo Chacón

Carné: 2021123138

Correo: kevincubillo@estudiantec.cr

Fecha de Entrega: 01/05/2023

Tabla de Contenido

Descripción del problema 3
Descripción de la solución 3
Diagrama de casos de uso 4
Descripción de casos de usos 4
Diagrama de clases 4
Diagrama Entidad-Relación 4
Análisis de resultados 4
  Objetivos alcanzados: 4
  Objetivos no alcanzados: 5
Conclusiones personales 5

Descripción del problema

Este proyecto busca implementar una aplicación web que permita la gestión de imágenes asociadas a especies, utilizando el paradigma de orientación a objetos (OO) y el patrón de diseño Modelo-Vista-Controlador (MVC), con el lenguaje de programación Java y el framework Spring Boot. La aplicación debe permitir la persistencia de datos en una base de datos relacional utilizando la herramienta de mapeo objeto-relacional Hibernate.

La aplicación debe manejar imágenes sobre biodiversidad, las cuales tienen metadatos obligatorios asociados, como una descripción, fecha de creación, un conjunto de palabras clave, el autor de la imagen entre otros. Además, la aplicación debe manejar la licencia de uso de las imágenes, utilizando solo licencias Creative Commons (CC) y una lista de opciones de licencia definidas mediante el uso de un Enum. También se debe permitir la asociación de uno o más taxones a una imagen, utilizando una lista de taxones asociados que pueden estar en cualquier nivel de la jerarquía taxonómica, como reino, filo, clase, orden, familia, género y/o especie.

Al desarrollar este proyecto se aplican metodologías de análisis y diseño de aplicaciones orientadas a objetos y su implementación en Java, utilizando tecnologías como Spring Boot e Hibernate para la persistencia de datos en una base de datos relacional, esto permite que los estudiantes pongan en práctica sus conocimientos en Programación orientada a objetos y diseño de software.

Descripción de la solución

La solución a este problema implica el diseño e implementación de una aplicación web que permita manejar imágenes asociadas a taxones específicos, utilizando el paradigma de programación orientada a objetos y el patrón de diseño Modelo-Vista-Controlador (MVC). Para ello se utilizará el lenguaje de programación Java y el framework Spring Boot, asi como Angular para la implementación de la interfaz gráfica.

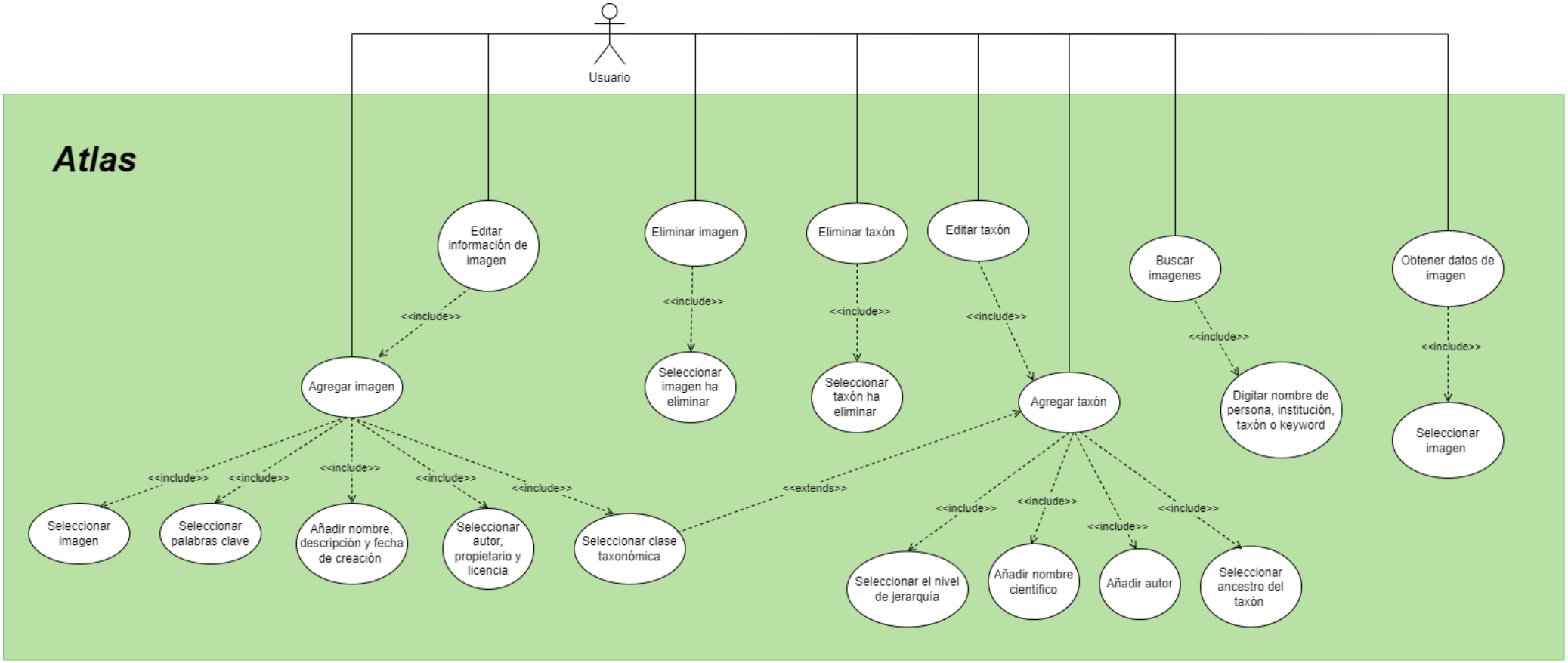
La aplicación web permitirá a los usuarios agregar, editar, eliminar y buscar imágenes, así como visualizar su información detallada y sus metadatos. Para persistir los datos se utilizará Hibernate como herramienta de mapeo objeto-relacional, que permite representar objetos de Java en tablas de una base de datos relacional.

