



Night Of Silence

Documentación.

Índice de contenido

Introducción:.....	3
Night Of Silence. ¿Que se puede hacer?.....	3
Diseño de N.O.S.....	3
Diseño de la BD.....	4
SERVLETS.....	5
BD_NOS.java.....	6
NOS.java.....	16
GUERRA.java.....	21
JSPs.....	36
index.jsp.....	37
login.jsp.....	40

ver_jugadores.jsp.....	42
registro_usuario.jsp.....	44
ficha_pj.jsp.....	47
menu.jsp.....	50
menu_juego.jsp.....	53
join_battle.jsp y entrada_mapa.jsp.....	54
join_battle.jsp.....	55
entrada_mapa.jsp.....	57
PARTIDA EN JUEGO.....	59
PARTIDA EN ESPERA DE TURNO.....	60
Guión de Instalación de la Plataforma:.....	61
Despliegue en Tomcat5.5:.....	62
Conclusión:.....	62
Bibliografía:.....	63

Introducción:

Night Of Silence nace de la nada intentando seguir el éxito de juegos como Ogame o TribalWars(Juegos Web). Para llegar al nivel de diseño de ambos juegos, se necesitan más de 3 y 4 meses e incluso años de duro trabajo picando código para crear algo comerciable.

En esta versión de NOS (v0.03) he intentado aplicar lo que sería un nivel base para la aplicación, con Servlets y JSP; altas y bajas de usuarios, control de sesiones, inserción aleatoria en un punto dentro de un mapa, ataques simples entre usuarios, gestión de turnos...

En este documento se refleja el trabajo que llevo haciendo desde hace 2 meses y medio, y que parece que es nada para todo lo que queda por desarrollar aún, con continuidad en un futuro próximo.

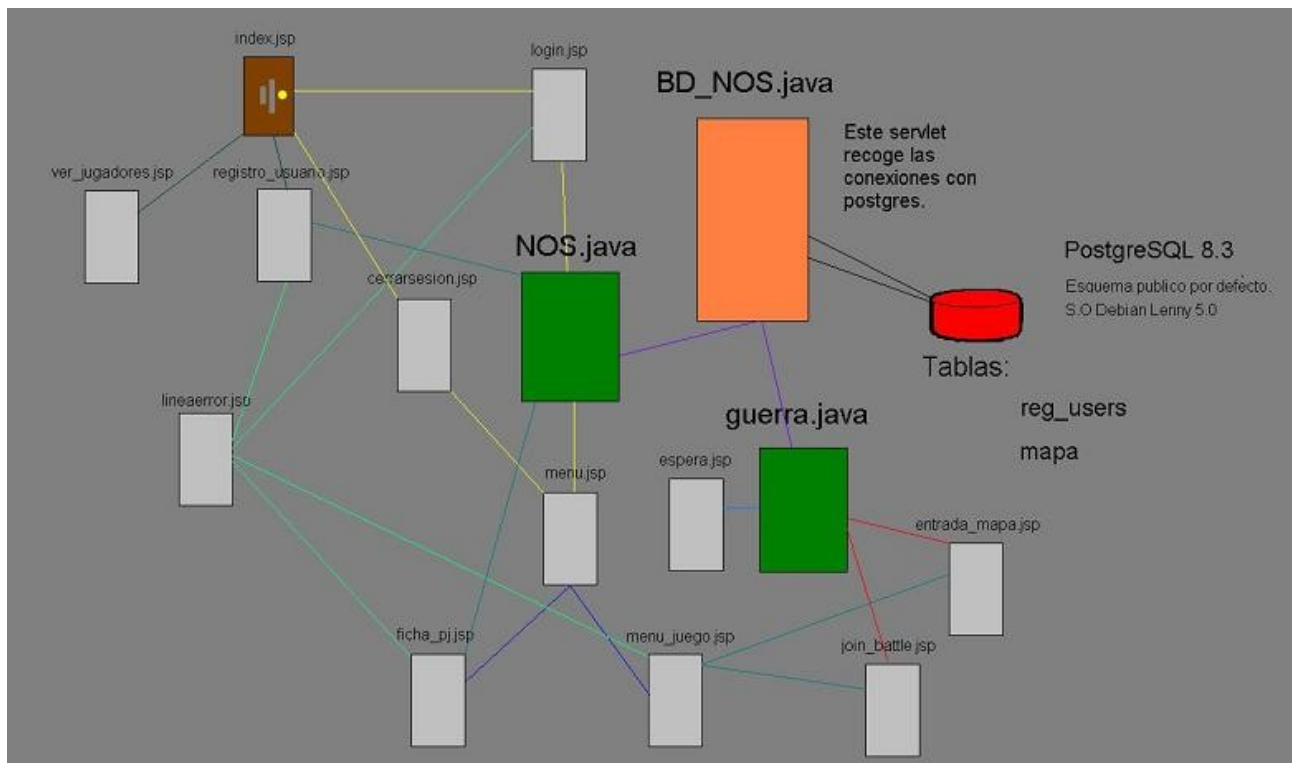
Mis fallos como programador son de noob pero ya serán cada vez menos...

Night Of Silence. ¿Que se puede hacer?

En esta versión se pueden introducir en una partida hasta un límite de 9 personajes (no testado), de los cuales, el primero que llegue será el primero en atacar. A medida que vayan entrando usuarios se sumaran a la partida y se neutralizarán sus acciones hasta que sea su turno por orden de entrada. Una vez que el primero lance el primer ataque, pasará turno al 2º y así se irán sucediendo los ataques. Además se irá viendo el estado de salud de cada jugador cuando sea nuestro turno. El último superviviente se llevará la gloria.

La salud de los personajes estará limitada a 7 y será un todos contra todos.

Diseño de N.O.S.



Diseño de la BD.

```
-- Table: reg_users

CREATE TABLE reg_users
(
  "Nombre" character varying(50),
  "Apodo" character varying(20) NOT NULL,
  "Email" character varying(100) NOT NULL,
  "Fecha_nacimiento" date,
  "Password" character varying(10),
  "Raza" character varying(30),
  "Clan" character varying(50),
  "Salud" integer,
  "Fuerza" integer,
  "Inteligencia" integer,
  "Destreza" integer,
  "Experiencia" double precision DEFAULT 0,
  "Dinero" double precision DEFAULT 0,
  "Moral" double precision,
  CONSTRAINT reg_users_pkey PRIMARY KEY ("Apodo", "Email")
)
WITH (OIDS=FALSE);
ALTER TABLE reg_users OWNER TO postgres;

-- Table: mapa

CREATE TABLE mapa
(
  "X" integer,
  "Y" integer,
  "Estado" character varying(1),
  "Apodo" character varying(30),
  "Nombre_partida" character varying(50),
  "Password_partida" character varying(10),
  "Tipo_jugador" character varying(10)
)
WITH (OIDS=FALSE);
ALTER TABLE mapa OWNER TO postgres;
```

Estas 2 tablas recogen toda la información que necesitamos para la aplicación.

Fallos contemplados:

- Guardar la password en texto plano no es para nada seguro. Habría que realizar un metodo que reciba la pass y la devuelva encriptada, por ejemplo en MD5.
- La tabla mapa no está en 3ª forma normal.

SERVLETS

BD_NOS.java

```
package pack1;

import java.sql.*;
import java.util.Date;

public class BD_NOS {

    private Connection con; //Objeto Connection.
    private Statement sentenciaSQL; //Objeto Statement.
    String motor = "";
    String controlador = "";
    String bd = "";
    String usuario = "";
    String contraseña = "";
    //PARAMETROS POR DEFECTO CONEXION MySQL (local) (Admin=JuanL)(NO modificar)
    //String motor = "MySQL";
    //String controlador = "com.mysql.jdbc.Driver";
    //String bd = "jdbc:mysql://localhost/nos";
    //String usuario = "root";
    //String contraseña = "root";
    //PARAMETROS POR DEFECTO CONEXION PostgreSQL (remota) (Admin=RaulR) (NO modificar)
    //String motor = "MySQL";
    //String controlador = "org.postgresql.Driver";
    //String bd = "jdbc:postgresql://laboratorioderhau.no-ip.org:5432/postgres";
    //String usuario = "postgres";
    //String contraseña = "postgres";
    //PARAMETROS POR DEFECTO CONEXION PostgreSQL (local) (Admin=RaulL) (NO modificar)
    //String motor = "MySQL";
    //String controlador = "org.postgresql.Driver";
    //String bd = "jdbc:postgresql://localhost:5432/postgres";
    //String usuario = "postgres";
    //String contraseña = "postgres";
    Date fechanac = null;
    String cadSQL = "";
    ResultSet resul = null;
    ResultSet resulQuery = null;
    Integer resulUpdate = 0;

    //Parametro "admin" inicial para definir la conexión con la BD.
    public BD_NOS(String admin0)
        throws ClassNotFoundException, SQLException, InstantiationException, IllegalAccessException {
        if (admin0.equals("JuanL")) {
            controlador = "com.mysql.jdbc.Driver";
            motor = "MySQL";
            bd = "jdbc:mysql://localhost:3306/nos";
            usuario = "root";
            contraseña = "root";
        } else if (admin0.equals("RaulL")) {
            controlador = "org.postgresql.Driver";
            bd = "jdbc:postgresql://localhost:5432/postgres";
            usuario = "postgres";
            contraseña = "postgres";
            motor = "PostgreSQL";
        } else if (admin0.equals("RaulR")) {
```

```

        controlador = "org.postgresql.Driver";
        bd = "jdbc:postgresql://laboratorioderhau.no-ip.org:5432/postgres";
        usuario = "postgres";
        contraseña = "postgres";
        motor = "PostgreSQL";
    }
    /* Cargar el driver según motor. Este forma es bastante rara
    pero util si se desea trabajar con distintas BD.*/
    Class.forName(controlador);
    conectar(); // conectar con la fuente de datos
}
// -----
// Conectar con la base de datos.
// -----

public void conectar() throws SQLException {
    // Conectar con la BD
    try {
        con = DriverManager.getConnection(bd, usuario, contraseña);
        // Crear objeto sentencia SQL
        sentenciaSQL = con.createStatement();
    } catch (SQLException e) {
        System.out.println("\nLa conexión no ha podido realizarse.\n" + e.toString());
    }
}
// -----
// Insertar nuevo usuario y nuevo apodo. Función:
/*Inserta en reg_users los datos introducidos, controlado previamente por
javascript.*/
// -----

public ResultSet insertar_email_password(String email, String password, String nombre, String apodo,
    String clan, String raza, String fecnac, Integer salud, Integer fuerza, Integer experiencia,
    Integer dinero, Integer moral, Integer inteligencia, Integer destreza) throws SQLException {
    resul = null;
    String dia = fecnac.substring(0, 2);
    String mes = fecnac.substring(3, 5);
    String año = fecnac.substring(6, 10);
    String fecnac0 = año + "/" + mes + "/" + dia;
    try {
        sentenciaSQL = con.createStatement();
        if (motor.equals("PostgreSQL")) {
            cadSQL = "INSERT INTO
reg_users(\"Nombre\", \"Apodo\", \"Email\", \"Fecha_nacimiento\", \"Password\", \"Clan\", \"Raza\", \"Salud\", \"Fuerza\",
\"Experiencia\", \"Dinero\", \"Moral\", \"Inteligencia\", \"Destreza\") "
                + "VALUES ('" + nombre + "','" + apodo + "','" + email + "','" + fecnac0 + "','" + password + "','" + clan
+ "','" + raza + "','" + salud + "','" + fuerza + "','" + experiencia + "','" + dinero + "','" + moral + "','" + inteligencia + "','"
+ destreza + "')";
        } else {
            cadSQL = "INSERT INTO reg_users ( Email,Password, Nombre, Apodo, Clan, Raza, F_nac,"
                + "Salud, Fuerza, Experiencia, Dinero, Moral, Inteligencia, Destreza)"
                + " VALUES ('" + email + "','" + password + "','" + nombre + "','" + apodo + "','" + clan + "','" + raza +
                + "','" + fecnac0 + "','"
                + salud + "','" + fuerza + "','" + experiencia + "','" + dinero + "','" + moral + "','" + inteligencia + "','" + destreza
+ "')";
        }
        sentenciaSQL.executeUpdate(cadSQL);
    } catch (SQLException e) {
        System.out.println("\nLa inserción del personaje no ha podido realizarse.\n" + e.toString());
        return resul;
    }
}

```

```

    }
    return resul;
}
// -----
// Comprobar Email Password. Función:
/*Extrae toda la informacion del email y pass asociado al registro en la tabla.*/
// -----

public ResultSet comprobar_email_password(String email, String password) throws SQLException {
    resulQuery = null;
    try {
        sentenciaSQL = con.createStatement();
        if (motor.equals("PostgreSQL")) {
            cadSQL = "SELECT \"Apodo\", \"Nombre\", \"Email\", \"Password\", \"Salud\", \"Fuerza\", \"Experiencia\",
\"Dinero\", \"
                + \" \"Moral\", \"Inteligencia\", \"Destreza\" from reg_users where \"Email\"=\"" + email + "\" and
\"Password\"=\"" + password + "\"";
        } else {
            cadSQL = "SELECT Email, Password, Apodo, Nombre, Salud, Fuerza, Experiencia, Dinero, \"
                + \" Moral, Inteligencia, Destreza \"
                + \" from reg_users where Email=\"" + email + "\" and Password=\"" + password + "\"";
        }
        resulQuery = sentenciaSQL.executeQuery(cadSQL);
    } catch (SQLException e) {
        System.out.println("\nLa comprobación del usuario no ha podido realizarse.\n" + e.toString());
        return resulQuery;
    }
    return resulQuery;
}
// -----
// Comprobar Email. Funcion:
/*Extrae la informacion que necesitamos sobre el email recibido.
Podemos comprobar que ya hay un usuario registrado con ese email.*/
// -----

public ResultSet comprobar_email(String email) throws SQLException {
    resulQuery = null;
    try {
        sentenciaSQL = con.createStatement();
        if (motor.equals("PostgreSQL")) {
            cadSQL = "SELECT \"Apodo\", \"Nombre\", \"Email\" from reg_users where \"Email\"=\"" + email + "\"";
        }
        resulQuery = sentenciaSQL.executeQuery(cadSQL);
    } catch (SQLException e) {
        System.out.println("\nLa comprobación del usuario no ha podido realizarse.\n" + e.toString());
        return resulQuery;
    }
    return resulQuery;
}
// -----
// Comprobar Apodo. Función:
/*No se pueden repetir los apodos.
Email y Apodo son Primarykey pero el Apodo debe poder repetirse ya que en el registro inicial
este se establece como 0. Por lo que surge la necesidad de que la aplicación lo controle.*/
// -----

public ResultSet comprobar_apodo(String apodo) throws SQLException {
    resulQuery = null;
    try {

