**Universidad de Costa Rica**

**Escuela de Ciencias de la Computación e Informática (ECCI)**

**CI-0112 Programación I**

**Grupo: 002**

**Documento de decisiones: Proyecto 2**

**Docente:**

**Luis Campos Duarte**

**Estudiantes:**

**Josué Rodríguez Aguilar I C4J023**

**Kevin Calderon Martinez I C4D511**

**I Ciclo**

**Julio 2025**

**a. Acuerdos de Pareja:** Describir los acuerdos alcanzados entre los miembros del equipo, como la división del trabajo y las responsabilidades.

El acuerdo logrado entre ambos estudiantes fue el de, principalmente, la subdivisión mutua del sistema. Primero, ambos hicieron la clase mascota para luego proceder con las estructuras; a Kevin se le ha asignado hacer la estructura dinámica de la Cola, y a Josué la del Árbol (Con sus respectivos nodos). Ambos lograron terminar su respectivas estructuras, y luego Kevin junto a Josué por llamada se encargaron de trabajar las demás clases (Veterinario, GUI). Para equiparar las cargas, Josué y Kevin trabajaron primero la GUI y Veterinario, pero posterior a ello Kevin realizó correcciones claves en las clases Veterinario e InterfazGráfica; puesto que el árbol es una estructura más pesada que la cola. Kevin también creó el repositorio del proyecto en GitHub, mientras que Josué se encargó de trabajar el Markdown del README y el doxygen para los puntos extra.

**b. Decisiones de Diseño:** Explicar cómo se diseñaron las clases, cómo se estructuraron los métodos y cualquier decisión significativa tomada durante el diseño.

**Mascota:**

Para esta clase no se diseñaron métodos en específico más que setters y getters, en conjunto, a la declaración de atributos para el objeto de mascotas que se implementa en las demás clases, con características como nombre, id y especie.

**NodoMascotaABB:**

De forma similar a mascotas los nodos del árbol solo cuenta con la declaración de atributos básicos para un árbol como nodo izquierdo y derecho, pues es un árbol binario y la mascota que debe de contener cada nodo. En conjunto a los setters y getters.

**NodoMascotaCola:**

Esta clase guarda tres variables; una Mascota m, que es guardada en los nodos de la misma, y referencias a los nodos head y tail, que son inicializadas en el constructor como nulas. Esta clase es clave para el desarrollo de la estructura de la cola.

**ArbolMascota:**

Para la clase ArbolMascota ya se empieza la implementación de métodos más complejos como lo fue la búsqueda por id de las mascotas en el árbol, la eliminación de mascotas del registro que llevaba el árbol, insertar mascotas en el árbol y el recorrido inorden por utilizar. Claramente ya entra el apartado recursivo con los métodos privados y los llamados mediante otros método públicos.

**ColaMascota:**

La clase ColaMascota es una estructura dinámica, clave para la atención de las mascotas en la clase Veterinario. La clase cola cuenta con los métodos convencionales de esta estructura (Enqueue, Dequeue, eliminar), pero trae algunos métodos claves extras, como peek que muestra la cabeza de la cola, y mostrarCola que llama a un recursivo, retornando los valores de los nodos de la Cola, desde la cabeza hasta la cola.

**Veterinario:**

Para la clase veterinario, su funcionalidad principal fue la de fusionar los métodos del árbol y la cola con la finalidad de lograr todo lo solicitado como lo era el registro de mascotas automatizando el id de cada una de ellas, el llamado público de cada método recursivo diseñado en las clases Cola y Árbol y se diseñaron los métodos para almacenar los datos en un archivo .txt necesarios para cargar los datos ingresados con anterioridad al programa.

**InterfazGrafica:**

La InterfazGrafica se encarga de mostrar todo el control de las estructuras del proyecto: Este consta de un PanelPrincipal, con tres paneles extra: Uno que registra las mascotas, con el nombre y la especie, junto a la lógica del botón registrar, un panel que muestra el historial mediante el árbol y un botón de eliminar registro, y una cola que se encarga de mostrar los animales que vamos agregando a la cola y atendiendo, mediante los botones Agregar a la Cola y Atender Mascota.

**c. Decisiones de Implementación:** Detallar las decisiones específicas de implementación, como la elección de estructuras de datos, algoritmos utilizados, etc.

Para esta decisión de implementación se comentará sobre los métodos principales o las decisiones más influyentes en el manejo del proyecto y el sistema de la clínica veterinaria como tal. Cabe recalcar que escogimos las estructuras de la Cola y el árbol binario, puesto que el proyecto requería como solución el implementarlos.:

**ArbolMascota:**

La clase de Árbol se encarga del registro en tiempo real de las mascotas registradas en el sistema, por lo que tiene métodos como:



Este método representa la recursividad a la hora de insertar una mascota donde se requiere a la mascota que se desea insertar y el nodo raiz del arbol para lograr analizar dónde colocarla, este método toma el id del nodo raíz del árbol, y lo compara con el id de la mascota indicada, para luego identificar si debe de continuar con el proceso de insertar en el nodo hijo derecho o izquierdo en caso que tenga, y si el método llega a posicionarse en un nodo null inserta la mascota. Importante, cada vez que chequea el árbol el método retorna la mascota pues necesita saber por así decirlo donde quedó con anterioridad.



Esta es la continuación del método recursivo de insertar, solamente que este se encarga de llamarlo en diferentes clases, con el único parámetro que se requiere que es la mascota.



