Universidad ORT Uruguay
Facultad de Ingeniería
Escuela de Tecnología

OBLIGATORIO PROGRAMACION 3
DOCUMENTACIÓN

Matías Poletti - 251602 **Federico Porta - 233276** **Fernando Ugarte - 175433**

Grupo: N3B

Docente: Plinio Gañi

Analista en Tecnologías de la Información

Fecha de entrega del documento: 14/10/2020

CONTENIDO

Contenido

REQUERIMIENTOS FUNCIONALES

RF-01-Registro de Usuario:

Registro de los datos del nuevo usuario para ingresarlo en la base de datos luego de cumplidas las validaciones correspondientes.

RF-02-Login:

Al ingresar con su cedula y contraseña correspondiente el usuario tiene acceso a las funcionalidades que le competen.

RF-03-Presentacion del proyecto:

Un usuario Solicitante puede pedir un préstamo ingresando los datos requeridos correspondientes de su proyecto

agregando además el monto solicitado y cantidad de cuotas para que de esta forma y según los estándares de la empresa pueda ser Aprobado o Rechazado.

La tasa de interés que ofrece la empresa varia tanto por tipo de proyecto como por la cantidad de cuotas.

RF-04 – Visualizar mis proyectos (Usuario Solicitante).

Se genera un listado de los proyectos del Solicitante logueado de los cuales puede verse no sólo los datos ingresados sino también el monto a pagar (monto solicitado más el interés generado) y su estado de transacción ("Pendiente de revisión", "Aprobado", "Rechazado")

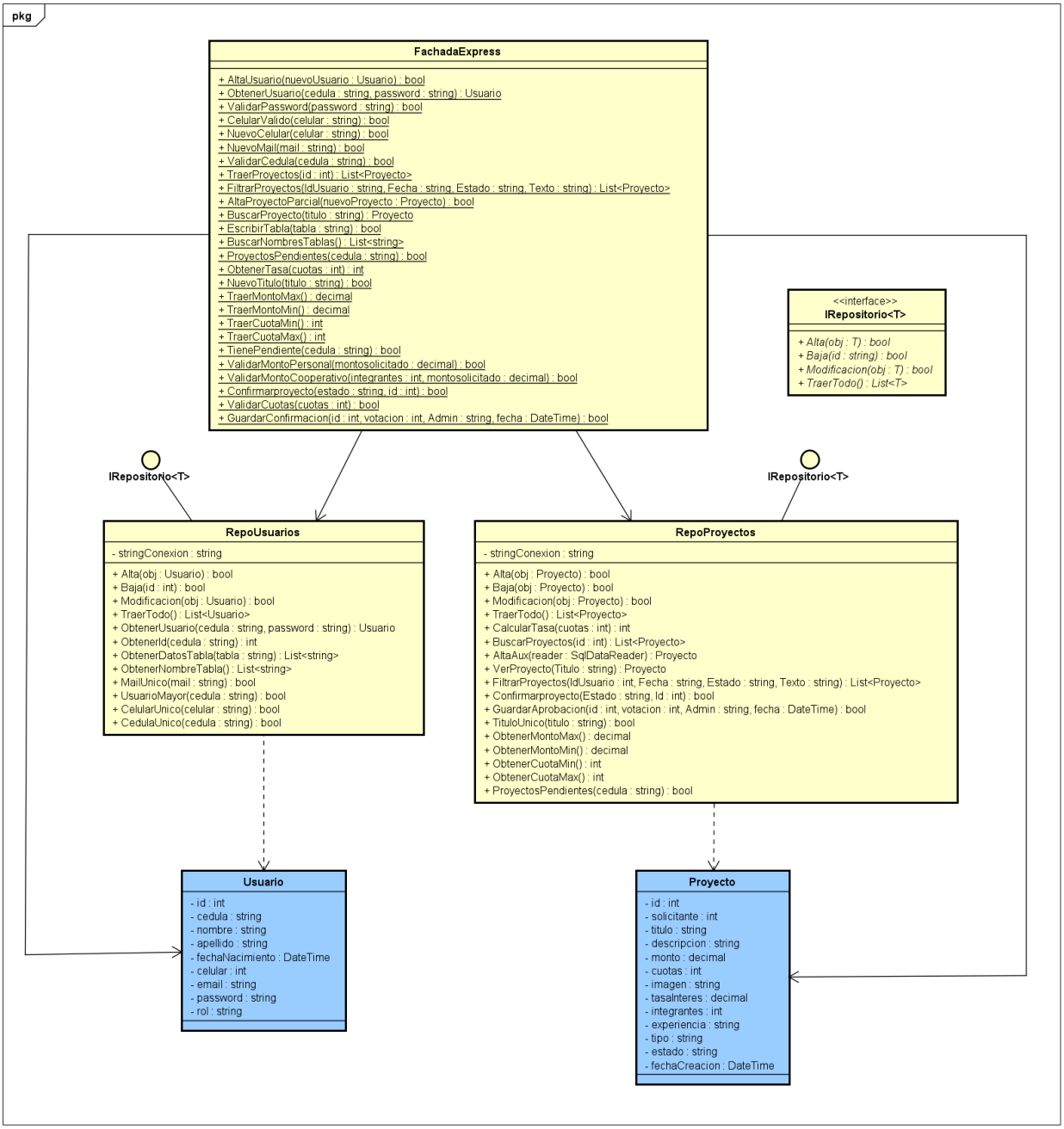
RF-05 – Aprobar/Rechaza proyecto (Usuario Admin)

El usuario Admin tiene la opción de acceder a todas las solicitudes que recibió la empresa y filtrar las mismas según su necesidad (por solicitante, por fecha, por estado de transacción y también por palabra clave). De esta manera puede, en caso de que el proyecto se encuentre en estado "Pendiente de Revisión", calificar el proyecto y de esta manera cambiar su estado según el puntaje dictado.

RF-06 – Exportar información en archivo de texto delimitado (Usuario Admin).

Seleccionando el nombre de la tabla, se exportará desde la base de datos en un archivo de texto delimitado para su fácil acceso.

DIAGRAMA DE CLASES UML



CODIGO

DOMINIO

Usuario

```
namespace Dominio
{
    public class Usuario
    {
        public int Id { get; set; }
        //[Required][MaxLength(10, ErrorMessage ="El nombre no puede tener más de 10 caracteres")]
        public string Cedula { get; set; }
        public string Nombre { get; set; }
        public string Apellido { get; set; }
        public DateTime FechaNacimiento { get; set; }
        public string Celular { get; set; }
        public string Email { get; set; }
        public string Password { get; set; }
        public string Rol { get; set; }

    }
}
```

Proyecto

```
namespace Dominio
{
    public class Proyecto
    {
        public int Id { get; set; }
        public int Solicitante { get; set; }
        public string Titulo { get; set; }
        public string Descripcion { get; set; }
        public decimal Monto { get; set; }
        public int Cuotas { get; set; }
        public string Imagen { get; set; } //COMO ES EL FORMATO?
        public decimal TasaInteres { get; set; }
        public int Integrantes { get; set; }
        public string Experiencia { get; set; }
        public string Tipo { get; set; }
        public string Estado { get; set; }
        public DateTime FechaCreacion { get; set; }

    }
}
```