```



```

        sentenciaSQL = con.createStatement();
        if (motor.equals("PostgreSQL")) {
            cadSQL = "SELECT \"Apodo\" from reg_users where \"Apodo\"=\"" + apodo + "\"";
        }
        resulQuery = sentenciaSQL.executeQuery(cadSQL);
    } catch (SQLException e) {
        System.out.println("\nLa comprobación del usuario no ha podido realizarse.\n" + e.toString());
        return resulQuery;
    }
    return resulQuery;
}

//-----
// Modificar Personaje. Funcion:
/*Una vez registrado el usuario tiene la opcion de crear personaje.
Actualiza con los parametros recibidos apodo, clan y raza en reg_user
donde el email y pass tb son recibidos.*/
//-----
public Integer modificar_personaje(String email, String password,
    String apodo, String clan, String raza) throws SQLException {
    resulUpdate = 0;
    try {
        sentenciaSQL = con.createStatement();
        if (motor.equals("PostgreSQL")) {
            cadSQL = "UPDATE reg_users SET \"Apodo\"=\"" + apodo
                + "\", \"Clan\"=\"" + clan + "\", \"Raza\"=\"" + raza + "\" where \"Email\"=\"" + email + "\" and \"Password\"=\"" +
password + "\"";
        } else {
            cadSQL = "UPDATE reg_users SET Apodo=\"" + apodo
                + "\", Clan=\"" + clan + "\", Raza=\"" + raza + "\" where Email=\"" + email + "\" and Password=\"" + password +
""";
        }
        resulUpdate = sentenciaSQL.executeUpdate(cadSQL);
    } catch (SQLException e) {
        System.out.println("\nLa actualizacion del personaje no ha podido realizarse.\n" + e.toString());
        return resulUpdate;
    }
    return resulUpdate;
}

//-----
//Comprobar Mapa. Funcion:
/*En un principio la idea es que sea un MASTER el que lleve nuestra partida*/
//-----
public ResultSet comprobar_mapa() throws SQLException {
    resulQuery = null;
    try {
        sentenciaSQL = con.createStatement();
        if (motor.equals("PostgreSQL")) {
            cadSQL = "Select \"Apodo\" from mapa where \"Tipo_jugador\"='Master'";
        }
        resulQuery = sentenciaSQL.executeQuery(cadSQL);
    } catch (SQLException e) {
        System.out.println("\nLa actualizacion del personaje no ha podido realizarse.\n" + e.toString());
        return resulQuery;
    }
    return resulQuery;
}

```

```

//-----
//Crear Mapa. Funcion:
/*Limpia la tabla con los valores que debe tener por defecto.
 * X e Y se reciben como parametros lo que facilitará que
 se introduzcan los datos ordenadamente y así poder mostrarlos
 luego con facilidad ordenados por X e Y.*/
//-----

public Integer crear_mapa(int i, int j) throws SQLException {
    resulUpdate = 0;
    try {
        sentenciaSQL = con.createStatement();
        if (motor.equals("PostgreSQL")) {
            cadSQL = "update mapa set \"X\"=\"" + i + "\",\"Y\"=\"" + j + "\",\"Estado\"='L', \"Apodo\"='L',
\\Nombre_partida\"='0', \"Password_partida\"='', \"P_accion\"=0, \"Tipo_jugador\"='\" where \"X\"=\"" + i + "\" and
\\Y\"=\"" + j + "\"";
        }
        resulUpdate = sentenciaSQL.executeUpdate(cadSQL);
    } catch (SQLException e) {
        System.out.println("\nLa actualizacion del personaje no ha podido realizarse.\n" + e.toString());
        return resulUpdate;
    }
    return resulUpdate;
}
//-----
//Mostrar Mapa. Función:
/*Esta sentencia se utiliza para extraer la información del mapa ordenadamente,
 así la podemos distribuir en un mapa de coordenadas.
 */
//-----

public ResultSet mostrar_mapa() throws SQLException {
    resulQuery = null;
    try {
        sentenciaSQL = con.createStatement();
        if (motor.equals("PostgreSQL")) {
            cadSQL = "Select \"X\", \"Y\", \"Estado\", \"Apodo\" from mapa where \"Tipo_jugador\"!='Master' order by
\\X\", \"Y\" ";
        }
        resulQuery = sentenciaSQL.executeQuery(cadSQL);
    } catch (SQLException e) {
        System.out.println("\nLa actualizacion del personaje no ha podido realizarse.\n" + e.toString());
        return resulQuery;
    }
    return resulQuery;
}
//-----
//Mostrar PJs. Funcion:
/*Muestra solo los jugadores que no son master.
 Recibe el master de la partida y muestra los que no son "Master".
 Como el permiso de accion deberia otorgarlo el Master y en esta versión
 no le hechamos cuenta, el parametro acción está inhabilitado.*/
//-----

public ResultSet mostrar_pjs(String apodo, Integer accion) throws SQLException {
    resulQuery = null;
    try {
        sentenciaSQL = con.createStatement();
        if (motor.equals("PostgreSQL")) {

```

```

        cadSQL = "Select \"X\", \"Y\", \"Estado\", \"Apodo\" from mapa where \"Apodo\"!=\"\" + apodo + \"\" and
        \"P_accion\"!=\"\" + accion + \"\" and \"Estado\"!=\"L\" and \"Tipo_jugador\"!=\"Master\" order by \"X\", \"Y\" ";
    }
    resulQuery = sentenciaSQL.executeQuery(cadSQL);
} catch (SQLException e) {
    System.out.println("\nLa actualizacion del personaje no ha podido realizarse.\n" + e.toString());
    return resulQuery;
}
return resulQuery;
}
//-----
//Introducir Master. Funcion:
/*Introduce al Master en el mapa. Lo mantenemos al margen dejandolo
fuera del mapa en la posicion ( 0 - 0 ).*/
//-----

public Integer introducir_master(String apodo, String nom_partida, String pass_partida) throws SQLException {
    resulUpdate = 0;
    try {
        sentenciaSQL = con.createStatement();
        if (motor.equals("PostgreSQL")) {

            cadSQL = "UPDATE mapa set \"X\"=0, \"Y\"=0, \"Apodo\"=\"\" + apodo + \"\", \"Estado\"='O', \"
            + \"Nombre_partida\"=\"\" + nom_partida + \"\", \"Password_partida\"=\"\" + pass_partida + \"\", \"
            + \"Tipo_jugador\"='Master' where \"X\"=0 and \"Y\"=0";

        }
        resulUpdate = sentenciaSQL.executeUpdate(cadSQL);
    } catch (SQLException e) {
        System.out.println("\nLa actualizacion del personaje no ha podido realizarse.\n" + e.toString());
        return resulUpdate;
    }
    return resulUpdate;
}
//-----
//Sacar Master. Funcion:
/*Saca al master del mapa. Esto posibilita a otro jugador ser el Master.*/
//-----

public Integer sacar_master() throws SQLException {
    resulUpdate = 0;
    try {
        sentenciaSQL = con.createStatement();
        if (motor.equals("PostgreSQL")) {

            cadSQL = "UPDATE mapa set \"X\"=0, \"Y\"=0, \"Apodo\"='L', \"Estado\"='L', \"
            + \"Nombre_partida\"=\"\", \"Password_partida\"=\"\", \"Tipo_jugador\"=\" \"
            + \"where \"X\"=0 and \"Y\"=0";

        }
        resulUpdate = sentenciaSQL.executeUpdate(cadSQL);
    } catch (SQLException e) {
        System.out.println("\nLa actualizacion del personaje no ha podido realizarse.\n" + e.toString());
        return resulUpdate;
    }
    return resulUpdate;
}
//-----
//Unir Jugador. Funcion:
/*Une a un jugador en un mapa creado por un master.*/
//-----

```

```

public Integer unir_jugador(String apodo, Integer x, Integer y) throws SQLException {
    resulUpdate = 0;
    try {
        sentenciaSQL = con.createStatement();
        if (motor.equals("PostgreSQL")) {
            cadSQL = "update mapa set \"Apodo\"=\"" + apodo + "\", \"Estado\"='O', "
                + "\"Tipo_jugador\"='PJ' where \"X\"=\"" + x + "\" and \"Y\"=\"" + y + "\"";
        }
        resulUpdate = sentenciaSQL.executeUpdate(cadSQL);
    } catch (SQLException e) {
        System.out.println("\nLa actualizacion del personaje no ha podido realizarse.\n" + e.toString());
        return resulUpdate;
    }
    return resulUpdate;
}
//-----
//Desconectar Jugador. Función:
/*Saca a un jugador de un mapa*/
//-----

public Integer desconectar_jugador(String apodo) throws SQLException {
    resulUpdate = 0;
    try {
        sentenciaSQL = con.createStatement();
        if (motor.equals("PostgreSQL")) {
            cadSQL = "update mapa set \"Apodo\"='L', \"Estado\"='L', \"Tipo_jugador\"=", "
                + "\"P_accion\"=0 where \"Apodo\"=\"" + apodo + "\"";
        }
        resulUpdate = sentenciaSQL.executeUpdate(cadSQL);
    } catch (SQLException e) {
        System.out.println("\nLa actualizacion del personaje no ha podido realizarse.\n" + e.toString());
        return resulUpdate;
    }
    return resulUpdate;
}
//-----
//Comprobar Partida. Función:
/*Comprueba que existe la partida creada en el mapa.
La idea del nombre y la pass surge de la posibilidad de que varios jugadores estén
en el mismo mapa pero en diferentes partidas. Ademas esto se resume en la BD a 2 campos añadidos
a la tabla mapa con el pensamiento de que el Master formara parte de la partida y pudiera estar
en cualquier punto.*
//-----

public ResultSet comprobar_partida(String nom_partida, String pass_partida) throws SQLException {
    resulQuery = null;
    try {
        sentenciaSQL = con.createStatement();
        if (motor.equals("PostgreSQL")) {
            cadSQL = "Select \"Apodo\", \"Nombre_partida\", \"Password_partida\" from mapa where
\"Nombre_partida\"=\"" + nom_partida + "\" and \"Password_partida\"=\"" + pass_partida + "\"";
        }
        resulQuery = sentenciaSQL.executeQuery(cadSQL);
    } catch (SQLException e) {
        System.out.println("\nLa actualizacion del personaje no ha podido realizarse.\n" + e.toString());
        return resulQuery;
    }
    return resulQuery;
}
}

```

```
//-----  
//Comprobar Usuarios Repetidos. Función:  
/*Comprueba usuarios repetidos en el mapa.*/  
//-----  
  
public ResultSet comprobar_usuarios_repetidos(String apodo) throws SQLException {  
    resulQuery = null;  
    try {  
        sentenciaSQL = con.createStatement();  
        if (motor.equals("PostgreSQL")) {  
            cadSQL = "Select \"Tipo_jugador\", \"Apodo\", \"P_accion\" from mapa where \"Apodo\"=\"\" + apodo + \"\"";  
        }  
        resulQuery = sentenciaSQL.executeQuery(cadSQL);  
    } catch (SQLException e) {  
        System.out.println("\nLa actualizacion del personaje no ha podido realizarse.\n" + e.toString());  
        return resulQuery;  
    }  
    return resulQuery;  
}  
//-----  
//Permitir Acción Jugador. Función:  
/*Permite a un jugador realizar acciones. El master tendrá poder sobre esta sentencia.*/  
//-----  
  
public Integer permitir_accion_jugador(String apodo) throws SQLException {  
    resulUpdate = 0;  
    try {  
        sentenciaSQL = con.createStatement();  
        if (motor.equals("PostgreSQL")) {  
            cadSQL = "update mapa set \"P_accion\"=1 where \"Apodo\"=\"\" + apodo + \"\"";  
        }  
        resulUpdate = sentenciaSQL.executeUpdate(cadSQL);  
    } catch (SQLException e) {  
        System.out.println("\nLa actualizacion del personaje no ha podido realizarse.\n" + e.toString());  
        return resulUpdate;  
    }  
    return resulUpdate;  
}  
//-----  
//Denegar Accion Jugador. Función:  
/*Al contrario que el metodo anterior, este deniega a un jugador realizar acciones.*/  
//-----  
  
public Integer denegar_accion_jugador(String apodo) throws SQLException {  
    resulUpdate = 0;  
    try {  
        sentenciaSQL = con.createStatement();  
        if (motor.equals("PostgreSQL")) {  
            cadSQL = "update mapa set \"P_accion\"=0 where \"Apodo\"=\"\" + apodo + \"\"";  
        }  
        resulUpdate = sentenciaSQL.executeUpdate(cadSQL);  
    } catch (SQLException e) {  
        System.out.println("\nLa actualizacion del personaje no ha podido realizarse.\n" + e.toString());  
        return resulUpdate;  
    }  
    return resulUpdate;  
}
```

```

//-----
//Atacar Jugador. Función:
/*Realiza un ataque a un usuario que recibe como parametro.*/
//-----
public Integer atacar_jugador(String apodo) throws SQLException {
    resulUpdate = 0;
    try {
        sentenciaSQL = con.createStatement();
        if (motor.equals("PostgreSQL")) {
            cadSQL = "update reg_users set \"Salud\"=\"Salud\"-1 where \"Apodo\"=\"" + apodo + "\"";
        }
        resulUpdate = sentenciaSQL.executeUpdate(cadSQL);
    } catch (SQLException e) {
        System.out.println("\nLa actualizacion del personaje no ha podido realizarse.\n" + e.toString());
        return resulUpdate;
    }
    return resulUpdate;
}
//-----
//Obtener Salud. Función:
/*Obtener la vida del usuario que recibe por parametro*/
//-----

public ResultSet obtener_salud(String apodo) throws SQLException {
    resulQuery = null;
    try {
        sentenciaSQL = con.createStatement();
        if (motor.equals("PostgreSQL")) {
            cadSQL = "Select \"Salud\", \"Apodo\" from reg_users where \"Apodo\"=\"" + apodo + "\"";
        }
        resulQuery = sentenciaSQL.executeQuery(cadSQL);
    } catch (SQLException e) {
        System.out.println("\nLa actualizacion del personaje no ha podido realizarse.\n" + e.toString());
        return resulQuery;
    }
    return resulQuery;
}
//-----
//Comprobar Coordenadas. Función:
/*Sirve para comprobar si hay un usuario en las coordenadas recibidas.*/
//-----

public ResultSet comprobar_coordenadas(Integer X, Integer Y) throws SQLException {
    resulQuery = null;
    try {
        sentenciaSQL = con.createStatement();
        if (motor.equals("PostgreSQL")) {
            cadSQL = "Select \"Apodo\" from mapa where \"X\"=\"" + X + "\" and \"Y\"=\"" + Y + "\"";
        }
        resulQuery = sentenciaSQL.executeQuery(cadSQL);
    } catch (SQLException e) {
        System.out.println("\nLa actualizacion del personaje no ha podido realizarse.\n" + e.toString());
        return resulQuery;
    }
    return resulQuery;
}

//-----
// Cerrar la conexion a la base de datos.
//-----

```

```
public void cerrarConexion() throws java.sql.SQLException {
    try {
        if (resul != null) {
            resul.close();
        }
        if (sentenciaSQL != null) {
            sentenciaSQL.close();
        }
        if (con != null) {
            con.close();
        }
    } catch (SQLException e) {
        System.out.println("\nProblemas en el cierre...\n" + e.toString());
    }
}
```

