Universidad de Málaga

ETSI Informática

Diseño orientado a objetos de un refugio de animales

Modelado y Diseño del Software (2024–25)

Daniil Gumeniuk
Angel Bayon Pazos
Diego Sicre Cortizo
Pablo Ortega Serapio
Angel Nicolás Escaño López
Francisco Javier Jordá Garay
Janine Bernadeth Olegario Laguit

Grupo 1.1

Diciembre 2024

Índice

Índice de figuras

Resumen

Esta práctica aborda el diseño e implementación de un sistema orientado a objetos para gestionar un refugio de animales utilizando Java y conceptos de diseño orientado a objetos vistos en el Tema 5.

El objetivo principal es analizar las posibles estrategias de diseño que permitan implementar este modelo, abordando desafíos como la necesidad de que un mismo socio pueda desempeñar múltiples roles simultáneamente. Se discute por qué las clases descritas inicialmente no pueden ser implementadas directamente en Java y se propone una posible solución mediante técnicas como composición, interfaces, y herencia múltiple simulada para garantizar la consistencia del sistema.

Finalmente, la solución propuesta acompañada de un diagrama de diseño que ilustra la arquitectura del sistema, muestra la reutilización de métodos, la integridad de los datos y la flexibilidad necesaria para adaptarse a los requerimientos del modelo conceptual.

1. Diseño del código de andamiaje

1.1. Introducción

En este apartado se discutirán las diferentes formas de diseñar el código de andamiaje necesario para implementar el modelo del refugio. Se incluirá un diagrama de diseño que represente las relaciones entre las entidades principales del sistema y la implementación en Java. Finalmente, se justificará la estrategia elegida.

1.2. Formas de diseñar el código de andamiaje

A continuación, se presentan las estrategias principales consideradas para implementar el código de andamiaje para las clases asociadas al modelo:

1.2.1. Estrategia 1: Uso de listas

- **Descripción:** Esta estrategia utiliza listas para manejar las asociaciones 1:M.
- **Ventajas:**
  - Implementación sencilla.
  - Flexibilidad para manejar entidades dinámicas.
- **Desventajas:**
  - Requiere validaciones adicionales para evitar duplicados.
  - Acceso menos eficiente para grandes conjuntos de datos.

1.2.2. Estrategia 2: Uso de conjuntos (`HashSet`)

- **Descripción:** Implementación basada en conjuntos para evitar duplicados y mejorar la eficiencia.
- **Ventajas:**
  - Garantiza la unicidad de los elementos.
  - Ofrece un acceso más eficiente.
- **Desventajas:**
  - Menos adecuado para mantener un orden específico de los elementos.

1.2.3. Estrategía 3: Uso de mapas

- **Descripción:** Utilización de estructuras como `HashMap` para optimizar las asociaciones.
- **Ventajas:**
  - Alta eficiencia para búsquedas.

- **Desventajas:**
  - Introduce mayor complejidad en el diseño del código.

1.3. Implementación en Java

A continuación, se presentan las implementaciones en Java para las clases `Socio`, `Voluntario`, `Adoptante` y `Donante`.

1.3.1. Clase `Socio`

La clase abstracta `Socio` sirve como base para las clases `Voluntario`, `Adoptante` y `Donante`. Se asegura que cada socio esté asociado a un refugio y que su información cumpla con las restricciones del modelo.

```
public abstract class Socio {
    private int ID;
    private Date fecha;
    private final Refugio refugioAsociado;

    public Socio(int ID, Date fecha, Refugio refugioAsociado) {
        assert ID > 0 : "El ID del socio debe ser valido.";
        assert fecha != null : "La fecha de registro no puede ser nula.";
        assert refugioAsociado != null : "El refugio asociado no puede ser
        this.ID = ID;
        this.fecha = fecha;
        this.refugioAsociado = refugioAsociado;
    }

    public int getID() { return ID; }
    public Date getDate() { return this.fecha; }
    public Refugio getRefugio() { return this.refugioAsociado; }
}
```

1.3.2. Clase `Adoptante`

Esta clase maneja las adopciones de animales y utiliza listas para almacenar las asociaciones.

```
public class Adoptante extends Socio {
    private List<Adopcion> adopciones;

    public Adoptante(int ID, Date date, Refugio r) {
        super(ID, date, r);
        adopciones = new ArrayList<>();
    }

    public void adoptar(Animal a, Voluntario v) {
```

```
        assert !adopciones.stream().anyMatch(ad -> ad.getAnimal().equals(a)
            "El adoptante ya tiene registrado este animal";
        v.tramitarAdopcion(a, this);
    }

    public void addAdopcion(Adopcion a) {
        if (!adopciones.contains(a)) adopciones.add(a);
    }
}
```

