PROGRAMACION I
SECCION A
ING. MARCO ANTONIO CULAJAY

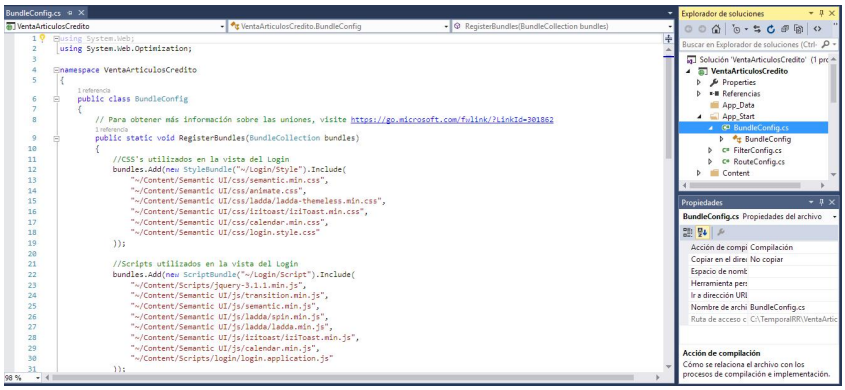
PROYECTO FINAL

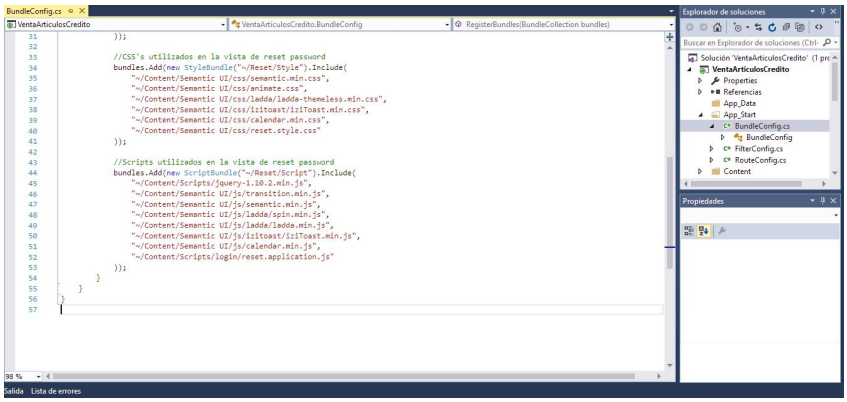
PROGRAMA DE CONTROL DE COMPRAS EN LINEA
.NET Y SQL

Integrantes del grupo:	Carné:
ANGEL GABRIEL CABRERA GUZMAN	5190-18-3012
ERVIN ALEXANDER LOPEZ RUEDA	5190-18-692
FAVIO EZEQUIEL URREA AGUILAR	5190-17-712
ROBIN YAN MARIA RIVAS CRUZ	5190-11-2280

Dentro de la carpeta AppStart encontramos las siguientes clases

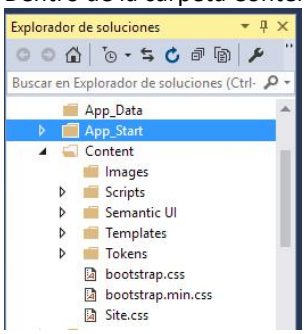
Clase donde se cargan lo styles y los scripts



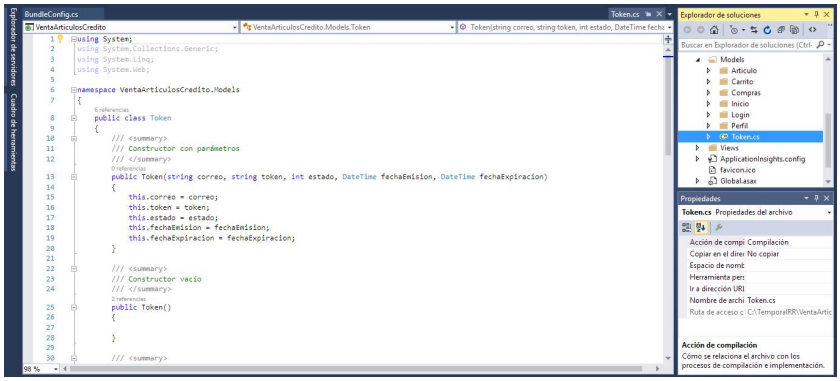


Clase en la que se indica cual es la vista principal

Dentro de la carpeta Content estan almacenados los archivos que se utilizan para dar estilo

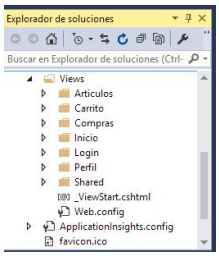


Dentro de la carpeta Models:
Dentro contiene las carpetas que contienen los modelos de las vista

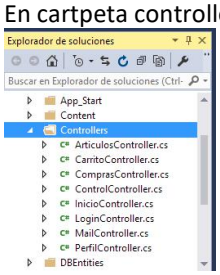


Encontramos una clase Tokens.cs que nos ayuda a validar los tokens de restauracion de contraseña

Carpeta Views
Dentro contiene las carpetas que contienen los las vistas de la aplicacion



En cartpeta controllers encontramos los controladores de las vistas



Descripcion por clase:
ArticulosController.cs
/*Lista los articulos y permite agreagar articulos*/

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using VentaArticulosCredito.DBEntities;
using VentaArticulosCredito.Models.Articulo;

namespace VentaArticulosCredito.Controllers
{
    public class ArticulosController : ControlController
    {
        /// <summary>
        /// Devuelve la vista de artículos
        /// </summary>
        /// <returns></returns>
        public ActionResult Articulos()
        {
            if (ValidaUsuario())
            {
                ArticulosPageModel model = new ArticulosPageModel();

                using (var db = new VentaArticulosCreditoEntities()) {
                    ViewBag.ListaCategorias = db.Categoria.ToList();
                    model.articulos = db.Articulo.Include("SubCategoria.Categoria").Include("Imagen_Articulo").ToList();
                }
```

