Lenguaje de Programación
Java - Trabajo Práctico 2020

CASINO “ASAN”

ENTREGA INICIAL

Ingeniería en Sistemas de Información

COMISIÓN ELECTIVA 306

44853 - Córdoba, Agustín Leonel

45131 - López, Augusto Amadeo

Índice:

1. CU resumen reestructurado - Jugar a la Ruleta

Nivel	*Estructura*	*Alcance*	*Caja*	*Instanciación*	*Interacción*
Resumen	Reestructurado	Sistema	Negra	Real	Semántica

Meta: Jugar a la ruleta.

Actor primario: Usuario.

Disparador: El usuario ingresa al casino.

Camino básico:

1. Usuario ingresa sus datos para crear una cuenta en el casino. Sistema valida datos y crea la cuenta. **Invocando <CUU01 Crear usuario>.**
2. Usuario se logeo en el sistema con sus credenciales. Sistema autentifica y concede acceso. **invocando <CUU02 Ingresar al sistema>.**
3. Usuario selecciona juego de la ruleta. Sistema concede acceso a la sala. **Invocando <CUU03 Entrar a la sala>.**
4. Usuario realiza apuesta en la ruleta. Sistema verifica apuesta, corre la ruleta y efectúa ganancias/pérdidas. **Invocando <CUU04 Realizar apuesta en ruleta>.**
   El paso 4 se repite hasta que el usuario proceda a paso 5.
5. Usuario se retira de la sala. Sistema acredita ganancias/pérdidas y redirige a la pantalla principal. **<CUU05 Salir de la sala>.**

Camino alternativo:

1.a <durante> El usuario no es mayor de edad (18 años):
    1.a.1 El sistema informa política del casino. Fin del caso de uso.
1.b <durante> Las contraseñas no coinciden:
    1.b.1 El sistema informa error. Vuelve al paso 1.
1.c <posterior> El mail ya está en uso:
    1.c.1 El sistema redirige a la pantalla de registro. Vuelve al paso 1.
2.a <posterior> Las credenciales del usuario son incorrectas:
    2.a.1 El sistema redirige a la pantalla de bienvenida. Vuelve al paso 2.
4.a <anterior> La apuesta que realiza el usuario excede su dinero:
    4.a.1 El sistema informa error. Vuelve al paso 4.

Post-condiciones:

- Éxito: Usuario juega a la ruleta.
- Éxito alternativo: Usuario crea credenciales de acceso al sistema.
- Fracaso: Usuario no juega a la ruleta.

2. CUU principales

- ❏ Ingresar al sistema.
- ❏ Crear usuario.
- ❏ Acreditar dinero.
- ❏ Retirar dinero.
- ❏ Entrar a la sala.
- ❏ Salir de la sala.
- ❏ Realizar apuesta en ruleta.
- ❏ Realizar apuesta en quiniela.
- ❏ Realizar apuesta en blackjack.
- ❏ Consultar movimientos.
- ❏ Consultar tops 10.
- ❏ Crear administrador.
- ❏ Eliminar usuario.
- ❏ Consultar usuarios.
- ❏ Cambiar rol de usuario.

3. Capturas de pantalla

Se pueden consultar en carpeta adjunta: “Screenshots”.

4. Diagrama de clase y modelo de datos

Se pueden consultar en el archivo “diagrama.pdf” adjunto en la misma carpeta de este informe.

5. Fragmentos de código - CU Jugar a la Ruleta

Tomando en cuenta, en este caso, como precondición que el usuario ya posee una cuenta válida en el sistema con dinero acreditado, el próximo paso es hacer eso de sus credenciales para acceder al mismo. Como se aprecia en la figura a continuación, el HTML de bienvenida (que se puede ver en Screenshots/home.png) hace uso del nombre de usuario y contraseña que se ingresan para autenticarlos luego en el servlet “signin”.

```
<form class="form-signin" action="signin" method="post">
    <p>Usuario</p>
    <input type="text" name="usuario" placeholder="Ingrese su usuario" required>
    <p>Contraseña</p>
    <input type="password" name="password" placeholder="Ingrese su contraseña" required>
    <input type="submit" value="Entrar">
    <a href="registro.html">¿No tienes cuenta?</a>
    <p style="margin-top:20px; line-height:1"> <small>©Casino ASAN by Córdoba-López. Todo
</form>
```

