5.3 Diseño de funcionalidades

En un proyecto con las dimensiones del que estamos exponiendo en este documento, existen multitud de componentes compartidas por toda la aplicación, que intervienen en diferentes funcionalidades, y que, por lo tanto, van a ser diseñadas por separado. Consecuentemente vamos a diferenciar el diseño en las siguientes partes:

- Componentes estándar.
- Funcionalidades.

5.3.1 Componentes estándar

Los diferentes componentes estándares que se forman parte de la aplicación ISM son los que se listan a continuación:

- Interfaz gráfica del sistema
- Módulo de gestión de base de datos (Consola y Agente)
- Módulo de gestión del Web Service.
- Módulo de navegadores
- Motor de la aplicación
- Scheduler
- Módulo de gestión de resultados
- Módulo de gestión de alertas a móvil/correo electrónico
- Notificación de alertas a OpenView
- Módulo de gestión de claves dinámicas
- Módulo de lectura de ficheros INI
- Módulo de conexión con RAS
- Módulo de encriptación

- Control del registro de Windows
- Módulo de FindAlerts

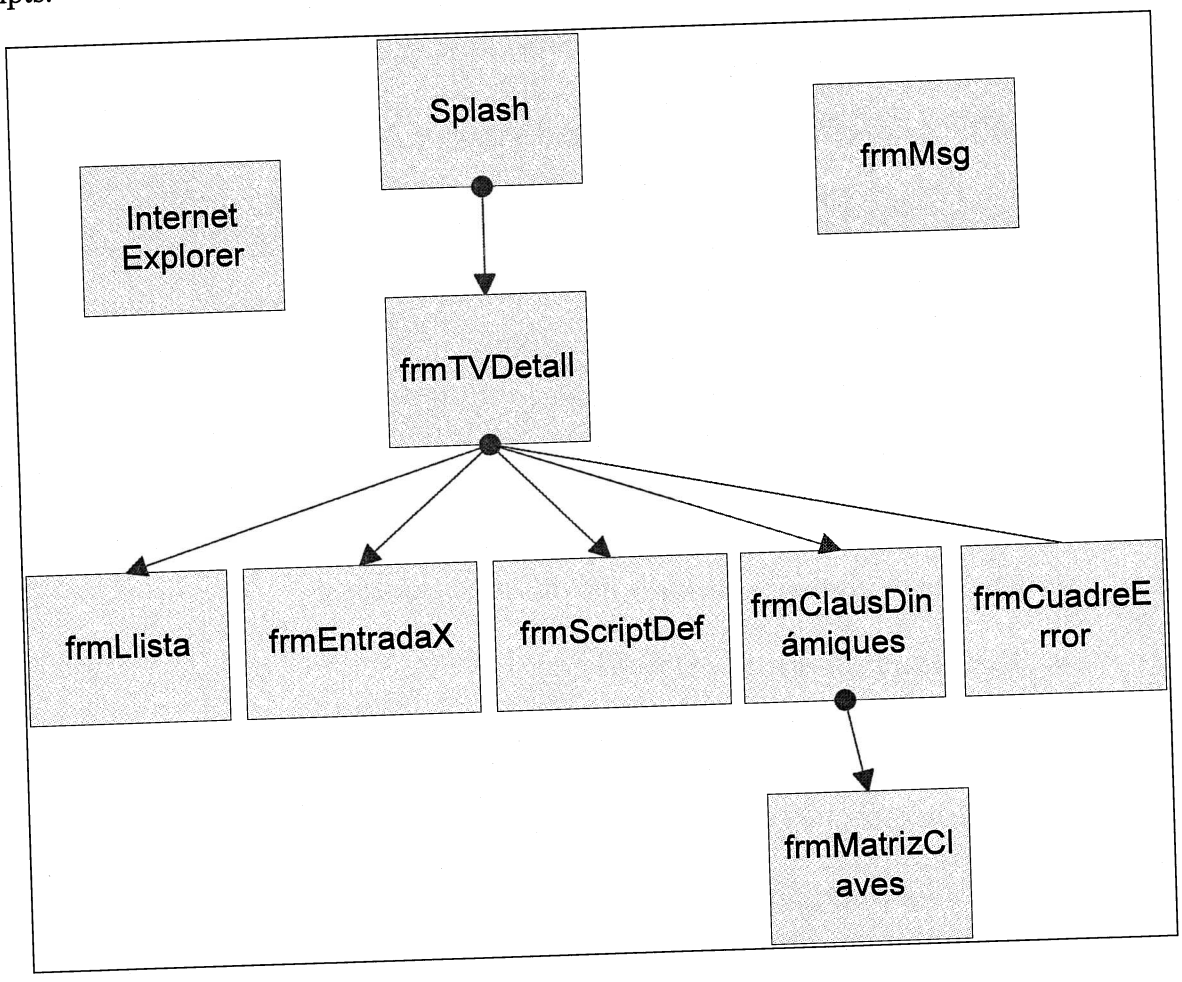
5.3.1.1 Interfaz gráfica del sistema

La interfaz gráfica se planteó basándose en la potencia y sencillez que plantea Visual Basic.

Toda la lógica de la interfaz gráfica se divide en formularios que se muestran al usuario dependiendo de las acciones que desea ejecutar.

Consola

A continuación, se muestra una estructura de las vistas y como están relacionadas entre ellas, está vista corresponde a la consola hasta el punto de entrar en el Generador de Scripts.



El formulario **Splash** es el formulario que se muestra al iniciar el aplicativo tanto la consola como el agente.

El formulario **frmTVDetall** muestra las interficie de la consola con el robot que actualmente se está tratando y el árbol de ejecución, con servicios, tests e ISP's. A partir de este formulario se accede a toda una serie de vistas que muestran información de configuración del Robot.

frmLlista es un formulario genérico, se utiliza para mostrar los listados seleccionables que se muestran a la hora de elegir test, ISP's, personas, robots,...
El formulario **frmClausDinamiques**, es el formulario de entrada para gestionar la funcionalidad de claves dinámicas.

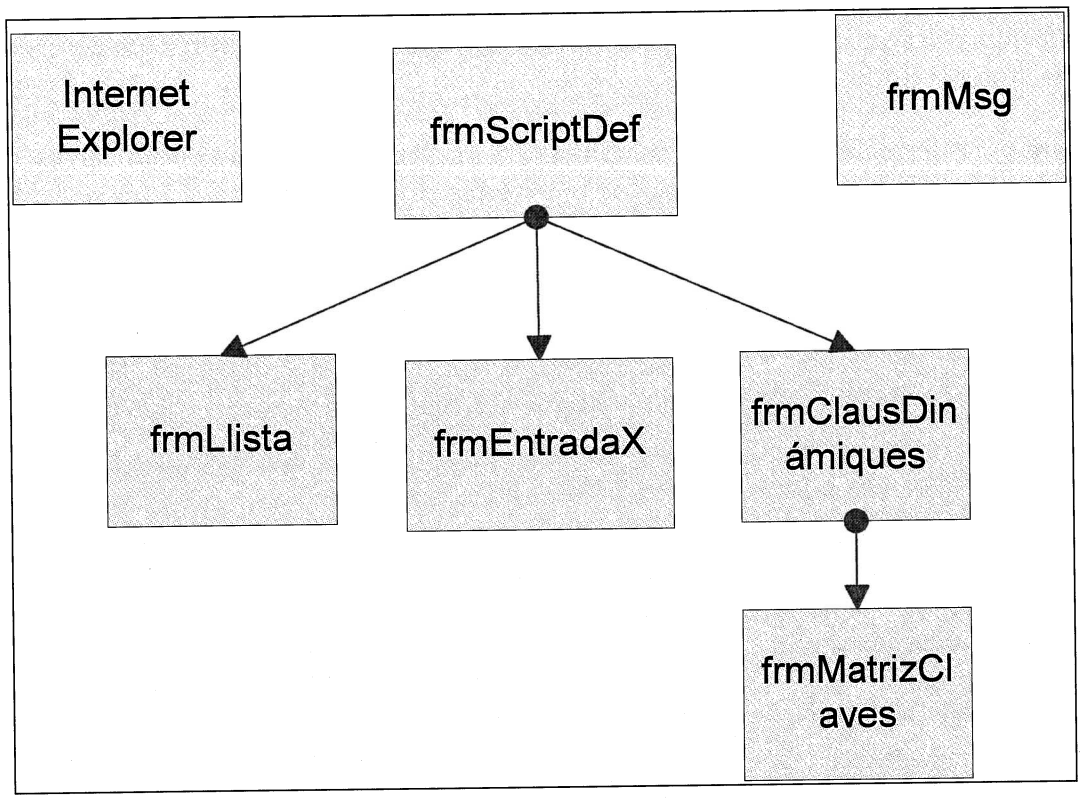
Los formularios con nombre **frmEntradaX** se corresponden a una serie de formularios de entrada de datos que se corresponden a los elementos del árbol de ejecución o elementos que intervienen en la monitorización;

- frmEntradaAturades
- frmEntradaCD
- frmEntradaError
- frmEntradaFrase
- frmEntradaFrases
- frmEntradaISP
- frmEntradaRobotCS
- frmEntradaScripts
- frmEntradaScriptsCitrix
- frmEntradaScriptsFTP
- frmEntradaScriptsMail
- frmEntradaServei
- frmEntradaTest

El formulario **frmMsg**, es un formulario genérico, se utiliza para mostrar los mensajes que se muestran al usuario.

En el objeto **InternetExplorer** se visualiza la página que se está grabando o testeando en ese momento.

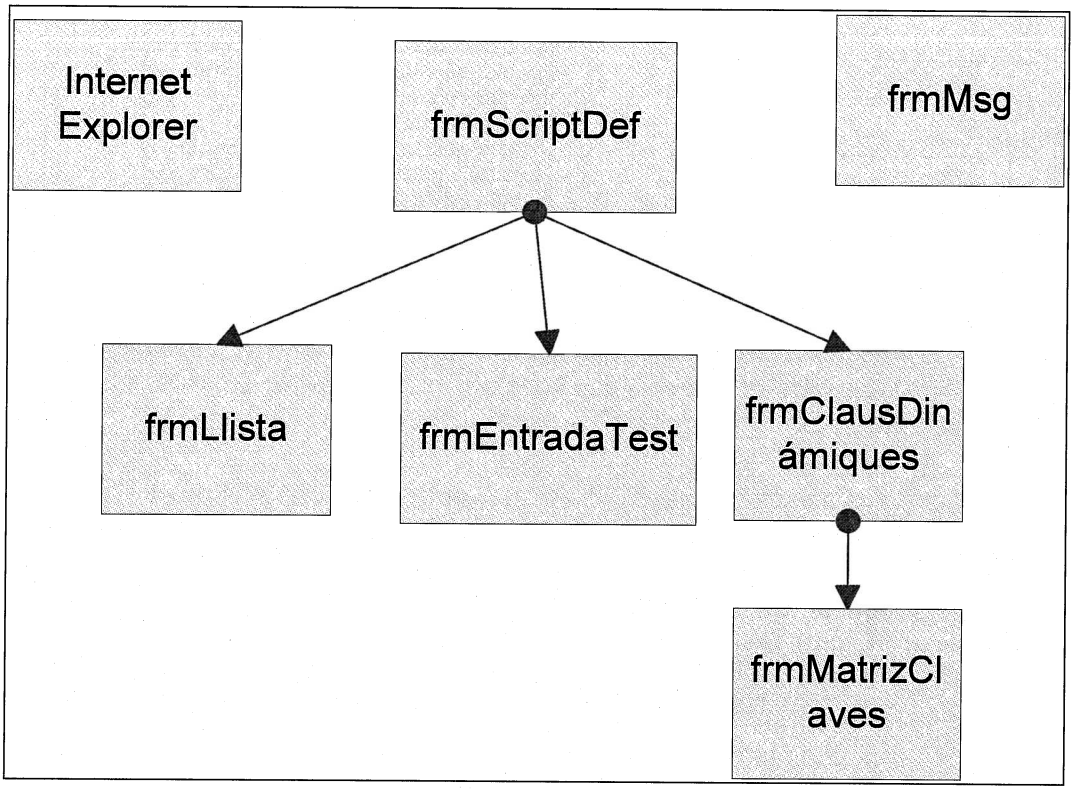
A continuación, se muestra una estructura de las vistas y como están relacionadas entre ellas del Generador de Scripts.



No se entra en detalle de como se configuran las ventanas ya que se utiliza el IDE proporcionado por Microsoft para generar vistas y código Visual Basic.

Agente

A continuación, se muestra una estructura de las vistas y como están relacionadas entre ellas de la aplicación Agente.