```
            return View("ArticuloListView", model);
        }

        return RedirectToAction("Login", "Login");
    }

    /// <summary>
    /// Agrega stock al artículo
    /// </summary>
    /// <param name="model"></param>
    /// <returns></returns>
    [HttpPost,ValidateAntiForgeryToken]
    public JsonResult AddStock(AddStockModel model)
    {
        int tipo = 0;
        string mensaje = "";

        if (!ModelState.IsValid)
        {
            //Extrae el primer mensaje de error que tenga el modelo
            mensaje = ModelState.Values.Select(e => e.Errors).Where(e => e.Count > 0).FirstOrDefault().Select(v => v.ErrorMessage).FirstOrDefault();
        }
        else
        {
            using(var db = new VentaArticulosCreditoEntities())
            {
                var stock = new Inventario(model.articulo, model.precioCompra, model.precioVenta, model.cantidad, model.fecha);

                db.Inventario.Add(stock);
                db.SaveChanges();
                tipo = 1;
                mensaje = "Stock agregado correctamente";
            }
        }

        return Json(new { tipo = tipo, mensaje = mensaje }, JsonRequestBehavior.AllowGet);
    }

    /// <summary>
    /// Crea un artículo
    /// </summary>
    /// <param name="model">Modelo</param>
    /// <returns>Json</returns>
    [HttpPost, ValidateAntiForgeryToken]
    public JsonResult CreateArticle(CreateArticleModel model)
    {
        int tipo = 0;
        string mensaje = "";

        if (!ModelState.IsValid)
        {
            //Extrae el primer mensaje de error que tenga el modelo
            mensaje = ModelState.Values.Select(e => e.Errors).Where(e => e.Count > 0).FirstOrDefault().Select(v => v.ErrorMessage).FirstOrDefault();
        }
        else
        {

        }

        return Json(new { tipo = tipo, mensaje = mensaje }, JsonRequestBehavior.AllowGet);
    }

    /// <summary>
    /// Recupera subcategorias
    /// </summary>
    /// <param name="categoria">Categoria</param>
    /// <returns>Json</returns>
    [HttpPost]
    public JsonResult GetSubCategorias(int categoria)
    {
        int tipo = 0;
        string mensaje = "";
        SelectList listaSub = null;

        using (var db = new VentaArticulosCreditoEntities())
        {
            try
            {
                listaSub = new SelectList(db.SubCategoria.Where(m => m.codigoCategoria == categoria).ToList(), "codigo", "nombre");
                tipo = 1;
            }
            catch (Exception ex)
            {
                mensaje = "Error al recuperar municipios";
            }
        }

        return Json(new { tipo = tipo, mensaje = mensaje, listaSub = listaSub }, JsonRequestBehavior.AllowGet);
    }
}
}
```

Clase CarritoController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using VentaArticulosCredito.DBEntities;


namespace VentaArticulosCredito.Controllers
{
    public class CarritoController : ControlController
    {
        /// <summary>
        /// Devuelve la vista del carrito de compras
        /// </summary>
        /// <returns>Vista</returns>
        [HttpGet]
        public ActionResult Carrito()
        {
            if (ValidaUsuario())
            {
                using (var db = new VentaArticulosCreditoEntities())
                {
                    var user = (Usuario)Session["usuario"];
                    var carrito = db.Carrito.Include("Inventario.Articulo.Imagen_Articulo").Where(c => c.codigoUsuario == user.codigo).ToList();
                    var direcciones = db.Direccion_Usuario.Include("Usuario").Include("Municipio.Departamento").Where(d => d.codigoUsuario ==
user.codigo).ToList();
                    var model = new VentaArticulosCredito.Models.Carrito.CarritoPageModel(carrito, direcciones);

                    return View(model);
                }
            }

            return RedirectToAction("Login","Login");
        }

        /// <summary>
        /// Elimina artículo del carrito
        /// </summary>
        /// <param name="correlativo"></param>
        /// <returns></returns>
        [HttpPost]
        public JsonResult EliminarArticulo(int correlativo)
        {
            int tipo = 0;
            string mensaje = "";

            using(var db = new VentaArticulosCreditoEntities())
            {
                try
                {
                    var carrito = db.Carrito.Where(c => c.correlativo == correlativo).FirstOrDefault();

                    if(carrito != null)
                    {
                        db.Carrito.Remove(carrito);
                        db.SaveChanges();
                        tipo = 1;
                    }
                }
                catch (Exception ex)
                {
                    mensaje = "Error al eliminar del carrito";
                }
            }

            return Json(new { tipo = tipo, mensaje = mensaje }, JsonRequestBehavior.AllowGet);
        }
    }
}
```