REPOSITORIOS

FachadaExpress

```
namespace Repositorios
{
    public static class FachadaExpress
```

```
    {
        #region USUARIO

        public static bool AltaUsuario(Usuario nuevoUsuario)
        {
            IRepositorio<Usuario> repoUsu = new RepoUsuarios();

            bool ret = false;
            Usuario usu = new Usuario()
            {
                Cedula = nuevoUsuario.Cedula,
                Nombre = nuevoUsuario.Nombre,
                Apellido = nuevoUsuario.Apellido,
                FechaNacimiento = nuevoUsuario.FechaNacimiento,
                Celular = nuevoUsuario.Celular,
                Email = nuevoUsuario.Email,
                Password = nuevoUsuario.Password,
                Rol = nuevoUsuario.Rol
            };
            if (usu.Password.Length > 6 && ValidarPassword(usu.Password) &&
CelularValido(usu.Celular))
            {
                ret = repoUsu.Alta(usu);
            }

            return ret;
        } // el alta de usuario

        public static Usuario ObtenerUsuario(string cedula, string password)

        {

            RepoUsuarios repoUsu = new RepoUsuarios();
            Usuario usu = new Usuario();

            usu = repoUsu.ObtenerUsuario(cedula, password);
            return usu;


        } // el login

        public static bool ValidarPassword(string password)
        {
            bool esValido = false;

            if (password.Length >= 6)
            {

                bool Mayuscula = false;
                bool Minuscula = false;
                bool Numero = false;

                int i = 0;

                while (!esValido && i < password.Length)
                {
                    char charActual = password[i];
```

```
            if (charActual == char.ToUpper(charActual))
            {
                Mayuscula = true;
            }
            if (charActual == char.ToLower(charActual))
            {
                Minuscula = true;
            }
            if (Char.IsNumber(charActual))
            {
                Numero = true;
            }

            if (Mayuscula && Minuscula && Numero)
            {
                esValido = true;
            }
            i++;
        }
    }
    return esValido;
}

public static bool CelularValido(string celular)
{
    bool valido = false;
    char ch = celular[2];
    int intVal = (int)Char.GetNumericValue(ch);
    if (celular[0] == '0' && celular[1] == '9' && intVal > 0)
    {
        valido = true;
    }
    return valido;
}

public static bool NuevoCelular(string celular)
{
    bool valido = false;
    bool correcto = CelularValido(celular);
    RepoUsuarios usu = new RepoUsuarios();
    bool unico = usu.CelularUnico(celular);
    if (correcto && unico)
    {
        valido = true;
    }
    return valido;
}

public static bool NuevoMail(string mail)
{
    bool valido = false;
    RepoUsuarios usu = new RepoUsuarios();
    valido = usu.MailUnico(mail);
    return valido;
}

public static bool ValidarCedula(string cedula)
{
```

```
            bool unica = false;
            RepoUsuarios usu = new RepoUsuarios();
            unica = usu.CedulaUnico(cedula);
            return unica;
        }
        #endregion


        public static List<Proyecto> TraerProyectos(int id)
        {

            RepoProyectos repoPro = new RepoProyectos();

            return repoPro.BuscarProyectos(id);

        } // trae todos los proyectos del USUARIO

        public static List<Proyecto> FiltrarProyectos(string IdUsuario, string Fecha,
string Estado, string Texto)
        {
           RepoUsuarios usuario = new RepoUsuarios();
            int idUsuario = usuario.ObtenerId(IdUsuario);
            RepoProyectos repoPro = new RepoProyectos();

            return repoPro.FiltrarProyectos(idUsuario, Fecha, Estado, Texto);
        } // ADMIN puede filtrar y hacer un listado de proyectos

       // ALTA DE PROYECTO
        public static bool AltaProyectoParcial(Proyecto nuevoProyecto)
        {

            {
                RepoProyectos RepoProyecto = new RepoProyectos();
               int tasa = RepoProyecto.CalcularTasa(nuevoProyecto.Cuotas);
                bool ret = false;
                Proyecto nuevo = new Proyecto()
                {
                    Solicitante = nuevoProyecto.Solicitante,
                    Titulo = nuevoProyecto.Titulo,
                    Descripcion = nuevoProyecto.Descripcion,
                    Monto = nuevoProyecto.Monto,
                    Cuotas = nuevoProyecto.Cuotas,
                    Imagen = nuevoProyecto.Imagen,
                    TasaInteres = nuevoProyecto.TasaInteres,
                    Integrantes = nuevoProyecto.Integrantes,
                    Experiencia = nuevoProyecto.Experiencia,
                    Tipo = nuevoProyecto.Tipo,
                    Estado = nuevoProyecto.Estado,
                    FechaCreacion = nuevoProyecto.FechaCreacion


                };// como hacemos el alta de cada proyecto



                ret = RepoProyecto.Alta(nuevo);

                return ret;
```

```
            }
        } // el controlador llama a este metodo para dar de alta un proyecto, ya sea COOP
o PERSONAL.

public static Proyecto buscarProyecto(string titulo)
        {
            RepoProyectos repo = new RepoProyectos();
            Proyecto pro = repo.VerProyecto(titulo);

            return pro;
        } // es cuando el admin entra para ver "detalladamente" un proyecto



        #region TEXTO
        public static bool EscribirTabla(string tabla)//Recibe un String con el nombre de
la tabla, retorna un bool cuando termina
        {
            bool termino = true;
            RepoUsuarios usu = new RepoUsuarios();
            List<string> datos = usu.ObtenerDatosTabla(tabla);
            string rutaAplicacion = AppDomain.CurrentDomain.BaseDirectory;
            string nombreArchivo = tabla + ".txt";
            string rutaCompleta = Path.Combine(rutaAplicacion, nombreArchivo);
            FileStream fs = new FileStream(rutaCompleta, FileMode.Create);
            StreamWriter sw = new StreamWriter(fs);
            foreach (string linea in datos)
            {
                sw.Write(linea);
            }
            sw.Close();
            return termino;
        }
        public static List<string> BuscarNombresTablas()
        {
            RepoUsuarios usu = new RepoUsuarios();
            List<string> nombres = usu.ObtenerNombreTabla();
            return nombres;
        }
        #endregion




        #region Validaciones

        public static bool ProyectosPendientes (string cedula)
        {
            RepoProyectos RepoProyecto = new RepoProyectos();

            return RepoProyecto.ProyectosPendientes(cedula);
        }
        public static int ObtenerTasa(int cuotas) {
            int tasa;
            RepoProyectos RepoProyecto = new RepoProyectos();
        tasa = RepoProyecto.CalcularTasa(cuotas);
```

```
            return tasa;
        }

        public static bool NuevoTitulo(string titulo)
        {
            bool valido = false;
            RepoProyectos pro = new RepoProyectos();
            valido = pro.TituloUnico(titulo);
            return valido;
        }
        public static decimal TraerMontoMax()
        {
            decimal max = 0;
            RepoProyectos pro = new RepoProyectos();
            max = pro.ObtenerMontoMax();
            return max;
        }
        public static decimal TraerMontoMin()
        {
            decimal min = 0;
            RepoProyectos pro = new RepoProyectos();
            min = pro.ObtenerMontoMin();
            return min;
        }
        public static int TraerCuotaMin()
        {
            int min = 0;
            RepoProyectos pro = new RepoProyectos();
            min = pro.ObtenerCuotaMin();
            return min;
        }
        public static int TraerCuotaMax()
        {
            int max = 0;
            RepoProyectos pro = new RepoProyectos();
            max = pro.ObtenerCuotaMax();
            return max;
        }
        public static bool TienePendiente(string cedula)
        {
            bool pendiente = false;
            RepoProyectos pro = new RepoProyectos();
            pendiente = pro.ProyectosPendientes(cedula);
            return pendiente;
        }
        public static bool ValidarMontoPersonal(decimal montosolicitado)
        {
            int bono = 20;

            decimal maximo = TraerMontoMax();
            decimal montoMax = maximo - (maximo * bono) / 100;

            return montosolicitado < montoMax && montosolicitado > TraerMontoMin();
        }
        public static bool ValidarMontoCooperativo(int integrantes, decimal
montosolicitado)
        {
            int bono = 20;
```

```
            if (integrantes <= 10)
            {
                bono = 2 * integrantes;

            }
            decimal montoMax = TraerMontoMax();
            decimal montoMin = TraerMontoMin();
            decimal nuevoMontoMax = montoMax + (montoMax * bono) / 100;
            return montosolicitado < nuevoMontoMax && montosolicitado > montoMin;

        }

        public static bool Confirmarproyecto(string estado, int id)
        {
            RepoProyectos repo = new RepoProyectos();
            return repo.Confirmarproyecto(estado,id);
        }
        public static bool ValidarCuotas(int cuotas)
        {
            return cuotas > TraerCuotaMin() && cuotas < TraerCuotaMax();
        }

        public static bool Guardarconfirmacion(int id, int votacion, string Admin,
DateTime fecha)
        {
            RepoProyectos repo = new RepoProyectos();

            return repo.GuardarAprobacion(id, votacion, Admin, fecha);
        }
        #endregion
    }
}
```

