**Documentación Código ICCP.Net**

**INTEGRATED CUSTOMER COMPLAINTS PLATFORM DEVELOPED IN ASP.NET**

**ROBOTECH INC**

**SANTIAGO - 2017**

Mariana Rivas

Joel Baez

Oscar Ortiz

Sebastián Miranda

Benjamín Ruiz

Programa tu Futuro – Robotech Inc

Última modificación: 30/11/2017 – 16:59

# Introducción

Integrated Customer Complaints Platform developed in .Net (ICCP.Net por sus siglas en inglés) es una aplicación web para <cliente> que desea tener un sistema de toma de reclamos, la cual tiene las siguientes características integradas:

* Acceso con perfiles, con una encriptación SHA1 en la base de datos.
* Pantalla donde se pueda ingresar el reclamo (Tipo de reclamo, Descripción, Fecha, Etc.) además los datos del cliente generador del reclamo.
* Permitir búsqueda de reclamos (por Rut o número de reclamo), poder agregar antecedentes y cambiar estado del reclamo.
* Visualizar el reclamo en PDF, custodiar el PDF en una estructura de directorio.
* Agrega un SLA por reclamo.
* Reporte de Reclamos por SLA.
* Reporte de Reclamos en gráficos con Chart.js

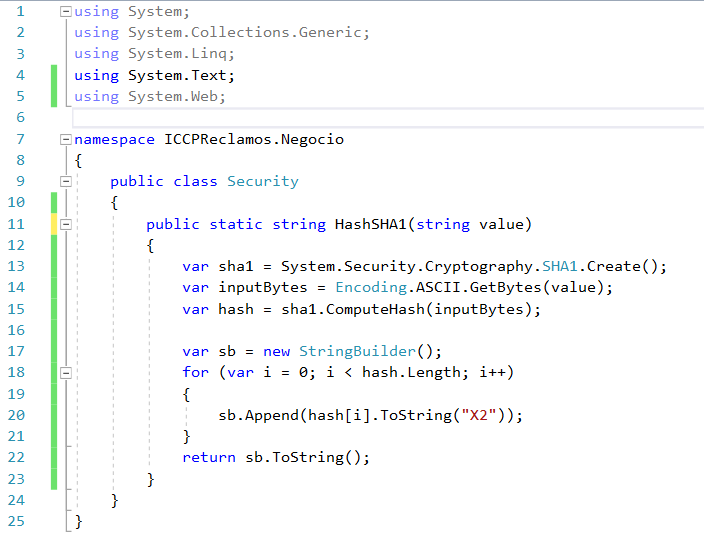
Se utilizaron las siguientes tecnologías para el desarrollo del programa:

* Visual Studio 2017 (IDE para la programación del sistema).
* Microsoft SQL Server 2014 (Motor de bases de datos).
* Librería iTextSharp para la generación de PDFs.
* Librería Chart.JS para la visualización de gráficos en el panel administración.

El sistema se encuentra constituido en 3 carpetas, un modelo de 3-capas las cuáles son:

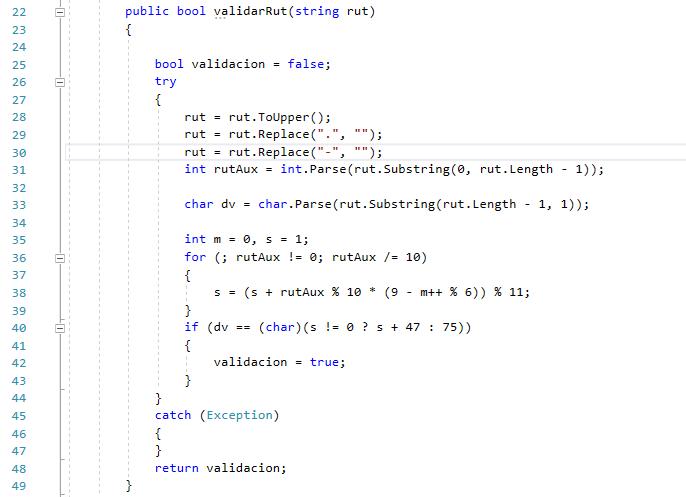
* Datos : Capa en la cual se manejan los datos con la Base de Datos, en esta tenemos una clase Datos.cs que recibe y entrega datos.
* Negocio: Capa en la cual se manejan las Clases para todos los tipos de Objetos que hay dentro del proyecto sus métodos y sus constructores.
* Presentación: Capa visual y vista directa del proyecto. Contiene archivos ASPX y CSHTML (Razor) que son los necesarios para la visualización de la página. También existen los métodos que hacen las llamadas a las funciones de la capa Datos mediante la capa Negocio.

