**Departamento de TIC**

**Algoritmos y Programación II**

**Unidad 2 - Laboratorio 1**

**Su entrega debe incluir los siguientes elementos**

Diseño de su solución

* (11%) Especificación de Requerimientos Funcionales.

No utilice el formato (tabla) de AP1, sino que escriba sus requerimientos como párrafos que inician con un verbo después de esta oración (ejemplo):

El sistema debe estar en la capacidad de:

**Req1.** ***Agregar***un nuevo usuario con nombre, dirección, teléfono y código. No permitirá que se agreguen usuarios con el mismo código *(y en el resto del párrafo deben ir todas las especificaciones relacionadas con este requerimiento)*.

* (20%) Diseño de casos de prueba.
* (8%) Diseño de diagrama de clases de las pruebas. (Visual Paradigm con FONDO BLANCO).
* (8%) Diseño del diagrama de clases del modelo. (Visual Paradigm con FONDO BLANCO).
* (8%) Diseño del diagrama de clases de las excepciones. (Visual Paradigm con FONDO BLANCO).

Construcción de la solución (lenguaje de programación)

* (10%) Implementación del modelo y la interfaz de usuario en consola con cada una de sus funcionalidades.
* (5%) Contratos de los métodos y generación de la documentación utilizando javadoc.
* (10%) Implementación de las excepciones.

Construcción de las pruebas unitarias automáticas (lenguaje de programación)

* (20%) Con base en el diseño de casos de prueba y utilizando JUnit implemente las pruebas unitarias automáticas..

**Enunciado**

**SISTEMA DE CONTROL DE TURNOS PARA ATENCIÓN AL USUARIO**

Una empresa lo ha contratado para que desarrolle un programa en su departamento de servicio al cliente que resuelva el problema del control y atención de las personas que llegan a sus centros de servicio para ser atendidas.

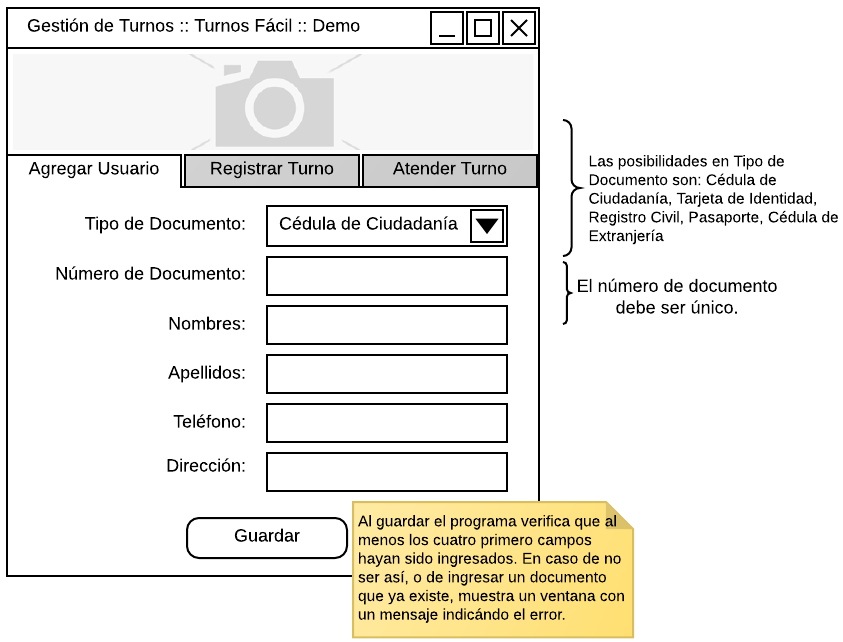
El programa será utilizado por un empleado de la empresa que se encuentra a la entrada del centro de servicio y con el cual siempre se comunicará, en primera instancia, la persona que ingrese para ser atendida.

