Universitat Autònoma
de Barcelona

GESTOR DE RECURSOS PARA INGENIEROS DE PRODUCTO DE I+D

Memoria del proyecto
de Ingeniería Técnica en
Informática de Gestión
realizado por
Diego Cano Nieto
y dirigido por
Marta Prim i Sabrià

Escola d'Enginyeria

Sabadell, Septiembre de 2010

La abajo firmante, Marta Prim i Sabrià,

profesora de la Escola d'Enginyeria de la UAB,

CERTIFICA:

Que el trabajo al que corresponde la presente memoria ha sido realizado bajo su dirección
por Diego Cano Nieto

Y para que conste firma la presente.

Sabadell, Septiembre de 2010

MARTA PRIM SABRIA

Firmado digitalmente por MARTA PRIM SABRIA
Nombre de reconocimiento (DN): c=ES, ou=Vegeu https://www.catcert.cat/veridCAT (c) 03, ou=Serveis Publics de Certificacio CPIXSA-2, sn=PRIM SABRIA, givenName=MARTA, serialNumber=40518823E, cn=MARTA PRIM SABRIA
Fecha: 2010.09.15 15:30:51 +02'00'

Firmado: Marta Prim i Sabrià

RESUMEN

El siguiente proyecto presenta todo el desarrollo de una aplicación que nace de una necesidad en un departamento de I+D localizado en el sector del automóvil, uno de los más exigentes del mundo que requiere la optimización de todos los recursos que intervienen hasta conseguir un producto altamente competitivo.

Esta necesidad es una herramienta capaz de gestionar validaciones de productos en sus fases de desarrollo. Estos productos son motores eléctricos, que deben cumplir unas especificaciones técnicas y unas exigencias de calidad muy severas. Para ello se diseñan prototipos, que posteriormente deben ser sometidos a ensayos de condiciones reales en cámaras. Un buen desarrollo de un producto pasa por una buena comunicación entre todas las unidades implicadas, es ahí donde interviene nuestro gestor, proveyendo información actualizada en todo momento.

Es importante destacar que, al mismo tiempo que nuestra herramienta gestiona el día a día, se va generando un histórico de información muy valiosa que puede ahorrar grandes cantidades de dinero. Actualmente, esa información no está bien cohesionada y produce muchas inconsistencias.

Para mejorar todo ello, hemos diseñado una aplicación basada en web y dirigida al principal implicado en el desarrollo del producto, el ingeniero, sin olvidar que otros actores forman parte y deben participar de forma pasiva, pero directa

CAPÍTULO 1

DESCRIPCIÓN DEL ESCENARIO

1.1. Introducción

Con este primer apartado buscamos ubicar y exponer el entorno donde vamos a implementar nuestro proyecto. Consideramos importante este punto porque vamos a dar a conocer la terminología específica del campo de trabajo, la organización que rige, la presentación de los actores y el papel que juegan los mismos.

1.2. Descripción general

La aplicación que vamos a desarrollar debemos encajarla en una empresa multinacional de componentes de automóvil con planta en Santa Perpètua de la Mogoda. Concretamente, la empresa fabrica, vende y desarrolla motores eléctricos de refrigeración de motores para turismos, siendo el departamento de I+D donde encontramos nuestro cometido.

El departamento de I+D se encarga de la parte técnica de los motores. Dentro de sus funciones encontramos tres: P0, P1 y P2. Los P0, son los encargados de las productividades, desarrollar mejoras en un proyecto actualmente en producción, así como de bajar el coste del producto aumentando su rendimiento. Los P1 nacen de los nuevos proyectos que plantean los clientes con sus nuevos requerimientos. Y los P2 son ideas más futuristas de cara a posicionarse en un mercado conociendo las tendencias del mismo; más encarado a un departamento de I+D+i.

El activo humano directo con el que cuenta I+D es: un director del departamento, 8 ingenieros de producto, un experto en motores, un responsable de laboratorio, 3 técnicos de laboratorio, un responsable de prototipos y 2 técnicos de prototipos.

1.3. Actores y localización en el organigrama.

1.3.1. Sinopsis de la empresa.

Desde un punto de vista más general, la empresa cuenta con 5 departamentos, producción y logística, recursos humanos, compras, línea de producto e investigación y desarrollo. El departamento de línea de producto es un departamento transversal, propio de una organización matricial, como se muestra en la Ilustración 1.

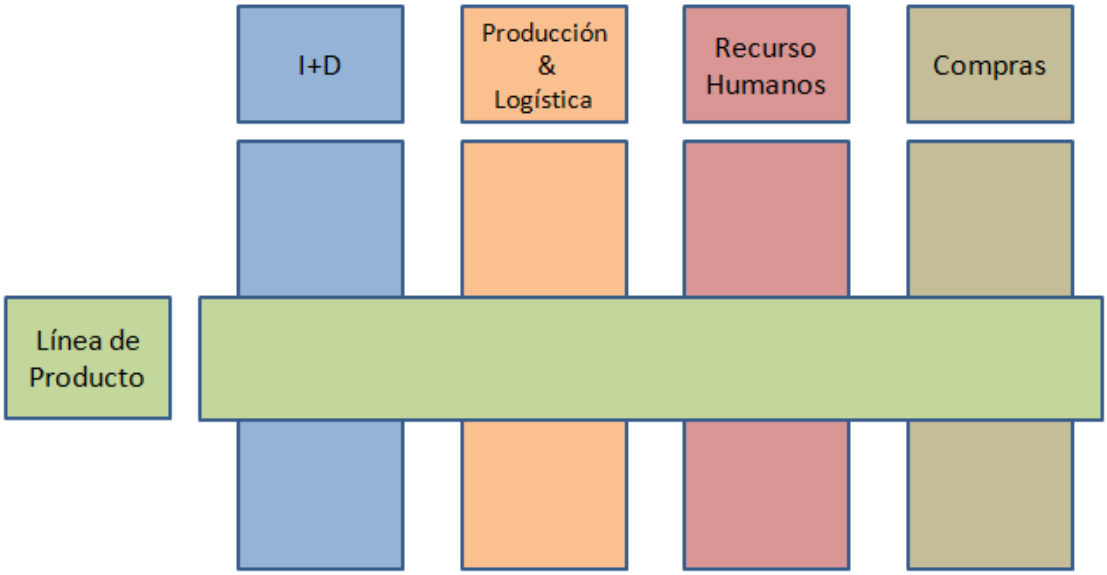


Ilustración 1. Organigrama general de la empresa.

En el apartado anterior (Descripción general) hemos introducido los principales miembros del departamento de I+D, en la Ilustración 2 vemos como está estructurado.

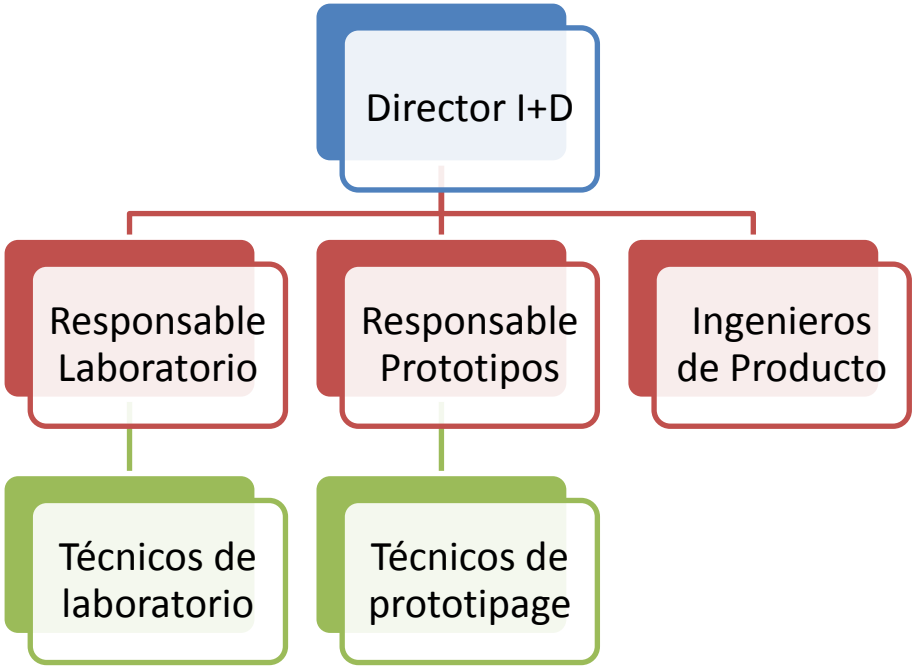


Ilustración 2. Estructura del departamento de I+D

1.3.2. Miembros del equipo de trabajo

Cada equipo de trabajo está implicado en uno o más proyectos. Para cada proyecto, se asignan equipos de trabajo formados por un jefe de proyecto, uno o varios ingenieros de producto, un ingeniero de calidad, un comprador y un vendedor.

El **jefe de proyecto**, es el máximo responsable de todo el proyecto, desde su inicio hasta que llega a producción en serie. Pertenece al departamento de línea de producto.

El **ingeniero de producto**, trabaja en las fases de diseño y validación del motor. Es el encargado de desarrollar un producto acorde a unos requisitos técnicos, sometiendo el producto a distintas pruebas pactadas con el cliente final. Para ello debe coordinar todo ese proceso junto con el responsable del laboratorio y el responsable de prototipos. Depende jerárquicamente del departamento de I+D y funcionalmente de línea de producto.

El **ingeniero de calidad**, controla durante todo el proceso de desarrollo que la calidad del producto cumpla los estándares de proceso y de componentes. Trabaja mucho con el cliente para atender sus peticiones de inmediato y aumentar su satisfacción. Depende jerárquicamente del departamento de calidad y funcionalmente de línea de producto.

El **comprador**, es el intermediario con el proveedor, pacta los precios de los componentes prototipos o de serie y gestiona las entregas en la fase de desarrollo. Pertenece al departamento de compras.

El **vendedor**, genera la propuesta del producto para concursar por el proyecto. Cotiza el precio del producto y es el intermediario principal con el cliente. Pertenece al departamento de línea de producto.

1.4. Motivaciones

Tras un periodo de trabajo como Ingeniero de producto en Nidec, he podido conocer perfectamente todo el proceso de diseño, desarrollo y validación de un motor. Además, he podido analizar todos los factores, conocer todos los flujos y valorar la forma de trabajo.

Teniendo en cuenta, la responsabilidad de cada proyecto y los medios a nuestra disposición para llevarlos a cabo, notaba una falta de eficiencia en dicho proceso. Veía que las herramientas de las que disponíamos no satisfacían las necesidades reales e impedían el óptimo rendimiento.

Por ello, decidí desarrollar una herramienta que permitiera trabajar mejor y con más recursos a un Ingeniero de producto, lo que no solo mejoraría su rendimiento particular sino que también mejoraría el de los proyectos en general.

1.5. Estructura de la memoria.

La memoria de este proyecto se estructura en 7 capítulos, organizados de la siguiente manera:

En el capitulo 1 encontramos la introducción al proyecto, donde detallamos la motivación personal para el desarrollo de esta aplicación y la descripción del escenario donde se pretende ubicar e instalar la herramienta de soporte.

Seguidamente, tenemos el capitulo 2 el cual contiene un estudio de viabilidad previo al inicio del proyecto, donde veremos la factibilidad del mismo además de algunos datos adicionales acerca de éste.

A continuación en el capitulo 3 exponemos el análisis de requerimientos, tanto a nivel funcional como no funcional y sus correspondientes casos de usos ordenados por actores.

En el capitulo 4 introducimos las tecnologías que hemos requerido para poder desarrollar el proyecto. También se introducen algunos nuevos conceptos de tipo teórico que tienen que ver con el diseño de la aplicación.

Posteriormente, encontramos el capitulo 5 donde exponemos todo el diseño del programa. Se matizan los conceptos teóricos introducidos en el capítulo anterior y se detallan algunos aspectos de la aplicación.

En el sexto capitulo, definimos las pruebas a las que hemos sometido nuestro programa para certificar su correcto funcionamiento y en que nos hemos basado.

Como séptimo y último capitulo, encontramos las conclusiones, donde valoramos tanto a nivel técnico como personal este proyecto, puntos a mejorar y futuras líneas de trabajo.

Finalmente, se puede encontrar toda la bibliografía utilizada como soporte para desarrollar este proyecto.

CAPÍTULO 2

ESTUDIO DE VIABILIDAD

2.1. Introducción

Este proyecto consiste en implementar una herramienta dirigida a un Ingeniero de Producto para proporcionarle soporte informatizado a la hora de desarrollar un motor eléctrico de refrigeración para motores de turismos.

2.2. Objeto

2.2.1. Descripción de la situación actual

Actualmente la relación entre el departamento de laboratorio, el departamento de prototipos y el departamento de Ingeniería sufre una falta de sincronización importante. El vínculo entre éstos está parcialmente informatizado, con el agravante del uso de diferentes plataformas. Además, no existe comunicación bidireccional entre ninguno de los departamentos y no hay enlace alguno entre laboratorio y el taller de prototipos.

Concretamente tenemos que entre ingeniería y laboratorio se trabaja con una base de datos compartida en la red corporativa en Access y, entre ingeniería y el taller de prototipos se utiliza Excel para gestionar los prototipos solicitados.

A continuación, se muestra un resumen visual de la situación. Ver Ilustración 3.