IRepositorio

```
namespace Repositorios
{
    interface IRepositorio<T>
    {
        bool Alta(T obj);
        bool Baja(int id);
        bool Modificacion(T obj);
        List<T> TraerTodo();


    }
}
```

RepoProyecto

```
namespace Repositorios
{
    class RepoProyectos : IRepositorio<Proyecto>
    {
```

```
        public string StringConexion { get; set; } =
ConfigurationManager.ConnectionStrings["MiConexion"].ConnectionString;


        public bool Alta(Proyecto obj)
        {
            bool ret = false;



            SqlConnection con = new SqlConnection(this.StringConexion);
            string imagenBD = obj.Titulo + ".jpg";
            string sql = "insert into Proyecto(Solicitante, Titulo, Fecha_Creacion,
Descripcion, Monto_Solicitado, Cuotas, Imagen, Tasa_Interes, Estado, Integrantes,
Experiencia, Tipo) values(@sol, @tit, @fecre, @des, @mon, @cuo, @img, @tas, @est, @inte,
@exp, @tipo);";
            SqlCommand com = new SqlCommand(sql, con);
            com.Parameters.AddWithValue("@sol", obj.Solicitante);
            com.Parameters.AddWithValue("@tit", obj.Titulo);
            com.Parameters.AddWithValue("@fecre", obj.FechaCreacion);
            com.Parameters.AddWithValue("@des", obj.Descripcion);
            com.Parameters.AddWithValue("@mon", obj.Monto);
            com.Parameters.AddWithValue("@cuo", obj.Cuotas);
            com.Parameters.AddWithValue("@img", imagenBD);
            com.Parameters.AddWithValue("@tas", obj.TasaInteres);
            com.Parameters.AddWithValue("@est", obj.Estado);
            com.Parameters.AddWithValue("@inte", obj.Integrantes);
            com.Parameters.AddWithValue("@exp", obj.Experiencia);
            com.Parameters.AddWithValue("@tipo", obj.Tipo);


            try
            {
                con.Open();
                int afectadas = com.ExecuteNonQuery();
                con.Close();

                ret = afectadas == 1;
            }
            catch
            {
                throw;
            }
            finally
            {
                if (con.State == ConnectionState.Open) con.Close();
            }

            return ret;
        }

        public bool ProyectosPendientes(string cedula)  //Si es true => tiene Pendientes
        {
            bool pendiente = false;
            SqlConnection con = new SqlConnection(this.StringConexion);
            string sql = "SELECT * FROM Usuario u, Proyecto p WHERE u.Cedula = @cedula
AND u.UsuarioId = p.Solicitante AND p.Estado = 'Pendiente de revisión'";
```

```
            SqlCommand com = new SqlCommand(sql, con);
            com.Parameters.AddWithValue("@cedula", cedula);
            try
            {
                con.Open();
                SqlDataReader reader = com.ExecuteReader();
                if (reader.HasRows)
                {
                    pendiente = true;
                }
            }
            catch
            {
                throw;
            }
            finally
            {
                if (con.State == ConnectionState.Open) con.Close();
            }
            return pendiente;
        }


        public int CalcularTasa(int Cuotas)
        {
            int tasa =0;

            SqlConnection con = new SqlConnection(this.StringConexion);

            string sql = " SELECT Interes FROM Tasa_Interes WHERE @cuotas >= Desde AND
@cuotas <= Hasta";
            SqlCommand com = new SqlCommand(sql, con);
           com.Parameters.AddWithValue("@cuotas", Cuotas);
            try
            {
                con.Open();
                SqlDataReader reader = com.ExecuteReader();

                while (reader.Read())
                {
                    string t = reader.GetValue(0).ToString();
                    tasa = int.Parse(t);

                }

                    con.Close();


            }
            catch
            {
                throw;
            }
            finally
            {
                if (con.State == ConnectionState.Open) con.Close();
            }
```

```
    return tasa;
}


public List<Proyecto> BuscarProyectos (int id)
{
    List<Proyecto> aux = new List<Proyecto>();

    SqlConnection con = new SqlConnection(this.StringConexion);

    string sql = "select * from Proyecto WHERE Solicitante = @id;";

    SqlCommand com = new SqlCommand(sql, con);
    com.Parameters.AddWithValue("@id", id);


    try
    {
        con.Open();
        SqlDataReader reader = com.ExecuteReader();

        while (reader.Read())
        {
            Proyecto pro = AltaAux(reader);
            aux.Add(pro);
        }

        con.Close();
    }
    catch
    {
        con.Close();
        throw;
    }
    finally
    {
        if (con.State == ConnectionState.Open) con.Close();
    }

        return aux;
}

private Proyecto AltaAux(SqlDataReader reader)
{
    string monto = reader["Monto_Solicitado"].ToString();
    string cuotas = reader["Cuotas"].ToString();
    string tasa = reader["Tasa_Interes"].ToString();
    string integrantes = reader["Integrantes"].ToString();
    string solicitante = reader["Solicitante"].ToString();
    string id = reader["ProyectoId"].ToString();
    DateTime nueva = Convert.ToDateTime(reader["Fecha_Creacion"].ToString());

    Proyecto pro = new Proyecto();

    pro.Id = int.Parse(id);
    pro.Titulo = reader["Titulo"].ToString();
    pro.Descripcion = reader["Descripcion"].ToString();
    pro.Imagen = reader["Imagen"].ToString();
```

```
            pro.Estado = reader["Estado"].ToString();
            pro.Monto = decimal.Parse(monto);
            pro.Cuotas = int.Parse(cuotas);
            pro.FechaCreacion = nueva;

            //Estado = reader.GetString(3),
            pro.Experiencia = reader["Experiencia"].ToString();
            pro.Integrantes = int.Parse(integrantes);
            pro.Solicitante = int.Parse(solicitante);
            pro.Tipo = reader["Tipo"].ToString();
            pro.TasaInteres = decimal.Parse(tasa);


            return pro;
        }

        public Proyecto VerProyecto(string Titulo)
        {
            SqlConnection con = new SqlConnection(this.StringConexion);

            string sql = "SELECT * FROM Proyecto WHERE Titulo = @tit ;";
            SqlCommand com = new SqlCommand(sql, con);
            com.Parameters.AddWithValue("@tit", Titulo);
            Proyecto pro = new Proyecto();

            try
            {
                con.Open();
                SqlDataReader reader = com.ExecuteReader();

                while (reader.Read())
                {
                   pro = AltaAux(reader);

                }
                con.Close();


            }
            catch
            {
                throw;
            }
            finally
            {
                if (con.State == ConnectionState.Open) con.Close();
            }

            return pro;

    }
        // ADMIN


        public List<Proyecto> FiltrarProyectos(int IdUsuario, string Fecha, string
Estado, string Texto)
        {
```

```
            List<Proyecto> aux = new List<Proyecto>();
            SqlConnection con = new SqlConnection(this.StringConexion);
            string where = "WHERE ";
            if(IdUsuario == 0 && Fecha == "" && Estado == "" && Texto == "")
            {
                where = "";
            }
            string c = "";
            if (IdUsuario != 0)
            {
                c = "Solicitante = "+IdUsuario;
            }

            if (Fecha != "")
            {   if(c != "") { c += " AND "; };
                c += "Fecha_Creacion = '"+Fecha+"'";
            }
            if (Estado != "")
            {
