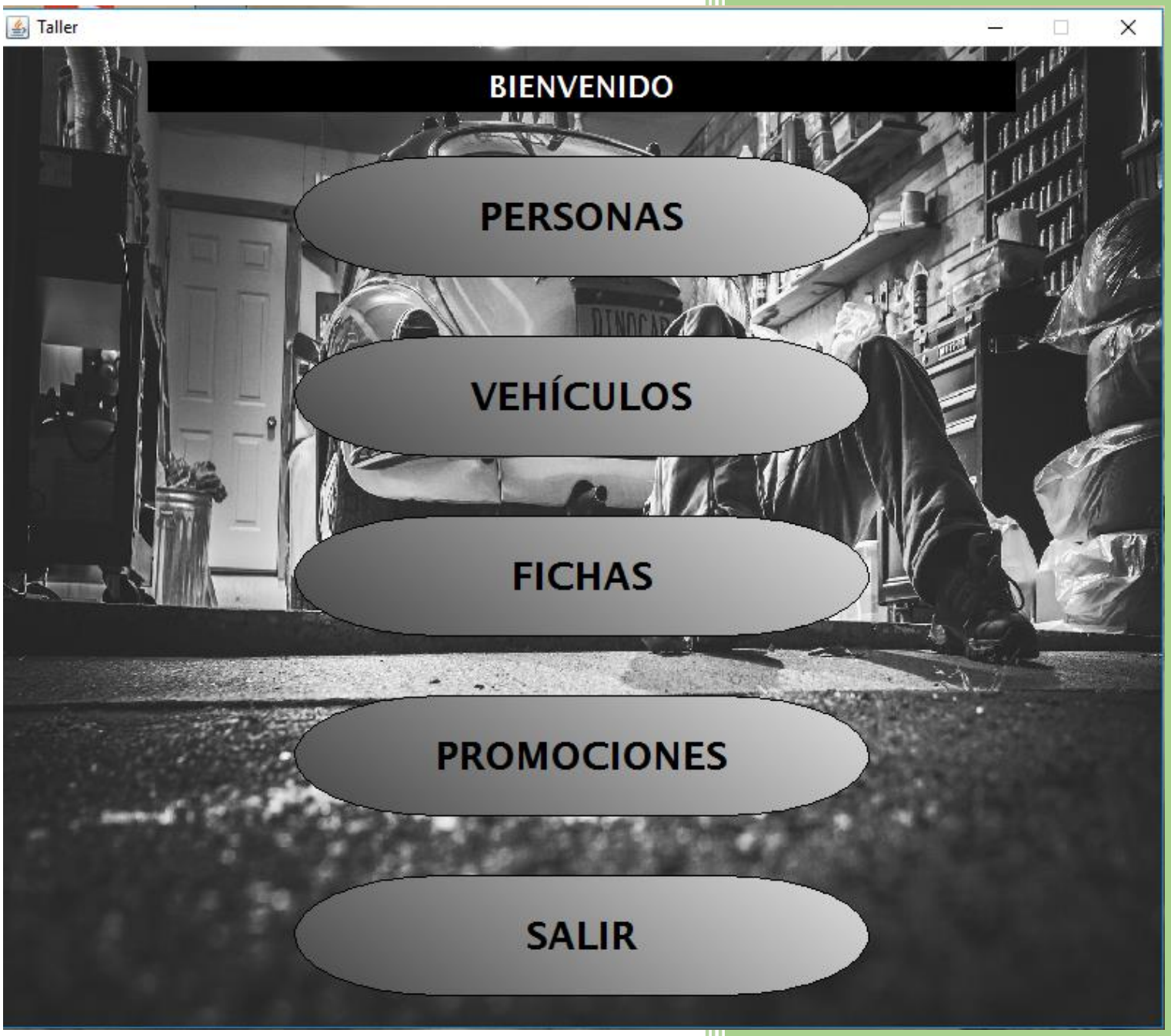
PRÁCTICA POO: Gestión de taller 2016/2017



Adriana Armental Tomé

53489431-X

Correo: adri.13ª@gmail.com

Telf: 606605912

Índice

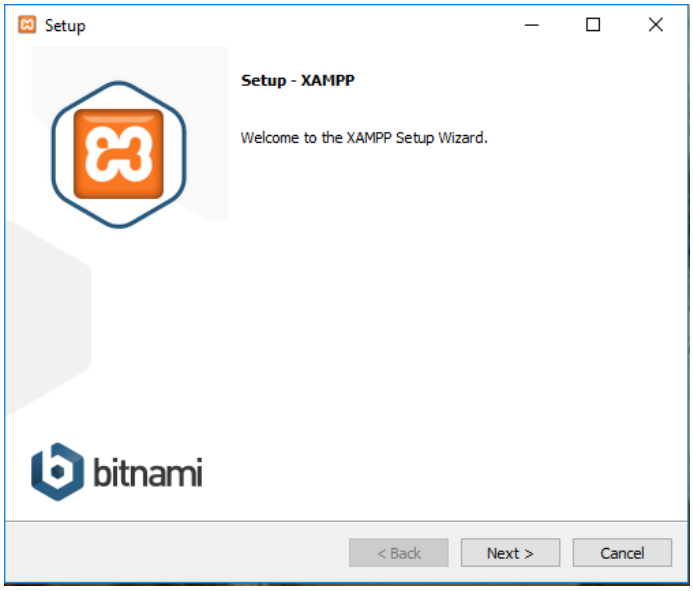
1. Introducción

El desarrollo de esta práctica da la posibilidad al alumno de escoger la manera de trabajar con los datos. Mi forma de hacerla es mediante bases de datos lo que implica tener un servidor apache y un gestor de bases de datos. El programa XAMPP nos proporciona ambas cosas por lo que voy a hacer un breve tutorial de instalación.

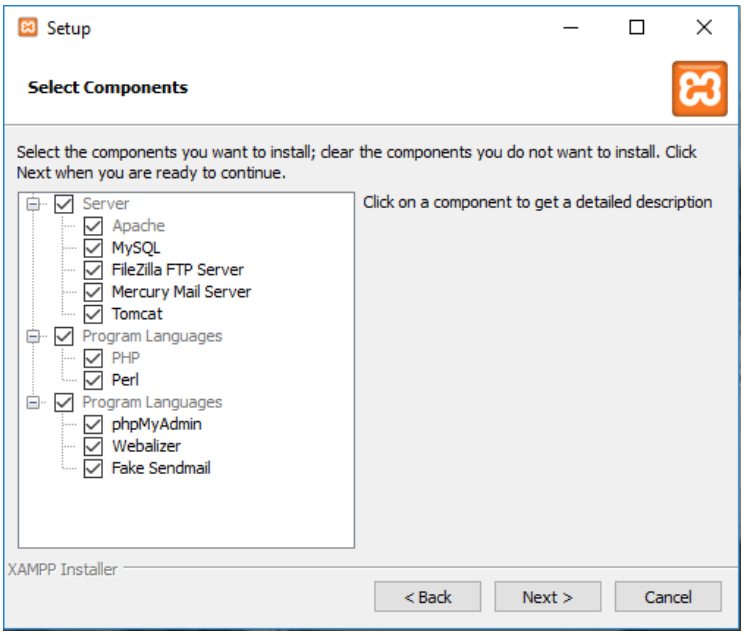
2. Descarga e instalación de XAMPP (en Windows)

Para empezar, iremos al enlace https://www.apachefriends.org/es/download.html y escogeremos la opción que más se adapte a nuestro equipo (mejor PHP 5 ya que PHP 7 aun no es estable).

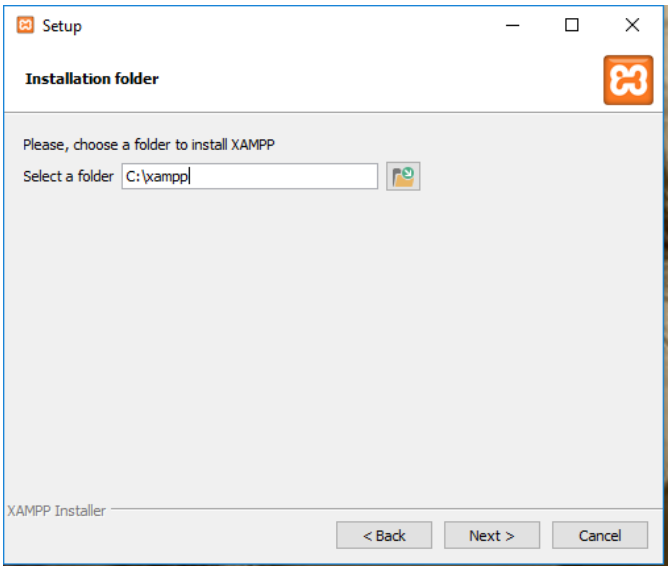
Una vez descargado, procedemos a su instalación. Nos abrirá un asistente.



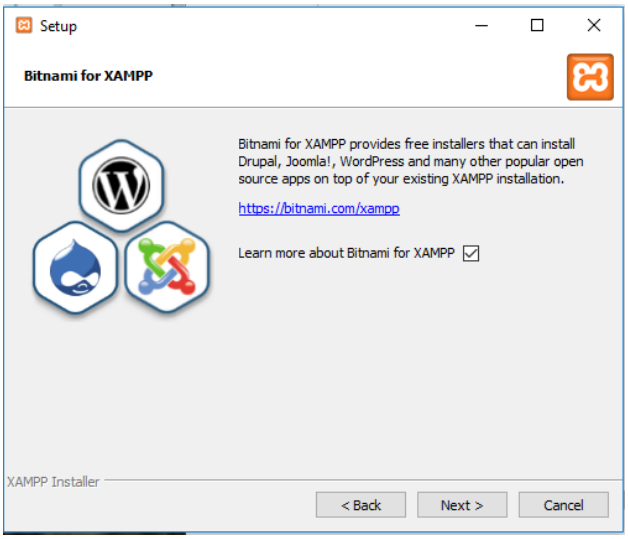
No tiene mayor complicación, asique le damos a “Next”.



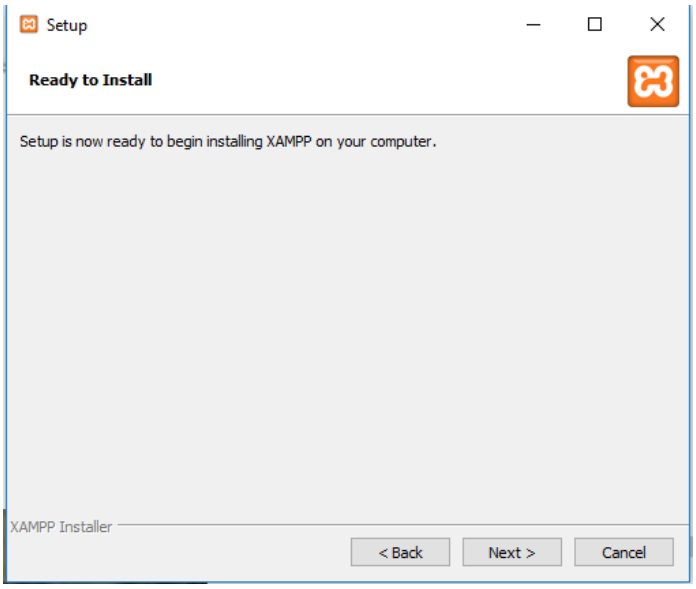
Seleccionamos los componentes por defecto, aunque para esta práctica solo son necesarios el Apache, el MySQL y phpMyAdmin.



Elegimos el directorio donde queremos almacenarlo. En mi caso en una carpeta creada en C llamada XAMPP.

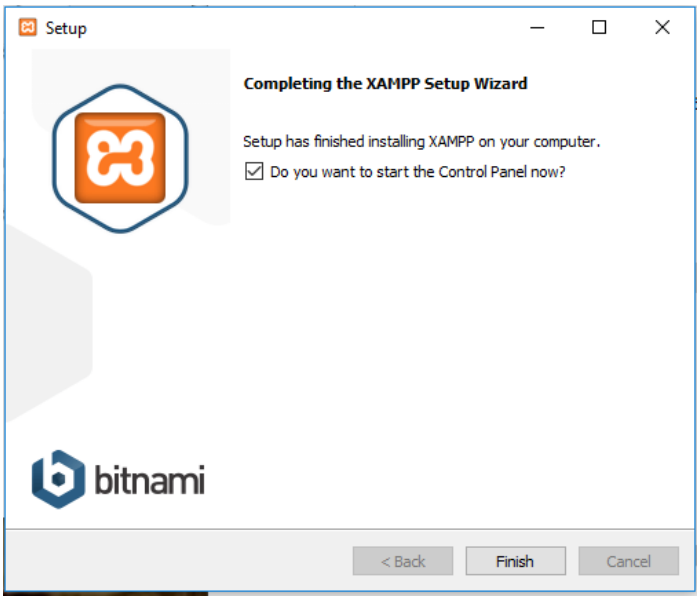


Seguimos dándole a “Next”, podemos marcar o desmarcar la casilla para saber más sobre XAMPP, la cual nos abrirá una ventana en el navegador por defecto con dicha información.



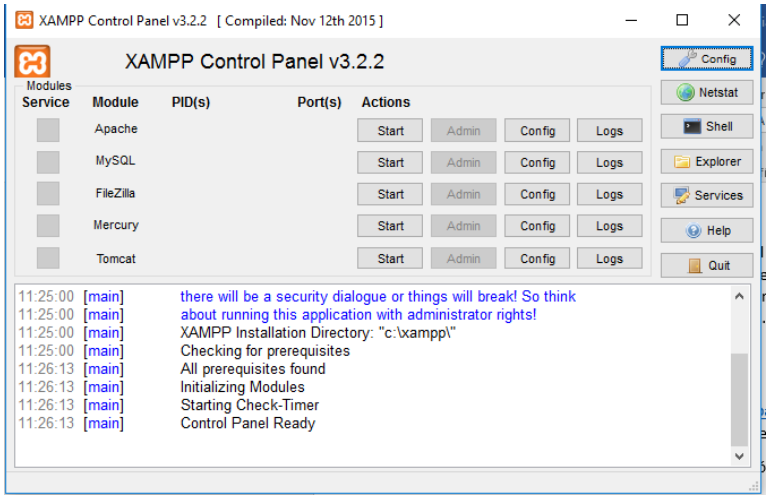
Al pulsar en “Next” dará comienzo la instalación.

Esperamos a que termine la instalación.

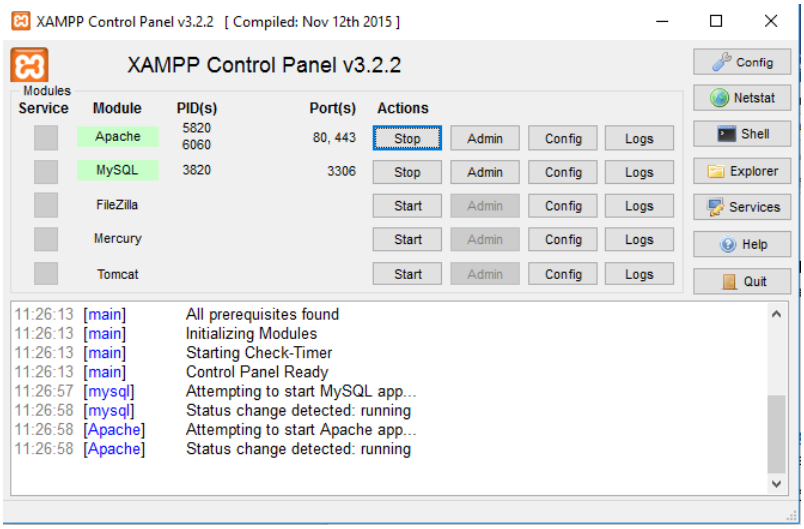


Tras finalizar la instalación nos permite abrir el panel de control directamente. Lo marcamos e igual tarda un poco hasta que abra el panel de administración.

La primera vez que lo abrimos nos dará la opción de elegir el idioma.



Finalmente se abre el panel de control y vemos las diferentes opciones que podemos habilitar, el servidor apache, MySQL, el FileZilla, etc.

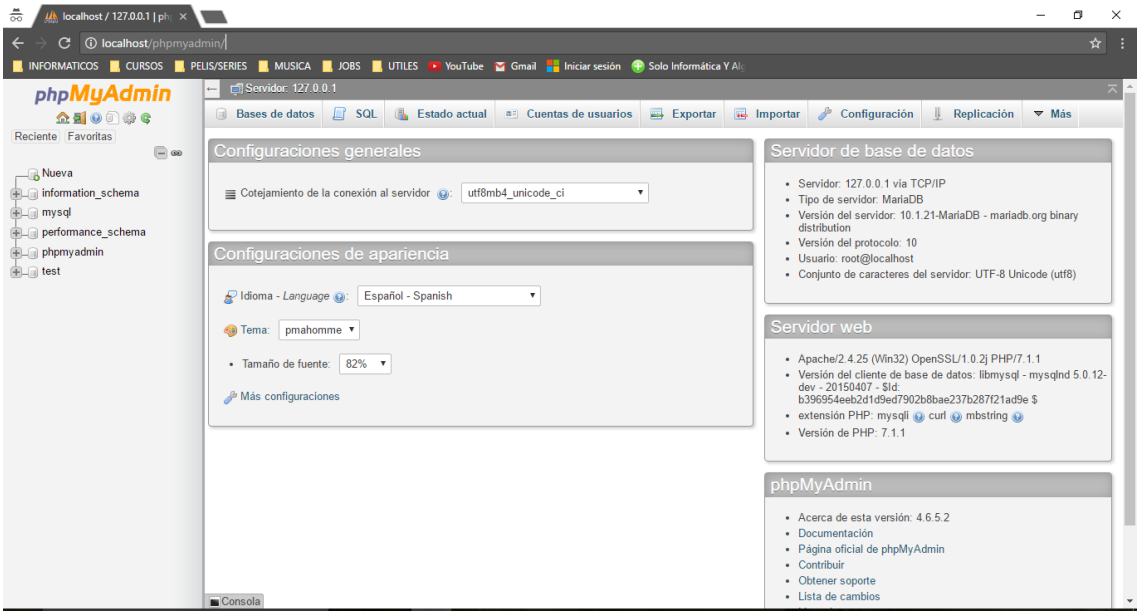


Para esta práctica solo es necesarios activar el Apache y el MySQL.

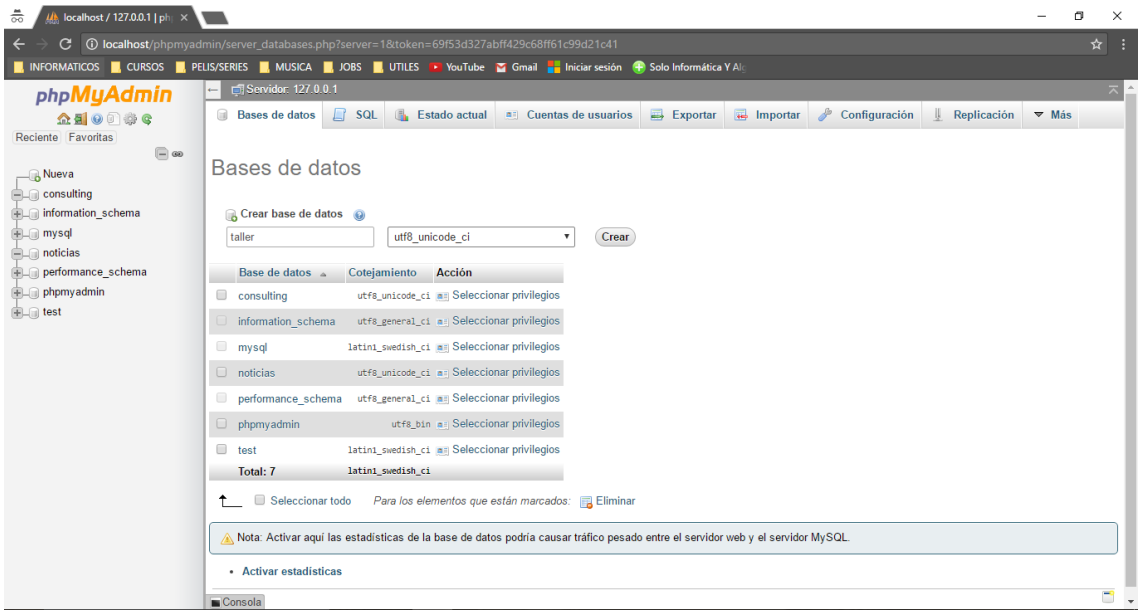
De esta forma se abre el panel de control directamente tras la finalización de la instalación del XAMPP, pero si ya lo tenemos instalado y lo queremos abrir, no hace falta más que ir a todos los programas y buscarlo ahí.

2.1. phpMyAdmin e importación de la Base de Datos

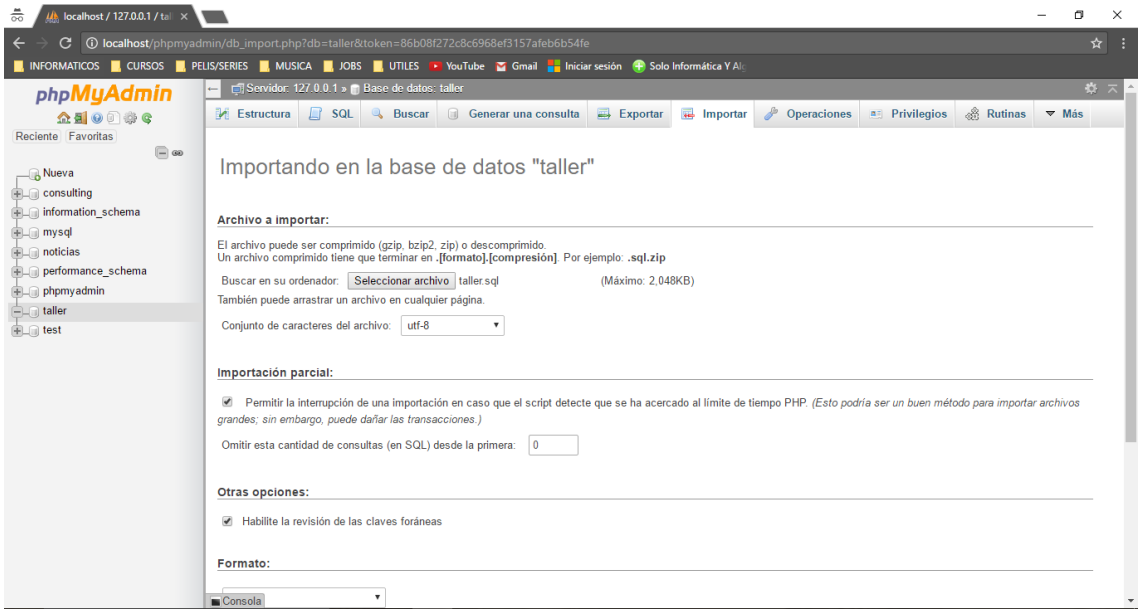
Una vez que tengamos XAMPP instalado y el servidor Apache y MySQL instalados accedemos a localhost: http://localhost/phpmyadmin/



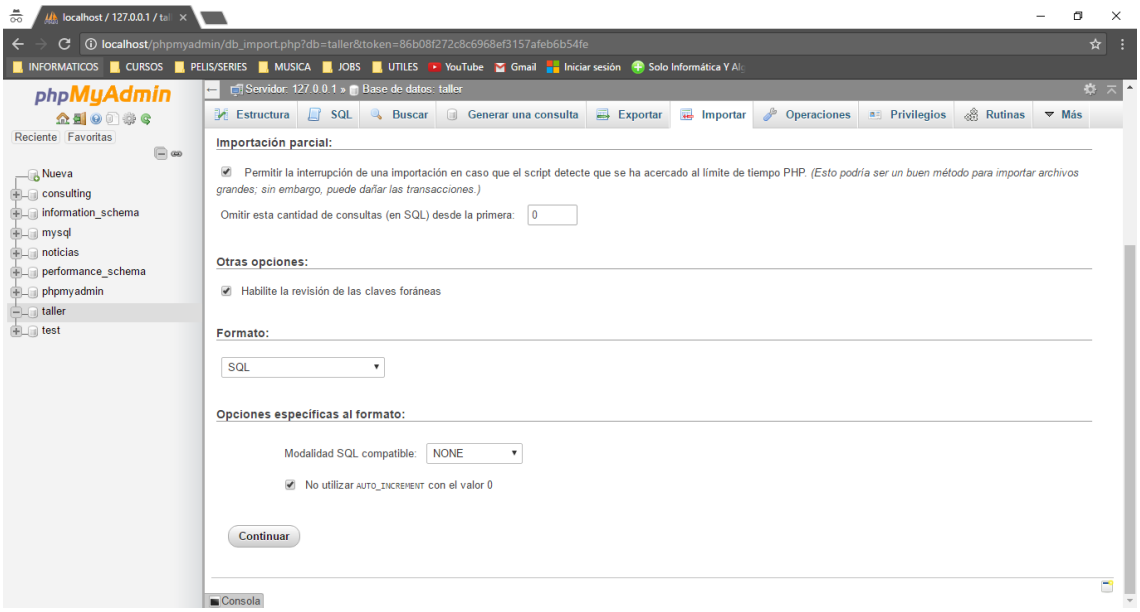
Ahora procedemos a crear la Base de Datos “taller”.



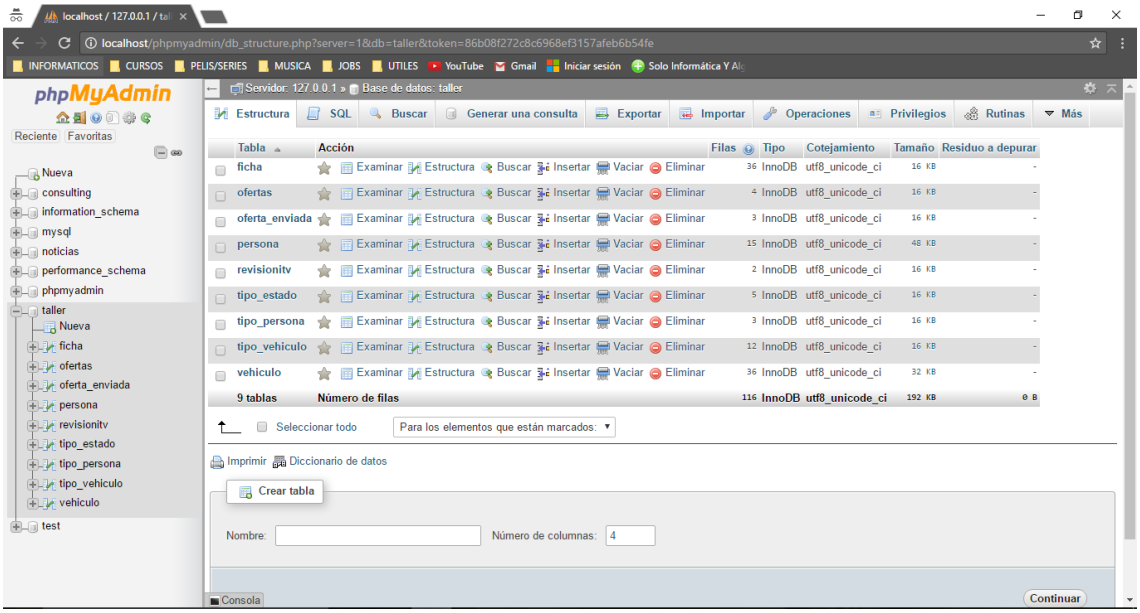
Le ponemos el nombre y elegimos el cotejamiento utf8_unicode_ci por tema de tildes y demás errores de sintaxis ortográfico.



Una vez creada la Base de Datos la seleccionamos en el menú lateral izquierdo y en el menú superior nos aparecerán una serie de opciones entre las que se encuentra “Importar”. Abrimos esa pestaña y nos da la opción de seleccionar el archivo SQL que estará almacenado en el directorio bd/taller.sql.



Una vez seleccionado no hay que darle más opciones a la importación asique le damos a "Continuar" y empezará la importación.



Una vez finalizada la importación vemos las tablas que se han importado a esa Base de Datos y si abrimos una tabla veremos los registros que están en esa tabla concreta.

NOTA: Para poder acceder al phpMyAdmin o utilizar la aplicación (taller.jar) siempre tendremos que tener iniciados el Apache y el MySQL. En el caso de la aplicación también será necesario tener la Base de Datos "taller" importada.

3. Descripción de la práctica

La práctica a realizar consiste en la realización de una aplicación para la gestión de un taller, con temas a tratar como personas (clientes, mecánicos, comerciales) vehículos (todo tipo), fichas de reparación en las que se incluyen los mencionados anteriormente y ofertas que los comerciales mandarán anualmente a los clientes.

Es realizada mediante la Programación Orientada a Objetos y el uso de Bases de Datos, así como de otras librerías como el driver JDBC de MySQL y la librería iText para la creación de pdfs.

La funcionalidad es bastante simple. Cualquier miembro del taller que pueda acceder a la aplicación tiene acceso a todas las funcionalidades que posee. Independientemente de si es mecánico o comercial (no dispone de sesiones para indicar quién es quién).

La primera funcionalidad que aporta es referente a las personas, tanto clientes como mecánicos o comerciales. Existe la posibilidad de registrar uno nuevo, modificar los datos de uno ya registrado o eliminarlo. A mayores existen varias consultas que se pueden hacer para localizarlos con mayor facilidad como pueden ser búsquedas por DNI, por nombre o por los apellidos. Todas estas consultas se realizan sobre la Base de Datos mediante el driver JDBC.

La segunda funcionalidad es la referente a los vehículos (todo ellos, incluidos en diversas categorías como coches que incluye turismos, deportivos, monovolúmenes o todoterrenos, motos, que pueden ser motos de campo o carretera, vehículos profesionales como ambulancias, coches de bomberos y policías y luego otros sin categorizar como autobuses, camiones o furgonetas). Es muy similar a la funcionalidad de personas, puedes registrar cualquier tipo de vehículo con sus características correspondientes. Poder modificarlos o eliminarlos. Para consultar se pueden ver todos o hacer búsquedas por matrícula, marca o modelo.

Otra funcionalidad que tiene son el registro, modificación, eliminación y búsqueda de fichas. Las fichas contienen la información sobre el vehículo, su dueño, el motivo de su ingreso, la fecha en la que entró, la fecha en la que sale (cuando se haya “arreglado”) el estado en el que se encuentra (pendiente, en proceso, parado, fase prueba o terminado si ya se ha acabado con el). En el caso de estar parado se habilita el motivo por el que se ha parado esa “reparación”. Las búsquedas se pueden hacer por el cliente, el mecánico o el estado, a mayores se puede incluir un rango de fechas para la búsqueda.

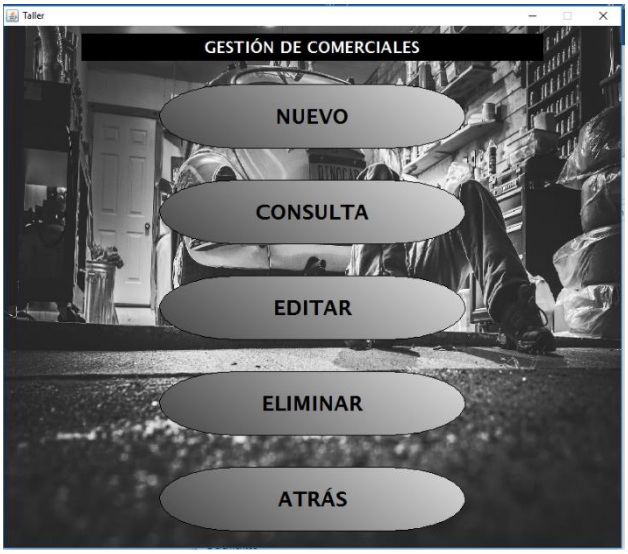
La última funcionalidad son las ofertas y la revisión de ITV en las cuales un comercial puede crear ofertas y después enviárselas a los clientes guardando la fecha en la que se envían puesto que hasta que transcurra un año esas ofertas no podrán ser enviadas a ese cliente. La ITV es lo más simple consiste en mantener un registro de los vehículos que se apuntan para acceder a los servicios que se ofrecen con esta promoción.

Si hay algún error en la edición o registro de un nuevo elemento tales como matriculas, correos o DNIs duplicados, introducir texto en campos de solo números nos llevará a una ventana de error de registro.

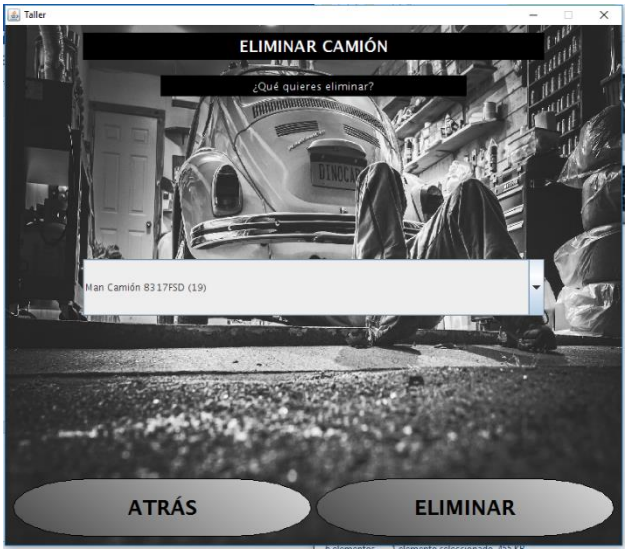
4. Diseño de la aplicación

El diseño de la aplicación consiste en una ventana en la que se manejan todas las funcionalidades de la aplicación sin diferenciar que tipo de usuario es el que la esté usando.

Interactuando con los distintos botones que aparecen en el menú de cada ventana irán apareciendo las diferentes funciones en función de la categoría en la que nos encontremos.

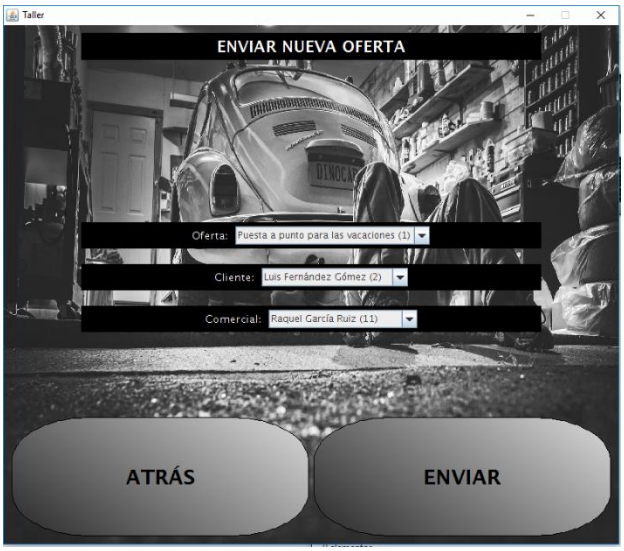



Taller
EDITAR CAMIÓN
Matrícula: 8317FSD
Marca: Man
Modelo: Camión
Nº Puertas: 2
Nº Ruedas: 6
Velocidad máx.: 200
Cap. Motor: 200
Cap. Almacenamiento: 300
Nº Pasajeros: 2
Motivo Visita: Reparación
Combustible: diésel
ABS: si
GPS: si
Airbags: si
Climatización: si
Descapotable: no
Dueño: Adriana Armental Tomé (1)
ENVIAR
ATRÁS


Taller
ELIMINAR CAMIÓN
¿Qué quieres eliminar?
Man Camión 8317FSD (19)
ATRÁS
ELIMINAR


Taller
BUSCADOR POR ESTADO DE FICHAS
¿Que estado quieres buscar?
Estado: pendiente
Rango Fecha Inicial:
Rango Fecha Final: 2017-05-02
ATRÁS
ENVIAR


Taller
LISTADO DE FICHAS
DESCRIPCIÓN ESTADO FECHA ENTRADA FECHA SALIDA MOTIVO PARADO CLIENTE
Revisión pendiente 2017-04-27 Luis Fernández Gómez Nissan GTR 2598
Revisión pendiente 2017-04-27 Martin Rodríguez Torres Suzuki Jimny 2589
Revisión pendiente 2017-04-27 Marcos González Castro SsangYong Tivoli
Reparación pendiente 2017-04-27 Adriana Armental Tomé Jeep Wrangel 987
Reparación pendiente 2017-04-27 Ana Calo Vila Citroën Grand C4
Reparación pendiente 2017-04-27 Martin Rodríguez Torres Opel Zafira 5871
Revisión pendiente 2017-04-27 Luis Fernández Gómez Renault Space 984
Reparación pendiente 2017-04-27 Ana Calo Vila KTM 250 EXC 25
Revisión pendiente 2017-04-27 Luis Fernández Gómez Yamaha WR 450
Reparación pendiente 2017-04-27 Marcos González Castro Honda CRF250 81
Revisión pendiente 2017-04-27 Adriana Armental Tomé BMW R 1200 Rt
ATRÁS


Taller
ENVIAR NUEVA OFERTA
Oferta: Puesta a punto para las vacaciones (1)
Cliente: Luis Fernández Gómez (2)
Comercial: Raquel García Ruiz (11)
ATRÁS
ENVIAR


Taller
LISTADO DE OFERTAS ENVIADAS
OFERTA CLIENTE CORREO CLIENTE COMERCIAL CORREO COMERCIAL
Revisión de neumáticos Luis Fernández Gómez luisfe@hotmail.com Vanesa Nuñez Abad vanesa@hotmail.com
Puesta a punto para las vacaciones Adriana Armental Tomé adri.13a@gmail.com Raquel García Ruiz raquel@hotmail.com
Puesta a punto para las vacaciones Adriana Armental Tomé adri.13a@gmail.com Carlos Rivas Dobarro carlos@gmail.com
ATRÁS

5. Herramientas usadas en el desarrollo de la aplicación

Las herramientas que se han usado en el desarrollo de la aplicación de la práctica son:

- ECLIPSE IDE JAVA EE DEVELOPERS NEON
  Herramienta para desarrolladores de Java que crean en entornos Java EE y aplicaciones web, incluyendo un Java IDE.
- JAVA SE 8
  Para la creación, compilación y ejecución de aplicaciones en el lenguaje de programación Java, el requisito es la instalación de un entorno de programación J2SE.
- MICROSOFT OFFICE WORD 2016
  Necesario para la creación de esta memoria. Se encarga del procesamiento de textos.
- GLIFFY
  Aplicación web para la creación de lo diagramas.
- DROPBOX
  Aplicación para el control de versiones que usé durante el transcurso de la realización de la aplicación.

Dentro de la propia aplicación también se usaron:

- JFRAME
  Componente básico que se requiere para implementar una interfaz visual con la librería Swing. Encapsula una ventana clásica de cualquiera sistema operativo con entorno gráfico (Windows, Linux, etc.). Tiene una serie de propiedades que se pueden modificar al gusto de cada uno.
- Componentes del JFRAME: JBUTTON, JLABEL, JPANEL, JCOMBOBOX, JTEXTFIELD
  Aunque el JButton no se usa directamente, se crea una clase que hereda de él y se modifican sus propiedades.
  Son componentes de la librería Swing, de las cuales, el JLabel muestra un texto, el JButton un botón a partir del cual se interactúa mediante eventos, el JComboBox es una lista desplegable y el JTextField es un cuadro de texto. El JPanel actúa como contenedor dentro del cual se pueden agrupar los anteriores mencionados para darle una distribución distinta al JFrame.
- ACTIONLISTENER
  Es la interfaz de escucha para recibir eventos de acción (ej. Pulsar un botón). Se registra un objeto con el componente addActionListener y cuando se produce el evento de acción se invoca al método actionPerformed que llevará a cabo una acción indicada.
- ITEMLISTENER
  Es la interfaz de escucha para recibir eventos de cambio (ej. Cambiar el objeto seleccionado de un seleccionador). Se registra un objeto con el componente addItemListener y cuando se produce el evento de cambio se invoca al método itemStateChanged que llevará a cabo una acción indicada.
- KEYLISTENER
  Similar al anterior para recibir eventos de teclado (ej. Pulsar una tecla). Se registra un objeto con el componente addKeyListener y cuando se pulsa una tecla se invoca al método KeyEvent que se subdivide en KeyPressed (tecla pulsada), KeyReleased (tecla liberada) y KeyType (tecla escrita).

6. Funcionalidades de la aplicación

PARTE JFRAMES

El juego está realizado en JFrames, a los cuales se les asigna un tamaño de 800px de ancho y 700px de alto. Los diferentes JFrames se irán eliminando y haciendo visibles según la ventana que se desee abrir. Fueron creados de forma que no se les pueda cambiar el tamaño y cada vez que se abra una ventana, esta se colocará en el centro de la pantalla.

El menú principal muestra 5 botones de los cuales, el último cierra la aplicación. Los otros llevan a la ventana correspondiente de la categoría seleccionada, como clientes, camiones, fichas, ofertas, etc.

En estas subventanas es donde se encuentras las ventanas que llevan a un nuevo registro, a la modificación de este, a eliminarlo o hacer búsquedas.

PARTE SQL DE LA BASE DE DATOS

El juego está realizado sobre una Base de Datos a la cual se le dan instrucciones mediante el conector JDBC para devolver unos registros, insertarlos, modificarlos o eliminarlos

Las operaciones de consulta siguen el patrón de “SELECT * FROM nombre_tabla WHERE condición”.

El asterisco indica que queremos que nos devuelva todos los campos de la tabla (nombre_tabla). El nombre_tabla ya indica que es el nombre de la tabla en la que vamos a trabajar y la condición se podría definir como una característica que debe poseer el registro para que sea devuelto. Por ejemplo, “WHERE campo_edad = 18, lo que nos devolver los registros de la tabla cuyo campo edad sea igual a 18.

Las operaciones de inserción siguen el patrón de “INSERT INTO nombre_tabla (campo1, campo2) VALUES (valor1, valor2)”

Hay que poner el nombre de la tabla juntos con el nombre de los campos que tiene y añadir los valores a dichos campos.

