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 describe el diseño del código de andamiaje necesario para implementar el modelo de gestión del refugio. Las clases principales y las estructuras de datos se han implementado para garantizar la correcta funcionalidad y la relación entre las entidades definidas. Se justifica la elección de estructuras de datos como `Set` para evitar duplicados y se asegura la integridad mediante el uso de aserciones (`assert`).

1.2. Implementación del Modelo

1.2.1. Clase Socio

La clase `Socio` es abstracta y representa la base para las subclases `Adoptante`, `Voluntario`, y `Donante`. Esta clase asegura que cada socio tenga un `ID` único, una fecha de registro válida y un refugio asociado.

```
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;
    }
}
```

1.2.2. Clase Donante

La clase `Donante` extiende de `Socio` y gestiona las donaciones realizadas por un socio. Las donaciones se almacenan en un `Set` para evitar duplicados.

```
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 donada debe ser mayor a
        donaciones = new HashSet<>();
        this.donar(cantidad);
    }

    public void donar(Double cantidad) {
        assert cantidad > 0 : "La cantidad donada debe ser mayor a cero.";
        LocalDate fechaDonacion = LocalDate.now();
        Donacion d = new Donacion(cantidad, Date.from(fechaDonacion.atStart
        donaciones.add(d);
        Refugio r = super.getRefugio();
        r.setLiquidez(r.getLiquidez() + cantidad);
        r.addSocio(this);
        assert donaciones.contains(d);
    }
}
```

1.2.3. Clase Adoptante

La clase `Adoptante` extiende de `Socio` y gestiona las adopciones realizadas por un adoptante. Las adopciones se almacenan en un `Set`.

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

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

    public void adoptar(Animal a, Voluntario v) {
        assert !adopciones.stream().anyMatch(ad -> ad.getAnimal().equals(a)
        v.tramitarAdopcion(a, this);
    }

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

1.2.4. Clase Voluntario

La clase `Voluntario` extiende de `Socio` y gestiona los trámites de adopción realizados por un voluntario.

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

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

    public void tramitarAdopcion(Animal a, Adoptante ad) {
        assert a.getEstadoAnimal() == EstadoAnimal.DISPONIBLE : "El animal
        LocalDate fechaAdopcion = LocalDate.now();
        Adopcion adopcion = new Adopcion(a, ad, this, Date.from(fechaAdopci
        tramites.add(adopcion);
    }
}
```

1.2.5. Clase Refugio

La clase `Refugio` gestiona el conjunto de `Socios` y `Animales`. Las operaciones están centralizadas para simplificar la gestión.

```
public class Refugio {
    private double liquidez;
    private Set<Animal> animalesRegistrados;
    private Set<Socio> socios;

    public Refugio(double liquidez) {
        assert liquidez >= 0 : "La liquidez debe ser no negativa.";
        this.liquidez = liquidez;
        animalesRegistrados = new HashSet<>();
        socios = new HashSet<>();
    }

    public void addSocio(Socio s) {
        assert s != null : "El socio no puede ser nulo.";
        socios.add(s);
    }
}
```

1.2.6. Clase Donacion

La clase `Donacion` representa una donación realizada por un `Donante`. Incluye la cantidad, la fecha de la donación y el donante asociado. Las validaciones aseguran que los

valores sean válidos en el momento de la creación de la instancia.

```
public class Donacion {
    private Double cantidad;
    private Date date;
    private final Donante donante;

    public Donacion(Double cantidad, Date date, Donante donante) {
        assert cantidad != null && cantidad > 0 : "La cantidad debe ser pos
        assert date != null && !date.after(new Date()) : "La fecha no puede
        assert donante != null : "El donante no puede ser nulo.";
        this.cantidad = cantidad;
        this.date = date;
        this.donante = donante;
    }

    public Double getCantidad() {
        assert cantidad != null && cantidad > 0 : "La cantidad no puede ser
        return cantidad;
    }

    public void setCantidad(Double cantidad) {
        this.cantidad = cantidad;
    }

    public Date getDate() {
        assert date != null : "La fecha no puede ser nula.";
        return date;
    }

    public void setDate(Date date) {
        this.date = date;
    }

    public Donante getDonante() {
        return this.donante;
    }

    @Override
    public String toString() {
        return String.format("Donacion: %.2f, %tY-%tB-%td", cantidad, date,
    }
}
```

1.2.7. Clase Adopcion

La clase `Adopcion` modela una adopción de un `Animal` realizada por un `Adoptante`, gestionada por un `Voluntario`. Implementa la bidireccionalidad entre estas entidades para mantener consistencia en las asociaciones.

