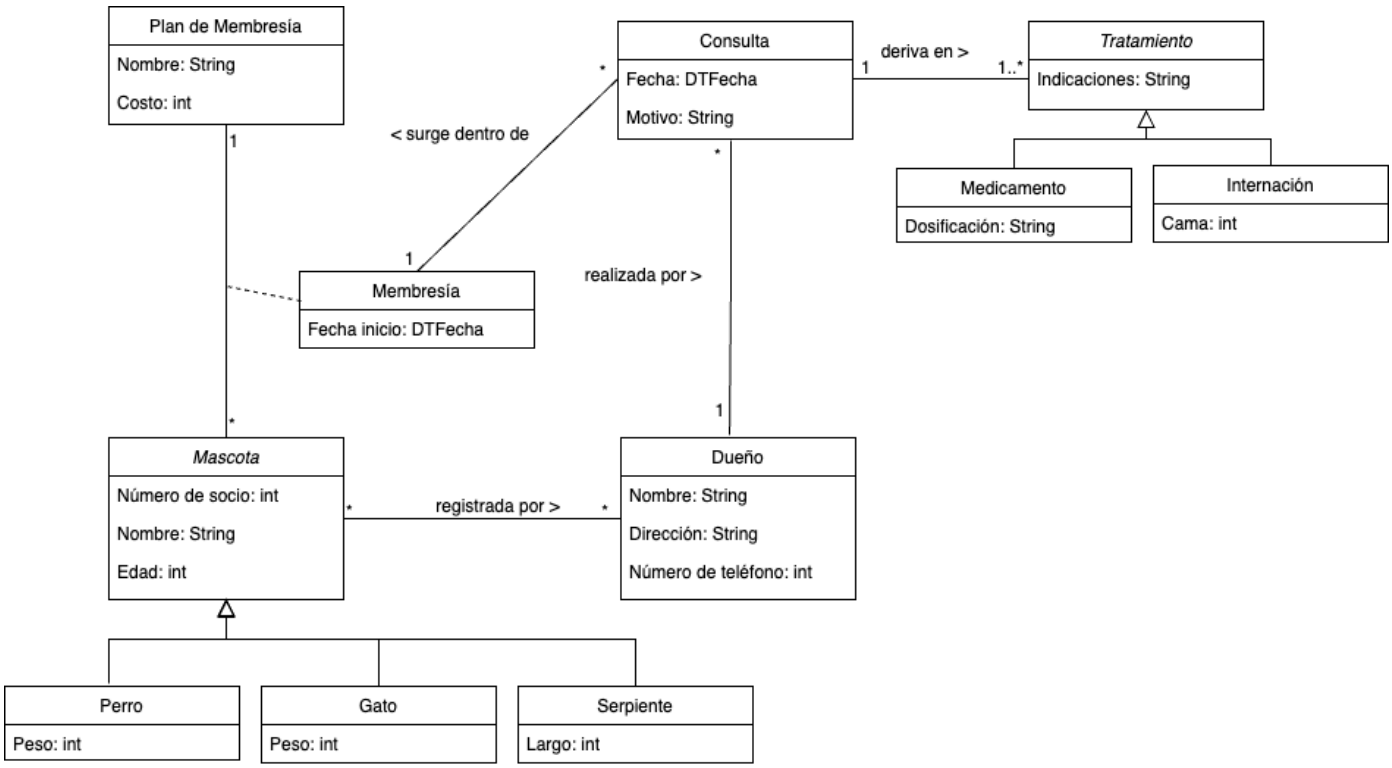
Programación 4

EXAMEN JULIO 2023
SOLUCIÓN

Problema 1 (30 puntos)

Se desea crear un sistema para registrar las consultas de mascotas de una veterinaria.