Las operaciones de edición siguen el patrón “UPDATE nombre_tabla SET campo1=valor1 WHERE campo2=valor2”

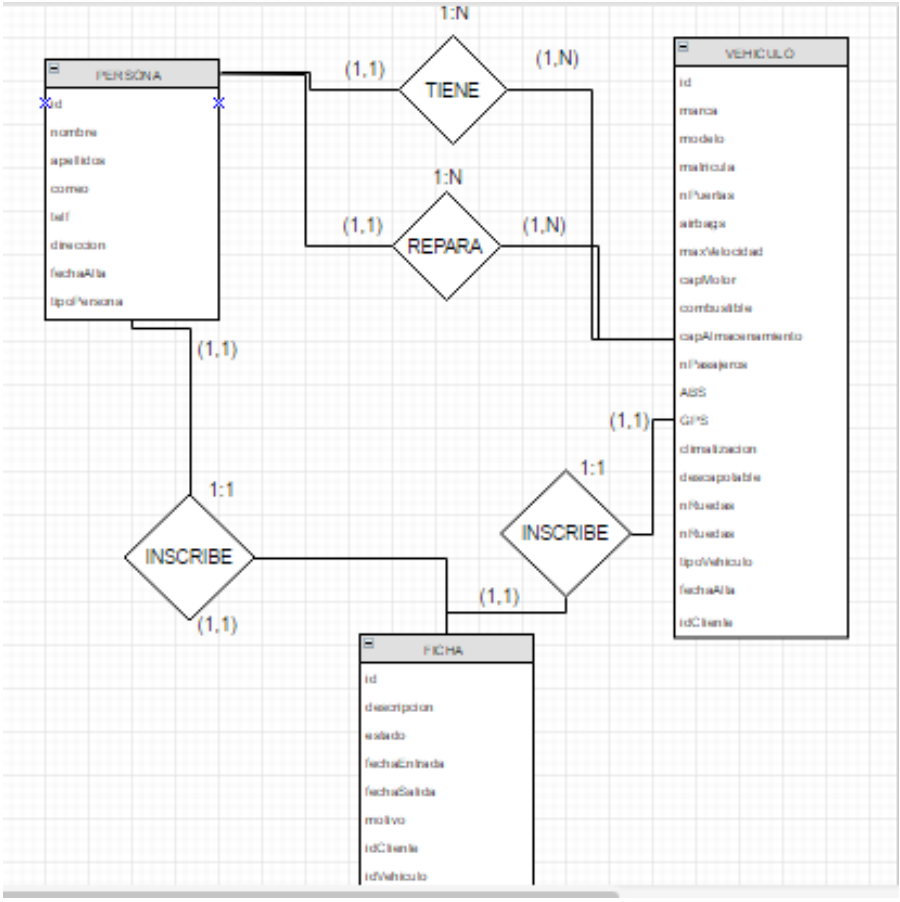
Poniendo el nombre de la tabla, se indica que campo se modifica seguido del nuevo valor y en la condición se pone una característica que tenga el registro para modificarlo. Frecuentemente se usa un id que sea distinto en todos los registros para ser más fácil su localización. De la otra forma puede afectar a varios registros más, además del que queremos seleccionar.

Las operaciones de borrado siguen el patrón “DELETE * FROM nombre_tabla WHERE condición”

Indicando la tabla en la que se quiere borrar, se indica que campos borrarse también (* = todos). Es importantísimo poner la condición en este tipo de operaciones ya que si falta la condición WHERE podemos hacer un borrado de todos los registros y dejar totalmente la tabla vacía. Por eso también se usa frecuentemente el id autoincremental.

7. Definición de Base de Datos

7.1. Modelo Entidad-Relación



7.2. Modelo Relacional

PERSONA (id, dni, nombre, apellidos, telf, correo, dirección, fechaAlta, FK tipoPersona)

TIPO_PERSONA (id, nombre_tipo)

VEHICULO (id, FK idCliente, FK tipoVehiculo, marca, modelo, matricula, nPuertas, airbags, maxVelocidad, capMotor, combustible, capAlmacenamiento, nPasajeros, ABS, GPS, climatizacion, descapotable, nRuedas, motivoVisita, fechaAlta)

TIPO_VEHICULO (id, nombre_tipo)

FICHA (id, FK idCliente, FK idVehiculo, FK idMecanico, descripcion, FK estado, fechaEntrada, fechaSalida, motivo)

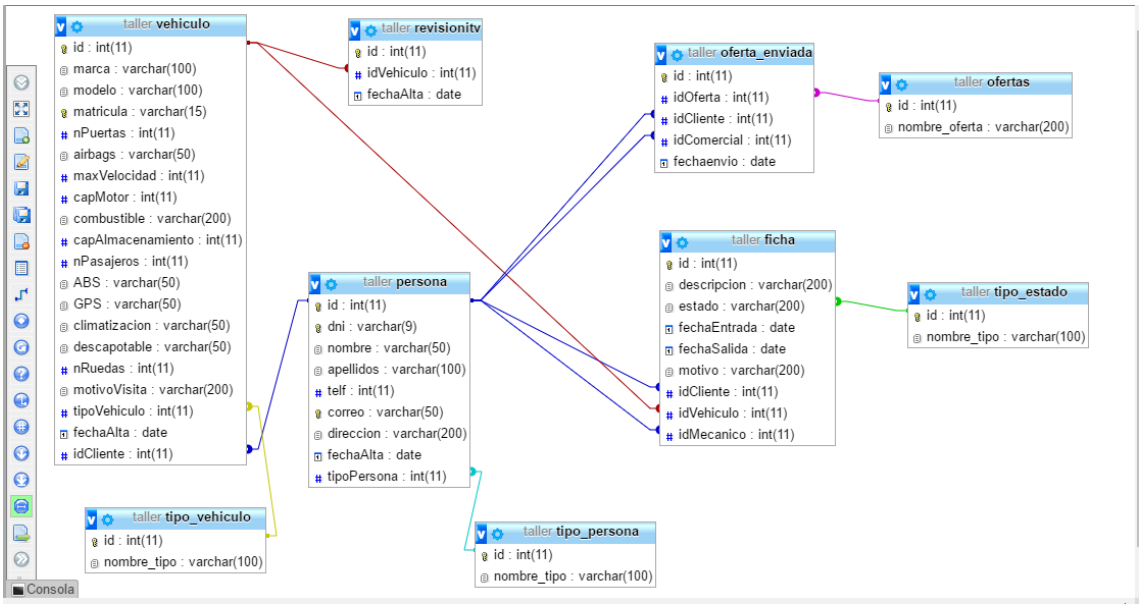
TIPO_ESTADO (id, nombre_tipo)

OFERTA (id, nombre_oferta)

OFERTA_ENVIADA (id, FK idOferta, FK idCliente, FK idComercial, fechaenvio)

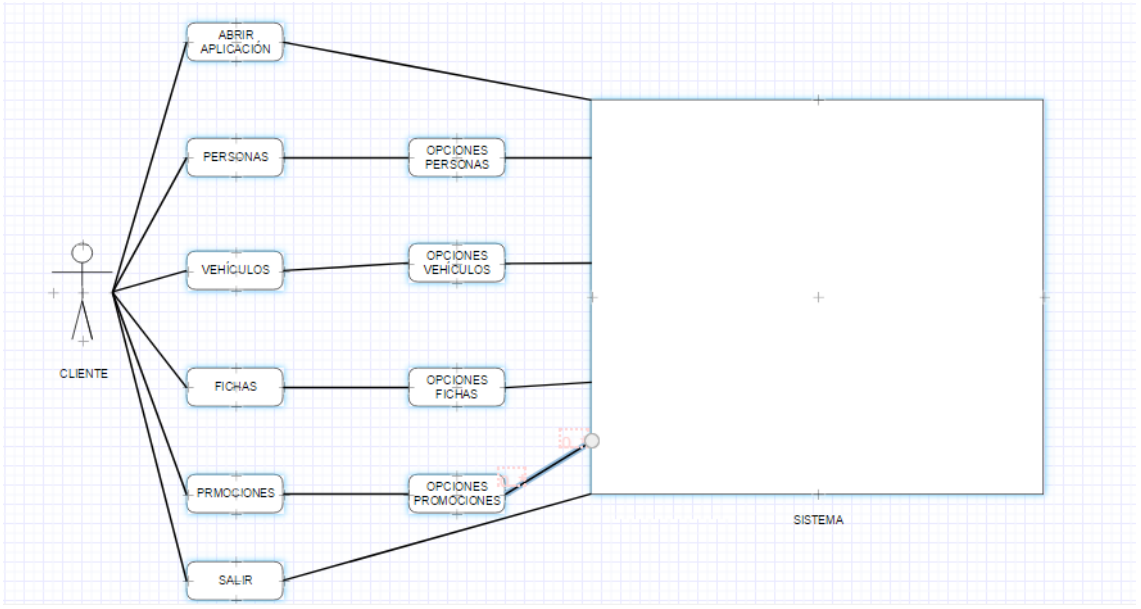
REVISIONITV (id, FK idVehiculo, fechaAlta)

7.3. Lista de tablas



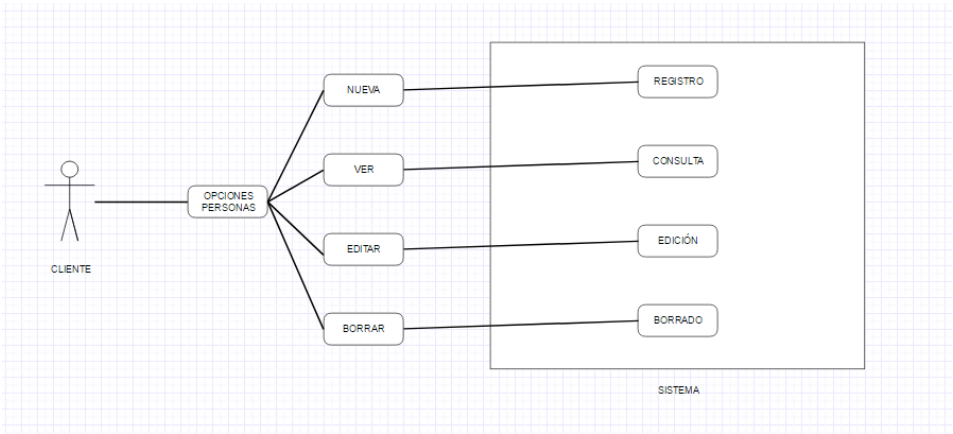
8. Casos de uso

El diagrama de casos de uso representa la forma en como un Cliente (Actor) opera con el sistema en desarrollo, además de la forma, tipo y orden en como los elementos interactúan. En este caso se muestran las distintas formas en las que se puede encontrar la aplicación.



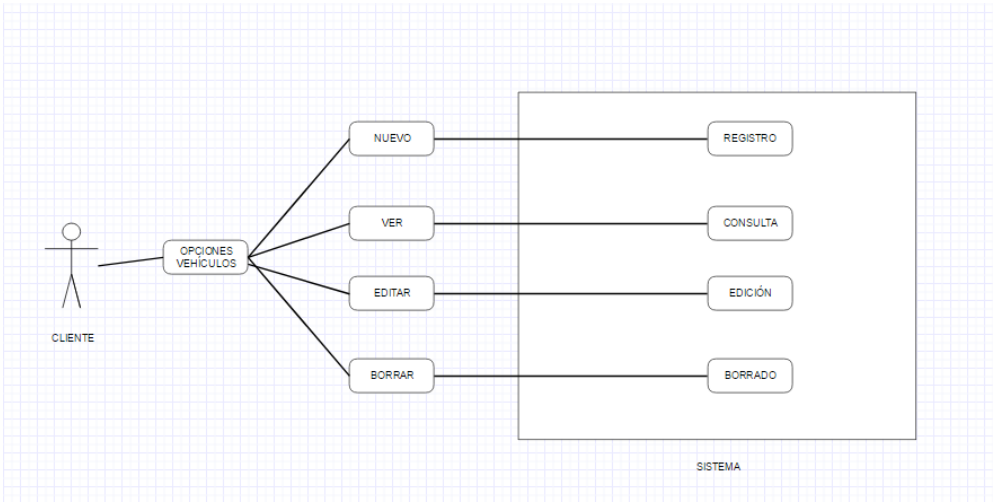
8.1. Caso de uso Persona

Tienen la misma funcionalidad para las ventanas Cliente, Comercial y Mecánico.

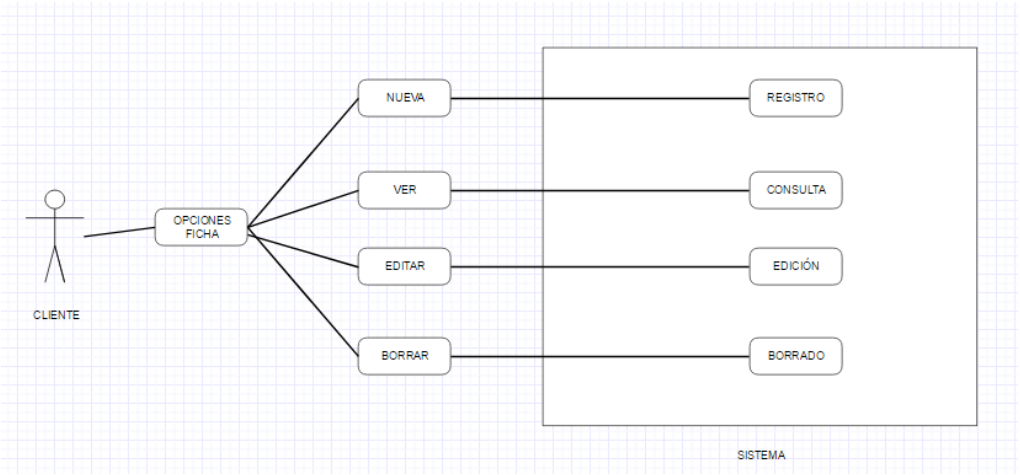


8.2. Caso de uso Vehículo

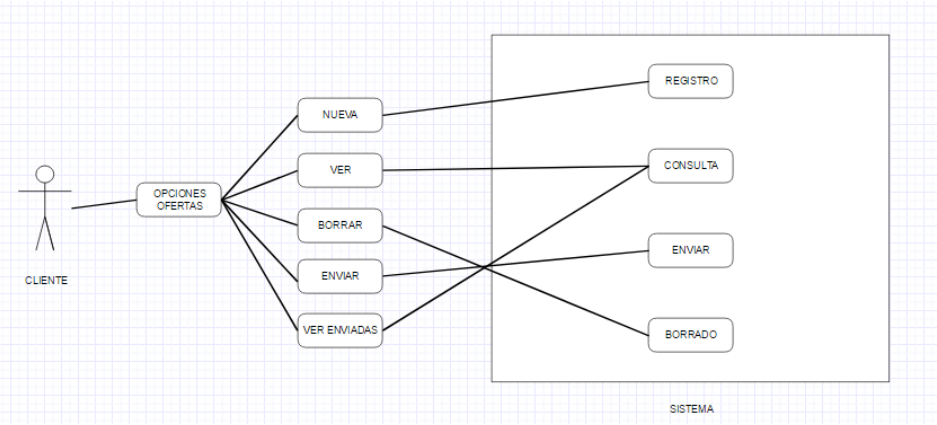
Tienen la misma funcionalidad para los distintos tipos de vehículos existentes.



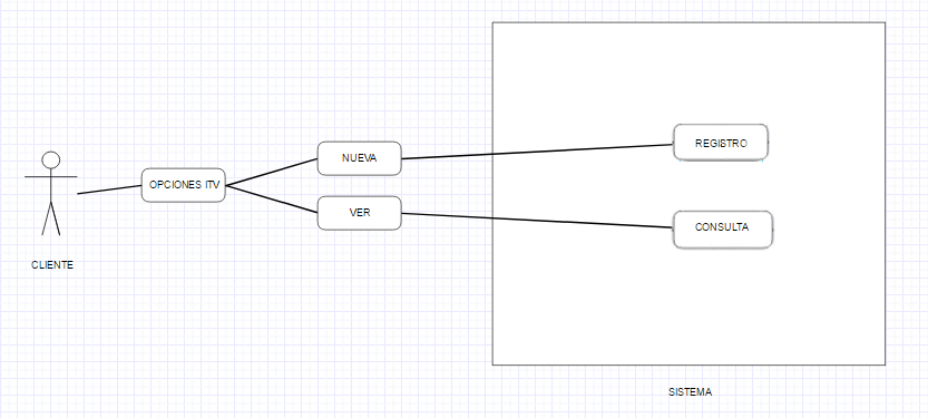
8.3. Caso de uso Ficha



8.4. Caso de uso Ofertas

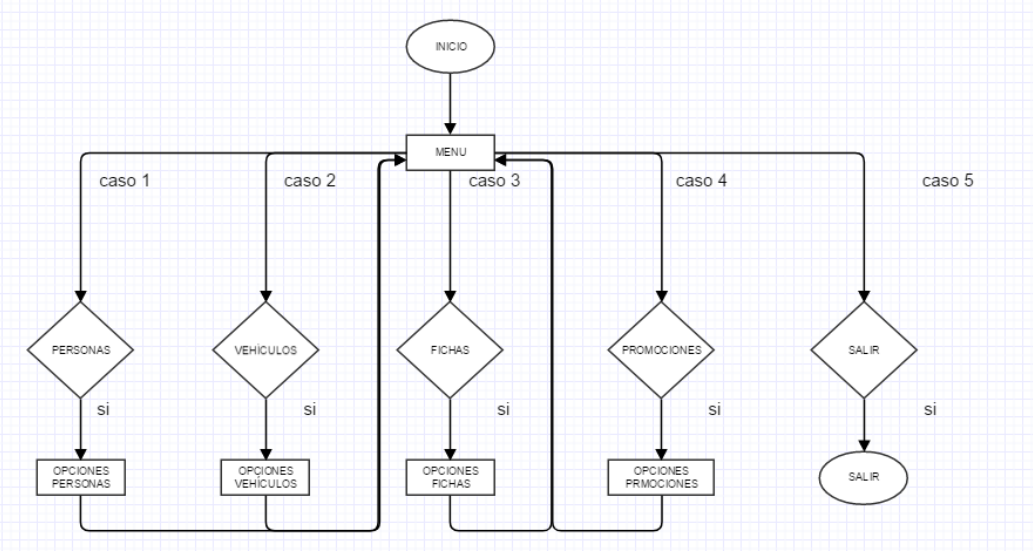


8.5. Caso de uso ITV



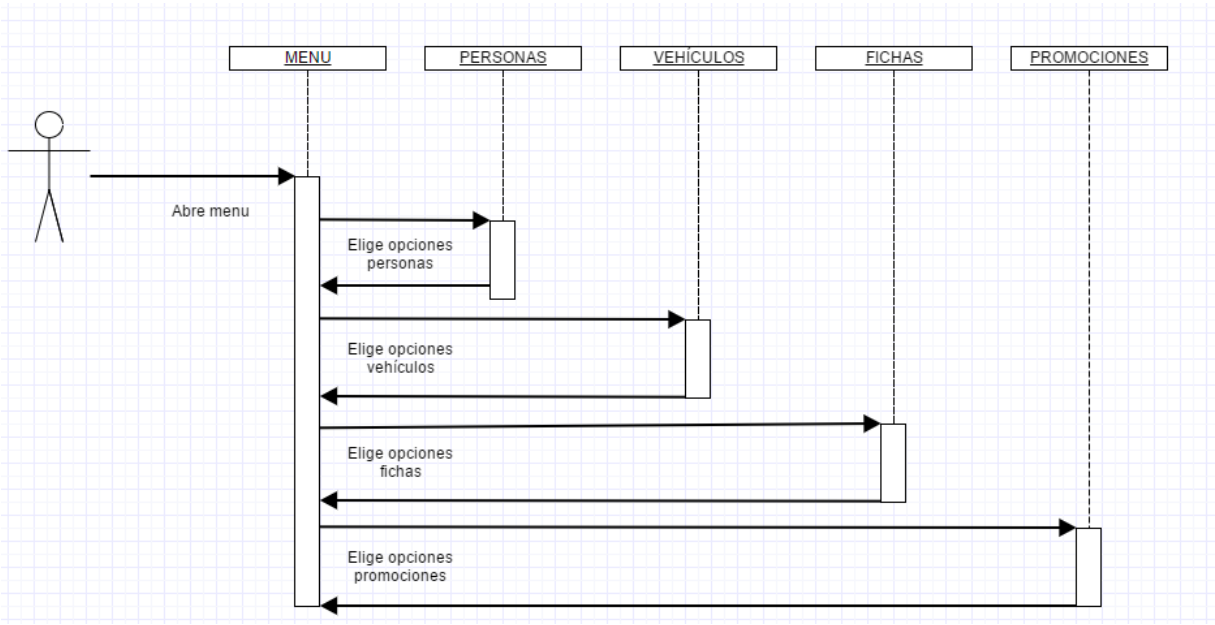
9. Diagrama de flujo

El diagrama de flujo es una representación gráfica de un proceso. Ofrece una descripción visual de las actividades implicadas en un proceso mostrando la relación secuencial entre ella, facilitando la rápida comprensión de cada actividad y su relación con las demás.



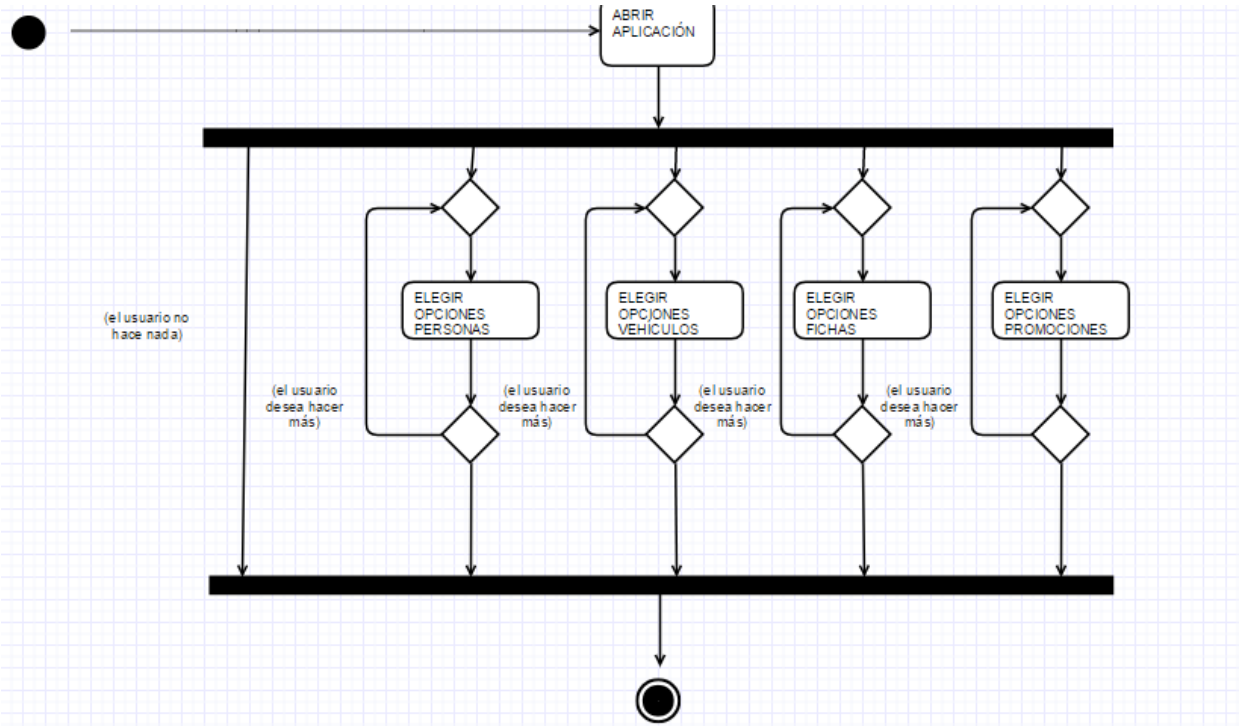
10. Diagramas de secuencia

Muestra una iteración que representa la secuencia de mensajes entre las instancias de las clases, componentes, subsistemas o actores. El tiempo fluye cara abajo en el diagrama y muestra el flujo de control de un participante a otro.



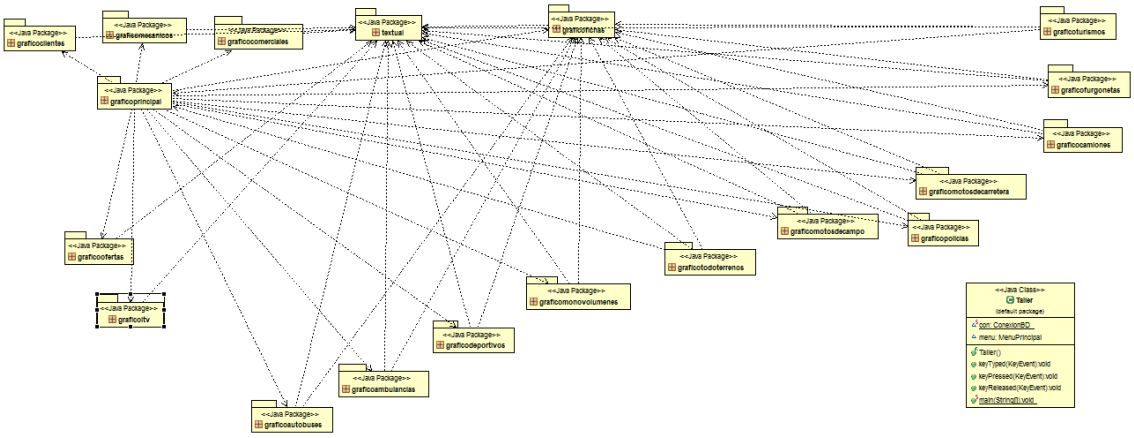
11. Diagramas de actividades

Se muestra un proceso de negocio o un proceso de software como un flujo de trabajo a través de una serie de acciones que pueden ser llevadas a cabo por personas, componentes de software o equipos.

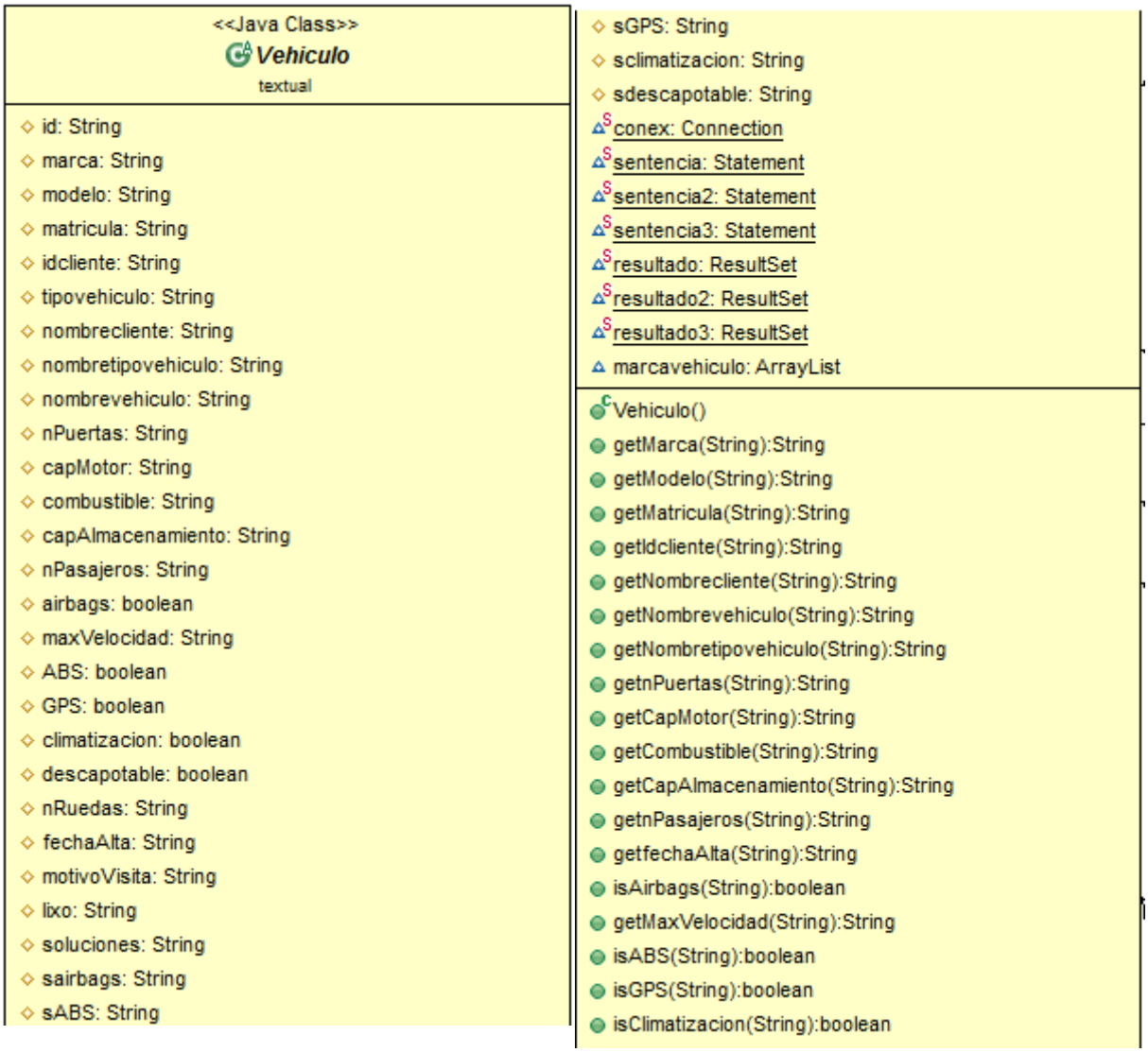


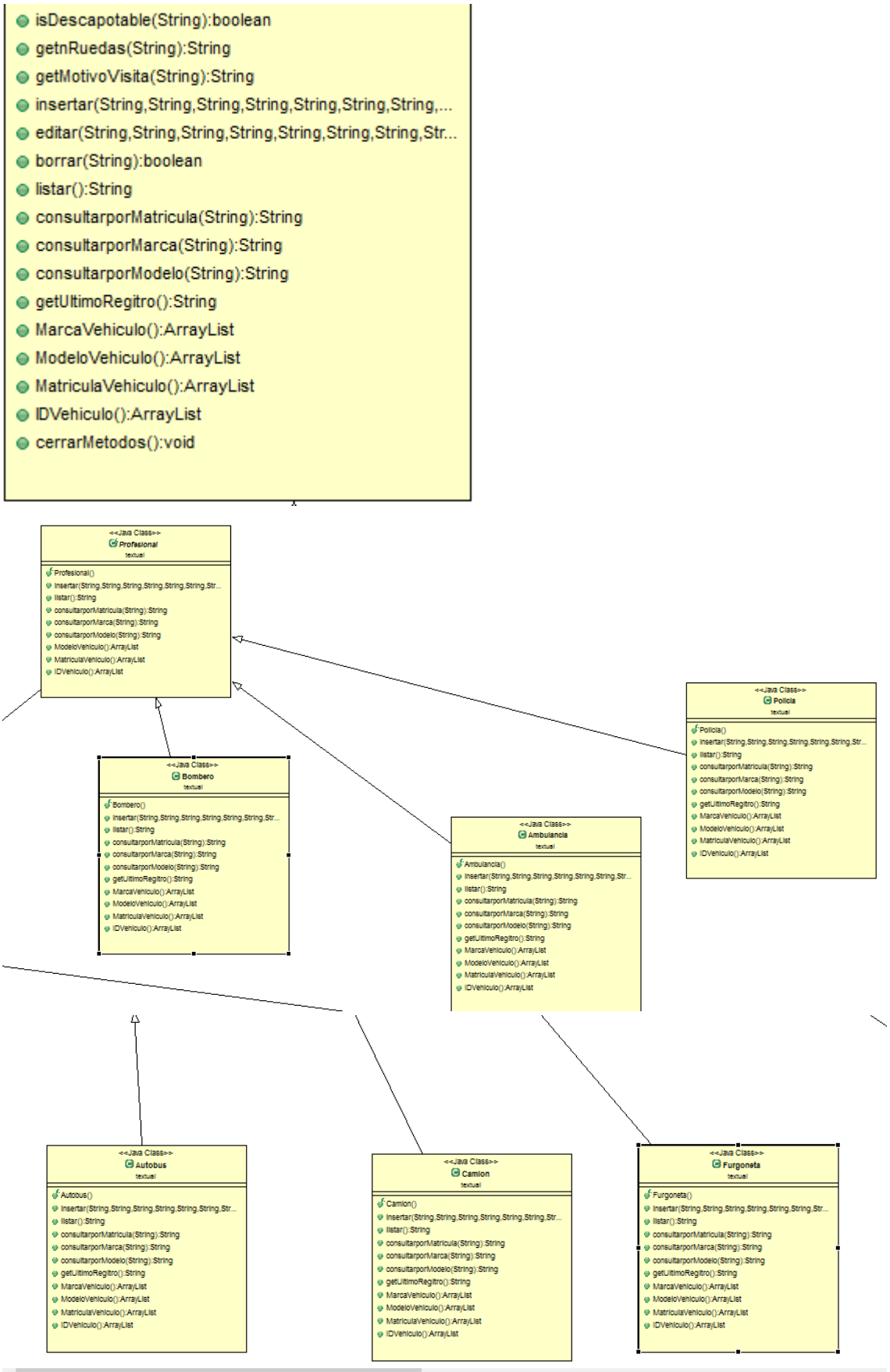
12. Diagrama de clases

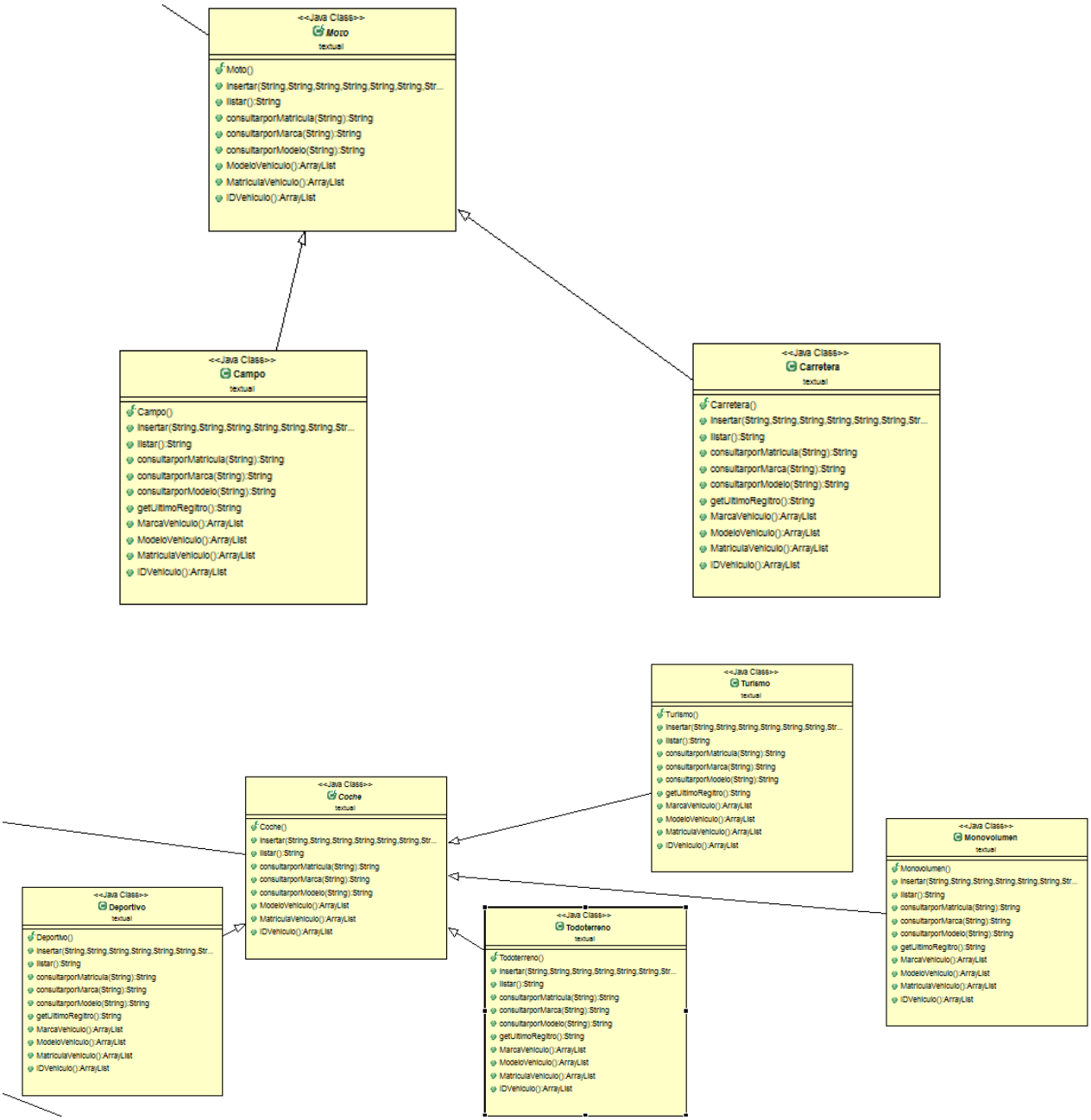
12.1. Paquetes



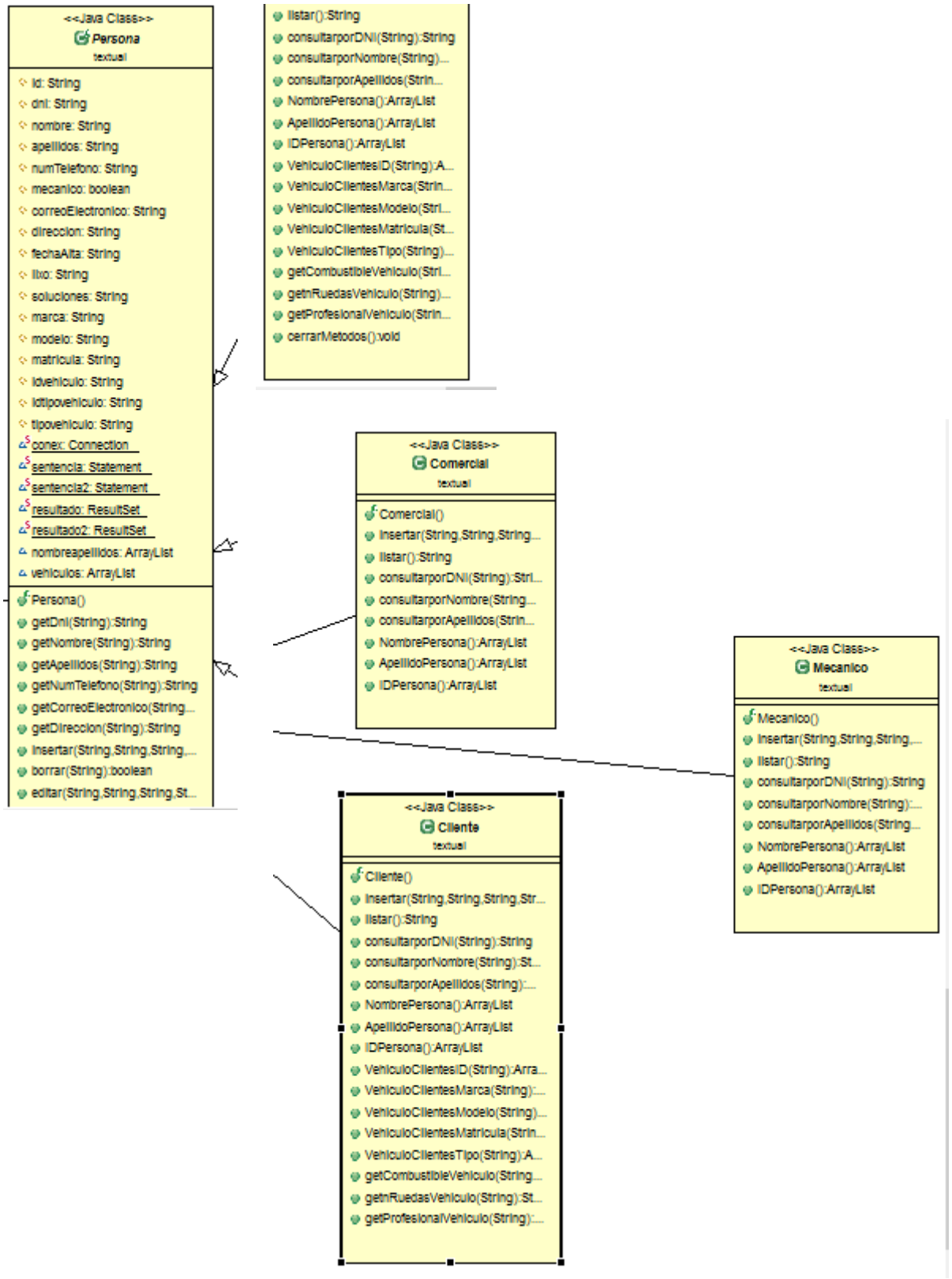
12.2. Paquete textual



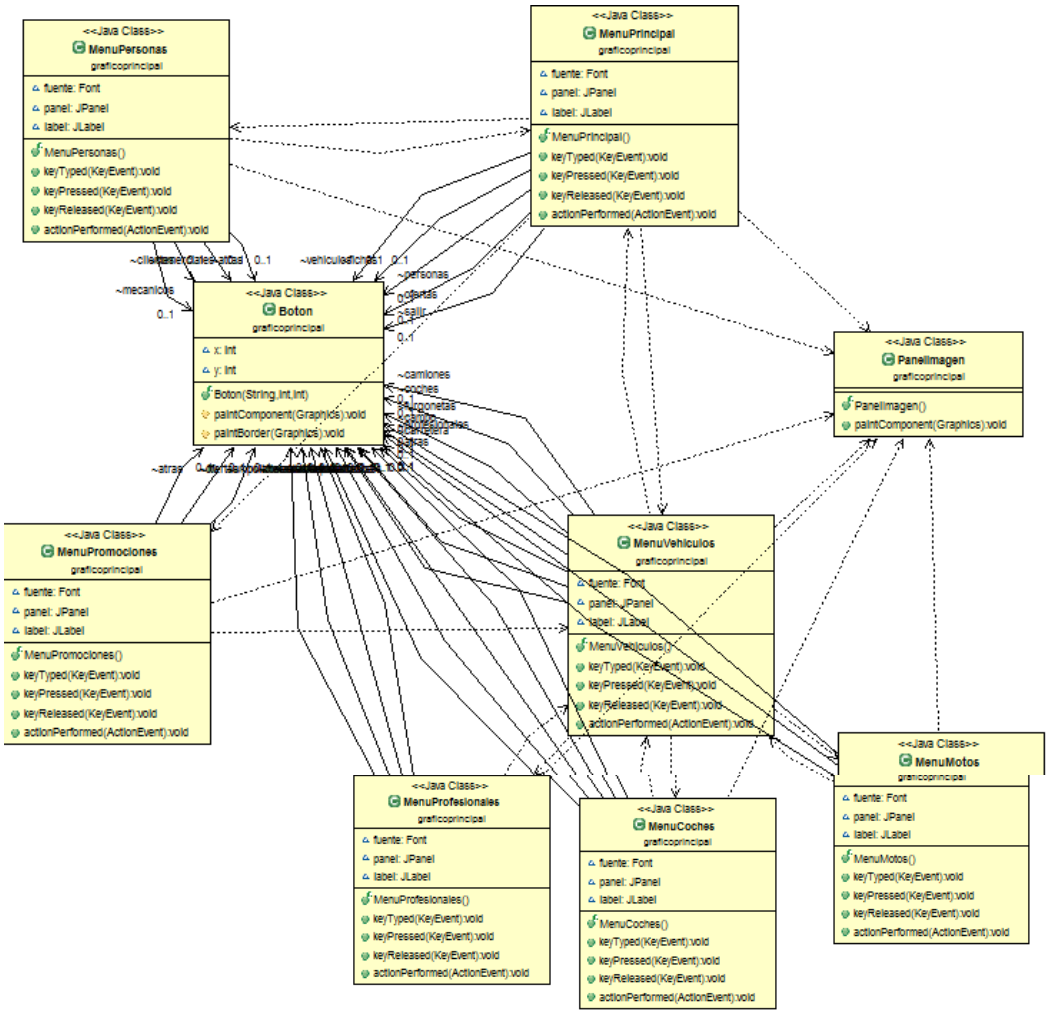

isDescapotable(String):boolean
getnRuedas(String):String
getMotivoVisita(String):String
insertar(String,String,String,String,String,String,String,...
editar(String,String,String,String,String,String,String,Str...
borrar(String):boolean
listar():String
consultarporMatricula(String):String
consultarporMarca(String):String
consultarporModelo(String):String
getUltimoRegitro():String
MarcaVehiculo():ArrayList
ModeloVehiculo():ArrayList
MatriculaVehiculo():ArrayList
IDVehiculo():ArrayList
cerrarMetodos():void
<<Java Class>>
Profesional
textual
Profesional()
Bombero
Policia
Ambulancia
Autobus
Camion
Furgoneta