Clase ComprasController.cs

```
/*Lista las compras*/
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using VentaArticulosCredito.DBEntities;
using VentaArticulosCredito.Models.Compras;

namespace VentaArticulosCredito.Controllers
{
    public class ComprasController : ControlController
    {
        /// <summary>
        /// Devuelve la lista con las compras realizadas
        /// </summary>
        /// <returns></returns>
        [HttpGet]
        public ActionResult Compras()
        {
```

```
            if (ValidaUsuario())
            {
                ComprasPageModel model = new ComprasPageModel();

                using (var db = new VentaArticulosCreditoEntities())
                {
                    var user = (Usuario)Session["usuario"];
                    model.compras = db.Factura.Include("Direccion_Usuario").Include("DetalleFactura.Articulo").Where(c => c.codigoUsuario ==
user.codigo).ToList();
                }

                return View("ComprasListView", model);
            }

            return RedirectToAction("Login", "Login");
        }

        /// <summary>
        /// Devuelve la vista con la orden de compra que se quiere ver
        /// </summary>
        /// <param name="orden"></param>
        /// <returns></returns>
        [HttpGet]
        public ActionResult Compra(String orden)
        {
            if (ValidaUsuario())
            {
                CompraPageModel model = new CompraPageModel();
                int fact = 0;

                using (var db = new VentaArticulosCreditoEntities())
                {
                    if (orden != null)
                    {
                        fact = Convert.ToInt32(orden);

                        model.factura =
db.Factura.Include("DetalleFactura.Articulo.Imagen_Articulo").Include("Direccion_Usuario.Municipio.Departamento").Where(f => f.codigo ==
fact).FirstOrDefault();
                        Session["orden-compra"] = orden;
                    }
                    else
                    {
                        fact = Convert.ToInt32(Session["orden-compra"]);

                        model.factura =
db.Factura.Include("DetalleFactura.Articulo.Imagen_Articulo").Include("Direccion_Usuario.Municipio.Departamento").Where(f => f.codigo ==
fact).FirstOrDefault();
                    }
                }

                return View("CompraView", model);
            }

            return RedirectToAction("Login","Login");
        }

        /// <summary>
        /// Confirma la orden de compra del carrito del usuario
        /// </summary>
        /// <returns></returns>
        [HttpPost]
        public JsonResult ConfirmarOrden(int direccionEntrega)
        {
            int tipo = 0;
            string mensaje = "";
            int orden = 0;

            using (var db = new VentaArticulosCreditoEntities())
            {
                try
                {
                    var user = (Usuario)Session["usuario"];
                    var carrito = db.Carrito.Include("Inventario.Articulo").Where(c => c.codigoUsuario == user.codigo).ToList();
                    var correlativo = new CorrelativoFactura(Guid.NewGuid().ToString());
                    int posicion = 1;

                    //Creo el códio de la factura
                    db.CorrelativoFactura.Add(correlativo);
                    db.SaveChanges();
                    //Creo el encabezado de la factura
                    var factura = new Factura(correlativo.correlativo, DateTime.Now, user.codigo, 0, direccionEntrega);
                    db.Factura.Add(factura);
                    db.SaveChanges();

                    foreach (var car in carrito)
                    {
                        //Creo detalle por cada posición del carrito
                        var detalle = new DetalleFactura(posicion, correlativo.correlativo, (decimal)car.Inventario.precioVenta, (int)car.cantidad,
car.Inventario.Articulo.codigo);
                        db.DetalleFactura.Add(detalle);
                        db.SaveChanges();
                        //Recupero el inventario de la posición y le resto la cantidad ordenada
                        var inventario = car.Inventario;
```

```
                    inventario.cantidad -= car.cantidad;
                    db.Entry(inventario).State = System.Data.Entity.EntityState.Modified;
                    db.SaveChanges();
                    //Elimino la posición del carrito
                    db.Carrito.Remove(car);
                    db.SaveChanges();
                    posicion += 1;
                }

                orden = correlativo.correlativo;
                //Envío correo
                MailController().SendConfirmOrder(user, ref tipo, ref mensaje, orden);
            }catch(Exception ex)
            {
                mensaje = "Error al confirmar orden";
            }
        }

        return Json(new { tipo = tipo, mensaje = mensaje, orden = orden }, JsonRequestBehavior.AllowGet);
    }

    /// <summary>
    /// Cancela la orden
    /// </summary>
    /// <param name="orden">Número orden</param>
    /// <returns>Json</returns>
    [HttpPost]
    public JsonResult CancelarOrden(int orden)
    {
        int tipo = 0;
        string mensaje = "";

        using (var db = new VentaArticulosCreditoEntities())
        {
            try
            {
                var order = db.Factura.Where(o => o.codigo == orden).FirstOrDefault();
                var user = (Usuario)Session["usuario"];

                order.estado = 4;
                db.Entry(order).State = System.Data.Entity.EntityState.Modified;
                db.SaveChanges();
                //Envío correo
                MailController().SendCancelOrder(user, ref tipo, ref mensaje, orden);
            }
            catch (Exception ex)
            {
                mensaje = "Error al cancelar orden";
            }
        }

        return Json(new { tipo = tipo, mensaje = mensaje }, JsonRequestBehavior.AllowGet);
    }
}
}
```

Clase ControlController.cs

/*Controlador principal Padres que tiene funciones y metodos globajes*/

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Net.Mail;
using System.Web;
using System.Web.Mvc;
using VentaArticulosCredito.DBEntities;
using VentaArticulosCredito.Models.Perfil;

namespace VentaArticulosCredito.Controllers
{
    public class ControlController : Controller
    {
        /// <summary>
        /// Constante de la ruta para registrarse
        /// </summary>
        public const string URL_Register = "http://localhost:54016/Login/Login";

        /// <summary>
        /// Constante de la ruta para restaurar contraseña
        /// </summary>
        public const string URL_Reset = "http://localhost:54016/Login/ResetPassword?token=";

        /// <summary>
        /// Constante de la ruta para ver confirmación de la orden
        /// </summary>
        public const string URL_ConfirmOrder = "http://localhost:54016/Compras/Compra?orden=";

        /// <summary>
        /// Devuelve el cliente SMTP para los correos
        /// </summary>
        /// <returns></returns>
        public SmtpClient ClienteSmtp()
        {
            SmtpClient cliente = new SmtpClient("smtp.gmail.com", 587);
```

```
    cliente.Credentials = new NetworkCredential("cyber.shop.register.gt@gmail.com", "CyberShop_GT");
    cliente.EnableSsl = true;
    return (cliente);
}

/// <summary>
/// Devuelve la ruta del template para el registro
/// </summary>
/// <returns></returns>
public string RegisterTemplate()
{
    return (AppDomain.CurrentDomain.BaseDirectory + @"Content\Templates\registerMail.html");
}

/// <summary>
/// Devuelve la ruta del template para restuarar la contraseña
/// </summary>
/// <returns></returns>
public string ResetPasswordTemplate()
{
    return (AppDomain.CurrentDomain.BaseDirectory + @"Content\Templates\resetPassword.html");
}

/// <summary>
/// Devuelve la ruta del template para de la confirmación de la orden
/// </summary>
/// <returns></returns>
public string ConfirmOrderTemplate()
{
    return (AppDomain.CurrentDomain.BaseDirectory + @"Content\Templates\confirmedOrder.html");
}

/// <summary>
/// Devuelve la ruta del template para de la cancelación de la orden
/// </summary>
/// <returns></returns>
public string CancelOrderTemplate()
{
    return (AppDomain.CurrentDomain.BaseDirectory + @"Content\Templates\cancelOrder.html");
}

/// <summary>
/// Retorna un objeto MailControler
/// </summary>
public MailController MailController()
{
    return (new MailController());
}