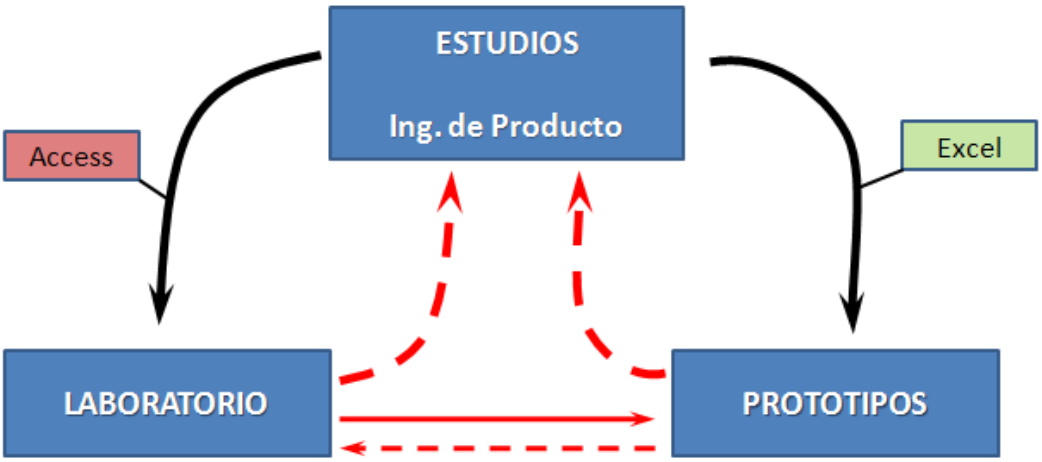


Ilustración 3. Situación actual de la comunicación entre departamentos.

2.2.2. Perfil del cliente – usuario

Podemos decir que como usuarios finales encontramos cuatro perfiles diferentes:

- **Ingeniero de Producto:** Podríamos definir a este usuario como principal. Su labor es generar las peticiones necesarias para el desarrollo del proyecto y que el flujo normal de la validación siga su curso sin la intervención de éste en dicho proceso. Simplemente, tiene que poder ver la información de estado de la situación actualizada.

- **Responsable de laboratorio:** Como representante del laboratorio y encargado de la gestión de dicho departamento; recibirá, programará y delegará en sus técnicos las peticiones que se vayan recibiendo en el sistema. Posteriormente deberá proporcionar la información correspondiente al término de su tarea.

- **Responsable de prototipos:** Recibe información detallada y desglosada del montaje del prototipo. Actualiza el estado de sus solicitudes.

- **Director de Proyectos:** Tiene la capacidad de gestionar a los usuarios, asignarles un rol y ubicarlos en los equipos formados para cada proyecto. Puede ver la situación del laboratorio y de prototipos. Además de gestionar los proyectos de los cuales es director y extraer información económica detallada.

2.2.3. Objetivos

El objetivo principal del proyecto es proporcionar una herramienta informatizada al Ingeniero de Producto que le permita hacer la validación del producto que se está desarrollando en su proyecto sin tener que sincronizar personalmente a los departamentos implicados en el flujo de trabajo.

Añadir que, son también objetivos del proyecto y a su vez funcionalidades de la aplicación:

- Mostrar información actualizada del proceso de validación.
- Evitar la intervención presencial del Ingeniero de Producto en la sincronización de los departamentos implicados.
- Generar un histórico de información relevante para el departamento de I+D.

2.2.4. Fuentes de información.

Como fuentes de información directa al desarrollo de la aplicación contamos con el material proporcionado para la realización de las prácticas de las asignaturas de Redes (herramientas para el 'front-end' para aplicaciones web) y Bases de datos (herramientas para el trato y almacenamiento de información).

Por lo que se refiere a la información de estudio y análisis del proceso que va a ser implementado, contamos con personas de gran experiencia en los diferentes departamentos implicados y de gran nivel formativo.

Además, para este tipo de aplicaciones y teniendo en cuenta que todas las tecnologías utilizadas van a ser de código libre, contamos con el material más actualizado colgado en la red con el plus de que existe una gran formación abierta a todos los públicos que nos proporcionan las diferentes webs oficiales de cada herramienta.

2.3. Valor de negocio

De todos es bien sabido que como regla general una empresa no realiza una inversión si no prevé una recuperación del capital y un rédito posterior. Es por ello que debemos definir muy bien la variante económica de este proyecto y de entre todos los objetivos ya anteriormente descritos, debemos añadir la capacidad económica que puede generar nuestro gestor.

En un mercado, como es el del automóvil, donde la reducción de costes es llevada al limite, poder reducir 20 o 30 céntimos de euro en un motor de 7 euros/unidad puede suponer la diferencia entre ganar un proyecto de 100.000 unidades anuales o perderlo.

Si tenemos en cuenta que la validación del motor se prorratea en cada unidad de motor y ésta supone aproximadamente el 10% del coste unitario. Todo aquello que sea reducir el coste del diseño, validación y desarrollo de un motor supondrá una incidencia directa en la bajada el precio unitario.

¿Qué nos ofrece nuestra herramienta para ello? Pues 3 opciones muy interesantes. La primera es el valor del histórico que se genera con los años. La rotación de personal es muy rápida en entornos multinacionales, por ello poder consultar de forma ordenada y cohesionada la información del pasado puede resultar de gran utilidad, por ejemplo para decidir si un ensayo se repite o es suficiente con el historial para garantizar al cliente la factibilidad del mismo.

En segundo lugar, la función de análisis de coste que debe ofrecer el gestor a los jefes de proyecto, les permitirá conocer más precisamente el coste real de una validación. De esta manera, poder ajustar más el precio a futuras cotizaciones.

La tercera es que, contando que son ensayos que pueden estar funcionando durante 10.000 horas seguidas (más de un año), definir correctamente el producto que va a ser sometido a prueba (trazabilidad) es vital para garantizar el resultado final. Tener que repetir un ensayo por mala configuración supone grandes pérdidas para el proyecto. En este caso, nuestro gestor almacena toda la información detallada del producto que es ensayado en cada caso.

Todo lo anteriormente explicado, es más entendible si aclaramos que el coste de un ensayo puede ir desde los 2.000€ el más barato hasta los 30.000€ el más caro.

2.4. Sistema a realizar

2.4.1. Descripción

Digamos que el proyecto consta de tres partes a realizar:

1. Análisis de los procesos.
2. Desarrollo de la aplicación en cuestión.
3. Documentación y memoria del proyecto.

La primera parte de este proyecto consiste en estudiar todos los escenarios que se puedan dar, junto con todas sus variantes. Con ello se pretende plasmar un marco real de trabajo compuesto por todos los posibles flujos.

A continuación se generan toda una serie de sub-fases dentro de la segunda parte propias del desarrollo de aplicaciones, tales como implementación, programación y test, entre otras.

En cuanto a la última fase, debemos recopilar toda la información que hemos generado durante el proyecto y documentarlo correctamente.

2.4.2. Recursos

A continuación, se van a detallar una serie de recursos recomendados para el funcionamiento mínimo de la aplicación aunque no considerados imprescindibles. Es decir, los requisitos que van a ser mencionados son una garantía de buen funcionamiento pero no excluyen otros sistemas de inferiores prestaciones o equivalentes. Dichos recursos pueden ser mejorados considerablemente y ello jugaría a favor del rendimiento final de nuestra herramienta.

Hardware: Puesto que nuestro objetivo es desarrollar una aplicación basada en web, esto indiscutiblemente conlleva una entidad cliente y otra entidad servidor. A lo que se refiere al lado servidor, se recomienda utilizar un sistema de configuración estándar tal como 4Gb de memoria RAM, Intel Dual Core 2.1 GHz, disco duro de 500Gb, Tarjeta Ethernet de red. Por el lado del cliente, podemos utilizar cualquier sistema genérico de recursos mínimos tales como 1Gb de memoria RAM, Intel Atom 230 a 1,66Ghz, disco duro, tarjeta de red Ethernet.

Software: En el lado del servidor se recomienda un sistema operativo tipo servidor. Todas las plataformas actuales son perfectamente funcionales ya que Apache trabaja con todas ellas puesto que implementan protocolo HTTP/1.1. Por lo que se interpreta, que Apache será nuestro servidor HTTP recomendado. Además como gestor de base de datos vamos a utilizar MySQL, también será entonces un requisito necesario. Del lado cliente, encontramos algo similar, el principal requisito será un navegador web, se recomienda Firefox, puesto que está disponible en todas las plataformas de sistemas operativos.

Recursos Humanos: Para el desarrollo de la aplicación, encontraremos como figura principal a Diego Cano Nieto, estudiante de la Escola d'Enginyeria de la Universitat Autònoma de Barcelona encargado y responsable de todo el desarrollo del proyecto y de la viabilidad del mismo. Además, se cuenta con la supervisión y apoyo de la figura del tutor del proyecto, representada por la doctora Marta Prim i Sabrià. También contamos con la colaboración y soporto del personal docente vinculado a algún aspecto del proyecto.

2.4.3. Análisis de coste

Tabla 1. Análisis de trabajo requerido en horas

Id	Tarea	Trabajo desarrollador (h)	Trabajo supervisora proyecto
1	***Análisis de los procesos***	***80***	***5***
1.1	Análisis de requerimientos	40	2,5
1.2	Diseño	40	2,5
2	***Desarrollo de la aplicación***	***116***	***7***
2.1	Codificación	100	6
2.2	Implementación y test	16	1
3	***Documentación y memoria del proyecto***	***70***	***3***
3.1	Introducción del escenario	20	0,5
3.2	Estudio de viabilidad	20	0,5
3.3	Documentación de la aplicación	30	2
TRABAJO TOTAL (h)		**266**	**15**

Una vez realizado el análisis de trabajo en horas, debemos cuantificar un coste para el trabajo de los miembros del proyecto. Se ha decidido establecer una cantidad de 20 € por hora para el trabajo del desarrollador y de 75 € por hora para la aportación de la tutora. Obteniendo, finalmente, un coste total de 6.445 €.

Cabe destacar que el coste de este proyecto es orientativo y está establecido acorde a un supuesto externo al contexto actual de proyecto de final de carrera.

2.4.4. Evaluación de riesgos

Como pasa en un gran porcentaje de proyectos de consultoría informática, podemos encontrarnos con el hecho de que el proyecto se encuentre finalizado, funcionando, cien por cien operativo y posiblemente cumpla los objetivos, pero que el cliente o usuario final no vea cubierta sus expectativas. No tanto porque la aplicación no desempeña su función sino porque la expectación generada ha sido excesiva o alejada de la realidad.

El origen de estas expectativas pueden venir del lado de los desarrolladores o bien, del lado del cliente final. Es por ello, que se debe manejar con especial cautela este tema, dejando bien especificado cada punto y detalle del proyecto. Además, es muy importante vigilar las relaciones interpersonales durante el desarrollo del proyecto para evitar desilusiones finales.

2.4.5. Alternativas

Es importante remarcar que la finalidad de la aplicación que vamos a desarrollar es bastante personalizada, por este motivo, resulta difícil encontrar otros productos en el mercado que puedan ofrecer un servicio parecido al que queremos proporcionar.

Pese a la idea anterior, no debemos descuidar el análisis del mercado actual, ya que cabe la posibilidad de encontrar alguna aplicación que abarque un extenso campo dentro de la gestión de I+D y podamos encontrar entre uno de sus muchos servicios nuestra necesidad.

Tras la búsqueda de algún posible candidato para ofrecer el mismo servicio, nos encontramos principalmente con:

- La mayoría de gestores de este tipo están orientados a gestión documental.
- Son aplicaciones modulares y solo cubren una parte del conjunto que ofrecerá nuestra aplicación. Es decir, no relacionan departamento (laboratorio, prototipos e ingeniería).
- Son programas muy genéricos que no cubren la necesidad específica.
- Ofrecen funcionalidades que no serán útiles y por lo tanto se desaprovecha su potencial.

2.5. Planificación

La organización de etapas que vamos a llevar a cabo para este proyecto la vamos a basar en un modelo lineal-secuencial. La siguiente fase debe comenzar una vez finalizada la anterior. Ver Tabla 2 e Ilustración 4. El motivo de esta elección es debido a que al iniciar el proyecto ya conocemos todos los requisitos del proyecto y no esperamos encontrar cambios ni a nivel de requerimientos ni peticiones de cambio desde el usuario final.

Tabla 2. Planificación de las etapas del proyecto.

Id	Tarea	Duración	Fecha inicio	Fecha fin	Preceden
1	***Análisis de los procesos***	***45d***	***01/10/09***	***02/12/09***	
1.1	Análisis de requerimientos	25d	01/10/09	04/11/09	
1.2	Diseño	20d	05/11/09	02/12/09	1.1
2	***Desarrollo de la aplicación***	***115d***	***03/12/09***	***12/05/10***	***1***
2.1	Codificación	70d	03/12/09	10/03/10	
2.2	Implementación y test	45d	11/03/10	12/05/10	2.1
3	***Documentación y memoria del proyecto***	***213d***	***01/10/09***	***26/07/10***	
3.1	Introducción del escenario	7d	01/10/09	09/10/09	
3.2	Estudio de viabilidad	40d	01/10/09	25/11/09	
3.3	Documentación de la aplicación	53d	13/05/10	26/07/10	2, 3.1, 3.2

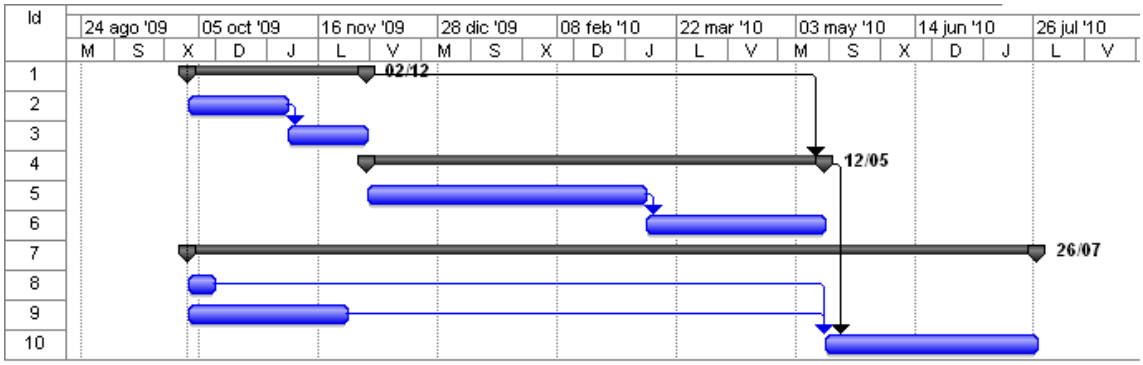


Ilustración 4. Diagrama de Gantt del proyecto.