```
public class Adopcion {
    private Date fecha;
    final private Animal animal;
    final private Adoptante adoptante;
    final private Voluntario voluntario;

    public Adopcion(Animal a, Adoptante ad, Voluntario v, Date fecha) {
        assert a != null : "El animal no puede ser nulo.";
        assert ad != null : "El adoptante no puede ser nulo.";
        assert v != null : "El voluntario no puede ser nulo.";
        assert a.getEstadoAnimal() == EstadoAnimal.DISPONIBLE : "El animal
        assert fecha != null && !fecha.after(new Date()) : "La fecha no pue

        this.animal = a;
        this.adoptante = ad;
        this.voluntario = v;
        this.fecha = fecha;

        a.setEstadoAnimal(EstadoAnimal.ADOPTADO);
        ad.addAdopcion(this);
        assert Collections.list(ad.getAdopciones()).contains(this) :
        "La adopcion no fue anadida correctamente al adoptante.";
        v.addTramite(this);
        assert Collections.list(v.getTramites()).contains(this) :
        "La adopcion no fue anadida correctamente al voluntario.";
    }

    public Date getFecha() {
        return this.fecha;
    }

    public void setFecha(Date fecha) {
        assert fecha != null && !fecha.after(new Date()) : "La fecha no pue
        this.fecha = fecha;
    }

    public Animal getAnimal() {
        return this.animal;
    }

    public Voluntario getVoluntario() {
```

```
        return this.voluntario;
    }

    public Adoptante getAdoptante() {
        return this.adoptante;
    }

    @Override
    public String toString() {
        return String.format("Adopcion: %tY-%tB-%td, %s, %s", fecha, fecha,
    }
}
```

1.2.8. Clase Animal

La clase `Animal` modela a un animal registrado en el sistema. Cada animal tiene un ID único, una fecha de nacimiento, un estado actual y está asociado a un `Refugio`.

```
public class Animal {
    private int ID;
    private Date nacimiento;
    private EstadoAnimal estadoAnimal;
    final private Refugio refugio;
    private Adopcion adopcion;

    public Animal(int ID, Date nacimiento, EstadoAnimal estadoAnimal, Refug
        assert ID > 0 : "El ID del animal debe ser valido.";
        assert nacimiento != null : "La fecha de nacimiento no puede ser nu
        assert estadoAnimal != null : "El estado del animal debe estar defi
        assert refugio != null : "El refugio debe existir.";

        this.ID = ID;
        this.nacimiento = nacimiento;
        this.estadoAnimal = estadoAnimal;
        this.refugio = refugio;
        this.adopcion = adopcion;
    }

    public EstadoAnimal getEstadoAnimal() {
        return estadoAnimal;
    }

    public void setEstadoAnimal(EstadoAnimal estadoAnimal) {
        assert estadoAnimal != null : "El estado del animal debe estar defi
        this.estadoAnimal = estadoAnimal;
    }
```

```
    public Date getNacimiento() {
        return nacimiento;
    }

    public void setNacimiento(Date nacimiento) {
        assert nacimiento != null : "La fecha de nacimiento no puede ser nu
        this.nacimiento = nacimiento;
    }

    public Refugio getRefugio() {
        return refugio;
    }

    public Adopcion getAdopcion() {
        return this.adopcion;
    }

    public void setAdopcion(Adopcion adopcion) {
        assert adopcion != null;
        this.adopcion = adopcion;
    }

    @Override
    public String toString() {
        return String.format("Animal: ID=%d, nacimiento=%tF, estado=%s", ID
    }
}
```

1.3. Conclusión

El código de andamiaje diseñado utiliza `Set` para evitar duplicados y asegura la bidireccionalidad de las asociaciones entre clases mediante comprobaciones con `assert`. Esto garantiza la consistencia e integridad del modelo.

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