                if (c != "") { c += " AND "; };
                c += "Estado = '" + Estado + "'";
            }
            if (Texto != "")
            {
                if (c != "") { c += " AND "; };
                c += "Titulo LIKE '%" + Texto+ "%' OR Descripcion LIKE '%" + Texto +
"%'";
            }
            if (Fecha != "")
            {
                c += "order by Convert(DATE, Fecha_Creacion)";
            }


            string sql = " SELECT * FROM Proyecto "+where + c;

            SqlCommand com = new SqlCommand(sql, con);


            try
            {
                con.Open();
                SqlDataReader reader = com.ExecuteReader();

                while (reader.Read())
                {
                    Proyecto pro = AltaAux(reader);
                    aux.Add(pro);
                }

                con.Close();
            }
            catch
            {
                con.Close();
                throw;
            }
            finally
```

```
            {
                if (con.State == ConnectionState.Open) con.Close();
            }

            return aux;
        }

        public bool Confirmarproyecto(string Estado, int Id)
        {
            bool ret = false;
            SqlConnection con = new SqlConnection(this.StringConexion);
            string sql = "UPDATE Proyecto SET Estado = '"+Estado+"' WHERE ProyectoId =
@Id";
            SqlCommand com = new SqlCommand(sql, con);
            com.Parameters.AddWithValue("@Id", Id);
            try
            {

                    con.Open();
                    int afectadas = com.ExecuteNonQuery();
                    con.Close();

                    ret = afectadas == 1;

            }
            catch
            {
                con.Close();
                throw;
            }
            finally
            {
                if (con.State == ConnectionState.Open) con.Close();
            }
            return ret;
        }

        public bool GuardarAprobacion (int id, int votacion, string Admin, DateTime
fecha)
        {
            bool ret = false;
            SqlConnection con = new SqlConnection(this.StringConexion);
            string sql = "insert into Aprobacion(ProyectoId, Puntaje, Administrador,
Fecha_Aprobacion) values(@id, @vot, @adm, @fecha);";
            SqlCommand com = new SqlCommand(sql, con);
            com.Parameters.AddWithValue("@id", id);
            com.Parameters.AddWithValue("@vot", votacion);
            com.Parameters.AddWithValue("@adm", Admin);
            com.Parameters.AddWithValue("@fecha", fecha);
            try
            {
                con.Open();
                int afectadas = com.ExecuteNonQuery();
                con.Close();

                ret = afectadas == 1;
            }
            catch
```

```
            {
                throw;
            }
            finally
            {
                if (con.State == ConnectionState.Open) con.Close();
            }
            return ret;
        }


        public bool TituloUnico(string titulo) //Consulta Si existe Titulo ingresado en
la columna Titulo de tabla Proyecto
        {
            bool unico = false;
            SqlConnection con = new SqlConnection(this.StringConexion);
            string sql = "SELECT Titulo FROM Proyecto WHERE Titulo = @titulo";
            SqlCommand com = new SqlCommand(sql, con);
            com.Parameters.AddWithValue("@titulo", titulo);
            try
            {
                con.Open();
                SqlDataReader reader = com.ExecuteReader();
                if (!reader.HasRows)
                {
                    unico = true;
                }
            }
            catch
            {
                throw;
            }
            finally
            {
                if (con.State == ConnectionState.Open) con.Close();
            }
            return unico;
        }

        #region Validar Montos // Monto max/min y comparacion con monto ingresado
        public decimal ObtenerMontoMax() //Consulta sql trae monto maximo
        {
            decimal max = 0;
            SqlConnection con = new SqlConnection(this.StringConexion);
            string sql = "SELECT Monto_Max FROM Financiamiento";
            SqlCommand com = new SqlCommand(sql, con);
            try
            {
                con.Open();
                SqlDataReader reader = com.ExecuteReader();
                while (reader.Read())
                {
                    string t = reader.GetValue(0).ToString();
                    max = decimal.Parse(t);
                }
                con.Close();
            }
```

```
        catch
        {
            throw;
        }
        finally
        {
            if (con.State == ConnectionState.Open) con.Close();
        }
        return max;
    }
    public decimal ObtenerMontoMin() //Consulta sql trae monto minimo
    {
        decimal min = 0;
        SqlConnection con = new SqlConnection(this.StringConexion);
        string sql = "SELECT Monto_Min FROM Financiamiento";
        SqlCommand com = new SqlCommand(sql, con);
        try
        {
            con.Open();
            SqlDataReader reader = com.ExecuteReader();
            while (reader.Read())
            {
                string t = reader.GetValue(0).ToString();
                min = decimal.Parse(t);
            }
            con.Close();
        }
        catch
        {
            throw;
        }
        finally
        {
            if (con.State == ConnectionState.Open) con.Close();
        }
        return min;
    }

    #endregion


    #region Validar Cuotas // rango cuota min/max y comparacion con cuotas ingresadas
    public int ObtenerCuotaMin() // Primer valor en columna Desde
    {
        int min = 0;
        List<int> aux = new List<int>();
        SqlConnection con = new SqlConnection(this.StringConexion);
        string sql = "SELECT Desde FROM Tasa_Interes";
        SqlCommand com = new SqlCommand(sql, con);
        try
        {
            con.Open();
            SqlDataReader reader = com.ExecuteReader();
            while (reader.Read())
            {
                string t = reader.GetValue(0).ToString();
                int valor = int.Parse(t);
                aux.Add(valor);
```

```
        }
        con.Close();
    }
    catch
    {
        con.Close();
        throw;
    }
    finally
    {
        if (con.State == ConnectionState.Open) con.Close();
    }
    min = aux[0];
    return min;
}
public int ObtenerCuotaMax() // Ultimo valor en la columna Hasta
{
    int max = 0;
    List<int> aux = new List<int>();
    SqlConnection con = new SqlConnection(this.StringConexion);
    string sql = "SELECT Hasta FROM Tasa_Interes";
    SqlCommand com = new SqlCommand(sql, con);
    try
    {
        con.Open();
        SqlDataReader reader = com.ExecuteReader();
        while (reader.Read())
        {
            string t = reader.GetValue(0).ToString();
            int valor = int.Parse(t);
            aux.Add(valor);
        }
        con.Close();
    }
    catch
    {
        con.Close();
        throw;
    }
    finally
    {
        if (con.State == ConnectionState.Open) con.Close();
    }
    max = aux[aux.Count - 1];
    return max;
}

#endregion
```

```
        public bool Baja(int id)
        {
            throw new NotImplementedException();
        }

        public bool Modificacion(Proyecto obj)
        {
            throw new NotImplementedException();
        }

        public List<Proyecto> TraerTodo()
        {
            throw new NotImplementedException();
        }


    }
}
```

