Universitat Autònoma
de Barcelona

AUTOMATIZACIÓN DE TESTING

Memòria del projecte

d'Enginyeria Tècnica en

Informàtica de Sistemes

realitzat per

Javier Venegas Adame

i dirigit per

Daniel Franco Puntes

Escola Universitària d'Informàtica

Sabadell, Setembre de 2009

El sotasignat, *Daniel Franco Puntes*,

professor/a de l'Escola Universitària d'Informàtica de la UAB,

CERTIFICA:

Que el treball al que correspon la present memòria
ha estat realitzat sota la seva direcció
per en *Javier Venegas Adame*

I per a que consti firma la present.
Sabadell, setembre de 2009

Signat: *Daniel Franco Puntes*

El sotasignat, *Joan Gutierrez Muñoz*,
de *SAGE Logic Control*,

CERTIFICA:

Que el treball al que correspon la present memòria
ha estat realitzat sota la seva supervisió
per en *Javier Venegas Adame*

I per a que consti firma la present.
Sabadell, setembre de 2009

Signat: *Joan Gutierrez Muñoz*

TABLA DE CONTENIDOS.

Capítulo 1.

Introducción.

En este capítulo muestra una exposición informativa acerca de los contenidos del proyecto. Las partes que lo componen son las siguientes:

- **Introducción.** Breve exposición de contenidos del proyecto.
- **Estado del arte.** En que se tecnología se basa el alumno para poderlo llevar a cabo.
- **Acercamiento histórico.** Breve exposición de hitos conseguidos hasta la fecha.
- **Objetivos.** Exposición de hasta donde se quiere llegar.
- **Organización de la memoria.** Breve exposición de los contenidos de este documento.

1.1 PRESENTACIÓN.

Sage España es el líder en desarrollo de soluciones de gestión empresarial para pequeñas y medianas empresas, desde el software y los servicios a la consultoría y la formación.

La compañía está integrada por tres divisiones: Gran Empresa, Mediana Empresa y Pequeña Empresa, cada una de ellas especializada en un segmento de mercado concreto para ofrecer las soluciones y servicios que mejor se adapten al cliente.

La división en la que se encuentra la sede de Sabadell (*Sage Logic Control*) es Mediana Empresa. Su objetivo es ofrecer soluciones avanzadas y adaptadas a las necesidades concretas de cada empresa, como la colección Sage *Logic Class*, última generación de soluciones ERP para pequeñas y medianas empresas, o la gama Sage Profesional Class, especialmente diseñada para los Despachos Profesionales.

Antes de lanzar una nueva aplicación al mercado, hay que hacerle una colección de pruebas para ver su correcto funcionamiento. El departamento encargado es el departamento de Calidad (QA). A la hora de hacer estas pruebas, se invierte demasiado tiempo ya que se hace de forma manual.

Sage Logic Control considera oportuno desarrollar unos tipos de test automatizados que reduzcan considerablemente la complejidad que supone hacer todo tipo de pruebas manualmente para unas funcionalidades que tiene la aplicación *Logic Class*: el *Cálculo de Nómina* y el *Cálculo de Seguros Sociales.*

1.2 ESTADO DEL ARTE.

Se tiene como referencia una colección de test ya implementados sobre una aplicación: *Logic Win Global*. La razón por la que se tienen en cuenta, es que fueron desarrollados con las mismas herramientas con las que se pretende llevar a cabo la implementación de nuevos test.

Concretamente, lo que se pretende es desarrollar una serie de test automáticos para *Logic Class,* pero la idea de Sage es, en un futuro, poder desarrollar la automatización a gran escala, es decir, poder testear las diferentes aplicaciones de forma automática de tal forma que facilite la detección y corrección de errores que puedan surgir al añadir nuevas funcionalidades.

1.3 ACERCAMIENTO HISTÓRICO A LA TECNOLOGÍA DEL PROYECTO.

Feb. 2007	Arranque del proyecto de Automatización de Testing Logic Win Global
Mayo 2007	Primeras ejecuciones desatendidas de ***pruebas de regresión*** [1] automáticas del Cálculo de Nómina 10 (versión 7.40). El sistema permite: • Programar la ejecución de las pruebas de regresión automáticas para que se ejecuten en diferido en cierta máquina. • Comprobar el progreso de la ejecución de dichas pruebas automáticas. • Revisar los resultados obtenidos. Estos resultados informan de las diferencias respecto a la versión de referencia.
Mayo 2007	Primeras ejecuciones desatendidas de pruebas de regresión automáticas del Cálculo de Nómina 10 (versión 7.40). El sistema permite: • Programar la ejecución de las pruebas de regresión automáticas para que se ejecuten en diferido en cierta máquina. • Comprobar el progreso de la ejecución de dichas pruebas automáticas. • Revisar los resultados obtenidos. Estos resultados informan de las diferencias respecto a la versión de referencia.
Sept. 2007	Puesta en marcha definitiva del test de Cálculo de Nómina 10(V. 7.60).

[1] Entendemos pruebas de regresión como **la comparación entre la versión a certificar y una anterior usada como referencia**.

A día de hoy se disponen 3 tests automáticos funcionando sobre dicha plataforma:

- Cálculo de Nómina 10
- Seguros Sociales de Nómina 10
- Impresos Oficiales 347

En la actualidad no existe ningún tipo de test automático para *Logic Class* por lo que las *pruebas de regresión* se hacen de forma manual, lo que surge la necesidad de implementar una automatización para esta aplicación. Teniendo en cuenta que *Logic Win Global* es una aplicación bastante antigua y lo que se pretende es que poco a poco vaya desapareciendo, la necesidad de automatizar *Logic Class* se considera imprescindible.

1.4 OBJETIVO Y ALCANCE PREVISTOS.

El objetivo del proyecto es el desarrollo de una herramienta de trabajo para un determinado departamento: *Calidad (QA*). A través de ella, se deben de poder ejecutar unos test automatizados sobre unas determinadas funcionalidades que tiene la aplicación *Logic Class*: el *Cálculo de Nómina y Seguros Sociales.*

Básicamente, hay que desarrollar una herramienta que facilite el trabajo para los *Técnicos de Calidad*. La ventaja es que *Calidad* no tiene por qué tener conocimientos de cómo se ha implementado todo, su labor es:

- Mantenimiento de los tests.
- Planificar la ejecución de los tests.
- Revisar los resultados de la ejecución de los tests.

Una vez finalizada la aplicación y visto su correcto funcionamiento, la finalidad es que pueda ser planificada desde cualquier estación de trabajo con acceso a la intranet de Sage.
Para ello, se añaden nuevas funcionalidades al website ya implementado para *Logic Win Global*, llamada Ciclo de Vida. Dicha aplicación, facilitará la programación de los test de forma automática y se puedan planificar para que sean ejecutados en una fecha y hora estipulados sin necesidad de estar presente físicamente mientras éstos se ejecutan.

En un principio, se probaran 2 funcionalidades de la aplicación *Logic Class* pero se puede ampliar para otras adaptando los cambios necesarios que sean requeridos.

1.5 ORGANIZACIÓN DE LA MEMORIA.

La estructura de este documento se basa en los siguientes capítulos:

- **Capítulo 2: Estudio de viabilidad.** Se analiza la viabilidad tanto técnica como económica para el desarrollo del proyecto. Se compone de los siguientes apartados.
- **Capítulo 3: Análisis y especificación.** Tiene como objetivo describir el diseño técnico del módulo de software objeto del proyecto de *automatización de testing*.
- **Capítulo 4: Fundamentos teóricos.** Se introducen algunos conceptos teóricos como ayuda para entender mejor el propósito de la realización de este proyecto.
- **Capítulo 5: Diseño.** Se expone detalladamente cada una de las partes de las que se compone para poder realizar las *pruebas de regresión.*
- **Capítulo 6: Implementación y pruebas.** Como ha se ha realizado e integrado y sus pruebas para ver su correcto funcionamiento.
- **Capítulo 7: Conclusiones.**

Capítulo 2.

Estudio de viabilidad.

En el siguiente capítulo se demostrará la viabilidad tanto técnica como económica para el desarrollo del proyecto. Los apartados de los que se compone son los siguientes:

- ***Objeto.*** Se expone la situación a resolver, los usuarios a los que va destinado así como los objetivos a cumplir.
- ***Descripción del sistema*****.** Descripción de la idea y funcionalidades del sistema con detalle.
- ***Recursos.*** Los recursos necesarios para poderlo llevar a cabo.
- ***Planificación.*** Una estimación de la duración para su desarrollo.
- ***Análisis de costos.*** El coste económico que tiene este proyecto.
- ***Conclusiones.*** Valoración final.

2. OBJETO.

2.1 Situación previa.

Como en cualquier empresa de hoy en día, a la hora de realizar un proyecto hay que analizar los costes y los beneficios que este obtendrá.

En este caso, el proyecto se presenta a *Dirección de Calidad*. En un principio, la previsión era la automatización para diferentes aplicaciones de las que dispone Sage España.

Debido a cambios de reestructuración de personal dentro de la empresa, le dieron prioridad a otros proyectos. Como consecuencia, el proyecto de automatización a gran escala se descarta.

Dirección de Calidad tiene en cuenta que *Logic Win Global*, por la que ya existe una serie de test automáticos, es una aplicación que en un periodo de tiempo no muy lejano, quedará obsoleta. Como consecuencia, se decide desarrollar un nuevo proyecto que incluya una colección de test automáticos para diversas funcionalidades de las que *Logic Class* dispone.

En vista que la gran mayoría de la infraestructura ya está montada y funciona correctamente, el coste para la realización de la automatización de *Class* es mínimo y, por tanto, un proyecto viable.

Debido a que el alumno no dispone de suficientes conocimientos en el funcionamiento de los test que *Calidad* desempeña, se puede decir que es una colaboración entre 2 departamentos: *Ingeniería y Calidad*. Gracias a la mutua colaboración ha sido posible realizar esta tarea.

2.2 Situación a resolver.

En la actualidad el departamento de *Calidad* hace las pruebas de regresión del *Cálculo de Nómina* y *Seguros Sociales* de forma manual. Este tipo de pruebas se considera que, por su excesiva complejidad, deben de automatizarse para intentar reducir al máximo el esfuerzo y los posibles errores que puedan surgir a la hora de hacer este tipo de pruebas.

Para llevar a cabo este procedimiento, *Calidad* lo hace en 2 fases: Ejecutar Cálculos en Class y posteriormente hacer comparativa entre bases de datos.

2.2.1 Ejecutar Cálculos Class.

Como ya se ha comentado anteriormente, se hacen los *cálculos de Nómina* y *Seguros Sociales* en la aplicación *Logic Class* de forma manual. Esto tiene una serie de inconvenientes:

- **Pérdida de tiempo**. Los datos a introducir son cuantiosos y hay que introducirlos de forma manual uno por uno, lo que hace que sea lento y pesado.
- **Error humano**. La cantidad de datos es muy elevada por lo que es fácil equivocarse a la hora de introducirlos.
- **Monotonía.** Son pruebas monótonas y repetitivas lo que hace que el propio cansancio también pueda provocar errores a la hora de introducir datos.

El resultado de los cálculos realizados por *Class*, se guardan en una base de datos que se le especifica a dicha aplicación a la hora de hacer la instalación.

Una vez realizados los cálculos, hay que comprobar que éstos hayan sido correctos. Esto también lleva cierto tiempo comprobarlo.

2.2.2 Comparativa entre Bases de Datos.

De una forma resumida, las pruebas de regresión consisten en cualquier tipo de pruebas de software que intentan descubrir las causas de nuevos errores que se producen en las aplicaciones, inducidos por cambios recientemente realizados en partes de la aplicación que, anteriormente al citado cambio, no eran propensas a este tipo de error.

Para hacer este tipo de comprobaciones se dispone de una base de datos de referencia. Una base de datos de referencia es aquella por la que ha pasado todo tipo de pruebas, por tanto, se considera la última versión de la aplicación que funciona correctamente.
En la siguiente ilustración podemos ver como sería una comparativa entre BD.

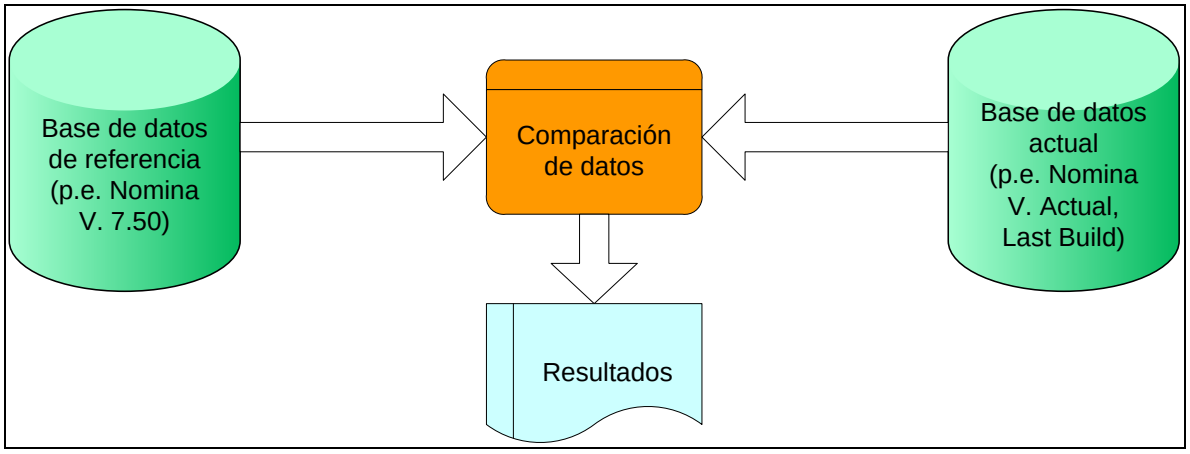

Figura 2. 1 Comparativa entre Base de Datos.

A la hora de hacer las comprobaciones entre Bases de datos, *Calidad* utiliza la BD en la cual se han realizado los cálculos manualmente con la BD de referencia. Para hacer la comparativa y ver los resultados utilizan un comparador de ficheros: *WinMerge*.

WinMerge es una herramienta para comparar el contenido de ficheros para Windows. La comparativa entre ficheros puede ser muy variada incluso entre bases de datos. Esta herramienta es capaz de contrastar un archivo con otro (mostrando los datos de ambos), remarcando con diferentes colores las modificaciones que se hayan hecho.

Si son unos archivos muy simples es fácil ver las diferencias, pero imaginemos una base de datos con unos 18.000 registros por ejemplo, buscar las diferencias ahí puede ser aparte de complicado, muy laborioso.

Una vez más pueden aparecer los errores comentados anteriormente como: error humano, cansancio, pérdida de tiempo… Todo esto hace que el proceso de hacer una prueba de regresión que se hace bastante a menudo, una tarea un tanto complicada y a la vez muy laboriosa.

2.3 Perfil del usuario del proyecto.

Los usuarios finales que utilizaran este tipo de test automáticos serán los *Técnicos de Calidad.*

Actualmente, no hay ningún proceso establecido de forma documentada ni procesos establecidos sobre la identificación y automatización de casos.

El siguiente gráfico define el flujo (simplificado) de información y acciones durante todas aquellas actividades de QA que tengan que ver con el proceso de automatización

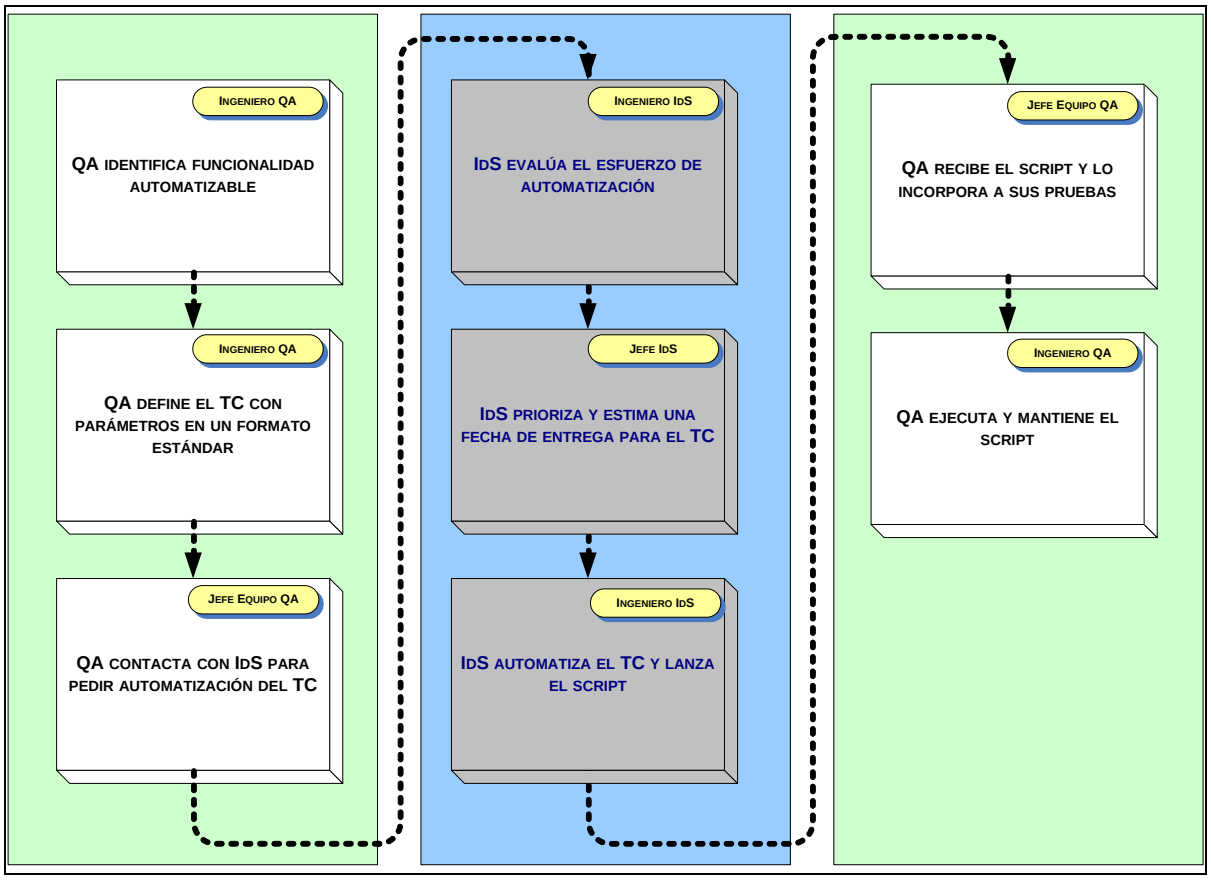

Figura 2.2: Flujo de datos automatización en QA

Nota: *IDS: Ingeniero de Software*

2.3.1 Identificación de casos de test

El proceso se inicia por lo tanto cuando en el área de QA se identifica un test case que, por características de distintos tipos: criticidad, alto nº de casuísticas parecidas, alto margen de error humano y alto grado de regularidad en ejecuciones (se explicará con más detalle en el capítulo 4). Esta identificación la harán los *Ingenieros de Calidad*.

2.3.2 Definición de los casos de test

Una vez identificado un TC, este se deberá pasar a un formato estándar definido en concordancia con el área de automatización del departamento de Ingeniería de Software. Los motivos de este proceso son los siguientes:

- Establecer unos criterios comunes a emplear.
- Poder definir bien los casos de prueba a automatizar y así evitar la necesidad de interacción continúa entre IdS y QA para resolver dudas.
- Documentar el proceso.

En la actualidad este formato aún se está ultimando.

2.3.3 Ejecución de los casos de test.

Una vez que el jefe informe al equipo de QA que el test automatizado ya puede ser ejecutado, su ejecución y mantenimiento recae en los ingenieros de QA. Sin embargo en caso de incidencias técnicas que impidan el correcto uso de un script, el ingeniero deberá contactar con una persona de contacto en IdS para solventar la incidencia; de esto también se dejará constancia en un listado gestionado centralmente y se informará a los jefes de equipo sobre los posibles impactos que esto supone.

2.3.4 Mantenimiento de test.

Una vez creado los test, hay que intentar actualizarlos. Se plantearán reuniones semanales entre los jefes de equipo de QA e IdS semanalmente para analizar las evaluaciones de los test cases propuestos para automatizar y por lo tanto evaluar la viabilidad económica y su utilidad.

Adicionalmente, se harán seguimientos mensuales entre los jefes de equipo de Calidad, el responsable de calidad y los responsables de automatización de IdS. Los puntos a tratar en estas reuniones serán los siguientes:

- Progreso real contra planificación
- Propuestas de mejora en los procesos
- Análisis de cobertura y proyecciones
- Análisis de incidencias
- Otros asuntos relevantes

2.4 Objetivos.

El objetivo del proyecto es crear una aplicación que a través de una plataforma permita la ejecución de tests automáticos de 2 casos de uso de *Logic Class*: *Cálculo de Nómina y Cálculo Seguros*. Las características que debe contemplar la aplicación deben de ser:

- Sistema único que soporte tests manuales y test automáticos.
- Facilitar la gestión de los casos de test y planes de prueba.
- Facilitar la gestión de los pases (o ejecuciones) de los tests.
- Ejecución de opciones básicas de pantallas de la aplicación emulando el comportamiento de un usuario.

Todo esto se pretende realizar para el *Técnico de Calidad* sin que éste tenga conocimientos de las herramientas utilizadas ni del lenguaje de programación utilizado para desarrollar esta aplicación. Éste se encargará de:

- Mantenimiento de los tests.
- Planificar la ejecución de los tests.
- Revisar los resultados de la ejecución de los tests.

Los beneficios expuestos a continuación son aplicables en la inmensa mayoría de casos, aunque según la situación su impacto pueda variar; cabe destacar que el impacto de automatizar test cases va más allá de un mero ahorro económico, ya que aporta dimensiones nuevas no cuantificables a la ejecución clásica manual:

- Ahorro de tiempo. En vez de probar una funcionalidad manualmente, se lanza el script (prácticamente inmediato) y sólo se debe comprobar el resultado del mismo (en caso de no fallar también prácticamente inmediato).
- Eliminación del riesgo por el factor humano.
- Posibilidad de ampliar el alcance de pruebas añadiendo un gran número de iteraciones que permita incrementar las casuísticas probadas.

Además de estos beneficios, el planteamiento especificado permite:

- Hacer un seguimiento detallado de las tareas, su estado y fechas
- Identificar oportunidades, puntos fuertes, debilidades y amenazas de forma prácticamente inmediata a través de un seguimiento programada establecido y formatos comunes consensuados
- A través de su adaptabilidad y flexibilidad, ser utilizado en una gran variedad de aplicativos.

2.5 DESCRIPCIÓN DEL SISTEMA.

2.5.1 Descripción.

La automatización en *Logic Class* pretende dotar al *Departamento de Calidad* de una herramienta para ejecutar pruebas de regresión automáticas sobre el *Cálculo de Nómina y Seguros Sociales.*

La herramienta de trabajo que se ha creado, tiene 3 partes bien diferenciadas:

- Programación pruebas en Ciclo de vida.
- Ejecución test con TestPartner
- Visualización resultados.

2.5.1.1 Ciclo de vida.

Para explicar la parte del proyecto de automatización de la colección de test para *Logic Class*, se explicará un poco por encima en que consiste y cuál es su propósito.

El Ciclo de vida es una aplicación web que se creó con los siguientes objetivos:

- Poder automatizar el proceso de compilaciones diarias de *Logic Class*.
- Poder realizar compilaciones parciales de componentes *Logic Class* (actualmente se realiza en su totalidad).
- Poder visualizar el resultado del proceso de compilaciones de *Logic Class.*

Su infraestructura es la siguiente:

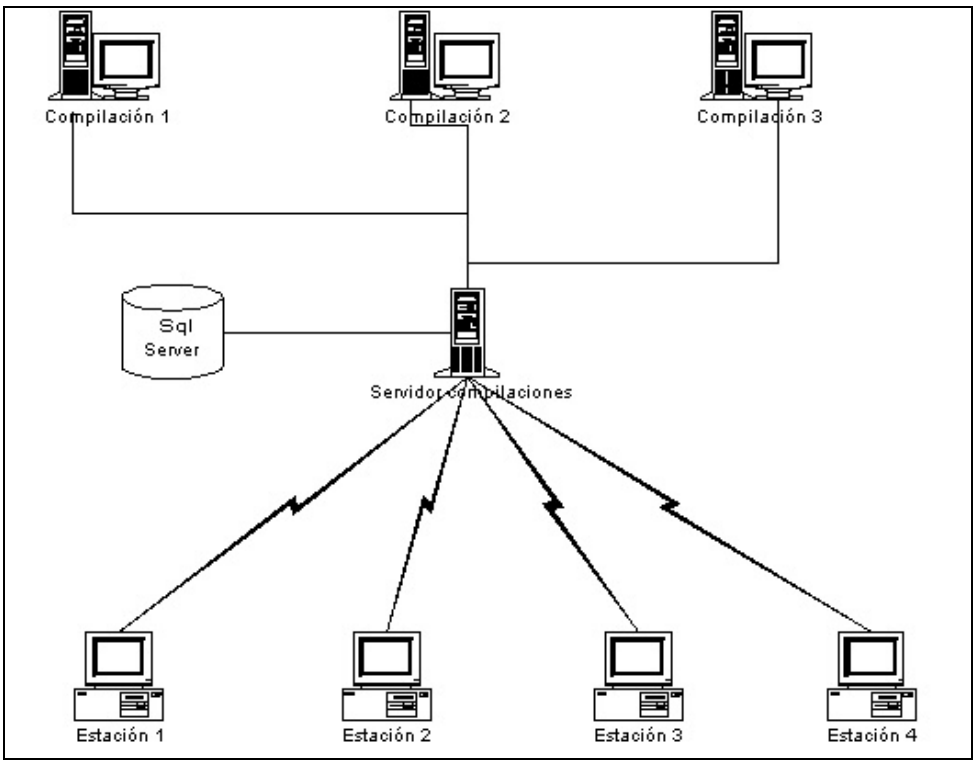

Figura 2.3: Infraestructura Ciclo de Vida

A continuación se explica con detalle en que consiste una de las partes que lo componen.

Compilación 1-n:
Equipos encargados de las compilaciones de *Logic Class*. Se necesitan tantos equipos como compilaciones activas se deseen.

- Compilación diaria.
- Compilación versión calle
- Compilación próxima versión
- Etc.

Servidor Compilaciones:
Equipo que controla todo el proceso de compilaciones, se encarga de lanzar las compilaciones en los equipos de Compilación, obtener los resultados y publicarlos

Estación 1-n:
Estaciones cliente desde donde se permite configurar el Ciclo de vida.

Una vez explicado las premisas por las que esta aplicación web fue creada, el *Departamento de Ingeniería* de Sage consideró que esta infraestructura muy útil, se le podían añadir nuevas funcionalidades para diferentes usos.