NOS.java

```
package pack1;
import java.io.*;
import javax.servlet.http.*;
import java.sql.*;
import javax.servlet.ServletException;
import javax.servlet.ServletConfig;

public class NOS extends HttpServlet {
    //Connection con; //Objeto de la conexion
    private BD_NOS BD; //Objeto Base de Datos.
    ResultSet res = null;
    Integer resInt = 0;

    String oculto0 = "";
    String email0 = "";
    String password0 = "";
    String nombre0 = "";
    String apodo0 = "0";

    String idsesion = "";
    String rango = "";

    String clan0 = "2";
    String raza0 = "";
    Integer salud0 = 0;
    Integer fuerza0 = 0;
    Integer experiencia0 = 0;
    Integer dinero0 = 0;
    Integer moral = 0;
    Integer inteligencia0 = 0;
    Integer destreza0 = 0;
    Integer moral0 = 0;

    //Date Datefecnac0;

    String fecnac0 = "sysdate";

    /*En el inicio realizamos la conexión a la base de datos
    por lo que las peticiones no producirán ningun retardo de conexión.*/

    @Override
    public void init(ServletConfig config) throws ServletException {
        //Capturamos los parametros iniales del servlet.
        /*Ciclo de Vida del servlet:
        Cuando un servidor carga un servlet, ejecuta el método init del servlet.
        La inicialización se completa antes de manejar peticiones de clientes
        y antes de que el servlet sea destruido. */
        super.init(config);
        /*Enviamos el parametro admin a BD_NOS para indicarle quien está
        administrando la aplicación.*/
        String admin = getServletConfig().getInitParameter("admin");
        try {
            //Creación de la clase BD_NOS.
            BD = new BD_NOS(admin);
            //Captura de distintos posibles errores.
        } catch (ClassNotFoundException e) {
```



```

        System.out.println("Clase no encontrada. " + e.getMessage());
    } catch (InstantiationException e) {
        System.out.println("Objeto no creado. " + e.getMessage());
    } catch (IllegalAccessException e) {
        System.out.println("Acceso ilegal. " + e.getMessage());
    } catch (SQLException e) {
        System.out.println("Excepción SQL. " + e.getMessage());
    }
}
} // fin init()

@Override
protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    //response.setContentType("text/html;charset=UTF-8");
    //PrintWriter out = response.getWriter();

    /*oculto0. Este objeto se recibe siempre y proviene de las paginas JSP(tambien
    las generadas por el propio servlet.) que hacen peticiones a NOS. El origen
    proviene en un campo oculto(hidden).*/
    oculto0 = request.getParameter("oculto");
    /*Si el parametro recibido es igual a login, es que proviene de login.jsp
    y se deberan realizar las siguientes operaciones.*/
    if (oculto0.equals("login")) {
        //Obtenemos los parametros(no ocultos) de login.
        email0 = request.getParameter("email");
        password0 = request.getParameter("password");
        try {
            //Comprobamos que el email y la pass estan en la BD
            res = BD.comprobar_email_password(email0, password0);
            //Si no existen datos...
            if (!(res.next())) {
                /*No existen ni email ni password...mandamos de vuelta con error por Get.
                Hacemos esto porque login.jsp contiene un include que apunta a lineaerror.jsp
                que capta el parametro pasado por get actua de 2 formas.
                login.jsp tambien está incluido en index.jsp.*/
                //Enviamos a la página index.jsp al segundo.
                response.setHeader("Refresh",
                    "0; URL=index.jsp?E=El usuario no existe...");

                //Si existen datos...
            } else {
                //Email y password encontrado.
                //Establecemos la sesion .
                HttpSession sesionOK = request.getSession();
                //Cada idsesion será unico para cada usuario.
                idsesion = (String) sesionOK.getId();

                //Establecemos todos los parametros para la sesión actual.
                sesionOK.setAttribute("idsesion", idsesion);
                sesionOK.setAttribute("Email", res.getString("Email"));
                sesionOK.setAttribute("Password", res.getString("Password"));
                sesionOK.setAttribute("Apodo", res.getString("Apodo"));
                sesionOK.setAttribute("Nombre", res.getString("Nombre"));
                sesionOK.setAttribute("Salud", res.getString("Salud"));
                sesionOK.setAttribute("Fuerza", res.getString("Fuerza"));
                sesionOK.setAttribute("Experiencia", res.getString("Experiencia"));
                sesionOK.setAttribute("Dinero", res.getString("Dinero"));
                sesionOK.setAttribute("Moral", res.getString("Moral"));
                sesionOK.setAttribute("Inteligencia", res.getString("Inteligencia"));
                sesionOK.setAttribute("Destreza", res.getString("Destreza"));
            }
        }
    }
}

```

```
//Una vez cargados los parametros devolvemos el menu de la aplicación.
response.setHeader("Refresh", "0; URL=menu.jsp");
//Enviamos a la página menu.jsp al segundo.
}
} catch (Exception e) {
    System.out.println(e.toString());
}
}
/*Si el parametro oculto que recibimos es modper, el servlet comprende
que la petición la está realizando ficha_pj.jsp y nos dice que
quiere actualizar el usuario. Esta pagina envia el apodo,
el clan y la raza introducida por el usuario.*/
} else if (oculto0.equals("modper")) {
    //Obtenemos los valores de la sesión.
    HttpSession sesionOK = request.getSession();
    idsesion = (String) sesionOK.getId();
    email0 = (String) sesionOK.getAttribute("Email");
    password0 = (String) sesionOK.getAttribute("Password");
    nombre0 = (String) sesionOK.getAttribute("Nombre");
    /*Obtenemos el valor de los parámetros tecleados por el usuario.
    Comprobados previamente.*/
    apodo0 = request.getParameter("apodo");
    clan0 = request.getParameter("clan");
    raza0 = request.getParameter("raza");

    try {
        //Comprobamos si el Apodo que recibimos está ya en la BD.
        res = BD.comprobar_apodo(apodo0);
        //Si lo estamos repitiendo...
        if (res.next()) {
            //Devolvemos a ficha_pj con un error.
            response.setHeader("Refresh", "0;"
                + " URL=ficha_pj.jsp?E=Ya hay un usuario usando el apodo: "
                + "" + apodo0 + ". Por favor, seleccione otro.");
            //Si no se esta repitiendo el apodo...
        } else {
            //Modificamos el registro en la tabla.
            resInt = BD.modificar_personaje(email0, password0, apodo0, clan0, raza0);
            //@@@@COMPROBACION DE INSERCCION SATISFACTORIA.

            //Actualizamos los parametros de la sesion.
            sesionOK.setAttribute("Apodo", apodo0);
            sesionOK.setAttribute("Clan", clan0);
            sesionOK.setAttribute("Raza", raza0);
            //Enviamos a la página menu.jsp al segundo.
            response.setHeader("Refresh", "0; URL=menu.jsp");
        }
        //Lanzamos la cadena del error.
    } catch (Exception e) {
        System.out.println(e.toString());
    }
}
}
/*Si el campo oculto es igual a crearper, el servlet comprende
que la pagina que esta llamando es registro_usuario.jsp y que
desea insertar un usuario nuevo.*/
} else if (oculto0.equals("crearper")) {
    //Obtenemos los valores de los parámetros introducidos por el usuario.
    password0 = request.getParameter("password");
    email0 = request.getParameter("email");
    nombre0 = request.getParameter("nombre");
    fecnac0 = request.getParameter("fecnac");
}
```

```
try {
    //Comprobamos si existe ya un usuario con ese email.
    /*Aquí surge un problema: al hacer la comprobación debe mandar
    tan solo el email, ya que, si el usuario introduce un email
    que ya existe y una pass que lo más logico es que sea diferente
    (o no), la consulta devolverá que no existe ese email con esa pass.
    Solución:Un metodo ESPECÍFICO para tal cuestion.
    res = BD.comprobar_email_password(email0, password0); NO NOS VALE*/
    //INSERTAR AQUI SI NO EXISTE.
    res = BD.comprobar_email(email0);
    //Si no encontramos el email...
    if (!(res.next())) {
        //Insertamos por defecto estos valores.
        //HISTORIA:
        /*Uno de los valores añadidos de los que he tenido que prescindir era
        el tema de las habilidades de los usuarios.
        En principio, la idea era que los usuarios pudieran elegir en un listbox,
        un personaje con unos "estados" predefinidos y unas características
        propias(Ej: vampiros, lobos,...según la temática que se le quiera dar;
        magos, dragones, barbaros...).
        Cada jugador tendra unos puntos de estado por defecto y estos se irán
        modificando con el paso del tiempo o cada vez que se alcance tal numero
        de experiencia. Esto podría ser por ejemplo: disparando un metodo
        en el servlet que mediante consulta a la tabla reg_users obtenga los campos
        fecha_registro y fecha_ultima_entrada, y despues ejecute un metodo
        que calcule la experiencia de un usuario pasadole por parametros
        los dos tipos de fecha y el apodo. O bien, creando un trigger que se ejecute
        ante un posible movimiento en una tabla fechas, actualizando el campo Experiencia
        en la tabla reg_users.

        Pero como el tiempo me apremiaba en todo momento he tenido que hacer algunos
        ReCoRTeS.... ESTADOS POR DEFECTO. Todo el mundo parte con las mismas características.*/

        //apodo clan y raza. 3 parametros que se obtienen al Crear Personaje.
        /*apodo: Recibe un 0 que indica a la aplicación cuando un usuario
        a creado un personaje y cuando no. Esto me sirve para en em menu de la aplicación,
        aparezca Crear Personaje o Entrar en función del contenido de apodo.*/
        apodo0 = "0";
        clan0 = "";
        raza0 = "";
        //Estado General a TODOS.
        salud0 = 7;
        fuerza0 = 1;
        experiencia0 = 1;
        dinero0 = 1;
        moral0 = 1;
        inteligencia0 = 1;
        destreza0 = 1;
        //Insertamos los valores obtenidos
        res = BD.insertar_email_password(email0, password0, nombre0, apodo0,
            clan0, raza0, fecnac0, salud0, fuerza0, experiencia0,
            dinero0, moral0, inteligencia0, destreza0);

        //Establecemos los parametros a la sesion.
        HttpSession sesionOK = request.getSession();
        sesionOK.setAttribute("Password", password0);
        sesionOK.setAttribute("Email", email0);
        sesionOK.setAttribute("Nombre", nombre0);
        idsesion = (String) sesionOK.getId();
```

```

        sesionOK.setAttribute("idsesion", idsesion);
        sesionOK.setAttribute("Apodo", apodo0);
        sesionOK.setAttribute("Clan", clan0);
        sesionOK.setAttribute("Raza", raza0);
        sesionOK.setAttribute("Salud", salud0);
        sesionOK.setAttribute("Fuerza", fuerza0);
        sesionOK.setAttribute("Experiencia", experiencia0);
        sesionOK.setAttribute("Dinero", dinero0);
        sesionOK.setAttribute("Moral", moral0);
        sesionOK.setAttribute("Inteligencia", inteligencia0);
        sesionOK.setAttribute("Destreza", destreza0);
        //Enviamos a la página menu.jsp al segundo.
        response.setHeader("Refresh", "0; URL=menu.jsp");
        //Vaciamos el buffer si está lleno.
        response.flushBuffer();
        //Si encontramos email.
    } else {
        //Enviamos a la página login.jsp al segundo con mensaje de error.
        response.setHeader("Refresh", "0;"
            + "URL=index.jsp?E=El usuario ya existe...");
    }
} catch (Exception e) {
    System.out.println(e.toString());
}
}
/*Si el campo oculto es igual a modificardatos, el servlet comprende
que la pagina que esta llamando es modificar_registro.jsp(X) y que
desea actualizar los datos del usuario.*/
} else if (oculto0.equals("modificardatos")) {
    //@@@@@MODIFICAR DATOS REGISTRO.
    /*Para implementar esta opción deberiamos:
    Generar una pagina en el servlet(NOS.java) para que muestre en un formulario con
    la información actual del usuario, y que este pueda modificar. Hacemos comprobación de datos.
    Una vez realizado y comprobados los cambios, se actualiza la BD.*/

    /*Si el campo oculto es igual a borrarusuario, el servlet comprende
    que la pagina que esta llamando es borrar_registro.jsp(X) y que
    desea eliminar los datos del usuario.*/
} else if (oculto0.equals("borrarusuario")) {
    //@@@@@MODIFICAR O ELIMINAR DATOS REGISTRO.
    /*Para implementar esta opción tan solo deberiamos
    obtener el apodo de la sesión que esta llamando y llamar a una
    metodo de BD_NOS que haga un delete en la BD*/
}
}

}
//Cerramos la conexión.
@Override
public void destroy() {
    super.destroy();
    /*@@@try {
        con.close();
    } catch (SQLException e) {
        System.out.println("Imposible cerrar conexión:" + e.getMessage());
    }*/
} //fin destroy()
}