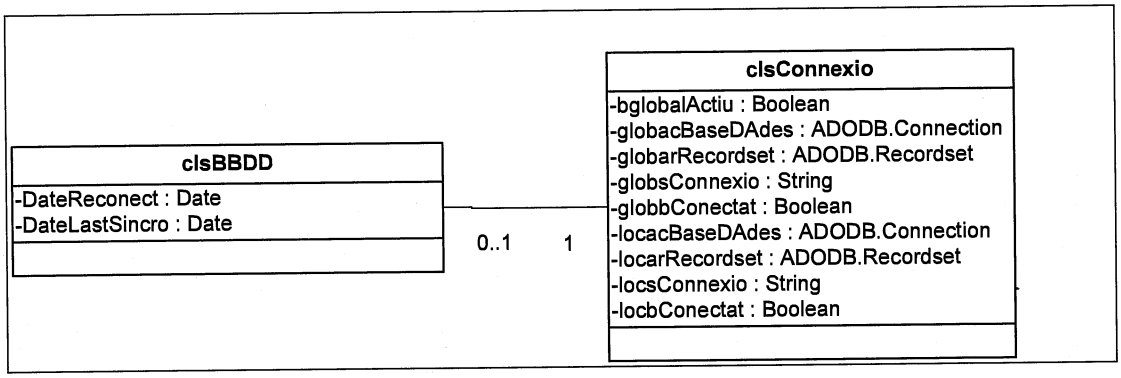


5.3.1.2 Módulo de gestión de base de datos (Consola y Agente)

El sistema de gestión de base de datos debe tener las siguientes características:

- Acceso a base de datos estándar, independiente del Sistema Gestor de Bases de datos que se este utilizando. Por lo tanto, se debe utilizar SQL estándar.
- Reconexión automática
- Control de errores
- Ejecución de procedimientos almacenados
- Gestión de una base de datos local y otra remota, en caso de no utilizar Servicios Web

Una vez definidos los objetivos que ha de cumplir el módulo, se diseñó una serie de clases que a continuación se detallan.



Existe una única clase de acceso a base de datos, tanto para la Consola como para el Agente. La case clsConnexio dispone de dos conexiones a la base de datos. Y existen dos utilizaciones diferentes:

1. Dos conexiones a bases de datos. La primera conexión corresponde al repositorio general alojado en un servidor dedicado, o como se suele llamar, en el proyecto, base de datos remota. La segunda conexión se usa para gestionar una conexión local. La conexión local, se utiliza en caso de perdida de conexión con la base de datos remota.

   Esta configuración, en estos momentos es obsoleta y se utilizaba cuando no existía la versión, de la consola y el agente, con Web Services.

2. La segunda configuración, corresponde a una conexión a una base de datos local, que hace de soporte al Web Service. Esta base de dalos, almacena los datos de resultados antes de enviarlos y almacena información de los árboles de ejecución de cada robot.

La clase clsConnexio gestiona las conexión con la base de datos y ejecuta las sentencias que la aplicación le envía mediante llamadas a funciones. Esta clase dispone de una serie de métodos para ejecutar acciones de Insert, Update, Select, ejecución de Storeds Procedures.

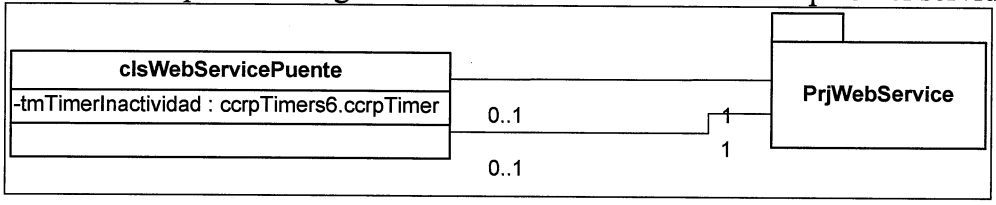
5.3.1.3 Módulo de gestión del Web Service.

ISM utiliza un módulo de Web Service propio para gestionar las conexiones con el servidor de WS.

Este WS utiliza un protocolo propio de comunicación, basado en el intercambio de XML's, como anteriormente se ha explicado.

El desarrollo de este módulo, se ha realizado creando una clase que se encarga de gestionar las comunicaciones con una DLL que se ha desarrollado en Visual Basic.

Este módulo que se encarga de realizar las comunicaciones http con el servidor.

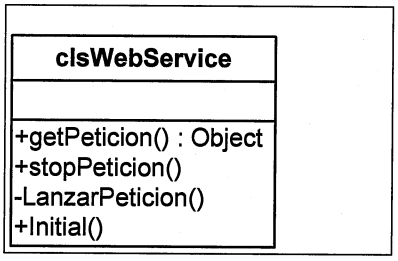


La clase **clsWebServicePuente**, dispone de dos instancias del paquete **PrjWebService** las cuales se utilizan para lo siguiente:

1. La primera para realizar todas las peticiones necesarias para el funcionamiento a petición del agente y la consola.

2. La segunda se utiliza para realizar pings periódicos al servidor web para informar de que la consola está funcionando.

El paquete PrjWebService está diseñado en Visual Basic, a continuación se muestra las funcionalidades que dispone:



getPeticion: Método de entrada para lanzar una petición http
LazarPetición: realiza la petición al servidor WS y espera la respuesta en caso de ser síncrona.

Este paquete se encarga de realizar la petición http al servidor web mediante el componente de Microsoft INET, este componente tiene las siguientes características:

INET proporciona la implementación para los dos protocolos más extendidos en Internet el HyperText Transfer Protocol (HTTP) y File Transfer Protocol (FTP).

Permite conectarse a la WWW usando el protocolo http para recibir documentos HTML o XML. Con el protocolo FTP, se nos permite conectarnos a servidores FTP y enviar y descargar ficheros.

- Una vez realizado el proyecto se han encontrado formas mejores de realizar esta comunicación con el servidor web. Hubiera sido interesante realizar las

comunicaciones con el Microsoft **XML 2.0** o superior que incluye un objeto para poder hacer está tarea, se trata de **XMLHTTPRequest**.

Este objeto contiene una serie de métodos que nos permite enviar un comando al servicio Web para mandarle la información y también nos permite recibir los datos que el servicio Web nos envía.

5.3.1.4 Módulo de navegadores

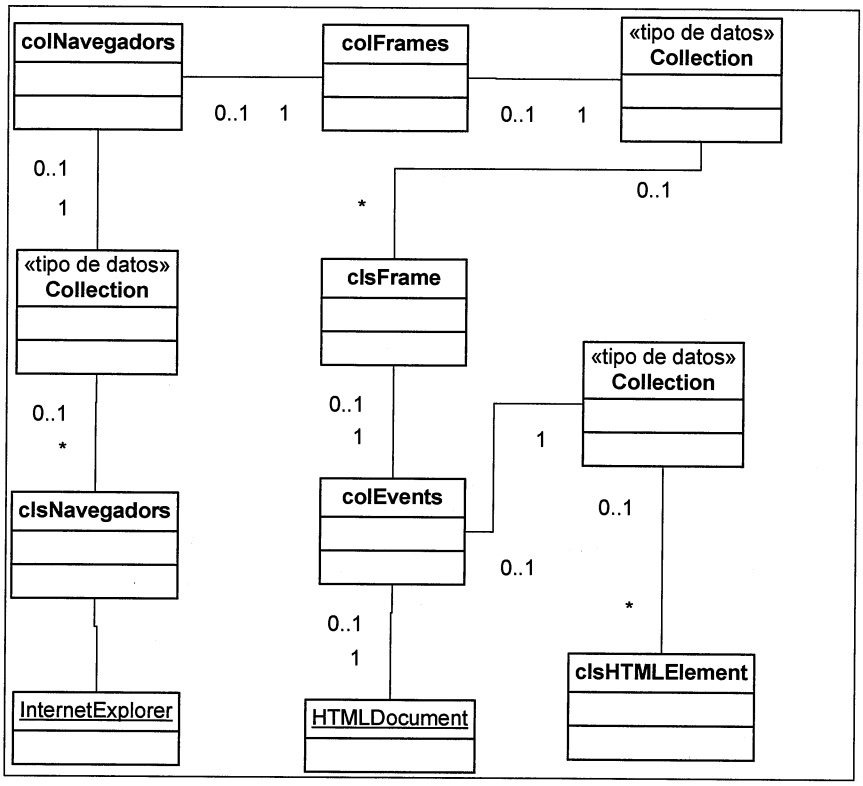
El módulo de navegadores controla los navegadores que intervienen en la grabación y o monitorización.

Debe tener las siguientes características:

- Poder controlar más de un navegador en caso de que la monitorización lo necesite. Disponer de los eventos de navegación iniciada, navegación completada, aparición de nuevas ventanas y posibles errores en la navegación
- Disponer de todas las acciones que se ejecutan en el navegador sobre los elementos que contiene el documento HTML.
- Poder controlar cualquier elemento que pueda aparecer en la página HTML y en caso de aparecer nuevos elementos también se controlen.
- Poder ampliar el control de las acciones, en caso de que aparezcan nuevos eventos, que se ejecutan en los elementos.

El desarrollo de este módulo se ha basado en una clase, **colNavegadors,** que contiene la colección de los navegadores que forman parte de la navegación.

A continuación, se ilustra las clases que intervienen en este módulo y sus relaciones.



La clase **clsNavegadors** es la que contiene una instancia del objeto Internet Explorer y es está clase la que recibe los eventos de:

BeforeNavigate2: Este evento se recibe antes de comenzar la navegación y tiene la función para ISM de control a donde se va a navegar y cortar en algunos casos la navegación, como puede ser, en el caso de los múltiples dominios.

DocumentComplete: Indica que la carga se ha completa correctamente y se utiliza para obtener el tiempo de visual de la navegación.

NavigateComplete2: Indica que la navegación se ha completado y se utiliza para obtener el tiempo de carga de la navegación.

NewWindow2: Cuando en una navegación aparece una nueva ventana se genera este evento y sirve para añadir el nuevo navegador a la colección de navegadores

NavigateError: Este evento se produce cuando se genera un error.

5.3.1.5 Motor de la aplicación

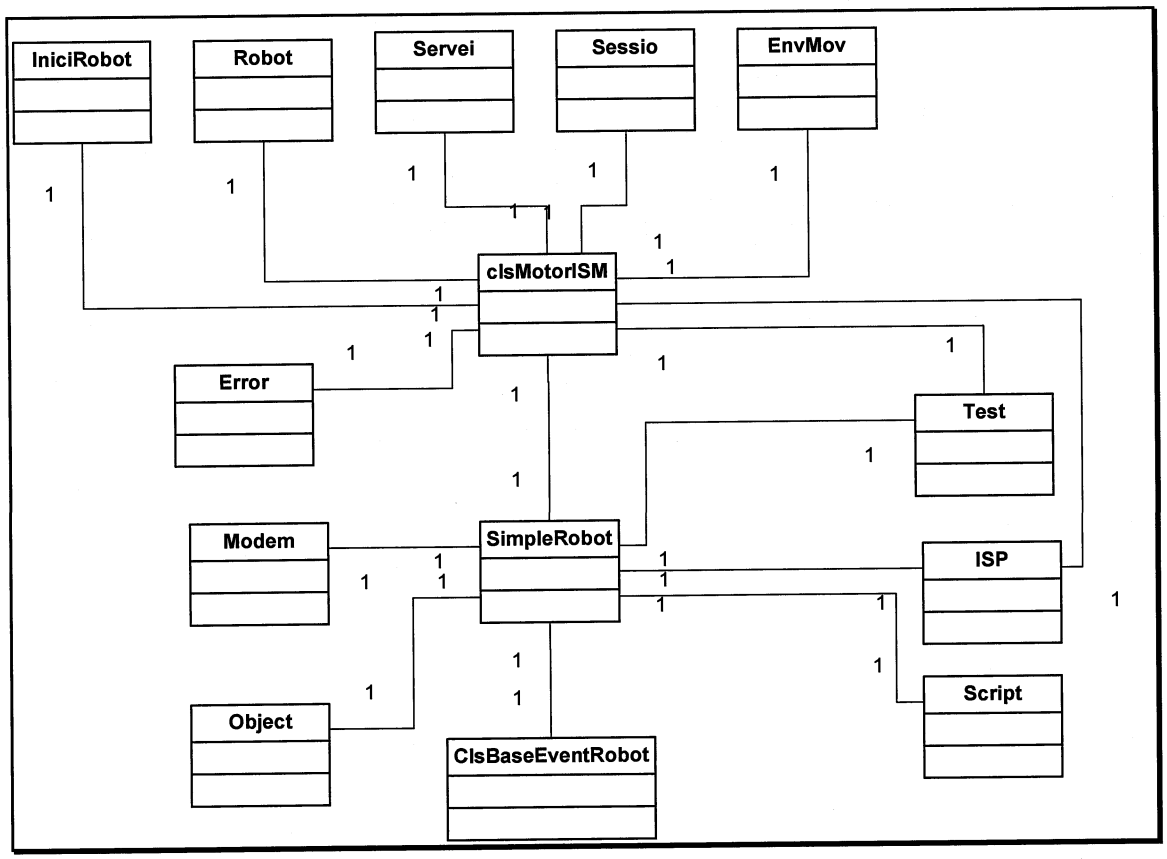
El motor de la aplicación es el módulo que se encarga de controlar toda la gestión de la aplicación y a los diferentes tipos de navegador que pueden existir en un futuro.

Debe tener las siguientes características:

- Debe ser un módulo escalable para poder incluir futuros tipos de test y navegadores. Por lo tanto, no se debe solo crear para controlar test de Internet Explorer

Este módulo se basa en la clase **clsMotorISM**, la cual se encarga de iniciar todas las navegaciones, controlar la finalización de las navegaciones, notificar de nuevo envíos de alertas a móvil o correo electrónico y avisar a las vistas de inicios de navegación, errores, resultados y finalizaciones de navegación.

La estructura de clases es la siguiente:



La clase **clsMotorISM** dispone de una instancia de cada uno de los elementos que intervienen en la navegación de un test.

