

README OBLIGATORIO BASES DE DATOS

Este documento se divide en dos partes, primero se va a explicar la función de cada una de las clases y como se relacionan y en la segunda parte el instructivo de uso para poder testear cada una de las posibles situaciones de manera correcta. El proyecto se encuentra en el siguiente repositorio: <https://github.com/brunoalbin2311/obligatorioBD.git>

Clases y sus funciones

En nuestro programa tenemos la clase que contiene el método main, la cual es desde donde vamos a ejecutar el programa llamada **ObligatorioBD2** (el nombre simplemente se debe a cuando estábamos haciendo el proyecto surgió otra idea completamente diferente y para no borrar lo que teníamos para usar de guía creamos otro con 2 al final), en esta clase se hacen solo dos cosas, se crea una instancia de la clase "PantallaIniciar", que es la primer pantalla que vamos a ver en el programa y está la configuración para el envío de mails, para poder testear de forma correcta el envío de mails, recomiendo cambiar las siguientes líneas de código:

```
scheduler.scheduleAtFixedRate(recordarFechaAgendaTask, 0, 24, TimeUnit.HOURS);  
scheduler.scheduleAtFixedRate(recordarVencimientoCarnetTask, 0, 24, TimeUnit.HOURS);
```

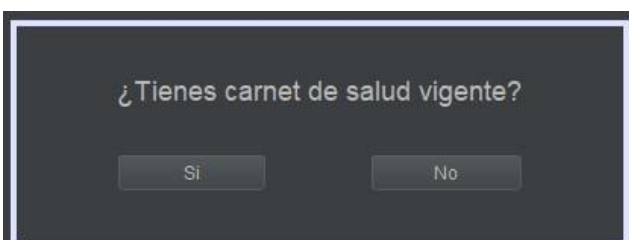
Los 2 últimos parámetros poner "1, TimeUnit.MINUTES", esto hace que los mails se envíen cada 1 minuto (siempre que ejecutemos el programa se van a enviar los mails al instante) en lugar de cada 24 horas

También tenemos 10 clases de tipo JFrame que podríamos decir que es el "front" de nuestro programa y estas son:

- **PantallaIniciar:** Esta pantalla es la primera que vemos al ejecutar nuestro programa, simplemente tiene 3 botones que nos permite elegir si vamos a registrarnos, iniciar sesión o ingresar como administrador (si en el momento que se ejecuta esta pantalla la fecha para realizar el formulario no es la correcta, solo se nos va a permitir iniciar sesión como administrador ya que este es el único que la puede cambiar.



- **PantallaPreguntaCarnetReg:** Esta pantalla la vemos cuando seleccionamos el botón "registrarse" en "PantallaIniciar", simplemente tiene dos botones que nos permite saber si tiene o no carnet vigente para saber si debemos pedir los datos de su carnet o agendar fecha.



•

PantallaRegistrarDatosCarnet: Esta pantalla la vemos cuando le damos “si” en la pantalla anterior, y su función es pedir todos los datos necesarios al funcionario que NO ESTABA REGISTRADO y además los datos del carnet que suponemos que al responder “si” en la pantalla anterior lo tiene vigente, podría pasar que se equivocó y no lo tiene vigente, alguna alerta o algo, pero por temas de tiempo y de no darle muchas vueltas a este tipo de decisiones tomamos esa suposición.

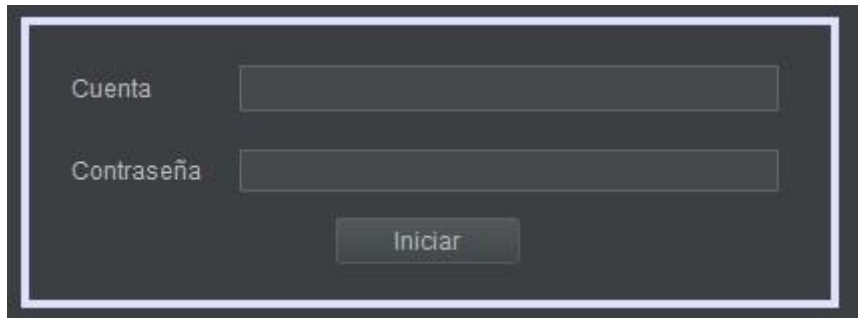
A screenshot of a web form titled 'PantallaRegistrarDatosCarnet'. The form is dark-themed with light-colored text. It contains several input fields for personal data: 'Cuenta', 'Contraseña', 'Cédula', 'Nombre', 'Apellido', 'Fecha de nacimiento' (with a calendar icon), 'Dirección', 'Correo', and 'Teléfono'. Below these is a section for the health card ('Carnet de salud') with a button 'Adjuntar comprobante', and two more date fields: 'Fecha emisión' and 'Fecha vencimiento' (both with calendar icons). At the bottom is a large 'Enviar' button.

