Conecta

Arquitectura de Conecta CerP12

Proyecto: **Conecta CerP12**

Elaboró: **Conecta México**

Fecha: **14 de Octubre 2015**

Historial de revisión

Fecha	Version	Descripción	Autor
14/10/2015	1.0	Producción	Conecta

CONECTA CERP12

INTRODUCCIÓN

Este documento provee un resumen de alto nivel y explica completamente la arquitectura empleada para el desarrollo del software Conecta CerP12 . Este trabajo contiene los componentes de la arquitectura, los cuales implican documentar el conjunto relevante a los puntos del desarrollo, maquetación del diseño y proceso del software. De manera secuencial explica la importancia de cada componente, sus constitución a un alto nivel con base en los requerimientos funcionales, de los cuales, se rige esta arquitectura. Conecta CerP12 es una aplicación que convierte archivos Cer y Key a archivos P12.

ALCANCE

El alcance de este documento es centrar las bases firmes del desarrollo integral de Conecta CerP12, con base en diagramas de casos de uso, diagramas de clases, diagramas de flujo y diagrama de arquitectura. Con base a lo anterior podemos establecer los patrones de diseño y paradigmas que más se adecuen a la solución. Una vez contemplando todos los posibles escenarios realizar las especificaciones de pruebas y matrices con base en los diagramas de flujo.

METAS Y RESTRICCIONES

Metas

Diseñar y modelar un sistema de software que pueda responder a las necesidades de conversión de archivos .cer y .key a .p12. Esto mediante una aplicación de escritorio.

Restricciones:

- La aplicación no genera archivos .key
- La aplicación no genera archivos .cer

REQUERIMIENTOS

ID	NOMBRE	DESCRIPCIÓN
RF1	**Obtener archivo .key**	Se requiere que el usuario tenga en su poder un archivo .key esta es una llave privada que el sistema Conecta Cer-P12 decodificara. El .key es opcional, por lo que el usuario puede no elegir el archivo.
RF2	**Obtener archivo .cer**	Se requiere que el usuario tenga en su poder un archivo .cer proporcionado por el SAT. Este archivo se debe localizar en la computadora para poder convertirlo a un p12, este archivo es obligatorio.
RF3	**Contraseña al archivo p12**	Se requiere que el archivo p12 tenga una contraseña por seguridad de resguardo. En caso de que se quiera instalar el p12, pedirá contraseña. La contraseña debe ser alfanumérica y debe contener una mayúscula cualquier número de minúsculas, y un caracter como [*,%,#,&].
RF4	**Generar archivo .p12**	Con base en el archivo .cer se crea un archivo p12, el archivo .key es opcional ya que este archivo es la llave privada del .p12.
RF5	**Interfaz Gráfica**	Se requiere que las operaciones de los requerimientos RF1, RF2, RF3 y RF4 se visualicen en una interfaz gráfica, ahí el usuario va a buscar los archivos .key y .cer; ingresará su contraseña y se seleccionará la carpeta o la ruta en donde será almacenado el archivo .p12 una vez que se haya creado. En caso de que se haya creado el p12 con éxito, existirá una opción para abrir la carpeta donde esta localizado el archivo, en caso de fallo debe mostrar el error ala descripción del error.

ACTORES

Actor	**Usuario operador de Conecta CerP12**	**Identificador: AU1**
Descripción	Usuario que utiliza Conecta CerP12 y desea convertir archivos .cer con una .key opcional a P12	
Características	Persona de la institución financiera	
Relación	Todo el sistema	
Referencias	Conecta CerP12	

Actor	**Conecta CerP12**	**Identificador: AU2**
Descripción	La aplicación conteniendo todos los requerimientos funcionales	
Características	• Recibe .key • Recibe .cer • Recibe contraseña • Genera archivo p12	
Relación	Todo el sistema	
Referencias	Conecta CerP12	

DESCRIPCIÓN DE LOS CASOS DE USO

Caso de Uso	**Obtener archivo .key**	**Identificador:** **CUT1**
Actores	• Usuario operador de Conecta Cer-P12 • Conecta Cer-P12	
Tipo	Primario	
Referencias	Con todo el sistema	
Precondición	• El usuario debe tener un archivo .key proporcionado por el SAT • En caso de que se haya elegido un archivo diferente al .key, se debe mencionar el error al usuario.	
Postcondición	• Se genera una referencia en Conecta Cer-P12, del archivo .key para poder obtener el archivo en modo de bytes y poder tratarlo dentro de la aplicación.	
Descripción	• Busca el archivo .key en la computadora del usuario, operador de Conecta Cer-P12 para poder obtener la referencia y el archivo.	
Resumen	Obtiene el archivo .key.	