RepoUsuario

```
namespace Repositorios
{
    public class RepoUsuarios : IRepositorio<Usuario>
    {
        public string StringConexion { get; set; } =
ConfigurationManager.ConnectionStrings["MiConexion"].ConnectionString;

        public bool Alta(Usuario obj)
        {
            bool ret = false;

            SqlConnection con = new SqlConnection(this.StringConexion);

            string sql = "insert into Usuario(Cedula, Nombre, Apellido, Pass,
Fecha_Nacimiento, Celular, Email, Rol) values(@ced, @nom, @ape, @pass, @nac, @cel, @mail,
@rol);";
            SqlCommand com = new SqlCommand(sql, con);
            com.Parameters.AddWithValue("@ced", obj.Cedula);
            com.Parameters.AddWithValue("@nom", obj.Nombre);
            com.Parameters.AddWithValue("@ape", obj.Apellido);
            com.Parameters.AddWithValue("@pass", obj.Password);
            com.Parameters.AddWithValue("@nac", obj.FechaNacimiento);
            com.Parameters.AddWithValue("@cel", obj.Celular);
            com.Parameters.AddWithValue("@mail", obj.Email);
            com.Parameters.AddWithValue("@rol", obj.Rol);
```

```
            try
            {
                con.Open();
                int afectadas = com.ExecuteNonQuery();
                con.Close();

                ret = afectadas == 1;
            }
            catch
            {
                con.Close();
                throw;
            }
            finally
            {
                if (con.State == ConnectionState.Open) con.Close();
            }

            return ret;
        }

        public Usuario ObtenerUsuario(string cedula, string password)
        {
            Usuario ret = new Usuario();

            SqlConnection con = new SqlConnection(this.StringConexion);

            string sql = " SELECT * FROM Usuario WHERE @cedula = Cedula AND @pass =
Pass";
            SqlCommand com = new SqlCommand(sql, con);
            com.Parameters.AddWithValue("@cedula", cedula);
            com.Parameters.AddWithValue("@pass", password);

            try
            {
                con.Open();
                SqlDataReader reader = com.ExecuteReader();

                while (reader.Read())
                {
                    DateTime fecha = reader.GetDateTime(5);


                    int id = int.Parse(reader["UsuarioId"].ToString());

                        ret.Id = id;
                        ret.Cedula = reader["Cedula"].ToString();
                        ret.Nombre = reader["Nombre"].ToString();
                        ret.Apellido = reader["Apellido"].ToString();
                        ret.FechaNacimiento = fecha;
                        ret.Celular = reader["Celular"].ToString(); ;
                        ret.Email = reader["Email"].ToString();
                        ret.Password = reader["Pass"].ToString();
                        ret.Rol = reader["Rol"].ToString();
```

```
                // Console.WriteLine("Nombre:  "+ret.Nombre +"    Id:  "+
reader["Id"].ToString());

                }

                con.Close();

            }
            catch
            {
                throw;
            }
            finally
            {
                if (con.State == ConnectionState.Open) con.Close();
            }

            return ret;
        }



        public int ObtenerId(string cedula)
        {
            int idusuario = 0;

            SqlConnection con = new SqlConnection(this.StringConexion);

            string sql = " SELECT UsuarioId FROM Usuario WHERE @cedula = Cedula";
            SqlCommand com = new SqlCommand(sql, con);
            com.Parameters.AddWithValue("@cedula", cedula);

            try
            {
                con.Open();
                SqlDataReader reader = com.ExecuteReader();

                while (reader.Read())
                {
                    idusuario = int.Parse(reader["UsuarioId"].ToString());
                }

                con.Close();

            }
            catch
            {
                throw;
            }
            finally
            {
                if (con.State == ConnectionState.Open) con.Close();
            }

            return idusuario;
        }

        #region TEXTO
```

```
        public List<string> ObtenerDatosTabla(string tabla) //Recibe una string nombre
tabla y devuelve una lista de strings
        {
            List<string> lista = new List<string>();
            SqlConnection con = new SqlConnection(this.StringConexion);
            string sql = "SELECT * FROM " + tabla;
            SqlCommand com = new SqlCommand(sql, con);

            try
            {
                con.Open();
                SqlDataReader reader = com.ExecuteReader();

                while (reader.Read())
                {
                    string linea = "";
                    for (int i = 0; i < reader.FieldCount; i++)
                    {
                        linea += reader.GetValue(i).ToString();
                        if (i != reader.FieldCount - 1)
                        {
                            linea += " | ";
                        }
                        else
                        {
                            linea += "\r\n";
                        }

                    }
                    lista.Add(linea);
                }
            }
            catch
            {
                throw;
            }
            finally
            {
                if (con.State == ConnectionState.Open) con.Close();
            }
            return lista;
        }
        public List<string> ObtenerNombreTabla()
        {
            List<string> lista = new List<string>();
            SqlConnection con = new SqlConnection(this.StringConexion);
            string sql = "SELECT TABLE_NAME FROM
PrestamosExpress.INFORMATION_SCHEMA.TABLES WHERE TABLE_TYPE = 'BASE TABLE' AND TABLE_NAME
!= 'sysdiagrams';";
            SqlCommand com = new SqlCommand(sql, con);
            try
            {
                con.Open();
                SqlDataReader reader = com.ExecuteReader();
                while (reader.Read())
                {
                    string linea = reader.GetValue(0).ToString();
```

```
                    lista.Add(linea);
                }
            }
            catch
            {
                throw;
            }
            finally
            {
                if (con.State == ConnectionState.Open) con.Close();
            }
            return lista;
        }
        #endregion

        public bool MailUnico(string mail) //Consulta Si existe mail ingresado en la
columna email de tabla usuario
        {
            bool unico = false;
            SqlConnection con = new SqlConnection(this.StringConexion);
            string sql = "SELECT Email FROM Usuario WHERE Email = @mail";
            SqlCommand com = new SqlCommand(sql, con);
            com.Parameters.AddWithValue("@mail", mail);
            try
            {
                con.Open();
                SqlDataReader reader = com.ExecuteReader();
                if (!reader.HasRows)
                {
                    unico = true;
                }
            }
            catch
            {
                throw;
            }
            finally
            {
                if (con.State == ConnectionState.Open) con.Close();
            }
            return unico;
        }
        public bool UsuarioMayor(string cedula) //Compara fecha nacimiento usuario con
fecha de hoy - 21 años
        {
            bool mayor = false;
            DateTime hoy = DateTime.Today;
            DateTime limite = hoy.AddYears(-21);
            SqlConnection con = new SqlConnection(this.StringConexion);
            string sql = "SELECT Fecha_Nacimiento FROM Usuario WHERE Cedula = @cedula";
            SqlCommand com = new SqlCommand(sql, con);
            com.Parameters.AddWithValue("@cedula", cedula);
            try
            {
                con.Open();
                SqlDataReader reader = com.ExecuteReader();
                while (reader.Read())
                {
```

```
                    DateTime t = reader.GetDateTime(0);
                    if (t <= limite)
                    {
                        mayor = true;
                    }

                }
                con.Close();
            }
            catch
            {
                throw;
            }
            finally
            {
                if (con.State == ConnectionState.Open) con.Close();
            }

            return mayor;
        }

        public bool CelularUnico(string celular) //Consulta Si existe Celular ingresado
en la columna Celular de tabla usuario
        {
            bool unico = false;
            SqlConnection con = new SqlConnection(this.StringConexion);
            string sql = "SELECT Celular FROM Usuario WHERE Celular = @celular";
            SqlCommand com = new SqlCommand(sql, con);
            com.Parameters.AddWithValue("@celular", celular);
            try
            {
                con.Open();
                SqlDataReader reader = com.ExecuteReader();
                if (!reader.HasRows)
                {
                    unico = true;
                }
            }
            catch
            {
                throw;
            }
            finally
            {
                if (con.State == ConnectionState.Open) con.Close();
            }
            return unico;
        }
        public bool CedulaUnico(string cedula) //Consulta Si existe Cedula ingresada en
la columna Cedula de tabla usuario
        {
            bool unico = false;
            SqlConnection con = new SqlConnection(this.StringConexion);
            string sql = "SELECT Cedula FROM Usuario WHERE Cedula = @cedula";
            SqlCommand com = new SqlCommand(sql, con);
            com.Parameters.AddWithValue("@cedula", cedula);
            try
            {
```

```
                con.Open();
                SqlDataReader reader = com.ExecuteReader();
                if (!reader.HasRows)
                {
                    unico = true;
                }
            }
            catch
            {
                throw;
            }
            finally
            {
                if (con.State == ConnectionState.Open) con.Close();
            }
            return unico;
        }



        public bool Baja(int id)
        {
            throw new NotImplementedException();
        }

        public bool Modificacion(Usuario obj)
        {
            throw new NotImplementedException();
        }

        public List<Usuario> TraerTodo()
        {
            throw new NotImplementedException();
        }
    }
}
```