Como se puede apreciar, ciertas tareas de la tercera fase han sido programadas para el inicio del proyecto. En este punto se rompe un poco el esquema típico de modelo lineal-secuencial. Esto es debido al tipo de tarea 3.1 y 3.2, que a pesar de que pertenecen al hito de documentación, su esencia nos permite irlas trabajando desde el inicio. De esta manera conseguimos cierta ventaja en el camino crítico del proyecto y tenemos un margen muy amplio para la finalización de estas dos tareas en concreto.

2.6. Conclusiones

Tras el estudio de viabilidad concluimos que la factibilidad del proyecto a desarrollar es ASUMIBLE.

No obstante, tendremos que tener presentes algunos puntos durante todo el desarrollo. Tras el análisis hemos detectado cuales son la fases, tareas y partes críticas del proyecto. Por este motivo, deberemos prestar especial atención y exigirnos el máximo de rigurosidad frente al cumplimiento estricto de dichos puntos críticos.

Además, hemos reflejado con claridad cuales van a ser nuestros objetivos principales, y cuales serán secundarios; lo que nos debe ayudar a mantener un rumbo claro y a no desviarnos de nuestro principal propósito.

CAPÍTULO 3

ESPECIFICACIÓN DE LOS REQUERIMIENTOS

3.1. Requerimientos no funcionales.

Vamos a hacer dos distinciones a la hora de especificar los requisitos no funcionales, los del lado del servidor y los del lado del cliente (usuario).

Por parte del servidor, se requiere tener instalado un servidor web con soporte para PHP5. Como vimos en el estudio de viabilidad, una propuesta puede ser Apache. Además, debe alojar una base de datos bajo MySQL.

Por lo que se refiere al cliente, éste tiene que tener previamente instalado un navegador web puesto que es la única forma de interacción con la aplicación. La aplicación garantiza su funcionamiento con todos los navegadores, pero se recomiendan aquellos que siguen los estándares marcados por el consorcio internacional W3C. Dicho explorador, debe tener permitido la ejecución de JavaScript.

Tanto servidor como cliente deben estar interconectados como mínimo en una red con permisos mínimos a la URL principal.

3.2. Seguridad

La seguridad tanto del terminal servidor como del terminal cliente es responsabilidad de la empresa que los aloje y deben estar protegidos por las reglas que rijan la intranet.

El acceso a la base de datos sólo estará permitido a través de la interfaz gráfica de la aplicación. Sólo se permite la modificación o eliminación de datos directamente sobre la base de datos al administrador de la misma, siempre bajo circunstancias puntuales como pudiera ser el mantenimiento.

3.3. Requerimientos funcionales.

Para nuestra aplicación vamos a definir un actor principal llamado Empleado, el cual tendrá definidas las actividades que pueden realizar todos los usuarios. A continuación encontramos cuatro actores más, que heredan funciones del actor principal (Empleado) y definen las suyas propias. Estos actores son Técnico de laboratorio (a partir de ahora TL), Ingeniero de producto (a partir de ahora IP), responsable de prototipos (a partir de ahora RP) y el jefe de Proyectos (a partir de ahora JP). Además, incorporamos un actor adicional denominado Responsable de Laboratorio (RL), extensión del actor TL que hereda todas sus funcionalidades.

3.3.1. Casos de uso de Empleados

3.3.1.1. Diagrama general

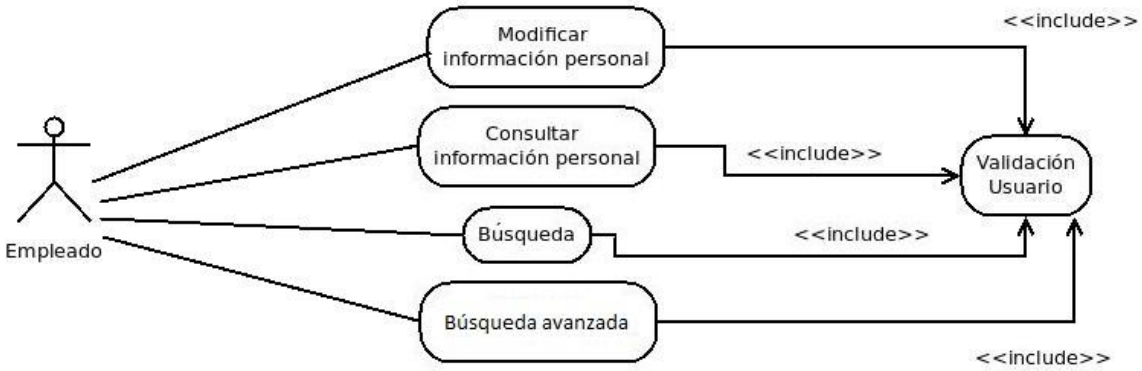


Ilustración 5. Diagrama UML para el actor Empleado

3.3.1.2. Consultar información personal

Nombre del caso de uso	Consultar información personal
Descripción	El usuario puede ver toda la información que define su perfil de empleado.

Disparadores	Seleccionar en el menú la opción de consultar perfil.
Precondición	El usuario se ha autentificado como tal.
Post condición	Se muestra por pantalla la información detallada.

3.3.1.3. Modificar información personal

Nombre del caso de uso	Modificar información personal
Descripción	Cada usuario puede modificar su propia información.
Disparadores	Seleccionar desde la vista de consulta de información personal la opción de modificar.
Precondición	El usuario se ha autentificado como tal.
Post condición	La información ha sido modificada.

3.3.1.4. Búsqueda

Nombre del caso de uso	Buscar

Descripción	Buscar información en la base de datos.
Disparadores	Rellenar con la palabra clave el formulario de búsqueda y clicar en el botón.
Precondición	El usuario se ha autentificado como tal.
Post condición	Si existen coincidencias, se muestran agrupadas por entidades. Si no existen coincidencias, se avisa al usuario por pantalla y se redirige a la página principal.

3.3.1.5. Búsqueda avanzada

Nombre del caso de uso	Búsqueda avanzada
Descripción	Se realiza una búsqueda mucho más detallada, permitiendo definir al usuario los criterios que le interesa encontrar.
Disparadores	Clicar el botón del formulario de búsqueda con el campo vacío.
Precondición	1. El usuario se ha autentificado como tal. 2. El campo del formulario está vacío.

Post condición	Si existen coincidencias, se muestran agrupadas por entidades. Si no existen coincidencias, se avisa al usuario por pantalla y se redirige a la página principal.

3.3.1.1. Validación de usuario

Nombre del caso de uso	Validación usuario
Descripción	Validar la identidad de la persona para que el sistema pueda identificarla y ofrecerle sus funcionalidades o restringirle otras.
Disparadores	Rellenar el formulario con el nombre de usuario y contraseña. Enviar la información para que sea validada mediante el botón de enviar.
Precondición	El usuario existe dado de alta en la base de datos.
Post condición	Si el nombre de usuario y contraseña son correctos, se permite la entrada al sistema y se carga la información pertinente para ese usuario. Si no son correctas las credenciales, se avisa al usuario y se le invita a volver a intentarlo.

3.3.2. Casos de uso del Técnico de Laboratorio

3.3.2.1. Diagrama general

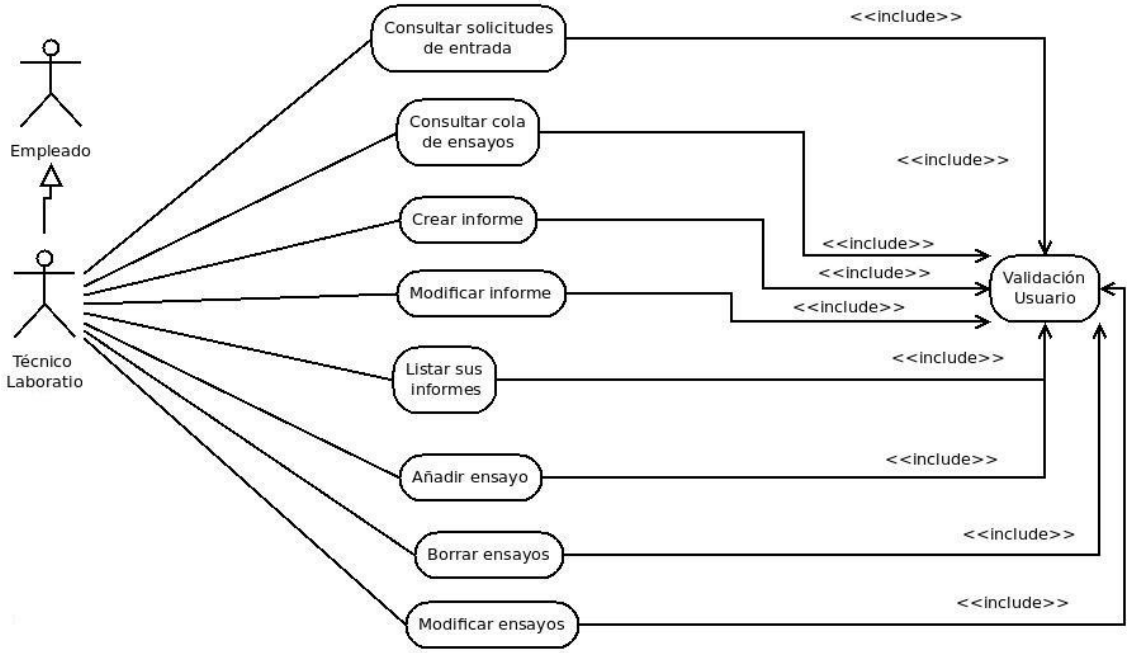


Ilustración 6. Diagrama UML para el actor Técnico de Laboratorio

3.3.2.2. Alta ensayos

Nombre del caso de uso	Alta ensayos.
Descripción	El usuario debe dar de alta todos los ensayos que su laboratorio puede realizar.
Disparadores	Seleccionar en el menú la opción de dar de alta un nuevo ensayo.

Precondición	El usuario se ha autentificado como tal.
Post condición	El ensayo ha sido introducido correctamente en la base de datos.

3.3.2.3. Baja ensayos

Nombre del caso de uso	Baja ensayos.
Descripción	El usuario debe dar de baja todos los ensayos que su laboratorio ya no puede realizar.
Disparadores	Seleccionar en el menú la opción de dar de baja un ensayo.
Precondición	1. El usuario se ha autentificado como tal. 2. El ensayo existe dado de alta en la base de datos. 3. El ensayo no debe estar vinculado a ninguna otra información, es decir, no se debe haber llevado a cabo nunca.
Post condición	El ensayo ha sido eliminado correctamente en la base de datos.

3.3.2.4. Modificación ensayos

Nombre del caso de uso	Modificación ensayos.

Descripción	El usuario puede modificar la información referente a todos los ensayos que su laboratorio puede realizar.
Disparadores	Seleccionar en el menú la opción de modificar un ensayo.
Precondición	El usuario se ha autentificado como tal. El ensayo existe en la base de datos.
Post condición	El ensayo ha sido modificado correctamente en la base de datos.

3.3.2.5. Crear informe

Nombre del caso de uso	Crear informe
Descripción	Introduce toda la información del ensayo que se ha realizado en el laboratorio.
Disparadores	Seleccionar en el menú la opción de crear informe.
Precondición	El usuario se ha autentificado como tal. Existe una solicitud de ensayo previa.
Post condición	Se ha vinculado un informe a una solicitud de ensayo y se da por cerrado el ensayo.

3.3.2.6. Consulta solicitudes de ensayos

Nombre del caso de uso	Consulta solicitudes ensayos
Descripción	Muestra los ensayos solicitados al usuario bajo los criterios de consulta seleccionados.
Disparadores	Seleccionar en el menú la opción de ver peticiones abiertas.
Precondición	El usuario se ha autentificado como tal.
Post condición	Se muestran por pantalla los ensayos.

3.3.3. Casos de uso del Responsable de Laboratorio

3.3.3.1. Diagrama general

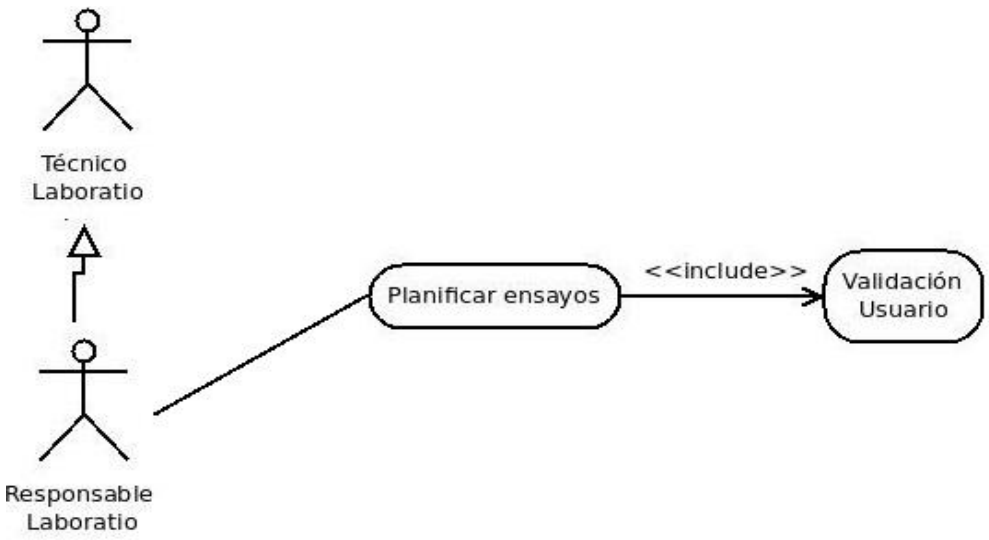


Ilustración 7. Diagrama UML para el actor Responsable de laboratorio

3.3.3.2. Planificación de ensayos

Nombre del caso de uso	Planificación de ensayos
Descripción	El RL asigna una fecha prevista para la realización del ensayo.
Disparadores	Seleccionar en el menú la opción de planificar o seleccionar directamente el ensayo desde la vista global.
Precondición	El usuario se ha autentificado como tal.
Post condición	Se muestran por pantalla la información actualizada del ensayo, incluyendo la fecha introducida. El estado de la solicitud se actualiza a “Planificado”.

