Pruebas unitarias usando MsTest

Gestión de la configuración

Integrantes

- Valeska Sofia Chica Valfre
- Jorge Steven Gualpa Gia
- Geovanny Alexander Ochoa Gilces
- Jordy Alejandro Vilcacundo Chiluisa

2024

Información General

- Proyecto: Sistema de facturación UTEQ
- Fecha: 17/07/2024
- Autores: [Valeska Chica, Jorge Gualpa, Geovanny Ochoa, Jordy Vilcacundo]

1. Introducción

Objetivo: El objetivo principal de estas pruebas unitarias es verificar el correcto funcionamiento de las clases csCliente, csEmpleados, csProducto y csFactura en el sistema de facturación SGFactuacion. Estas pruebas buscan asegurar que las operaciones fundamentales como registro, edición, búsqueda y listado de clientes, empleados y productos, así como el proceso de facturación, funcionen correctamente y de manera consistente. Esto ayuda a garantizar la integridad y confiabilidad del sistema antes de su implementación o después de realizar cambios.

2. Herramientas y entorno

IDE utilizado: Visual Studio 2022 Community 17.10.4

Framework de pruebas: MsTest.TestFramework 2.2.10

Sistema Operativo: Windows 10 Pro

Lenguaje de programación: C#

3. Pruebas Unitarias

3.1.RegistrarCliente_ShouldReturnTrue_WhenRegistrationIsSuccessful

Descripción:

Esta prueba unitaria verifica que el método RegistrarCliente() de la clase csCliente funcione correctamente al registrar un nuevo cliente en el sistema. La prueba crea un nuevo objeto cliente con datos válidos y espera que el método de registro retorne true, indicando un registro exitoso.

Precondiciones:

- La clase csCliente está implementada y accesible.
- El método RegistrarCliente() está implementado en la clase csCliente.
- La base de datos o el sistema de almacenamiento está disponible y accesible.
- No existe un cliente con la cédula "2100562384" en el sistema.

Pasos:

- Arrange (Preparación):

  Crear un nuevo objeto csCliente con los siguientes datos:

  ID: 1

  Cédula: "2100562384"

  Nombre: "John"

  Apellido: "Doe"

  Fecha: Fecha y hora actual

Email: "john@example.com"

- Act (Acción):

  Llamar al método RegistrarCliente() del objeto cliente creado.

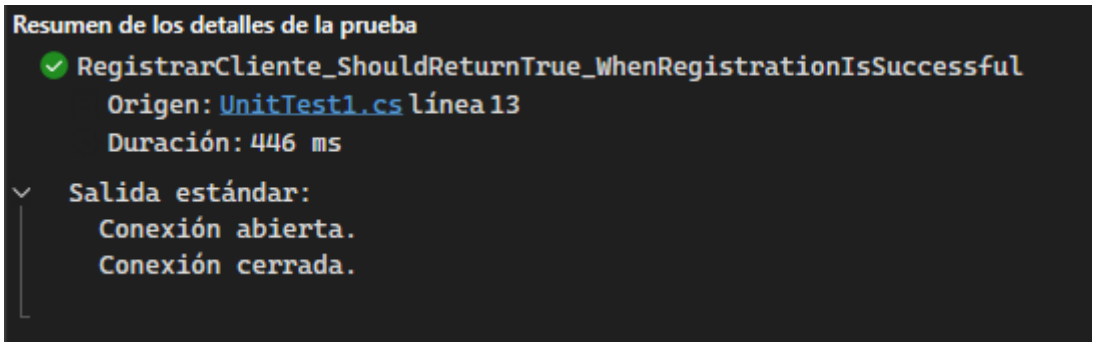
  Almacenar el resultado booleano en la variable 'registrado'.

- Assert (Afirmación):

  Verificar que el valor de 'registrado' sea true utilizando Assert.IsTrue().

Resultado esperado:

El método RegistrarCliente() debe retornar true, indicando que el cliente se registró exitosamente en el sistema.



Código:

```
public   void   RegistrarCliente_ShouldReturnTrue_WhenRegistrationIsSuccess-
ful()
{
    // Arrange
    var cliente = new csCliente(1, "2100562384", "John", "Doe", DateTime.Now,
"john@example.com");

    // Act
    bool registrado = cliente.RegistrarCliente();

    // Assert
    Assert.IsTrue(registrado);
}
```

3.2.RegistrarCliente_ShouldReturnFalse_OnDuplicateCedula

Descripción:

Esta prueba unitaria verifica que el método RegistrarCliente() de la clase csCliente retorne false cuando se intenta registrar un cliente con una cédula que ya existe en el sistema. La prueba simula un intento de registro de un cliente con una cédula duplicada y espera que el método rechace el registro.

Precondiciones:

- La clase csCliente está implementada y accesible.
- El método RegistrarCliente() está implementado en la clase csCliente.
- La base de datos o el sistema de almacenamiento está disponible y accesible.
- Ya existe un cliente en el sistema con la cédula "1234567890".

Pasos:

- Arrange (Preparación):

  Crear un nuevo objeto csCliente con los siguientes datos:

  ID: 1

  Cédula: "1234567890" (una cédula que se asume ya existe en el sistema)

  Nombre: "John"

  Apellido: "Doe"

  Fecha: Fecha y hora actual

  Email: "john@example.com"

- Act (Acción):

  Llamar al método RegistrarCliente() del objeto cliente creado.

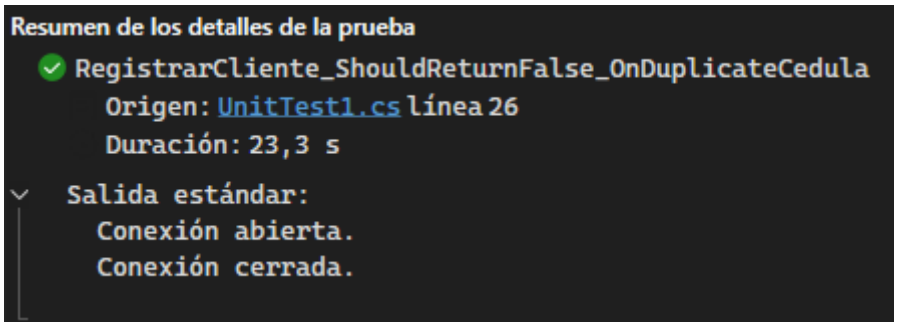
  Almacenar el resultado booleano en la variable 'registrado'.

