ÍNDICE

CAPÍTULO 1. INTRODUCCIÓN

Bnet es el nombre del Proyecto de Fin de Carrera desarrollado por Lilia González y dirigido por Germán Rigau i Claramunt para la titulación de Ingeniería Técnica en Informática de Sistemas en convenio con la empresa Gisma Asesoramiento.

El objetivo de este proyecto es desarrollar un sistema web de comunicación empresarial enfocado a crear nuevas relaciones comerciales utilizando como medio la valoración de productos y/o servicios realizada por los contactos de cada empresa usuaria de la aplicación.

1.1 Antecedentes y motivación

Como bien sabemos Internet ha supuesto una revolución en el mundo de las comunicaciones. Tanto la descentralización de los datos como la ruptura de barreras espacio-temporales ha permitido un crecimiento exponencial de la "Red de Redes" y ha abierto un sinfín de nuevas posibilidades de explotación comercial.

Además, en estos últimos tiempos las redes sociales han experimentado un gran auge entre los usuarios de Internet. Aplicaciones sociales como Facebook, Tuenti, Twitter,.. han permitido a los usuarios mantener el contacto con sus amistades y compañeros así como crear nuevas relaciones.

Sin embargo, "La Red" es un espacio demasiado grande, la información está dispersa por todas partes y muchas veces encontrar información fiable resulta difícil.

Motivado por esta situación, la empresa Gisma Asesoramiento propuso la creación de un prototipo de red social empresarial destinada a convertirse en un canal para mejorar la búsqueda de nuevos distribuidores o fabricantes basándose en la opinión contactos (empresas “amigas”).

1.2 Descripción del proyecto

El principal objetivo de la aplicación será crear un sistema que permita que cada usuario obtenga la información que necesita para poder seleccionar nuevos fabricantes o proveedores basándose en las opiniones de sus contactos y no únicamente de usuarios anónimos, permitiendo así una valoración mas fiable.

Una vez realizada la aplicación un usuario podrá:

- Invitar a otras empresas a registrarse en el sistema.
- Buscar empresas y ver su perfil.
- Gestionar sus contactos
- Registrar ventas a otros usuarios
- Valorar sus compras
- Ver las valoraciones de los contactos
- Ver sus balances sobre compras y ventas

CAPÍTULO 2. DOCUMENTO DE OBJETIVOS DE PROYECTO

2.1 Descripción

Bnet es el nombre del Proyecto de Fin de Carrera desarrollado por Lilia González y dirigido por Germán Rigau i Claramunt para la titulación de Ingeniería Técnica en Informática de Sistemas de la UPV-EHU para el curso 2009-2010 y coordinado por la empresa Gisma Asesoramiento.

El proyecto consiste en crear un prototipo de aplicación web que tendrá como objetivo obtener financiación por diferentes medios para lanzar al mercado, en un futuro, el producto completo.

La idea principal de la aplicación puede verse como una red social de empresas que servirá a éstas como herramienta para encontrar nuevos proveedores/fabricantes de calidad basándose en las opiniones de los contactos de cada empresa y así crear círculos de confianza empresarial.

2.2 Objetivos

Existen tres objetivos principales que corresponden a la totalidad de la realización del proyecto:

Objetivo 1 – Realización de un prototipo funcional para la empresa.

Objetivo 2 – Creación de toda la documentación necesaria para la aprobación del proyecto por parte de la Universidad.

Objetivo 3 – Formación en distintos campos (programación, documentación, metodología de trabajo en el mercado laboral,...) para el alumno.

2.3 Método de trabajo

2.3.1 Equipo de trabajo:

G. Sedano: Es el responsable de la empresa y será el supervisor del proyecto. Su tarea principal será notificar los requisitos necesarios para la aplicación, así como proporcionar feedback sobre el desarrollo del prototipo.

K. Mola: Realizará junto con G.Sedano de las tareas de supervisión del alumno en la empresa además de formar al alumno en los aspectos técnicos para que pueda realizar su trabajo.

G. Rigau: Será el director del proyecto en la Universidad y por lo tanto supervisará el trabajo desarrollado por el alumno.

L.González: La alumna encargada del proyecto y por tanto estará presente en todas las fases del proyecto. Será tarea de ésta desarrollar la aplicación así como documentar todo lo necesario y defender al final el trabajo realizado, en la Universidad.

2.3.2 Calendario de Trabajo

Está previsto que el proyecto sea presentado en Junio del 2010 por lo que las previsiones se harán de acuerdo a esta fecha. Sin embargo, es posible que las horas reales no coincidan con las planificadas y sea por tanto prorrogado el plazo de entrega hasta Septiembre del 2010.

El alumno dedicará a realizar el proyecto 4 horas diarias dentro de la empresa. El horario será de 15:00 a 19:00 de lunes a viernes, durante todo el periodo del 01-01-2010 al 31-05-2010, lo que serán unas 400 horas, en las que se incluirá la formación necesaria, el tiempo de diseño e implementación de la aplicación y la realización de la documentación necesaria. El alumno, se reserva además la posibilidad de dedicar más horas fuera de este horario si lo cree necesario.

2.3.3 Reuniones

Para garantizar la calidad del trabajo realizado por el alumno y realizar un seguimiento apropiado se realizarán dos tipos de reuniones.

Reuniones con el profesor: que asegurarán que el alumno está siendo capaz de desarrollar su función.

Reuniones con los miembros de la empresa: Se realizarán dos veces por semana, para garantizar que se cumplen los plazos acordados y que el trabajo sigue los requisitos marcados.

2.3.4 Soporte técnico

La empresa cuenta con un programa de registro de horas dedicadas a proyectos, por lo que es requisito que cada día a última hora sea registrado el trabajo realizado y el tiempo dedicado a cada tarea para ayudar a la estimación temporal de futuros proyectos

El alumno realizará una copia diaria de los archivos del proyecto, por otro lado la empresa, como política ya creada, realizará una copia de seguridad de la aplicación y de la base de datos subidos al servidor.

2.3.5 Proceso de desarrollo del software

Un proceso de desarrollo de software es una metodología que tiene como propósito producir y mantener software eficaz y eficiente. A lo largo de los años y a medida que el software ha ido cambiando se han desarrollado distintos procesos de desarrollo como el ciclo de vida en cascada, el modelo en espiral o el proceso unificado de software (PUD).

El Proceso Unificado de Desarrollo ha adquirido una serie de características de otros procesos que le hacen ser una metodología genérica y adaptable a distintos tipos de proyectos.
Estas características son:

- **Guiado por los casos de uso:**

  Un caso de uso es una manera de plantear un problema que el software tiene que resolver. Indica cómo los usuarios interactúan con el sistema y define los posibles escenarios que pueden darse en cada situación.
  El Modelo de Casos de Uso (MCU) es la parte del PUD con el que se logra describir la funcionalidad total del sistema.

- **Centrado en la arquitectura**

  La arquitectura del sistema tiene que realizarse pensando en aspectos cómo cuáles serán los equipos que alojaran las aplicaciones, qué componentes externos se utilizarán y cómo pueden afectar a nuestro sistema, etc.
  La arquitectura para seguir la metodología del PUD debe ser flexible y fácil de modificar, compresible y que en lo posible que permita la reutilización de componentes.

- **Enfocado en los riesgos**

  Esta metodología procura abordar los obstáculos más grandes al inicio del

proyecto para poder analizar la factibilidad del proyecto lo antes posible. Esta razón es la que lleva a realizar un núcleo consistente en las primeras iteraciones del proyecto.

- **Sigue un ciclo de vida iterativo e incremental**

  Los proyectos se dividen en iteraciones que se tratan igual que subproyectos con fases bien diferenciadas y con plazos establecidos.

  En el proyecto esta metodología se traduce en la realización de varias iteraciones que contemplan las siguientes fases (captura de requisitos, análisis, diseño,implementación, pruebas y desarrollo) y que permitirán dotar a la aplicación de la flexibilidad y estabilidad necesaria para la consecución de los objetivos marcados.

2.4 Alcance

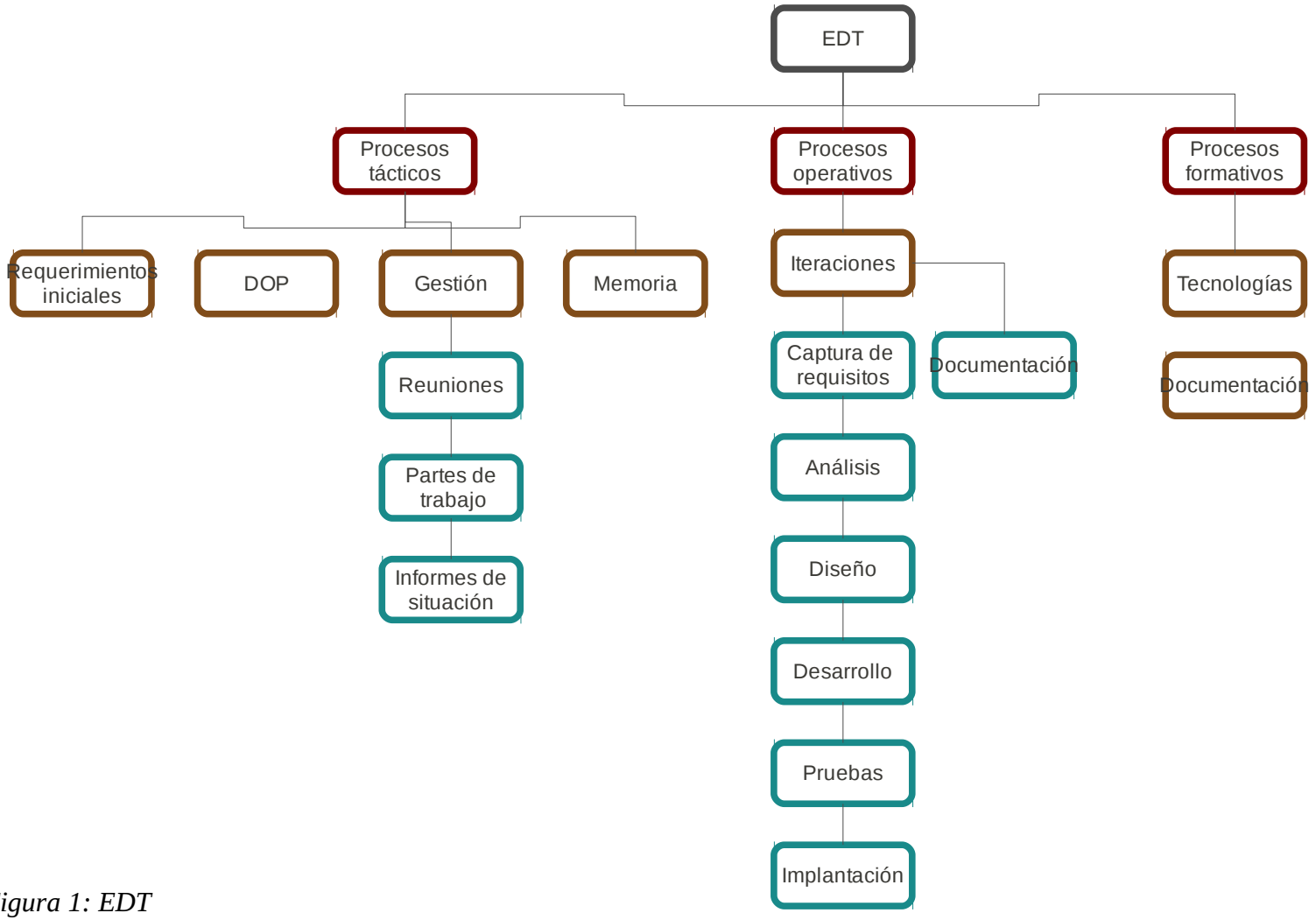


Figura 1: EDT

2.4.1 Tareas

Procesos tácticos

Requisitos Iniciales

Esta tarea consistirá en realizar primero una toma de contacto con la empresa, analizar cuales son los requisitos del sistema que se quiere desarrollar y la factibilidad del proyecto en el contexto que nos ocupa. En este momento se decidirá también cuales serán las tecnologías que se usarán para el desarrollo de la aplicación.

DOP

La redacción del presente documento será el objetivo de este proceso. Este documento será la pauta a seguir, la base sobre la que empezar a trabajar. Además servirá posteriormente para analizar los resultados y comprobar todos los cambios que se han realizado a lo largo del proyecto.

Gestión

Una parte importante para el buen desarrollo del proyecto es llevar una gestión adecuada de todos los procesos del proyecto. Esto se plasmará en tres documentos: las actas de las reuniones relevantes, los partes diarios de trabajo y los informes de situación.

Memoria

Como requisito para la aprobación del proyecto es necesario redactar una memoria que será presentada para su evaluación.

Procesos operativos

Se realizarán varias iteraciones para el desarrollo de la aplicación. La primera iteración estará compuesta de un único caso de uso que será evaluado por el profesor para dar una guía inicial al alumno. A partir de ahí, se irán añadiendo casos de uso. En cada iteración se realizarán las subtareas marcadas en el EDT y se documentarán todas.

Procesos formativos

Será necesario dedicar gran parte del tiempo disponible a aprender sobre las tecnologías que se usarán para la realización de la aplicación, las pruebas y la implantación en el equipo final.

También será necesario repasar y mejorar los conocimientos aprendidos en Ingeniería del Software para realizar una gestión y memoria adecuadas.

2.4.2 Hitos y entregas

Se planearán varios hitos. Algunos son obligatorios y otros están para comprobar el buen ritmo de trabajo.

Hito	Fecha	Descripción
Toma de contacto	02/12/09	Empiezan las negociaciones con la empresa, universidad,...
Entrega del boceto del DOP	15/12/09	Confirmación del profesor como director de proyecto y presentación del boceto de DOP
Inicio	01/01/10	Inicio del proyecto
Iteración 1	15/01/10	Muestra de la primera iteración
Iteración 2	15/02/10	Segunda iteración
Iteración 3	15/04/10	Tercera iteración
Iteración 4	15/05/10	Iteración final y memoria
Entrega memoria	30/05/10	Entrega de las copias en secretaría
Presentación pública	Segunda quincena de junio	Presentación del proyecto ante el jurado

Tabla 1. Hitos y entregas

2.4.3 Recursos

Materiales

La empresa pone a disposición del alumno los siguientes recursos: un ordenador de sobremesa, conexión a Internet, una impresora, material de oficina y un servidor para colgar la aplicación. Además beneficia al alumno con un salario mensual por su trabajo.

Horas

Se estima más o menos que el proyecto llevará unas 400 horas de trabajo en la empresa, más el tiempo que el alumno estime necesario.

2.5 Planificación

Este apartado muestra la estimación temporal que se ha establecido para el desarrollo de todas las fases. La *Figura 2* plasma dicha estimación.

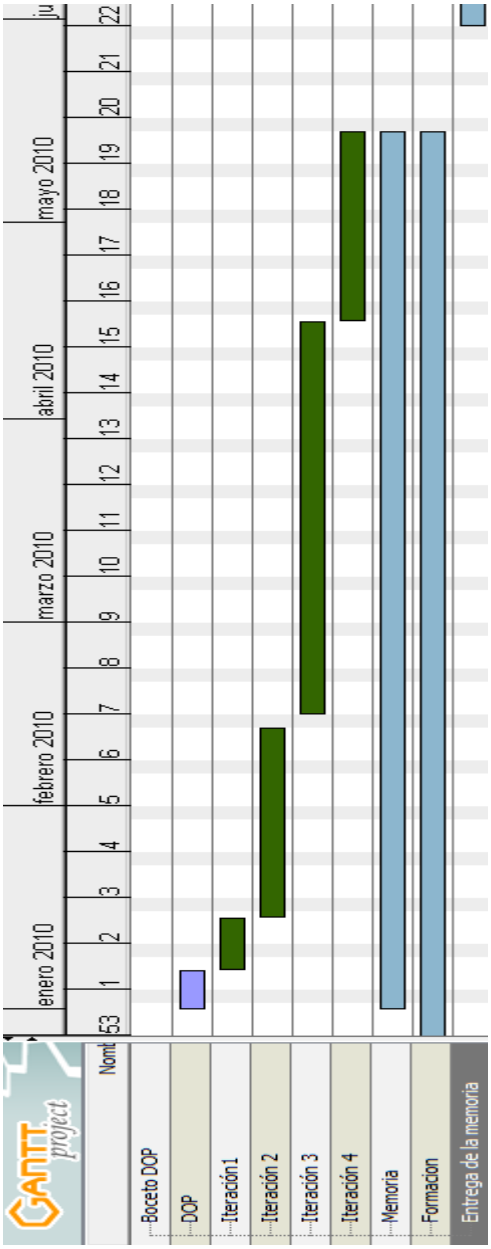


Figura 2. Diagrama de Gantt

El proyecto comenzará redactando el presente documento, para ello será necesaria una reunión inicial con el alumno y los encargados del proyecto en Gisma Asesoramiento. En ella se tratarán los aspectos más relevantes para la estimación temporal como funcionalidad del sistema, disponibilidad horaria, número de iteraciones, etc.

Cómo ya se ha indicado anteriormente se realizarán cuatro iteraciones. La primera iteración se hará antes de los exámenes universitarios y será la más corta ya que se realizará un único caso de uso completo que será comprobado por el profesor para que sirva como guía válida para el resto de casos de uso.

En la segunda iteración se añadirán algunos casos de uso más para comprobar que se sigue la metodología adecuada y no alargar demasiado los plazos.

La tercera iteración será la más larga ya que incluye las vacaciones de Semana Santa. Si los plazos se desajustaran mucho este sería el momento idóneo para corregirlo.

La última iteración procurará mejorar y solucionar los últimos detalles de la aplicación. Finalizará en Mayo para tener tiempo suficiente por si surgieran problemas antes del plazo de entrega de la memoria.

También podemos observar que la redacción de los distintos apartados de la memoria se realizarán a lo largo de todo el proceso así como la adquisición de formación necesaria para el alumno.

Horas planificadas por tarea	
DOP	8
Captura de Requisitos	30
Análisis	50
Diseño	80
Implementación	100
Pruebas	30
Implantación	4
Interfaz gráfica	20
Formación	40
Reuniones y documentación	38
Total	400

Tabla 2. Horas planificadas por tarea

En la *tabla 2* podemos ver la repartición de horas por tarea, para poder comparar en la memoria final las diferencias entre lo planificado y lo real.

Como puede observarse esta previsto que el mayor tiempo se dedique al diseño e implementación de la aplicación.

2.6 Plan de contingencia

Para facilitar la comprensión de los riesgos presentes durante la realización del proyecto y el coste que puede suponer cualquiera de estos sucesos, se han incluido, además de las posibles soluciones de cada riesgo la manera de preeverlos, la probabilidad de que este suceda y el nivel de gravedad con que pone en riesgo el proyecto (ambos factores según la escala: bajo, medio, alto, muy alto)

Riesgo: Incumplimiento de un plazo de entrega	
Probabilidad: Muy alta	Gravedad: Media
Si se prevé que si una entrega no se hará a tiempo será necesario realizar una reunión o notificarlo por email al director de proyecto para comentar que ha llevado a esa situación y si es necesario replantear las fechas de entrega siguientes.	