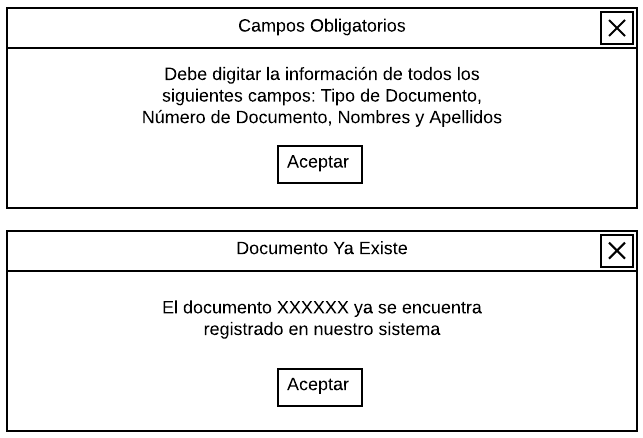
El empleado lo buscará con su número de documento de identidad y luego le asignará un turno para ser atendido. El turno está formado por una letra y un número entre 00 y 99. El primer turno es el A00, el siguiente el A01 y así sucesivamente. Cuando se llegue al último número de una letra (el 99), se cambiará al primer número de la letra siguiente y así continuará. Por ejemplo del A99 sigue el B00.

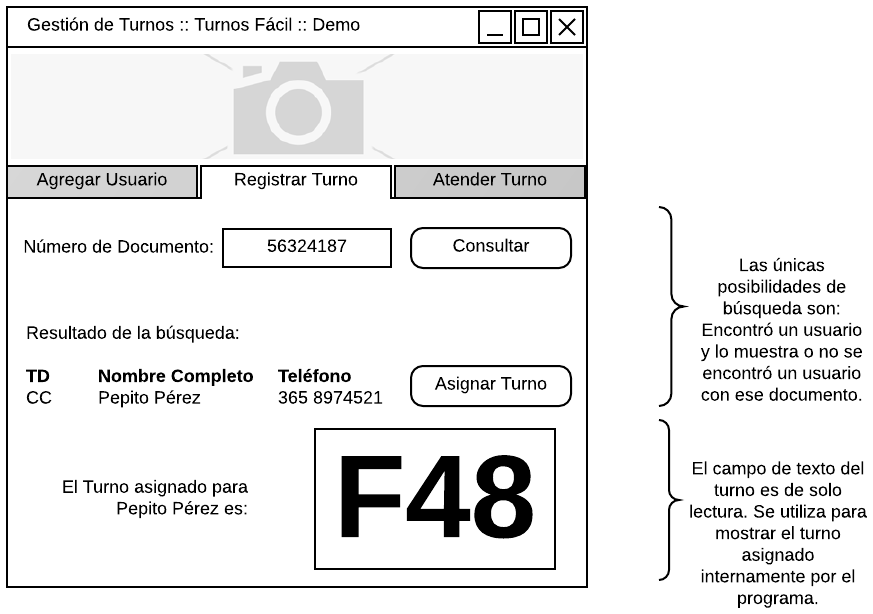
El programa también tendrá la posibilidad de registrar a un nuevo usuario en caso de que no se encuentre ya registrado en el sistema. Los datos de un usuario son tipo de documento de identidad, número de documento, nombres, apellidos, teléfono y dirección. Al agregar a un nuevo usuario se debe validar que se diligencien al menos el tipo de documento, el número de documento, los nombres y los apellidos de la persona (es decir son obligatorios).

Por otro lado el programa debe permitir ir avanzando en el turno en la medida en que otro empleado va atendiendo a cada cliente de acuerdo con el turno que tiene asignado. Esa persona que va atendiendo tiene la posibilidad de indicar al programa si atendió realmente al usuario con ese turno o si no lo atendió porque ya no se encontraba en el lugar cuando fue llamado para ser atendido.