- Assert (Afirmación):

  Verificar que el valor de 'registrado' sea false utilizando Assert.IsFalse().

Resultado esperado:

El método RegistrarCliente() debe retornar false, indicando que el cliente no se pudo registrar debido a que la cédula ya existe en el sistema.



Código:

```
public void RegistrarCliente_ShouldReturnFalse_OnDuplicateCedula()
    {
        // Arrange
         var cliente = new csCliente(1, "1234567899", "Jona", "Martin", Date-
Time.Now, "jasasahn@example.com");

        // Act
        bool registrado = cliente.RegistrarCliente();

        // Assert
        Assert.IsFalse(registrado);
    }
```

3.3.EditarCliente_ShouldReturnTrue_WhenEditIsSuccessful

Descripción:

Esta prueba unitaria verifica que el método EditarCliente() de la clase csCliente funcione correctamente al editar un cliente existente en el sistema. La prueba crea un objeto cliente con datos actualizados y espera que el método de edición retorne true, indicando una actualización exitosa.

Precondiciones:

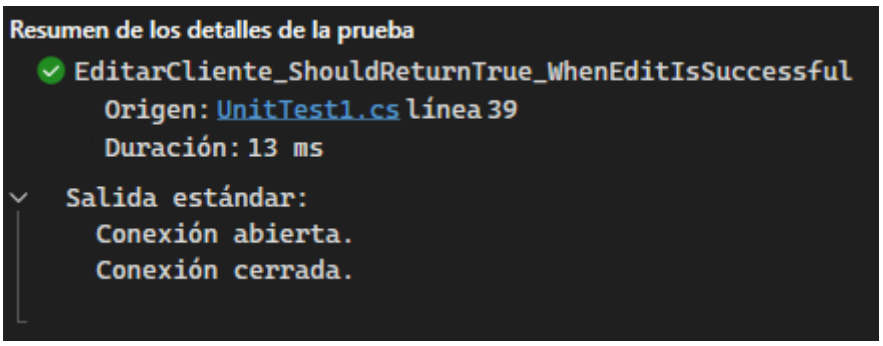
- La clase csCliente está implementada y accesible.
- El método EditarCliente() está implementado en la clase csCliente.
- La base de datos o el sistema de almacenamiento está disponible y accesible.
- Existe un cliente en el sistema con ID 1.

Pasos:

- Arrange (Preparación):
  Crear un objeto csCliente con los siguientes datos:
  ID: 1 (asumiendo que este cliente ya existe en el sistema)
  Cédula: "1234567890"
  Nombre: "John"
  Apellido: "Doe"
  Fecha: Fecha y hora actual
  Email: "john@example.com"
- Act (Acción):
  Llamar al método EditarCliente() del objeto cliente creado.
  Almacenar el resultado booleano en la variable 'editado'.
- Assert (Afirmación):
  Verificar que el valor de 'editado' sea true utilizando Assert.IsTrue().

Resultado esperado:

El método EditarCliente() debe retornar true, indicando que la información del cliente se actualizó exitosamente en el sistema.



Código:

```
public void EditarCliente_ShouldReturnTrue_WhenEditIsSuccessful()
{
    // Arrange
    var cliente = new csCliente(1, "1234567890", "John", "Doe", DateTime.Now,
"john@example.com");

    // Act
    bool editado = cliente.EditarCliente();

    // Assert
    Assert.IsTrue(editado);
}
```

3.4.ListarClientes_ShouldReturnListOfClients

Descripción:

Esta prueba unitaria verifica que el método estático ListarClientes() de la clase csCliente funcione correctamente al recuperar una lista de clientes del sistema. La prueba espera que el método devuelva una lista no nula y no vacía de objetos csCliente.

Precondiciones:

- La clase csCliente está implementada y accesible.
- El método estático ListarClientes() está implementado en la clase csCliente.
- La base de datos o el sistema de almacenamiento está disponible y accesible.
- Existen uno o más clientes registrados en el sistema.

Pasos:

- Act (Acción):

  Llamar al método estático ListarClientes() de la clase csCliente.

  Almacenar el resultado (una lista de objetos csCliente) en la variable 'clientes'.

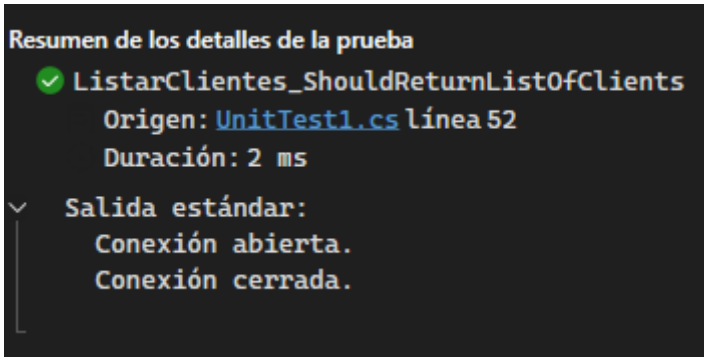
- Assert (Afirmación):

  Verificar que la lista 'clientes' no sea nula utilizando Assert.IsNotNull().

  Verificar que la lista 'clientes' contenga al menos un elemento utilizando Assert.IsTrue(clientes.Count > 0).

Resultado esperado:

El método ListarClientes() debe retornar una lista no nula que contenga al menos un objeto csCliente.



Código:

```
public void ListarClientes_ShouldReturnListOfClients()
{
    // Act
    List<csCliente> clientes = csCliente.ListarClientes();

    // Assert
    Assert.IsNotNull(clientes);
    Assert.IsTrue(clientes.Count > 0);
}
```

3.5.BuscarClientes_ShouldReturnMatchingClients

Descripción:

Esta prueba unitaria verifica que el método estático BuscarClientes() de la clase csCliente funcione correctamente al buscar clientes en el sistema basándose en un término de búsqueda. La prueba espera que el método devuelva una lista no nula y no vacía de objetos csCliente que coincidan con el criterio de búsqueda.