<<Java Class>>
Moto
textual
Moto()
insertar(String,String,String,String,String,String,Str...
llistar():String
consultarporMatricula(String):String
consultarporMarca(String):String
consultarporModelo(String):String
ModeloVehiculo():ArrayList
MatriculaVehiculo():ArrayList
IDVehiculo():ArrayList
<<Java Class>>
Campo
textual
Campo()
insertar(String,String,String,String,String,String,Str...
llistar():String
consultarporMatricula(String):String
consultarporMarca(String):String
consultarporModelo(String):String
getUltimoRegitro():String
MarcaVehiculo():ArrayList
ModeloVehiculo():ArrayList
MatriculaVehiculo():ArrayList
IDVehiculo():ArrayList
<<Java Class>>
Carretera
textual
Carretera()
insertar(String,String,String,String,String,String,Str...
llistar():String
consultarporMatricula(String):String
consultarporMarca(String):String
consultarporModelo(String):String
getUltimoRegitro():String
MarcaVehiculo():ArrayList
ModeloVehiculo():ArrayList
MatriculaVehiculo():ArrayList
IDVehiculo():ArrayList
<<Java Class>>
Turismo
textual
Turismo()
insertar(String,String,String,String,String,String,Str...
llistar():String
consultarporMatricula(String):String
consultarporMarca(String):String
consultarporModelo(String):String
getUltimoRegitro():String
MarcaVehiculo():ArrayList
ModeloVehiculo():ArrayList
MatriculaVehiculo():ArrayList
IDVehiculo():ArrayList
<<Java Class>>
Coche
textual
Coche()
insertar(String,String,String,String,String,String,Str...
llistar():String
consultarporMatricula(String):String
consultarporMarca(String):String
consultarporModelo(String):String
ModeloVehiculo():ArrayList
MatriculaVehiculo():ArrayList
IDVehiculo():ArrayList
<<Java Class>>
Monovolumen
textual
Monovolumen()
insertar(String,String,String,String,String,String,Str...
llistar():String
consultarporMatricula(String):String
consultarporMarca(String):String
consultarporModelo(String):String
getUltimoRegitro():String
MarcaVehiculo():ArrayList
ModeloVehiculo():ArrayList
MatriculaVehiculo():ArrayList
IDVehiculo():ArrayList
<<Java Class>>
Deportivo
textual
Deportivo()
insertar(String,String,String,String,String,String,Str...
llistar():String
consultarporMatricula(String):String
consultarporMarca(String):String
consultarporModelo(String):String
getUltimoRegitro():String
MarcaVehiculo():ArrayList
ModeloVehiculo():ArrayList
MatriculaVehiculo():ArrayList
IDVehiculo():ArrayList
<<Java Class>>
Todoterreno
textual
Todoterreno()
insertar(String,String,String,String,String,String,Str...
llistar():String
consultarporMatricula(String):String
consultarporMarca(String):String
consultarporModelo(String):String
getUltimoRegitro():String
MarcaVehiculo():ArrayList
ModeloVehiculo():ArrayList
MatriculaVehiculo():ArrayList
IDVehiculo():ArrayList

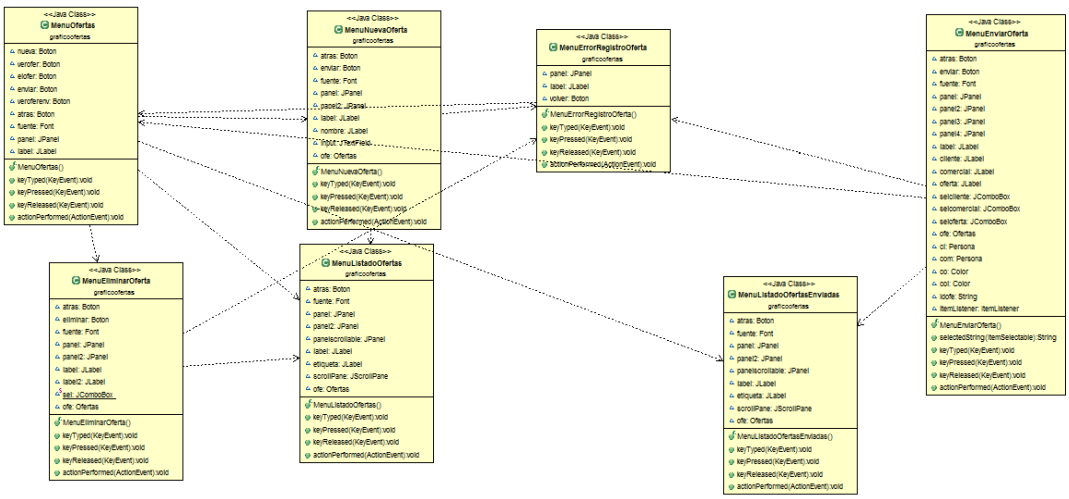

<<Java Class>>
ITV
textual
soluciones: String
idvehiculo: String
marcavehiculo: String
modelovehiculo: String
matriculavehiculo: String
nombrevehiculo: String
fecha: String
fechaActual: String
nombrevehiculos: ArrayList
conex: Connection
sentencia: Statement
sentencia2: Statement
resultado: ResultSet
resultado2: ResultSet
ITV()
insertar(String):boolean
listar():String
VehiculoID():ArrayList
VehiculoMarca():ArrayList
VehiculoModelo():ArrayList
VehiculoMatricula():ArrayList
cerrarMetodos():void
<<Java Class>>
Ofertas
textual
idofertaenviada: String
nombreoferta: String
clienteoferta: String
clienteapeloferta: String
correoclienteoferta: String
correocomercialoferta: S...
comercialoferta: String
idoferta: String
idcomercial: String
idcliente: String
soluciones: String
fecha: String
fechaActual: String
nombreofertas: ArrayList
conex: Connection
sentencia: Statement
sentencia2: Statement
sentencia3: Statement
sentencia4: Statement
resultado: ResultSet
resultado2: ResultSet
resultado3: ResultSet
resultado4: ResultSet
Ofertas()
getFecha(String):String
insertar(String):boolean
enviar(String,String,Strin...
listar():String
listarEnv():String
borrar(String):boolean
NombreOfertas():ArrayList
IDOfertas():ArrayList
ClienteDisponiblesOferta...
cerrarMetodos():void
<<Java Class>>
Ficha
textual
id: String
idvehiculo: String
idcliente: String
idmecanico: String
idestado: String
descripcion: String
estado: String
motivoParado: String
fechaEntrada: String
fechaSalida: String
soluciones: String
nombrecliente: String
nombremecanico: String
nombrevehiculo: String
nombreficha: ArrayList
conex: Connection
sentencia: Statement
sentencia2: Statement
sentencia3: Statement
sentencia4: Statement
sentencia5: Statement
sentencia6: Statement
resultado: ResultSet
resultado2: ResultSet
resultado3: ResultSet
resultado4: ResultSet
resultado5: ResultSet
resultado6: ResultSet
Ficha()
getidvehiculo(String):Str...
getNombrevehiculo(Stri...
setidvehiculo(String):void
getidcliente(String):String
getNombrecliente(String...
setidcliente(String):void
getidmecanico(String):S...
getNombremecanico(Str...
setidmecanico(String):v...
getDescripcion(String):...
setDescripcion(String):...
getMotivoParado(String)...
setMotivoParado(String)...
getFechaEntrada(String...
setFechaEntrada(String...
getFechaSalida(String):...
setFechaSalida(String):...
getEstado(String):String
setEstado(String):void
insertar(String,String,St...
editar(String,String,Strin...
borrar(String):boolean
listar():String
consultarporMecanico(...
consultarporCliente(Stri...
consultarporEstado(Stri...
FichaCliente():ArrayList
FichaVehiculo():ArrayList
FichaMecanico():Array...
FichaID():ArrayList
cerrarMetodos():void


<<Java Class>>
Persona
textual
id: String
dni: String
nombre: String
apellidos: String
numTelefono: String
mecanico: boolean
correoElectronico: String
direccion: String
fechaAlta: String
libro: String
soluciones: String
marca: String
modelo: String
matricula: String
idvehiculo: String
idtipovehiculo: String
tipovehiculo: String
conex: Connection
sentencia: Statement
sentencia2: Statement
resultado: ResultSet
resultado2: ResultSet
nombreapellidos: ArrayList
vehiculos: ArrayList
Persona()
getDni(String):String
getNombre(String):String
getApellidos(String):String
getNumTelefono(String):String
getCorreoElectronico(String...
getDireccion(String):String
insertar(String,String,String,...
borrar(String):boolean
editar(String,String,String,St...
listar():String
consultarporDNI(String):String
consultarporNombre(String)...
consultarporApellidos(Strin...
NombrePersona():ArrayList
ApellidoPersona():ArrayList
IDPersona():ArrayList
VehiculoClientesID(String):A...
VehiculoClientesMarca(Strin...
VehiculoClientesModelo(Stri...
VehiculoClientesMatricula(St...
VehiculoClientesTipo(String)...
getCombustibleVehiculo(Stri...
getnRuedasVehiculo(String)...
getProfesionalVehiculo(Strin...
cerrarMetodos():void
<<Java Class>>
Comercial
textual
Comercial()
insertar(String,String,String...
listar():String
consultarporDNI(String):Stri...
consultarporNombre(String...
consultarporApellidos(Strin...
NombrePersona():ArrayList
ApellidoPersona():ArrayList
IDPersona():ArrayList
<<Java Class>>
Mecanico
textual
Mecanico()
insertar(String,String,String,...
listar():String
consultarporDNI(String):String
consultarporNombre(String):...
consultarporApellidos(String...
NombrePersona():ArrayList
ApellidoPersona():ArrayList
IDPersona():ArrayList
<<Java Class>>
Cliente
textual
Cliente()
insertar(String,String,String,Str...
listar():String
consultarporDNI(String):String
consultarporNombre(String):St...
consultarporApellidos(String):...
NombrePersona():ArrayList
ApellidoPersona():ArrayList
IDPersona():ArrayList
VehiculoClientesID(String):Arra...
VehiculoClientesMarca(String):...
VehiculoClientesModelo(String)...
VehiculoClientesMatricula(Strin...
VehiculoClientesTipo(String):A...
getCombustibleVehiculo(String...
getnRuedasVehiculo(String):St...
getProfesionalVehiculo(String):...

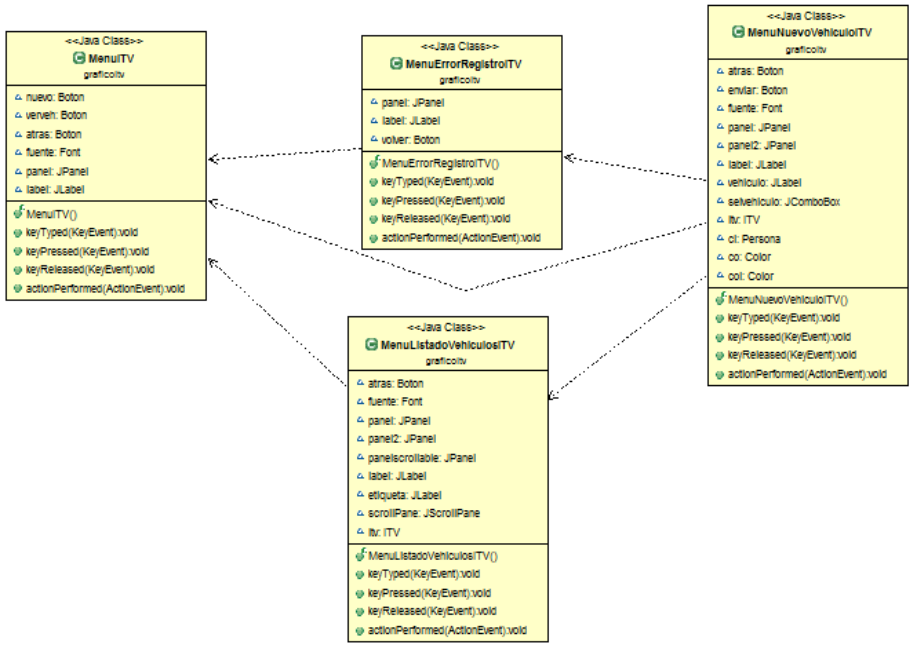
12.3. Paquete graficoprincipal



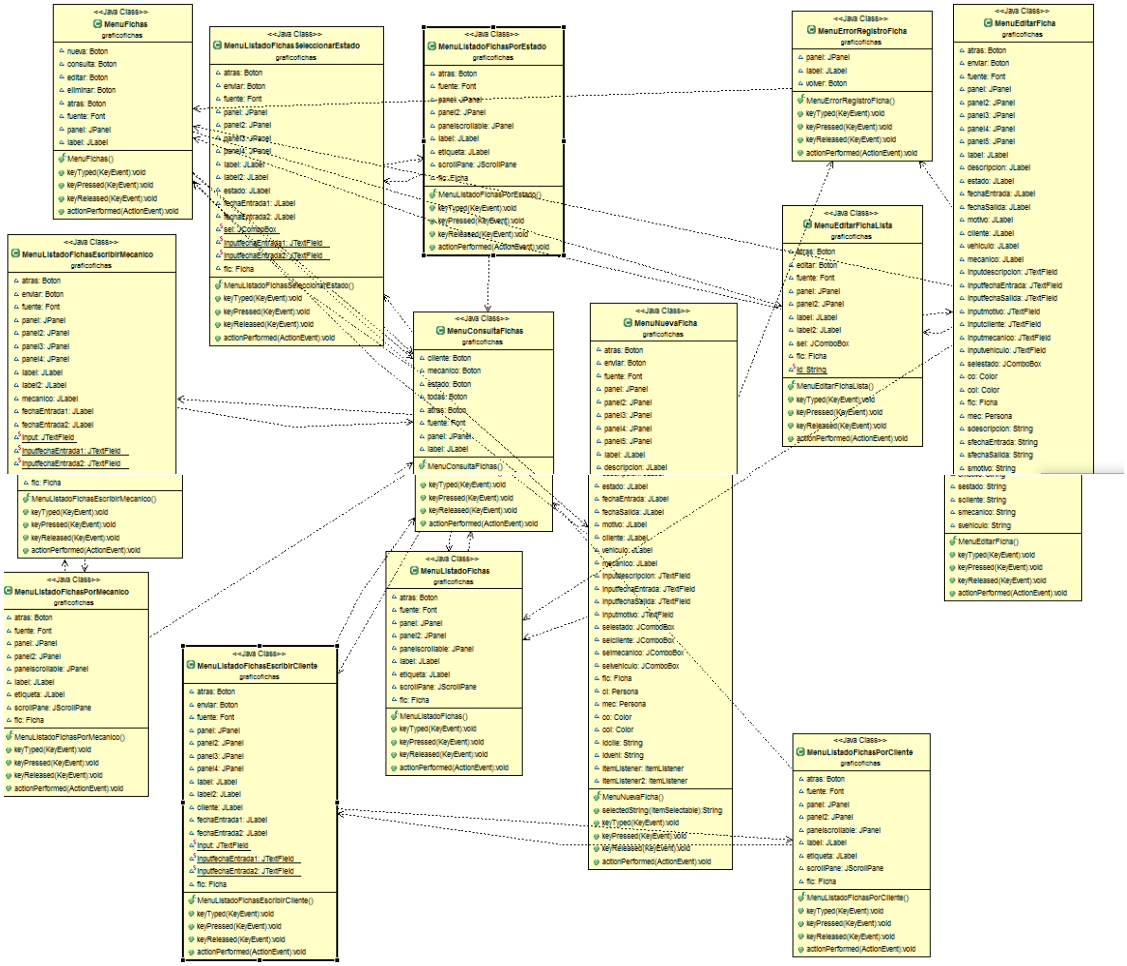
12.4. Paquete graficoofertas



12.5. Paquete graficoitv

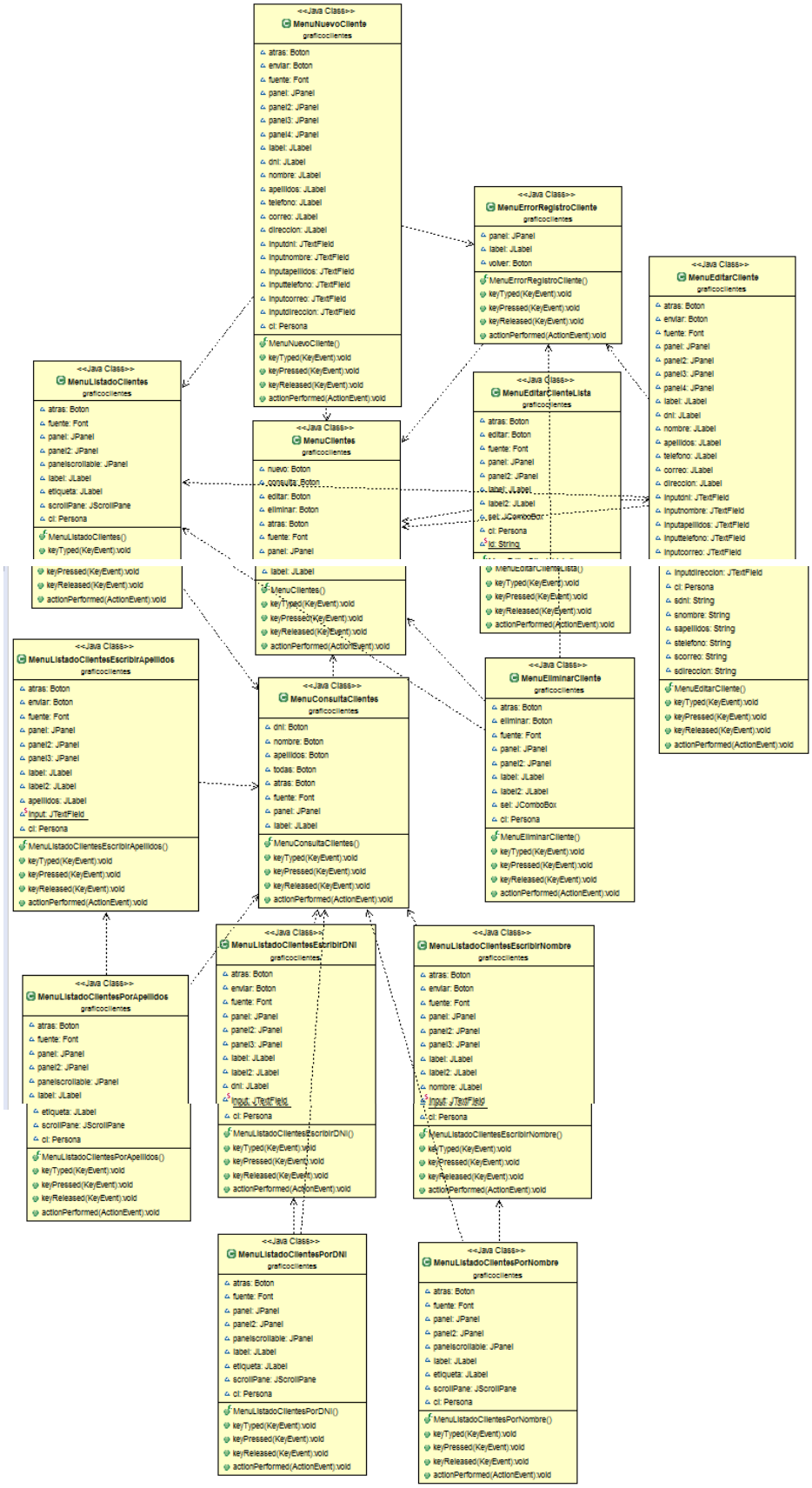


12.6. Paquete graficofichas



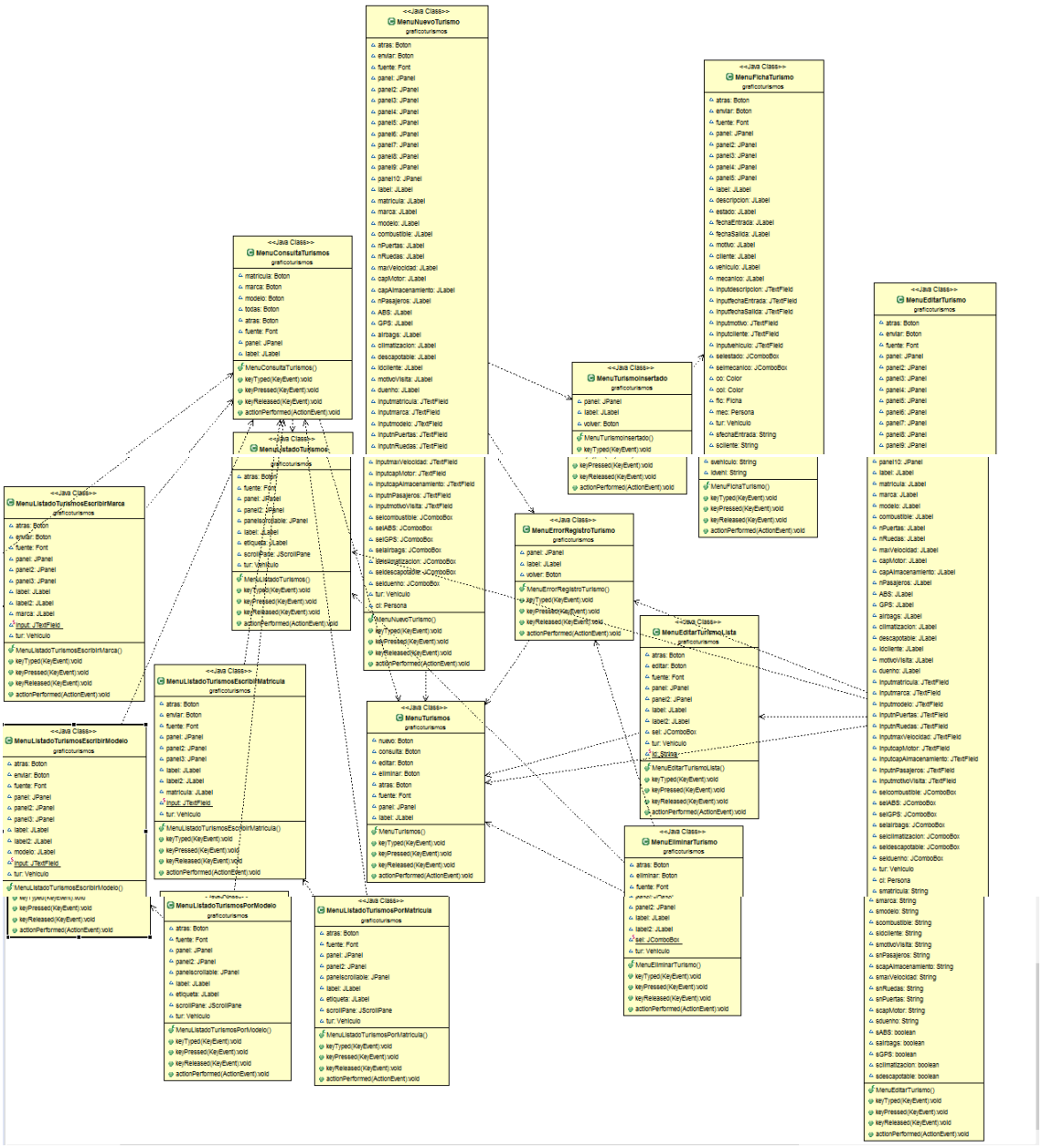
12.7. Paquete graficoclientes

Es igual para todos los gráficos de tipo Persona (graficoclientes, graficomecanicos, graficocomerciales)



12.8. Paquete textual graficoturismos

Es igual para todos los gráficos de tipo Vehiculo (graficoturismos, graficodeportivos, graficomonovolumenes, graficotodoterrenos, graficomotosdecampo, graficomotosdecarretera, graficoambulancias, graficobomberos, graficopolicias, graficoautobuses, graficocamiones, graficofurgonetas)



13. Pruebas

13.1. Pruebas de contenido

Estas pruebas como su nombre indica, buscan verificar que el contenido del sistema sea coherente y consistente a la vez. También debe verificar que las palabras usadas para transmitir una idea al usuario sean las acertadas y que la idea transmitida sea la misma.

Gran parte de las pruebas de contenido realizadas al sistema están enfocadas a los cuestionarios, ya que el objetivo principal del sistema es la iteración del usuario con los cuestionarios.

13.2. Pruebas de código

Para las pruebas de código la herramienta más eficaz sería la herramienta “profesional” del w3c, que es de pago y mucho más completa y eficiente que las gratuitas a las que se pueden acceder desde su web

13.3. Pruebas unitarias

Comprobar los módulos individuales del sistema.

Durante el desarrollo de las clases de la aplicación, se hicieron pruebas individuales para detectar más fácilmente cualquier problema que pudiese aparecer. Finalmente, su resultado fue el correcto.

13.4. Pruebas de integración

Comprobar que las partes del sistema de un cierto nivel funcionan correctamente todas juntas. Todas las clases, tanto las del JFrame con las del entorno gráfico interactúan entre ellas correctamente, incluidos todos los eventos por parte de los botones.

13.5. Pruebas funcionales

Este tipo de pruebas comprueban las necesidades de funcionamiento, acorde con el diseño. Verifican que el sistema lleve a cabo correctamente todas las funciones requeridas, la validación de los datos y se deben realizar pruebas de comportamiento ante distintos escenarios.
Estas pruebas deben estar enfocadas a tareas, a límites del sistema, las condiciones planeadas de errores y de exploración. Para estas pruebas usamos los esquemas de pruebas de caja negra ya que nos interesa saber si funcionan o no, independientemente de la forma que lo haga.
Debe garantizar que el programa es ejecutado de forma deseada y según los requisitos funcionales para las pruebas de software.

13.6. Pruebas del sistema

Comprobar que el sistema funcione correctamente. La preba de todo en sistema en conjunto. Como es una aplicación de gestión se centra en las operaciones sobre la Base de Datos como consultas, registros, etc. Como en toda aplicación siempre hay la posibilidad de abarcar más, con nuevas clases o nuevos métodos y una vez acabados todos se volvería a realizar esta prueba.

13.7. Pruebas de caja negra

Los sistemas de pruebas de caja negra no consideran la codificación dentro de sus parámetros a evaluar, es decir, que no están basados en el conocimiento del diseño interno del programa. Estas pruebas se enfocan en los requerimientos establecidos en la funcionalidad del sistema. Dichas pruebas son llevadas a cabo sobre la interfaz del software, es decir, de la función, actuando sobre ella como una caja negra, proporcionando unas entradas y estudiando las salidas para ver si concuerdan con las esperadas.

13.8. Pruebas de caja blanca

Al contrario de las pruebas de caja negra, estas se basan en el conocimiento de la lógica interna del código del sistema. Las pruebas contemplan los distintos caminos que se pueden generar gracias a las estructuras condicionales, los distintos estados del mismo, etc.
Así como prueba de caja negra ejercitan los requisitos funcionales de fuera del módulo, las de caja blanca están destinadas a funciones internas. Pruebas de caja blanca son llevadas a cabo en primer lugar en un módulo en particular, para a continuación, realizar la caja negra en varios subsistemas.

13.9. Pruebas de usabilidad

Las pruebas de usabilidad tienen la finalidad de verificar que tan fácil de usar es el sistema. Las pruebas de usabilidad deben verificar aprendizaje (tareas básicas que el usuario hace la primera vez que se entra en el sistema), eficiencia (cuando el usuario ya aprendió algo del sistema, que rápido pueden llegar a hacer las tareas), manejo de errores (cuantos errores realiza el usuario, la gravedad de estés es como se recupera el usuario de ellos) y el grado de satisfacción (que tan satisfactorio es emplear el sistema). Para obtener resultados realistas, es recomendable dejar que las personas que prueben el sistema resuelvan los problemas que se le presentan por sí mismos, porque si les ayudamos ya no tendría sentido el motivo de esta prueba.

13.10. Pruebas de rendimiento, calidad y aceptabilidad

Para las pruebas de rendimiento calidad y adaptabilidad podemos comprobarlas en la herramienta gratuita woorank, que nos ofrece información sobre los diseños, estadísticas muy útiles del posicionamiento de la web en las redes y muchas otras funcionalidades.

13.11. Seguimiento de errores e incidencias

Para el seguimiento de los errores e incidencias bugzilla es una herramienta que nos ayudará con la gestión del desarrollo del software, con seguimiento de bugs y errores, así como cambios en el código.

13.12. Usuarios

Una vez definidas las pruebas, el siguiente paso es buscar a los usuarios que nos ayudarán a realizarlas:

• Empresas. La primera parte de estas pruebas se realiza en paralelo con el desarrollo del sistema ya que son las pruebas de contenido de los cuestionarios. Con estas pruebas se busca que cuando las preguntas se integren al sistema, ya tengan un filtro de pruebas anterior.

Se genera un cuestionario que servirá como retroalimentación y se acompaña de una descripción general del proceso y de lo que se espera de las pruebas. Las preguntas del cuestionario están enfocadas a las pruebas de usabilidad.

• Usuarios comunes. Llevar a cabo las pruebas con usuarios comunes ya que muchas veces los usuarios que realizan las pruebas ya tienen experiencia anterior con sistemas similares, lo que significa que ya pudieron estar familiarizados con muchos aspectos del sistema y habrá puntos del mismo que no se considerarían. Las pruebas realizadas por los usuarios comunes son de usabilidad y funcionalidad, ya que para hacer las evaluaciones de contenido se requiere experiencia en el campo.

• Desarrolladores. Las pruebas realizadas por los desarrolladores son pruebas de caja blanca y de integración, con la finalidad de buscar fallos a partir del conocimiento del código fuente.

13.13. Resultado de las pruebas

Cada fase fue evaluada en una escala del 1 al 5 siendo 1 “muy bajo” y 5 “muy alto”.

Según los resultados obtenidos se podrían dar distintas propiedades.

• **Amigable**. Se refiere a la facilidad de iteración del sistema con el usuario sin tener que consultar un manual o ayuda en línea.

• **Legibilidad**. En esta prueba se evalúa el color de los textos, el contraste de los mismos con el fondo y el tamaño de la fuente, que debe ser adecuado para su legibilidad por la mayoría de los usuarios.

• **Efectividad**. Es cuando una tarea puede ser realizada sin complicación.

• **Eficiencia**. Es cuando las tareas que se llevan a cabo, pueden ser realizadas rápida y fácilmente.

• **Satisfacción**. Es que tan a gusto se quedó una persona con las tareas realizadas en el sistema.

• **Reversibilidad**. Es la capacidad de un sistema para permitir deshacer las acciones realizadas.

• **Autonomía**. Se refiere a que los usuarios deben tener el control sobre el sistema en todo momento.

• **Interfaz gráfica**. Significa que tan buena resulta la navegación en el sistema gracias a la interfaz gráfica. Esto incluye las imágenes, colores y posición de los elementos que conforman el sistema.

14. Propuestas de mejora

Tratándose de una aplicación de gestión, con el crecimiento gradual de un taller, la aplicación también tendría que ir aumentando sus funcionalidades.

- Pueden considerarse mejoras en función de la empresa que lo requiera, añadir, consultar, modificar o eliminar proveedores.
- Añadir un nuevo servicio al taller para venta de vehículos, nuevos y de ocasión, lo cual requeriría consultas, añadir nuevos, eliminarlos, modificarlos, así como el cliente que lo compra.

- Un nuevo apartado para el control de almacén y saber que piezas o herramientas hay, cuantas hay, cuales se necesitan y en caso de necesitarlas solicitarlas al proveedor.

15. Conclusiones

La aplicación proporciona toda las características y funcionalidades necesarias para la administración de un taller a baja escala.

Aunque tal vez algunos talleres necesiten mayor manejo de datos, como proveedores o venta de vehículos, tratándose de bases de datos no supondría mayor complicación dado que las Bases de Datos tienen capacidad para alojar grandes cantidades de datos.

En conclusión, la finalidad para la que la aplicación fue creada la cumple perfectamente y sería cuestión de añadir un par de horas de trabajo para poder realizar las mejoras oportunas en función de quién la solicite.

16. Bibliografía

- https://help.eclipse.org/neon/index.jsp?topic=%2Forg.eclipse.platform.doc.isv%2Freference%2Fapi%2Foverview-summary.html
- http://docs.oracle.com/javase/8/docs/api/
- http://developers.itextpdf.com/apis
- https://dev.mysql.com/downloads/connector/j/

17. Anexo: Código

17.1. Taller.java (main)

```
import java.awt.event.KeyEvent;

import java.awt.event.KeyListener;


import javax.swing.JFrame;

import javax.swing.JPanel;


import graficoprincipal.MenuPrincipal;

import textual.*;

/**

 * @author Adriana Armental Tomé

 * @version 06.03.2017

 */

public class Taller extends JFrame implements KeyListener

{

        static ConexionBD con;

   MenuPrincipal menu;



   public Taller()

   {

        menu = new MenuPrincipal();

        menu.setVisible(true);


                //para desactivar o espacio no control de menu

                JPanel panel = (JPanel) menu.getContentPane();


        panel.addKeyListener(this);

                panel.setFocusable(true);
```

```
        con = new ConexionBD();
    }

        @Override
        public void keyTyped(KeyEvent e) {}

        //control de espacio
        @Override
        public void keyPressed(KeyEvent e) {
                int key = e.getKeyCode(); //collese a tecla soltada

                if(key == KeyEvent.VK_SPACE) { //si e o espacio
                        e.consume();
                }
        }

        @Override
        public void keyReleased(KeyEvent e) {
        }

    public static void main(String[] args) {
        new Taller();

    }

}
```

17.2. ConexionBD.java

```
package textual;
```

```
import java.sql.*;
/**
 * Conexion es una clase para conectar con la BD .
 *
 * @author Adriana Armental Tomé
 * @version 06.03.2017
 */
public class ConexionBD
{
   static final String DATABASE_URL = "jdbc:mysql://localhost/taller";
   static final String USER = "root";
   static final String PASSWORD = "";
   Connection con;

   public ConexionBD()
   {
      try {
          //Carregamos o driver JDBC
         Class.forName("com.mysql.jdbc.Driver");
         con = DriverManager.getConnection(DATABASE_URL, USER, PASSWORD);
      } catch (ClassNotFoundException e) {
         e.printStackTrace();
      } catch(SQLException e) {
         e.printStackTrace();
      }
   }

          public Connection getCon() {
                    return con;
          }
```

```
}
```

17.3. Persona.java

```
package textual;


import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;


/**
 * @author Adriana Armental Tomé
 * @version 06.03.2017
 */
public abstract class Persona
{
   // instance variables - replace the example below with your own
   protected String id;
   protected String dni;
   protected String nombre;
   protected String apellidos;
   protected String numTelefono;
   protected boolean mecanico;
   protected String correoElectronico;
   protected String direccion;
   protected String fechaAlta;
   protected String lixo;
   protected String soluciones;
   protected String marca, modelo, matricula, idvehiculo, idtipovehiculo, tipovehiculo;
```

```
   static ConexionBD con;
   static Connection conex;
   static Statement sentencia, sentencia2;
   static ResultSet resultado, resultado2;
         ArrayList nombreapellidos, vehiculos;

   public Persona() {
      con = new ConexionBD();
      conex = con.getCon();
      try {
                        sentencia = conex.createStatement();
                        sentencia2 = conex.createStatement();
                  } catch (SQLException e) {
                        e.printStackTrace();
                  }
   }

   public String getDni(String id) {
         try {
                        resultado = sentencia.executeQuery("SELECT * FROM persona WHERE
id='"+id+"'");
                        while (resultado.next()) {
                              dni = resultado.getString(2); //dni
                        }

                  } catch (SQLException e) {
                        e.printStackTrace();
                  }
                  return dni;
         }
```

```
	public String getNombre(String id) {
	try {
			resultado = sentencia.executeQuery("SELECT * FROM persona WHERE
id='"+id+"'");
			while (resultado.next()) {
				nombre = resultado.getString(3); //nombre
			}

		} catch (SQLException e) {
			e.printStackTrace();
		}
		return nombre;
	}

	public String getApellidos(String id) {
	try {
			resultado = sentencia.executeQuery("SELECT * FROM persona WHERE
id='"+id+"'");
			while (resultado.next()) {
				apellidos = resultado.getString(4); //apellidos
			}

		} catch (SQLException e) {
			e.printStackTrace();
		}
		return apellidos;
	}

	public String getNumTelefono(String id) {
	try {
			resultado = sentencia.executeQuery("SELECT * FROM persona WHERE
id='"+id+"'");
```

```
                while (resultado.next()) {
                        numTelefono = resultado.getString(5); //telefono
                }

        } catch (SQLException e) {
                e.printStackTrace();
        }
        return numTelefono;
    }

    public String getCorreoElectronico(String id) {
    try {
                resultado = sentencia.executeQuery("SELECT * FROM persona WHERE
id='"+id+"'");
                while (resultado.next()) {
                        correoElectronico = resultado.getString(6); //correo
                }

        } catch (SQLException e) {
                e.printStackTrace();
        }
        return correoElectronico;
    }

    public String getDireccion(String id) {
    try {
                resultado = sentencia.executeQuery("SELECT * FROM persona WHERE
id='"+id+"'");
                while (resultado.next()) {
                        direccion = resultado.getString(7); //direccion
                }
```

```
		} catch (SQLException e) {
			e.printStackTrace();
		}
		return direccion;
	}

	public boolean insertar(String dni, String nombre, String apellidos, String numTelefono,
String correoElectronico, String direccion) {
		return false;
	}

	public boolean borrar(String id) {

		try {
		sentencia.executeUpdate("DELETE FROM persona WHERE id='"+id+"'");
		return true;

		} catch(SQLException e) {
			e.printStackTrace();
			return false;
		}
	}

	public boolean editar(String id, String nombre, String apellidos, String telefono, String
correo, String direccion) {

		try {
		sentencia.executeUpdate("UPDATE persona SET dni='"+dni+"',
nombre='"+nombre+"', apellidos='"+apellidos+"', telf='"+telefono+"', correo='"+correo+"',
direccion='"+direccion+"' WHERE id = '"+id+"'");
```

```
                return true;

                } catch(SQLException e) {
                        e.printStackTrace();
                        return false;
                }

        }

        public String listar(){
                return null;
        }

  public String consultarporDNI(String dni) {
                return null;
        }

  public String consultarporNombre(String nombre) {
                return null;
        }

  public String consultarporApellidos(String apellidos) {
                return null;
        }

  public ArrayList NombrePersona(){
                return null;
        }
  public ArrayList ApellidoPersona(){
                return null;
        }
```

```
public ArrayList IDPersona(){
                return null;
        }
public ArrayList VehiculoClientesID(String id) {
        return null;
}
public ArrayList VehiculoClientesMarca(String id) {
        return null;
}
public ArrayList VehiculoClientesModelo(String id) {
        return null;
}
public ArrayList VehiculoClientesMatricula(String id) {
        return null;
}
public ArrayList VehiculoClientesTipo(String id) {
        return null;
}

public String getCombustibleVehiculo(String id) {
                return null;
        }
public String getnRuedasVehiculo(String id) {
                return null;
        }
public String getProfesionalVehiculo(String id) {
                return null;
        }

public void cerrarMetodos() {
   try {
```

```
            resultado.close();
            resultado2.close();
            sentencia.close();
            sentencia2.close();
            conex.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

}
```