- **IniciRobot**, indica el inicio del robot y tiene una relación con el servicio

- **Robot**, la clase MotorISM tiene una instancia del Robot, para tener un acceso rápido a sus métodos

- **Servei**, la clase servei indica el servicio que se está ejecutando en un momento determinado,

- **Sessio**, indica la sesión que se ejecuta en ese momento, por lo tanto, indica el test y el ISP que se está utilizando para monitorizar una transacción

- **Test**, instancia del test que indica el test que se está ejecutando

- **ISP**, instancia de la clase ISP, que indica el ISP por el cual se está ejecutando el test

- **EnvMov**, la clase clsMotorISM contiene una instancia a la clase que se encarga de enviar las alertas por correo electrónico o por SMS.

- **Error**, dispone de una instancia a la clase error, para poder gestionar los errores que le informa la navegación y poder mostrarlos.

- **clsSimpleRobot,** está clase es la clase principal y clsMotorISM contiene una instancia para poder gestionar diferentes tipos de Navegadores, es está clase la que contiene la generalización de Navegadores

La clase **clsSimpleRobot**, es la que se encarga de recibir notificaciones de los diferentes tipos de navegadores y abstrae al motor de ISM de que tipo de test y navegador se está utilizando para realizar una prueba. Para conseguir esto se ha creado un objeto del tipo Object en Visual Basic que puede contener los diferentes tipos de navegadores.
Esta clase también contiene una instancia de la clase **Modem**, que se encarga de gestionar las conexiones a ISP's y marcaciones de acceso telefónico a redes.

Esta clase contiene instancias de las siguientes clases:

- **Test**, instancia a la clase de Test, para conocer que Test se está ejecutando

- **ISP**, instancia de la clase ISP del ISP actual

- **Script**, instancia de script actual

- **Object**, está clase se utiliza para instancia de forma genérica cada uno de los tipos de navegadores que existen.

- **Modem,** instancia a la clase que controla el modem.

- **clsBaseEvRobot,** clase que se utiliza para dar los eventos del motor de ejecución de cada uno de los tipos de navegadores. Sirve de interficie entre el motor y el scheduler.

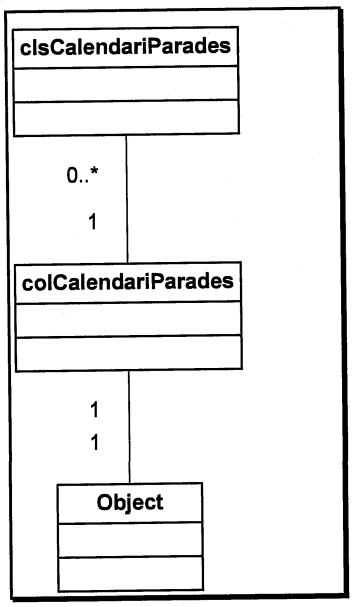
5.3.1.6 Scheduler

El módulo de scheduler se compone de una clase que contiene una colección de calendarios.

Este módulo debe disponer de las siguientes funcionalidades:

- Contener todos los calendarios de un elemento del árbol de ejecución
- Discernir de entre todos los calendarios, si el elemento debe ejecutarse o no

Las clases que intervienen en este módulo son las siguientes:



La clase **clsColcalendariParades** contiene una referencia al objeto al cual pertenece, de la misma forma contiene una colección de Calendarios

La clase **clsCalendariParades** tiene una serie de parámetros que se utilizan para configura un calendario:

```
Public HoraInici As clsDataTemps
Public HoraFi As clsDataTemps
Public DataAlta As clsDataTemps
Public DataModificacio As clsDataTemps
```

```
Public HoraDiaInici As String
Public HoraDiaFi As String
Public DiaMes As Long
Public IntMin As Long
Public DiaSetmana As String

Public bHoraDia As Boolean
Public bDiaMes As Boolean
Public bIntMin As Boolean
Public bDiaSetmana As Boolean

Public bActualizado As Boolean

Public MinIntervalEntreExec As Long
```

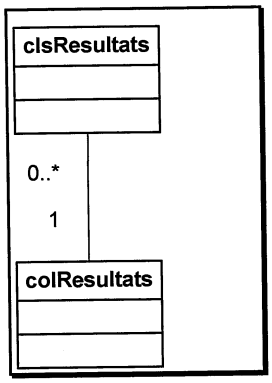
5.3.1.7 Módulo de gestión de resultados

Este módulo se encarga de gestionar los resultados obtenidos por la monitorización del ISM, su envío a la base de datos y posterior envío al servidor central por medio del Web Service.

Debe tener las siguientes funcionalidades:

- Almacenamiento en memoria de los resultados obtenidos por cada test
- Mantener los resultados en caso de parada o caída del servicio.
- Poder recuperar los datos de la base de datos local
- Enviar los resultados al repositorio central, mediante base de datos o Web Service
- Debe ser capaz de enviar los resultados sin esperar la respuesta del servidor
- Debe poder comprobar, una vez enviado un resultado, si se ha insertado correctamente

El diagrama de clases de este módulo es el siguiente:



La clase colResultats dispone de una serie de funciones para gestionar los resultados:

- **AfegirResultat**, añade una nueva instancia de la clase clsResultats a la colección
- **SaveWS**, enviar los resultados al servidor central
- **SaveWSAsincrono**, envía los resultados de forma asíncrona

5.3.1.8 Módulo de gestión de alertas a móvil/correo electrónico

El módulo de envío de alertas se debe encargar de enviar los errores que recibe del MotorISM.

Para ello se dispone de una clase **clsEnvMov** que se encarga de gestionar el envío y los errores durante el envío.

Esta clase dispone de métodos como los siguientes

- **Configurar**, se utiliza para configurar los servidores de correo
- **EnviarErrores**, se encarga de leer de la base de datos local los errores a enviar y enviarlos a sus destinatarios

5.3.1.9 Notificación de alertas a OpenView

El módulo de alertas a OpenView se encarga de comunicar eventos del sistema a consolas de OpenView.

Debe tener las funcionalidades que se exponen a continuación:

- Lectura de resultados directamente de la base de datos

- Enviar errores a OpenView, informando de robot, test, ISP, script donde se ha producido el error. Enviar el error indicando el código interno y la descripción del error

- Enviar tiempos de respuesta por test y por script

5.3.1.10 Módulo de gestión de claves dinámicas

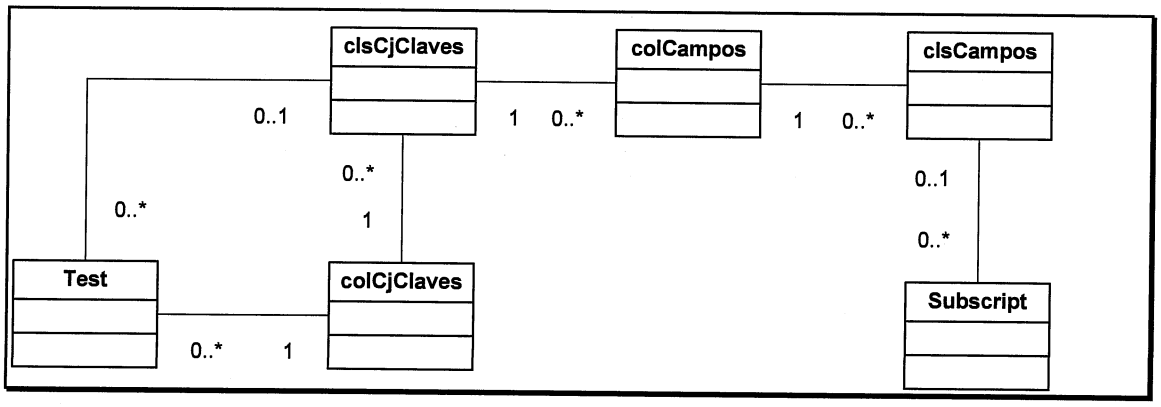
El presente módulo se encarga de gestionar las claves dinámicas de los tests a la hora de ejecutar una acción con un paso dinámico.

Para ello debe tener las siguientes funcionalidades:

- Debe contener todas las claves dinámicas presentes en la base de datos

- Cada clave dinámica debe tener sus campos configurados

La estructura de este módulo es la siguiente:



Este módulo se utiliza desde un Test, cada uno de los tests contiene una instancia de la clase colCjClaves
A la hora de realizar una acción si un subscript tiene asociado un campo se obtendrá el valor dinámicamente

5.3.1.11 Módulo de lectura de ficheros INI

Este módulo se debe encarga de leer el fichero de configuración de la Consola y el Agente que se utilizara para leer todos los parámetros de inicio.

Las funcionalidades necesarias son las siguientes:

- Abrir un fichero de configuración
- Leer parámetros, tanto texto como números
- Leer parámetros divididos en bloques

Este módulo se compone de una clase **clsFicheroINI**, que utiliza llamadas a sistema para abrir ficheros y leer parámetros. Estas llamadas se encuentran en la librería Kernel32 y son las siguientes:

```
Function GetPrivateProfileString Lib "Kernel32" Alias "GetPrivateProfileStringA"
(ByVal lpApplicationName As String, ByVal lpKeyName As Any, ByVal lpDefault As
String, ByVal lpReturnedString As String, ByVal nSize As Long, ByVal lpFileName
As String) As Long
```

```
Function GetPrivateProfileSection Lib "Kernel32" Alias "GetPrivateProfileSectionA"
(ByVal lpAppName As String, ByVal lpReturnedString As String, ByVal nSize As
Long, ByVal lpFileName As String) As Long
```

```
Function GetPrivateProfileInt Lib "Kernel32" Alias "GetPrivateProfileIntA" (ByVal
lpApplicationName As String, ByVal lpKeyName As String, ByVal nDefault As Long,
ByVal lpFileName As String) As Long
```

```
Function OpenFile Lib "Kernel32" (ByVal lpFileName As String, lpReOpenBuff As
OFSTRUCT, ByVal wStyle As Integer) As Integer
```

5.3.1.12 Módulo de conexión con RAS

El módulo de conexión a ISP's se encarga de realizar la conexión mediante el acceso telefónico a redes.

Las funcionalidades son las siguientes:

- Realizar conexiones a un ISP mediante el RAS
- Controlar los errores en la conexión
- Cerrar conexiones abiertas
- Controlar que el ISP este disponible en la máquina

Para realizar estas funcionalidades se disponen una clase **clsModem**, la cual debe utilizar las funcionalidades de la librería WinInet para gestionar las conexiones

5.3.1.13 Módulo de seguridad/encriptación

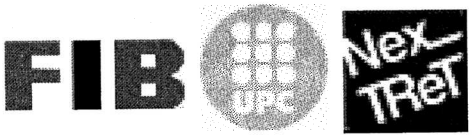
El actual módulo se encarga de controlar la licencia del ISM, tanto de la consola como del agente, y se encarga de encriptar y desencriptar las licencias introducidas.

Este módulo se encarga de realizar las siguientes funcionalidades:

- Comprobar la licencia de agente y consola
- Desencriptar/encriptar las licencias
- Permitir/denegar la ejecución de ISM

Sobre este módulo no se va a entrar en detalle en las clases, ya que se utilizan una serie de clases implementadas por segundas personas.

Se utiliza algoritmos de encriptación Blowfish y utilizando Radix64.



5.3.1.14 Control del registro de Windows

El módulo se encarga de interactuar con el registro de Windows para realizar las tareas de:

- Guardar la licencia en el registro
- Leer la configuración del registro

Debe tener las siguientes funcionalidades:

- Leer las claves
- Leer valores alfanuméricos, DWORD, binarios
- Cargar las claves
- Cargar valores alfanuméricos, DWORD, binarios

Par esto se dispone de una clase **ManageRegistry**, que dispone de funciones específicas para guardar leer cada uno de los tipos.

5.3.1.15 Módulo de FindAlerts

Este módulo se debe ejecutar como una aplicación independiente tanto de la consola como del agente.

Se encarga de controlar la aparición de ventanas emergente que puedan aparecer durante la navegación y que pueden parar la navegación.
Estas ventanas son:

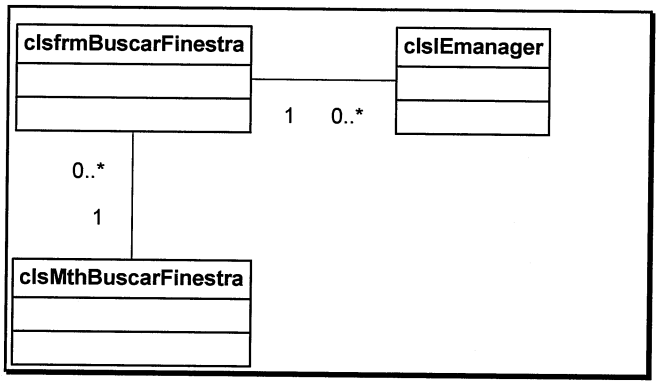
- Ventana de petición usuario/password
- Ventanas de mensajes emergentes.
- Avisos de seguridad
- Prompts

Debe tener las siguientes funcionalidades:

- Detectar la aparición de ventanas
- Cerrar las ventanas que aparecen

- Informar del tipo de ventana aparecida
- Interactuar con las ventanas, para presionar botones e introducir valores en caja de texto.

Las clases que interviene en este módulo son las siguientes:



La clase **clsMTHBuscarFinestra** es la clase de entrada para la consola y el agente, y desde está, a través de la instancia de **clsfrmBuscarFinestra** se debe arrancar la búsqueda de ventanas activa.

Esta búsqueda debe ser activa y debe pararse. En caso de encontrar una ventana informa a la clase clsMthBuscarFinestra por medio de eventos y está informa al agente y la consola que lo hayan creado.