VALIDACIONES

Validaciones

```
namespace Validaciones
{
    class Validaciones
    {

    }

    public sealed class CedulaUnica : ValidationAttribute
    {
        protected override ValidationResult IsValid(object value, ValidationContext
validationContext)
        {
            string cedula = Convert.ToString(value);
            bool unica = FachadaExpress.ValidarCedula(cedula);
            if (unica)
            {
                return ValidationResult.Success;
            }
            else
```

```
            {
                return new ValidationResult(ErrorMessage);
            }

        }
    }
    public sealed class CelularUnico : ValidationAttribute
    {
        protected override ValidationResult IsValid(object value, ValidationContext
validationContext)
        {
            string celular = Convert.ToString(value);
            bool unico = FachadaExpress.NuevoCelular(celular);
            bool valido = FachadaExpress.CelularValido(celular);
            if (valido && unico)
            {
                return ValidationResult.Success;
            }
            else
            {
                return new ValidationResult(ErrorMessage);
            }
        }
    }
    public sealed class PassValida : ValidationAttribute
    {
        protected override ValidationResult IsValid(object value, ValidationContext
validationContext)
        {
            string pass = Convert.ToString(value);
            bool valido = FachadaExpress.ValidarPassword(pass);
            if (valido)
            {
                return ValidationResult.Success;
            }
            else
            {
                return new ValidationResult(ErrorMessage);
            }
        }
    }
    public sealed class EmailUnico : ValidationAttribute
    {
        protected override ValidationResult IsValid(object value, ValidationContext
validationContext)
        {
            string mail = Convert.ToString(value);
            bool valido = FachadaExpress.NuevoMail(mail);
            if (valido)
            {
                return ValidationResult.Success;
            }
            else
            {
                return new ValidationResult(ErrorMessage);
            }
        }
    }
```

```
    public sealed class TituloUnico : ValidationAttribute
    {
        protected override ValidationResult IsValid(object value, ValidationContext
validationContext)
        {

            string titulo = Convert.ToString(value);
            bool valido = FachadaExpress.NuevoTitulo(titulo);
            if (valido)
            {
                return ValidationResult.Success;
            }
            else
            {
                return new ValidationResult(ErrorMessage);
            }
        }
    }

    public sealed class RangoCuotas : ValidationAttribute
    {
        protected override ValidationResult IsValid(object value, ValidationContext
validationContext)
        {
            int cuotasIngresadas = Convert.ToInt32(value);
            int max = FachadaExpress.TraerCuotaMax();
            int min = FachadaExpress.TraerCuotaMin();
            if (cuotasIngresadas >= min && cuotasIngresadas <= max)
            {
                return ValidationResult.Success;
            }
            else
            {
                return new ValidationResult(ErrorMessage);
            }
        }
    }
    public sealed class EdadUsuario : ValidationAttribute
    {
        protected override ValidationResult IsValid(object value, ValidationContext
validationContext)
        {
            DateTime FechaNacimiento = Convert.ToDateTime(value);

            DateTime hoy = DateTime.Today;

            DateTime limite = hoy.AddYears(-21);
            RepoUsuarios usu = new RepoUsuarios();
            if (FechaNacimiento <= limite)
            {
                return ValidationResult.Success;
            }
            else
            {
                return new ValidationResult(ErrorMessage);
            }
        }
    }
```

```
}
```