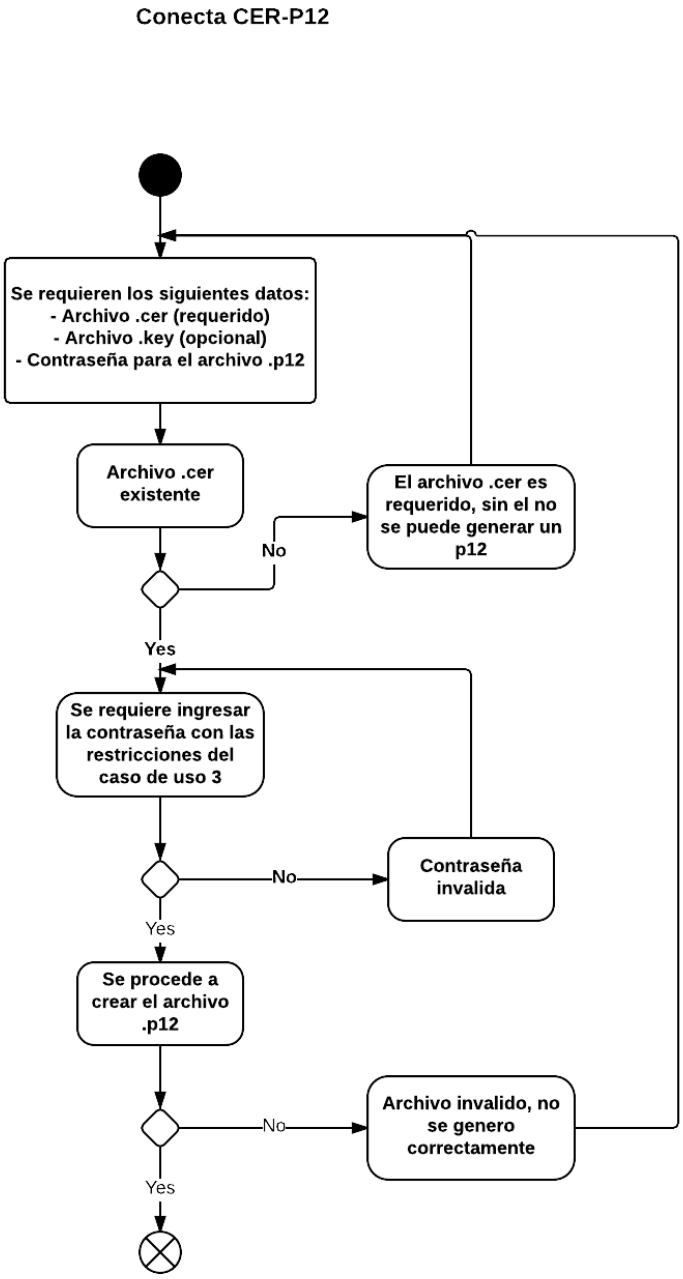
Caso de Uso	**Obtener archivo .cer**	**Identificador:** **CUT2**
Actores	• Usuario operador de Conecta Cer-P12 • Conecta Cer-P12	
Tipo	Primario	
Referencias	Con todo el sistema	
Precondición	• El usuario debe tener un archivo .cer proporcionado por el SAT • En caso de que se haya elegido un archivo diferente al .cer, se debe mencionar el error al usuario.	
Postcondición	• Se genera una referencia en Conecta Cer-P12, del archivo .cer para poder obtener el archivo en modo de bytes y poder tratarlo dentro de la aplicación.	
Descripción	• Busca el archivo .cer en la computadora del usuario, operador de Conecta Cer-P12 para poder obtener la referencia y el archivo.	
Resumen	Obtiene el archivo .cer.	

Caso de Uso	**Contraseña al archivo p12**	**Identificador:** **CUT3**
Actores	• Usuario operador de Conecta Cer-P12 • Conecta Cer-P12	
Tipo	Primario	
Referencias	Con todo el sistema	
Precondición	• La contraseña debe contener las siguientes características: - Longitud mínima de 6 caracteres y máxima de 18 - Al menos una mayúscula - Al menos un número - Al menos un caracter especial [*, $, %, &], máximo 3	
Postcondición	• Si la contraseña es correcta y válida, se puede proceder a los siguientes requerimientos. En caso de que la contraseña no sea validad, se debe anunciar al usuario que validación de la contraseña no paso.	
Descripción	• El usuario debe ingresar una contraseña en la aplicación, esta contraseña servirá para resguardar el archivo p12.	
Resumen	Contraseña para el archivo p12.	