5.3.2 Funcionalidades

Se expone, a continuación, una selección de funcionalidades de las llevadas a cabo por el autor del documento, concretamente se ha seleccionado una de las funcionalidades sirviendo de ejemplo para el resto de funcionalidades y, evitando así, extender excesivamente el documento con el diseño de la totalidad de funcionalidades. Éstas son las siguientes:

- Mantenimiento de Robots.
- Mantenimiento de Test.
- Mantenimiento de Servicios
- Mantenimiento de ISP
- Mantenimiento de Personas
- Mantenimiento de Usuarios/Test
- Mantenimiento Agente

5.3.2.1 Mantenimiento de Test

Se ha elegido está funcionalidad ya que dispone de casi la totalidad de casos que nos podemos encontrar en la aplicación. Desde el mantenimiento de test se pueden realizar las siguientes acciones:

- Alta de un test
- Eliminar un test
- Modificar un test
- Grabar un test
- Play de un test

Alta de un test

El elemento básico de navegación es el test, por lo tanto, la alta de un test es el punto inicial para crear una navegación y su futura navegación.

Para dar de alta un nuevo test se debe informar de los siguientes datos:

Nombre	*Descripción*	*Tipo*	*Defecto*
Nombre	Nombre del test	Texto	"Ninguno"
Descripción	Descripción del test	Texto	"Ninguno"
Intentos	Número de intentos hasta informar un error	Número	2
Tiempo de Espera	Tiempo de espera entre test	Número	0
Tipo de Test	Indica que tipo de test va a corresponder al actual	Combo	Test Internet
Fichero	Indica el fichero del html original, obsoleto	Texto	"Ninguno"
Validar HTML	Indica que se va a validad el HTML	CheckBox	Deshabilitado

Acciones

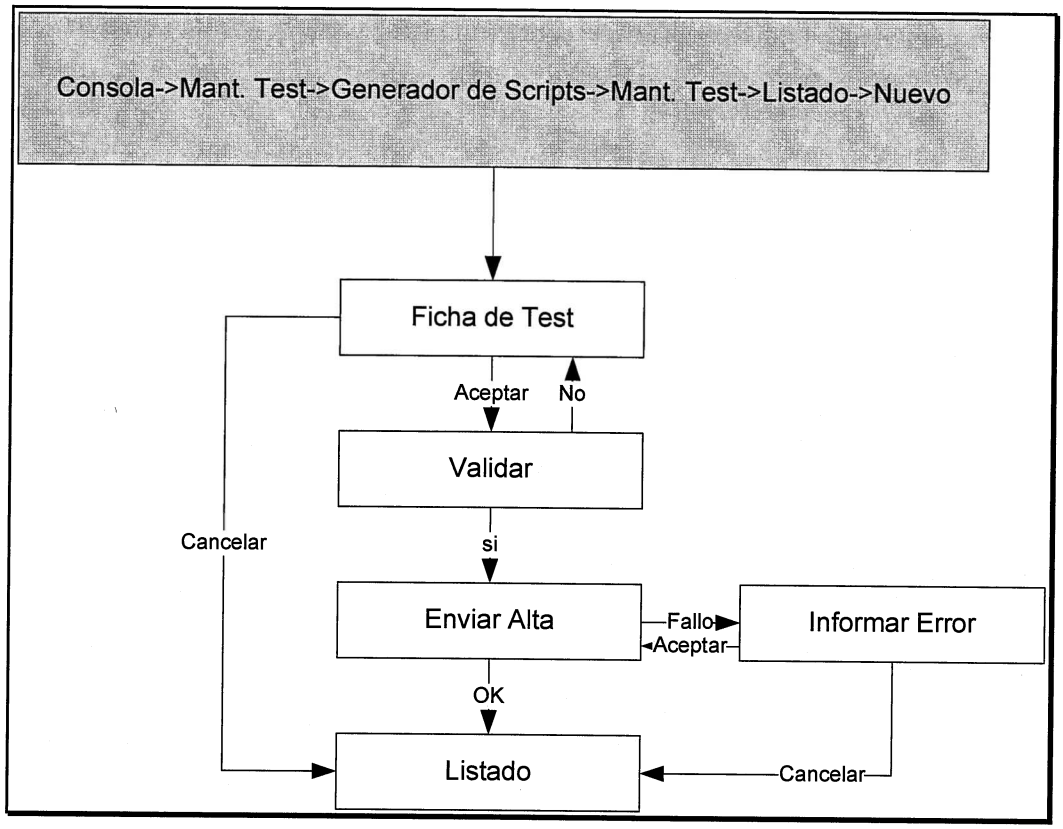
- **Aceptar**, da de alta el test y lo envía por Web Service al repositorio central
- **Cancelar**, cancela el alta del test

Validaciones

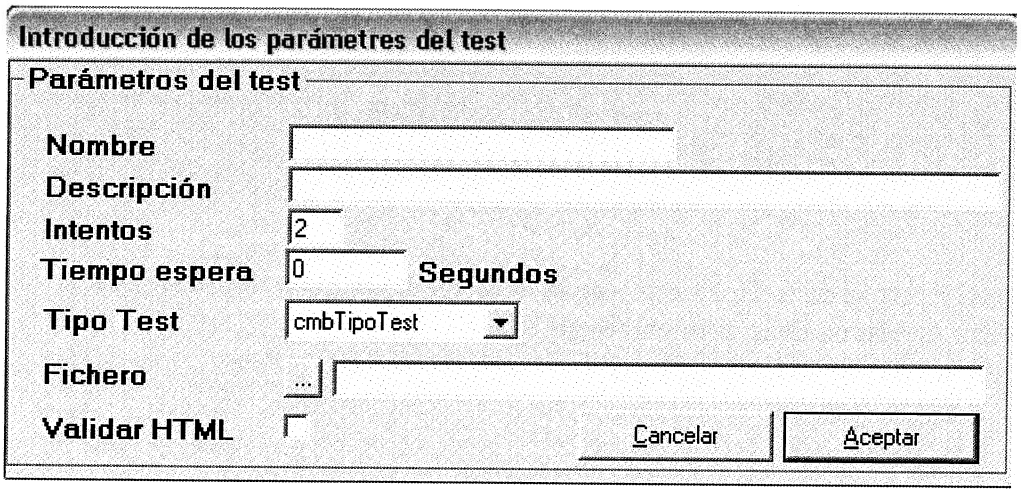
Las validaciones que se tienen en cuenta en esta funcionalidad son las siguientes:

- Formato de los datos introducidos, que se han informado en la tabla anterior
- Tiempo de espera e intentos positivos

Flujo de navegaciones

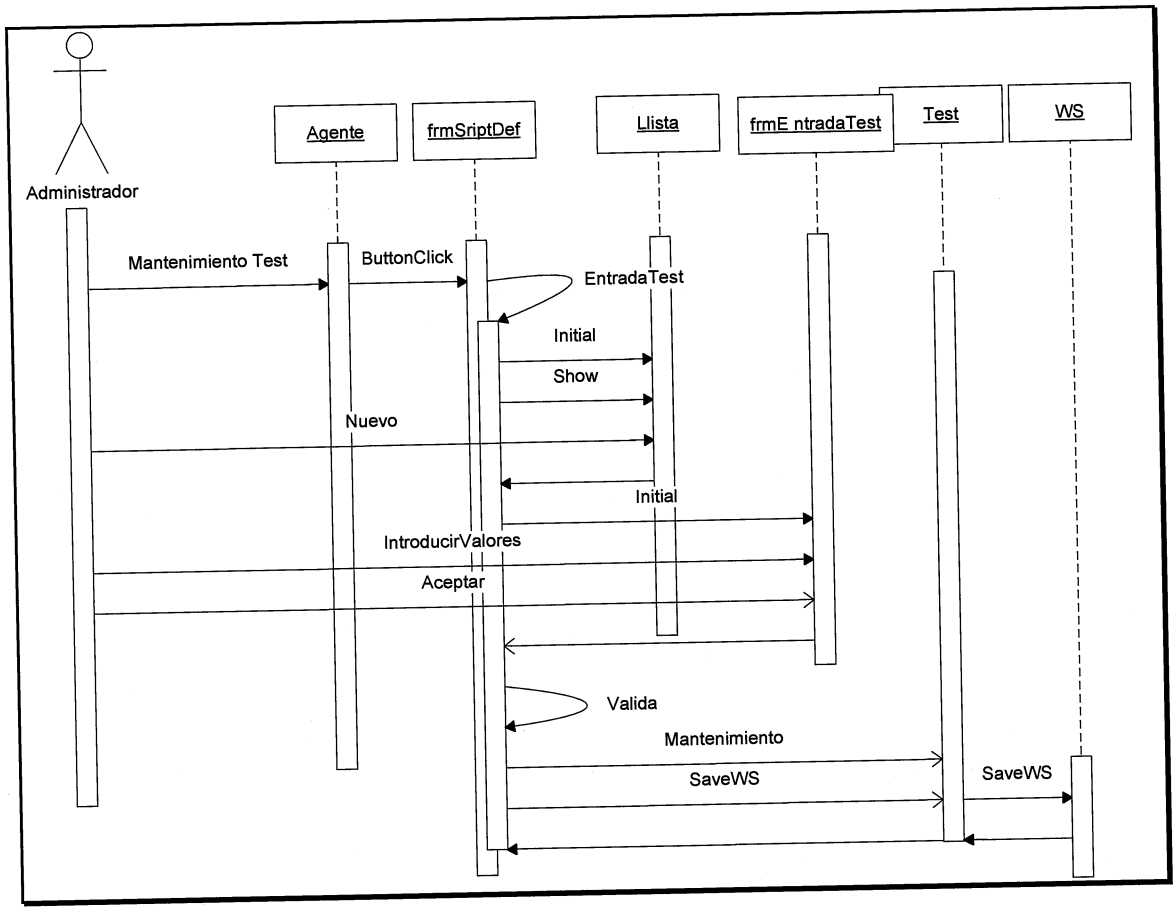


Interfaz de usuario



Diagramas de secuencia

El diagrama de secuencias siguiente es el que representa la secuencia de alta de un test.



Eliminar un test

Para eliminar un test se debe acceder al listado de test y seleccionar uno. Para que se pueda eliminar un test no debe estar en ningún árbol de ejecución.

Acciones

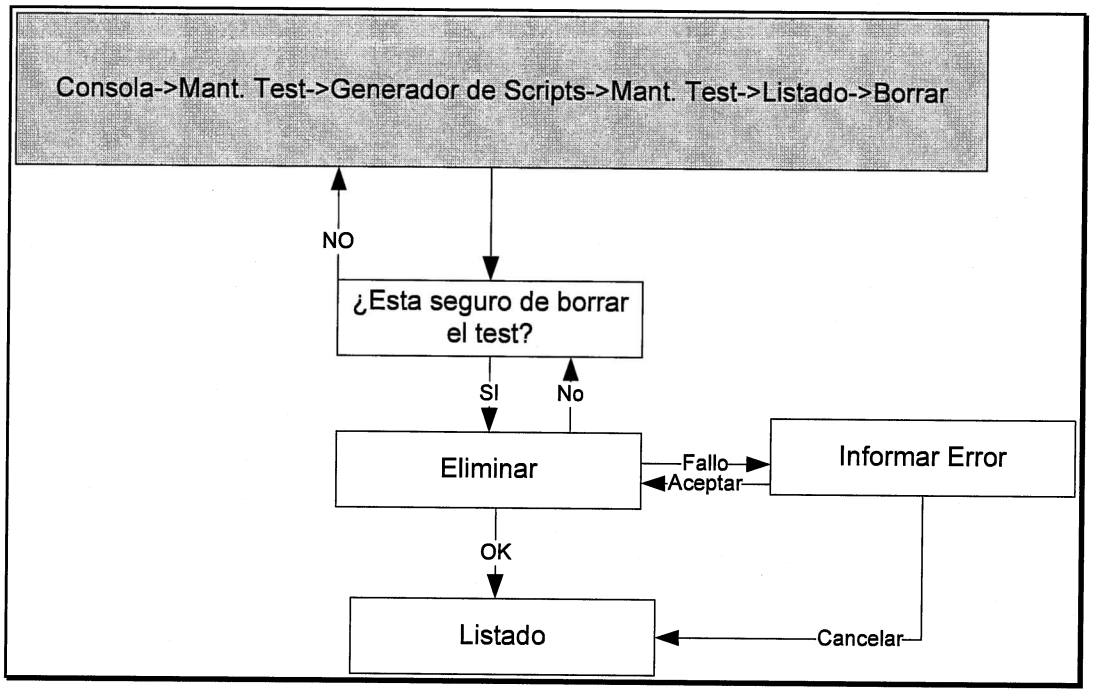
- **Borrar**, eliminar un test
- **Cancelar**, cancelar la acción de eliminar, ocultado el listado