- **PantallaRegistrarDatosCarnet:** Funciona exactamente igual que la anterior, pero esta la vemos cuando el funcionario presiona el botón de “no”, por lo que, en lugar de pedir los datos del carnet, pide una posible fecha para agendarse a sacar el carnet.

A screenshot of the same web form, but in a state where it asks for a date. The input fields for personal data are still present. Below the 'Teléfono' field, there is a text prompt: 'Seleccione una fecha para sacar su carnet de salud'. Below this prompt is a date picker widget. At the bottom is the 'Enviar' button.

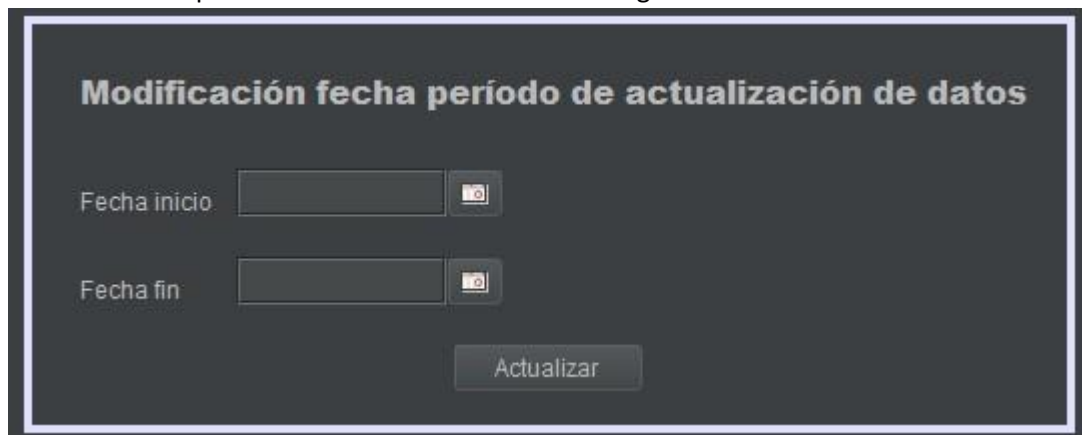
-

PantallaLoginAdmin: Esta pantalla la vemos cuando presionamos el botón “administración” en la pantalla al iniciar y su función es pedir los datos para iniciar sesión como administrador (si se intenta iniciar con una cuenta válida de un funcionario no nos va a dejar).



A screenshot of a login form titled "PantallaLoginAdmin". It features two input fields: "Cuenta" (Account) and "Contraseña" (Password). Below these fields is a button labeled "Iniciar" (Start/Log In). The form is enclosed in a dark border with a light blue inner border.

- **PantallaAdmin:** Esta pantalla nos permite modificar la fecha la cual va a estar permitido tanto como completar el formulario como también registrarse.



A screenshot of a form titled "Modificación fecha período de actualización de datos". It contains two date input fields: "Fecha inicio" (Start Date) and "Fecha fin" (End Date). Each field has a calendar icon to its right. Below the date fields is a button labeled "Actualizar" (Update). The form is enclosed in a dark border with a light blue inner border.

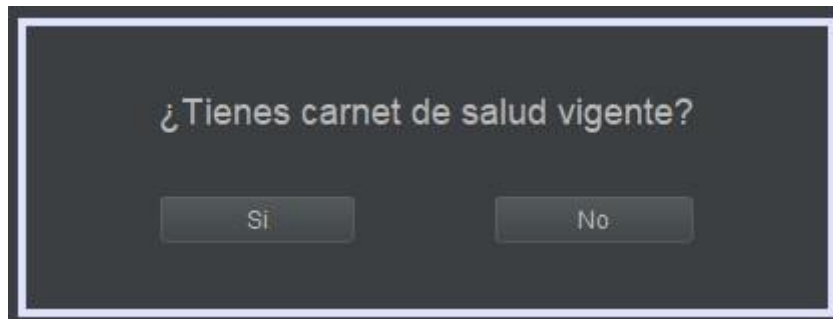
- **PantallaLogin:** Esta pantalla la vemos cuando presionamos el botón “iniciar sesión” en la pantalla para iniciar y su función es pedir los datos para iniciar sesión como funcionario, es exactamente igual a la que se usa para iniciar como administrador pero este obviamente debe haber tenido ya una previo registro, si se intenta iniciar con la cuenta del administrador aquí, nos va a avisar que hay un error en la cuenta o contraseña ya que no tiene sentido que el administrador quiera iniciar sesión para completar el formulario.