17.4. Cliente.java

```
package textual;

import java.sql.SQLException;
import java.util.ArrayList;
import java.util.Calendar;



/**
 * @author Adriana Armental Tomé
 * @version 06.03.2017
 */
public class Cliente extends Persona
{

            public Cliente() {}


            @Override
            public boolean insertar(String dni, String nombre, String apellidos, String numTelefono,
String correoElectronico, String direccion) {
```

```
                String dia2 = "";
                String mes2 = "";

    Calendar c = Calendar.getInstance();
        int dia = c.get(Calendar.DATE);
        if (dia<10) {
                dia2 = "0"+dia;
        }
        else {
                dia2 = ""+dia;
        }
        int mes = c.get(Calendar.MONTH) + 1;
        if (mes<10) {
                mes2 = "0"+mes;
        }
        else {
                mes2 = ""+mes;
        }
        int annio = c.get(Calendar.YEAR);

        fechaAlta = annio + "-" + mes2 + "-" + dia2;

        try {
            String sql = "INSERT INTO persona (`dni`, `nombre`, `apellidos`, `telf`, `correo`,
`direccion`, `fechaAlta`, `tipoPersona`) VALUES ("

        +"'"+dni+"','"+nombre+"','"+apellidos+"','"+numTelefono+"','"+correoElectronico+"','"+
direccion+"','"+fechaAlta+"','1')";
            sentencia.executeUpdate(sql);
            return true;

            } catch(SQLException e) {
```

```
                e.printStackTrace();
                return false;
            }
        }

        @Override
        public String listar() {

            try {
                resultado = sentencia.executeQuery("SELECT * FROM persona WHERE
tipoPersona = '1'");

                if (resultado.next() == true) //move o cursor
                {
                    soluciones =
"<html><body><table><tr><th>DNI</th><th>NOMBRE</th><th>APELLIDOS</th><th>TELÉFON
O</th><th>CORREO</th><th>DIRECCIÓN</th><th>FECHA ALTA</th></tr>";
                    resultado.beforeFirst(); //volve a poñer de primeiro o cursor
                    while (resultado.next()) {
                        dni = resultado.getString(2); //dni
                        nombre = resultado.getString(3); //nombre
                        apellidos = resultado.getString(4); //apellidos
                        numTelefono = resultado.getString(5); //telefono
                        correoElectronico = resultado.getString(6); //correo
                        direccion = resultado.getString(7); //direccion
                        fechaAlta = resultado.getString(8); //fechaalta

        soluciones+="<tr><td>"+dni+"</td><td>"+nombre+"</td><td>"+apellidos+"</td><td>"
+numTelefono+"</td><td>"+correoElectronico+"</td><td>"+direccion+"</td><td>"+fechaAlta
+"</td></tr>";

                    }
```

```
                    soluciones+="</table></body></html>";
                }
                else {
                    soluciones = "No hay resultados.";
                }

            } catch (SQLException e) {
                e.printStackTrace();
            }
            return soluciones;

        }

        @Override
        public String consultarporDNI(String dni) {

            try {
                resultado = sentencia.executeQuery("SELECT * FROM persona WHERE
UPPER(dni) LIKE UPPER('%"+dni+"%') AND tipoPersona='1'");

                if (resultado.next() == true) //move o cursor
                {
                    soluciones =
"<html><body><table><tr><th>DNI</th><th>NOMBRE</th><th>APELLIDOS</th><th>TELÉFON
O</th><th>CORREO</th><th>DIRECCIÓN</th><th>FECHA ALTA</th></tr>";
                    resultado.beforeFirst(); //volve a poñer de primeiro o cursor
                    while (resultado.next()) {
                        dni = resultado.getString(2); //dni
                        nombre = resultado.getString(3); //nombre
                        apellidos = resultado.getString(4); //apellidos
                        numTelefono = resultado.getString(5); //telefono
                        correoElectronico = resultado.getString(6); //correo
```

```
                                        direccion = resultado.getString(7); //direccion
                                        fechaAlta = resultado.getString(8); //fechaalta

        soluciones+="<tr><td>"+dni+"</td><td>"+nombre+"</td><td>"+apellidos+"</td><td>"
+numTelefono+"</td><td>"+correoElectronico+"</td><td>"+direccion+"</td><td>"+fechaAlta
+"</td></tr>";

                                }
                                soluciones+="</table></body></html>";
                        }
                        else {
                                soluciones = "No hay resultados.";
                        }

                } catch (SQLException e) {
                        e.printStackTrace();
                }
                return soluciones;
        }

        @Override
        public String consultarporNombre(String nombre) {

                try {
                        resultado = sentencia.executeQuery("SELECT * FROM persona WHERE
UPPER(nombre) LIKE UPPER('%"+nombre+"%') AND tipoPersona='1'");

                        if (resultado.next() == true) //move o cursor
                        {
                                soluciones =
"<html><body><table><tr><th>DNI</th><th>NOMBRE</th><th>APELLIDOS</th><th>TELÉFON
O</th><th>CORREO</th><th>DIRECCIÓN</th><th>FECHA ALTA</th></tr>";
```

```
                    resultado.beforeFirst(); //volve a poñer de primeiro o cursor
                    while (resultado.next()) {
                        dni = resultado.getString(2); //dni
                        nombre = resultado.getString(3); //nombre
                        apellidos = resultado.getString(4); //apellidos
                        numTelefono = resultado.getString(5); //telefono
                        correoElectronico = resultado.getString(6); //correo
                        direccion = resultado.getString(7); //direccion
                        fechaAlta = resultado.getString(8); //fechaalta

    soluciones+="<tr><td>"+dni+"</td><td>"+nombre+"</td><td>"+apellidos+"</td><td>"
+numTelefono+"</td><td>"+correoElectronico+"</td><td>"+direccion+"</td><td>"+fechaAlta
+"</td></tr>";

                    }
                    soluciones+="</table></body></html>";
                }
                else {
                    soluciones = "No hay resultados.";
                }

            } catch (SQLException e) {
                e.printStackTrace();
            }
            return soluciones;
        }

        @Override
        public String consultarporApellidos(String apellidos) {

            try {
```

```
                    resultado = sentencia.executeQuery("SELECT * FROM persona WHERE
UPPER(apellidos) LIKE UPPER('%"+apellidos+"%') AND tipoPersona='1'");


                    if (resultado.next() == true) //move o cursor
                    {
                        soluciones =
"<html><body><table><tr><th>DNI</th><th>NOMBRE</th><th>APELLIDOS</th><th>TELÉFON
O</th><th>CORREO</th><th>DIRECCIÓN</th><th>FECHA ALTA</th></tr>";
                        resultado.beforeFirst(); //volve a poñer de primeiro o cursor
                        while (resultado.next()) {
                            dni = resultado.getString(2); //dni
                            nombre = resultado.getString(3); //nombre
                            apellidos = resultado.getString(4); //apellidos
                            numTelefono = resultado.getString(5); //telefono
                            correoElectronico = resultado.getString(6); //correo
                            direccion = resultado.getString(7); //direccion
                            fechaAlta = resultado.getString(8); //fechaalta


        soluciones+="<tr><td>"+dni+"</td><td>"+nombre+"</td><td>"+apellidos+"</td><td>"
+numTelefono+"</td><td>"+correoElectronico+"</td><td>"+direccion+"</td><td>"+fechaAlta
+"</td></tr>";


                        }
                        soluciones+="</table></body></html>";
                    }
                    else {
                        soluciones = "No hay resultados.";
                    }


                } catch (SQLException e) {
                    e.printStackTrace();
                }
```

```
            return soluciones;
        }

        @Override
        public ArrayList NombrePersona() {
            nombreapellidos = new ArrayList();
            try {
                resultado = sentencia.executeQuery("SELECT * FROM persona WHERE
tipoPersona = '1'");

                if (resultado.next() == true) //move o cursor
                {
                    resultado.beforeFirst(); //volve a poñer de primeiro o cursor
                    while (resultado.next()) {
                        nombre = resultado.getString(3); //nombre

                        nombreapellidos.add(nombre);
                    }
                }
                else {
                    nombreapellidos.add("No hay resultados.");
                }

            } catch (SQLException e) {
                e.printStackTrace();
            }
            return nombreapellidos;
        }

        @Override
        public ArrayList ApellidoPersona() {
```

```
                nombreapellidos = new ArrayList();
                try {
                        resultado = sentencia.executeQuery("SELECT * FROM persona WHERE
tipoPersona = '1'");

                        if (resultado.next() == true) //move o cursor
                        {
                                resultado.beforeFirst(); //volve a poñer de primeiro o cursor
                                while (resultado.next()) {
                                        apellidos = resultado.getString(4); //apellidos

                                        nombreapellidos.add(apellidos);
                                }
                        }
                        else {
                                nombreapellidos.add("No hay resultados.");
                        }

                } catch (SQLException e) {
                        e.printStackTrace();
                }
                return nombreapellidos;
        }

        @Override
        public ArrayList IDPersona() {
                nombreapellidos = new ArrayList();
                try {
                        resultado = sentencia.executeQuery("SELECT * FROM persona WHERE
tipoPersona = '1'");

                        if (resultado.next() == true) //move o cursor
```

```
                {
                        resultado.beforeFirst(); //volve a poñer de primeiro o cursor
                        while (resultado.next()) {
                                id = resultado.getString(1); //id

                                nombreapellidos.add(id);
                        }
                }
                else {
                        nombreapellidos.add("No hay resultados.");
                }

        } catch (SQLException e) {
                e.printStackTrace();
        }
        return nombreapellidos;
    }

    @Override
    public ArrayList VehiculoClientesID(String id) {
        vehiculos = new ArrayList();
        try {
            resultado = sentencia.executeQuery("SELECT * FROM vehiculo WHERE
idCliente = '"+id+"'");

                if (resultado.next() == true) //move o cursor
                {
                        resultado.beforeFirst(); //volve a poñer de primeiro o cursor
                        while (resultado.next()) {
                                idvehiculo = resultado.getString(1); //idvehiculo
                                vehiculos.add(idvehiculo);
```

```
                        }
                    }
                    else {
                        vehiculos.add("No hay resultados.");
                    }

                } catch (SQLException e) {
                    e.printStackTrace();
                }
                return vehiculos;
            }

            @Override
            public ArrayList VehiculoClientesMarca(String id) {
                vehiculos = new ArrayList();
                try {
                    resultado = sentencia.executeQuery("SELECT * FROM vehiculo WHERE
idCliente = '"+id+"'");

                    if (resultado.next() == true) //move o cursor
                    {
                        resultado.beforeFirst(); //volve a poñer de primeiro o cursor
                        while (resultado.next()) {
                            marca = resultado.getString(2); //marca
                            vehiculos.add(marca);
                        }
                    }
                    else {
                        vehiculos.add("No hay resultados.");
                    }
```

```
            } catch (SQLException e) {
                e.printStackTrace();
            }
            return vehiculos;
        }

        @Override
        public ArrayList VehiculoClientesModelo(String id) {
            vehiculos = new ArrayList();
            try {
                resultado = sentencia.executeQuery("SELECT * FROM vehiculo WHERE
idCliente = '"+id+"'");

                if (resultado.next() == true) //move o cursor
                {
                    resultado.beforeFirst(); //volve a poñer de primeiro o cursor
                    while (resultado.next()) {
                        modelo = resultado.getString(3); //modelo
                        vehiculos.add(modelo);
                    }
                }
                else {
                    vehiculos.add("No hay resultados.");
                }

            } catch (SQLException e) {
                e.printStackTrace();
            }
            return vehiculos;
        }
```

```
	@Override
	public ArrayList VehiculoClientesMatricula(String id) {
		vehiculos = new ArrayList();
		try {
			resultado = sentencia.executeQuery("SELECT * FROM vehiculo WHERE idCliente = '"+id+"'");

			if (resultado.next() == true) //move o cursor
			{
				resultado.beforeFirst(); //volve a poñer de primeiro o cursor
				while (resultado.next()) {
					matricula = resultado.getString(4); //matricula
					vehiculos.add(matricula);
				}
			}
			else {
				vehiculos.add("No hay resultados.");
			}

		} catch (SQLException e) {
			e.printStackTrace();
		}
		return vehiculos;
	}

	@Override
	public ArrayList VehiculoClientesTipo(String id) {
		vehiculos = new ArrayList();
		try {
			resultado = sentencia.executeQuery("SELECT * FROM vehiculo WHERE idCliente = '"+id+"'");
```

```
                        if (resultado.next() == true) //move o cursor
                        {
                                resultado.beforeFirst(); //volve a poñer de primeiro o cursor
                                while (resultado.next()) {
                                        idtipovehiculo = resultado.getString(18); //tipo
                                        resultado2 = sentencia2.executeQuery("SELECT *
FROM tipo_vehiculo WHERE id = '"+idtipovehiculo+"'");

                                        if (resultado2.next() == true) //move o cursor
                                        {
                                                resultado2.beforeFirst(); //volve a poñer de
primeiro o cursor
                                                while (resultado2.next()) {
                                                        tipovehiculo = resultado2.getString(2);
//tipovehiculo
                                                }
                                        }
                                        vehiculos.add(tipovehiculo);
                                }
                        }
                        else {
                                vehiculos.add("No hay resultados.");
                        }

                } catch (SQLException e) {
                        e.printStackTrace();
                }
                return vehiculos;
        }

        @Override
        public String getCombustibleVehiculo(String id) {
```

```
        try {
                        resultado = sentencia.executeQuery("SELECT * FROM vehiculo WHERE
id='"+id+"'");
                        while (resultado.next()) {
                           soluciones = resultado.getString(9);
                        }

                } catch (SQLException e) {
                        e.printStackTrace();
                }
                return soluciones;

        }
   @Override
        public String getnRuedasVehiculo(String id) {
        try {
                        resultado = sentencia.executeQuery("SELECT * FROM vehiculo WHERE
id='"+id+"'");
                        while (resultado.next()) {
                           soluciones = resultado.getString(16);
                        }

                } catch (SQLException e) {
                        e.printStackTrace();
                }
                return soluciones;
        }
   @Override
        public String getProfesionalVehiculo(String id) {
        try {
                        resultado = sentencia.executeQuery("SELECT * FROM vehiculo WHERE
id='"+id+"'");
```

```
                        while (resultado.next()) {
                                tipovehiculo = resultado.getString(18);
                                resultado2 = sentencia2.executeQuery("SELECT * FROM
tipo_vehiculo WHERE id='"+tipovehiculo+"'");
                                while (resultado2.next())
                                {
                                        soluciones = resultado2.getString(2);
                                }
                        }

                } catch (SQLException e) {
                        e.printStackTrace();
                }
                return soluciones;
        }
}
```

17.5. Vehiculo.java

```
package textual;

import java.sql.*;
import java.util.ArrayList;

/**
 * Vehiculo es una clase abstracta para los vehiculos.
 * Ofrece las caracteristicas comunes a todos los vehiculos
 * como combustible, almacenamiento, nº de puertas, etc.
 *
 * @author Adriana Armental Tomé
 * @version 06.03.2017
 */
public abstract class Vehiculo
```

```
{
   // instance variables - replace the example below with your own
          protected String id;
   protected String marca;
   protected String modelo;
   protected String matricula;
   protected String idcliente;
   protected String tipovehiculo;
   protected String nombrecliente, nombretipovehiculo, nombrevehiculo;
   protected String nPuertas;
   protected String capMotor;
   protected String combustible;
   protected String capAlmacenamiento;
   protected String nPasajeros;
   protected boolean airbags;
   protected String maxVelocidad;
   protected boolean ABS;
   protected boolean GPS;
   protected boolean climatizacion;
   protected boolean descapotable;
   protected String nRuedas;
   protected String fechaAlta;
   protected String motivoVisita;
   protected String lixo;
   protected String soluciones;

   protected String sairbags, sABS, sGPS, sclimatizacion, sdescapotable;
   //estas son as que se usan para sacar no formulario de registro de coche

   static ConexionBD con;
   static Connection conex;
```

```
    static Statement sentencia, sentencia2, sentencia3;
    static ResultSet resultado, resultado2, resultado3;
            ArrayList marcavehiculo;

    public Vehiculo() {
        con = new ConexionBD();
        conex = con.getCon();
        try {
                                sentencia = conex.createStatement();
                                sentencia2 = conex.createStatement();
                                sentencia3 = conex.createStatement();
                        } catch (SQLException e) {
                                e.printStackTrace();
                        }
    }



    public String getMarca(String id) {
            try {
                                resultado = sentencia.executeQuery("SELECT * FROM vehiculo WHERE
id='"+id+"'");
                                while (resultado.next()) {
                                        marca = resultado.getString(2);
                                }


                        } catch (SQLException e) {
                                e.printStackTrace();
                        }
                        return marca;
    }
```

```
   public String getModelo(String id) {
         try {
                  resultado = sentencia.executeQuery("SELECT * FROM vehiculo WHERE
id='"+id+"'");
                  while (resultado.next()) {
                        modelo = resultado.getString(3);
                  }

            } catch (SQLException e) {
                  e.printStackTrace();
            }
            return modelo;
   }

   public String getMatricula(String id) {
         try {
                  resultado = sentencia.executeQuery("SELECT * FROM vehiculo WHERE
id='"+id+"'");
                  while (resultado.next()) {
                        matricula = resultado.getString(4);
                  }

            } catch (SQLException e) {
                  e.printStackTrace();
            }
            return matricula;
   }

   public String getIdcliente(String id) {
         try {
```

```
                    resultado = sentencia.executeQuery("SELECT * FROM vehiculo WHERE
id='"+id+"'");
                    while (resultado.next()) {
                        idcliente = resultado.getString(20);
                    }

                } catch (SQLException e) {
                    e.printStackTrace();
                }
                return idcliente;
    }

    public String getNombrecliente(String id) {
        try {
                    resultado = sentencia.executeQuery("SELECT * FROM vehiculo WHERE
id='"+id+"'");
                    while (resultado.next()) {
                        idcliente = resultado.getString(20);
                        resultado2 = sentencia2.executeQuery("SELECT * FROM
persona WHERE id='"+idcliente+"'");
                        while (resultado2.next())
                        {
                            nombrecliente = resultado2.getString(3) + " " +
resultado2.getString(4);
                        }
                    }

                } catch (SQLException e) {
                    e.printStackTrace();
                }
                return nombrecliente;
        }
```

```
        public String getNombrevehiculo(String id) {
                try {
                        resultado = sentencia.executeQuery("SELECT * FROM vehiculo WHERE
id='"+id+"'");
                        while (resultado.next()) {
                                nombrevehiculo = resultado.getString(2) + " " +
resultado.getString(3) + " " + resultado.getString(4);
                        }
                } catch (SQLException e) {
                        e.printStackTrace();
                }
                return nombrevehiculo;
        }

   public String getNombretipovehiculo(String id) {
        try {
                        resultado = sentencia.executeQuery("SELECT * FROM vehiculo WHERE
id='"+id+"'");
                        while (resultado.next()) {
                                tipovehiculo = resultado.getString(20);
                                resultado3 = sentencia3.executeQuery("SELECT * FROM
tipo_vehiculo WHERE id='"+tipovehiculo+"'");
                                while (resultado3.next())
                                {
                                        nombretipovehiculo = resultado3.getString(2);
                                }
                        }

                } catch (SQLException e) {
                        e.printStackTrace();
                }
                return nombretipovehiculo;
```

```
        }

    public String getnPuertas(String id) {
        try {
                    resultado = sentencia.executeQuery("SELECT * FROM vehiculo WHERE
id='"+id+"'");
                    while (resultado.next()) {
                       nPuertas = resultado.getString(5);
                    }

                } catch (SQLException e) {
                    e.printStackTrace();
                }
                return nPuertas;
    }


        public String getCapMotor(String id) {
        try {
                    resultado = sentencia.executeQuery("SELECT * FROM vehiculo WHERE
id='"+id+"'");
                    while (resultado.next()) {
                       capMotor = resultado.getString(8);
                    }

                } catch (SQLException e) {
                    e.printStackTrace();
                }
                return capMotor;
    }

        public String getCombustible(String id) {
```

```
        try {
                resultado = sentencia.executeQuery("SELECT * FROM vehiculo WHERE
id='"+id+"'");
                while (resultado.next()) {
                   combustible = resultado.getString(9);
                }

            } catch (SQLException e) {
                e.printStackTrace();
            }
            return combustible;
    }

        public String getCapAlmacenamiento(String id) {
        try {
                resultado = sentencia.executeQuery("SELECT * FROM vehiculo WHERE
id='"+id+"'");
                while (resultado.next()) {
                   capAlmacenamiento = resultado.getString(10);
                }

            } catch (SQLException e) {
                e.printStackTrace();
            }
            return capAlmacenamiento;
    }

        public String getnPasajeros(String id) {
        try {
                resultado = sentencia.executeQuery("SELECT * FROM vehiculo WHERE
id='"+id+"'");
                while (resultado.next()) {
```

```
                    nPasajeros = resultado.getString(11);
                }

            } catch (SQLException e) {
                e.printStackTrace();
            }
            return nPasajeros;
    }

    public String getfechaAlta(String id) {
    try {
                resultado = sentencia.executeQuery("SELECT * FROM vehiculo WHERE
id='"+id+"'");
                while (resultado.next()) {
                    fechaAlta = resultado.getString(19);
                }

            } catch (SQLException e) {
                e.printStackTrace();
            }
            return fechaAlta;
    }

    public boolean isAirbags(String id) {
    try {
                resultado = sentencia.executeQuery("SELECT * FROM vehiculo WHERE
id='"+id+"'");
                while (resultado.next()) {
                    airbags = resultado.getBoolean(6);
                }

            } catch (SQLException e) {
```

```
                e.printStackTrace();
            }
            return airbags;
    }

    public String getMaxVelocidad(String id) {
    try {
                resultado = sentencia.executeQuery("SELECT * FROM vehiculo WHERE
id='"+id+"'");
                while (resultado.next()) {
                   maxVelocidad = resultado.getString(7);
                }

            } catch (SQLException e) {
                e.printStackTrace();
            }
            return maxVelocidad;
    }

    public boolean isABS(String id) {
    try {
                resultado = sentencia.executeQuery("SELECT * FROM vehiculo WHERE
id='"+id+"'");
                while (resultado.next()) {
                        ABS = resultado.getBoolean(12);
                }

            } catch (SQLException e) {
                e.printStackTrace();
            }
            return ABS;
    }
```

```
	public boolean isGPS(String id) {
	try {
			resultado = sentencia.executeQuery("SELECT * FROM vehiculo WHERE
id='"+id+"'");
			while (resultado.next()) {
				GPS = resultado.getBoolean(13);
			}

		} catch (SQLException e) {
			e.printStackTrace();
		}
		return GPS;
  }

	public boolean isClimatizacion(String id) {
	try {
			resultado = sentencia.executeQuery("SELECT * FROM vehiculo WHERE
id='"+id+"'");
			while (resultado.next()) {
				climatizacion = resultado.getBoolean(14);
			}

		} catch (SQLException e) {
			e.printStackTrace();
		}
		return climatizacion;
  }

	public boolean isDescapotable(String id) {
	try {
```

```
                        resultado = sentencia.executeQuery("SELECT * FROM vehiculo WHERE
id='"+id+"'");
                        while (resultado.next()) {
                                descapotable = resultado.getBoolean(15);
                        }

                } catch (SQLException e) {
                        e.printStackTrace();
                }
                return descapotable;
  }


        public String getnRuedas(String id) {
        try {
                        resultado = sentencia.executeQuery("SELECT * FROM vehiculo WHERE
id='"+id+"'");
                        while (resultado.next()) {
                           nRuedas = resultado.getString(16);
                        }

                } catch (SQLException e) {
                        e.printStackTrace();
                }
                return nRuedas;
  }

        public String getMotivoVisita(String id) {
        try {
                        resultado = sentencia.executeQuery("SELECT * FROM vehiculo WHERE
id='"+id+"'");
                        while (resultado.next()) {
```

```
                motivoVisita = resultado.getString(17);
            }

        } catch (SQLException e) {
            e.printStackTrace();
        }
        return motivoVisita;
    }

    public boolean insertar(String marca, String modelo, String matricula, String
combustible, String nPuertas, String nRuedas, String maxVelocidad, String capMotor, String
capAlmacenamiento, String nPasajeros, String ABS, String GPS, String airbags, String
climatizacion, String descapotable, String idcliente, String motivoVisita) {
        return false;
    }

    public boolean editar(String id, String marca, String modelo, String matricula, String
combustible, String nPuertas, String nRuedas, String maxVelocidad, String capMotor, String
capAlmacenamiento, String nPasajeros, String ABS, String GPS, String airbags, String
climatizacion, String descapotable, String idcliente, String motivoVisita) {

        try {
            sentencia.executeUpdate("UPDATE vehiculo SET marca='"+marca+"',
modelo='"+modelo+"', matricula='"+matricula+"', nPuertas='"+nPuertas+"',
airbags='"+airbags+"', maxVelocidad='"+maxVelocidad+"', capMotor='"+capMotor+"',
combustible='"+combustible+"', capAlmacenamiento='"+capAlmacenamiento+"',
nPasajeros='"+nPasajeros+"', ABS='"+ABS+"', GPS='"+GPS+"', climatizacion='"+climatizacion+"',
descapotable='"+descapotable+"', nRuedas='"+nRuedas+"', motivoVisita='"+motivoVisita+"',
idCliente='"+idcliente+"' WHERE id = '"+id+"'");
            return true;
        } catch(SQLException e) {
            e.printStackTrace();
            return false;
        }
```

```
        }

        public boolean borrar(String id) {

            try {
            sentencia.executeUpdate("DELETE FROM vehiculo WHERE id='"+id+"'");
            return true;

            } catch(SQLException e) {
                    e.printStackTrace();
                    return false;
            }
        }

    public String listar() {
                return null;
        }

    public String consultarporMatricula(String matricula) {
                return null;
        }

    public String consultarporMarca(String marca) {
                return null;
        }

    public String consultarporModelo(String modelo) {
                return null;
        }

    public String getUltimoRegitro() {
```

```
                return null;
    }

    public ArrayList MarcaVehiculo(){
                return null;
        }
    public ArrayList ModeloVehiculo(){
                return null;
        }
    public ArrayList MatriculaVehiculo(){
                return null;
        }
    public ArrayList IDVehiculo(){
                return null;
        }

    public void cerrarMetodos() {
        try {
                resultado.close();
                resultado2.close();
                resultado3.close();
                        sentencia.close();
                        sentencia2.close();
                        sentencia3.close();
                conex.close();
                } catch (SQLException e) {
                        e.printStackTrace();
                }
    }
}
```

17.6. Turismo

```
package textual;


import java.sql.SQLException;

import java.util.ArrayList;

import java.util.Calendar;


/**

 * @author Adriana Armental Tomé

 * @version 06.03.2017

 */

public class Turismo extends Coche

{


	public Turismo() {}


	@Override

	public boolean insertar(String marca, String modelo, String matricula, String combustible, String nPuertas, String nRuedas, String maxVelocidad, String capMotor, String capAlmacenamiento, String nPasajeros, String ABS, String GPS, String airbags, String climatizacion, String descapotable, String idcliente, String motivoVisita) {

		String dia2 = "";

		String mes2 = "";


    Calendar c = Calendar.getInstance();

	int dia = c.get(Calendar.DATE);

	if (dia<10) {

		dia2 = "0"+dia;

	}

	else {

		dia2 = ""+dia;

	}
```

```
int mes = c.get(Calendar.MONTH) + 1;
if (mes<10) {
        mes2 = "0"+mes;
}
else {
        mes2 = ""+mes;
}
int annio = c.get(Calendar.YEAR);

fechaAlta = annio + "-" + mes2 + "-" + dia2;

try {
    String sql = "INSERT INTO vehiculo (`marca`, `modelo`, `matricula`, `nPuertas`,
`airbags`, `maxVelocidad`, `capMotor`, `combustible`, `capAlmacenamiento`, `nPasajeros`,
`ABS`, `GPS`, `climatizacion`, `descapotable`, `nRuedas`, `motivoVisita`, `tipoVehiculo`,
`fechaAlta`, `idCliente`) VALUES ("

    +"'"+marca+"','"+modelo+"','"+matricula+"','"+nPuertas+"','"+airbags+"','"+maxVelocid
ad+"','"+capMotor+"','"+combustible+"','"+capAlmacenamiento+"','"+nPasajeros+"','"+ABS+"','
"+GPS+"','"+climatizacion+"','"+descapotable+"','"+nRuedas+"','"+motivoVisita+"','1','"+fechaAl
ta+"','"+idcliente+"')";
        sentencia.executeUpdate(sql);
        return true;

        } catch(SQLException e) {
                e.printStackTrace();
                return false;

        }
}

@Override
public String listar() {
```

```
                try {
                        resultado = sentencia.executeQuery("SELECT * FROM vehiculo WHERE
tipoVehiculo='1'");

                        if (resultado.next() == true) //move o cursor
                        {
                                soluciones =
"<html><body><table><tr><th>MATRÍCULA</th><th>MARCA</th><th>MODELO</th><th>Nº
PUERTAS</th><th>AIRBAGS</th><th>VELOCIDAD
MAX.</th><th>CAP.MOTOR</th><th>COMBUSTIBLE</th><th>CAP.ALMACENAMIENTO</th><t
h>Nº PASAJEROS</th><th>ABS</th><th>GPS</th><th>CLIMATIZACION</th><th>Nº
RUEDAS</th><th>DESCAPOTABLE</th><th>MOT.VISITA</th><th>FECHA
ALTA</th><th>DUEÑO</th></tr>";
                                resultado.beforeFirst(); //volve a poñer de primeiro o cursor
                                while (resultado.next()) {
                                        marca = resultado.getString(2);
                                        modelo = resultado.getString(3);
                                        matricula = resultado.getString(4);
                                   nPuertas = resultado.getString(5);
                                   airbags = resultado.getBoolean(6);
                                   if(airbags == true) {
                                        sairbags = "si";
                                   }
                                   else {
                                        sairbags = "no";
                                   }
                                   maxVelocidad = resultado.getString(7);
                                   capMotor = resultado.getString(8);
                                   combustible = resultado.getString(9);
                                   capAlmacenamiento = resultado.getString(10);
                                   nPasajeros = resultado.getString(11);
                                   ABS = resultado.getBoolean(12);
                                   if(ABS == true) {
                                        sABS = "si";
```

```
}
else {
	sABS = "no";
}
GPS = resultado.getBoolean(13);
if(GPS == true) {
	sGPS = "si";
}
else {
	sGPS = "no";
}
climatizacion = resultado.getBoolean(14);
if(climatizacion == true) {
	sclimatizacion = "si";
}
else {
	sclimatizacion = "no";
}
descapotable = resultado.getBoolean(15);
if(descapotable == true) {
	sdescapotable = "si";
}
else {
	sdescapotable = "no";
}
nRuedas = resultado.getString(16);
motivoVisita = resultado.getString(17);
	tipovehiculo = resultado.getString(18);
fechaAlta = resultado.getString(19);
	idcliente = resultado.getString(20);
```

```
                                        resultado2 = sentencia2.executeQuery("SELECT
* FROM persona WHERE id='"+idcliente+"'");
                                        while (resultado2.next())
                                        {
                                                nombrecliente =
resultado2.getString(3) + " " + resultado2.getString(4);
                                        }


                                        resultado3 = sentencia3.executeQuery("SELECT
* FROM tipo_vehiculo WHERE id='"+tipovehiculo+"'");
                                        while (resultado3.next())
                                        {
                                                nombretipovehiculo =
resultado3.getString(2);
                                        }



        soluciones+="<tr><td>"+matricula+"</td><td>"+marca+"</td><td>"+modelo+"</td><t
d>"+nPuertas+"</td><td>"+sairbags+"</td><td>"+maxVelocidad+"</td><td>"+capMotor+"</t
d><td>"+combustible+"</td><td>"+capAlmacenamiento+"</td><td>"+nPasajeros+"</td><td>
"+sABS+"</td><td>"+sGPS+"</td><td>"+sclimatizacion+"</td><td>"+nRuedas+"</td><td>"+sd
escapotable+"</td><td>"+motivoVisita+"</td><td>"+fechaAlta+"</td><td>"+nombrecliente+"
</td></tr>";


                                }
                                soluciones+="</table></body></html>";
                        }
                        else {
                                soluciones = "No hay resultados.";
                        }


                } catch (SQLException e) {
                        e.printStackTrace();
                }
                return soluciones;
```

```
}

    @Override
    public String consultarporMatricula(String matricula) {

        try {
            resultado = sentencia.executeQuery("SELECT * FROM vehiculo WHERE
UPPER(matricula) LIKE UPPER('%"+matricula+"%') AND tipoVehiculo='1'");

            if (resultado.next() == true) //move o cursor
            {
                soluciones =
"<html><body><table><tr><th>MATRÍCULA</th><th>MARCA</th><th>MODELO</th><th>Nº
PUERTAS</th><th>AIRBAGS</th><th>VELOCIDAD
MAX.</th><th>CAP.MOTOR</th><th>COMBUSTIBLE</th><th>CAP.ALMACENAMIENTO</th><t
h>Nº PASAJEROS</th><th>ABS</th><th>GPS</th><th>CLIMATIZACION</th><th>Nº
RUEDAS</th><th>DESCAPOTABLE</th><th>MOT.VISITA</th><th>FECHA
ALTA</th><th>DUEÑO</th></tr>";
                resultado.beforeFirst(); //volve a poñer de primeiro o cursor
                while (resultado.next()) {
                    marca = resultado.getString(2);
                    modelo = resultado.getString(3);
                    matricula = resultado.getString(4);
                  nPuertas = resultado.getString(5);
                  airbags = resultado.getBoolean(6);
                  if(airbags == true) {
                    sairbags = "si";
                  }
                  else {
                    sairbags = "no";
                  }
                  maxVelocidad = resultado.getString(7);
                  capMotor = resultado.getString(8);
                  combustible = resultado.getString(9);
```

```
capAlmacenamiento = resultado.getString(10);
nPasajeros = resultado.getString(11);
ABS = resultado.getBoolean(12);
if(ABS == true) {
        sABS = "si";
}
else {
        sABS = "no";
}
GPS = resultado.getBoolean(13);
if(GPS == true) {
        sGPS = "si";
}
else {
        sGPS = "no";
}
climatizacion = resultado.getBoolean(14);
if(climatizacion == true) {
        sclimatizacion = "si";
}
else {
        sclimatizacion = "no";
}
descapotable = resultado.getBoolean(15);
if(descapotable == true) {
        sdescapotable = "si";
}
else {
        sdescapotable = "no";
}
nRuedas = resultado.getString(16);
```

```
                    motivoVisita = resultado.getString(17);
                        tipovehiculo = resultado.getString(18);
                    fechaAlta = resultado.getString(19);
                        idcliente = resultado.getString(20);

                                    resultado2 = sentencia2.executeQuery("SELECT
* FROM persona WHERE id='"+idcliente+"'");
                                    while (resultado2.next())
                                    {
                                        nombrecliente =
resultado2.getString(3) + " " + resultado2.getString(4);
                                    }

                                    resultado3 = sentencia3.executeQuery("SELECT
* FROM tipo_vehiculo WHERE id='"+tipovehiculo+"'");
                                    while (resultado3.next())
                                    {
                                        nombretipovehiculo =
resultado3.getString(2);
                                    }

        soluciones+="<tr><td>"+matricula+"</td><td>"+marca+"</td><td>"+modelo+"</td><t
d>"+nPuertas+"</td><td>"+sairbags+"</td><td>"+maxVelocidad+"</td><td>"+capMotor+"</t
d><td>"+combustible+"</td><td>"+capAlmacenamiento+"</td><td>"+nPasajeros+"</td><td>
"+sABS+"</td><td>"+sGPS+"</td><td>"+sclimatizacion+"</td><td>"+nRuedas+"</td><td>"+sd
escapotable+"</td><td>"+motivoVisita+"</td><td>"+fechaAlta+"</td><td>"+nombrecliente+"
</td></tr>";

                    }
                    soluciones+="</table></body></html>";
                }
                else {
                    soluciones = "No hay resultados.";
                }
```

```
                } catch (SQLException e) {
                        e.printStackTrace();
                }
                return soluciones;
  }

        @Override
        public String consultarporMarca(String marca) {

                try {
                        resultado = sentencia.executeQuery("SELECT * FROM vehiculo WHERE
UPPER(marca) LIKE UPPER('%"+marca+"%') AND tipoVehiculo='1'");

                        if (resultado.next() == true) //move o cursor
                        {
                                soluciones =
"<html><body><table><tr><th>MATRÍCULA</th><th>MARCA</th><th>MODELO</th><th>Nº
PUERTAS</th><th>AIRBAGS</th><th>VELOCIDAD
MAX.</th><th>CAP.MOTOR</th><th>COMBUSTIBLE</th><th>CAP.ALMACENAMIENTO</th><t
h>Nº PASAJEROS</th><th>ABS</th><th>GPS</th><th>CLIMATIZACION</th><th>Nº
RUEDAS</th><th>DESCAPOTABLE</th><th>MOT.VISITA</th><th>FECHA
ALTA</th><th>DUEÑO</th></tr>";
                                resultado.beforeFirst(); //volve a poñer de primeiro o cursor
                                while (resultado.next()) {
                                        marca = resultado.getString(2);
                                        modelo = resultado.getString(3);
                                        matricula = resultado.getString(4);
                                   nPuertas = resultado.getString(5);
                                   airbags = resultado.getBoolean(6);
                                   if(airbags == true) {
                                        sairbags = "si";
                                   }
                                   else {
```

```
        sairbags = "no";
}
maxVelocidad = resultado.getString(7);
capMotor = resultado.getString(8);
combustible = resultado.getString(9);
capAlmacenamiento = resultado.getString(10);
nPasajeros = resultado.getString(11);
ABS = resultado.getBoolean(12);
if(ABS == true) {
        sABS = "si";
}
else {
        sABS = "no";
}
GPS = resultado.getBoolean(13);
if(GPS == true) {
        sGPS = "si";
}
else {
        sGPS = "no";
}
climatizacion = resultado.getBoolean(14);
if(climatizacion == true) {
        sclimatizacion = "si";
}
else {
        sclimatizacion = "no";
}
descapotable = resultado.getBoolean(15);
if(descapotable == true) {
        sdescapotable = "si";
```

```
                                        }
                                        else {
                                                sdescapotable = "no";
                                        }
                                        nRuedas = resultado.getString(16);
                                        motivoVisita = resultado.getString(17);
                                                tipovehiculo = resultado.getString(18);
                                        fechaAlta = resultado.getString(19);
                                                idcliente = resultado.getString(20);


                                                        resultado2 = sentencia2.executeQuery("SELECT
* FROM persona WHERE id='"+idcliente+"'");
                                                        while (resultado2.next())
                                                        {
                                                                nombrecliente =
resultado2.getString(3) + " " + resultado2.getString(4);
                                                        }


                                                        resultado3 = sentencia3.executeQuery("SELECT
* FROM tipo_vehiculo WHERE id='"+tipovehiculo+"'");
                                                        while (resultado3.next())
                                                        {
                                                                nombretipovehiculo =
resultado3.getString(2);
                                                        }



        soluciones+="<tr><td>"+matricula+"</td><td>"+marca+"</td><td>"+modelo+"</td><t
d>"+nPuertas+"</td><td>"+sairbags+"</td><td>"+maxVelocidad+"</td><td>"+capMotor+"</t
d><td>"+combustible+"</td><td>"+capAlmacenamiento+"</td><td>"+nPasajeros+"</td><td>
"+sABS+"</td><td>"+sGPS+"</td><td>"+sclimatizacion+"</td><td>"+nRuedas+"</td><td>"+sd
escapotable+"</td><td>"+motivoVisita+"</td><td>"+fechaAlta+"</td><td>"+nombrecliente+"
</td></tr>";


                                }
```

```
                    soluciones+="</table></body></html>";
                }
                else {
                    soluciones = "No hay resultados.";
                }

            } catch (SQLException e) {
                e.printStackTrace();
            }
            return soluciones;
    }

    @Override
    public String consultarporModelo(String modelo) {

            try {
                resultado = sentencia.executeQuery("SELECT * FROM vehiculo WHERE
UPPER(modelo) LIKE UPPER('%"+modelo+"%') AND tipoVehiculo='1'");

                if (resultado.next() == true) //move o cursor
                {
                    soluciones =
"<html><body><table><tr><th>MATRÍCULA</th><th>MARCA</th><th>MODELO</th><th>Nº
PUERTAS</th><th>AIRBAGS</th><th>VELOCIDAD
MAX.</th><th>CAP.MOTOR</th><th>COMBUSTIBLE</th><th>CAP.ALMACENAMIENTO</th><t
h>Nº PASAJEROS</th><th>ABS</th><th>GPS</th><th>CLIMATIZACION</th><th>Nº
RUEDAS</th><th>DESCAPOTABLE</th><th>MOT.VISITA</th><th>FECHA
ALTA</th><th>DUEÑO</th></tr>";
                    resultado.beforeFirst(); //volve a poñer de primeiro o cursor
                    while (resultado.next()) {
                        marca = resultado.getString(2);
                        modelo = resultado.getString(3);
                        matricula = resultado.getString(4);
                      nPuertas = resultado.getString(5);
```

```
airbags = resultado.getBoolean(6);
if(airbags == true) {
	sairbags = "si";
}
else {
	sairbags = "no";
}
maxVelocidad = resultado.getString(7);
capMotor = resultado.getString(8);
combustible = resultado.getString(9);
capAlmacenamiento = resultado.getString(10);
nPasajeros = resultado.getString(11);
ABS = resultado.getBoolean(12);
if(ABS == true) {
	sABS = "si";
}
else {
	sABS = "no";
}
GPS = resultado.getBoolean(13);
if(GPS == true) {
	sGPS = "si";
}
else {
	sGPS = "no";
}
climatizacion = resultado.getBoolean(14);
if(climatizacion == true) {
	sclimatizacion = "si";
}
else {
```

```
                    sclimatizacion = "no";
                }
                descapotable = resultado.getBoolean(15);
                if(descapotable == true) {
                    sdescapotable = "si";
                }
                else {
                    sdescapotable = "no";
                }
                nRuedas = resultado.getString(16);
                motivoVisita = resultado.getString(17);
                    tipovehiculo = resultado.getString(18);
                fechaAlta = resultado.getString(19);
                    idcliente = resultado.getString(20);


                                resultado2 = sentencia2.executeQuery("SELECT
* FROM persona WHERE id='"+idcliente+"'");
                                while (resultado2.next())
                                {
                                    nombrecliente =
resultado2.getString(3) + " " + resultado2.getString(4);
                                }


                                resultado3 = sentencia3.executeQuery("SELECT
* FROM tipo_vehiculo WHERE id='"+tipovehiculo+"'");
                                while (resultado3.next())
                                {
                                    nombretipovehiculo =
resultado3.getString(2);
                                }


        soluciones+="<tr><td>"+matricula+"</td><td>"+marca+"</td><td>"+modelo+"</td><t
```

```
d>"+nPuertas+"</td><td>"+sairbags+"</td><td>"+maxVelocidad+"</td><td>"+capMotor+"</t
d><td>"+combustible+"</td><td>"+capAlmacenamiento+"</td><td>"+nPasajeros+"</td><td>
"+sABS+"</td><td>"+sGPS+"</td><td>"+sclimatizacion+"</td><td>"+nRuedas+"</td><td>"+sd
escapotable+"</td><td>"+motivoVisita+"</td><td>"+fechaAlta+"</td><td>"+nombrecliente+"
</td></tr>";

                                        }

                                        soluciones+="</table></body></html>";

                                }

                                else {

                                        soluciones = "No hay resultados.";

                                }


                        } catch (SQLException e) {

                                e.printStackTrace();

                        }

                        return soluciones;

  }


  @Override

        public String getUltimoRegitro() {

        try {

                                resultado = sentencia.executeQuery("SELECT MAX(id) FROM vehiculo
WHERE tipoVehiculo='7'");

                                while (resultado.next()) {

                                        id = resultado.getString(1);

                                }


                        } catch (SQLException e) {

                                e.printStackTrace();

                        }

                        return id;

  }
```

```
	@Override
	public ArrayList MarcaVehiculo() {
		marcavehiculo = new ArrayList();
		try {
			resultado = sentencia.executeQuery("SELECT * FROM vehiculo WHERE
tipoVehiculo='1'");

			if (resultado.next() == true) //move o cursor
			{
				resultado.beforeFirst(); //volve a poñer de primeiro o cursor
				while (resultado.next()) {
					marca = resultado.getString(2); //marca

					marcavehiculo.add(marca);
				}
			}
			else {
				marcavehiculo.add("No hay resultados.");
			}

		} catch (SQLException e) {
			e.printStackTrace();
		}
		return marcavehiculo;
	}

	@Override
	public ArrayList ModeloVehiculo() {
		marcavehiculo = new ArrayList();
		try {
```

```
                    resultado = sentencia.executeQuery("SELECT * FROM vehiculo WHERE
tipoVehiculo='1'");

                    if (resultado.next() == true) //move o cursor
                    {
                        resultado.beforeFirst(); //volve a poñer de primeiro o cursor
                        while (resultado.next()) {
                            modelo = resultado.getString(3); //modelo

                            marcavehiculo.add(modelo);
                        }
                    }
                    else {
                        marcavehiculo.add("No hay resultados.");
                    }

                } catch (SQLException e) {
                    e.printStackTrace();
                }
                return marcavehiculo;
            }

            @Override
            public ArrayList MatriculaVehiculo() {
                marcavehiculo = new ArrayList();
                try {
                    resultado = sentencia.executeQuery("SELECT * FROM vehiculo WHERE
tipoVehiculo='1'");

                    if (resultado.next() == true) //move o cursor
                    {
                        resultado.beforeFirst(); //volve a poñer de primeiro o cursor
```

```
                while (resultado.next()) {
                        matricula = resultado.getString(4); //matricula

                        marcavehiculo.add(matricula);
                }
            }
            else {
                marcavehiculo.add("No hay resultados.");
            }

        } catch (SQLException e) {
            e.printStackTrace();
        }
        return marcavehiculo;
    }

    @Override
    public ArrayList IDVehiculo() {
        marcavehiculo = new ArrayList();
        try {
            resultado = sentencia.executeQuery("SELECT * FROM vehiculo WHERE
tipoVehiculo='1'");

            if (resultado.next() == true) //move o cursor
            {
                resultado.beforeFirst(); //volve a poñer de primeiro o cursor
                while (resultado.next()) {
                    id = resultado.getString(1); //nombre

                    marcavehiculo.add(id);
                }
```

```
                        }
                        else {
                                marcavehiculo.add("No hay resultados.");
                        }

                } catch (SQLException e) {
                        e.printStackTrace();
                }
                return marcavehiculo;
        }
}
```