/// <summary>
/// Valida si hay un usuario logueado
/// </summary>
/// <returns>True o false si hay usuario</returns>
public Boolean ValidaUsuario()
{
    if (Session["usuario"] != null)
    {
        return true;
    }

    return false;
}

/// <summary>
/// Valida si hubo cambios en los datos del usuario
/// </summary>
/// <param name="oldUser"></param>
/// <param name="newUser"></param>
/// <returns></returns>
public Boolean HuboCambios(Usuario oldUser, UpdateInfoModel newUser, ref Boolean validaNIT, ref Boolean validaCorreo)
{
    Boolean hubo = false;

    if (newUser.email != oldUser.correo || newUser.nombre != oldUser.nombre || newUser.apellido != oldUser.apellido
        || newUser.fechaNacimiento != oldUser.fechaNacimiento || newUser.nit != oldUser.nit)
        hubo = true;

    if (newUser.nit != oldUser.nit)
        validaNIT = true;

    if (newUser.email != oldUser.correo)
        validaCorreo = true;

    return hubo;
}

/// <summary>
/// Valida si hubo cambios en los datos de la dirección
/// </summary>
/// <param name="oldUser"></param>
/// <param name="newUser"></param>
/// <returns></returns>
public Boolean HuboCambios(Direccion_Usuario oldDirec, EditDirectionModel newDirec)
{
```

```
            if (newDirec.nombre != oldDirec.nombre || newDirec.apellido != oldDirec.apellido || Convert.ToInt32(newDirec.telefono) != oldDirec.telefono
                || newDirec.direccion != oldDirec.direccion || newDirec.municipio != oldDirec.codigoMunicipio)
                return true;

            return false;
        }
    }
}
```

Clase InicioController.cs

/*Maneja la vista principal donde se enlistan los articulos*/

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using VentaArticulosCredito.DBEntities;
using VentaArticulosCredito.Helpers;
using VentaArticulosCredito.Models.Inicio;
using Microsoft.Ajax.Utilities;

namespace VentaArticulosCredito.Controllers
{
    public class InicioController : ControlController
    {
        /// <summary>
        /// Devuelve la vista principal de la página
        /// </summary>
        /// <returns>Vista</returns>
        [HttpGet]
        public ActionResult Inicio(string filter, string subCategoria)
        {
            if (ValidaUsuario())
            {
                using (var db = new VentaArticulosCreditoEntities())
                {
                    List<Categoria> categorias = db.Categoria.Include("SubCategoria").ToList();
                    List<Inventario> stock = db.Inventario.Include("Articulo.Imagen_Articulo").Where(i => i.cantidad > 0).OrderBy(i =>
i.fechaCompra).DistinctBy(i => i.codigoArticulo).ToList();

                    var model = new InicioPageModel(stock, categorias);

                    return View(model);
                }
            }

            return RedirectToAction("Login","Login");
        }

        /// <summary>
        /// Elimina las variables de sesión del usuario logueado
        /// </summary>
        /// <returns>Json con tipo y mensaje</returns>
        [HttpGet]
        public JsonResult CloseSession()
        {
            int tipo = 0;
            string mensaje = "";

            try
            {
                Session.Clear();
                Session.Abandon();
                Session.RemoveAll();
                tipo = 1;
            }
            catch (Exception ex)
            {
                mensaje = "Error al cerrar sesión";
            }

            return Json(new { tipo = tipo, mensaje = mensaje }, JsonRequestBehavior.AllowGet);
        }

        /// <summary>
        /// Retorna la infomación del inventario que se desea ver
        /// </summary>
        /// <param name="correlativoInventario">Codigo inventario</param>
        /// <returns>Modelo</returns>
        [HttpGet]
        public ActionResult ArticuloModalPartialView(int correlativoInventario)
        {
            Inventario model = null;

            using (var db = new VentaArticulosCreditoEntities())
            {
                model = db.Inventario.Include("Articulo.Imagen_Articulo").Where(i => i.correlativo == correlativoInventario).FirstOrDefault();
            }

            return PartialView("ArticuloModalPartialView", model);
        }

        /// <summary>
```

```
        /// Devuelve la cantidad de artículos que tiene el carrito del usuario logueado
        /// </summary>
        /// <returns>Json</returns>
        [HttpGet]
        public JsonResult ActualizarIndicadorCarrito()
        {
            int tipo = 0;
            string mensaje = "";
            int cantidad = 0;

            using (var db = new VentaArticulosCreditoEntities())
            {
                try
                {
                    var user = (Usuario)Session["usuario"];
                    var carrito = db.Carrito.Where(c => c.codigoUsuario == user.codigo).ToList();

                    cantidad = carrito.Count;
                    tipo = 1;
                }
                catch (Exception ex)
                {
                    mensaje = "Error al cargar información del carrito";
                }
            }

            return Json(new { tipo = tipo, mensaje = mensaje, cantidad = cantidad }, JsonRequestBehavior.AllowGet);
        }

        /// <summary>
        /// Agrega un artículo al carrito de compras
        /// </summary>
        /// <returns>Json</returns>
        [HttpPost]
        public JsonResult GuardarEnCarrito(int inventario, int cantidad)
        {
            int tipo = 0;
            string mensaje = "";
            Boolean exitoso = false;

            using (var db = new VentaArticulosCreditoEntities())
            {
                try
                {
                    var user = (Usuario)Session["usuario"];
                    var carritoExistente = db.Carrito.Where(c => c.codigoUsuario == user.codigo && c.codigoInventario == inventario).FirstOrDefault();

                    if (carritoExistente == null)
                    {
                        var carrito = new Carrito(user.codigo, inventario, cantidad);

                        db.Carrito.Add(carrito);
                        db.SaveChanges();
                        exitoso = true;
                    }
                    else
                    {
                        carritoExistente.cantidad += cantidad;
                        db.Entry(carritoExistente).State = System.Data.Entity.EntityState.Modified;
                        db.SaveChanges();
                        exitoso = true;
                    }

                    if (exitoso)
                    {
                        tipo = 1;
                        mensaje = "Agregado al carrito";
                    }

                }
                catch (Exception ex)
                {
                    mensaje = "Error al agregar al carrito";
                }
            }

            return Json(new { tipo = tipo, mensaje = mensaje }, JsonRequestBehavior.AllowGet);
        }
    }
}
```

Clase LoginController.cs

/*Valida la vista de Login o Registro de usuarios y recuperar contraseña*/

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text.RegularExpressions;
using System.Web;
using System.Web.Mvc;
using VentaArticulosCredito.DBEntities;
```