Todo esto hizo que en el año 2007, se aprovechara esta infraestructura para crear una colección de test automáticos para la aplicación *Logic Win Global*, con lo que la web dispone de nuevos recursos.

Con la automatización de Class ocurre lo mismo, se aprovecha este gran recurso del cual Sage dispone, para implementar una herramienta de trabajo que permita automatizar una serie de pruebas para que se ejecuten de manera automática.

Para empezar, hay que comentar que el acceso esta restringido, no es como una web cualquiera que te puedes registrar y tener acceso. Para poder acceder a ella, un usuario administrador te tiene que dar previamente de alta, por lo que tiene un acceso bastante limitado de usuarios.

Una vez hemos hecho login en la página web, nos aparecerá un menú en la parte izquierda de la pantalla. Las nuevas funcionalidades añadidas son las que aparecen en último lugar, dentro de la sección "*LC / Nómina*". Aquí podemos ver que hay diferentes opciones: *"Configuración, Casos de Test, Pruebas de Regresión y Programar Pruebas de regresión".*

Configuración ("ScriptConfiguration.aspx").
Aquí el usuario permite hacer es especificar la BD de referencia, así como sus tablas y sus campos. Aquí el usuario puede realizar 2 funciones:

- Añadir una nueva tabla
- Editar una tabla.

Estas operaciones se hacen en la página "ScriptableFicha.aspx" que consiste en rellenar los campos necesarios para conectarse a la BD de referencia.

En la siguiente ilustración se puede ver con más claridad:

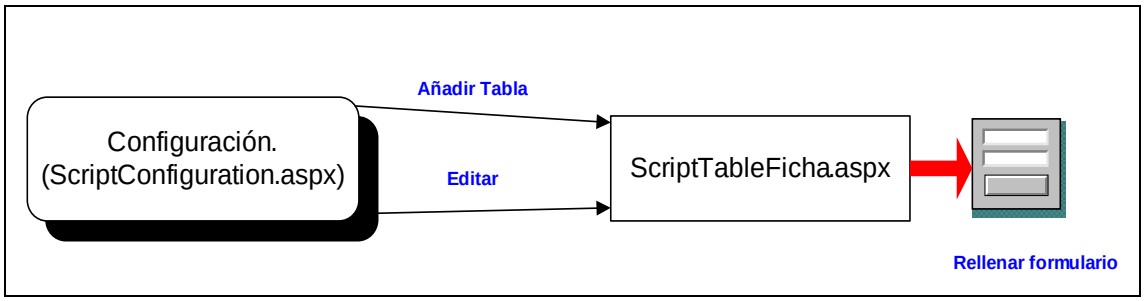

Figura 2.4: Enlaces "Configuración"

Casos de Test ("TestCaseLista.aspx").
Aquí es donde se definen los Test Cases. Se puede ver en una tabla el resumen de los TC de que se dispone. El usuario puede realizar 2 funciones:

- Añadir un nuevo TC.
- Editar un TC ya existente.

Dichas operaciones son enlazan con la siguiente página "TestCaseFicha.aspx". El usuario debe de poner un nombre al TC y una descripción. Posteriormente puede, como con anterioridad hacer las 2 siguientes funciones:

- Editar límites /opciones.
- Añadir limites/opciones.

Los límites son los cálculos que queremos que haga la aplicación Logic Class y las opciones las diferentes formas en que Class nos permite hacerlas. Para ello hay que rellenar un formulario que es un enlace a la página "TestCaseDetailsFicha.aspx". En la siguiente ilustración se puede observar como sería su funcionamiento.

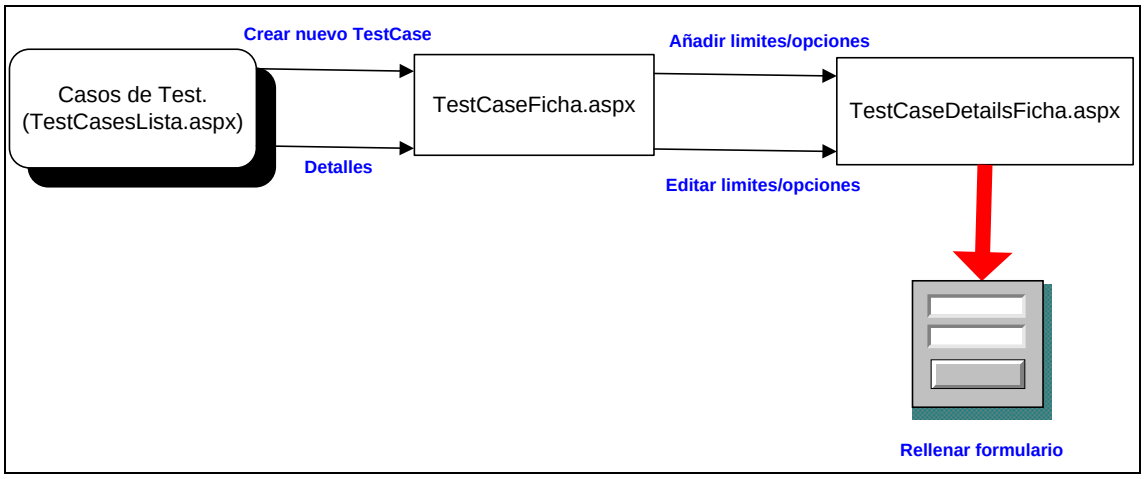

Figura 2.5: Enlaces "Casos de Test"

Pruebas de Regresión. ("JobsLista.aspx").

En este apartado es donde se pueden visualizar toda la colección de pruebas de regresión realizadas. Se puede ver con detalle todas las pruebas realizadas. Si queremos visualizar alguna en concreto, disponemos de un filtro para poder ver las que más convenga.

Disponemos de 2 enlaces dentro de los apartados "Nº ejecución" y "Nº incidencias". Al acceder a este enlace, accedemos a la pagina "JobTestCases.aspx".

Una vez en "JobTestCases.aspx", se puede ver un resumen del resultado del trabajo. Hay 3 partes bien diferenciadas.

- **Información General:** Donde se puede visualizar datos como el usuario que ejecutó el test, la máquina donde se ejecutó, el inicio y el fin de la ejecución.
- **Resultado:** Se visualiza una vez procesado el test datos como si se ejecutó correctamente, los registros procesados y el nº de incidencias encontradas.
- **Casos de test procesados:** Tabla donde se ve el contenido de los test que se han ejecutado.

En esta página se puede acceder a 3 páginas diferentes:

- **JobSchedule.aspx:** Para repetir las mismas pruebas directamente.
- **JobSummary.aspx:** Se puede ver un resumen de las incidencias encontradas.
- **JobDetailFicha.aspx:** Donde se muestra con detalle en que registro se ha producido alguna incidencia. Para ello se visualiza el valor de la tabla de referencia con el valor de la tabla donde se ha ejecutado el cálculo.

En la siguiente figura se puede visualizar como seria la estructura de estas páginas explicadas en este apartado.

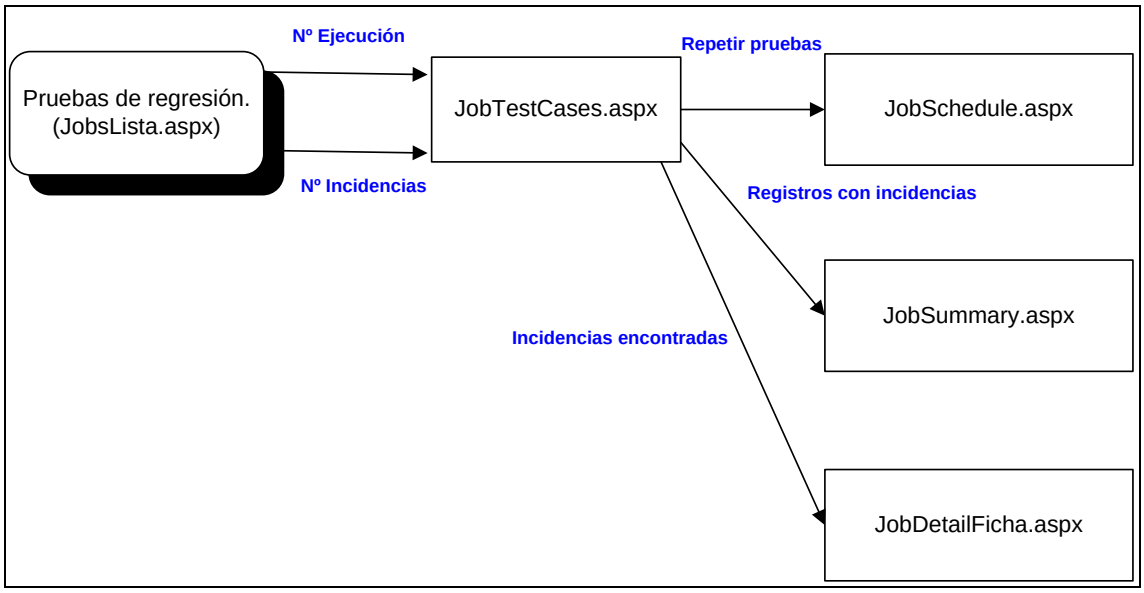

Figura 2.6: Enlaces "Prueba de Regresión"

Programar pruebas de regresión. ("JobSchedule.aspx").

Apartado en el cual se introducirán todos los parámetros para la ejecución de una *prueba de regresión*: selección de los TestCases, los procesos a ejecutar y la programación (fecha, hora, lugar, etc.).

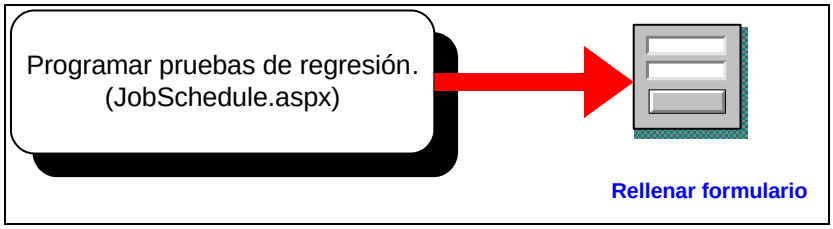

Figura 2.7: Programar prueba de regresión.

A continuación, mediante el siguiente diagrama de actividad, se puede observar como sería el proceso de la programación de una prueba de regresión en el *Ciclo de Vida.*

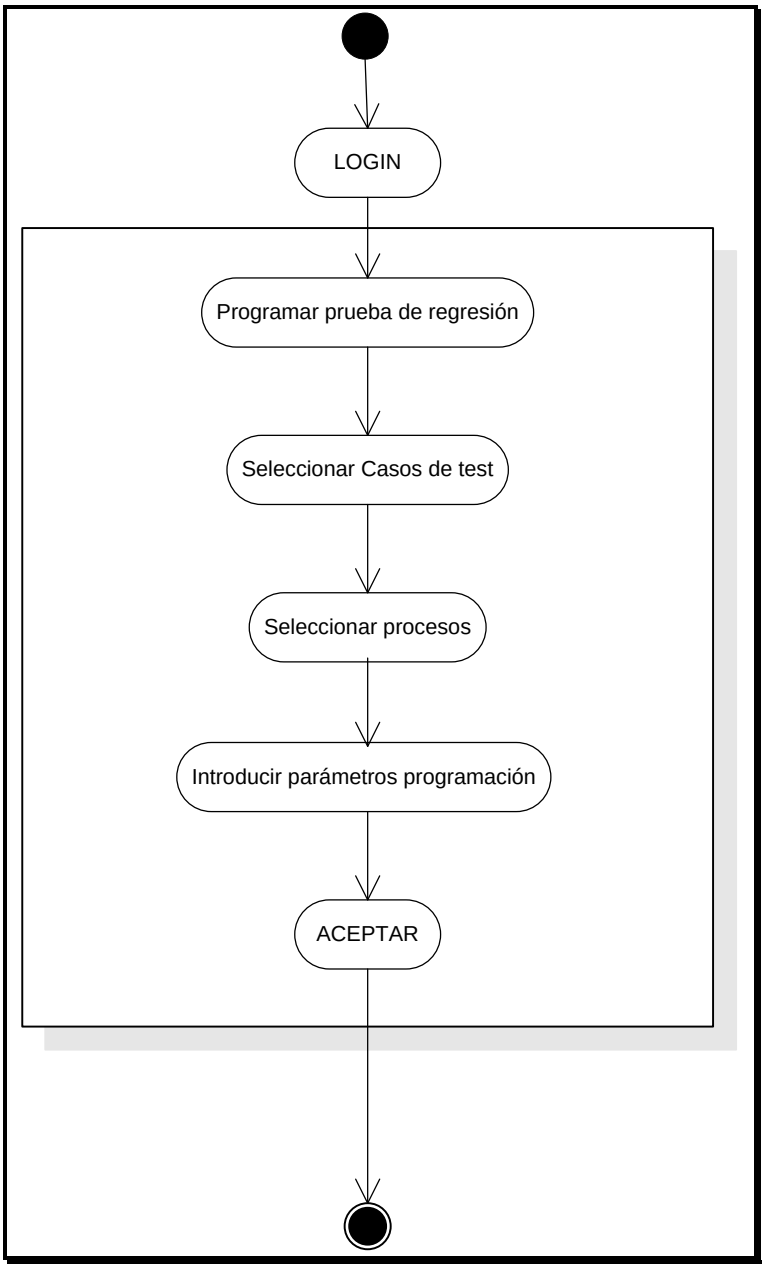

Figura 2.8: Diagrama actividad de planificación de test

2.5.1.2 Ejecución Test con TestPartner.

Una vez hecha la programación de una prueba en el Ciclo de Vida, el siguiente paso es la ejecución del test la máquina especificada con los parámetros correspondientes tales como fecha y hora establecidas.

En esta máquina se encuentra instalada una pequeña aplicación realizada en Visual Basic 6.0, llamada TPLauncher. Esta se encarga de lanzar el script de TestPartner una vez ha sido programado en el Ciclo de Vida. Para hacer eso lo que hace es leer de manera síncrona sobre la BD esperando que haya un nuevo trabajo programado.

Una vez ha sido creado un nuevo caso de Test a través del Ciclo de Vida, y comprobado que es la máquina donde nos encontramos, muestra una notificación y abre la aplicación TestPartner. Una vez abierto TestPartner comienza la ejecución del test.
En la siguiente ilustración se puede ver como funciona.

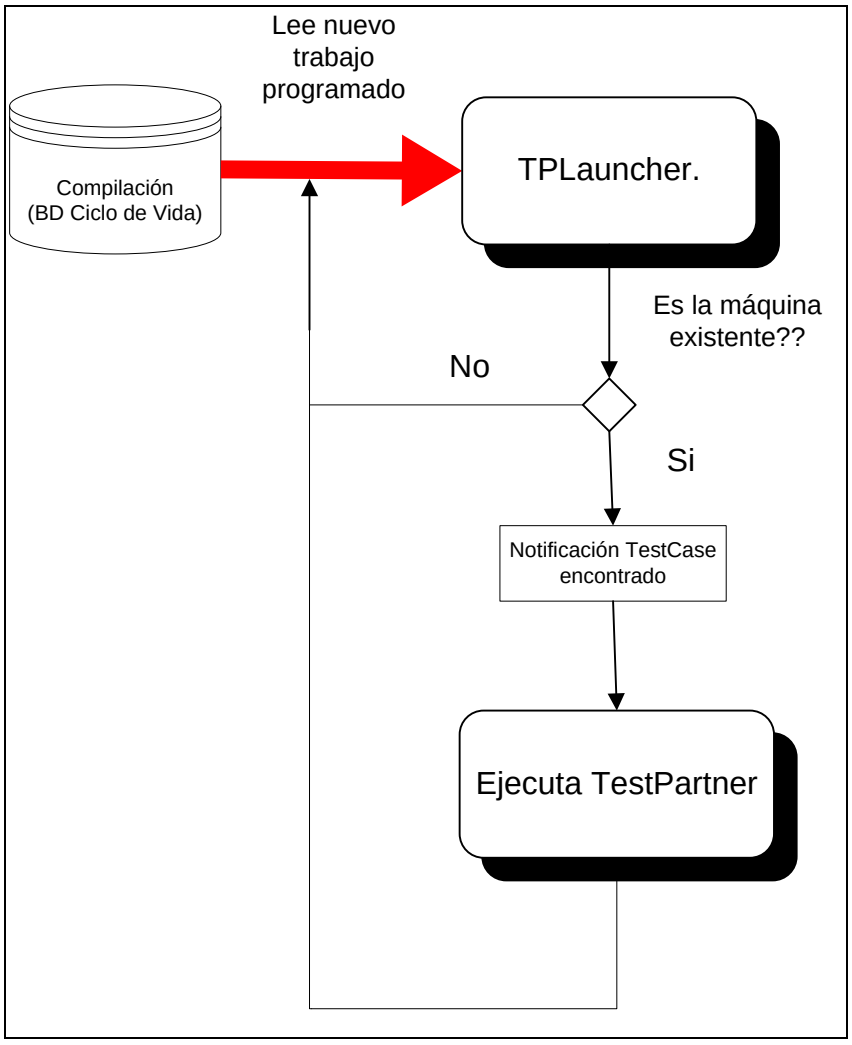

Figura 2.1: Funcionamiento TPLauncher

Una vez el TPLauncher ha podido encontrar el Test Case correspondiente, ejecuta el script de TestPartner.
En el siguiente diagrama se puede observar como funciona el script.

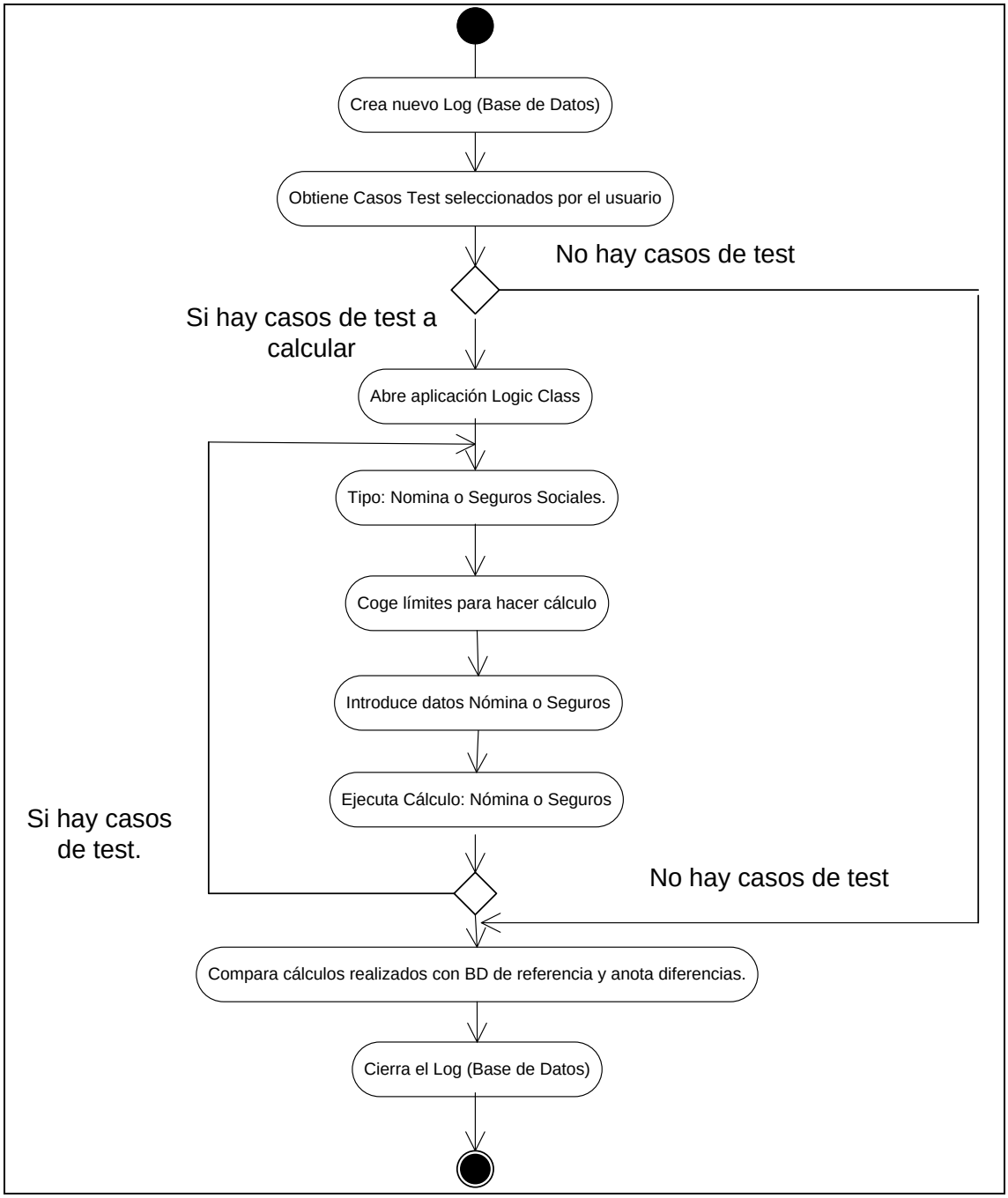

Figura 2.10: Funcionamiento script TestPartner

A continuación, se explica con detalle el funcionamiento de cada una de las partes vistas en la figura anterior.

1. ***Crea nuevo Log.*** Tal y como dice su definición, un log es un registro oficial de eventos durante un periodo de tiempo en particular. Por tanto lo que se hace es crear uno nuevo, ya que se crea uno diferente por cada ejecución.
2. ***Obtiene Casos de Test seleccionados por el usuario.*** En el *Ciclo de Vida*, el usuario seleccionó unos TC al programar la ejecución, por tanto lo que se hace es comprobar cuales fueron seleccionados. Esta parte es importante ya que si no hay TC, el script solo hará una comparativa entre las BD saltándose así los cálculos de *Nómina y Seguros.* Pasan directamente al paso 8
3. ***Abre aplicación Logic Class.*** Una vez comprobado que hay TC para ejecutar en Class, se procede a abrir ésta teniendo en cuenta que, para poder entrar, se necesita introducir un usuario y una contraseña. Por tanto lo que hace TestPartner es abrir la aplicación y hacer login.
4. ***Tipo: Nómina o Seguros.*** Aquí hay que tener en cuenta que, el usuario cuando programó realizar esta prueba, tenía 2 opciones: *Nómina y Seguros*. Se hace esta comprobación porque se encuentran en diferentes apartados de *Class* y, por tanto, habrá que acceder a una pantalla o a otra.
5. ***Escoge límites y opciones.*** Antes de realizar cualquiera de los 2 cálculos, hay que rellenar 2 formularios.
    - ***Opciones:*** Donde se le introducen parámetros como el año, el mes el tipo de proceso a ejecutar…etc.
    - ***Límites:*** Aquí se le introducen los límites. Estos son 2: empresas que se quieren calcular así como sus trabajadores. Se introduce mediante unos intervalos (limite inferior-superior).

    Esto se hace mediante lectura de la BD.
6. ***Introducir datos:*** Una vez se han podido obtener las opciones y los límites correspondientes, se introducen en sus respectivos formularios.
7. ***Ejecuta cálculo:*** Al finalizar de rellenar el formulario, se procede al cálculo según los datos introducidos con anterioridad. Hay que decir que si son muchos límites y opciones el proceso tarda un cierto tiempo. Una vez concluidas todas las operaciones, se comprueba si hay más TC, si es así, volvemos al paso 4. Si no es así, se pasa automáticamente al siguiente paso.
8. ***Compara datos con BD referencia y anota diferencias:*** Todo cálculo que realiza Class queda anotado en su BD. Por tanto, una vez realizados se puede proceder a comparar estos con la BD de referencia para ver las posibles diferencias. Al mismo tiempo que se va haciendo la comparativa, si se encuentra una diferencia se anota en el Log. Esta es la operación que más tiempo invierte ya que va registro a registro.
9. ***Cierre de Log:*** Una vez se han acabado todo tipo de ejecuciones, lo que se hace es cerrar cualquier enlace con BD para futuras ejecuciones posteriores.

2.5.1.3 Visualización resultados.

Una vez finalizada la ejecución del script de Testpartner en una máquina, todas las incidencias encontradas son escritas en el Log de la BD
Cualquier tipo de incidencia, podrá ser vista desde el *Ciclo de Vida*. La principal ventaja es que, al tratarse de un website, se podrán visualizar los resultados de cualquier ejecución independientemente de la máquina donde se haya ejecutado.

Para ver el funcionamiento de todas las partes explicadas en este apartado, se puede visualizar mediante la siguiente ilustración.

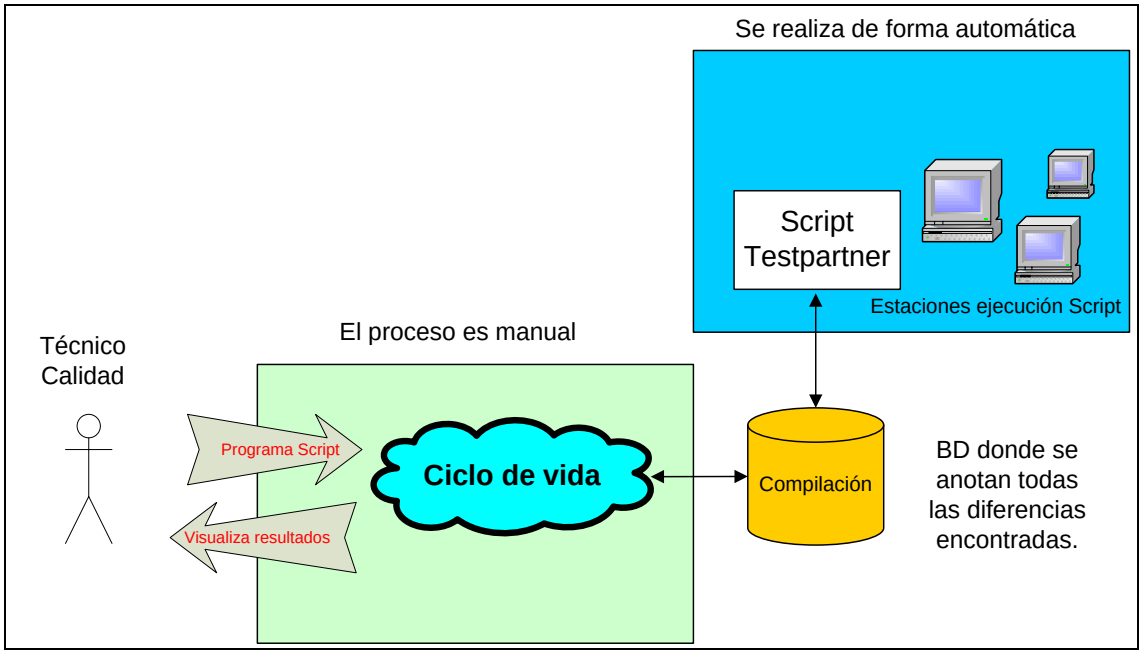

Figura 2.2: Funcionamiento plataforma para la creación de las pruebas de regresión

2.6 RECURSOS.

A continuación se explican los recursos necesarios para el desarrollo del proyecto tanto humano, como técnicos.