Precondiciones:

- La clase csCliente está implementada y accesible.
- El método estático BuscarClientes() está implementado en la clase csCliente.
- La base de datos o el sistema de almacenamiento está disponible y accesible.
- Existen uno o más clientes registrados en el sistema que coinciden con el término de búsqueda "John".

Pasos:

- Arrange (Preparación):

  Definir un término de búsqueda. En este caso, se usa "John".
- Act (Acción):

  Llamar al método estático BuscarClientes() de la clase csCliente, pasando el término de búsqueda "John".

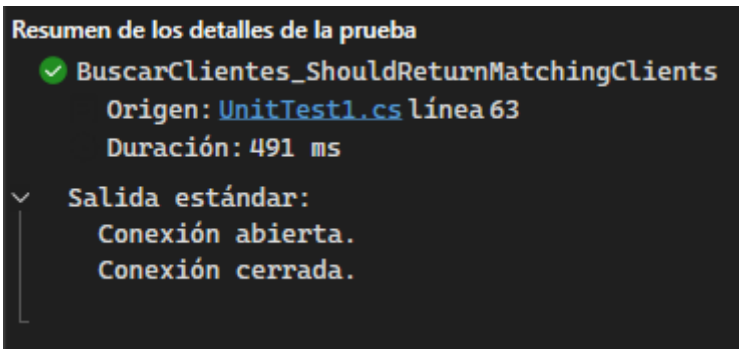
  Almacenar el resultado (una lista de objetos csCliente) en la variable 'clientes'.
- Assert (Afirmación):

  Verificar que la lista 'clientes' no sea nula utilizando Assert.IsNotNull().

  Verificar que la lista 'clientes' contenga al menos un elemento utilizando Assert.IsTrue(clientes.Count > 0).

Resultado esperado:

El método BuscarClientes() debe retornar una lista no nula que contenga al menos un objeto csCliente que coincida con el término de búsqueda "John".



Código:

```
public void BuscarClientes_ShouldReturnMatchingClients()
{
    // Arrange
    string busqueda = "John"; // Ejemplo de término de búsqueda

    // Act
     List<csCliente> clientes = csCliente.BuscarClientes(bus-
queda);

    // Assert
    Assert.IsNotNull(clientes);
    Assert.IsTrue(clientes.Count > 0);
}
```

3.6.GetFacturaIDByClienteIDAndFecha_ShouldReturnValidID

Descripción:

Esta prueba unitaria verifica que el método estático GetFacturaIDByClienteIDAndFecha() de la clase csCliente funcione correctamente al recuperar el ID de una factura basándose en el ID de un cliente y una fecha específica. La prueba espera que el método devuelva un ID de factura válido (mayor que cero) para los parámetros proporcionados.

Precondiciones:

- La clase csCliente está implementada y accesible.
- El método estático GetFacturaIDByClienteIDAndFecha() está implementado en la clase csCliente.
- La base de datos o el sistema de almacenamiento está disponible y accesible.
- Existe al menos una factura en el sistema para el cliente con ID 1 en la fecha actual.

Pasos:

- Arrange (Preparación):

  Definir un ID de cliente válido (en este caso, 1).

  Obtener la fecha actual (sin la parte de tiempo) usando DateTime.Now.Date.
- Act (Acción):

Llamar al método estático GetFacturaIDByClienteIDAndFecha() de la clase csCliente, pasando el ID del cliente y la fecha.

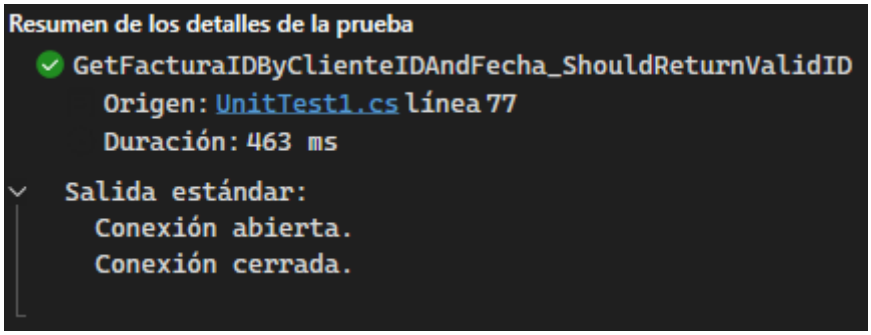
Almacenar el resultado (un long que representa el ID de la factura) en la variable 'idFactura'.

- Assert (Afirmación):

  Verificar que el valor de 'idFactura' sea mayor que cero utilizando Assert.IsTrue(idFactura > 0).

Resultado esperado:

El método GetFacturaIDByClienteIDAndFecha() debe retornar un ID de factura válido (un número entero mayor que cero) para el cliente y la fecha especificados.



Código:

```
public void GetFacturaIDByClienteIDAndFecha_ShouldReturnValidID()
{
    // Arrange
    long idCliente = 1; // ID de cliente válido
    DateTime fecha = DateTime.Now.Date; // Fecha válida

    // Act
    long idFactura = csCliente.GetFacturaIDByClienteIDAndFecha(idCliente, fecha);

    // Assert
    Assert.IsTrue(idFactura > 0); // Se espera que el ID de factura sea válido
}
```

3.7.RegistrarEmpleado_ShouldReturnTrue_WhenRegistrationIsSuccessful

Descripción:

Esta prueba unitaria verifica que el método RegistrarEmpleado() de la clase csEmpleados funcione correctamente al registrar un nuevo empleado en el sistema, incluyendo sus credenciales. La prueba espera que el método retorne true, indicando un registro exitoso.

Precondiciones:

- La clase csEmpleados y su clase interna csCredenciales están implementadas y accesibles.
- El método RegistrarEmpleado() está implementado en la clase csEmpleados.
- La base de datos o el sistema de almacenamiento está disponible y accesible.
- No existe un empleado con la cédula "1234567890" en el sistema.

- No existe un usuario con el nombre de usuario "johndoe" en el sistema.