```
using VentaArticulosCredito.Helpers;
using VentaArticulosCredito.Models.Login;

namespace VentaArticulosCredito.Controllers
{
    public class LoginController : ControlController
    {
        /// <summary>
        /// Método que devuelve la vista del login
        /// </summary>
        /// <returns>Vista del login</returns>
        [HttpGet]
        public ActionResult Login()
        {
            if (!ValidaUsuario()) return View();

            return RedirectToAction("Inicio","Inicio");
        }

        /// <summary>
        /// Valida las credenciales del usuario
        /// </summary>
        /// <param name="model">Modelo para el login</param>
        /// <returns>Json con tipo y mensaje</returns>
        [HttpPost, ValidateAntiForgeryToken]
        public JsonResult Login(LoginViewModel model)
        {
            int tipo = 0;
            string mensaje = "";
            string correo = "";

            if (!ModelState.IsValid)
            {
                //Extrae el primer mensaje de error que tenga el modelo
                mensaje = ModelState.Values.Select(e => e.Errors).Where(e => e.Count > 0).FirstOrDefault().Select(v => v.ErrorMessage).FirstOrDefault();
            }
            else
            {
                using (var db = new VentaArticulosCreditoEntities())
                {
                    var userMail = db.Usuario.Where(u => u.correo == model.email).FirstOrDefault();

                    if (userMail != null)
                    {
                        if (userMail.estado == 1)
                        {
                            var passwordEncrypt = Encrypt.Encripta(model.password);

                            var userCredentials = db.Usuario.Where(u => u.correo == model.email && u.password == passwordEncrypt).FirstOrDefault();

                            if (userCredentials != null)
                            {
                                tipo = 1;
                                correo = userCredentials.correo;
                                Session["usuario"] = userCredentials;
                                Session["rolUsuario"] = userCredentials.codigoRol;
                            }
                            else
                            {
                                mensaje = "Contraseña incorrecta";
                            }
                        }
                        else
                        {
                            tipo = 2;
                            mensaje = "Aún no ha compleado su registro";
                        }
                    }
                    else
                    {
                        mensaje = "Correo no registrado";
                    }
                }
            }

            return Json(new { tipo = tipo, mensaje = mensaje, correo = correo }, JsonRequestBehavior.AllowGet);
        }

        /// <summary>
        /// Registra un nuevo cliente en la plataforma
        /// </summary>
        /// <param name="model">Modelo de registro</param>
        /// <returns>Json con mensaje y tipo de mensaje</returns>
        [HttpPost, ValidateAntiForgeryToken]
        public JsonResult Register(RegisterViewModel model)
        {
            int tipo = 0;
            string mensaje = "";

            if (!ModelState.IsValid)
```

```
    {
        //Extrae el primer mensaje de error que tenga el modelo
        mensaje = ModelState.Values.Select(e => e.Errors).Where(e => e.Count > 0).FirstOrDefault().Select(v => v.ErrorMessage).FirstOrDefault();
    }
    else
    {
        using (var db = new VentaArticulosCreditoEntities())
        {
            try
            {
                var userMail = db.Usuario.Where(u => u.correo == model.email).FirstOrDefault();
                var userNIT = db.Usuario.Where(u => u.nit == model.nit).FirstOrDefault();

                if (userMail != null)
                {
                    mensaje = "Este correo ya está registrado";
                }
                else if (userNIT != null)
                {
                    mensaje = "Este nit ya está siendo utilizado";
                }
                else
                {
                    if (ValidarNIT.ValidaNIT(model.nit))
                    {
                        var newUser = new Usuario(model.email, Encrypt.Encripta(model.password), model.nombre, model.apellido, model.fechaNacimiento, model.nit, 1, 0);

                        db.Usuario.Add(newUser);
                        db.SaveChanges();
                        MailController().SendRegisterMail(model, ref mensaje, ref tipo);
                    }
                    else
                    {
                        mensaje = "Ingrese un nit válido";
                    }
                }
            }
            catch (Exception ex)
            {
                mensaje = "Error al registrarse";
            }
        }
    }

    return Json(new { tipo = tipo, mensaje = mensaje }, JsonRequestBehavior.AllowGet);
}

/// <summary>
/// Completa el registro de un usuario nuevo, es la confirmación
/// </summary>
/// <param name="email">Correo</param>
/// <returns>Json con tipo de mensaje y mensaje</returns>
[HttpPost]
public JsonResult CompleteRegister(string email)
{
    int tipo = 0;
    string mensaje = "";

    using (var db = new VentaArticulosCreditoEntities())
    {
        var user = db.Usuario.Where(u => u.correo == email).FirstOrDefault();

        if (user != null) {
            if (user.estado == 0)
            {
                try
                {
                    user.estado = 1;
                    db.Entry(user).State = System.Data.Entity.EntityState.Modified;
                    db.SaveChanges();
                    tipo = 1;
                    mensaje = "Registro completado exitosamente";
                }
                catch (Exception ex)
                {
                    mensaje = "Error al completar registro";
                }
            }
            else
            {
                tipo = 2;
            }
        }
        else
        {
            mensaje = "El registro no puede completarse";
        }
    }

        return Json(new { tipo = tipo, mensaje = mensaje }, JsonRequestBehavior.AllowGet);
```

```
}

/// <summary>
/// Envia el correo para cambiar la contraseña por olvido
/// </summary>
/// <param name="model">Modelo de la vista</param>
/// <returns>Json con tipo y mensaje</returns>
public JsonResult SendForgotPassword(ChangePasswordModel model)
{
   int tipo = 0;
   string mensaje = "";

   if (!ModelState.IsValid)
   {
      //Extrae el primer mensaje de error que tenga el modelo
      mensaje = ModelState.Values.Select(e => e.Errors).Where(e => e.Count > 0).FirstOrDefault().Select(v => v.ErrorMessage).FirstOrDefault();
   }
   else
   {
      using (var db = new VentaArticulosCreditoEntities())
      {
         var user = db.Usuario.Where(u => u.correo == model.email).FirstOrDefault();

         if(user != null)
         {
            MailController().SendResetPassword(user, ref mensaje, ref tipo);
         }
         else
         {
            mensaje = "Este correo no está registrado";
         }
      }
   }

   return Json(new { tipo = tipo, mensaje = mensaje }, JsonRequestBehavior.AllowGet);
}