3.3.4. Casos de uso del Responsables de Prototipos

3.3.4.1. Diagrama general

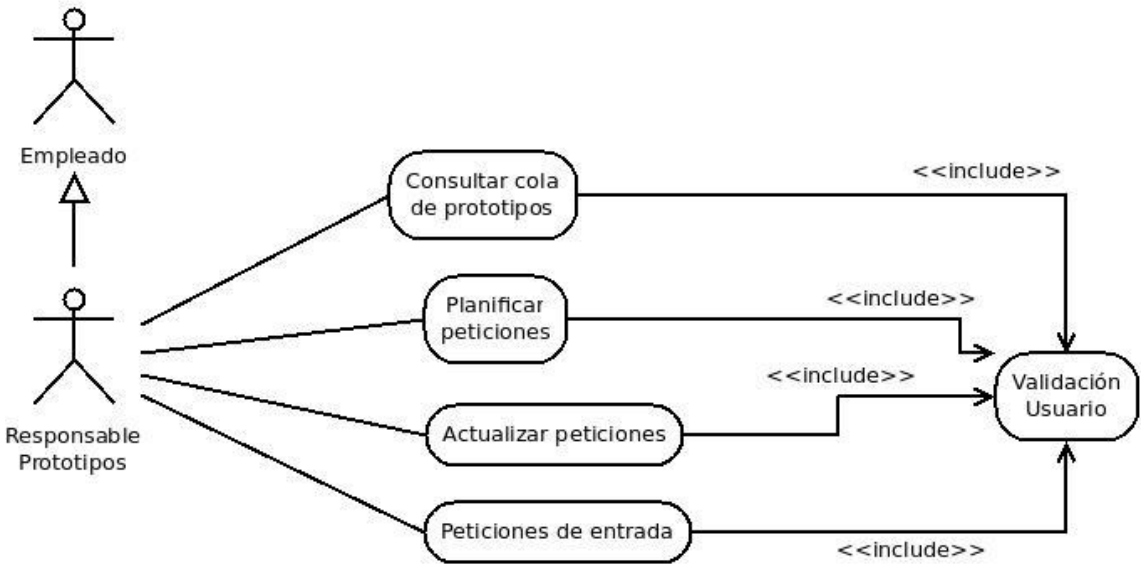


Ilustración 8. Diagrama UML para el actor Responsable de prototipos

3.3.4.2. Actualización estado peticiones

Nombre del caso de uso	Actualización estado peticiones
Descripción	Mantiene al día el estado de las peticiones que se han realizado a su departamento. Los estados son: En marcha (la petición se está realizando), Cerrada (la petición está terminada).
Disparadores	Seleccionar en el menú la opción de actualizar solicitudes.

Precondición	El usuario se ha autentificado como tal. Existe una entrada de solicitud previa.
Post condición	El estado de dicha solicitud de prototipo queda actualizado.

3.3.4.3. Consulta peticiones de prototipos

Nombre del caso de uso	Consulta petición de prototipos
Descripción	Muestra las peticiones de prototipos solicitadas al usuario bajo los criterios de consulta seleccionados.
Disparadores	Seleccionar en el menú la opción de ver peticiones abiertas.
Precondición	El usuario se ha autentificado como tal.
Post condición	Se muestran por pantalla las peticiones de prototipos.

3.3.4.1. Planificación de la petición

Nombre del caso de uso	Planificación de ensayos

Descripción	El RP asigna una fecha prevista para la construcción de los prototipos.
Disparadores	Seleccionar en el menú la opción de planificar o seleccionar directamente el ensayo desde la vista global.
Precondición	El usuario se ha autentificado como tal.
Post condición	Se muestran por pantalla la información actualizada del ensayo, incluyendo la fecha introducida. El estado de la petición se actualiza a "Planificado".

3.3.5. Casos de uso del Ingeniero de Producto

3.3.5.1. Diagrama general I

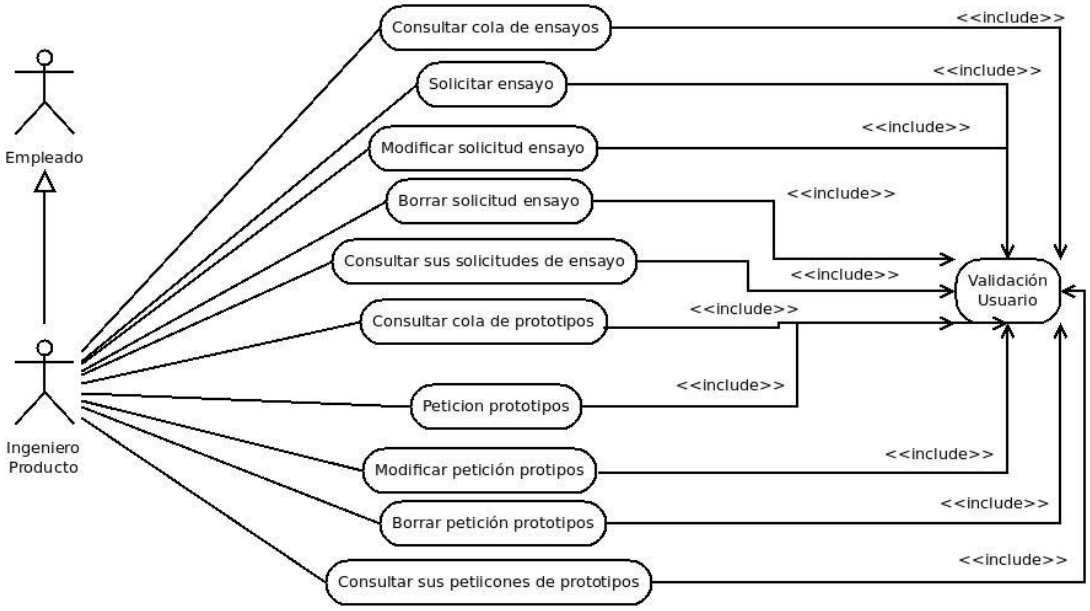


Ilustración 9. Primer diagrama UML para el actor ingeniero de producto

3.3.5.2. Diagrama general II

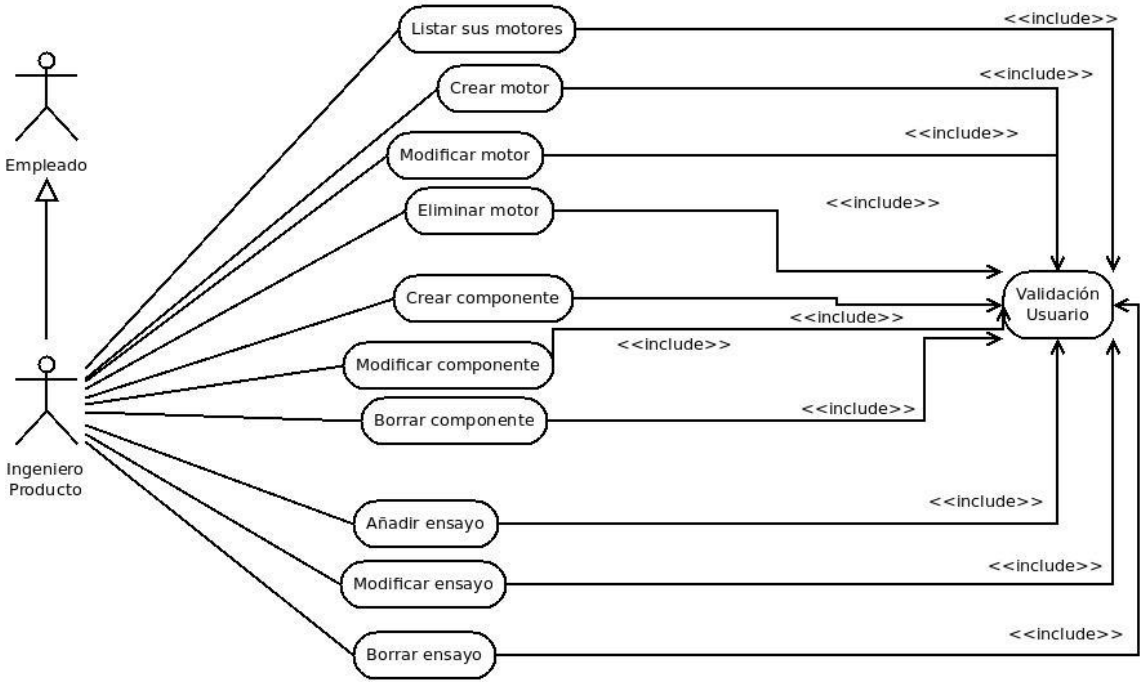


Ilustración 10. Segundo diagrama UML para el actor ingeniero de producto

3.3.5.3. Nueva solicitud de ensayos

Nombre del caso de uso	Nueva solicitud ensayo
Descripción	El usuario solicita realizar un ensayo al laboratorio. La petición va acompañada de información relevante sobre su elaboración.
Disparadores	Seleccionar en el menú la opción de solicitar ensayo.
Precondición	El usuario se ha autentificado como tal.

Post condición	La solicitud se ha almacenado correctamente en la base de datos.

3.3.5.4. Baja solicitud

Nombre del caso de uso	Baja solicitud
Descripción	El usuario da de baja una solicitud que ya no desea realizar.
Disparadores	Seleccionar en el menú la opción de dar de baja una solicitud.
Precondición	1. El usuario se ha autentificado como tal. 2. La solicitud existe dada de alta en la base de datos. 3. La solicitud no debe estar vinculada a ninguna otra información, como por ejemplo, tener un informe ya creado para esa solicitud.
Post condición	La solicitud de ensayo ha sido eliminada correctamente en la base de datos.

3.3.5.5. Modificar solicitud

Nombre del caso de uso	Modificar solicitud
Descripción	El usuario puede modificar la información de la solicitud de ensayo.

Disparadores	Seleccionar en el menú la opción de modificar solicitud.
Precondición	1. El usuario se ha autentificado como tal. 2. El ensayo existe en la base de datos.
Post condición	La solicitud ha sido modificada correctamente en la base de datos.

3.3.5.6. Consulta sus solicitudes de ensayo

Nombre del caso de uso	Consulta solicitudes de ensayo
Descripción	Muestra las peticiones de ensayos que el usuario ha solicitado bajo los criterios de consulta seleccionados.
Disparadores	Seleccionar en el menú la opción de ver solicitudes de ensayo.
Precondición	El usuario se ha autentificado como tal.
Post condición	Se muestran por pantalla las solicitudes de ensayo.

3.3.5.7. Nueva petición de prototipos.

Nombre del caso de uso	Nueva petición prototipos.
Descripción	El usuario encarga la realización de prototipos al taller. La petición va acompañada de información relevante sobre su elaboración.
Disparadores	Seleccionar en el menú la opción de petición de prototipos.
Precondición	El usuario se ha autentificado como tal.
Post condición	La solicitud se ha creado correctamente en la base de datos.

3.3.5.8. Baja petición de prototipos.

Nombre del caso de uso	Baja petición prototipos.
Descripción	El usuario da de baja una petición que ya no desea realizar.
Disparadores	Seleccionar en el menú la opción de dar de baja una petición.
Precondición	1. El usuario se ha autentificado como tal. 2. La solicitud existe dada de alta en la base de datos. 3. La solicitud no debe estar vinculada a ninguna

	otra información, como por ejemplo, tener un ensayo realizado con esos prototipos.
Post condición	La petición ha sido eliminada correctamente en la base de datos.

3.3.5.9. Modificar petición de prototipos.

Nombre del caso de uso	Modificar petición de prototipos
Descripción	El usuario puede modificar la información que describe la petición a producir.
Disparadores	Seleccionar en el menú la opción de modificar solicitud.
Precondición	1. El usuario se ha autentificado como tal. 2. La petición existe en la base de datos.
Post condición	La petición ha sido modificada correctamente en la base de datos.

3.3.5.10. Consulta peticiones de prototipos

Nombre del caso de uso	Consulta petición de prototipos
Descripción	Muestra las peticiones de prototipos solicitadas por el usuario bajo los criterios de consulta seleccionados.

Disparadores	Seleccionar en el menú la opción de ver peticiones.
Precondición	El usuario se ha autentificado como tal.
Post condición	Se muestran por pantalla las peticiones de prototipos.

