**REQUERIMIENTOS FUNCIONALES:**

El sistema debe estar en la capacidad de:

**Req1. Gestionar** *los turnos por tipos de turnos dependiendo del nombre del tipo de turno y su tiempo de duración el cual es un numero decimal positivo, este número representa la cantidad de minutos de duración del turno por lo cual 1,25 representa 1min con 15segundos o 3 minutos y medio sería igual a 3,5 etc. El usuario tendrá la capacidad de crear nuevos turnos, pero cada uno de ser diferente es decir que tengan nombre único.*

**Req2.** ***Asignar*** *un turno a un usuario por medio de su número y tipo de documento de identidad, y el tipo de turno. Cada usuario solo puede tener un turno asignado a la vez el mismo día, es decir solo puede pedir otro turno después de ser atendido o pedirlo para otro día. Una vez se asigne el turno a la persona este se activará y apenas sea atendido, es decir, cuando se acabe el tiempo del turno, este se desactiva y el usuario puede volver a pedir otro turno. El turno asignado será una continuación del ultimo turno que se asigno antes de este, si es el primero se asignara el primer turno que corresponde al valor A00.*

***Req3. Registrar*** *un usuario nuevo al sistema con su tipo de documento de identidad, numero de documento, nombres y apellido (todos estos datos son obligatorios), además también puede diligenciar la dirección y número telefónico, pero estos son opcionales. El sistema no puede registrar dos veces el mismo usuario, es decir, que no tengan el mismo número y tipo de documento de identidad.*

***Req4. Avanzar*** *todos los turnos que hallan pasado hasta la fecha, es decir que si en el Sistema ya pasaron 10 turnos, al darle avanzar los 10 turnos se pondrán como atendidos y se genera aleatoriamente si el usuario estuvo o no presente durante su llamado.*

**Req5. Desplegar** *visualmente la hora y fecha del Sistema, indicándola en el formato: Año/mes/día/hora/minuto/segundo. Teniendo en cuenta que la hora puede ser modificada por el usuario. Esta se ejecutará cada vez que el usuario termine de ejecutar una acción, es decir cuando se vuelva a mostrar el menú o cuando le de a la opción de desplegar directamente.*

**Req6. Actualizar** *fecha y hora del Sistema, teniendo en cuenta como base la Hora actual de Sistema de cómputo. Esta puede actualización puede ocurrir de dos maneras diferentes, la primera es que le usuario indique la fecha y hora deseada, la cual debe ser mayor a la hora actual del sistema y la segunda es que se iguale a la fecha y hora del Sistema de cómputo. No importa cual de los dos se use, el tiempo debe seguir transcurriendo común y corriente, es decir, que los datos se vallan actualizando de igual manera a como pasa el tiempo. Una vez actualizado la fecha y hora se debe colocar todos los turnos como atendidos y generar aleatoriamente si dichas personas estuvieron presentes o no en el momento de ser atendidas*

**Req7. Generar.**

* *Generar un reporte con todos los turnos que un usuario ha generado en cuestión indicando el código del turno, si fue o no atendido y si el usuario estaba presente a la hora de atención del turno. guardando la información en un archivo de texto o mostrando por pantalla si así lo desea el usuario, también puede escoger ambas si lo desea.*
* *Generar un reporte con todos los usuarios que han tenido asignado el turno indicado por el usuario, guardando la información en un archivo de texto o mostrando por pantalla si así lo desea el usuario, también puede escoger ambas si lo desea.*
* *Generar aleatoriamente usuario registrados en el Sistema, la cantidad de usuarios debe ser indicados al sistema, y deben cumplir las condiciones requeridas para asignar un usuario, también el numero de usuario que puede generar máximo son de 4000 y esto incluye también a los generado anteriormente, es decir que la suma de todos los usuarios generados al azar es de 4000.*
* *Generar aleatoriamente turnos para los usuarios existentes, el usuario tendrá la capacidad de escoger cuantos días de turnos automáticos se generarán, teniendo en cuenta que siempre empezarán desde el día actual y que cantidad de turnos por día, ya que un usuario no puede repetir turno hasta ser atendido esta cantidad no puede superar cantidad máxima de usuarios registrados.*

**Req8. Suspender** *a un usuario que no ha estado presente durante sus dos últimos turnos, impidiéndole poder pedir otro turno en los siguientes dos días*