/// <summary>
/// Retorna la vista de reset password
/// </summary>
/// <returns>Vista</returns>
[HttpGet]
public ActionResult ResetPassword(string token, string correo)
{
   if (Token.LeerToken(correo, token))
   {
      return View();
   }

   return RedirectToAction("Login", "Login");
}

/// <summary>
/// Restaura la contraseña del usuario
/// </summary>
/// <returns>Json con tipo y mensaje</returns>
[HttpPost, ValidateAntiForgeryToken]
public JsonResult ResetPassword(ResetPasswordViewModel model)
{
   string mensaje = "";
   int tipo = 0;

   if (!ModelState.IsValid)
   {
      //Extrae el primer mensaje de error que tenga el modelo
      mensaje = ModelState.Values.Select(e => e.Errors).Where(e => e.Count > 0).FirstOrDefault().Select(v => v.ErrorMessage).FirstOrDefault();
   }
   else
   {
      using (var db = new VentaArticulosCreditoEntities())
      {
         var user = db.Usuario.Where(u => u.correo == model.correo).FirstOrDefault();

         if (user != null)
         {
            try
            {
               user.password = Encrypt.Encripta(model.password);
               db.Entry(user).State = System.Data.Entity.EntityState.Modified;
               db.SaveChanges();
               Helpers.Token.ModificarToken(model.correo, model.token);
               tipo = 1;
               mensaje = "Se ha restaurado su contraseña correctamente";
            }
            catch (Exception ex)
            {
               mensaje = "Error al restaurar contraseña";
            }
         }
         else
```

```
            {
               mensaje = "El usuario ya no existe";
            }
          }
       }

       return Json(new { tipo = tipo, mensaje = mensaje }, JsonRequestBehavior.AllowGet);
    }

    /// <summary>
    /// Recarga la sesión del usuario que está logueado
    /// </summary>
    /// <param name="correo">Correo del usuario</param>
    /// <returns>Json con tipo</returns>
    [HttpPost]
    public JsonResult ReloadSession(string correo)
    {
       int tipo = 0;

       using (var db = new VentaArticulosCreditoEntities())
       {
          var user = db.Usuario.Where(u => u.correo == correo).FirstOrDefault();

          if(user != null)
          {
             tipo = 1;
             Session["usuario"] = user;
             Session["rolUsuario"] = user.codigoRol;
          }
       }

       return Json(new { tipo = tipo }, JsonRequestBehavior.AllowGet);
    }
  }
}
```

Clase MailController.cs

/*Controlador para enviar correos*/

```
using System;
using System.Collections.Generic;
using System.Configuration;
using System.Linq;
using System.Net.Mail;
using System.Web;
using VentaArticulosCredito.DBEntities;
using VentaArticulosCredito.Helpers;
using VentaArticulosCredito.Models.Login;

namespace VentaArticulosCredito.Controllers
{
    public class MailController : ControlController
    {
        /// <summary>
        /// Envía el correo de registro
        /// </summary>
        /// <param name="correo">Correo del usuario registrado</param>
        /// <returns>True o false si fue enviado o no</returns>
        public Boolean SendRegisterMail(RegisterViewModel model, ref string mensaje, ref int tipo)
        {
            Boolean enviado = false;
            string content = "";
            MailMessage mail = new MailMessage();

            try
            {
                content = System.IO.File.ReadAllText(RegisterTemplate());
                content = content.Replace("$$$_nombreUsuario", model.nombre + " " + model.apellido);
                content = content.Replace("$$$_link", URL_Register + "?email=" + model.email);
                mail.From = new MailAddress("cyber.shop.register.gt@gmail.com", "Cyber Shop");
                mail.To.Add(new MailAddress(model.email, model.nombre + " " + model.apellido));
                mail.Subject = "Registro Cyber Shop";
                mail.IsBodyHtml = true;
                mail.Body = content;

                ClienteSmtp().Send(mail);
                tipo = 1;
                mensaje = "Le hemos enviado un email para completar el registro";
            }
            catch (Exception ex)
            {
                mensaje = "Error al enviar correo";
            }

            return enviado;
        }

        /// <summary>
        /// Envía el correo con el link para restaurar la contraseña
        /// </summary>
        /// <param name="usuario">Usuario al que debe enviarse el correo</param>
        public void SendResetPassword(Usuario usuario, ref string mensaje, ref int tipo)
        {
```

```
    if (!Helpers.Token.LeerToken(usuario.correo))
    {
        string content = "";
        Guid token = Guid.NewGuid();
        MailMessage mail = new MailMessage();

        try
        {
            content = System.IO.File.ReadAllText(ResetPasswordTemplate());
            content = content.Replace("$$$_nombreUsuario", usuario.nombre + " " + usuario.apellido);
            content = content.Replace("$$$_link", URL_Reset + token.ToString().Replace("-", "_") + "&correo=" + usuario.correo);
            mail.From = new MailAddress("cyber.shop.register.gt@gmail.com", "Cyber Shop");
            mail.To.Add(new MailAddress(usuario.correo, usuario.nombre + " " + usuario.apellido));
            mail.Subject = "Restaurar Contraseña";
            mail.IsBodyHtml = true;
            mail.Body = content;

            ClienteSmtp().Send(mail);
            Token.GuardarToken(usuario.correo, token.ToString().Replace("-", "_"));
            tipo = 1;
            mensaje = "Le hemos enviado un email para restaurar su contraseña";
        }
        catch (Exception ex)
        {
            mensaje = "Error al enviar correo";
        }
    }
    else
    {
        tipo = 2;
        mensaje = "Ya le hemos enviado un correo";
    }
}

/// <summary>
/// Envía correo al confirmar orden
/// </summary>
/// <param name="usuario">Usuario</param>
/// <param name="tipo">Tipo mensaje</param>
/// <param name="mensaje">Mensaje</param>
public void SendConfirmOrder(Usuario usuario, ref int tipo, ref string mensaje, int orden)
{
    string content = "";
    MailMessage mail = new MailMessage();

    try
    {
        content = System.IO.File.ReadAllText(ConfirmOrderTemplate());
        content = content.Replace("$$$_nombreUsuario", usuario.nombre + " " + usuario.apellido);
        content = content.Replace("$$$_ordenCompra", orden.ToString());
        content = content.Replace("$$$_link", URL_ConfirmOrder + orden.ToString());
        mail.From = new MailAddress("cyber.shop.register.gt@gmail.com", "Cyber Shop");
        mail.To.Add(new MailAddress(usuario.correo, usuario.nombre + " " + usuario.apellido));
        mail.Subject = "Confirmación Orden";
        mail.IsBodyHtml = true;
        mail.Body = content;