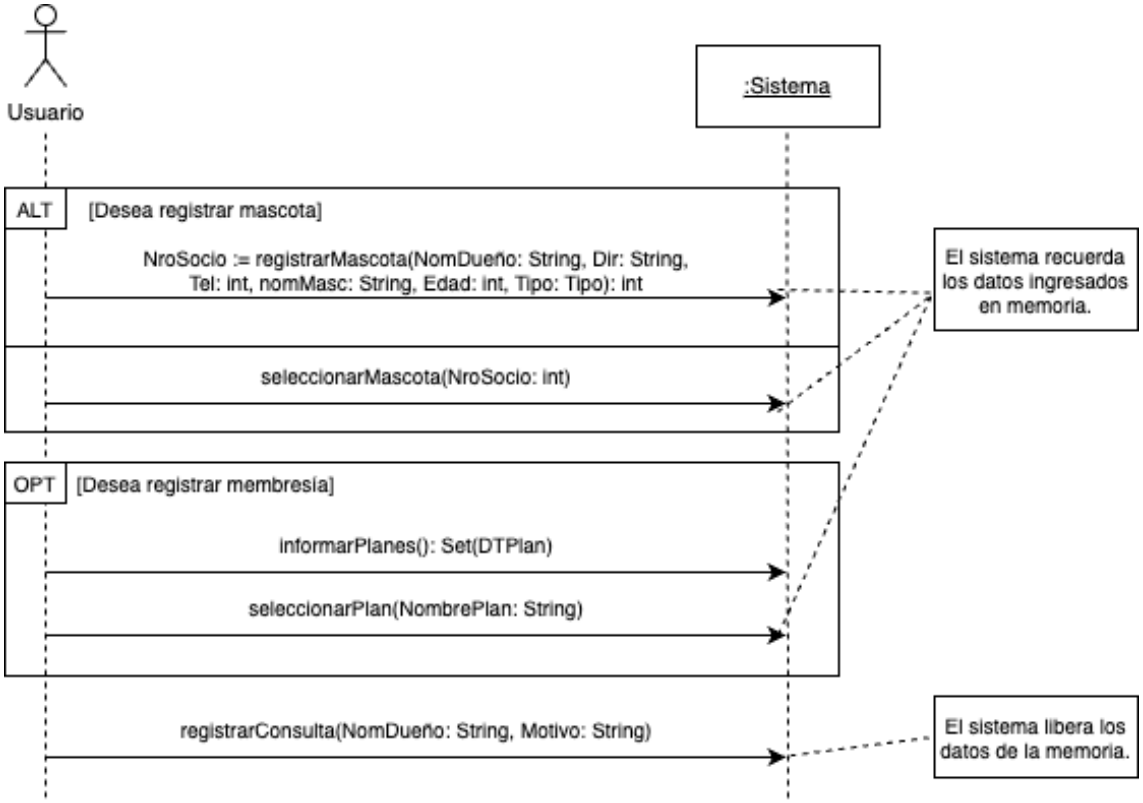
a) Realizar el Modelo de Dominio de la realidad planteada, incluyendo todas las restricciones que considere necesarias en lenguaje natural.



Restricciones

- No existen dos instancias de Dueño con igual valor para el atributo Nombre.
- No existen dos instancias de Mascota con igual valor para el atributo Número de Socio.
- No existen dos instancias de Plan de Membresía con igual valor para el atributo Nombre.
- Un Dueño está asociado a una Consulta solo si la Consulta surge dentro de una Membresía de una Mascota que está registrada por ese Dueño.

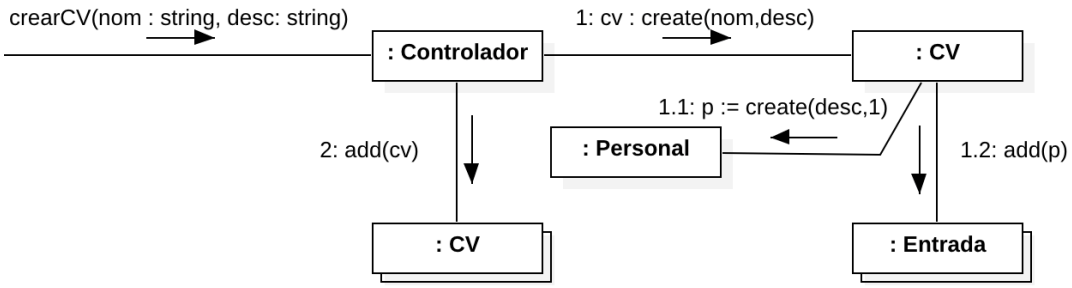
b) Realizar un Diagrama de Secuencia del Sistema (DSS) para el Caso de Uso. Indique el uso de memoria del Sistema y de datatypes, si corresponde.