Validaciones

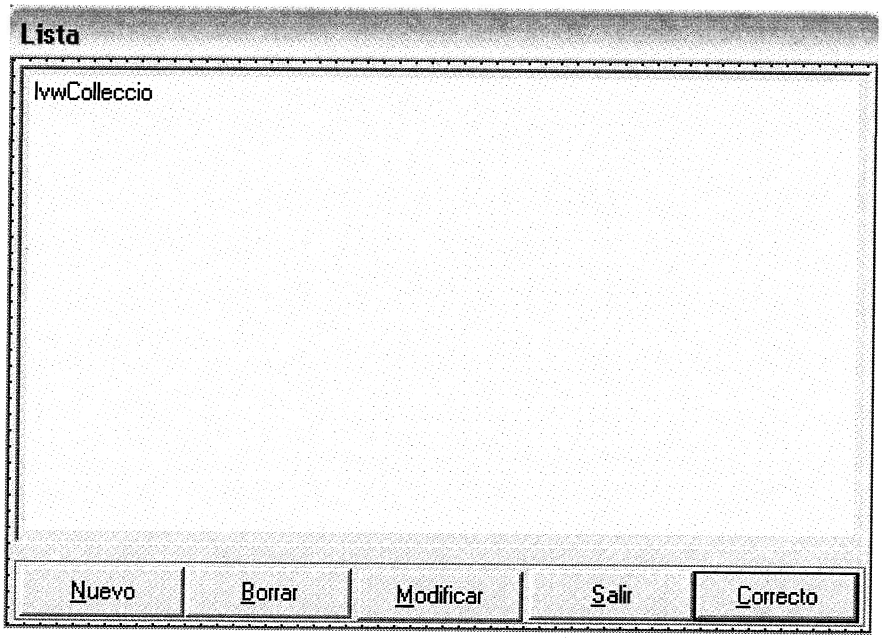
Las validaciones que se tienen en cuenta en esta funcionalidad son las siguientes:

- Comprobar que el test no este en algún árbol de un Robot

Flujo de navegaciones

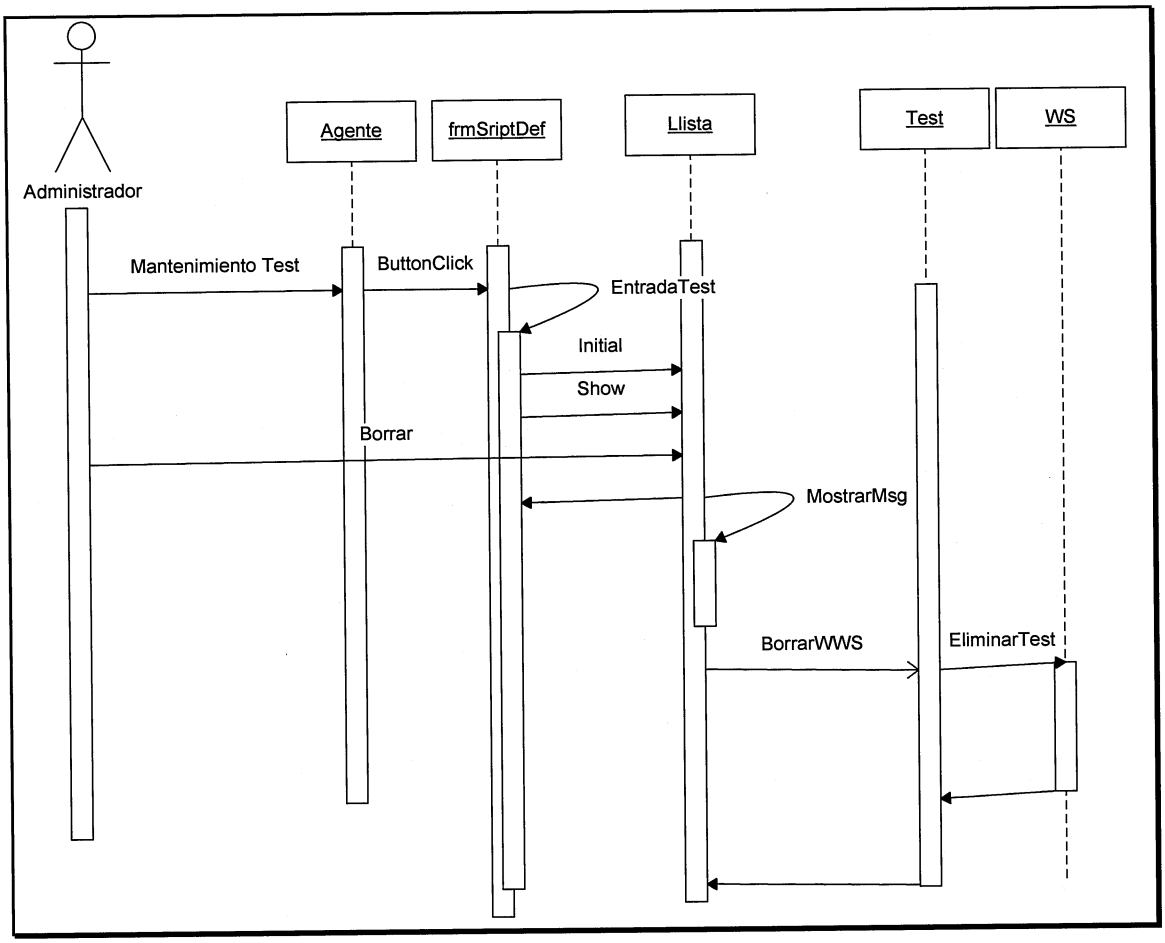


Interfaz de usuario



Diagramas de secuencia

El diagrama de secuencias siguiente es el que representa la secuencia de alta de un test.



Modificar un test

Para modificar un test, se realiza con la misma operativa que el alta.

Para modificar un test se debe informar de los siguientes datos:

Nombre	*Descripción*	*Tipo*	*Defecto*
Nombre	Nombre del test	Texto	"Ninguno"
Descripción	Descripción del test	Texto	"Ninguno"
Intentos	Número de intentos hasta informar un error	Número	2
Tiempo de Espera	Tiempo de espera entre test	Número	0
Tipo de Test	Indica que tipo de test va a corresponder al actual	Combo	Test Internet
Fichero	Indica el fichero del html original, obsoleto	Texto	"Ninguno"
Validar HTML	Indica que se va a validad el HTML	CheckBox	Deshabilitado

Acciones

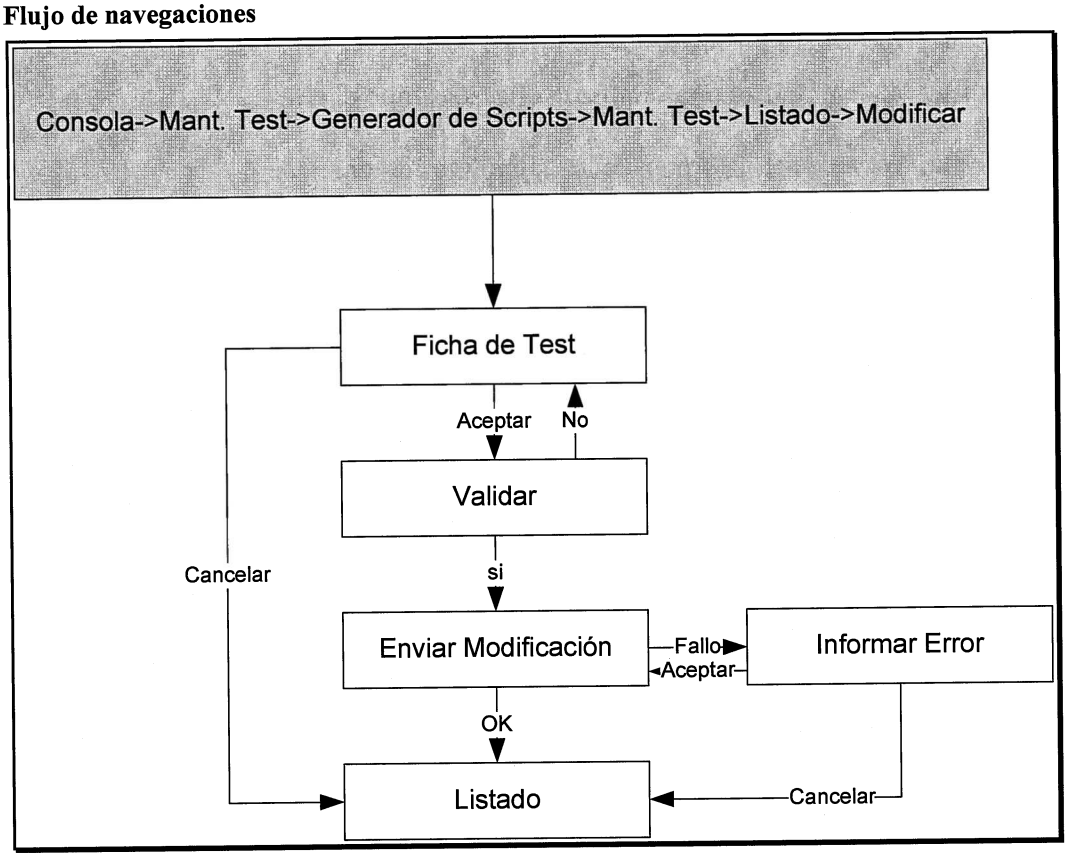
- **Aceptar**, modifica el test y lo envía por Web Service al repositorio central
- **Cancelar**, cancela la modificación del test

Validaciones

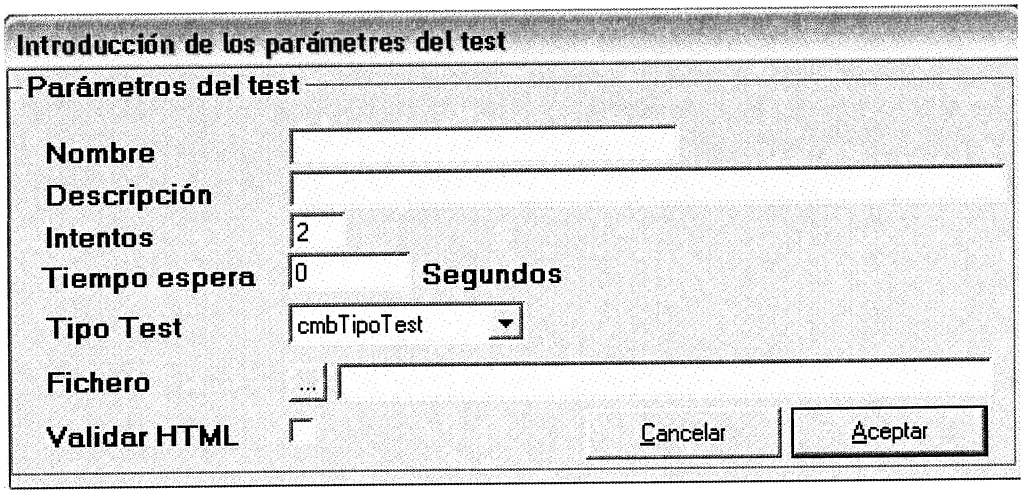
Las validaciones que se tienen en cuenta en esta funcionalidad son las siguientes:

- Formato de los datos introducidos, que se han informado en la tabla anterior
- Tiempo de espera e intentos positivos

Flujo de navegaciones

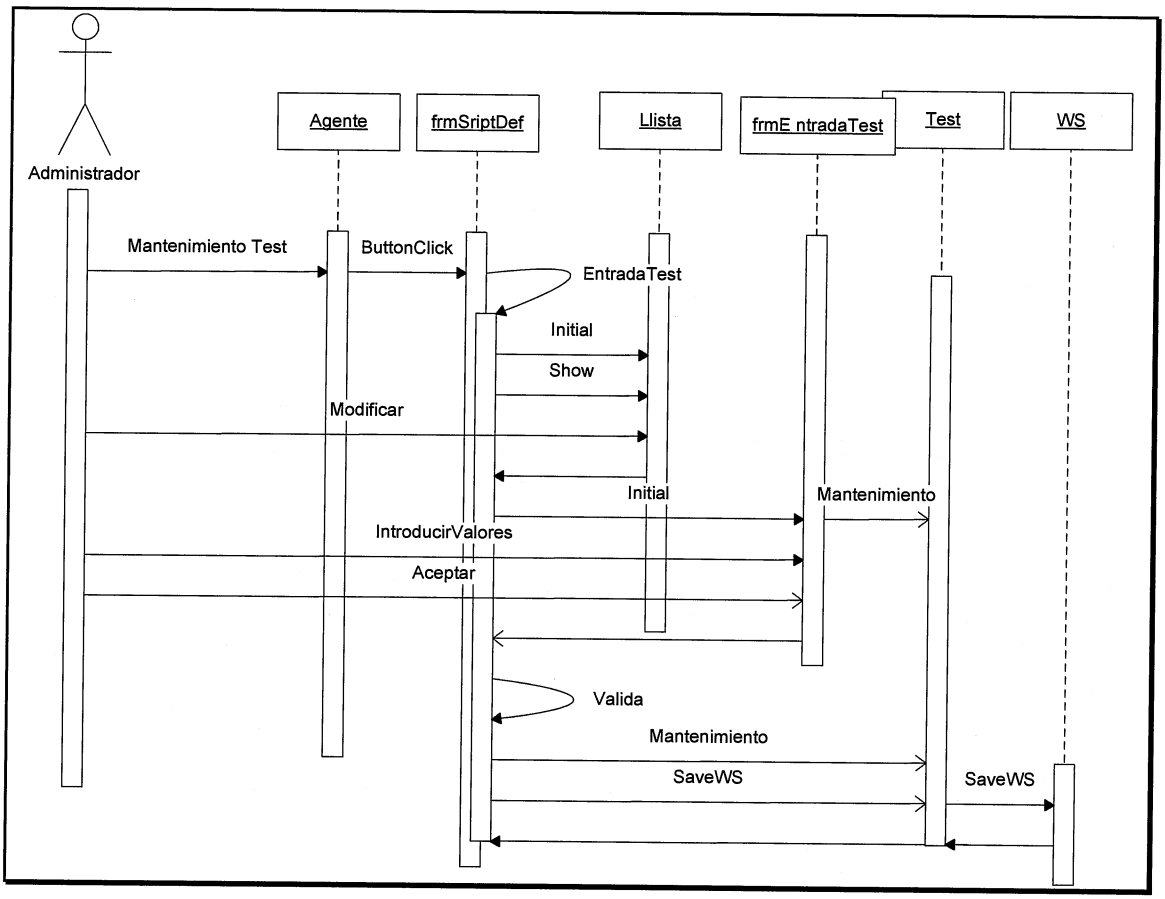


Interfaz de usuario



Diagramas de secuencia

El diagrama de secuencias siguiente es el que representa la secuencia de alta de un test.