La última funcionalidad explicada debería estar en un programa diferente (o al menos en un perfil diferente) al de las demás funcionalidades ya que es utilizado por un empleado con un rol diferente al de las primeras funcionalidades explicadas, sin embargo por ser ésta la primera versión del programa y con motivos de demostración, todas las funcionalidades serán implementadas en el mismo programa. Cada una en una pestaña diferente, tal como se muestra en los siguientes mockups. Los mockups fueron diseñados para una segunda versión del programa, usted deberá basarse en ellas para guiarse en las opciones que el programa debe tener pero debe desarrollar su interfaz utilizando un menú de opciones en una consola de texto.







|  |  |
| --- | --- |
| e. Consulta correctamente el siguiente turno por atender:  i. Habiendo turnos por atender  f. Atiende un turno asignado a una persona:  i. Habiendo turnos por atender  **REQUERIMIENTOS FUNCIONALES**  El sistema debe estar en la capacidad de: | Usted debe diseñar e implementar al menos las siguientes pruebas a la operación que:   1. Agrega un nuevo usuario funciona correctamente:    1. Si el usuario no existe    2. Si el usuario ya existe    3. Si no hay usuarios en el programa    4. Si hay usuarios en el programa 2. Busca un usuario con base en su número de documento de identidad:    1. Si ya existe el usuario a buscar    2. Si no existe el usuario a buscar    3. Si no hay usuarios en el programa    4. Si hay usuarios en el programa 3. Asigna un turno a un usuario:    1. Si el usuario tiene un turno activo, entonces debe probar que retorne el turno activo y no genere otro    2. Si el usuario no tiene un turno activo entonces le genera uno nuevo 4. Genera un nuevo turno:    1. Consecutivo al último turno asignado    2. Si es el primer turno genera el turno A00    3. Si el último turno generado es el D99 entonces genera el turno E00    4. Si el último turno generado es el Z99 entonces genera el turno A00 |

* **Req1.** ***Agregar***un nuevo cliente con nombres, apellidos, tipo de documento, número de documento, dirección y teléfono. No permitirá que se agreguen usuarios con el mismo tipo de documento y número.
* **Req2. Buscar** un cliente dentro del sistema de control de turnos, el cliente se buscará ingresando su tipo de documento y su número. Esto con el objetivo de agregar un cliente, si al buscarlo ya existe no se puede agregar el mismo, y en el momento de asignar un turno se debe buscar el cliente de igual manera para ser asignado a un turno correspondiente.
* ***Req3. Asignar*** un cliente a un turno, el turno está formado por una letra y un número entre 00 y 99. Cada vez que el número llegue a 99 se avanza a la siguiente letra. Hasta llegar a la Z99, en ese momento se reinician los turnos y vuelve a iniciar en el A00.
* ***Req3.1 Crear*** cada uno de los turnos, cada vez que se va asignando el turno se va avanzando de número y cuando se acaban los números vuelve a iniciar con la siguiente letra. Luego de esto vamos creando los turnos dentro de nuestro control de turnos, cada turno con el cliente que se desea atender.
* ***Req3.2 Cambiar*** el estado de un turno, al crear un turno se cambia su estado a pendiente, que quiere decir que existe un cliente pendiente de ser atendido.
* ***Req4. Atender***  un turno, cuando un empleado decida atender el siguiente turno el control de turnos va a buscar el siguiente turno que tenga un estado de pendiente, en ese momento el control nos dice el turno que sigue por atender, y cambia el estado del turno a atendiendo, para que mientras se espera a atender al cliente, el control de turnos identifique el turno que sigue por atender en caso de que existan más empleados atendiendo clientes.
* ***Req5. Finalizar*** un turno, empleado que atenderá este turno tiene dos opciones, marcar que ya atendió al cliente e inmediatamente el estado del turno cambia a atendido, y marcar que el cliente no se presentó, esto para finalizar el turno y que el sitio de trabajo del empleado pueda recibir el siguiente turno pendiente.

