Documentación de Tests

Módulo `DbTest`

Contiene las pruebas implementadas en el módulo `DbTest` para verificar la funcionalidad de las operaciones d`e nuestra base de datos`.

Pruebas del método `Db.del_element`

1. **Eliminar el primer elemento**: Verifica que el método elimina correctamente un elemento ubicado al inicio de la lista.
2. **Clave no encontrada**: Comprueba que si la clave no existe en la lista, esta permanece sin cambios.
3. **Eliminar un elemento intermedio**: Asegura que se elimina correctamente un elemento ubicado en medio de la lista.
4. **Lista vacía como resultado**: Confirma que al eliminar el único elemento de la lista, esta queda vacía.

Pruebas del método `Db.find_element`

1. **Encontrar un elemento único**: Verifica que el método localiza correctamente un elemento en la lista cuando este existe.
2. **Encontrar un elemento en una lista más grande**: Asegura que el método identifica el elemento correcto en una lista con múltiples elementos.
3. **Elemento no encontrado**: Comprueba que el método retorna `{:not_found}` cuando el elemento no está en la lista.

Pruebas del método `Db.update`

1. **Actualizar un único elemento**: Asegura que un elemento puede ser actualizado correctamente con nuevos valores.
2. **Actualizar un elemento en una lista grande**: Verifica que el método actualiza correctamente un elemento específico en una lista con varios elementos.
3. **Actualizar en una lista vacía**: Confirma que si la lista está vacía, el método retorna una lista vacía sin errores.
4. **Actualizar elemento no encontrado**: Asegura que, si la clave no coincide con ningún elemento, la lista permanece sin cambios.

Módulo `EncriptadoTest`

Este documento describe las pruebas implementadas en el módulo `EncriptadoTest`, que verifican la funcionalidad de los métodos `encriptar/1` y `desencriptar/1` del módulo `Encriptado`.

Pruebas de los métodos `encriptar` y `desencriptar`

1. **Contraseña básica**: Comprueba que una contraseña simple como `"password"` se puede encriptar y luego desencriptar correctamente.
2. **Un solo carácter**: Verifica que una contraseña de un solo carácter (`"e"`) puede ser encriptada y desencriptada sin errores.
3. **Múltiples caracteres repetidos**: Asegura que contraseñas con varias repeticiones, como `"holaholaholas"`, son procesadas correctamente.
4. **Combinaciones complejas**: Comprueba que contraseñas que incluyen combinaciones de letras, números y símbolos especiales (`"as12*@3"`) se encriptan y desencriptan adecuadamente.

Módulo `PistasDataBaseTest`

Este fragmento describe las pruebas realizadas para validar las funcionalidades relacionadas con la gestión de pistas y usuarios en las tablas `TablaPistas` y `TablaUsuarios`.

Pruebas de gestión de pistas

1. **Inicializar pistas**: Verifica que, al inicializar las pistas, se cargan correctamente 20 pistas en el sistema.
2. **Reservar pista correctamente**: Asegura que un usuario registrado puede reservar una pista existente, y la reserva se registra correctamente.
3. **Reservar pista no existente**: Comprueba que intentar reservar una pista fuera del rango válido (por ejemplo, la número 25) retorna un error de tipo `:pista_no_existente`.
4. **Reservar pista con usuario no existente**: Verifica que no es posible realizar una reserva para un usuario que no está registrado en `TablaUsuarios`.
5. **Liberar pista correctamente**: Valida que un usuario puede liberar una pista que ha reservado previamente y que esta vuelve a estar disponible.
6. **Liberar pista de otro usuario**: Asegura que un usuario no puede liberar una pista reservada por otro usuario, retornando un error `:no_autorizado`.
7. **Liberar pista no existente**: Confirma que intentar liberar una pista que no existe retorna un error de tipo `:pista_no_existente`.
8. **Liberar pista no reservada**: Comprueba que, si un usuario intenta liberar una pista que no está reservada, se retorna un error de tipo `:pista_no_reservada`.

Pruebas de gestión de reservas por usuario

1. **Listar reservas de un usuario**: Verifica que las reservas realizadas por un usuario específico se listan correctamente, mostrando los números de las pistas reservadas y el identificador del usuario.

Módulo `ServidorAdminTest`

A continuación se describen las pruebas implementadas para verificar las funcionalidades del servidor de administración (`ServidorAdmin`) y su interacción con `ServidorCliente` y las tablas de usuarios y pistas.

Configuración inicial

- **Setup**: Antes de cada prueba:
  - Se inicializan las tablas `TablaUsuarios` y `TablaPistas`.
  - Se crean procesos independientes para `ServidorAdmin` y `ServidorCliente`.
  - Los `PID` de estos procesos se comparten en cada prueba mediante un mapa.

