15-5-2018

Clínica Privada Zaidín

Proyecto final EDD

Mario Pérez Domínguez, José Miguel Yáñez Martínez, Plácido Ramírez Calvero, Laura Ruiz Sánchez, Alberto Privado Moreno, David Serrano Alonso.

**Índice**

[**Enunciado** 2](#_Toc513659161)

[**Diagrama de Clases** 3](#_Toc513659162)

[**JForms** 4](#_Toc513659163)

[**Ficheros** 5](#_Toc513659164)

[**Medicamentos** 8](#_Toc513659165)

[**Pacientes** 9](#_Toc513659166)

[**MySQL** 13](#_Toc513659167)

[**Pruebas del Software** 16](#_Toc513659168)

[**Control de Versiones** 19](#_Toc513659169)

[**Unificación** 20](#_Toc513659170)

[**Conclusiones** 26](#_Toc513659171)

# **Enunciado**

Una clínica privada de nuestra ciudad nos ha encargado que desarrollemos una aplicación informática que gestione toda la información de los pacientes y la de propia clínica.

Dicho software, tiene asociada una base de datos de pacientes, para los cuales, hay una asociación con la seguridad social y en la que el NUSS de la esta, corresponderá con el identificador de cada paciente de dicha clínica.

La base de datos, tendrá las siguientes tablas:

* Consulta.
* Medicamento.
* Medico.
* Paciente.
* Receta

En la tabla consulta, se guarda toda la información referente a la consulta. En la tabla paciente, se guardará toda la información referente a los pacientes. En la tabla Medicamento, se guardará toda la información referente a los medicamentos. En la tabla Médico se guarda toda la información referente al médico. En la tabla Receta se guardan todas las recetas emitidas para el paciente con el NUSS asociado. En la tabla historial, se guardará toda la información clínica del paciente y será mostrado en un cuadro de texto.

Cuando inicie la aplicación solicitará introducir el NUSS el cual nos mostrará todos los datos del paciente. Si es la primera vez que asiste un paciente a la consulta, la aplicación debe permitir la inserción de un nuevo paciente.

El formulario, deberá constar con un botón de guardar, un botón volver y un botón para emitir una receta:

* Guardar: Guardará todas las modificaciones que se realicen en el formulario en la base de datos asociada a este proyecto.
* Volver: Retrocedemos hasta la selección de un paciente.
* Recetar: En una nueva ventana, permitirá elegir medicamento, periodicidad y la toma de ese medicamento.

La nueva ventana de recetas constará de los siguientes botones y los siguientes campos:

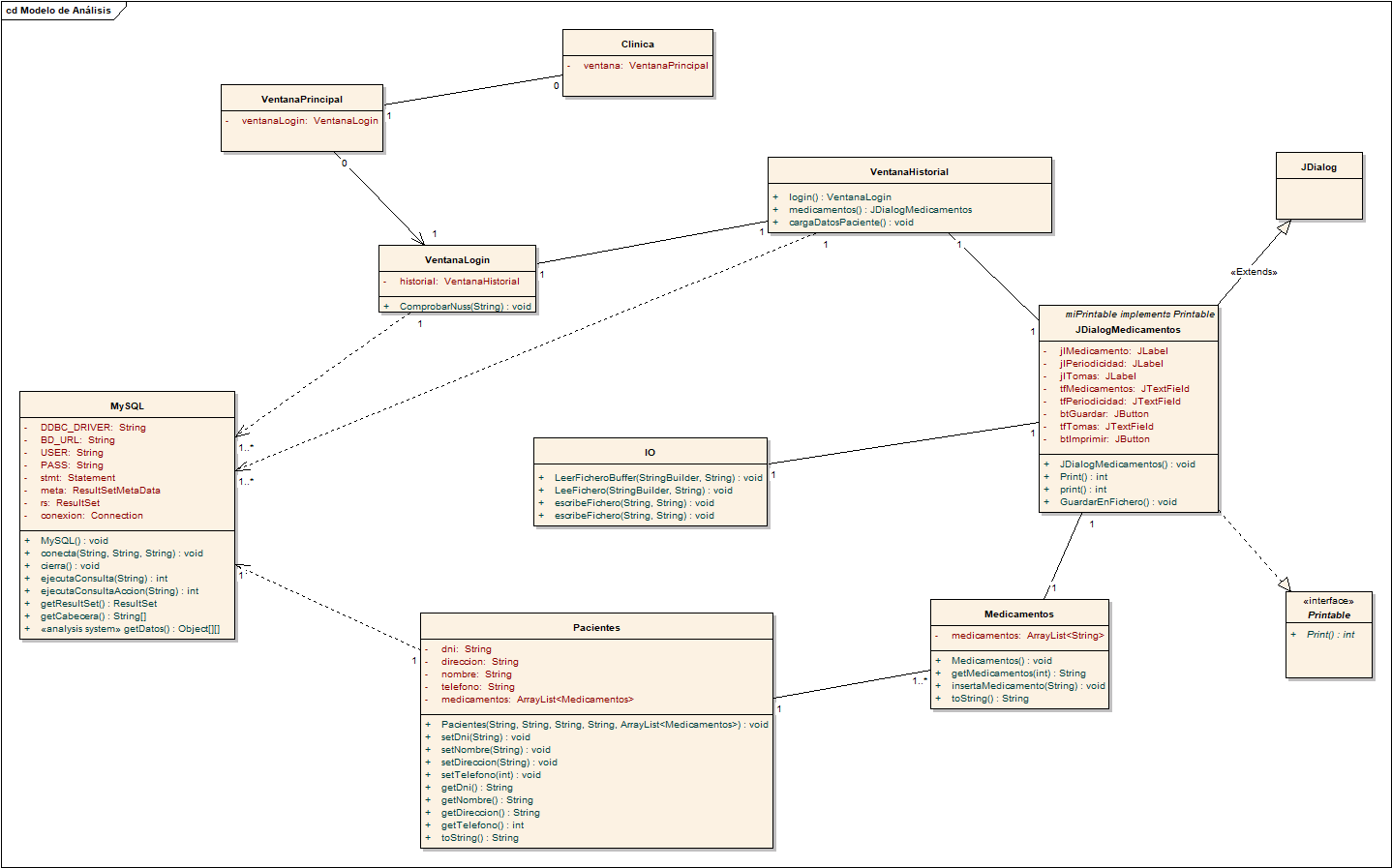
Campos:

* Medicamento:
* Periodicidad:
* Tomas:

Botones:

* Guardar, el cual guarda la receta en la base de datos.
* Imprimir, que nos permitirá imprimir la receta por la impresora predeterminada o en un PDF.

# **Diagrama de Clases**



# **JForms**

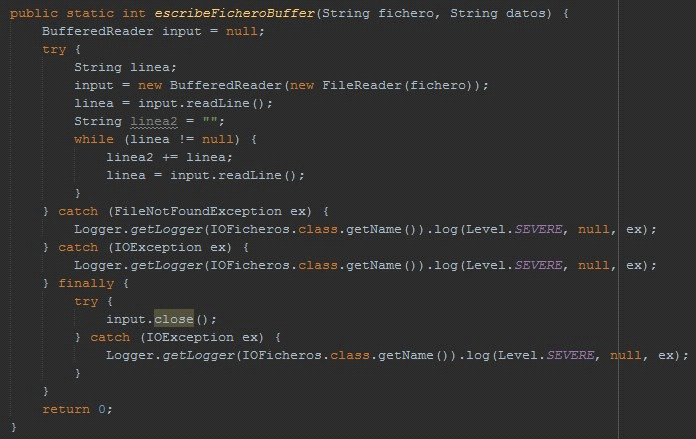
# **Ficheros**

La clase IOFicheros se encargará de leer y escribir en un fichero de texto los datos que introduce el usuario.

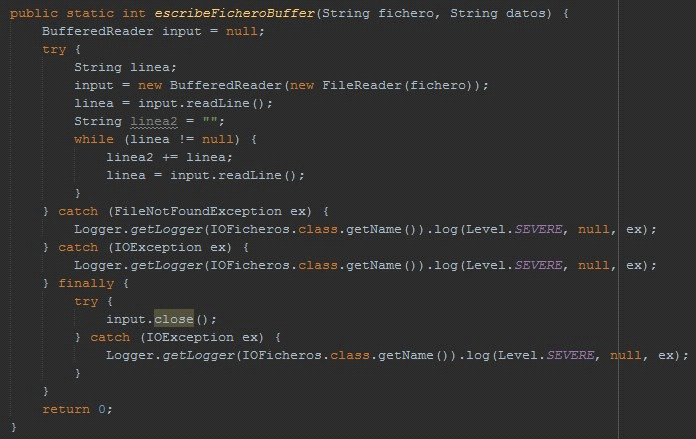
El primer método de la clase es el *"leeFicheroBuffer"*, el cual recibe el nombre del fichero de texto y lee todo el contenido de este, aunque al final solo nos muestra el contenido que le hemos pedido, y el resto lo almacena en memoria.

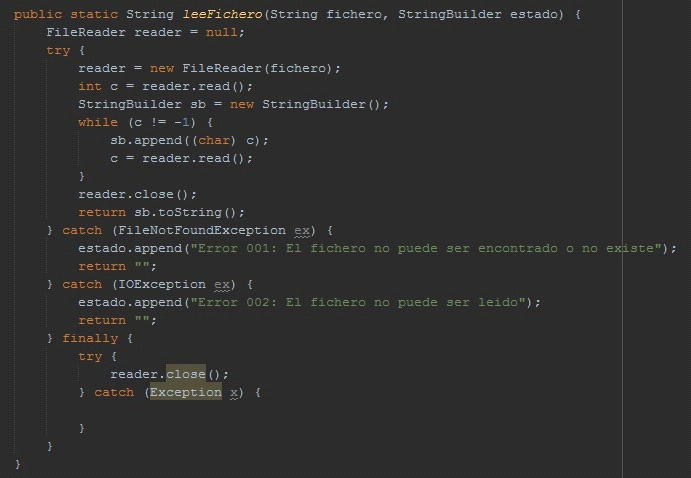


A su vez disponemos del método *"escribeFicheroBuffer",* la cual recibe unos datos por parte del usuario y los almacena en memoria hasta que tenga bastantes datos como para realizar una escritura eficiente



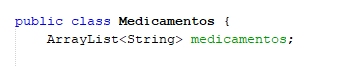
Por último tenemos los métodos *"leeFichero"* y *"escribeFichero",* las cuales realizan las mismas funciones que las clases anteriores, con la peculiaridad de que estas no almacenan datos en memoria, sino que leen y escriben todo el contenido del fichero de texto





# **Medicamentos**

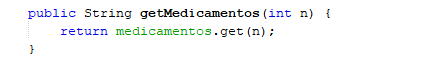
La clase medicamentos está compuesta por un constructor, tres métodos y un atributo de clase.