En el servlet, se efectúa el traspaso de dichos parámetros a la entidad “usuario” la cual se encuentra preparada para contener datos de usuario y hace la validación, que veremos a continuación, para luego asignar el usuario a la sesión y concederle acceso al casino.

```
User ctrl=new User();
String nombre_usuario=request.getParameter("usuario");
String password=request.getParameter("password");

Usuario u = new Usuario();

u.setNombre_usuario(nombre_usuario);
u.setPassword(password);

u = ctrl.validate(u);

if (u==null) {
    response.sendRedirect("index.html");
} else {
    if(u.getRol()==2) {
        request.getSession().setAttribute("usuario", u);
        request.getRequestDispatcher("WEB-INF/casino.jsp").forward(request, response);
    } else {
        request.getSession().setAttribute("usuario", u);
        request.getRequestDispatcher("WEB-INF/admin/admin.jsp").forward(request, response);
    }
}
```

La función “validate”, haciendo uso de la estructuración en capas, llama posteriormente a la función “getUsuario” la cual ejecuta una query sobre la base de datos asignada previamente. La misma se puede ver a continuación, y si es exitosa devuelve todos los datos correspondientes al usuario para asignarlo a la sesión. Posteriormente, el servlet verifica si la cuenta pertenece a un administrador y efectúa la redirección correspondiente.

```
public Usuario getUsuario(Usuario user) {
    Usuario u=null;
    PreparedStatement stmt=null;
    ResultSet rs=null;
    try {
        stmt=Conexion.getInstancia().getConn().prepareStatement(
                "select id,nombre,apellido,email,genero,dinero,rol from usuarios where usuario=? "
                + "and contraseña=?"
                );
        stmt.setString(1, user.getNombre_usuario());
        stmt.setString(2, user.getPassword());
        rs=stmt.executeQuery();
        if(rs!=null && rs.next()) {
            u=new Usuario();
```

Suponiendo que la cuenta no es de administrador, el usuario accede a la UI principal del casino (Screenshots/casino.png).

Una vez dentro, para jugar a la ruleta, el usuario solamente debe hacer uso del botón “jugar” que se encuentra adyacente a la descripción del juego.

```
<div class="row" style="margin-top:10px; margin-bottom:10px">
    <div class="col-md-3">
        <img src="images/ruleta.jpg" width="200px" height="150px" style="border-radius:25px">
    </div>
    <div class="col-md-7" style="color:gold; text-align: justify" >
            <p><small>La ruleta se juega en la mesa de la ruleta, en donde se enumeran todos los resultados
            posibles y en donde los jugadores ponen sus apuestas en función del resultado que predicen.
            El crupier hace girar la rueda de la ruleta y arroja la bola de la ruleta en la rueda. La bola
            finalmente caerá en un número y se darán a conocer a los ganadores de esa ronda particular de la
            ruleta. La ruleta se juega en contra del casino y no en contra de otros jugadores. El juego
            ofrece diferentes opciones de apuestas. Prueba tus estrategias aquí: ASAN'S ROULETTE.</small></p>
     </div>
    <div class="cold-md-4" style="margin:auto">
    <form action="ruleta" method="get">
        <input type="hidden" name="id" value=<%=u.getId()%>>
        <input type="hidden" name="dinero" value=<%=u.getDinero()%>>
        <button type="submit" class="btn btn-success">JUGAR</button>
    </form>
    </div>
</div>
```

El botón hace referencia a un formulario el cual sirve para, una vez presionado el botón, llamar al servlet “ruleta” (en su método get) junto con los atributos que va a necesitar el juego y que se encuentran escondidos para la vista del usuario.

Una vez dentro de la sala (Screenshot/roulette.png), el usuario puede hacer uso de su dinero para efectuar distintas apuestas sin exceder el mismo. Si lo hace, el sistema mostrará un mensaje de error en javascript:

```
if (apuesta><%=session.getAttribute("dinero")%>){
    alert("No puedes apostar más dinero del que tienes");
    return false;
}
```