Riesgo: Baja temportal o permanente	
Probabilidad: Muy baja	Gravedad: Muy alta
Se notificará inmediatamente al profesor y se planteará si hay que tomar algún tipo de medida especial.	

Riesgo: Pérdida de datos del proyecto	
Probabilidad: Baja	Gravedad: Muy alta
Se seguirá una política de copias de seguridad (Véase la sección método de trabajo)	

Riesgo: Estimaciones temporales mal realizadas	
Probabilidad: Muy alta	Gravedad: Media
Al no ser un software que vaya a implantarse actualmente y no existe un cliente como tal, las fechas pueden ajustarse sin demasiados problemas	

Riesgo: Falta de conocimiento	
Probabilidad: Alta	Gravedad: Baja
Está previsto que sea necesario mejorar la formación en varios campos. Método a proceder: - Se buscará material de ayuda en Internet o la biblioteca de la Universidad - Se pedirá ayuda a personas con experiencia: profesores, compañeros de empresa, foros,... - Si fuera necesario se buscarán tecnologías o métodos más sencillos e intuitivos	

Riesgo: Error de diseño	
Probabilidad: Muy alta	Gravedad: Media
Si se produce un error en el diseño de una parte ya implementada será necesario valorar hasta que punto conviene rediseñar esa parte, si fuera necesario se volverá a realizar	

2.7 Factibilidad

Realizado el estudio de costes y horas, se considera que el proyecto Bnet será factible y se prevé que las entregas podrán ser terminadas en el plazo fijado y se cuenta con un margen flexible para la entrega final (septiembre)

CAPITULO 3. CAPTURA DE REQUISITOS

3.1 Actores

Esta aplicación cuenta únicamente con dos actores:

- Persona: es cualquier usuario no inscrito en el sistema
- Usuario: cualquier empresa registrada en el sistema

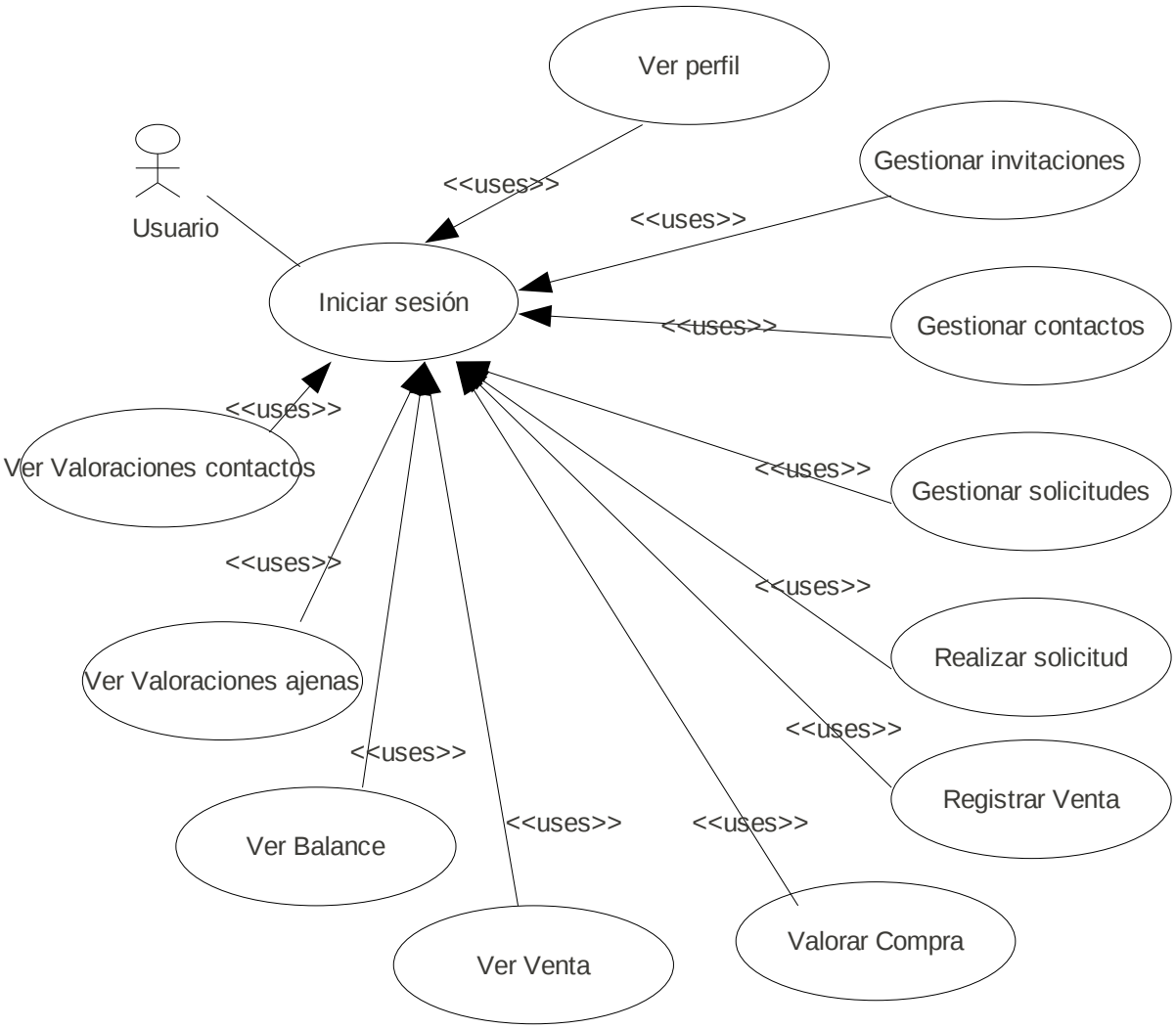


Figura 3. Casos de uso del usuario

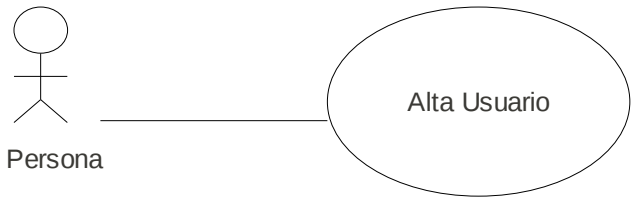


Figura 4. Casos de uso de persona

3.2 Requisitos

Rf1: Posibilidad de dar de alta otros usuarios a través de invitaciones por email (limitadas a 5)

Rf2: Acceso sólo a usuarios registrados para cualquier operación

Rf3: Posibilidad de comprobar los datos de registro

Rf4: Posibilidad de buscar usuarios del sistema para añadir como contactos

Rf5: Gestionar las peticiones

Rf6: Ver ventas

Rf7: Registrar nuevas ventas

Rf8: Ver compras y poder valorarlas del 1 al 5

Rf9: Ver balances, es decir, compras y ventas mensuales o anuales

Rf10: Ver la valoración media de una empresa por todos los usuarios del sistema según la valoración de sus compras.

Rf11: Ver la valoración media de una empresa por los contactos del usuario

Rf12: Buscar la empresa a la que ver la valoración media, buscandola por sector y provincia o mediante un patrón de búsqueda

Rf13: El proyecto debe ser desarrollado en PHP o JSP

3.3 Casos de uso

En esta sección se presenta el Modelo de Casos de Uso que contiene los distintos situaciones que se le presentan al usuario y sus posibles escenarios alternativos.

3.3.1 Iniciar Sesión

Curso normal de los eventos

1 . Usuario: Un usuario accede a través de un navegador a la página inicial de la web Bnet. Introduce su nombre de usuario y contraseña.
2 . Sistema: Comprueba que los datos son válidos para ese usuario y carga la página de bienvenida para usuarios registrados que incluye las opciones a las que puede acceder.

Curso alternativo

Paso 2 . Los datos no son válidos.
Sistema: Vuelve a cargar la página inicial mostrando un mensaje de error

3.3.2 Ver Perfil

Curso normal de los eventos

1 . Usuario: Un usuario registrado quiere ver su perfil
2 . Sistema: Le muestra los datos asociados a su perfil.

3.3.3 Gestionar invitaciones

Curso normal de los eventos

1 . Usuario: Un usuario registrado quiere gestionar sus invitaciones

2 . Sistema: Le muestra el número de invitaciones disponibles y un formulario para que el usuario pueda introducir los datos de la invitación

3 . Usuario: Rellena los datos necesarios

4 . Sistema: Comprueba que los datos son válidos y envía la invitación por email

Curso alternativo

Paso 2 . El usuario no dispone de más invitaciones

Sistema: Muestra un mensaje informando de la situación

Paso 3 . El usuario sólo quiere ver el número de invitaciones pero no quiere realizar ninguna

Paso 4. La dirección de email no es válida.

Sistema: Muestra un mensaje de error

3.3.4 Eliminar contacto

Curso normal de los eventos

1 . Usuario: Un usuario registrado quiere eliminar un contacto

2 . Sistema: Le muestra toda su lista de contactos y le da la opción de borrarlos

3 . Usuario: Selecciona el contacto que quiere eliminar

4 . Sistema: Elimina el usuario

Curso alternativo

Paso 2 . El usuario no posee ningún contacto

Sistema: Muestra un mensaje informando de la situación

Paso 3 . No selecciona ningún contacto

3.3.5 Gestionar solicitudes

Curso normal de los eventos

1 . Usuario: Un usuario registrado quiere gestionar sus solicitudes de amistad

2 . Sistema: Le muestra toda su lista de solicitudes (las realizadas por el usuario y las peticiones que le han llegado). Permite al usuario aceptar o rechazar una solicitud de amistad recibida

3 . Usuario: Selecciona sobre un contacto la opción aceptar

4 . Sistema: Acepta la petición y lo añade a contactos

Curso alternativo

Paso 2 . No hay solicitudes

Sistema: Muestra un mensaje informando que no hay solicitudes.

Paso 3 . No selecciona ningún contacto

Paso 3' . Quiere rechazar la petición

Sistema: Rechaza la petición, la elimina de la lista y no permite al solicitante volver a realizar la petición

3.3.6 Realizar solicitud

Curso normal de los eventos

1 . Usuario: Un usuario registrado quiere realizar una petición de amistad a otro usuario del sistema. Introduce un patrón de búsqueda y le da a buscar.
2 . Sistema: Devuelve una lista con usuarios del sistema que coinciden con ese patrón.
3 . Usuario: Selecciona al que quiere realizar la petición de contacto
4 . Sistema: Envía la petición de amistad al usuario seleccionado

Curso alternativo

Paso 2 . No hay contactos que coincidan con la búsqueda
Sistema: Muestra un mensaje informando de la situación
Paso 3 . No selecciona ningún contacto

3.3.7 Registrar venta

Curso normal de los eventos

1 . Usuario: Un usuario registrado quiere registrar una venta realizada a otro usuario. Introduce el nombre de usuario.
2 . Sistema: Si el usuario existe, permite introducir el resto de los datos necesarios para registrar la venta
3 . Usuario: Introduce los datos
4 . Sistema: Registra la venta

Curso alternativo

Paso 2 . No existe el comprador al que quiere registrar esa venta
Sistema: Informa del error

3.3.8 Valorar compra

Curso normal de los eventos

1 . Usuario: Un usuario registrado quiere valorar una compra
2 . Sistema: Muestra la lista de compras de ese usuario
3 . Usuario: Selecciona la compra a valorar
4 . Sistema: Le muestra los datos de la compra y le da la opción de valorarla con una puntuación del 1 al 5.
5 . Usuario: Valora la compra
6 . Sistema: Registra la valoración

Curso alternativo

Paso 2 . No existen compras para ese usuario

Sistema: informa de la situación

Paso 5. Decide salir del caso de uso

3.3.9 Ver ventas

Curso normal de los eventos

1 . Usuario: Un usuario registrado quiere ver sus ventas

2 . Sistema: Muestra la lista de ventas

3.3.10 Ver balance

Curso normal de los eventos

1 . Usuario: Un usuario registrado quiere ver sus balances. Selecciona el año y/o el mes del que quiere ver su balance

2 . Sistema: Muestra el balance del periodo indicado

Curso alternativo

Paso 2 . No existen compras o ventas asociadas a ese periodo para ese usuario

Sistema: informa de la situación

3.3.11 Ver valoraciones ajenas

Curso normal de los eventos

1 . Usuario: Un usuario registrado quiere ver la valoración media de una empresa. Introduce un patrón de búsqueda.
2 . Sistema: Muestra las empresas coincidentes (que no sean ella misma) y sus valoraciones medias

Curso alternativo

Paso 1: El usuario no quiere buscar las empresas por el nombre sino que quiere ver las de un sector concreto en cierta provincia.
Usuario: Selecciona un sector y una provincia.
Sistema: Muestra las empresas coincidentes y sus valoraciones medias

Paso 2 . No hay empresas que coincidan con el patrón de búsqueda
Sistema: informa de la situación

3.3.12 Ver valoraciones contactos

Curso normal de los eventos

1 . Usuario: Un usuario registrado quiere ver como está valorada una empresa por sus contactos. Introduce un patrón de búsqueda.
2 . Sistema: Muestra las empresas coincidentes (que no sean ella misma) y las valoraciones medias de los contactos del usuario o se muestran como "no valoradas" si no hay valoración de los contactos para esa empresa

Curso alternativo

Paso 1: El usuario no quiere buscar las empresas por el nombre sino que quiere ver las de un sector concreto en cierta provincia.
Usuario: Selecciona un sector y una provincia.
Sistema: Muestra las empresas coincidentes y sus valoraciones medias.
Paso 2 . No hay empresas que coincidan con el patrón de búsqueda
Sistema: informa de la situación

3.3.13 Alta Usuario

Curso normal de los eventos

1 . Usuario: Un usuario recibe un email con una invitación a darse de alta en el sistema. Entra en la dirección indicada

2 . Sistema: Le muestra un formulario con los datos a rellenar

3 . Usuario: Rellena los datos de la empresa y envia el alta

4 . Sistema: Comprueba que los datos son válidos, registra el alta y envía un email con la contraseña para entrar en el sistema

Curso alternativo

Paso 4: Los datos no son válidos

Sistema: Informa de la situación y le permite volver a introducir los datos

3.4 Modelo de dominio

La figura 5 representa el modelo de dominio de Bnet que incluye la información sobre las empresas, las relaciones entre los usuarios del sistema y las ventas y compras.

Un **usuario** tiene asociado un nombre de usuario y una contraseña, además tiene un estado por si quiere desactivarse el usuario en algún momento, además cuenta con un número limitado de invitaciones (5 al principio).

Una **empresa** tiene un id asociado que tiene su equivalente en la tabla usuario, ya que cada usuario es en realidad una empresa. Se ha separado para tener por un lado la información del sistema (usuario, estado, número de invitaciones,...) y por otro la información de la empresa(dónde se encuentra, nombre,...). Además cada empresa puede elegir hasta 3 **sectores** con los que identificarse.

Cada usuario tiene asociado una **lista de contactos**, es decir, relación con otros usuarios.

Por último un usuario puede registrar una venta a otro usuario y esta queda registrada como compra para el otro. Cada comprador puede valorar cada compra, esta acción es lo que hace que el sistema pueda lograr su objetivo principal.

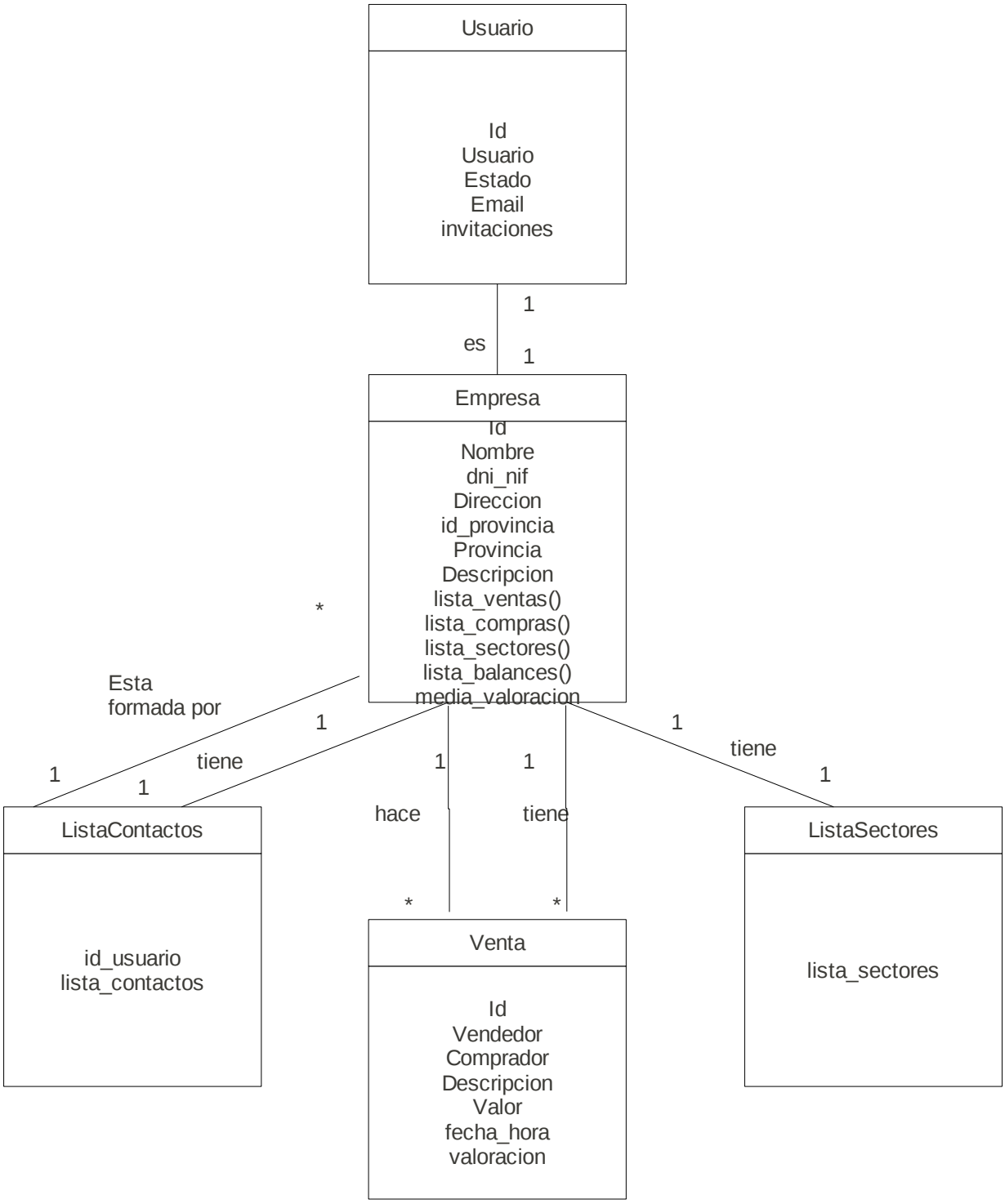


Figura 5. Modelo de Dominio del Sistema

CAPITULO 4. ANÁLISIS

4.1 Iniciar Sesión

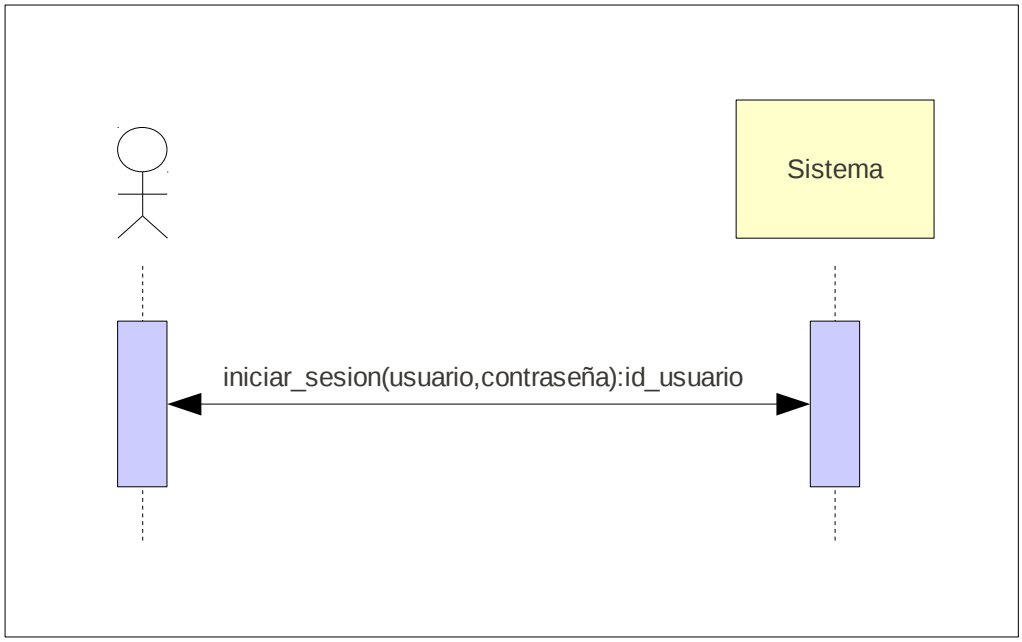


Figura 6. Diagrama de Secuencia de Iniciar Sesión

4.1.1Contratos

Operación: identificar_usuario(usuario, contraseña)

Responsalibidades: Comprobar que el usuario es un usuario del sistema y que la contraseña corresponde a la de ese usuario.