```

GUERRA.java

```
package pack1;

import java.io.*;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.servlet.http.*;
import java.sql.*;
import javax.servlet.ServletException;
import javax.servlet.ServletConfig;
import java.util.*;

public class guerra extends HttpServlet {

    private BD_NOS BD; //Objeto Base de Datos.
    ResultSet res = null;
    Integer resInt = 0;
    //DECLARACIONES LOCALES DEL USUARIO
    String oculto0 = "";
    String email0 = "";
    String password0 = "";
    String nombre0 = "";
    String apodo0 = "0";
    String idsesion = "";
    //DECLARACIONES LOCALES PARA MANEJO DE TURNOS.
    /*String[] vector;
    vector = {"0","0","0","0","0","0","0","0","0"};*/
    /*Declaramos el vector con cadenas vacias, ya que
    si la declaramos con algun valor no podremos utilizarlo
    para realizar consultas en la BD*/
    String[] vector = {"", "", "", "", "", "", "", "", ""};
    Integer acumulador = 0;
    Integer turno = 0;
    //DECLARAMOS E INICIALIZAMOS VARIABLES LOCALES PARA EL MAPA.
    String nom_partida = "";
    String pass_partida = "";
    Integer X = 0;
    Integer Y = 0;
    //cont nos ayuda a construir la tabla donde se asientan las imagenes.
    Integer cont = 0;
    //Creamos un hilo para mas tarde detener la aplicación por tiempo que deseemos.
    Thread th;
    //Declaramos una variable auxiliar que podamos utilizar en cualquier momento.
    Integer aux;
    //DECLARAMOS E INICIALIZAMOS VARIABLES LOCALES PARA EL PERSONAJE.
    String clan0 = "";
    String raza0 = "";
    Integer salud0 = 0;
    Integer fuerza0 = 0;
    Integer experiencia0 = 0;
    Integer dinero0 = 0;
    Integer moral = 0;
    Integer inteligencia0 = 0;
    Integer destreza0 = 0;
    Integer moral0 = 0;
    //String fecnac0 = "sysdate";

    //Realizamos la conexión a la base de datos.
```

```
@Override
public void init(ServletConfig config) throws ServletException {
    //Capturamos los parametros iniales del servlet.
    super.init(config);
    String admin = getServletConfig().getInitParameter("admin");
    try {
        //Creación de la clase BD_NOS.
        BD = new BD_NOS(admin);
        //Captura de distintos posibles errores.
    } catch (ClassNotFoundException e) {
        System.out.println("Clase no encontrada. " + e.getMessage());
    } catch (InstantiationException e) {
        System.out.println("Objeto no creado. " + e.getMessage());
    } catch (IllegalAccessException e) {
        System.out.println("Acceso ilegal. " + e.getMessage());
    } catch (SQLException e) {
        System.out.println("Excepción SQL. " + e.getMessage());
    }
}
} // fin init()

@Override
protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    PrintWriter out = response.getWriter();
    //Obtenemos el parametro oculto.
    oculto0 = request.getParameter("oculto");
    /*Si el campo oculto es igual a activar_acción...
    LAS ACCIONES ESTAN ANULADAS.*/
    if (oculto0.equals("activar_accion")) {
        apodo0 = request.getParameter("lista_pjs");
        try {
            resInt = BD.permitir_accion_jugador(apodo0);
            cargar_mapa_partida(request, response);
        } catch (Exception e) {
            System.out.println(e.toString());
        }
        /*Si es igual a denegar_acción...
        LAS ACCIONES ESTAN ANULADAS.*/
    } else if (oculto0.equals("denegar_accion")) {
        apodo0 = request.getParameter("lista_pjs");
        try {
            resInt = BD.denegar_accion_jugador(apodo0);
            cargar_mapa_partida(request, response);
        } catch (Exception e) {
            System.out.println(e.toString());
        }
        /*Si el campo oculto es igual a crearpartida...
        El servlet entiende que entrada_mapa.jsp ha hecho la peticion de crear partida.
        La idea es que cualquier jugador pueda crear partida.
        EL MASTER ESTÁ ANULADO.*/
    } else if (oculto0.equals("crearpartida")) {
        //Obtenemos la sesión.
        HttpSession sesionOK = request.getSession();

        try { //Traemos los parametros de la sesión.
            nom_partida = (String) request.getParameter("nom_partida");
            pass_partida = (String) request.getParameter("pass_partida");
            apodo0 = (String) sesionOK.getAttribute("Apodo");
```

```

/*Comprobamos si hay un usuario MASTER en la partida. Esto permite
saber si ya hay una partida creada.*/
res = BD.comprobar_mapa();
//Si hay partida...
if (res.next()) {
    //Devolvemos a menu_juego.jsp con error por metodo Get.
    response.setHeader("Refresh", "0; URL=menu_juego.jsp?A=Ya hay creada una partida");
} else {
    //Si no hay ningun MASTER, limpiamos el mapa.
    for (int i = 1; i <= 3; i++) {
        for (int j = 1; j <= 3; j++) {
            //Creamos el mapa con la ayuda de este bucle for.
            resInt = BD.crear_mapa(i, j);
        }
    }
    //Una vez limpio podemos introducir al usuario Master en la partida.
/*Como comentaba, la idea era que un usuario tambien pudiera formar parte de la partida.
Así lo colocariamos en un punto aleatorio.(Si no tuvieramos un metodo especifico para ello).*/
    //x = (Integer) sesionOK.getAttribute("X");
    //y = (Integer) sesionOK.getAttribute("Y");

    /*Llamamos a introducir master con los parametros obtenidos.
Como queremos dejar al margen al MASTER, el metodo introducir master está
preparado para insertar al MASTER en la posición ( 0 - 0 ).*/
    resInt = BD.introducir_master(apodo0, nom_partida, pass_partida);
    //Cargamos el mapa para el MASTER.
    cargar_mapa_partida(request, response);
}
} catch (SQLException ex) {
    Logger.getLogger(NOS.class.getName()).log(Level.SEVERE, null, ex);
}
}
/*Si el campo oculto es igual a unirpartida...
El servlet entiende que join_battle.jsp ha hecho la peticion de unir a la partida.
Todos los usuarios que quieran unirse a la partida pasarán por aquí.*/
} else if (oculto0.equals("unirpartida")) {
    //Obtenemos la sesion del usuario.
    HttpSession sesionOK = request.getSession();
    try {
        //Traemos los parametros introducidos en join_battle.jsp.
        nom_partida = (String) request.getParameter("nom_partida");
        pass_partida = (String) request.getParameter("pass_partida");
        apodo0 = (String) sesionOK.getAttribute("Apodo");
        //Comprobamos si existe una partida con ese nombre y esa pass.
        res = BD.comprobar_partida(nom_partida, pass_partida);
        //Si no existe...
        if (!res.next()) {
            //Devolvemos al segun a menu_juego con un error pasado por Get.
            response.setHeader("Refresh", "0; URL=menu_juego.jsp?E=No existe la partida");
            //Si existe...
        } else {
            /*Comprobamos que el usuario no este ya en la partida. Esto me sirve por
si el usuario se a desconectado sin querer.*/
            res = BD.comprobar_usuarios_repetidos(apodo0);
            //Si existe ya el usuario en la partida...
            if (res.next()) {
                //Comprobamos si tiene salud.
                salud0 = comprobar_salud(apodo0);
                //Si es menor o igual a 0...
                if (salud0 <= 0) {
                    //El jugador está muerto.
                }
            }
        }
    }
}

```

```
        response.setHeader("Refresh", "0;"
            + "URL=menu_juego.jsp?A=No tienes salud suficiente para entrar en la partida.");
        //Si es mayor que 0...
    } else {
        //Metemos el usuario en el vector. El vector contiene los jugadores online.
        meter_en_vector(apodo0);
        //Cargamos la partida.
        cargar_mapa_partida(request, response);
    }
    //Si no existe el usuario en la partida...
} else {
    //Comprobamos la salud del usuario.
    salud0 = comprobar_salud(apodo0);
    //Si salud es menos o igual a 0...
    if (salud0 <= 0) {
        //Si el jugador esta muerto...devolvemos a menu juego con error.
        response.setHeader("Refresh", "0;"
            + "URL=menu_juego.jsp?A=No tienes salud suficiente para entrar en la partida.");
        //Si es mayor que 0...
    } else {
        //Repetiremos el siguiente proceso mientras las coordenadas estén ocupadas.
        do {
            //Traemos los numeros aleatorios.
            X = traer_num_aleatorio();
            Y = traer_num_aleatorio();
            /*aux tiene su primer uso. Devolvera 0 si las coordenadas están ocupadas
            y 1 cuando estén libres.*/
            aux = comprobar_coordenadas(X, Y);
        } while (aux == 0);
        /*Como estamos seguros de las coordenadas no están ocupadas,
        procedemos a unir al jugador.*/
        resInt = BD.unir_jugador(apodo0, X, Y);
        try {
            out.println("Conectando...");
            //Metemos el jugador en el vector. Online.
            meter_en_vector(apodo0);
            //Dormimos la ejecución del codigo 10 segundos. Ganamos tiempo.
            th.sleep(10000);
            //Si todo ha ido correctamente, se cargará el mapa.
            cargar_mapa_partida(request, response);
        } catch (Exception e) {
            System.out.println(e.toString());
        }
    }
}
}
} catch (SQLException ex) {
    Logger.getLogger(NOS.class.getName()).log(Level.SEVERE, null, ex);
}
}
/*Si el campo oculto es igual a actualizar_partida...
El servlet entiende que espera.jsp ha hecho la peticion de atilizar tu estado en la partida.
Mientras estemos esperando turno, pasaremos por quí cada 5 segundos.*/
} else if (oculto0.equals("actualizar_mapa")) {
    //Traemos la sesión.
    HttpSession sesionOK = request.getSession();
    //Traemos el apodo.
    apodo0 = (String) sesionOK.getAttribute("Apodo");
    /*Pasamos al "cerebro de los turnos". Este se encarga de controlar los turnos de
    cada usuario.*/
    control_turno(request, response, apodo0);
}
```



```
/*Si el campo oculto es igual a cerrar_mapa...
El servlet entiende que se ha hecho una petición desde el menu del
MASTER para desconectar a todos los usuarios de la partida, y abandonar su puesto.*/
} else if (oculto0.equals("cerrar_mapa")) {
//Traemos la sesion del usuario.
HttpSession sesionOK = request.getSession();
//traemos el apodo.
apodo0 = (String) sesionOK.getAttribute("Apodo");
try {
//Obtenemos tos los personajes de la partida.
res = BD.mostrar_pjs(apodo0, 0);
//Mientras encontremos usuarios...
while (res.next()) {
//Lo borramos de la partida.
BD.desconectar_jugador(res.getString("Apodo"));
//Lo desconectamos del mapa.
sacar_de_vector(res.getString("Apodo"));
}
//Abandonamos el puesto de master.
BD.sacar_master();
//Sacamos al usuario de la partida.
sacar_de_vector(apodo0);
//Retornamos a menu juego con aviso de exito al cerrar el mapa.
response.setHeader("Refresh", "0; URL=menu_juego.jsp?A=Se ha eliminado la partida correctamente");
} catch (Exception e) {
System.out.println(e.toString());
}
}
/*Si el campo oculto es igual a desconectar_mapa...
El servlet entiende que se ha hecho una petición desde el menu del
usuario para que lo desconecte. y abandone el mapa. DEBEREMOS ESPERAR NUESTRO TURNO.*/ else if
(oculto0.equals("desconectar_mapa")) {
//Traemos la sesión.
HttpSession sesionOK = request.getSession();
//Traemos el apodo.
apodo0 = (String) sesionOK.getAttribute("Apodo");
try {
//Desconectamos al jugador.
BD.desconectar_jugador(apodo0);
//Lo sacamos de la partida.
sacar_de_vector(apodo0);
//Pasamos nuestro turno.
pasar_turno(apodo0);
//Devolvemos al menu del juego. Con un aviso.
response.setHeader("Refresh", "0; URL=menu_juego.jsp?A=Has salido de la partida correctamente");
} catch (Exception e) {
System.out.println(e.toString());
}
}
/*Si el campo oculto es igual a atacar...
El servlet entiende que se ha hecho una petición desde el menu del
jugador para atacar al usuario que recibe de un formulario.*/
else if (oculto0.equals("atacar")) {
//Traemos las sesion
HttpSession sesionOK = request.getSession();
try {
//Traemos el apodo.
apodo0 = (String) sesionOK.getAttribute("Apodo");
//Si es el turno del usuario...
if (consulta_turno_usuario(apodo0) == 1) {
//Traemos el apodo seleccionado.
String apodo = request.getParameter("lista_pjs");
```

```

//Si este apodo está vacío...
if (apodo.equals("")) {
    //Pasamos a control turno para que controle la situación.
    control_turno(request, response, apodo0);
} //Si contiene un apodo...
else {
    //Atacamos al jugador seleccionado.
    BD.atacar_jugador(apodo);
    //Pasamos turno.
    pasar_turno(apodo0);
    //Controlamos la nueva situación del usuario.
    control_turno(request, response, apodo0);
}
//Si no es el turno del usuario...
} else {
    //Controlamos la situación.
    control_turno(request, response, apodo0);
}
} catch (Exception e) {
    System.out.println(e.toString());
}
}
}

@Override
//Metodo destroy del servlet.
public void destroy() {
    super.destroy();
}

////////////////////////////////////METODOS
/*CARGAR MAPA PARTIDA:
Metodo que carga la partida a un usuario.*/
protected void cargar_mapa_partida(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {

    //Establecemos el contenido de la respuesta.
    response.setContentType("text/html;charset=UTF-8");
    //Traemos la sesión del usuario.
    HttpSession sesionOK = request.getSession();
    //Establecemos una salida para mostrar por pantalla.
    PrintWriter out = response.getWriter();
    //Limpiamos las respuestas al usuario.
    response.reset();
    //cont. nos permite mostrar correctamente el mapa.
    cont = 0;
    //Traemos el apodo
    apodo0 = (String) sesionOK.getAttribute("Apodo");
    //Traemos la salud del jugador.
    salud0 = comprobar_salud(apodo0);
    //Si la salud es menor o igual a 0...
    if (salud0 <= 0) { //Jugador muerto.
        //Si le llegaba el turno en el momento de morir...
        if (consulta_turno_usuario(apodo0) == 1) {
            //Sacamos al usuario del vector. Online
            sacar_de_vector("ok-" + apodo0);
            //Y lo desconectamos de la partida.
            desconectar_de_la_partida(request, response);
            //Retornamos al menú del juego con un mensaje que le informe de su estado.
            response.setHeader("Refresh", "0;URL=menu_juego.jsp?A=Te han matado PAQUETE!");
        }
    }
}

```

```

    } //Si moria y no era su turno...
else {
    //Lo sacamos del vector. Online.
    sacar_de_vector(apodo0);
    //Lo desconectamos del mapa.
    desconectar_de_la_partida(request, response);
    //Retornamos al menu del juego con un mensaje que informe de su estado.
    response.setHeader("Refresh", "0;URL=menu_juego.jsp?A=Te han matado PAQUETE!");
}

} //Si la salud es mayor que 0...
else { //Jugador vivo.
    try {
        //Cargamos la cabecera de la pagina de la partida. html y jsp.
        cabecera_mapa(request, response, apodo0);
        /*Reutilizamos comprobar usuarios repetidos. Como sabemos que nos va devolver
        la información del usuario que recibe por parametros...*/
        res = BD.comprobar_usuarios_repetidos(apodo0);
        //Si existen filas...
        if (res.next()) {
            //Si es el MASTER...
            if (res.getString("Tipo_jugador").equals("Master")) {
                /*Mostramos el menu del master, que permite o restringe la acción de un usuario.
                Recibe el apodo para pasarlo a los metodos que utiliza internamente.*/
                menu_master(request, response, res.getString("Apodo"));
            } //Si es el PJ(jugador)...
            else if (res.getString("Tipo_jugador").equals("PJ")) {
                //Mostramos el menu del jugador.
                menu_pj(request, response, res.getString("Apodo"));
            }
        } //Si no existen filas...
        else { //Probablemente el MASTER ha desconectado al usuario o se ha cerrado el mapa.
            //Mostramos un mensaje al usuario.
            out.println("Has sido desconectado de la partida");
            //Retornamos a menu de juego.
            response.setHeader("Refresh", "2; URL=menu_juego.jsp");
        }
    } catch (Exception e) {
        System.out.println(e.toString());
    }
} //Controlamos la seccion del mapa en otro try diferente.
try {
    //Traemor la información de la tabla reg_users. Excepto al MASTER.
    res = BD.mostrar_mapa();
    //Cargamos los script que usan los formularios de la partida.
    script_mapa(request, response);
    /*Formulario que llama al mismo guerra.java con parametro oculto.
    La función refresco(); proviene del script. Realiza un submit cada 10 segundos.*/
    out.println("<center><form id='formMapa' name='formMapa' method='post' action='guerra'
onSubmit='\"javascript:return refresco();\">");
    //Comienzo de la tabla que mostrará las imagenes.
    out.println("<table border='0' cellspacing='0' cellpadding='0' >");
    //Si existen filas que mostrar...
    while (res.next()) {
        //cont, controla los comienzos de las coordenadas X del mapa.
        if (cont == 0 || cont == 3 || cont == 6) {
            //Nueva fila.
            out.println("<tr>");
        }
        //Si la fila tiene estado L de libre...