En el constructor es donde instanciamos el arraylist

C:\Users\Plácido\Desktop\Captura.PNG

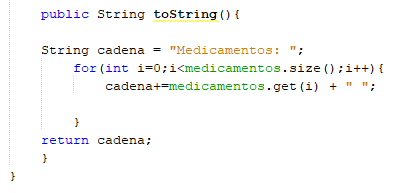
A continuación tenemos los tres métodos, el primero de ellos se encarga de coger los medicamentos del arraylist



El segundo método se encarga de añadir medicamentos al arraylist donde le pasamos el nombre de un medicamento



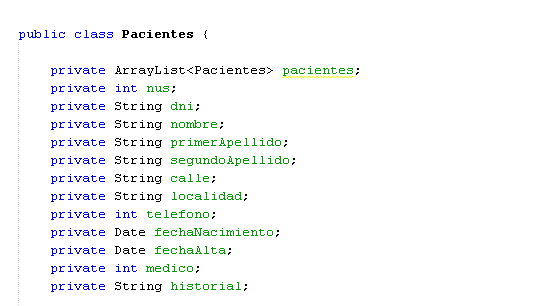
Y el tercer método se encarga de imprimir el arraylist de medicamentos



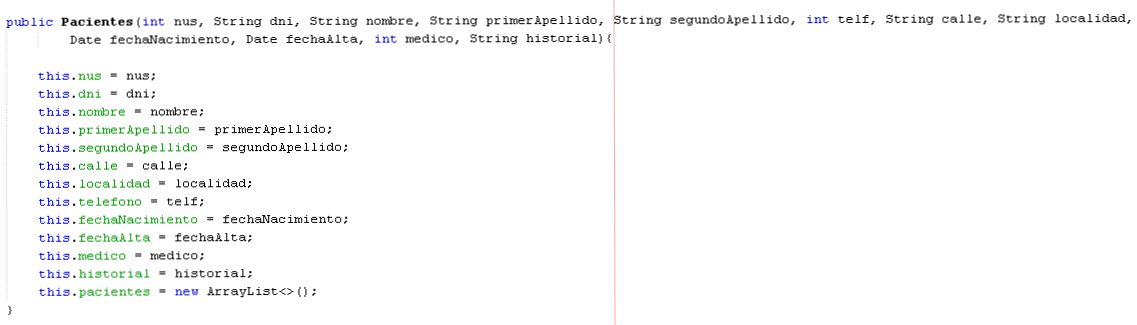
# **Pacientes**

La clase Pacientes va a ser la encargada de trabajar con los datos de los pacientes de la clínica y algunos datos relacionados con estos.

Lo primero que contiene esta clase, es una serie de declaraciones de atributos.

El primer atributo es un ArrayList de Pacientes con el nombre de pacientes. Después se declaran todos los datos del paciente. Estos son el nus, dni, el nombre, el primer apellido, el segundo apellido, la calle, la localidad, el teléfono, la fecha de nacimiento, la fecha de alta, el médico de este y el historial del paciente. Entre ellos está el nus, que va a ser número de identificación del paciente de la seguridad social, el cual se va a usar para acceder a los datos del paciente en el inicio de la aplicación.

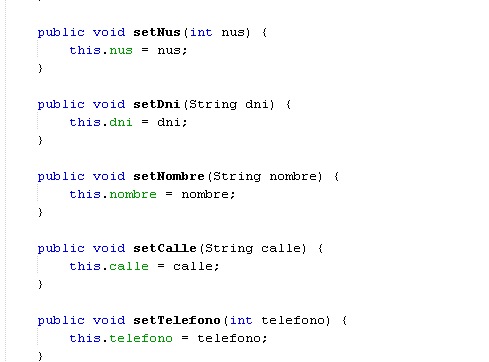
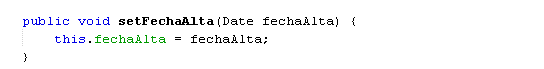
En la siguiente imagen tenemos el constructor de los atributos anteriormente explicados, en el cual, asignamos un valor a estos objetos recién declarados.

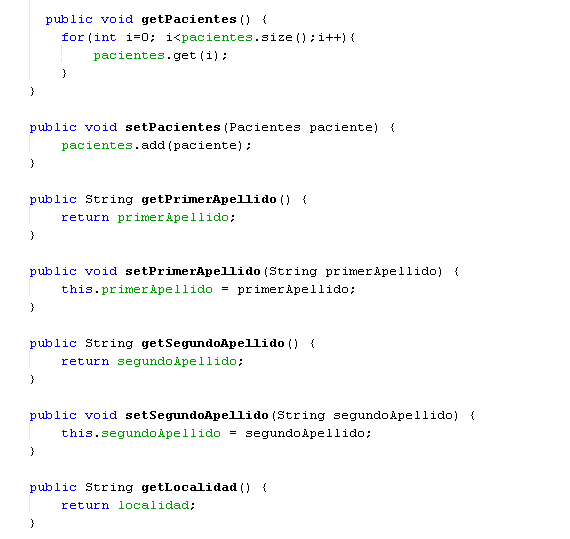
Una vez declarados y construidos estos atributos, utilizamos los Setters y Getters, los cuales son métodos de acceso lo que indica que son siempre declarados públicos.

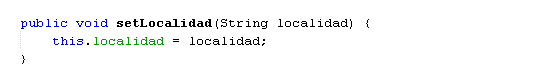
* Los setters nos sirven para asignar un valor inicial a un atributo, pero de forma explícita, además este no retorna nada, por eso siempre es void y solo nos permite dar acceso público a ciertos atributos que deseemos que el usuario pueda modificar.
* Los getters nos sirven para obtener el valor ya asignado a un atributo y utilizarlo para cierto método.

