|  |
| --- |
|  |

|  |
| --- |
| **Creación de un Sistema de Reserva En Java Con MySQL Server** |
| [Escriba el subtítulo del documento] |
|  |
| Contiene el desarrollo de una aplicación de escritorio hecha para un hotel. Comenzando desde como crear una base de datos en MySQL workbench, implementarla en un servidor XAMPP, desarrollar la aplicación en Netbeans 8.2 con Java 8 e implementarla en una computadora o equipo servidor a nivel local o en red. |
|  |
| **Luis Ángel Vanegas Martinez** |
| **08/01/2018** |
|  |

Contenido

[ARQUITECTURA DEL SISTEMA CON PATRÓN MVC 2](#_Toc503190743)

[CREAR BASE DE DATOS A TRAVÉS DEL MODELO ENTIDAD-RELACIÓN (ER) EN MYSQL WORKBENCH 3](#_Toc503190744)

[Crear tablas 4](#_Toc503190745)

[Crear relaciones 5](#_Toc503190746)

[Exportar script de la base de datos 7](#_Toc503190747)

[Importar la base de datos creada en MySQL WorkBench a phpMyAdmin 8](#_Toc503190748)

[CREAR LA APLICACIÓN EN NETBEANS 9](#_Toc503190749)

[1. Creamos un proyecto Java Application 9](#_Toc503190750)

[2. Creamos los paquetes: logica, datos, presentación, files(para guardar imágenes a utilizar) 10](#_Toc503190751)

[3. Adjuntamos el JDBC para podernos conectar a la base de datos de MySQL 10](#_Toc503190752)

[4. creamos cada una de las clases correspondientes a cada paquete 10](#_Toc503190753)

[Paquete Datos (Modelo) 10](#_Toc503190754)

[Paquete Logica (Controlador) 11](#_Toc503190755)

[Paquete Presentación (Vista) 15](#_Toc503190756)

[Agregar un control Jcalendar mediante una librería: 29](#_Toc503190757)

[CREAR REPORTES EN JAVA CON JASPER REPORTS Y IREPORT DESIGNER 31](#_Toc503190758)

[Desarrollar reportes en iReport 35](#_Toc503190759)

[Implementar el reporte en netbeans 35](#_Toc503190760)

[GENERAR ARCHIVOS DE INSTALACION DEL SISTEMA 37](#_Toc503190761)

[1. Exportar nuestra base de datos: 37](#_Toc503190762)

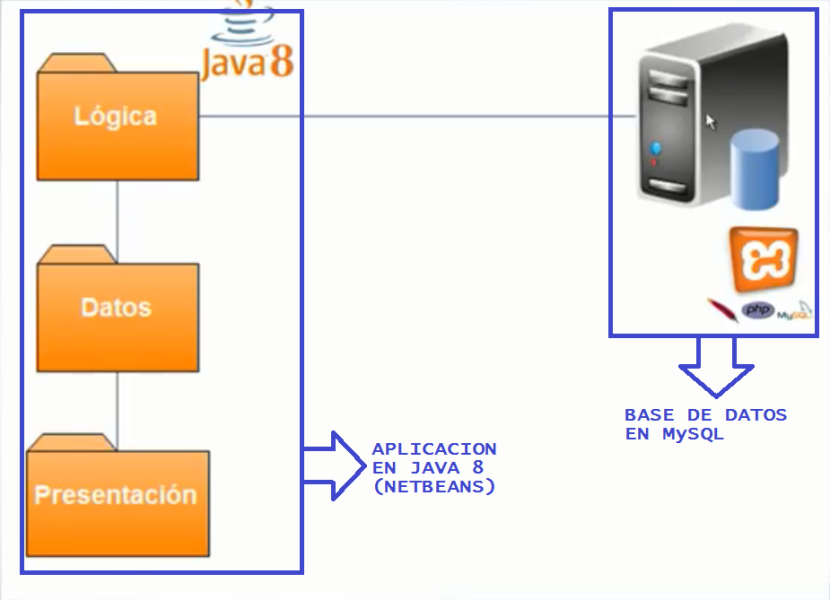
[2. Configuracion de la aplicación antes de generar el .jar 37](#_Toc503190763)

[3. Crear el .jar 39](#_Toc503190764)

[IMPLEMENTACION DEL SISTEMA DE RESERVA EN UNA COMPUTADORA O RED 40](#_Toc503190765)

[Configurando la maquina servidor 40](#_Toc503190766)

# ARQUITECTURA DEL SISTEMA CON PATRÓN MVC



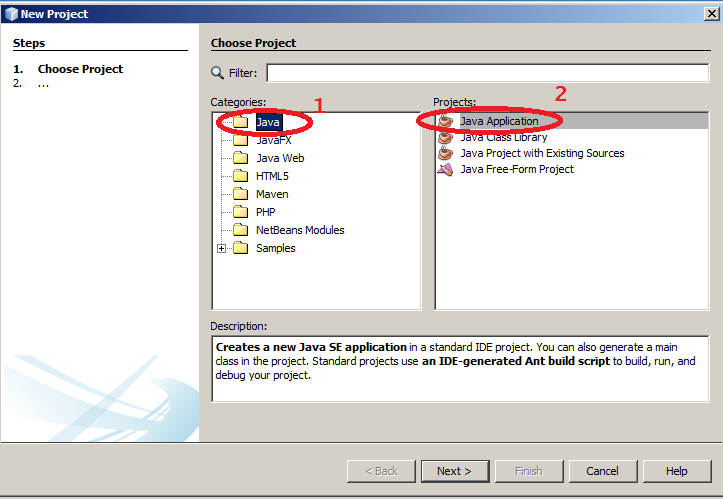
* Paquete Datos (Modelo):
* Clases persistentes (una clase por cada tabla en la base de datos) con sus metodos getter, setter y constructores de objetos
* Paquete Lógica (Controlador):
* Clase de conexión a la base de datos
* Clases con los métodos para poder insertar, modificar, eliminar, mostrar, filtrar registros de las tablas de la base de datos y funciones tales como para controlar accesos, entre otras.
* Paquete Presentación (Vista):
* Interfaces
* Formularios de registros de datos (formularios que sirven para almacenar datos a la BD)
* Formularios vistas (son los formulario que permiten realizar busquedas, ya sea de habitación, cliente y productos)
* Formularios de accesos (formulario de loggeo)
* Menú del sistema (formulario MDI )
* Imágenes (contiene la imágenes utilizadas por el sistema)
* Reportes (contiene el formato de los reportes )
* Servidor: como servidor utilizamos XAMPP con sus conexiones Apache y MySQL. El cual tendrá como motor de base de datos MySQL

Herramientas Extras

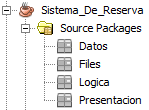
* Librería JDBC MySQL: se encarga de conectar la aplicación en Java 8 con el servidor en MySQL
* IDE iReport Designer: nos permite crear los reportes utilizando sus librerias de Jasper Reports
* Librerías Jasper Reports: se encarga de enlazar e imprimir los reportes

# CREAR LA APLICACIÓN EN NETBEANS

## Creamos un proyecto Java Application



## Creamos los paquetes: logica, datos, presentación, files(para guardar imágenes a utilizar)

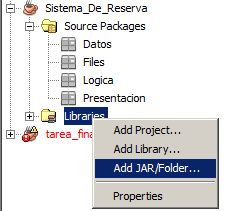


## Adjuntamos el JDBC para podernos conectar a la base de datos de MySQL

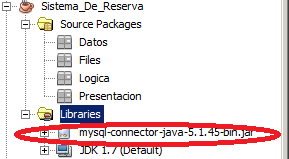
* 1. Descargamos nuestro JDBC escribiendo en google JDBC MYSQL y descargamos desde la pagina principal el archivo .zip y lo descomprimimos.

NOTA: el conector debe de tener un nombre similar a: mysql-connector-java-5.1.45-bin.jar y este es el que se va importar en nuestro proyecto.