```

```

if ((res.getString("Estado")).equals("L")) {
    //Mostramos la imagen que representa que esas coordenadas están libres.
    out.println("<td><img src=image_base/1.jpg ><td>");
    //Aumentamos contador para pasar a la siguiente coordenada.
    cont++;
} //Si la fila tiene estado O de ocupado...
else if ((res.getString("Estado")).equals("O")) {
    //Traemos la X.
    X = res.getInt("X");
    //Traemos la Y.
    Y = res.getInt("Y");
    //Traemos el apodo.
    apodo0 = res.getString("Apodo");
    //Mostramos la imagen que tiene asociado las coordenadas ocupadas.
    /*Ademas tiene una función deshabilitada por ahora. Esta permite que
    al hacer click en la imagen muestre las coordenadas y el apodo del usuario
    en un os campos de texto en la misma pagina.*/
    out.println("<td><img src=image_base/2.jpg title='" + apodo0
        + "' onMouseOver='MOver()' onClick='\"MClick(\" + X + \",\" + Y + \",\" + apodo0 + \")\" ><td>");
    //Aumentamos el contador.
    cont++;
}
}
if (cont == 0 || cont == 3 || cont == 6 || cont == 9) {
    out.println("</tr>");
}
}
out.println("</table>");
out.println("<input type='hidden' name='oculto' value='actualizar_mapa' >");
//out.println("<input type='submit' value='Actualizar Mapa'>");
out.println("</form></center><br><br>");

pie_mapa(request, response);

} catch (Exception e) {
    System.out.println(e.toString());
}
}
}
}
/*Traer Numero Aleatorio. Devuelve un numero aleatorio del 1 al 3.*/
protected Integer traer_num_aleatorio() throws ServletException, IOException {
    Integer num = 0;
    Random x = new Random();
    for (int i = 0; i < 1; i++) {
        num = ((Integer) (x.nextInt(3) + 1));
    }
    return num;
}
/*Comprobar Coordenadas. Recibe las coordenadas(X e Y) y devuelve 1 si están libres, o 0 si están ocupadas. */
protected Integer comprobar_coordenadas(Integer X, Integer Y) throws ServletException, IOException {
    try {
        res = BD.comprobar_coordenadas(X, Y);
        if (res.next()) {
            if (res.getString("Apodo").equals("L")) {
                return 1;
            } else {
                return 0;
            }
        }
    } else {
        return 0;
    }
}
}

```

```
    } catch (Exception e) {
        System.out.println(e.toString());
        return 0;
    }
}
/*Comprobar Salud. Devuelve la salud del apodo que recibe por parametro, o devuelve 0 si no encuentra a este..*/
protected Integer comprobar_salud(String apodo) throws ServletException, IOException {

    try {
        res = BD.obtener_salud(apodo);
        if (res.next()) {
            return res.getInt("Salud");
        } else {
            return 0;
        }
    } catch (Exception e) {
        System.out.println(e.toString());
        return 0;
    }
}

/*Desconectar de la Partida. Recibe la sesion del usuario que llama y toma su apodo para desconectarle.*/
protected void desconectar_de_la_partida(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
    HttpSession sesionOK = request.getSession();
    apodo0 = (String) sesionOK.getAttribute("Apodo");
    try {
        BD.desconectar_jugador(apodo0);
        //response.setHeader("Refresh", "0; URL=menu_juego.jsp?A=Has salido de la partida correctamente");
    } catch (Exception e) {
        System.out.println(e.toString());
    }
}

/*Menu Master. Solo visible al MASTER. INHABILITADO.*/
protected void menu_master(HttpServletRequest request, HttpServletResponse response, String apodo) throws
ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    PrintWriter out = response.getWriter();
    /*Menu admin permiso. Recibe un apodo y establece a 1 el permiso de acción.*/
    menu_admin_permiso(request, response, apodo0);
    /*Menu admin deniega. Recibe un apodo y establece a 0 el permiso de acción.*/
    menu_admin_deniega(request, response, apodo0);

    //Formulario que muestra un botón para cerrar el mapa.
    /*Tambien so podria mostrar el botón solo con un id y un action. Pero lo dejo así para añadir mas información
    en un futuro.*/
    out.println("<center><form id='formMapa1' name='formMapa1' method='post' action='guerra'>");
    out.println("<input type='hidden' name='oculto' value='cerrar_mapa'>");
    out.println("<input type='submit' value='Cerrar Mapa'>");
    out.println("</form></center>");
}

/*Menu PJ. Solo visible al PJ.*/
protected void menu_pj(HttpServletRequest request, HttpServletResponse response, String apodo) throws
ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    PrintWriter out = response.getWriter();
    try {
        //Traemos los usuarios que está en el mapa, excepto el apodo.
        res = BD.mostrar_pjs(apodo, 0);
        //Formulario que presenta un listbox con los usuarios obtenidos.
```

```

out.println("<form id='formMapa2' name='formMapa2' method='post' action='guerra'>");
out.println("Selecciona jugador a atacar:");
out.println("<select name='lista_pjs'>");
out.println("<option selected label='Selecciona Jugador'></option>");
//Mientras existan usuarios.
while (res.next()) {
    //Lo presentamos en forma de opcion para el listbox.
    out.println("<option value='\" + res.getString("Apodo") + "\">\" + res.getString("Apodo") + "\"</option>");
}
//Cerramos el listbox y el formulario con campo oculto y un botón. Submit.
out.println("</select>");
out.println("<br><center>");
out.println("<input type='hidden' name='oculto' value='atacar' >");
out.println("<input type='submit' value='Atacar'>");
out.println("</center></form><br>");

//Formulario que contiene un campo oculto y un botón. Desconectar de la partida.
out.println("<br><center><form id='formMapa3' name='formMapa3' method='post' action='guerra'>");
out.println("<input type='hidden' name='oculto' value='desconectar_mapa' >");
out.println("<input type='submit' value='Desconectar Partida'>");
out.println("</form></center><br>");
} catch (Exception e) {
    System.out.println(e.toString());
}
}

//Cabecera de la pagina de acción en NOS. Mapa Partida. html y jsp.
protected void cabecera_mapa(HttpServletRequest request, HttpServletResponse response, String apodo) throws
ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    PrintWriter out = response.getWriter();
    //Inicializacion de aux. Servira pa indicarnos si es nuestro turno.
    aux = 0;
    //Cabecera HTML.
    out.println("<html>");
    out.println("<head>");
    out.println("<meta http-equiv='Content-Type' content='text/html; charset=UTF-8' >");
    out.println("<title>PARTIDA</title>");
    out.println("</head>");
    //Cargamos la funcion del script al iniciar body.
    out.println("<body onLoad='refresco()'>");
    //Tabla. Muestra los jugadores ONLINE(vector) junto con su salud.
    out.println("<table border='1'>");
    out.println("<tr><td>Jugador");
    out.println("<td>Salud</tr>");
    //Recorremos el vector.
    for (int i = 0; i < vector.length; i++) {
        out.println("<tr>");
        /*Si el turno es del jugador que llama...Mostramos solo el apodo*/
        if (vector[i].equals("ok-" + apodo)) {
            out.println("<td>" + apodo);
            out.println("<td>" + comprobar_salud(apodo));
            //aux recibe un 1 para indicar que es nuestro turno.
            aux = 1;
        } //Si es el turno de otro jugador...
        else {
            /*Si empieza por ok-...*/
            if (vector[i].startsWith("ok-")) {
                /*Mostramos solo el Apodo. A partir del caracter 3 incluyendo este. Así saco el apodo

```

```

        limpio, del vector.*/
        out.println("<td>" + vector[i].substring(3));
        //Si su vida es mayor que 0.
        if (comprobar_salud(vector[i].substring(3)) > 0) {
            //Muestro la vida.
            out.println("<td>" + comprobar_salud(vector[i].substring(3)));
        }
    } //Si es jugador en espera de turno...Diferente a cadena vacia.
    else if (!vector[i].equals("")) {
        //Mostramos apodo.
        out.println("<td>" + vector[i]);
        //Si la vida es mayor que 0.
        if (comprobar_salud(vector[i]) > 0) {
            //Mostramos vida.
            out.println("<td>" + comprobar_salud(vector[i]));
        }
    }
}
//Cerramos fila.
out.println("</tr>");
}
//Cerramos la tabla.
out.println("</table>");
//Solo si es el turno del usuario que carga el mapa, aparecerá el siguiente mensaje.
if (aux == 1) {
    out.println("<center><h2>ES TU TURNO</h2></center>");
}
}
//Cierre de la pagina de accion de la partida. Mapa Partida. html y jsp.
protected void pie_mapa(HttpServletRequest request, HttpServletResponse response) throws ServletException,
IOException {
    response.setContentType("text/html;charset=UTF-8");
    PrintWriter out = response.getWriter();
    //Linea error estará operativa en el momento que realice un metodo Get en guerra.java.
    out.println("<jsp:include page='lineaerror.jsp'></jsp:include>");
    out.println("</body>");
    out.println("</html>");
}
//Script de la pagina de acción de la partida. Mapa Partida. JavaScript.
protected void script_mapa(HttpServletRequest request, HttpServletResponse response) throws ServletException,
IOException {
    response.setContentType("text/html;charset=UTF-8");
    PrintWriter out = response.getWriter();
    out.println("<script type='text/javascript' language='javascript'>");
    out.println("function refresco(){}");
    out.println("setTimeout('\document.formMapa.submit();\",'',10000);");
    out.println("{}");
    //-----
    //FUNCIONES DESHABILITADAS
    /*
    out.println("function MOver() {}");
    //out.println("function MClick() {alert('MClick')}");
    out.println("function MClick(X,Y,apodo) {}");
    //out.println("alert(X);");
    //out.println("alert(Y);");
    out.println("document.formMapa.X.value=X;");
    out.println("document.formMapa.Y.value=Y;");
    out.println("document.formMapa.apodo.value=apodo;");
    out.println("{}");*/
}

```

```

//-----
out.println("</script>");
}
/*Menu admin permiso. Muestra un formulario con un listbox que contiene los jugadores
que NO tienen permiso de acción.
Solo visible al MASTER.*/
protected void menu_admin_permiso(HttpServletRequest request, HttpServletResponse response, String apodo)
throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    PrintWriter out = response.getWriter();
    try {
        //Traemos los jugadores.
        res = BD.mostrar_pjs(apodo, 0);
        out.println("<FORM name='form1' ACTION='guerra' METHOD='Post' onSubmit='\"javascript:return
validar2();\">");
        out.println("Insertar jugador:");
        out.println("<select name='\"lista_pjs\">");
        out.println("<option selected name='\"Selecciona Jugador\"></option>");
        //Mientras existan...
        while (res.next()) {
            //Mostramos el apodo como opción del listbox.
            out.println("<option value='\" + res.getString("Apodo") + "\">\" + res.getString("Apodo") + "\"</option>");
        }
        //Cerramos el listbox
        out.println("</select>");
        //Boton que hace el submit.
        out.println("<INPUT TYPE='submit' NAME='Enviar' VALUE='Permitir Acción' >");
        //Campo oculto que interpretara guerra.java.
        out.println("<input type='hidden' name='oculto' value='activar_accion'>");
        //Cerramos el form.
        out.println("</FORM>");
    } catch (Exception e) {
        System.out.println(e.toString());
    }
}
}
/*Menu admin deniega. Muestra un formulario con un listbox que contiene los jugadores ONLINE.
Solo visible al MASTER. Exactamente igual anterior, a diferencia de que solo muestra
usuarios que SI tienen permiso de acción.*/
protected void menu_admin_deniega(HttpServletRequest request, HttpServletResponse response, String apodo)
throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    PrintWriter out = response.getWriter();
    try {
        res = BD.mostrar_pjs(apodo, 1);
        out.println("<FORM name='form1' ACTION='guerra' METHOD='Post' onSubmit='\"javascript:return
validar2();\">");
        out.println("Eliminar jugador:");
        out.println("<select name='\"lista_pjs\">");
        out.println("<option selected name='\"Selecciona Jugador\"></option>");
        while (res.next()) {
            out.println("<option value='\" + res.getString("Apodo") + "\">\" + res.getString("Apodo") + "\"</option>");
        }
        out.println("</select>");
        out.println("<INPUT TYPE='submit' NAME='Enviar' VALUE='Denegar Acción' >");
        out.println("<input type='hidden' name='oculto' value='denegar_accion'>");
        out.println("</FORM>");
    } catch (Exception e) {
        System.out.println(e.toString());
    }
}
}
}

```



```
//METODOS PARA EL MANEJO DE SALIDAS Y ENTRDAS DE LA PARTIDA CON UN VECTOR
```

```
/*Inicializamos el vector "por si las moscas".*/
```

```
protected void iniciar_vector() throws ServletException, IOException {
```

```
    //Recorremos el vector añadiendo cadenas vacias.
```

```
    for (int i = 0; i < vector.length; i++) {
```

```
        vector[i] = "";
```

```
    }
```

```
}
```

```
/*Añadimos a un usuario al vector.*/
```

```
protected void meter_en_vector(String apodo) throws ServletException, IOException {
```

```
    /*La primera vez que se llama al servlet, acumulador valdrá 0. Cuando un 2º jugador
```

```
    llame a meter en vector, acumulador valdrá 1.*/
```

```
    vector[acumulador] = apodo;
```

```
    acumulador++;
```

```
    optimizar_vector();
```

```
}
```

```
/*Con optimizar vector soluciono el problema de que un usuario pueda desconectarse y dejar un espacio vacio entre el usuario anterior y el siguiente.*/
```

```
protected void optimizar_vector() throws ServletException, IOException {
```

```
    for (int i = 0; i < ((vector.length) - 1); i++) {
```

```
        String var = vector[i];
```

```
        /*Si encuentra una cadena vacía, trae la posicion posterior a esta.
```

```
        Pasa el hueco a la derecha*/
```

```
        if (var.equals("")) {
```

```
            vector[i] = vector[(i + 1)];
```

```
            vector[i + 1] = "";
```

```
        }
```

```
    }
```

```
}
```

```
/*Sacar vector, saca a cualquier usuario del vector sea o no su turno.*/
```

```
protected void sacar_de_vector(String apodo) throws ServletException, IOException {
```

```
    for (int i = 0; i < vector.length; i++) {
```

```
        if (vector[i].equals(apodo) || vector[i].equals("ok-" + apodo)) {
```

```
            vector[i] = "";
```

```
            acumulador--;
```

```
            optimizar_vector();
```

```
        }
```

```
    }
```

```
}
```

```
////////////////////////////////////METODOS PARA EL MANEJO DE TURNOS
```

```
/*Establecer turno, inserta la cadena "ok-" delante del apodo del usuario.
```

```
Cuando la aplicación vea esto entendera que es el turno del usuario.*/
```

```
protected void establecer_turno(String apodo) throws ServletException, IOException {
```

```
    for (int i = 0; i < vector.length; i++) {
```

```
        if (vector[i].equals(apodo)) {
```

```
            vector[i] = "ok-" + apodo;
```

```
        }
```

```
    }
```

```
}
```

```
/*Pasar turno, permite a un usuario soltar el turno a su predecesor en el vector.
```

```
Solo el usuario con turno podria realizar las siguientes acciones.*/
```

```
protected void pasar_turno(String apodo) throws ServletException, IOException {
```

```
    for (int i = 0; i < vector.length; i++) {
```

```
        //Si es el turno del usuario que llama...
```

```
        if (vector[i].equals("ok-" + apodo)) {
```

```
            //Si va a pasar el turno a una cadena vacia... Entiende que no hay mas jugadores.
```

```
            if (vector[i + 1].equals("")) {
```

```
                //Pasamos el turno al primer jugador.
```