Los métodos que utilizamos a continuación son los que se encargan asignar un valor inicial y obtenerlo de los datos del paciente.

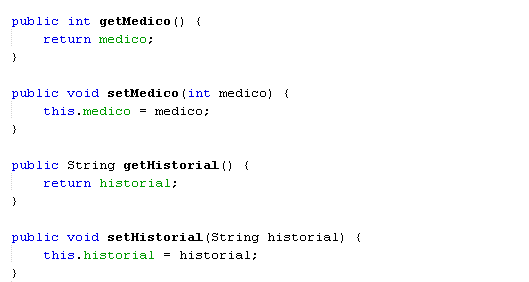
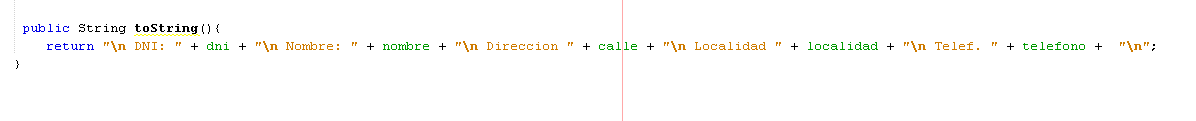








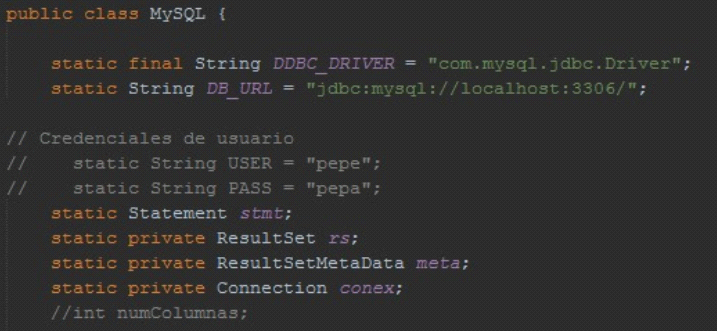
Por último tenemos un método llamado ToString, el cual devuelve una cadena que representa al conjunto de objetos con los cuales hemos trabajado anteriormente.



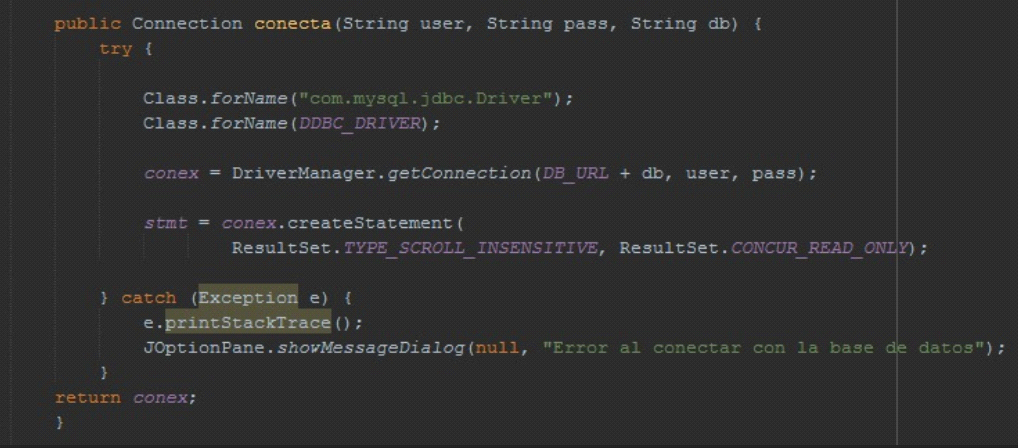
# **MySQL**

La clase MySQL realiza una conexión con una base de datos en el lenguaje MySQL.

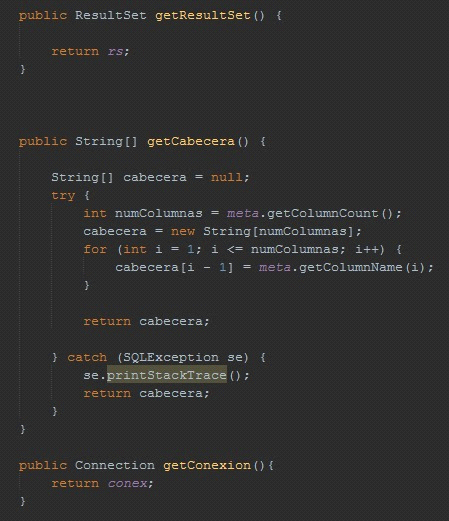
Lo primero que contiene esta clase son declaraciones de objetos necesarios para el funcionamiento de los métodos que contiene dicha clase, además del controlador necesario para el funcionamiento de ella y el puerto de conexión de la base de datos.

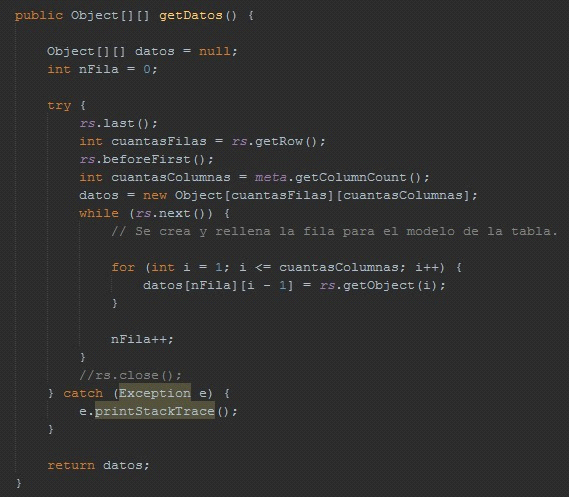


La clase *"conecta"* se encarga de realizar la conexión con la base de datos, e informa al usuario en caso de que haya un error de conexión.