**REQUERIMIENTOS NO FUNCIONALES**

*El sistema debe estar en capacidad de:*

**Req1. Definir** *la clase Fecha y Hora deben estar definidas en el paquete modelo.*

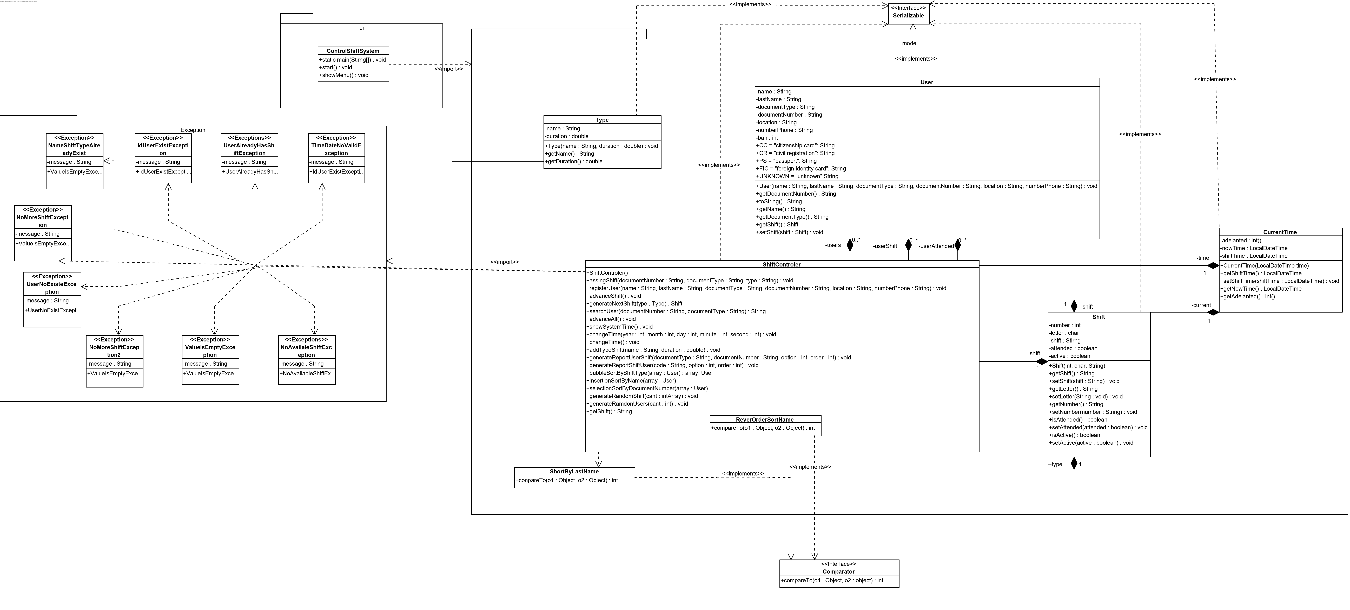
**Req2. Guardar** *toda la información del sistema debe ser persistente, es decir que guarde toda la información una vez se cierre el programa, además se debe usar el método de serialización para lograr esto*

**Req3. Tomar** *los valores de los nombres y apellidos, de los usuarios generados aleatoriamente, de un archivo generado por alguna herramienta web que facilite este trabaja, también se incluyen los archivos de documento de identidad, dirección y teléfono.*

**Req4. Indicar** *cuanto tiempo tardo el usuario en ejecutar una acción específica, es decir se cuenta el tiempo de cuando inicia una acción y cuando la termina y se muestra la diferencia por pantalla indicando que ese fue el tiempo que si acción tardo en ejecutarse. Debe tenerse en cuanta que ese tiempo no incluye el tiempo que se demore el usuario es digitar lo que le pide el sistema sino, el tiempo que tarda el sistema en ejecutar la acción.*

**Req5. Utilizar** *en alguna parte del modelo usando los tres siguientes algoritmos los cuales deben estar implementados por lo menos una vez:*

* *Ordenamientos de selección, burbuja e inserción, los cuales pueden ser utilizados sobre primitivos o sobre objeto.*
* *Búsqueda secuencial y binaria*
* *Ordenamientos con el método sort de la clase Arrays o la clase Collections, utilizando las interfaces Comparable y Comparator como una clase del modelo, y otra clase Comparator como una clase anónima, un Comparator al revés con el método reverseOrder de la clase Collections, y un comparador de una clase ya implementada en el API de java, como las Clases String o Integer. Cada uno de estos ordenamientos deben estar implementados por lo menos una vez.*



**INFORME DE LA IMPLEMENTACION**

Los algoritmos de ordenamiento se utilizaron al generar los reportes, los cuales se le pregunta al usuario como quería que el reporte estuviera ordenado.

* Inserción, se ordenó los reportes por nombre de manera ascendente
* burbuja, para ordenarlos por tipos de turnos en este también se usó la comparadora de la clase String para ejecutarlo
* selección, para ordenar el reporte por el número de documento

los ordenamientos con la interfaz Comparator y Comparable se implementaron de la siguiente manera:

* Comparable, se compara el código de turno de los usuarios, para ordenar el reporte de todos los turnos que ha tenido un usuario.
* Comparator externo, se compara los apellidos del usuario y se utiliza en un algoritmo de Arrays.sort para ordenar de manera ascendente.
* Collections.reverseOrder se use para invertir el orden del Comparator externo de ordenar por apellidos, para hacer este de manera descendente.
* Comparator anónimo, se crea dentro de la clase ShiftControler y se usa para ordenar los nombres de los usuarios de manera descendente.

La búsqueda secuencial esta implementada en el algoritmo para buscar Usuarios.