* 1. vamos a la carpeta del proyecto que se llama Libraries y damos clic derecho – Add JAR/FOLDER…



* 1. buscamos la direccion del conector jdbc y lo importamos. Luego nos debe de aparecer:



NOTA: la librería debe de estar en la carpeta fuente de la aplicación

## creamos cada una de las clases correspondientes a cada paquete

### Paquete Datos (Modelo)

Estas clases cuentan con una única estructura en la cual solo varía los atributos de cada clase. Por cada tabla de nuestra base de datos, haremos una clase parecida a la clase que tenemos como ejemplo de Estructura

#### Estructura detallada en codigo

|  |
| --- |
| **package** Datos;  /\*\*\* **@author** Luis Angel\*/  /\* esta es una clase persistente que nos permitirá formar los objetos que  serán almacenados en la tabla habitación  \*/  **public** **class** vHabitacion {  **private** **int** id\_habitacion;  **private** String numero;  **private** String piso;  **private** String descripcion;  **private** String caracteristicas;  **private** **double** precio\_diario;  **private** String estado;  **private** String tipo\_habitacion;  //constructor vacio  //constructor  //getter and setter  } |

### Paquete Logica (Controlador)

Estas clases se comunican directamente con la base de datos y ejecutan funciones de tipo CRUD

#### Estructura detallada en codigo

La primera estructura es de la clase *conexión* que nos permite conectarnos con la base de datos:

|  |
| --- |
| **package** Logica;  **import** java.sql.Connection;  **import** java.sql.DriverManager;  **import** java.sql.SQLException;  **import** javax.swing.JOptionPane;  /\* @author Luis Angel \*/  **public** **class** conexion {  **public** String db="sistemareservadb";//nombre de la base de datos  **public** String url="jdbc:mysql://127.0.0.1/" +db;/\*en esta dirección puede ir la ip  o el nombre del servidor que es localhost o si estamos en red, le indicamos la ip  de la computadora que contiene el servidor\*/  **public** String user="root";//usuario de nuestro administrador de bd  **public** String pass="";//la contraseña del administrador de bd  **public** conexion() {  }    **public** Connection conectar(){  Connection link=**null**;//variable que hace instancia a la clase conecction  **try** {  /\*cargamos el driver de la conexion jdbc \*/  Class.forName("org.gjt.mm.mysql.Driver");  /\*para obtener la ruta que va en forName(), ampliamos nuestro  mysql-connectr-java-5.1.45 que se encuentra en Libraries y nos vamos  al final en org.gjt.mm.mysql la ruta sería: org.gjt.mm.mysql.Driver\*/    /\*le decimos a la variable  link que cree un enlace hacia la base de datos\*/  link=DriverManager.getConnection(**this**.url, **this**.user, **this**.pass);  } **catch** (ClassNotFoundException | SQLException e) {  JOptionPane.showConfirmDialog(**null**, e);  }  **return** link;//al final se debe de devolver la cadena de conexion  }  } |

La siguiente estructura es propia de aquellas clases principales y que derivan de las tablas de nuestra base de datos, solo varíarian algunos nombres de atributos y comportamientos funcionales:

|  |
| --- |
| **package** Logica;  **import** Datos.vhabitacion;  **import** java.sql.\*;  **import** javax.swing.JOptionPane;  **import** javax.swing.table.DefaultTableModel;  /\*\*\* **@author** Luis Angel\*/  **public** **class** fhabitacion {  **private** conexion mysql=**new** conexion();//creamos una instancia a la cadena de conexion  **private** Connection cn=mysql.conectar();//llamamos al metodo conectar de la clase conexion  **private** String sSQL="";//variable donde se almacenará la cadena de conexion  **public** **int** totalRegistros;//variable que cuenta el total de registro que tiene la consulta  /\*metodo para mostrar los registros de la tabla habitacion  \*/  **public** DefaultTableModel mostrar(String buscar){  DefaultTableModel modelo ;  //vector que contendrá los cabezales de la tabla  String [] titulos = {"ID", "Número", "Piso", "Descripción",  "Caracteristicas", "Precio", "Estado", "Tipo Habitacion"};  String [] registro = **new** String [8];//vector que contendrá los registros  totalRegistros=0;  modelo = **new** DefaultTableModel(**null**,titulos);/\*le llevamos los titulos al modelo\*/    /\* la siguiente linea es la instrucion que obtendrá todos los registros  de la habitacion (consulta sql)  \*/  sSQL = "select \* from habitacion where piso like '%"+ buscar +"%' "  + "order by idhabitacion";  **try** {  Statement st = cn.createStatement();//declara la conexion  ResultSet rs = st.executeQuery(sSQL);//ejecuta la consulta  /\* vamos a recorrer uno a uno cada registro obtenido al ejecutar  la consulta  \*/  **while**(rs.next()){//recorre rs (resultado)  /\*almacenamos uno a uno en el vector registro, todos los  registros consultados en la base de datos.  ponemos el campo tal y como esta en la bd  \*/  registro[0]=rs.getString("idhabitacion");  registro[1]=rs.getString("numero");  registro[2]=rs.getString("piso");  registro[3]=rs.getString("descripcion");  registro[4]=rs.getString("caracteristicas");  registro[5]=rs.getString("precio\_diario");  registro[6]=rs.getString("estado");  registro[7]=rs.getString("tipo\_habitacion");  totalRegistros ++;  modelo.addRow(registro);  }  **return** modelo;  } **catch** (Exception e) {  JOptionPane.showConfirmDialog(**null**, e);  **return** **null**;  }  }  /\* metodo para insertar datos a la tabla habitaciones de la base de datos  \*/  **public** **boolean** insertar (vhabitacion dts){  //consulta sql de inserción de datos  sSQL="insert into habitacion (numero, piso, descripcion, "  + "caracteristicas, precio\_diario, estado, tipo\_habitacion) "  + "values (?,?,?,?,?,?,?)";    **try** {  //se almacena en el pst la consulta  PreparedStatement pst=cn.prepareCall(sSQL);  /\*al pst se le almacena los valores de acuerdo al indice (indice de acuerdo  con los ?,?,?) en su mismo orden \*/  pst.setString(1, dts.getNumero());  pst.setString(2, dts.getPiso());  pst.setString(3, dts.getDescripcion());  pst.setString(4, dts.getCaracteristicas());  pst.setDouble(5, dts.getPrecio\_diario());  pst.setString(6, dts.getEstado());  pst.setString(7, dts.getTipo\_habitacion());  **int** n=pst.executeUpdate();/\*el executeUpdate ejecuta la consulta y la  variable n almacenará el resultado de ejecucion del statement \*/    /\*condicion si se cumple o no la insercion de registros  en la tabla habitaciones  \*/  **if**(n!=0){//cuando es cero, quiere decir que no se ejecuto ni 1 consulta  **return** **true**;  }**else**{  **return** **false**;  }  } **catch** (Exception e) {  JOptionPane.showInputDialog(**null**, e);  **return** **false**;  }  }  /\* metodo para editar datos de la tabla habitaciones de la base de datos  \*/  **public** **boolean** editar (vhabitacion dts){  //consulta sql de inserción de datos  sSQL="update habitacion set numero=?, piso=?, descripcion=?, "  + "caracteristicas=?, precio\_diario=?, estado=?, tipo\_habitacion=? "  + "where idhabitacion=?";  **try** {  //se almacena en el pst la consulta  PreparedStatement pst=cn.prepareCall(sSQL);  /\*al pst se le almacena los valores de acuerdo al indice (indice de acuerdo  con los ?,?,?) en su mismo orden \*/  pst.setString(1, dts.getNumero());  pst.setString(2, dts.getPiso());  pst.setString(3, dts.getDescripcion());  pst.setString(4, dts.getCaracteristicas());  pst.setDouble(5, dts.getPrecio\_diario());  pst.setString(6, dts.getEstado());  pst.setString(7, dts.getTipo\_habitacion());  pst.setInt(8, dts.getId\_habitacion());  **int** n=pst.executeUpdate();/\*el executeUpdate ejecuta la consulta y la  variable n almacenará el resultado de ejecucion del statement \*/    /\*condicion si se cumple o no la insercion de registros  en la tabla habitaciones  \*/  **if**(n!=0){//cuando es cero, quiere decir que no se ejecuto ni 1 consulta  **return** **true**;  }**else**{  **return** **false**;  }  } **catch** (Exception e) {  JOptionPane.showInputDialog(**null**, e);  **return** **false**;  }  }  /\* metodo que elimina datos de la tabla habitacion de la base de datos  \*/  **public** **boolean** eliminar(vhabitacion dts){  //consulta sql de inserción de datos  sSQL="delete from habitacion where idhabitacion=?";  **try** {  //se almacena en el pst la consulta  PreparedStatement pst=cn.prepareCall(sSQL);  /\*al pst se le almacena los valores de acuerdo al indice (indice de acuerdo  con los ?,?,?) en su mismo orden \*/  pst.setInt(1, dts.getId\_habitacion());  **int** n=pst.executeUpdate();/\*el executeUpdate ejecuta la consulta y la  variable n almacenará el resultado de ejecucion del statement \*/    /\*condicion si se cumple o no la insercion de registros  en la tabla habitaciones  \*/  **if**(n!=0){  **return** **true**;  }**else**{  **return** **false**;  }  } **catch** (Exception e) {  JOptionPane.showInputDialog(**null**, e);  **return** **false**;  }  }  } |