Negocio/Security.cs



Función que recibe un valor de tipo string como “value”, que retorna el string mismo pasado por HASH SHA1 (método de seguridad)

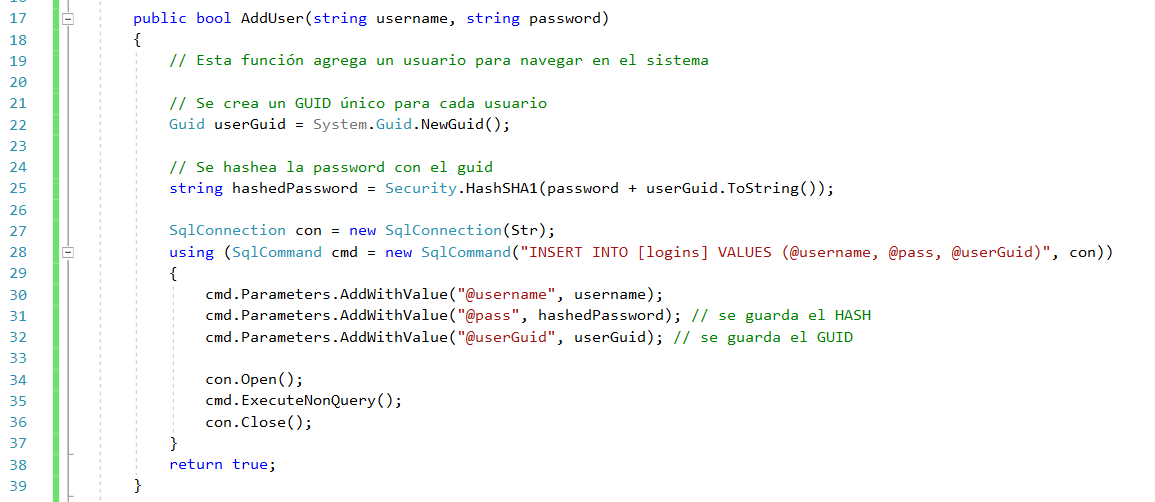
Negocio/Validadores.cs



Función validarRut que recibe un valor de tipo string, en referencia a un RUT chileno (de forma 11.111.111-K / 11111111-K / 11111111K), se pasa por módulo 11 para saber si su dígito verificador es correcto.