Precondiciones: La contraseña está cifrada con md5

Postcondiciones: Permite acceder a la página de bienvenida

Salida: id_usuario si es válido, -1 eoc

4.2 Ver Perfil

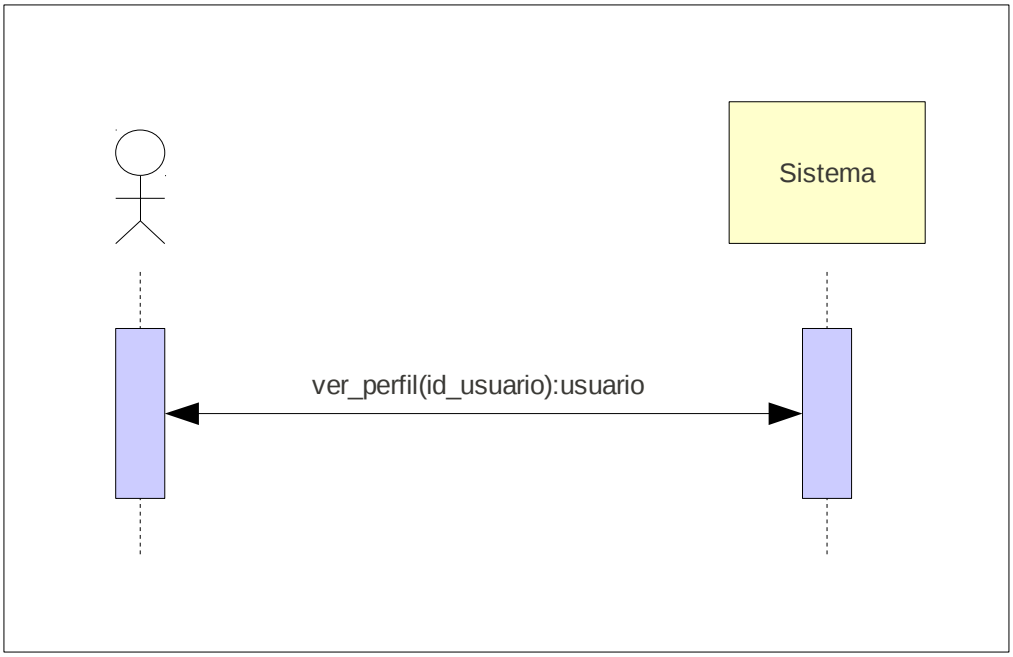


Figura 7. Diagrama de Secuencia de Ver Perfil

4.2.1 Contratos

Operación: ver_perfil(Id_usuario)

Responsalibidades: Obtiene los datos del perfil de la empresa

Precondiciones: Usuario identificado

Postcondiciones: Muestra los datos

Salida: usuario

4.3 Gestionar Invitaciones

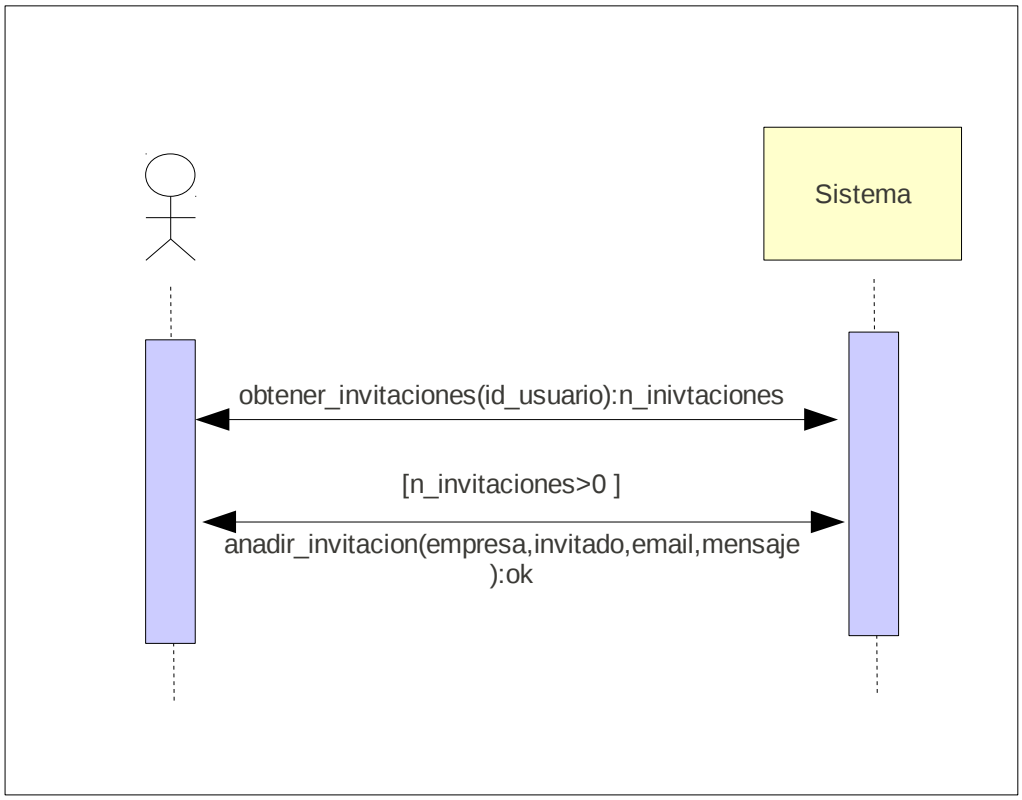


Figura 8. Diagrama de Secuencia de Gestionar Invitaciones

4.3.1 Contratos

Operación: obtener_invitaciones(Id_usuario): n_invitaciones

Responsalibidades: Obtiene el número de invitaciones disponibles para un usuario

Precondiciones: Usuario identificado

Postcondiciones:

Salida: n_invitaciones=numero de invitaciones

Operación: anadir_invitacion(empresa,invitado,email,mensaje)

Responsalibidades: Envia una invitación con un mensaje personalizado

Precondiciones: Usuario identificado, hay invitaciones disponibles

Postcondiciones: Se elimina una invitación al usuario de total de posibles n_invitaciones=n_invitaciones-1.

Salida: ok=-1 si se produce un error al enviar el email

4.4 Eliminar contacto

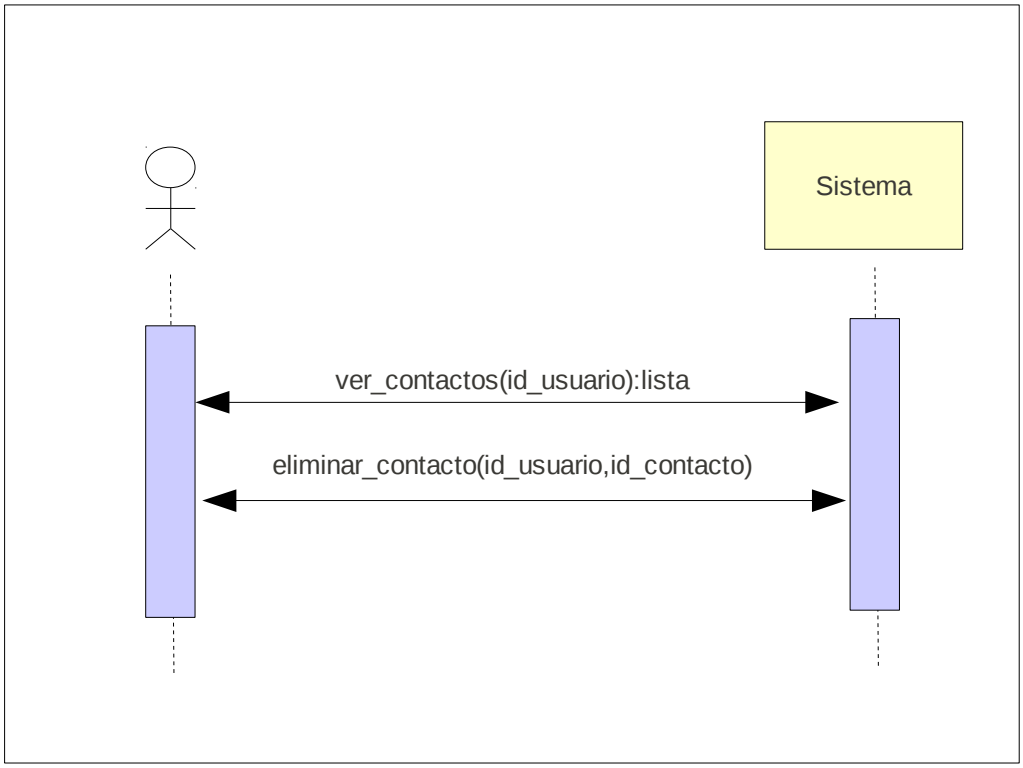


Figura 9. Diagrama de Secuencia de Eliminar Contacto

4.4.1 Contratos

Operación: ver_contactos(Id_usuario)

Responsalibidades: Obtiene la lista de contactos del usuario

Precondiciones: Usuario identificado

Postcondiciones:

Salida: lista(empresa)=lista de empresas que son contacto del usuario

Operación: eliminar_contacto(id_usuario,id_contacto)

Responsalibidades: Elimina el contacto de la lista de contactos del usuario

Precondiciones: Usuario identificado, es contacto del usuario

Postcondiciones: id_contacto deja de ser contacto del usuario

Salida:

4.5 Gestionar solicitudes

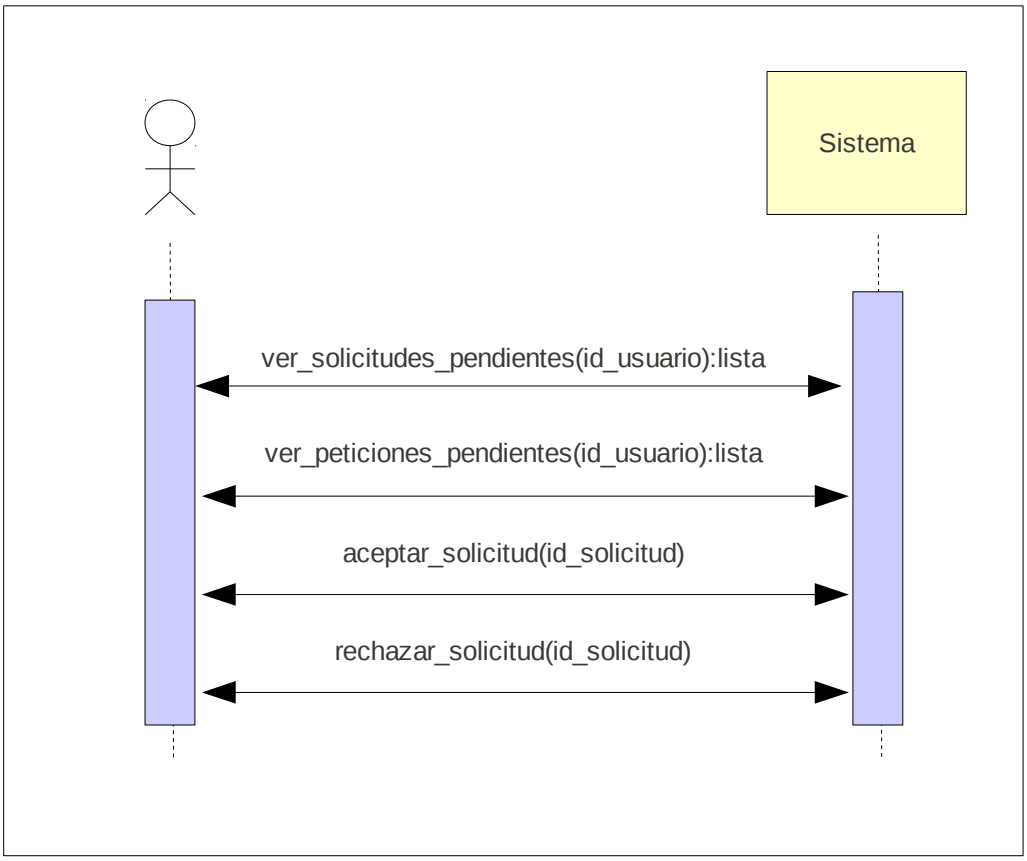


Figura 10. Diagrama de Secuencia de Gestionar Solicitudes

4.5.1 Contratos

Operación: ver_solicitudes_pendientes(Id_usuario)

Responsalibidades: Obtiene la lista de solicitudes de amistad que ha recibido el usuario

Precondiciones: Usuario identificado

Postcondiciones:

Salida: lista(id_solicitudes)=lista con las solicitudes de amistad

Operación: ver_peticiones_pendientes(Id_usuario)

Responsalibidades: Obtiene la lista de solicitudes de amistad que ha realizado el usuario

Precondiciones: Usuario identificado

Postcondiciones:

Salida: lista(id_peticiones)=lista con las peticiones de amistad

Operación: aceptar_solicitud(id_usuario,id_contacto)

Responsalibidades: Acepta la petición de amistad

Precondiciones: Usuario identificado, es contacto del usuario

Postcondiciones: id_contacto =2, id_usuario=2

Salida:

4.6 Realizar solicitud

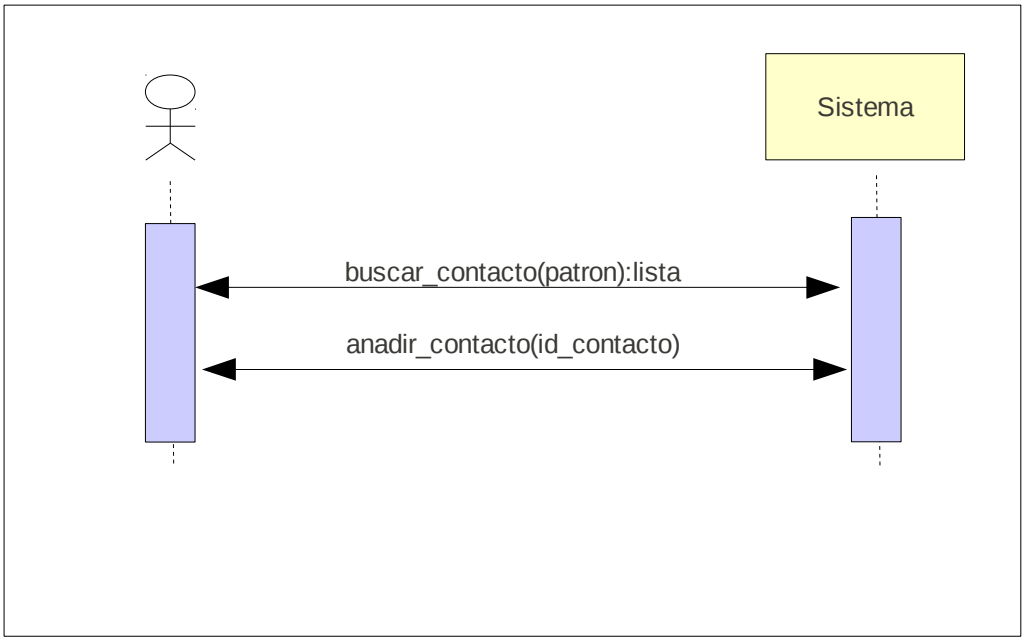


Figura 11. Diagrama de Secuencia de Realizar Solicitud

4.6.1 Contratos

Operación: buscar_contacto(patron)

Responsalibidades: Obtiene la lista de empresas cuyos datos coincidan con el patrón de búsqueda.

Precondiciones: Usuario identificado

Postcondiciones:

Salida: lista(empresas)=lista de empresas que contienen el patrón de búsqueda en algunos de sus campos.