La solución también contempla la implementación de la jerarquía taxonómica de los seres vivos, permitiendo que las imágenes estén asociadas a uno o varios

taxones. Para ello se definirán los datos asociados a cada taxón, como su código, nombre científico, autor, año de publicación y ancestro.

Además, se implementará una lista de licencias de uso de imágenes, utilizando el Enum de Java para definir los tipos de licencias. Se definirán también palabras clave estandarizadas para las imágenes, que los usuarios deberán seleccionar de una lista predefinida.

Diagrama de casos de uso



Descripción de casos de usos

1.

Nombre: Agregar Imagen

Actores: Usuario

Precondiciones:

- El usuario debe tener conexión a la aplicación.

- El sistema de archivos debe estar disponible.

Flujo principal:

1. El usuario selecciona la opción "Crear".

2. El sistema muestra el formulario para ingresar los metadatos de la imagen.

3. El usuario carga la imagen en el sistema de archivos.

4. El usuario completa el formulario, ingresando la descripción, fecha de creación, palabras clave, autor, propietario, licencia y taxón asociado.

5. El sistema crea una referencia en la base de datos, almacenando los metadatos de la imagen y la ubicación en el sistema de archivos.

Flujos alternativos:

3.1. El usuario carga una imagen con un formato no válido.

- El sistema muestra un mensaje de error indicando que el formato de imagen no es válido.

4.1. El usuario no ingresa los campos obligatorios del formulario.

- El sistema no envía el formulario y no se completa la transacción.

4.2. No se han agregado taxones al sistema.

- El sistema no envía el formulario y no se completa la transacción.

Postcondiciones:

- La imagen se ha almacenado en el sistema de archivos.

- Se ha creado una referencia en la base de datos para la imagen.

- El usuario puede acceder a la imagen en la aplicación y visualizar su información.

2.

Nombre: Modificar Imagen

Actores: Usuario

Precondiciones:

- El usuario debe tener conexión a la aplicación.

- La imagen a modificar debe existir en la base de datos y estar disponible en el sistema de archivos.

Flujo principal:

1. El usuario ingresa al backoffice.

2. El usuario selecciona la imagen que desea modificar.

3. El sistema muestra los metadatos actuales de la imagen.

4. El usuario edita los campos que desea modificar, como la descripción, fecha de creación, palabras clave, autor, propietario, licencia y taxón asociado.

5. El usuario confirma los cambios.

6. El sistema actualiza la referencia en la base de datos y en el sistema de archivos con los nuevos metadatos.

Flujos alternativos:

4.1. El usuario no desea realizar cambios.

- El usuario cancela la operación y se retorna a la página anterior.

5.1. El usuario no ha ingresado información válida en los campos modificados.

- El sistema no completa la transacción.

Postcondiciones:

- La imagen se ha actualizado en la base de datos y en el sistema de archivos con los nuevos metadatos.