2.6.1 Recursos humanos.

Mediante este diagrama se puede observar todas las partes implicadas en este proyecto:

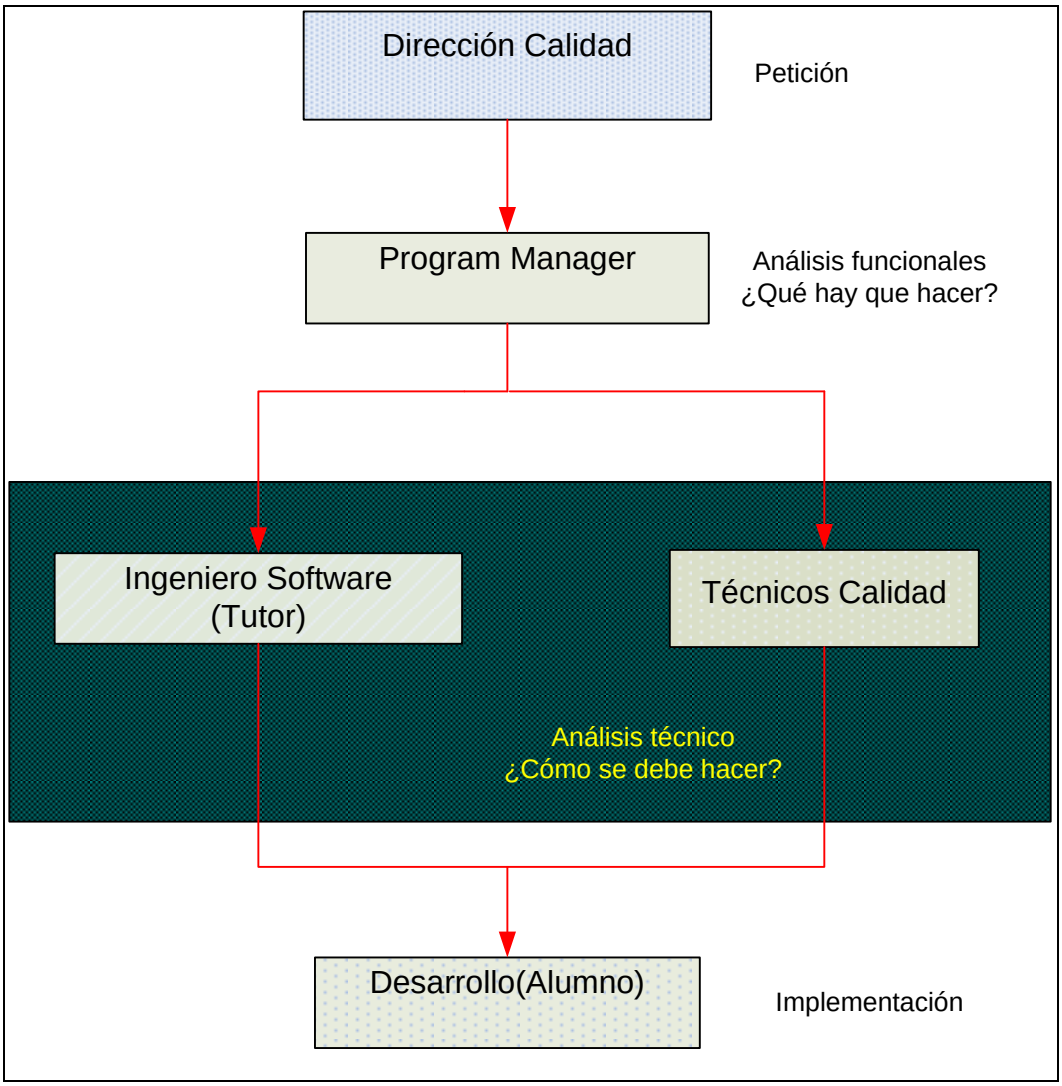

Figura 2.3: Personas implicadas en el proyecto

Como se puede ver en la figura anterior, se explica a continuación en que consiste cada una de las fases:
Dirección de Calidad es quien da el visto bueno para que se pueda llevar a cabo, una vez analizado las ventajas y sus inconvenientes.
Una vez se ha dado el visto bueno, el proyecto pasa a manos del *Program Manager* que es quien hace su análisis funcional, es decir, viendo las peticiones que se han hecho, decide qué se debe de hacer y qué no.
A continuación se hace el análisis técnico, es decir, una vez visto que es lo que se debe de hacer, en este caso el tutor, elabora un informe de cómo se debe de implementar en colaboración de los *Técnicos de Calidad* ya que ellos supervisan todo el trabajo que se deba de implementar.
Una vez hecho el informe, este pasa a manos del desarrollador, en este caso el alumno, que es quien hará todo el desarrollo en cuanto a programación. Cualquier duda que el alumno tenga, lo consulta entre su tutor y los *Técnicos de Calidad.*

2.6.2 Recursos hardware (Infraestructura).

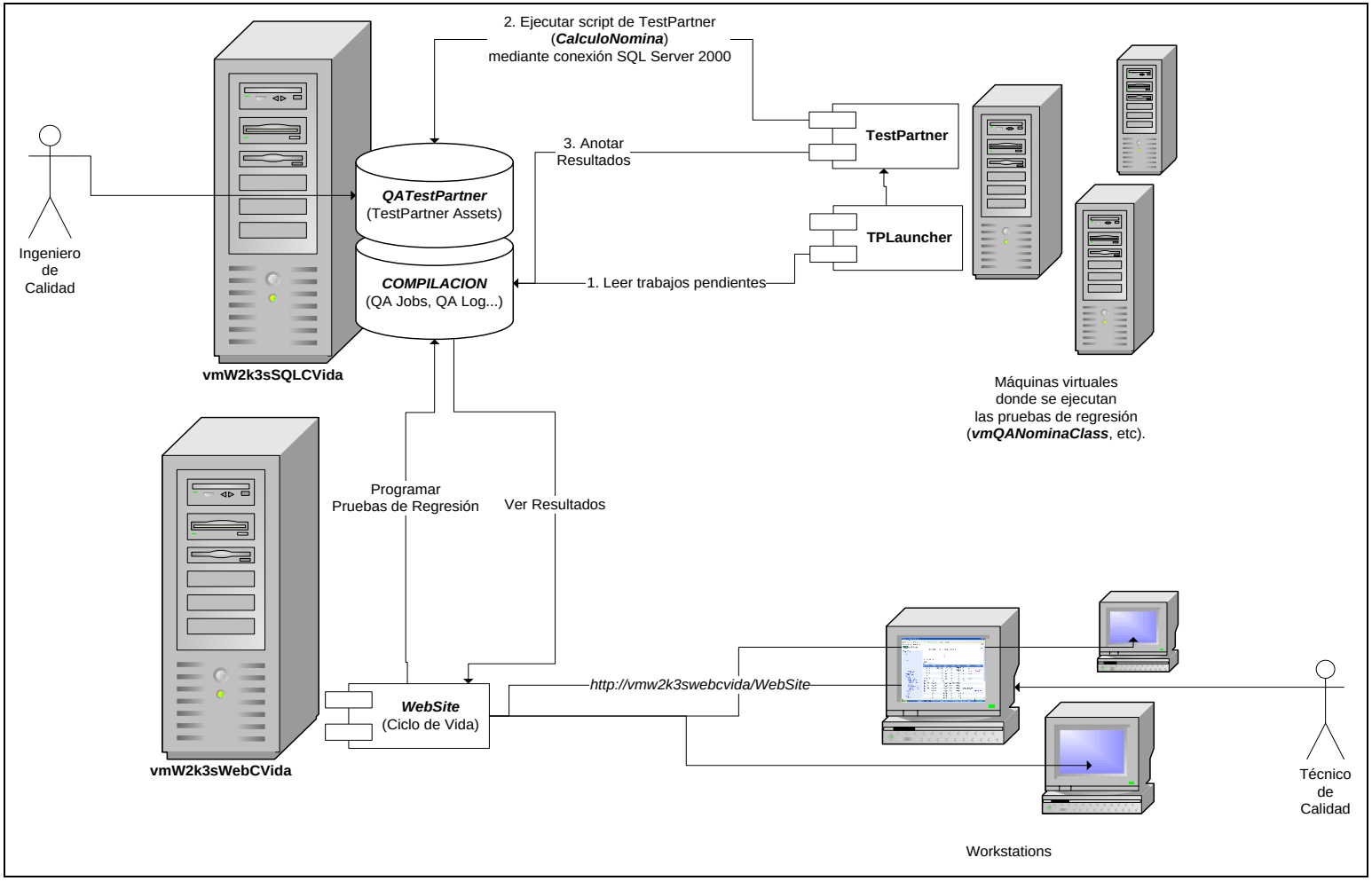

Figura 2.4: Infraestructura para la ejecución de las pruebas de regresión

Para explicar detalladamente la infraestructura utilizada, como se puede observar en la figura, hay 4 partes bien diferenciadas:

- ***vmW2k3sSQLCVida:*** Servidor donde se almacenan las Bases de Datos del *Ciclo de Vida*.
- ***vmW2k3sWebCVida:*** Servidor donde se encuentra físicamente el website del *Ciclo de Vida.*
- ***vmQANominaClass:*** Máquina donde se ejecutaran los test automáticos. De momento hay solo una pero se pueden implementar más.
- ***Workstations:*** Cualquier máquina física con acceso a Internet desde la que se pueda acceder al website y, por tanto, utilizar el *Ciclo de Vida.* Se puede utilizar tanto como para programar una nueva ejecución de test o para visualizar resultados de pruebas.

Es importante destacar que de las partes explicadas anteriormente, los nombres de las máquinas que contienen el prefijo "vm" quieren decir que son máquinas virtuales (virtual machines). Esto quiere decir que tanto las máquinas que hacen de servidores como las máquinas donde se ejecutan los test, se encuentran físicamente en un servidor llamado *"Virtualización4"*.

"Virtualización4", utiliza un software para poder crear estas máquinas virtuales: Virtual Server 2005 R2. Los requisitos mínimos que Virtual Server necesita son:

- ***Sistema operativo****:* Win 2003 Server o XP con SP2.
- ***Procesador:*** 2.1 GHz.
- ***Memoria RAM****:* 1 GB.
- ***Vídeo:*** 256 MB.
- ***Espacio en disco*** de 10 GB.
- ***Lector CD-Rom o DVD-Rom.***
- ***Teclado y Mouse.***

2.6.3 Recursos software.

Tal y como se ha comentado en el apartado anterior, para poder crear y ejecutar máquinas virtuales, se utiliza el software de Microsoft Virtual Server 2005 R2 Enterprise.

Virtual Server 2005 R2 es una solución para gestionar varios servidores (Sistemas Operativos) en una misma plataforma. Por tanto se utilizarán 2 Sistemas Operativos diferentes en la infraestructura de "*Virtualizacion4"*. Las máquinas servidoras utilizan Windows 2003 Server mientras que las máquinas cliente utilizan Windows XP SP2.

Por tanto, los requisitos Software para las máquinas servidoras son:

- ***vmW2k3sSQLCVida:*** Servidor donde quedan almacenadas las BD.
  - Microsoft Windows 2003 Server.
  - Microsoft SQL Server 2000.

- ***vmW2k3sWebCVida:*** Servidor donde está físicamente el Website.
  - Microsoft Windows 2003 Server.
  - Tener instalado el IIS (Internet Information Services). Este servicio convierte a un ordenador en un servidor de Internet o Intranet es decir que en las computadoras que tienen este servicio instalado se pueden publicar páginas web tanto local como remotamente (servidor web).

Para las máquinas clientes:

- ***vmQANominaClass:*** Máquina donde se ejecuta el test de forma automática. En un principio se ha creado esta máquina virtual, pero pueden ser varias.
  - Microsoft Windows XP.
  - TestPartner.
  - TPLauncher.
  - Logic Class.

- ***Workstations:*** Cualquier máquina física desde de la que se puede acceder al *Ciclo de Vida***.** Para estas máquinas hay 2 requisitos:
  - Pertenecer a la red de Sage ya que es un website exclusivo para los trabajadores.
  - Disponer de un navegador web. No hace falta tener conexión a Internet ya que al tener acceso al dominio *"logiccontrol.com"*, se puede acceder a toda la Intranet de la empresa.

2.7 PLANIFICACIÓN DEL PROYECTO.

Como todo proyecto informático la planificación es un tanto complicada de realizar. En el caso de este, no es ninguna excepción.

En un principio la tarea a desarrollar del alumno era la de realizar una parte de un proyecto superior. Concretamente, este debía solamente desarrollar la parte de TestPartner, es decir, desarrollar un script que fuera capaz de realizar el Cálculo de Nómina y Seguros Sociales automáticamente y a partir de ahí, hiciera la comparativa entre las Bases de Datos.

A principios del mes de Abril, hubo cambios en el organigrama de la empresa, que afectaron de lleno en la elaboración de este proyecto. Se le asignó un nuevo tutor al alumno e incluso un enfoque totalmente distinto al que en principio estaba previsto. Esto ha tenido consecuencias tales como una demora importante y que los plazos inicialmente previstos no se hayan podido cumplir.

Finalmente se decidió que, aprovechando los conocimientos que el alumno había adquirido durante los meses en los que estuvo trabajando, se hiciera la automatización de unas funcionalidades de *Class* que se consideraban críticas y, por tanto, imprescindibles.

La planificación inicial se puede ver mediante el siguiente diagrama de Gantt.

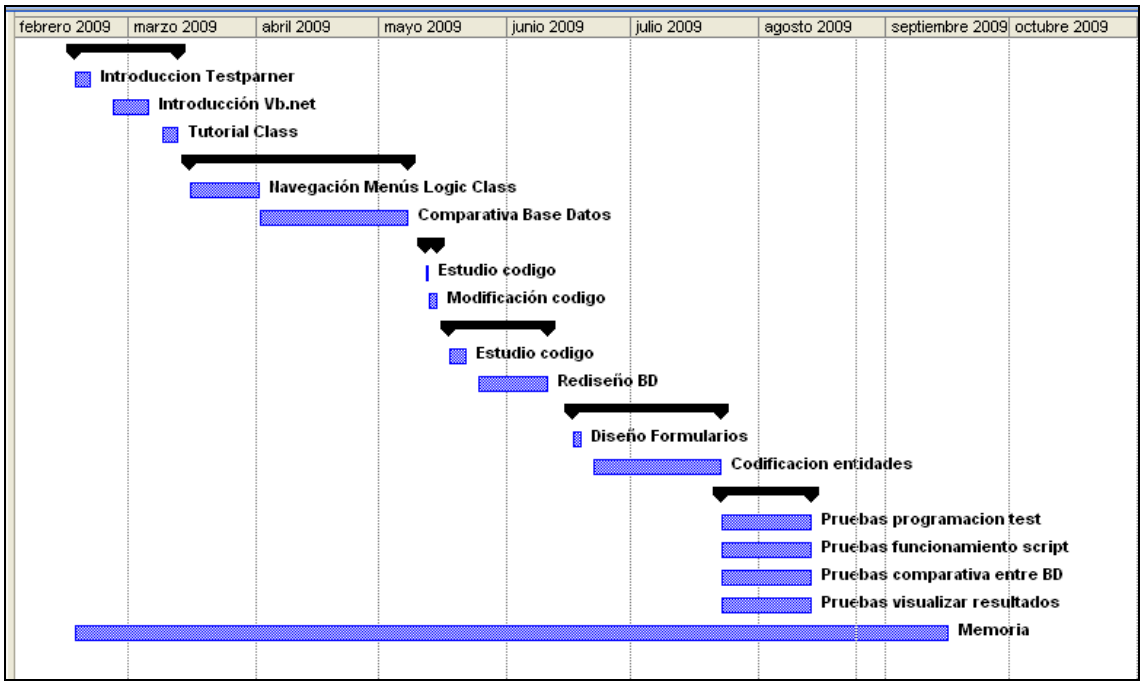

Figura 2.5: Diagrama de Gantt proyecto.

En la siguiente figura se puede ver con más detalle cada una de las etapas en las que se divide el proyecto.

	Nombre de tarea	Comienzo	Fin	Nombres de los recursos
1	**Estudio previo**	**lun 16/02/09**	**jue 12/03/09**	
2	Introducción a Testpartner	lun 16/02/09	jue 19/02/09	Introduccion Testparner
3	Introduccion a Vb.net	mié 25/02/09	jue 05/03/09	Introducción Vb.net
4	Tutorial Logic Class	lun 09/03/09	jue 12/03/09	Tutorial Class
5	**Codificación TestPartner**	**lun 16/03/09**	**jue 07/05/09**	
6	Ejecución automática Logic Class	lun 16/03/09	mié 01/04/09	Navegación Menús Logic Class
7	Comparativas Base de Datos	jue 02/04/09	jue 07/05/09	Comparativa Base Datos
8	**TPLauncher**	**mar 12/05/09**	**jue 14/05/09**	
9	Estudio código	mar 12/05/09	mar 12/05/09	Estudio codigo
10	Modificación código	mié 13/05/09	jue 14/05/09	Modificación codigo
11	**Estudio Ciclo de Vida**	**lun 18/05/09**	**mié 10/06/09**	
12	Estudio código	lun 18/05/09	jue 21/05/09	Estudio codigo
13	Rediseñar Base de Datos	lun 25/05/09	mié 10/06/09	Rediseño BD
14	**Codificación Ciclo de Vida**	**mié 17/06/09**	**mié 22/07/09**	
15	Diseño Formularios	mié 17/06/09	jue 18/06/09	Diseño Formularios
16	Codificación entidades	lun 22/06/09	mié 22/07/09	Codificacion entidades
17	**Pruebas**	**jue 23/07/09**	**jue 13/08/09**	**Pruebas**
18	Programación de test	jue 23/07/09	jue 13/08/09	Pruebas programacion test
19	Script TestPartner	jue 23/07/09	jue 13/08/09	Pruebas funcionamiento script
20	Comparativa Base de Datos	jue 23/07/09	jue 13/08/09	Pruebas comparativa entre BD
21	Visualización resultados en Website	jue 23/07/09	jue 13/08/09	Pruebas visualizar resultados
22	Memoria	lun 16/02/09	mar 15/09/09	Memoria

Figura 2.6: Planificación detallada del proyecto

Las etapas en las que se ha dividido el proyecto, son las siguientes:

- ***Estudio previo:*** El alumno no dispone de los conocimientos suficientes por lo que en esta fase lo que hace es aprender como funcionan las herramientas que utilizará para su futuro desarrollo.
- ***Codificación TestPartner:*** En esta etapa, el alumno desarrolla el script que hará posible la realización de las pruebas de regresión automáticas. Para ello hay 2 partes bien diferenciadas.
  - ***Ejecución automática Logic Class:*** Esta es la parte donde el script ejecuta automáticamente la aplicación.
  - ***Comparativa entre Base de Datos:*** Una vez ejecutado Logic Class, se procede a la comparación entre los resultados obtenidos y una BD de referencia.
- ***TPLauncher:*** Esta herramienta está ya implementada y funcionando para otra aplicación anterior, lo que se hace es adaptarla para que pueda funcionar con *Logic Class.*

- ***Estudio Ciclo de Vida:*** La aplicación *Ciclo de Vida* es muy compleja por lo que el alumno lo que hace aquí es entender el código y su funcionamiento ya que tiene que añadir nuevas funcionalidades. Además de entender su funcionamiento, utiliza una Base de Datos que debe de ser modificada para que estas nuevas funcionalidades puedan llevarse a cabo.
- ***Codificación Ciclo de Vida:*** Una vez el alumno ha podido comprender el funcionamiento de la aplicación, se procederá a modificar y añadir el código necesario para poder implementar las pruebas de regresión para *Logic Class.* Para ello, primeramente deberá de diseñar los nuevos formularios que hay que añadir y posteriormente su respectivo código para que puedan interactuar con el usuario.
- ***Pruebas:*** Se prueba su correcto funcionamiento de cada una de las partes y su puesta en marcha.
- ***Memoria:*** Es la parte donde se redacta todo lo relacionado con la documentación.

2.8 ANÁLISIS DE COSTES.

El hecho de aprovechar una infraestructura que ya esta montada y funcionando, hace que los costes a nivel de software / hardware sean inexistentes. La empresa dispone de un número de licencias necesarias para el montaje de las máquinas así como de sus aplicaciones, por lo que no necesita solicitar ninguna adicional.
El único coste que tiene este proyecto es la mano de obra por parte del alumno al que le reportan una compensación económica mensual.

Teniendo en cuenta que este proyecto se inició en febrero y su fecha de finalización es a finales de agosto y, la compensación económica es de unos 500 € mensuales, el coste asciende a:

2.8.1 Costes Hardware.

Teniendo en cuenta que las máquinas donde van a ser ejecutadas los test son máquinas virtuales, el único coste sería la del software que se utiliza para su montaje. En este caso es Virtual Server 2005 R2 del que hay una versión gratuita por lo tanto el coste HW no se tiene en cuenta.

2.8.2 Costes Software

Las máquinas virtuales deben de disponer de:

Sistema operativo Windows XP Home SP3	→	80 €
TestPartner	→	6500 €
Total	→	6580 €.

Es un coste estimativo ya que en realidad forma parte de una infraestructura que esta ya implementada y funcionando por lo que SAGE no tiene que hacer ninguna inversión, ya dispone de las licencias necesarias.

2.8.3 Costes Personal.

A continuación se muestra cual sería el coste estimativo del proyecto.

Program manager (análisis funcional)	→	43 horas * 18,75 €/h = 806,25 €
Ingeniero Software (análisis orgánico)	→	83 horas * 11,25 €/h = 933,75 €
Desarrollo (Codificación)	→	415 horas * 9,37 €/h = 3888,55 €
Tester (QA)	→	95 horas * 10,62 €/h = 1008,09 €
Total	→	6636,64 €.

2.9 CONCLUSIONES.

Lo que se ha conseguido es el desarrollo de una herramienta de trabajo para los *Técnicos de Calidad* que facilite mucho su tarea a la hora de encontrar errores en el *Cálculo de Nómina* y *Seguros Sociales* de *Logic Class*.

Hay que tener en cuenta que solo se han implementado estas 2 funcionalidades al considerarse "criticas", es decir que cumple todos los requisitos para su automatización. Esto quiere decir que no tiene porqué automatizarse todo tipo de pruebas, primero hay que hacer un análisis para ver si el tiempo invertido en la realización de estos test es rentable o no.

Esta automatización aporta los siguientes beneficios:

- ***Mayor fiabilidad*** en la certificación de los productos, reduciendo el factor humano.
- ***Mayor cobertura***, ya que podremos probar muchas más situaciones en menos tiempo.
- ***Detección temprana de errores de programación***.
- ***Automatización de las tareas manuales más tediosas***, podrán dedicar un mayor porcentaje de su tiempo a interpretar los resultados obtenidos en las pruebas de regresión, en vez de dedicarlo a la ejecución de las pruebas en sí.

Capítulo 3.

Fundamentos teóricos.

En este capítulo se expondrán una serie de conocimientos teóricos para entender mejor en qué consisten las pruebas que se van a automatizar. El capítulo se divide en las siguientes partes:

- ***ERP, Logic Class.*** Pequeña exposición de los contenidos de esta aplicación.
- ***Ciclo de vida software.*** Las etapas para el desarrollo del software.
- ***Pruebas de software.*** Conceptos teóricos sobre los tipos de pruebas de software que se realizan
- ***Pruebas de regresión***. Se explica en qué consisten así como las que se realizan en Sage Logic Control.

3.1 INTRODUCCIÓN.

Para explicar en que consiste la aplicación *Logic Class* se hará una pequeña introducción en que consiste una aplicación ERP.

3.2 ERP

Los sistemas de planificación de recursos de la empresa (en inglés ERP, enterprise resource planning) son sistemas de gestión de información que integran y automatizan muchas de las prácticas de negocio asociadas con los aspectos operativos o productivos de una empresa.

Existen 3 características que definen un ERP.

- Son sistemas integrales.
- Son sistemas modulares.
- Son adaptables.

Los objetivos principales de los sistemas ERP son:

- Optimización de los procesos empresariales.
- Acceso a la información confiable, precisa y oportuna.
- Posibilidad de compartir información entre todos los componentes de la organización.
- Eliminación de datos y operaciones innecesarias.
- Reducción de tiempos y de los costes de los procesos.

El propósito fundamental de un ERP es otorgar apoyo a los clientes del negocio, tiempos rápidos de respuesta a sus problemas así como un eficiente manejo de información que permita la toma oportuna de decisiones y disminución de los costes totales de operación.

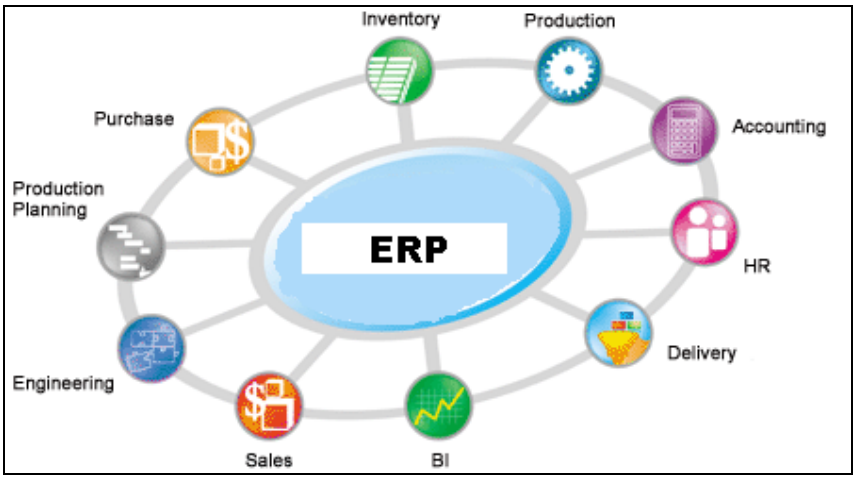

Figura 3.1: Ejemplo ERP

3.3 LOGIC CLASS.

La automatización de las *pruebas de regresión* se hace sobre 2 funcionalidades de la aplicación Logic Class, por tanto a continuación se explicará en que consiste.

Logic Class es una aplicación ERP que se compone de módulos específicos para llevar el control de un negocio, la aplicación abarca:

- Área contable y financiera.
- Ventas y distribución.
- Producción y logística.
- Proyectos.
- CRM.
- Postventa.
- Nómina y Recursos Humanos.
- Análisis de Negocio.
- Gestión Documental.
- Contenidos Legales.

Nómina y Recursos Humanos, el cual se van a hacer las pruebas, está dentro del apartado Laboral de la aplicación.

La solución de Gestión Laboral *de Logic Class* está dividida en dos grandes áreas:

- El área de Nómina permite realizar todo tipo de cálculos salariales, un completo tratamiento multiconvenio, gestión de atrasos y otras funcionalidades avanzadas. Incluye además una completa analítica de costes salariales.