1.3.3. Clase `Voluntario`

Esta clase gestiona los trámites de adopción y registro de animales en el refugio.

```
public class Voluntario extends Socio {
    private List<Adopcion> tramites;

    public Voluntario(int ID, Date date, Refugio r) {
        super(ID, date, r);
        tramites = new ArrayList<>();
    }

    public void tramitarAdopcion(Animal a, Adoptante ad) {
        assert a.getEstadoAnimal() == EstadoAnimal.DISPONIBLE :
            "El animal ya esta adoptado.";
        Adopcion adopcion = new Adopcion(a, ad, this, new Date());
        this.tramites.add(adopcion);
    }
}
```

1.3.4. Clase `Donante`

Esta clase utiliza conjuntos (`HashSet`) para manejar las donaciones, asegurando unicidad.

```
public class Donante extends Socio {
    private Set<Donacion> donaciones;

    public Donante(int ID, Date date, Refugio r, Double cantidad) {
        super(ID, date, r);
        assert cantidad > 0 : "La cantidad inicial debe ser mayor a cero.";
        donaciones = new HashSet<>();
        this.donar(cantidad);
    }

    public void donar(Double cantidad) {
        assert cantidad > 0 : "La cantidad donada debe ser mayor a cero.";
```

```
        Donacion d = new Donacion(cantidad, new Date(), this);
        donaciones.add(d);
    }
}
```

1.4. Justificación de la estrategia elegida

Se ha optado por una combinación de listas y conjuntos debido a:

- Modularidad: Cada clase maneja sus propias asociaciones de manera clara y estructurada.
- Escalabilidad: Las listas y conjuntos son adecuadas para manejar escenarios con un número moderado de asociaciones.
- Mantenimiento: La separación de responsabilidades facilita futuras modificaciones.

1.5. Conclusión

El diseño y la implementación presentados cumplen con los requisitos del modelo, garantizando claridad, eficiencia y modularidad en el código.

RF2.1 Crear Producto

Descripción

Los usuarios deben de poder crear productos mientras sea posible, definiendo sus atributos y asignándoles sus respectivas categorías y relaciones.

Pre-condición
El usuario debe haber iniciado sesión en su cuenta en Mini PIM.

Post-condición

- Caso de éxito: Todos los productos que el usuario creó se reflejan en la base de datos del sistema y en su interfaz gráfica.
- Caso mínimo: El sistema notifica al usuario el resultado de la acción de crear producto; exitosa o fallida.

Prioridad: Alta

Autor: Francisco Javier Jordá Garay

Control de cambios: Versión 1: Definición del caso de uso

Escenario principal

1. El usuario se encuentra en el apartado de productos y selecciona la opción de "Añadir".
2. El sistema muestra el menú de creación solicitando al usuario:
    - GTIN (atributo sistema − comprueba validez de longitud)
    - SKU (atributo sistema)
    - Thumbnail (atributo sistema − comprueba tamaño 200×200px y formato)
    - Label (atributo sistema − comprueba máximo de 250 caracteres)
    - Atributos (opcional − comprueba máximo 5 nuevos atributos usuario)
    - Categorías (opcional)
3. El usuario introduce los datos obligatorios y los que decida de opcionales y selecciona "Confirmar".
4. El sistema comprueba la validez de los datos introducidos por el usuario.
5. El sistema almacena el producto creado en la base de datos registrando la fecha de creación.
6. El sistema actualiza la información del total de datos registrados en la base de datos.
7. El sistema muestra el apartado de "Productos" todos los recursos almacenados para esta sección.

Escenarios alternativos

2.a. El sistema no puede almacenar el producto por superar el máximo de almacenamiento ligado al plan de suscripción del usuario.

2.a.1 El sistema notifica al usuario que ha llegado al máximo de capacidad permitida en el plan de almacenamiento.

***.a** El usuario cancela la acción de crear un nuevo producto seleccionando la opción que cierra el menú de creación.

*.a.1 El sistema regresa al apartado de "Productos".

4.a El sistema detecta un fallo en la comprobación de los datos obligatorios.

4.a.1 El sistema notifica del error de comprobación al usuario mostrando el atributo del producto afectado.

4.a.2 El sistema regresa al menú de creación permitiendo edición de los datos.

Casos de Prueba

Escenario: Principal

Dado que inicié sesión con mi cuenta de usuario correspondiente
Y estoy en el apartado de Productos
Cuando selecciono la opción de "Añadir"
E introduzco correctamente los atributos del producto que deseo crear
Y selecciono "confirmar" para guardar los datos
Entonces el sistema almacena la información en la base de datos de Mini PIM
Y actualiza la información del total de datos registrados en la base de datos
Y muestra el apartado de Productos con todos los recursos almacenados para esta sección.