A screenshot of a login form titled "PantallaLogin". It features two input fields: "Cuenta" (Account) and "Contraseña" (Password). Below these fields is a button labeled "Iniciar" (Start/Log In). The form is enclosed in a dark border with a light blue inner border.

PantallaPreguntaCarnetComp: Verifica exactamente igual que la otra pantalla, pero los botones nos abren las pantallas para completar los datos y no registrarnos de 0.

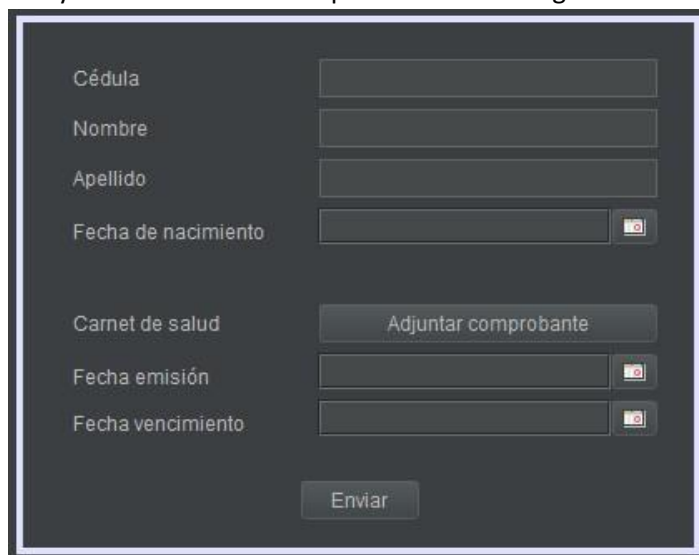
-



¿Tienes carnet de salud vigente?

Si No


- **PantallaCompletarDatosCarnet:** Si le damos a si en la pantalla anterior se nos abre esta pantalla que pide los datos a actualizar y los datos del carnet que funciona igual que al registrarse y actualiza los datos dependiendo de el logID con el cual inició sesión,




Cédula


Nombre

Apellido

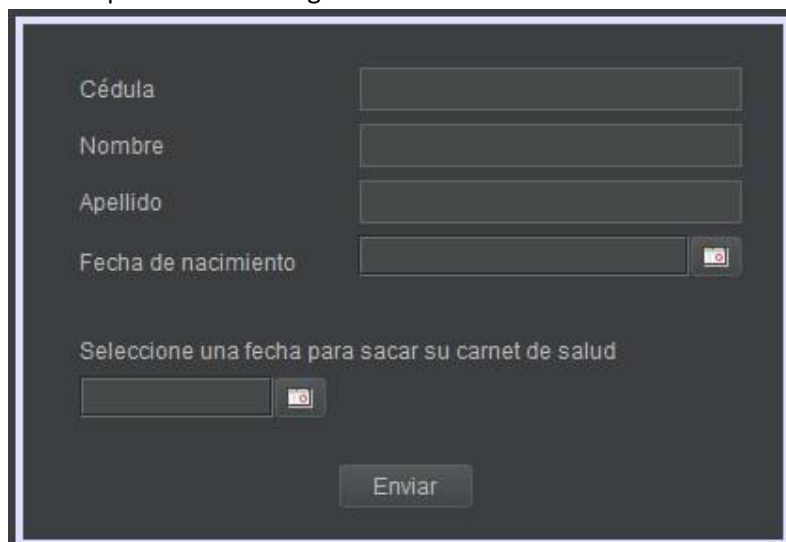
Fecha de nacimiento 

Carnet de salud Adjuntar comprobante

Fecha emisión 

Fecha vencimiento 


- **PantallaCompletarDatosNoCarnet:** Se abre si ponemos que no tenemos carnet y al igual que al registrarse nos pide una fecha de agenda y al igual que en la anterior actualiza los datos dependiendo del logID.