3.3.6. Casos de uso del Jefe de Proyecto

3.3.6.1. Diagrama general

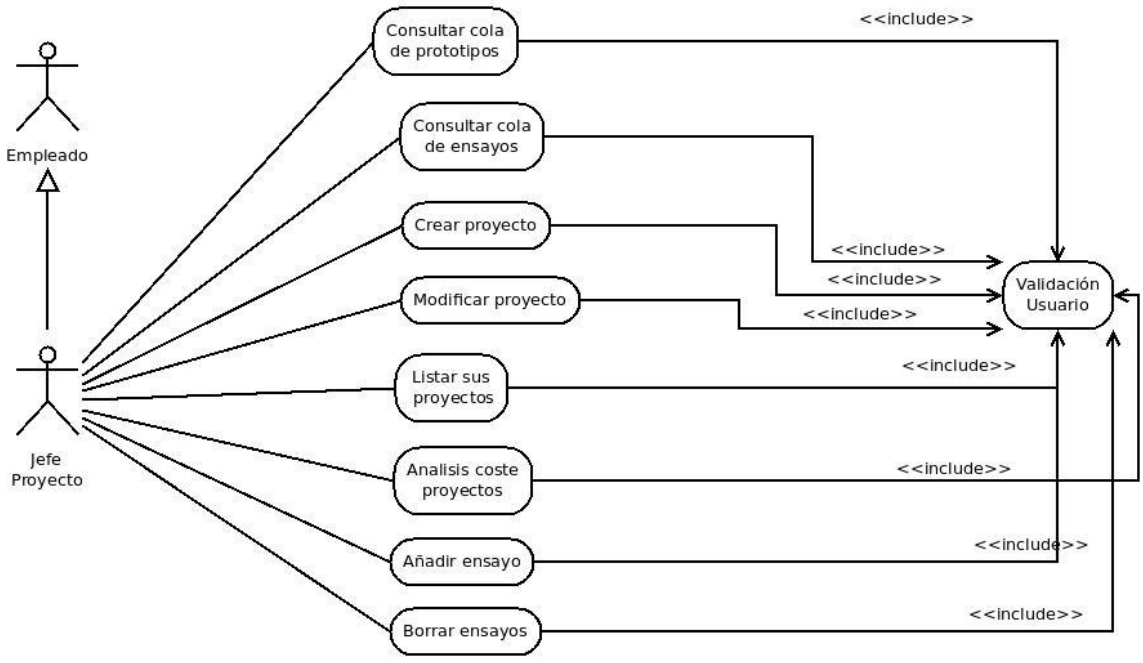


Ilustración 11. Diagrama UML para el actor jefe de proyecto

3.3.6.2. Consulta sus solicitudes de ensayo

Nombre del caso de uso	Consulta solicitudes de ensayo
Descripción	Muestra las peticiones de ensayos que el usuario ha solicitado bajo los criterios de consulta seleccionados.
Disparadores	Seleccionar en el menú la opción de ver solicitudes de ensayo.
Precondición	El usuario se ha autentificado como tal.
Post condición	Se muestran por pantalla las solicitudes de ensayo.

3.3.6.3. Consulta peticiones de prototipos

Nombre del caso de uso	Consulta petición de prototipos
Descripción	Muestra las peticiones de prototipos solicitadas por el usuario bajo los criterios de consulta seleccionados.
Disparadores	Seleccionar en el menú la opción de ver peticiones.
Precondición	El usuario se ha autentificado como tal.

Post condición	Se muestran por pantalla las peticiones de prototipos.

3.3.6.1. Desglose de coste de un proyecto

Nombre del caso de uso	Desglose coste proyecto
Descripción	Muestra una tabla con el detalle de toda la validación del proyecto. Agrupa coste total ensayos, coste total prototipos y coste total de la validación.
Disparadores	Seleccionar en el menú la opción de analizar coste.
Precondición	El usuario se ha autentificado como tal.
Post condición	Se muestran por pantalla el desglose del coste del proyecto.

CAPÍTULO 4

TECNOLOGÍAS EMPLEADAS

4.1. Introducción.

Para el desarrollo de esta aplicación hemos utilizado como tecnologías base las principales relacionadas con el mundo web. Del lado del servidor encontramos PHP como lenguaje de programación y del lado del cliente tenemos HTML, CSS, JavaScript y XML.

Al mismo tiempo, cabe matizar que hemos utilizado herramientas de trabajo, conocidas como *framework*[1] basadas en las tecnologías base. Para PHP, hemos elegido Zend Framework por ciertas razones que comentaremos más adelante y para JavaScript, hemos utilizado parcialmente jQuery y AJAX.

4.2. PHP y Zend Framework

4.2.1. ¿Qué es y porque Zend Framework?

Zend Framework (a partir de ahora ZF), como su propio nombre indica, es un *framework* de código abierto para desarrollar aplicaciones web bajo el lenguaje de programación PHP 5. Del mismo modo que PHP 5 está orientado a objetos y es código libre, ZF también lo es.

Principalmente ZF ha sido desarrollado por Zend Technologies. Esta empresa fue fundada por Andi Gutmans y Zeev Suraski, quienes junto con otros desarrolladores extendieron la versión inicial de PHP de su creador Rasmus Lerdof. El gran vínculo entre PHP y ZF es evidente, motivo adicional que lo hace un candidato más atractivo frente a otros competidores del mismo estilo. Cabe destacar, que al proyecto ZF se han sumado grandes empresas como Google y Microsoft para aportar diferentes interfaces que facilitan la interacción con sus propias aplicaciones.

El proyecto ZF cuenta también con un amplio soporte de documentación oficial de libre acceso y completamente gratuito. Además, al encontramos ante un proyecto muy extendido en el mundo de la web dinámica, podemos encontrar grandes aportaciones desde fuentes externas, no oficiales, pero sí de gran utilidad.

[1] Framework: estructura conceptual y tecnológica de soporte definida, que proporciona herramientas concretas, con base en la cual otro proyecto puede ser organizado y desarrollado.

Pese a que ZF puede ser usado simplemente como librería para manejar formularios, filtro de datos y validación, uno de los motivos más interesante por el que decantarse por ZF es por su característica más global, y es que trabaja bajo el patrón de diseño[2] Modelo, Vista, Controlador (MVC).

4.2.2. Modelo-Vista-Controlador (MVC)

MVC es perfecto para aplicaciones web ya que busca separar los datos de la aplicación (Modelo), la interfaz de usuario (Vista) y la lógica de control (Controlador), ver Ilustración 12, que traducido a entornos web los podemos distinguir como:

- Modelo: alberga todo aquello relacionado con los datos almacenados en la base de datos del programa, es decir, las sentencias SQL.
- Vista: interfaz de usuario definida con HTML y CSS. Elimina la necesidad de mezclar código PHP con *tags* de HTML.
- Controlador: Toda la lógica que controla y relaciona todas las partes de la aplicación. En definitiva, todo el lenguaje PHP.

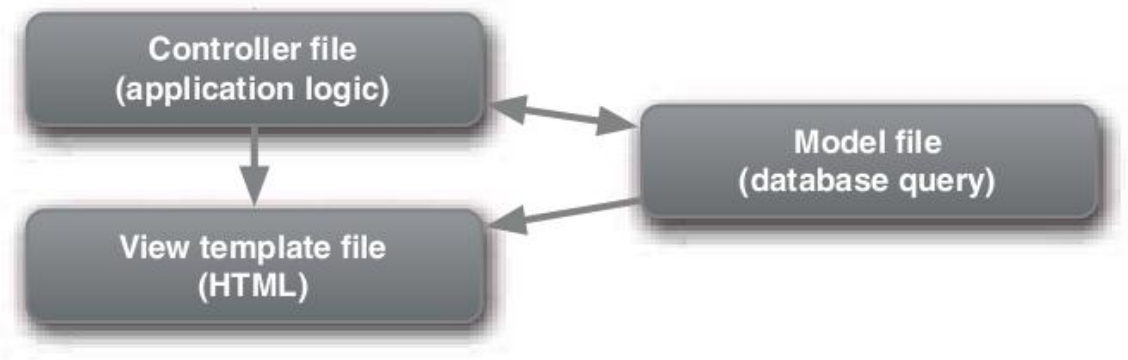


Ilustración 12. Patrón de diseño MVC.

Todo ello, nos extiende un marco de trabajo que proporciona elementos reusables y escalables en el diseño de sistemas de software, evita la reiteración en la búsqueda de soluciones a problemas ya conocidos y solucionados anteriormente y estandariza el modo en que se realiza el diseño. Eso facilita futuras ampliaciones, o simplemente, el mantenimiento del mismo.

[2] Patrón de diseño: conjunto estandarizado de conceptos y buenas prácticas para enfocar un tipo de problemática particular, que sirve como referencia para enfrentar y resolver nuevos problemas de índole similar.

4.3. JavaScript

4.3.1. AJAX

AJAX es el acrónimo de Asynchronous JavaScript And XML. El cliente mantiene una conexión abierta (asíncrona) en segundo plano con el servidor lo que permite transferir información (en XML) más precisa, lo que proporciona mayor velocidad. Una vez recibida ésta, el navegador la muestra mediante JavaScript.

Es importante matizar que no todo el contenido de la aplicación usa AJAX. Esto es debido a que nuestro software tiene como utilidad formar parte de una intranet, lugar donde no existe una gran deficiencia de velocidad.

No obstante, hemos considerado oportuno emplear esta tecnología ante situaciones es la que se valora muy positivamente la velocidad de respuesta de la aplicación. Como por ejemplo, cuando un campo de selección de lista depende de otro. En nuestro caso, al seleccionar una referencia de un motor, a continuación el usuario debe escoger un índice (nivel de ingeniería) para esa referencia. Es de gran utilidad que dicha información se cargue dinámicamente sin que el usuario deba esperar a la recarga de la página.

4.3.2. JQuery

JQuery también es considerado un *framework* basado en el lenguaje del lado del cliente JavaScript. A pesar de ello, en esta ocasión no hemos necesitado usar jQuery como *framework*, simplemente hemos utilizado sus librerías como complementos a nuestra aplicación para obtener soporte en menús, utilidades en fechas y algunos efectos visuales destacados para la interfaz gráfica.

4.3.3. ZF y jQuery

A pesar de que la última versión de ZF incluye librerías de soporte para algunos de los *widgets*[3] más destacados de jQuery, hemos preferido utilizar directamente las herramientas que jQuery proporciona para disponer de un abanico más amplio y no limitarnos a las que ZF ofrece.

[3] Widget: es una pequeña aplicación o programa, usualmente presentado en archivos o ficheros pequeños, que pretenden dar fácil acceso a funciones frecuentemente usadas y proveer de información visual.

4.3.4. Widgets empleados

A continuación, mostramos todas las utilidades junto con su contextualización de las herramientas que jQuery ha aportado a nuestro proyecto:

DatePicker

DatePicker es un calendario visual que permite estandarizar las fechas de forma visual de manera que el usuario escoge la fecha deseada sin problemas de formato (ver Ilustración 13). Tales problemas pueden venir dados, por ejemplo, entre el formato americano (mm/dd/aaaa) o el europeo (dd/mm/aaaa), pero sabiendo que ambos usan el calendario gregoriano, la estandarización pasa por forzar la elección desde un entorno visual. Todas las fechas que el usuario debe introducir, pasan por elegir la misma desde el calendario visual, no dejando que el usuario modifique el formato de ésta en ningún caso.

Ilustración 13. Calendario de tipo *Datepicker*.

Accordion

Esta herramienta esconde al ojo del usuario partes de la página web, permitiéndole elegir que contenidos desea mostrar y cuales no necesita en ese momento (ver Ilustración 14). Este efecto, se utiliza para la función de búsqueda avanzada, donde el usuario elige que desea buscar para mostrar los campos requeridos para ello, y repliega el resto para no confundir en caso de desplazamiento horizontal por la pantalla.

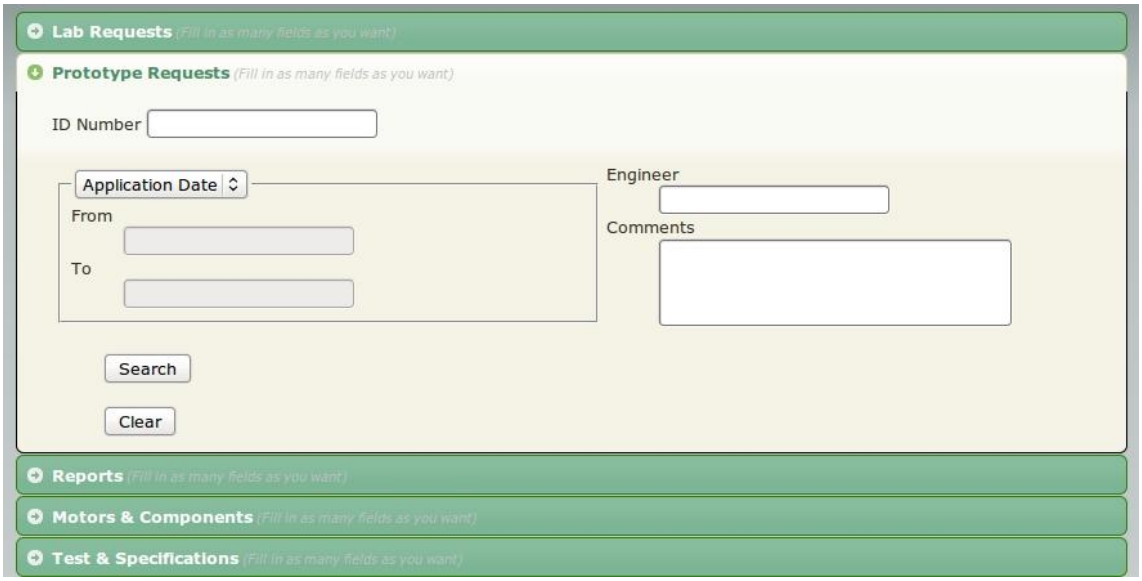


Ilustración 14. El efecto acordeón usado en la búsqueda avanzada.

Autocomplete

La herramienta autocompletar busca cambiar la forma en la que el usuario elige un valor determinado de entre una serie de registros estipulados. En otras palabras, pretende substituir a la lista desplegable ante casos de listas muy largas que hacen que éstas dejen de ser útiles en esos casos.

Autocompletar, tiene un motor de búsqueda interno que proporciona una lista de registros coincidentes con los primeros caracteres introducidos por el usuario (ver Ilustración 15). De este modo restringe los valores mostrados al usuario.

Cabe destacar que autocompletar es un útil de elección ante registros conocidos o semiconocidos, puesto que la lista completa no está disponible para el usuario. Este factor no es una adversidad para nuestra aplicación, todo lo contrario. Ha sido utilizado en puntos donde el usuario conoce o intuye bien el contenido, como son referencias y nombre de proyectos, ya que son valores con los que trabajan diariamente y maneja con agilidad.

