

PRUEBAS UNITARIAS

EXPLOTACIÓN DE IKER



ORDEN DE EJECUCIÓN DE CADA PRUEBA UNITARIA

Probar a ejecutar todo el código primero. En caso de que el test ***testActualizarUpdate*** de error, debe comentar el test ***testBorrarPorDni***.

EN CASO DE QUE OCURRA OTRO ERROR

Nos situamos en la clase ***tJugadoresTest*** y ejecutamos los siguientes test en este orden:

- 1. TEST DE INSERTAR***
- 2. TEST DE SELECCIONAR POR DNI***
- 3. TEST PARA BORRAR***
- 4. TEST PARA ACTUALIZAR***

PRUEBA UNITARIA ABRIR CONEXIÓN DE BASE DE DATOS E INSERTAR JUGADOR

Hemos realizado esta prueba para comprobar la conexión de la Base de Datos y para realizar la prueba unitaria de insertar jugadores. En este caso, como podemos ver en la captura del código le hemos dado unos valores válidos y hemos podido insertar el jugador.

Consulta SQL utilizada y valor que nos devuelve la función:

```
2 usages
public static int insert(Jugador jugador) throws Exception{
    System.out.println("Comienzo");
    BaseDato.abrirConexion();
    PreparedStatement ps = BaseDato.getCon().prepareStatement( sql: "insert into jugadores(nombre, apellido, dni, posicion, tipo_jugador) values (?, ?, ?, ?, ?)");
    ps.setString( parameterIndex: 1, jugador.getNombre());
    ps.setString( parameterIndex: 2, jugador.getApellido());
    ps.setString( parameterIndex: 3, jugador.getDNI());
    ps.setString( parameterIndex: 4, jugador.getPosicion().toString());
    ps.setString( parameterIndex: 5, jugador.getTipoJugador().toString());
    System.out.println("Insertado");
    int n = ps.executeUpdate();
    System.out.println(n);

    BaseDato.cerrarConexion();

    return n;
}
```

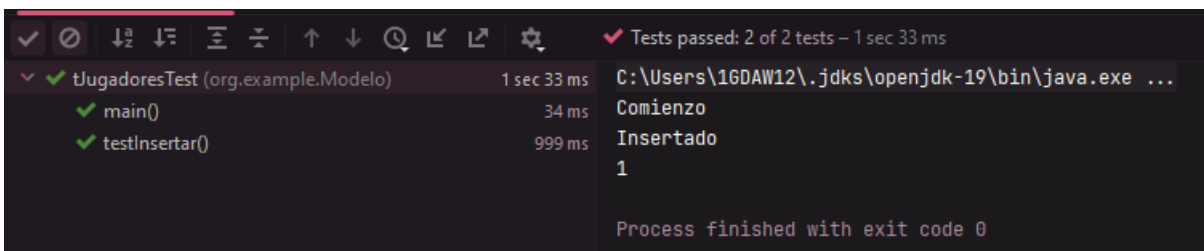
Test y resultado esperado:

```
1 usage
Jugador jugador = new Jugador( nombre: "koldo", apellido: "san juan",
    DNI: "87654321Z", Jugador.tPosicion.MC, Jugador.tTipoJugador.HABITUAL);

@Test
void testInsertar() throws Exception {

    assertEquals( expected: 1, tJugadores.insert(jugador));
}
```

Resultado:



✓ Tests passed: 2 of 2 tests – 1 sec 33 ms

Test	Duration	Output
tJugadoresTest (org.example.Modelo)	1 sec 33 ms	C:\Users\16DAW12\.jdk\openjdk-19\bin\java.exe ...
main()	34 ms	Comienzo
testInsertar()	999 ms	Insertado 1

Process finished with exit code 0

121	138	Juan	Arriba	12345678B	DC	HABITUAL
122	139	iKER	MACIAS	22222222C	MC	HABITUAL
123	142	koldo	san juan	87654321Z	MC	HABITUAL

PRUEBA UNITARIA DE VISUALIZAR JUGADOR POR DNI

En esta prueba unitaria comprobamos la Select del DNI de un jugador ya insertado.

Consulta SQL utilizada y valor que nos devuelve la función:

```
1 usage
2 public static Jugador buscarPorDNI(Jugador jugador) throws Exception{
3     BaseData.abrirConexion();
4     PreparedStatement ps = BaseData.getConnection().prepareStatement("select * from jugadores where dni=?");
5     ps.setString(1, jugador.getDNI());
6     ResultSet result = ps.executeQuery();
7     if (result.next()){
8         Jugador tPosicion tPosicion = null;
9         switch (result.getString( columnLabel: "posicion")){
10             case "DC" → tPosicion = Jugador.tPosicion.DC;
11             case "DF" → tPosicion = Jugador.tPosicion.DF;
12             case "P" → tPosicion = Jugador.tPosicion.P;
13             case "MC" → tPosicion = Jugador.tPosicion.MC;
14         }
15         Jugador.tTipoJugador tTipoJugador = null;
16         switch (result.getString( columnLabel: "tipo_jugador")){
17             case "HABITUAL" → tTipoJugador = Jugador.tTipoJugador.HABITUAL;
18             case "WILDCARD" → tTipoJugador = Jugador.tTipoJugador.WILDCARD;
19         }
20         jugador = new Jugador(result.getInt( columnLabel: "id_jugador"), result.getString( columnLabel: "nombre"), result.getString( columnLabel: "apellido"), result.getString( columnLabel: "dni"), tPosicion, tTipoJugador);
21     }
22     BaseData.cerrarConexion();
23     return jugador;
24 }
```