Cédula

Nombre

Apellido

Fecha de nacimiento 

Seleccione una fecha para sacar su carnet de salud



En cada botón de cada pantalla se puede ver la lógica como se utilizan los métodos y las validaciones, por ejemplo este es el botón de enviar de la pantalla “CompletarDatosCarnet”:

```
private void botonCompletarDatosActionPerformed(java.awt.event.ActionEvent evt) {
    Funcionario funcionario = new Funcionario();
    CarnetDeSalud carnet = new CarnetDeSalud();

    switch (funcionario.verificarFuncionario(jTextFieldCedula, jTextFieldNombre, jTextFieldApellido, jTextFieldFechaNacimiento)) {

        case 3 -> JOptionPane.showMessageDialog(null, "¡ERROR! El campo 'CEDULA' se encuentra vacío.");
        case 33 -> JOptionPane.showMessageDialog(null, "¡ERROR! El campo 'CEDULA' no tiene una cédula válida.");
        case 333 -> JOptionPane.showMessageDialog(null, "¡ERROR! El campo 'CEDULA' tiene un cédula no disponible, porfavor ver.");
        case 4 -> JOptionPane.showMessageDialog(null, "¡ERROR! El campo 'NOMBRE' se encuentra vacío.");
        case 44 -> JOptionPane.showMessageDialog(null, "¡ERROR! El campo 'NOMBRE' tiene caracteres no permitidos, ingrese solo.");
        case 5 -> JOptionPane.showMessageDialog(null, "¡ERROR! El campo 'APELLIDO' se encuentra vacío.");
        case 55 -> JOptionPane.showMessageDialog(null, "¡ERROR! El campo 'APELLIDO' tiene caracteres no permitidos, ingrese solo.");
        case 6 -> JOptionPane.showMessageDialog(null, "¡ERROR! El campo 'FECHA DE NACIMIENTO' se encuentra vacío.");
        case 66 -> JOptionPane.showMessageDialog(null, "¡ERROR! El campo 'FECHA DE NACIMIENTO' tiene una fecha no válida.");
        default -> {
            switch (carnet.verificarCarnet(jTextFieldFechaEmision, jTextFieldFechaVencimiento, getUbicacionArchivo())) {
                case 1 -> JOptionPane.showMessageDialog(null, "¡ERROR! No ingreso un comprobante aún");
                case 2 -> JOptionPane.showMessageDialog(null, "¡ERROR! El campo 'Fecha Emision' tiene una fecha no válida.");
                case 3 -> JOptionPane.showMessageDialog(null, "¡ERROR! El campo 'Fecha Vencimiento' tiene una fecha no válida.");
                case 4 -> JOptionPane.showMessageDialog(null, "¡ERROR! Su carnet no puede estar vencido");
                case 5 -> JOptionPane.showMessageDialog(null, "¡ERROR! La fecha final no puede ser anterior a la inicial.");
                default -> {
                    funcionario.actualizarDatos(jTextFieldCedula, jTextFieldNombre, jTextFieldApellido, jTextFieldFechaNacimiento);
                    carnet.insertarCarnet(jTextFieldCedula, jTextFieldFechaEmision, jTextFieldFechaVencimiento, getUbicacionArchivo());
                    dispose();
                }
            }
        }
    }
}
```

Las clases que manejan toda la lógica y hacen de nuestro “back” son las siguientes:

- **Administración:** Esta clase se usa para crear un admin con sus respectivas responsabilidades.
 - **actualizarPeriodo:** Este método hace un insert en nuestra base a la tabla “Periodos_Actualizacion” con las fechas que se seleccionen, (se inserta con un valor que se va incrementando lo cual hace que en el momento de insertar esta fecha queda como la última)
 - **fechaDisponible:** Este método hace un select a la tabla “Peridos_Actualizacion” seleccionando así la ultima fecha que se ingreso, entonces por ejemplo al iniciar sesión, se ejecuta este método y la fecha es válida le permite iniciar, sino le tira un cartel que no es la fecha disponible.
 - **verificarFechas:** Método para simplemente revisar que las fechas no sean vacías y válidas, con válidas nos referimos a que existen y que no sean anteriores al momento de la ejecución.
- **Agenda:** Esta clase se encarga de agendar funcionarios que no tienen el carnet válido.
 - **insertarAgenda:** Este método hace un insert a la tabla agenda con la cedula y la fecha que el funcionario elija.
 - **verificarFechaAgenda:** Verifica que la fecha no sea vacía y sea válida.
- **CConection:** Esta clase es la que usamos para conectarnos con la base de datos.
- **CarnetDeSalud:** Esta clase tiene toda la lógica para manejar el funcionamiento de los carnets.

