**Requerimientos Funcionales:**

**Req1.** Agregar un nuevo usuario. El programa debe estar en la capacidad de poder agregar un nuevo cliente en el sistema para hacer su respectiva asignación de turno. No se debe permitir agregar dos usuarios con el mismo tipo y numero de documento. Las entradas son: Tipo de documento, numero de documento, nombres y apellidos, teléfono y dirección. Dichas entradas deben ser verificadas al menos Tipo y numero del documento, nombres y apellidos. El resultado es un registro exitoso del usuario.

**Req2.** Buscar un usuario con base en su numero de documento de identidad. El programa tiene la capacidad de buscar por medio de dos datos a un usuario antes de poder asignarle un turno. Las entradas son: El tipo y numero de identidad. El usuario debe estar ya registrado en el programa. Acceder al usuario buscado se tiene como resultado.

**Req3.** Asignar un turno a un usuario. El software puede asignarle un turno a un usuario anteriormente buscado por el documento de identidad. No se le debe generar un turno a un cliente que tenga un turno activo. El resultado es un cliente con su respectivo turno para ser atendido.

**Req4.** Generar un nuevo turno. El Software debe generar un turno a un cliente que sea consecutivo al numero de turnos actual. Los turnos son con una letra empezando de la A – Z y con un numero de 00 – 99; cada vez que llega al numero 99 y la letra no sea Z, el numero vuelve a 00 con la letra consecutiva. Si llega al turno Z99 se volverá a contar desde A00.

**Req5.** Consultar el siguiente turno correctamente. El sistema tiene la facilidad de generar un turno correctamente de forma consecutiva, teniendo en cuenta los turnos ya asignados a los clientes.

**Req6.** Atender turnos. El sistema puede cambiar los estados de los turnos que ya han sido atendidos teniendo en cuenta los turnos registrados en el sistema. El resultado que se tiene, es que cada turno que se atiende pasa a un estado “inactivo”, así mismo, el mismo cliente puede volver a pedir otro turno el mismo día con un solo registro en el sistema.

**Pruebas Unitarias**

**Configuración de los escenarios:**

|  |  |  |
| --- | --- | --- |
| **Nombre** | **Clase** | **Escenario** |
| Setup1 | ControlTest | Un objeto de la clase User con documentType = TI, con id = 10068947 , name = “Andres”, lastName = “Cordoba”, phone = 300595465, address = Calle 73 # 3-69 |
| Setup2 | ControlTest | el método addUser("CC","123456789", "Pepito","Perez",3009596,"calle 30 # 3-89 ") |
| Setup3 | ControlTest | Un objeto de clase User("PA", "1006549872", "Federico", "Prieto", 300595436, "Calle 70 # 3-56")  El método addUser("PA", "1006549872", "Federico", "Prieto", 300595436, "Calle 70 # 3-56") |
| Setup4 | ControlTest | El método addUser("CC", "1006051976", "Armando", "Paredes", 300595436, "Calle 50 # 3-70")  El método addTurn("1006051976") |
| Setup5 | ControlTest | Los métodos addUser("CC", "1006051976", "Armando", "Paredes", 300595436, "Calle 50 # 3-70")  addUser("TI", "31899407", "Gloria", "Salamanc", 301528611, "Calle 70 # 3-90")  addUser("CE", "19281218", "Armando", "Paredes", 316441558, "Calle 50 # 3-10")  addUser("PA", "1234568975", "Armando", "Paredes", 301528353, "Calle 50 # 3-00") |

**Diseño de casos de prueba**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo de la Prueba:** Probar el método agregar un usuario, cuando existe o no el usuario y cuando existen o no usuarios en el sistema. | | | | |
| **Clase** | **Método** | **Escenario** | **Valores de entrada** | **Resultado** |
| Control | addUser |  | "TI", "100658947", "andres", "cordoba", 300595465, "Calle 73 # 3-69" | "The user has been created" |
| Control | addUser |  | "CC", "1006051976", "Armando", "Paredes", 300595436, "Calle 50 # 3-70" |  |
| Control | addUser | Setup5 |  | “the user is already created”  “the user has been created” |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo de la Prueba:** Probar el buscar un usuario, cuando existe o no el usuario y cuando existen o no usuarios en el sistema. | | | | |
| **Clase** | **Método** | **Escenario** | **Valores de entrada** | **Resultado** |
| Control | searchUser | Setup1 | "100658947" | "the user has not been created" |
| Control | searchUser | Setup2 | "123456789" | SearchUser != null |
| Control | SearchUser |  | "31899407"  "19281218" | searchUser == null |
| Control | SearchUser | Setup3 | new User("CC", "123456798" , "Alexander", "Prieto", 315894650, "Calle 9 # 2-48")  addUser("CC", "123456798" , "Alexander", "Prieto", 315894650, "Calle 9 # 2-48")  "1006549872";  "123456798" | SearchUser == user.getId() |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo de la Prueba:** Probar el método mostrar turnos habiendo turnos por atender | | | | |
| **Clase** | **Método** | **Escenario** | **Valores de entrada** | **Resultado** |
| Control | showTurns | Setup4 | Turn [A0] | The turns are the same |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo de la Prueba:** Probar el método cambiar estado para atender un turno habiendo turnos por atender | | | | |
| **Clase** | **Método** | **Escenario** | **Valores de entrada** | **Resultado** |
| Control | setStatus | Setup4 | A 0 “yes” | "the Turn [A0] had been attended" |