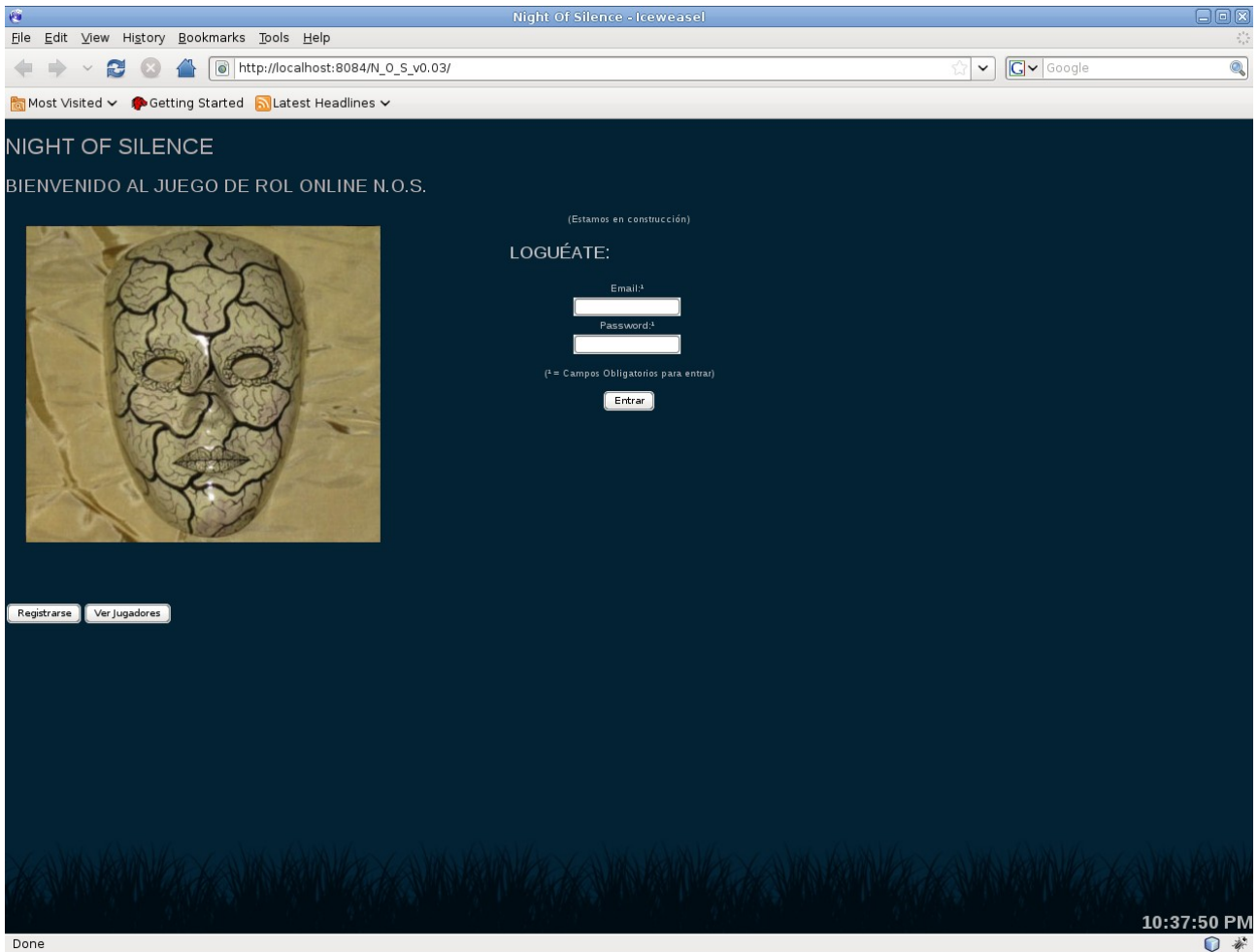
```

else if (consulta_turno_inicial() == 0) {
    //Si es mi turno...
    if (consulta_turno_usuario(apodo) == 1) {
        /*Turno ahora valdrá 2. Si pasamos una 2ª vez por aquí,
        empezaremos a controlar si llega el fin de la partida.*/
        turno++;
        //Si turno es mayor a 2...
        if (turno > 2) {
            //Controlar fin de la partida.
            control_fin_partida(request, response, apodo);
            turno = 0;
        }
        //Cargamos la partida.
        cargar_mapa_partida(request, response);
    }
    //Si hay partida y no es mi turno.
    else {
        //Controlamos fin de la partida.
        control_fin_partida(request, response, apodo);
        /*Enviamos a una pagina de espera que se refresca cada 5 segundos.
        Enviando campo oculto actualizar_mapa a guerra.java..*/
        response.setHeader("Refresh", "0; URL=espera.jsp");
    }
}
}
}
/*Control fin de partida, permite saber cuando un usuario se ha quedado solo en el mapa.*/
protected void control_fin_partida(HttpServletRequest request, HttpServletResponse response, String apodo) throws
ServletException, IOException {
    PrintWriter out = response.getWriter();
    Integer acum = 0;
    try {
        //Recorremos el vector.
        for (int i = 0; i < vector.length; i++) {
            //Si es diferente al apodo del usuario que llama y a cadena vacía...
            if (!vector[i].equals(apodo) && !vector[i].equals("")) {
                /*Obtenemos la salud del usuario del vector.*/
                res = BD.obtener_salud(vector[i]);
                //Si existen filas...
                if (res.next()) {
                    //Acumulamos la salud de los personajes ONLINE.
                    acum = acum + res.getInt("Salud");
                }
            }
        }
    } catch (Exception e) {
        System.out.println(e.toString());
    }
    //Si el acumulador es 0 o null...
    if (acum == 0 || acum == null) { //No hay jugadores vivos a los que atacar.
        //Salimos del vector.
        sacar_de_vector(apodo);
        //Desconectamos de la partida.
        desconectar_de_la_partida(request, response);
        //Retornamos al menu del juego con Aviso de triunfo.
        response.setHeader("Refresh", "0; URL=menu_juego.jsp?A=Has ganado la partida");
    }
}
}
}
}

```

JSPs

index.jsp



– index.jsp

Muestra la pantalla de Bienvenida.

Botones que enlazan con: `registro_usuario.jsp`, `ver_jugadores.jsp`.

Ademas con *include* incrustamos a `login.jsp`.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<link rel="stylesheet" href="estilo.css">
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title class="art-header">Night Of Silence</title>
</head>
<body>
<span id="liveclock" style="position:absolute;right:0;bottom:0;"></span><script language="JavaScript"
type="text/javascript">
<!--
function show5(){
if (!document.layers&&!document.all&&!document.getElementById)
return

var Digital=new Date()
var hours=Digital.getHours()
var minutes=Digital.getMinutes()
var seconds=Digital.getSeconds()

var dn="PM"
if (hours<12)
dn="AM"
if (hours>12)
hours=hours-12
if (hours==0)
hours=12

if (minutes<=9)
minutes="0"+minutes
if (seconds<=9)
seconds="0"+seconds
//change font size here to your desire
myclock="<font size='5' face='Arial' ><b><font size='1'></font></br>" +hours+":"+minutes+":"
+seconds+" "+dn+ "</b></font>"
if (document.layers){
document.layers.liveclock.document.write(myclock)
document.layers.liveclock.document.close()
}
}
```

```
else if (document.all)
    liveclock.innerHTML=myclock
else if (document.getElementById)
    document.getElementById("liveclock").innerHTML=myclock
setTimeout("show5()",1000)
}
window.onload=show5
//-->
</script>
<center>
    <div id="titulo">
        <h1>Night Of Silence</h1>
        <h2>Bienvenido al Juego de Rol online N.O.S.</h2>
    </div>
    <div id="menuhori">
        <table>
            <tr>
                <td><input type="button" onclick="location='registro_usuario.jsp'" value="Registrarse"></td>
                <td><input type="button" onclick="location='ver_jugadores.jsp'" value="Ver Jugadores" ></td>
            </tr>
        </table>
    </div>
    (Estamos en construccion)<br>
    <div id="central">
        
    </div>
    <div id="login">
        <h3>Logueate:</h3>

        <jsp:include page="login.jsp"></jsp:include><br>
    </div>
</center>
</body>
</html>
```

login.jsp

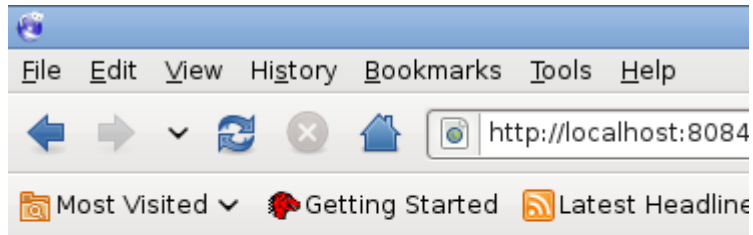
```
<%@page contentType="text/html; charset=UTF-8" session="true" language="java" import="java.util.*" %>
<html>
  <head><title>Proceso de login</title>
</head>
<body>
  <script type="text/javascript" language="javascript">
    function mensaje() {alert(document.URL)};

    //Validamos el nombre
    function validar() {
      //validar email (No puede ser blanco)
      if (document.form1.email.value.length==0){
        alert("El campo email no puede estar vac\u00edo ...")
        document.form1.email.focus()
        return false;
      }
      //validar email (Tiene que tener una estructura dada)(JTagua)
      var er = new RegExp(/^[A-Za-z0-9_-\.\.]+\@[A-Za-z0-9_-\.\.]{2,}\.[A-Za-z0-9]{2,}(\.[A-Za-z0-9])?/);
      var valemail = document.form1.email.value;
      //var er = /\w+([\.-]?\w+)*@\w+([\.-]?\w+)*(\.\w{2,3,4})+$/; (Otra expresion regular)
      if (!er.test(valemail)){
        alert("La direcci\u00f3n email es incorrecta.");
        document.form1.email.focus();
        return false;
      }
      //validar password (No puede ser blanco).
      if (document.form1.password.value.length==0){
        alert("El campo password no puede estar vac\u00edo ...")
        document.form1.password.focus();
        return false;
      }
      //Controlados TODOS los errores enviamos el formulario.
      document.form1.submit();
      return true;
    }
  </script>
```



```
<form name ="form1" action="NOS" method="Post" onreset="mensaje()" onsubmit="javascript:return validar();" >
  <table>
    <tr>
      <th>Email:&sup1;</th>
    </tr>
    <tr>
      <td> <input type="text" name="email" size=15></td>
    </tr>
    <tr>
      <th>Password:&sup1; </th>
    </tr>
    <tr>
      <td><input type="password" name="password" size=15></td>
    </tr>
  </table>
  <br>(&sup1 = Campos Obligatorios para entrar)
  <br><br><input type="hidden" name="oculto" value="login" >
  <input type="submit" value="Entrar">
</form>
<jsp:include page="lineaerror.jsp"></jsp:include>
</body>
</html>
```

ver_jugadores.jsp



Apodos:

aaaa
klamus
Ruly
juan
Pepe
u
Aliena
a

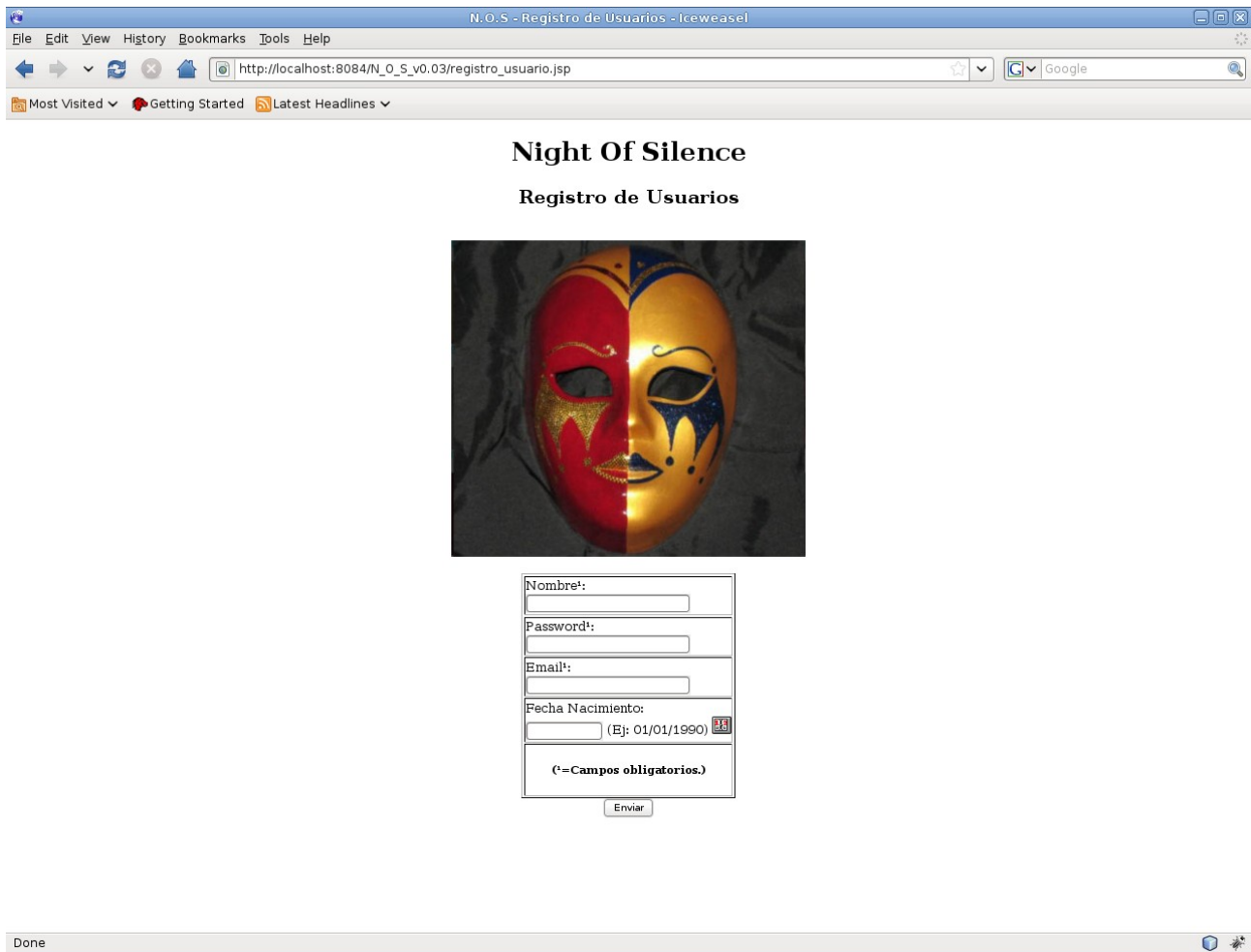
– ver_jugadores.jsp

Página que presenta una lista con los Apodos de los usuarios registrados que han creado personaje.


(Ejemplo de mi antigua metodología)

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">
<%@ page language = "java"%>
<%@ page import = "java.sql.Connection"%>
<%@ page import = "java.sql.DriverManager"%>
<%@ page import = "java.sql.ResultSet"%>
<%@ page import = "java.sql.Statement"%>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>JSP Page</title>
  </head>
  <body>
    <%
      //Conexión con Postgres
      Class.forName("org.postgresql.Driver");
      Connection conexion = DriverManager.getConnection("jdbc:postgresql://localhost:5432/postgres",
"postgres", "postgres");
      if (!conexion.isClosed()) {
        Statement st = conexion.createStatement();
        ResultSet rs = st.executeQuery("select \"Apodo\" from reg_users");
        String aux = "";
        out.println("<h3>Apodos:</h3><br>");
        while (rs.next()) {
          aux = rs.getString("Apodo");
          if(!aux.equals("0"))
            {
              out.println(aux + "<br>");
            }
        }
        conexion.close();
      }
    %>
  </body>
</html>
```

registro_usuario.jsp



The screenshot shows a web browser window with the following content:

- Browser title: N.O.S - Registro de Usuarios - Iceweasel
- Address bar: http://localhost:8084/N_O_S_v0.03/registro_usuario.jsp
- Page title: **Night Of Silence**
- Page subtitle: **Registro de Usuarios**
- Image: A colorful, stylized mask with red, yellow, and blue sections.
- Form fields:
 - Nombre*:
 - Password*:
 - Email*:
 - Fecha Nacimiento: (Ej: 01/01/1990) 
- Footer: (*=Campos obligatorios.)
- Button:

– registro_usuario.jsp

Presenta un formulario con los datos(Nombre*, Email*, Fecha_Nacimiento, Password*) a rellenar por el usuario para su registro.