- El usuario puede acceder a la imagen en la aplicación y visualizar su información actualizada.

3.

Nombre: Borrar Imagen

Actores: Usuario

Precondiciones:

- El usuario debe tener conexión a la aplicación.

- La imagen que se desea borrar debe existir en la base de datos.

Flujo principal:

1. El usuario ingresa al backoffice.

2. El usuario selecciona la imagen que desea borrar.

3. El sistema elimina la referencia de la imagen en la base de datos y la imagen del sistema de archivos, así como sus metadatos asociados.

Flujos alternativos:

Ninguno.

Postcondiciones:

- La imagen se ha eliminado del sistema de archivos y la base de datos, así como sus metadatos asociados.

- El usuario ya no puede acceder a la imagen en la aplicación.

4.

Nombre: Buscar Imágenes

Actores: Usuario

Precondiciones:

- El usuario debe tener conexión a la aplicación.

- Debe haber imágenes almacenadas en la base de datos.

Flujo principal:

1. El usuario digita las palabras claves en el campo "Buscar".

2. El usuario presiona el icono de la “lupa”.

3. El sistema realiza una búsqueda en la base de datos y muestra una lista paginada de imágenes que coinciden con el término de búsqueda.

5. El usuario puede hacer clic en una imagen para ver más detalles.

Flujos alternativos:

3.1. No se encuentran imágenes que coincidan con el término de búsqueda.

- El sistema muestra un mensaje indicando que no se han encontrado imágenes que coincidan con el término de búsqueda.

Postcondiciones:

- El usuario puede ver una lista paginada de imágenes que coinciden con el término de búsqueda.

- El usuario puede hacer clic en una imagen para ver más detalles.

5.

Nombre: Modificar Taxón

Actores: Usuario

Precondiciones:

- El usuario debe tener conexión a la aplicación.

- Debe existir el taxón que se desea modificar en la base de datos.

Flujo principal:

1. El usuario ingresa al backoffice

2. El sistema muestra una lista de los taxones existentes en la aplicación.

3. El usuario selecciona el taxón que desea modificar.

4. El sistema muestra los detalles del taxón seleccionado.

5. El usuario edita los campos de nombre científico, autor, año de publicación y ancestro.

6. El sistema verifica que el ancestro seleccionado corresponda a un nivel válido en la jerarquía taxonómica.

7. El usuario confirma los cambios.

8. El sistema actualiza los datos del taxón en la base de datos y en todas las imágenes asociadas al taxón modificado.

Flujos alternativos:

5.1. El usuario no desea modificar el taxón.

- El usuario cancela la operación y vuelve al menú principal.

6.1. El usuario selecciona un ancestro que no corresponde a un nivel válido en la jerarquía taxonómica.

- El sistema muestra un mensaje de error indicando que el ancestro seleccionado no es válido.

- El usuario debe seleccionar un ancestro válido.

7.1. El usuario no confirma los cambios.

- El sistema no realiza los cambios y vuelve a mostrar los detalles del taxón seleccionado.

Postcondiciones:

- Los datos del taxón se han actualizado en la base de datos.

- Todas las imágenes asociadas al taxón modificado se han actualizado con los nuevos datos del taxón.