Operación: anadir_contacto(Id_contacto)

Responsalibidades: Realiza una petición de amistad a la empresa

Precondiciones: Usuario identificado, la empresa existe

Postcondiciones: Id_contacto=0, id_usuario=1

Salida:

4.7 Registrar Venta

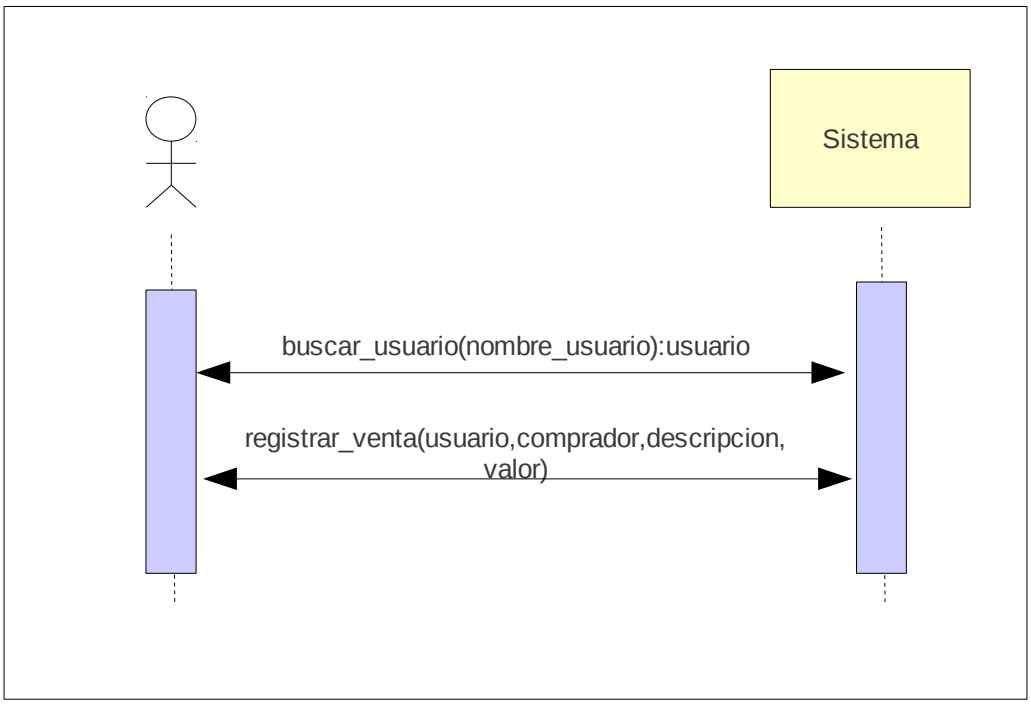


Figura 12. Diagrama de Secuencia de Registrar Venta

4.7.1 Contratos

Operación: buscar_usuario(nombre_usuario)

Responsalibidades: Obtiene el usuario si existe

Precondiciones: Usuario identificado

Postcondiciones:

Salida: usuario

Operación: registrar_venta(usuario,descripcion,venta)

Responsalibidades: Registra una venta a ese cliente

Precondiciones: Usuario identificado, cliente existente,valor válido

Postcondiciones: Se ha registrado la venta

Salida:

4.8 Valorar Compra

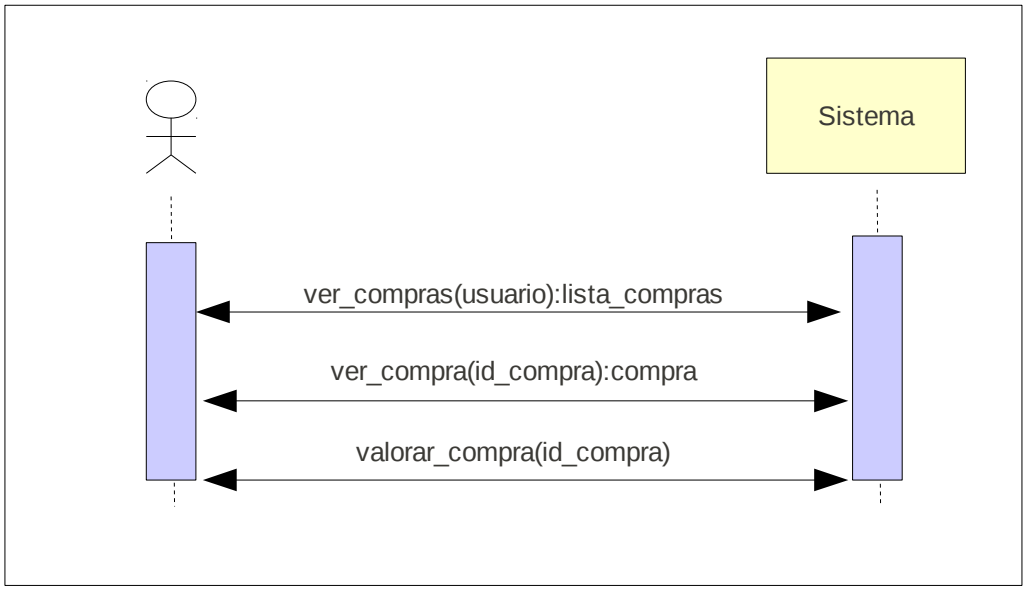


Figura 13. Diagrama de Secuencia de Valorar Compra

4.8.1 Contratos

Operación: ver_compras(usuario)

Responsalibidades: Obtiene las compras del usuario

Precondiciones: Usuario identificado

Postcondiciones:

Salida: lista(compras)

Operación: ver_compra(id_compra)

Responsalibidades: Obtiene los datos de esa compra concreta.

Precondiciones: Usuario identificado, compra existente

Postcondiciones:

Salida: compra

Operación: valorar_compra(valoracion)

Responsalibidades: Valora esa compra con un número del 1 al 5

Precondiciones: Usuario identificado, compra existente,valoración es un número del 1 al 5

Postcondiciones: valoracion(campo)=valoracion que introduce el usuario

Salida:

4.9 Ver Ventas

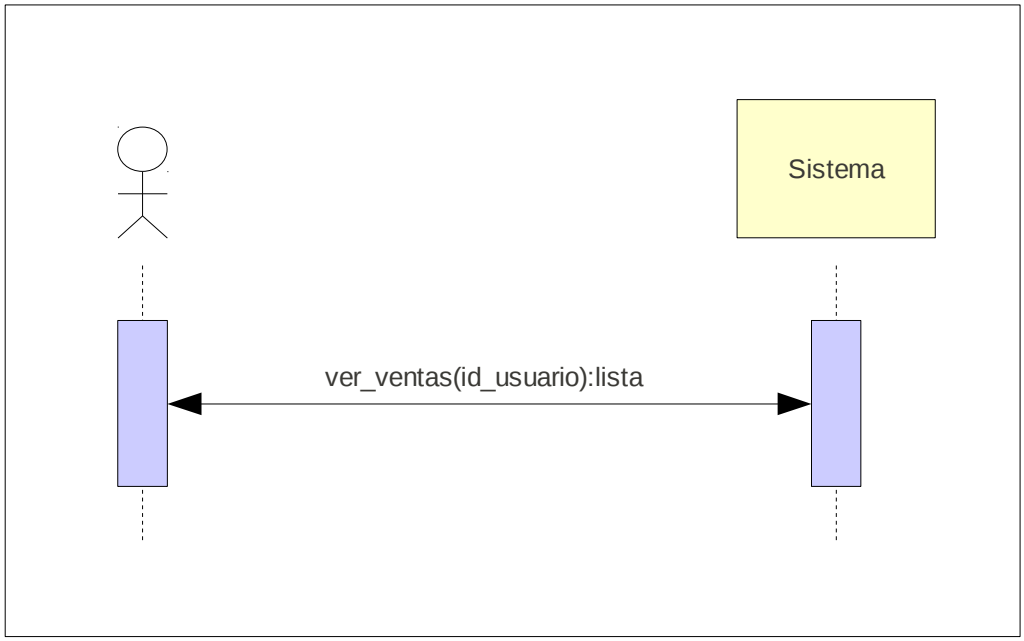


Figura 14. Diagrama de Secuencia de Ver Ventas

4.9.1 Contratos

Operación: ver_ventas(Id_usuario)

Responsalibidades: Obtiene los datos de las ventas de esa empresa

Precondiciones: Usuario identificado

Postcondiciones:

Salida: lista(ventas)

4.10 Ver Balance

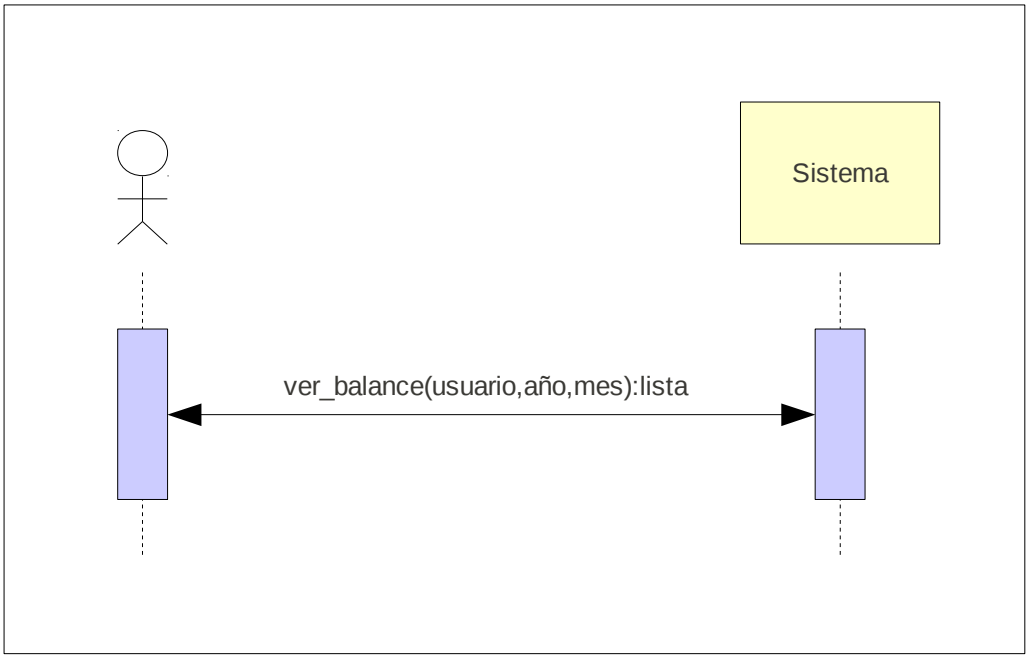


Figura 15. Diagrama de Secuencia de Ver Balance

4.11.1 Contratos

Operación: ver_balance(usuario,año,mes)

Responsalibidades: Obtiene las compras y ventas del periodo seleccionado por el usuario

Precondiciones: Usuario identificado, año y mes válido

Postcondiciones:

Salida: lista(compras): lista con las compras y ventas del usuario

4.12 Ver Valoraciones Ajenas (patrón)

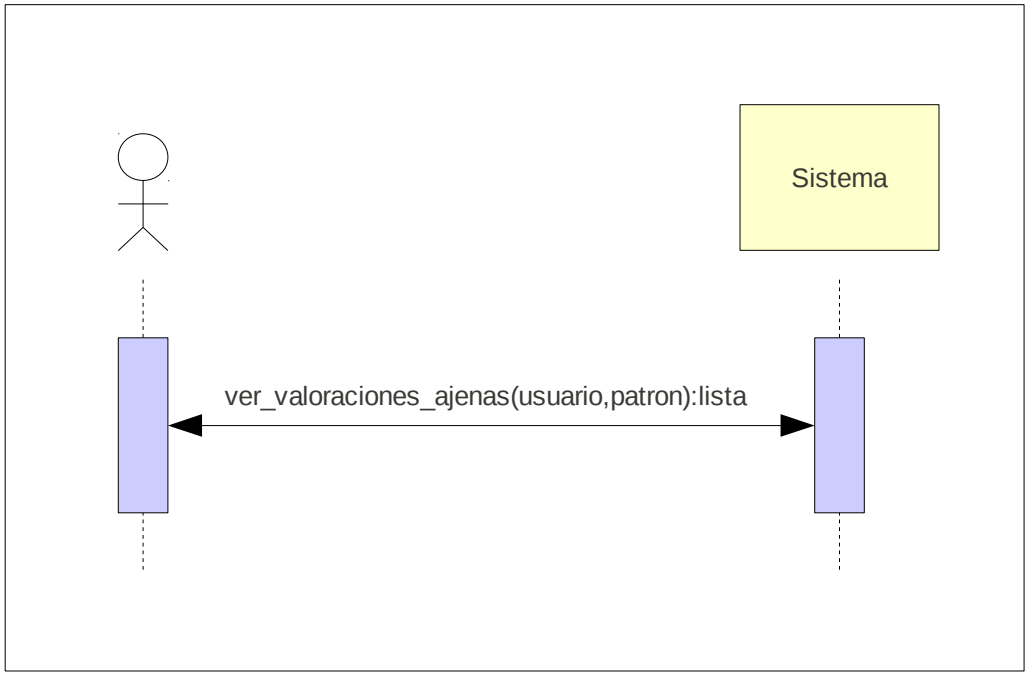


Figura 16. Diagrama de Secuencia de Ver Valoraciones Ajenas

4.12.1Contratos

Operación: ver_valoraciones_ajenas(usuario,patron)

Responsalibidades: Obtiene las empresas que coinciden con el patrón de búsqueda y las valoraciones medias que han recibido de los usuarios del sistema

Precondiciones: Usuario identificado

Postcondiciones:

Salida: lista(empresas)

4.13 Ver Valoraciones Ajenas (sector y provincia)

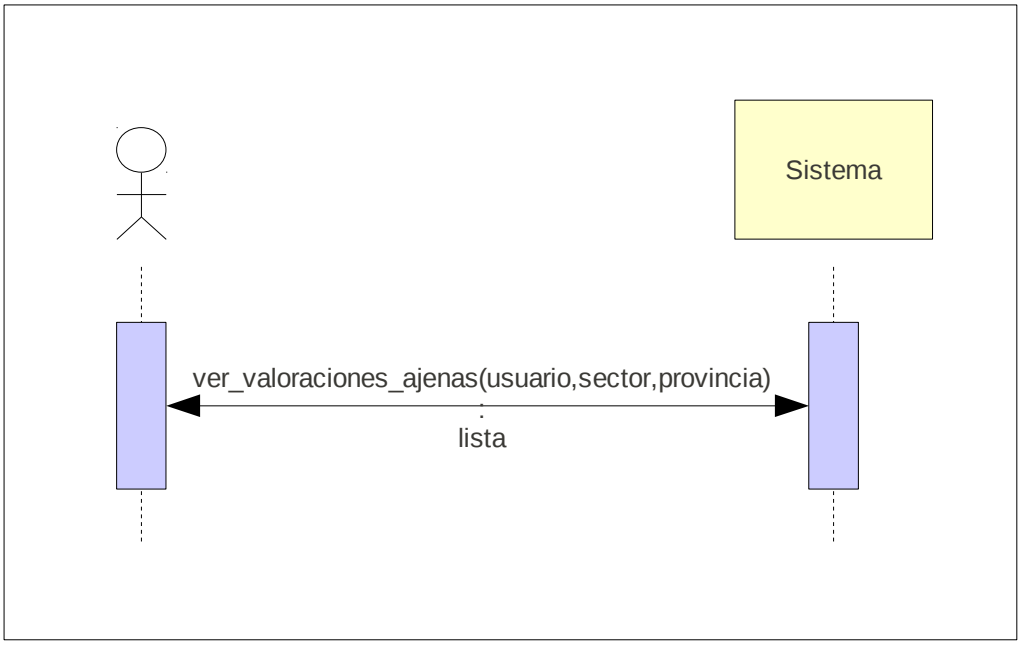


Figura 17. Diagrama de Secuencia de Ver Valoraciones Ajenas

4.13.1 Contratos

Operación: ver_valoraciones_ajenas(usuario,sector,provincia)

Responsalibidades: Obtiene las empresas del sector y provincia seleccionadas y las valoraciones medias que han recibido de los usuarios del sistema

Precondiciones: Usuario identificado

Postcondiciones:

Salida: lista(empresas)

4.14 Ver Valoraciones Contactos (patrón)

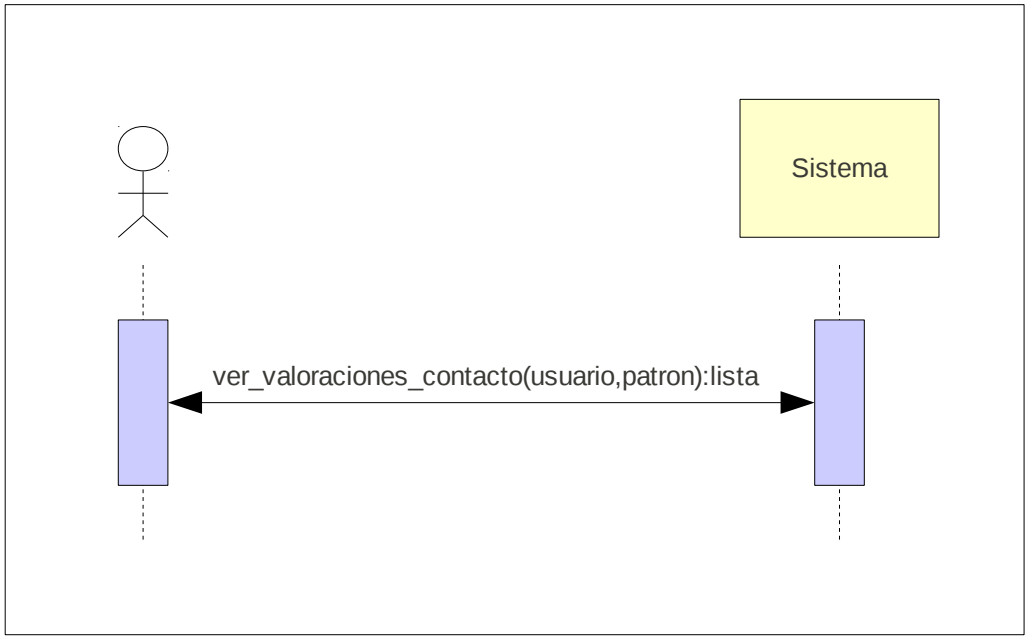


Figura 18. Diagrama de Secuencia de Ver Valoraciones Contactos

4.14.1 Contratos

Operación: ver_valoraciones_contacto(usuario,patron)

Responsalibidades: Obtiene las empresas que coinciden con el patrón de búsqueda y las valoraciones medias que han recibido de los contactos de la empresa

Precondiciones: Usuario identificado

Postcondiciones:

Salida: lista(empresas)

4.14 Ver Valoraciones Contactos (sector y provincia)

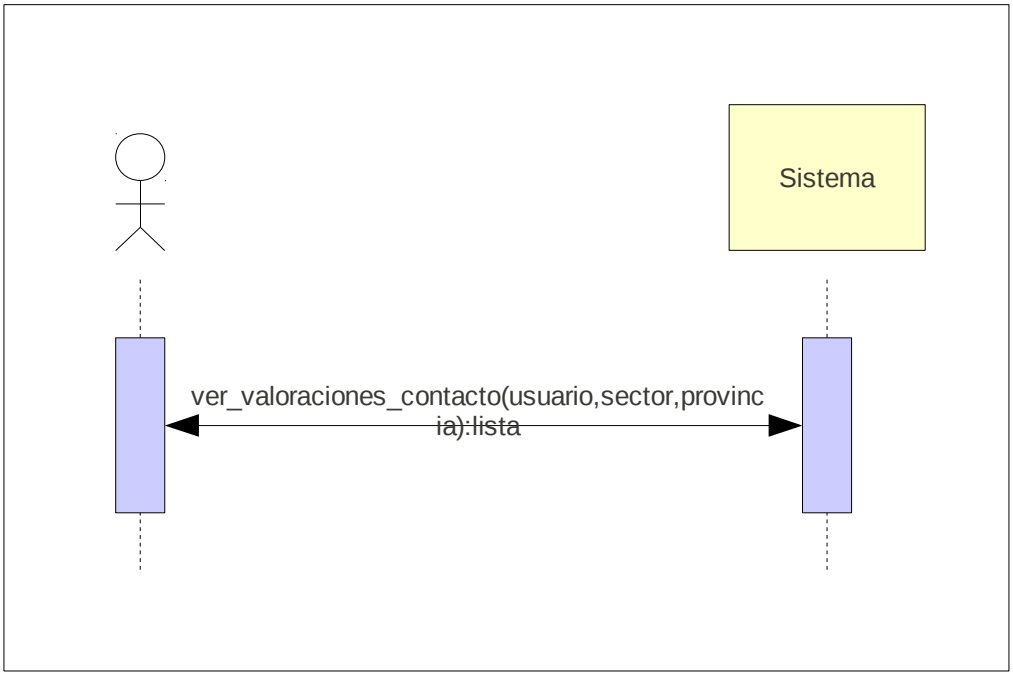


Figura 18. Diagrama de Secuencia de Ver Valoraciones Contactos

4.14.1 Contratos

Operación: ver_valoraciones_contacto(usuario,sector,provincia)