### Paquete Presentación (Vista)

#### Formularios principales

NOTA: Cada formulario que hace referencia a las tablas de nuestra base de datos cuenta basicamente con la misma estructura del siguiente ejemplo: (solo varían algunos campos, nombres de estos y en su mayor caso su comportamiento funcional):

**Estructura detallada en codigo**

|  |
| --- |
| **package** Presentacion;  /\*\*\* **@author** Luis Angel\*/  **public** **class** frmhabitacion **extends** javax.swing.JFrame {  /\*constructor, se ejecuta al invocar al formulario y muestra todos los registros en la tabla  E inhabilita todos los controles mediante el metodo inhabilitar()\*/  **public** frmhabitacion() {  initComponents();//este metodo viene por defecto  mostrar("");//metodo mostrar con parametro en blanco para que muestre todos  inhabilitar();  }  /\*desde un principio, el atributo accion será guardar, si se desea aditar, mas adelante el  atributo accion cambia de valor\*/  **private** String accion = "guardar";  /\*metodo para ocultar las columnas que no quiero visualizar en la tabla  en este caso solo la columna cero que es donde va el ID  \*/  **void** ocultar\_columnas() {  //el maximo y minimo tamaño de la columna cero (0) será 0  tablalistado.getColumnModel().getColumn(0).setMaxWidth(0);  tablalistado.getColumnModel().getColumn(0).setMinWidth(0);  tablalistado.getColumnModel().getColumn(0).setPreferredWidth(0);  }  /\*metodo para desabilitar cajas de textos y controles para ciertas operaciones  \*/  **void** inhabilitar() {  txtidhabitacion.setVisible(**false**);  cbopiso.setEnabled(**false**);  txtnumero.setEnabled(**false**);  txtdescripcion.setEnabled(**false**);  txtcaracteristicas.setEnabled(**false**);  txtpreciodiario.setEnabled(**false**);  cboestado.setEnabled(**false**);  cbotipohabitacion.setEnabled(**false**);  btnguardar.setEnabled(**false**);  btncancelar.setEnabled(**false**);  btneliminar.setEnabled(**false**);  txtidhabitacion.setText("");  txtpreciodiario.setText("");  txtcaracteristicas.setText("");  txtdescripcion.setText("");  }  /\*metodo para habilitar cajas de textos y controles para ciertas operaciones  \*/  **void** habilitar() {  txtidhabitacion.setVisible(**false**);  cbopiso.setEnabled(**true**);  txtnumero.setEnabled(**true**);  txtdescripcion.setEnabled(**true**);  txtcaracteristicas.setEnabled(**true**);  txtpreciodiario.setEnabled(**true**);  cboestado.setEnabled(**true**);  cbotipohabitacion.setEnabled(**true**);  btnguardar.setEnabled(**true**);  btncancelar.setEnabled(**true**);  btneliminar.setEnabled(**true**);  txtidhabitacion.setText("");  txtpreciodiario.setText("");  txtcaracteristicas.setText("");  txtdescripcion.setText("");  }  /\*metodo que permite mostrar datos en la tabla\*/  **void** mostrar(String buscar) {  **try** {  DefaultTableModel modelo;  fHabitacion func = **new** fHabitacion();  modelo = func.mostrar(buscar);//ejecutamos el metodo mostrar de fHabitacion  tablalistado.setModel(modelo);//en la tabla almacenamos lo que retornó el metodo mostrar()  ocultar\_columnas();  lbltotalregistros.setText("Total Registros " + Integer.toString(func.totalRegistros));  } **catch** (Exception e) {  JOptionPane.showConfirmDialog(rootPane, e);  }  }  @SuppressWarnings("unchecked")  /\*metodo que se ejecuta al activar el boton nuevo  \*/  **private** **void** btnnuevoActionPerformed(java.awt.event.ActionEvent evt) {  habilitar(); //habilita los controles de entrada de datos  btnguardar.setText("Guardar"); //el boton guardar cambia de nombre  accion = "guardar"; //el atrbituo accion cambia a guardar  }  /\*metodo que se ejecuta al activar el boton guardar, este boton no solo sirve para guardar sino  tambien para editar  \*/  **private** **void** btnguardarActionPerformed(java.awt.event.ActionEvent evt) {  //valido que todas las cajas de texto tengan datos digitados  **if** (txtnumero.getText().length() == 0) {  JOptionPane.showConfirmDialog(rootPane, "debes ingresar un numero de habitacion");  txtnumero.requestFocus();//poner el foco en el campo de texto  **return**;  }  **if** (txtdescripcion.getText().length() == 0) {  JOptionPane.showConfirmDialog(rootPane, "debes ingresar una descripcion para la habitacion");  txtdescripcion.requestFocus();//poner el foco en el campo de texto  **return**;  }  **if** (txtpreciodiario.getText().length() == 0) {  JOptionPane.showConfirmDialog(rootPane, "debes ingresar un precio diario para la habitacion");  txtpreciodiario.requestFocus();//poner el foco en el campo de texto  **return**;  }  **if** (txtcaracteristicas.getText().length() == 0) {  JOptionPane.showConfirmDialog(rootPane, "debes ingresar una caracteristica para la habitacion");  txtcaracteristicas.requestFocus();//poner el foco en el campo de texto  **return**;  }  vHabitacion dts = **new** vHabitacion();  fHabitacion func = **new** fHabitacion();  //el siguiente bloque de codigo es para almacenar en la clase molde, los datos del objeto  dts.setNumero(txtnumero.getText());  **int** seleccionado = cbopiso.getSelectedIndex();//seleccionado almacenará el indice seleccionado en el combo  dts.setPiso((String) cbopiso.getItemAt(seleccionado));//en el dts se almacena el texto del indice seleccionado en el combo  dts.setDescripcion(txtdescripcion.getText());  dts.setCaracteristicas(txtcaracteristicas.getText());  dts.setPrecio\_diario(Double.parseDouble(txtpreciodiario.getText()));  seleccionado = cboestado.getSelectedIndex();  dts.setEstado((String) cboestado.getItemAt(seleccionado));  seleccionado = cbotipohabitacion.getSelectedIndex();  dts.setTipo\_habitacion((String) cbotipohabitacion.getItemAt(seleccionado));  /\*si la accion es guardar, se ejecuta el metodo insertar de la clase controladora, a cual dicho metodo  se le llevan los datos que se almacenaron en la clase moldeadora del objeto dts\*/  **if** (accion.equals("guardar")) {  **if** (func.insertar(dts)) {//si si se almacena, entonces  JOptionPane.showMessageDialog(rootPane, "la habitacion fue registrada satisfactoriamente");  mostrar("");//muestra todos los registros  inhabilitar();//inhabilita los controles  }  //con la accion editar pasa similar a lo anterior, solo que este invoca al metodo editar  } **else** **if** (accion.equals("editar")) {  dts.setId\_habitacion(Integer.parseInt(txtidhabitacion.getText()));  **if** (func.editar(dts)) {  JOptionPane.showMessageDialog(rootPane, "la habitacion fue editada satisfactoriamente");  mostrar("");  inhabilitar();  }  }  }  /\* metodo del boton eliminar, el cual invoca al metodo eliminar de fhabitacion  mediante func  \*/  **private** **void** btneliminarActionPerformed(java.awt.event.ActionEvent evt) {  **if** (!txtidhabitacion.getText().equals("")) {//si no es blanco el id de la habitacion, eliminamos la habitacion  **int** confirmacion = JOptionPane.showConfirmDialog(rootPane, "estas seguro de eliminar la habitacion?", "Confirmar", 2);  **if** (confirmacion == 0) {//si confirmacion es 0 quiere decir que si esta seguro de eliminar, entonces  fHabitacion func = **new** fHabitacion();  vHabitacion dts = **new** vHabitacion();  /\*en la clase molde (dts) se modifica el id\_habitacion por lo que tiene el txtidhabitacion  con el fin de que mas adelante se envie ese mismo dts como parametro al metodo eliminar  \*/  dts.setId\_habitacion(Integer.parseInt(txtidhabitacion.getText()));  func.eliminar(dts);/\*ejecuta el metodo eliminar del controlador y se le lleva el dts (en este  caso el dts solo contendrá el id de la habitacion a eliminar)\*/  mostrar("");//muestran todos los datos  inhabilitar();//inhabilita todos los controles  }  }  }  /\*metodo buscar que ejecuta al metodo de esta clase llamado mostrar y que le  lleva como parametro el criterio de busqueda  \*/  **private** **void** btnbuscarActionPerformed(java.awt.event.ActionEvent evt) {  mostrar(txtbuscar.getText());  }  /\*estos metodos txt y cbo son de los campos y en ellos se esta trasfiriendo el foco  al siguiente control, mediante el metodo transferFocus()  \*/  **private** **void** txtnumeroActionPerformed(java.awt.event.ActionEvent evt) {  txtnumero.transferFocus();  }  **private** **void** cbopisoActionPerformed(java.awt.event.ActionEvent evt) {  cbopiso.transferFocus();  }  **private** **void** txtpreciodiarioActionPerformed(java.awt.event.ActionEvent evt) {  txtpreciodiario.transferFocus();  }  **private** **void** cboestadoActionPerformed(java.awt.event.ActionEvent evt) {  cboestado.transferFocus();  }  **private** **void** cbotipohabitacionActionPerformed(java.awt.event.ActionEvent evt) {  cbotipohabitacion.transferFocus();  }  /\*este metodo se crea dandole clic derecho a la tabla - Events - Mouse -  MouseClicked y para cuando uno le de clic a una fila, entonces haga cualquier cosa.  en este caso: al darle clic a un registro, se entiende que se va a editar o eliminar  \*/  **private** **void** tablalistadoMouseClicked(java.awt.event.MouseEvent evt) {  btnguardar.setText("Editar");//al boton guardar, se le cambia la propiedad text por editar  habilitar(); //se habilitan los controles  btneliminar.setEnabled(**true**); //se habilita el boton eliminar, para que si desea, elimine el registro  accion = "editar";//el atributo accion toma el valor de editar  /\* en fila se almacena de la tabla listado la fila donde se ha hecho clic  \*/  **int** fila = tablalistado.rowAtPoint(evt.getPoint());  /\*ahora capturamos los datos de cada columna y enviarlo a las cajas de texto  \*/  txtidhabitacion.setText(tablalistado.getValueAt(fila, 0).toString());  txtnumero.setText(tablalistado.getValueAt(fila, 1).toString());  cbopiso.setSelectedItem(tablalistado.getValueAt(fila, 2).toString());  txtdescripcion.setText(tablalistado.getValueAt(fila, 3).toString());  txtcaracteristicas.setText(tablalistado.getValueAt(fila, 4).toString());  txtpreciodiario.setText(tablalistado.getValueAt(fila, 5).toString());  cboestado.setSelectedItem(tablalistado.getValueAt(fila, 6).toString());  cbotipohabitacion.setSelectedItem(tablalistado.getValueAt(fila, 7).toString());  }  /\* metodo del boton salir que permite cerrar el programa mediante el metodo  dispose()  \*/  **private** **void** btnsalirActionPerformed(java.awt.event.ActionEvent evt) {  **this**.dispose();  }  **public** **static** **void** main(String args[]) {  java.awt.EventQueue.invokeLater(**new** Runnable() {  **public** **void** run() {  **new** frmhabitacion().setVisible(**true**);  }  });  }  // Variables declaration - do not modify  **private** javax.swing.JButton btnbuscar;  **private** javax.swing.JButton btncancelar;  **private** javax.swing.JButton btneliminar;  **private** javax.swing.JButton btnguardar;  **private** javax.swing.JButton btnnuevo;  **private** javax.swing.JButton btnsalir;  **private** javax.swing.JComboBox cboestado;  **private** javax.swing.JComboBox cbopiso;  **private** javax.swing.JComboBox cbotipohabitacion;  **private** javax.swing.JLabel jLabel1;  **private** javax.swing.JLabel jLabel2;  **private** javax.swing.JLabel jLabel3;  **private** javax.swing.JLabel jLabel4;  **private** javax.swing.JLabel jLabel5;  **private** javax.swing.JLabel jLabel6;  **private** javax.swing.JLabel jLabel7;  **private** javax.swing.JLabel jLabel8;  **private** javax.swing.JLabel jLabel9;  **private** javax.swing.JPanel jPanel1;  **private** javax.swing.JPanel jPanel2;  **private** javax.swing.JScrollPane jScrollPane1;  **private** javax.swing.JScrollPane jScrollPane2;  **private** javax.swing.JScrollPane jScrollPane3;  **private** javax.swing.JLabel lbltotalregistros;  **private** javax.swing.JTable tablalistado;  **private** javax.swing.JTextField txtbuscar;  **private** javax.swing.JTextArea txtcaracteristicas;  **private** javax.swing.JTextArea txtdescripcion;  **private** javax.swing.JTextField txtidhabitacion;  **private** javax.swing.JTextField txtnumero;  **private** javax.swing.JTextField txtpreciodiario;  // End of variables declaration  } |