Diagrama de clases

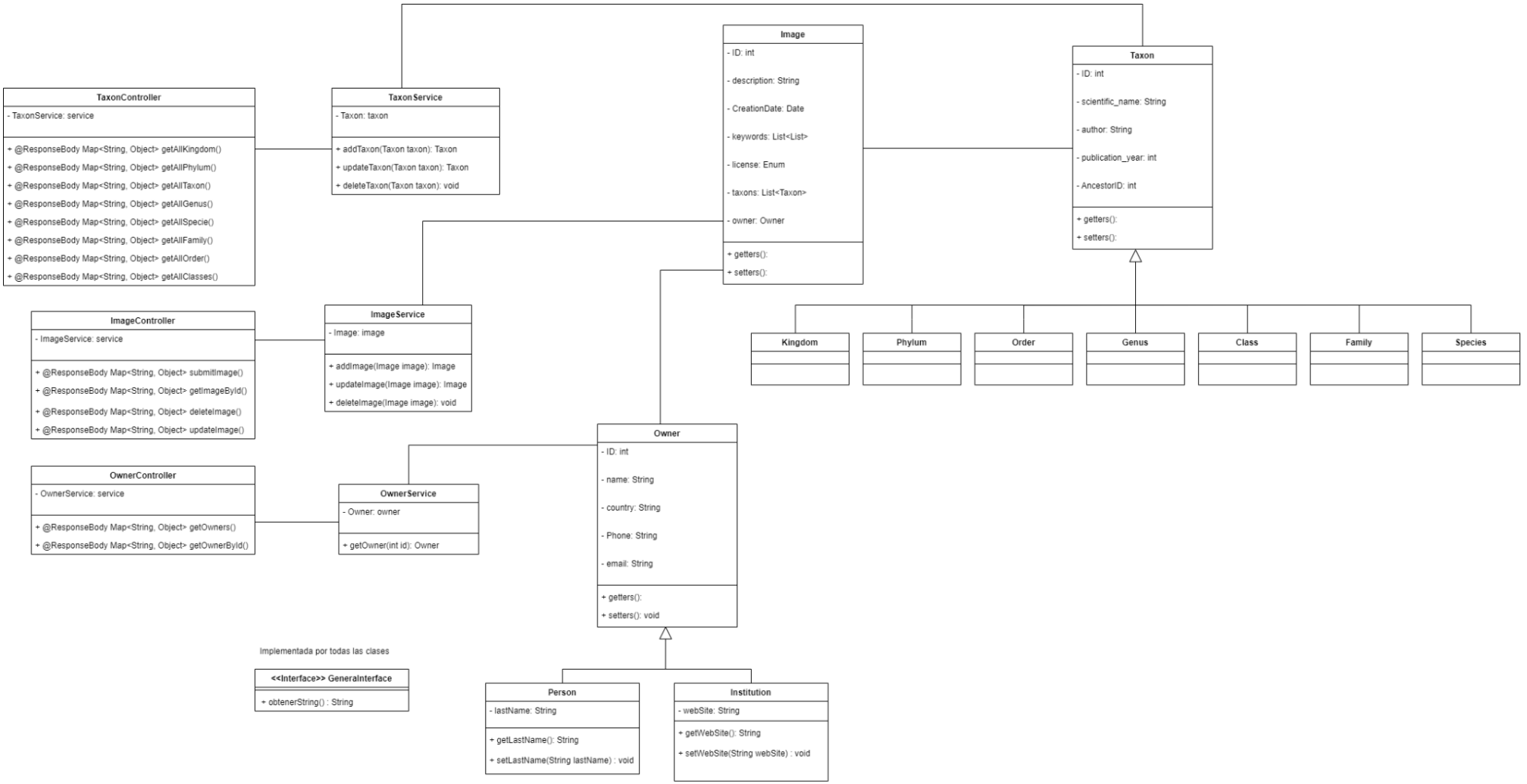
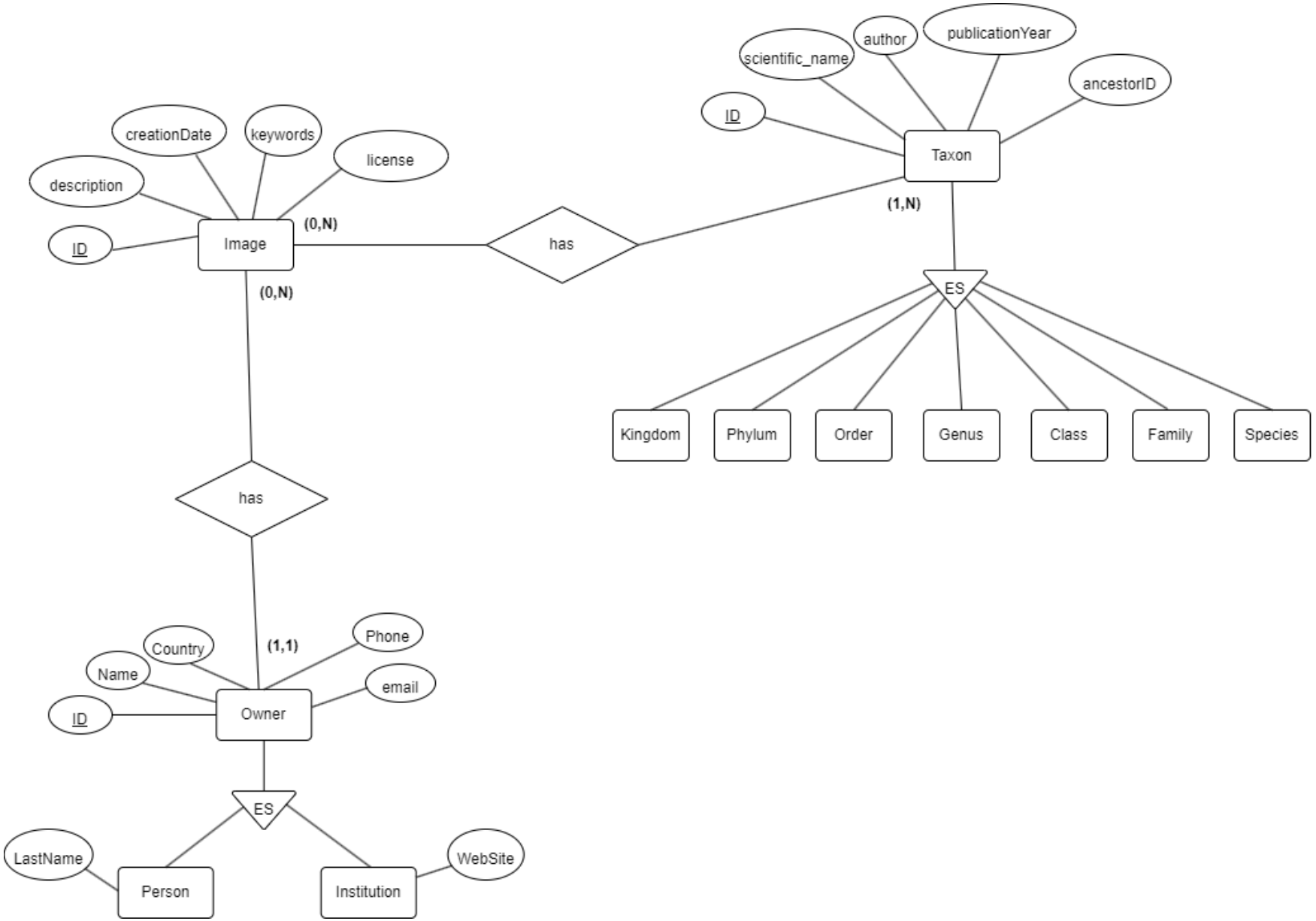