17.7. Ficha

```
package textual;

import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;

/**
 * @author Adriana Armental Tomé
 * @version 06.03.2017
 */
public class Ficha
{
        protected String id;
        protected String idvehiculo;
   protected String idcliente;
   protected String idmecanico;
```

```
protected String idestado;
protected String descripcion;
protected String estado;
protected String motivoParado;
protected String fechaEntrada;
protected String fechaSalida;
protected String soluciones;
protected String nombrecliente;
protected String nombremecanico;
protected String nombrevehiculo;

protected ArrayList nombreficha;

static ConexionBD con;
static Connection conex;
static Statement sentencia, sentencia2, sentencia3, sentencia4, sentencia5, sentencia6;
static ResultSet resultado, resultado2, resultado3, resultado4, resultado5, resultado6;

public Ficha()
{
   con = new ConexionBD();
   conex = con.getCon();
   try {
                        sentencia = conex.createStatement();
                        sentencia2 = conex.createStatement();
                        sentencia3 = conex.createStatement();
                        sentencia4 = conex.createStatement();
                        sentencia5 = conex.createStatement();
                        sentencia6 = conex.createStatement();
               } catch (SQLException e) {
                        e.printStackTrace();
```

```
                }
  }

        public String getIdvehiculo(String id) {
        try {
                resultado = sentencia.executeQuery("SELECT * FROM ficha WHERE
id='"+id+"'");
                while (resultado.next()) {
                        idvehiculo = resultado.getString(8); //idvehiculo
                }

                } catch (SQLException e) {
                        e.printStackTrace();
                }
                return idvehiculo;
        }

        public String getNombrevehiculo(String id) {
        try {
                resultado = sentencia.executeQuery("SELECT * FROM ficha WHERE
id='"+id+"'");
                while (resultado.next()) {
                        idvehiculo = resultado.getString(8); //idvehiculo

                        resultado2 = sentencia2.executeQuery("SELECT * FROM
vehiculo WHERE id='"+idvehiculo+"'");
                        while (resultado2.next())
                        {
                                nombrevehiculo = resultado2.getString(2) + " " +
resultado2.getString(3) + " " + resultado2.getString(4);
                        }
                }
```

```
		} catch (SQLException e) {
			e.printStackTrace();
		}
		return nombrevehiculo;
	}

	public void setIdvehiculo(String idvehiculo) {
		this.idvehiculo = idvehiculo;
	}

	public String getIdcliente(String id) {
	try {
			resultado = sentencia.executeQuery("SELECT * FROM ficha WHERE
id='"+id+"'");
			while (resultado.next()) {
				idcliente = resultado.getString(7); //idcliente
			}

		} catch (SQLException e) {
			e.printStackTrace();
		}
		return idcliente;
	}

	public String getNombrecliente(String id) {
	try {
			resultado = sentencia.executeQuery("SELECT * FROM ficha WHERE
id='"+id+"'");
			while (resultado.next()) {
				idcliente = resultado.getString(7); //idcliente
```

```
                                resultado2 = sentencia2.executeQuery("SELECT * FROM
persona WHERE id='"+idcliente+"'");
                                while (resultado2.next())
                                {
                                        nombrecliente = resultado2.getString(3) + " " +
resultado2.getString(4);
                                }
                        }

                } catch (SQLException e) {
                        e.printStackTrace();
                }
                return nombrecliente;
        }

        public void setIdcliente(String idcliente) {
                this.idcliente = idcliente;
        }

        public String getIdmecanico(String id) {
        try {
                        resultado = sentencia.executeQuery("SELECT * FROM ficha WHERE
id='"+id+"'");
                        while (resultado.next()) {
                                idmecanico = resultado.getString(9); //idmecanico
                        }

                } catch (SQLException e) {
                        e.printStackTrace();
                }
                return idmecanico;
        }
```

```
	public String getNombremecanico(String id) {
	try {
			resultado = sentencia.executeQuery("SELECT * FROM ficha WHERE
id='"+id+"'");
			while (resultado.next()) {
				idmecanico = resultado.getString(9); //idmecanico
				resultado2 = sentencia2.executeQuery("SELECT * FROM
persona WHERE id='"+idmecanico+"'");
				while (resultado2.next())
				{
					nombremecanico = resultado2.getString(3) + " " +
resultado2.getString(4) + " "+resultado2.getString(2);
				}
			}

		} catch (SQLException e) {
			e.printStackTrace();
		}
		return nombremecanico;
	}

	public void setIdmecanico(String idmecanico) {
		this.idmecanico = idmecanico;
	}

	public String getDescripcion(String id) {
	try {
			resultado = sentencia.executeQuery("SELECT * FROM ficha WHERE
id='"+id+"'");
			while (resultado.next()) {
				descripcion = resultado.getString(2); //descripcion
```

```
                }

            } catch (SQLException e) {
                e.printStackTrace();
            }
            return descripcion;
        }

        public void setDescripcion(String descripcion) {
            this.descripcion = descripcion;
        }

        public String getMotivoParado(String id) {
        try {
                resultado = sentencia.executeQuery("SELECT * FROM ficha WHERE
id='"+id+"'");
                while (resultado.next()) {
                    motivoParado = resultado.getString(6); //motivoParado
                }

            } catch (SQLException e) {
                e.printStackTrace();
            }
            return motivoParado;
        }

        public void setMotivoParado(String motivoParado) {
            this.motivoParado = motivoParado;
        }

        public String getFechaEntrada(String id) {
```

```
	try {
			resultado = sentencia.executeQuery("SELECT * FROM ficha WHERE
id='"+id+"'");
			while (resultado.next()) {
				fechaEntrada = resultado.getString(4); //fechaEntrada
			}

		} catch (SQLException e) {
			e.printStackTrace();
		}
		return fechaEntrada;
	}

	public void setFechaEntrada(String fechaEntrada) {
		this.fechaEntrada = fechaEntrada;
	}

	public String getFechaSalida(String id) {
	try {
			resultado = sentencia.executeQuery("SELECT * FROM ficha WHERE
id='"+id+"'");
			while (resultado.next()) {
				fechaSalida = resultado.getString(5); //fechaSalida
			}

		} catch (SQLException e) {
			e.printStackTrace();
		}
		return fechaSalida;
	}

	public void setFechaSalida(String fechaSalida) {
```

```
		this.fechaSalida = fechaSalida;
	}

	public String getEstado(String id) {
	try {
			resultado = sentencia.executeQuery("SELECT * FROM ficha WHERE
id='"+id+"'");
			while (resultado.next()) {
				idestado = resultado.getString(3); //estado
				resultado2 = sentencia2.executeQuery("SELECT * FROM
tipo_estado WHERE id='"+idestado+"'");
				while (resultado2.next())
				{
					estado = resultado2.getString(2);
				}
			}

		} catch (SQLException e) {
			e.printStackTrace();
		}
		return estado;
	}

	public void setEstado(String estado) {
		this.estado = estado;
	}

	public boolean insertar(String descripcion, String estado, String fechaEntrada, String
fechaSalida, String motivoParado, String idcliente, String idvehiculo, String idmecanico) {

		if (fechaSalida.isEmpty()) {
```

```
                try {
                    String sql = "INSERT INTO ficha (`descripcion`, `estado`, `fechaEntrada`, `fechaSalida`, `motivo`, `idCliente`, `idVehiculo`, `idMecanico`) VALUES ("
                            +"'"+descripcion+"','"+estado+"','"+fechaEntrada+"', NULL,'"+motivoParado+"','"+idcliente+"','"+idvehiculo+"','"+idmecanico+"')";
                    sentencia.executeUpdate(sql);
                    return true;

                } catch(SQLException e) {
                        e.printStackTrace();
                        return false;
                    }
                }
                else {
                try {
                    String sql = "INSERT INTO ficha (`descripcion`, `estado`, `fechaEntrada`, `fechaSalida`, `motivo`, `idCliente`, `idVehiculo`, `idMecanico`) VALUES ("

          +"'"+descripcion+"','"+estado+"','"+fechaEntrada+"','"+fechaSalida+"','"+motivoParado+"','"+idcliente+"','"+idvehiculo+"','"+idmecanico+"')";
                    sentencia.executeUpdate(sql);
                    return true;

                } catch(SQLException e) {
                        e.printStackTrace();
                        return false;
                    }

                }
        }

   public boolean editar(String id, String descripcion, String estado, String fechaSalida, String motivoParado) {
```

```
        if (fechaSalida.isEmpty()) {
                try {
            sentencia.executeUpdate("UPDATE ficha SET descripcion='"+descripcion+"',
estado='"+estado+"', fechaSalida = NULL, motivo='"+motivoParado+"' WHERE id = '"+id+"'");
            return true;
                } catch(SQLException e) {
                        e.printStackTrace();
                        return false;
                }
        }
        else {
                try {
            sentencia.executeUpdate("UPDATE ficha SET descripcion='"+descripcion+"',
estado='"+estado+"', fechaSalida='"+fechaSalida+"', motivo='"+motivoParado+"' WHERE id =
'"+id+"'");
            return true;
                } catch(SQLException e) {
                        e.printStackTrace();
                        return false;
                }
        }
        }

   public boolean borrar(String id) {

                try {
            sentencia.executeUpdate("DELETE FROM ficha WHERE id='"+id+"'");
            return true;

                } catch(SQLException e) {
                        e.printStackTrace();
```

```
                return false;
            }
        }

    public String listar() {

            try {
                resultado = sentencia.executeQuery("SELECT * FROM ficha");

                if (resultado.next() == true) //move o cursor
                {
                    soluciones =
"<html><body><table><tr><th>DESCRIPCIÓN</th><th>ESTADO</th><th>FECHA
ENTRADA</th><th>FECHA SALIDA</th><th>MOTIVO
PARADO</th><th>CLIENTE</th><th>COCHE</th><th>MECÁNICO</th></tr>";
                    resultado.beforeFirst(); //volve a poñer de primeiro o cursor
                    while (resultado.next()) {
                        descripcion = resultado.getString(2); //descripcion
                        idestado = resultado.getString(3); //estado
                        resultado2 = sentencia2.executeQuery("SELECT *
FROM tipo_estado WHERE id='"+idestado+"'");
                        while (resultado2.next())
                        {
                            estado = resultado2.getString(2);
                        }
                        fechaEntrada = resultado.getString(4); //fechaentrada
                        fechaSalida = resultado.getString(5); //fechasalida
                        if (fechaSalida == null) {
                            fechaSalida = "";
                        }
                        motivoParado = resultado.getString(6);
//motivoparado
                        if (motivoParado == null || motivoParado.isEmpty()) {
```

```
                                        motivoParado = "";
                                    }
                                    idcliente = resultado.getString(7); //idcliente
                                    idvehiculo = resultado.getString(8); //idvehiculo
                                    idmecanico = resultado.getString(9); //idmecanico

                                    resultado3 = sentencia3.executeQuery("SELECT *
FROM persona WHERE id='"+idcliente+"'");
                                    while (resultado3.next())
                                    {
                                        nombrecliente = resultado3.getString(3) + " " +
resultado3.getString(4);
                                    }

                                    resultado4 = sentencia4.executeQuery("SELECT *
FROM persona WHERE id='"+idmecanico+"'");
                                    while (resultado4.next())
                                    {
                                        nombremecanico = resultado4.getString(3) + "
" + resultado4.getString(4) + " "+resultado4.getString(2);
                                    }

                                    resultado5 = sentencia5.executeQuery("SELECT *
FROM vehiculo WHERE id='"+idvehiculo+"'");
                                    while (resultado5.next())
                                    {
                                        nombrevehiculo = resultado5.getString(2) + " "
+ resultado5.getString(3) + " " + resultado5.getString(4);
                                    }

        soluciones+="<tr><td>"+descripcion+"</td><td>"+estado+"</td><td>"+fechaEntrada+
"</td><td>"+fechaSalida+"</td><td>"+motivoParado+"</td><td>"+nombrecliente+"</td><td>
"+nombrevehiculo+"</td><td>"+nombremecanico+"</td></tr>";
```

```
                    }
                    soluciones+="</table></body></html>";
                }
                else {
                    soluciones = "No hay resultados.";
                }

            } catch (SQLException e) {
                e.printStackTrace();
            }
            return soluciones;

        }

        public String consultarporMecanico(String nombre, String fechaEnt1, String fechaEnt2)
{

            if (fechaEnt1.isEmpty() || fechaEnt1.equals("") || fechaEnt2.isEmpty() ||
fechaEnt2.equals("")) {
                try {
                    resultado = sentencia.executeQuery("SELECT * FROM persona
WHERE UPPER(nombre) LIKE UPPER('%"+nombre+"%') AND tipoPersona='2'");

                    if (resultado.next() == true) //move o cursor
                    {
                        resultado.beforeFirst(); //volve a poñer de primeiro o
cursor
                        while (resultado.next()) {
                            idmecanico = resultado.getString(1);
//idmecanico
                            resultado2 = sentencia2.executeQuery("SELECT
* FROM ficha WHERE idMecanico='"+idmecanico+"'");
```

```
                                        if (resultado2.next() == true) //move o cursor
                                        {
                                                soluciones =
"<html><body><table><tr><th>DESCRIPCIÓN</th><th>ESTADO</th><th>FECHA
ENTRADA</th><th>FECHA SALIDA</th><th>MOTIVO
PARADO</th><th>CLIENTE</th><th>COCHE</th><th>MECÁNICO</th></tr>";
                                                resultado2.beforeFirst(); //volve a
poñer de primeiro o cursor
                                                while (resultado2.next()) {
                                                        descripcion =
resultado2.getString(2); //descripcion
                                                        idestado =
resultado2.getString(3); //estado
                                                        resultado3 =
sentencia3.executeQuery("SELECT * FROM tipo_estado WHERE id='"+idestado+"'");
                                                        while (resultado3.next())
                                                        {
                                                                estado =
resultado3.getString(2);
                                                        }
                                                        fechaEntrada =
resultado2.getString(4); //fechaentrada
                                                        fechaSalida =
resultado2.getString(5); //fechasalida
                                                        if (fechaSalida == null) {
                                                                fechaSalida = "";
                                                        }
                                                        motivoParado =
resultado2.getString(6); //motivoparado
                                                        if (motivoParado == null) {
                                                                motivoParado = "";
                                                        }
                                                        idcliente =
resultado2.getString(7); //idcliente
                                                        idvehiculo =
resultado2.getString(8); //idvehiculo
```

```
                                                    idmecanico =
resultado2.getString(9); //idmecanico


                                                    resultado4 =
sentencia4.executeQuery("SELECT * FROM persona WHERE id='"+idcliente+"'");
                                                    while (resultado4.next())
                                                    {
                                                        nombrecliente =
resultado4.getString(3) + " " + resultado4.getString(4);
                                                    }


                                                    resultado5 =
sentencia4.executeQuery("SELECT * FROM persona WHERE id='"+idmecanico+"'");
                                                    while (resultado5.next())
                                                    {
                                                        nombremecanico =
resultado5.getString(3) + " " + resultado5.getString(4) + " "+resultado5.getString(2);
                                                    }


                                                    resultado6 =
sentencia6.executeQuery("SELECT * FROM vehiculo WHERE id='"+idvehiculo+"'");
                                                    while (resultado6.next())
                                                    {
                                                        nombrevehiculo =
resultado6.getString(2) + " " + resultado6.getString(3) + " " + resultado6.getString(4);
                                                    }



        soluciones+="<tr><td>"+descripcion+"</td><td>"+estado+"</td><td>"+fechaEntrada+
"</td><td>"+fechaSalida+"</td><td>"+motivoParado+"</td><td>"+nombrecliente+"</td><td>
"+nombrevehiculo+"</td><td>"+nombremecanico+"</td></tr>";
                                                }
                                            }
                                        }
                                        soluciones+="</table></body></html>";
```

```
                                        }
                                        else {
                                                soluciones = "No hay resultados.";
                                        }

                                } catch (SQLException e) {
                                        e.printStackTrace();
                                }
                                return soluciones;
                        }
                        else {
                                try {
                                        resultado = sentencia.executeQuery("SELECT * FROM persona
WHERE UPPER(nombre) LIKE UPPER('%"+nombre+"%') AND tipoPersona='2'");

                                        if (resultado.next() == true) //move o cursor
                                        {
                                                resultado.beforeFirst(); //volve a poñer de primeiro o
cursor
                                                while (resultado.next()) {
                                                        idmecanico= resultado.getString(1);
//idmecanico
                                                        resultado2 = sentencia2.executeQuery("SELECT
* FROM ficha WHERE fechaEntrada >= '"+fechaEnt1+"' AND fechaEntrada <= '"+fechaEnt2+"'
AND idMecanico = '"+idmecanico+"'");
                                                        if (resultado2.next() == true) //move o cursor
                                                        {
                                                                soluciones =
"<html><body><table><tr><th>DESCRIPCIÓN</th><th>ESTADO</th><th>FECHA
ENTRADA</th><th>FECHA SALIDA</th><th>MOTIVO
PARADO</th><th>CLIENTE</th><th>COCHE</th><th>MECÁNICO</th></tr>";
                                                                resultado2.beforeFirst(); //volve a
poñer de primeiro o cursor
                                                                while (resultado2.next()) {
```

```
                descripcion =
resultado2.getString(2); //descripcion
                idestado =
resultado2.getString(3); //estado
                resultado3 =
sentencia3.executeQuery("SELECT * FROM tipo_estado WHERE id='"+idestado+"'");
                while (resultado3.next())
                {
                        estado =
resultado3.getString(2);
                }
                fechaEntrada =
resultado2.getString(4); //fechaentrada
                fechaSalida =
resultado2.getString(5); //fechasalida
                if (fechaSalida == null) {
                        fechaSalida = "";
                }
                motivoParado =
resultado2.getString(6); //motivoparado
                if (motivoParado == null) {
                        motivoParado = "";
                }
                idcliente =
resultado2.getString(7); //idcliente
                idvehiculo =
resultado2.getString(8); //idvehiculo
                idmecanico =
resultado2.getString(9); //idmecanico


                resultado4 =
sentencia4.executeQuery("SELECT * FROM persona WHERE id='"+idcliente+"'");
                while (resultado4.next())
                {
```

```
                                                    nombrecliente =
resultado4.getString(3) + " " + resultado4.getString(4);
                                                }


                                                resultado5 =
sentencia4.executeQuery("SELECT * FROM persona WHERE id='"+idmecanico+"'");
                                                while (resultado5.next())
                                                {
                                                    nombremecanico =
resultado5.getString(3) + " " + resultado5.getString(4) + " "+resultado5.getString(2);
                                                }


                                                resultado6 =
sentencia6.executeQuery("SELECT * FROM vehiculo WHERE id='"+idvehiculo+"'");
                                                while (resultado6.next())
                                                {
                                                    nombrevehiculo =
resultado6.getString(2) + " " + resultado6.getString(3) + " " + resultado6.getString(4);
                                                }



        soluciones+="<tr><td>"+descripcion+"</td><td>"+estado+"</td><td>"+fechaEntrada+
"</td><td>"+fechaSalida+"</td><td>"+motivoParado+"</td><td>"+nombrecliente+"</td><td>
"+nombrevehiculo+"</td><td>"+nombremecanico+"</td></tr>";
                                                }
                                            }
                                        }
                                        soluciones+="</table></body></html>";
                                    }
                                    else {
                                        soluciones = "No hay resultados.";
                                    }

                                } catch (SQLException e) {
```

```
                    e.printStackTrace();
                }
                return soluciones;
            }
        }

        public String consultarporCliente(String nombre, String fechaEnt1, String fechaEnt2) {

            if (fechaEnt1.isEmpty() || fechaEnt1.equals("") || fechaEnt2.isEmpty() ||
fechaEnt2.equals("")) {
                try {
                    resultado = sentencia.executeQuery("SELECT * FROM persona
WHERE UPPER(nombre) LIKE UPPER('%"+nombre+"%') AND tipoPersona='1'");

                    if (resultado.next() == true) //move o cursor
                    {
                        resultado.beforeFirst(); //volve a poñer de primeiro o
cursor
                        while (resultado.next()) {
                            idcliente = resultado.getString(1); //idcliente
                            resultado2 = sentencia2.executeQuery("SELECT
* FROM ficha WHERE idCliente='"+idcliente+"'");
                            if (resultado2.next() == true) //move o cursor
                            {
                                soluciones =
"<html><body><table><tr><th>DESCRIPCIÓN</th><th>ESTADO</th><th>FECHA
ENTRADA</th><th>FECHA SALIDA</th><th>MOTIVO
PARADO</th><th>CLIENTE</th><th>COCHE</th><th>MECÁNICO</th></tr>";
                                resultado2.beforeFirst(); //volve a
poñer de primeiro o cursor
                                while (resultado2.next()) {
                                    descripcion =
resultado2.getString(2); //descripcion
                                    idestado =
resultado2.getString(3); //estado
```

```
                                                        resultado3 =
sentencia3.executeQuery("SELECT * FROM tipo_estado WHERE id='"+idestado+"'");
                                                        while (resultado3.next())
                                                        {
                                                                estado =
resultado3.getString(2);
                                                        }
                                                        fechaEntrada =
resultado2.getString(4); //fechaentrada
                                                        fechaSalida =
resultado2.getString(5); //fechasalida
                                                        if (fechaSalida == null) {
                                                                fechaSalida = "";
                                                        }
                                                        motivoParado =
resultado2.getString(6); //motivoparado
                                                        if (motivoParado == null) {
                                                                motivoParado = "";
                                                        }
                                                        idcliente =
resultado2.getString(7); //idcliente
                                                        idvehiculo =
resultado2.getString(8); //idvehiculo
                                                        idmecanico =
resultado2.getString(9); //idmecanico


                                                        resultado4 =
sentencia4.executeQuery("SELECT * FROM persona WHERE id='"+idcliente+"'");
                                                        while (resultado4.next())
                                                        {
                                                                nombrecliente =
resultado4.getString(3) + " " + resultado4.getString(4);
                                                        }
```

```
                                                        resultado5 =
sentencia4.executeQuery("SELECT * FROM persona WHERE id='"+idmecanico+"'");
                                                        while (resultado5.next())
                                                        {
                                                                nombremecanico =
resultado5.getString(3) + " " + resultado5.getString(4) + " "+resultado5.getString(2);
                                                        }

                                                        resultado6 =
sentencia6.executeQuery("SELECT * FROM vehiculo WHERE id='"+idvehiculo+"'");
                                                        while (resultado6.next())
                                                        {
                                                                nombrevehiculo =
resultado6.getString(2) + " " + resultado6.getString(3) + " " + resultado6.getString(4);
                                                        }

        soluciones+="<tr><td>"+descripcion+"</td><td>"+estado+"</td><td>"+fechaEntrada+
"</td><td>"+fechaSalida+"</td><td>"+motivoParado+"</td><td>"+nombrecliente+"</td><td>
"+nombrevehiculo+"</td><td>"+nombremecanico+"</td></tr>";
                                                                }
                                                        }
                                                }
                                                soluciones+="</table></body></html>";
                                        }
                                        else {
                                                soluciones = "No hay resultados.";
                                        }

                                } catch (SQLException e) {
                                        e.printStackTrace();
                                }
                                return soluciones;
                        }
```

```
                else {
                        try {
                                resultado = sentencia.executeQuery("SELECT * FROM persona
WHERE UPPER(nombre) LIKE UPPER('%"+nombre+"%') AND tipoPersona='1'");

                                if (resultado.next() == true) //move o cursor
                                {
                                        resultado.beforeFirst(); //volve a poñer de primeiro o
cursor
                                        while (resultado.next()) {
                                                idcliente = resultado.getString(1); //idcliente
                                                resultado2 = sentencia2.executeQuery("SELECT
* FROM ficha WHERE fechaEntrada >= '"+fechaEnt1+"' AND fechaEntrada <= '"+fechaEnt2+"'
AND idCliente = '"+idcliente+"'");
                                                if (resultado2.next() == true) //move o cursor
                                                {
                                                        soluciones =
"<html><body><table><tr><th>DESCRIPCIÓN</th><th>ESTADO</th><th>FECHA
ENTRADA</th><th>FECHA SALIDA</th><th>MOTIVO
PARADO</th><th>CLIENTE</th><th>COCHE</th><th>MECÁNICO</th></tr>";
                                                        resultado2.beforeFirst(); //volve a
poñer de primeiro o cursor
                                                        while (resultado2.next()) {
                                                                descripcion =
resultado2.getString(2); //descripcion
                                                                idestado =
resultado2.getString(3); //estado
                                                                resultado3 =
sentencia3.executeQuery("SELECT * FROM tipo_estado WHERE id='"+idestado+"'");
                                                                while (resultado3.next())
                                                                {
                                                                        estado =
resultado3.getString(2);
                                                                }
                                                                fechaEntrada =
resultado2.getString(4); //fechaentrada
```

```
                                                fechaSalida =
resultado2.getString(5); //fechasalida
                                                if (fechaSalida == null) {
                                                        fechaSalida = "";
                                                }
                                                motivoParado =
resultado2.getString(6); //motivoparado
                                                if (motivoParado == null) {
                                                        motivoParado = "";
                                                }
                                                idcliente =
resultado2.getString(7); //idcliente
                                                idvehiculo =
resultado2.getString(8); //idvehiculo
                                                idmecanico =
resultado2.getString(9); //idmecanico


                                                resultado4 =
sentencia4.executeQuery("SELECT * FROM persona WHERE id='"+idcliente+"'");
                                                while (resultado4.next())
                                                {
                                                        nombrecliente =
resultado4.getString(3) + " " + resultado4.getString(4);
                                                }


                                                resultado5 =
sentencia4.executeQuery("SELECT * FROM persona WHERE id='"+idmecanico+"'");
                                                while (resultado5.next())
                                                {
                                                        nombremecanico =
resultado5.getString(3) + " " + resultado5.getString(4) + " "+resultado5.getString(2);
                                                }


                                                resultado6 =
sentencia6.executeQuery("SELECT * FROM vehiculo WHERE id='"+idvehiculo+"'");
```

```
                                                        while (resultado6.next())
                                                        {
                                                                nombrevehiculo =
resultado6.getString(2) + " " + resultado6.getString(3) + " " + resultado6.getString(4);
                                                        }


        soluciones+="<tr><td>"+descripcion+"</td><td>"+estado+"</td><td>"+fechaEntrada+
"</td><td>"+fechaSalida+"</td><td>"+motivoParado+"</td><td>"+nombrecliente+"</td><td>
"+nombrevehiculo+"</td><td>"+nombremecanico+"</td></tr>";
                                                                }
                                                        }
                                                }
                                                soluciones+="</table></body></html>";
                                        }
                                        else {
                                                soluciones = "No hay resultados.";
                                        }

                                } catch (SQLException e) {
                                        e.printStackTrace();
                                }
                                return soluciones;
                        }
                }

                public String consultarporEstado(String estado, String fechaEnt1, String fechaEnt2) {

                        if (fechaEnt1.isEmpty() || fechaEnt1.equals("") || fechaEnt2.isEmpty() ||
fechaEnt2.equals("")) {
                                try {
                                        resultado = sentencia.executeQuery("SELECT * FROM
tipo_estado WHERE nombre_tipo='"+estado+"'");
```

```
                    if (resultado.next() == true) //move o cursor
                    {
                        resultado.beforeFirst(); //volve a poñer de primeiro o
cursor
                        while (resultado.next()) {
                            idestado = resultado.getString(1); //idestado
                            resultado2 = sentencia2.executeQuery("SELECT
* FROM ficha WHERE estado='"+idestado+"'");
                            if (resultado2.next() == true) //move o cursor
                            {
                                soluciones =
"<html><body><table><tr><th>DESCRIPCIÓN</th><th>ESTADO</th><th>FECHA
ENTRADA</th><th>FECHA SALIDA</th><th>MOTIVO
PARADO</th><th>CLIENTE</th><th>COCHE</th><th>MECÁNICO</th></tr>";
                                resultado2.beforeFirst(); //volve a
poñer de primeiro o cursor
                                while (resultado2.next()) {
                                    descripcion =
resultado2.getString(2); //descripcion
                                    idestado =
resultado2.getString(3); //estado
                                    resultado3 =
sentencia3.executeQuery("SELECT * FROM tipo_estado WHERE id='"+idestado+"'");
                                    while (resultado3.next())
                                    {
                                        estado =
resultado3.getString(2);
                                    }
                                    fechaEntrada =
resultado2.getString(4); //fechaentrada
                                    fechaSalida =
resultado2.getString(5); //fechasalida
                                    if (fechaSalida == null) {
                                        fechaSalida = "";
```

```
                                                        }
                                                        motivoParado =
resultado2.getString(6); //motivoparado
                                                        if (motivoParado == null) {
                                                                motivoParado = "";
                                                        }
                                                        idcliente =
resultado2.getString(7); //idcliente
                                                        idvehiculo =
resultado2.getString(8); //idvehiculo
                                                        idmecanico =
resultado2.getString(9); //idmecanico


                                                        resultado4 =
sentencia4.executeQuery("SELECT * FROM persona WHERE id='"+idcliente+"'");
                                                        while (resultado4.next())
                                                        {
                                                                nombrecliente =
resultado4.getString(3) + " " + resultado4.getString(4);
                                                        }


                                                        resultado5 =
sentencia4.executeQuery("SELECT * FROM persona WHERE id='"+idmecanico+"'");
                                                        while (resultado5.next())
                                                        {
                                                                nombremecanico =
resultado5.getString(3) + " " + resultado5.getString(4) + " "+resultado5.getString(2);
                                                        }


                                                        resultado6 =
sentencia6.executeQuery("SELECT * FROM vehiculo WHERE id='"+idvehiculo+"'");
                                                        while (resultado6.next())
                                                        {
                                                                nombrevehiculo =
resultado6.getString(2) + " " + resultado6.getString(3) + " " + resultado6.getString(4);
```

```
                                                }


        soluciones+="<tr><td>"+descripcion+"</td><td>"+estado+"</td><td>"+fechaEntrada+
"</td><td>"+fechaSalida+"</td><td>"+motivoParado+"</td><td>"+nombrecliente+"</td><td>
"+nombrevehiculo+"</td><td>"+nombremecanico+"</td></tr>";
                                                }
                                        }
                                        }
                                        soluciones+="</table></body></html>";
                                }
                                else {
                                        soluciones = "No hay resultados.";
                                }

                        } catch (SQLException e) {
                                e.printStackTrace();
                        }
                        return soluciones;
                }
                else {
                        try {
                                resultado = sentencia.executeQuery("SELECT * FROM
tipo_estado WHERE nombre_tipo='"+estado+"'");

                                if (resultado.next() == true) //move o cursor
                                {
                                        resultado.beforeFirst(); //volve a poñer de primeiro o
cursor
                                        while (resultado.next()) {
                                                idestado = resultado.getString(1); //idestado
                                                resultado2 = sentencia2.executeQuery("SELECT
* FROM ficha WHERE fechaEntrada >= '"+fechaEnt1+"' AND fechaEntrada <= '"+fechaEnt2+"'
AND estado = '"+idestado+"'");
```

```
                                        if (resultado2.next() == true) //move o cursor
                                        {
                                                soluciones =
"<html><body><table><tr><th>DESCRIPCIÓN</th><th>ESTADO</th><th>FECHA
ENTRADA</th><th>FECHA SALIDA</th><th>MOTIVO
PARADO</th><th>CLIENTE</th><th>COCHE</th><th>MECÁNICO</th></tr>";
                                                resultado2.beforeFirst(); //volve a
poñer de primeiro o cursor
                                                while (resultado2.next()) {
                                                        descripcion =
resultado2.getString(2); //descripcion
                                                        idestado =
resultado2.getString(3); //estado
                                                        resultado3 =
sentencia3.executeQuery("SELECT * FROM tipo_estado WHERE id='"+idestado+"'");
                                                        while (resultado3.next())
                                                        {
                                                                estado =
resultado3.getString(2);
                                                        }
                                                        fechaEntrada =
resultado2.getString(4); //fechaentrada
                                                        fechaSalida =
resultado2.getString(5); //fechasalida
                                                        if (fechaSalida == null) {
                                                                fechaSalida = "";
                                                        }
                                                        motivoParado =
resultado2.getString(6); //motivoparado
                                                        if (motivoParado == null) {
                                                                motivoParado = "";
                                                        }
                                                        idcliente =
resultado2.getString(7); //idcliente
                                                        idvehiculo =
resultado2.getString(8); //idvehiculo
```

```
                                                    idmecanico =
resultado2.getString(9); //idmecanico


                                                    resultado4 =
sentencia4.executeQuery("SELECT * FROM persona WHERE id='"+idcliente+"'");
                                                    while (resultado4.next())
                                                    {
                                                        nombrecliente =
resultado4.getString(3) + " " + resultado4.getString(4);
                                                    }


                                                    resultado5 =
sentencia4.executeQuery("SELECT * FROM persona WHERE id='"+idmecanico+"'");
                                                    while (resultado5.next())
                                                    {
                                                        nombremecanico =
resultado5.getString(3) + " " + resultado5.getString(4) + " "+resultado5.getString(2);
                                                    }


                                                    resultado6 =
sentencia6.executeQuery("SELECT * FROM vehiculo WHERE id='"+idvehiculo+"'");
                                                    while (resultado6.next())
                                                    {
                                                        nombrevehiculo =
resultado6.getString(2) + " " + resultado6.getString(3) + " " + resultado6.getString(4);
                                                    }



        soluciones+="<tr><td>"+descripcion+"</td><td>"+estado+"</td><td>"+fechaEntrada+
"</td><td>"+fechaSalida+"</td><td>"+motivoParado+"</td><td>"+nombrecliente+"</td><td>
"+nombrevehiculo+"</td><td>"+nombremecanico+"</td></tr>";
                                                }
                                            }
                                        }
                                        soluciones+="</table></body></html>";
```

```
                        }
                        else {
                                soluciones = "No hay resultados.";
                        }

                } catch (SQLException e) {
                        e.printStackTrace();
                }
                return soluciones;
        }
}

public ArrayList FichaCliente() {
        nombreficha = new ArrayList();
        try {
                resultado = sentencia.executeQuery("SELECT * FROM ficha");

                if (resultado.next() == true) //move o cursor
                {
                        resultado.beforeFirst(); //volve a poñer de primeiro o cursor
                        while (resultado.next()) {
                                idcliente = resultado.getString(7); //idcliente

                                resultado2 = sentencia2.executeQuery("SELECT *
FROM persona WHERE id='"+idcliente+"'");
                                while (resultado2.next())
                                {
                                        nombrecliente = resultado2.getString(3) + " " +
resultado2.getString(4);
                                        nombreficha.add(nombrecliente);
                                }
```

```
                }
            }
            else {
                soluciones = "No hay resultados.";
            }

        } catch (SQLException e) {
            e.printStackTrace();
        }
        return nombreficha;
    }

    public ArrayList FichaVehiculo() {
        nombreficha = new ArrayList();
        try {
            resultado = sentencia.executeQuery("SELECT * FROM ficha");

            if (resultado.next() == true) //move o cursor
            {
                resultado.beforeFirst(); //volve a poñer de primeiro o cursor
                while (resultado.next()) {
                    idvehiculo = resultado.getString(8); //idvehiculo

                    resultado2 = sentencia2.executeQuery("SELECT *
FROM vehiculo WHERE id='"+idvehiculo+"'");
                    while (resultado2.next())
                    {
                        nombrevehiculo = resultado2.getString(2) + " "
+ resultado2.getString(3) + " " + resultado2.getString(4);
                        nombreficha.add(nombrevehiculo);
                    }
                }
```

```
                        }
                        else {
                                soluciones = "No hay resultados.";
                        }

                } catch (SQLException e) {
                        e.printStackTrace();
                }
                return nombreficha;
        }

        public ArrayList FichaMecanico() {
                nombreficha = new ArrayList();
                try {
                        resultado = sentencia.executeQuery("SELECT * FROM ficha");

                        if (resultado.next() == true) //move o cursor
                        {
                                resultado.beforeFirst(); //volve a poñer de primeiro o cursor
                                while (resultado.next()) {
                                        idmecanico = resultado.getString(9); //idmecanico
                                        if (idmecanico == null) {
                                                nombreficha.add("mecánico no asignado");
                                        }
                                        else {
                                                resultado2 = sentencia2.executeQuery("SELECT
* FROM persona WHERE id='"+idmecanico+"'");
                                                while (resultado2.next())
                                                {
                                                        nombremecanico =
resultado2.getString(3) + " " + resultado2.getString(4) + " "+resultado2.getString(2);
                                                        nombreficha.add(nombremecanico);
```

```
                                        }
                                }

                        }
                }
                else {
                        soluciones = "No hay resultados.";
                }

        } catch (SQLException e) {
                e.printStackTrace();
        }
        return nombreficha;
}

public ArrayList FichaID() {
        nombreficha = new ArrayList();
        try {
                resultado = sentencia.executeQuery("SELECT * FROM ficha");

                if (resultado.next() == true) //move o cursor
                {
                        resultado.beforeFirst(); //volve a poñer de primeiro o cursor
                        while (resultado.next()) {
                                id = resultado.getString(1); //id

                                nombreficha.add(id);
                        }
                }
                else {
                        soluciones = "No hay resultados.";
```

```
                        }


                } catch (SQLException e) {
                        e.printStackTrace();
                }
                return nombreficha;
        }

   public void cerrarMetodos() {
        try {
                resultado.close();
                resultado2.close();
                resultado3.close();
                resultado4.close();
                resultado5.close();
                resultado6.close();
                        sentencia.close();
                        sentencia2.close();
                        sentencia3.close();
                        sentencia4.close();
                        sentencia5.close();
                        sentencia6.close();
                conex.close();
                } catch (SQLException e) {
                        e.printStackTrace();
                }
   }
}
```