Pasos:

- Arrange (Preparación):

  Crear un objeto csCredenciales con nombre de usuario "johndoe" y contraseña "password".

  Crear un objeto csEmpleados con los siguientes datos:

  ID: 1

  Cédula: "1234567890"

  Nombre: "John"

  Apellido: "Doe"

  Fecha: Fecha y hora actual

  Email: "john@example.com"

- Act (Acción):

  Llamar al método RegistrarEmpleado() del objeto empleado creado, pasando el objeto credenciales como parámetro.

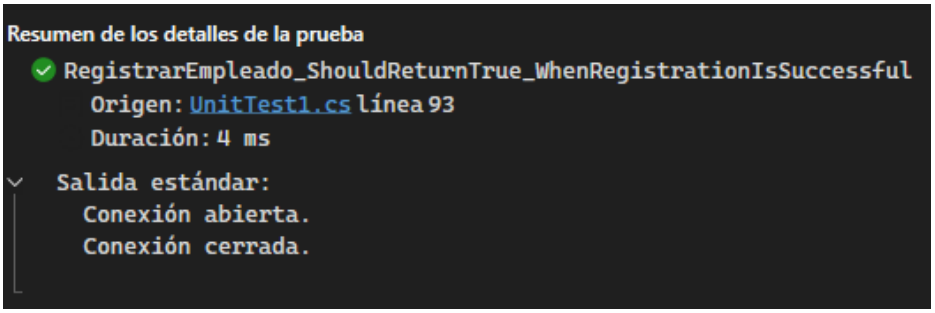
  Almacenar el resultado booleano en la variable 'registrado'.

- Assert (Afirmación):

  Verificar que el valor de 'registrado' sea true utilizando Assert.IsTrue().

Resultado esperado:

El método RegistrarEmpleado() debe retornar true, indicando que el empleado y sus credenciales se registraron exitosamente en el sistema.



Código:

```
public void RegistrarEmpleado_ShouldReturnTrue_WhenRegistrationIsSuccessful()
{
    // Arrange
    var credenciales = new csEmpleados.csCredenciales("johndoe", "password");
     var empleado = new csEmpleados(1, "1234567890", "John", "Doe", DateTime.Now,
"john@example.com");

    // Act
    bool registrado = empleado.RegistrarEmpleado(credenciales);

    // Assert
    Assert.IsTrue(registrado);
}
```

3.8.EditarEmpleado_ShouldReturnTrue_WhenEditIsSuccessful

Descripción:

Esta prueba unitaria verifica que el método EditarEmpleado() de la clase csEmpleados funcione correctamente al editar la información de un empleado existente en el sistema, incluyendo la actualización de sus credenciales. La prueba espera que el método retorne true, indicando una edición exitosa.

Precondiciones:

- La clase csEmpleados y su clase interna csCredenciales están implementadas y accesibles.
- El método EditarEmpleado() está implementado en la clase csEmpleados.
- La base de datos o el sistema de almacenamiento está disponible y accesible.
- Existe un empleado en el sistema con ID 1.
- Existe un usuario con el nombre de usuario "johndoe" en el sistema.

Pasos:

- Arrange (Preparación):

  Crear un objeto csCredenciales con nombre de usuario "johndoe" y nueva contraseña "newpassword".

  Crear un objeto csEmpleados con los siguientes datos actualizados:

  ID: 1 (asumiendo que este empleado ya existe en el sistema)

  Cédula: "1234567890"

  Nombre: "John"

  Apellido: "Doe"

  Fecha: Fecha y hora actual

  Email: "john@example.com"

- Act (Acción):

  Llamar al método EditarEmpleado() del objeto empleado creado, pasando el objeto credenciales como parámetro.

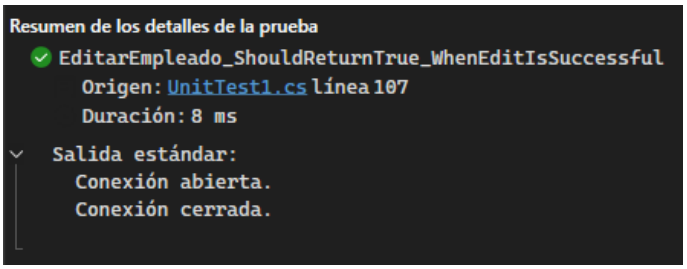
Almacenar el resultado booleano en la variable 'editado'.

- Assert (Afirmación):

  Verificar que el valor de 'editado' sea true utilizando Assert.IsTrue().

Resultado esperado:

El método EditarEmpleado() debe retornar true, indicando que la información del empleado y sus credenciales se actualizaron exitosamente en el sistema.



Código:

```
public void EditarEmpleado_ShouldReturnTrue_WhenEditIsSuccessful()
{
    // Arrange
    var credenciales = ne csEmpleados.csCredenciales("jp", "root");
   var empleado = new csEmpleados(1, "1234567890", "John", "Doe", DateTime.Now,
"john@example.com");

    // Act
    bool editado = empleado.EditarEmpleado(credenciales);

    // Assert
    Assert.IsTrue(editado);
}
```

3.9.IniciarSesion_ShouldReturnTrue_WhenCredentialsAreCorrect

Descripción:

Esta prueba unitaria verifica que el método IniciarSesion() de la clase csCredenciales (una clase interna de csEmpleados) funcione correctamente cuando se proporcionan credenciales válidas. La prueba espera que el método retorne true, indicando un inicio de sesión exitoso.

Precondiciones:

- La clase csEmpleados y su clase interna csCredenciales están implementadas y accesibles.
- El método IniciarSesion() está implementado en la clase csCredenciales.
- El sistema de autenticación está disponible y accesible.
- Existe un usuario en el sistema con nombre de usuario "johndoe" y contraseña "newpassword".