Diagrama Entidad-Relación



Análisis de resultados

Objetivos alcanzados:

- La aplicación implementada utiliza el paradigma de programación orientada a objetos (OO) y se apegó fielmente al modelo UML, de tal manera que las clases que representan en el modelo son objetos en la aplicación.

- Las imágenes no se almacenan directamente en la base de datos, sino en el sistema de archivos, y se hace referencia a ellas a través de los metadatos de la imagen.

- Se estructuró una jerarquía taxonómica de los seres vivos, con herencia en la representación, y se verificó que el taxón padre corresponda a un nivel válido cuando se crea o modifica un taxon_ancestor_id.

- La interfaz web de usuario cuenta con funcionalidades para la captura de datos y la búsqueda y visualización de estos.

- En la búsqueda y visualización de datos, el usuario puede digitar el nombre de una persona, institución, un taxón o una palabra clave, y se despliega una lista paginada de las imágenes que cumplen con ese criterio de búsqueda.

- Se utilizaron Java, Spring Boot, Hibernate y una base de datos relacional (MySQL).

- Se definieron modificadores de visibilidad de los atributos y métodos de las clases, al menos una variable y método de clase (static) y herencia en las clases Taxon y Owner.

- La clase Taxon se definió como abstracta y se definió una interfaz que aplica a todas las clases que la implementan.

- Se programó la aplicación web utilizando el patrón de diseño MVC.

- Se prepararon los datos de prueba necesarios para ejecutar la aplicación.

Objetivos no alcanzados:

- No hay.

Conclusiones personales:

Brandon

Fue un proyecto interesante que me ayude a entender fundamentos de la programación orientada a objetos que antes no entendía, además me permitió adentrarme en el mundo de las taxonomías de las plantas, lo cual también fue interesante.

Kevin

El diseño e implementación de este proyecto de software me permitió comprender conceptos de programación orientada a objetos, como herencia, polimorfismo y encapsulamiento. Además de adquirir nuevas habilidades en el manejo de herramientas como spring boot para la creación de aplicaciones web siguiendo el patrón de diseño Modelo vista controlador (MVC)