Responsalibidades: Obtiene las empresas del sector y provincia seleccionadas y las valoraciones medias que han recibido de los contactos de la empresa

Precondiciones: Usuario identificado

Postcondiciones:

Salida: lista(empresas)

4.15 Alta usuario

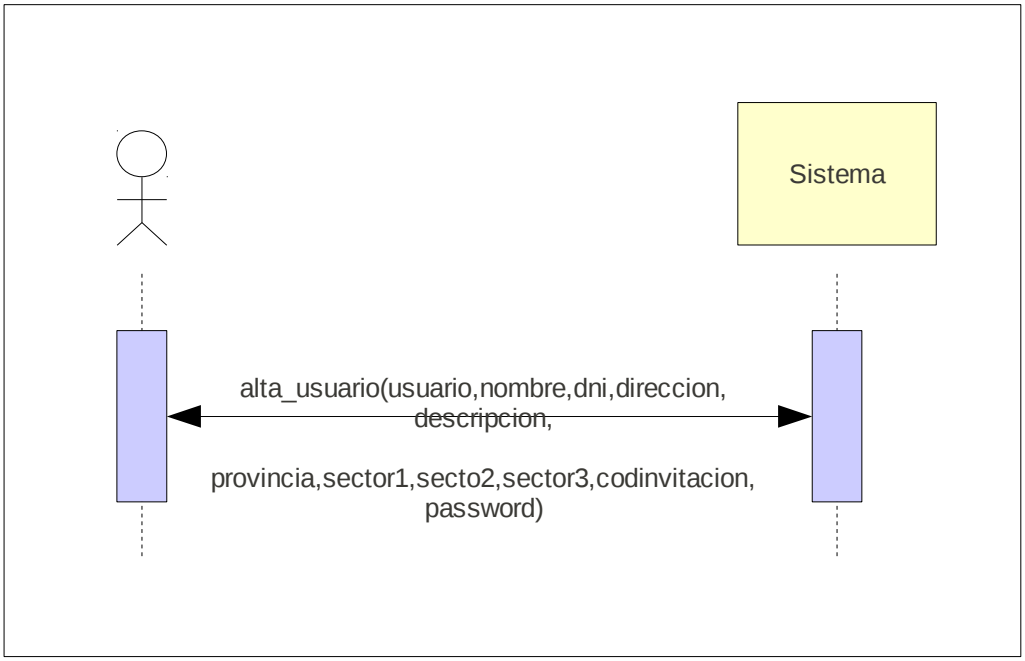


Figura 19. Diagrama de Secuencia de Alta Usuario

4.16 Contratos

Operación:

alta_usuario(usuario,nombre,dni,direccion,descripcion,provincia,sector1,secto r2,sector3,codinvitacion,password)

Responsalibidades: Da de alta un usuario y envía un email con la contraseña

Precondiciones: los datos son válidos

Postcondiciones: Se ha dado de alta un usuario

Salida:

CAPÍTULO 5. ARQUITECTURA

5.1 Planteamiento inicial

Un punto clave a la hora de iniciar un nuevo proyecto software es elegir una arquitectura adecuada para el sistema que ayude a las labores de desarrollo y mantenimiento del software.

La empresa no tenía claro que tipo de arquitectura seguir ni el lenguaje de programación a usar pero deseaban realizar la aplicación usando programación orientada a objetos para comprobar si este tipo de metodología podría aplicarse a alguno de sus proyectos.

5.2 Arquitectura General del Sistema

Con los requisitos establecidos por la empresa se decidió seguir una arquitectura que combinara los siguientes estilos arquitectónicos:

5.2.1 Patrón en 3 capas:

El objetivo de este estilo de programación es separar en tres capas la aplicación de modo que los cambios en una capa afecten lo menos posible a las otras.

La **capa de presentación** es la interfaz que se le muestra al usuario y en la que éste puede interactuar con el sistema.

La **capa de negocio** establece la lógica del programa. Recibe las peticiones del usuario y se comunica con el nivel de datos para que le ofrezca los datos

que necesita para enviarlos como respuesta a la capa de presentación.

Por último la **capa de datos** es la encargada de obtener los datos del sistema de almacenamiento elegido (distintos SGBDs, ficheros,...) y los entrega a la capa de negocio.

5.2.2 Patrón OO:

Siguiendo este patrón las entidades del mundo real tiene su correspondiente "objeto" en el sistema. Según Wikipedia los objetos son:

> "*entidades que combinan estado (atributo), comportamiento (método) e identidad:*
>
> - *El estado está compuesto de datos, será uno o varios atributos a los que se habrán asignado unos valores concretos (datos).*
> - *El comportamiento está definido por los procedimientos o métodos con que puede operar dicho objeto, es decir, qué operaciones se pueden realizar con él.*
> - *La identidad es una propiedad de un objeto que lo diferencia del resto"*

5.2.3 Patrón Modelo -Vista-Controlador

Para ayudar todavía más a la separación entre las capas se utiliza este patrón que crea una nueva capa intermedia cuya responsabilidad es manejar la comunicación entre vista y modelo de manera que evite lo máximo posible la programación en la capa de presentación. Este nuevo nivel y relaciones pueden verse en la figura X.

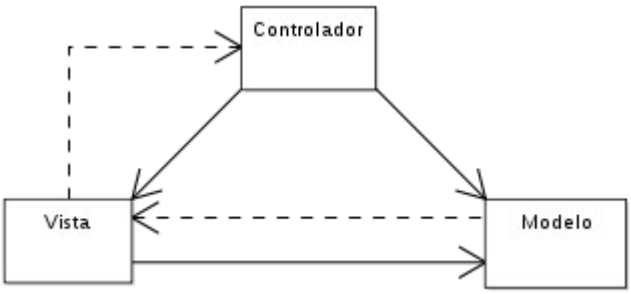


Figura 20. Esquema del patrón MVC

Su funcionamiento según Wikipedia es el siguiente:

"Aunque se pueden encontrar diferentes implementaciones de MVC, el flujo que sigue el control generalmente es el siguiente:

1. *El usuario interactúa con la interfaz de usuario de alguna forma (por ejemplo, el usuario pulsa un botón, enlace, etc.)*

2. *El controlador recibe (por parte de los objetos de la interfaz-vista) la notificación de la acción solicitada por el usuario. El controlador gestiona el evento que llega, frecuentemente a través de un gestor de eventos (handler) o callback.*

3. *El controlador accede al modelo, actualizándolo, posiblemente modificándolo de forma adecuada a la acción solicitada por el usuario (por ejemplo, el controlador actualiza el carro de la compra del usuario). Los controladores complejos están a menudo estructurados usando un patrón de comando que encapsula las acciones y simplifica su extensión.*

4. *El controlador delega a los objetos de la vista la tarea de desplegar la interfaz de usuario. La vista obtiene sus datos del modelo para generar la interfaz apropiada para el usuario donde se refleja los cambios en el modelo (por ejemplo, produce un listado del contenido del carro de la compra). El modelo no debe tener conocimiento directo sobre la vista. Sin embargo, se podría utilizar el patrón Observador para proveer cierta indirección entre el modelo y la vista, permitiendo al modelo notificar a los interesados de cualquier cambio. Un objeto vista puede registrarse con el modelo y esperar a los cambios, pero aun así el modelo en sí mismo sigue sin saber nada de la vista. El controlador no pasa objetos de dominio (el modelo) a la vista aunque puede dar la orden a la vista para que se actualice. Nota: En algunas implementaciones la vista no tiene acceso directo al modelo, dejando que el controlador envíe los datos del modelo a la vista.*

5. *La interfaz de usuario espera nuevas interacciones del usuario, comenzando el ciclo nuevamente."*

5.3 Elección tecnológica

5.3.1 Capa de negocio

Debido a la naturaleza del proyecto y a los usuarios potenciales de la aplicación, era evidente la necesidad de seleccionar un lenguaje que permitiera la creación de aplicaciones web.

Inicialmente, la empresa mostró preferencia por realizar la aplicación en PHP ya que otros proyectos ya se habían realizado utilizando este lenguaje. Otra opción que se barajó fue utilizar JsP para realizar la aplicación.

Por lo tanto se realizó un pequeño estudio, que se detalla a continuación, sobre estos lenguajes y adaptabilidad al proyecto

Java/JSP

Varios factores nos decantaban hacia esta opción. Era un lenguaje conocido por el alumno ya que lo había utilizado en varias asignaturas a lo largo de la carrera. El proyecto de Ingeniería del Software se hizo utilizando Jsp y por lo tanto era una baza segura.

Además existen otros detalles técnicos interesantes como:

- Portabilidad, se puede ejecutar en cualquier sistema operativo que tenga la máquina virtual de Java instalada. Esto es interesante ya que la máquina en la que se programa la aplicación no suele ser la misma en la que al final se aloja.
- Cada JSP se ejecuta en una hebra diferente, por lo que no es necesario volver a cargar cada vez que el cliente hace una petición.
- Se integra con los módulos de Java

Desventajas

- Los hostings son más díficiles de encontrar y más caros como se ve en la *tabla 3* (la mayoría de los paquetes económicos ofrece siempre el mismo paquete Apache+mysql+PHP)

Hosting	JSP	PHP
www.hospedaxes.com	44.95€/año	26.95€/año
www.astrahosting.com	12.50€/año	20.00€/año

Tabla 3. Comparativa de precios según el lenguaje de programación

- Si se usa servidor propio es más difícil de configurar que uno normal.

PHP

Como ya se ha mencionado la empresa ya había realizado varias aplicaciones en este lenguaje y ofrecía formación al alumno en este aspecto.

Alguna de las ventajas de este lenguaje es que contiene muchísima documentación, librerías, frameworks... además, es bastante sencillo de aprender. Se integra como módulo de Apache, y es aceptado por la mayoría de los hostings, a precios muy baratos.

Desventajas

- Es un lenguaje tan extendido que a veces se hace difícil encontrar código hecho de calidad.
- No es un lenguaje orientado a objetos puro, por lo que muy comúnmente encontraremos código poco estructurado

La elección

Finalmente nos decantamos por PHP ya que este proyecto permitiría al alumno aplicar los conocimientos de POO adquirida a lo largo de estos últimos cursos a un lenguaje diferente.

El hosting también fue un factor que nos decantó por la segunda opción ya que la empresa ya tenía servidores propios preparados para PHP pero no para JSP

¿Framework o no?

Una vez elegido el lenguaje, el director de proyecto, Germán, sugirió que se realizara una pequeña investigación sobre frameworks y su posible utilización para este caso.

Existen múltiples frameworks para PHP y cada uno usa una metodología distinta por lo que nos centramos en CakePHP[1] que tenía la ventaja de ser de los más sencillos y fáciles de instalar.

Introducción ¿Qué es?

Es un framework para el desarrollo de aplicaciones web basadas en el lenguaje de programación PHP. Se basa en dos patrones de programación:

- ActiveRecord: Según este patrón cada fila de cada tabla de la BBDD se engloba en una clase, es decir que cada fila es una instancia de una clase del lenguaje de programación usado.
- MVC(Modelo-Vista-Controlador): Patrón que permite separar un programa en

1 http://cakephp.org/

tres partes bien diferenciadas: datos de la aplicación, interfaz de usuario y lógica de control

Estructura de petición simple

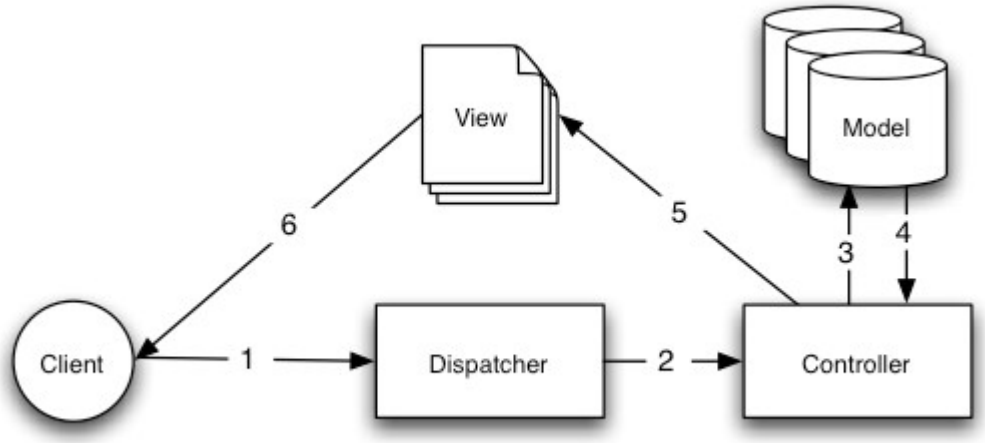


Figura 21. Estructura MVC de CakePHP

CakePHP utiliza la estructura básica mostrada en la Figura 21 que corresponde con el siguiente funcionamiento:

1. Cuando un cliente realiza una petición, ésta es recogida por el Dispatcher.
2. El dispatcher según la ruta desde la que se ha recibido la petición asigna la responsabilidad a un controlador correspondiente.
3. El controlador realiza la lógica correspondiente y obtiene los datos de las entidades del modelo.
4. El modelo devuelve los datos al controlador.
5. Una vez que el controlador tiene listos los datos de la petición se los pasa a la vista correspondiente para presentárselos al usuario (la vista puede ser HTML,PDF..)

Estructura de carpetas y convenciones

El buen funcionamiento del framework se basa en la estructura de directorios utilizada y en las convenciones en cuanto a la nomenclatura de archivos,..

Estructura

- cake : Aquí se encuentran todos los archivos necesarios para el funcionamiento del framework
- app : Aquí se colocarán los archivos de la aplicación que desarrollemos
- vendors**:** En esta carpeta se añadirá el software de terceros

Convenciones

Al usar estas convenciones, CakePHP es capaz de relacionar todas las diferentes capas de la aplicación y así facilitar la tarea del programador, además de ayudar a mantener una estructura clara

Los nombres de las clases(Modelos) están en singular y con CamelCase(Cada primera letra Mayuscula). Ejemplo: UsuarioEmpresa, UsuarioAdministrador

Los nombres de las tablas correspondientes a los modelos están en plural y usando guión bajo. Ejemplo: usuarios_empresas

Los nombres de los controladores son plurales, CamelCased y terminan en “Controller”. La primera accion del controlador debe ser index(). Ejemplo: UsuariosPersonasController

Las vistas se escriben igual que las acciones del controlador pero usando guion bajo para separar las palabras y tienen todas extensión ctp. Ejemplo: get_ready.ctp.

Ventajas de usar un framework

- Permite programar más rápido.
- Funcionamiento probado (por la comunidad).
- Soporte y documentación (en muchos casos).
- Ayuda a escribir código más limpio y ordenado.
- Ayuda a la separación en capas (Modelo, Negocio, Vista..).

Desventajas

- Incompatibilidades entre código (framework y código de terceros).
- Tiempo inicial de aprendizaje (convenciones).
- Parte del código no es conocido (puede ser difícil adaptarlo).
- Dependiente del soporte .
- Problemas para desarrollar diseños muy específicos o con tecnologías muy concretas.
- Aumenta el peso de las aplicaciones.

Elección

Al ser uno de los objetivos prioritarios del alumno aprender sobre todos los aspectos del diseño e implementación del desarrollo de una aplicación se decidió no utilizar ningún framework. Su uso hace que algunas partes de la implementación se abstraigan a un nivel que no interesa en este proyecto (por ejemplo, no es necesario saber cómo CakePHP obtiene los datos de la base de datos, solo es necesario saber cómo llamar a la función adecuada y ese concimiento no podrá ser exportado si se utiliza otro framework distinto). Además las formalidades necesarias a aprender al principio podrían retrasar bastante el proyecto respecto a las fechas planificadas.

5.3.2 Capa de datos

Por un lado, el SGBD más utilizado junto a PHP es MySQL[2]. Su instalación cómo módulo del servidor web Apache le hace el candidato mas propicio. Permite la creación de bases de datos relacionales, es libre, gratuito y cuenta con herramientas de apoyo gratuitas también.

Por otro lado, ya estábamos familiarizados con él y por lo tanto fue necesario pensarlo mucho.

5.3.3 Capa de presentación

Al ser una tecnología web estaba claro la necesidad de utilizar HTML que junto a las Hojas de estilos son una combinación sencilla e intuitiva.

Además para la creación y modificación de imágenes se utilizó una versión de evaluación de Photoshop[3].

5.3.4 Herramientas de desarrollo

Para trabajar el código se ha utilizado la plataforma de desarrollo Eclipse[4].

En cuanto a la base de datos primeramente se utilizó PHPMyAdmin[5] para crear la estructura. Más tarde para realizar pruebas y llamar a los procedimientos almacenados se utilizó MySQL Query Browser ya que su usabilidad era mejor.

2 http://www.mysql.com/
3 https://www.adobe.com
4 http://www.eclipse.org/
5 http://www.phpmyadmin.net

Para crear el esquema de la base de datos se utilizó el paquete MySQL Workbench[6]

5.3.5 Herramientas para documentación

Se ha elegido la suite ofimática OpenOffice[7] para crear todos los documentos necesarios para redactar el proyecto. También se ha utilizado la aplicación GanttProject[8] para la realización del diagrama de Gantt.

5.3.6 Librerias externas

- PHPMailer[9] ha sido utilizado para enviar los emails.
- Simpletest[10] ha sido usado para realizar las pruebas unitarias.

5.3.7 Otros

- Plugin de Firefox FireFTP[11] para subir los archivos al servidor de la empresa en la que se encuentra alojada la aplicación.
- Plugin de Firefox FireBug[12] para la búsqueda de errores en la interfaz gráfica.

6 http://dev.mysql.com/downloads/
7 http://es.openoffice.org/
8 http://www.ganttproject.biz/
9 http://sourceforge.net/projects/phpmailer/
10 http://www.simpletest.org/
11 https://addons.mozilla.org/es-ES/firefox/addon/684/
12 https://addons.mozilla.org/es-ES/firefox/addon/1843/

5.4 Arquitectura Final del Sistema

Después de mucho investigar (hay tanta información sobre PHP que a veces resulta difícil encontrar algo de calidad), leer sobre el funcionamiento de los frameworks, orientación a objetos en php,.. nos hizo decantarnos el esquema de funcionamiento basado en MVC que se muestra en la figura 22.