Caso de Uso	**Generar archivo .p12**	**Identificador:** **CUT4**
Actores	• Usuario operador de Conecta Cer-P12 • Conecta Cer-P12	
Tipo	Primario	
Referencias	Con todo el sistema	
Precondición	• Se debió obtener el archivo .cer y este debió ser valido. • [Opcional] Se debió obtener el archivo .key valido • Se debió crear una contraseña válida para el archivo .p12 • Haber localizado una ruta válida de almacenamiento para el archivo p12	
Postcondición	• Se genera una referencia en Conecta Cer-P12, del archivo .cer para poder obtener el archivo en modo de bytes y poder tratarlo dentro de la aplicación. Si hay un error se debe notificar al usuario lo que esta sucediendo, caso de falta de espacio en disco, nombre de archivo invalido, etc.	
Descripción	• Busca el archivo .cer en la computadora del usuario, operador de Conecta Cer-P12 para poder obtener la referencia y el archivo.	
Resumen	Una vez obtenido el .cer, un .key (opcional) y asignado una contraseña, se generará el archivo .p12.	

Caso de Uso	**Exposición de los requerimientos en un entorno gráfico**	**Identificador:** **CUT5**
Actores	• Usuario operador de Conecta Cer-P12 • Conecta Cer-P12	
Tipo	Primario	
Referencias	Con todo el sistema	
Precondición	• Se deben haber realizo los Casos de uso del 1 al 4 • Se debe tener espacio en el disco para instalar la aplicación • La aplicación corre en arquitectura 64-bits	
Postcondición	• Se expondrá en una ventana con entorno gráfico la funcionalidad de Conecta Cer-P12. En el caso de uso 1 donde se tiene que buscar un .key, se debe de crear un botón de “examinar archivos”, con el cual se localizará el archivo .key. Con respecto al caso de uso 2, se debe crear un botón de “examinar archivos”, con el cual se localizará el archivo .cer. Para el caso de uso 3 se debe de crear un campo de texto con el texto ofuscado, para que no se pueda leer la contraseña, así mismo a un lado de este, deben aparecer las validaciones de la contraseña. En el caso de uso 4 se debe de generar un botón que diga “Guardar como”, con el cual, se abrirá una pantalla con el sistema de archivos abierto. Ahí el usuario podrá seleccionar el nombre y en que directorio se almacenará el archivo p12. • Después de llenar todos los campos se deben generar dos ventanas más, una de error en donde aparecerá la imagen de una equis roja, donde especifique el error y la posible causa. Así mismo una ventana con una paloma verde, en donde diga que fue exitosa la creación y aparezca un botón que te dirige al directorio en donde se creó el archivo .p12. En ambas ventanas se puede regresar al inicio de la aplicación.	
Descripción	• Se crea una aplicación de entorno gráfico y de escritorio. Exponiendo las funcionalidades Conecta Cer-P12.	
Resumen	La exposición final de Conecta Cer-P12	

CAPA LÓGICA


Conecta CER-P12
Se requieren los siguientes datos:
- Archivo .cer (requerido)
- Archivo .key (opcional)
- Contraseña para el archivo .p12
Archivo .cer existente
No
El archivo .cer es requerido, sin el no se puede generar un p12
Yes
Se requiere ingresar la contraseña con las restricciones del caso de uso 3
No
Contraseña invalida
Yes
Se procede a crear el archivo .p12
No
Archivo invalido, no se genero correctamente
Yes

