REQUERIMIENTOS FUNCIONALES

Req1. Agregar nuevo usuario, verificando que no exista previamente, es necesario el tipo de documento de identidad, el número de documento, nombre apellido, teléfono, dirección. Siendo los cuatro primeros obligatorios para el registro.

Req2. Buscar usuario, mediante el número de documento de identidad, verificado que el usuario está registrado el sistema puede:

a). Asignar turno al usuario, teniendo en cuenta que deben haber creado previamente tipos de turno.

Req3. El usuario puede registrar los tipos de turnos que se van a implementar en el programa, ingresando el nombre y el tiempo que se tarda en atender el tipo de turno.

Req4. Desplegar la fecha y hora actual del sistema

Req5. Permitir al usuario modificar la hora y fecha actual, ya sea con una fecha y hora dada o la hora actual del sistema.

Req6. Atender todos los turnos hasta el momento, según la duración de los turnos y teniendo en cuenta que hay una solo persona atendiendo.

Req7. El usuario puede generar dos tipos de reportes:

a). Los turnos que una persona ha solicitado, indicando el código de turno y si estaba o no al momento de atenderlo.

b). Las personas que han llegado a tener un determinado turno.

Req8. El usuario puede suspender a una persona que no estuvo presente en los últimos dos turnos, la persona suspendida no puede pedir turnos durante 2 días.

Req9. Registrar una persona aleatoria al sistema.

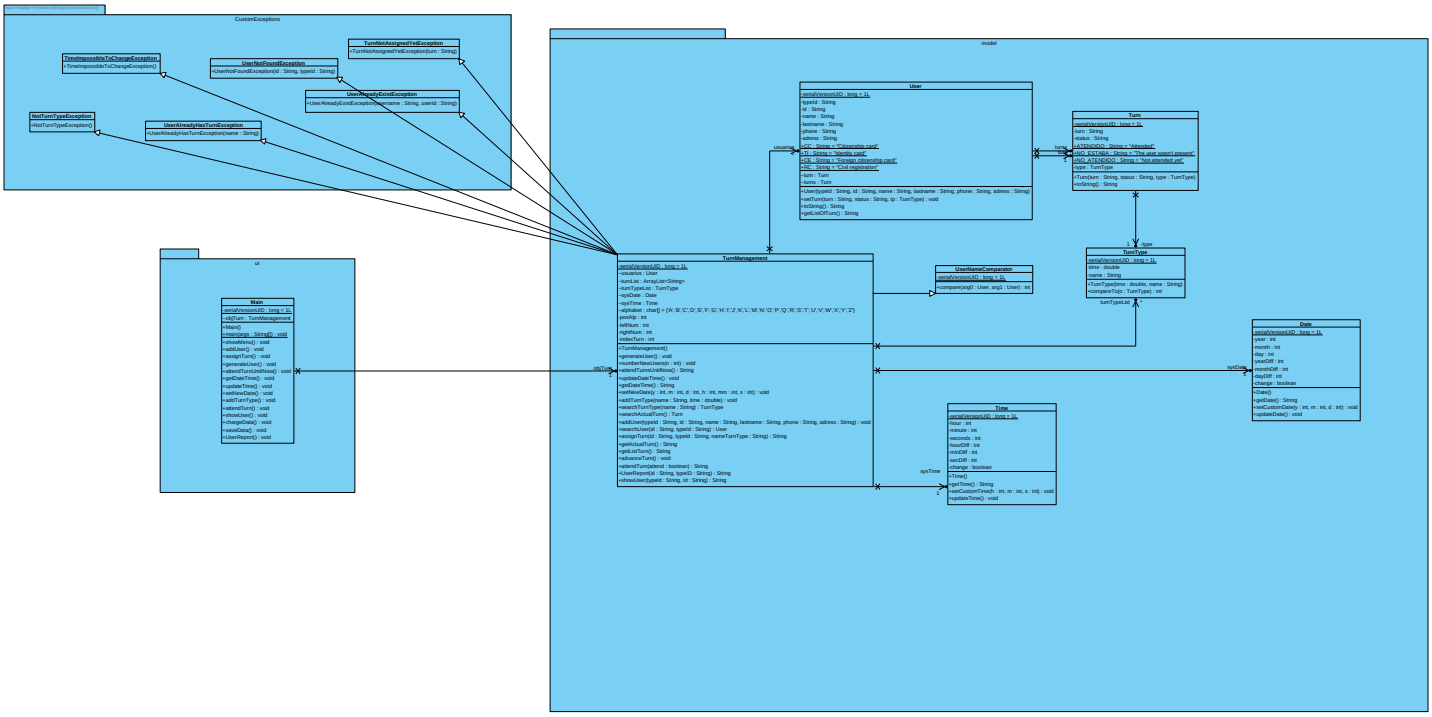
NO FUNCIONALES

Se implementó la Búsqueda binaria en el método **searchTurnType()** de la clase principal del modelo. Se ordena el **ArrayList** de **TurnType** usando la interfaz Comparable implementada en la clase **TurnType**.