**Estructura general**

* **Al abrir el formulario:** se encuentra inhabilitado los controles de entrada de datos (excepto el campo buscar), los botones guardar, cancelar, eliminar. se cargarán todos los registros en la tabla y se mostrará el total de registros.
* **Al darle clic en nuevo:** se habilitarán todos los controles inhabilitados al principio y el boton guardar, se llamará “guardar”
* **Al seleccionar un registro de la tabla:** se habilitarán todos los controles inhabilitados desde un principio, y los datos del registro pasrán a los campos de texto y el boton “guardar” cambiará su nombre a editar
* **Al buscar:** se filtrarán los registros de la tabla por aquellos registros que contendrán palabras del criterio de busqueda

#### Formulario MDI (menú) frminicio

Será el formulario contenedor de todos los formularios secundarios o internos

Para crearlo:

* Nos dirigimos al paquete Presentacion – clic derecho – other
* Luego en Categorias vamos a Swing GUI Forms y en file type elegimos MDI Application Sample…

Nota: a los formularios que llamaremos desde nuestro DMI y que esten extendidos de javax.swing.Jframe, lo pondremos a que extiendan de javax.swing.JinternalFrame para que se puedan visualizar.

|  |
| --- |
| **package** Presentacion;  /\*\*\* **@author** Luis Angel\*/  **public** **class** frminicio **extends** javax.swing.JFrame {  **public** frminicio() {  initComponents();  **this**.setExtendedState(frminicio.MAXIMIZED\_BOTH);//para que ocupe toda la pantalla maximize  **this**.setTitle("Sistema de Reserva de Habitaciones y Gestión de ventas - Hotel Paraíso"); //titulo formulario  }  @SuppressWarnings("unchecked")    **private** **void** cutMenuItemActionPerformed(java.awt.event.ActionEvent evt) {  frmHabitacion form =**new** frmHabitacion();//instanciamos el formulario a llamar  /\*el DMI tiene el menú y tiene el Desktop panel, es aquí en el desktop panel donde se abrirán los formularios  que hayamos escogido abrir en nuestro menú. le he cambiado de nombre, ya no es desktop\_panel sino escritorio  \*/  escritorio.add(form);//en el desktop panel de nuestro MDI agregamos el formulario que instanciamos  form.toFront(); //le decimos que aparezca en frente  form.setVisible(**true**);// le decimos que se haga visible  }  //llama al formulario producto  **private** **void** copyMenuItemActionPerformed(java.awt.event.ActionEvent evt) {  frmproducto form =**new** frmproducto();  escritorio.add(form);  form.toFront();  form.setVisible(**true**);  }  //llama al formulario cliente  **private** **void** aboutMenuItemActionPerformed(java.awt.event.ActionEvent evt) {  frmcliente form =**new** frmcliente();  escritorio.add(form);  form.toFront();  form.setVisible(**true**);  }  //llama al formulario trabajador  **private** **void** jMenuItem2ActionPerformed(java.awt.event.ActionEvent evt) {  frmtrabajador form =**new** frmtrabajador();  escritorio.add(form);  form.toFront();  form.setVisible(**true**);  }  **private** **void** mnusalirMouseClicked(java.awt.event.MouseEvent evt) {  //se cerrara cuando se haga clic en el menu salir. por eso está en el evento mouseClicked  **this**.dispose();  }  //llama al formulario reserva y le lleva algunos datos  **private** **void** contentMenuItemActionPerformed(java.awt.event.ActionEvent evt) {  frmreserva form = **new** frmreserva();  escritorio.add(form);  form.toFront();  form.setVisible(**true**);  frmreserva.txtidtrabajador.setText(lblidpersona.getText());  frmreserva.txttrabajador.setText(lblnombre.getText() + " " + lblapaterno.getText());  frmreserva.idusuario=Integer.parseInt(lblidpersona.getText());  }  **public** **static** **void** main(String args[]) {  java.awt.EventQueue.invokeLater(**new** Runnable() {  **public** **void** run() {  **new** frminicio().setVisible(**true**);  }  });  }  // Variables declaration - do not modify  **private** javax.swing.JMenuItem aboutMenuItem;  **private** javax.swing.JMenuItem contentMenuItem;  **private** javax.swing.JMenuItem copyMenuItem;  **private** javax.swing.JMenuItem cutMenuItem;  **public** **static** javax.swing.JDesktopPane escritorio;  **private** javax.swing.JMenuItem jMenuItem1;  **private** javax.swing.JMenuItem jMenuItem2;  **private** javax.swing.JMenuItem jMenuItem3;  **private** javax.swing.JMenuItem jMenuItem4;  **public** **static** javax.swing.JLabel lblacceso;  **public** **static** javax.swing.JLabel lblamaterno;  **public** **static** javax.swing.JLabel lblapaterno;  **public** **static** javax.swing.JLabel lblidpersona;  **public** **static** javax.swing.JLabel lblnombre;  **private** javax.swing.JMenuBar menuBar;  **public** **static** javax.swing.JMenu mnuarchivo;  **private** javax.swing.JMenu mnuayuda;  **public** **static** javax.swing.JMenu mnuconfiguraciones;  **private** javax.swing.JMenu mnuconsultas;  **private** javax.swing.JMenu mnuherramientas;  **private** javax.swing.JMenu mnureservas;  **private** javax.swing.JMenu mnusalir;  **private** javax.swing.JMenu mnusisreserva;  // End of variables declaration  } |