MÓDULO DE COMPOSICIÓN WEB

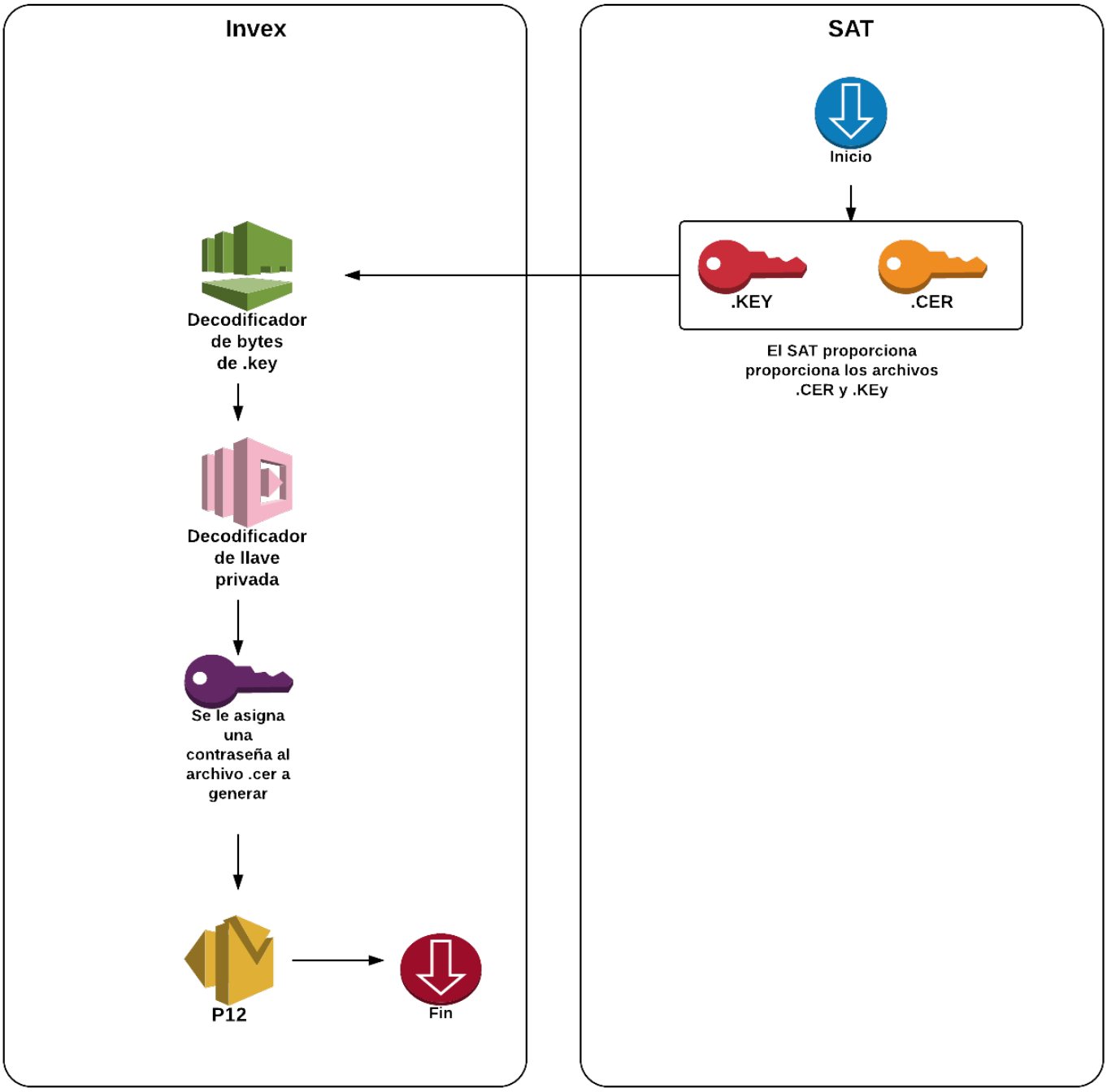


DIAGRAMA DE CASOS DE USO