- **insertCarnet:** Este método hace un insert a la tabla Carnet_Salud con los datos correspondientes del carnet que el funcionario ingrese al completar los datos o registrarse por primera vez.
- **verificarCarnet:** Verifica que se ingresen todos los campos, también el comprobante y además que las fechas sean válidas.
- **Correo:** Esta clase de encarga de crear y enviar correos.
- **Recordatorio:** Relacionada con la clase correo, esta clase se encarga de seleccionar las personas a las cuales hay que hacerles un recordatorio.
 - **recordarFechaAgenda:** Este método hace un select seleccionado todos los mails de las personas cuya fecha de agenda es futura a la de hoy y llama al método siguiente
 - **enivarRecordatorioAgenda:** Este método envía un correo a la persona recordándole la fecha que tiene agendada creando un objeto de la clase correo.
 - **recordarVencimientoCarnet y enviarRecordatorioVencimientoCarnet:** Funcionan igual que los dos anteriores pero el select se hace a los mails de las personas cuya fecha de vencimiento del carnet es pasada a la de hoy.
- **Funcionario:** Esta clase maneja casi toda la lógica del programa ya que del funcionario depende casi todas las tablas.
 - **nuevoLogin:** Este método lo que hace es ingresar a la tabla Ultimo_inicio la cuenta en el momento que se inicia sesión, esto se hace con un número que se auto incrementa, esto lo hicimos porque, cuando nosotros queremos usar la cuenta en la pantalla de registrar, no tenemos problema ya que al estar completando un campo, simplemente tenemos que tomar el valor de ese campo pero en la pantalla para completar los datos no tenemos el campo “cuenta” (siempre que hablo de cuenta hablo del logID) entonces cuando queremos completar los datos, tenemos que hacer un update pero en referencia a algo, bueno lo hacemos con haciendo un select a esta tabla con el ultimo valor insertado, ya que sería el ultimo inicio de sesión y actualizamos la cedula..nombre..etc en base a esa cuenta (explico mejor en otro método mas abajo)
 - **insertarFuncionario:** Este método toma todos los parámetros que se le pasa en la pantalla “PantallaRegistroDatos” ya sea con o sin carnet, (solo toma los datos del funcionario) y hace un insert a administración con la cuenta y el rol de funcionario, también hace un insert a la tabla login con la cuenta y la contraseña hashada y obviamente inserta el funcionario a la tabla funcionario con sus datos correspondientes
 - **actualizarDatos:** Este método hace exactamente lo mismo que el anterior con unos pequeños cambios, toma los parametros de la pantalla

“PantallaCompletarDatos” ya sea con o sin carnet y actualiza los datos de la tabla funcionario (update) dependiendo de la cuenta con la cual se inicio sesión (logId)

- **obtenerUltimoLogId:** Este método hace un select a la tabla Ultimo_inicio para poder sacar el logId de la persona a la cual se le va a hacer el update de datos con el método actualizarDatos.
- **verificarUsuario:** Este método simplemente verifica que la cuenta que va a iniciar sesión, se le pasa por parámetro el logId y la contraseña, hashlea la contraseña y compara estos valores con los de la tabla logId y si hay una coincidencia se puede decir que “inicia” sino le tira un cartel que la contraseña o la cuenta son incorrectos
- **verificarNuevoFuncionario y verificarFuncionario:** Estos métodos son todas las validaciones, por ejemplo el logID tiene q tener mínimo 8 caracteres y menos de 20, la cedula tiene que tener solo números y ser 8 dígitos, el mail tiene que terminar en “@gmail.com” o “@correo.ucu.edu.uy”, el teléfono tiene que arrancar con 09 y ser 9 números, los nombres solo letras, y las fechas tienen que ser validas y con sentido, ya que suponemos que los funcionarios son todos mayores de 18, hay varios para probar.
- **obtenerMD5:** Hashlea la contraseña, este método no lo hicimos nosotros lo de acá: <https://www.youtube.com/watch?v=ef3kenC4xa0>
- **edAdmin:** Hace un select a la tabla Administración y si la columna rol es true, devuelve true, sino false, esto lo usamos para que el admin no pueda iniciar sesión con su cuenta como funcionario y un funcionario no pueda iniciar sesión como un admin.
- **cuentaExiste, cedulaExiste, correoExiste y telefonoExiste:** Estos métodos hacen un select en la tabla funcionario revisando si ya hay un funcionario ya ingresado con alguno de esos datos.
- **Validaciones....:** Después hay varios métodos simples validaciones que mencione arriba.