Escenario: Alternativo 2.a

Dado que inicié sesión con mi cuenta de usuario correspondiente
Y tengo el límite de productos
Y estoy en el apartado de Productos
Cuando selecciono la opción de "Añadir"
E introduzco correctamente los atributos del producto que deseo crear
Y selecciono "confirmar" para guardar los datos
Entonces el sistema me notifica que no puede almacenar el producto por superar el máximo de almacenamiento ligado a mi plan de suscripción actual

Escenario: Alternativo 3.a

Dado que inicié sesión con mi cuenta de usuario correspondiente
Y estoy en el apartado de Productos
Cuando selecciono la opción de "Añadir"
Y selecciono la opción de cancelar
Entonces el sistema muestra el apartado de Productos mostrando todos los recursos almacenados sin ningún cambio.

Escenario: Alternativo 4.a

Dado que inicié sesión con mi cuenta de usuario correspondiente
Y estoy en el apartado de Productos
Cuando selecciono la opción de "Añadir"
Y escribo los datos del producto y dejo uno obligatorio vacío
Entonces el sistema me muestra cual es el atributo que falla
Y regresa al menú de creación.

Bocetos

RF2.1 Crear Producto

Descripción

Los usuarios deben de poder crear productos mientras sea posible, definiendo sus atributos y asignándoles sus respectivas categorías y relaciones.

Pre-condición
El usuario debe haber iniciado sesión en su cuenta en Mini PIM.

Post-condición

- Caso de éxito: Todos los productos que el usuario creó se reflejan en la base de datos del sistema y en su interfaz gráfica.
- Caso mínimo: El sistema notifica al usuario el resultado de la acción de crear producto; exitosa o fallida.

Prioridad: Alta

Autor: Francisco Javier Jordá Garay

Control de cambios: Versión 1: Definición del caso de uso

Escenario principal

1. El usuario se encuentra en el apartado de productos y selecciona la opción de "Añadir".
2. El sistema muestra el menú de creación solicitando al usuario:
    - GTIN (atributo sistema − comprueba validez de longitud)
    - SKU (atributo sistema)
    - Thumbnail (atributo sistema − comprueba tamaño 200×200px y formato)
    - Label (atributo sistema − comprueba máximo de 250 caracteres)
    - Atributos (opcional − comprueba máximo 5 nuevos atributos usuario)
    - Categorías (opcional)
3. El usuario introduce los datos obligatorios y los que decida de opcionales y selecciona "Confirmar".
4. El sistema comprueba la validez de los datos introducidos por el usuario.
5. El sistema almacena el producto creado en la base de datos registrando la fecha de creación.
6. El sistema actualiza la información del total de datos registrados en la base de datos.
7. El sistema muestra el apartado de "Productos" todos los recursos almacenados para esta sección.

Escenarios alternativos

2.a. El sistema no puede almacenar el producto por superar el máximo de almacenamiento ligado al plan de suscripción del usuario.

2.a.1 El sistema notifica al usuario que ha llegado al máximo de capacidad permitida en el plan de almacenamiento.

***.a** El usuario cancela la acción de crear un nuevo producto seleccionando la opción que cierra el menú de creación.

*.a.1 El sistema regresa al apartado de "Productos".

4.a El sistema detecta un fallo en la comprobación de los datos obligatorios.

4.a.1 El sistema notifica del error de comprobación al usuario mostrando el atributo del producto afectado.

4.a.2 El sistema regresa al menú de creación permitiendo edición de los datos.

Casos de Prueba

Escenario: Principal

Dado que inicié sesión con mi cuenta de usuario correspondiente
Y estoy en el apartado de Productos
Cuando selecciono la opción de "Añadir"
E introduzco correctamente los atributos del producto que deseo crear
Y selecciono "confirmar" para guardar los datos
Entonces el sistema almacena la información en la base de datos de Mini PIM
Y actualiza la información del total de datos registrados en la base de datos
Y muestra el apartado de Productos con todos los recursos almacenados para esta sección.

Escenario: Alternativo 2.a

Dado que inicié sesión con mi cuenta de usuario correspondiente
Y tengo el límite de productos
Y estoy en el apartado de Productos
Cuando selecciono la opción de "Añadir"
E introduzco correctamente los atributos del producto que deseo crear
Y selecciono "confirmar" para guardar los datos
Entonces el sistema me notifica que no puede almacenar el producto por superar el máximo de almacenamiento ligado a mi plan de suscripción actual

Escenario: Alternativo 3.a

Dado que inicié sesión con mi cuenta de usuario correspondiente
Y estoy en el apartado de Productos
Cuando selecciono la opción de "Añadir"
Y selecciono la opción de cancelar
Entonces el sistema muestra el apartado de Productos mostrando todos los recursos almacenados sin ningún cambio.

Escenario: Alternativo 4.a

Dado que inicié sesión con mi cuenta de usuario correspondiente
Y estoy en el apartado de Productos
Cuando selecciono la opción de "Añadir"
Y escribo los datos del producto y dejo uno obligatorio vacío
Entonces el sistema me muestra cual es el atributo que falla
Y regresa al menú de creación.

Bocetos