        ClienteSmtp().Send(mail);
        tipo = 1;
        mensaje = "Le hemos enviado un email con la información de su orden";
    }
    catch (Exception ex)
    {
        mensaje = "Error al enviar correo";
    }
}

/// <summary>
/// Envía correo al cancelar orden
/// </summary>
/// <param name="usuario">Usuario</param>
/// <param name="tipo">Tipo mensaje</param>
/// <param name="mensaje">Mensaje</param>
public void SendCancelOrder(Usuario usuario, ref int tipo, ref string mensaje, int orden)
{
    string content = "";
    MailMessage mail = new MailMessage();

    try
    {
        content = System.IO.File.ReadAllText(CancelOrderTemplate());
        content = content.Replace("$$$_nombreUsuario", usuario.nombre + " " + usuario.apellido);
        content = content.Replace("$$$_ordenCompra", orden.ToString());
        content = content.Replace("$$$_link", URL_ConfirmOrder + orden.ToString());
        mail.From = new MailAddress("cyber.shop.register.gt@gmail.com", "Cyber Shop");
        mail.To.Add(new MailAddress(usuario.correo, usuario.nombre + " " + usuario.apellido));
        mail.Subject = "Cancelación Orden";
        mail.IsBodyHtml = true;
        mail.Body = content;

        ClienteSmtp().Send(mail);
        tipo = 1;
        mensaje = "Le hemos enviado un email con la información de su orden";
    }
```