Calendario desplegable. Clase date-picker.js.

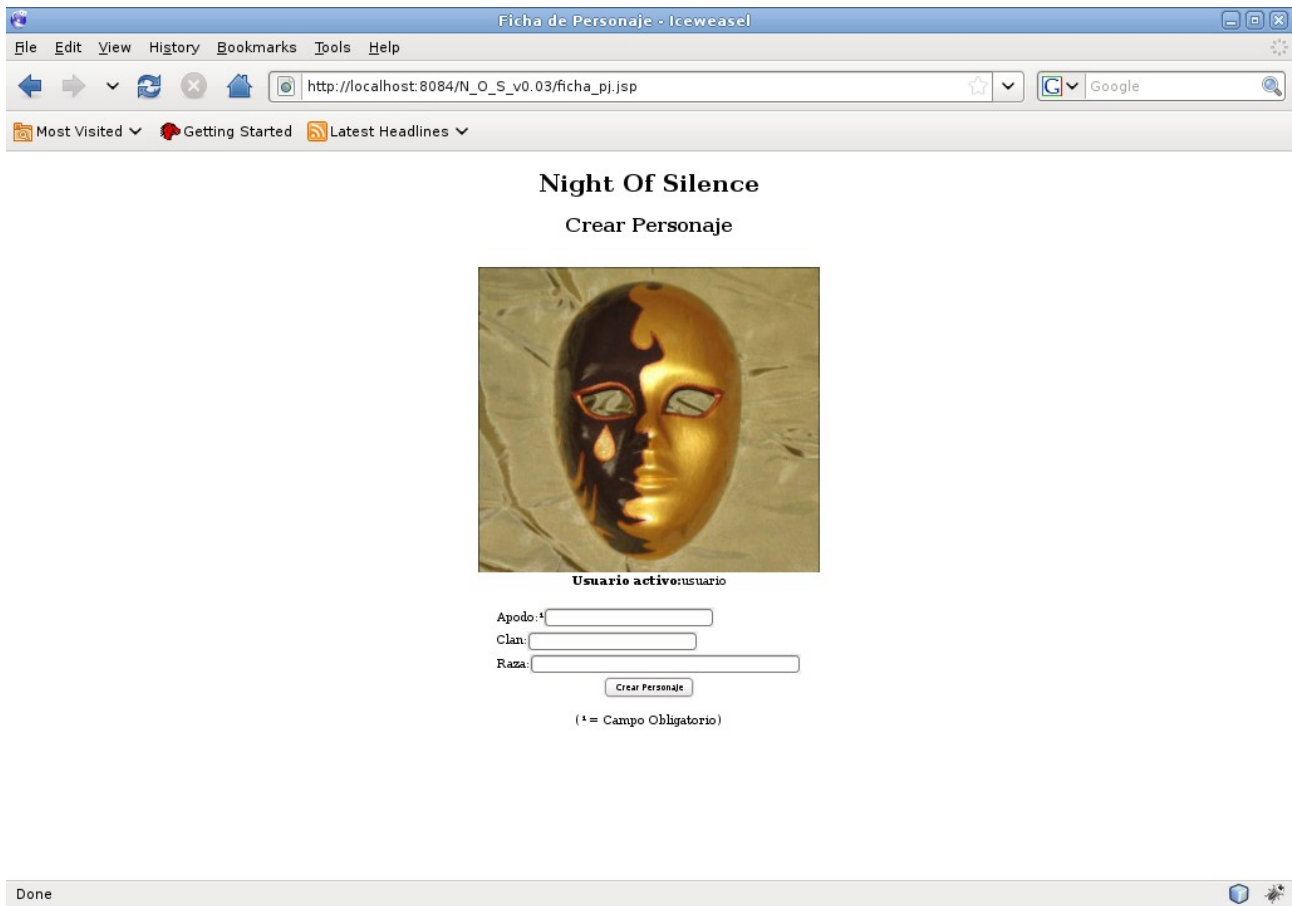
```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<%@page contentType="text/html; charset=UTF-8" language="java" import="java.util.*" %>
<%--
Document : index
Created on : 18-abr-2010, 16:23:45
Author : raul
--%>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>N.O.S - Registro de Usuarios</title>
<SCRIPT type="text/javascript" src="date-picker.js"> </SCRIPT>
</head>
<body>
<script type="text/javascript" language="javascript">
//Validamos el nombre
function validar2() {
//validar el nombre (No puede estar vacío)
if (document.form2.nombre.value.length==0){
alert("El campo nombre no puede estar vacío ...")
document.form2.nombre.focus()
return false;
}
//validar el password (No puede estar vacío)
if (document.form2.password.value.length==0){
alert("El campo nombre no puede estar vacío ...")
document.form2.password.focus()
return false;
}
//validar email (Tiene que tener una estructura dada)
var er = new RegExp(/^([A-Za-z0-9_\-\.]+@[A-Za-z0-9_\-\.]{2,}\.[A-Za-z0-9]{2,}(\.[A-Za-z0-9])?)/);
var valemil = document.form2.email.value;
//var er = /^[w+([\.-]?w+)*@w+([\.-]?w+)(\.[w]{2,3,4})+$/; (Otra expresion regular)
if (!er.test(valemil)){
alert("La dirección email es incorrecta.");
document.form2.email.focus();
return false;
}
}
```

```

//Controlados los errores enviamos el formulario.
document.form2.submit();
return true;
}
</script>
<center>
<h1>Night Of Silence</h1>
<h2>Registro de Usuarios</h2><br>
<br><br>
<FORM name="form2" ACTION="NOS" METHOD="Post" onSubmit="javascript:return validar2();">
  <TABLE border="1"
    <TR><TD>Nombre<sup>1</sup>;<br><INPUT TYPE="text" NAME="nombre" SIZE="25"></TD></TR>
    <TR><TD>Password<sup>1</sup>;<br><INPUT TYPE="password" NAME="password"
SIZE="25"></TD></TR>
    <TR><TD>Email<sup>1</sup>;<br><INPUT TYPE="text" NAME="email" SIZE="25"></TD></TR>
    <TR><TD>Fecha Nacimiento:<br><INPUT name="fecnac" size="10" (Ej: 01/01/1990)
    <A href="javascript:show_calendar('form2.fecnac',null,null,'DD/MM/YYYY');"
onmouseout="window.status="";return true;" onmouseover="window.status='Date Picker';return
true;">
      <IMG border=0 height=22 src="images0/show-calendar.gif" width=24
alt="Calendario"></A></TR>
    <TR><TD><center><h5>(<sup>1</sup>=Campos obligatorios.)</h5></center></TD></TR>
  </TABLE>
  <INPUT TYPE="submit" NAME="Enviar" VALUE="Enviar" >
  <input type="hidden" name="oculto" value="crearper">
</FORM>
</center>
</body>
</html>

```

ficha_pj.jsp



- ficha_pj.jsp

Presenta un formulario con los datos(Apodo*, Clan y Raza) a rellenar por el usuario para su registro.

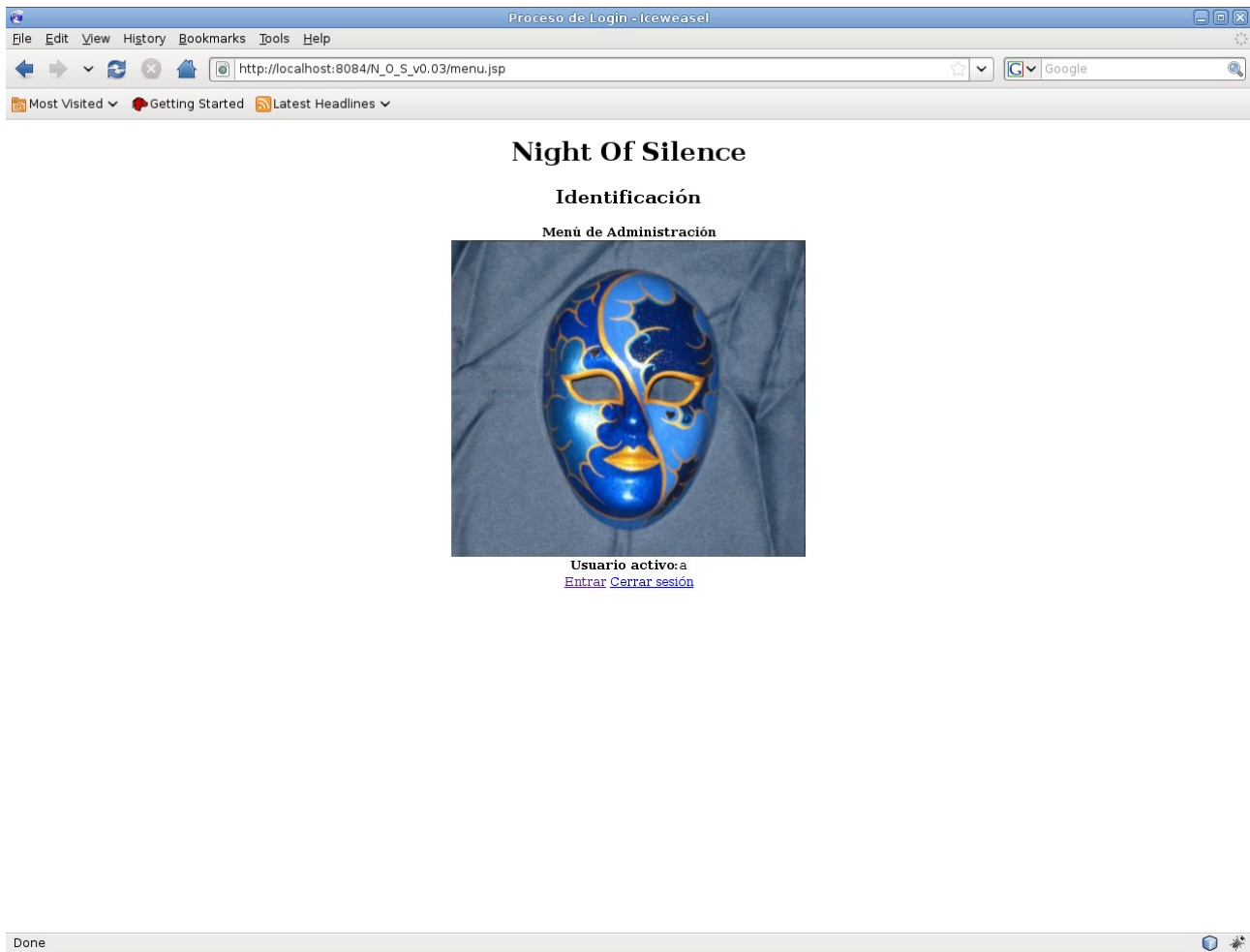
Solo el apodo es obligatorio.

```
<%@ page session="true" %>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Ficha de Personaje</title>
  </head>
  <body>
    <script type="text/javascript" language="javascript">
      //Validamos el apodo
      function validar() {
        //validar apodo (No puede ser blanco)
        if (document.form3.apodo.value.length==0){
          alert("El campo apodo no puede estar vac\u00edo ...")
          document.form3.apodo.focus()
          return false;
        }
        //validar raza (No puede ser blanco)
        if (document.form3.raza.value.length==0){
          alert("El campo raza no puede estar vac\u00edo ...")
          document.form3.raza.focus()
          return false;
        }
        //Controlados TODOS los errores enviamos el formulario.
        document.form3.submit();
        return true;
      }
    }
  </script>
  <% //CONTROL DE SESION -----
  String nombre = "";
  String password = "";
  String idsesion = "";
  String apodo = "";
  //Obtenemos la sesion (si existe)
  HttpSession sesionOK = request.getSession();
  idsesion = (String) sesionOK.getAttribute("idsesion");
  if (idsesion == null) {
```



```
//Enviamos inmediatamente a index.jsp al tramposo.
response.setHeader("Refresh", "0; URL=index.jsp?E=Es necesario identificarse...");
response.flushBuffer();
} else {
//Capturamos nombre, password y apodo de la sesi3n
nombre = (String) sesionOK.getAttribute("Nombre");
password = (String) sesionOK.getAttribute("Password");
apodo = (String) sesionOK.getAttribute("Apodo");
}
%>
<center>
<h1>Night Of Silence</h1>
<h2>Crear Personaje</h2><br>
 <br>
<% if (nombre == null) {out.println("<b>(Ning&uacute;n usuario activo)</b><BR>");}
else {out.println("<b>Usuario activo:</b>"+nombre+"<BR>");}
out.println("<b>Ident. sesion :</b>"+idsesion+"<BR>"); %>
<FORM name="form3" ACTION="NOS" METHOD="Post" onSubmit="javascript:return validar();" >
<TABLE>
<TR><TD>Apodo:&sup1<INPUT TYPE="text" NAME="apodo" SIZE="30"></TD></TR>
<TR><TD>Clan:<INPUT TYPE="text" NAME="clan" SIZE="30"></TD></TR>
<TR><TD>Raza:<INPUT TYPE="text" NAME="raza" SIZE="50"></TD></TR>
</TABLE>
<INPUT TYPE="submit" NAME="Enviar" VALUE="Crear Personaje">
<input type="hidden" name="oculto" value="modper">
</FORM>
<br>(&sup1 = Campo Obligatorio)<br><br>
<jsp:include page="lineaerror.jsp"></jsp:include>
</center>
</body>
</html>
```

menu.jsp



- menu.jsp

Comprueba que el usuario ha iniciado sesión y presenta un menu de opciones dinamico. Este menu inicialmente se compone de:

- Crear Personaje. Nos llevará a un formulario donde introduciremos los datos relativos al personaje. Actualmente solo son Apodo, Clan, Raza.
- Cerrar Sesión. Nos lleva a cerrarsesion.jsp. Este mata la sesión y nos envia al index.jsp.