#### Formulario(inicio de sesión) frmusuariologin

|  |
| --- |
| **package** Presentacion;  **import** Datos.vtrabajador;  **import** Logica.ftrabajador;  **import** javax.swing.JOptionPane;  **import** javax.swing.table.DefaultTableModel;  /\*\*\* **@author** Luis Angel\*/  **public** **class** frmusuariologin **extends** javax.swing.JFrame {  **public** frmusuariologin() {  initComponents();  **this**.setTitle("Acceso al Sistema");  **this**.setLocationRelativeTo(**null**);  }    @SuppressWarnings("unchecked")  **private** **void** btningresarActionPerformed(java.awt.event.ActionEvent evt) {  **try** {  DefaultTableModel modelo;  ftrabajador func =**new** ftrabajador();  vtrabajador dts=**new** vtrabajador();  //le enviamos al dts los valores del usuario y del password  dts.setLogin(txtusuario.getText());  dts.setPassword(txtpassword.getText());  modelo=func.login(dts.getLogin(),dts.getPassword());  tablalistado.setModel(modelo);    **if** (func.totalregistros >0) {//si es mayor que 0 quiere decir que el usuario si existe  **this**.dispose();//cierra el formulario frmusuario login  frminicio form = **new** frminicio();//instancia al formulario inicio  form.toFront();//pone en frente el formulario inicio  form.setVisible(**true**);//pone visible el formulario inicio  //enviamos al formulario inicio en cada label, los datos correspondientes que se  //encuentran en la tabla listado  frminicio.lblidpersona.setText(tablalistado.getValueAt(0, 0).toString());  frminicio.lblnombre.setText(tablalistado.getValueAt(0, 1).toString());  frminicio.lblapaterno.setText(tablalistado.getValueAt(0, 2).toString());  frminicio.lblamaterno.setText(tablalistado.getValueAt(0, 3).toString());  frminicio.lblacceso.setText(tablalistado.getValueAt(0, 4).toString());  **if** (!frminicio.lblacceso.getText().equals("Administrador")) {//si el usuario no  //es administrador, entonces  frminicio.mnuarchivo.setEnabled(**false**);//el menu archivo se deshabilita  frminicio.mnuconfiguraciones.setEnabled(**false**);//el menú configuraciones se deshabilita  }  }**else** {  JOptionPane.showMessageDialog(rootPane, "Acceso Denegado","Acceso al Sistema",JOptionPane.ERROR\_MESSAGE);  }  } **catch** (Exception e) {  }  }  **public** **static** **void** main(String args[]) {  java.awt.EventQueue.invokeLater(**new** Runnable() {  **public** **void** run() {  **new** frmusuariologin().setVisible(**true**);  }  });  }  // Variables declaration - do not modify  **private** javax.swing.JButton btningresar;  **private** javax.swing.JButton btnsalir;  **private** javax.swing.JLabel jLabel1;  **private** javax.swing.JLabel jLabel2;  **private** javax.swing.JLabel jLabel3;  **private** javax.swing.JPanel jPanel1;  **private** javax.swing.JScrollPane jScrollPane1;  **private** javax.swing.JTable tablalistado;  **private** javax.swing.JPasswordField txtpassword;  **private** javax.swing.JTextField txtusuario;  // End of variables declaration  } |

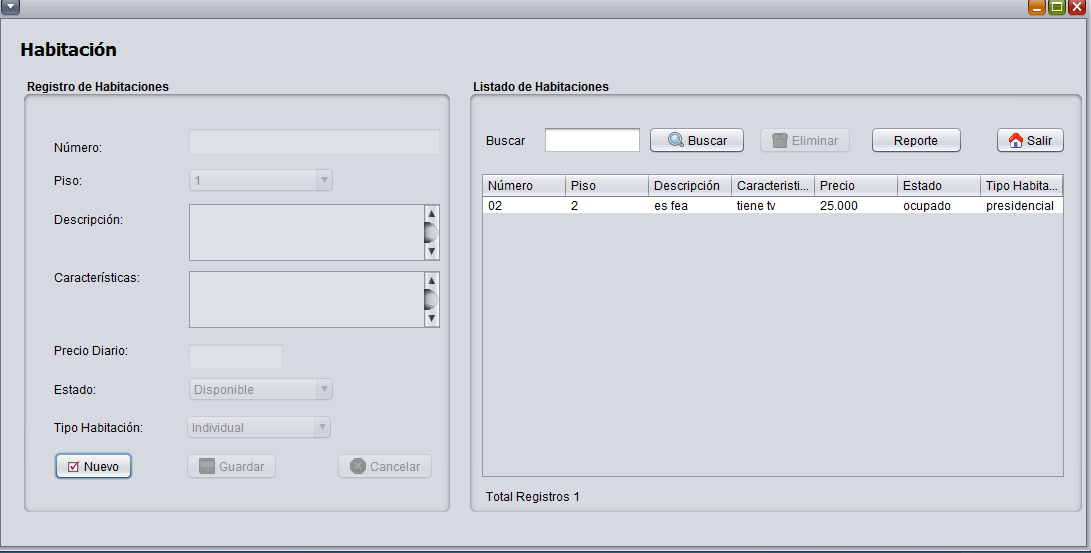
#### Formularios vistas

Los demás formularios vistas cuentan basicamente con la misma estructura del siguiente ejemplo:

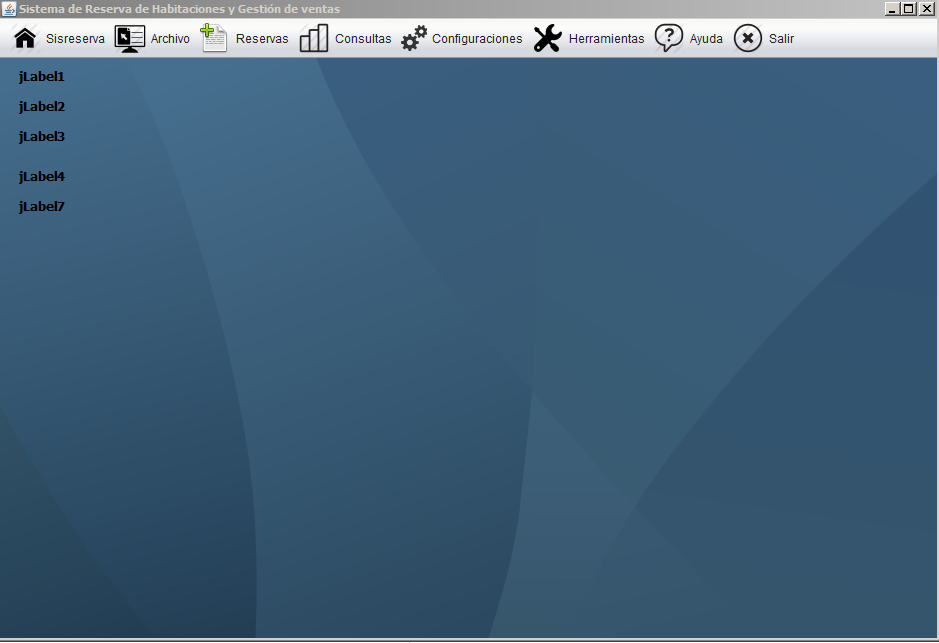
|  |
| --- |
| **package** Presentacion;  **import** Logica.fhabitacion;  **import** javax.swing.JOptionPane;  **import** javax.swing.table.DefaultTableModel;  /\*\*\* **@author** Luis Angel\*/  **public** **class** frmvistahabitacion **extends** javax.swing.JFrame {  **public** frmvistahabitacion() {  initComponents();  mostrar("");  **this**.setLocationRelativeTo(**null**);//hace que el formulario aparezca en la parte central de la pantalla  }  //metodo silimar al de la calse fhabitaciones  **void** ocultar\_columnas() {  tablalistado.getColumnModel().getColumn(0).setMaxWidth(0);  tablalistado.getColumnModel().getColumn(0).setMinWidth(0);  tablalistado.getColumnModel().getColumn(0).setPreferredWidth(0);  }  //metodo similar al de la clase fhabitaciones  **void** mostrar(String buscar) {  **try** {  DefaultTableModel modelo;  fhabitacion func = **new** fhabitacion();  modelo = func.mostrarvista(buscar);  tablalistado.setModel(modelo);  ocultar\_columnas();  lbltotalregistros.setText("Total Registros " + Integer.toString(func.totalregistros));  } **catch** (Exception e) {  JOptionPane.showConfirmDialog(rootPane, e);  }  }  @SuppressWarnings("unchecked")  **private** **void** btnbuscarActionPerformed(java.awt.event.ActionEvent evt) {  mostrar(txtbuscar.getText());  }  **private** **void** tablalistadoMousePressed(java.awt.event.MouseEvent evt) {  **if** (evt.getClickCount() == 2) {//si hago 2 clic, debemos tener en cuenta el evento  **int** fila= tablalistado.getSelectedRow();//selecciona la fila  String cod;  String valor;  //obtiene los valores de cada columna de la fila seleccionada  cod=tablalistado.getValueAt(fila, 0).toString();  valor=tablalistado.getValueAt(fila, 1).toString();  //los valores obtenidos los pasa a dichos campos del formulario reserva  frmreserva.txtidhabitacion.setText(cod);  frmreserva.txtnumero.setText(valor);  **this**.dispose();  }  }  **public** **static** **void** main(String args[]) {  java.awt.EventQueue.invokeLater(**new** Runnable() {  **public** **void** run() {  **new** frmvistahabitacion().setVisible(**true**);  }  });  }  // Variables declaration - do not modify  **private** javax.swing.JButton btnbuscar;  **private** javax.swing.JLabel jLabel9;  **private** javax.swing.JPanel jPanel2;  **private** javax.swing.JScrollPane jScrollPane3;  **private** javax.swing.JLabel lbltotalregistros;  **private** javax.swing.JTable tablalistado;  **private** javax.swing.JTextField txtbuscar;  // End of variables declaration  } |

#### Diseño de los formularios

##### Formularios Principales



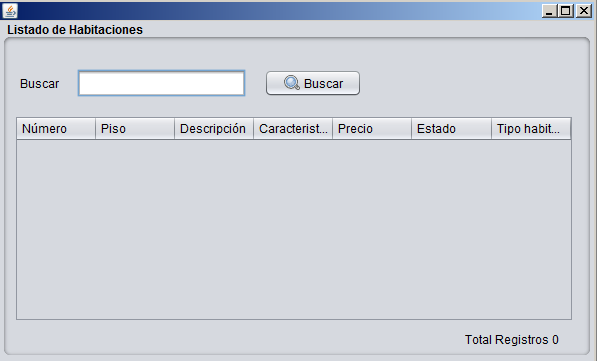
##### Formulario DMI (menú) frminicio



##### Formulario (inicio de sesión) frmusuariologin



##### Formularios vistas



## Agregar un control Jcalendar mediante una librería:

* Vamos al palette – clic derecho – palette manager…
* Abrirá una ventana y le daremos en Add From JAR… y buscamos la librería Jcalendar y luego seleccionamos Jcalendar y Jdata Chooser
* Luego seleccionamos donde queremos que aparezca, en este caso en la carpeta Swing Controls y le damos finish

Invocar Controles de un formulario desde otro formulario diferente:

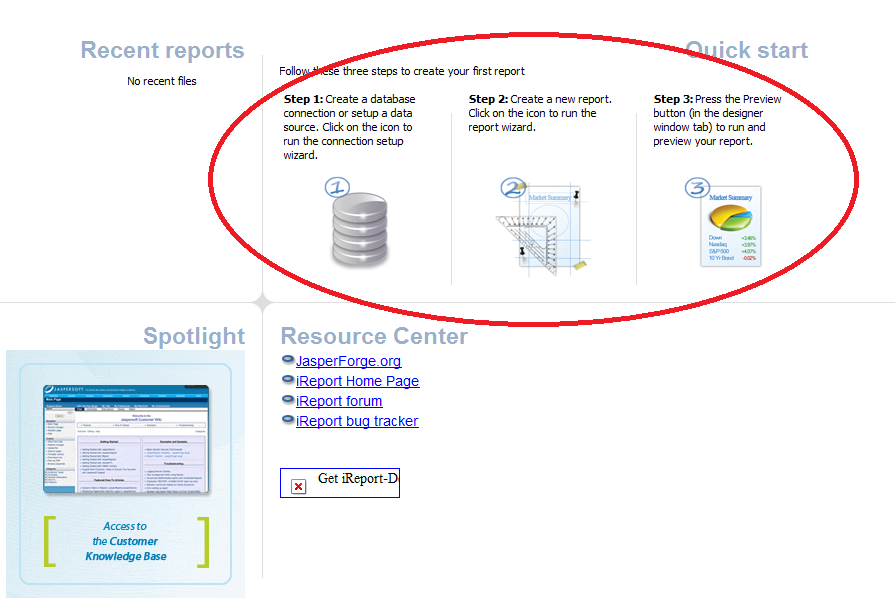
para inhabilitar o habilitar algún menú de nuestra barra de menú, tenemos que ir a las propiedades del menú – Code – Variable Modifiers – access – seleccionamos public y seleccionamos static.

# CREAR REPORTES EN JAVA CON JASPER REPORTS Y IREPORT DESIGNER

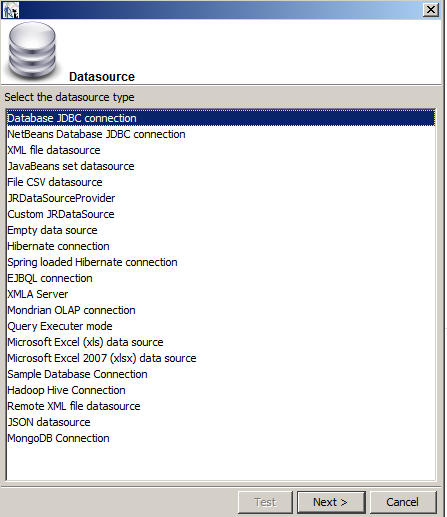
Diseñador de reportes con ireport 5.6.0

* Vamos a ireport y abrimos la carpeta bin – ireport (el archivo aplicacion)
* Nos da la opcion de crear el reporte mediante 3pasos, Los seguimos paso a paso.

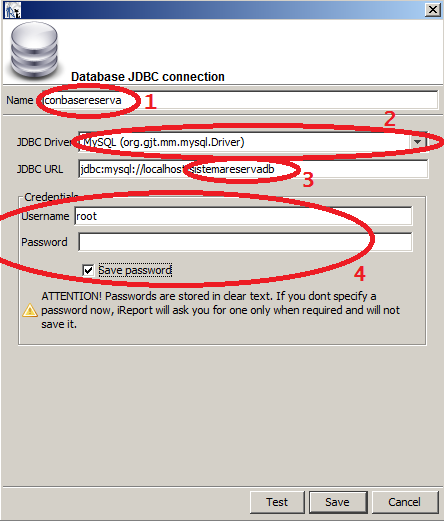
1. Paso 1: antes de dirigirnos a este paso, activamos el servidor y luego damos clic en:



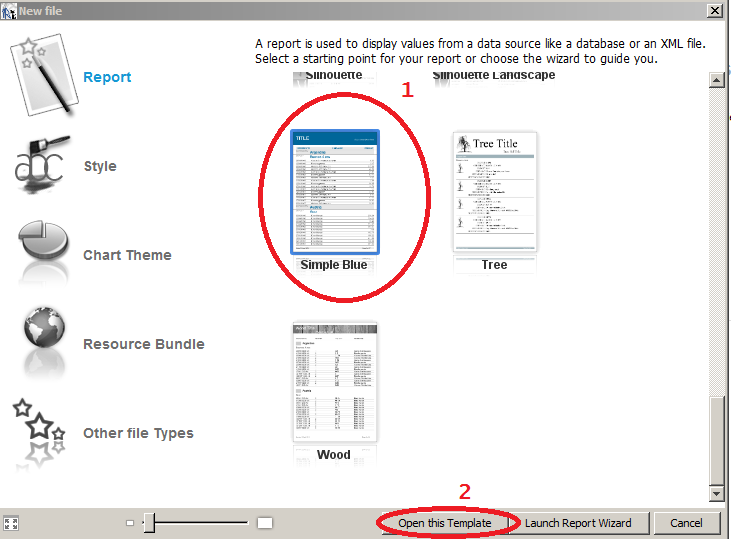
* 1. damos clic en el step1 y se nos abrirá una ventana en la cual seleccionaremos el tipo de conexión a la base de datos



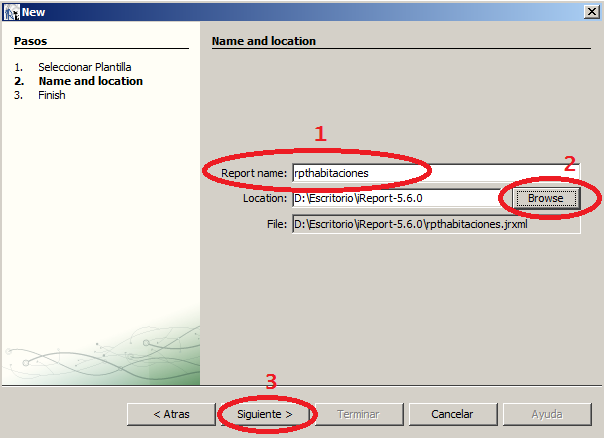
* 1. en la nueva ventana asignamos un nombre a la conexión, seleccionamos nuestro driver jdbc que es el de mysql (con el que estamos trabajando), en JDBC URL ponemos la URL de nuestra clase conexión de la aplicación y en credenciales. Ponemos el username y password que se encuentra en nuestra clase conexión



1. paso 2: hacemos clic al step 2
   1. seleccionamos una plantilla y le damos en open



* 1. en la nueva ventana, le damos un nombre al reporte y luego buscamos la locacización donde se almacenará el reporte hecho (esta localizacion debe ser en la carpeta src de nuestra aplicación, ahí creamos una carpeta que se llame reportes y ahí almacenamos nuestros reportes)



* 1. luego nos debe aparecer Congratulations donde indica que el reporte fue creado exitosamente

1. step 3: se nos abre el asistente de diseño del reporte, donde crearemos el reporte a través de dichos controles

## Desarrollar reportes en iReport

## Implementar el reporte en netbeans

una vez creado el reporte en ireport designer. Para implementarlo en java a través de netbeans, importamos las librerias de la API Jasper Reports que nos van a permitir visualizar el reporte.

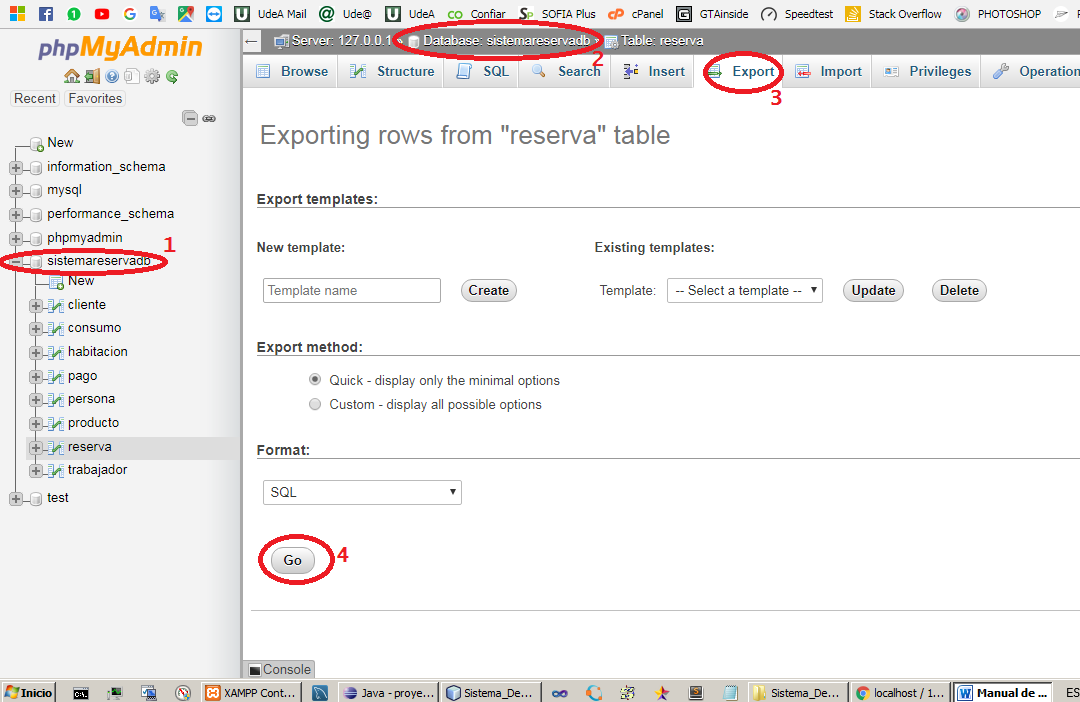
Para darle funcion a nuestro reporte desde nuestra aplicación, a través de un boton. Utilizamos el codigo:

|  |
| --- |
| /\*instanciamos a la clase conexion con el nuevo atributo que llamaremos conexion  \*/  **private** Connection conexion=**new** conexion().conectar();  /\*metodo que da accion al boton de generar reporte  \*/  **private** **void** btnReporteActionPerformed(java.awt.event.ActionEvent evt) {  Map p=**new** HashMap();//  JasperReport report;//variable para visulaizar reporte  JasperPrint print;//variable para imprimir reporte  **try** {  //asignamos el reporte a nuestra variable report  report=JasperCompileManager.compileReport(**new** File("").getAbsolutePath()+"/src/Reportes/rpHabitaciones.jrxml");  //imprimimos nuestro reporte  print=JasperFillManager.fillReport(report, p,conexion);  //visualizamos nuestro reporte  JasperViewer view=**new** JasperViewer(print,**false**);  //le ponemos un titulo a nuestro reporte  view.setTitle("Reporte de Habitaciones");  //le decimos que esté visible  view.setVisible(**true**);  } **catch** (Exception e) {  e.printStackTrace();  }  } |

# GENERAR ARCHIVOS DE INSTALACION DEL SISTEMA

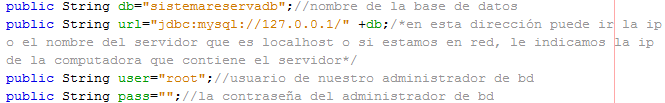
## Exportar nuestra base de datos:

vamos al phpmyadmin y nos paramos en nuestra base de datos y le damos en exportar y luego en go.



## Configuracion de la aplicación antes de generar el .jar

1. Vamos a la clase conexión que se encuentra en nuestro paquete logica o controlador y modificamos los siguientes valores



En db ponemos el nombre de la base de datos que hemos creado en el servidor

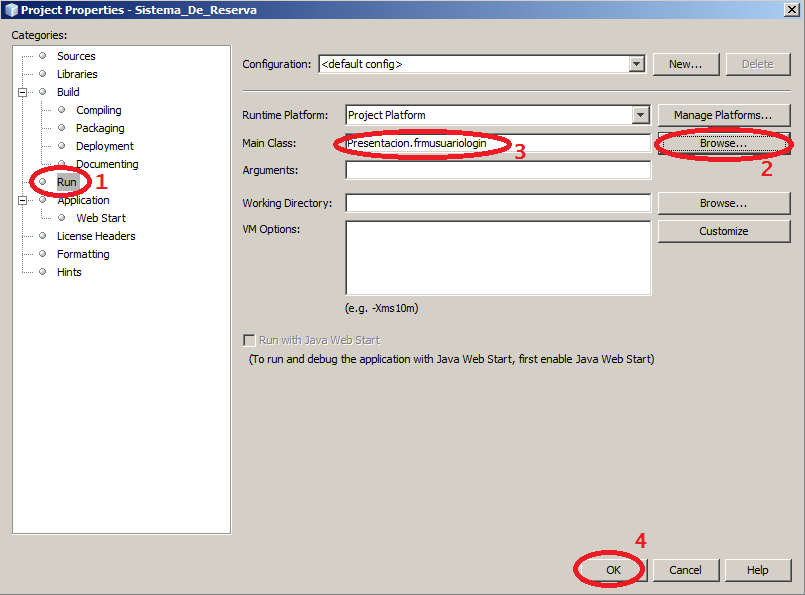
En url donde está la ip 127.0.0.1 ponemos

* La ip de la maquina si la aplicación quedará en red
* “Localhost” o la ip de la maquina si la aplicación se utilizará solamente a nivel de la misma maquina que contiene el servidor

En user ponemos nuestro usuario de administrador de la base de datos

En pass ponemos la contraseña del usuario administrador de la base de datos

1. Vamos a nuestra aplicación en netbeans y le damos clic derecho a nuestro proyecto - Propiedades y luego le damos en run y seleccionamos el primer formulario que queremos que abra la aplicación



## Generar el .jar

1. En netbeans, seleccionamos nuestro proyecto, le damos clic derecho – Clean and Build y esperamos…
2. Luego vamos a la carpeta donde tenemos el proyecto y vemos que se nos crea una carpeta llamada dist y ahí se encuentra nuestro .jar
3. Despues copiamos nuestra carpeta src y la pegamos en la carpeta dist y solo dejamos la carpeta reporte y las demás las eliminamos.