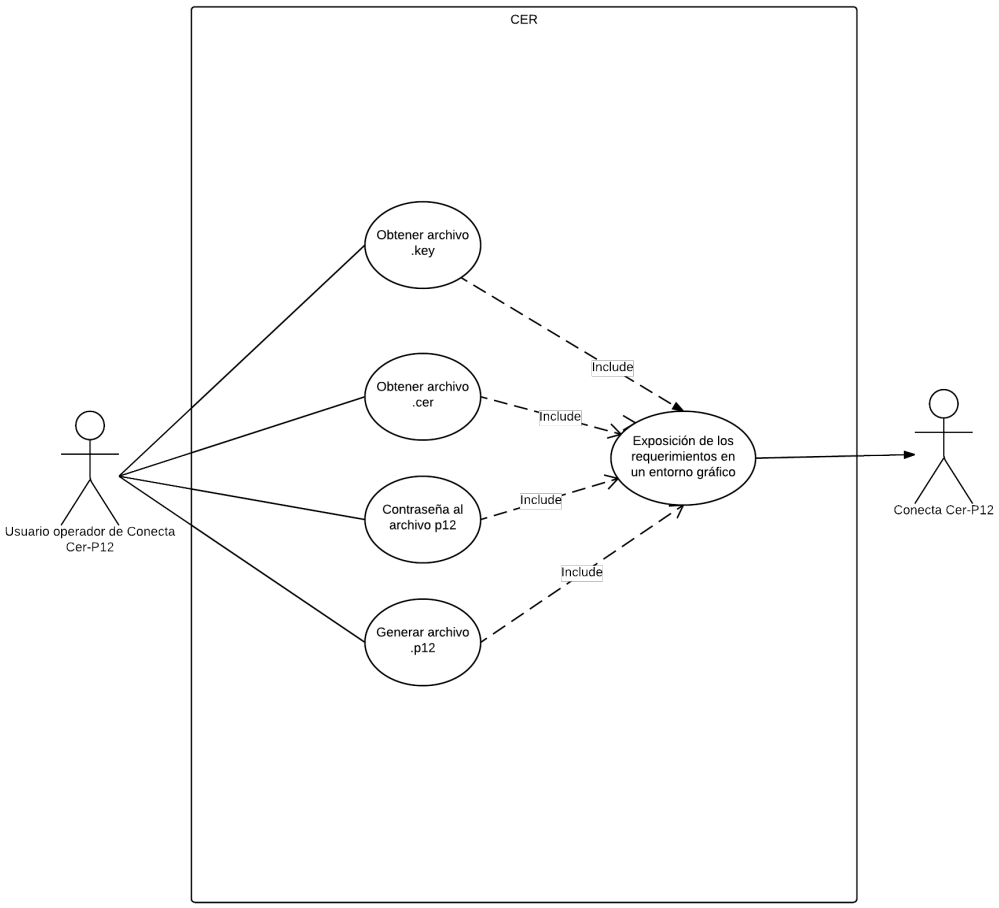
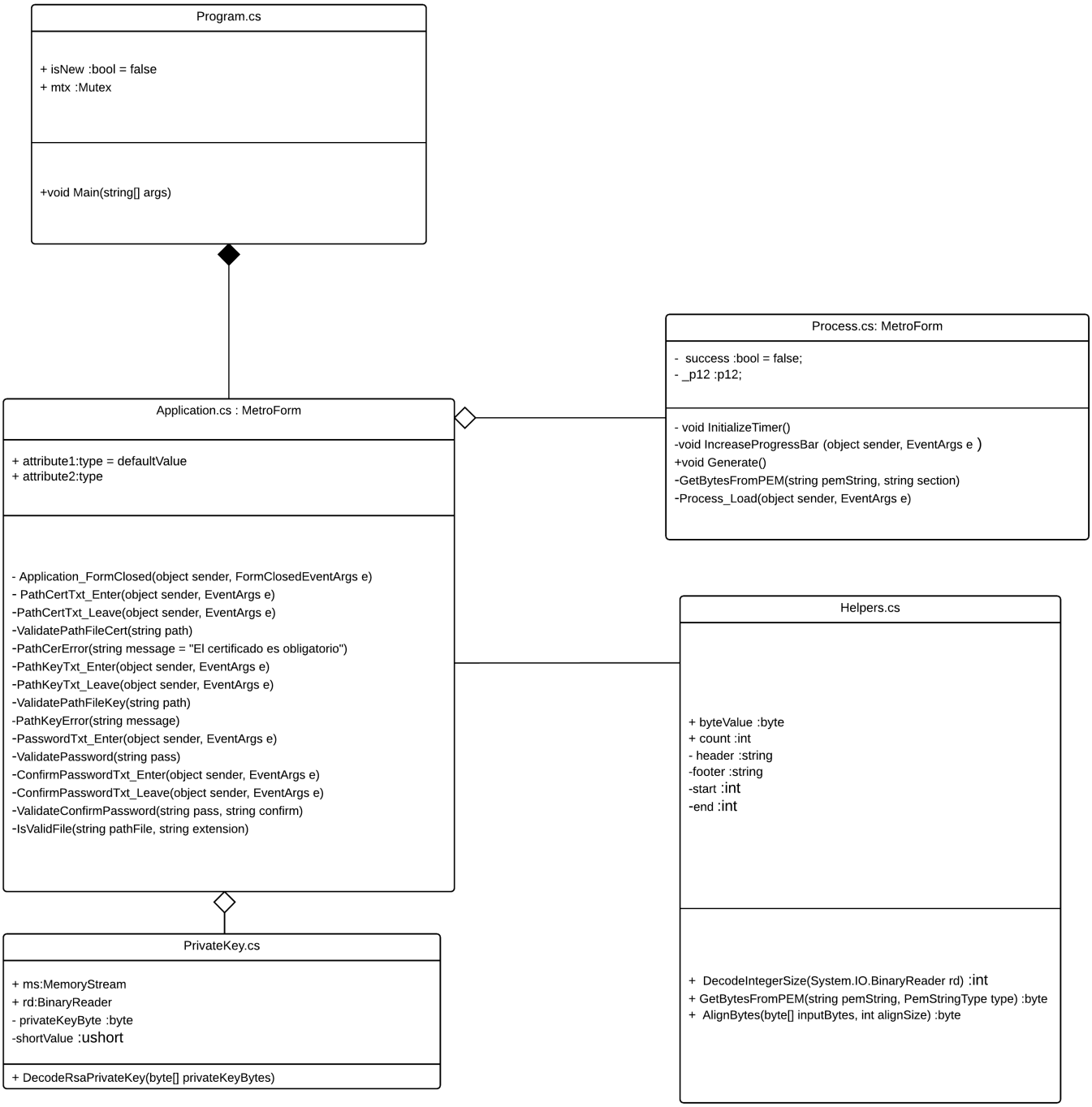