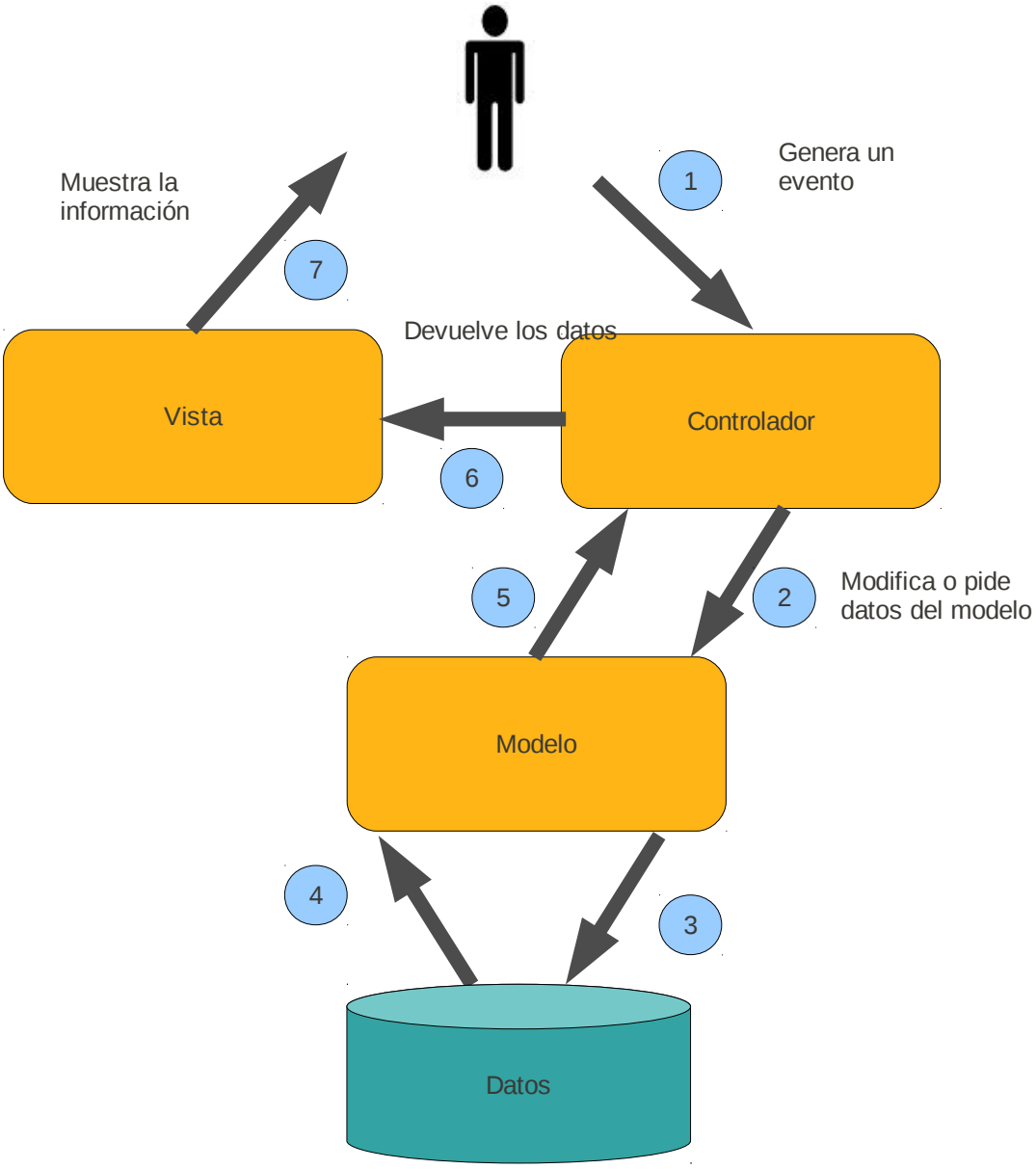


Figura 22. Esquema de funcionamiento del sistema

Más concretamente la arquitectura funciona según el esquema de la figura 23.

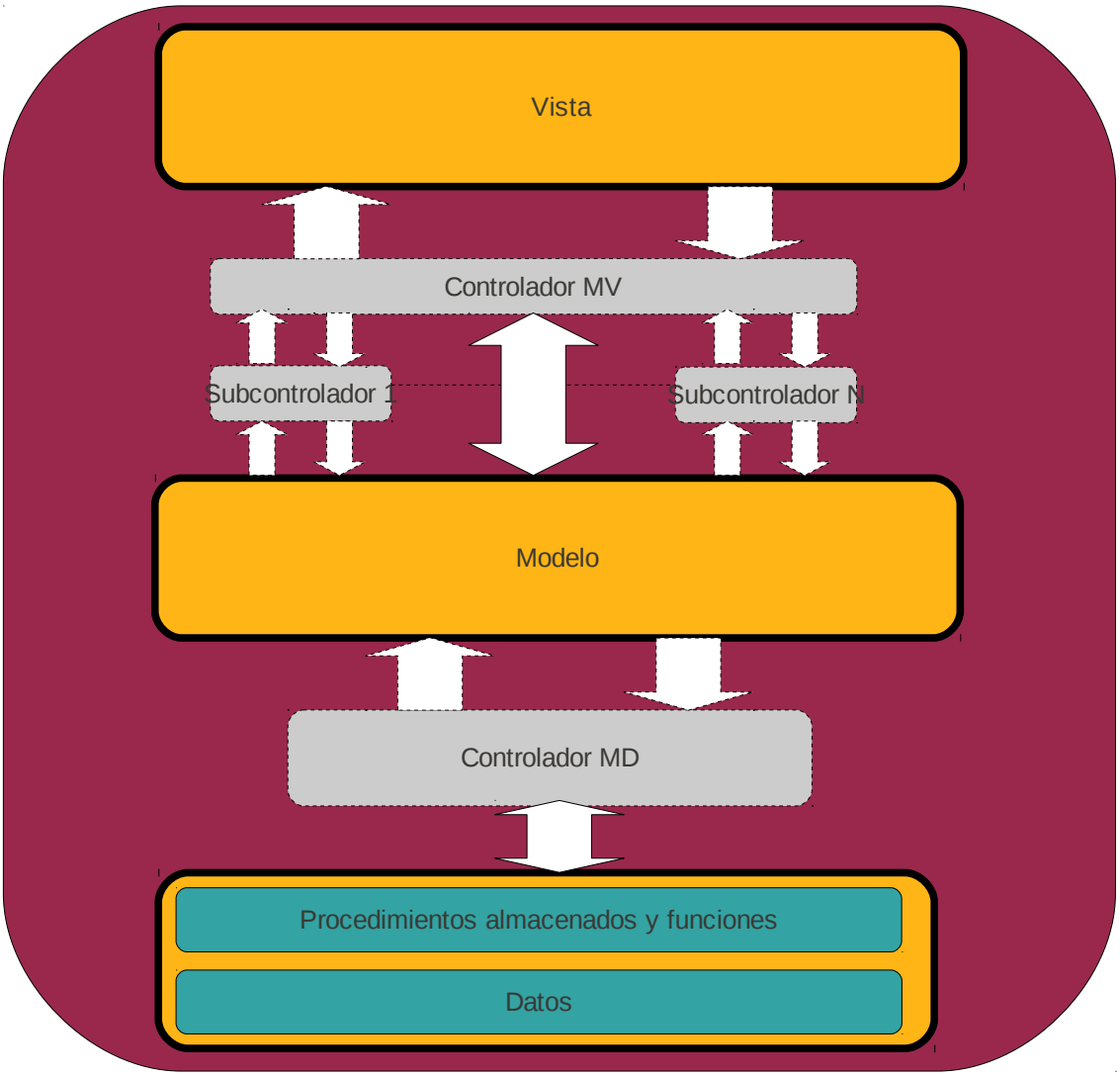


Figura 23. Arquitectura por capas del Sistema

El funcionamiento es el siguiente:

Paso 1: Un usuario genera un evento (pulsa un botón, enlace,...) en la vista.

Paso 2: Es recogido por el Controlador Principal. Éste puede pedir los datos al modelo directamente o delegar el proceso a un subcontrolador.

Paso 3: El modelo se encarga de realizar las operaciones con los datos que pide al controlador MD

Paso 4: El controlador MD simplemente hace las llamadas a los procedimientos.

Paso 5: El procedimiento almacenado obtiene o almacena los datos de la base de datos y devuelve los resultados.

Paso 6: El controlador MD recoge los datos de los procedimientos almacenados y transforma los resultados que le dan estos a un formato tipo tabla para que sean tratados por el modelo más fácilmente.

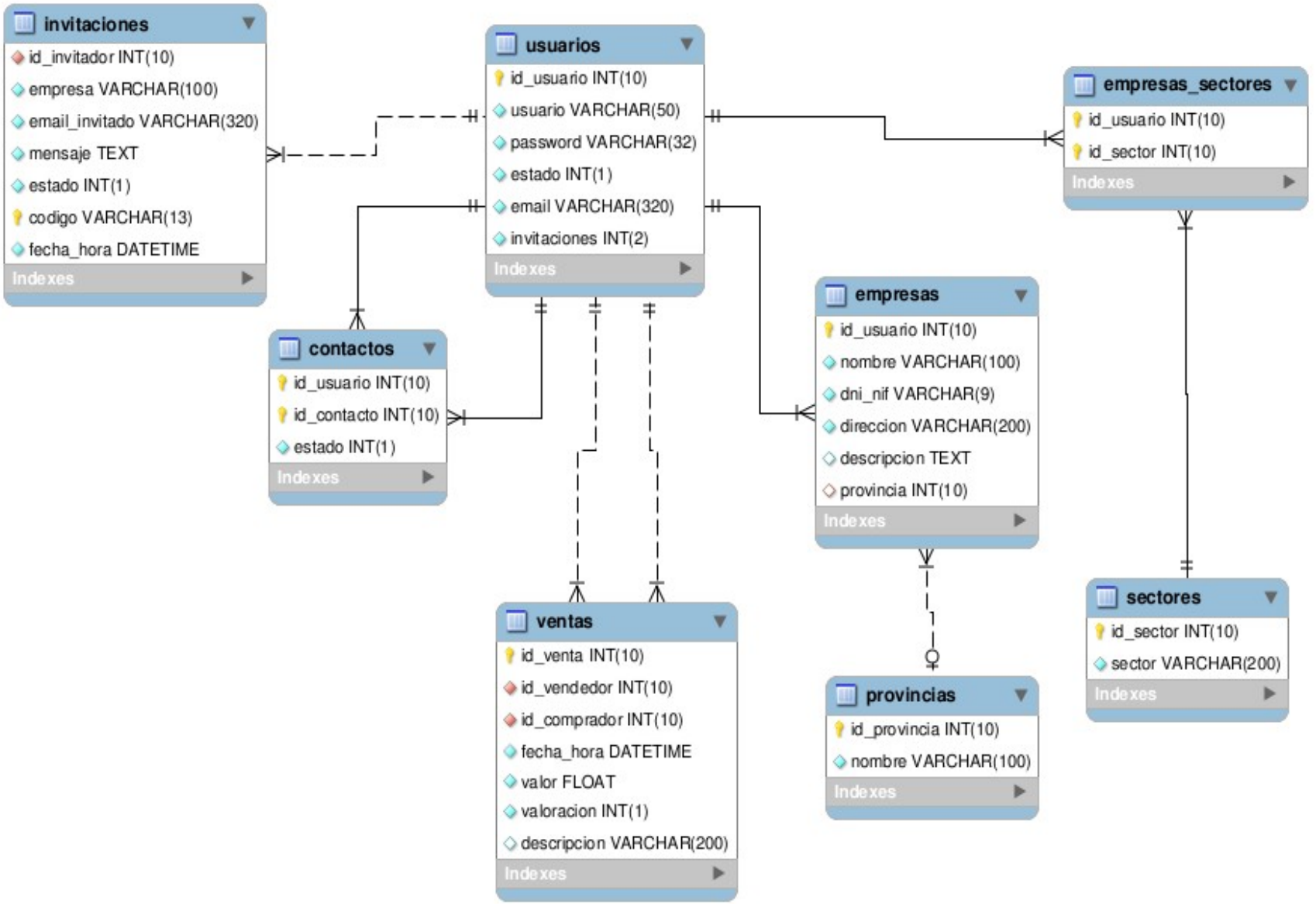
Paso 7: El controlador principal o los subcontroladores recogen los datos del modelo y cargan la vista asociada al evento

CAPÍTULO 6. DISEÑO

6.1 Introducción

Debido a que el proyecto usa tecnologías web en el que influyen factores como el tiempo de sesión, la necesidad de recargar los datos en cada petición, etc, puede resultar complicado hacer diagramas de secuencia. Investigando por Internet uno se da cuenta que la mayor parte de los proyectos en PHP no usan este tipo de diagramas (en muchos caso directamente no documentan nada) y por eso lo mostrado a continuación son esquemas que intentan seguir una metodología parecida pero un poco adaptada al caso concreto que nos ocupa.

6.2 Base de datos



Cada usuario del sistema es una empresa por lo que su id es el mismo. Además cada empresa tiene asociada una provincia y una lista de hasta 3 sectores.

Por otro lado cada empresa tiene en sus contactos otras empresas usuarias del sistema. Según el valor del estado cada relación se encuentra en una fase.

Por cada relación se guardan 2 registros.
Si el usuario 1 realiza una petición de amistad al usuario 2 estos serian los registros guardados.

id_usuario	id_contacto	Estado
1	2	1
2	1	0

El primer registro indica que la relación ha sido aceptada por el usuario 1, el segundo registro indica en cambio que el usuario 2 no se ha pronunciado todavía acerca de esa petición de amistad.
Si el usuario 2 decide aceptar la relación de amistad ambos estados se encontrarán a 2.

id_usuario	id_contacto	Estado
1	2	2
2	1	2

Sin embargo, si rechaza la petición los estados quedarian así.

id_usuario	id_contacto	Estado
1	2	2
2	1	3

Además las empresas se pueden hacer compras y ventas entre ellas. Estas acciones quedan registradas en la tabla ventas. En la que por cada venta se recoge el usuario que realiza la venta, el que ejerce de cliente, la fecha y hora del registro, la descripción de la venta, el valor y la valoración que da el cliente a esta compra.

Por último en la tabla invitaciones se registran los datos de envío de las invitaciones al sistema.

6.3 Estructura de diseño común

Al utilizar la arquitectura planteada anteriormente con los patrones OO y MVC la mayor parte de los diseños siguen la misma estructura. Es por esta razón que primero explicaremos el diagrama de funcionamiento común, figura 24, y después explicaremos el diseño del contrato Añadir Invitacion con ejemplos de código para que el lector pueda comprender cómo el formato de los diseños se aplica a la implementación.

6.3.1 Esquema común.

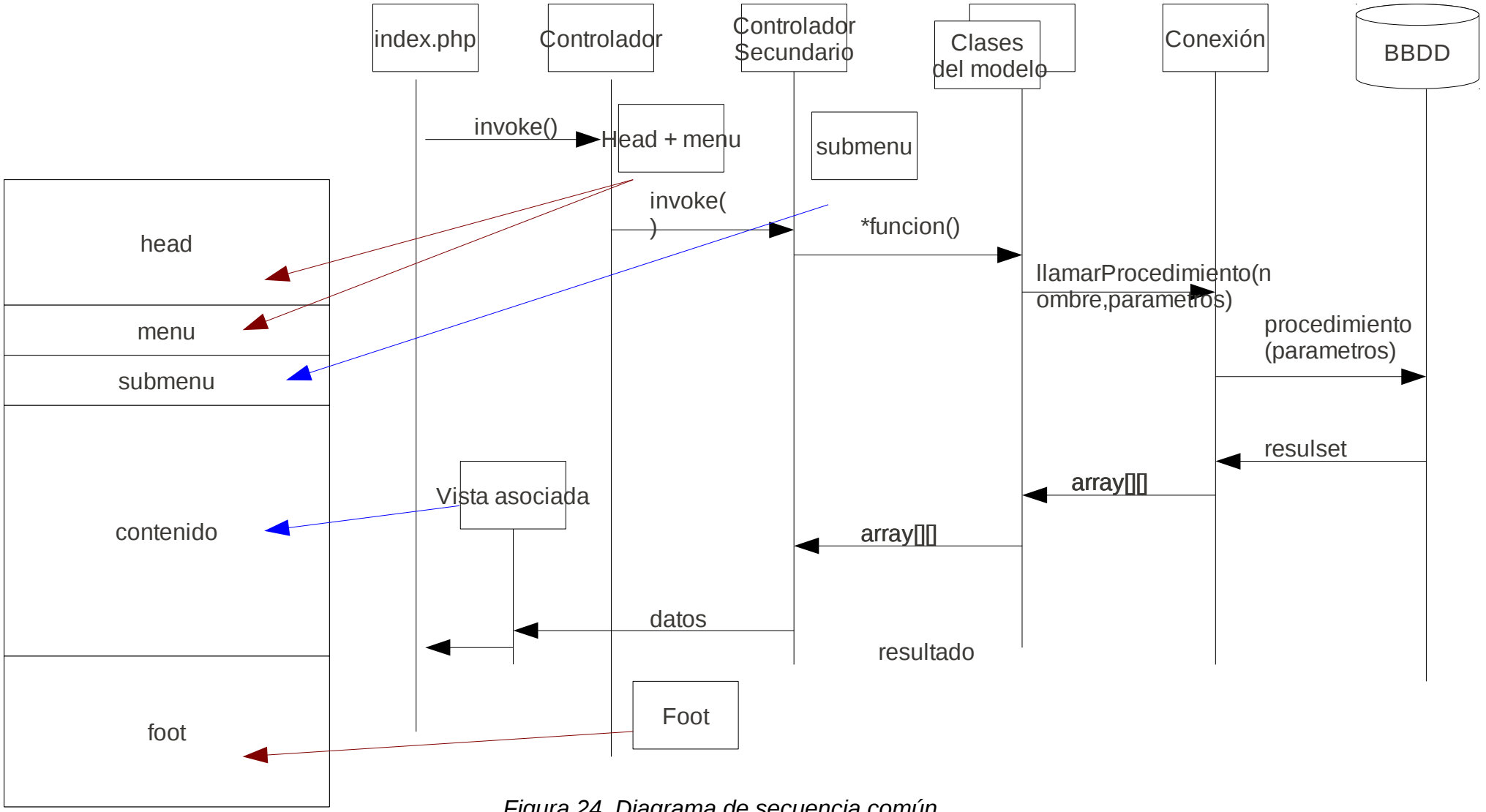


Figura 24. Diagrama de secuencia común.

En el diagrama vemos cómo el usuario trabaja siempre con la página index que únicamente envía las peticiones al controlador principal. Este es el encargado de añadir el head y el foot a la página de salida. Por otro lado, el controlador delega responsabilidades en los subcontroladores que añaden los submenús asociados a esa sección y se encargan de crear las clases del modelo y usar sus métodos para obtener los datos que posteriormente se mostrarán en la vista.

En la *tabla 4* podemos ver el formato de diseño que se ha planteado para no repetir continuamente los mismos diagramas y explicaciones.

Evento	Qué produce el evento (botón, menú,etc)
Subcontrolador asociado	Indica a qué subcontrolador se le delegan las responsabilidades, si es que existe.
$_GET	Nombre de las variables pasadas por el método GET que se utilizan en este caso y que indican la subsección en que se ha producido el evento
$_POST	Nombre de las variables pasadas por el método POST que se utilizan en este caso que indican que objeto ha producido el evento
Modelo	Se explica a qué funciones de las clases del modelo se llama
Procedimientos almacenados	Prodecimientos utilizados.
Vista asociada	Vistas incluidas en la página index

Tabla 4. Tabla-diseño genérico

Para comprender mejor como se puede plasmar los datos de la tabla-diseño mostraremos a continuación un ejemplo concreto, el contrato Añadir Invitacion del caso de uso Gestionar Invitaciones.