- El área de Recursos Humanos cubre todo el ciclo de vida del empleado, desde la definición del puesto de trabajo y confección de organigramas, hasta la selección, formación, evaluación y retribución de los empleados y la gestión de riesgos laborales. Consta además de un amplio catálogo de informes de seguimiento y gestión.

La gestión Laboral Logic Class dispone de las siguientes funciones:

- Único entorno de trabajo que integra todas las tareas del área laboral.
- Tutoriales en línea y paneles de acceso a los datos y procesos más comunes
- Código único de empleado, con histórico de los datos de los diferentes periodos de alta y/o actividad
- Elaboración de circulares y etiquetas de envío
- Envío automático, por correo electrónico, de informes y listados (incluyendo recibos salariales a empleados)
- Mediante el centro de información del empleado, se puede visualizar toda la información relacionada con el mismo. Permite centralizar y gestionar todas sus relaciones laborales: contrastos, categorías, puestos de trabajo, nóminas, formación, evaluación, desempeño, etc.
- Cumplimiento de la normativa LOPD. Con estricto control de acceso de los usuarios a la información y los datos.
- Multi-empresa, multi.ejercicio y multi-convenio con diferentes periodos de vigencia
- Trazabilidad e integración total de la información
- Gestión de avisos y alarmas relacionados con los principales hitos en la gestión laboral.

Esta aplicación tiene una versión comercial estándar pero también es posible adaptarla según las necesidades del negocio personalizado, permitiendo la personalización de la base de datos, diseño de pantallas e informes específicos, cálculos, etc. Además de todo esto se suelen implementar actualizaciones para un mejor funcionamiento y corrección de posibles errores que pueda tener.

3.4 CICLO DE VIDA DEL SOFTWARE.

Se entiende el ciclo de vida como el proceso que se sigue para construir, entregar y hacer evolucionar el software, desde la concepción de una idea hasta la entrega y el retiro del sistema.
Regularmente hay cinco pasos en el ciclo de vida del software:

***a) Entender el problema*.** En la primera fase del ciclo de vida del software, se enlistan las tareas que el software debe desarrollar, los problemas a ser resueltos, y en esta fase se estudian sus causas y efectos.

La tarea específica que se requiere del programa se deriva del establecimiento del problema, que es una descripción concisa del problema en cuestión. El software debe proporcionar una solución utilizable a este problema.

***b) Diseñar el programa*.** En la fase de diseño, el objetivo es conocer las relaciones entre los módulos del programa, y garantizar que se cumplen cabalmente los requerimientos solicitados de una manera eficiente, lógica y completa.

Los diseñadores de software consideran los recursos de hardware y software disponibles para poder alcanzar su objetivo. Si se llega a la conclusión de que no es posible utilizar algún hardware o software, se planea utilizar una estrategia diferente.

Primero se diseña la estructura general del programa. Entonces el problema se divide en subproblemas en tareas más y más pequeñas hasta que tengan un tamaño manejable.

c) Codificar el programa. Durante la fase de codificación, el programa se escribe en un lenguaje de programación. Hay muchos lenguajes de programación, cada uno de ellos es especialista en algún tipo de problemas. Por ejemplo, FORTRAN es especialista en cálculos numéricos, mientras que LISP es especialista en problemas de inteligencia artificial y procesamiento simbólico. El código del programa debe desarrollar la tarea solicitada, y debe ser legible de modo que otros programadores lo puedan mantener. Los programas se escriben usualmente en módulos separados, cada módulo desarrolla alguna tarea específica y debe funcionar independientemente y en relación con el resto del programa.

d) Probar el programa. Durante la fase de pruebas, el programa se ejecuta y se revisa. Las tareas deben ejecutarse sin errores en los resultados y también sin errores fatales. Los defectos en los programas se llaman bugs.

Se examinan primero los módulos de manera individual, en forma independientemente, luego, se prueba todo el programa para encontrar bugs que puedan ocurrir en la interacción de los módulos. Cuando se encuentra un bug, se aísla la causa y se resuelve. Este proceso se llama depuración. El programador se debe asegurar de al resolver un bug, no se crean otros más en alguna otra parte del programa.

e) Mantener el programa. Durante la fase de mantenimiento, de determina cualquier error y deficiencia en el programa, y se realizan cualquier acción para resolverla, mientras se preserva la integridad del programa. El uso de notas de diseño, códigos bien documentados y variables entendibles, puede ayudar al mantenimiento futuro del programa.

Este proyecto se basa en las etapas de prueba y de mantenimiento del programa que es para lo que se pretende desarrollar una colección de test automatizados.

3.5 PRUEBAS DE SOFTWARE.

Las pruebas de software son los procesos que permiten verificar y revelar la calidad de un producto software.

Las pruebas de software se integran dentro de las diferentes fases del Ciclo del software dentro de la Ingeniería de software. Así se ejecuta un programa y mediante técnicas experimentales se trata de descubrir que errores tiene.

Para determinar el nivel de calidad se deben efectuar unas medidas o pruebas que permitan comprobar el grado de cumplimiento respecto de las especificaciones iniciales del sistema.

Las pruebas de software, testing o beta testing es un proceso usado para identificar posibles fallos de implementación, calidad, o usabilidad de un programa de ordenador o videojuego. Básicamente es una fase en el desarrollo de software consistente en probar las aplicaciones construidas.

Hay muchos planteamientos a la hora de abordar el proceso de pruebas de software, pero para verificar productos complejos de forma efectiva requiere de un proceso de investigación más que seguir un procedimiento al pie de la letra. Una definición de "testing" es: “proceso de evaluación de un producto desde un punto de vista crítico”, donde el "tester" (persona que realiza las pruebas) somete el producto a una serie de acciones inquisitivas, y el producto responde con su comportamiento como reacción.

Por supuesto, nunca se debe testear el software en un entorno de producción. Es necesario testear los nuevos programas en un entorno de pruebas separado físicamente del de producción. Para crear un entorno de pruebas en una máquina independiente de la máquina de producción es necesario crear las mismas condiciones que en la máquina de producción. Existen a tal efecto varias herramientas vendidas por los mismos fabricantes de hardware (IBM, Sun, HP etc.). Esas utilidades reproducen automáticamente las bases de datos para simular un entorno de producción.

En general, se distinguen entre errores de programación (o "bugs") y defectos de forma. En un defecto de forma, el programa no realiza lo que el usuario espera. Por el contrario, un error de programación puede describirse como un fallo en la semántica de un programa de ordenador. Éste podría presentarse, o no, como un defecto de forma si se llegan a dar ciertas condiciones de cálculo.

Una práctica común es que el proceso de pruebas de un programa sea realizado por un grupo independiente de "testers" al finalizar su desarrollo y antes de sacarlo al mercado. Una práctica que viene siendo muy popular es distribuir de forma gratuita una versión no final del producto para que sean los propios consumidores los que la prueben. En ambos casos, a la versión del producto en pruebas y que es anterior a la versión final (o "master") se denomina beta, y a dicha fase de pruebas, beta testing.

Puede además existir una versión anterior en el proceso de desarrollo llamada *alpha*, en la que el programa, aunque incompleto, dispone de funcionalidad básica y puede ser testeado.

Finalmente y antes de salir al mercado, es cada vez más habitual que se realice una fase de RTM testing (Release To Market), dónde se comprueba cada funcionalidad del programa completo en entornos de producción.

En la cadena de valor del desarrollo de un software específico, el proceso de prueba es clave a la hora de detectar errores o fallas. Conceptos como estabilidad, escalabilidad, eficiencia y seguridad se relacionan a la calidad de un producto bien desarrollado. Las aplicaciones de software han crecido en complejidad y tamaño, y por consiguiente también en costos. Hoy en día es crucial verificar y evaluar la calidad de lo construido de modo de minimizar el costo de su reparación. Mientras antes se detecte una falla, más barato es su corrección.

El proceso de prueba es un proceso técnico especializado de investigación que requiere de profesionales altamente capacitados en lenguajes de desarrollo, métodos y técnicas de pruebas y herramientas especializadas. El conocimiento que debe manejar un ingeniero de prueba es muchas veces superior al del desarrollador de software.

3.5.1 Tipos de pruebas Software.

Existen diferentes tipos de pruebas para probar el correcto funcionamiento del software:

a) Pruebas unitarias: es una forma de probar el correcto funcionamiento de un módulo de código. Esto sirve para asegurar que cada uno de los módulos funcione correctamente por separado.

b) Pruebas funcionales: es una prueba basada en la ejecución, revisión y retroalimentación de las funcionalidades previamente diseñadas para el software. Las pruebas funcionales se hacen mediante el diseño de modelos de prueba que buscan evaluar cada una de las opciones con las que cuenta el paquete informático.

c) Pruebas de integración: son aquellas que se realizan en el ámbito del desarrollo de software una vez que se han aprobado las pruebas unitarias. Únicamente se refieren a la prueba o pruebas de todos los elementos unitarios que componen un proceso, hecha en conjunto, de una sola vez.

Consiste en realizar pruebas para verificar que un gran conjunto de partes de software funcionan juntos.

d) Pruebas de validación: Se trata de evaluar el sistema o parte de este durante o al final del desarrollo para determinar si satisface los requisitos iniciales. La pregunta a realizarse es: ¿Es esto lo que el cliente quiere?

e) Pruebas de regresión**:** Cualquier tipo de pruebas de software que intentan descubrir las causas de nuevos errores (bugs), carencias de funcionalidad, o divergencias funcionales con respecto al comportamiento esperado del software, inducidos por cambios recientemente realizados en partes de la aplicación que anteriormente al citado cambio no eran propensas a este tipo de error. Esto implica que el error tratado se reproduce como consecuencia inesperada del citado cambio en el programa.

Este tipo de cambio puede ser debido a prácticas no adecuadas de control de versiones, falta de consideración acerca del ámbito o contexto de producción final y extensibilidad del error que fue corregido (fragilidad de la corrección), o simplemente una consecuencia del rediseño de la aplicación.

Por lo tanto, en la mayoría de las situaciones del desarrollo de software se considera una buena práctica que cuando se localiza y corrige un bug, se grabe una prueba que exponga el bug y se vuelvan a probar regularmente después de los cambios subsiguientes que experimente el programa.

3.5.2 Pruebas de regresión en Sage Logic Control.

Sage Logic Control considera que se tienen que automatizar las pruebas de regresión sobre el *Cálculo de Nómina* y *Seguros Sociales* por que son las más utilizadas y a la vez más delicadas. Los motivos son los siguientes:

- ***Criticidad:*** Al igual que a la hora de identificar qué test cases ejecutar con prioridad en función de su factor de riesgo, tiene más prioridad un test case de criticidad alta sobre otros.
- ***Alto nº de casuísticas parecidas:*** El uso de parámetros y Test Cases atómicos permite optimizar los scripts de TC automáticos para ampliar las casuísticas cubiertas con un incremento de esfuerzo muy limitado en el caso de TC automatizados. Si un TC automático puede ser usado para cubrir una amplia gama de escenarios, tiene más valor que uno equivalente que sólo cubra pocas casuísticas.

- ***Alto margen de error humano:*** Casos que por su naturaleza – monótonos, gran cantidad de datos, cálculos complejos, etc. – sean proclives a provocar un error humano en su ejecución.

- ***Alto grado de regularidad en ejecuciones:*** Cuántas más veces se ejecute un TC, antes se recuperará la inversión del esfuerzo de automatización.

La responsabilidad de identificar test cases considerados útiles para ser automatizados recae en los ingenieros de QA.

Una vez identificado un TC, este se deberá pasar a un formato estándar definido en concordancia con el área de automatización del departamento de Ingeniería de Software. Los motivos de este proceso son los siguientes:

- Establecer unos criterios comunes a emplear
- Poder definir bien los casos de prueba a automatizar y así evitar la necesidad de interacción continua entre Ingenieros de Software y QA para resolver dudas
- Documentar el proceso.

Los beneficios expuestos a continuación son aplicables en la inmensa mayoría de casos, aunque según la situación su impacto pueda variar; cabe destacar que el impacto de automatizar test cases va más allá de un mero ahorro económico, ya que aporta dimensiones nuevas no cuantificables a la ejecución clásica manual

- Ahorro de tiempo. En vez de probar una funcionalidad manualmente, se lanza el script (prácticamente inmediato) y sólo se debe comprobar el resultado del mismo (en caso de no fallar también prácticamente inmediato).
- Eliminación del riesgo por el factor humano.
- Posibilidad de ampliar el alcance de pruebas añadiendo un gran número de iteraciones que permita incrementar las casuísticas probadas.

Capítulo 4.

Análisis y Especificación.

En el siguiente capítulo se mostrará el análisis técnico del proyecto, requisito previo antes de poder empezar a desarrollar código. Los apartados que lo componen son:

- ***Casos de uso*.** Cuales son los actores que intervienen en el sistema así como la función que desempeñan.
- ***Diagrama de secuencia*.** Se explica el funcionamiento de la descripción de los casos de uso.

4.1 DIAGRAMA DE CASOS DE USO.

A continuación se muestran los principales actores y casos de uso que intervienen en el sistema:

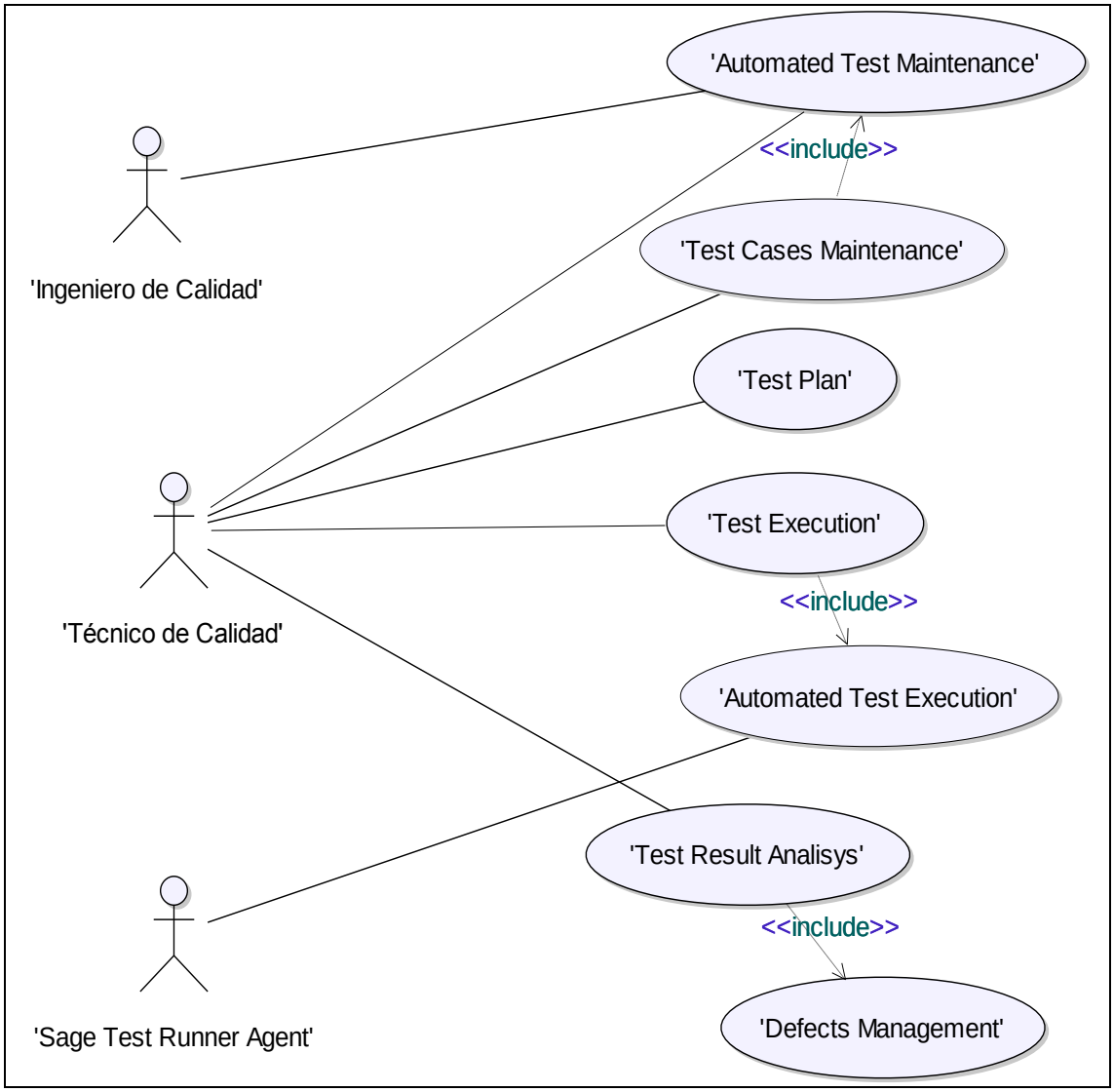

Figura 4. 1 Diagrama de casos de uso

4.1.1 Actores.

Los actores que intervienen en los procesos de QA que pretende cubrir el sistema son:

- **Ingeniero de Calidad**: Es el encargado del mantenimiento de la infraestructura y también de los scripts de tests automáticos complejos.
- **Técnico de Calidad**: Es el encargado del mantenimiento de los casos de test, los planes de ejecución y, de comprobar y analizar los resultados de los tests automáticos.
- **Sage Test Runner Agent**: Se trata de la aplicación que lanza y ejecuta los tests automáticos.

4.1.2 Descripción de los Casos de Uso.

A continuación se detallan cada uno de los principales casos de uso que contempla el sistema:

- **Mantenimiento de Casos de Test**: Cada caso de test puede llevar asociada *automatización* o no. Esta automatización permitirá que el test se ejecute de forma totalmente desatendida reportando el resultado de las validaciones que contenga. Dependiendo de la complejidad de dicha *automatización* las labores de mantenimiento las deberá realizar un *Técnico de Calidad* o un *Ingeniero de Calidad.*
- **Planificación de ejecución de tests**: El *técnico de calidad* debe planificar quién, dónde y cuándo se debe ejecutar cada caso de test.
- **Ejecución test automático**: Cuando un caso de test lleva asociada *automatización* éste será ejecutado automáticamente por un *agente de ejecución de pruebas automáticas*. Esta automatización normalmente contendrá una o más validaciones, el resultado de las cuales deberá registrarse del mismo modo que lo haría un técnico de calidad.
- **Análisis resultados de la ejecución de tests**: El *técnico de calidad* es el encargado de analizar el resultado de la ejecución de los casos de test automáticos. Cuando haya validaciones que no han dado el resultado esperado, se creará un defecto en el sistema de gestión de defectos.

4.2 DIAGRAMAS DE SECUENCIA.

A continuación se explican los diagramas de secuencia de la descripción de los casos de uso anteriores comentados.

Mantenimiento de los Casos de Test:

Secuencia donde se hace el mantenimiento de los Casos de Test.

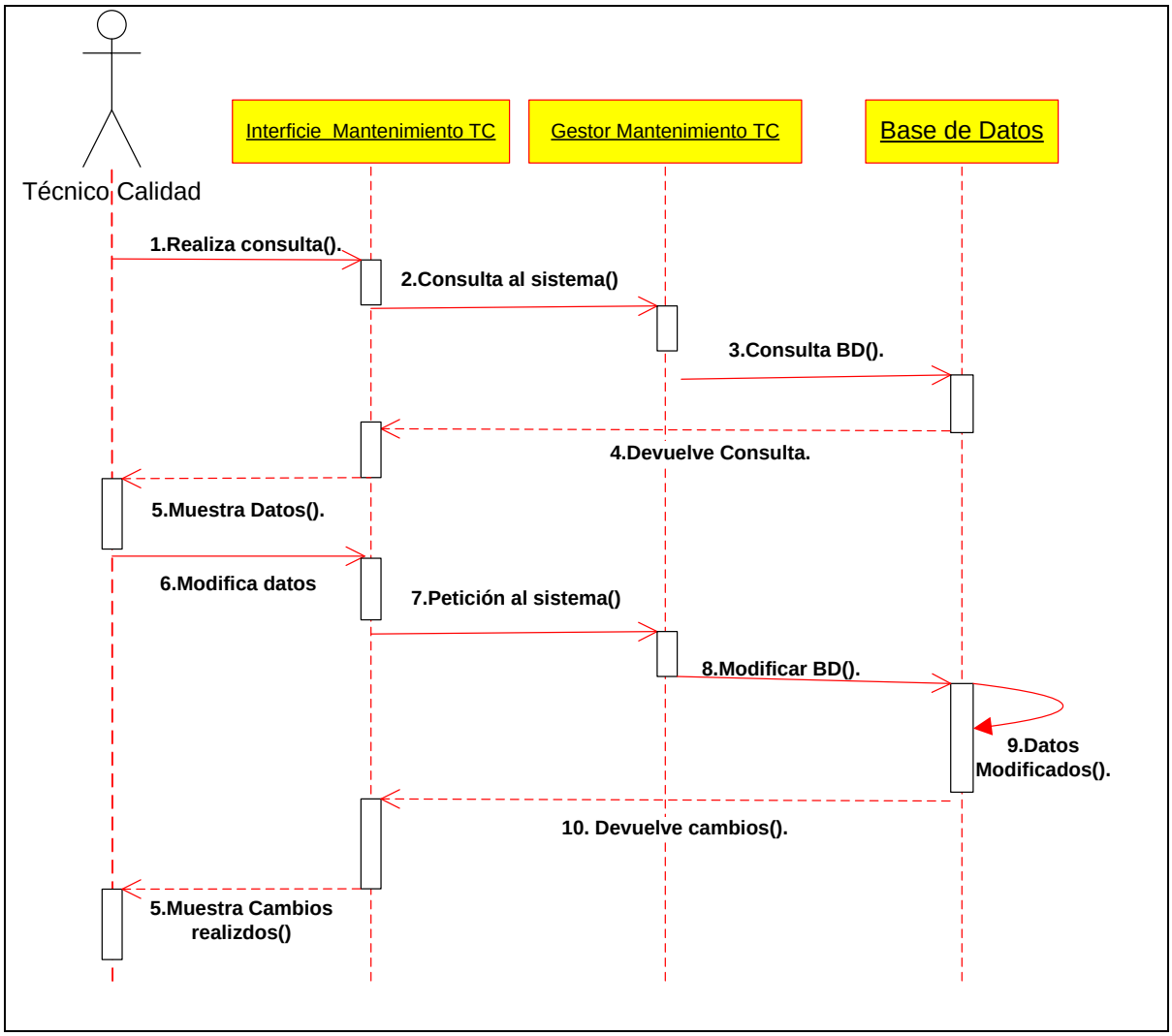

Figura 3.2. Diagrama secuencia Mantenimiento de los Casos de Test.

A continuación se explican cada una de las clases que intervienen.

- **Técnico Calidad:** Actor que será quien haga las peticiones al sistema.
- **Interficie Mantenimiento TC:** Formulario web donde realizara el actor todas sus funciones.
- **Gestor Mantenimiento:** Se encarga de hacer todas las peticiones con la BD, tanto de lectura como de escritura.
- **Base de Datos:** Clase de donde se obtienen los datos y son modificados.

Planificación de ejecución de Test.

Secuencia donde los test son programados para su ejecución.

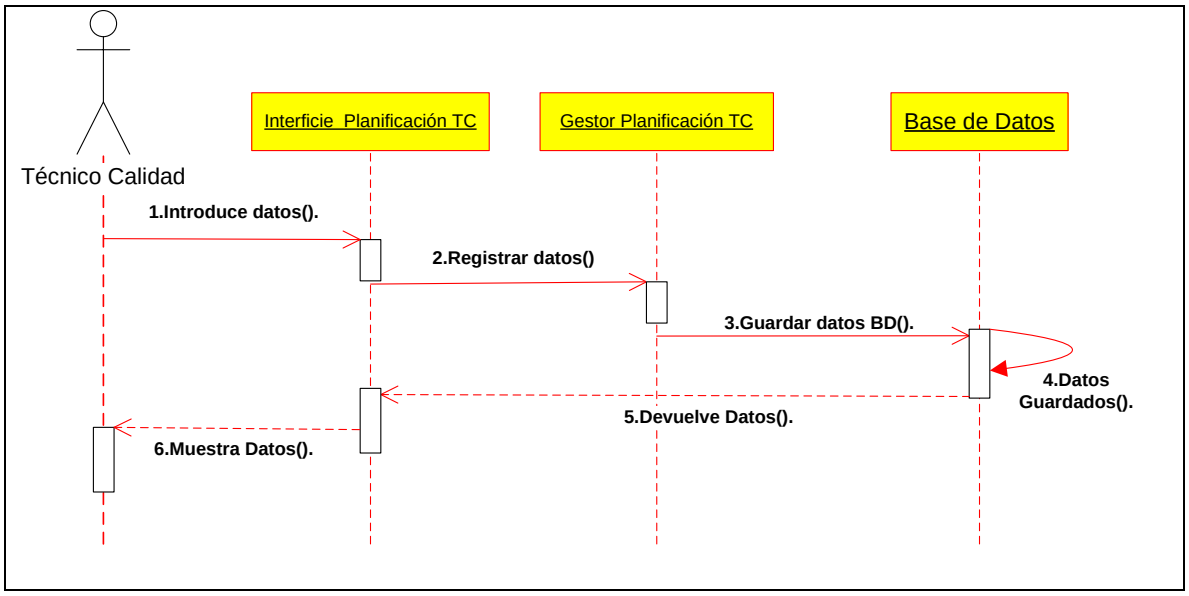

Figura 3.3. Diagrama de secuencia de planificación ejecución de test.

Como en el caso anterior, se procede a explicar en qué consiste cada componente:

- **Técnico Calidad:** Actor que será quien haga las peticiones al sistema.
- **Interficie Planificación TC:** Formulario web donde realizara el actor todas sus funciones.
- **Gestor Planificación:** Se encarga de hacer todas las peticiones con la BD, en este caso solo guardar datos
- **Base de Datos:** Clase de donde se modifican datos y posteriormente se muestran los resultados.

Ejecución de Test.

Secuencia donde se muestra el procedimiento para la ejecución de un TC.

