# Funcionamiento de la aplicación

**Inicio**

Para iniciar la aplicación, esta requiere que haya presente múltiples elementos:

* Critico
  + Gitcredentials
  + Settings
  + ConfiguracionAplicacion
  + Idiomas [Carpeta](el inicial mínimo)
  + Encriptkeys
  + Logs
* Necesario
  + Creditos y credits
  + Emailcredentials
  + Googlecalendarcredentials
* Conveniente
  + Logodavid1.ico

**MainWindows**

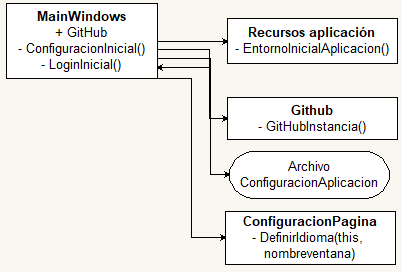
Lo primero que se hace es ajustar el tamaño de la ventana al 90% de la pantalla y centrarla, luego si no se había creado antes se crea el entorno inicial, para esto se llama a **RecursosAplicacion.EntornoInicialAplicacion()**

Se le adjunta a la propiedad **github** la instancia singleton de **Github.GithubInstancia()**

Luego leemos la configuración inicial con el método *configuracioninicial()*

Ahora se lee la configuración cargada en ConfiguracionAplicacion y se comprueba si esta marcada recordar usuario o login, en caso afirmativo se llama al método *LoginInicial()*

Ahora se define la página, esto carga en la pagina actual el resourcedictionary del idioma actual esto se hace con **ConfiguracionPagina.DefinirIdioma(this, “login”)**



**Recursos Aplicación.EntornoInicialAplicacion()**

Añade a la propiedad directoriosaplicacion, que no es más que una colección ordenada según la clave con un par clave, valor (sortedlist), la lista de todos los directorios necesarios e importantes para la aplicación y luego crea estos si no existieran ya.

**//Añadimos todas las direcciones a las que tendremos que hacer referencia, estas son, idiomas, una para cada idioma**

**//Logs, recursos, imagenes, usuario, credenciales, etc.**

**Directorios.Add("logs", DireccionBase + "\\Logs\\");**

**Directorios.Add("idiomas", DireccionBase + "\\Idiomas\\");**

**Directorios.Add("idioma\_ingles", DireccionBase + "\\Idiomas\\EN\\");**

**Directorios.Add("idioma\_espanol", DireccionBase + "\\Idiomas\\ES\\");**

**Directorios.Add("clave\_encriptado", DireccionBase + "\\Utils\\Keys\\");**

**Directorios.Add("credenciales", DireccionBase + "\\Utils\\Credentials\\");**

**Directorios.Add("iconos", DireccionBase + "\\Imagenes\\Icons");**

**Directorios.Add("imagenes\_defecto", DireccionBase + "\\Imagenes\\Default\\");**

**Directorios.Add("usuario", DireccionBase + "\\Usuarios\\");**

**Directorios.Add("imagenes\_usuario", DireccionBase + "\\Usuarios\\Imagenes\\");**

**Directorios.Add("archivos\_usuario", DireccionBase + "\\Usuarios\\Archivos\\");**

**Directorios.Add("recursos", DireccionBase + "\\Resources\\");**

**//Nos aseguramos de que estan todas las carpetas importantes creadas, creandolas si faltan.**

**foreach (string direccion in Directorios.Values)**

**{**

**Directory.CreateDirectory(direccion);**

**}**

Esto genera una estructura con la siguiente forma

* App
  + Logs
  + Utils
    - Keys
    - Credentials
  + Imágenes
    - Icons
    - Default
  + Idiomas
    - EN
      * Una entrada por cada ventana a traducir
    - ES
      * Una entrada por cada ventana a traducir
  + Usuarios
    - Por cada usuario
      * Campaña
        + Imágenes
        + Archivos

**Github (creación de instancia, elemento singleton, constructor)**

Creamos un cliente de github, con el nombre del proceso actual es decir ElEscribaDelDJ si estamos en español o Master’scribe si estamos en inglés, el encargado de esto es.

**new GitHubClient(new ProductHeaderValue(System.Diagnostics.Process.GetCurrentProcess().ProcessName));**

Que guarda dicha información en la propiedad cliente.

Se llama a la clase externa para encriptar o desencriptar AES, la cual desencripta las credenciales usando la clave almacenada en Utils/Claves/

**AESencription encriptar = new AESencription();**

Guardamos la dirección completa de la clave encriptada la cual es un objecto JSON de tipo AESencription, y hacemos que el objeto encriptar sea equivalente al contenido de este archivo, para que posea dichas claves.