Test y resultado esperado:

```
2 usages
3 Jugador jugadorDni = new Jugador( DNI: "87654321Z");
4 @Test
5 void testSeleccionarPorDni() throws Exception {
6     assertEquals( jugadorDni.getDNI(), tJugadores.buscarPorDNI(jugadorDni).getDNI());
7 }
8 }
```

Resultado:

✓	Tests passed: 2 of 2 tests – 1 sec 116 ms
✓	tJugadoresTest (org.example.Modelo) 1 sec 116 ms C:\Users\16DAW12\.jdk\openjdk-19\bin\java.exe ...
✓	main() 28 ms
✓	testSeleccionarPorDni() 1 sec 88 ms Process finished with exit code 0

ID_JUGADOR	NOMBRE	APELLIDO	DNI	POSICION	TIPO_JUGADOR
1	142	koldo	san juan	87654321Z MC	HABITUAL

PRUEBA UNITARIA DE BORRAR JUGADOR

Generamos el test de borrar un jugador.

Consulta SQL utilizada y valor que nos devuelve la función:

```
1 usage
public static int delete(Jugador jugador) throws Exception {
    BaseDato.abrirConexion();
    PreparedStatement ps = BaseDato.getCon().prepareStatement( sql: "delete from jugadores where dni = ?");
    ps.setString( parameterIndex: 1, jugador.getDNI());
    int n = ps.executeUpdate();
    BaseDato.cerrarConexion();
    return n;
}
```

Test y resultado esperado:

```
1 usage
Jugador jugadorBorrar = new Jugador( DNI: "876543212");
@Test
void testBorrarPorDni() throws Exception {
    assertEquals( expected: 1, tJugadores.delete(jugadorBorrar));
}
```

Resultado:

✓ Tests passed: 4 of 4 tests – 1 sec 296 ms		
✓ tJugadoresTest (org.example.Modelo)	1 sec 296 ms	C:\Users\16DAW12\.jdk\openjdk-19\bin\java.exe ...
✓ main()	25 ms	
✓ testBorrarPorDni()	1 sec 14 ms	Process finished with exit code 0
✓ testSeleccionarPorDni()	190 ms	
✓ testInsertar()	67 ms	

PRUEBA UNITARIA DE ACTUALIZAR JUGADOR

Actualizamos el jugador que hemos creado anteriormente. Como podemos observar el test de insertar nos da error, ya que el jugador ya estaba creado y para poder actualizarlo debemos comentar el test de borrado.

Consulta SQL utilizada y valor que nos devuelve la función:

```

1 usage
public static int update(Jugador jugador) throws Exception{
    BaseDatos.abrirConexion();
    PreparedStatement ps = BaseDatos.getCon().prepareStatement( sql: "update jugadores set nombre=?, " +
        "apellido=?, posicion=?, tipo_jugador=? where dni=?");
    ps.setString( parameterIndex: 1, jugador.getNombre());
    ps.setString( parameterIndex: 2, jugador.getApellido());
    ps.setString( parameterIndex: 3, jugador.getPosicion().toString());
    ps.setString( parameterIndex: 4, jugador.getTipoJugador().toString());
    ps.setString( parameterIndex: 5, jugador.getDNI());
    int n = ps.executeUpdate();
    BaseDatos.cerrarConexion();
    return n;
}

```

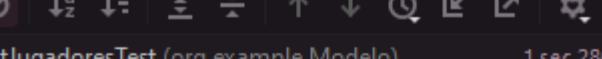
Test y resultado esperado:

```

1 usage
Jugador jugadorUpdate = new Jugador( nombre: "jose", apellido: "juan carlos", DNI: "87654321Z", Jugador.tPosicion.DC, Jugador.tTipoJugador.WILDCARD);
@Test
void testActualizarPorDNI() throws Exception {
    assertEquals( expected: 1, tJugadores.update(jugadorUpdate));
}

```

Resultado:



Run console output for `jugadoresTest (org.example.Modelo)`:

Test Method	Duration	Status
<code>main()</code>	25 ms	Passed
<code>testSeleccionarPorDni()</code>	1 sec 69 ms	Passed
<code>testInsertar()</code>	107 ms	Failed
<code>testActualizarPorDNI()</code>	79 ms	Passed