Instructivo de uso

Primero que nada para ejecutar el programa de forma exitosa, lo primero que se debe hacer es levantar la base de datos, para esto debemos ir con la consola a la ubicación en la que tengamos la carpeta en la que se encuentra el docker-compose.yml y ejecutar docker-compose up (en caso de no tener docker debemos instalarlo previamente), luego, una vez esté levantado, simplemente ejecutamos el archivo llamado obligatorio.jar para iniciar la aplicación"

Cuando ejecutemos el programa se nos abre la pantalla para elegir si registrarnos, iniciar como admin o iniciar sesión como funcionario previamente registrado, si intentamos registrarnos o iniciar no nos debería dejar, ya que el período del 1 al 15 de noviembre de este año ya paso, por ende debemos iniciar como administrador, con el usuario “admin” y la contraseña “admin”, seleccionamos fechas las cuales nos convenga para que la aplicación funcione (aclarar que la fecha de inicio tiene que ser si o si futura ya que no tendría sentido por ejemplo si hoy es 5, abrir el periodo del 4 al 10, es ilógico) por lo que lo recomendable es seleccionar como inicio la

fecha de hoy y la de fin si cualquiera que sea futura, una vez le damos a actualizar, la tabla Periodos_Actualizacion se agrega estas fechas con el año y el semestre del momento que se actualiza, y un valor que se va incrementando a medida que se hacen nuevos inserts para poder tomar el ultimo, ya que es el que a nosotros nos interesa. Una vez que estamos en una fecha válida para completar el formulario tenemos varias opciones.

- **Sin registro y sin carnet:** En este caso le damos al botón de registrarnos y luego a no, vamos a completar todos los datos (todos los campos deben estar completos, aquí se pueden probar varias validaciones que están mencionadas cuando explico la clase “Funcionario” en el los métodos de verificar. Básicamente al completar todos los datos bien aquí hacemos 3 cosas, un insert a la tabla “Administracion” con una nueva cuenta y su rol, que es false ya que es funcionario, un insert a la tabla “Login” con su cuenta y su contraseña hasheada, un insert a la tabla Funcionario con toda su información y por ultimo un insert a la tabla Agenda con la cedula de este funcionario y la fecha la cual selecciono para agendarse (aquí lo recomendame es agendarse con un correo valido para poder ver la notificación de la agenda, hay que correr el programa denuevo para que esto se actualice). El programa termina la ejecución una vez le damos al botón para enviar.
- **Sin registro y con carnet:** Funciona casi igual pero en lugar de hacer un insert a la tabla Agenda, hace un insert a la tabla Carnet_Salud con los campos de la fecha completos con fechas validas y que por ejemplo al fecha de emisión sea antes a la de hoy y la de vencimiento futura a la de hoy y se debe añadir un comprobante, si no hay un comprobante adjuntado no te permite enviar el formulario.
- **Con registro y sin carnet:** En este caso le damos iniciar sesión y debemos obviamente tener un previo registro (en la base de datos hay varios inserts ilegales podríamos decir para modificar y poder ingresar usuarios con por ejemplo su carnet vencido y así poder revisar las notificaciones por correo de que no tenemos el carnet de salud vigente) luego le damos a no ya que no tenemos carnet de salud, y aquí vamos a ingresar en los campos los valores para actualizar (en la tabla funcionario se van a actualizar los valores dependiendo del logId con el cual nosotros iniciamos sesión, recomiendo usar alguno de los inserts de la base de datos para poder simular de esta forma que este usuario se registro hace 1 año por ejemplo) la parte del carnet funciona igual que en el punto de “Sin registro y sin carnet”.
- **Con registro y con carnet:** Bueno y esto básicamente la primera parte funciona como en el punto “Con registro y sin carnet” y la segunda parte como “Sin registro y con carnet”, simplemente actualiza todos los datos del usuario con el que inició sesión y se agrega un nuevo carnet a la tabla Carnet_Salud, no se eliminan ni se actualizan los carnet ya que creemos que la idea es mantener el historial de estos, pero si actualizamos a los funcionarios ya que no tendría sentido básicamente tener los datos que un funcionario ya no usa como un cambio de nombre o tal vez ingreso un dato mal.. etc..etc.

Una vez hacemos cualquiera de estas operaciones el programa se cierra y si queremos hacer otra tendremos que ejecutar el programa de nuevo.