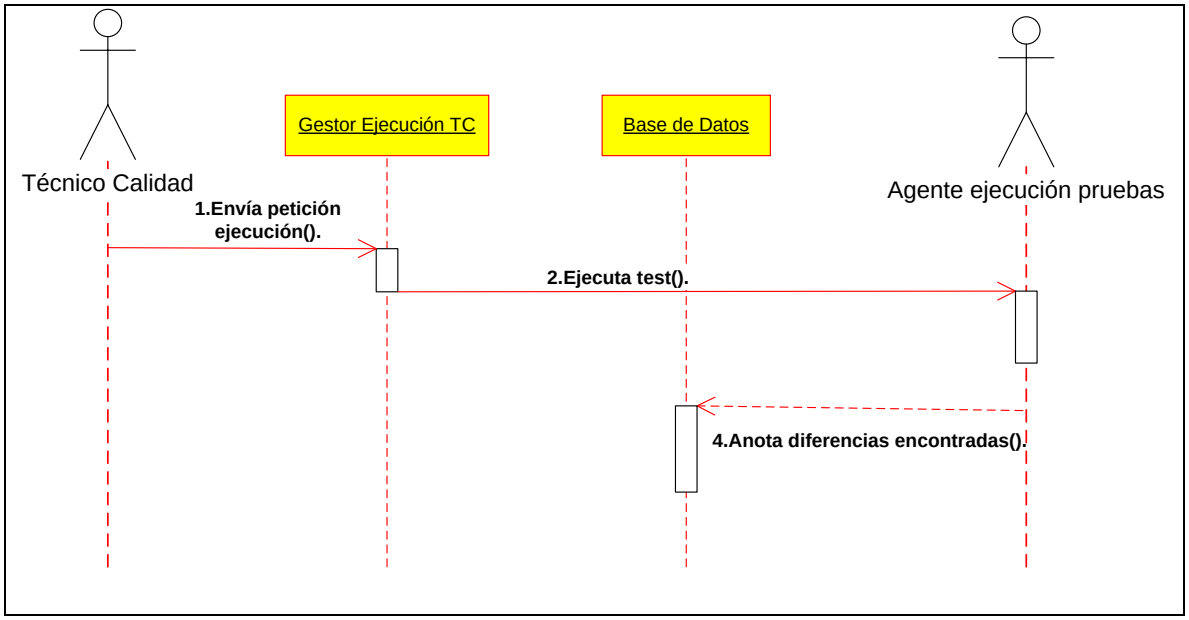

Figura 4.4. Diagrama de secuencia de ejecución de test.

A continuación se explican cada una de las clases que intervienen.

- **Técnico Calidad:** Actor que será quien haga las peticiones al sistema.
- **Gestor Ejecución TC:** Se encarga de hacer todas las peticiones que hace el técnico de calidad con el agente de ejecución de pruebas
- **Base de Datos:** Clase de donde se obtienen los datos y son modificados.
- **Agente ejecución pruebas:** Corresponde a la máquina donde se ejecutará el script. Ejecuta el test y posteriormente anota los resultados en la BD**.**

Análisis resultados de test.

Secuencia donde se muestra la visualización de resultados del test.

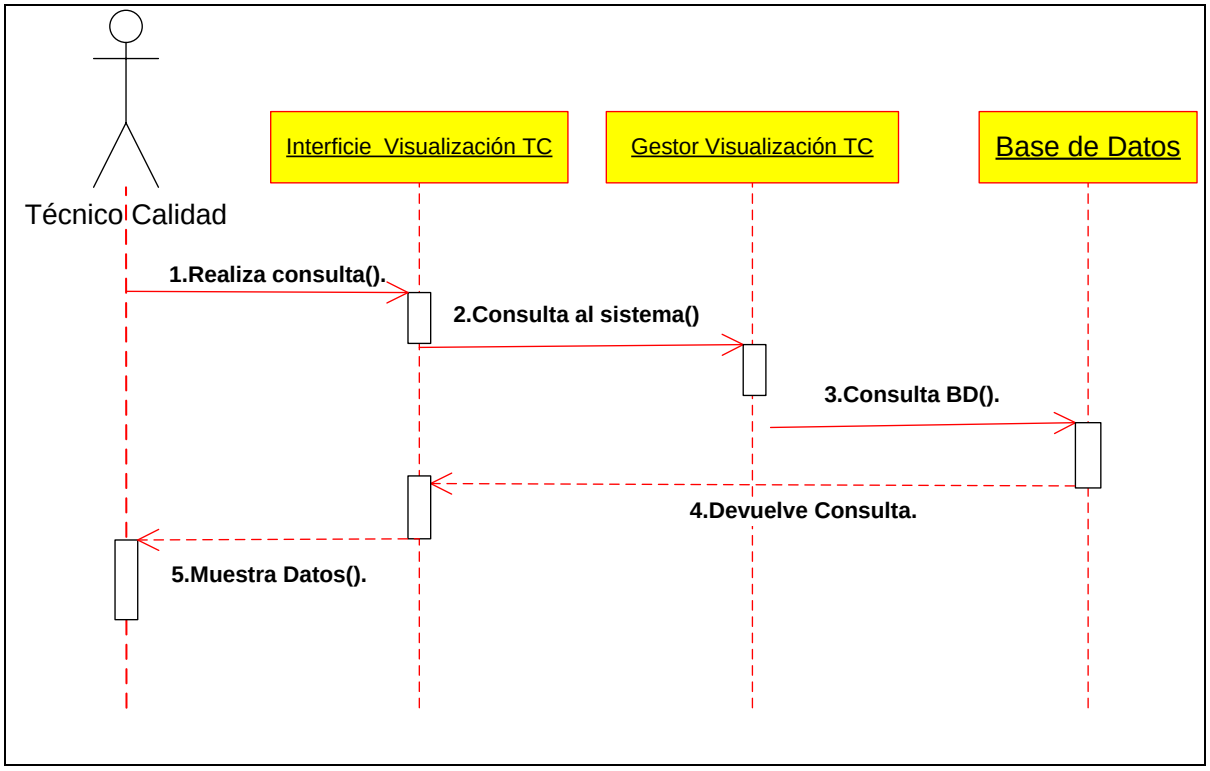

Figura 4.5. Diagrama secuencia visualización resultados del test.

Los diferentes componentes que lo componen, son:

- ***Técnico Calidad:*** Actor que será quien haga las peticiones al sistema.
- ***Interficie Visualización TC:*** Donde se hace la solicitud para visualizar datos y se muestran.
- ***Gestor Visualización TC*****:** Encargado de hacer la petición de leer los datos solicitados.
- ***Base de Datos:*** Se hace la consulta de los datos solicitados y los envía para su visualización.

Capítulo 5.

Diseño.

En el siguiente capítulo, se explicará con detalle en que consiste el diseño de la plataforma para la automatización de las *pruebas de regresión* para la aplicación *Logic Class.* Los apartados son los siguientes:

- ***Selección del entorno de desarrollo.*** Herramientas utilizadas para llevar a cabo el desarrollo de las aplicaciones.
- ***Configuración de la plataforma*****.** Configuración de herramientas y servicios necesarios en una estación de trabajo.
- ***Arquitectura*****.** Qué tipo de arquitectura de software se ha implementado así como sus partes.
- ***Módulos del sistema*****.** Cada una de las partes lógicas de las que se compone el sistema.
- ***Diseño de la interfaz gráfica*****.** Diseños de pantallas así como su funcionamiento.
- ***Diseño de la Base de Datos.*** Su estructura y las tablas que la componen.

5.1 SELECCIÓN DEL ENTORNO DE DESARROLLO.

Como ya se ha visto en capítulos anteriores, hay 2 partes bien diferenciadas: el Website *Ciclo de Vida* y ejecución de test con *TestPartner*. A continuación se explica las herramientas que se han utilizado para su desarrollo.

Ciclo de Vida.

Se han añadido nuevas funcionalidades a un proyecto ya existente por tanto no ha habido elección posible respecto a la herramienta para su desarrollo. Se ha utilizado Visual Studio 2005 y concretamente se ha desarrollado en lenguaje C #.

Para la manipulación de la Base de Datos la herramienta utilizada ha sido SQL Server 2000, en este caso pasa lo mismo, es un proyecto con una BD implícita lo que no hay elección posible.

TestPartner.

Para la automatización la aplicación utilizada es *TestPartner*. E**s** una **GUI de pruebas de software**[1] de herramientas de Micro Focus que tiene por objeto permitir a los equipos de proyecto de desarrollo de software para automatizar y funcionalmente aplicación de prueba las interfaces gráficas de usuario, con el objetivo de poder cumplir con las pruebas más aplicación en una determinada cantidad de tiempo que podría llevarse a cabo manual.
El código de desarrollo utilizado es Microsoft Visual Basic para Aplicaciones (VBA), que tiene licencia de Microsoft. Con VBA con los usuarios tienen acceso a un entorno de desarrollo integrado (IDE) que proporciona el núcleo VisualBasic aplicación bibliotecas, así como todas las capacidades inherentes a VBA (depuración, manejo de errores, la capacidad para hacer referencia a las bibliotecas públicas, etc.).

[1].GUI de pruebas de software: es el proceso de pruebas de un producto que utiliza una interfaz gráfica de usuario, para garantizar que se cumplen sus especificaciones escritas. Esto se hace normalmente a través de la utilización de una variedad de casos de prueba.

5.2 CONFIGURACIÓN DE LA PLATAFORMA.

Como ya se ha explicado en capítulos anteriores, la interfaz la cual el usuario utilizará es el website llamado *Ciclo de Vida.* Desde allí podrá tanto programar las tareas correspondientes así como visualizar los resultados. Su estructura es la siguiente:

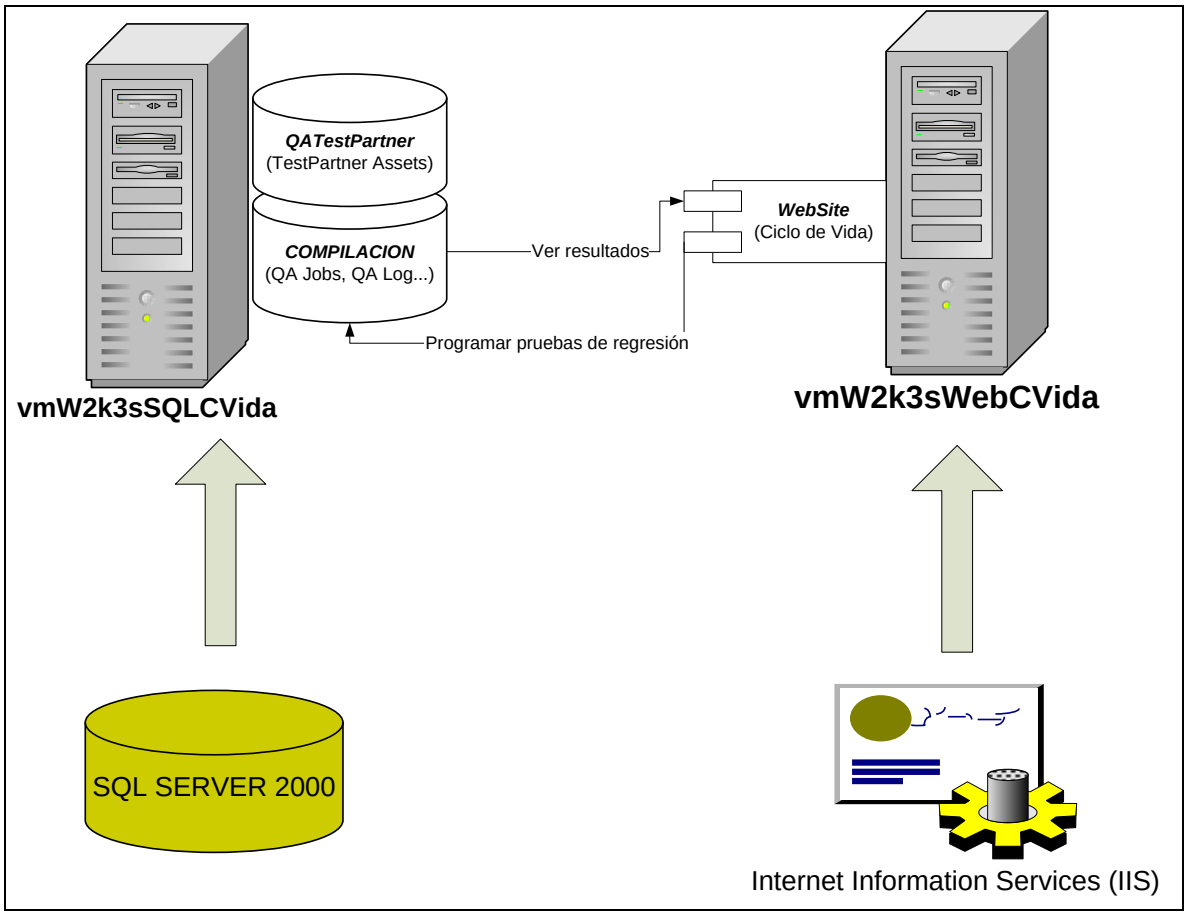

Figura 5. 1 Estructura website Ciclo de Vida.

Nota: Se da por supuesto que los servidores están incluidos en el dominio logiccontrol.com (dominio corporativo de la empresa)

Como se puede observar en la figura anterior, Ciclo de Vida se compone de 2 servidores:

- ***vmW2k3sSQLCVida:*** Servidor donde se almacenan las Base de Datos por la cual se programarán los test y, posteriormente, se podrán ver los resultados de las pruebas.

  La aplicación utilizada es *SQL Server 2000*. Este es un Sistema Gestor de Base de Datos, es decir, que sirve de interfaz entre la BD, el usuario y las aplicaciones que lo utilizan.

- ***vmW2k3sWebCVida:*** Servidor que contiene el website. Para ello hay que tener instalado, a parte de Windows 2003 Server, el servicio de Windows Internet Information Services (ISS). Este servicio convierte a un ordenador en un servidor de Internet o Intranet es decir que en las computadoras que tienen este servicio instalado se pueden publicar páginas web tanto local como remotamente.

Una vez visto de las partes que se compone el website, se procede a explicar qué herramientas debe de disponer cualquier máquina para poder ejecutar dichas pruebas.

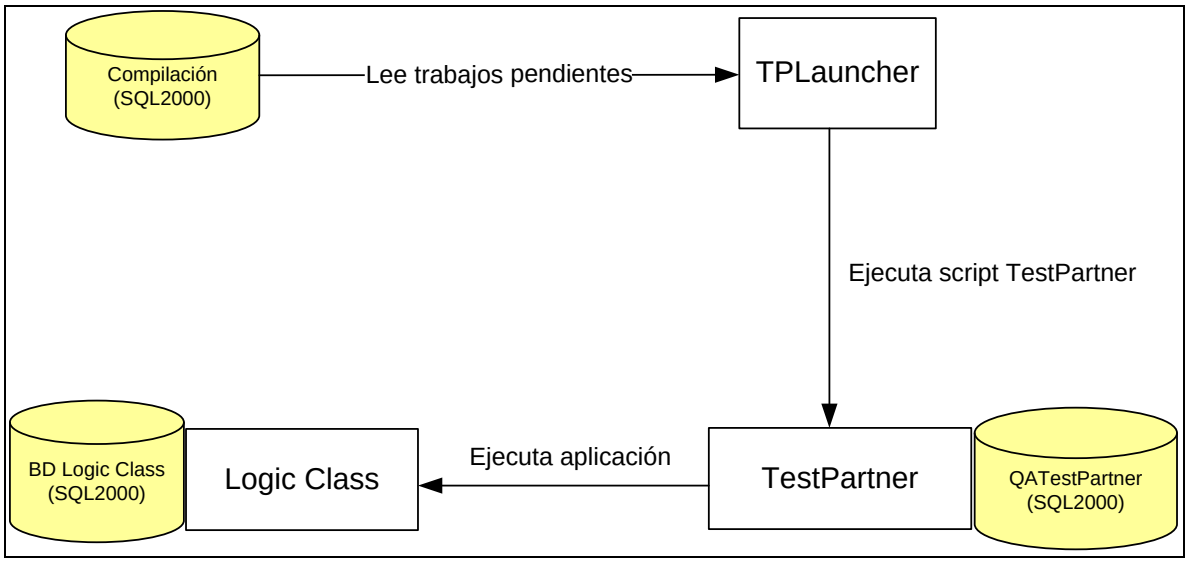

Figura 5. 2. Herramientas necesarias para la realización de las pruebas de regresión.

Como se puede observar en la figura anterior, la máquina la cual tiene que ejecutar las pruebas de regresión dispone de 3 aplicaciones en concreto:

- TPLauncher.
- TestPartner.
- Logic Class.

Todas estas aplicaciones se conectan a BD diferentes de SQL 2000. Para poder conectarse a estas BD, se necesita un componente: MDAC (Microsoft Data Acces Components).
Este componente se instala con la instalación de *Logic Class* por lo que no es necesario instalarlo.

En resumen, lo único que hay que tener instalado son las aplicaciones vistas con anterioridad: TPLauncher, TestPartner y Logic Class. Una vez instaladas, se podrán ejecutar las *pruebas de regresión* sin ningún problema.

5.3 ARQUITECTURA.

La arquitectura que está implementada, se basa en el modelo de 3 capas. La ventaja principal de este estilo es que el desarrollo se puede llevar a cabo en varios niveles y, en caso de que sobrevenga algún cambio, sólo se ataca al nivel requerido sin tener que revisar entre código mezclado.
Las capas son las siguientes:

- ***Capa de presentación:*** es la que ve el usuario, le comunica la información y captura la información del usuario en un mínimo de proceso (realiza un filtrado previo para comprobar que no hay errores de formato). Esta capa se comunica únicamente con la capa de negocio.
- ***Capa de negocio:*** en esta capa, residen los programas que se ejecutan, se reciben las peticiones del usuario y se envían las respuestas tras el proceso. Se denomina capa de negocio (e incluso de lógica del negocio) porque es aquí donde se establecen todas las reglas que deben cumplirse. Esta capa se comunica con la capa de presentación, para recibir las solicitudes y presentar los resultados, y con la capa de datos, para solicitar al gestor de base de datos para almacenar o recuperar datos de él. También se consideran aquí los programas de aplicación.

- ***Capa de datos:*** es donde residen los datos y es la encargada de acceder a los mismos. Está formada por uno o más gestores de bases de datos que realizan todo el almacenamiento de datos, reciben solicitudes de almacenamiento o recuperación de información desde la capa de negocio.

Mediante la siguiente ilustración se puede observar como se estructuran las diferentes capas para poder realizar las *pruebas de regresión*.

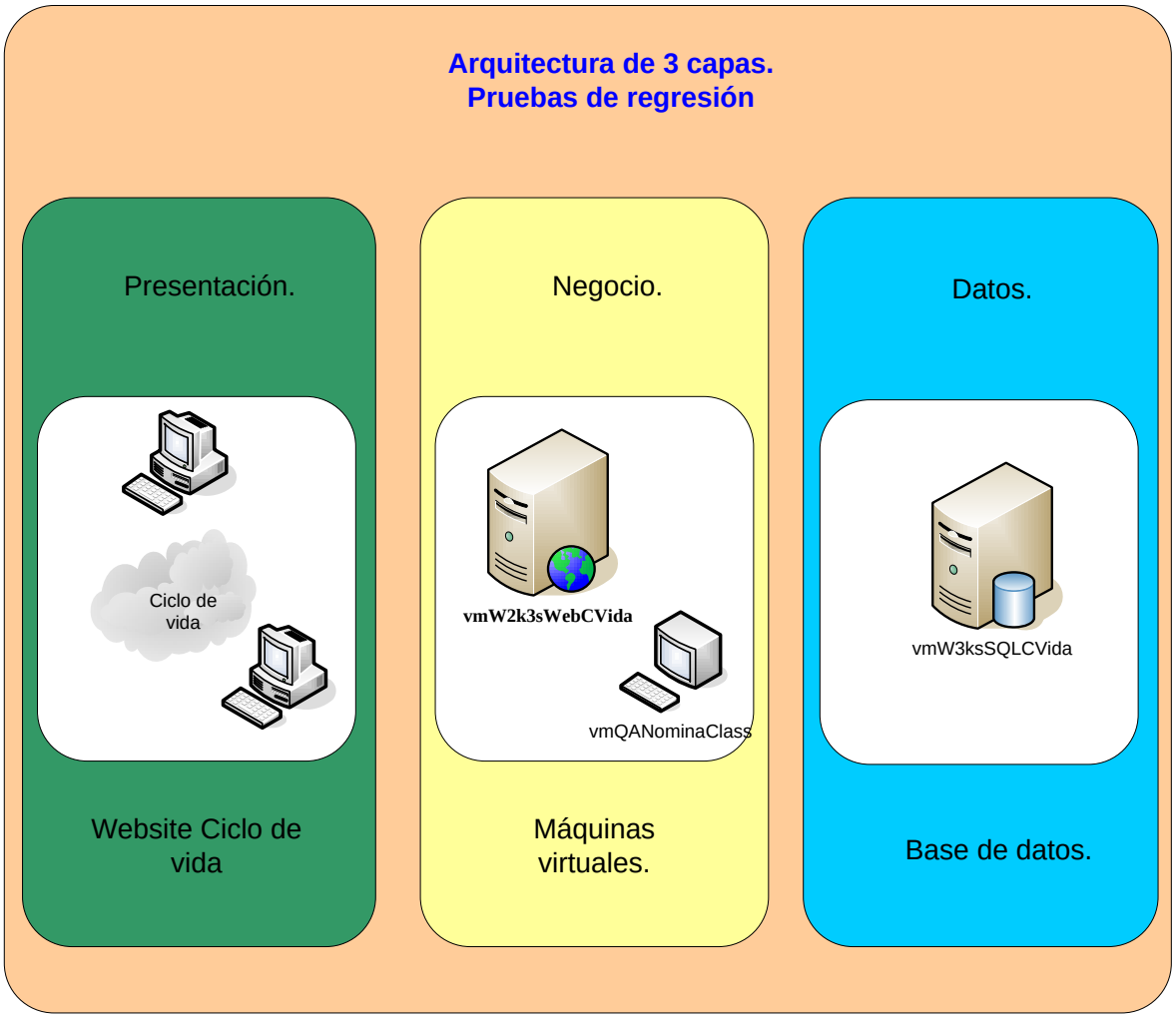

Figura 5.3. Arquitectura para las pruebas de regresión.

Como se puede observar en la ilustración anterior, las capas que se utilizan para la automatización son las siguientes:

- ***Presentación:*** Los técnicos de calidad son los que podrán hacer todo tipo de pruebas a través del website.
- ***Negocio****:* Donde se realizan todas las operaciones, corresponde a la parte automática, es decir las máquinas virtuales donde se ejecutan los test.
- ***Datos:*** Es donde se almacenan todos los resultados de las pruebas para que estos puedan ser consultados y analizados con detenimiento.

5.4 MÓDULOS DEL SISTEMA.

Los módulos que conforman el sistema se pueden agrupar en 2 subsistemas, el *Test Manager y el Test Runner.*

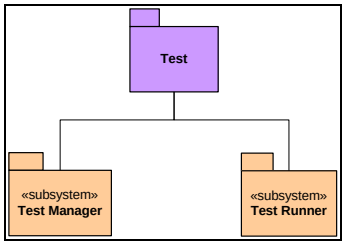

Figura 5. 4. Subsitemas.

El subsistema *Test Manager* aglutina los módulos que permitirán al usuario gestionar (mantener) los casos de test, planificar la ejecución de los tests, controlar el progreso de los tests que se estén ejecutando, y revisar el resultado de los tests ejecutados.
El subsistema *Test Runner* agrupa los módulos encargados de la ejecución de los tests, tanto manuales como automáticos. Esta ejecución incluye registrar el resultado de las validaciones.

Cada subsistema se compone de los siguientes módulos, a continuación se puede observar mediante la siguiente ilustración.

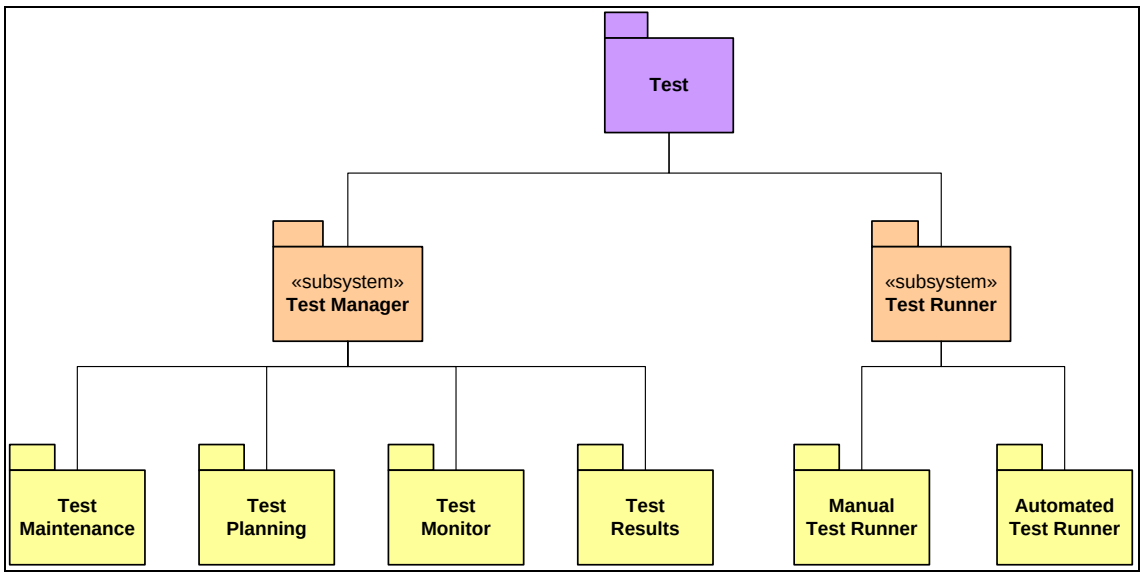

Figura 5.5 Módulos del sistema

Como se puede apreciar en la figura, el subsistema *Test Runner* es bien simple por lo tanto no se entrará en detalle en que consisten. En cambio el *Test Manager*, es algo más complejo por lo que se procederá a explicar sus módulos:

Test Maintenance.

Toda nueva funcionalidad, corrección o mejora, que tenga que ser implementada por el equipo de desarrollo en una nueva versión de un producto, debería llevar asociada al menos un test, ya sea manual o automático. Este test debe servir para comprobar que dicha implementación cumple con las especificaciones funcionales (análisis) y con las no funcionales si las hubiera (rendimiento, concurrencia, etc.).

Test Planning.

El *departamento de Calidad* es el encargado de planificar los test según crean convenientes. Tal y como se ha explicado en el módulo anterior, se suelen planificar al hacerse una nueva versión de un producto o al detectarse algún reportado por algún cliente. A partir de aquí lo que se permite es programar a un día y hora determinados.

Test Monitor.

Toda ejecución de test, tiene que tener un seguimiento de cómo se van procesando los datos, es esencial a la hora de poder detectar algún tipo de error durante su ejecución así como, en un futuro, poder introducir mejoras para un mejor funcionamiento.

Test Results.

Una vez finalizado el test, hay que comprobar el funcionamiento y poder visualizar todo tipo de errores que se hayan podido producir. Una vez visualizado los resultados se procede a su posible corrección, ya sea un error de programación o un error de concepto.

5.5 DISEÑO DE INTERFAZ GRÁFICA.

5.5.1 Introducción.

Para el diseño de la interfaz gráfica, solo se explicará el website ya que aquí es donde el usuario puede realizar alguna acción. El website ya existía así que simplemente lo que se ha hecho es añadir nuevas páginas y formularios para poder ejecutar las nuevas pruebas.