17.8. Ofertas.java

```
package textual;
```

```
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.Calendar;

/**
 * @author Adriana Armental Tomé
 * @version 06.03.2017
 */
public class Ofertas
{
   protected String idofertaenviada, nombreoferta, clienteoferta, clienteapeloferta,
correoclienteoferta, correocomercialoferta, comercialoferta, idoferta, idcomercial, idcliente,
soluciones, fecha, fechaActual;
   protected ArrayList nombreofertas;

   static ConexionBD con;
   static Connection conex;
   static Statement sentencia, sentencia2, sentencia3, sentencia4;
   static ResultSet resultado, resultado2, resultado3, resultado4;

   public Ofertas()
   {
         String dia2 = "";
                  String mes2 = "";

      Calendar cal = Calendar.getInstance();
         int dia = cal.get(Calendar.DATE);
         if (dia<10) {
```

```
                dia2 = "0"+dia;
        }
        else {
                dia2 = ""+dia;
        }
        int mes = cal.get(Calendar.MONTH) + 1;
        if (mes<10) {
                mes2 = "0"+mes;
        }
        else {
                mes2 = ""+mes;
        }
        int annio = cal.get(Calendar.YEAR);

        fechaActual = annio + "-" + mes2 + "-" + dia2;

    con = new ConexionBD();
    conex = con.getCon();
    try {
                        sentencia = conex.createStatement();
                        sentencia2 = conex.createStatement();
                        sentencia3 = conex.createStatement();
                        sentencia4 = conex.createStatement();
                } catch (SQLException e) {
                        e.printStackTrace();
                }
  }

        public String getFecha(String idoferta) {
        try {
```

```
                        resultado = sentencia.executeQuery("SELECT MAX(id) FROM
oferta_enviada WHERE idOferta='"+idoferta+"'");
                        while (resultado.next()) {
                                idofertaenviada = resultado.getString(1); //id
                                resultado2 = sentencia2.executeQuery("SELECT * FROM
oferta_enviada WHERE id='"+idofertaenviada+"'");
                                while (resultado2.next()) {
                                        fecha = resultado2.getString(5); //id
                                }
                        }

                } catch (SQLException e) {
                        e.printStackTrace();
                }
                return fecha;
        }

        public boolean insertar(String descripcion) {

        try {
            String sql = "INSERT INTO ofertas (`nombre_oferta`) VALUES ('"+descripcion+"')";
            sentencia.executeUpdate(sql);
            return true;

                } catch(SQLException e) {
                        e.printStackTrace();
                        return false;
                }
        }

        public boolean enviar(String idcli, String idcom, String idofe) {
                String dia2 = "";
```

```
                String mes2 = "";


    Calendar c = Calendar.getInstance();
        int dia = c.get(Calendar.DATE);
        if (dia<10) {
                dia2 = "0"+dia;
        }
        else {
                dia2 = ""+dia;
        }
        int mes = c.get(Calendar.MONTH) + 1;
        if (mes<10) {
                mes2 = "0"+mes;
        }
        else {
                mes2 = ""+mes;
        }
        int annio = c.get(Calendar.YEAR);


        fecha = annio + "-" + mes2 + "-" + dia2;
        try {
            String sql = "INSERT INTO oferta_enviada (`idOferta`, `idCliente`, `idComercial`,
`fechaenvio`) VALUES ('"+idofe+"','"+idcli+"','"+idcom+"','"+fecha+"')";
            sentencia.executeUpdate(sql);
            return true;


            } catch(SQLException e) {
                    e.printStackTrace();
                    return false;
            }
        }
```

```
	public String listar() {

		try {
			resultado = sentencia.executeQuery("SELECT * FROM ofertas");

			if (resultado.next() == true) //move o cursor
			{
				soluciones =
"<html><body><table><tr><th>OFERTAS</th></tr>";
				resultado.beforeFirst(); //volve a poñer de primeiro o cursor
				while (resultado.next()) {
					nombreoferta = resultado.getString(2);

					soluciones+="<tr><td>"+nombreoferta+"</td></tr>";

				}
				soluciones+="</table></body></html>";
			}
			else {
				soluciones = "No hay resultados.";
			}

		} catch (SQLException e) {
			e.printStackTrace();
		}
		return soluciones;
	}

	public String listarEnv() {
```

```
                try {
                        resultado = sentencia.executeQuery("SELECT * FROM
oferta_enviada");

                        if (resultado.next() == true) //move o cursor
                        {
                                soluciones =
"<html><body><table><tr><th>OFERTA</th><th>CLIENTE</th><th>CORREO
CLIENTE</th><th>COMERCIAL</th><th>CORREO COMERCIAL</th><th>FECHA
ENVÍO</th></tr>";
                                resultado.beforeFirst(); //volve a poñer de primeiro o cursor
                                while (resultado.next()) {
                                        idoferta = resultado.getString(2);
                                        idcliente = resultado.getString(3);
                                        idcomercial = resultado.getString(4);
                                        fecha = resultado.getString(5);
                                        resultado2 = sentencia2.executeQuery("SELECT *
FROM persona WHERE id='"+idcliente+"'");
                                        while (resultado2.next())
                                        {
                                                clienteoferta = resultado2.getString(3) + " " +
resultado2.getString(4);
                                                correoclienteoferta = resultado2.getString(6);
                                        }
                                        resultado3 = sentencia3.executeQuery("SELECT *
FROM persona WHERE id='"+idcomercial+"'");
                                        while (resultado3.next())
                                        {
                                                comercialoferta = resultado3.getString(3) + " "
+ resultado3.getString(4);
                                                correocomercialoferta =
resultado3.getString(6);
                                        }
                                        resultado4 = sentencia4.executeQuery("SELECT *
FROM ofertas WHERE id='"+idoferta+"'");
```

```
                                        while (resultado4.next())
                                        {
                                                nombreoferta = resultado4.getString(2);
                                        }


        soluciones+="<tr><td>"+nombreoferta+"</td><td>"+clienteoferta+"</td><td>"+corre
oclienteoferta+"</td><td>"+comercialoferta+"</td><td>"+correocomercialoferta+"</td><td>"
+fecha+"</td></tr>";


                                }
                                soluciones+="</table></body></html>";
                        }
                        else {
                                soluciones = "No hay resultados.";
                        }

                } catch (SQLException e) {
                        e.printStackTrace();
                }
                return soluciones;
        }

        public boolean borrar(String id) {

                try {
            sentencia.executeUpdate("DELETE FROM ofertas WHERE id='"+id+"'");
            return true;

                } catch(SQLException e) {
                        e.printStackTrace();
                        return false;
```

```
            }
        }

        public ArrayList NombreOfertas() {
            nombreofertas = new ArrayList();
            try {
                resultado = sentencia.executeQuery("SELECT * FROM ofertas");

                if (resultado.next() == true) //move o cursor
                {
                    resultado.beforeFirst(); //volve a poñer de primeiro o cursor
                    while (resultado.next()) {
                        nombreoferta = resultado.getString(2); //nombre

                        nombreofertas.add(nombreoferta);
                    }
                }
                else {
                    nombreofertas.add("No hay resultados.");
                }

            } catch (SQLException e) {
                e.printStackTrace();
            }
            return nombreofertas;
        }

        public ArrayList IDOfertas() {
            nombreofertas = new ArrayList();
            try {
                resultado = sentencia.executeQuery("SELECT * FROM ofertas");
```

```
                    if (resultado.next() == true) //move o cursor
                    {
                            resultado.beforeFirst(); //volve a poñer de primeiro o cursor
                            while (resultado.next()) {
                                    idoferta = resultado.getString(1); //id

                                    nombreofertas.add(idoferta);
                            }
                    }
                    else {
                            nombreofertas.add("No hay resultados.");
                    }

            } catch (SQLException e) {
                    e.printStackTrace();
            }
            return nombreofertas;
    }

    public ArrayList ClienteDisponiblesOfertas(String idofe) {
            nombreofertas = new ArrayList();
            try {
                    resultado = sentencia.executeQuery("SELECT * FROM persona WHERE
tipoPersona = '1' AND id NOT IN (SELECT idCliente from oferta_enviada WHERE fechaenvio>
DATE_ADD('"+fechaActual+"', INTERVAL -1 YEAR) AND idOferta='"+idofe+"')");

                    if (resultado.next() == true) //move o cursor
                    {
                            resultado.beforeFirst(); //volve a poñer de primeiro o cursor
                            while (resultado.next()) {
                                    idcliente = resultado.getString(1); //id
```

```
                                        clienteoferta = resultado.getString(3); //nombre
                                        clienteapeloferta = resultado.getString(4); //apellidos
                                        nombreofertas.add(clienteoferta+"
"+clienteapeloferta+" ("+idcliente+")");
                                }
                        }
                        else {
                                nombreofertas.add("No hay resultados.");
                        }

                } catch (SQLException e) {
                        e.printStackTrace();
                }
                return nombreofertas;
        }

  public void cerrarMetodos() {
        try {
                resultado.close();
                resultado2.close();
                resultado3.close();
                resultado4.close();
                        sentencia.close();
                        sentencia2.close();
                        sentencia3.close();
                        sentencia4.close();
                conex.close();
                } catch (SQLException e) {
                        e.printStackTrace();
                }
  }
```

```
}
```

17.9. ITV.java

```
package textual;


import java.sql.Connection;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

import java.util.ArrayList;

import java.util.Calendar;


/**

 * @author Adriana Armental Tomé

 * @version 06.03.2017

 */

public class ITV

{

   protected String soluciones, idvehiculo, marcavehiculo, modelovehiculo, matriculavehiculo,
nombrevehiculo, fecha, fechaActual;

   protected ArrayList nombrevehiculos;


   static ConexionBD con;

   static Connection conex;

   static Statement sentencia, sentencia2;

   static ResultSet resultado, resultado2;


   public ITV()

   {

         String dia2 = "";

                  String mes2 = "";


     Calendar cal = Calendar.getInstance();
```

```
        int dia = cal.get(Calendar.DATE);
        if (dia<10) {
                dia2 = "0"+dia;
        }
        else {
                dia2 = ""+dia;
        }
        int mes = cal.get(Calendar.MONTH) + 1;
        if (mes<10) {
                mes2 = "0"+mes;
        }
        else {
                mes2 = ""+mes;
        }
        int annio = cal.get(Calendar.YEAR);

        fechaActual = annio + "-" + mes2 + "-" + dia2;

    con = new ConexionBD();
    conex = con.getCon();
    try {
                        sentencia = conex.createStatement();
                        sentencia2 = conex.createStatement();
                } catch (SQLException e) {
                        e.printStackTrace();
                }
  }

        public boolean insertar(String idvehiculo) {

        try {
```

```
            String sql = "INSERT INTO revisionitv (`idVehiculo`, `fechaAlta`) VALUES
('"+idvehiculo+"','"+fechaActual+"')";

            sentencia.executeUpdate(sql);

            return true;


            } catch(SQLException e) {

                  e.printStackTrace();

                  return false;

            }

      }


      public String listar() {


            try {

                  resultado = sentencia.executeQuery("SELECT * FROM revisionitv");


                  if (resultado.next() == true) //move o cursor

                  {

                        soluciones =
"<html><body><table><tr><th>VEHÍCULO</th><th>FECHA</th></tr>";

                        resultado.beforeFirst(); //volve a poñer de primeiro o cursor

                        while (resultado.next()) {

                              idvehiculo = resultado.getString(2);

                              fecha = resultado.getString(3);

                              resultado2 = sentencia2.executeQuery("SELECT *
FROM vehiculo WHERE id='"+idvehiculo+"'");

                              while (resultado2.next()) {

                                    nombrevehiculo = resultado2.getString(2) + " "
+ resultado2.getString(3) + " "+resultado2.getString(4);


      soluciones+="<tr><td>"+nombrevehiculo+"</td><td>"+fecha+"</td></tr>";

                              }

                        }
```

```
                    soluciones+="</table></body></html>";
                }
                else {
                    soluciones = "No hay resultados.";
                }

            } catch (SQLException e) {
                e.printStackTrace();
            }
            return soluciones;
        }

        public ArrayList VehiculoID() {
            nombrevehiculos = new ArrayList();
            try {
                resultado = sentencia.executeQuery("SELECT * FROM vehiculo");

                if (resultado.next() == true) //move o cursor
                {
                    resultado.beforeFirst(); //volve a poñer de primeiro o cursor
                    while (resultado.next()) {
                        idvehiculo = resultado.getString(1); //idvehiculo
                        nombrevehiculos.add(idvehiculo);
                    }
                }
                else {
                    nombrevehiculos.add("No hay resultados.");
                }

            } catch (SQLException e) {
                e.printStackTrace();
```

```
        }
        return nombrevehiculos;
}

public ArrayList VehiculoMarca() {
        nombrevehiculos = new ArrayList();
        try {
                resultado = sentencia.executeQuery("SELECT * FROM vehiculo");

                if (resultado.next() == true) //move o cursor
                {
                        resultado.beforeFirst(); //volve a poñer de primeiro o cursor
                        while (resultado.next()) {
                                marcavehiculo = resultado.getString(2); //marca
                                nombrevehiculos.add(marcavehiculo);
                        }
                }
                else {
                        nombrevehiculos.add("No hay resultados.");
                }

        } catch (SQLException e) {
                e.printStackTrace();
        }
        return nombrevehiculos;
}

public ArrayList VehiculoModelo() {
        nombrevehiculos = new ArrayList();
        try {
                resultado = sentencia.executeQuery("SELECT * FROM vehiculo");
```

```
                if (resultado.next() == true) //move o cursor
                {
                        resultado.beforeFirst(); //volve a poñer de primeiro o cursor
                        while (resultado.next()) {
                                modelovehiculo = resultado.getString(3); //modelo
                                nombrevehiculos.add(modelovehiculo);
                        }
                }
                else {
                        nombrevehiculos.add("No hay resultados.");
                }

        } catch (SQLException e) {
                e.printStackTrace();
        }
        return nombrevehiculos;
}

public ArrayList VehiculoMatricula() {
        nombrevehiculos = new ArrayList();
        try {
                resultado = sentencia.executeQuery("SELECT * FROM vehiculo");

                if (resultado.next() == true) //move o cursor
                {
                        resultado.beforeFirst(); //volve a poñer de primeiro o cursor
                        while (resultado.next()) {
                                matriculavehiculo = resultado.getString(4); //matricula
                                nombrevehiculos.add(matriculavehiculo);
                        }
```

```
                        }
                        else {
                                nombrevehiculos.add("No hay resultados.");
                        }

                } catch (SQLException e) {
                        e.printStackTrace();
                }
                return nombrevehiculos;
        }

   public void cerrarMetodos() {
        try {
                resultado.close();
                resultado2.close();
                        sentencia.close();
                        sentencia2.close();
                conex.close();
                } catch (SQLException e) {
                        e.printStackTrace();
                }
   }
}
```

17.10. Boton

```
package graficoprincipal;

import java.awt.Color;
import java.awt.Font;
import java.awt.GradientPaint;
import java.awt.Graphics;
```

```
import java.awt.Graphics2D;

import javax.swing.JButton;

/**
 * @author Adriana Armental Tomé
 * @version 06.03.2017
 */
public class Boton extends JButton{

	int x, y;

	//Constructor forma rectangulo bordes curvos con un tamanho e colores dado
	public Boton(String rotulo, int x, int y) {
		super(rotulo);
		this.x = x;
		this.y = y;
		setContentAreaFilled(false); //que non se pinte o que "sobra"
	}

	@Override
	protected void paintComponent(Graphics g) {
		//Colores para o gradiante
		Color blanco = new Color(255, 255, 255);
		Color negro = new Color(0, 0, 0);
		//Cambiar letra
		g.setFont(new Font("Lucida Sans", Font.BOLD, 26));
		//Gradiente
		Graphics2D g2d=(Graphics2D)g;
		g2d.setPaint(new GradientPaint(500, 0, blanco, 200, 400, negro));
		if (getModel().isArmed()) {	//si se pulsa cambiaselle o color
```

```
                g.setColor(blanco);
            }
            //pintase o boton
            g2d.fillRoundRect(0, 0, getSize().width - 1, getSize().height - 1, x, y);
            super.paintComponent(g);
        }

        //Sobreescritura del borde
        @Override
        protected void paintBorder(Graphics g) {
            g.setColor(Color.black);
            g.drawRoundRect(0, 0, getSize().width - 1, getSize().height - 1, x, y);
        }
}
```

17.11. PanelImagen.java

```
package graficoprincipal;

import java.awt.Dimension;
import java.awt.Graphics;
import javax.swing.ImageIcon;
import javax.swing.JPanel;

/**
 * @author Adriana Armental Tomé
 * @version 06.03.2017
 */
public class PanelImagen extends JPanel {
        public PanelImagen(){
            this.setSize(1000,900);
        }
```

```
        @Override
        public void paintComponent (Graphics g){
                Dimension tamanio = getSize(); //Tamanho que se lle vai dar a imaxe(toda a
ventana)
                //Collese a imagen
                ImageIcon imagenFondo = new
ImageIcon(getClass().getResource("/imagenes/fondoprincipal.jpg"));
                //Debuxase na posicion asignada
                g.drawImage(imagenFondo.getImage(),0,0,tamanio.width, tamanio.height,
null);
                //Que non sea opaca
                setOpaque(false);
                super.paintComponent(g);
        }
}
```

17.12. MenuPrincipal.java

```
package graficoprincipal;

import java.awt.Color;
import java.awt.Font;
import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;
import java.awt.Insets;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;

import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
```

```
import graficofichas.MenuFichas;

/**
 * @author Adriana Armental Tomé
 * @version 06.03.2017
 */

public class MenuPrincipal extends JFrame implements KeyListener, ActionListener {

        //Botons menu
        Boton personas, vehiculos, fichas, ofertas, salir;
        Font fuente;
        JPanel panel;
        JLabel label;

        public MenuPrincipal(){
                super("Taller"); //Nome
    //declarar e colocar fondo
    PanelImagen p = new PanelImagen(); //Panel que conten a imaxe
    setContentPane(p); //Asignar panel
    setSize(800,700); //Tamanho ventana
                setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); //Poder cerrar a ventana.
    setResizable(false); //Que non se poda cambiar o tamaanho da ventana
    setLayout(new GridBagLayout()); //Distribucion da ventana
    setLocationRelativeTo(null); //Colocar a ventana no centro

    //Crear botons e ponher a escoita
    personas = new Boton("PERSONAS", 200, 110);
    personas.setForeground(Color.BLACK); //cambiar color de letras
    personas.setFocusPainted(false); //para que non sala o cuadro o redor das letras
```

```
vehiculos = new Boton("VEHÍCULOS", 200, 110);
vehiculos.setForeground(Color.BLACK);
vehiculos.setFocusPainted(false);
                fichas = new Boton("FICHAS", 200, 110);
                fichas.setForeground(Color.BLACK);
                fichas.setFocusPainted(false);
                ofertas = new Boton("PROMOCIONES", 200, 110);
                ofertas.setForeground(Color.BLACK);
                ofertas.setFocusPainted(false);
                salir = new Boton("SALIR", 200, 110);
                salir.setForeground(Color.BLACK);
                salir.setFocusPainted(false);
                personas.addActionListener(this);
                vehiculos.addActionListener(this);
                fichas.addActionListener(this);
                ofertas.addActionListener(this);
                salir.addActionListener(this);

                //Declaro colores para o fondo e as etiquetas dos paneles
Color co= new Color(0);
Color col= new Color(255,255,255);

//Creanse paneles e daselle un color de fondo
 panel = new JPanel();
 panel.setBackground(co);

 label = new JLabel("BIENVENIDO");
 label.setFont(new Font("Lucida Sans", Font.BOLD, 20));
 label.setForeground(col);

                //Añadense as etiquetas os paneles
```

```
panel.add(label);

GridBagConstraints c = new GridBagConstraints();
c.gridx=0; // especifica a coordenada x
c.gridy=0;         // coordenada y
c.gridwidth=1; //numero de celdas que ocupa no GridBagLayout
c.gridheight=1;
c.anchor=GridBagConstraints.CENTER; //posicion dentro dunha celda
c.fill=GridBagConstraints.BOTH; //espacio que ocupara dentro dunha celda
c.weightx=1.0; //porcentaxe de espazo libre que ocupara
c.weighty=0;
c.insets=new Insets(10,100,10,100); //ponher marxes
add(panel,c);

c.gridx=0; // especifica a coordenada x
c.gridy=1;         // coordenada y
c.gridwidth=1; //numero de celdas que ocupa no GridBagLayout
c.gridheight=1;
c.anchor=GridBagConstraints.CENTER; //posicion dentro dunha celda
c.fill=GridBagConstraints.BOTH; //espacio que ocupara dentro dunha celda
c.weightx=1.0; //porcentaxe de espazo libre que ocupara
c.weighty=1.0;
c.insets=new Insets(20,200,20,200); //ponher marxes
add(personas,c);

c.gridx=0;
c.gridy=2;
c.gridwidth=1;
c.gridheight=1;
c.anchor=GridBagConstraints.CENTER;
c.fill=GridBagConstraints.BOTH;
```

```
c.weightx=1.0;
c.weighty=1.0;
c.insets=new Insets(20,200,20,200);
add(vehiculos,c);

c.gridx=0;
c.gridy=3;
c.gridwidth=1;
c.gridheight=1;
c.anchor=GridBagConstraints.CENTER;
c.fill=GridBagConstraints.BOTH;
c.weightx=1.0;
c.weighty=1.0;
c.insets=new Insets(20,200,20,200);
add(fichas,c);

c.gridx=0;
c.gridy=4;
c.gridwidth=1;
c.gridheight=1;
c.anchor=GridBagConstraints.CENTER;
c.fill=GridBagConstraints.BOTH;
c.weightx=1.0;
c.weighty=1.0;
c.insets=new Insets(20,200,20,200);
add(ofertas,c);

c.gridx=0;
c.gridy=5;
c.gridwidth=1;
c.gridheight=1;
```

```
        c.anchor=GridBagConstraints.CENTER;
        c.fill=GridBagConstraints.BOTH;
        c.weightx=1.0;
        c.weighty=1.0;
        c.insets=new Insets(20,200,20,200);
        add(salir,c);
}

@Override
public void keyTyped(KeyEvent e) {}

//control de espacio
@Override
public void keyPressed(KeyEvent e) {
        int key = e.getKeyCode(); //collese a tecla soltada

        if(key == KeyEvent.VK_SPACE) { //si e o espacio
                e.consume();
        }
}

@Override
public void keyReleased(KeyEvent e) {
}

@Override
public void actionPerformed(ActionEvent e) {
        if (e.getActionCommand().equals("PERSONAS")) {
                MenuPersonas per = new MenuPersonas();
                dispose(); //elimina ventana
                per.setVisible(true);
```

```
            //para desactivar o espacio no control de menu
            JPanel panel1 = (JPanel) per.getContentPane();

            panel1.addKeyListener(this);
            panel1.setFocusable(true);
        }
        if (e.getActionCommand().equals("VEHÍCULOS")) {
            MenuVehiculos veh = new MenuVehiculos();
            dispose(); //elimina ventana
            veh.setVisible(true);

            //para desactivar o espacio no control de menu
            JPanel panel1 = (JPanel) veh.getContentPane();

            panel1.addKeyListener(this);
            panel1.setFocusable(true);
        }
        if (e.getActionCommand().equals("FICHAS")) {
            MenuFichas fic = new MenuFichas();
            dispose(); //elimina ventana
            fic.setVisible(true);

            //para desactivar o espacio no control de menu
            JPanel panel1 = (JPanel) fic.getContentPane();

            panel1.addKeyListener(this);
            panel1.setFocusable(true);
        }
        if (e.getActionCommand().equals("PROMOCIONES")) {
            MenuPromociones ofe = new MenuPromociones();
```

```
                dispose(); //elimina ventana
                ofe.setVisible(true);

                //para desactivar o espacio no control de menu
                JPanel panel1 = (JPanel) ofe.getContentPane();

                panel1.addKeyListener(this);
                panel1.setFocusable(true);
            }
            if (e.getActionCommand().equals("SALIR")) {
                System.exit(0); //Pechar programa
            }
        }
}
```

17.13. MenuPersonas.java

```
package graficoprincipal;

import java.awt.Color;
import java.awt.Font;
import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;
import java.awt.Insets;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;

import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
```

```
import graficoclientes.MenuClientes;
import graficocomerciales.MenuComerciales;
import graficomecanicos.MenuMecanicos;

/**
 * @author Adriana Armental Tomé
 * @version 06.03.2017
 */

public class MenuPersonas extends JFrame implements KeyListener, ActionListener {

        //Botons menu
        Boton clientes, comerciales, mecanicos, atras;
        Font fuente;
        JPanel panel;
        JLabel label;

        public MenuPersonas(){
                super("Taller"); //Nome
    //declarar e colocar fondo
    PanelImagen p = new PanelImagen(); //Panel que conten a imaxe
    setContentPane(p); //Asignar panel
    setSize(800,700); //Tamanho ventana
                setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); //Poder cerrar a ventana.
    setResizable(false); //Que non se poda cambiar o tamaanho da ventana
    setLayout(new GridBagLayout()); //Distribucion da ventana
    setLocationRelativeTo(null); //Colocar a ventana no centro

    //Crear botons e ponher a escoita
    clientes = new Boton("CLIENTES", 200, 110);
```

```
clientes.setForeground(Color.BLACK); //cambiar color de letras
clientes.setFocusPainted(false); //para que non sala o cuadro o redor das letras
comerciales = new Boton("COMERCIALES", 200, 110);
comerciales.setForeground(Color.BLACK);
comerciales.setFocusPainted(false);
mecanicos = new Boton("MECÁNICOS", 200, 110);
mecanicos.setForeground(Color.BLACK);
mecanicos.setFocusPainted(false);
atras = new Boton("ATRÁS", 200, 110);
atras.setForeground(Color.BLACK);
                atras.setFocusPainted(false);
                clientes.addActionListener(this);
                comerciales.addActionListener(this);
                mecanicos.addActionListener(this);
                atras.addActionListener(this);

                //Declaro colores para o fondo e as etiquetas dos paneles
Color co= new Color(0);
Color col= new Color(255,255,255);

//Creanse paneles e daselle un color de fondo
panel = new JPanel();
panel.setBackground(co);

label = new JLabel("GESTIÓN DE PERSONAS");
label.setFont(new Font("Lucida Sans", Font.BOLD, 20));
label.setForeground(col);

                //Añadense as etiquetas os paneles
                panel.add(label);
```

```
GridBagConstraints c = new GridBagConstraints();
c.gridx=0; // especifica a coordenada x
c.gridy=0;         // coordenada y
c.gridwidth=1; //numero de celdas que ocupa no GridBagLayout
c.gridheight=1;
c.anchor=GridBagConstraints.CENTER; //posicion dentro dunha celda
c.fill=GridBagConstraints.BOTH; //espacio que ocupara dentro dunha celda
c.weightx=1.0; //porcentaxe de espazo libre que ocupara
c.weighty=0;
c.insets=new Insets(10,100,10,100); //ponher marxes
add(panel,c);

c.gridx=0; // especifica a coordenada x
c.gridy=1;         // coordenada y
c.gridwidth=1; //numero de celdas que ocupa no GridBagLayout
c.gridheight=1;
c.anchor=GridBagConstraints.CENTER; //posicion dentro dunha celda
c.fill=GridBagConstraints.BOTH; //espacio que ocupara dentro dunha celda
c.weightx=1.0; //porcentaxe de espazo libre que ocupara
c.weighty=1.0;
c.insets=new Insets(20,200,20,200); //ponher marxes
add(clientes,c);

c.gridx=0;
c.gridy=2;
c.gridwidth=1;
c.gridheight=1;
c.anchor=GridBagConstraints.CENTER;
c.fill=GridBagConstraints.BOTH;
c.weightx=1.0;
c.weighty=1.0;
```

```
        c.insets=new Insets(20,200,20,200);
        add(comerciales,c);

        c.gridx=0;
        c.gridy=3;
        c.gridwidth=1;
        c.gridheight=1;
        c.anchor=GridBagConstraints.CENTER;
        c.fill=GridBagConstraints.BOTH;
        c.weightx=1.0;
        c.weighty=1.0;
        c.insets=new Insets(20,200,20,200);
        add(mecanicos,c);

        c.gridx=0;
        c.gridy=4;
        c.gridwidth=1;
        c.gridheight=1;
        c.anchor=GridBagConstraints.CENTER;
        c.fill=GridBagConstraints.BOTH;
        c.weightx=1.0;
        c.weighty=1.0;
        c.insets=new Insets(20,200,20,200);
        add(atras,c);

}

@Override
public void keyTyped(KeyEvent e) {}

//control de espacio
```

```
	@Override
	public void keyPressed(KeyEvent e) {
		int key = e.getKeyCode(); //collese a tecla soltada

		if(key == KeyEvent.VK_SPACE) { //si e o espacio
			e.consume();
		}
	}

	@Override
	public void keyReleased(KeyEvent e) {
	}

	@Override
	public void actionPerformed(ActionEvent e) {
		if (e.getActionCommand().equals("CLIENTES")) {
			MenuClientes cli = new MenuClientes();
			dispose();
			cli.setVisible(true);

			//para desactivar o espacio no control de menu
			JPanel panel = (JPanel) cli.getContentPane();

			panel.addKeyListener(this);
			panel.setFocusable(true);

		}
		if (e.getActionCommand().equals("COMERCIALES")) {
			MenuComerciales come = new MenuComerciales();
			dispose();
			come.setVisible(true);
```

```
            //para desactivar o espacio no control de menu
            JPanel panel = (JPanel) come.getContentPane();

            panel.addKeyListener(this);
            panel.setFocusable(true);

}
if (e.getActionCommand().equals("MECÁNICOS")) {
            MenuMecanicos meca = new MenuMecanicos();
            dispose();
            meca.setVisible(true);

            //para desactivar o espacio no control de menu
            JPanel panel = (JPanel) meca.getContentPane();

            panel.addKeyListener(this);
            panel.setFocusable(true);
}

if (e.getActionCommand().equals("ATRÁS")) {
            MenuPrincipal men = new MenuPrincipal();
            dispose();
            men.setVisible(true);

            //para desactivar o espacio no control de menu
            JPanel panel = (JPanel) men.getContentPane();

            panel.addKeyListener(this);
            panel.setFocusable(true);
}
```

```
        }
}
```

17.14. MenuVehiculos.java

```
package graficoprincipal;

import java.awt.Color;
import java.awt.Font;
import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;
import java.awt.Insets;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;

import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;

import graficoautobuses.MenuAutobuses;
import graficocamiones.MenuCamiones;
import graficofurgonetas.MenuFurgonetas;

/**
 * @author Adriana Armental Tomé
 * @version 06.03.2017
 */

public class MenuVehiculos extends JFrame implements KeyListener, ActionListener {
```

```
    //Botons menu
    Boton coches, motos, camiones, furgonetas, autobuses, profesionales, atras;
    Font fuente;
    JPanel panel;
    JLabel label;

    public MenuVehiculos(){
            super("Taller"); //Nome
//declarar e colocar fondo
PanelImagen p = new PanelImagen(); //Panel que conten a imaxe
setContentPane(p); //Asignar panel
setSize(800,700); //Tamanho ventana
            setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); //Poder cerrar a ventana.
setResizable(false); //Que non se poda cambiar o tamaanho da ventana
setLayout(new GridBagLayout()); //Distribucion da ventana
setLocationRelativeTo(null); //Colocar a ventana no centro

//Crear botons e ponher a escoita
coches = new Boton("COCHES", 200, 110);
coches.setForeground(Color.BLACK); //cambiar color de letras
coches.setFocusPainted(false); //para que non sala o cuadro o redor das letras
motos = new Boton("MOTOS", 200, 110);
motos.setForeground(Color.BLACK);
motos.setFocusPainted(false);
camiones = new Boton("CAMIONES", 200, 110);
camiones.setForeground(Color.BLACK);
camiones.setFocusPainted(false);
furgonetas = new Boton("FURGONETAS", 200, 110);
furgonetas.setForeground(Color.BLACK);
furgonetas.setFocusPainted(false);
autobuses = new Boton("AUTOBUSES", 200, 110);
```

```
        autobuses.setForeground(Color.BLACK);
        autobuses.setFocusPainted(false);
        profesionales = new Boton("PROFESIONALES", 200, 110);
        profesionales.setForeground(Color.BLACK);
        profesionales.setFocusPainted(false);
        atras = new Boton("ATRÁS", 200, 110);
        atras.setForeground(Color.BLACK);
                        atras.setFocusPainted(false);
                        coches.addActionListener(this);
                        motos.addActionListener(this);
                        camiones.addActionListener(this);
                        furgonetas.addActionListener(this);
                        autobuses.addActionListener(this);
                        profesionales.addActionListener(this);
                        atras.addActionListener(this);

                        //Declaro colores para o fondo e as etiquetas dos paneles
        Color co= new Color(0);
        Color col= new Color(255,255,255);

       //Creanse paneles e daselle un color de fondo
        panel = new JPanel();
        panel.setBackground(co);

        label = new JLabel("GESTIÓN DE VEHÍCULOS");
        label.setFont(new Font("Lucida Sans", Font.BOLD, 20));
        label.setForeground(col);

                        //Añadense as etiquetas os paneles
                        panel.add(label);
```

```
GridBagConstraints c = new GridBagConstraints();
c.gridx=0; // especifica a coordenada x
c.gridy=0;          // coordenada y
c.gridwidth=1; //numero de celdas que ocupa no GridBagLayout
c.gridheight=1;
c.anchor=GridBagConstraints.CENTER; //posicion dentro dunha celda
c.fill=GridBagConstraints.BOTH; //espacio que ocupara dentro dunha celda
c.weightx=1.0; //porcentaxe de espazo libre que ocupara
c.weighty=0;
c.insets=new Insets(10,100,10,100); //ponher marxes
add(panel,c);

c.gridx=0; // especifica a coordenada x
c.gridy=1;          // coordenada y
c.gridwidth=1; //numero de celdas que ocupa no GridBagLayout
c.gridheight=1;
c.anchor=GridBagConstraints.CENTER; //posicion dentro dunha celda
c.fill=GridBagConstraints.BOTH; //espacio que ocupara dentro dunha celda
c.weightx=1.0; //porcentaxe de espazo libre que ocupara
c.weighty=1.0;
c.insets=new Insets(20,200,20,200); //ponher marxes
add(coches,c);

c.gridx=0;
c.gridy=2;
c.gridwidth=1;
c.gridheight=1;
c.anchor=GridBagConstraints.CENTER;
c.fill=GridBagConstraints.BOTH;
c.weightx=1.0;
c.weighty=1.0;
```

```
c.insets=new Insets(20,200,20,200);
add(motos,c);

c.gridx=0;
c.gridy=3;
c.gridwidth=1;
c.gridheight=1;
c.anchor=GridBagConstraints.CENTER;
c.fill=GridBagConstraints.BOTH;
c.weightx=1.0;
c.weighty=1.0;
c.insets=new Insets(20,200,20,200);
add(camiones,c);

c.gridx=0;
c.gridy=4;
c.gridwidth=1;
c.gridheight=1;
c.anchor=GridBagConstraints.CENTER;
c.fill=GridBagConstraints.BOTH;
c.weightx=1.0;
c.weighty=1.0;
c.insets=new Insets(20,200,20,200);
add(furgonetas,c);

c.gridx=0;
c.gridy=5;
c.gridwidth=1;
c.gridheight=1;
c.anchor=GridBagConstraints.CENTER;
c.fill=GridBagConstraints.BOTH;
```

```
        c.weightx=1.0;
        c.weighty=1.0;
        c.insets=new Insets(20,200,20,200);
        add(autobuses,c);

        c.gridx=0;
        c.gridy=6;
        c.gridwidth=1;
        c.gridheight=1;
        c.anchor=GridBagConstraints.CENTER;
        c.fill=GridBagConstraints.BOTH;
        c.weightx=1.0;
        c.weighty=1.0;
        c.insets=new Insets(20,200,20,200);
        add(profesionales,c);

        c.gridx=0;
        c.gridy=7;
        c.gridwidth=1;
        c.gridheight=1;
        c.anchor=GridBagConstraints.CENTER;
        c.fill=GridBagConstraints.BOTH;
        c.weightx=1.0;
        c.weighty=1.0;
        c.insets=new Insets(20,200,20,200);
        add(atras,c);

}

@Override
public void keyTyped(KeyEvent e) {}
```

```
//control de espacio
@Override
public void keyPressed(KeyEvent e) {
	int key = e.getKeyCode(); //collese a tecla soltada

	if(key == KeyEvent.VK_SPACE) { //si e o espacio
		e.consume();
	}
}

@Override
public void keyReleased(KeyEvent e) {
}

@Override
public void actionPerformed(ActionEvent e) {
	if (e.getActionCommand().equals("COCHES")) {
		MenuCoches coc = new MenuCoches();
		dispose();
		coc.setVisible(true);

		//para desactivar o espacio no control de menu
		JPanel panel = (JPanel) coc.getContentPane();

		panel.addKeyListener(this);
		panel.setFocusable(true);

	}
	if (e.getActionCommand().equals("MOTOS")) {
		MenuMotos mot = new MenuMotos();
```

```
            dispose();
            mot.setVisible(true);

            //para desactivar o espacio no control de menu
            JPanel panel = (JPanel) mot.getContentPane();

            panel.addKeyListener(this);
            panel.setFocusable(true);

    }
    if (e.getActionCommand().equals("CAMIONES")) {
            MenuCamiones cam = new MenuCamiones();
            dispose();
            cam.setVisible(true);

            //para desactivar o espacio no control de menu
            JPanel panel = (JPanel) cam.getContentPane();

            panel.addKeyListener(this);
            panel.setFocusable(true);
    }

    if (e.getActionCommand().equals("FURGONETAS")) {
            MenuFurgonetas fur = new MenuFurgonetas();
            dispose();
            fur.setVisible(true);

            //para desactivar o espacio no control de menu
            JPanel panel = (JPanel) fur.getContentPane();

            panel.addKeyListener(this);
```

```
                panel.setFocusable(true);
        }

        if (e.getActionCommand().equals("AUTOBUSES")) {
                MenuAutobuses aut = new MenuAutobuses();
                dispose();
                aut.setVisible(true);

                //para desactivar o espacio no control de menu
                JPanel panel = (JPanel) aut.getContentPane();

                panel.addKeyListener(this);
                panel.setFocusable(true);
        }

        if (e.getActionCommand().equals("PROFESIONALES")) {
                MenuProfesionales pro = new MenuProfesionales();
                dispose();
                pro.setVisible(true);

                //para desactivar o espacio no control de menu
                JPanel panel = (JPanel) pro.getContentPane();

                panel.addKeyListener(this);
                panel.setFocusable(true);
        }

        if (e.getActionCommand().equals("ATRÁS")) {
                MenuPrincipal men = new MenuPrincipal();
                dispose();
                men.setVisible(true);
```

```
                        //para desactivar o espacio no control de menu
                        JPanel panel = (JPanel) men.getContentPane();

                        panel.addKeyListener(this);
                        panel.setFocusable(true);
                    }
                }
            }
```

17.15. MenuPromociones.java

```
package graficoprincipal;

import java.awt.Color;
import java.awt.Font;
import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;
import java.awt.Insets;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;

import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;

import graficoitv.MenuITV;
import graficoofertas.MenuOfertas;

/**
```

```
 * @author Adriana Armental Tomé
 * @version 06.03.2017
 */

public class MenuPromociones extends JFrame implements KeyListener, ActionListener {

	//Botons menu
	Boton ofertas, itv, atras;
	Font fuente;
	JPanel panel;
	JLabel label;

	public MenuPromociones(){
		super("Taller"); //Nome
	//declarar e colocar fondo
	PanelImagen p = new PanelImagen(); //Panel que conten a imaxe
	setContentPane(p); //Asignar panel
	setSize(800,700); //Tamanho ventana
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); //Poder cerrar a ventana.
	setResizable(false); //Que non se poda cambiar o tamaanho da ventana
	setLayout(new GridBagLayout()); //Distribucion da ventana
	setLocationRelativeTo(null); //Colocar a ventana no centro

	//Crear botons e ponher a escoita
	ofertas = new Boton("OFERTAS", 200, 110);
	ofertas.setForeground(Color.BLACK); //cambiar color de letras
	ofertas.setFocusPainted(false); //para que non sala o cuadro o redor das letras
	itv = new Boton("REVISIÓN ITV", 200, 110);
	itv.setForeground(Color.BLACK);
	itv.setFocusPainted(false);
	atras = new Boton("ATRÁS", 200, 110);
```

```
 atras.setForeground(Color.BLACK);
                atras.setFocusPainted(false);
                ofertas.addActionListener(this);
                itv.addActionListener(this);
                atras.addActionListener(this);

                //Declaro colores para o fondo e as etiquetas dos paneles
 Color co= new Color(0);
 Color col= new Color(255,255,255);

//Creanse paneles e daselle un color de fondo
 panel = new JPanel();
 panel.setBackground(co);

 label = new JLabel("GESTIÓN DE MOTOS");
 label.setFont(new Font("Lucida Sans", Font.BOLD, 20));
 label.setForeground(col);

                //Añadense as etiquetas os paneles
                panel.add(label);

                GridBagConstraints c = new GridBagConstraints();
                c.gridx=0; // especifica a coordenada x
                c.gridy=0;          // coordenada y
                c.gridwidth=1; //numero de celdas que ocupa no GridBagLayout
                c.gridheight=1;
                c.anchor=GridBagConstraints.CENTER; //posicion dentro dunha celda
                c.fill=GridBagConstraints.BOTH; //espacio que ocupara dentro dunha celda
                c.weightx=1.0; //porcentaxe de espazo libre que ocupara
                c.weighty=0;
                c.insets=new Insets(10,100,10,100); //ponher marxes
```

```
add(panel,c);

c.gridx=0; // especifica a coordenada x
c.gridy=1;          // coordenada y
c.gridwidth=1; //numero de celdas que ocupa no GridBagLayout
c.gridheight=1;
c.anchor=GridBagConstraints.CENTER; //posicion dentro dunha celda
c.fill=GridBagConstraints.BOTH; //espacio que ocupara dentro dunha celda
c.weightx=1.0; //porcentaxe de espazo libre que ocupara
c.weighty=1.0;
c.insets=new Insets(20,200,20,200); //ponher marxes
add(ofertas,c);

c.gridx=0;
c.gridy=2;
c.gridwidth=1;
c.gridheight=1;
c.anchor=GridBagConstraints.CENTER;
c.fill=GridBagConstraints.BOTH;
c.weightx=1.0;
c.weighty=1.0;
c.insets=new Insets(20,200,20,200);
add(itv,c);

c.gridx=0;
c.gridy=3;
c.gridwidth=1;
c.gridheight=1;
c.anchor=GridBagConstraints.CENTER;
c.fill=GridBagConstraints.BOTH;
c.weightx=1.0;
```

```
        c.weighty=1.0;
        c.insets=new Insets(20,200,20,200);
        add(atras,c);

}

@Override
public void keyTyped(KeyEvent e) {}

//control de espacio
@Override
public void keyPressed(KeyEvent e) {
        int key = e.getKeyCode(); //collese a tecla soltada

        if(key == KeyEvent.VK_SPACE) { //si e o espacio
                e.consume();
        }
}

@Override
public void keyReleased(KeyEvent e) {
}

@Override
public void actionPerformed(ActionEvent e) {
        if (e.getActionCommand().equals("OFERTAS")) {
                MenuOfertas men = new MenuOfertas();
                dispose();
                men.setVisible(true);

                //para desactivar o espacio no control de menu
```

```
                JPanel panel = (JPanel) men.getContentPane();

                panel.addKeyListener(this);
                panel.setFocusable(true);

            }
            if (e.getActionCommand().equals("REVISIÓN ITV")) {
                MenuITV men = new MenuITV();
                dispose();
                men.setVisible(true);

                //para desactivar o espacio no control de menu
                JPanel panel = (JPanel) men.getContentPane();

                panel.addKeyListener(this);
                panel.setFocusable(true);

            }

            if (e.getActionCommand().equals("ATRÁS")) {
                MenuVehiculos men = new MenuVehiculos();
                dispose();
                men.setVisible(true);

                //para desactivar o espacio no control de menu
                JPanel panel = (JPanel) men.getContentPane();

                panel.addKeyListener(this);
                panel.setFocusable(true);
            }
        }
```

```
}
```