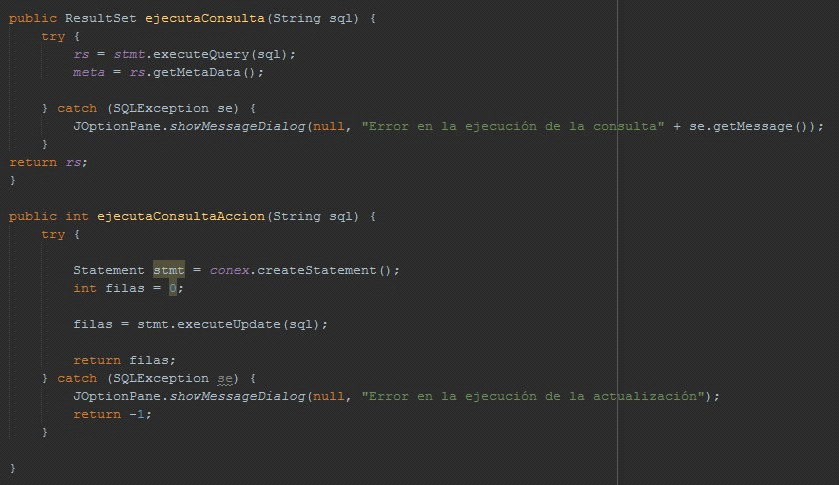


Las clase *"getResultSet"*, *"getCabecera"* y *"getDatos"* cogen los datos que contiene la base de datos y los nombres de las tablas.

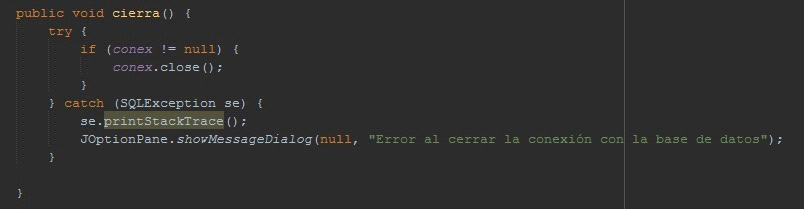




Las siguientes clases, *"ejecutaConsulta"* y *"ejecutaConsultaAccion"* permiten la ejecución de consultas de inserción, actualización y borrado en la base de datos.

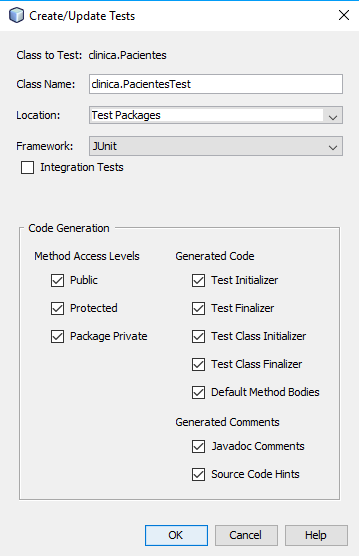


Por último, la clase *"cierra"* finaliza la conexión con la base de datos.



# **Pruebas del Software**

Para crear una clase de Test, hay que seleccionar una clase del proyecto pulsar botón derecho del ratón, navegar hasta el menú tolos y posteriormente “Create/Update tests” y nos aparecerá está ventana:



Seleccionaremos todas las casillas y al pulsar sobre ok, nos creará un paquete nuevo en nuestro proyecto



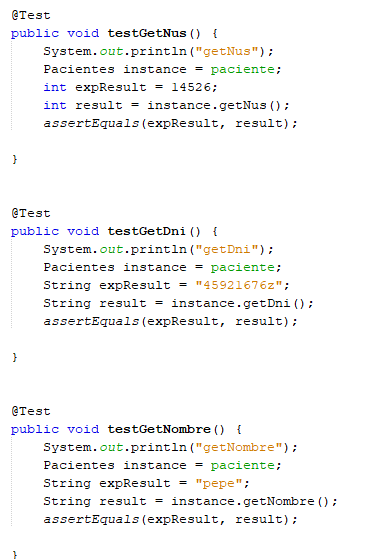
En el cual se encuentra nuestra clase con las pruebas con tantos métodos antecedidos de @Test como método tuviéramos en nuestra clase original.

Hay que configurar la clase para ejecutar los distintos métodos; En este caso primeramente vamos a crear varios objetos Pacientes.

También en todos los métodos que vayamos a probar hay que eliminar la línea que empieza con “Fail” porque si no provocará que nuestras pruebas fallen.