Las máquinas donde se ejecutan los test, todo se hace automático por lo que el usuario no realiza ninguna acción. Como consecuencia, no se ha hecho ninguna interfaz para el usuario y, por tanto, no se explica en este apartado.
A continuación se explicarán las nuevas páginas añadidas al *Ciclo de Vida*, así como sus formularios.

5.5.2 Diseño Ciclo de Vida.

Para poder acceder al website, hay que hacer login, por tanto si no es un usuario dado de alta, no podrá acceder. Para estar registrados y poder acceder al website, un administrador será el que tenga que dar de alta al nuevo usuario.

El funcionamiento de esta página es simple, solo se debe introducir usuario y contraseña. En la siguiente figura, se muestra la pantalla de acceso a *Ciclo de Vida.*

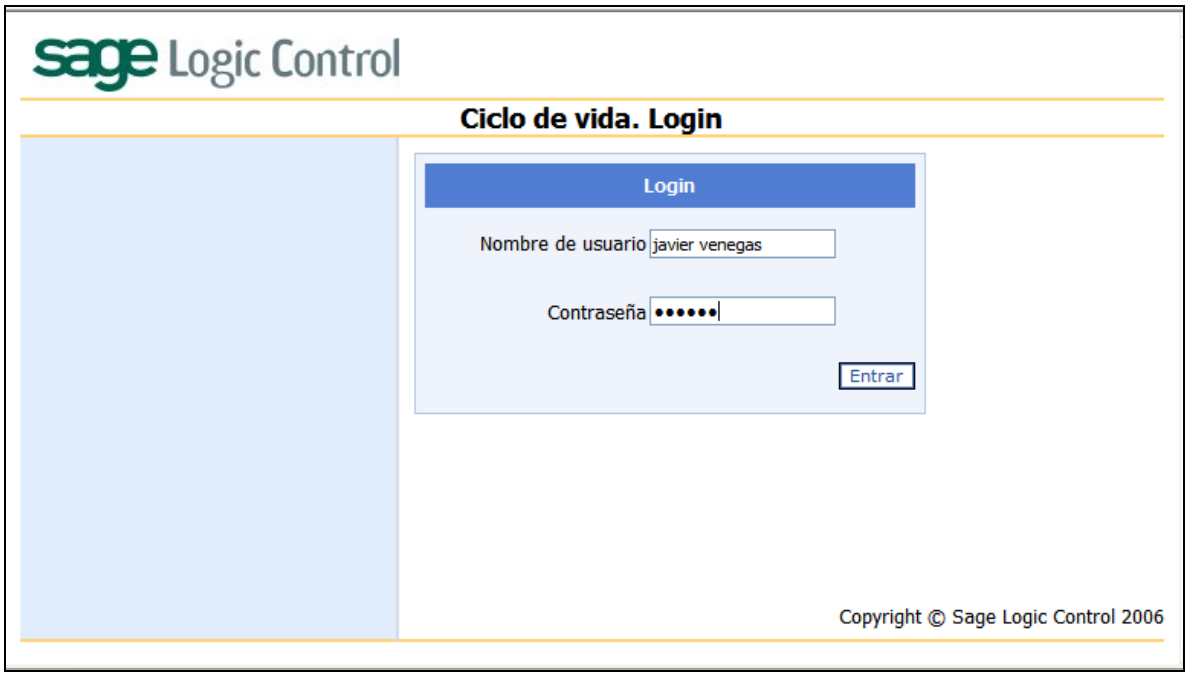

Figura 5.6. Acceso al ciclo de vida.

Una vez hecho login, se accede al menú principal. De este menú, lo que se ha añadido son nueva funcionalidades disponibles: *"LC / Nómina 10"*. Desde aquí se podrá hacer todo lo relacionado con las *pruebas de regresión* para *Logic Class*.
En la siguiente figura, se pueden observar cuales son las nuevas funcionalidades añadidas marcadas con un recuadro rojo.

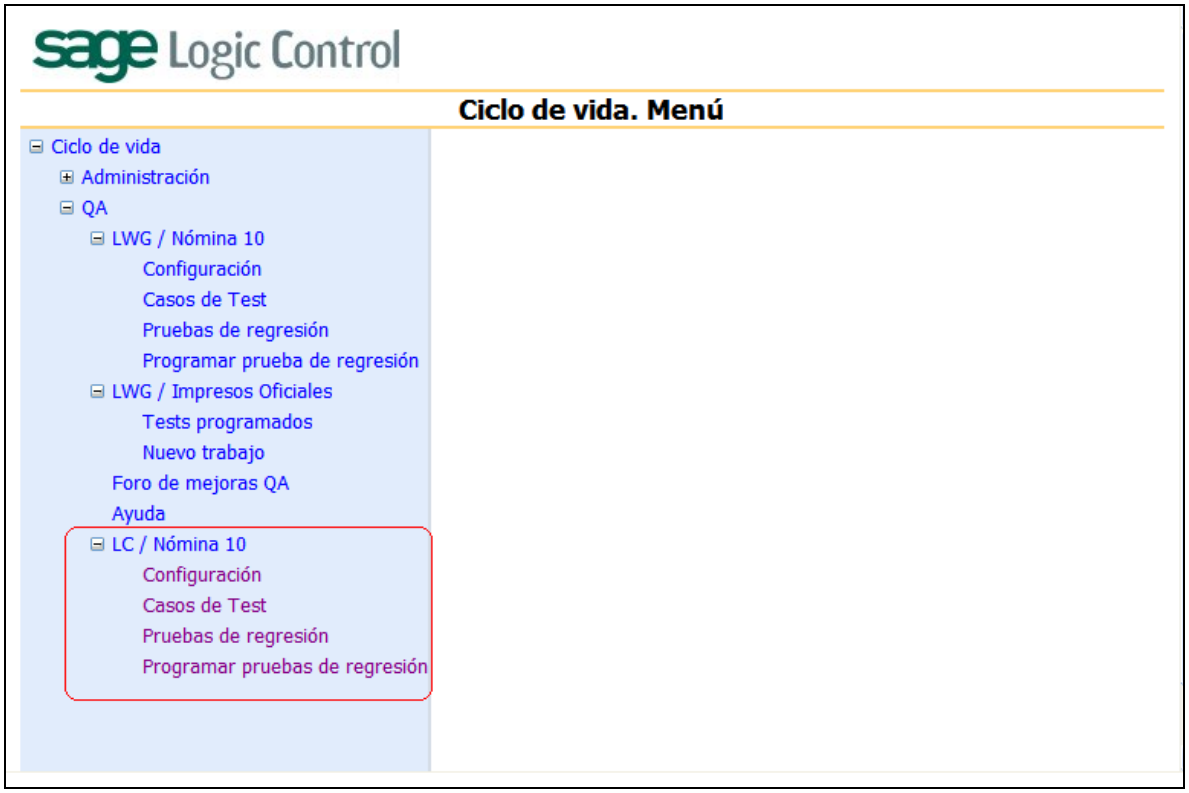

Figura 5. 7. Captura del menú principal Ciclo de Vida.

A continuación se explicará cada una de las partes del menú, se hará en orden descendente, es decir se empezará por explicar el apartado *"Configuración"* hasta acabar con *"Programar Pruebas de Regresión"*.

Configuración.

En este apartado, la función que se realiza es especificar la Base de Datos de Referencia, es decir, los datos con los que comparar una vez hechos los *Cálculos de Nómina y Seguros Sociales.*

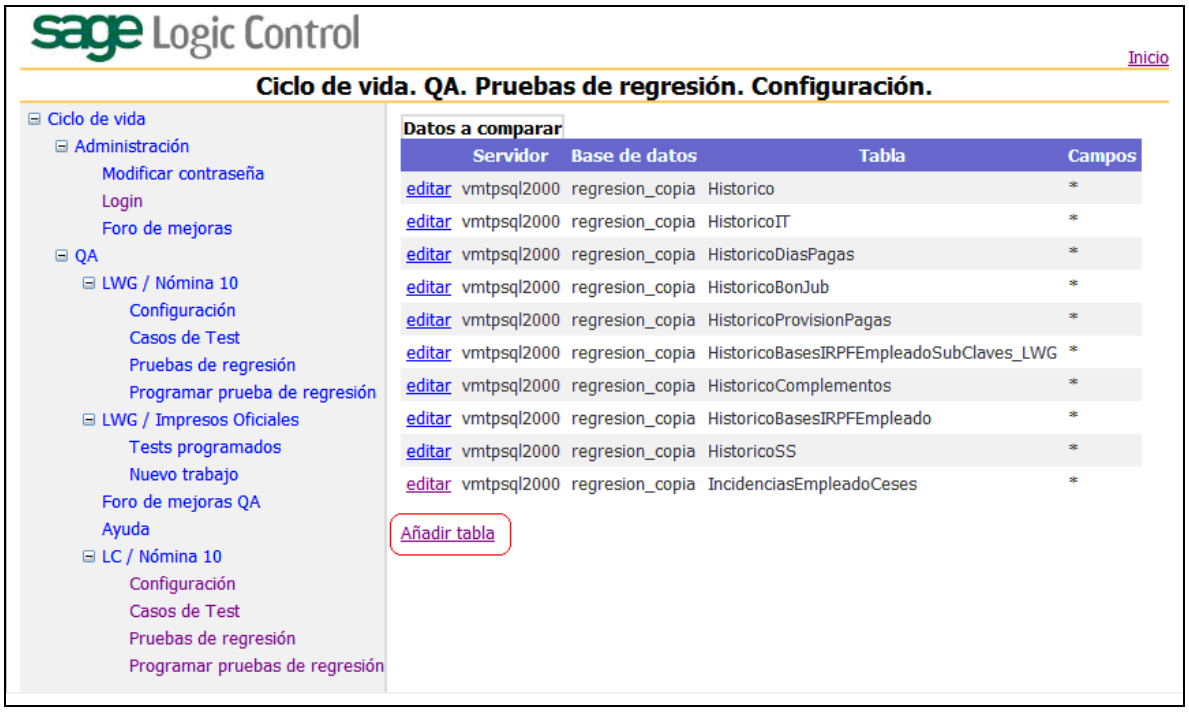

Figura 5. 8. Tablas de la Base de Datos de regencia.

Se pueden añadir tantas tablas como sean necesarias. Para añadir una nueva tabla, se deberá de acceder al enlace “Añadir Tabla”. Como se puede observar, este está marcado en rojo en la figura anterior.

Una vez accedido a este enlace, se rellena el formulario con los datos correspondientes. Como requisito es que ningún campo debe de quedar vacío.
Cuando se han introducido los datos, se pulsa el botón “Guardar” y quedará almacenada nueva tabla. En la siguiente figura se puede observar el formulario para rellenar los campos correspondientes.

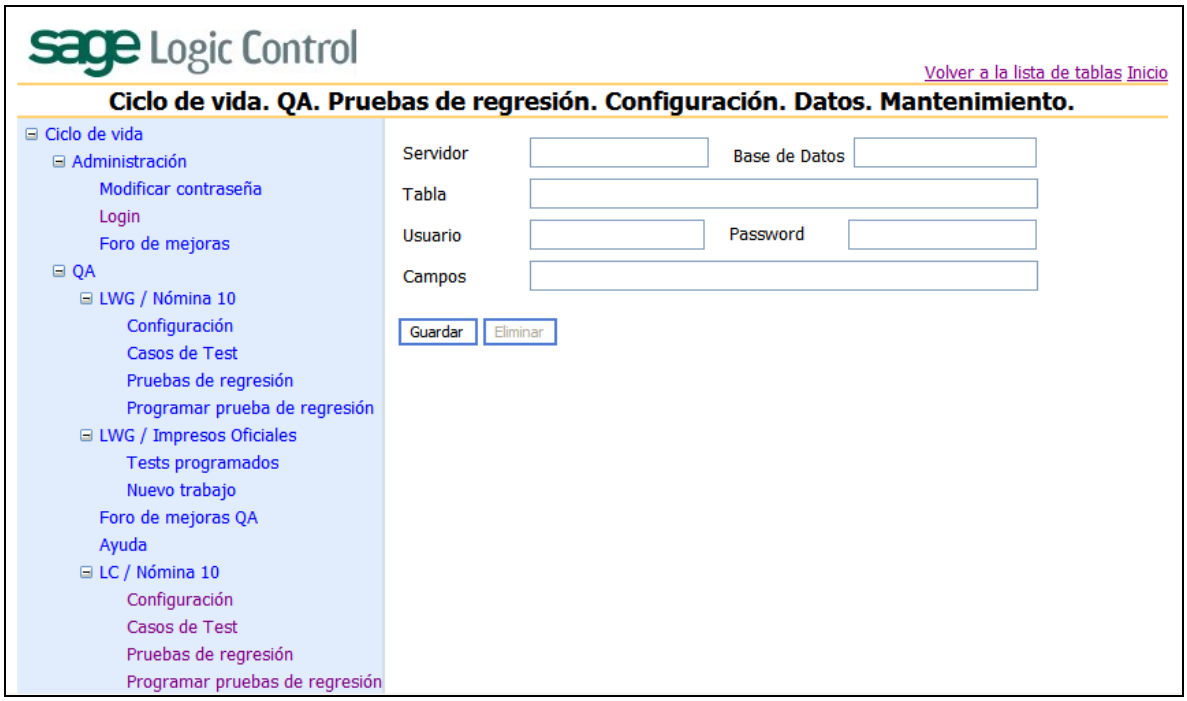

Figura 5. 9. Mantenimiento, creación de una nueva tabla.

Para eliminar o modificar una tabla, lo que se hace es acceder al apartado “Configuración” y acceder al enlace *“editar”* de una tabla ya dada de alta.

Figura 5. 10 Edición de una tabla.

Se accede al siguiente formulario y aquí se puede tanto modificar los campos que se deseen. Para ello, una vez finalizado se pulsa el botón “Guardar”.
Si se quiere eliminar completamente la tabla, se pulsa el botón “Eliminar”.

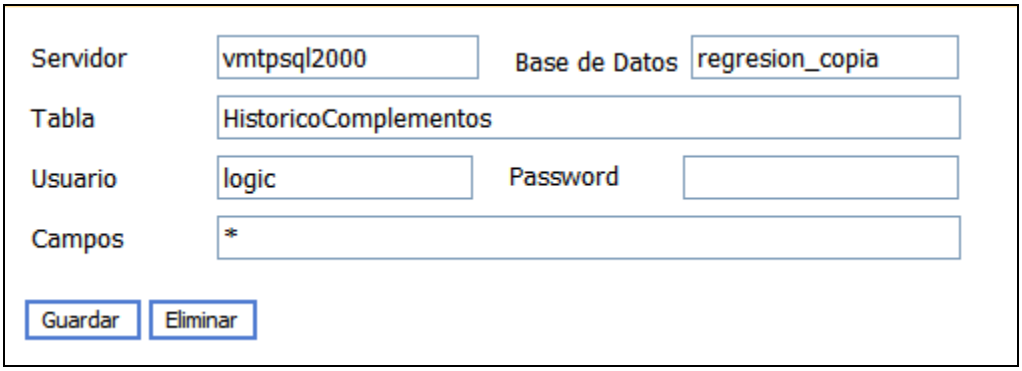

Figura 5. 11. Modificación/eliminación de una tabla.

Casos de Test.

Se accede mediante el menú pulsando sobre *"Casos de Test"*.A continuación se puede ver el contenido si se pulsa ese enlace, que no es más un resumen de todos los *Casos de Test* que se han dado de alta.

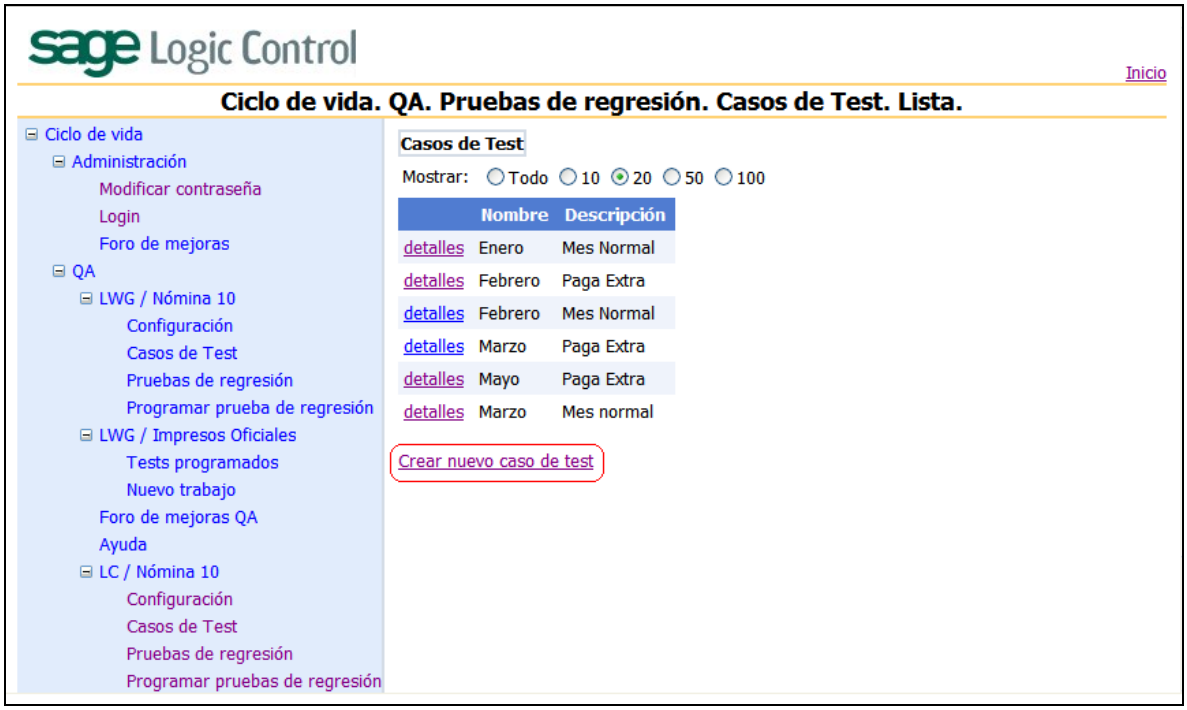

Figura 5.12. Como crear un nuevo Caso de Test.

Para añadir uno nuevo, se accede al enlace marcado en rojo en la figura anterior: "*Crear nuevo caso de Test"*. También se pueden añadir/modificar los ya existentes.

La siguiente figura muestra el formulario a rellenar para la creación de uno nuevo.

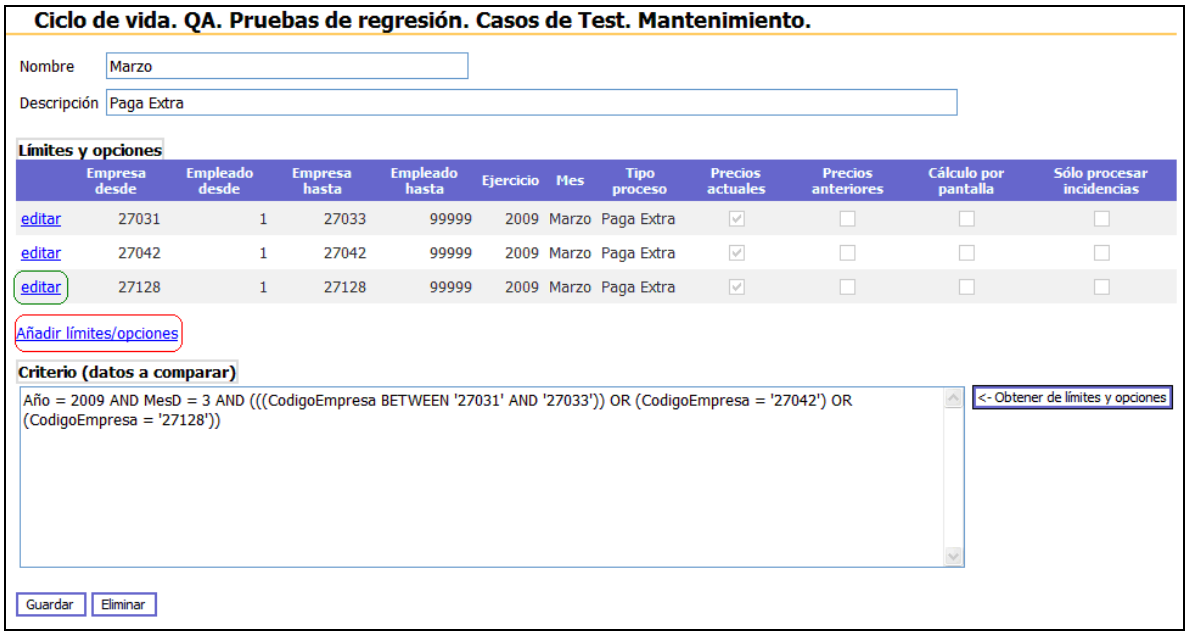

Figura 5. 3. Formulario para la creación de un Caso de Test.

Tal y como se ve en la figura anterior, aquí es donde se ve un resumen general del *Test Case.* En este apartado es donde se hace todo lo relacionado con el mantenimiento de un TC: se puede crear, modificar e incluso eliminar.

Hay que tener en cuenta que un TC puede tener varios límites/opciones. Para añadir uno nuevo se accede al enlace *"Añadir límites/opciones"*, remarcado con un recuadro rojo.
Si lo que se quiere es modificar/eliminar "Limites y opciones", lo que se debe de hacer es acceder al enlace "editar" (marcado en la figura anterior con un recuadro de color verde.)

Como se puede observar en la figura, se accede al mantenimiento de opciones y límites de los Casos de Test. Como ya se ha explicado con anterioridad, en el mantenimiento las opciones son desde crear uno nuevo, modificarlo e incluso eliminarlo.

Para crear uno nuevo, se llenan los campos con sus respectivos valores y cuando se ha acabado se pulsa el botón “Guardar”. Para ello previamente se debe de acceder desde el enlace “añadir límites y opciones”.

Para modificar, se accede tal y como se explicó anteriormente desde el enlace “editar”. Se modifican los respectivos campos y una vez modificados se pulsa el botón “Guardar”.

Para eliminar, simplemente se accede como en el caso de modificar pero se pulsa el botón “Eliminar” y estos límites quedarán borrados.

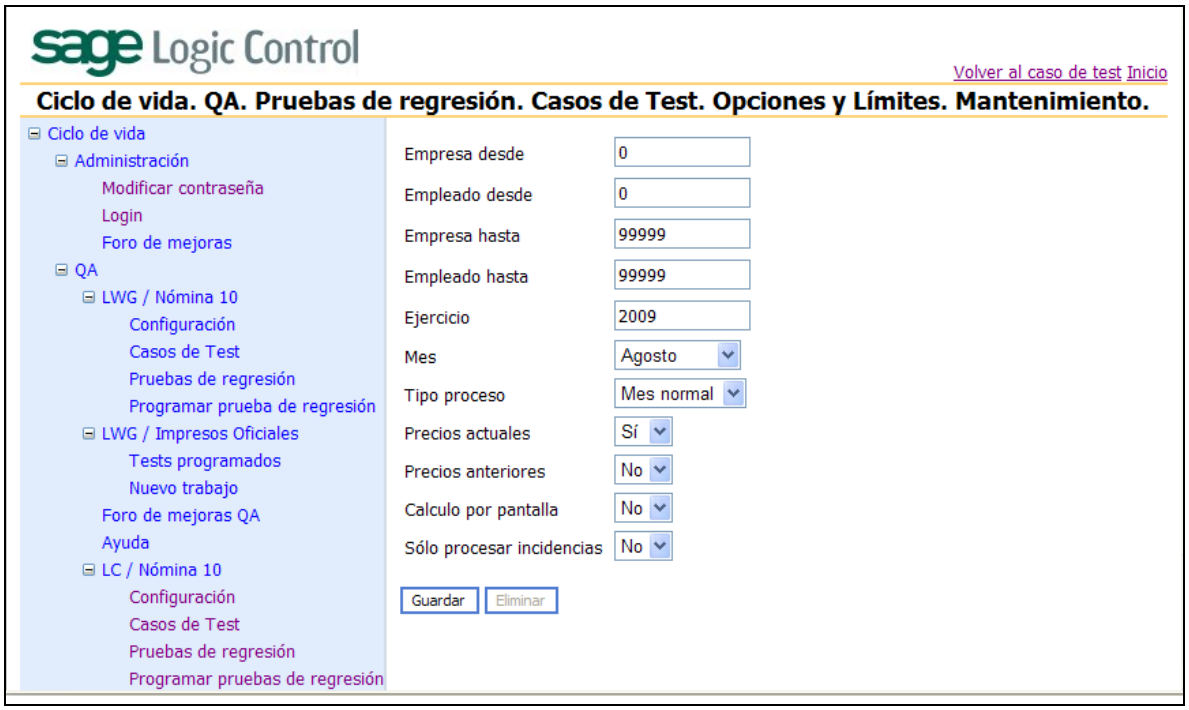

Figura 5. 4. Mantenimiento de los límites y opciones de los Casos de Test.

Pruebas de Regresión.

Una vez creados los TC con sus límites y opciones se procede a explicar en que consiste este apartado. Como se puede ver en la figura de más abajo, consiste en una lista de todos los trabajos realizados.

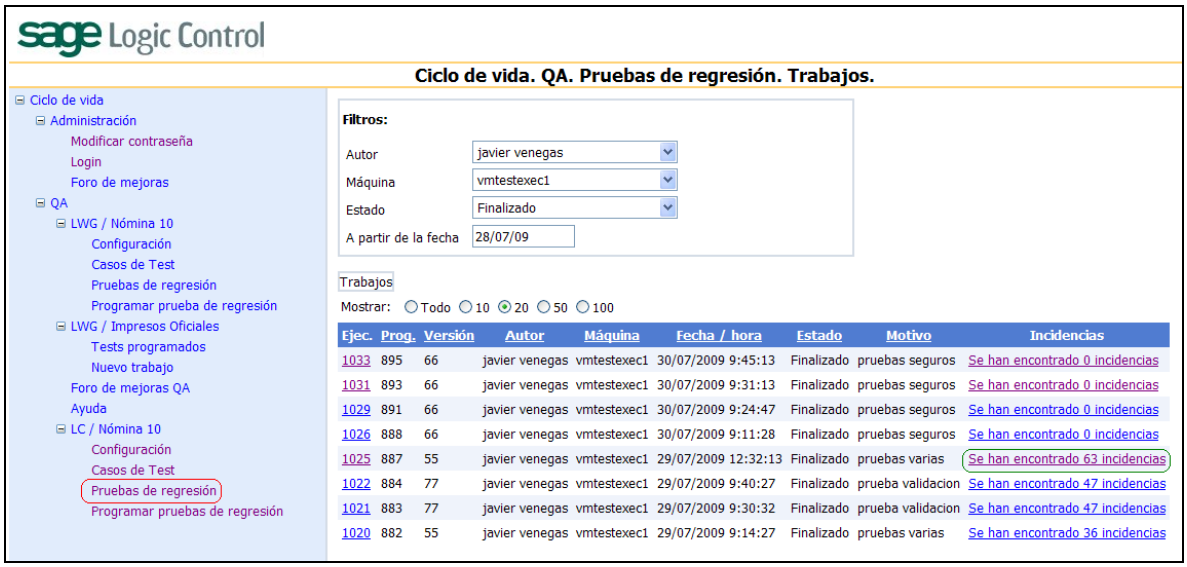

Figura 5. 5 Resumen de los trabajos realizados.