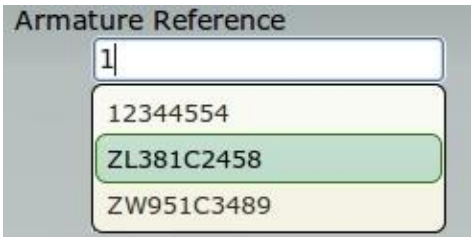


Ilustración 15. Ejemplo de autocompletar. Muestra valores que contienen “1”.

CAPÍTULO 5

DISEÑO

5.1. Introducción

Para modelar y diseñar la aplicación hemos utilizado UML (Unified Modeling Language). UML es un lenguaje visual de modelado orientado a objetos que unifica los métodos de OO de Booch, Jacobson y Rumbaugh. En definitiva, lo que busca UML entre otras muchas cosas es:

- Permitir especificar todas las decisiones del análisis, diseño e implementación, construyendo modelos precisos y nada ambiguos.
- Documentar todos los elementos de un proceso de desarrollo (requerimientos, arquitectura, pruebas, etcétera.)

5.2. Arquitectura de la aplicación

Como hemos comentado anteriormente, Zend Framework trabaja bajo el patrón Modelo-Vista-Controlador, por la cual cosa, hemos diseñado la aplicación entorno a este paradigma.

¿Y como se traduce MVC en nuestra aplicación? En nuestro caso, se definió tanto un modelo como un controlador por entidad. Por ejemplo, para las solicitudes teníamos un modelo (SolicitudesModel) y un controlador (SolicitudesController). Y cada entidad tenia definida una serie de vistas para cada acción. Por ejemplo, crear, modificar o borrar solicitud.

Además del propio diseño MVC que ofrece ZF, nos gustaría añadir un nuevo elemento al concepto. Se trataría del Modelo-Usuario, es decir los formularios. Algo que consideramos que se encuentra a medio camino entre la lógica de control y los datos de la aplicación (ver Ilustración 16). ¿Por qué decimos a medio camino? Pues porque tienen una lógica de control propia, tales como la validación de campos o el filtro de los datos; y además, su cometido es claramente el de proveer datos al modelo para que este luego se encargue.

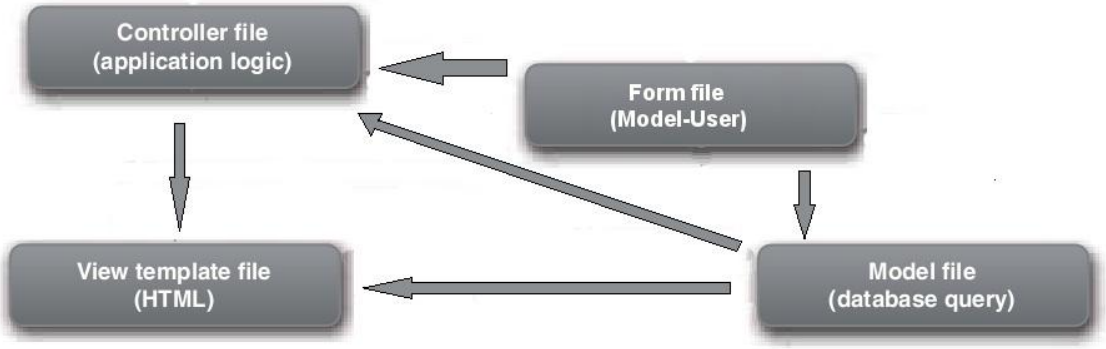


Ilustración 16. Patrón de diseño MVC con el concepto Modelo-Usuario añadido.

A continuación podemos ver la clase que define el formulario para seleccionar un informe que deberá ser modificado:

```
class InformeForm extends Zend_Form{

      public function __construct($options = null,$action){
            parent::__construct($options);
            $this->setName('informes');

            /* CREAMOS EL ELEMENTO DE ENTRADA DE DATOS */

            $informe = new
            ZendX_JQuery_Form_Element_AutoComplete('numero');
            $db = new Db_InformesModel();
            $indu_list = $db->getInformeAc
            (UserModel::getIdentity()->num_empleado);
            $val = new Zend_Validate_InArray($indu_list);
            $val->setMessage('Report "%value%" cannot be modified
            or it is not stored in the data base');
            $informe->setLabel('Report number :')
                  ->setRequired(true)
                  ->addFilter('StripTrim') /* Filtro 1[4] */
                  ->addValidator('NotEmpty',true)/*Validación 1[5]*/
                  ->addValidator($val)       /* Validación 2[6] */
                  ->setJQueryParams(array('source' =>
            $indu_list));

            /* CREAMOS EL BOTON DE ENVÍO DE DATOS */
```

[4] Filtro 1: Elimina los espacios en blanco del inicio y final de la cadena.

[5] Validación 1: Si el campo no es rellenado, el formulario no es válido y se marca el elemento como erróneo.

[6] Validación 2: Solo se pueden modificar ciertos informes permitidos, si el número de informe no está en la lista de válidos, el formulario advierte que el número de informe introducido no es correcto.

```
            $submit = new Zend_Form_Element_Submit('submit');
            $submit->setLabel('Select');

            /* CREAMOS EL BOTON DE RESET */

            $reset = new Zend_Form_Element_Reset('reset');
            $reset->setLabel('Clear');
            $this->addElements(array($informe,$submit,$reset));
      }
}
```

5.1. Diseño de la base de datos

5.1.1. Diagrama UML

A continuación se presenta el diagrama UML que representa el diseño de la base de datos (ver Ilustración 17)

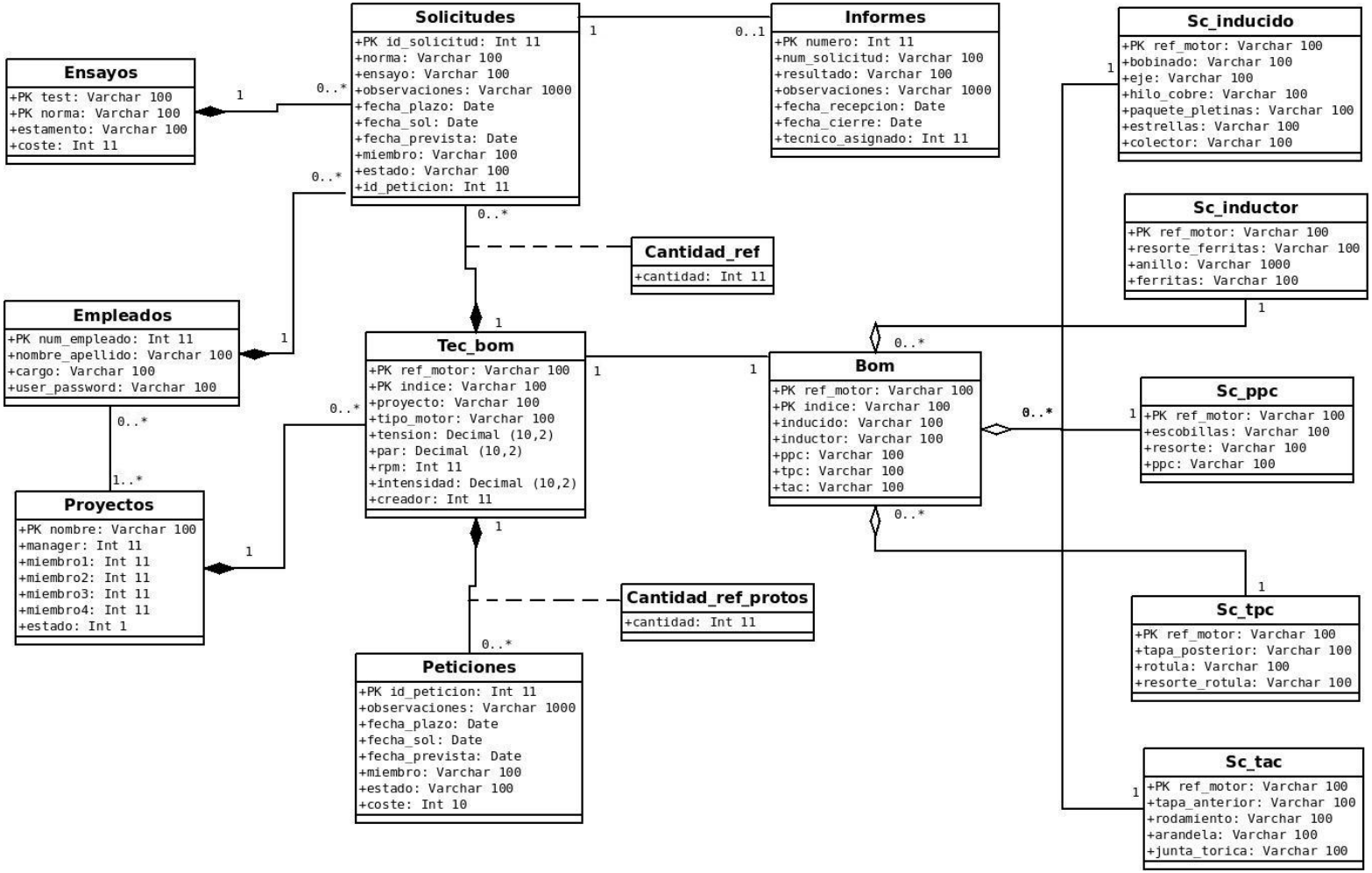


Ilustración 17. Diagrama UML para las base de datos.

5.1.2. Descripción de las tablas

A continuación vamos a describir de forma sintetizada cual es el contenido de cada tabla. Por orden alfabético, siguen de la siguiente manera:

Bom[7], contiene la relación de componentes que forman cada motor. Cada motor está definido por una referencia y un índice (nivel de ingeniería).

Cantidad_ref, guarda la relación de motores que se piden en una solicitud de ensayo y las cantidades pedidas para cada uno de ellos.

Cantidad_ref_protos, guarda la relación de motores que se piden en una petición de prototipos y las cantidades pedidas para cada uno de ellos.

Empleados, almacena información de cada uno de los empleados que tienen un rol dentro de la aplicación. Será importante para que el programa pueda identificar correctamente al usuario que intenta acceder a las funciones de la aplicación.

Ensayos, relación de ensayos que se pueden realizar en el laboratorio. Destacar el atributo coste, que establece el coste de realización del ensayo.

Informes, recoge la información del laboratorio después de haber realizado una solicitud.

Peticiones, guarda todas las peticiones de construcción de prototipos que se realizan al taller.

Proyectos, contiene información de cada proyecto, la más destacable es quien lo dirige y los ingenieros que lo integran.

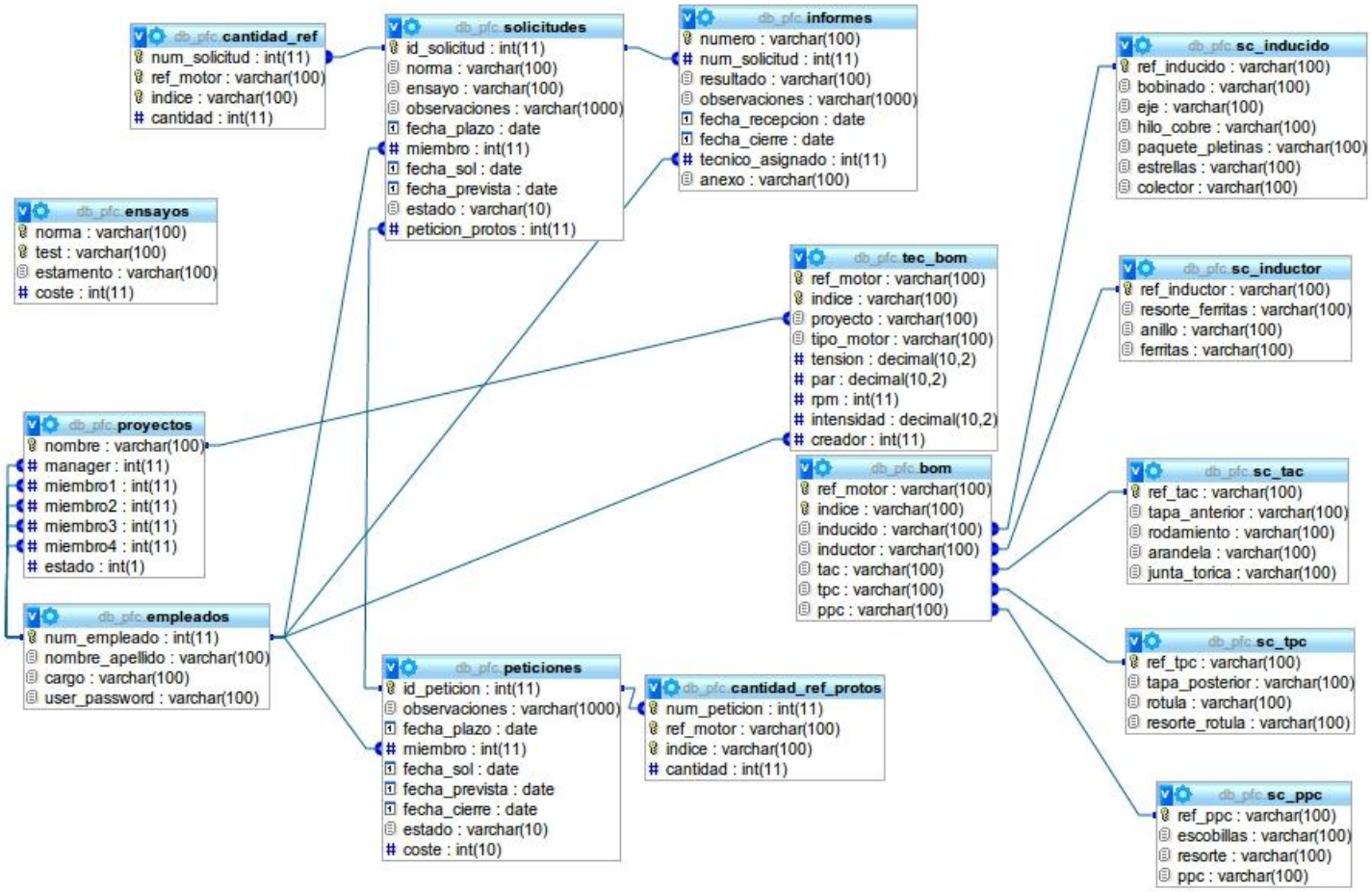
Sc_inducido, sc_inductor, sc_ppc, sc_tac, sc_tpc son todos los subconjuntos necesarios para formar un motor. Cada tabla recoge las referencias de componentes que forman un subconjunto.