Pasos:

- Arrange (Preparación):

  Crear un objeto csCredenciales con nombre de usuario "johndoe" y contraseña "newpassword".
- Act (Acción):

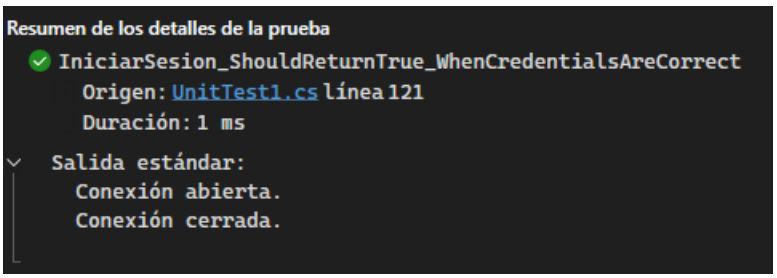
  Llamar al método IniciarSesion() del objeto credenciales creado.

  Almacenar el resultado booleano en la variable 'sesionIniciada'.
- Assert (Afirmación):

  Verificar que el valor de 'sesionIniciada' sea true utilizando Assert.IsTrue().

Resultado esperado:

El método IniciarSesion() debe retornar true, indicando que las credenciales proporcionadas son correctas y el inicio de sesión fue exitoso.



Código:

```
public void IniciarSesion_ShouldReturnTrue_WhenCredentialsAreCorrect()
{
    // Arrange
    var credenciales = new csEmpleados.csCredenciales("johndoe", "newpassword");

    // Act
    bool sesionIniciada = credenciales.IniciarSesion();

    // Assert
    Assert.IsTrue(sesionIniciada);
}
```

3.10. IniciarSesion_ShouldReturnFalse_WhenIncorrectCredentials

Descripción:

Esta prueba unitaria verifica que el método IniciarSesion() de la clase csCredenciales (una clase interna de csEmpleados) funcione correctamente cuando se proporcionan credenciales incorrectas. La prueba espera que el método retorne false, indicando un intento de inicio de sesión fallido.

Precondiciones:

- La clase csEmpleados y su clase interna csCredenciales están implementadas y accesibles.
- El método IniciarSesion() está implementado en la clase csCredenciales.
- El sistema de autenticación está disponible y accesible.
- Existe un usuario en el sistema con nombre de usuario "johndoe", pero su contraseña no es "wrongpassword".

Pasos:

- Arrange (Preparación):

  Crear un objeto csCredenciales con nombre de usuario "johndoe" y contraseña incorrecta "wrongpassword".

- Act (Acción):

  Llamar al método IniciarSesion() del objeto credenciales creado.

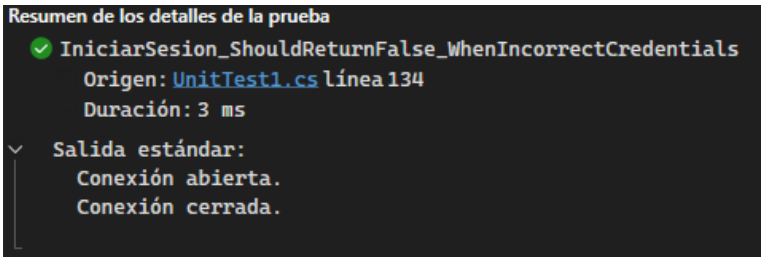
  Almacenar el resultado booleano en la variable 'sesionIniciada'.

- Assert (Afirmación):

  Verificar que el valor de 'sesionIniciada' sea false utilizando Assert.IsFalse().

Resultado esperado:

El método IniciarSesion() debe retornar false, indicando que las credenciales proporcionadas son incorrectas y el intento de inicio de sesión ha fallado.



Código:

```
public void IniciarSesion_ShouldReturnFalse_WhenIncorrectCredentials()
{
    // Arrange
      var credenciales = new csEmpleados.csCredenciales("johndoe", "wrong-
password");

    // Act
    bool sesionIniciada = credenciales.IniciarSesion();

    // Assert
    Assert.IsFalse(sesionIniciada);
}
```

3.11. ListarEmpleados_ShouldReturnListOfEmployees

Descripción:

Esta prueba unitaria verifica que el método estático ListarEmpleados() de la clase csEmpleados funcione correctamente al recuperar una lista de empleados del sistema. La prueba espera que el método devuelva una lista no nula y no vacía de objetos csEmpleados.

Precondiciones:

- La clase csEmpleados está implementada y accesible.
- El método estático ListarEmpleados() está implementado en la clase csEmpleados.
- La base de datos o el sistema de almacenamiento está disponible y accesible.
- Existen uno o más empleados registrados en el sistema.

Pasos:

- Act (Acción):

  Llamar al método estático ListarEmpleados() de la clase csEmpleados.

  Almacenar el resultado (una lista de objetos csEmpleados) en la variable 'empleados'.

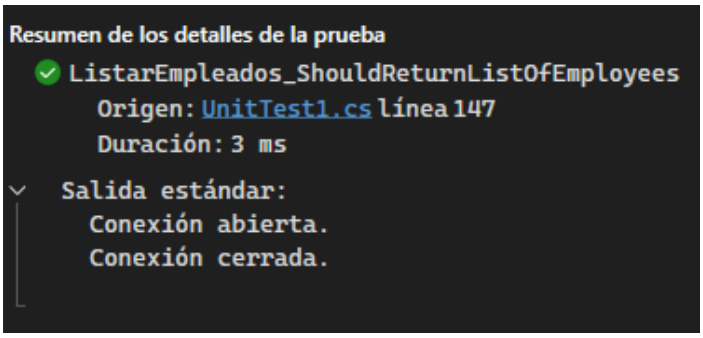
- Assert (Afirmación):

  Verificar que la lista 'empleados' no sea nula utilizando Assert.IsNotNull().

  Verificar que la lista 'empleados' contenga al menos un elemento utilizando Assert.IsTrue(empleados.Count > 0).

Resultado esperado:

El método ListarEmpleados() debe retornar una lista no nula que contenga al menos un objeto csEmpleados, representando los empleados registrados en el sistema.



Código:

```
public void ListarEmpleados_ShouldReturnListOfEmployees()
{
    // Act
    List<csEmpleados> empleados = csEmpleados.ListarEmpleados();

    // Assert
    Assert.IsNotNull(empleados);
    Assert.IsTrue(empleados.Count > 0);
}
```

3.12. BuscarEmpleados_ShouldReturnMatchingEmployees

Descripción:

Esta prueba unitaria verifica que el método estático BuscarEmpleados() de la clase csEmpleados funcione correctamente al buscar empleados que coincidan con el término de búsqueda "John". La prueba espera que el método devuelva una lista no nula y no vacía de objetos csEmpleados que contengan empleados cuyos nombres coincidan con "John".

