# Definición del Problema

Realice un programa que simule un turnero de un centro de ensayos médicos de la siguiente manera:

a. El programa tendrá dentro de sí un conjunto de estudios médicos fijos realizables por distintos pacientes que serán introducidos por el usuario, donde para cada estudio médico se guardará la siguiente información (esta información no será digitada por el usuario, sino que la tendremos en la base de datos):

- Número/ID de estudio.

- Nombre de estudio.

- Departamento al que pertenece dicho estudio.

- Duración que tarda en ser realizado dicho estudio.

b. Se le pedirá al usuario que especifique la cantidad de pacientes que se quieren considerar, y para cada paciente se le preguntarán al usuario los estudios (permitir seleccionar tanto por ID como por nombre) que se realizará.

c. Se le preguntará al usuario para cada paciente si se desea ordenarle sus estudios por los más duraderos primero o los más prontos primero.

d. Se ordenarán los estudios que se realizará cada paciente (usando un algoritmo eficiente de ordenamiento), según lo especificado por el usuario, y luego se procederá a asignar cada estudio en la línea de tiempo de su respectivo departamento, según la disponibilidad de éste.

Se presentará por pantalla:

a. La línea de tiempo de las cargas de trabajo de cada departamento, donde cada carga de trabajo es:

- El número del paciente que se hará el estudio en dicho departamento.

- El número (ID) del estudio que se realizará el paciente (y por lo tanto, el departamento en el que será realizado).

- El momento en el cual se realizará dicho estudio (su posición en la línea de tiempo).

b. El calendario que llevará el paciente para realizar sus estudios, donde cada elemento del calendario es lo equivalente a una carga de trabajo.

Se permitirá al usuario salir en cualquier momento de la aplicación y luego continuar donde quedó, empleando bases de datos para almacenar toda la información.

Se deberán cumplir las siguientes restricciones:

a. No se aceptarán más de 10 estudios para cada paciente.

b. No habrán más de 4 departamentos en total.

c. No habrán más de 10 estudios, cada uno perteneciendo a un departamento.

d. Un estudio no durará más de 10 unidades de tiempo en ser realizado.

e. Un paciente no se podrá de hacer más de 10 estudios en total.

Tabla de contenido

[Definición del Problema 1](#_Toc321982988)

[Análisis 1](#_Toc321982989)

[Diseño 2](#_Toc321982990)

[Código Fuente 3](#_Toc321982991)

[Condiciones de Error 4](#_Toc321982992)

[Ejemplos de Uso 5](#_Toc321982993)

[Notas al Profesor 6](#_Toc321982994)

# Análisis

Bueno este problemita a simple vista lo único que le veo de dificultad seria el implemento de algoritmos de ordenamiento para cuando pide que ordenen los estudios. Lo que se puede hacer es tener todo en un solo form bien amplio ya que no veo utilidad de tener varios forms.

Un inconveniente es como vamos a implementar el tiempo, ya que dice que el usuario podrá cerrar la aplicación en cualquier momento, y luego continuar en donde quedo. Entonces tengo una hay varias opciones: si usaremos el tiempo actual o un tiempo para la aplicación, que correrá mientras esta esté abierta.

El form se podrá dividir por áreas. Otra cosa que pienso que será un poco tediosa será la integración de la base de datos, para verificar cuando un paciente no tenga inscrito más de 10 estudios médicos. Se podría tener una clase llamada paciente que guarda la data del paciente y tener otra clase de una lista que herede a la clase Paciente.

-Para lo del tiempo, se podría guardar la hora actual, y cuando uno cargue el programa de nuevo se verá que hora es, y que tiempo ha pasado desde la última vez que se abrió la aplicación, en base a ese tiempo que paso vamos a empezar desde el primer estudio que está en la lista para eliminar los turnos que ya hayan pasado durante ese lapso de tiempo que se cerró el programa.

# Diseño

Bueno en base al análisis, tomamos un nuevo rumbo en torno a como hicimos el programa.

Primero que todo, tenemos 3 clases principales. Estas son la clase Paciente, Estudio, Departamento. Luego tenemos tres clases más que son una lista que heredan de las clases mencionadas anteriormente. La lista de esas clases la hicimos como clases nuevas ya que vimos que era más higiénico tenerla que declarar la lista de la forma normal.

Aparte tenemos una clase estática llama Funciones, esta clase mayormente es donde tenemos las funciones "Resources" del programa. Luego tenemos una clase llamada DiccionarioDept, su función esta tal cual el nombre. Por ultimo tenemos la clase llamada SaveData, que es donde guardamos toda la información de la corrida actual del programa.

Utilizamos base de datos para la lectura y muestreo de los estudios. Para guardar la información actual de la corrida utilizamos Serializacion (Archivos) como almacenamiento persistente.

De acuerdo a la interfaz que elegimos para el programa, cubrimos con facilidad todas las "restricciones" del programa.