6.3.2 Un ejemplo: Contrato Añadir invitación

La *tabla 5* muestra el diseño del contrato Añadir Invitación

Evento	El usuario que está en la sección de invitaciones rellena el formulario y pulsa el botón enviar invitación
Subcontrolador asociado	Controlador de perfil
$_GET	$_GET['menu']=perfil, $_GET['accion']=ver_invitaciones
$_POST	$_POST['enviar_invitacion'], $_POST[empresa],$_POST[email], $_POST[cuerpo_email]
Modelo	Se llama a la función de usuario anadir_invitacion, pasando los datos del formulario y el nombre de la empresa usuaria.
Procedimientos almacenados	anadir_invitacion
Vista asociada	Invitaciones.php

Tabla 5. Tabla-diseño Añadir Invitación

El usuario trabaja siempre sobre la página index.php y genera los eventos en esta: pulsar botones, enlaces,...

El contenido de index.php es este:

```
include("Controladores/controlador.php");
Incluye la clase para poder ser utilizada

$controlador = new Controlador();
Crea una instancia del controlador

$controlador->invoke();
Invoca el controlador
```

El controlador tiene un método constructor que incluye todas las clases necesarias para poder trabajar con el modelo.

```
public function __construct(){
      require("Conexion/conexion.php");
      require("Modelo/usuario.php");
      require("Modelo/empresa.php");
      require("Modelo/lista_contactos.php");
      require("Modelo/venta.php");
      require("phpmailer/class.phpmailer.php");
      require("phpmailer/class.smtp.php");
}
```

Si el usuario está logueado,

```
include("Vistas/estructura_pagina/head.php");
Esta línea incluye el head de la página, que contiene
la estructura inicial del html, es decir, el head,
titulo, body... y el menú principal

Aquí recoge la opción pulsada del menú y delega la
responsabilidad al subcontrolador asociado a esa
sección del menú o sino hay subcontrolador asociado se
incluye la vista  asociada


      switch($_GET['menu']){
      case 'bienvenido':
            include('Vistas/app/bienvenido.php');
      break;
      case 'perfil':

      include("Controladores/perfil_controlador.php");
            $controlador_perfil = new PerfilControlador();
            $controlador_perfil->invoke();
       break;
```

```
        . . .
    include("Vistas/estructura_pagina/foot.php");
```

Aquí se incluye el foot de la página.

Una vez delegado a un subcontrolador la tarea a realizar este añade a la vista el submenú asociado.

```
public function __construct(){

include('Vistas/estructura_pagina/menu_izquierda/perfil.php');
      Constructor del controlador de perfil
```

si la acción ha sido pulsar una opción del submenú esta se recoge en una variable get

```
    switch($_GET['accion']){
        case 'no': case 'ver_perfil':
            $empresa=new empresa($this->id_usuario);
            include('Vistas/app/perfil/perfil.php');
        break;
```

si se trata de un evento dentro de una subsección se recoge mediante una variable post, como es nuestro caso.

```
        case 'ver_invitaciones':
            $usuario=new usuario($this->id_usuario);
            $empresa=new empresa($this->id_usuario);
            if($_POST['enviar_invitacion']){
                $invitado=$_POST['empresa'];
                $email=$_POST['email'];
                $mensaje=$_POST['cuerpo_email'];
                $ok=$usuario->anadir_invitacion($empresa-
                        >get_nombre(),$invitado,
                        $email,$mensaje);
                . . .
```

Como vemos los controladores son los que se encargan de hacer las llamadas

a las clases del modelo

Las clases del modelo si necesitan obtener datos de la base de datos o modificarlos llaman a la clase Conexión, que abre la conexión con la base de datos, llama al procedimiento almacenado que necesite y devuelve los datos en forma de array.

```
$c=new conexion();
$parametros="'".$codigo_invitacion."','".$this->id."','".
$invitado."','".$email."','".$mensaje."','".date('Y-m-d
H:m:s')."'";
$resultado=$c->llamarProcedimiento('anadir_invitacion',
            $parametros,0);
```

Por último, el modelo incluye la vista asociada.

```
    include('Vistas/app/perfil/invitaciones.php');
```

Ésta obtiene los datos de las clases instanciadas en el modelo y construye el HTML que visualizará el usuario

```
<h1>Mi Perfil</h1>
<table class="tabla2">
      <tr>
            <td>Nombre: </td>
            <td><?php echo $empresa->get_nombre(); ?></td>
            <td></td>
      </tr>
      <tr>
            <td>Dni o Nif:</td>
            <td><?php echo $empresa->get_dni_nif(); ?></td>
            <td></td>
      </tr>

      . . .
```

Como podemos observar, aunque todavía queda algo de programación en la vista, es muy sencilla y se basa únicamente en mostrar datos.

6.4 Diseño de los contratos

En este apartado se mostrarán las tablas-diseño de cada contrato de los casos de uso.

6.4.1 Iniciar Sesión

Evento	El usuario introduce usuario y contraseña y pulsa el botón entrar.
Subcontrolador asociado	No
$_GET	$_GET['menu']=bienvenido
$_POST	$_POST['identificar'], $_POST[usuario],$_POST[password]
Modelo	Se comprueba si es un usuario válido llamando al método identificar_usuario pasando las variables usuario y password cifrada con md5. Si esto devuelve un número distinto de -1 se da acceso,sino vuelve a la página de logueo.
Procedimientos almacenados	Existe_usuario(nombre,password)

```
CREATE DEFINER=`root`@`localhost` FUNCTION `existe_usuario`(c varchar(320), p varchar(32)) RETURNS int(1)
BEGIN
 declare passcorrecto int;
 declare valorretorno int;
  select count(*) into passcorrecto from usuarios where usuario = c and password = p and estado=1;
 if(passcorrecto > 0) then
```

select id_usuario into valorretorno from usuarios where usuario = c and estado=1; end if; return valorretorno;	
Vista asociada	Bienvenido.php

Tabla 6. Tabla-diseño de Iniciar Sesión

Paso 1. Evento: El usuario introduce usuario y contraseña y pulsa el botón entrar. Estas variables se pasan por $_POST

Paso 2. Se recarga la página, se crea el controlador y se invoca.

Paso 3. Como este caso de uso no tiene controlador asociado, se trata ahí mismo. Se comprueba si es un usuario válido llamando al método identificar_usuario() de la clase usuario pasando como parámetros las variables recogidas por $_POST.

Paso 4. La clase usuario crea una instancia de Conexión que realiza una petición de datos a la función existe_usuario de la base de datos. Si los datos del usuario son válidos devuelve el id de usuario, que será mantenido a lo largo de la sesión. En otro caso devolvería -1.

Paso 5. El controlador en caso de haber recibido una respuesta positiva crea la variable de sesión S_SESSION['usuario']. Carga el head, muestra la pantalla de bienvenida al sistema y carga por último el foot. Si la respuesta es negativa muestra un mensaje de error.

6.4.2 Ver Perfil

Paso 1. El usuario pulsa en el submenú “Ver perfil”

Paso 2. Se recarga la página, se crea el controlador y se invoca.

Paso 3. Este caso de uso está asociado al controlador del perfil. Por lo tanto, se crea e invoca. Este recoge la acción mediante las variables GET. Crea una nueva instancia de empresa pasándole como parámetro el id de usuario que coincide con el id de la empresa.

Paso 4. La clase empresa comprueba que clase de constructor necesita. Y carga los datos de la empresa. Para ello tiene que obtener los datos de la empresa (procedimiento obtener_empresa), tambien carga la valoración de la empresa(procedimiento obtener_valoracion_empresa) y por último los sectores a los que pertenece (obtener_sectores_empresa)

Paso 5. El controlador de perfil carga la vista perfil.php que recoge los datos de la empresa y los muestra.

Evento	El usuario pulsa en el submenú “Ver perfil”
Subcontrolador asociado	Controlador de perfil
$_GET	$_GET['menu']=perfil, $_GET['accion']=ver_perfil
$_POST	
Modelo	Se crea una instancia de empresa pasándole al constructor el id del usuario que se encuentra en una

	variable de sesión
Procedimientos almacenados	Obtener_empresa(id),obtener_valoracion_empresa(id), obtener_sectores_empresa(id)

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `obtener_empresa`(in id int(10))
BEGIN
    Select e.id_usuario ,e.dni_nif,e.descripcion,e.direccion,e.nombre as 'nombre',p.nombre as 'provincia',p.id_provincia from empresas as e,provincias as p where e.provincia=p.id_provincia and e.id_usuario=id;
END

CREATE DEFINER=`root`@`localhost` PROCEDURE `obtener_valoracion_empresa`(in id int(10))
BEGIN
  Select avg(v.valoracion) as valoracion from empresas as e,ventas as v where e.id_usuario=v.id_vendedor and v.id_vendedor=id and v.valoracion <> 0;
END