Fuente: <https://www.leychile.cl/Navegar?idNorma=3559>

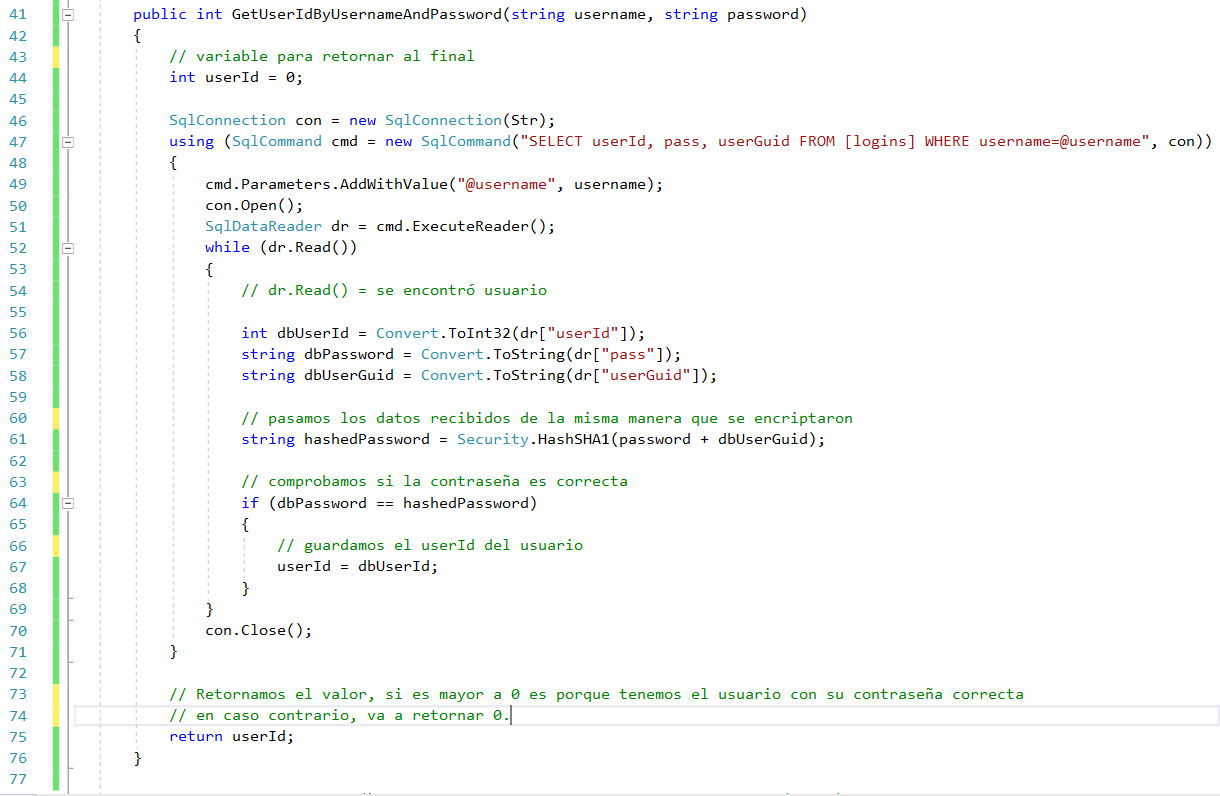
Datos/Datos.cs



Función AddUser, retorna un boolean para agregar un usuario a la base de datos. Recibe dos valores de tipo string: username y password, los cuales representan el usuario y contraseña a guardar.

Esta función obtiene un valor Guid para guardarlo en la base de datos (dato único) y así crear una especie de “SALT” para agregar más seguridad a la contraseña. Luego, se genera un string “hashedPassword” el cuál va a almacenar la contraseña concatenada con el salt y va a pasar por SHA1 para su protección llamando a la clase Negocio/Security.cs y el método HashSHA1.

Luego de lo anterior, se ejecuta la query que va a almacenar los datos en la base de datos. Si todo se encuentra bien, nuestro boolean retornará “verdadero”.



Función GetUserIdByUserUsernameAndPassword, retorna un valor de tipo int para validar a un usuario desde la base de datos. Recibe dos valores de tipo string: username y password, los cuales representan el usuario y contraseña del usuario.

Esta función, tal como la de guardar usuario, necesitan pasar por un hash para comprobar su integridad. Si ésta existe y el usuario con su contraseña coinciden con la base de datos, el valor que entregará será el ID del usuario (mayor que 0). Por lo tanto, si no encontramos al usuario, su contraseña no coincide o no hay registros en la base de datos, la función retornará 0 lo que será un “Not-Authed” o no autenticado.