MVC

HomeController

```
namespace MVC.Controllers
{
    public class HomeController : Controller
    {



        public ActionResult Home(string mensaje)
        {

            if(Session["rol"] == null)
            {
                return RedirectToAction("Login", "Usuario");
            }
            ViewBag.mensaje = mensaje;
            return View();
        }

        public ActionResult misProyectos()
        {
            if (Session["rol"] == null || Session["rol"].ToString() != "Solicitante")
            {
                return RedirectToAction("Login", "Usuario");
            }


            string usuario = Session["id"].ToString();

            int buscarusuario = int.Parse(usuario);

            ViewBag.jpg = ".jpg";
            List<Proyecto> Pro = FachadaExpress.TraerProyectos(buscarusuario);
            if (Pro == null)
            {
                ViewBag.Mensaje = "Usuario o contraseña incorrectos";
                return View();
            }

            return View(Pro);
        }

#region Alta Proyecto Personal
        public ActionResult AltaProyectoPersonal(string mensaje)
        {
            if (Session["rol"] == null || Session["rol"].ToString() != "Solicitante")
            {
                return RedirectToAction("Login", "Usuario");
            }
            ViewBag.Mensaje = mensaje;
            return View();

            //  return RedirectToAction("home", new { mensaje = textoMensaje });
        }
```

```
        [HttpPost]
        [ValidateInput(false)]
        public ActionResult AltaProyectoPersonal(ViewModelProyecto nuevoProyecto)
        {

            string errormensaje = "Hubo un problema, intentelo de nuevo mas tarde.";
            ViewBag.mensaje = errormensaje;

            #region GuardarImagen
            string rutaAplicacion = HttpRuntime.AppDomainAppPath;
            string nombreCarpetaImagenes =
ConfigurationManager.AppSettings["ImagenesGuardadas"];
            string rutaDirectorioImagenes = Path.Combine(rutaAplicacion,
nombreCarpetaImagenes);
            string rutaCompletaArchivo = Path.Combine(rutaDirectorioImagenes,
nuevoProyecto.Titulo + ".jpg");
            if (nuevoProyecto.Imagen != null)
            {
                nuevoProyecto.Imagen.SaveAs(rutaCompletaArchivo);
            }
            else if (nuevoProyecto.Imagen == null)
            {
                return View(nuevoProyecto);
            }

            #endregion

            nuevoProyecto.TasaInteres = FachadaExpress.ObtenerTasa(nuevoProyecto.Cuotas);
            nuevoProyecto.Integrantes = 1;
            nuevoProyecto.Tipo = "Personal";
            nuevoProyecto.Estado = "Pendiente de revisión";
            nuevoProyecto.FechaCreacion = DateTime.Today;



            if (ModelState.IsValid)
            {

                if (!FachadaExpress.ValidarMontoPersonal(nuevoProyecto.Monto))
                {
                    errormensaje = "El Monto esta fuera del rango";
                    return RedirectToAction("AltaProyectoPersonal", new { mensaje =
errormensaje });
                }
                if (!FachadaExpress.ValidarCuotas(nuevoProyecto.Cuotas))
                {
                    errormensaje = "Las cuotas estan fuera del rango";
                    return RedirectToAction("AltaProyectoPersonal", new { mensaje =
errormensaje });
                }
                if (!FachadaExpress.NuevoTitulo(nuevoProyecto.Titulo))
                {
                    errormensaje = "El Titulo ya existe";
                    return RedirectToAction("AltaProyectoPersonal", new { mensaje =
errormensaje });
                }
                return RedirectToAction("confirmarProyecto", nuevoProyecto);
```

```
            }

            //return RedirectToAction("AltaProyectoPersonal", new { mensaje =
errormensaje });
            return View(nuevoProyecto);
        }
        #endregion

        #region Alta Proyecto Cooperativo
        public ActionResult AltaProyectoCooperativo()
        {
            if (Session["rol"] == null || Session["rol"].ToString() != "Solicitante")
            {
                return RedirectToAction("Login", "Usuario");
            }
            return View();
        }

        [HttpPost]
        [ValidateInput(false)]
        public ActionResult AltaProyectoCooperativo(ViewModelProyectoCoop
nuevoProyectoCoop)
        {
            string errormensaje = "Hubo un problema, intentelo de nuevo mas tarde.";
            ViewBag.mensaje = errormensaje;

            string rutaAplicacion = HttpRuntime.AppDomainAppPath;
            string nombreCarpetaImagenes =
ConfigurationManager.AppSettings["ImagenesGuardadas"];
            string rutaDirectorioImagenes = Path.Combine(rutaAplicacion,
nombreCarpetaImagenes);
            string rutaCompletaArchivo = Path.Combine(rutaDirectorioImagenes,
nuevoProyectoCoop.Titulo + ".jpg");

            nuevoProyectoCoop.Imagen.SaveAs(rutaCompletaArchivo);

            nuevoProyectoCoop.TasaInteres =
FachadaExpress.ObtenerTasa(nuevoProyectoCoop.Cuotas);
            ViewModelProyectoCoop pro = nuevoProyectoCoop;
            pro.Tipo = "Cooperativo";
            pro.Experiencia = "--";
            pro.Estado = "Pendiente de revisión";
            pro.FechaCreacion = DateTime.Today;

            if (ModelState.IsValid)
            {
                if (!FachadaExpress.ValidarMontoCooperativo(pro.Integrantes, pro.Monto))
                {
                    errormensaje = "El Monto esta fuera del rango";
                    return RedirectToAction("AltaProyectoCooperativo", new { mensaje =
errormensaje });
                }
                if (!FachadaExpress.ValidarCuotas(pro.Cuotas))
                {
                    errormensaje = "Las cuotas estan fuera del rango";
                    return RedirectToAction("AltaProyectoCooperativo", new { mensaje =
errormensaje });
                }
```

```
                if (!FachadaExpress.NuevoTitulo(pro.Titulo))
                {
                    errormensaje = "El Titulo ya existe";
                    return RedirectToAction("AltaProyectoCooperativo", new { mensaje =
errormensaje });
                }

                return RedirectToAction("confirmarProyecto", pro);
            }

            return RedirectToAction("AltaProyectoCooperativo", new { mensaje =
errormensaje });

        }
        #endregion

        #region Confirmacion y Alta de Proyectos
        public ActionResult confirmarProyecto(ViewModelProyecto pro)
        {
            #region calcular los montos y cuotas
            decimal traerinteres = FachadaExpress.ObtenerTasa(pro.Cuotas); // trae el
calculo de la tasa de interes %%
            traerinteres = 1 + traerinteres / 100;
            decimal montototal = pro.Monto * traerinteres;     // Calcula el % de interes
que se le va a sumar al monto
            ViewBag.montoconinteres = montototal;                                  // se le
pasa el monto total con interes a la vista

            montototal = decimal.Round(montototal, 2);
            decimal montoporcuota = montototal / pro.Cuotas;
            montoporcuota = decimal.Round(montoporcuota, 2);
            ViewBag.montoconinteres = montototal;                                  // se le
pasa el monto total con interes a la vista
            ViewBag.montoporcuota = montoporcuota;
            #endregion
            // calcular los montos y cuotas
            ViewBag.imagen = "/Imagenes/" + pro.Titulo + ".jpg";
            if (pro.Solicitante == 0)
            {

                return View(pro);
            }

            return RedirectToAction("AltaProyecto", pro);
            //  return RedirectToAction("home", new { mensaje = textoMensaje });
        }
        public ActionResult AltaProyecto(Proyecto Alta)
        {
            string txtmensaje = "Se Confirmo el Proyecto con Exito";
            string cedula = Session["cedula"].ToString();

            string usuario = Session["id"].ToString();
            int buscarusuario = int.Parse(usuario);
            Alta.Solicitante = buscarusuario;

            if (FachadaExpress.ProyectosPendientes(cedula))
            {
                txtmensaje = "Aún tienes un Proyecto pendiente";
```

```
                return RedirectToAction("home", new { mensaje = txtmensaje });

            }
            if (!FachadaExpress.AltaProyectoParcial(Alta))
            {
                txtmensaje = "Ocurrio un error, verifica los datos";
                return RedirectToAction("home", new { mensaje = txtmensaje });
            }
            return RedirectToAction("home", new { mensaje = txtmensaje });
        }
        #endregion

        public ActionResult VerProyecto(string titulo)
        {
            if (Session["rol"] == null || Session["rol"].ToString() != "Administrador")
            {
                return RedirectToAction("Login", "Usuario");
            }
            Proyecto verProyecto = FachadaExpress.buscarProyecto(titulo);


            #region calcular los montos y cuotas
            decimal traerinteres = FachadaExpress.ObtenerTasa(verProyecto.Cuotas); //
trae el calculo de la tasa de interes %%
            traerinteres = 1 + traerinteres / 100;
            decimal montototal = verProyecto.Monto * traerinteres;     // Calcula el % de
interes que se le va a sumar al monto
            ViewBag.montoconinteres = montototal;                                  // se le
pasa el monto total con interes a la vista

            montototal = decimal.Round(montototal, 2);
            decimal montoporcuota = montototal / verProyecto.Cuotas;
            montoporcuota = decimal.Round(montoporcuota, 2);
            ViewBag.montoconinteres = montototal;                                  // se le
pasa el monto total con interes a la vista
            ViewBag.montoporcuota = montoporcuota;
            #endregion
            ViewBag.imagen = "/Imagenes/" + verProyecto.Titulo + ".jpg";

            return View(verProyecto);
            //  return RedirectToAction("home", new { mensaje = textoMensaje });
        }

        [HttpPost]
        public ActionResult CambiarEstado(int id, int votacion)
        {
            string errormensaje = "ocurrio un error, intentelo de nuevo mas tarde";
            string estado = "";
            if (votacion > 10)
            {
                return View();
            }
            else if (votacion >= 0 && votacion < 6)
            {
                estado = "Rechazado";
            }
            else if (votacion >= 6 && votacion <= 10)
            {
```

```
                estado = "Aprobado";
            }
           if(FachadaExpress.Confirmarproyecto(estado, id))
            {
                string Admin = Session["cedula"].ToString();
                DateTime fecha = DateTime.Today;
                if (FachadaExpress.Guardarconfirmacion(id, votacion, Admin, fecha))
                {
                    errormensaje = " El proyecto fue confirmado, su estado fue : " +
estado;
                    return RedirectToAction("Home", new { mensaje = errormensaje });
                }
            }
            return RedirectToAction("Home", new { mensaje = errormensaje });
        }

        #region Filtrar Proyecto ADMIN
        public ActionResult FiltrarProyectos()
        {
            if (Session["rol"] == null || Session["rol"].ToString() != "Administrador")
            {
                return RedirectToAction("Login", "Usuario");
            }
            return View();
        }

        [HttpPost]
        public ActionResult FiltrarProyectos(string cedula, string estado, string
palabra, DateTime? fecha)
        {
            string fe = "";
            if (fecha != null)
            {
                DateTime nueva = Convert.ToDateTime(fecha);

                string anio = nueva.Day.ToString();
                string mes = nueva.Month.ToString();
                string dia = nueva.Year.ToString();
                fe = dia + "-" + mes + "-" + anio;
            }
            List<Proyecto> lista = FachadaExpress.FiltrarProyectos(cedula, fe, estado,
palabra);
            ViewBag.jpg = ".jpg";
            return View(lista);

        }
        #endregion

        #region Armar Texto
        public ActionResult CrearTexto()
        {
            if (Session["rol"] == null || Session["rol"].ToString() != "Administrador")
            {
                return RedirectToAction("Login", "Usuario");
            }
            ViewBag.Nombres = FachadaExpress.BuscarNombresTablas();

            return View();
```

```
        }

        [HttpPost]
        public ActionResult CrearTexto(string tablas)
        {
            string mensajedevuelto = "imposible exportar las tablas. intentelo de
nuevo.";
            if (FachadaExpress.EscribirTabla(tablas))
            {
                mensajedevuelto = "la tabla se ha impreso correctamente";
                return RedirectToAction("Home", new { mensaje = mensajedevuelto });
            }

            return RedirectToAction("Home", new { mensaje = mensajedevuelto });
        }

        #endregion
    }
}
```