Si se quiere ver con detalle las incidencias ocurridas durante su ejecución, se pueden ver mediante el enlace como el que aparece en la figura marcado con un recuadro verde.

Una vez pulsado ese enlace, lo que se puede observar es un resumen del resultado del trabajo.

Figura 5. 6. Resumen de un trabajo en concreto.

Primero, se puede observar que se tiene una información general tal como el usuario que lo ha ejecutado, la máquina donde se ejecutó, el inicio, el fin, etc.

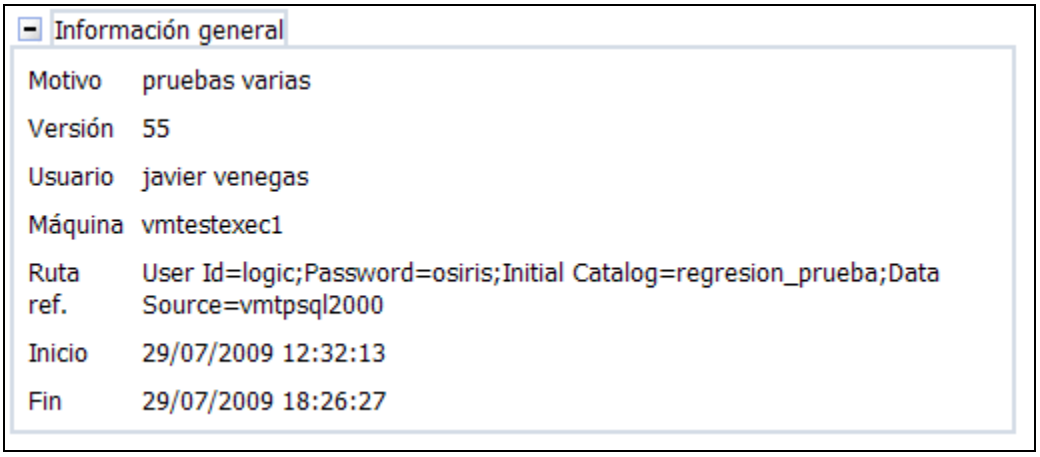

Figura 5. 7 Información general de un trabajo realizado.

En segundo lugar, aparece del resultado del test, las incidencias encontradas así como los registros procesados.

Figura 5. 8. Resultado de un trabajo realizado.

Finalmente, se puede observar un resumen de los casos de test procesados. Como se puede ver en la figura, existe un enlace para poder repetir las mismas pruebas con los mismos casos de test.

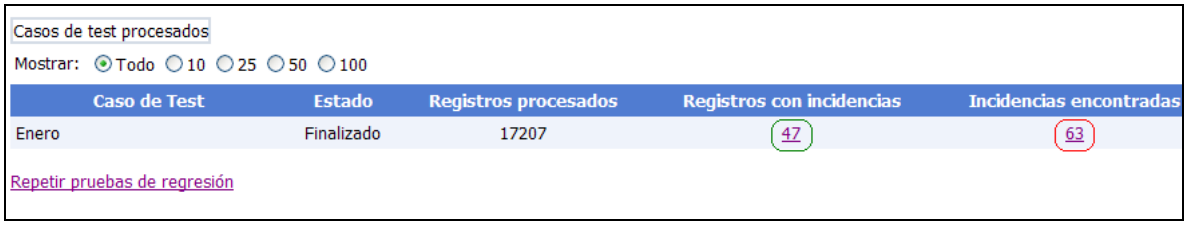

Figura 5. 9. Casos de test procesados en un trabajo.

Si se observa con detenimiento la figura anterior, hay marcados 2 enlaces: “Registros con incidencias” (recuadro verde) y “Incidencias encontradas” (recuadro rojo).

Si se accede al enlace “Registros con incidencias”, lo que se podrá observar es un resumen del resultado del trabajo.

Registros procesados

Mostrar: ○ Todo ○ 10 ◉ 20 ○ 50 ○ 100

Empresa	Empleado	Tipo proceso	Año	Mes	Incidencias
27024	340	MES	2009	1	Se han encontrado 1 incidencias
27024	343	MES	2009	1	Se han encontrado 1 incidencias
27024	390	MES	2009	1	Se han encontrado 1 incidencias
27024	393	MES	2009	1	Se han encontrado 1 incidencias
27120	5	MES	2009	1	Se han encontrado 6 incidencias
27120	5	MES	2009	1	Se han encontrado 15 incidencias
27120	7	MES	2009	1	Se han encontrado 1 incidencias
27125	10	MES	2009	1	Se han encontrado 1 incidencias
27125	15	MES	2009	1	Se han encontrado 1 incidencias
27125	25	MES	2009	1	Se han encontrado 1 incidencias
27125	5	MES	2009	1	Se han encontrado 1 incidencias
27125	6	MES	2009	1	Se han encontrado 1 incidencias
27125	8	MES	2009	1	Se han encontrado 1 incidencias
27126	10	MES	2009	1	Se han encontrado 1 incidencias
27126	9	MES	2009	1	Se han encontrado 1 incidencias
27129	41	MES	2009	1	Se han encontrado 1 incidencias
27180	106	MES	2009	1	Se han encontrado 1 incidencias
27180	128	MES	2009	1	Se han encontrado 1 incidencias
27180	129	MES	2009	1	Se han encontrado 1 incidencias
27180	155	MES	2009	1	Se han encontrado 1 incidencias

1 2 3

Figura 5. 20 Resumen de los registros encontrados con incidencias.

Al acceder al enlace siguiente “Incidencias encontradas” lo que se observa es con detalle el resultado del trabajo. Aquí es donde se visualizan los errores. Se pueden observar en los campos “Valor de referencia” y “Valor actual” si son diferentes, es cuando se interpreta que hay un error.

Incidencias

Mostrar: ○ Todo ○ 10 ◉ 25 ○ 50 ○ 100

Empresa	Empleado	Tipo Proceso	Año	Mes	Tabla	Resto de PK	Campo	Valor de referencia	Valor actual
27120	5	MES	2009	1	Historico		ImporteNom	56	112
27120	5	MES	2009	1	Historico		ImporteNom	400	800
27120	5	MES	2009	1	Historico		OImporte	M	I
27120	5	MES	2009	1	Historico		PrecioNom	40	80
27120	5	MES	2009	1	Historico		ImporteNom	18,8	37,6
27120	5	MES	2009	1	Historico		ImporteNom	6,6	13,2
27120	7	MES	2009	1	Historico		OImporte	M	I
27024	343	MES	2009	1	HistoricoSS		PendienteBonReforma	0	-15,61
27024	390	MES	2009	1	HistoricoSS		PendienteBonReforma	0	-13,75
27024	393	MES	2009	1	HistoricoSS		PendienteBonReforma	0	-15,61
27024	340	MES	2009	1	HistoricoSS		PendienteBonReforma	0	-13,75
27120	5	MES	2009	1	HistoricoSS		RemuneracionTotal	400	800
27120	5	MES	2009	1	HistoricoSS		TotalDeduccionesNom	81,4	162,8
27120	5	MES	2009	1	HistoricoSS		TotalDevengosNom	400	800
27120	5	MES	2009	1	HistoricoSS		LiquidoaPercibir	318,6	637,2
27120	5	MES	2009	1	HistoricoSS		PenCotBaseA	1766,2	1366,2
27120	5	MES	2009	1	HistoricoSS		PenCotBaseU	1766,2	1366,2
27120	5	MES	2009	1	HistoricoSS		CostesEmpMutua	16	20
27120	5	MES	2009	1	HistoricoSS		ImporteDtoDesempleo	6,2	12,4
27120	5	MES	2009	1	HistoricoSS		ImporteDtoFormacion	0,3999999999	0,7999999999
27120	5	MES	2009	1	HistoricoSS		BaseCCTotal	1400	1800
27120	5	MES	2009	1	HistoricoSS		CostesEmpCC	418,6	538,2
27120	5	MES	2009	1	HistoricoSS		CostesEmpCCConBo	167,44	215,28
27120	5	MES	2009	1	HistoricoSS		BaseAccDiasTra	400	800
27120	5	MES	2009	1	HistoricoSS		BaseAccTotal	1400	1800

1 2 3

Figura 5. 21. Incidencias encontradas con detalle.

Programar trabajo.

Para acabar, se procede a explicar en que consiste este apartado.

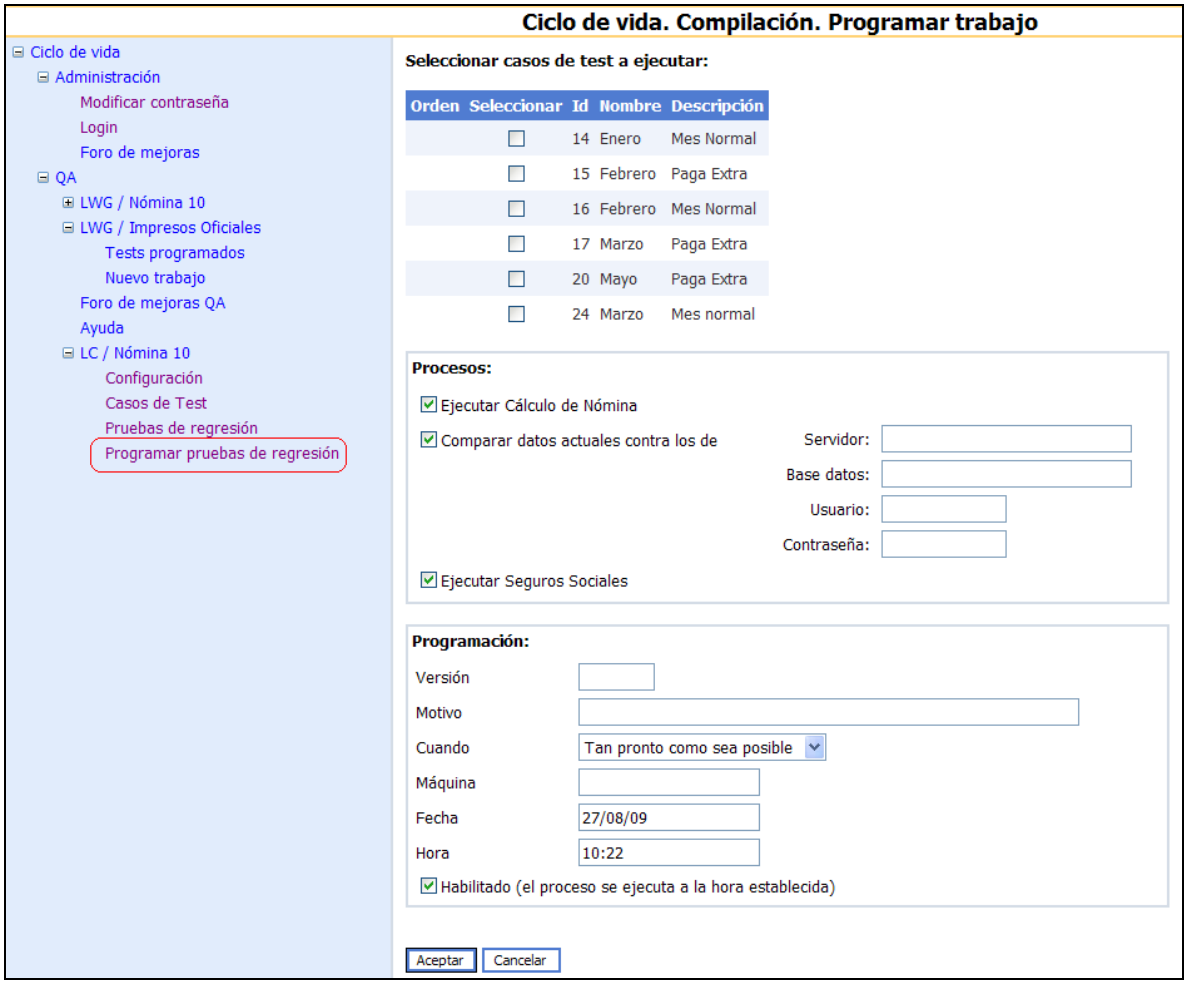

Figura 5. 22. Vista de cómo programar una Prueba de Regresión.

Tal y como su nombre indica, lo que se hace en este apartado es programar un test para ser ejecutado.

Hay 3 partes bien diferenciadas: “Seleccionar casos de test a ejecutar”, “Procesos” y “Programación”.

El primer apartado “Seleccionar casos de test a ejecutar” lo que se debe de hacer es seleccionar mediante un “checkbox” las pruebas que el usuario desee que se ejecuten. (Nota: Estos casos de test ya fueron creados en el apartado “Casos de Test” explicado con anterioridad).

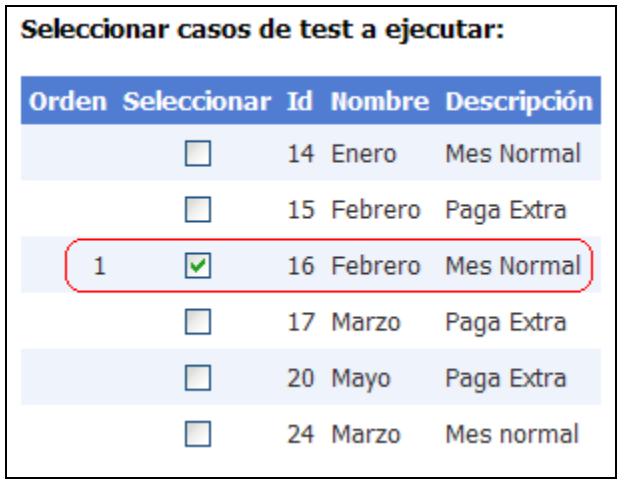

Figura 5. 23. Casos de test disponibles.

En el siguiente, “Procesos”, se eligen el tipo: “Cálculo de Nómina o Seguros Sociales”, así como la base de datos con la que se compararán los datos.

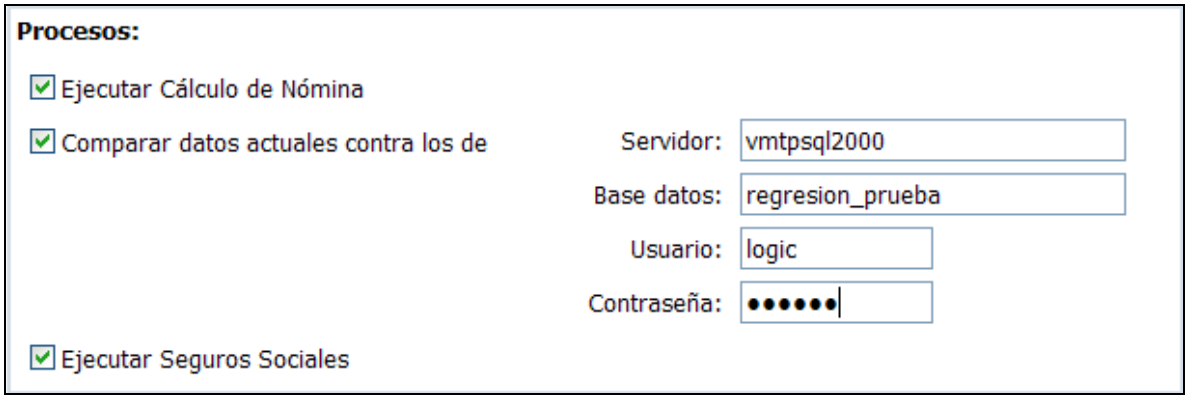

Figura 5. 10. Procesos a ejecutar en un mismo caso de test.

Finalmente en el apartado “Programación”, tal y como su nombre indica se introducen los parámetros para su ejecución, es decir, en qué máquina y cuando se ejecutará el test.

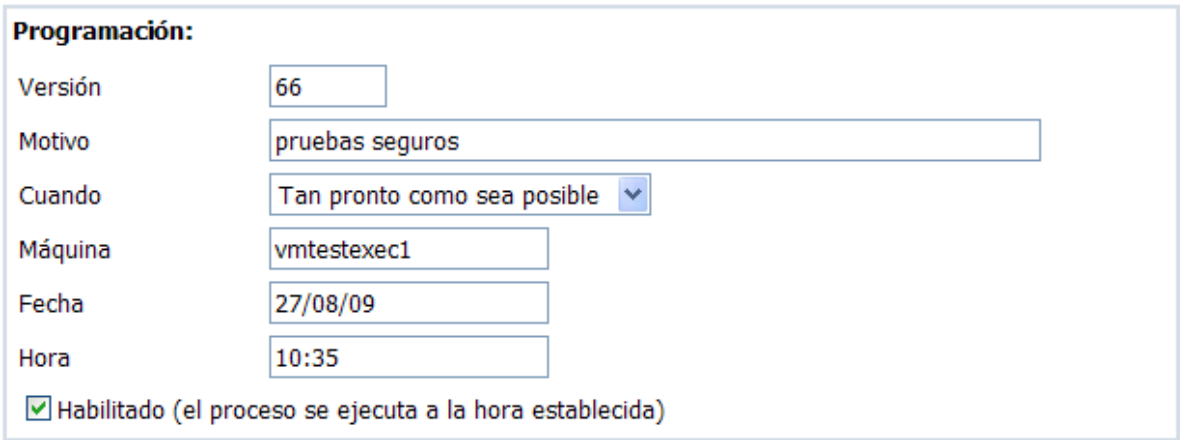

Figura 5. 11. Formulario para programar una prueba.

Una vez rellenados los apartados explicados con anterioridad se pulsa el botón “Aceptar” y la tarea quedará guardada lista para su ejecución.

5.6 DISEÑO DE LA BASE DE DATOS.

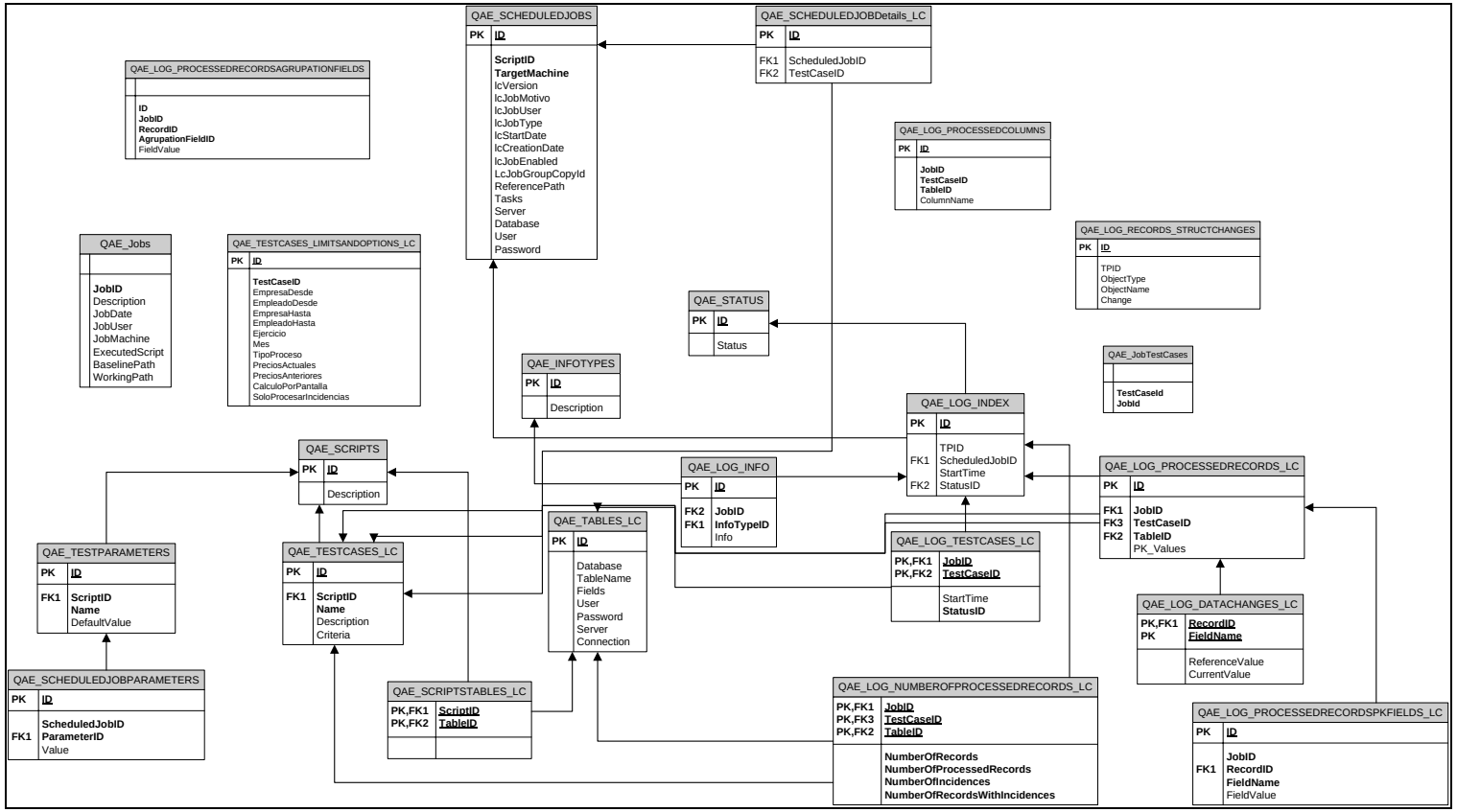

Figura 5. 12 Estructura de la Base de Datos.

Como se puede observar en la ilustración anterior, la estructura de la Base de Datos es un tanto compleja por lo que se explicará con detalle en que consisten las tablas, para ello las agruparemos en diferentes bloques

- *Programación de trabajos.*
- *Scripts.*
- *Registro eventos ejecución test*.

Programación de trabajos.

Como se puede ver en la ilustración de más abajo, este conjunto de tablas son las involucradas en la programación de una trabajo. *"QAE_SCHEDULEDJOBS, QAE_SCHEDULEDJOBDetails_LC y QAE_Jobs"* tienen que ver con la planificación, es decir, "donde" y "cuando" se ejecutará la tarea. Toda tarea tiene que ir relacionada con unos Casos de Test, en este caso se relaciona con las tablas *"QAE_TESTCASES_LC y QAE_TESTCASES_LIMITSANDOPTIONS_LC".* La planificación y sus Casos de Test van relacionados con la tabla *"QAE_JobTestCases".*

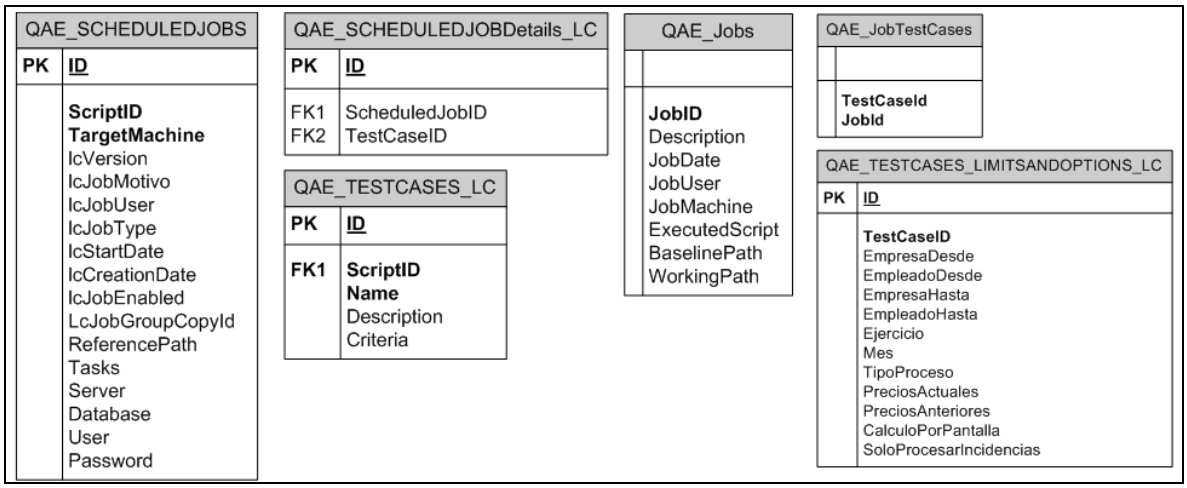

Figura 5. 13. Tablas relacionadas con la programación de un trabajo.

Scripts.

El siguiente conjunto de tablas tiene que ver todo lo relacionado con la ejecución de un Script: un nombre y un identificador (id) al cual hacen referencia, "*QAE_TESTPARAMETERS*" así como una base de datos de referencia asociado con la cual se compararán los datos "*QAE_TABLES_LC*".
La relación de estos datos se puede observar que se hace mediante la tabla "*QAE_SCRIPTSTABLES_LC*".

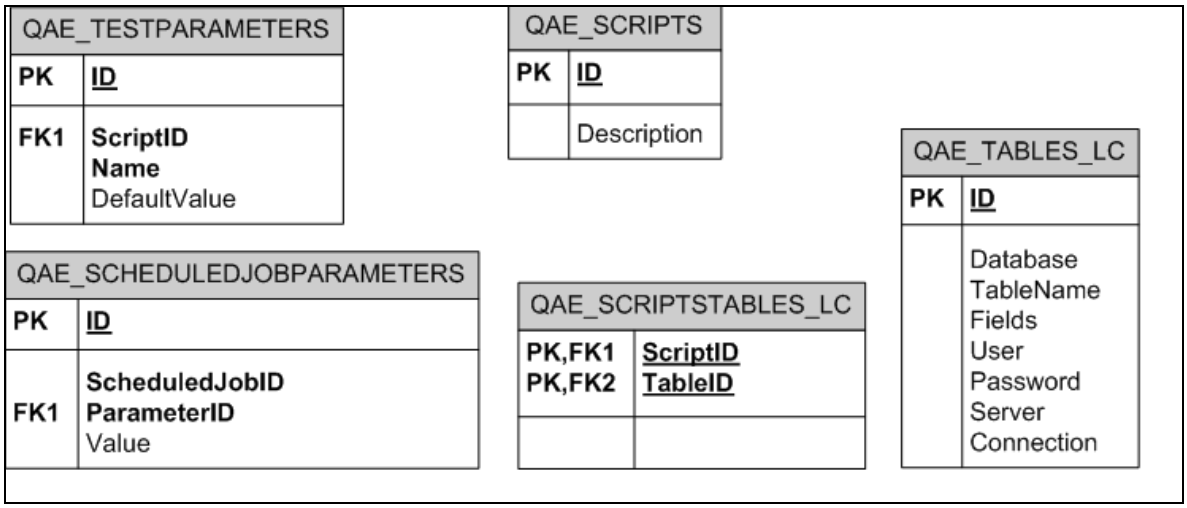

Figura 5. 14 Tablas relacionadas con la ejecución de un script.