# IMPLEMENTACION DEL SISTEMA DE RESERVA EN UNA COMPUTADORA O RED

Requisitos:

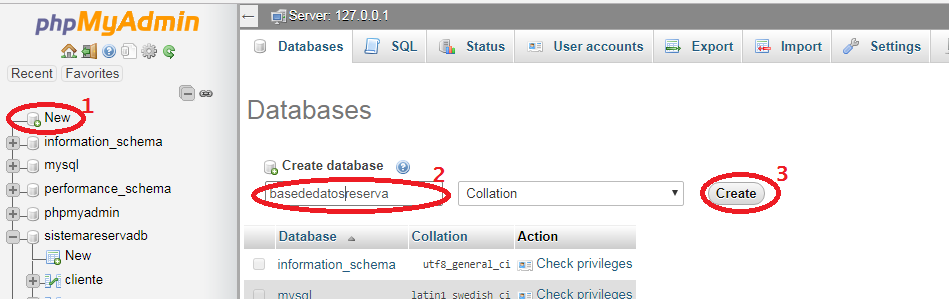
* Servidor MySQL :

Puede ser WAMP, LAMP, MAMP O XAMPP

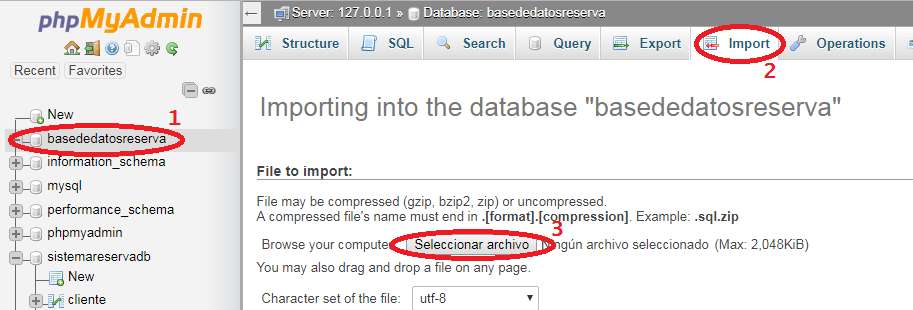
* JRE
* Archivo Jar y Librerías

## Configurando la maquina servidor

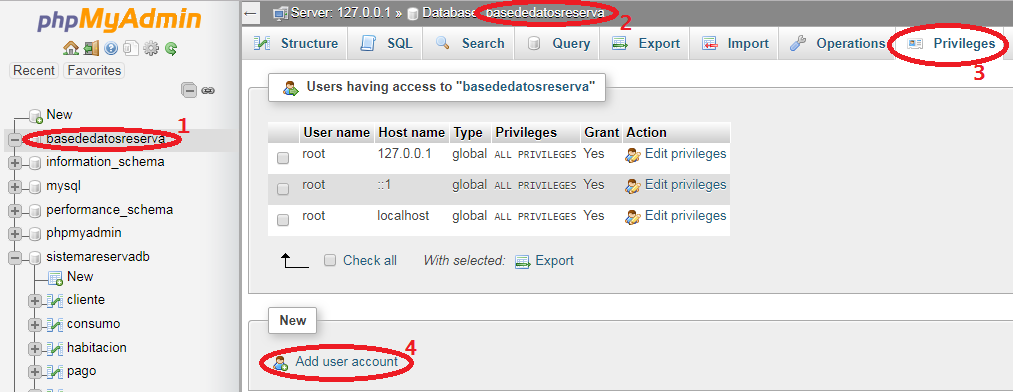
1. En la maquina servidor instalamos el servidor que ejecutará la maquina ejemplo: XAMPP
2. Luego nos dirigimos al phpMyAdmin <http://localhost/phpmyadmin/> y creamos una base de datos:



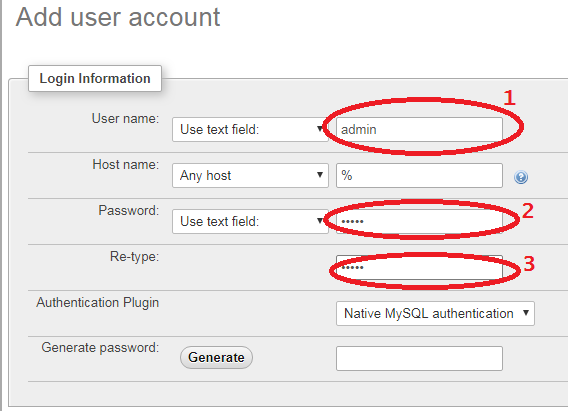
1. Seleccionamos la base de datos creada anteriormente y luego damos en import y despues seleccionamos el archivo de base de datos



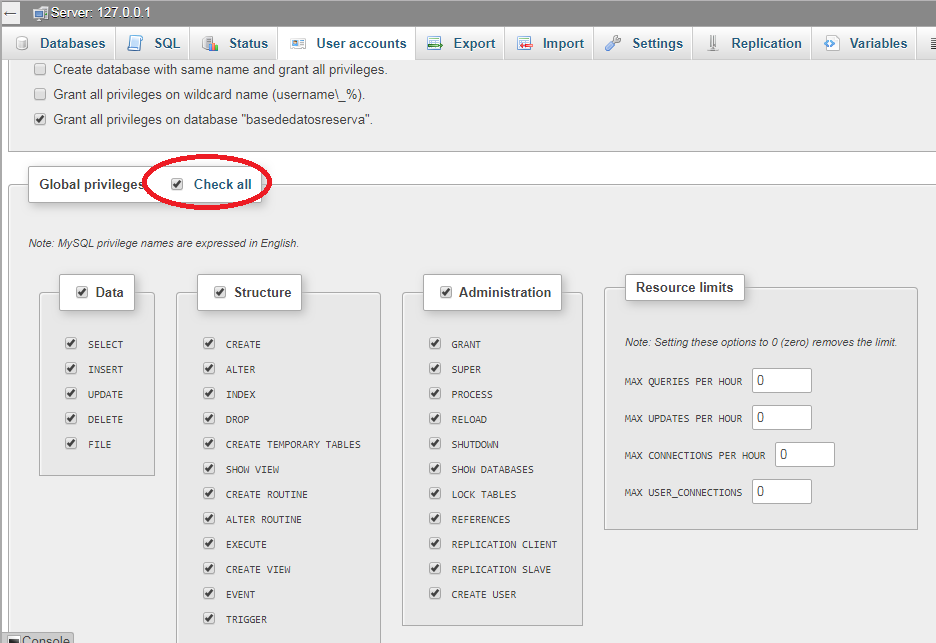
1. Seleccionamos nuestra base de datos creada y vamos donde dice privilegios y agregamos un usuario para nuestra base de datos



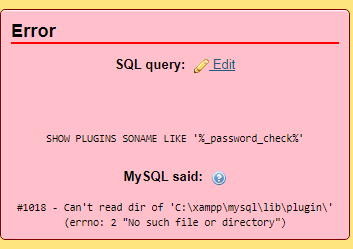
1. Asignamos un nombre de usuario, una contraseña y confirmamos la contraseña



1. Luego bajamos un poco y le asignamos todos los privilegios que deseamos y le damos en go



Nota: en caso de este error



Vamos a la direccion c:\xampp\mysql\ y ahí dentro creamos la carpeta “lib” y dentro de ella creamos la carpeta “plugin”