DIAGRAMA DE CLASES



DESCRIPCIÓN DE MÉTODOS DE PROGRAMACIÓN

Descripción de método	**Main**	**Identificador del método: MET1**
Sintaxis del método	void Main(string[] args)	
Descripción del método	Punto de entrada de la aplicación, cuando la aplicación se inicia puede tomar un conjunto de parámetros de entrada.	
Parámetros del método	args: String[]	
Salidas del método	No devuelve valores ya que es un método **VOID**	

Descripción de método	**Cierre del Formulario**	**Identificador del método: MET2**
Sintaxis del método	`void Application_FormClosed(object sender, FormClosedEventArgs e)`	
Descripción del método	Se dispara al ser cerrado el FORM, si tiene un FORM padre es cerrado.	
Parámetros del método	Object: Control(FORM) desata el evento **FormClosed** FormClosedEventArgs: Datos del evento se pasa al controlador	
Salidas del método	No devuelve valores ya que es un método **VOID**	

Descripción de método	**Entrada al control de la ruta del certificado**	**Identificador del método: MET3**

Sintaxis del método	`void PathCertTxt_Enter(object sender, EventArgs e)`
Descripción del método	Modifica la propiedad **MetroColorStyle** al entra al foco del control.
Parámetros del método	Object: Referencia al control que lanza el evento **ENTER** EventArgs: Datos del evento se pasa al controlador
Salidas del método	No devuelve valores ya que es un método **VOID**

Descripción de método	**Salida del foco del control de la ruta del certificado**	**Identificador del método: MET4**
Sintaxis del método	`void PathCertTxt_Leave(object sender, EventArgs e)`	
Descripción del método	Modifica la propiedad **MetroColorStyle** al salir del foco del control.	
Parametros del método	Object: Referencia al control que lanza el evento **LEAVE** EventArgs: Datos del evento se pasa al controlador	
Salidas del método	No devuelve valores ya que es un método **VOID**	

Descripción de método	**Valida si es un PATH valido (.cer)**	**Identificador del método: MET5**

Sintaxis del método	`bool ValidatePathFileCert(string path)`
Descripción del método	Verifica si es un **PATH** valido (.cer) del control **PathCertTxt**
Parámetros del método	String: Cadena con la ruta hacia un archivo (.cer)
Salidas del método	bool: **TRUE** si la ruta existe **FALSE** en caso contrario

Descripción de método	**Despliega el mensajes de error**	**Identificador del método: MET6**
Sintaxis del método	`void PathCerError(string message = "El certificado es obligatorio")`	
Descripción del método	Despliega el mensaje de error del control **PathCertLbl**	
Parámetros del método	String: cadena con el mensaje del error	
Salidas del método	No devuelve valores ya que es un método **VOID**	

Descripción de método	**Búsqueda del archivo (.key) a través del asistente**	**Identificador del método: MET7**