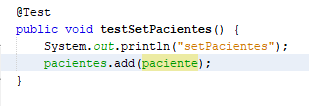
Cuando hayamos modificado todos los métodos de nuestra clase test, debería quedar algo similar:



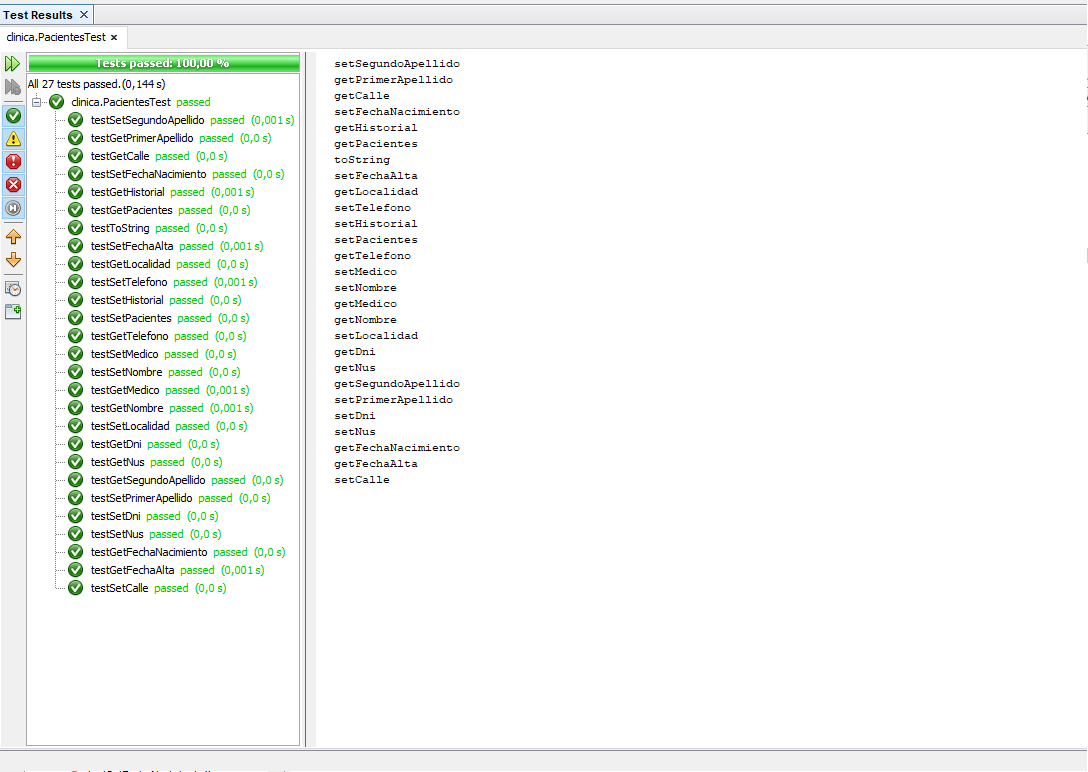
Ahora un objeto Paciente e instanciaremos el arrayList:



Con nuestro objeto Paciente creado iremos probando los distintos métodos, como por ejemplo el setPacientes() que lo que hará será insertar un paciente en el arrayList:



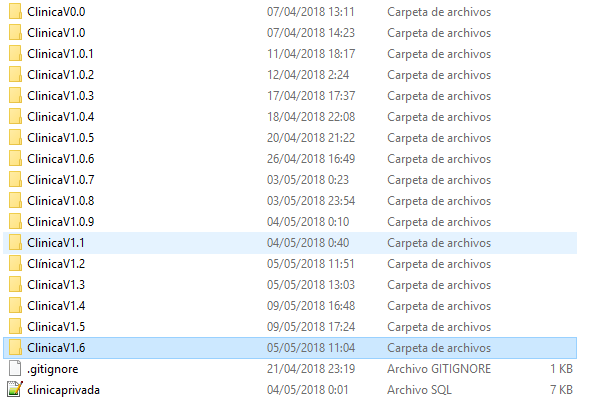
Seguidamente iremos probando y configurando los distintos métodos hasta que terminemos de evaluar satisfactoriamente nuestro test:



Este sería nuestro resultado final

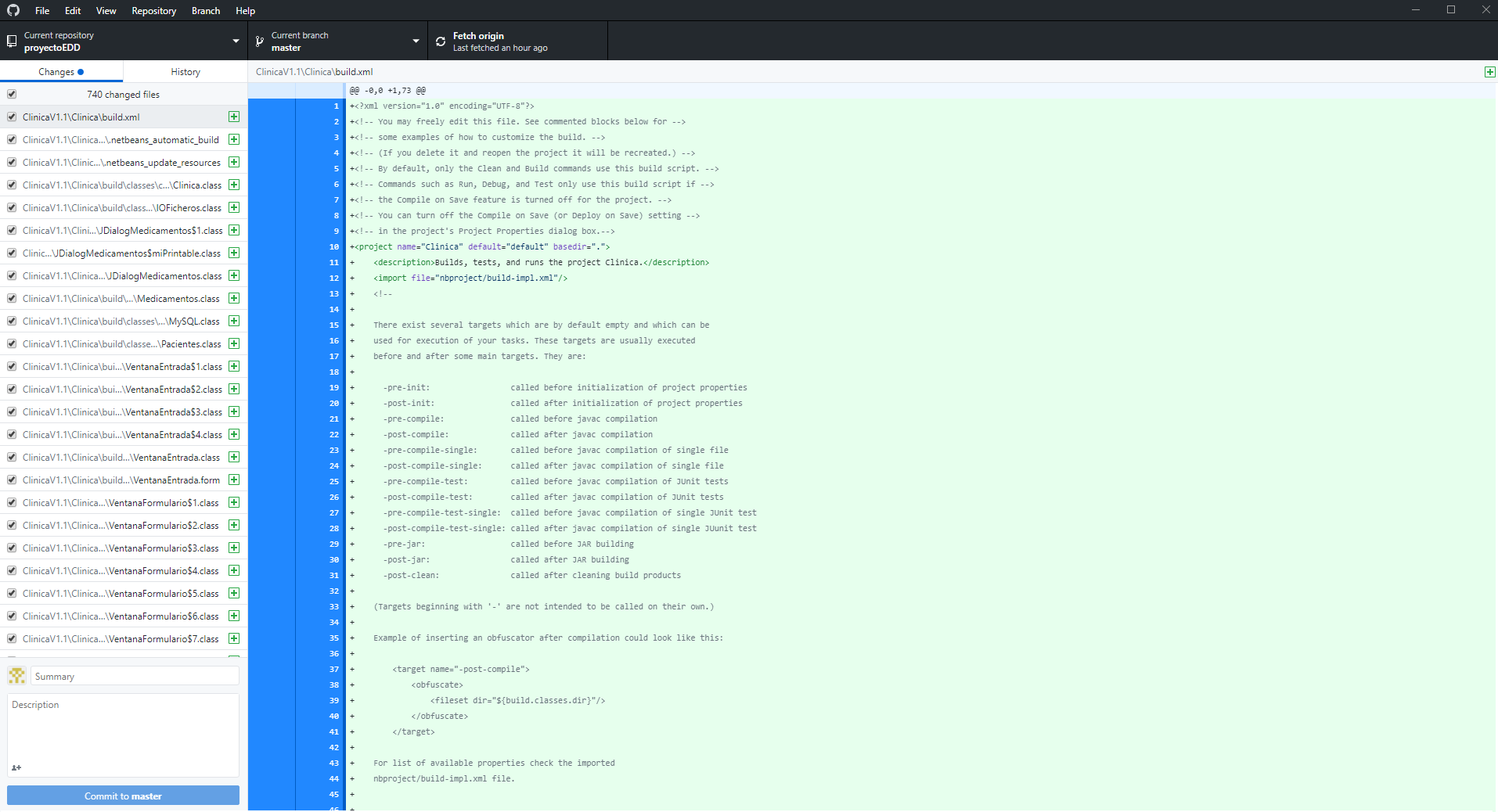
# **Control de Versiones**

Para el control de versiones, hemos usado gitHub, ya que era lo más cómo gracias a su interfaz gráfica. En total en el proyecto, hemos realizado 17 versiones diferentes desde la 0.0 a la 1.6



Para usar gitHub es necesario registrarse y descargar su aplicación. Una vez descargada e iniciada la sesión

Tendremos esta interfaz que mostrará si hay algún cambio:



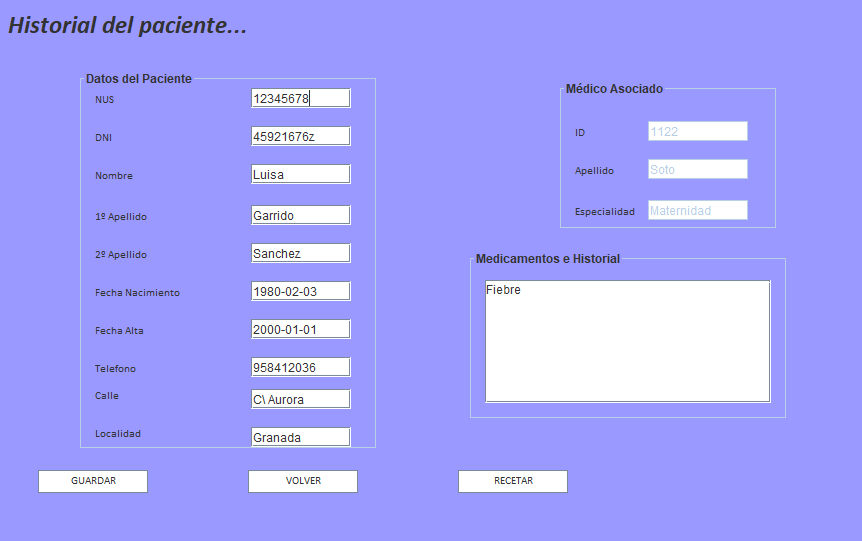
Abajo a la izquierda, introducimos un título y una descripción y se habilitará el botón “commit to master” pinchamos ahí y se habilitará arriba “Push origin” pulsamos en push y ya hemos compartido nuestros cambios.

# **Unificación**

El proyecto consta de 9 clases una principal que instancia VentanaPrincipal, la cual es un botón que instancia VentanaEntrada que muestra un JComboBox rellenándolo con un ResultSet y obteniendo los NUS de los pacientes. Una vez seleccionado el NUS podremos acceder a ver los datos de dicho paciente, instanciando VentanaFormulario y pasando como parámetro un ArrayList cargado con todos los datos de los pacientes a través de un resultSet, en VentanaEntrada:

|  |  |
| --- | --- |
|  |  |
|  |  |

Al haber pasado como parámetro el arrayList<Paciente> tenemos acceso a los distintos métodos que tiene la clase Paciente detallada anteriormente. Gracias a estos métodos, nos permite rellenar todo el formulario con los datos personales y clínicos de cada paciente:

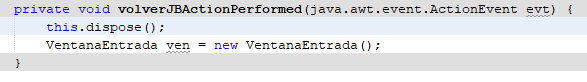


Desde este JFrame podremos modificar cualquier dato del paciente, excepto su médico ya que esto es una llave Externa de MySQL que apunta a la tabla médico y de la cual extraemos los datos con un ResultSet. Si estos datos se pudieran modificar, podría causar un conflicto en la llave externa que produciría un error fatal en nuestra aplicación.