Se crea la interfaz **UserNameComparator <User>** para ordenar los objetos de la clase **User** por sus nombres, esta se ejecuta cada vez que un usuario es registrado en el sistema en el método **addUser().**

URL Proyect GitHub:

https://github.com/Diego-ds/TurnControl.git


CustomExceptions
TimeImpossibleToChangeException
+TimeImpossibleToChangeException()
TurnNotAssignedYetException
+TurnNotAssignedYetException(turn : String)
UserNotFoundException
+UserNotFoundException(id : String, typeId : String)
UserAlreadyExistException
+UserAlreadyExistException(username : String, userId : String)
NotTurnTypeException
+NotTurnTypeException()
UserAlreadyHasTurnException
+UserAlreadyHasTurnException(name : String)
model
User
-serialVersionUID : long = 1L
-typeId : String
-id : String
-name : String
-lastname : String
-phone : String
-adress : String
+CC : String = "Citizenship card"
+TI : String = "Identity card"
+CE : String = "Foreign citizenship card"
+RC : String = "Civil registration"
-turn : Turn
-turns : Turn
+User(typeId : String, id : String, name : String, lastname : String, phone : String, adress : String)
+setTurn(turn : String, status : String, tp : TurnType) : void
+toString() : String
+getListOfTurn() : String
Turn
-serialVersionUID : long = 1L
-turn : String
-status : String
+ATTENDED : String = "Attended"
+NO_ESTABA : String = "The user wasn't present"
+NO_ATENDIDO : String = "Not attended yet"
-type : TurnType
+Turn(turn : String, status : String, type : TurnType)
+toString() : String
TurnType
-serialVersionUID : long = 1L
-time : double
-name : String
+TurnType(time : double, name : String)
+compareTo(o : TurnType) : int
TurnManagement
-serialVersionUID : long = 1L
-usuarios : User
-turnList : ArrayList<String>
-turnTypeList : TurnType
-sysDate : Date
-sysTime : Time
-alphabet : char[] = {'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z'}
-posAlp : int
-leftNum : int
-rightNum : int
-indexTurn : int
+TurnManagement()
+generateUser() : void
-numberNewUsersIn : int) : void
+attendTurnsLimitNow() : String
+updateDateTime() : void
+getDateTime() : String
+setNewDate(y : int, m : int, d : int, h : int, mm : int, s : int) : void
+addTurnType(name : String, time : double) : void
+searchTurnType(name : String) : TurnType
+searchActualTurn() : Turn
+addUser(typeId : String, id : String, name : String, lastname : String, phone : String, adress : String) : void
+searchUser(id : String, typeId : String) : User
+assignTurn(id : String, typeId : String, nameTurnType : String) : String
+getActualTurn() : String
+getListTurn() : String
+advanceTurn() : void
+attendTurn(attend : boolean) : String
+UserReport(id : String, typeID : String) : String
+showUser(typeId : String, id : String) : String
UserNameComparator
-serialVersionUID : long = 1L
+compare(arg0 : User, arg1 : User) : int
Time
-serialVersionUID : long = 1L
-hour : int
-minute : int
-seconds : int
-hourDiff : int
-minDiff : int
-secDiff : int
-change : boolean
+Time()
+getTime() : String
+setCustomTime(h : int, m : int, s : int) : void
+updateTime() : void
Date
-serialVersionUID : long = 1L
-year : int
-month : int
-day : int
-yearDiff : int
-monthDiff : int
-dayDiff : int
-change : boolean
+Date()
+getDate() : String
+setCustomDate(y : int, m : int, d : int) : void
+updateDate() : void
ui
Main
-serialVersionUID : long = 1L
-objTurn : TurnManagement
+Main()
+main(args : String[]) : void
+showMenu() : void
-addUser() : void
-assignTurn() : void
-generateUser() : void
-attendTurnLimitNow() : void
-getDateTime() : void
-updateTime() : void
-setNewDate() : void
-addTurnType() : void
-attendTurn() : void
-showUser() : void
+changeData() : void
+saveData() : void
+UserReport() : void
usuarios
turns
1
*
type
1
turnTypeList
*
objTurn
1
sysDate
1
sysTime
1