**DISEÑO DE CASOS DE PRUEBA**

**Configuración de los Escenarios**

|  |  |  |
| --- | --- | --- |
| **Nombre** | **Clase** | **Escenario** |
| setup1 | TurnControlTest | Un objeto de la clase TurnControl con siete objetos de la clase cliente:  - El primero con typeOfDocument=”CC”, ID=” 1005977”, name=” Carlos”, surnames=”Restrepo”, phone= “3124567890”, address=” Quinta con quinta”  - El segundo con typeOfDocument=”CC”, ID=” 324567”, name=”David”, surnames=”Medina ”, phone= “3234514587”, address=””  - El tercero con typeOfDocument=”CC”, ID=” 123453456”, name=”Fernando ”, surnames=”Carvajal ”, phone= “”, address=””  - El cuarto con typeOfDocument=”CC”, ID=” 334441”, name=” Camilo”, surnames=” Realpe”, phone= “”, address=””  - El quinto con typeOfDocument=”CC”, ID=” 1005977”, name=” Carlos”, surnames=”Restrepo”, phone= “3124567890”, address=” Quinta con quinta”  - El sexto con typeOfDocument=”CC”, ID=” 13588732”, name=”Camilo ”, surnames=”Carvajal ”, phone= “”, address=””  - El séptimo con typeOfDocument=”CC”, ID=””, name=”Daniel ”, surnames=”Rodríguez ”, phone= “”, address=”” |

**Diseño de Casos de Prueba**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo de la Prueba:** (**setup1**) Verificar que el método addClient de la clase TurnControl funcione correctamente, cuando se intenten agregar clientes que ya estén agregados no serán agregados, además de clientes a los que les falten datos. Sin embargo, si el cliente no existe y tiene todos sus datos completos será agregado. | | | | |
| **Clase** | **Método** | **Escenario** | **Valores de Entrada** | **Resultado** |
| TurnContol | addClient | setup1 | Ninguno | Se agrega los objetos:   * Primero. * Segundo. * Tercero. * Cuarto. * Sexto   Los otros dos no se agregan debido a que uno está repetido y otro tiene los datos incompletos. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo de la Prueba: (compareSize)** Verificar que el método addClient si agregé los clientes que debe agregar, por lo tanto verificamos que el tamaño de nuestra lista sea el que se espera dependiendo de los objetos agregados en el setup1. | | | | |
| **Clase** | **Método** | **Escenario** | **Valores de Entrada** | **Resultado** |
| TurnControl | addClient | Setup1 |  | El tamaño de la lista debe ser igual a 5. Ya que se deben agregar 5 objetos. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo de la Prueba: (searchTest)** Verificar que el primer usuario si está agregado a través del método buscar cliente. Se le ingresa su tipo y número de documento y el método debe retornar el cliente. Si el método no encuentra al cliente, quiere decir que este no existe, y por lo tanto retorna null. | | | | |
| **Clase** | **Método** | **Escenario** | **Valores de Entrada** | **Resultado** |
| TurnControl | searchClient | Setup1 | Tipo de documento: “CC”  Número de documento:  “1005977” | Cliente en la primera posición. |
| TurnControl | searchClient | Setup1 | Tipo de documento: “Passport”  Número de documento:  “Aa124576g” | Null |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo de la Prueba: (assignTurnTest)** Verificar que los turnos si son asignados de la manera correcta. | | | | |
| **Clase** | **Método** | **Escenario** | **Valores de Entrada** | **Resultado** |
| TurnControl | assignTurn | Setup1 | Ninguno | Turno asignado es =” A00” |
| TurnControl | assignTurn | Setup1 | Letra actual=’D’  Número actual= 99 | El siguiente turno asignado debe ser = “E00” |
| TurnControl | assignTurn | Setup1 | Letra actual=’Z’  Número actual= 99 | El siguiente turno asignado debe ser =” A00” |