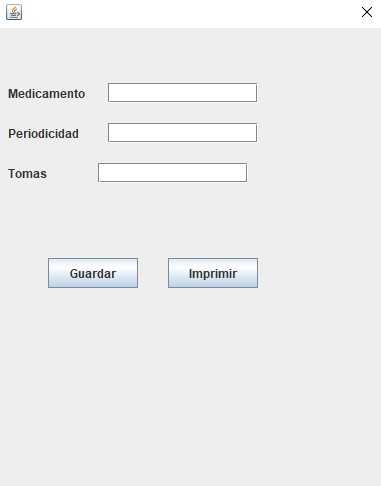
Todas las inserciones y modificaciones que se hacen en la BD, se llevan a cabo con PreparedStatement, esto nos permite más flexibilidad y seguridad a la hora de ejecutar nuestras consultas.



El botón volver, simplemente no devuelve a la ventanaEntrada



Por último el botón recetar, instancia un JDialog en el cual podremos recetar un medicamento a nuestro paciente e imprimir dicha receta:



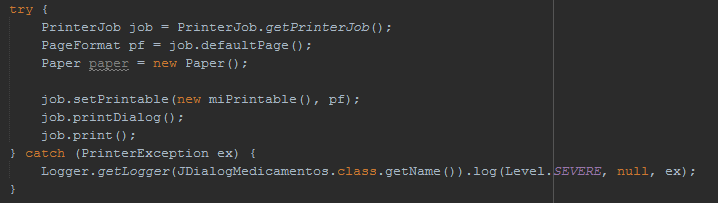
El botón de guardar, nos genera un archivo de texto con el contenido de los textField usando para ello la clase IOFicheros explicada anteriormente.

También el botón de guardar, hacer una inserción en la base de datos en la tabla recetas y dicha receta estará asociada a un médico y a un paciente.



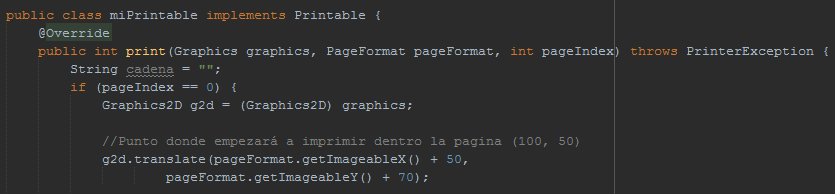
En el botón imprimir creamos un objeto PageFormat y un PrinterJob que es el que será enviado a la impresora seleccionada.

Establecemos que sea imprimible, le pasamos como parámetro nuestra clase interna en la cual redefinimos el método print y el PageFormat y seleccionamos el dialogo de impresión. Por último imprimimos.

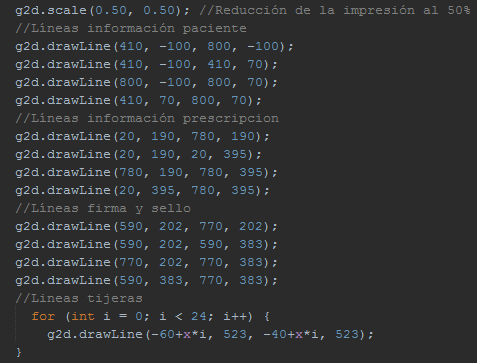


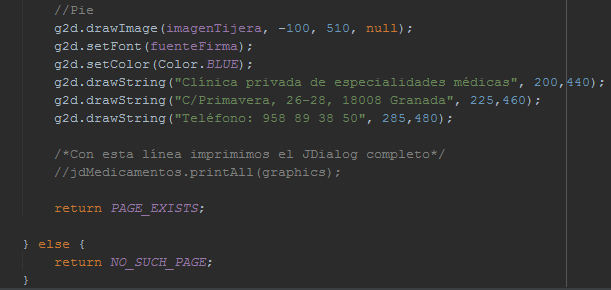
En la clase interna donde redefinimos el método printable lo primero que hacemos es un If si la página es igual a 0 para saber si es la primera. Si lo es, creamos un objeto Graphics2D y le asignamos el valor del parámetro Graphics que se nos pasa haciendo un casting.

Tras ello, usamos el método Translate() de Graphics2D para traducir (mover) cada punto desde Graphics2D al punto correspondiente del formato de página (PageFormat) pasándole los valores del eje X y el eje Y.



Una vez hecho esto, cambiamos la escala de la página un 50% con el método scale() y comenzamos a escribir en la página con los métodos drawLine() para hacer líneas, drawString() para escribir Strings en la página y drawImage() para colocar una imagen.





Terminamos de redefinir el método haciendo que devuelva PAGE\_EXISTS en caso de que se escriba la página o NO\_SUCH\_PAGE en caso de que no se haga nada.

Una vez compilado el software, para ejecutarlo necesitamos Java Enviroment y nuestro archivo Jar por defecto Netbeans lo crea en la carpeta dist dentro del proyecto

# **Conclusiones**

Plácido: El trabajo me ha ayudado bastante a coger nuevas perspectivas a la hora de trabajar en un proyecto, también me ha ayudado a corregir algunos errores muy básicos que tenía y me ha enseñado que la realización de un proyecto lleva su tiempo y trabajo

David: Este proyecto me ha servido para afianzar conocimientos en general de toda la asignatura así como la asignatura de programación. Como estructurar un proyecto con varios desarrolladores y como organizar el trabajo entre todos.

Mario: En mi opinión hacer este proyecto en grupo nosotros desde 0 sin ayuda ha sido una idea bastante acertada ya que nos ha ayudado mucho a pensar y ver cómo crear y estructurar nosotros mismos un proyecto totalmente nuevo, y en lo personal, sobre todo en mi parte me ha servido mucho para aprender más sobre el printable.