17.16. MenuClientes.java

```
package graficoclientes;

import java.awt.Color;
import java.awt.Font;
import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;
import java.awt.Insets;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;

import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;

import graficoprincipal.Boton;
import graficoprincipal.MenuPersonas;
import graficoprincipal.PanelImagen;

/**
 * @author Adriana Armental Tomé
 * @version 06.03.2017
 */

public class MenuClientes extends JFrame implements KeyListener, ActionListener {

	//Botons menu
```

```
        Boton nuevo, consulta, editar, eliminar, atras;
        Font fuente;
        JPanel panel;
        JLabel label;

        public MenuClientes(){
                super("Taller"); //Nome
    //declarar e colocar fondo
    PanelImagen p = new PanelImagen(); //Panel que conten a imaxe
    setContentPane(p); //Asignar panel
    setSize(800,700); //Tamanho ventana
                setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); //Poder cerrar a ventana.
    setResizable(false); //Que non se poda cambiar o tamaanho da ventana
    setLayout(new GridBagLayout()); //Distribucion da ventana
    setLocationRelativeTo(null); //Colocar a ventana no centro

    //Crear botons e ponher a escoita
    nuevo = new Boton("NUEVO", 200, 110);
    nuevo.setForeground(Color.BLACK); //cambiar color de letras
    nuevo.setFocusPainted(false); //para que non sala o cuadro o redor das letras
    consulta = new Boton("CONSULTA", 200, 110);
    consulta.setForeground(Color.BLACK);
    consulta.setFocusPainted(false);
    editar = new Boton("EDITAR", 200, 110);
    editar.setForeground(Color.BLACK);
    editar.setFocusPainted(false);
    eliminar = new Boton("ELIMINAR", 200, 110);
    eliminar.setForeground(Color.BLACK);
    eliminar.setFocusPainted(false);
    atras = new Boton("ATRÁS", 200, 110);
    atras.setForeground(Color.BLACK);
```

```
                atras.setFocusPainted(false);
                nuevo.addActionListener(this);
                consulta.addActionListener(this);
                editar.addActionListener(this);
                eliminar.addActionListener(this);
                atras.addActionListener(this);

                //Declaro colores para o fondo e as etiquetas dos paneles
 Color co= new Color(0);
 Color col= new Color(255,255,255);

//Creanse paneles e daselle un color de fondo
 panel = new JPanel();
 panel.setBackground(co);

 label = new JLabel("GESTIÓN DE CLIENTES");
 label.setFont(new Font("Lucida Sans", Font.BOLD, 20));
 label.setForeground(col);

                //Añadense as etiquetas os paneles
                panel.add(label);

                GridBagConstraints c = new GridBagConstraints();
                c.gridx=0; // especifica a coordenada x
                c.gridy=0;          // coordenada y
                c.gridwidth=1; //numero de celdas que ocupa no GridBagLayout
                c.gridheight=1;
                c.anchor=GridBagConstraints.CENTER; //posicion dentro dunha celda
                c.fill=GridBagConstraints.BOTH; //espacio que ocupara dentro dunha celda
                c.weightx=1.0; //porcentaxe de espazo libre que ocupara
                c.weighty=0;
```

```
c.insets=new Insets(10,100,10,100); //ponher marxes
add(panel,c);

c.gridx=0; // especifica a coordenada x
c.gridy=1;          // coordenada y
c.gridwidth=1; //numero de celdas que ocupa no GridBagLayout
c.gridheight=1;
c.anchor=GridBagConstraints.CENTER; //posicion dentro dunha celda
c.fill=GridBagConstraints.BOTH; //espacio que ocupara dentro dunha celda
c.weightx=1.0; //porcentaxe de espazo libre que ocupara
c.weighty=1.0;
c.insets=new Insets(20,200,20,200); //ponher marxes
add(nuevo,c);

c.gridx=0;
c.gridy=2;
c.gridwidth=1;
c.gridheight=1;
c.anchor=GridBagConstraints.CENTER;
c.fill=GridBagConstraints.BOTH;
c.weightx=1.0;
c.weighty=1.0;
c.insets=new Insets(20,200,20,200);
add(consulta,c);

c.gridx=0;
c.gridy=3;
c.gridwidth=1;
c.gridheight=1;
c.anchor=GridBagConstraints.CENTER;
c.fill=GridBagConstraints.BOTH;
```

```
        c.weightx=1.0;
        c.weighty=1.0;
        c.insets=new Insets(20,200,20,200);
        add(editar,c);

        c.gridx=0;
        c.gridy=4;
        c.gridwidth=1;
        c.gridheight=1;
        c.anchor=GridBagConstraints.CENTER;
        c.fill=GridBagConstraints.BOTH;
        c.weightx=1.0;
        c.weighty=1.0;
        c.insets=new Insets(20,200,20,200);
        add(eliminar,c);

        c.gridx=0;
        c.gridy=5;
        c.gridwidth=1;
        c.gridheight=1;
        c.anchor=GridBagConstraints.CENTER;
        c.fill=GridBagConstraints.BOTH;
        c.weightx=1.0;
        c.weighty=1.0;
        c.insets=new Insets(20,200,20,200);
        add(atras,c);

}

@Override
public void keyTyped(KeyEvent e) {}
```

```
//control de espacio
@Override
public void keyPressed(KeyEvent e) {
	int key = e.getKeyCode(); //collese a tecla soltada

	if(key == KeyEvent.VK_SPACE) { //si e o espacio
		e.consume();
	}
}

@Override
public void keyReleased(KeyEvent e) {
}

@Override
public void actionPerformed(ActionEvent e) {
	if (e.getActionCommand().equals("NUEVO")) {
		MenuNuevoCliente con = new MenuNuevoCliente();
		dispose(); //elimina ventana
		con.setVisible(true);

		//para desactivar o espacio no control de menu
		JPanel panel1 = (JPanel) con.getContentPane();

		panel1.addKeyListener(this);
		panel1.setFocusable(true);

	}
	if (e.getActionCommand().equals("CONSULTA")) {
		MenuConsultaClientes con = new MenuConsultaClientes();
```

```
            dispose(); //elimina ventana
            con.setVisible(true);

            //para desactivar o espacio no control de menu
            JPanel panel1 = (JPanel) con.getContentPane();

            panel1.addKeyListener(this);
            panel1.setFocusable(true);

    }
    if (e.getActionCommand().equals("EDITAR")) {
            MenuEditarClienteLista con = new MenuEditarClienteLista();
            dispose(); //elimina ventana
            con.setVisible(true);

            //para desactivar o espacio no control de menu
            JPanel panel1 = (JPanel) con.getContentPane();

            panel1.addKeyListener(this);
            panel1.setFocusable(true);

    }
    if (e.getActionCommand().equals("ELIMINAR")) {
            MenuEliminarCliente con = new MenuEliminarCliente();
            dispose(); //elimina ventana
            con.setVisible(true);

            //para desactivar o espacio no control de menu
            JPanel panel1 = (JPanel) con.getContentPane();

            panel1.addKeyListener(this);
```

```
                panel1.setFocusable(true);


            }

            if (e.getActionCommand().equals("ATRÁS")) {
                MenuPersonas men = new MenuPersonas();
                dispose();
                men.setVisible(true);

                //para desactivar o espacio no control de menu
                JPanel panel = (JPanel) men.getContentPane();

                panel.addKeyListener(this);
                panel.setFocusable(true);
            }
        }
}
```

17.17. MenuNuevoCliente.java

```
package graficoclientes;

import java.awt.Color;
import java.awt.FlowLayout;
import java.awt.Font;
import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;
import java.awt.Insets;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;
```

```
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JTextField;

import graficoprincipal.Boton;
import graficoprincipal.PanelImagen;
import textual.Cliente;
import textual.Persona;

/**
 * @author Adriana Armental Tomé
 * @version 06.03.2017
 */

public class MenuNuevoCliente extends JFrame implements KeyListener, ActionListener {

	//Botons menu
	Boton atras, enviar;
	Font fuente;
	JPanel panel, panel2, panel3, panel4;
	JLabel label, dni, nombre, apellidos, telefono, correo, direccion;
	JTextField inputdni, inputnombre, inputapellidos, inputtelefono, inputcorreo,
inputdireccion;
	Persona cl;

	public MenuNuevoCliente(){
		super("Taller"); //Nome
      //declarar e colocar fondo
      PanelImagen p = new PanelImagen(); //Panel que conten a imaxe
```

```
        setContentPane(p); //Asignar panel
        setSize(800,700); //Tamanho ventana
                        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); //Poder cerrar a ventana.
        setResizable(false); //Que non se poda cambiar o tamaanho da ventana
        setLayout(new GridBagLayout()); //Distribucion da ventana
        setLocationRelativeTo(null); //Colocar a ventana no centro

        cl = new Cliente();

        atras = new Boton("ATRÁS", 200, 110);
        atras.setForeground(Color.BLACK);
                        atras.setFocusPainted(false);
                        atras.addActionListener(this);
                        enviar = new Boton("ENVIAR", 200, 110);
                        enviar.setForeground(Color.BLACK);
                        enviar.setFocusPainted(false);
                        enviar.addActionListener(this);

                        //Declaro colores para o fondo e as etiquetas dos paneles
        Color co= new Color(0);
        Color col= new Color(255,255,255);

       //Creanse paneles e daselle un color de fondo
        panel = new JPanel();
        panel.setBackground(co);

        panel2 = new JPanel();
        panel2.setBackground(co);
        panel2.setLayout(new FlowLayout());
        panel3 = new JPanel();
        panel3.setBackground(co);
```

```
panel3.setLayout(new FlowLayout());
panel4 = new JPanel();
panel4.setBackground(co);
panel4.setLayout(new FlowLayout());

label = new JLabel("REGISTRO DE NUEVO CLIENTE");
label.setFont(new Font("Lucida Sans", Font.BOLD, 20));
label.setForeground(col);

dni = new JLabel("DNI: ");
dni.setFont(new Font("Lucida Sans", Font.PLAIN, 14));
dni.setForeground(col);
nombre = new JLabel("Nombre: ");
nombre.setFont(new Font("Lucida Sans", Font.PLAIN, 14));
nombre.setForeground(col);
apellidos = new JLabel("Apellidos: ");
apellidos.setFont(new Font("Lucida Sans", Font.PLAIN, 14));
apellidos.setForeground(col);
telefono = new JLabel("Teléfono: ");
telefono.setFont(new Font("Lucida Sans", Font.PLAIN, 14));
telefono.setForeground(col);
correo = new JLabel("Correo: ");
correo.setFont(new Font("Lucida Sans", Font.PLAIN, 14));
correo.setForeground(col);
direccion = new JLabel("Dirección: ");
direccion.setFont(new Font("Lucida Sans", Font.PLAIN, 14));
direccion.setForeground(col);

inputdni = new JTextField(10);
inputdni.setFont(new Font("Lucida Sans", Font.PLAIN , 12));
inputdni.setForeground(col);
```

```
        inputdni.setBackground(co);
        inputdni.setEditable(true);
        inputdni.setEnabled(true);
        inputdni.setFocusable(true);
        inputdni.requestFocus();
        inputnombre = new JTextField(10);
        inputnombre.setFont(new Font("Lucida Sans", Font.PLAIN , 12));
        inputnombre.setForeground(col);
        inputnombre.setBackground(co);
        inputnombre.setEditable(true);
        inputnombre.setEnabled(true);
        inputnombre.setFocusable(true);
        inputnombre.requestFocus();
        inputapellidos = new JTextField(10);
        inputapellidos.setFont(new Font("Lucida Sans", Font.PLAIN , 12));
        inputapellidos.setForeground(col);
        inputapellidos.setBackground(co);
        inputapellidos.setEditable(true);
        inputapellidos.setEnabled(true);
        inputapellidos.setFocusable(true);
        inputapellidos.requestFocus();
        inputtelefono = new JTextField(10);
        inputtelefono.setFont(new Font("Lucida Sans", Font.PLAIN , 12));
        inputtelefono.setForeground(col);
        inputtelefono.setBackground(co);
        inputtelefono.setEditable(true);
        inputtelefono.setEnabled(true);
        inputtelefono.setFocusable(true);
        inputtelefono.requestFocus();
        inputcorreo = new JTextField(10);
        inputcorreo.setFont(new Font("Lucida Sans", Font.PLAIN , 12));
```

```
inputcorreo.setForeground(col);
inputcorreo.setBackground(co);
inputcorreo.setEditable(true);
inputcorreo.setEnabled(true);
inputcorreo.setFocusable(true);
inputcorreo.requestFocus();
inputdireccion = new JTextField(10);
inputdireccion.setFont(new Font("Lucida Sans", Font.PLAIN , 12));
inputdireccion.setForeground(col);
inputdireccion.setBackground(co);
inputdireccion.setEditable(true);
inputdireccion.setEnabled(true);
inputdireccion.setFocusable(true);
inputdireccion.requestFocus();

        //Añadense as etiquetas os paneles
        panel.add(label);
        panel2.add(dni);
        panel2.add(inputdni);
        panel2.add(nombre);
        panel2.add(inputnombre);
        panel3.add(apellidos);
        panel3.add(inputapellidos);
        panel3.add(telefono);
        panel3.add(inputtelefono);
        panel4.add(correo);
        panel4.add(inputcorreo);
        panel4.add(direccion);
        panel4.add(inputdireccion);

        GridBagConstraints c = new GridBagConstraints();
```

```
c.gridx=0; // especifica a coordenada x
c.gridy=0;         // coordenada y
c.gridwidth=1; //numero de celdas que ocupa no GridBagLayout
c.gridheight=1;
c.anchor=GridBagConstraints.CENTER; //posicion dentro dunha celda
c.fill=GridBagConstraints.BOTH; //espacio que ocupara dentro dunha celda
c.weightx=1.0; //porcentaxe de espazo libre que ocupara
c.weighty=0;
c.insets=new Insets(10,100,10,100); //ponher marxes
add(panel,c);

c.gridx=0; // especifica a coordenada x
c.gridy=1;         // coordenada y
c.gridwidth=1; //numero de celdas que ocupa no GridBagLayout
c.gridheight=1;
c.anchor=GridBagConstraints.CENTER; //posicion dentro dunha celda
c.fill=GridBagConstraints.BOTH; //espacio que ocupara dentro dunha celda
c.weightx=1.0; //porcentaxe de espazo libre que ocupara
c.weighty=0;
c.insets=new Insets(200,200,10,200); //ponher marxes
add(panel2,c);

c.gridx=0; // especifica a coordenada x
c.gridy=2;         // coordenada y
c.gridwidth=1; //numero de celdas que ocupa no GridBagLayout
c.gridheight=1;
c.anchor=GridBagConstraints.CENTER; //posicion dentro dunha celda
c.fill=GridBagConstraints.BOTH; //espacio que ocupara dentro dunha celda
c.weightx=1.0; //porcentaxe de espazo libre que ocupara
c.weighty=0;
c.insets=new Insets(10,200,10,200); //ponher marxes
```

```
add(panel3,c);

c.gridx=0; // especifica a coordenada x
c.gridy=3;          // coordenada y
c.gridwidth=1; //numero de celdas que ocupa no GridBagLayout
c.gridheight=1;
c.anchor=GridBagConstraints.CENTER; //posicion dentro dunha celda
c.fill=GridBagConstraints.BOTH; //espacio que ocupara dentro dunha celda
c.weightx=1.0; //porcentaxe de espazo libre que ocupara
c.weighty=0;
c.insets=new Insets(10,200,10,200); //ponher marxes
add(panel4,c);

c.gridx=0;
c.gridy=4;
c.gridwidth=1;
c.gridheight=1;
c.anchor=GridBagConstraints.CENTER;
c.fill=GridBagConstraints.BOTH;
c.weightx=1.0;
c.weighty=1.0;
c.insets=new Insets(175,10,10,400);
add(atras,c);

c.gridx=0;
c.gridy=4;
c.gridwidth=1;
c.gridheight=1;
c.anchor=GridBagConstraints.CENTER;
c.fill=GridBagConstraints.BOTH;
c.weightx=1.0;
```

```
            c.weighty=1.0;
            c.insets=new Insets(175,400,10,10);
            add(enviar,c);

    }

    @Override
    public void keyTyped(KeyEvent e) {}

    //control de espacio
    @Override
    public void keyPressed(KeyEvent e) {
            int key = e.getKeyCode(); //collese a tecla soltada

            if(key == KeyEvent.VK_SPACE) { //si e o espacio
                    e.consume();
            }
    }

    @Override
    public void keyReleased(KeyEvent e) {
    }

    @Override
    public void actionPerformed(ActionEvent e) {

            if (e.getActionCommand().equals("ATRÁS")) {
                    MenuClientes men = new MenuClientes();
                    dispose();
                    men.setVisible(true);
```

```
                    //para desactivar o espacio no control de menu
                    JPanel panel = (JPanel) men.getContentPane();

                    panel.addKeyListener(this);
                    panel.setFocusable(true);
                }
                if (e.getActionCommand().equals("ENVIAR")) {
                    if (cl.insertar(inputdni.getText(), inputnombre.getText(),
inputapellidos.getText(), inputtelefono.getText(), inputcorreo.getText(),
inputdireccion.getText())) {
                        MenuListadoClientes men = new MenuListadoClientes();
                        dispose();
                        men.setVisible(true);

                        //para desactivar o espacio no control de menu
                        JPanel panel = (JPanel) men.getContentPane();

                        panel.addKeyListener(this);
                        panel.setFocusable(true);
                    }
                    else {
                        MenuErrorRegistroCliente men = new
MenuErrorRegistroCliente();
                        men.setVisible(true);
                        dispose();

                        //para desactivar o espacio no control de menu
                        JPanel panel = (JPanel) men.getContentPane();

                        panel.addKeyListener(this);
                        panel.setFocusable(true);
                    }
```

```
                    }
            }
}
```

17.18. MenuEditarClienteLista.java

```
package graficoclientes;

import java.awt.Color;
import java.awt.Font;
import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;
import java.awt.Insets;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;

import javax.swing.JComboBox;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;

import graficoprincipal.Boton;
import graficoprincipal.PanelImagen;
import textual.Cliente;
import textual.Persona;

/**
 * @author Adriana Armental Tomé
 * @version 06.03.2017
 */
```

```
public class MenuEditarClienteLista extends JFrame implements KeyListener, ActionListener {

	//Botons menu
	Boton atras, editar;
	Font fuente;
	JPanel panel, panel2;
	JLabel label, label2;
	JComboBox sel;
	Persona cl;
	static String id;

	public MenuEditarClienteLista(){
		super("Taller"); //Nome
	//declarar e colocar fondo
	PanelImagen p = new PanelImagen(); //Panel que conten a imaxe
	setContentPane(p); //Asignar panel
	setSize(800,700); //Tamanho ventana
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); //Poder cerrar a ventana.
	setResizable(false); //Que non se poda cambiar o tamaanho da ventana
	setLayout(new GridBagLayout()); //Distribucion da ventana
	setLocationRelativeTo(null); //Colocar a ventana no centro

	cl = new Cliente();

	atras = new Boton("ATRÁS", 200, 110);
	atras.setForeground(Color.BLACK);
		atras.setFocusPainted(false);
		atras.addActionListener(this);
		editar = new Boton("EDITAR", 200, 110);
		editar.setForeground(Color.BLACK);
```

```
                editar.setFocusPainted(false);
                editar.addActionListener(this);

                //Declaro colores para o fondo e as etiquetas dos paneles
    Color co= new Color(0);
    Color col= new Color(255,255,255);

   //Creanse paneles e daselle un color de fondo
    panel = new JPanel();
    panel.setBackground(co);

    panel2 = new JPanel();
    panel2.setBackground(co);

    label = new JLabel("EDITAR CLIENTES");
    label.setFont(new Font("Lucida Sans", Font.BOLD, 20));
    label.setForeground(col);

    label2 = new JLabel("¿Qué cliente quieres editar?");
    label2.setFont(new Font("Lucida Sans", Font.PLAIN, 14));
    label2.setForeground(col);

    sel = new JComboBox();
    sel.setFont(new Font("Lucida Sans", Font.PLAIN , 12));


    String nombreper = cl.NombrePersona().toString();
    nombreper = nombreper.replace("[", "").replace("]", "");
    String[] parts = nombreper.split(",");
    String apellidoper = cl.ApellidoPersona().toString();
    apellidoper = apellidoper.replace("[", "").replace("]", "");
```

```
String[] parts2 = apellidoper.split(",");
String idcliente2 = cl.IDPersona().toString();
idcliente2 = idcliente2.replace("[", "").replace("]", "");
String[] parts3 = idcliente2.split(",");
for (int i=0; i< parts.length; i++) {
   sel.addItem(parts[i].trim()+" "+parts2[i].trim()+" ("+parts3[i].trim()+")");
}

                //Añadense as etiquetas os paneles
                panel.add(label);
                panel2.add(label2);

                GridBagConstraints c = new GridBagConstraints();
                c.gridx=0; // especifica a coordenada x
                c.gridy=0;          // coordenada y
                c.gridwidth=1; //numero de celdas que ocupa no GridBagLayout
                c.gridheight=1;
                c.anchor=GridBagConstraints.CENTER; //posicion dentro dunha celda
                c.fill=GridBagConstraints.BOTH; //espacio que ocupara dentro dunha celda
                c.weightx=1.0; //porcentaxe de espazo libre que ocupara
                c.weighty=0;
                c.insets=new Insets(10,100,10,100); //ponher marxes
                add(panel,c);

                c.gridx=0; // especifica a coordenada x
                c.gridy=1;          // coordenada y
                c.gridwidth=1; //numero de celdas que ocupa no GridBagLayout
                c.gridheight=1;
                c.anchor=GridBagConstraints.CENTER; //posicion dentro dunha celda
                c.fill=GridBagConstraints.BOTH; //espacio que ocupara dentro dunha celda
                c.weightx=1.0; //porcentaxe de espazo libre que ocupara
```

```
c.weighty=0;
c.insets=new Insets(10,200,10,200); //ponher marxes
add(panel2,c);

c.gridx=0; // especifica a coordenada x
c.gridy=2;          // coordenada y
c.gridwidth=1; //numero de celdas que ocupa no GridBagLayout
c.gridheight=1;
c.anchor=GridBagConstraints.CENTER; //posicion dentro dunha celda
c.fill=GridBagConstraints.BOTH; //espacio que ocupara dentro dunha celda
c.weightx=1.0; //porcentaxe de espazo libre que ocupara
c.weighty=1.0;
c.insets=new Insets(200,100,200,100); //ponher marxes
add(sel,c);

c.gridx=0;
c.gridy=4;
c.gridwidth=1;
c.gridheight=1;
c.anchor=GridBagConstraints.CENTER;
c.fill=GridBagConstraints.BOTH;
c.weightx=1.0;
c.weighty=1.0;
c.insets=new Insets(10,10,10,400);
add(atras,c);

c.gridx=0;
c.gridy=4;
c.gridwidth=1;
c.gridheight=1;
c.anchor=GridBagConstraints.CENTER;
```

```
        c.fill=GridBagConstraints.BOTH;
        c.weightx=1.0;
        c.weighty=1.0;
        c.insets=new Insets(10,400,10,10);
        add(editar,c);

}

@Override
public void keyTyped(KeyEvent e) {}

//control de espacio
@Override
public void keyPressed(KeyEvent e) {
        int key = e.getKeyCode(); //collese a tecla soltada

        if(key == KeyEvent.VK_SPACE) { //si e o espacio
                e.consume();
        }
}

@Override
public void keyReleased(KeyEvent e) {
}

@Override
public void actionPerformed(ActionEvent e) {

        if (e.getActionCommand().equals("ATRÁS")) {
                MenuClientes men = new MenuClientes();
                dispose();
```

```
                men.setVisible(true);

                //para desactivar o espacio no control de menu
                JPanel panel = (JPanel) men.getContentPane();

                panel.addKeyListener(this);
                panel.setFocusable(true);
            }
            if (e.getActionCommand().equals("EDITAR")) {
                String resul = (String) sel.getSelectedItem();

                int inicio = resul.indexOf("(");
                int fin = resul.indexOf(")");

                id = resul.substring(inicio + 1, fin);

                MenuEditarCliente men = new MenuEditarCliente();
                dispose();
                men.setVisible(true);

                //para desactivar o espacio no control de menu
                JPanel panel = (JPanel) men.getContentPane();

                panel.addKeyListener(this);
                panel.setFocusable(true);


            }
        }
}
```

17.19. MenuEditarCliente.java

```
package graficoclientes;

import java.awt.Color;
import java.awt.FlowLayout;
import java.awt.Font;
import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;
import java.awt.Insets;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;

import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JTextField;

import graficoprincipal.Boton;
import graficoprincipal.PanelImagen;
import textual.Cliente;
import textual.Persona;

/**
 * @author Adriana Armental Tomé
 * @version 06.03.2017
 */

public class MenuEditarCliente extends JFrame implements KeyListener, ActionListener {
```

```
        //Botons menu
        Boton atras, enviar;
        Font fuente;
        JPanel panel, panel2, panel3, panel4;
        JLabel label, dni, nombre, apellidos, telefono, correo, direccion;
        JTextField inputdni, inputnombre, inputapellidos, inputtelefono, inputcorreo,
inputdireccion;
        Persona cl;
        String sdni, snombre, sapellidos, stelefono, scorreo, sdireccion;

        public MenuEditarCliente(){
                super("Taller"); //Nome
    //declarar e colocar fondo
    PanelImagen p = new PanelImagen(); //Panel que conten a imaxe
    setContentPane(p); //Asignar panel
    setSize(800,700); //Tamanho ventana
                setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); //Poder cerrar a ventana.
    setResizable(false); //Que non se poda cambiar o tamaanho da ventana
    setLayout(new GridBagLayout()); //Distribucion da ventana
    setLocationRelativeTo(null); //Colocar a ventana no centro

    cl = new Cliente();
    sdni = cl.getDni(MenuEditarClienteLista.id);
    snombre = cl.getNombre(MenuEditarClienteLista.id);
    sapellidos = cl.getApellidos(MenuEditarClienteLista.id);
    stelefono = cl.getNumTelefono(MenuEditarClienteLista.id);
    scorreo = cl.getCorreoElectronico(MenuEditarClienteLista.id);
    sdireccion = cl.getDireccion(MenuEditarClienteLista.id);

    atras = new Boton("ATRÁS", 200, 110);
    atras.setForeground(Color.BLACK);
```

```
                atras.setFocusPainted(false);
                atras.addActionListener(this);
                enviar = new Boton("ENVIAR", 200, 110);
                enviar.setForeground(Color.BLACK);
                enviar.setFocusPainted(false);
                enviar.addActionListener(this);

                //Declaro colores para o fondo e as etiquetas dos paneles
 Color co= new Color(0);
 Color col= new Color(255,255,255);

//Creanse paneles e daselle un color de fondo
 panel = new JPanel();
 panel.setBackground(co);

 panel2 = new JPanel();
 panel2.setBackground(co);
 panel2.setLayout(new FlowLayout());
 panel3 = new JPanel();
 panel3.setBackground(co);
 panel3.setLayout(new FlowLayout());
 panel4 = new JPanel();
 panel4.setBackground(co);
 panel4.setLayout(new FlowLayout());

 label = new JLabel("EDITAR CLIENTE");
 label.setFont(new Font("Lucida Sans", Font.BOLD, 20));
 label.setForeground(col);

 dni = new JLabel("DNI: ");
 dni.setFont(new Font("Lucida Sans", Font.PLAIN, 14));
```

```
dni.setForeground(col);
nombre = new JLabel("Nombre: ");
nombre.setFont(new Font("Lucida Sans", Font.PLAIN, 14));
nombre.setForeground(col);
apellidos = new JLabel("Apellidos: ");
apellidos.setFont(new Font("Lucida Sans", Font.PLAIN, 14));
apellidos.setForeground(col);
telefono = new JLabel("Teléfono: ");
telefono.setFont(new Font("Lucida Sans", Font.PLAIN, 14));
telefono.setForeground(col);
correo = new JLabel("Correo: ");
correo.setFont(new Font("Lucida Sans", Font.PLAIN, 14));
correo.setForeground(col);
direccion = new JLabel("Dirección: ");
direccion.setFont(new Font("Lucida Sans", Font.PLAIN, 14));
direccion.setForeground(col);

inputdni = new JTextField(10);
inputdni.setFont(new Font("Lucida Sans", Font.PLAIN , 12));
inputdni.setForeground(col);
inputdni.setBackground(co);
inputdni.setEditable(true);
inputdni.setEnabled(true);
inputdni.setFocusable(true);
inputdni.requestFocus();
inputdni.setText(sdni);
inputnombre = new JTextField(10);
inputnombre.setFont(new Font("Lucida Sans", Font.PLAIN , 12));
inputnombre.setForeground(col);
inputnombre.setBackground(co);
inputnombre.setEditable(true);
```

```
inputnombre.setEnabled(true);
inputnombre.setFocusable(true);
inputnombre.requestFocus();
inputnombre.setText(snombre);
inputapellidos = new JTextField(10);
inputapellidos.setFont(new Font("Lucida Sans", Font.PLAIN , 12));
inputapellidos.setForeground(col);
inputapellidos.setBackground(co);
inputapellidos.setEditable(true);
inputapellidos.setEnabled(true);
inputapellidos.setFocusable(true);
inputapellidos.requestFocus();
inputapellidos.setText(sapellidos);
inputtelefono = new JTextField(10);
inputtelefono.setFont(new Font("Lucida Sans", Font.PLAIN , 12));
inputtelefono.setForeground(col);
inputtelefono.setBackground(co);
inputtelefono.setEditable(true);
inputtelefono.setEnabled(true);
inputtelefono.setFocusable(true);
inputtelefono.requestFocus();
inputtelefono.setText(stelefono);
inputcorreo = new JTextField(10);
inputcorreo.setFont(new Font("Lucida Sans", Font.PLAIN , 12));
inputcorreo.setForeground(col);
inputcorreo.setBackground(co);
inputcorreo.setEditable(true);
inputcorreo.setEnabled(true);
inputcorreo.setFocusable(true);
inputcorreo.requestFocus();
inputcorreo.setText(scorreo);
```

```
inputdireccion = new JTextField(10);
inputdireccion.setFont(new Font("Lucida Sans", Font.PLAIN , 12));
inputdireccion.setForeground(col);
inputdireccion.setBackground(co);
inputdireccion.setEditable(true);
inputdireccion.setEnabled(true);
inputdireccion.setFocusable(true);
inputdireccion.requestFocus();
inputdireccion.setText(sdireccion);

        //Añadense as etiquetas os paneles
        panel.add(label);
        panel2.add(dni);
        panel2.add(inputdni);
        panel2.add(nombre);
        panel2.add(inputnombre);
        panel3.add(apellidos);
        panel3.add(inputapellidos);
        panel3.add(telefono);
        panel3.add(inputtelefono);
        panel4.add(correo);
        panel4.add(inputcorreo);
        panel4.add(direccion);
        panel4.add(inputdireccion);

        GridBagConstraints c = new GridBagConstraints();
        c.gridx=0; // especifica a coordenada x
        c.gridy=0;         // coordenada y
        c.gridwidth=1; //numero de celdas que ocupa no GridBagLayout
        c.gridheight=1;
        c.anchor=GridBagConstraints.CENTER; //posicion dentro dunha celda
```

```
c.fill=GridBagConstraints.BOTH; //espacio que ocupara dentro dunha celda
c.weightx=1.0; //porcentaxe de espazo libre que ocupara
c.weighty=0;
c.insets=new Insets(10,100,10,100); //ponher marxes
add(panel,c);

c.gridx=0; // especifica a coordenada x
c.gridy=1;          // coordenada y
c.gridwidth=1; //numero de celdas que ocupa no GridBagLayout
c.gridheight=1;
c.anchor=GridBagConstraints.CENTER; //posicion dentro dunha celda
c.fill=GridBagConstraints.BOTH; //espacio que ocupara dentro dunha celda
c.weightx=1.0; //porcentaxe de espazo libre que ocupara
c.weighty=0;
c.insets=new Insets(200,200,10,200); //ponher marxes
add(panel2,c);

c.gridx=0; // especifica a coordenada x
c.gridy=2;          // coordenada y
c.gridwidth=1; //numero de celdas que ocupa no GridBagLayout
c.gridheight=1;
c.anchor=GridBagConstraints.CENTER; //posicion dentro dunha celda
c.fill=GridBagConstraints.BOTH; //espacio que ocupara dentro dunha celda
c.weightx=1.0; //porcentaxe de espazo libre que ocupara
c.weighty=0;
c.insets=new Insets(10,200,10,200); //ponher marxes
add(panel3,c);

c.gridx=0; // especifica a coordenada x
c.gridy=3;          // coordenada y
c.gridwidth=1; //numero de celdas que ocupa no GridBagLayout
```

```
        c.gridheight=1;
        c.anchor=GridBagConstraints.CENTER; //posicion dentro dunha celda
        c.fill=GridBagConstraints.BOTH; //espacio que ocupara dentro dunha celda
        c.weightx=1.0; //porcentaxe de espazo libre que ocupara
        c.weighty=0;
        c.insets=new Insets(10,200,10,200); //ponher marxes
        add(panel4,c);

        c.gridx=0;
        c.gridy=4;
        c.gridwidth=1;
        c.gridheight=1;
        c.anchor=GridBagConstraints.CENTER;
        c.fill=GridBagConstraints.BOTH;
        c.weightx=1.0;
        c.weighty=1.0;
        c.insets=new Insets(175,10,10,400);
        add(atras,c);

        c.gridx=0;
        c.gridy=4;
        c.gridwidth=1;
        c.gridheight=1;
        c.anchor=GridBagConstraints.CENTER;
        c.fill=GridBagConstraints.BOTH;
        c.weightx=1.0;
        c.weighty=1.0;
        c.insets=new Insets(175,400,10,10);
        add(enviar,c);

    }
```

```
@Override
public void keyTyped(KeyEvent e) {}

//control de espacio
@Override
public void keyPressed(KeyEvent e) {
	int key = e.getKeyCode(); //collese a tecla soltada

	if(key == KeyEvent.VK_SPACE) { //si e o espacio
		e.consume();
	}
}

@Override
public void keyReleased(KeyEvent e) {
}

@Override
public void actionPerformed(ActionEvent e) {

	if (e.getActionCommand().equals("ATRÁS")) {
		MenuClientes men = new MenuClientes();
		dispose();
		men.setVisible(true);

		//para desactivar o espacio no control de menu
		JPanel panel = (JPanel) men.getContentPane();

		panel.addKeyListener(this);
		panel.setFocusable(true);
```

```
                }
                if (e.getActionCommand().equals("ENVIAR")) {
                        if (cl.editar(MenuEditarClienteLista.id, inputnombre.getText(),
inputapellidos.getText(), inputtelefono.getText(), inputcorreo.getText(),
inputdireccion.getText())) {
                                MenuListadoClientes men = new MenuListadoClientes();
                                dispose();
                                men.setVisible(true);

                                //para desactivar o espacio no control de menu
                                JPanel panel = (JPanel) men.getContentPane();

                                panel.addKeyListener(this);
                                panel.setFocusable(true);
                        }
                        else {
                                MenuErrorRegistroCliente men = new
MenuErrorRegistroCliente();
                                men.setVisible(true);
                                dispose();

                                //para desactivar o espacio no control de menu
                                JPanel panel = (JPanel) men.getContentPane();

                                panel.addKeyListener(this);
                                panel.setFocusable(true);
                        }
                }
        }
}
```

17.20. MenuEliminarCliente.java

```
package graficoclientes;

import java.awt.Color;
import java.awt.Font;
import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;
import java.awt.Insets;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;

import javax.swing.JComboBox;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;

import graficoprincipal.Boton;
import graficoprincipal.PanelImagen;
import textual.Cliente;
import textual.Persona;

/**
 * @author Adriana Armental Tomé
 * @version 06.03.2017
 */

public class MenuEliminarCliente extends JFrame implements KeyListener, ActionListener {
```

```
        //Botons menu
        Boton atras, eliminar;
        Font fuente;
        JPanel panel, panel2;
        JLabel label, label2;
        JComboBox sel;
        Persona cl;

        public MenuEliminarCliente(){
                super("Taller"); //Nome
    //declarar e colocar fondo
    PanelImagen p = new PanelImagen(); //Panel que conten a imaxe
    setContentPane(p); //Asignar panel
    setSize(800,700); //Tamanho ventana
                setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); //Poder cerrar a ventana.
    setResizable(false); //Que non se poda cambiar o tamaanho da ventana
    setLayout(new GridBagLayout()); //Distribucion da ventana
    setLocationRelativeTo(null); //Colocar a ventana no centro

    cl = new Cliente();

    atras = new Boton("ATRÁS", 200, 110);
    atras.setForeground(Color.BLACK);
                atras.setFocusPainted(false);
                atras.addActionListener(this);
                eliminar = new Boton("ELIMINAR", 200, 110);
                eliminar.setForeground(Color.BLACK);
                eliminar.setFocusPainted(false);
                eliminar.addActionListener(this);

                //Declaro colores para o fondo e as etiquetas dos paneles
```

```
 Color co= new Color(0);
 Color col= new Color(255,255,255);

//Creanse paneles e daselle un color de fondo
 panel = new JPanel();
 panel.setBackground(co);

 panel2 = new JPanel();
 panel2.setBackground(co);

 label = new JLabel("ELIMINAR CLIENTES");
 label.setFont(new Font("Lucida Sans", Font.BOLD, 20));
 label.setForeground(col);

 label2 = new JLabel("¿A quién quieres eliminar?");
 label2.setFont(new Font("Lucida Sans", Font.PLAIN, 14));
 label2.setForeground(col);

 sel = new JComboBox();
 sel.setFont(new Font("Lucida Sans", Font.PLAIN , 12));


 String nombreper = cl.NombrePersona().toString();
 nombreper = nombreper.replace("[", "").replace("]", "");
 String[] parts = nombreper.split(",");
 String apellidoper = cl.ApellidoPersona().toString();
 apellidoper = apellidoper.replace("[", "").replace("]", "");
 String[] parts2 = apellidoper.split(",");
 String idcliente2 = cl.IDPersona().toString();
 idcliente2 = idcliente2.replace("[", "").replace("]", "");
 String[] parts3 = idcliente2.split(",");
```

```
    for (int i=0; i< parts.length; i++) {
       sel.addItem(parts[i].trim()+" "+parts2[i].trim()+" ("+parts3[i].trim()+")");
    }

                //Añadense as etiquetas os paneles
                panel.add(label);
                panel2.add(label2);

                GridBagConstraints c = new GridBagConstraints();
                c.gridx=0; // especifica a coordenada x
                c.gridy=0;          // coordenada y
                c.gridwidth=1; //numero de celdas que ocupa no GridBagLayout
                c.gridheight=1;
                c.anchor=GridBagConstraints.CENTER; //posicion dentro dunha celda
                c.fill=GridBagConstraints.BOTH; //espacio que ocupara dentro dunha celda
                c.weightx=1.0; //porcentaxe de espazo libre que ocupara
                c.weighty=0;
                c.insets=new Insets(10,100,10,100); //ponher marxes
                add(panel,c);

                c.gridx=0; // especifica a coordenada x
                c.gridy=1;          // coordenada y
                c.gridwidth=1; //numero de celdas que ocupa no GridBagLayout
                c.gridheight=1;
                c.anchor=GridBagConstraints.CENTER; //posicion dentro dunha celda
                c.fill=GridBagConstraints.BOTH; //espacio que ocupara dentro dunha celda
                c.weightx=1.0; //porcentaxe de espazo libre que ocupara
                c.weighty=0;
                c.insets=new Insets(10,200,10,200); //ponher marxes
                add(panel2,c);
```

```
c.gridx=0; // especifica a coordenada x
c.gridy=2;         // coordenada y
c.gridwidth=1; //numero de celdas que ocupa no GridBagLayout
c.gridheight=1;
c.anchor=GridBagConstraints.CENTER; //posicion dentro dunha celda
c.fill=GridBagConstraints.BOTH; //espacio que ocupara dentro dunha celda
c.weightx=1.0; //porcentaxe de espazo libre que ocupara
c.weighty=1.0;
c.insets=new Insets(200,100,200,100); //ponher marxes
add(sel,c);

c.gridx=0;
c.gridy=4;
c.gridwidth=1;
c.gridheight=1;
c.anchor=GridBagConstraints.CENTER;
c.fill=GridBagConstraints.BOTH;
c.weightx=1.0;
c.weighty=1.0;
c.insets=new Insets(10,10,10,400);
add(atras,c);

c.gridx=0;
c.gridy=4;
c.gridwidth=1;
c.gridheight=1;
c.anchor=GridBagConstraints.CENTER;
c.fill=GridBagConstraints.BOTH;
c.weightx=1.0;
c.weighty=1.0;
c.insets=new Insets(10,400,10,10);
```

```
            add(eliminar,c);

        }

        @Override
        public void keyTyped(KeyEvent e) {}

        //control de espacio
        @Override
        public void keyPressed(KeyEvent e) {
            int key = e.getKeyCode(); //collese a tecla soltada

            if(key == KeyEvent.VK_SPACE) { //si e o espacio
                e.consume();
            }
        }

        @Override
        public void keyReleased(KeyEvent e) {
        }

        @Override
        public void actionPerformed(ActionEvent e) {

            if (e.getActionCommand().equals("ATRÁS")) {
                MenuClientes men = new MenuClientes();
                dispose();
                men.setVisible(true);

                //para desactivar o espacio no control de menu
                JPanel panel = (JPanel) men.getContentPane();
```

```
                panel.addKeyListener(this);
                panel.setFocusable(true);
            }
            if (e.getActionCommand().equals("ELIMINAR")) {
                String resul = (String) sel.getSelectedItem();

                int inicio = resul.indexOf("(");
                int fin = resul.indexOf(")");

                String ident = resul.substring(inicio + 1, fin);

                if (cl.borrar(ident)) {
                    MenuListadoClientes men = new MenuListadoClientes();
                    dispose();
                    men.setVisible(true);

                    //para desactivar o espacio no control de menu
                    JPanel panel = (JPanel) men.getContentPane();

                    panel.addKeyListener(this);
                    panel.setFocusable(true);
                }
                else {
                    MenuErrorRegistroCliente men = new
MenuErrorRegistroCliente();
                    men.setVisible(true);
                    dispose();

                    //para desactivar o espacio no control de menu
                    JPanel panel = (JPanel) men.getContentPane();
```

```
                                panel.addKeyListener(this);
                                panel.setFocusable(true);
                        }

                }
        }
}
```