Pruebas del servidor de administración

1. **Registrar un usuario y verificar su existencia**:
   - **Objetivo**: Validar que un usuario registrado aparece correctamente en la tabla de usuarios.
   - **Proceso**:
     - Registrar el usuario con `ServidorAdmin.singUp`.
     - Comprobar la existencia del usuario con `DB.TablaUsuarios.existsUser`.
   - **Resultado esperado**: El usuario es registrado y detectado como existente.
2. **Borrar un usuario y verificar que no existe**:
   - **Objetivo**: Confirmar que un usuario eliminado ya no está en la tabla de usuarios.
   - **Proceso**:
     - Registrar y reservar una pista para el usuario.
     - Borrar al usuario con `ServidorAdmin.deleteUser`.
     - Comprobar su ausencia en la tabla de usuarios con `DB.TablaUsuarios.existsUser`.
   - **Resultado esperado**: El usuario se elimina correctamente.
3. **Comprobar contraseña de un usuario**:
   - **Objetivo**: Verificar la autenticación de un usuario mediante su contraseña.
   - **Proceso**:
     - Registrar un usuario.
     - Comprobar su contraseña con `ServidorAdmin.checkPassword`.
     - Validar tanto contraseñas correctas como incorrectas.
   - **Resultado esperado**: La contraseña válida se acepta; la inválida devuelve un error.

4. **Actualizar contraseña de un usuario**:

- **Objetivo**: Validar la capacidad de actualizar la contraseña de un usuario.
- **Proceso**:
    - Registrar un usuario.
    - Actualizar su contraseña con `ServidorAdmin.updatePassword`.
    - Verificar que la nueva contraseña funciona correctamente.
- **Resultado esperado**: La contraseña se actualiza y es aceptada en la autenticación.

5. **Ver todas las pistas**:
    - **Objetivo**: Asegurar que se puede consultar el estado actual de todas las pistas.
    - **Proceso**:
        - Registrar dos usuarios con `ServidorCliente.singUp`.
        - Reservar dos pistas para ellos con `ServidorCliente.reservar_pista`.
        - Llamar a `ServidorAdmin.seeAllPistas` y verificar el estado de todas las pistas.
    - **Resultado esperado**: Las pistas reservadas muestran el usuario correspondiente, y las demás están disponibles (`reservada_por: nil`).

Módulo `ServidorClienteTest`

Define una serie de pruebas unitarias para verificar el comportamiento del servidor ServidorCliente.

- **Registrar un usuario y verificar su existencia**: Verifica que un usuario puede registrarse correctamente y que su existencia puede ser comprobada en la base de datos.
- **Intentar registrar un usuario ya existente**: Prueba el caso donde se intenta registrar un usuario que ya existe en la base de datos.
- **Borrar un usuario y verificar que no existe**: Verifica que un usuario puede ser eliminado correctamente y que ya no existe en la base de datos.
- **Comprobar contraseña de un usuario**: Verifica si la contraseña de un usuario puede ser validada correctamente, tanto si es correcta como incorrecta.
- **Actualizar contraseña de un usuario**: Verifica que se puede actualizar la contraseña de un usuario y que la nueva contraseña es válida.
- **Reservar una pista correctamente**: Verifica que un usuario pueda reservar una pista correctamente.
- **Reservar una pista con un usuario no registrado**: Asegura que no se pueda reservar una pista si el usuario no está registrado.
- **Reservar una pista ya reservada**: Verifica que no se puede reservar una pista si ya está reservada por otro usuario.
- **Liberar una pista correctamente**: Verifica que un usuario pueda liberar correctamente una pista que ha reservado.
- **Liberar una pista no reservada**: Asegura que no se puede liberar una pista que no ha sido reservada.
- **Liberar una pista reservada por otro usuario**: Verifica que no se puede liberar una pista que ha sido reservada por otro usuario.
- **Ver reservas de un usuario**: Verifica que se pueda obtener la lista de reservas de un usuario (aunque esta prueba aún está comentada como un ejemplo).

Módulo UserDataBaseTest

Explicación de las pruebas:

1. **Añadir y comprobar usuario**:
    - Se prueba la adición de un usuario a la base de datos y se verifica si el usuario existe.
    - Se espera que al añadir un usuario con éxito, `TablaUsuarios.existsUser` devuelva `{:user_exists}`.
2. **Comprobar contraseña**:
    - Se añade un usuario y se verifican varias situaciones:
        - Si la contraseña es correcta (`{:ok, usuario}`).
        - Si la contraseña es incorrecta (`{:error, :invalid_password}`).
        - Si el usuario no existe (`{:error, :user_not_found}`).
3. **Actualizar contraseña**:
    - Se prueba la actualización de la contraseña de un usuario.
    - Se espera que el sistema devuelva `{:updated}` tras una actualización exitosa de la contraseña.
4. **Borrar usuario y verificar que no existe**:
    - Se elimina un usuario y luego se verifica que efectivamente no exista en la base de datos.
    - Se espera que después de la eliminación, `TablaUsuarios.existsUser` devuelva `{:user_does_not_exist}`.
5. **Ver usuarios**:
    - Se añade varios usuarios y luego se verifica si el listado completo de usuarios es correcto.
    - Se espera que la lista de usuarios esté ordenada, por lo que se comparan ambas listas ordenadas.

Casos sin cubrir

No se realizaron test específicos para las interfaces tanto de usuario como de administrador debido a que estas practicamente solo utilizan los métodos probados anteriormente y no añaden ninguna funcionalidad compleja a mayores que se deba probar.