Grabar un test

Para grabar un test, se debe primero seleccionar un test, previamente creado.

Acciones

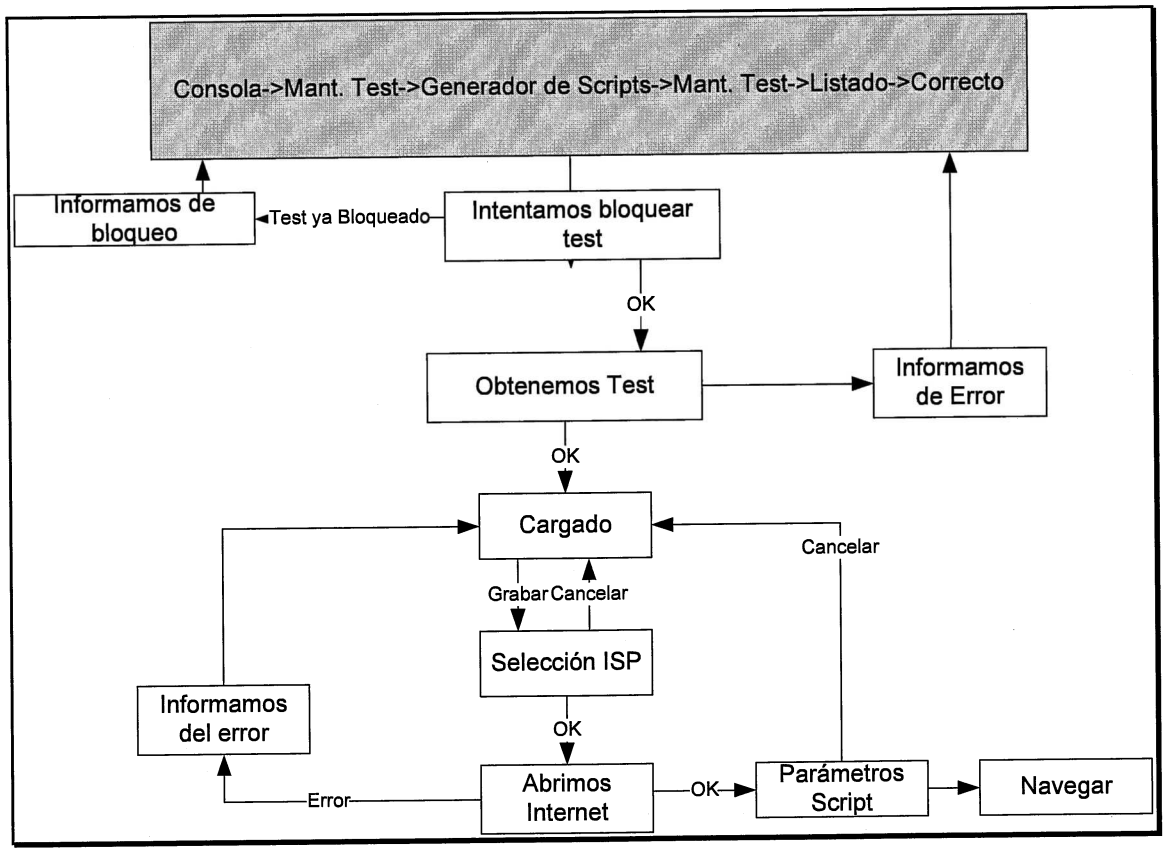
- **Grabar**, inicia el proceso de grabación de un test.
- **Cancelar**, cancela la grabación.
- **Seleccionar Test**, seleccionar un test de la lista de test.
- **Correcto**, seleccionar definitivamente el test para poder trabajar con él.

Validaciones

Las validaciones que se tienen en cuenta en esta funcionalidad son las siguientes:

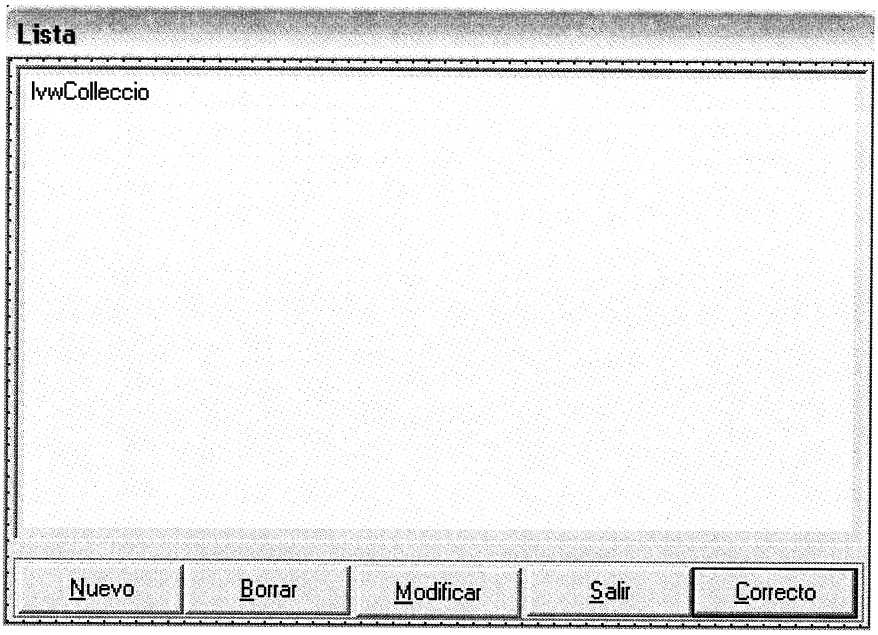
- El test no ha de estar bloqueado

Flujo de navegaciones

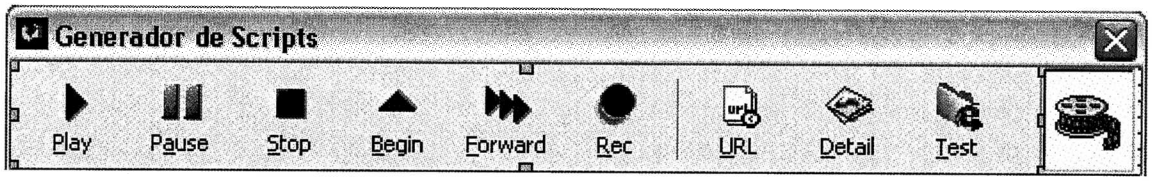


Interfaz de usuario

Listado de Test

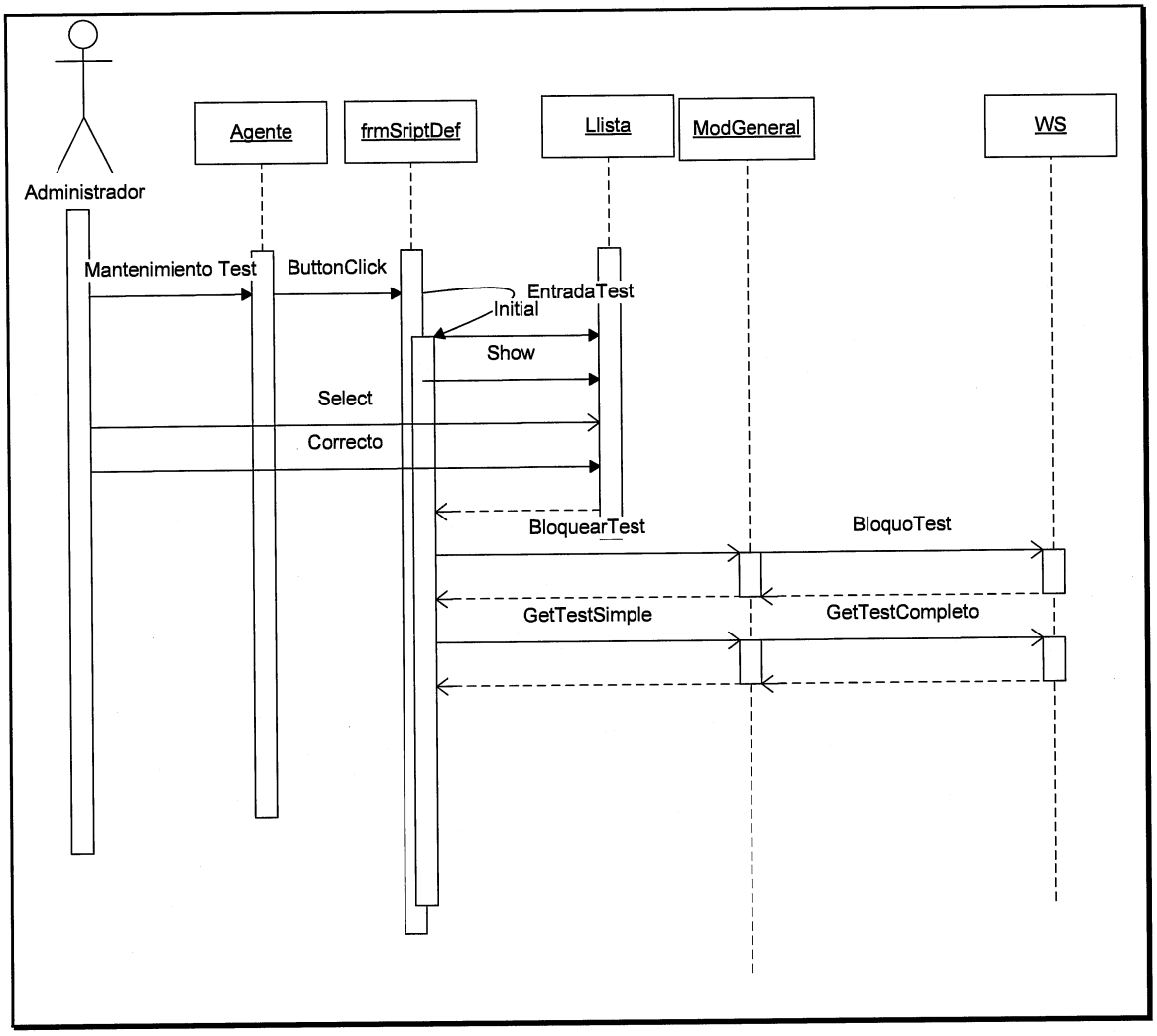


Generador de Scripts

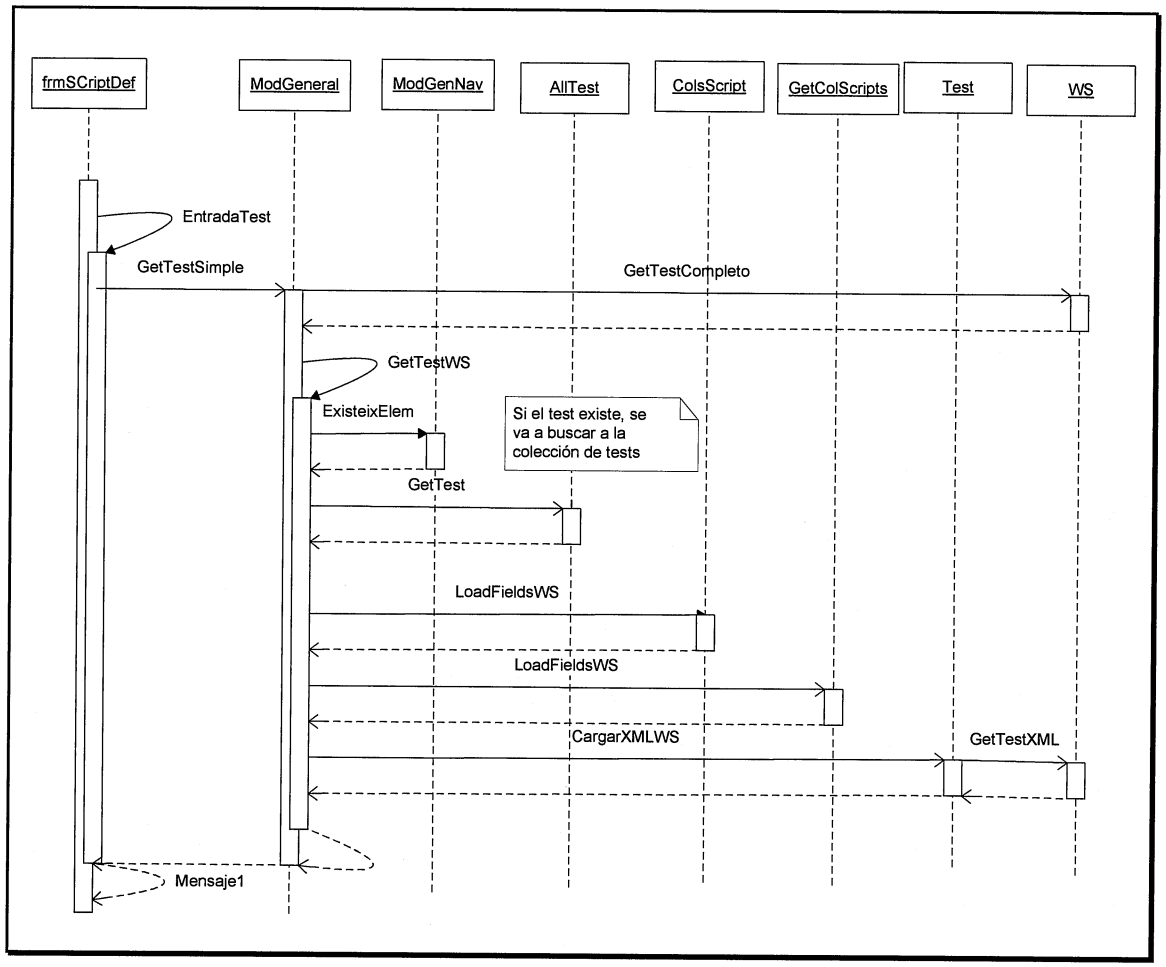


Diagramas de secuencia

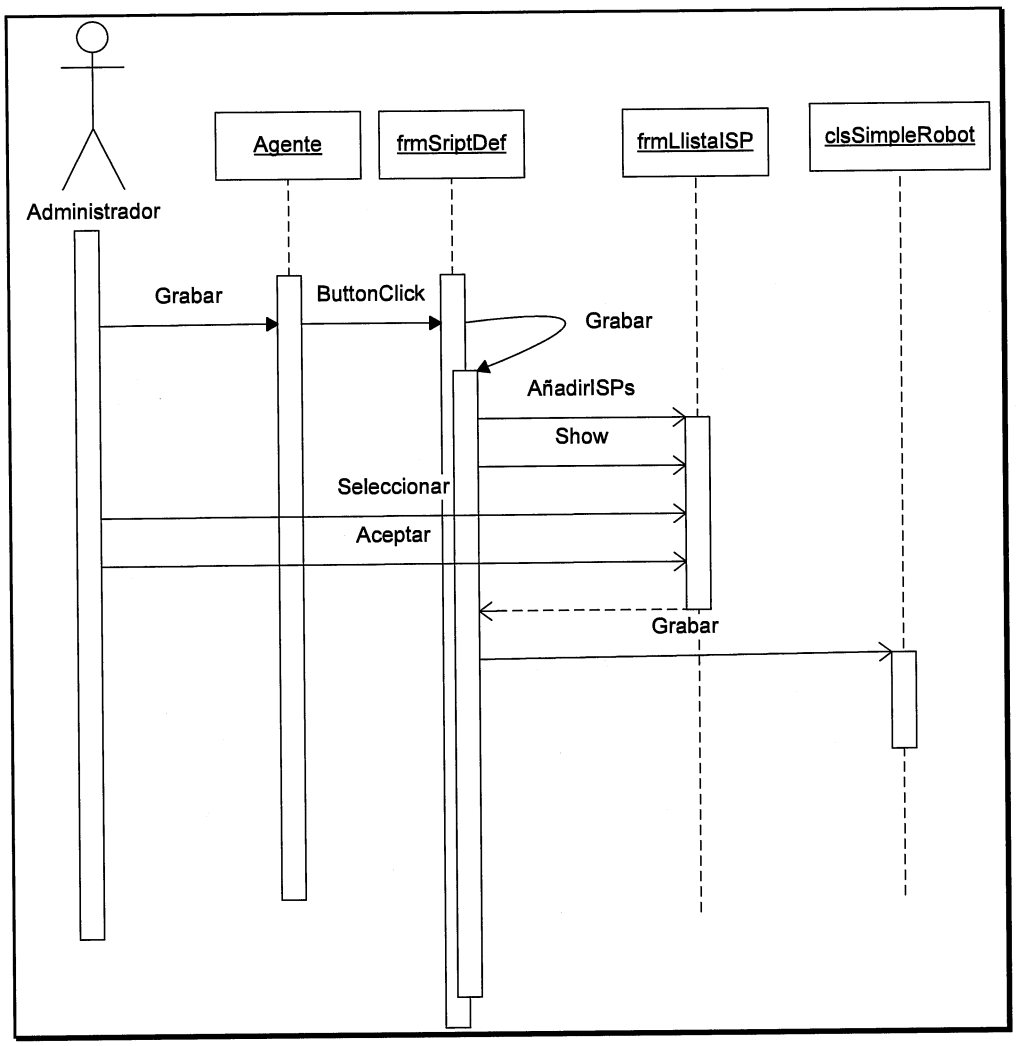
El diagrama de secuencias siguiente es el que representa la selección de un test para grabar.



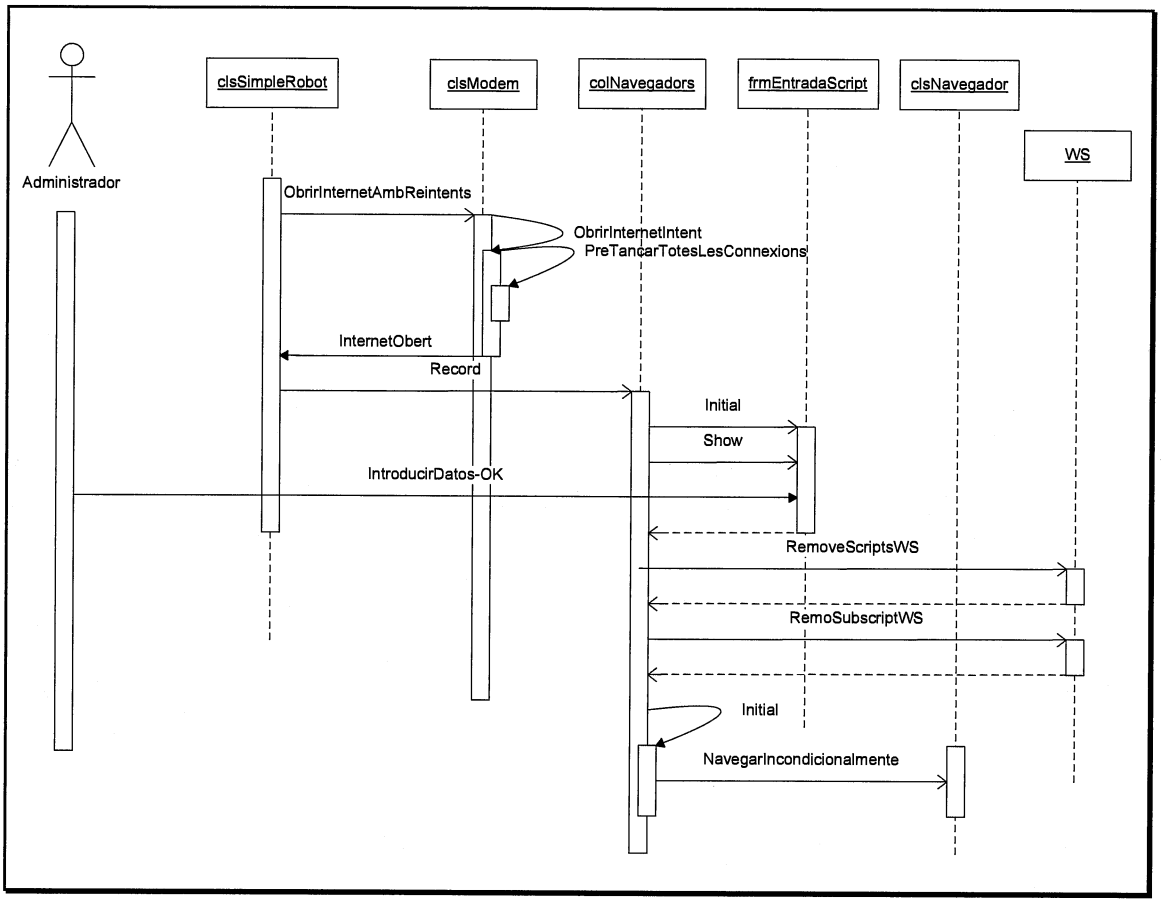
A continuación, se entra en detalle de la llamada a **GetTestSimple**



Una vez cargado el test en memoria, se muestra el diagrama de secuencias desde que el Administrador presiona el botón de Grabar de la venta del Generador de Scripts



A continuación, se muestra la llamada a **Grabar** de la clase **clsSimpleRobot**



La función, de la clase **clsNavegador**, **NavegarIncondionalmente** realiza las llamadas al navegador mediante la llamada
IE.Navigate2

Play de un test

Una vez grabado un test, los usuarios del ISM desean comprobar la navegación que han grabado.

Para ello, acceden con el botón de play a la ejecución de la prueba.

Acciones

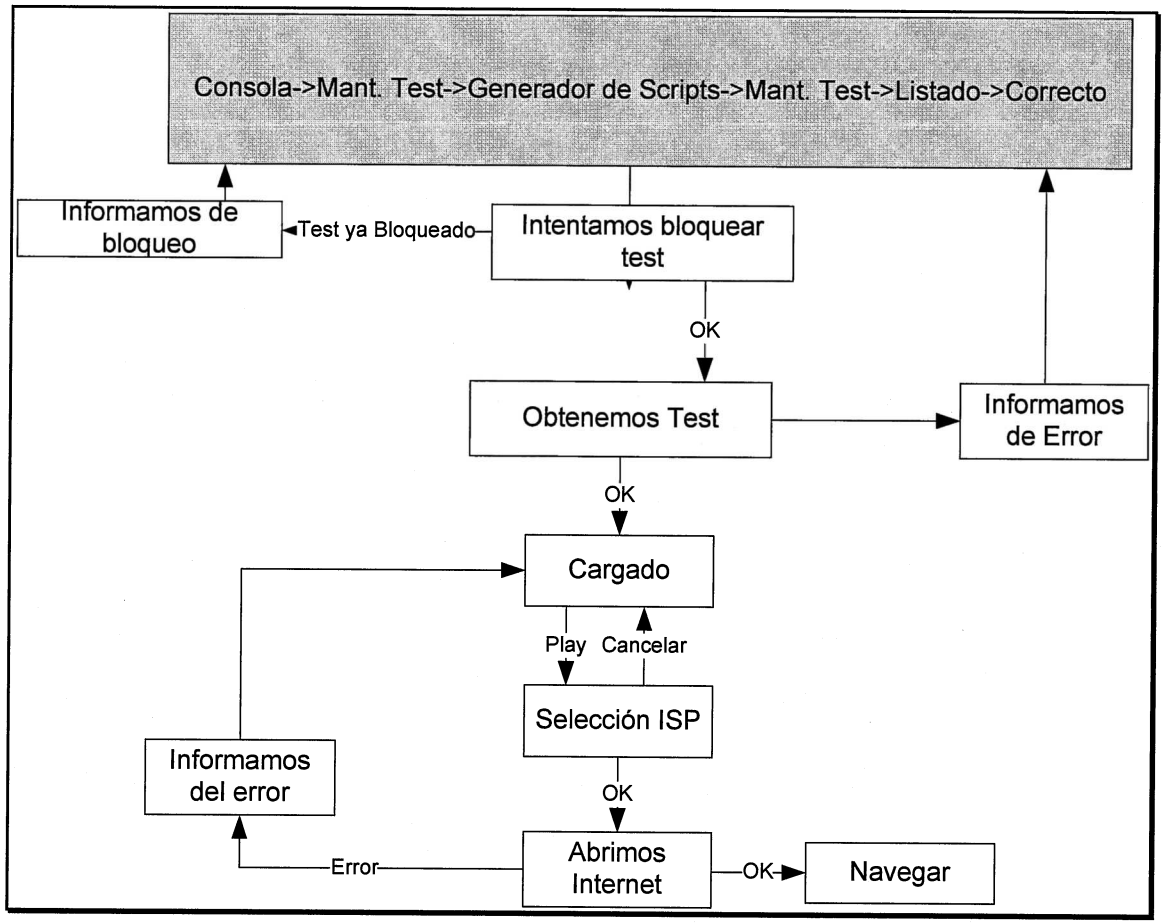
- **Play**, inicia el proceso de prueba de un test.
- **Cancelar**, cancela el testeo.
- **Stop**, para la navegación actual
- **Seleccionar Test**, seleccionar un test de la lista de test.
- **Correcto**, seleccionar definitivamente el test para poder trabajar con él.