Precondiciones:

- La clase csEmpleados está implementada y accesible.
- El método estático BuscarEmpleados(string busqueda) está implementado en la clase csEmpleados.
- Existen empleados registrados en el sistema cuyos nombres coincidan con "John".

Pasos:

- Arrange (Preparación)

  Asignar el valor “John” a la variable ‘busqueda’
- Act (Acción):

  Llamar al método estático BuscarEmpleados(busqueda) de la clase csEmpleados.

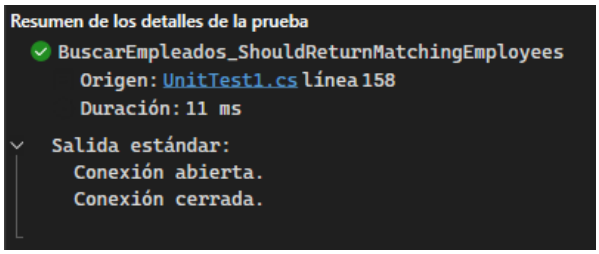
  Almacenar el resultado (una lista de objetos csEmpleados) en la variable empleados.
- Assert (Afirmación):

  Verificar que la lista empleados no sea nula utilizando Assert.IsNotNull(empleados).

Verificar que la lista empleados contenga al menos un elemento utilizando Assert.IsTrue(empleados.Count > 0).

Resultado esperado:

El método BuscarEmpleados("John") debe retornar una lista no nula que contenga al menos un objeto csEmpleados cuyo nombre coincida con "John", representando los empleados encontrados en el sistema que coinciden con el término de búsqueda.



Código:

```
public void BuscarEmpleados_ShouldReturnMatchingEmployees()
{
    // Arrange
    string busqueda = "John"; // Ejemplo de término de búsqueda

    // Act
    List<csEmpleados> empleados = csEmpleados.BuscarEmpleados(busqueda);

    // Assert
    Assert.IsNotNull(empleados);
    Assert.IsTrue(empleados.Count > 0);
}
```

3.13. RegistrarProducto_ShouldReturnTrue_WhenRegistrationIsSuccessful

Descripción:

Esta prueba unitaria verifica que el método RegistrarProducto() de la clase csProducto funcione correctamente al intentar registrar un nuevo producto con los parámetros especificados. La prueba espera que el método retorne true cuando el registro sea exitoso.

Precondiciones:

- La clase csProducto está implementada y accesible.
- Se ha creado una instancia de csProducto con nombre "Producto de Prueba", precio de 10.5 y cantidad inicial de 100.

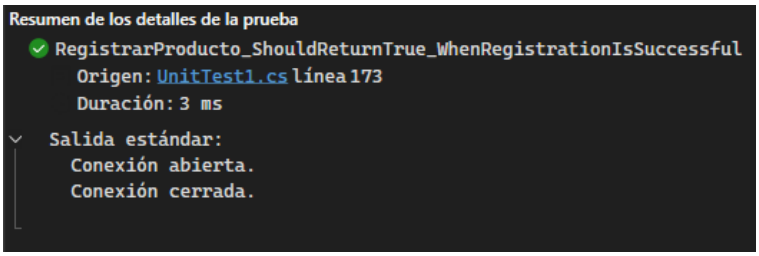
Pasos:

- Arrange (Preparación):
  Crear una nueva instancia de csProducto llamada producto con los valores especificados: nombre "Producto de Prueba", precio 10.5 y cantidad 100.
- Act (Acción):
  Llamar al método RegistrarProducto() en la instancia producto.
- Assert (Afirmación):

Verificar que la variable registrada sea true utilizando Assert.IsTrue(registrado).

Resultado esperado:

Se espera que el método RegistrarProducto() retorne true, indicando que el registro del producto fue exitoso.



Código:

```
public void RegistrarProducto_ShouldReturnTrue_WhenRegistrationIsSuccessful()
{
    // Arrange
    var producto = new csProducto("Producto de Prueba", 10.5m, 100);

    // Act
    bool registrado = producto.RegistrarProducto();

    // Assert
    Assert.IsTrue(registrado);
}
```

3.14. EditarProducto_ShouldReturnTrue_WhenEditIsSuccessful

Descripción:

Esta prueba unitaria verifica que el método EditarProducto() de la clase csProducto funcione correctamente al intentar editar un producto existente con los parámetros especificados. La prueba espera que el método retorne true cuando la edición sea exitosa.

Precondiciones:

- La clase csProducto está implementada y accesible.
- Se ha creado una instancia de csProducto con el ID 1, nombre "Producto Actualizado", precio de 15.75 y cantidad de 200, representando un producto existente que se desea editar.

Pasos:

- Arrange (Preparación):

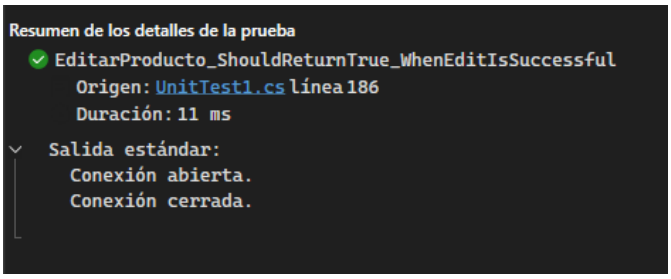
  Crear una nueva instancia de csProducto llamada producto con los valores especificados: ID 1, nombre "Producto Actualizado", precio 15.75 y cantidad 200.
- Act (Acción):

  Llamar al método EditarProducto() en la instancia producto.
- Assert (Afirmación):

  Verificar que la variable editada sea true utilizando Assert.IsTrue(editado).

Resultado esperado:

Se espera que el método EditarProducto() retorne true, indicando que la edición del producto fue exitosa.