El enfoque que utilizamos para el registro de pacientes es:

1. Seleccionamos el/los estudios de el/los pacientes.

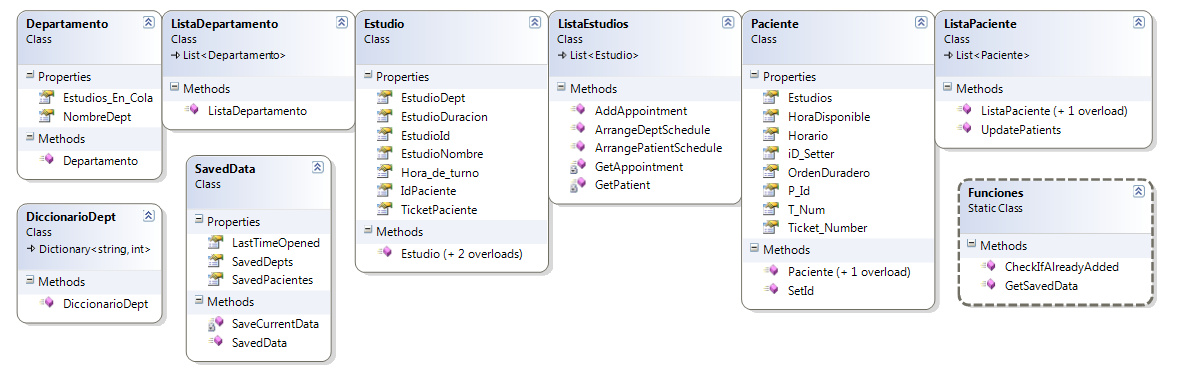
2. Se crea el paciente y con su lista de estudios.

3. Se organizan los estudios del paciente por más/menos duradero.

4. Se procede a agregar el/los estudio/s de cada paciente en horario disponible del departamento de ese estudio.

Luego si se cierra el programa mientras todavía haya pacientes con estudios pendientes se llama a la función GetSavedData y se buscan todo lo guardado. Luego se llama a una función para que chequee cuales estudios ya han pasado durante ese lapso de tiempo y borrarlos.

Diagrama de Clases:



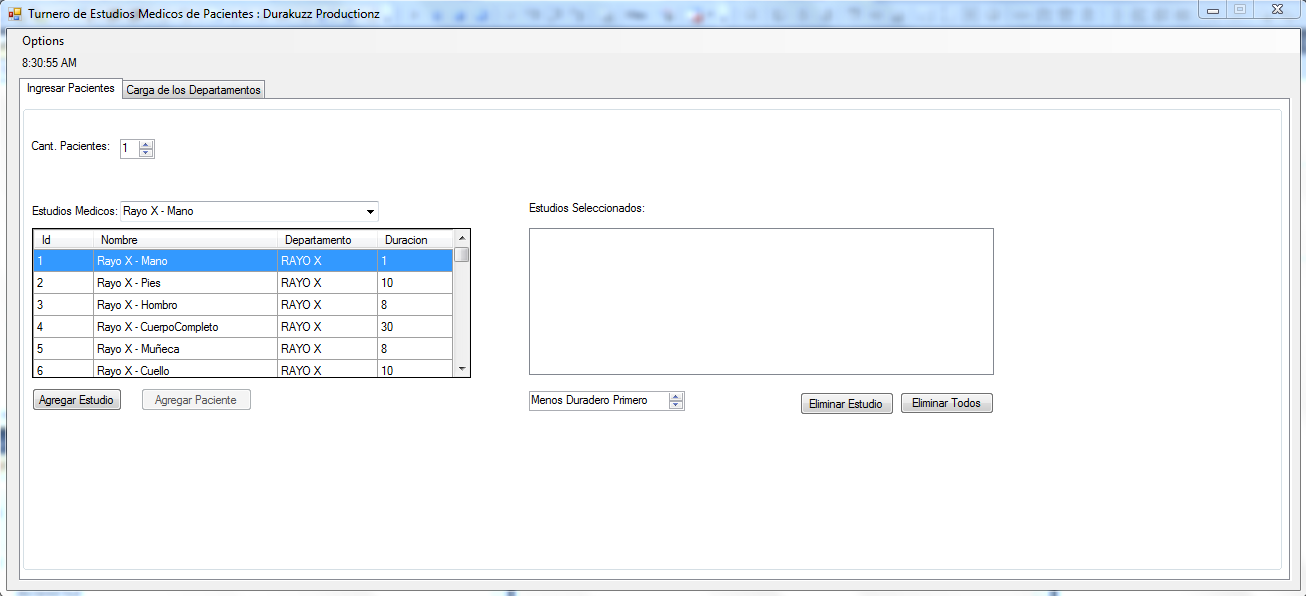
# Código Fuente

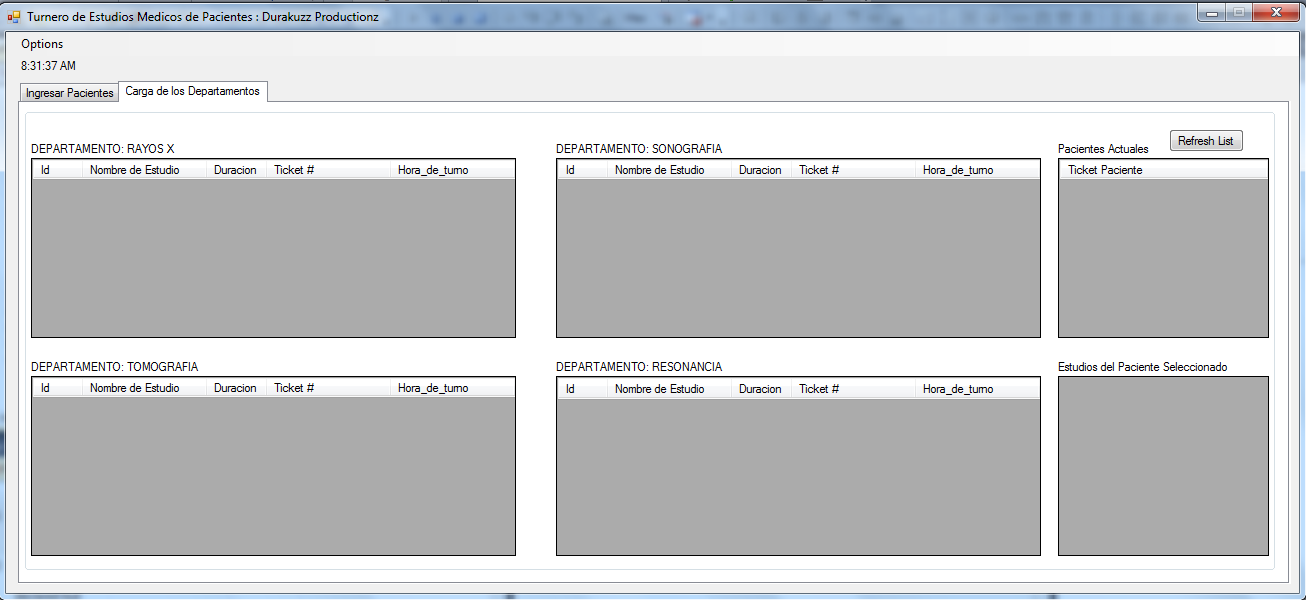
Esta adjunto al archivo .rar

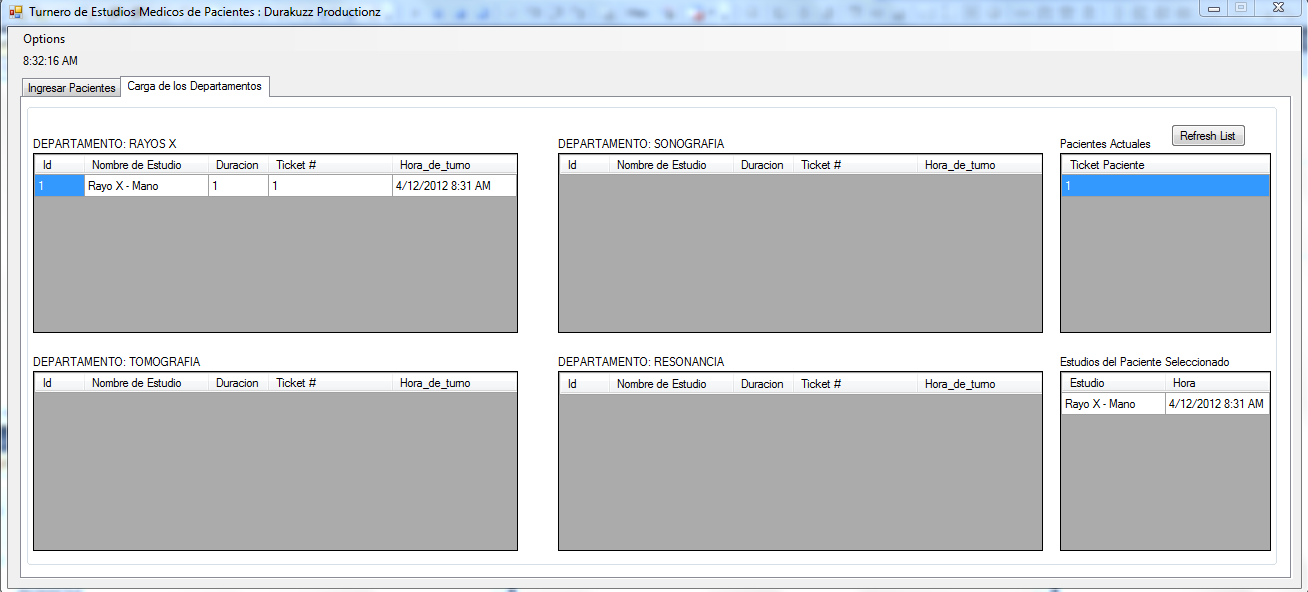
# Condiciones de Error

Por ahora no tiene.

# Ejemplos de Uso







# Notas al Profesor

En caso de que desee hacer un señalamiento al profesor acerca de su trabajo, incluirlo bajo este título al final.