**string path = RecursosAplicacion.Directorios["clave\_encriptado"] + "encriptkeys.txt";**

**encriptar = JsonConvert.DeserializeObject<AESencription>(File.ReadAllText(path));**

Posteriormente una vez temenos el objeto que se ocupa de la encriptación y desencriptación con las claves, ahora cambiamos la dirección para leer el archivo de las credenciales de github para desencriptarlas posteriormente (estas credenciales están encriptadas porque github no permite subir archivos con credenciales en texto plano, si detecta esto lo considera inseguro y bloquea las credenciales).

**path = RecursosAplicacion.Directorios["credenciales"] + "gitcredentials.txt";**

Ahora para desencriptar las credenciales, necesitamos leer los bytes del archivo

**var bytes2 = System.IO.File.ReadAllBytes(path);**

Con esto generamos las credenciales de github y posteriormente recuperamos el repositorio, indicando el nombre del propietario del repositorio y el nombre del repositorio.

**this.cliente.Credentials = new Credentials(encriptar.Decrypt(bytes2, encriptar.AesKey, encriptar.AesIv));**

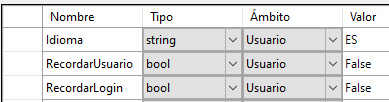
**this.repositorio = cliente.Repository.Get("davidgmd", "Proyecto-de-fin-de-grado").Result;**

**Archivo – ConfiguracionAplicacion.settings –**

Este archivo permite poder acceder a la configuración en tiempo de ejecución de forma fácil a través de toda la aplicación.

Hay aparte un archivo settings.ini que es el que guarda la configuración para que esta se conserve al abrir la aplicación de nuevo.

El contenido del archivo es simplemente



Es decir, el idioma por defecto, y las opciones de recordar usuario y recordar login que por defecto son false, es decir desmarcadas, luego esto se comprueba al leer el archivo .ini y se modifica.

**MainWindows.ConfiguracionInicial()**

Se guardan los valores iniciales de la configuración en una propiedad vector de strings, del archivo settings.ini

**this.valoresinicialesconf = System.IO.File.ReadAllLines(RecursosAplicacion.DireccionBase + "\\Settings.ini");**

Como vamos a recorrer con un foreach, vamos a usar una variable auxiliar contadora

**int i = 0;**

Ahora recorremos ese vector de cadenas y una por una vamos dividiendo cada línea separándola por “:” así por ejemplo Idioma:ES, separamos la categoría del valor y almacenamos el valor, la variable auxiliar nos permite almacenar los 3 valores cada uno con un índice diferente.

**foreach (string cadenainicial in this.valoresinicialesconf)**

**{**

**var cadenaresultante = cadenainicial.Split(':');**

**this.valoresinicialesconf[i] = cadenaresultante[1].Trim().ToString();**

**i += 1;**

**}**

Ahora uso una herramienta llamada regex.replace esto permite usar una expresión regular sobre una cadena, se utiliza porque al leer las líneas de configuración se toma como si las cadenas tuvieran unas comillas, para evitar este comportamiento anomalo que a veces ocurre, se utiliza una expresión regular que elimina cualquier comilla y espacios en blanco.

Tras eso se almacena en el archivo de configuración accesible fácilmente todos los valores necesarios.

**Regex.Replace(this.valoresinicialesconf[1], @"\s+", "");**

**ConfiguracionAplicacion.Default.Idioma = this.valoresinicialesconf[0];**

**ConfiguracionAplicacion.Default.RecordarUsuario = Convert.ToBoolean(this.valoresinicialesconf[1].Trim());**

**ConfiguracionAplicacion.Default.RecordarLogin = Convert.ToBoolean(this.valoresinicialesconf[2].Trim());**

Veamos ahora por último lo que define que idioma se usa en cada ventana

**ConfiguracionPagina.DefinirIdioma(this, "Login")**

Primero definimos una variable para guardar las direcciones diferentes para cada ventana, ahora comprobamos según la variable nombreventana

**string path;**

**switch (nombreventana)**

**{**

**case ("Options"):**

**path = RecursosAplicacion.DireccionBase + "\\Idiomas\\" + ConfiguracionAplicacion.Default.Idioma + "\\View\\Options\\" + nombreventana + ".xaml";**

**break;**

**case ("MainMenu"):**

**path = RecursosAplicacion.DireccionBase + "\\Idiomas\\" + ConfiguracionAplicacion.Default.Idioma + "\\View\\" + nombreventana + ".xaml";**

**break;**

**default:**

**path = RecursosAplicacion.DireccionBase + "\\Idiomas\\" + ConfiguracionAplicacion.Default.Idioma + "\\" + nombreventana + ".xaml";**

**break;**

**}**

Como idioma es una propiedad ResourceDicionary establecemos como source la dirección antes definida y a la ventana le añadimos como recurso ese diccionario que hemos establecido.

**idioma.Source = new Uri(path, UriKind.Absolute);**

**ventana.Resources.MergedDictionaries.Add(idioma);**