Código:

```
public void EditarProducto_ShouldReturnTrue_WhenEditIsSuccessful()
{
    // Arrange
    var producto = new csProducto(1, "Producto Actualizado", 15.75m, 200);

    // Act
    bool editado = producto.EditarProducto();

    // Assert
    Assert.IsTrue(editado);
}
```

3.15. ListarProductos_ShouldReturnListOfProducts

Descripción:

Esta prueba unitaria verifica que el método estático ListarProductos() de la clase csProducto funcione correctamente al listar los productos existentes en el sistema. La prueba espera que el método devuelva una lista no nula y que contenga al menos un producto.

Precondiciones:

- La clase csProducto está implementada y accesible.
- Existen productos registrados en el sistema que puedan ser listados por el método ListarProductos().

Pasos:

- Act (Acción):

  Llamar al método estático ListarProductos() de la clase csProducto para obtener la lista de productos.

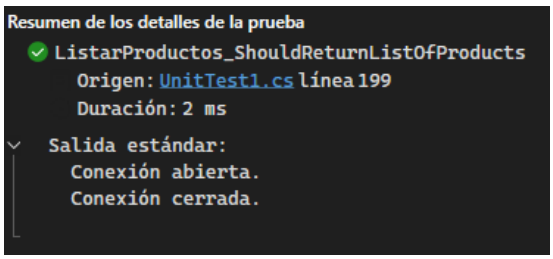
- Assert (Afirmación):

  Verificar que la variable productos no sea null utilizando Assert.IsNotNull(productos).

  Verificar que la lista productos contenga al menos un elemento utilizando Assert.IsTrue(productos.Count > 0).

Resultado esperado:

Se espera que el método ListarProductos() retorne una lista no nula que contenga al menos un producto, representando los productos existentes en el sistema.



Código:

```
public void ListarProductos_ShouldReturnListOfProducts()
{
    // Act
    List<csProducto> productos = csProducto.ListarProductos();

    // Assert
    Assert.IsNotNull(productos);
    Assert.IsTrue(productos.Count > 0);
}
```

3.16. BuscarProductoPorNombre_ShouldReturnMatchingProducts

Descripción:

Esta prueba unitaria verifica que el método estático BuscarProductoPorNombre() de la clase csProducto funcione correctamente al buscar productos cuyo nombre coincida con el patrón especificado. La prueba espera que el método devuelva una lista no nula y que contenga al menos un producto cuyo nombre coincida con el patrón de búsqueda.

Precondiciones:

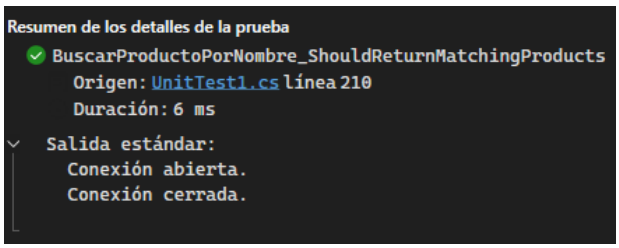
- La clase csProducto está implementada y accesible.
- Existen productos registrados en el sistema cuyos nombres coincidan con el patrón de búsqueda especificado ("Banano" en este caso).

Pasos:

- Arrange (Preparación):
  Asignar el valor "Banano" a la variable patronNombre como ejemplo de término de búsqueda.
- Act (Acción):
  Llamar al método estático BuscarProductoPorNombre(patronNombre) de la clase csProducto para buscar productos que coincidan con el nombre especificado por patronNombre.
- Assert (Afirmación):
  Verificar que la variable productos no sea null utilizando Assert.IsNotNull(productos).
  Verificar que la lista productos contenga al menos un elemento utilizando Assert.IsTrue(productos.Count > 0).

Resultado esperado:

Se espera que el método BuscarProductoPorNombre("Banano") retorne una lista no nula que contenga al menos un producto cuyo nombre coincida con el patrón de búsqueda "Banano", representando los productos encontrados en el sistema que coinciden con el término de búsqueda.



Código:

```
public void BuscarProductoPorNombre_ShouldReturnMatchingProducts()
{
    // Arrange
    string patronNombre = "Banano"; // Ejemplo de término de búsqueda

    // Act
    List<csProducto> productos = csProducto.BuscarProductoPorNombre(patronNombre);

    // Assert
    Assert.IsNotNull(productos);
    Assert.IsTrue(productos.Count > 0);
}
```

3.17. RegistrarFactura_ShouldReturnTrue_WhenRegistrationIsSuccessfulç

Descripción:

Esta prueba unitaria verifica que el método RegistrarFactura() de la clase csFactura funcione correctamente al intentar registrar una nueva factura con los detalles y parámetros especificados. La prueba espera que el método retorne true cuando el registro de la factura sea exitoso.

Precondiciones:

- La clase csFactura está implementada y accesible.
- Se han creado instancias de csFactura.FacturaDetalle para representar los detalles de la factura.
- Se ha creado una lista de detalles de factura (detalles) que contiene al menos dos elementos.
- Se ha creado una instancia de csFactura llamada factura.
- Se ha definido una fecha actual (fecha).
- Se han especificado IDs válidos de cliente (idCliente) y empleado (idEmpleado) en la base de datos.

Pasos:

- Arrange (Preparación):
  - Se han creado dos instancias de csFactura.FacturaDetalle (detalle1 y detalle2) con diferentes IDs de producto, precios, cantidades y valores de IVA.
  - Se ha inicializado una lista detalles que contiene los detalles de factura creados anteriormente.
  - Se ha creado una instancia de csFactura llamada factura.
  - Se ha definido la fecha actual en la variable fecha.
  - Se han especificado IDs válidos de cliente (idCliente) y empleado (idEmpleado).

- Act (Acción):

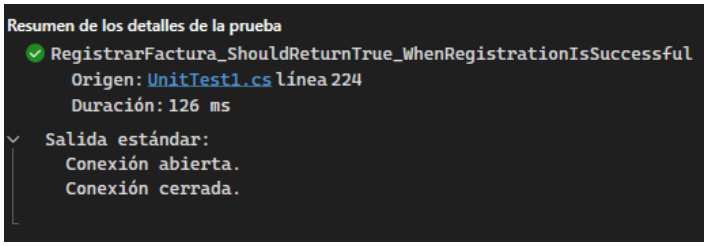
  Llamar al método RegistrarFactura(idCliente, fecha, detalles, idEmpleado) en la instancia factura para intentar registrar la factura con los detalles proporcionados.