Registro eventos ejecución test.

Como su nombre indica, son las tablas donde se anotan las incidencias encontradas durante la ejecución del test. Como se puede apreciar en la ilustración, la tabla principal la cual contiene la información y todas están relacionadas es *"QAE_LOG_INDEX".*
Esta tabla relaciona 3 partes.

- **Información general (color naranja):** Información como su descripción y su estado (finalizado, en ejecución, etc)
- **Registros en proceso (color verde):** A través de estas tablas, se puede saber el número de registros que han sido procesados.
- **Cambios (color azul):** En estas tablas se anotarán todas las diferencias encontradas en la comparación entre Bases de Datos.

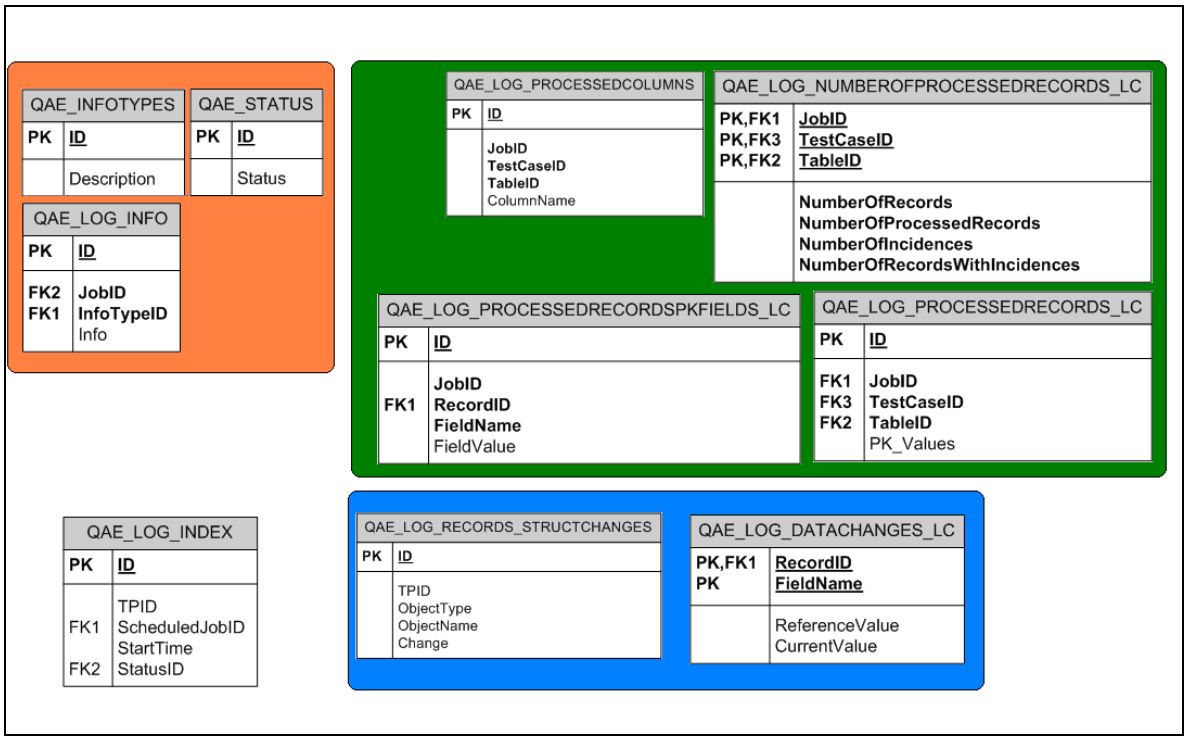

Figura 5. 15. Tablas relacionadas con las incidencias encontradas durante la ejecución de un test.

Capítulo 6.

Implementación y pruebas.

En el siguiente capítulo se mostrará cómo se ha hecho la implementación de cada una da las partes, así como las pruebas realizadas para su correcto funcionamiento.

- ***Implementación.*** Cómo se ha desarrollado del código así como su estructura**.**
- ***Pruebas.*** Diferentes tipos de pruebas que se han hecho una vez acabada la codificación para ver su correcto funcionamiento. En este caso, son de rendimiento y compatibilidad.

6.1 IMPLEMENTACIÓN.

Para el desarrollo de la plataforma, hay que diferenciar 2 partes: la máquina virtual donde se ejecutarán los test automáticos y el *Ciclo de Vida*, que fueron implementados en el mismo orden.

Máquina virtual.

Lo primero, ha sido la instalación de las aplicaciones necesarias para poder tener todos los recursos disponibles:

1. Instalación Windows XP en máquina virtual.
2. Instalación de *TestPartner* en máquina virtual. *TPLauncher* no necesita instalación, es un archivo ejecutable.
3. Instalación *Logic Class* en la misma máquina. *Logic Class* necesita una base de datos SQL Server 2000, por lo que se crea una en un servidor virtual disponible en la infraestructura de la empresa.

En la figura siguiente, se muestra un esquema de cómo serían los recursos necesarios.

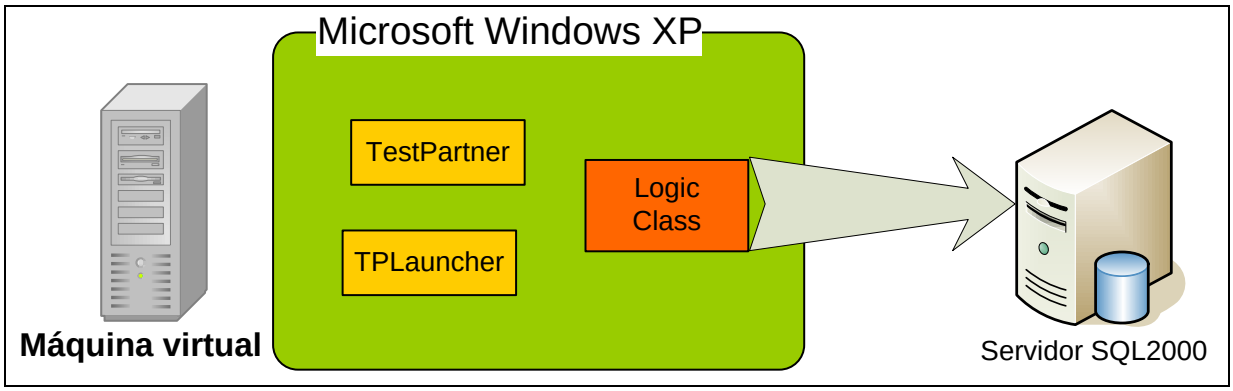

Figura 6. 1. Contenido aplicaciones máquina virtual.

Una vez instaladas las herramientas se procede a la elaboración del script de TestPartner. Se ha realizado en 2 fases:

1. ***Emulación del usuario en los menús de Class.*** Es decir, lo primero que hace el test es hacer funcionar *Logic Class* como si de un usuario común se tratara, ejecutando el *Cálculo de Nómina o Seguros Sociales*.
2. ***Comparativa entre Base de Datos.*** Una vez ha finalizado la ejecución de Class, se procede a hacer la comparativa entre los datos procesados por el cálculo de la aplicación, con una Base de Datos de referencia. Esta segunda parte ha sido implementada a la misma vez que se desarrollaban las nuevas funcionalidades del *Ciclo de Vida*, ya que sino es imposible probar su funcionamiento.

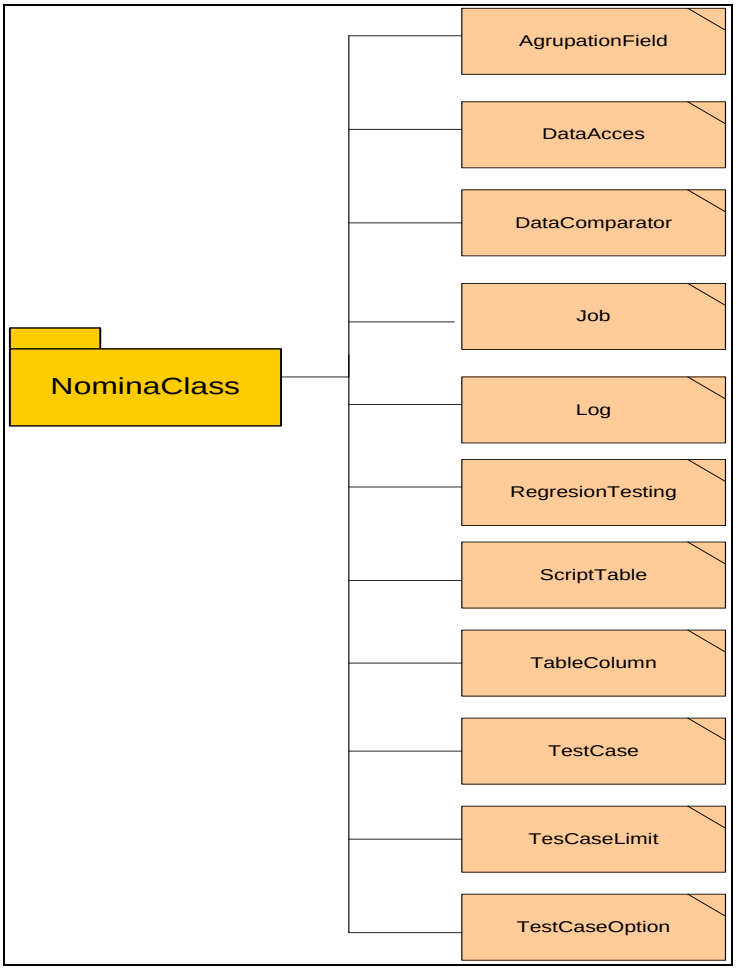

Figura 6. 2. Estructura del código script TestPartner.

Como se ha podido apreciar en la figura anterior, el código se estructura de una clase principal *("main")* llamada *NominaClass* y de ella se derivan todas las demás. A continuación se explica cada una de ellas la función que realizan.

- ***NominaClass:*** Clase principal donde se realiza toda la emulación del usuario al utilizar Logic Class. Desde aquí se llama a las diferentes clases.
- ***AgrupationFields:*** Clase-Tipo AgrupationFields (campos de agrupación de un script).
- ***DataAcces:*** Mini acceso a datos.
- ***DataComparator:*** Encargada de la comparación entre BD.
- ***Job:*** Clase-Tipo Job (Identificador de trabajo).
- ***Log:*** Log de pruebas de regresión.
- ***RegressionTesting:*** Clase que realiza las pruebas de regresión y anota las diferencias encontradas en el Log.
- ***ScriptTable:*** Clase-Tipo ScriptTable (tablas de un script).
- ***TableColumn:*** Estructura para guardar las características de una tabla.
- ***TestCase:*** Clase-Tipo TestCase (casos de test).
- ***TestCaseLimit:*** Clase-Tipo TestCaseLimit (limites asociados a un caso de test).
- ***TestCaseOption:*** Clase-Tipo TesCaseOption (opciones asociadas a un caso de test).

Nota: Para ver con más detalle el código, se puede ver en el anexo entregado**.**

Ciclo de Vida.

Este ha sido desarrollado en 2 fases:

1. Copia de la BD original para hacer pruebas ya que utiliza diariamente y, posteriormente, agregar las tablas necesarias para el desarrollo necesario para los nuevos test para Logic Class.

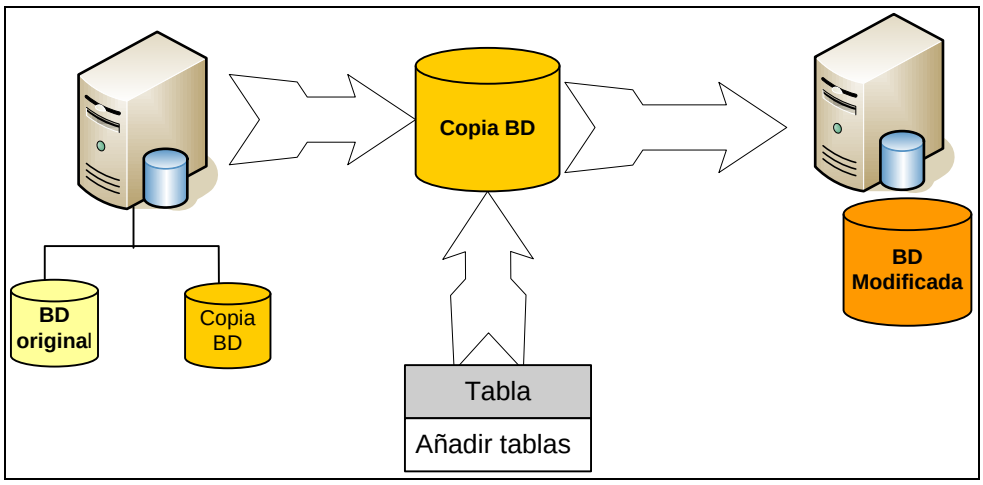

Figura 6.3 Modificación de las Bases de Datos.

2. Agregar las diferentes clases y páginas necesarias para agregar las nuevas funcionalidades que son requeridas. Primero de todo, se codifica en la máquina física de la que el alumno dispone (Visual Basic 2005 permite trabajar en este modo) una vez comprobado su correcto funcionamiento se agregan las nuevas clases al servidor donde se encuentra el website.

6.2 PRUEBAS.

A continuación se explica el tipo de pruebas al que ha sido sometida la plataforma para ver su correcto funcionamiento. Cabe destacar que es un entorno en el cual ya estaba funcionando correctamente, por tanto, las pruebas a realizar son mínimas.

6.2.1 Pruebas de compatibilidad.

Para la realización de este tipo de pruebas, se ha tenido en cuenta los requisitos de las aplicaciones que están instaladas en la máquina virtual para la ejecución de los test. Hay que tener en cuenta que es un entorno hecho a medida, por lo que en un principio la instalación de las aplicaciones, junto con el sistema operativo, se consideran que son las más óptimas.

El Ciclo de Vida se ha comprobado que es compatible con los navegadores más comúnmente utilizados:

- Internet Explorer 7.0 o superior.
- Mozilla Firefox 3.0 o superior.

En cambio, para el montaje de la máquina virtual, tiene sus limitaciones. Logic Class no tiene problemas de compatibilidad ya que funciona en cualquier SO actual: Windows 2000, Windows 2003, Windows XP, Windows Vista, etc.

En cambio, la versión utilizada de TestPartner 5.6.0 no es compatible con Windows Vista. Cabe también tener en cuenta que son aplicaciones que funcionan en entornos Microsoft, por lo que en otros SO como Linux no funcionan. Por tanto, los requisitos a nivel de SO son:

- ***Sistemas Operativos:*** Windows 2000, 2003 y Windows XP.

6.2.2 Pruebas de rendimiento.

Para la realización de estas pruebas, lo que se ha hecho es probar con casos prácticos para ver de qué manera responde la plataforma creada.

Mediante el siguiente gráfico se puede ver que el conjunto de las pruebas tiene un rendimiento lineal, es decir, cuantos más registros tiene que procesar, mayor es el tiempo que tarda en ejecutarse el test.

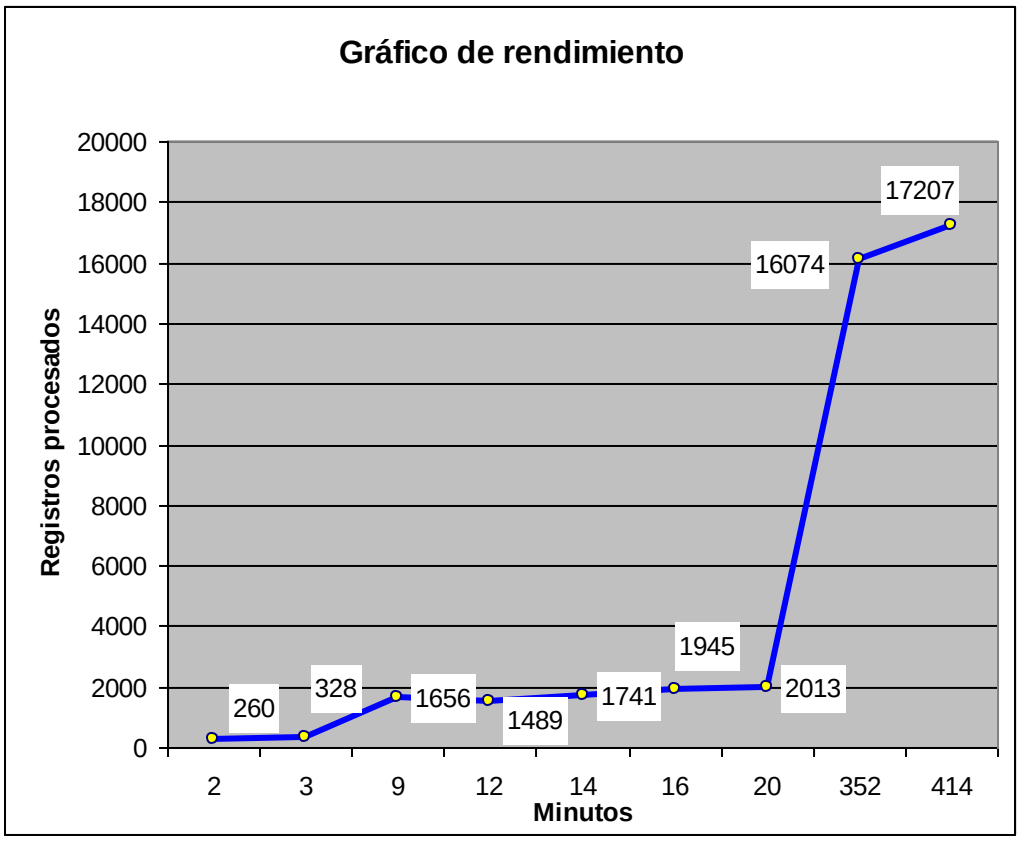

Figura 6.4 Gráfico rendimiento. Registros procesados / tiempo.

Los datos utilizados para la elaboración del gráfico han sido los siguientes.

Tiempo proceso (minutos)	Nº registros procesados
2	260
3	328
9	1656
12	1489
14	1741
16	1945
20	2013
352	16074
414	17207

Todos estos datos han sido obtenidos mediante pruebas de diferentes casos de test para probar su correcto funcionamiento.

Como se puede observar, el rendimiento de la aplicación es un poco lento ya que para procesar unos 17000 registros, cantidad que se considera como un caso normal, tarda casi 6 horas. De todas formas, el propósito es programar la ejecución de los test una vez acabada la jornada laboral.

Si durante la ejecución de las pruebas se demora en unas pocas horas, se le resta importancia. Mientras estén listos para la mañana del día siguiente, se pueden considerar válidos.

Una vez realizadas estas pruebas, se puede extraer como conclusión que tiene un margen de mejora. Se debería de intentar mejorar el tiempo de procesamiento de datos para que se pudiera agilizar aún más el proceso.

Capítulo 7.

Conclusión y resultados.

En el siguiente capítulo consiste en una vez desarrollado todo, el alumno hace una valoración final. Los apartados son

- ***Consecución de objetivos*.** Cuáles han sido los objetivos conseguidos.
- ***Futuras líneas de ampliación.*** Que se va a hacer en un futuro.
- ***Valoración final.*** Opinión personal del alumno.

7.1 CONSECUCIÓN DE OBJETIVOS.

Hablar si se han conseguido las los objetivos desde que se empezó a desarrollar este proyecto es un poco complejo.

En un principio, los objetivos eran bastante distintos respecto a lo que finalmente se ha desarrollado. La idea inicial que tenia Sage, era crear una plataforma única para todas las sedes que se tiene en España y para sus diferentes aplicaciones. Por tanto no era más que la integración de un proyecto pequeño dentro de uno mayor. Finalmente lo que se decide es implementar la automatización solo de *Logic Class*, por lo que se convierte en uno único y, por tanto, la responsabilidad completa recae sobre el alumno.

Si se compara la idea inicial, que era la de globalizar la automatización de testing para las diferentes sedes de España, con lo que se ha desarrollado al final, se puede observar que las expectativas iniciales no se han cumplido ni mucho menos.

En cambio, sí que se puede decir que se han cumplido los objetivos que, una vez iniciado el proyecto, se han propuesto. Se descarta la automatización a gran escala pero, aprovechando los conocimientos adquiridos por el alumno, lo que se hace finalmente es ampliar las funcionalidades de una herramienta que Sage considera de gran utilidad.

7.2 FUTURAS LINEAS DE AMPLIACIÓN.

En un futuro, la idea inicial de hacer una automatización a gran escala dentro de Sage, está entre uno de los objetivos de la empresa. La gran diferencia, es que las herramientas que se pretenden utilizar, que aún no están decididas, son bastante diferentes a las utilizadas hasta la fecha por tanto, no tendrá nada que ver con lo que hay implementado a día de hoy.

7.4 VALORACIÓN PERSONAL DE LA EXPERIENCIA.

Esta experiencia me ha servido para poner en práctica toda la base adquirida durante la carrera. Sin esos conocimientos me hubiera sido imposible llegar hasta donde he llegado.

Es algo que recomiendo ya que considero que me ha marcado mucho por 2 motivos. El primero, es conocer el mundo de la informática empresarial, es decir, conocer la metodología de trabajo que tiene esta empresa y las técnicas que utiliza para el desarrollo de sus aplicaciones. También cabe destacar que he adquirido nuevos conocimientos sobre lenguajes de programación y diferentes aplicaciones que desconocía.

El segundo, y creo que el más importante, es la satisfacción personal de poder haber realizado una tarea de este tipo de forma autónoma que, en un principio, veía imposible realizar. Pese a las dificultades encontradas en el transcurso de los días, te das cuenta que mientras no se pierda la ilusión y las ganas de trabajar, toda tarea es posible de realizarse.

Para acabar, quería agradecer a la empresa *Sage Logic Control*, poder haber contado conmigo para el desarrollo de mi proyecto, así como al *Departamento de I+D y de Calidad* por toda la ayuda que me han prestado ya que sin ellos no hubiera sido posible.

BIBLIOGRAFÍA.

- Arias, José Antonio "*Análisis técnico automatización Testing".* Departamento I+D, Sage Logic Control, 2 Febrero 2009.
- Till Wagner. "*Automatización de TestCases*". Departamento Calidad, Sage Logic Control. 6 Febrero 2009.
- Simon Robinson, Jay Glynn, Burton Harvey, Craig Moqueen, Jerod Moemeka, Christian Ángel, Mogan Skinner, Karli Watson. *"Professional C#".* Third Edition, Ed Wrox Press Ltd, 2001.
- Robert Vieira. "*Professional SQL Server 2005 Programming".* Wiley Publising, 2007.
- Jérôme Gabillaud. "*SQL 2005, SQL, Transact SQL".* Ediciones Eni, Mayo 2006.
- Jorge Serrano Perez. *"Visual Basic 2005. Manual Avanzado".* Ed. Anaya Multimedia, 2006.

FUENTES ELECTRÓNICAS.

- Wikipedia. "Arquitectura de software" (22 Agosto 2009).[online]. Disponible en: http://es.wikipedia.org/wiki/Arquitectura_de_software (Agosto 2009)
- C# Station. (Copyright 2000-2009) "*C# Tutorial*"[online]. Disponible en: http://www.csharp-station.com/Tutorial.aspx (Mayo, 2009).
- Wikipedia. "Caso de uso" (24 Agosto 2009).[online]. Disponible en: http://es.wikipedia.org/wiki/Caso_de_uso (Agosto 2009).
- Wikipedia. "Diagrama de secuencia" (30 Junio 2009).[online]. Disponible en: http://es.wikipedia.org/wiki/Diagrama_de_secuencia (Agosto 2009)
- Realltech (Copyright 2001). *"Funciones en sql Server 2000".* [online]. Disponible en: http://www.sqlmax.com/func1.asp (Abril, 2009).

- Claudio Casares (7/09/2004). "*Introducción a SQL*"[online]. Disponible en: http://www.maestrosdelweb.com/editorial/tutsql1/ (Abril, 2009).
- Canal Visual Basic .Net "*Manuales .Net: Manual C#*". (Copyright 2009([online] Disponible en: http://www.canalvisualbasic.net/manual-net/c-sharp/ (Junio, 2009)
- Luis Artola (30 Julio 2009).*"Metodologías ágiles".* [online]. Disponible en: http://www.programania.net/category/desarrollo-agil/ (Abril, 2009)
- Wikipedia. "Programación por capas" (6 Septiembre 2009).[online]. Disponible en: http://es.wikipedia.org/wiki/Arquitectura_de_tres_niveles (Septiembre 2009).
- Wikipedia. "Pruebas de software" (31 Agosto 2009).[online]. Disponible en: http://es.wikipedia.org/wiki/Pruebas_de_software (Agosto 2009)
- Wikipedia. "Pruebas de regresión" (3 Agosto 2009).[online]. Disponible en: http://es.wikipedia.org/wiki/Pruebas_de_regresi%C3%B3n (Agosto 2009)
- Sage Logic Control.(Copyright 2009).[online]. Disponible en: http://www.sagelogiccontrol.com/websage/index.aspx (Septiembre, 2009)
- Alex Solano (26/09/2002) "*SQL Server 2000*" [online]. Disponible en: http://www.netveloper.com/contenido2.aspx?IDC=64_0 (Abril, 2009).
- TestPartner MicroFocus. (Copyright 2001-2009).[online].Disponible en: http://www.microfocus.com/products/TestPartner/index.asp
- ConnectionStrings, "*The connection String Reference*" (Copyright 2009). [online]. Disponible en: http://www.connectionstrings.com (Mayo, 2009)
- El Guille. "*Trucos y rutinas para Visual Basic 2ª parte*". (31 Agosto 1997) [online]. Disponible en: http://www.elguille.info/vb/VB_TIP2.HTM#vb2_005 (Mayo, 2009).
- Freetutes.com (Copyright 2007-2008) "Visual Basic 6 (VB6) - Beginners Tutorial" [online]. Disponible en: http://visualbasic.freetutes.com/learn-vb6/ (Abril,2009).

ANEXO.

En el cd, se incluyen las siguientes carpetas.

- **Código fuente:** Donde hay 2 subcarpetas.
    - **Fuente txt:** Está incluido el código fuente en formato txt.
    - **Fuente TP:** TestPartner solo permite visualizar su código a través de la importación de archivos xml. Por tanto, en esta carpeta se incluye una versión de evaluación de TestPartner, un archivo xml (NóminaClass) ,las instrucciones para poder importar y visualizar el código, así como un User's Guide del propio TestPartner.

El código fuente correspondiente al Ciclo de Vida, así como la Base de Datos, es propiedad de SAGE, por lo que no permite la inclusión como anexo.

- **Memoria:** Se incluyen los documentos correspondientes a la elaboración de la memoria en formato pdf. A través del índice se puede acceder a los diferentes capítulos de los que se compone.