Solicitudes, guarda todas las solicitudes de ensayos que realizan los ingenieros al laboratorio.

Tec_bom (BOM técnico), contiene la información más técnica que define cada referencia de motor e índice, además del proyecto al que pertenece.

[7] BOM: acrónimo del inglés *"Bill Of Material"* que significa lista de materiales. En el mundo industrial, es el desglose de componentes y subcomponentes que forman un producto.

5.1.3. Relación de tablas



5.2. Seguridad, autentificación y control de acceso.

A nivel de seguridad externa cabe remarcar que no se han desarrollado nuevas medidas. Como se ha comentado en capítulos anteriores, la protección de no intrusión por parte de ajenos a la aplicación viene dada por los propios mecanismos implementados en la red local de la empresa.

Sí es cierto, que hay que vigilar e impedir posibles alteraciones voluntarias o no voluntarias dentro de los propios usuarios de nuestro gestor. De forma que se ha creado un sistema de autentificación y control de acceso. Es importante resaltar el matiz que las separa. La autentificación es el proceso de identificación de un individuo en base a sus credenciales (en nuestro caso nombre de usuario y contraseña), mientras que el control de acceso es el proceso de decidir si el usuario está autorizado a hacer algo concreto (Ver Ilustración 18). Como es evidente el control de acceso requiere previamente una autentificación.

Ilustración 18. Gráfico explicativo del control de acceso.

De modo que lo que se busca en primer lugar es impedir la alteración de datos de terceros y el acceso a información restringida tanto a nivel interno, como adicionalmente en caso de intrusión externa. Y en segundo lugar, ofrecer al usuario mejores prestaciones enfocadas directamente a sus necesidades. Esto último es gracias a la autentificación, que le permite a la aplicación identificar al usuario para interaccionar contra un rol determinado.

En un primer nivel, las acciones que un actor puede realizar están plenamente definidas en los menús de navegación. Pero al tratarse de una aplicación web, todas las acciones tienen una URL definida. De modo que cualquiera conocedor de una URL concreta podría acceder a funciones no dirigidas a su usuario. Por ello todas las clases Controlador, tienen definidas listas de acciones permitidas o denegadas.

5.3. Interfaz gráfica

5.3.1. Visión global

Es importante empezar matizando que toda la interfaz gráfica ha sido realizada en inglés. El motivo es muy sencillo, como hemos comentado al inicio, el contexto donde iría implantada la aplicación es un entorno completamente multicultural (propio de una multinacional), donde se toma el inglés como lengua común.

El diseño de la interfaz grafica se ha hecho basándonos en una estructura de plantilla (*template*), marcadores de posición (*placeholders*) e inserción (ver Ilustración 19). Es decir, se ha definido una plantilla madre que aloja diferentes marcadores de posición, en los cuales se insertan vistas independientes, de la forma siguiente:

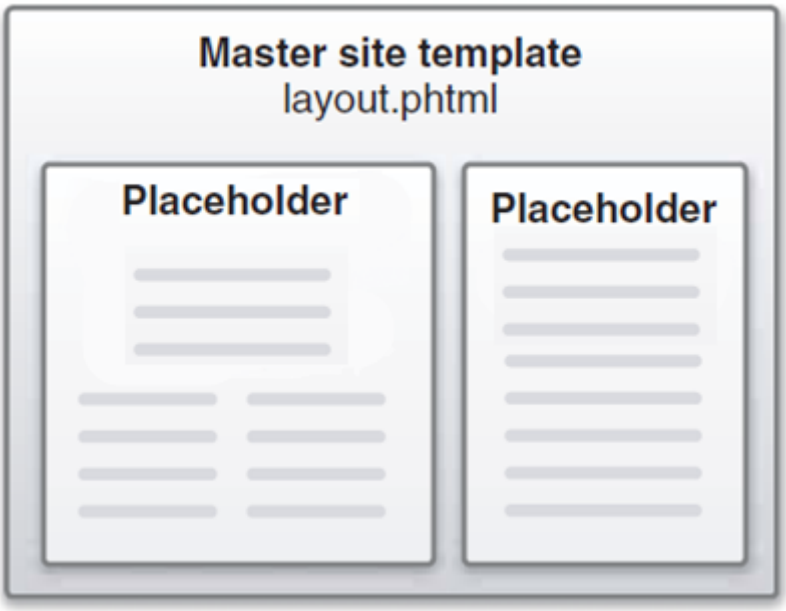


Ilustración 19. Esquema de la organización para el diseño de la interfaz gráfica.

De modo que dispones de 3 marcadores de posición definidos en nuestra plantilla, como son el menú, el contenido y el *login* como se muestra a continuación (ver Ilustración 20):

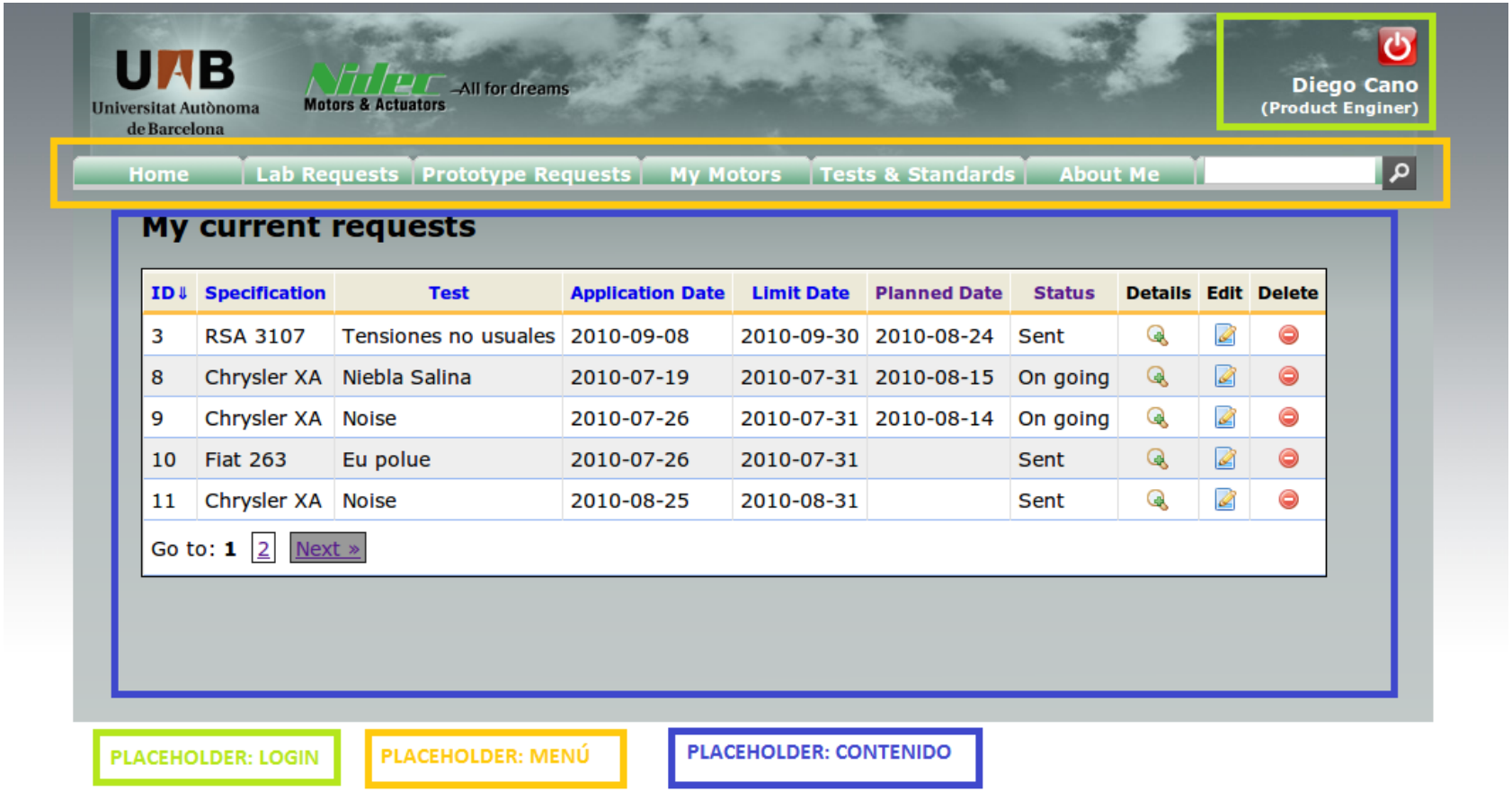


Ilustración 20. Los marcadores utilizados. En verde: login, en naranja: menú, en azul: contenido.

5.3.2. Menús

Los menús han sido definidos particularmente para cada actor, de manera que a priori solo puedan acceder a aquellas funciones que les han sido asignadas. Y decimos a priori, porque en caso de acceder a una URL no estipulada en el menú, el control de acceso es el que pasa a gestionar si el usuario tiene permisos suficientes (véase 5.2. Seguridad, autentificación y control de acceso.).

Todos los menús están diseñados bajo estructuras similares. Los usuarios se encontrarán ante menús horizontales basados en pestañas, algunas de ellas disponen de submenús desplegables.

En el extremo derecho, se ha habilitado un formulario de acceso a la búsqueda rápida. Simplemente hay que rellenar el campo y clicar sobre el icono de lupa para realizar una búsqueda rápida sobre las claves primarias de los elementos. Si se clica el icono lupa sin rellenar el campo, se redirige hacia la búsqueda avanzada.

La búsqueda avanzada dispone de menú propio. Es un menú de estilo acordeón que despliega o recoge las pestañas en función de la búsqueda que queremos realizar. Cada pestaña representa un elemento real (Solicitudes de ensayo, peticiones de prototipos, informes, motores y componentes, ensayos y normas)

Como mencionábamos anteriormente cada actor dispone de un menú propio. A continuación se muestran todos los menús desplegados (ver Ilustración 21, Ilustración 22, Ilustración 23, Ilustración 24, Ilustración 25):

Ilustración 21. Menú desplegado para un Ingeniero de Producto

Ilustración 22. Menú desplegado para el Responsable de Laboratorio

Ilustración 23. Menú desplegado para un Técnico de Laboratorio

Ilustración 24. Menú desplegado para el Jefe de Proyecto

Ilustración 25. Menú para el Responsable de Prototipos

5.3.3. Mensajes de interacción con el usuario

Nos gustaría destacar tres tipos de mensajes que interaccionan directamente con el usuario en nuestro programa. Puesto que la mensajería con el usuario no debe hacerse muy repetitiva, ya que ésta pierde eficacia (el usuario termina haciendo caso omiso sin apenas leer el contenido), hemos optado por utilizar esta técnica en los casos más destacables.

El primer caso de mensajes hacia el usuario son los errores en los formularios (ver Ilustración 26). Dicha validación es realizada por PHP. Cuando un campo no ha sido validado correctamente aparece un mensaje de error personalizado para cada caso advirtiendo de la anomalía. De modo que el formulario no sigue hacia delante, sin previa rectificación.

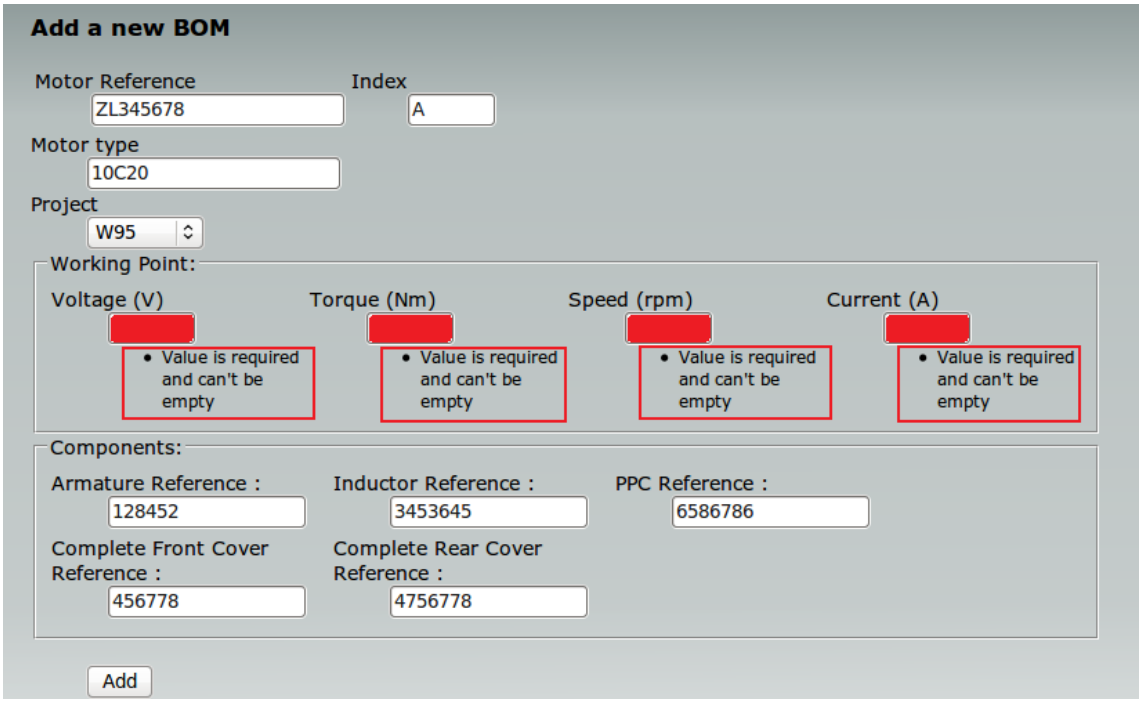


Ilustración 26. Mensajes de error en un formulario.