Sintaxis del método	`void SearchPathKeyBtn_Click(object sender, EventArgs e)`
Descripción del método	Obtiene la ruta de la llave (.key) a través de un asistente.
Parametros del método	Object: Referencia al control que lanza el evento **CLICK** EventArgs: Datos del evento se pasa al controlador
Salidas del método	No devuelve valores ya que es un método **VOID**

Descripción de método	**Modifica la propiedad MetroColorStyle al entra al foco del control.**	**Identificador del método: MET8**
Sintaxis del método	`private void PathKeyTxt_Enter(object sender, EventArgs e)`	
Descripción del método	Modifica la propiedad **MetroColorStyle** al entra al foco del control.	
Parametros del método	Object: Referencia al control que lanza el evento **ENTER** EventArgs: Datos del evento se pasa al controlador	
Salidas del método	No devuelve valores ya que es un método **VOID**	

Descripción de método	**Modifica la propiedad MetroColorStyle al salir del foco del control.**	**Identificador del método: MET9**
Sintaxis del método	`void PathKeyTxt_Leave(object sender, EventArgs e)`	
Descripción del método	Modifica la propiedad **MetroColorStyle** al salir del foco del control.	
Parametros del método	Object: Referencia al control que lanza el evento **LEAVE** EventArgs: Datos del evento se pasa al controlador	
Salidas del método	No devuelve valores ya que es un método **VOID**	

Descripción de método	**Valida si es un PATH valido .key del control PathKeyTxt**	**Identificador del método: MET10**
Sintaxis del método	`bool ValidatePathFileKey(string path)`	
Descripción del método	Valida si es un **PATH** valido .key del control **PathKeyTxt**	
Parámetros del método	path: cadena con la ruta del archivo (.key)	
Salidas del método	**Bool: TRUE** si la ruta existe **FALSE** en caso contrario	

Descripción de método	**Despliega el mensage de error del control PathKeyLbl**	**Identificador del método: MET11**
Sintaxis del método	`void PathKeyError(string message)`	
Descripción del método	Despliega el mensaje de error del control **PathKeyLbl**	
Parámetros del método	String: cadena con el mensaje	
Salidas del método	No devuelve valores ya que es un método **VOID**	

Descripción de método	**Entrada password**	**Identificador del método: MET12**
Sintaxis del método	`void PasswordTxt_Enter(object sender, EventArgs e)`	
Descripción del método	Se encarga de manejar el campo del password	
Parámetros del método	Object: Referencia al control que lanza el evento **ENTER** EventArgs: Datos del evento se pasa al controlador	
Salidas del método	No devuelve valores ya que es un método **VOID**	

Descripción de método	**Salida del Foco del Control PasswordTxt**	**Identificador del método: MET13**
Sintaxis del método	`void PasswordTxt_Leave(object sender, EventArgs e)`	
Descripción del método	Método que se ejecuta cuando se deja el campo del password	
Parámetros del método	Object: Referencia al control que lanza el evento **LEAVE** EventArgs: Datos del evento se pasa al controlador	
Salidas del método	No devuelve valores ya que es un método **VOID**	

Descripción de método	**Valida si una cadena es una contraseña valida**	**Identificador del método: MET14**
Sintaxis del método	`bool ValidatePassword(string pass)`	
Descripción del método	Valida si una cadena es una contraseña valida	
Parámetros del método	String: cadena de contraseña a validar	
Salidas del método	Bool: **TRUE** si es una contraseña valida **FALSE** de lo contrario	

Descripción de método	**Entrada del foco del control ConfirmPasswordTxt**	**Identificador del método: MET15**
Sintaxis del método	`void ConfirmPasswordTxt_Enter(object sender, EventArgs e)`	
Descripción del método	Método que se ejecuta cuando se entra en el campo contraseña cuando se confirma	
Parámetros del método	Object: Referencia al control que lanza el evento **ENTER** EventArgs: Datos del evento se pasa al controlador	
Salidas del método	No devuelve valores ya que es un método **VOID**	

Descripción de método	**Salida de foco del control ConfirmPasswordTxt**	**Identificador del método: MET16**
Sintaxis del método	`void ConfirmPasswordTxt_Leave(object sender, EventArgs e)`	
Descripción del método	Método que se ejecuta cuando se sale del campo contraseña cuando se confirma	
Parámetros del método	Object: Referencia al control que lanza el evento **LEAVE** EventArgs: Datos del evento se pasa al controlador	
Salidas del método	No devuelve valores ya que es un método **VOID**	