UsuarioController

```
namespace MVC.Controllers
{
    public class UsuarioController : Controller
    {
        // GET: Usuario
        public ActionResult Login(string mensaje)
        {
            Session.Abandon();
            return View();

        }

        [HttpPost]
        public ActionResult Login(string cedula, string password)
        {
            string errormensaje = "";
            Usuario unU = FachadaExpress.ObtenerUsuario(cedula, password);
            if (unU == null)
            {
                errormensaje = "Usuario o contraseña incorrectos";
                return RedirectToAction("Login", new { mensaje = errormensaje });

            }

            Session["rol"] = unU.Rol;
            Session["cedula"] = unU.Cedula;
            Session["id"] = unU.Id;


            return Redirect("/Home/home");
        }


        public ActionResult AltaUsuario(string mensaje)
        {
            ViewBag.mensaje = mensaje;
```

```
            return View();
        }

        [HttpPost]
        [ValidateInput(false)]
        public ActionResult AltaUsuario(ViewModelUsuario nuevoUsuario, string pass)
        {
            if (ModelState.IsValid) {
                nuevoUsuario.Rol = "Solicitante";
               if (pass == nuevoUsuario.Password)
                    {
                        return RedirectToAction("Nuevo", nuevoUsuario);
                        // bool ret = FachadaExpress.AltaUsuario(nuevoUsuario);
                    }
            }
            return View(nuevoUsuario);

        }

        public ActionResult Nuevo(Usuario nuevoUsuario)
        {
            nuevoUsuario.Rol = "Solicitante";
           if(FachadaExpress.AltaUsuario(nuevoUsuario))
          { return RedirectToAction("Login", new { mensaje = "Se dio de alta con Exito"
}); }

            return RedirectToAction("AltaUsuario", new { mensaje = "Verifique los datos"
});
        }

        public ActionResult Logout()
        {
            Session.Abandon();
            return RedirectToAction("login");
        }
    }
}
```

ViewModelProyecto

```
namespace MVC.Models
{
    public class ViewModelProyecto
    {
        public int Id { get; set; }
        public int Solicitante { get; set; }
        [Required]
        [TituloUnico(ErrorMessage = "El nombre del proyecto ingresado ya existe.")]
        [StringLength(50)]
        public string Titulo { get; set; }
        [Required]
        [StringLength(150)]
        public string Descripcion { get; set; }
        [Required]
        public decimal Monto { get; set; }
        [Required]
        [RangoCuotas(ErrorMessage = "El valor de cuotas ingresado está fuera de rango.")]
        public int Cuotas { get; set; }
```

```
        public HttpPostedFileBase Imagen { get; set; } //COMO ES EL FORMATO?

        public decimal TasaInteres { get; set; }

        public int Integrantes { get; set; }
        [Required]
        [StringLength(150)]
        public string Experiencia { get; set; }

        public string Tipo { get; set; }

        public string Estado { get; set; }

        public DateTime FechaCreacion { get; set; }
    }
}
```

ViewModelProyectoCoop

```
namespace MVC.Models
{
    public class ViewModelProyectoCoop
    {
        public int Id { get; set; }
        public int Solicitante { get; set; }
        [Required]
        [TituloUnico(ErrorMessage = "El nombre del proyecto ingresado ya existe.")]
        [StringLength(50)]
        public string Titulo { get; set; }
        [Required]
        [StringLength(150)]
        public string Descripcion { get; set; }
        [Required]
        public decimal Monto { get; set; }
        [Required]
        [RangoCuotas(ErrorMessage = "El valor de cuotas ingresado está fuera de rango.")]
        public int Cuotas { get; set; }

        public HttpPostedFileBase Imagen { get; set; } //COMO ES EL FORMATO?

        public decimal TasaInteres { get; set; }
        [Required]
        public int Integrantes { get; set; }

        [StringLength(150)]
        public string Experiencia { get; set; }

        public string Tipo { get; set; }

        public string Estado { get; set; }

        public DateTime FechaCreacion { get; set; }
    }
}
```

ViewModelUsuario

```
namespace MVC.Models
```

```
{
    public class ViewModelUsuario
    {
        public int Id { get; set; }
        //[Required][MaxLength(10, ErrorMessage ="El nombre no puede tener más de 10
caracteres")]
        [Required]
        [CedulaUnica(ErrorMessage = "La cédula ingresada ya está registrada.")]
        [StringLength(8)]
        public string Cedula { get; set; }

        [Required]
        [StringLength(30)]
        public string Nombre { get; set; }

        [Required]
        [StringLength(30)]
        public string Apellido { get; set; }

        [Required]
        [DataType(DataType.Date)]
        [Display(Name = "Fecha De Nacimiento")]
        [EdadUsuario(ErrorMessage = "Tiene que ser mayor a 21 años.")]
        public DateTime FechaNacimiento { get; set; }

        [Required]
        [StringLength(9)]
        [CelularUnico(ErrorMessage = "El numero que ingresó ya está registrado.")]
        public string Celular { get; set; }

        [Required]
        [StringLength(100)]
        [RegularExpression(@"^[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,4}$",
ErrorMessage = "El correo ingresado no es válido.")]
        [EmailUnico(ErrorMessage = "El correo que ingresó ya está registrado.")]
        public string Email { get; set; }

        [Required]
        [DataType(DataType.Password)]
        [Display(Name = "Contraseña")]
        [StringLength(30)]
        [PassValida(ErrorMessage = "Contraseña debe tener al menos 6 caracteres, 1
mayúscula, 1 minúscula y un numero.")]
        public string Password { get; set; }


        public string Rol { get; set; }
    }
}
```