17.21. MenuErrorRegistroCliente.java

```
package graficoclientes;

import java.awt.Color;
import java.awt.Font;
import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;
import java.awt.Insets;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;

import graficoprincipal.Boton;
import graficoprincipal.PanelImagen;

/**
 * @author Adriana Armental Tomé
 * @version 06.03.2017
```

```
 */


public class MenuErrorRegistroCliente extends JFrame implements KeyListener,
ActionListener{


        //Declaracion de paneles, etiquetas e botons

        JPanel panel;

        JLabel label;

        Boton volver;


        public MenuErrorRegistroCliente() {

                super("Taller"); //Nome

                PanelImagen p = new PanelImagen();

    setContentPane(p);

                setSize(400,350); //TamaÃ±o ventana

                setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); //Poder cerrar a ventana.

                setResizable(false); //Que non se poda cambiar o tamaÃ±o da ventana

            setLayout(new GridBagLayout());

            setLocationRelativeTo(null);


            //Declaro colores para o fondo e as etiquetas dos paneles

            Color co= new Color(0);

    Color col= new Color(255,255,255);


    //Creanse paneles e daselle un color de fondo

    panel = new JPanel();

    panel.setBackground(co);


    String texto =  "<html><body><p align='center'>ERROR</p><br>"

                + "<p align='center'>Se ha detectado un error en los datos
introducidos.</p></body></html>" ;
```

```
label = new JLabel(texto);
label.setFont(new Font("Lucida Sans", Font.PLAIN , 12));
label.setForeground(col);

        volver = new Boton("VOLVER", 200, 110);
        volver.setFocusPainted(false);
        volver.addActionListener(this);

        panel.add(label);

        GridBagConstraints c = new GridBagConstraints();
        c.gridx=0; // especifica a coordenada x
        c.gridy=0;      // coordenada y
        c.gridwidth=1; //nÃºmero de celdas que ocupa no GridBagLayout
        c.gridheight=1;
        c.anchor=GridBagConstraints.NORTH; //posiciÃ³n dentro dunha celda

        c.fill=GridBagConstraints.NORTH; //espacio que ocupara dentro dunha celda

        c.weightx=1.0; //porcentaxe de espazo libre que ocupara
        c.weighty=1.0;
        c.insets=new Insets(10,10,10,10); //poÃ±er marxes
        add(panel,c);

        c.gridx=0;
        c.gridy=1;
        c.gridwidth=1;
        c.gridheight=1;
        c.anchor=GridBagConstraints.CENTER;
        c.fill=GridBagConstraints.BOTH;
        c.weightx=1.0;
```

```
   c.weighty=1.0;
   //c.insets=new Insets(20,200,20,200);
   add(volver,c);
}

@Override
public void keyTyped(KeyEvent e) {}

//control de espacio
@Override
public void keyPressed(KeyEvent e) {
        int key = e.getKeyCode(); //collese a tecla soltada

        if(key == KeyEvent.VK_SPACE) { //si e o espacio
                e.consume();
        }
}

@Override
public void keyReleased(KeyEvent e) {
}

@Override
public void actionPerformed(ActionEvent e) {

        if (e.getActionCommand().equals("VOLVER")) {
                MenuClientes men = new MenuClientes();
                dispose();
                men.setVisible(true);

                //para desactivar o espacio no control de menu
```

```
                        JPanel panel = (JPanel) men.getContentPane();

                        panel.addKeyListener(this);
                        panel.setFocusable(true);

                }

        }

}
```

17.22. MenuListadoClientes.java

```
package graficoclientes;

import java.awt.Color;
import java.awt.Dimension;
import java.awt.Font;
import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;
import java.awt.Insets;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;

import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.ScrollPaneConstants;
```

```
import graficoprincipal.Boton;
import graficoprincipal.PanelImagen;
import textual.Cliente;
import textual.Persona;

/**
 * @author Adriana Armental Tomé
 * @version 06.03.2017
 */

public class MenuListadoClientes extends JFrame implements KeyListener, ActionListener {

	//Botons menu
	Boton atras;
	Font fuente;
	JPanel panel, panel2, panelscrollable;
	JLabel label, etiqueta;
	JScrollPane scrollPane;
	Persona cl;

	public MenuListadoClientes(){
		super("Taller"); //Nome
      //declarar e colocar fondo
      PanelImagen p = new PanelImagen(); //Panel que conten a imaxe
      setContentPane(p); //Asignar panel
      setSize(800,700); //Tamanho ventana
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); //Poder cerrar a ventana.
      setResizable(false); //Que non se poda cambiar o tamaanho da ventana
      setLayout(new GridBagLayout()); //Distribucion da ventana
      setLocationRelativeTo(null); //Colocar a ventana no centro
```

```
        cl = new Cliente();

        atras = new Boton("ATRÁS", 200, 110);
        atras.setForeground(Color.BLACK);
                        atras.setFocusPainted(false);
                        atras.addActionListener(this);

                        //Declaro colores para o fondo e as etiquetas dos paneles
        Color co= new Color(0);
        Color col= new Color(255,255,255);

       //Creanse paneles e daselle un color de fondo
        panel = new JPanel();
        panel.setBackground(co);

        panel2 = new JPanel();
        panel2.setBackground(co);

        label = new JLabel("LISTADO DE CLIENTES");
        label.setFont(new Font("Lucida Sans", Font.BOLD, 20));
        label.setForeground(col);

        etiqueta = new JLabel(cl.listar());
        etiqueta.setFont(new Font("Lucida Sans", Font.PLAIN , 12));
        etiqueta.setForeground(col);

                        //Añadense as etiquetas os paneles
                        panel.add(label);
                        panel2.add(etiqueta);
                        scrollPane = new JScrollPane(panel2);
```

```
        scrollPane.setHorizontalScrollBarPolicy(ScrollPaneConstants.HORIZONTAL_SCROLLBAR
_AS_NEEDED);

scrollPane.setVerticalScrollBarPolicy(ScrollPaneConstants.VERTICAL_SCROLLBAR_AS_NEEDED);
    scrollPane.setBounds(0, 10, 770, 300);
    panelscrollable = new JPanel(null);
    panelscrollable.setPreferredSize(new Dimension(800, 700));
    panelscrollable.add(scrollPane);

                GridBagConstraints c = new GridBagConstraints();
                c.gridx=0; // especifica a coordenada x
                c.gridy=0;         // coordenada y
                c.gridwidth=1; //numero de celdas que ocupa no GridBagLayout
                c.gridheight=1;
                c.anchor=GridBagConstraints.CENTER; //posicion dentro dunha celda
                c.fill=GridBagConstraints.BOTH; //espacio que ocupara dentro dunha celda
                c.weightx=1.0; //porcentaxe de espazo libre que ocupara
                c.weighty=0;
                c.insets=new Insets(10,100,10,100); //ponher marxes
                add(panel,c);

                c.gridx=0; // especifica a coordenada x
                c.gridy=1;         // coordenada y
                c.gridwidth=1; //numero de celdas que ocupa no GridBagLayout
                c.gridheight=1;
                c.anchor=GridBagConstraints.CENTER; //posicion dentro dunha celda
                c.fill=GridBagConstraints.BOTH; //espacio que ocupara dentro dunha celda
                c.weightx=1.0; //porcentaxe de espazo libre que ocupara
                c.weighty=1.0;
                c.insets=new Insets(10,10,10,10); //ponher marxes
                add(panelscrollable,c);
```

```
            c.gridx=0;
            c.gridy=6;
            c.gridwidth=1;
            c.gridheight=1;
            c.anchor=GridBagConstraints.CENTER;
            c.fill=GridBagConstraints.BOTH;
            c.weightx=1.0;
            c.weighty=0.5;
            c.insets=new Insets(50,10,50,400);
            add(atras,c);

    }

    @Override
    public void keyTyped(KeyEvent e) {}

    //control de espacio
    @Override
    public void keyPressed(KeyEvent e) {
            int key = e.getKeyCode(); //collese a tecla soltada

            if(key == KeyEvent.VK_SPACE) { //si e o espacio
                    e.consume();
            }
    }

    @Override
    public void keyReleased(KeyEvent e) {
    }
```

```
        @Override
        public void actionPerformed(ActionEvent e) {

                if (e.getActionCommand().equals("ATRÁS")) {
                        MenuConsultaClientes men = new MenuConsultaClientes();
                        dispose();
                        men.setVisible(true);

                        //para desactivar o espacio no control de menu
                        JPanel panel = (JPanel) men.getContentPane();

                        panel.addKeyListener(this);
                        panel.setFocusable(true);
                }
        }
}
```

17.23. MenuListadoClientesEscribirDNI.java

```
package graficoclientes;

import java.awt.Color;
import java.awt.FlowLayout;
import java.awt.Font;
import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;
import java.awt.Insets;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;
```

```
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JTextField;

import graficoprincipal.Boton;
import graficoprincipal.PanelImagen;
import textual.Cliente;
import textual.Persona;

/**
 * @author Adriana Armental Tomé
 * @version 06.03.2017
 */

public class MenuListadoClientesEscribirDNI extends JFrame implements KeyListener,
ActionListener {

	//Botons menu
	Boton atras, enviar;
	Font fuente;
	JPanel panel, panel2, panel3;
	JLabel label, label2, dni;
	static JTextField input;
	Persona cl;

	public MenuListadoClientesEscribirDNI(){
		super("Taller"); //Nome
      //declarar e colocar fondo
      PanelImagen p = new PanelImagen(); //Panel que conten a imaxe
      setContentPane(p); //Asignar panel
```

```
setSize(800,700); //Tamanho ventana
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); //Poder cerrar a ventana.
setResizable(false); //Que non se poda cambiar o tamaanho da ventana
setLayout(new GridBagLayout()); //Distribucion da ventana
setLocationRelativeTo(null); //Colocar a ventana no centro

cl = new Cliente();

atras = new Boton("ATRÁS", 200, 110);
atras.setForeground(Color.BLACK);
        atras.setFocusPainted(false);
        atras.addActionListener(this);
        enviar = new Boton("ENVIAR", 200, 110);
        enviar.setForeground(Color.BLACK);
        enviar.setFocusPainted(false);
        enviar.addActionListener(this);

        //Declaro colores para o fondo e as etiquetas dos paneles
Color co= new Color(0);
Color col= new Color(255,255,255);

//Creanse paneles e daselle un color de fondo
panel = new JPanel();
panel.setBackground(co);

panel2 = new JPanel();
panel2.setBackground(co);

panel3 = new JPanel();
panel3.setBackground(co);
panel3.setLayout(new FlowLayout());
```

```
label = new JLabel("BUSCADOR POR DNI DE CLIENTES");
label.setFont(new Font("Lucida Sans", Font.BOLD, 20));
label.setForeground(col);

label2 = new JLabel("¿Que DNI quieres buscar?");
label2.setFont(new Font("Lucida Sans", Font.PLAIN, 14));
label2.setForeground(col);

dni = new JLabel("DNI");
dni.setFont(new Font("Lucida Sans", Font.PLAIN, 14));
dni.setForeground(col);

input = new JTextField(30);
input.setFont(new Font("Lucida Sans", Font.PLAIN , 12));
input.setForeground(col);
input.setBackground(co);
input.setEditable(true);
input.setEnabled(true);
input.setFocusable(true);
input.requestFocus();

        //Añadense as etiquetas os paneles
        panel.add(label);
        panel2.add(label2);
        panel3.add(dni);
        panel3.add(input);

        GridBagConstraints c = new GridBagConstraints();
        c.gridx=0; // especifica a coordenada x
        c.gridy=0;         // coordenada y
```

```
c.gridwidth=1; //numero de celdas que ocupa no GridBagLayout
c.gridheight=1;
c.anchor=GridBagConstraints.CENTER; //posicion dentro dunha celda
c.fill=GridBagConstraints.BOTH; //espacio que ocupara dentro dunha celda
c.weightx=1.0; //porcentaxe de espazo libre que ocupara
c.weighty=0;
c.insets=new Insets(10,100,10,100); //ponher marxes
add(panel,c);

c.gridx=0; // especifica a coordenada x
c.gridy=1;          // coordenada y
c.gridwidth=1; //numero de celdas que ocupa no GridBagLayout
c.gridheight=1;
c.anchor=GridBagConstraints.CENTER; //posicion dentro dunha celda
c.fill=GridBagConstraints.BOTH; //espacio que ocupara dentro dunha celda
c.weightx=1.0; //porcentaxe de espazo libre que ocupara
c.weighty=0;
c.insets=new Insets(10,200,10,200); //ponher marxes
add(panel2,c);

c.gridx=0; // especifica a coordenada x
c.gridy=2;          // coordenada y
c.gridwidth=1; //numero de celdas que ocupa no GridBagLayout
c.gridheight=1;
c.anchor=GridBagConstraints.CENTER; //posicion dentro dunha celda
c.fill=GridBagConstraints.BOTH; //espacio que ocupara dentro dunha celda
c.weightx=1.0; //porcentaxe de espazo libre que ocupara
c.weighty=0;
c.insets=new Insets(200,100,200,100); //ponher marxes
add(panel3,c);
```

```
            c.gridx=0;
            c.gridy=4;
            c.gridwidth=1;
            c.gridheight=1;
            c.anchor=GridBagConstraints.CENTER;
            c.fill=GridBagConstraints.BOTH;
            c.weightx=1.0;
            c.weighty=1.0;
            c.insets=new Insets(10,10,10,400);
            add(atras,c);

            c.gridx=0;
            c.gridy=4;
            c.gridwidth=1;
            c.gridheight=1;
            c.anchor=GridBagConstraints.CENTER;
            c.fill=GridBagConstraints.BOTH;
            c.weightx=1.0;
            c.weighty=1.0;
            c.insets=new Insets(10,400,10,10);
            add(enviar,c);

    }

    @Override
    public void keyTyped(KeyEvent e) {}

    //control de espacio
    @Override
    public void keyPressed(KeyEvent e) {
            int key = e.getKeyCode(); //collese a tecla soltada
```

```
        if(key == KeyEvent.VK_SPACE) { //si e o espacio
                e.consume();
        }
}

@Override
public void keyReleased(KeyEvent e) {
}

@Override
public void actionPerformed(ActionEvent e) {

        if (e.getActionCommand().equals("ATRÁS")) {
                MenuConsultaClientes men = new MenuConsultaClientes();
                dispose();
                men.setVisible(true);

                //para desactivar o espacio no control de menu
                JPanel panel = (JPanel) men.getContentPane();

                panel.addKeyListener(this);
                panel.setFocusable(true);
        }
        if (e.getActionCommand().equals("ENVIAR")) {
                MenuListadoClientesPorDNI men = new MenuListadoClientesPorDNI();
                dispose();
                men.setVisible(true);

                //para desactivar o espacio no control de menu
                JPanel panel = (JPanel) men.getContentPane();
```

```
                panel.addKeyListener(this);
                panel.setFocusable(true);
            }
        }
}
```

17.24. MenuListadoClientesPorDNI.java

```
package graficoclientes;

import java.awt.Color;
import java.awt.Dimension;
import java.awt.Font;
import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;
import java.awt.Insets;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;

import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.ScrollPaneConstants;

import graficoprincipal.Boton;
import graficoprincipal.PanelImagen;
import textual.Cliente;
import textual.Persona;
```

```
/**
 * @author Adriana Armental Tomé
 * @version 06.03.2017
 */

public class MenuListadoClientesPorDNI extends JFrame implements KeyListener,
ActionListener {

	//Botons menu
	Boton atras;
	Font fuente;
	JPanel panel, panel2, panelscrollable;
	JLabel label, etiqueta;
	JScrollPane scrollPane;
	Persona cl;

	public MenuListadoClientesPorDNI(){
		super("Taller"); //Nome
	//declarar e colocar fondo
	PanelImagen p = new PanelImagen(); //Panel que conten a imaxe
	setContentPane(p); //Asignar panel
	setSize(800,700); //Tamanho ventana
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); //Poder cerrar a ventana.
	setResizable(false); //Que non se poda cambiar o tamaanho da ventana
	setLayout(new GridBagLayout()); //Distribucion da ventana
	setLocationRelativeTo(null); //Colocar a ventana no centro

	cl = new Cliente();

	atras = new Boton("ATRÁS", 200, 110);
```

```
        atras.setForeground(Color.BLACK);
                        atras.setFocusPainted(false);
                        atras.addActionListener(this);

                        //Declaro colores para o fondo e as etiquetas dos paneles
        Color co= new Color(0);
        Color col= new Color(255,255,255);

       //Creanse paneles e daselle un color de fondo
        panel = new JPanel();
        panel.setBackground(co);

        panel2 = new JPanel();
        panel2.setBackground(co);

        label = new JLabel("LISTADO DE CLIENTES");
        label.setFont(new Font("Lucida Sans", Font.BOLD, 20));
        label.setForeground(col);

        etiqueta = new
JLabel(cl.consultarporDNI(MenuListadoClientesEscribirDNI.input.getText()));
        etiqueta.setFont(new Font("Lucida Sans", Font.PLAIN , 12));
        etiqueta.setForeground(col);

                        //Añadense as etiquetas os paneles
                        panel.add(label);
                        panel2.add(etiqueta);
                        scrollPane = new JScrollPane(panel2);

            scrollPane.setHorizontalScrollBarPolicy(ScrollPaneConstants.HORIZONTAL_SCROLLBAR
_AS_NEEDED);
```

```
scrollPane.setVerticalScrollBarPolicy(ScrollPaneConstants.VERTICAL_SCROLLBAR_AS_NEEDED);
        scrollPane.setBounds(0, 10, 770, 300);
        panelscrollable = new JPanel(null);
        panelscrollable.setPreferredSize(new Dimension(800, 700));
        panelscrollable.add(scrollPane);

                GridBagConstraints c = new GridBagConstraints();
                c.gridx=0; // especifica a coordenada x
                c.gridy=0;         // coordenada y
                c.gridwidth=1; //numero de celdas que ocupa no GridBagLayout
                c.gridheight=1;
                c.anchor=GridBagConstraints.CENTER; //posicion dentro dunha celda
                c.fill=GridBagConstraints.BOTH; //espacio que ocupara dentro dunha celda
                c.weightx=1.0; //porcentaxe de espazo libre que ocupara
                c.weighty=0;
                c.insets=new Insets(10,100,10,100); //ponher marxes
                add(panel,c);

                c.gridx=0; // especifica a coordenada x
                c.gridy=1;         // coordenada y
                c.gridwidth=1; //numero de celdas que ocupa no GridBagLayout
                c.gridheight=1;
                c.anchor=GridBagConstraints.CENTER; //posicion dentro dunha celda
                c.fill=GridBagConstraints.BOTH; //espacio que ocupara dentro dunha celda
                c.weightx=1.0; //porcentaxe de espazo libre que ocupara
                c.weighty=1.0;
                c.insets=new Insets(10,10,10,10); //ponher marxes
                add(panelscrollable,c);

                c.gridx=0;
```

```
            c.gridy=4;
            c.gridwidth=1;
            c.gridheight=1;
            c.anchor=GridBagConstraints.CENTER;
            c.fill=GridBagConstraints.BOTH;
            c.weightx=1.0;
            c.weighty=0.5;
            c.insets=new Insets(50,10,50,400);
            add(atras,c);

    }

    @Override
    public void keyTyped(KeyEvent e) {}

    //control de espacio
    @Override
    public void keyPressed(KeyEvent e) {
            int key = e.getKeyCode(); //collese a tecla soltada

            if(key == KeyEvent.VK_SPACE) { //si e o espacio
                    e.consume();
            }
    }

    @Override
    public void keyReleased(KeyEvent e) {
    }

    @Override
    public void actionPerformed(ActionEvent e) {
```

```
                if (e.getActionCommand().equals("ATRÁS")) {
                        MenuConsultaClientes men = new MenuConsultaClientes();
                        dispose();
                        men.setVisible(true);

                        //para desactivar o espacio no control de menu
                        JPanel panel = (JPanel) men.getContentPane();

                        panel.addKeyListener(this);
                        panel.setFocusable(true);
                }
        }
}
```

17.25. MenuListadoClientesEscribirNombre.java

```
package graficoclientes;

import java.awt.Color;
import java.awt.FlowLayout;
import java.awt.Font;
import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;
import java.awt.Insets;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;

import javax.swing.JFrame;
import javax.swing.JLabel;
```

```
import javax.swing.JPanel;
import javax.swing.JTextField;

import graficoprincipal.Boton;
import graficoprincipal.PanelImagen;
import textual.Cliente;
import textual.Persona;

/**
 * @author Adriana Armental Tomé
 * @version 06.03.2017
 */

public class MenuListadoClientesEscribirNombre extends JFrame implements KeyListener,
ActionListener {

	//Botons menu
	Boton atras, enviar;
	Font fuente;
	JPanel panel, panel2, panel3;
	JLabel label, label2, nombre;
	static JTextField input;
	Persona cl;

	public MenuListadoClientesEscribirNombre(){
		super("Taller"); //Nome
	//declarar e colocar fondo
	PanelImagen p = new PanelImagen(); //Panel que conten a imaxe
	setContentPane(p); //Asignar panel
	setSize(800,700); //Tamanho ventana
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); //Poder cerrar a ventana.
```

```
setResizable(false); //Que non se poda cambiar o tamaanho da ventana
setLayout(new GridBagLayout()); //Distribucion da ventana
setLocationRelativeTo(null); //Colocar a ventana no centro

cl = new Cliente();

atras = new Boton("ATRÁS", 200, 110);
atras.setForeground(Color.BLACK);
        atras.setFocusPainted(false);
        atras.addActionListener(this);
        enviar = new Boton("ENVIAR", 200, 110);
        enviar.setForeground(Color.BLACK);
        enviar.setFocusPainted(false);
        enviar.addActionListener(this);

        //Declaro colores para o fondo e as etiquetas dos paneles
Color co= new Color(0);
Color col= new Color(255,255,255);

//Creanse paneles e daselle un color de fondo
panel = new JPanel();
panel.setBackground(co);

panel2 = new JPanel();
panel2.setBackground(co);

panel3 = new JPanel();
panel3.setBackground(co);
panel3.setLayout(new FlowLayout());

label = new JLabel("BUSCADOR POR NOMBRE DE CLIENTES");
```

```
label.setFont(new Font("Lucida Sans", Font.BOLD, 20));
label.setForeground(col);

label2 = new JLabel("¿Que nombre quieres buscar?");
label2.setFont(new Font("Lucida Sans", Font.PLAIN, 14));
label2.setForeground(col);

nombre = new JLabel("Nombre");
nombre.setFont(new Font("Lucida Sans", Font.PLAIN, 14));
nombre.setForeground(col);

input = new JTextField(30);
input.setFont(new Font("Lucida Sans", Font.PLAIN , 12));
input.setForeground(col);
input.setBackground(co);
input.setEditable(true);
input.setEnabled(true);
input.setFocusable(true);
input.requestFocus();

        //Añadense as etiquetas os paneles
        panel.add(label);
        panel2.add(label2);
        panel3.add(nombre);
        panel3.add(input);

        GridBagConstraints c = new GridBagConstraints();
        c.gridx=0; // especifica a coordenada x
        c.gridy=0;         // coordenada y
        c.gridwidth=1; //numero de celdas que ocupa no GridBagLayout
        c.gridheight=1;
```

```
c.anchor=GridBagConstraints.CENTER; //posicion dentro dunha celda
c.fill=GridBagConstraints.BOTH; //espacio que ocupara dentro dunha celda
c.weightx=1.0; //porcentaxe de espazo libre que ocupara
c.weighty=0;
c.insets=new Insets(10,100,10,100); //ponher marxes
add(panel,c);

c.gridx=0; // especifica a coordenada x
c.gridy=1;          // coordenada y
c.gridwidth=1; //numero de celdas que ocupa no GridBagLayout
c.gridheight=1;
c.anchor=GridBagConstraints.CENTER; //posicion dentro dunha celda
c.fill=GridBagConstraints.BOTH; //espacio que ocupara dentro dunha celda
c.weightx=1.0; //porcentaxe de espazo libre que ocupara
c.weighty=0;
c.insets=new Insets(10,200,10,200); //ponher marxes
add(panel2,c);

c.gridx=0; // especifica a coordenada x
c.gridy=2;          // coordenada y
c.gridwidth=1; //numero de celdas que ocupa no GridBagLayout
c.gridheight=1;
c.anchor=GridBagConstraints.CENTER; //posicion dentro dunha celda
c.fill=GridBagConstraints.BOTH; //espacio que ocupara dentro dunha celda
c.weightx=1.0; //porcentaxe de espazo libre que ocupara
c.weighty=0;
c.insets=new Insets(200,100,200,100); //ponher marxes
add(panel3,c);

c.gridx=0;
c.gridy=4;
```

```
            c.gridwidth=1;
            c.gridheight=1;
            c.anchor=GridBagConstraints.CENTER;
            c.fill=GridBagConstraints.BOTH;
            c.weightx=1.0;
            c.weighty=1.0;
            c.insets=new Insets(10,10,10,400);
            add(atras,c);

            c.gridx=0;
            c.gridy=4;
            c.gridwidth=1;
            c.gridheight=1;
            c.anchor=GridBagConstraints.CENTER;
            c.fill=GridBagConstraints.BOTH;
            c.weightx=1.0;
            c.weighty=1.0;
            c.insets=new Insets(10,400,10,10);
            add(enviar,c);

    }

    @Override
    public void keyTyped(KeyEvent e) {}

    //control de espacio
    @Override
    public void keyPressed(KeyEvent e) {
            int key = e.getKeyCode(); //collese a tecla soltada

            if(key == KeyEvent.VK_SPACE) { //si e o espacio
```

```
                e.consume();
            }
        }

        @Override
        public void keyReleased(KeyEvent e) {
        }

        @Override
        public void actionPerformed(ActionEvent e) {

            if (e.getActionCommand().equals("ATRÁS")) {
                MenuConsultaClientes men = new MenuConsultaClientes();
                dispose();
                men.setVisible(true);

                //para desactivar o espacio no control de menu
                JPanel panel = (JPanel) men.getContentPane();

                panel.addKeyListener(this);
                panel.setFocusable(true);
            }
            if (e.getActionCommand().equals("ENVIAR")) {
                MenuListadoClientesPorNombre men = new
MenuListadoClientesPorNombre();
                dispose();
                men.setVisible(true);

                //para desactivar o espacio no control de menu
                JPanel panel = (JPanel) men.getContentPane();
```

```
                panel.addKeyListener(this);
                panel.setFocusable(true);
            }
        }
}
```

17.26. MenuListadoClientesPorNombre.java

```
package graficoclientes;

import java.awt.Color;
import java.awt.Dimension;
import java.awt.Font;
import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;
import java.awt.Insets;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;

import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.ScrollPaneConstants;

import graficoprincipal.Boton;
import graficoprincipal.PanelImagen;
import textual.Cliente;
import textual.Persona;
```

```
/**
 * @author Adriana Armental Tomé
 * @version 06.03.2017
 */

public class MenuListadoClientesPorNombre extends JFrame implements KeyListener,
ActionListener {

	//Botons menu
	Boton atras;
	Font fuente;
	JPanel panel, panel2, panelscrollable;
	JLabel label, etiqueta;
	JScrollPane scrollPane;
	Persona cl;

	public MenuListadoClientesPorNombre(){
		super("Taller"); //Nome
	//declarar e colocar fondo
	PanelImagen p = new PanelImagen(); //Panel que conten a imaxe
	setContentPane(p); //Asignar panel
	setSize(800,700); //Tamanho ventana
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); //Poder cerrar a ventana.
	setResizable(false); //Que non se poda cambiar o tamaanho da ventana
	setLayout(new GridBagLayout()); //Distribucion da ventana
	setLocationRelativeTo(null); //Colocar a ventana no centro

	cl = new Cliente();

	atras = new Boton("ATRÁS", 200, 110);
	atras.setForeground(Color.BLACK);
```

```
                atras.setFocusPainted(false);
                atras.addActionListener(this);

                //Declaro colores para o fondo e as etiquetas dos paneles
        Color co= new Color(0);
        Color col= new Color(255,255,255);

       //Creanse paneles e daselle un color de fondo
        panel = new JPanel();
        panel.setBackground(co);

        panel2 = new JPanel();
        panel2.setBackground(co);

        label = new JLabel("LISTADO DE CLIENTES");
        label.setFont(new Font("Lucida Sans", Font.BOLD, 20));
        label.setForeground(col);

        etiqueta = new
JLabel(cl.consultarporNombre(MenuListadoClientesEscribirNombre.input.getText()));
        etiqueta.setFont(new Font("Lucida Sans", Font.PLAIN , 12));
        etiqueta.setForeground(col);

                //Añadense as etiquetas os paneles
                panel.add(label);
                panel2.add(etiqueta);
                scrollPane = new JScrollPane(panel2);

          scrollPane.setHorizontalScrollBarPolicy(ScrollPaneConstants.HORIZONTAL_SCROLLBAR
_AS_NEEDED);

scrollPane.setVerticalScrollBarPolicy(ScrollPaneConstants.VERTICAL_SCROLLBAR_AS_NEEDED);
```

```
scrollPane.setBounds(0, 10, 770, 300);
panelscrollable = new JPanel(null);
panelscrollable.setPreferredSize(new Dimension(800, 700));
panelscrollable.add(scrollPane);

        GridBagConstraints c = new GridBagConstraints();
        c.gridx=0; // especifica a coordenada x
        c.gridy=0;         // coordenada y
        c.gridwidth=1; //numero de celdas que ocupa no GridBagLayout
        c.gridheight=1;
        c.anchor=GridBagConstraints.CENTER; //posicion dentro dunha celda
        c.fill=GridBagConstraints.BOTH; //espacio que ocupara dentro dunha celda
        c.weightx=1.0; //porcentaxe de espazo libre que ocupara
        c.weighty=0;
        c.insets=new Insets(10,100,10,100); //ponher marxes
        add(panel,c);

        c.gridx=0; // especifica a coordenada x
        c.gridy=1;         // coordenada y
        c.gridwidth=1; //numero de celdas que ocupa no GridBagLayout
        c.gridheight=1;
        c.anchor=GridBagConstraints.CENTER; //posicion dentro dunha celda
        c.fill=GridBagConstraints.BOTH; //espacio que ocupara dentro dunha celda
        c.weightx=1.0; //porcentaxe de espazo libre que ocupara
        c.weighty=1.0;
        c.insets=new Insets(10,10,10,10); //ponher marxes
        add(panelscrollable,c);

        c.gridx=0;
        c.gridy=4;
        c.gridwidth=1;
```

```
            c.gridheight=1;
            c.anchor=GridBagConstraints.CENTER;
            c.fill=GridBagConstraints.BOTH;
            c.weightx=1.0;
            c.weighty=0.5;
            c.insets=new Insets(50,10,50,400);
            add(atras,c);

    }

    @Override
    public void keyTyped(KeyEvent e) {}

    //control de espacio
    @Override
    public void keyPressed(KeyEvent e) {
            int key = e.getKeyCode(); //collese a tecla soltada

            if(key == KeyEvent.VK_SPACE) { //si e o espacio
                    e.consume();
            }
    }

    @Override
    public void keyReleased(KeyEvent e) {
    }

    @Override
    public void actionPerformed(ActionEvent e) {

            if (e.getActionCommand().equals("ATRÁS")) {
```

```
                    MenuConsultaClientes men = new MenuConsultaClientes();
                    dispose();
                    men.setVisible(true);

                    //para desactivar o espacio no control de menu
                    JPanel panel = (JPanel) men.getContentPane();

                    panel.addKeyListener(this);
                    panel.setFocusable(true);
                }
            }
}
```

17.27. MenuListadoClientesEscribirApellidos.java

```
package graficoclientes;

import java.awt.Color;
import java.awt.FlowLayout;
import java.awt.Font;
import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;
import java.awt.Insets;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;

import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JTextField;
```

```
import graficoprincipal.Boton;
import graficoprincipal.PanelImagen;
import textual.Cliente;
import textual.Persona;

/**
 * @author Adriana Armental Tomé
 * @version 06.03.2017
 */

public class MenuListadoClientesEscribirApellidos extends JFrame implements KeyListener,
ActionListener {

        //Botons menu
        Boton atras, enviar;
        Font fuente;
        JPanel panel, panel2, panel3;
        JLabel label, label2, apellidos;
        static JTextField input;
        Persona cl;

        public MenuListadoClientesEscribirApellidos(){
                super("Taller"); //Nome
    //declarar e colocar fondo
    PanelImagen p = new PanelImagen(); //Panel que conten a imaxe
    setContentPane(p); //Asignar panel
    setSize(800,700); //Tamanho ventana
                setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); //Poder cerrar a ventana.
    setResizable(false); //Que non se poda cambiar o tamaanho da ventana
    setLayout(new GridBagLayout()); //Distribucion da ventana
```

```
setLocationRelativeTo(null); //Colocar a ventana no centro

cl = new Cliente();

atras = new Boton("ATRÁS", 200, 110);
atras.setForeground(Color.BLACK);
        atras.setFocusPainted(false);
        atras.addActionListener(this);
        enviar = new Boton("ENVIAR", 200, 110);
        enviar.setForeground(Color.BLACK);
        enviar.setFocusPainted(false);
        enviar.addActionListener(this);

        //Declaro colores para o fondo e as etiquetas dos paneles
Color co= new Color(0);
Color col= new Color(255,255,255);

//Creanse paneles e daselle un color de fondo
panel = new JPanel();
panel.setBackground(co);

panel2 = new JPanel();
panel2.setBackground(co);

panel3 = new JPanel();
panel3.setBackground(co);
panel3.setLayout(new FlowLayout());

label = new JLabel("BUSCADOR POR APELLIDOS DE CLIENTES");
label.setFont(new Font("Lucida Sans", Font.BOLD, 20));
label.setForeground(col);
```

```
label2 = new JLabel("¿Que apellidos quieres buscar?");
label2.setFont(new Font("Lucida Sans", Font.PLAIN, 14));
label2.setForeground(col);

apellidos = new JLabel("Apellidos");
apellidos.setFont(new Font("Lucida Sans", Font.PLAIN, 14));
apellidos.setForeground(col);

input = new JTextField(30);
input.setFont(new Font("Lucida Sans", Font.PLAIN , 12));
input.setForeground(col);
input.setBackground(co);
input.setEditable(true);
input.setEnabled(true);
input.setFocusable(true);
input.requestFocus();

        //Añadense as etiquetas os paneles
        panel.add(label);
        panel2.add(label2);
        panel3.add(apellidos);
        panel3.add(input);

        GridBagConstraints c = new GridBagConstraints();
        c.gridx=0; // especifica a coordenada x
        c.gridy=0;         // coordenada y
        c.gridwidth=1; //numero de celdas que ocupa no GridBagLayout
        c.gridheight=1;
        c.anchor=GridBagConstraints.CENTER; //posicion dentro dunha celda
        c.fill=GridBagConstraints.BOTH; //espacio que ocupara dentro dunha celda
```

```
c.weightx=1.0; //porcentaxe de espazo libre que ocupara
c.weighty=0;
c.insets=new Insets(10,100,10,100); //ponher marxes
add(panel,c);

c.gridx=0; // especifica a coordenada x
c.gridy=1;          // coordenada y
c.gridwidth=1; //numero de celdas que ocupa no GridBagLayout
c.gridheight=1;
c.anchor=GridBagConstraints.CENTER; //posicion dentro dunha celda
c.fill=GridBagConstraints.BOTH; //espacio que ocupara dentro dunha celda
c.weightx=1.0; //porcentaxe de espazo libre que ocupara
c.weighty=0;
c.insets=new Insets(10,200,10,200); //ponher marxes
add(panel2,c);

c.gridx=0; // especifica a coordenada x
c.gridy=2;          // coordenada y
c.gridwidth=1; //numero de celdas que ocupa no GridBagLayout
c.gridheight=1;
c.anchor=GridBagConstraints.CENTER; //posicion dentro dunha celda
c.fill=GridBagConstraints.BOTH; //espacio que ocupara dentro dunha celda
c.weightx=1.0; //porcentaxe de espazo libre que ocupara
c.weighty=0;
c.insets=new Insets(200,100,200,100); //ponher marxes
add(panel3,c);

c.gridx=0;
c.gridy=4;
c.gridwidth=1;
c.gridheight=1;
```

```
            c.anchor=GridBagConstraints.CENTER;
            c.fill=GridBagConstraints.BOTH;
            c.weightx=1.0;
            c.weighty=1.0;
            c.insets=new Insets(10,10,10,400);
            add(atras,c);

            c.gridx=0;
            c.gridy=4;
            c.gridwidth=1;
            c.gridheight=1;
            c.anchor=GridBagConstraints.CENTER;
            c.fill=GridBagConstraints.BOTH;
            c.weightx=1.0;
            c.weighty=1.0;
            c.insets=new Insets(10,400,10,10);
            add(enviar,c);

    }

    @Override
    public void keyTyped(KeyEvent e) {}

    //control de espacio
    @Override
    public void keyPressed(KeyEvent e) {
            int key = e.getKeyCode(); //collese a tecla soltada

            if(key == KeyEvent.VK_SPACE) { //si e o espacio
                    e.consume();
            }
```

```
    }

    @Override
    public void keyReleased(KeyEvent e) {
    }

    @Override
    public void actionPerformed(ActionEvent e) {

        if (e.getActionCommand().equals("ATRÁS")) {
            MenuConsultaClientes men = new MenuConsultaClientes();
            dispose();
            men.setVisible(true);

            //para desactivar o espacio no control de menu
            JPanel panel = (JPanel) men.getContentPane();

            panel.addKeyListener(this);
            panel.setFocusable(true);
        }
        if (e.getActionCommand().equals("ENVIAR")) {
            MenuListadoClientesPorApellidos men = new
MenuListadoClientesPorApellidos();
            dispose();
            men.setVisible(true);

            //para desactivar o espacio no control de menu
            JPanel panel = (JPanel) men.getContentPane();

            panel.addKeyListener(this);
            panel.setFocusable(true);
```

```
                }
        }
}
```

17.28. MenuListadoClientesPorApellidos.java

```
package graficoclientes;

import java.awt.Color;
import java.awt.Dimension;
import java.awt.Font;
import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;
import java.awt.Insets;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;

import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.ScrollPaneConstants;

import graficoprincipal.Boton;
import graficoprincipal.PanelImagen;
import textual.Cliente;
import textual.Persona;

/**
 * @author Adriana Armental Tomé
```

```
 * @version 06.03.2017
 */

public class MenuListadoClientesPorApellidos extends JFrame implements KeyListener,
ActionListener {

	//Botons menu
	Boton atras;
	Font fuente;
	JPanel panel, panel2, panelscrollable;
	JLabel label, etiqueta;
	JScrollPane scrollPane;
	Persona cl;

	public MenuListadoClientesPorApellidos(){
		super("Taller"); //Nome
    //declarar e colocar fondo
    PanelImagen p = new PanelImagen(); //Panel que conten a imaxe
    setContentPane(p); //Asignar panel
    setSize(800,700); //Tamanho ventana
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); //Poder cerrar a ventana.
    setResizable(false); //Que non se poda cambiar o tamaanho da ventana
    setLayout(new GridBagLayout()); //Distribucion da ventana
    setLocationRelativeTo(null); //Colocar a ventana no centro

    cl = new Cliente();

    atras = new Boton("ATRÁS", 200, 110);
    atras.setForeground(Color.BLACK);
		atras.setFocusPainted(false);
		atras.addActionListener(this);
```

```
                    //Declaro colores para o fondo e as etiquetas dos paneles
        Color co= new Color(0);
        Color col= new Color(255,255,255);

      //Creanse paneles e daselle un color de fondo
        panel = new JPanel();
        panel.setBackground(co);

        panel2 = new JPanel();
        panel2.setBackground(co);

        label = new JLabel("LISTADO DE CLIENTES");
        label.setFont(new Font("Lucida Sans", Font.BOLD, 20));
        label.setForeground(col);

        etiqueta = new
JLabel(cl.consultarporApellidos(MenuListadoClientesEscribirApellidos.input.getText()));
        etiqueta.setFont(new Font("Lucida Sans", Font.PLAIN , 12));
        etiqueta.setForeground(col);

                    //Añadense as etiquetas os paneles
                    panel.add(label);
                    panel2.add(etiqueta);
                    scrollPane = new JScrollPane(panel2);

          scrollPane.setHorizontalScrollBarPolicy(ScrollPaneConstants.HORIZONTAL_SCROLLBAR
_AS_NEEDED);

scrollPane.setVerticalScrollBarPolicy(ScrollPaneConstants.VERTICAL_SCROLLBAR_AS_NEEDED);
        scrollPane.setBounds(0, 10, 770, 300);
        panelscrollable = new JPanel(null);
```

```
panelscrollable.setPreferredSize(new Dimension(800, 700));
panelscrollable.add(scrollPane);

        GridBagConstraints c = new GridBagConstraints();
        c.gridx=0; // especifica a coordenada x
        c.gridy=0;         // coordenada y
        c.gridwidth=1; //numero de celdas que ocupa no GridBagLayout
        c.gridheight=1;
        c.anchor=GridBagConstraints.CENTER; //posicion dentro dunha celda
        c.fill=GridBagConstraints.BOTH; //espacio que ocupara dentro dunha celda
        c.weightx=1.0; //porcentaxe de espazo libre que ocupara
        c.weighty=0;
        c.insets=new Insets(10,100,10,100); //ponher marxes
        add(panel,c);

        c.gridx=0; // especifica a coordenada x
        c.gridy=1;         // coordenada y
        c.gridwidth=1; //numero de celdas que ocupa no GridBagLayout
        c.gridheight=1;
        c.anchor=GridBagConstraints.CENTER; //posicion dentro dunha celda
        c.fill=GridBagConstraints.BOTH; //espacio que ocupara dentro dunha celda
        c.weightx=1.0; //porcentaxe de espazo libre que ocupara
        c.weighty=1.0;
        c.insets=new Insets(10,10,10,10); //ponher marxes
        add(panelscrollable,c);

        c.gridx=0;
        c.gridy=4;
        c.gridwidth=1;
        c.gridheight=1;
        c.anchor=GridBagConstraints.CENTER;
```

```
            c.fill=GridBagConstraints.BOTH;
            c.weightx=1.0;
            c.weighty=0.5;
            c.insets=new Insets(50,10,50,400);
            add(atras,c);

    }

    @Override
    public void keyTyped(KeyEvent e) {}

    //control de espacio
    @Override
    public void keyPressed(KeyEvent e) {
            int key = e.getKeyCode(); //collese a tecla soltada

            if(key == KeyEvent.VK_SPACE) { //si e o espacio
                    e.consume();
            }
    }

    @Override
    public void keyReleased(KeyEvent e) {
    }

    @Override
    public void actionPerformed(ActionEvent e) {

            if (e.getActionCommand().equals("ATRÁS")) {
                    MenuConsultaClientes men = new MenuConsultaClientes();
                    dispose();
```

```
                men.setVisible(true);


                //para desactivar o espacio no control de menu
                JPanel panel = (JPanel) men.getContentPane();


                panel.addKeyListener(this);
                panel.setFocusable(true);
            }
        }
}
```