Si de lo contrario, la apuesta es válida, presionando el botón “tirar” se asignará automáticamente un resultado y se comparará en código javascript las ganancias o pérdidas correspondiente que serán computadas en la cuenta.

```
if(ganancia === 0){
    let h2 = document.getElementById('h2');
    apuesta = apuesta.toString();
    h2.innerHTML="PERDISTE $" + apuesta;
    h2.removeAttribute("hidden");
    apuesta = "-" + apuesta;
    document.getElementById("ganancia_0").value = apuesta; //botón jugar de nuevo
    document.getElementById("ganancia_1").value = apuesta; //botón volver
} else{
    let h2 = document.getElementById('h2');
    ganancia = ganancia.toString();
    h2.innerHTML="GANASTE $" + ganancia;
    h2.removeAttribute("hidden");
    document.getElementById("ganancia_0").value = ganancia; //botón jugar de nuevo
    document.getElementById("ganancia_1").value = ganancia; //botón volver
}
document.getElementById("again").removeAttribute("hidden");
document.getElementById('volver').removeAttribute("hidden");
document.getElementById("volver_menu").disabled = true;
return true;
```

Si la ganancia es nula, la apuesta que el usuario efectuó se convierte en pérdida y se le asigna a los formularios correspondientes a los botones “volver” y “jugar de nuevo”. Para ello se debe convertir la variable que se viene trabajando como numérica a string. Además, se despliegan los mensajes de ganancia o pérdida y se muestran dichos botones para poder continuar.

```
<form action="ruleta" method="post">
        <input type="hidden" name="id" id="id" value=<%=session.getAttribute("id")%>>
        <input type="hidden" name="ganancia" id="ganancia_0" value="">
        <input type="hidden" name="tipo" id="tipo" value="0">
        <button hidden type="submit" id="again" class="btn btn-primary" style="margin-top:20px">JUGAR DE NUEVO</button>
</form>
```

El “tipo” hace referencia al botón que se presiona para poder diferenciarlo en el mismo servlet “ruleta” sin la necesidad de crear otro.

En el servlet, primeramente se agrega a la tabla de historiales, el resultado de la mano junto con el número “3” que hace referencia al juego de la ruleta (“1” y “2” para los demás). Luego, se procede a recargar (positiva o negativamente) el dinero a la cuenta del usuario. Esto último implica acreditar o descontar de la base de datos el monto y actualizar el usuario en el sistema a los nuevos valores.

Por último, dependiendo del tipo de botón que se precede, el sistema redirige hacia la pantalla deseada por el usuario. Esto es, una nueva sala de ruleta o la pantalla principal del casino.

En las imágen a continuación se puede ver el proceso implementado en el servlet ruleta (en su método post):

```
User ctrl=new User();
Historial hist = new Historial();
MovimientosDinero rcg=new MovimientosDinero();
Integer id = Integer.parseInt(request.getParameter("id"));
Integer tipo = Integer.parseInt(request.getParameter("tipo"));
Float ganancia_f = Float.parseFloat(request.getParameter("ganancia"));
Integer ganancia = Math.round(ganancia_f);

Usuario u = new Usuario();
u.setId(id);
u.setDinero(ganancia);

rcg.recargar(u);
hist.add(u, 3);

Usuario nuevo = new Usuario();
nuevo.setId(id);
nuevo = ctrl.getById(nuevo);

if(tipo==0) {
    request.getSession().setAttribute("id", id);
```

Para la parte de acreditar o descontar se procede con la siguiente query (previamente pasando por el sistema de capas):

```
public void recargar(Usuario u){
    PreparedStatement stmt= null;
    try {
        stmt=Conexion.getInstancia().getConn().prepareStatement(
        "update usuarios set dinero=dinero+? where id=?");
        stmt.setInt(1, u.getDinero());
        stmt.setInt(2, u.getId());
        stmt.executeUpdate();
    }  catch (SQLException e) {
        e.printStackTrace();
```

Y luego, para computar los historiales (teniendo en cuenta que al atributo “dinero” de la entidad “usuario” se rellena con la ganancias/pérdida del juego, tal como se puede ver en el servlet):

```
public void add_historial(Usuario u, Integer juego) {
    PreparedStatement stmt= null;
    try {
        stmt=Conexion.getInstancia().getConn().prepareStatement(
        "insert into historial (id_usuario, ganancia, id_juego) values (?,?,?)");
        stmt.setInt(1, u.getId());
        stmt.setInt(2, u.getDinero());
        stmt.setInt(3, juego);
        stmt.executeUpdate();
    } catch (SQLException e) {
```

Una vez finalizado el caso de uso, el usuario puede seguir utilizando el casino a su gusto.