Descripción de método	**Compara y valida dos contraseña**	**Identificador del método: MET17**
Sintaxis del método	`bool ValidateConfirmPassword(string pass, string confirm)`	
Descripción del método	Valida si dos cadenas son contraseñas validas e iguales.	
Parámetros del método	String: cadena de contraseña String: cadena de confirmación de contraseña	
Salidas del método	Bool: **TRUE** si las contraseñas son válidas **FALSE** de lo contrario	

Descripción de método	**Valida una ruta**	**Identificador del método: MET18**
Sintaxis del método	`bool IsValidFile(string pathFile, string extension)`	
Descripción del método	Valida si un archivo existe y si la extensión del archivo coincide.	
Parámetros del método	String: cadena con la ruta del archivo a validar String: cadena con la extensión del archivo a validar	
Salidas del método	Bool: **TRUE** si el archivo existe y la extensión es la solicitada **FALSE** de lo contrario	

Descripción de método	**Decodifica una RSA Private Key**	**Identificador del método: MET19**
Sintaxis del método	`RSACryptoServiceProvider DecodeRsaPrivateKey(byte[] privateKeyBytes)`	
Descripción del método	Decodifica una **RSA** Private Key	
Parámetros del método	Bytes[]: Arreglo de Bytes de un archivo (.key)	
Salidas del método	`RSACryptoServiceProvider: Provedor de servicios RSACrypto`	

Descripción de método	**Inicia el proceso de creación del .p12 (PKCS)**	**Identificador del método: MET20**
Sintaxis del método	`void Process_Load(object sender, EventArgs e)`	
Descripción del método	Inicia el proceso de creación del .p12 (PKCS)	
Parámetros del método	Object: Referencia al control que lanza el evento **LOAD** EventArgs: Datos del evento se pasa al controlador	
Salidas del método	No devuelve valores ya que es un método **VOID**	

Descripción de método	**Crea el .p12 (PKCS) archivo**	**Identificador del método: MET21**
Sintaxis del método	`void Generate()`	
Descripción del método	Crea el .p12 (PKCS) archivo	
Parámetros del método	Sin parametros	
Salidas del método	No devuelve valores ya que es un método **VOID**	

Descripción de método	**Transforma un cadena en un arreglo de bytes**	**Identificador del método: MET22**
Sintaxis del método	`byte[] GetBytesFromPEM(string pemString, string section)`	
Descripción del método	`Transforma un cadena desde una sección especifica regresando un arreglo de bytes`	
Parámetros del método	String: cadena a convertir String: punto de inicio	
Salidas del método	Bytes[]: arreglo de bytes de la cadena convertida desde un punto especificado	

Descripción de método	**Inicializa Timer**	**Identificador del método: MET23**
Sintaxis del método	`void InitializeTimer()`	
Descripción del método	Inicializa el timer.	
Parámetros del método	Sin parametros	
Salidas del método	No devuelve valores ya que es un método **VOID**	

Descripción de método	**Incrementa el ProgressBar hasta iniciar la aplicación**	**Identificador del método: MET24**
Sintaxis del método	`void IncreaseProgressBar(object sender, EventArgs e)`	
Descripción del método	Incrementa el ProgressBar, hasta que un evento lo detenga	
Parámetros del método	Object: Referencia al control que lanza el evento **TICK** EventArgs: Datos del evento se pasa al controlador	
Salidas del método	No devuelve valores ya que es un método **VOID**	

Descripción de método	**Decodifica y obtiene el tamaño un BinaryReader**	**Identificador del método: MET25**
Sintaxis del método	`int DecodeIntegerSize(System.IO.BinaryReader rd)`	
Descripción del método	Esta helper function analiza el tamaño de un entero y lee su contenido usando un formato ASN.1	
Parámetros del método	BinaryReader:	
Salidas del método	Int: obtenemos el tamaño del BinaryReader	

Descripción de método	**Obtiene los bytes de una cadena PEM**	**Identificador del método: MET26**
Sintaxis del método	`byte[] AlignBytes(byte[] inputBytes, int alignSize)`	
Descripción del método	Obtiene los bytes de una cadena PEM	
Parámetros del método	inputBytes: bytes a parsear alignSize: tamaño del arreglo	
Salidas del método	Bytes[]: bytes de una cadena PEM	