- Assert (Afirmación):

  Verificar que la variable registrada sea true utilizando Assert.IsTrue(registrado).

Resultado esperado:

Se espera que el método RegistrarFactura() retorne true, indicando que el registro de la factura fue exitoso con los parámetros y detalles proporcionados.



Código:

```
public void RegistrarFactura_ShouldReturnTrue_WhenRegistrationIsSuccessful()
{
    // Arrange
    csFactura.FacturaDetalle detalle1 = new csFactura.FacturaDetalle
    {
        ID_Produc = 1,
        Precio = 10.5m,
        Cantidad = 2,
        IVA = 0.12m
    };

    csFactura.FacturaDetalle detalle2 = new csFactura.FacturaDetalle
    {
        ID_Produc = 2,
        Precio = 15.75m,
        Cantidad = 1,
        IVA = 0.12m
    };

    List<csFactura.FacturaDetalle> detalles = new List<csFactura.FacturaDetalle>();
    detalles.Add(detalle1);
    detalles.Add(detalle2);

    csFactura factura = new csFactura();
    DateTime fecha = DateTime.Now;
    long idCliente = 1; // ID de cliente válido en tu base de datos
    long idEmpleado = 1; // ID de empleado válido en tu base de datos

    // Act
    bool registrado = factura.RegistrarFactura(idCliente, fecha, detalles, idEmpleado);

    // Assert
    Assert.IsTrue(registrado);
}
```

3.18. ObtenerUltimoIDFactura_ShouldReturnValidID

Descripción:

Esta prueba unitaria verifica que el método ObtenerUltimoIDFactura() de la clase csFactura funcione correctamente al obtener el último ID válido de factura registrado en el sistema. La prueba espera que el método retorne un ID mayor que 0, indicando que se ha obtenido correctamente el último ID de factura.

Precondiciones:

- La clase csFactura está implementada y accesible.
- Se ha creado una instancia de csFactura llamada factura.

Pasos:

- Arrange (Preparación):

  Se ha creado una instancia de csFactura llamada factura.

- Act (Acción):

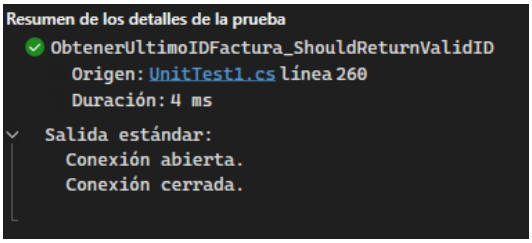
  Llamar al método ObtenerUltimoIDFactura() en la instancia factura para obtener el último ID de factura registrado en el sistema.

- Assert (Afirmación):

  Verificar que el valor de ultimoID sea mayor que 0 utilizando Assert.IsTrue(ultimoID > 0).

Resultado esperado:

Se espera que el método ObtenerUltimoIDFactura() retorne un ID válido (mayor que 0), indicando que se ha obtenido correctamente el último ID de factura registrado en el sistema.



Código:

```
public void ObtenerUltimoIDFactura_ShouldReturnValidID()
{
    // Arrange
    csFactura factura = new csFactura();

    // Act
    long ultimoID = factura.ObtenerUltimoIDFactura();

    // Assert
    Assert.IsTrue(ultimoID > 0);
}
```

3.19. FechaFacturaCliente_ShouldReturnListOfDates

Descripción:

Esta prueba unitaria verifica que el método estático FechaFacturaCliente() de la clase csFactura funcione correctamente al obtener las fechas de factura asociadas a un cliente específico en el sistema. La prueba espera que el método devuelva una lista no nula y que contenga al menos una fecha de factura para el cliente especificado.

Precondiciones:

- La clase csFactura está implementada y accesible.

- Existe al menos una factura asociada al cliente con el ID especificado (idCliente).

Pasos:

- Arrange (Preparación):

  Se ha definido el ID de cliente válido (idCliente) en la base de datos.

- Act (Acción):

  Llamar al método estático FechaFacturaCliente(idCliente) de la clase csFactura para obtener las fechas de factura asociadas al cliente especificado.

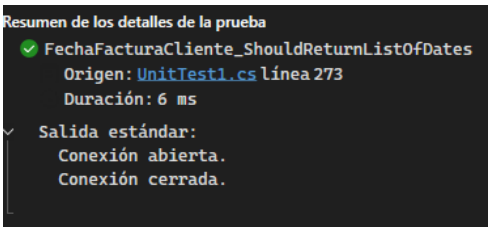
- Assert (Afirmación):

  Verificar que la variable fechasFacturas no sea null utilizando Assert.IsNotNull(fechasFacturas).

  Verificar que la lista fechasFacturas contenga al menos un elemento utilizando Assert.IsTrue(fechasFacturas.Count > 0).

Resultado esperado:

Se espera que el método FechaFacturaCliente(idCliente) retorne una lista no nula que contenga al menos una fecha de factura asociada al cliente con el ID especificado, representando las fechas de factura encontradas en el sistema para ese cliente.



Código:

```
public void FechaFacturaCliente_ShouldReturnListOfDates()
{
    // Arrange
    long idCliente = 1; // ID de cliente válido en tu base de datos

    // Act
    List<csFactura> fechasFacturas = csFactura.FechaFacturaCliente(idCliente);

    // Assert
    Assert.IsNotNull(fechasFacturas);
    Assert.IsTrue(fechasFacturas.Count > 0);
}
```

4. Conclusión

El proceso de pruebas unitarias para las clases csCliente, csEmpleados, csProducto y csFactura en el sistema de facturación de la UTEQ ha sido crucial para asegurar que el software funcione correctamente. Estas pruebas nos han permitido verificar paso a paso cómo se registran, editan, buscan y listan clientes, empleados, productos y facturas. Detectamos errores desde el principio y aseguramos que el sistema cumpla con lo que necesita antes de lanzarlo. Es una parte clave para mantener el software confiable y listo para futuras mejoras.