Este es el método recursivo de eliminar que nuevamente requiere de la mascota y el nodo raiz del arbol, este examina el id de la mascota otorgada y lo compara con el nodo existente en el árbol, luego en dependencia de la cantidad de hijos puede: Eliminar el nodo hoja, eliminar el nodo padre y dejar su único nodo hijo como sucesor, o encontrar el mínimo id de la sub rama derecha del nodo padre que se examina para eliminar y sustituirlo por el mismo, así eliminandolo del árbol.



Este método fue un auxiliar recursivo necesario para eliminar nodos con dos hijos pues necesitábamos encontrar el nodo más “pequeño” de la sub rama derecha del nodo padre, este simplemente buscaba el nodo izquierdo siempre que no fuera nulo y paraba cuando lo fuese.



Método público para llamar al eliminar de otras clases, que únicamente requiere la mascota por eliminar (bueno técnicamente solo su id).



Método recursivo de buscar por id las mascotas, nuevamente requiere de la mascota y el nodo raiz del arbol para iniciar la búsqueda, en este caso es el mismo sistema de comparación de ids para viajar alrededor de los nodos izquierdos y derechos del árbol, hasta que da con el nodo con el id buscado de la mascota dada.



Método público del llamado recursivo de buscar que únicamente requiere del id, pues inicializa una nueva mascota con ese id indicado y la busca mediante el llamado recursivo anterior.



Método recursivo del recorrido inorden, donde se muestra primero el nodo izquierdo, luego nodo padre y por último nodo derecho, en este caso se retorna un string con separadores de líneas basados en la muestra de los datos en dicho orden.



Método público que llama el recorrido simplemente retorna todo el recorrido desde la raíz del árbol conforme lo anteriormente descrito.



Verifica que el árbol esté o no esté vacío, mediante false y true respectivamente.



Método auxiliar necesario para la clase veterinario, pues necesitábamos obtener la raíz del árbol de una clase externa.

**ColaMascota**

Esta clase es la estructura de la Cola de atención. Trae sus métodos convencionales mencionados anteriormente, pero además también:



Hace llamado recursivo a un método privado que retorna los valores de los nodos llenos de la cola.



Método clave para verificar si los nodos de la cola no están vacíos, se encarga de verificar que se repitan las ids de las mascotas.

**Veterinario**

****

Le da un id random a la mascota, y mientras que no se repita, la guarda junto a la mascota en el árbol y guarda la información en un archivo de texto.

****

Se llama al método enqueue de la cola.



Guarda la información de las mascotas con un PrintWriter separados por una “,”.



Lee y escribe la información de las mascotas registradas y guardadas en el árbol.



Hace llamado al privado recursivo que guarda cada nodo existente en ese momento en el árbol.

**InterfazGrafica**

Los elementos más claves de la GUI son:



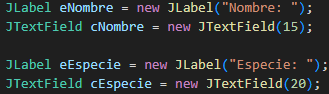
La declaración del panel principal.



El ordenamiento de los demás paneles en el principal de manera horizontal.



El panel de los datos a registrar que contiene:



Los campos de texto y etiquetas para agregar el nombre de la mascota y su especie.



El botón de registrar junto a su ActionListener, que trata de captar todos los errores existentes a la hora de registrar una mascota (especie sin números, datos no vacíos). Llama al método registrar de Veterinario, así que agrega a la cola a las mascotas que se registran.



Panel de la cola de atención, contiene:



Área no editable que muestra la cola.



Botón para encolar a las mascotas, diciéndole al usuario que digite el ID del animal que desea encolar. Capta los problemas si no hay mascotas registradas, el id no es válido o contiene letras, que no ingresó id, y luego llama al método de agregarCola de Veterinario.



Botón para atender a las mascotas, verifica que la cola no esté vacía y llama al método de atender, que es el dequeue, esto lo hace en orden FIFO (Pues es una cola).

  
 El panel del árbol que contiene:



Área de texto no editable que muestra el historial de mascotas registradas.



El botón de eliminar el registro de mascotas; primero capta los errores de que no hayan mascotas registradas, luego le pide al usuario la id (Si el id es invalido o no se encuentran mascotas con ese id, capta esos errores), y le pregunta al usuario si quiere eliminarla; si le da yes llama al método de eliminar del árbol, actualiza el historial con el método de verHistoria, que retorna los nuevos nodos del árbol.

**d. Puntos de Mejora:** Identificar áreas que podrían mejorarse o ampliarse en futuras versiones del proyecto.

En general, el desarrollo del proyecto se ha realizado en un ambiente ameno. Los aspectos por mejorar en este caso son realmente pocos; la comunicación y el acuerdo entre los autores de este trabajo ha sido muy clara y el acuerdo justo, y la división del trabajo ha sido lo suficientemente equitativa.

Si bien hubo algunos detalles por mejorar en el proyecto, para el correcto desarrollo del mismo (Principalmente en cómo se implementa la InterfazGrafica), la cercana comunicación con el profesor ayudó al grupo de trabajo a poder implementar las soluciones.

Lo único por mejorar sería el desarrollar los programas con la lógica correcta, pero esto en la programación se termina resolviendo mediante la metodología de prueba-error. Puesto que el consultarle al profesor también fue clave para resolver los errores del código.

Se desea que futuros proyectos sean desarrollados mediante el mismo grupo (Aunque tal vez no en este curso), puesto que el trabajo fue muy ordenado y ameno.