```
            catch (Exception ex)
            {
                mensaje = "Error al enviar correo";
            }
        }
    }
}
```

Clase PerfilController.cs

/*maneja la vista del perfil para actualizar información agregar direcciones o cambiar contraseñas*/

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using VentaArticulosCredito.DBEntities;
using VentaArticulosCredito.Helpers;
using VentaArticulosCredito.Models.Perfil;

namespace VentaArticulosCredito.Controllers
{
  public class PerfilController : ControlController
  {
    /// <summary>
    /// Devuelve la vista del perfil
    /// </summary>
    /// <returns></returns>
    [HttpGet]
    public ActionResult Perfil()
    {
      if (ValidaUsuario()) {
        using (var db = new VentaArticulosCreditoEntities())
        {
          var usuario = (Usuario)Session["usuario"];
          ViewBag.ListaDepartamentos = db.Departamento.ToList();
          return View("PerfilView", new PerfilPageModel(db.Usuario.Include("Direccion_Usuario.Municipio.Departamento").Where(u => u.codigo == usuario.codigo).FirstOrDefault()));
        }
      }

      return RedirectToAction("Login", "Login");
    }

    /// <summary>
    /// Actualiza la información del usuario
    /// </summary>
    /// <param name="model">Model update</param>
    /// <returns>Json</returns>
    [HttpPost, ValidateAntiForgeryToken]
    public JsonResult UpdateInfo(UpdateInfoModel model)
    {
      int tipo = 0;
      string mensaje = "";
      string correo = "";
      Boolean validaNIT = false;
      Boolean guardar = true;
      Boolean validaCorreo = false;

      try
      {
        if (!ModelState.IsValid)
        {
          //Extrae el primer mensaje de error que tenga el modelo
          mensaje = ModelState.Values.Select(e => e.Errors).Where(e => e.Count > 0).FirstOrDefault().Select(v => v.ErrorMessage).FirstOrDefault();
        }
        else
        {
          var user = (Usuario)Session["usuario"];

          if (HuboCambios(user, model, ref validaNIT, ref validaCorreo))
          {
            using (var db = new VentaArticulosCreditoEntities())
            {
              if (validaNIT)
              {
                if (ValidarNIT.ValidaNIT(model.nit))
                {
                  var userNIT = db.Usuario.Where(u => u.nit == model.nit).FirstOrDefault();

                  if (userNIT == null)
                  {
                    user.nit = model.nit;
                  }
                  else
                  {
                    guardar = false;
                    mensaje = "Este NIT ya está siendo utilzado";
                  }
```

```
                }
                else
                {
                    guardar = false;
                    mensaje = "Ingrese un NIT válido";
                }
            }

            if (validaCorreo)
            {
                var userMail = db.Usuario.Where(u => u.correo == model.email).FirstOrDefault();

                if (userMail != null)
                {
                    guardar = false;
                    mensaje = "Este correo ya está siendo utilizado";
                }
            }

            if (guardar)
            {
                user.correo = model.email;
                user.nombre = model.nombre;
                user.apellido = model.apellido;
                user.fechaNacimiento = model.fechaNacimiento;
                db.Entry(user).State = System.Data.Entity.EntityState.Modified;
                db.SaveChanges();
                tipo = 1;
                mensaje = "Información actualizada correctamente";
                Session["usuario"] = user;
                correo = user.correo;
            }
        }
    }
    else
    {
        mensaje = "Asegurese de modificar al menos un registro";
    }
  }
  catch (Exception ex)
  {
    mensaje = "Error al cambiar contraseña";
  }

  return Json(new { tipo = tipo, mensaje = mensaje, correo = correo }, JsonRequestBehavior.AllowGet);
}

/// <summary>
/// Cambia la contraseña del usuario
/// </summary>
/// <returns>Json con tipo y mensaje</returns>
[HttpPost, ValidateAntiForgeryToken]
public JsonResult ChangePassword(ChangePasswordModel model)
{
  string mensaje = "";
  int tipo = 0;

  if (!ModelState.IsValid)
  {
    //Extrae el primer mensaje de error que tenga el modelo
    mensaje = ModelState.Values.Select(e => e.Errors).Where(e => e.Count > 0).FirstOrDefault().Select(v => v.ErrorMessage).FirstOrDefault();
  }
  else
  {
    using (var db = new VentaArticulosCreditoEntities())
    {
        var passEncrypt = Encrypt.Encripta(model.passwordActual);
        var user = (Usuario)Session["usuario"];
        var passUser = db.Usuario.Where(u => u.codigo == user.codigo && u.password == passEncrypt).FirstOrDefault();

        if(passUser != null)
        {
            try
            {
                passUser.password = Encrypt.Encripta(model.newPassword);
                db.Entry(passUser).State = System.Data.Entity.EntityState.Modified;
                db.SaveChanges();
                tipo = 1;
                mensaje = "Se ha cambiado su contraseña correctamente";
            }
            catch (Exception ex)
            {
                mensaje = "Error al cambiar contraseña";
            }
        }
        else
        {
            mensaje = "Contraseña incorrecta";
        }
```

```
        }
    }

    return Json(new { tipo = tipo, mensaje = mensaje }, JsonRequestBehavior.AllowGet);
}

/// <summary>
/// Devuelve los municipios del departamento seleccionado
/// </summary>
/// <param name="departamento"></param>
/// <returns></returns>
[HttpPost]
public JsonResult GetMunicipios(int departamento)
{
    int tipo = 0;
    string mensaje = "";
    SelectList listaMunicipios = null;

    using (var db = new VentaArticulosCreditoEntities())
    {
        try
        {
            listaMunicipios = new SelectList(db.Municipio.Where(m => m.codigoDepartamento == departamento).ToList(), "codigo", "nombre");
            tipo = 1;
        }catch(Exception ex)
        {
            mensaje = "Error al recuperar municipios";
        }
    }

    return Json(new { tipo = tipo, mensaje = mensaje, listaMunicipios = listaMunicipios }, JsonRequestBehavior.AllowGet);
}

/// <summary>
/// Crea la dirección del usuario
/// </summary>
/// <param name="model"></param>
/// <returns></returns>
public JsonResult CreateDirection(CreateDirectionModel model)
{
    int tipo = 0;
    string mensaje = "";

    if (!ModelState.IsValid)
    {
        //Extrae el primer mensaje de error que tenga el modelo
        mensaje = ModelState.Values.Select(e => e.Errors).Where(e => e.Count > 0).FirstOrDefault().Select(v => v.ErrorMessage).FirstOrDefault();
    }
    else
    {
        using (var db = new VentaArticulosCreditoEntities())
        {
            try
            {
                var user = (Usuario)Session["usuario"];
                var direccion = new Direccion_Usuario(user.codigo, Convert.ToInt32(model.telefono), model.direccion, model.municipio, model.nombre, model.apellido);

                db.Direccion_Usuario.Add(direccion);
                db.SaveChanges();
                tipo = 1;
                mensaje = "Dirección agregada correctamente";
            }catch(Exception ex)
            {
                mensaje = "Error al crear dirección";
            }

        }
    }

    return Json(new { tipo = tipo, mensaje = mensaje }, JsonRequestBehavior.AllowGet);
}

/// <summary>
/// Devuelve la dirección a editar
/// </summary>
/// <param name="direccion"></param>
/// <returns></returns>
public ActionResult EditDirectionPartialView(int direccion)
{
    PerfilPageModel model = new PerfilPageModel();

    using (var db = new VentaArticulosCreditoEntities())
    {
        model.direccion = db.Direccion_Usuario.Include("Municipio.Departamento").Where(d => d.codigo == direccion).FirstOrDefault();
        ViewBag.ListaDepartamentos = db.Departamento.ToList();
        ViewBag.ListaMunicipios = db.Municipio.Where(m => m.codigoDepartamento == model.direccion.Municipio.Departamento.codigo).ToList();
    }

    return PartialView("EditDirectionPartialView", model);
}
```

```
    /// <summary>
    /// Edita una dirección
    /// </summary>
    /// <param name="model"></param>
    /// <returns></returns>
    [HttpPost, ValidateAntiForgeryToken]
    public JsonResult EditDirection(EditDirectionModel model)
    {
        int tipo = 0;
        string mensaje = "";

        try
        {
            if (!ModelState.IsValid)
            {
                //Extrae el primer mensaje de error que tenga el modelo
                mensaje = ModelState.Values.Select(e => e.Errors).Where(e => e.Count > 0).FirstOrDefault().Select(v => v.ErrorMessage).FirstOrDefault();
            }
            else
            {
                using (var db = new VentaArticulosCreditoEntities())
                {
                    var user = (Usuario)Session["usuario"];
                    var oldDirec = db.Direccion_Usuario.Include("Municipio.Departamento").Where(d => d.codigo == model.codigo).FirstOrDefault();

                    if (HuboCambios(oldDirec, model))
                    {
                        oldDirec.nombre = model.nombre;
                        oldDirec.apellido = model.apellido;
                        oldDirec.telefono = Convert.ToInt32(model.telefono);
                        oldDirec.direccion = model.direccion;
                        oldDirec.codigoMunicipio = model.municipio;

                        db.Entry(oldDirec).State = System.Data.Entity.EntityState.Modified;
                        db.SaveChanges();
                        tipo = 1;
                        mensaje = "Dirección modificada correctamente";
                    }
                    else
                    {
                        mensaje = "Asegurese de modificar al menos un registro";
                    }
                }
            }
        }
        catch (Exception ex)
        {
            mensaje = "Error al cambiar contraseña";
        }

        return Json(new { tipo = tipo, mensaje = mensaje }, JsonRequestBehavior.AllowGet);
    }

    /// <summary>
    /// Elimina la dirección seleccionada
    /// </summary>
    /// <param name="direccion">Dirección</param>
    /// <returns><Json/returns>
    [HttpPost]
    public JsonResult DeleteDirection(int direccion)
    {
        int tipo = 0;
        string mensaje = "";

        using (var db = new VentaArticulosCreditoEntities())
        {
            try
            {
                var direct = db.Direccion_Usuario.Where(d => d.codigo == direccion).FirstOrDefault();

                db.Direccion_Usuario.Remove(direct);
                db.SaveChanges();
                tipo = 1;
                mensaje = "Dirección eliminada correctamente";
            }catch(Exception ex)
            {
                mensaje = "Error al eliminar dirección";
            }
        }

        return Json(new { tipo = tipo, mensaje = mensaje }, JsonRequestBehavior.AllowGet);
    }
  }
}
```

Diagrama entidad relación Base de datos