Validaciones

Las validaciones que se tienen en cuenta en esta funcionalidad son las siguientes:

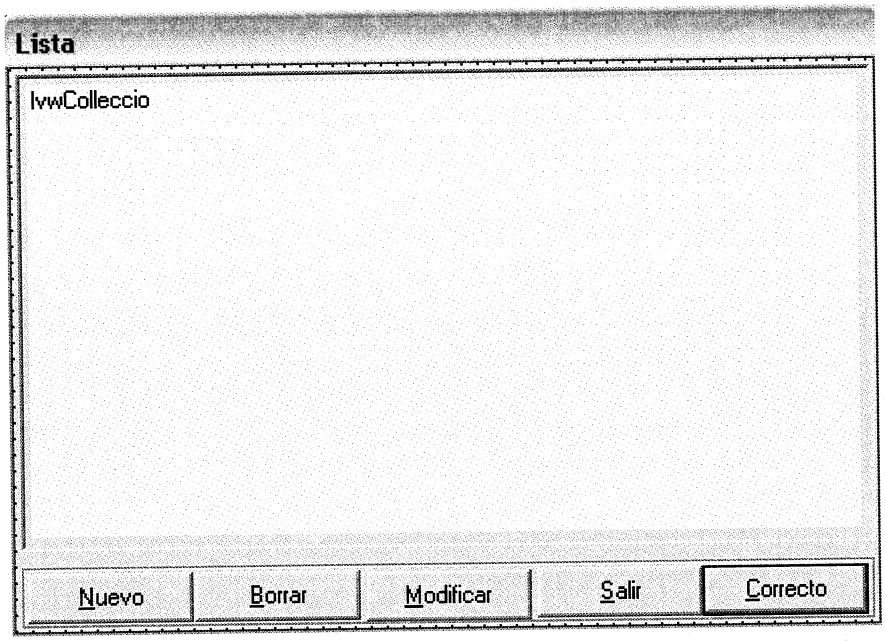
- El test debe tener scripts y subscripts para poder ejecutarse

Flujo de navegaciones

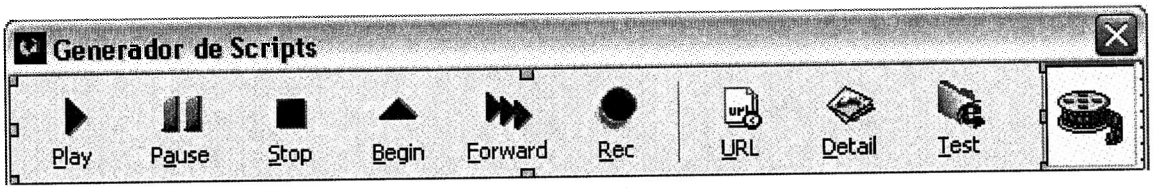


Interfaz de usuario

Listado de Test

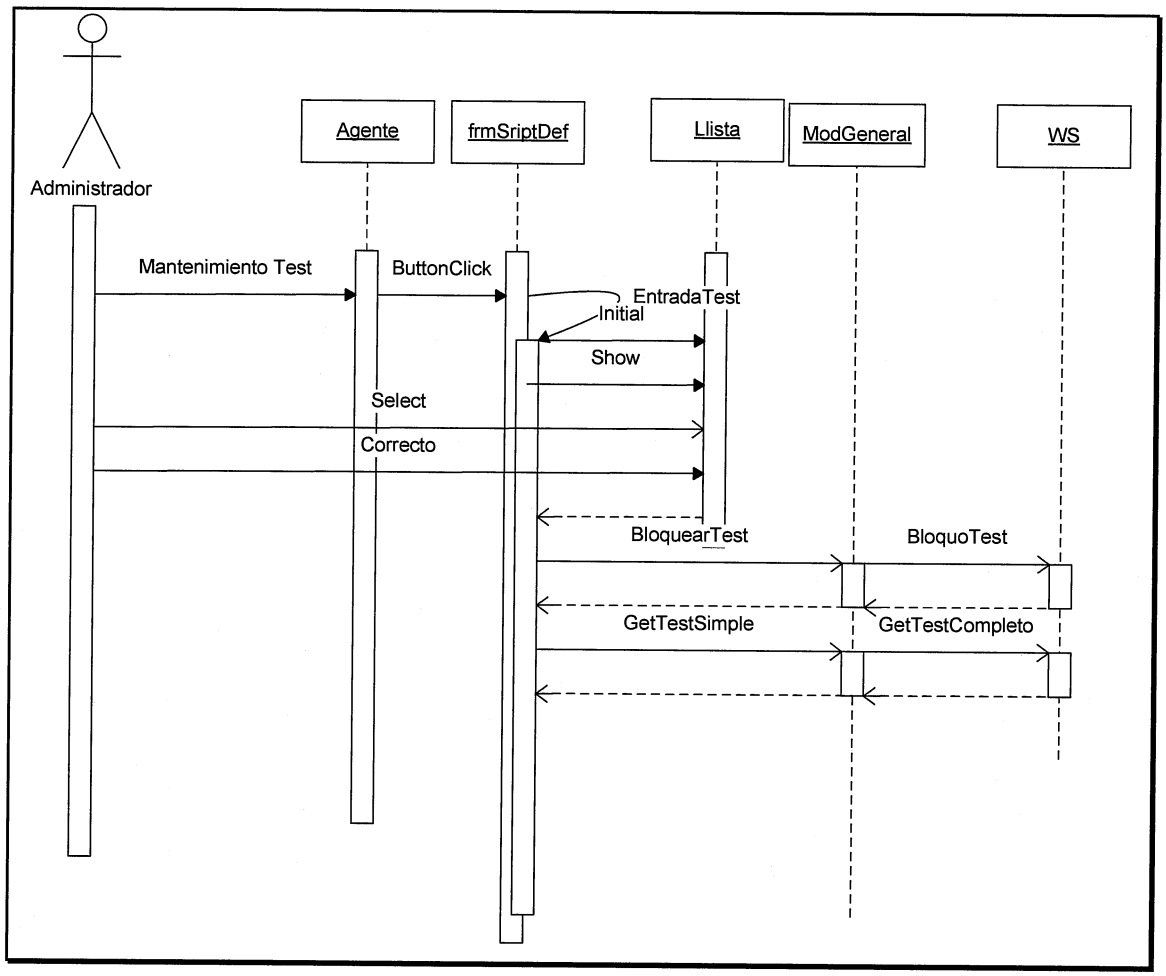


Generador de Scripts

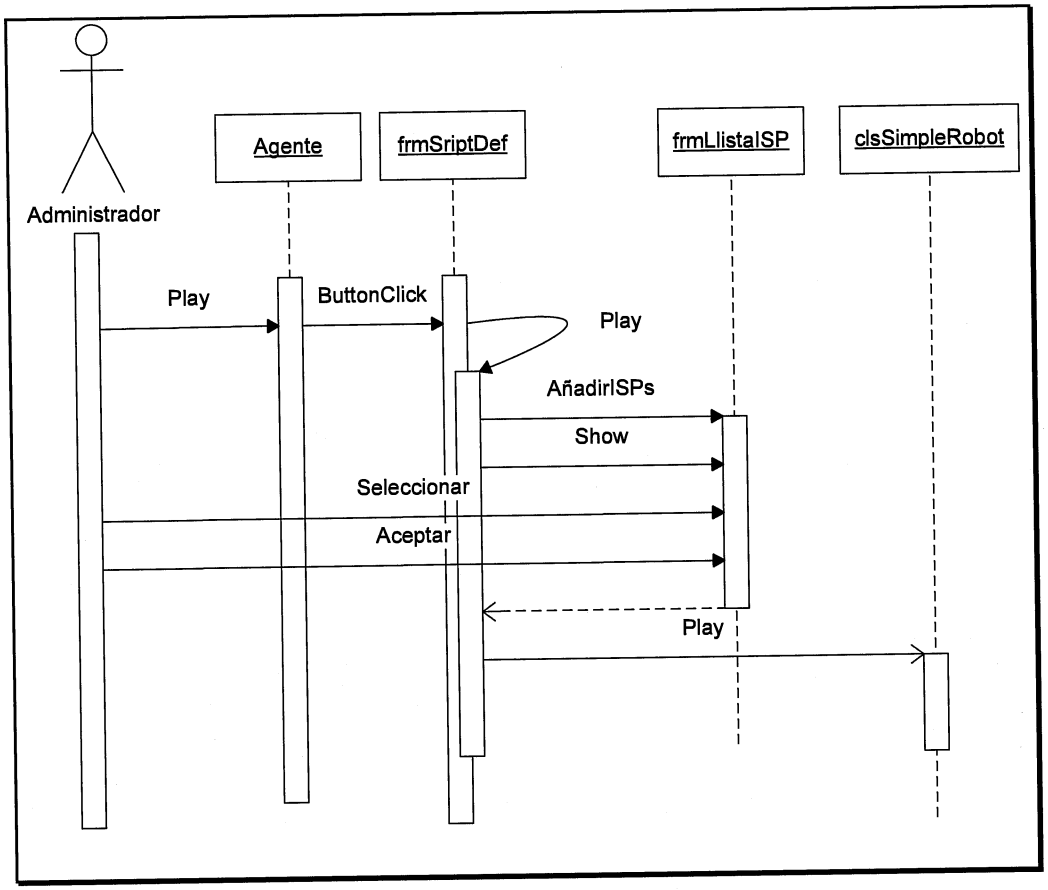


Diagramas de secuencia

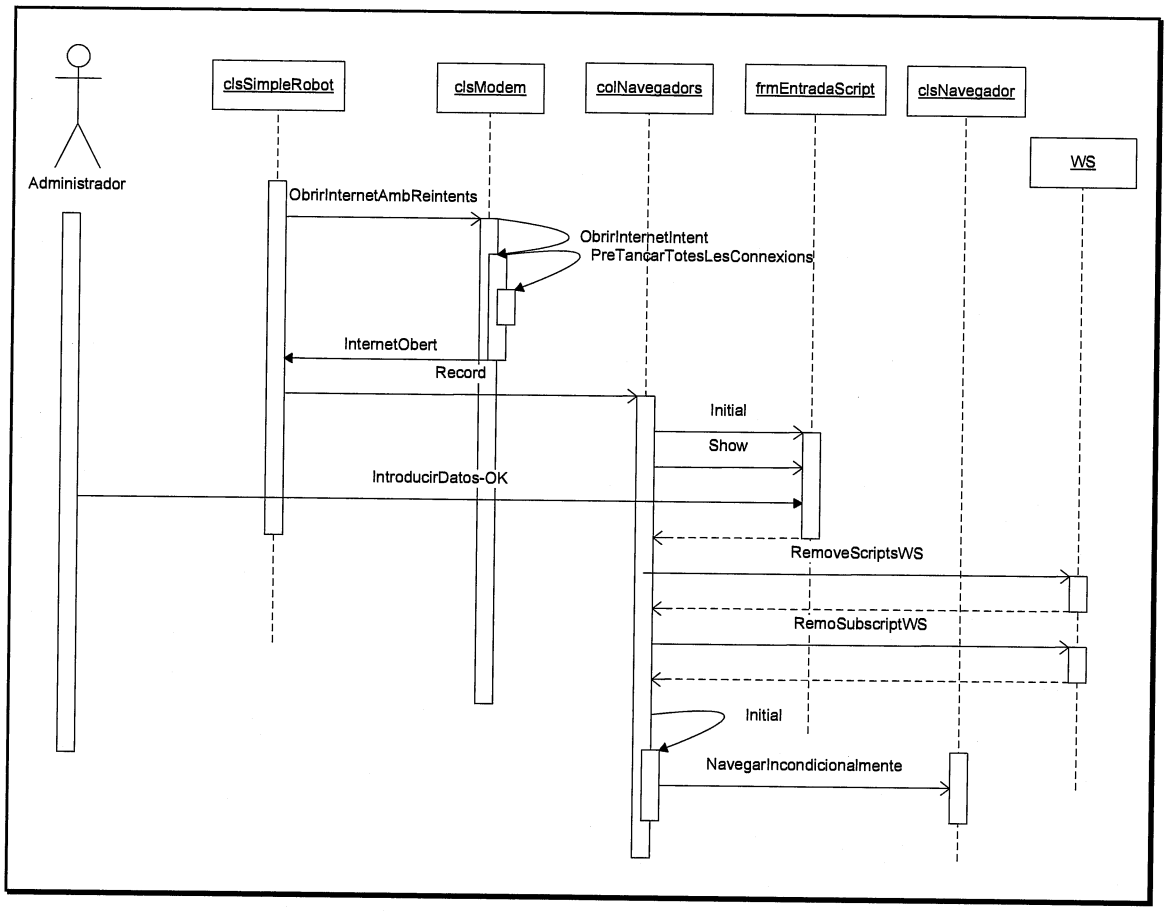
El diagrama de secuencias siguiente es el que representa la selección de un test para play.



A continuación, se muestra el diagrama de secuencias par realizar la funcionalidad de Play del test seleccionado con anterioridad.



El siguiente diagrama corresponde al Play del test en la clase **clsSimpleRobot**



5.4 Diseño de la base de datos

El diseño de la base de datos se divide en cuatro entornos entrelazados entre ellos, los cuales tienen la base en el árbol de ejecución:

1. Árbol de Ejecución, corresponde a los datos que se deben guardar para generar el árbol de ejecución de un robot.

2. Resultados de la aplicación, en este apartado el autor presenta las tablas que almacenan los datos de monitorización obtenidos por la aplicación así como los datos tratados y almacenados en el datawerehouse de la aplicación.

3. Reporting Web, en este apartado se presentan las tablas que se utilizan para gestionar el Reporting Web donde se presentan los datos.

4. Tablas Auxiliares, en el presente apartado se presentaran las tablas auxiliares que se utilizan para el funcionamiento de las diferentes funcionalidades de la aplicación, como son envío de SMS o correos electrónicos de alerta, claves dinámicas y frases de validación.

De cada una de las tablas que se detallarán a continuación, se explicita el nombre de la tabla que tendrá en el gestor de base de datos, y una breve explicación de su finalidad. De igual modo, también se detalla de cada uno de sus campos la siguiente información:

- Nombre. Nombre del campo que se le pondrá en la tabla del gestor relacional.
- Tipo de campo.
- Tamaño. Si se trata de un campo de tipo texto, indica el tamaño máximo de éste.
- Obligatorio. Indica si el campo es de obligado cumplimiento.
- PK (Primary Key), FK (Foreign Key). Valor que indica si el campo forma parte de la primary key de la tabla, o si la columna es una foreign key de otra tabla (en ese caso, se indica el nombre de la tabla y el campo, de la foreign key)