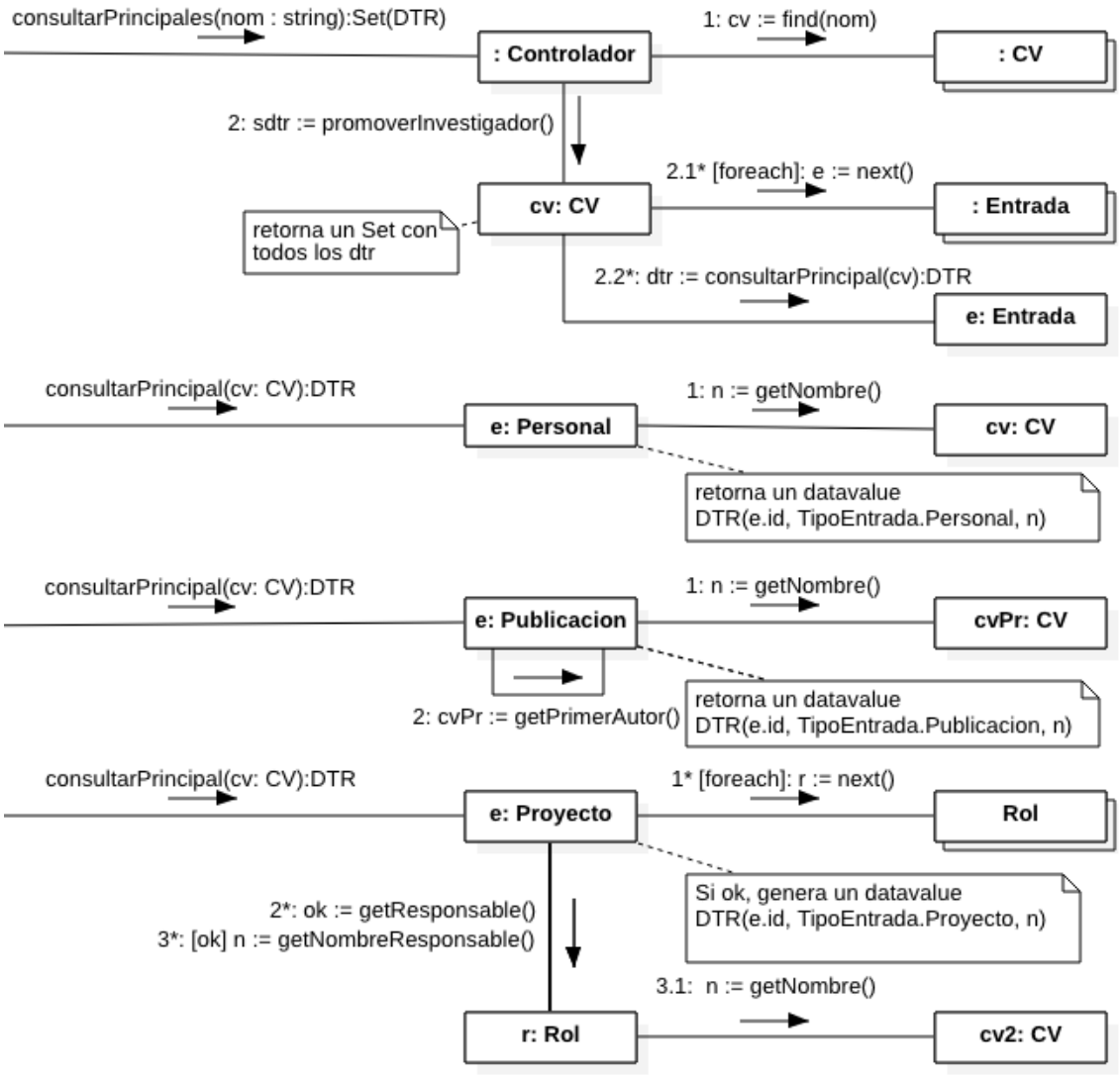
Problema 2 (35 puntos)

Se está desarrollando un sistema de gestión de currículum para investigadores.