En segundo lugar, tenemos los mensajes de información al usuario. Estos aparecen al realizar (ver Ilustración 27) o al no realizar (ver Ilustración 28) alguna acción que estaba prevista ejecutar en ese instante de prioridad alta, tales como inserción, eliminación o modificación de datos en una base de datos.

Ilustración 27. Cuadro de dialogo de acción satisfactoria.

Ilustración 28. Cuadro de dialogo de acción fallida.

En tercer y último lugar, encontramos los mensajes de verificación (ver Ilustración 29). Estos son solo requeridos cuando el usuario elige una acción destructiva desde una tabla con múltiples entradas. Situación muy probable de confundir filas y elegir la opción no deseada.

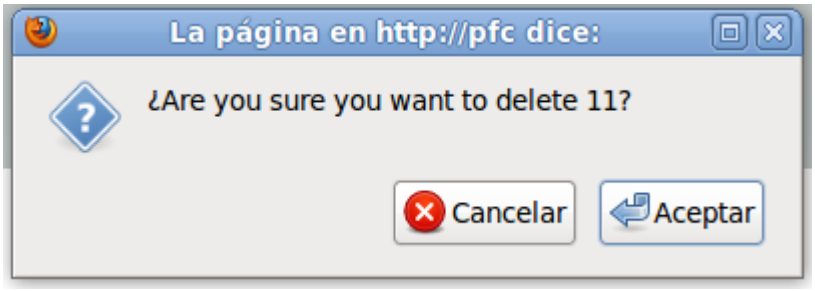


Ilustración 29. Cuadro de dialogo para confirmación del usuario.

Capitulo 6

PRUEBAS Y COMPATIBILIDAD

6.1. Casos de prueba basados en casos de uso

Para completar un diseño robusto de una aplicación software hay que aplicar distintas pruebas de análisis que acreditan su correcto funcionamiento. El objetivo principal de dichas pruebas es detectar posibles anomalías en la ejecución que no fueron detectadas o previstas en la fase de programación.

En nuestro caso hemos optado por realizar pruebas basas en casos de uso. Dichas pruebas se definen de la misma forma que un caso de uso, donde se especifican escenarios, flujos básicos y flujos alternativos. Entendemos por escenario, un camino concreto que va del estado inicial, pasando por el flujo básico y/o flujos alternativos. El flujo básico se define como el camino que sucede más frecuentemente o el más bueno. Y los alternativos son casos opcionales, excepcionales o fuera de la norma.

Cabe destacar que no todos los escenarios y flujos han podido ser testeados debido a su exponencial número de permutaciones posibles que se llegan a generar. Es por ello que se ha tratado de testear los casos más esenciales. A continuación se muestra un ejemplo de ello, donde el caso de estudio es la creación de un motor.

6.1.1. Caso de prueba: Creación de un motor

En este caso tenemos el flujo básico, 3 flujos alternativos y en consecuencia 6 escenarios posibles. Se especifican en la Tabla 3 y se complementa con la Ilustración 30.

Tabla 3. Especificación del caso de uso “creación de un motor”.

Flujo básico (1)	Se rellena el formulario con todos los datos deseados, se envían los datos y se almacenan correctamente en la base de datos. Se informa al usuario de acción satisfactorio (ver Ilustración 27).
Flujo alternativo (2)	La referencia del motor ya existe y se pide modificarla.

Flujo alternativo (3)	El formulario no es validado correctamente porque un campo falta o es erróneo.
Flujo alternativo (4)	Los datos no pueden ser almacenados en la base de datos. Se avisa al usuario del error (ver Ilustración 28).
Escenarios	1-2 1-2-3 1-2-3-4 1-3 1-3-4 1-4

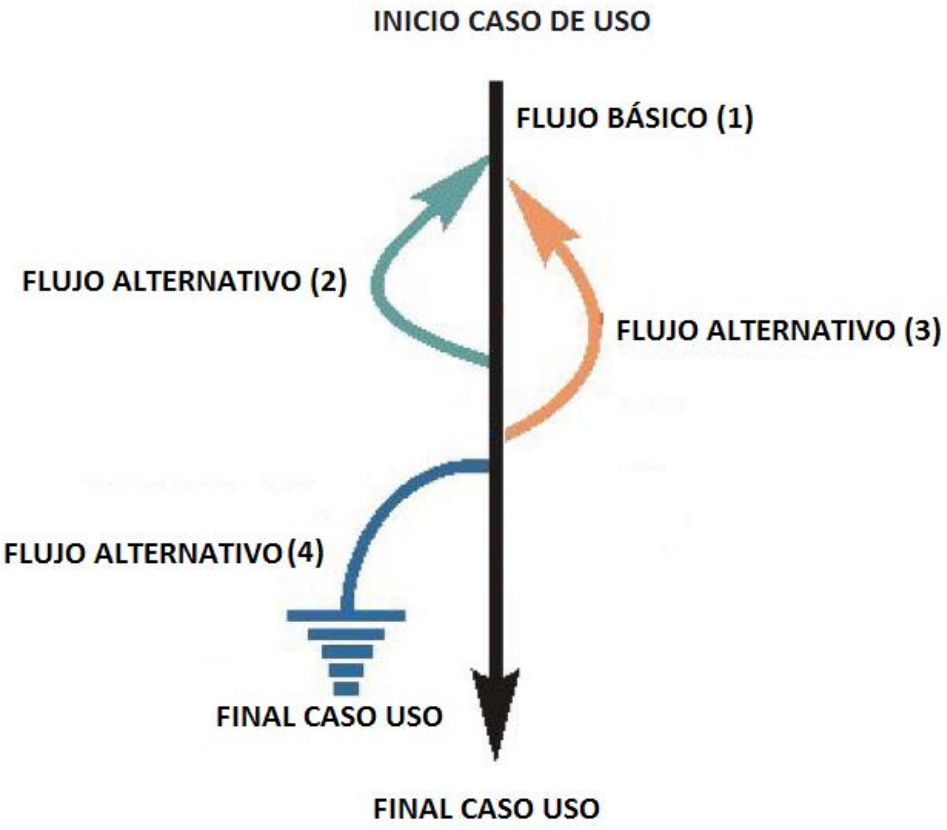


Ilustración 30. Diagrama de flujos y escenarios.

6.2. Compatibilidad

La compatibilidad para interaccionar correctamente con esta aplicación es un tema a tener en cuenta, aunque no crítico. Como ya se sabe, en el ámbito relacionado con la web, hay un factor no controlable por el programador como es el navegador o explorador. Nos gustaría remarcar que tras las pruebas de compatibilidad realizadas no existen diferencias entre aquellos navegadores que siguen las reglas de estandarización del consorcio W3C que son Internet Explorer 7 o superior, Firefox, Chrome o Safari. Sí existe, por otro lado, algunas diferencias de posicionamiento visual con Internet Explorer 6 (que no seguía los estándares del W3C).

Por otro lado, nos podemos encontrar con la opción de que JavaScript no esté permitido en el explorador (desactivado), y a pesar de que está contemplado como requerimiento no funcional (ver punto 3.1), siempre cabe la oportunidad de encontrarnos con ello. En cualquier caso, detectaríamos una falta de ayuda visual pero la aplicación puede seguir actuando correctamente sin ningún punto crítico. Esto es debido a que JavaScript no tiene asignada ninguna función imprescindible, sino solo de complemento a PHP. Un ejemplo sería que la validación de los formularios depende única y exclusivamente de algoritmos PHP.

CAPÍTULO 7

CONCLUSIONES

Las sensaciones finales son muy positivas. A nivel de proyecto, creemos haber cumplido con todos los objetivos que nos habíamos propuesto, habiendo logrado una aplicación muy competitiva y preparada para ser implantada en el mundo real. Una aplicación que sigue las reglas de la ingeniería del software y que está lista para los nuevos tiempos. El diseño robusto le permitirá crecer en varias líneas sin ningún problema.

Es por seguro, que nuestro gestor no podrá permanecer intacto con el tiempo. Todos conocemos el exponencial ritmo de crecimiento que lleva el sector de las nuevas tecnologías, por ello es que vemos a nuestra aplicación capaz de afrontar nuevos retos y ofrecer nuevos valores de negocio. Como futuras líneas de ampliación, que pueden tenerse en cuenta a corto plazo serian:

- Desarrollar funciones de predicción del tiempo. Para prever al ingeniero el *timing* medio que se está tomando en realizar un ensayo.
- Generador de planes de validación automáticos para un proyecto y norma en concreto.

Si bien hay que hacer un punto de autocrítica, éste seria sobre la planificación a nivel de *timing*. Hemos sufrido un retraso bastante considerable a tener muy en cuenta para el futuro. Concretamente las tareas relativas al desarrollo de la aplicación, 2.1 Codificación y 2.2 Implementación y test (ver Tabla 4 e Ilustración 31), son las que han sufrido más retraso y en consecuencia el proyecto, puesto que ya teníamos conocimiento de que eran las más críticas y cualquier retraso en ellas afectaría directamente sobre la fecha final.

Tabla 4. Corrección del análisis del trabajo requerido en horas.

Id	Tarea	Trabajo desarrollador (h)	Trabajo supervisora proyecto
1	***Análisis de los procesos***	***80***	***5***
1.1	Análisis de requerimientos	40	2,5
1.2	Diseño	40	2,5
2	***Desarrollo de la aplicación***	***205***	***7***
2.1	Codificación	180	6
2.2	Implementación y test	25	1
3	***Documentación y memoria***	***70***	***3***

	del proyecto		
3.1	Introducción del escenario	20	0,5
3.2	Estudio de viabilidad	20	0,5
3.3	Documentación de la aplicación	30	2
TRABAJO TOTAL (h)		**355**	**15**

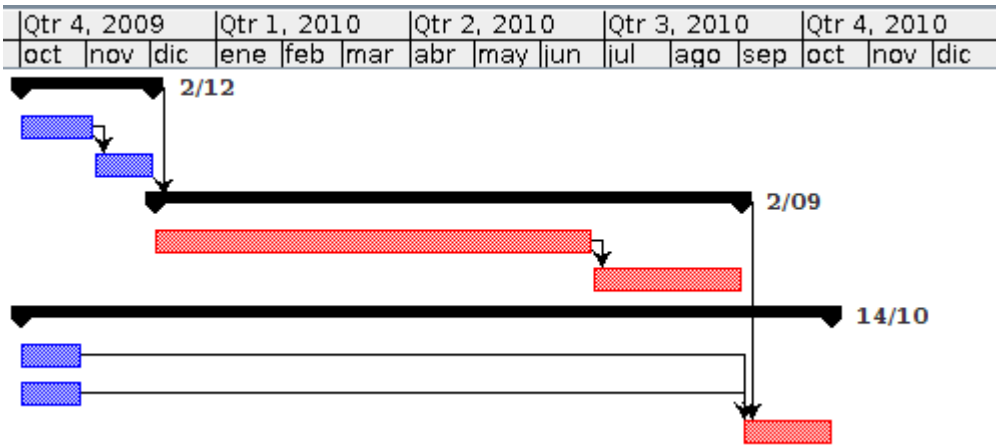


Ilustración 31. Diagrama de Gantt actualizado.

Los motivos concretos que nos han afectado al retraso del proyecto, son dos. El primero fue el cálculo de horas empleadas en la parte más práctica. La estimación no fue acertada y la realidad ha sido un incremento de horas de diferencia considerable. El segundo motivo, fue el tiempo disponible para la dedicación del proyecto. La combinación de trabajo y estudios, en ocasiones provoca no poder prestar todo el tiempo requerido a uno en reprimenda del otro. Ello nos llevó a largar en el tiempo las tareas de programación y test.

A nivel más personal, la valoración es totalmente positiva. No solo he conseguido plasmar los conocimientos aprendidos durante estos últimos años, sino que me ha permitido aprender y conocer nuevos campos totalmente desconocidos para mí.

BIBLIOGRAFÍA

Publicaciones impresas:

- Rob Allen, Nick Lo, Steven Brown, “Zend Framework in Action”, Manning Publications, 2009. Idioma: Inglés.

- Rob Allen, “Getting started with Zend Framework”, Akrabat, 2006. Idioma: Inglés.

Recursos electrónicos:

- Zend community. Guía de referencia para programadores. Especificación detallada de todas las clases de Zend Framework. Disponible en: http://framework.zend.com/manual/en/. (3 de septiembre 2010). Idioma: Inglés.

- Mehdi Achour, Friedhelm Betz, Antony Dovgal, Nuno Lopes, Hannes Magnusson, Georg Richter, Damien Seguy, Jakub Vrana. Manual de PHP. Disponible en: http://www.php.net/manual/en/index.php. (11 de septiembre 2010). Idioma: Inglés.

- Matthew Weier O'Phinney. Trata los decoradores de Zend Framework. Disponible en: http://devzone.zend.com/article/3450 (5 de mayo de 2008). Idioma: Inglés.

- Andrés Guzmán. Blog que ofrece un tutorial de iniciación a Zend Framework. Disponible en: http://bolsadeideas.cl/zsamer/category/php5/zend- framwork/ page/2/. (11 de enero 2009). Idioma: Español.

- Andrés Guzmán. Proyectos de código libre ofrecido por Google. Disponible en: http://code.google.com/p/zend-framework-datagrid/. (11 agosto 2010). Idioma: Inglés

Sabadell, a septiembre de 2010

Diego Cano Nieto