Si el usuario que llega a menu.jsp ya tiene creado un personaje este transformará en:

- Entrar. Nos lleva al menu principal del juego, menu_juego.jsp.
- Cerrar Sesión.

```

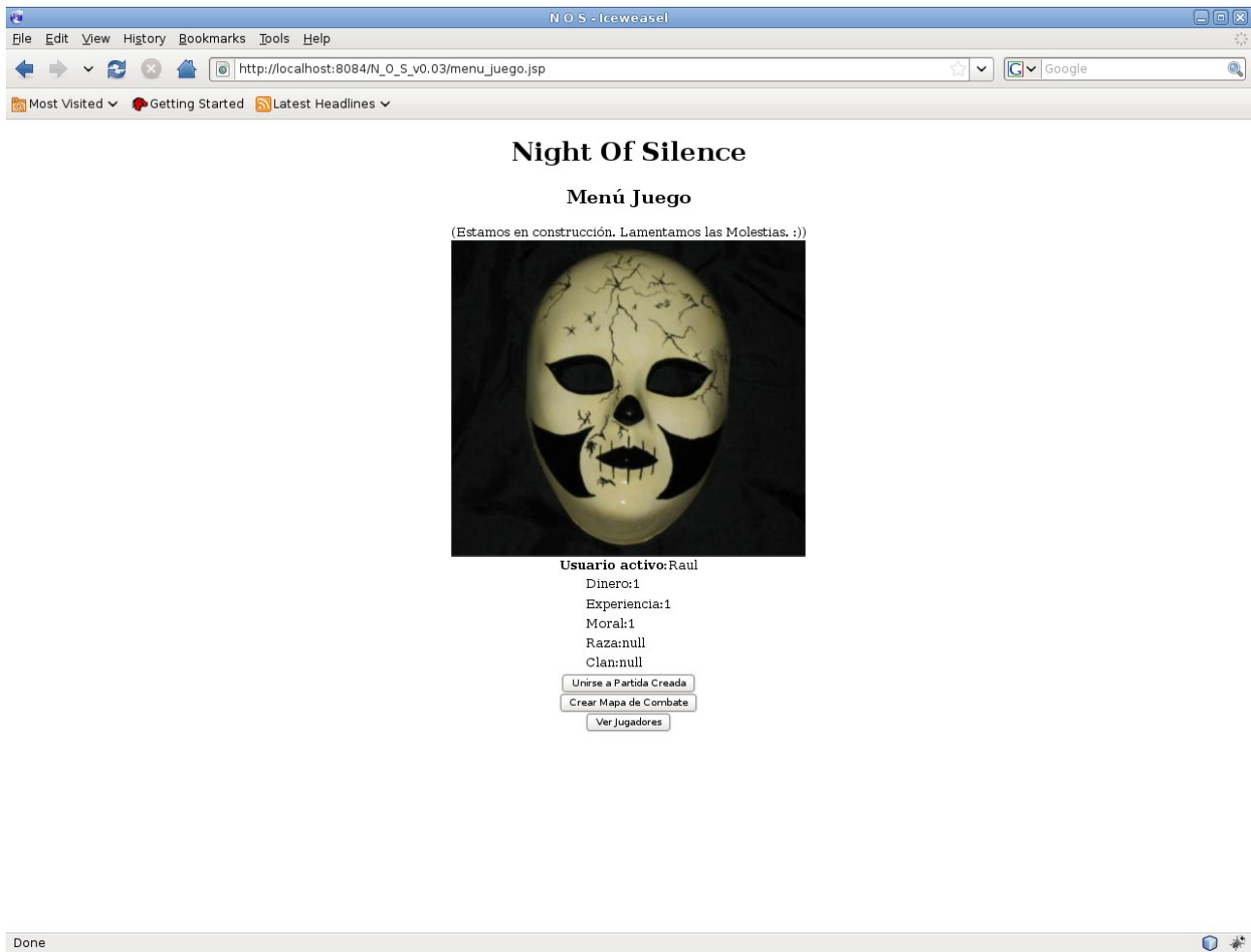
<%@page contentType="text/html; charset=UTF-8" session="true" language="java" import="java.util.*" %>
<html>
  <head><title>Proceso de Login</title>
  </head>
  <body>
    <center>
<%
  //COMPROBACION DE SESION -----
  String nombre = "";
  String password = "";
  String idsesion = "";
  String apodo = "";
  //Obtenemos la sesion (si existe)
  HttpSession sesionOK = request.getSession();
  idsesion = (String) sesionOK.getAttribute("idsesion");
  if (idsesion == null) {
    //Enviamos inmediatamente a index.jsp al tramposo.
    response.setHeader("Refresh", "0; URL=index.jsp?E=Es necesario identificarse...");
    response.flushBuffer();
  } else {
    //Capturamos nombre, password y apodo de la sesiÃ³n
    nombre = (String) sesionOK.getAttribute("Nombre");
    password = (String) sesionOK.getAttribute("Password");
    apodo = (String) sesionOK.getAttribute("Apodo");
  }
  //-----
%>
  <h1>Night Of Silence</h1>
  <h2>Identificaci&oacute;n</h2>
  <b>Men&uacute; de Administraci&oacute;n</b><br>
  <!-- IOS ERRORES APARENTES NO SON TALES. -->
  <br>
  <% if (nombre == null) {out.println("<b>(Ning&Aacute;n usuario activo)</b><BR>");}
  else {out.println("<b>Usuario activo:</b>"+nombre+"<BR>");}
  //out.println("<b>Ident. sesion :</b>"+idsesion+"<BR>");
  if (apodo.equals("0")) {%>

    <a href="ficha_pj.jsp">Crear Personaje</a>
  <%} else {%>
    <a href="menu_juego.jsp">Entrar</a>

```

```
<!--<a href="opc3.jsp">Borrar Personaje</a>-->
<% } %>
<!--<a href="opc4.jsp">Cambiar clave</a>-->
<a href="cerrarsesion.jsp">Cerrar sesi&oacute;n</a>
</center>
</body>
</html>
```

menu_juego.jsp



- menu_juego.jsp

En principio, solo presenta el Dinero, la Experiencia, la Moral del jugador (en un futuro *Random*), la Raza y el Clan al que pertenece (Más adelante permitirá ataques conjuntos entre jugadores).

Unirse a Partida Creada, botón que nos lleva al `join_battle`.

Crear Mapa de Combate, botón que nos lleva a `entrada_mapa`.

Ver Jugadores, botón que no lleva a `ver_jugadores.jsp`.

join_battle.jsp y entrada_mapa.jsp

JSP Page - Iceweasel

File Edit View History Bookmarks Tools Help

http://localhost:8084/N_0_S_v0.03/join_battle.jsp

Most Visited Getting Started Latest Headlines

Entrando a Mapa...

Nombre Partida:

Password:

(† = Campos Obligatorios para entrar)

Enviar

Done

- entrada_mapa.jsp y join_battle.jsp.

Son dos páginas que presentan el mismo formulario pero actúan de diferente forma.

- entrada_mapa.jsp nos solicita nombre y password para crear la partida.
- join_battle.jsp nos solicita el nombre y la password de la partida para comenzar a jugar.

join_battle.jsp

```
<%@page session = "true"%>
<%@page import="java.util.Random" %>
<%@page contentType="text/html" pageEncoding="UTF-8" language="java" import="java.util.*"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>JSP Page</title>
  </head>
  <body>
<%
  //COMPROBACION DE SESION -----
  String nombre = "";
  String password = "";
  String idsesion = "";
  String apodo = "";
  //Obtenemos la sesion (si existe)
  HttpSession sesionOK = request.getSession();
  idsesion = (String) sesionOK.getAttribute("idsesion");
  if (idsesion == null) {
    //Enviamos inmediatamente a index.jsp al trampos.
    response.setHeader("Refresh", "0; URL=index.jsp?E=Es necesario identificarse...");
    response.flushBuffer();
  } else {
    //Capturamos nombre, password y apodo de la sesiÃ³n
    nombre = (String) sesionOK.getAttribute("Nombre");
    password = (String) sesionOK.getAttribute("Password");
    apodo = (String) sesionOK.getAttribute("Apodo");
  }
  -----
  }
%>
<script type="text/javascript" language="javascript">
  //Validamos el nombre
  function mensaje() {alert(document.URL)}
```

```
//validar el nombre (No puede estar vacío)
if (document.form3.nom_partida.value.length==0){
    alert("El campo nombre no puede estar vacío ...")
    document.form3.nom_partida.focus();
    return false;
}
//validar el password (No puede estar vacío)
if (document.form3.pass_partida.value.length==0){
    alert("El campo password no puede estar vacío ...")
    document.form3.pass_partida.focus();
    return false;
}
//Controlados los errores enviamos el formulario.
document.form3.submit();
return true;
}
</script>
<h1>Entrando a Mapa...</h1>
<FORM name="form3" ACTION="guerra" METHOD="Post" onreset="mensaje()" onsubmit="javascript:return
validar3();">
    <TABLE>
        <TR><TD>Nombre Partida:<sup>1;<INPUT TYPE="text" NAME="nom_partida"
SIZE="40"></TD></TR>
        <TR><TD>Password:<sup>1;<INPUT TYPE="password" NAME="pass_partida"
SIZE="16"></TD></TR>
    </TABLE>
    <br>(<sup>1 = Campos Obligatorios para entrar)
    <br><input type="hidden" name="oculto" value="unirpartida">
    <br><INPUT TYPE="submit" NAME="Enviar" VALUE="Enviar" >
</FORM>
</body>
</html>
```


entrada_mapa.jsp

```
<%@page session = "true"%>
<%@page import="java.util.Random" %>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>JSP Page</title>
  </head>
  <body>
    <%
//COMPROBACION DE SESION -----
String nombre = "";
String password = "";
String idsesion = "";
String apodo = "";
//Obtenemos la sesion (si existe)
HttpSession sesionOK = request.getSession();
idsesion = (String) sesionOK.getAttribute("idsesion");
if (idsesion == null) {
    //Enviamos inmediatamente a index.jsp al trampos.
    response.setHeader("Refresh", "0; URL=index.jsp?E=Es necesario identificarse...");
    response.flushBuffer();
} else {
    //Capturamos nombre, password y apodo de la sesiÃ³n
    nombre = (String) sesionOK.getAttribute("Nombre");
    password = (String) sesionOK.getAttribute("Password");
    apodo = (String) sesionOK.getAttribute("Apodo");
}%>
<script type="text/javascript" language="javascript">
    //Validamos el nombre
    function validar2() {
        //validar el nombre (No puede estar vacio)
        if (document.form2.nom_partida.value.length==0){
            alert("El campo nombre no puede estar vac\u00edo ...")
            document.form2.nom_partida.focus();
        }
    }
</script>
    </body>
</html>
```

```
        return false;
    }
    //validar el password (No puede estar vacío)
    if (document.form2.pass_partida.value.length==0){
        alert("El campo password no puede estar vacío ...")
        document.form2.pass_partida.focus();
        return false;
    }
    //Controlados los errores enviamos el formulario.
    document.form2.submit();
    return false;
};
</script>
<h1>Entrando a Mapa...</h1>
<FORM name="form2" ACTION="guerra" METHOD="Post" onSubmit="javascript:return validar2();">
    <TABLE>
        <TR><TD>Nombre Partida<sup>1</sup>;<INPUT TYPE="text" NAME="nom_partida"
SIZE="40"></TD></TR>
        <TR><TD>Password<sup>1</sup>;<INPUT TYPE="text" NAME="pass_partida" SIZE="16"></TD></TR>
    </TABLE>
    <INPUT TYPE="submit" NAME="Enviar" VALUE="Enviar" >
    <input type="hidden" name="oculto" value="crearpartida">
</FORM>
</body>
</html>
```

PARTIDA EN JUEGO



– Dentro de NOS

Página generada por guerra.java dependiendo de los datos recogidos en cada momento. Cargar Mapa Partida.

Muestra la tabla de usuarios online junto a su salud.

Listbox que nos permite seleccionar a un jugador online y atacarle.

También podemos desconectarnos de la partida.

PARTIDA EN ESPERA DE TURNO



ESPERANDO TURNO

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" >
<title>PARTIDA</title>
</head>
<body onLoad="refresco()">
  <script type='text/javascript' language='javascript'>
    function refresco(){
      setTimeout("document.formMapa.submit();",1000);
    }
  </script>
  <form id='formMapa' name='formMapa' method='post' action='guerra' onSubmit="javascript:return refresco();">
    <input type='hidden' name='oculto' value='actualizar_mapa' >
  </form>
  <center><h1>ESPERANDO TURNO</h1></center>
</body>
</html>
```

Guión de Instalación de la Plataforma:

Comienzo instalando Debian Lenny y actualizando.

```
#aptitude update
```

```
#aptitude [upgrade|safe-upgrade|full-upgrade]
```

Elijo tomcat5.5 como motor servlet y jsp para albergar la aplicación.

```
#aptitude install tomcat5.5
```

*Tomcat parece que necesita una version actualizada de java jdk.

```
#aptitude search java
```

```
#aptitude install sun-java6-jdk
```

*Al hacer esto deberá corregirse el fallo, sino...

```
#update-alternatives --config java
```

*Seleccionamos la opción deseada.

Compruebo que el servidor está corriendo.

```
#!/etc/init.d/tomcat5.5 status
```

```
#netstat -putan
```

*Tomcat empieza a escuchar por defecto en los puertos 8009 y 8180.

*Puedo ver donde se ha almacenado la instalación de tomcat.

```
#dpkg -L tomcat5.5
```

Descargo Netbeans6.8 y lo instalo.

*Damos permiso de ejecución al script de instalación.

```
#chmod +x netbeans-6.8-ml-linux.sh
```

```
#!/netbeans-6.8-ml-linux.sh
```

*Como motor en tiempos de ejecución en Netbeans, elijo tomcat.

Como base de datos elijo finalmente PostgreSQL ya que es libre y no tendre problema de licencias.

```
#aptitude search postgresql
```

```
#aptitude install postgresql
```

Asigno password al usuario postgres.

```
#passwd postgres
```

Me logeo con el usuario postgres y ejecuto:

```
$su postgres
```

```
postgres$psql
```

Como no dispongo de mucho tiempo optaré por instalar una herramienta grafica que me facilite el trabajo con la BD.

```
#aptitude install pgadmin3
```

Para conectarnos a la BD desde pgadmin3 hay que modificar la password de postgres desde psql.

```
postgres=#alter user postgres with password 'postgres';
```

Una vez hecho esto podremos conectarnos sin problemas a la BD desde pgadmin.

Una vez dentro voy a crear 2 tablas para la v.0.01 de la aplicación.

*Voy a crear:

- “reg_users”, para el registro de los usuarios.
- “mapa”, para guardar el mapa de batalla.

Despliegue en Tomcat5.5:

Antes de desplegar podría configurar tomcat para que sea el que controle las conexiones con la BD, así, la aplicación podría funcionar en diversas plataformas. Pero haciendo pruebas e investigando un poco, parece que hay un pequeño bug(en Ubuntu y Debian)a la hora de otorgar permisos a los servlets de la aplicación para acceder a la BD.**Bug #234127 Ubutnu.**

Java.security.AccessControlException...

¡Da igual!, tengo un servlet que ya controla las conexiones, lo aprovecharé. Para poder utilizarlo necesito desactivar el uso de java security manager en tomcat. Esto se hace:

- `#nano /etc/init.d/tomcat5.5`
- `TOMCAT5_SECURITY=no`

Una vez hecho esto cojo el N_O_S_v0.03.war de la carpeta dist del proyecto en netbeans y lo muevo a la carpeta donde se despliengan las aplicaciones en tomcat.

```
#cp /home/usuario/NetBeansProjects/N_O_S_v0.03/dist/ N_O_S_v0.03.war  
/usr/share/tomcat5.5/webapps
```

Automáticamente se despliega al arrancar tomcat..

Conclusión:

Para construir la aplicación he hecho uso de:

- Netbeans 6.8.
- PostgreSQL 8.3.
- Kenai.
- Artisteer.
- JSP y Servlets.

Bibliografía:

<http://chuwiki.chuidiang.org/>

<http://www.esdebian.org/>

<https://bugs.launchpad.net/ubuntu/+source/tomcat5.5/+bug/234127>