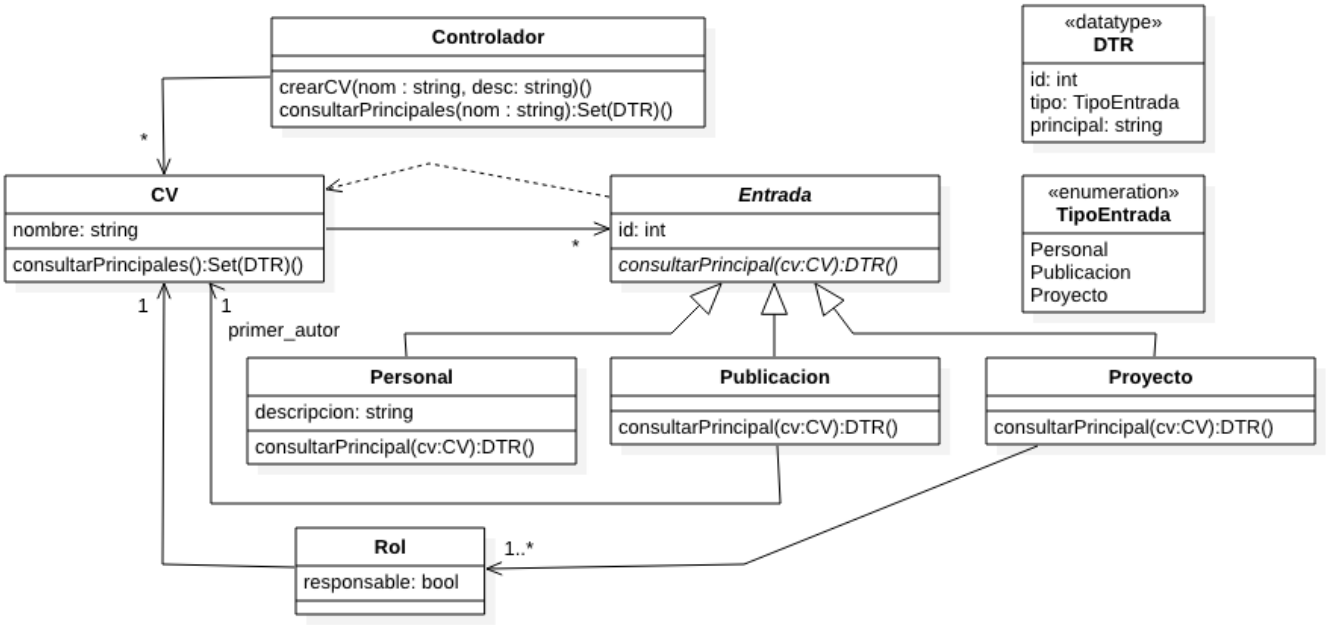
a) Realizar el Diagrama de Comunicación correspondiente a la operación del sistema `crearCV(nom : string, desc: string)`.



b) Realizar el Diagrama de Comunicación correspondiente a la operación del sistema `consultarPrincipales(nom : string):Set(DTR)`.



c) Realizar el Diagrama de Clases de Diseño (DCD) resultante. No incorporar setters, getters ni operaciones de colecciones, e incorporar sólo los constructores y destructores que se utilicen en los diagramas de comunicación.



Problema 3 (35 puntos)

Se está construyendo un sistema para la gestión de documentos.

a) Identifique qué patrones se utilizan en este diseño, exceptuando el patrón Singleton. Para cada patrón identificado, indicar las clases participantes y sus roles.

Se utilizan los patrones Template Method y Strategy de la siguiente forma:

- Template Method: Documento (Clase abstracta), Texto y Planilla (Clases concretas), mostrar (esqueleto), mostrarContenido (operación primitiva).
- Strategy: Usuario (Contexto), Visualizador (Estrategia), VisAndroid y VisIOS (Estrategias concretas).

b) Implementar en C++ el .h de la clase `Usuario`. Incluir el(los) constructor(es) que sea(n) necesario(s). No incluir destructor ni operaciones get/set.

```
class Usuario {
      private:
            int idUsr;
            std::map<std::string, Documento *> docs;
            Visualizador *vis;
      public:
            Usuario(int);
            void nuevoDoc(DTDataDoc *);
            void setVisualizador(TipoVisualizacion);
            void visualizar(std::string);
};
```

c) Implementar en C++ las operaciones `Usuario::nuevoDoc`, `Usuario::setVisualizador` y `Usuario::visualizar`.

```
void Usuario::nuevoDoc(DTDataDoc *dtd) {
      Documento *nuevoDoc;
      int nuevoId = Utils::getNuevoIdDoc();
      if (dynamic_cast<DTDataTexto *>(dtd) != NULL)
            nuevoDoc = new Texto(nuevoId,
                                 dynamic_cast<DTDataTexto *>(dtd));
      else
            nuevoDoc = new Planilla(nuevoId,
                                    dynamic_cast<DTDataPlanilla*>(dtd));
      docs->insert(std::pair<int, Documento *>(nuevoId, nuevoDoc));
}

void Usuario::setVisualizador(TipoVisualizacion t) {
      CtrlVis *cVis = CtrlVis::getInstance();
      switch(t) {
            case vAndroid:
                  vis = cVis->getVisAndroid();
                  break;
            case vIOS:
                  vis = cVis->getVisIOS();
      }
}

void Usuario::visualizar(std::string idDoc) {
      vis->generarVisualizacion(docs[idDoc]);
}
```