CREATE DEFINER=`root`@`localhost` PROCEDURE `obtener_sectores_empresa`(in id int(10))
BEGIN
  Select s.id_sector,s.sector from sectores as s,empresas_sectores as es where es.id_sector=s.id_sector and es.id_usuario=id;
END
```

Vista asociada	Perfil.php

6.4.3 Obtener Invitaciones

Paso 1. Evento: El usuario pulsa en el submenú “Invitaciones”

Paso 2. Se recarga la página, se crea el controlador y se invoca.

Paso 3. Este caso de uso está asociado,al igual que el caso anterior, al controlador del perfil. Por lo tanto, se crea e invoca. Este recoge la acción mediante las variables GET. Crea una nueva instancia de usuario para poder obtener el número de invitaciones que le quedan

Paso 4. El controlador de perfil carga la vista invitaciones.php que recoge el número de invitaciones y muestra el formulario para realizar una nueva si aún quedan

Evento	El usuario pulsa en el submenú “Invitaciones”
Subcontrolador asociado	Controlador de perfil
$_GET	$_GET['menu']=perfil, $_GET['accion']=ver_invitaciones
$_POST	
Modelo	Se obtiene el número de invitaciones disponibles de la clase usuario.
Procedimientos almacenados	obtener_usuario(id)
CREATE DEFINER=`root`@`localhost` PROCEDURE `obtener_usuario`(in id int(10))	

BEGIN Select id_usuario,usuario,estado,email,invitaciones from usuarios where id_usuario=id; END	
Vista asociada	Invitaciones.php

Tabla 8. Tabla-diseño Obtener invitaciones

6.4.4 Ver contactos

Paso 1. Evento: El usuario pulsa en el submenú “Mis contactos”

Paso 2. Se recarga la página, se crea el controlador y se invoca.

Paso 3. Este caso de uso está asociado al controlador de contactos. Se crea e invoca. Este recoge la acción mediante las variables GET. Crea una nueva instancia de lista_contactos, que carga toda la lista de contactos de ese usuario

Paso 4. La lista de contactos obtiene los datos del procedimiento obtener_contactos.

Paso 5. El controlador de contactos carga la vista ver_contactos.php que recoge los datos de la empresa y los muestra.

Evento	Pulsado submenú “Mis Contactos”
Subcontrolador asociado	Controlador de contactos
$_GET	$_GET['menu']=contactos $_GET['accion']=ver_contactos
$_POST	

Modelo	Se crea una lista con todos los contactos del usuario, existen distintos tipos de lista de contactos,
Procedimientos almacenados	obtener_contactos
CREATE DEFINER=`root`@`localhost` PROCEDURE `obtener_contactos`(in id int(10)) BEGIN Select id_contacto from contactos where id_usuario=id and estado=2; END	
Vista asociada	ver_contactos.php

Tabla 9. Tabla-diseño Ver Contactos

6.4.5 Eliminar contacto

Paso 1. Evento: El usuario pulsa eliminar uno de los contactos

Paso 2. Se recarga la página, se crea el controlador y se invoca.

Paso 3. Este caso de uso está asociado al controlador de contactos. Se crea e invoca. Este recoge la acción mediante las variables $_POST, recoge el id pasado en el nombre de contacto

Paso 4. La lista de contactos obtiene los datos del procedimiento obtener_contactos

Paso 5. El controlador carga la vista asociada.

Evento	Pulsado el botón de eliminar un contacto concreto
Subcontrolador asociado	Controlador de contactos
$_GET	$_GET['menu']=contactos $_GET['accion']=ver_contactos
$_POST	$_POST[eliminar_id_contacto]
Modelo	Elimina el contacto de la lista, llamando a la funcón elimminar_contacto
Procedimientos almacenados	eliminar_contacto(id_usuario,id_contacto)
CREATE DEFINER=`root`@`localhost` PROCEDURE `eliminar_contacto`(in usuario int(10),in contacto int(10)) BEGIN delete from contactos where id_usuario=usuario and id_contacto=contacto; delete from contactos where id_usuario=contacto and id_contacto=usuario; END	
Vista asociada	ver_contactos.php

6.4.6 Ver Solicitudes Pendientes& Ver Contactos Pendientes

Paso 1. Evento: El usuario pulsa en el submenú “Contactos pendientes”

Paso 2. Se recarga la página, se crea el controlador y se invoca.

Paso 3. Este caso de uso está asociado al controlador de contactos. Se crea e invoca. Este recoge la acción mediante las variables GET. Se crea una lista de contactos con las peticiones de amistad recibidas y otra con las peticiones realizadas, usando distintas opciones del constructor.

Paso 4. La lista de contactos obtiene los datos del procedimiento obtener_contactos_solicitud y obtener_contactos_pendientes

Paso 5. El controlador de contactos carga la vista asociada.

Evento	Pulsado el menú “Contactos pendientes”
Subcontrolador asociado	Controlador de contactos
$_GET	$_GET['menu']=contactos $_GET['accion']=ver_contactos_pendiente
$_POST	
Modelo	Muestra la lista de contactos pendientes de aceptación y la lista de solicitudes
Procedimientos almacenados	obtener_contactos_solicitud (id_usuario)

	obtener_contactos_pendientes(id_usuario)
CREATE DEFINER=`root`@`localhost` PROCEDURE `obtener_contactos_solicitud`(in id int(10)) BEGIN Select id_contacto from contactos where id_usuario=id and estado=0; END CREATE DEFINER=`root`@`localhost` PROCEDURE `obtener_contactos_pendientes`(in id int(10)) BEGIN Select id_contacto from contactos where id_usuario=id and estado=1; END	
Vista asociada	ver_contactos_pendientes.php

Tabla 11. Tabla-diseño Ver Solicitudes Pendientes

6.4.7 Aceptar o rechazar contacto

Paso 1 y 2. Igual

Paso 3 y 4: Comprueba las variables $_POST para ver si la opción elegida es aceptar o rechazar. Obtiene el id del nombre del botón y llama al método acpetar_contacto o rechazar contacto que cambia el campo estado de la tabla 2 aceptar o 3 rechazar.

Paso 5. Carga la vista asociada.

Evento	Pulsado el botón aceptar contacto
Subcontrolador asociado	Controlador de contactos
$_GET	$_GET['menu']=contactos $_GET['accion']=ver_contactos_pendiente

$_POST	$_POST[aceptar_id_contacto]
Modelo	Llama a la función aceptar_contacto de la clase lista_contactos
Procedimientos almacenados	aceptar_contacto(id_usuario,id_contacto)
CREATE DEFINER=`root`@`localhost` PROCEDURE `aceptar_contacto`(in usuario int(10),in contacto int(10)) BEGIN UPDATE contactos SET estado=2 where id_usuario=usuario and id_contacto=contacto; UPDATE contactos SET estado=2 where id_usuario=contacto and id_contacto=usuario; END	
Vista asociada	ver_contactos_pendientes.php

Tabla 12. Tabla-diseño Aceptar Contacto

Evento	Pulsado el botón rechaza contacto
Subcontrolador asociado	Controlador de contactos
$_GET	$_GET['menu']=contactos $_GET['accion']=ver_contactos_pendiente
$_POST	$_POST[aceptar_id_contacto]
Modelo	Llama a la función rechazar_contacto de la clase lista_contactos
Procedimientos almacenados	rechazar_contacto(id_usuario,id_contacto)
CREATE DEFINER=`root`@`localhost` PROCEDURE `rechazar_contacto`(in usuario int(10),in contacto int(10)) BEGIN	

update contactos set estado=3 where id_usuario=usuario and id_contacto=contacto; END	
Vista asociada	ver_contactos_pendientes.php

Tabla 13. Tabla-diseño Rechazar Contacto

6.4.8 Buscar contacto

Paso 1. Evento: El usuario pulsa en el botón buscar contacto pasándole un patrón de búsqueda

Paso 2. Se recarga la página, se crea el controlador y se invoca.

Paso 3. El controlador de contacto crea una lista de contactos que coincide con el patrón de búsqueda y recoge la información de esas empresas

Paso 4. La lista de contactos obtiene los datos del procedimiento obtener_empresas_busqueda

Paso 5. El controlador de contactos carga la vista asociada.

Evento	Pulsado el botón buscar
Subcontrolador asociado	Controlador de contactos
$_GET	$_GET['menu']=contactos $_GET['accion']=nuevo_contacto
$_POST	$_POST[buscar_contacto]
Modelo	Crea una nueva lista de contactos tipo búsqueda
Procedimientos almacenados	obtener_empresas_busqueda(id_usuario,id_contacto)
CREATE DEFINER=`root`@`localhost` PROCEDURE	

`obtener_empresas_busqueda`(in usuario int(10),in patron varchar(100))
BEGIN
 	
declare ncontactos int;
select count(e.id_usuario) into ncontactos from empresas as e,contactos as c where e.id_usuario=c.id_contacto and e.id_usuario=usuario;
 	
if(ncontactos = 0) then
select id_usuario as 'id' from empresas where id_usuario!=usuario and (nombre like concat('%',patron,'%') or descripcion like concat('%',patron,'%') or provincia like concat('%',patron,'%'));
else
select id_usuario as 'id' from empresas where id_usuario!=usuario and (nombre like concat('%',patron,'%') or descripcion like concat('%',patron,'%') or provincia like concat('%',patron,'%')) and id_usuario not in (select e.id_usuario as 'id' from empresas as e,contactos as c where e.id_usuario=c.id_contacto and c.id_contacto=usuario);
end if;
END	

Vista asociada	ver_contactos_busqueda.php

Tabla 14. Tabla-diseño Buscar Contacto

6.4.9 Anadir contacto

Paso 1. Evento: El usuario pulsa en el botón de añadir sobre un contacto de la lista.

Paso 2. Se recarga la página, se crea el controlador y se invoca.

Paso 3. El controlador llama a añadir contacto de la lista de contactos, pasándole el id de usuario.

Paso 4. El controlador de contactos carga la vista asociada.

Evento	Pulsado el botón añadir
Subcontrolador asociado	Controlador de contactos
$_GET	$_GET['menu']=contactos $_GET['accion']=nuevo_contacto
$_POST	$_POST[añadir_id_contacto]
Modelo	Llama añadir contacto.
Procedimientos almacenados	anadir_contacto(id_usuario,id_contacto)

CREATE DEFINER=`root`@`localhost` PROCEDURE `anadir_contacto`(in usuario int(10),in contacto int(10))
BEGIN
insert into contactos(id_usuario,id_contacto,estado) values (usuario,contacto,1);
insert into contactos(id_usuario,id_contacto,estado) values (contacto,usuario,0);
END	

Vista asociada	ver_contactos_nuevo.php

Tabla 15. Tabla-diseño Añadir contacto

6.4.10 Buscar usuario

Paso 1. Evento: El usuario introduce el nombre de usuario de una empresa y pulsa el botón buscar

Paso 2. Se recarga la página, se crea el controlador y se invoca.

Paso 3. Comprueba si el usuario cliente existe.

Paso 4. Muestra la vista asociada

Evento	Pulsado el botón buscar
Subcontrolador asociado	Controlador de compras
$_GET	$_GET['menu']=compras $_GET['accion']=registrar_venta
$_POST	$_POST[comprobar_usuario], $_POST[cliente]
Modelo	Se crea una nueva instancia de usuario y se llama a existe_usuario que comprueba si el usuario es válido
Procedimientos almacenados	existe_usuario_compra(usuario,vendedor)
CREATE DEFINER=`root`@`localhost` PROCEDURE `existe_usuario_compra`(in u varchar(50), in v varchar(50)) BEGIN Select usuario from usuarios where usuario=u and usuario<>v; END	
Vista asociada	registrar_compra.php

Tabla 15. Tabla-diseño Buscar Usuario

6.4.11 Anadir Venta

Paso 1. Evento: El usuario introduce los datos del formulario y pulsa registrar

Paso 2. Se recarga la página, se crea el controlador y se invoca.

Paso 3. El controlador de compras recoge los valores del formulario y llama a la función registrar_venta de la clase empresa

Paso 4. Muestra la vista asociada

Evento	Pulsado el botón buscar
Subcontrolador asociado	Controlador de compras
$_GET	$_GET['menu']=compras $_GET['accion']=registrar_venta
$_POST	$_POST[registrar_venta], $_POST[cliente], $_POST['descripcion_venta], $_POST['valor_venta']
Modelo	Se le registra la venta a la empresa usuaria pasándole los datos del formulario
Procedimientos almacenados	anadir_venta(id_usuario,id_contacto)

CREATE DEFINER=`root`@`localhost` PROCEDURE `anadir_venta`(in cliente int(10),in vendedor int(10),in f_h datetime,in d text,in v float)
BEGIN
insert into ventas(id_vendedor,id_comprador,fecha_hora,descripcion,valor) values (vendedor,cliente,f_h,d,v);	

Vista asociada	buscar_cliente.php

Tabla 16. Tabla-diseño Añadir Venta

6.4.12 Ver compras

Paso 1. Evento: El usuario pulsa en el menú “mis compras”

Paso 2. Se recarga la página, se crea el controlador y se invoca.

Paso 3. El controlador de compras carga la lista de compras de la empresa usuaria

Paso 4. Muestra la vista asociada

Evento	Se ha entrado en el menú “mis compras”
Subcontrolador asociado	Controlador de compras
$_GET	$_GET['menu']=compras $_GET['accion']=ver_compras
$_POST	
Modelo	Muestra lista de compras
Procedimientos almacenados	obtener_compras_usuario(id_usuario)
CREATE DEFINER=`root`@`localhost` PROCEDURE `obtener_compras_usuario`(in id_c int(10)) BEGIN Select id_venta from ventas where id_comprador=id_c; END	
Vista asociada	ver_compras.php

Tabla 17. Tabla-diseño Ver Compras

6.4.13 Valorar compra

Paso 1. Evento: El usuario pulsa el botón valorar

Paso 2. Se recarga la página, se crea el controlador y se invoca.

Paso 3. El controlador de compras obtiene la valoración realizada y llama al procedimiento valorar de esa compra.

Paso 4. Muestra la vista asociada

Evento	Se ha pulsado el botón valorar
Subcontrolador asociado	Controlador de compras
$_GET	$_GET['menu']=compras $_GET['accion']=ver_compras
$_POST	$_POST['guardarvaloracion_id-usuario'], $_POST[valoración]
Modelo	Obtiene el id de la compra y la valoración y los pasa como parámetros a la función valorar de esa compra.
Procedimientos almacenados	valorar_compra(id_compra,valoracion)
CREATE DEFINER=`root`@`localhost` PROCEDURE `valorar_compra`(in venta int(10),in val int(1)) begin UPDATE ventas SET valoracion=val where id_venta=venta; END	
Vista asociada	ver_compras.php

Tabla 18. Tabla-diseño Valorar Compra

6.4.14 Ver Ventas

Paso 1. Evento: El usuario pulsa en el menú la opción “mis ventas”

Paso 2. Se recarga la página, se crea el controlador y se invoca.

Paso 3. El controlador de compras carga la lista de ventas de esa empresa

Paso 4. Muestra la vista asociada

Evento	Se ha pulsado en el menú “mis ventas”
Subcontrolador asociado	Controlador de compras
$_GET	$_GET['menu']=compras $_GET['accion']=ver_ventas
$_POST	
Modelo	Obtiene las ventas de la empresa (cargar_lista_ventas())
Procedimientos almacenados	obtener_ventas_usuario(id_usuario)
CREATE DEFINER=`root`@`localhost` PROCEDURE `obtener_ventas_usuario`(in id_v int(10)) BEGIN Select id_venta from ventas where id_vendedor=id_v; END	
Vista asociada	ver_ventas.php

Tabla 19. Tabla-diseño Ver Ventas

6.4.15 Ver Balance

Paso 1. Evento: El usuario elige un mes y año y pulsa ver balance

Paso 2. Se recarga la página, se crea el controlador y se invoca.

Paso 3. El controlador de compras carga la lista de compras y ventas de la fecha seleccionada.

Evento	Se ha pulsado en el menú “ver balance”
Subcontrolador asociado	Controlador de compras
$_GET	$_GET['menu']=compras $_GET['accion']=busca_balances
$_POST	$_POST[filtrar_balance], $_POST[anno],$_POST[mes]
Modelo	La función cargar_balances(año,mes) obtiene la lista de compras y ventas y muestra el total.
Procedimientos almacenados	obtener_balances(id_usuario,fecha1,fecha2)
CREATE DEFINER=`root`@`localhost` PROCEDURE `obtener_balances`(in id_c int(10), in f1 datetime, in f2 datetime) BEGIN Select id_venta from ventas where (id_comprador=id_c or id_vendedor=id_c) and fecha_hora>=f1 and fecha_hora<f2 order by fecha_hora desc; END	
Vista asociada	ver_balances.php

Tabla 20. Tabla-diseño Ver Balance

6.4.16 Buscar Balance

Paso 1. Evento: El usuario pulsa en el menú la opción “balance”

Paso 2. Se recarga la página, se crea el controlador y se invoca.

Paso 3. El controlador de compras carga la vista asociada

Evento	Se ha pulsado en el menú “mis ventas”
Subcontrolador asociado	Controlador de compras
$_GET	$_GET['menu']=compras $_GET['accion']=busca_balances
$_POST	
Modelo	
Procedimientos almacenados	
Vista asociada	buscar_balances.php

Tabla 21. Tabla-diseño Buscar Balance

6.4.17 Ver valoraciones ajenas patrón

Paso 1. Evento: El usuario introduce un patrón de búsqueda y le da a buscar

Paso 2. Se recarga la página, se crea el controlador y se invoca.

Paso 3. El controlador de valoraciones obtiene las empresas que coinciden con el patrón de búsqueda y sus valoraciones medias

Paso 4. Presenta la vista asociada

Evento	El usuario ha introducido un patrón de búsqueda y ha pulsado el botón buscar
Subcontrolador asociado	Controlador de valoraciones
$_GET	$_GET['menu']=valoraciones $_GET['accion']=no
$_POST	$_POST['buscar_empresa'], $_POST['valoracion']=2, $_POST['empresa']
Modelo	Obtiene las empresas que coinciden con el patrón de búsqueda y sus valoraciones medias
Procedimientos almacenados	Obtener_valoraciones_ajenas
CREATE DEFINER=`root`@`localhost` PROCEDURE `obtener_valoraciones_ajenas`(in usuario int(10),in patron varchar(100)) BEGIN	

select e.id_usuario as 'id' from empresas e, ventas v where v.id_vendedor=e.id_usuario and e.id_usuario<>usuario and v.id_comprador<>usuario and (e.nombre like concat('%',patron,'%') or e.descripcion like concat('%',patron,'%') or e.provincia like concat('%',patron,'%')) and v.valoracion<>0 group by e.id_usuario; END	
Vista asociada	ver_valoraciones.php

Tabla 22. Tabla-diseño Ver Valoraciones Ajenas

Hasta ahora los procedimientos almacenados eran muy sencillitos,inserts,updates y selects simples por lo que no he añadido ningún ejemplo, pero estos pueden resultar un poco menos claros.

CREATE DEFINER=`root`@`localhost` PROCEDURE `obtener_valoraciones_ajenas`(in usuario int(10),in patron varchar(100))
BEGIN

select e.id_usuario as 'id' from empresas e, ventas v where v.id_vendedor=e.id_usuario and e.id_usuario<>usuario and v.id_comprador<>usuario and (e.nombre like concat('%',patron,'%') or e.descripcion like concat('%',patron,'%') and v.valoracion<>0 group by e.id_usuario;

END

Devuelve el id de las empresas (entre las que no estoy yo,e.id_usuario<>usuario,v.id_comprador<>usuario), que coincidan con el patrón de búsqueda (e.nombre like concat('%',patron,'%') or e.descripcion like concat('%',patron,'%')) y que estén valorados(v.valoracion<>0)

6.4.18 Ver valoraciones ajenas sector y provincia

Paso 1. Evento: El usuario introduce un patrón de búsqueda y le da a buscar

Paso 2. Se recarga la página, se crea el controlador y se invoca.

Paso 3. El controlador de valoraciones obtiene las empresas que con el sector y la provincia seleccionados.

Paso 4. Presenta la vista asociada

Evento	El usuario ha seleccionado un sector y una provincia y ha pulsado el botón buscar
Subcontrolador asociado	Controlador de valoraciones
$_GET	$_GET['menu']=valoraciones $_GET['accion']=no
$_POST	$_POST['buscar_empresa_sector'], $_POST['valoracion']=2, $_POST['sector1'], $_POST['provincia']
Modelo	Obtiene las empresas que coinciden con el sector y provincia introducidos.
Procedimientos almacenados	Obtener_valoraciones_ajenas_sector
CREATE DEFINER=`root`@`localhost` PROCEDURE `obtener_valoraciones_ajenas_sector`(in usuario int(10),in sec int(10),in prov int(10))	

BEGIN select e.id_usuario as 'id', avg(v.valoracion) from empresas e, ventas v,empresas_sectores as es where es.id_usuario=e.id_usuario and v.id_vendedor=e.id_usuario and e.id_usuario!=usuario and e.provincia=prov and es.id_sector=sec; END	
Vista asociada	ver_valoraciones.php

Tabla 23. Tabla-diseño Ver Valoraciones Ajenas

6.4.19 Ver valoraciones contactos patrón

Paso 1. Evento: El usuario introduce un patrón de búsqueda y le da a buscar

Paso 2. Se recarga la página, se crea el controlador y se invoca.

Paso 3. El controlador de valoraciones obtiene las empresas que coinciden con el patrón de búsqueda y sus valoraciones medias

Paso 4. Presenta la vista asociada

Evento	El usuario ha introducido un patrón de búsqueda y ha pulsado el botón buscar
Subcontrolador asociado	Controlador de valoraciones
$_GET	$_GET['menu']=valoraciones $_GET['accion']=no
$_POST	$_POST['buscar_empresa'], $_POST['valoracion']=1, $_POST['empresa']
Modelo	Obtiene las empresas que coinciden con el patrón de búsqueda y sus valoraciones medias por los contactos del usuario
Procedimientos almacenados	Obtener_valoraciones_ajenas, obtener_valoraciones_contactos
CREATE DEFINER=`root`@`localhost` PROCEDURE `obtener_valoraciones_ajenas`(in usuario int(10),in patron varchar(100)) BEGIN select e.id_usuario as 'id' from empresas e, ventas v where v.id_vendedor=e.id_usuario and	

e.id_usuario<>usuario and v.id_comprador<>usuario and (e.nombre like concat('%',patron,'%') or e.descripcion like concat('%',patron,'%') or e.provincia like concat('%',patron,'%')) and v.valoracion<>0 group by e.id_usuario;
END
 	
CREATE DEFINER=`root`@`localhost` PROCEDURE `obtener_valoraciones_contactos`(in usuario int(10),in id_buscar int(10))
BEGIN
select avg(v.valoracion) as valoracion from empresas e, ventas v where v.id_vendedor=id_buscar and e.id_usuario!=usuario and v.id_comprador in (Select c.id_contacto from contactos as c where c.id_usuario=usuario and c.estado=2) and v.valoracion <>0;
END	

Vista asociada	ver_valoraciones_contactos.php

Tabla 24. Tabla-diseño de Ver Valoraciones Contactos

Este procedimiento selecciona media de valoraciones de las empresas (que no son el usuario, e.id_usuario<>usuario) y cuyo comprador ha sido uno de nuestros contactos (v.id_comprador in (Select c.id_contacto from contactos as c where c.id_usuario=usuario and c.estado=2))

6.4.20 Ver valoraciones ajenas sector y provincia

Paso 1. Evento: El usuario introduce un patrón de búsqueda y le da a buscar

Paso 2. Se recarga la página, se crea el controlador y se invoca.

Paso 3. El controlador de valoraciones obtiene las empresas que con el sector y la provincia seleccionados y su valoración media por los contactos de la empresa.

Paso 4. Presenta la vista asociada

Evento	El usuario ha seleccionado un sector y una provincia y ha pulsado el botón buscar
Subcontrolador asociado	Controlador de valoraciones
$_GET	$_GET['menu']=valoraciones $_GET['accion']=no
$_POST	$_POST['buscar_empresa_sector'], $_POST['valoracion']=2, $_POST['sector1'], $_POST['provincia']
Modelo	Obtiene las empresas que coinciden con el sector y provincia introducidos y la valoración media de los contactos del usuario
Procedimientos almacenados	Obtener_valoraciones_ajenas_sector, obtener_valoraciones_contactos
Vista asociada	ver_valoraciones_contactos.php

Tabla 24. Tabla-diseño de Ver Valoraciones Ajenas

6.5 Capa de presentación

El desarrollo de la capa de presentación ha llevado una parte importante de la carga de trabajo.

Es importante señalar que el diseño y la creación de la interfaz gráfica sólo empezó a realizarse una vez probadas todas las clases del modelo y su integración con la parte de programación no fue muy costosa gracias sobre todo a la utilización del patrón MVC.

La metodología seguida para desarrollar la interfaz gráfica fue la siguiente

Paso 1: Diseño previo
Diseño “en papel” de la estructura de la páginas, las secciones, menús,...

Paso 2: División en capas
Creación de un html que estructure con capas todas las secciones de la página y asi, poder asignar una funcionalidad a cada parte

Paso 3: Creación de imágenes
Creación o edición de imágenes con para darle “vida” a la interfaz. Se utilizó una versión de evaluación del programa Photoshop para realizar esta tarea.

Paso 4: Hojas de estilo
Crear estilos css para los distintos elementos títulos, tablas, …

Paso 5: Incluir la interfaz
Aunque aparentemente cada vez se carga una página diferente, en realidad siempre es la misma, index.php que llama al controlador principal y este

carga el head, la pagina asociada a esa vista y el foot. Por lo tanto, fue necesario adaptar la página creada en el paso 2 y separarla en varias.

A continuación se muestra la estructura de capas de la página y donde se encuentra cada parte en el árbol de ficheros

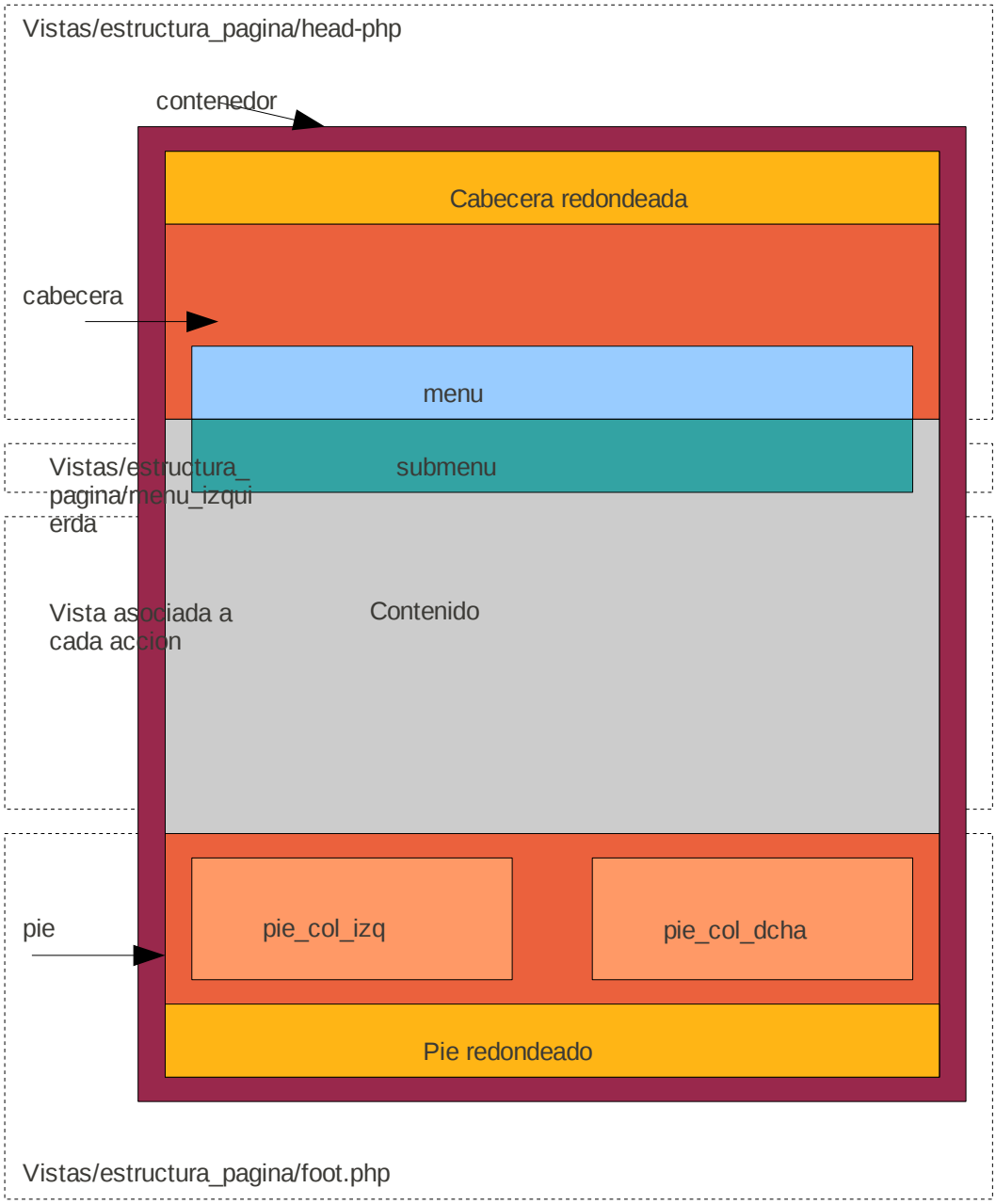


Figura 25. Estructura de capas de la página

CAPÍTULO 7. IMPLEMENTACIÓN

En esta sección se mostrarán los detalles más importantes de la implementación que no se hayan mencionado hasta el momento y que puedan ser de utilidad al desarrollador.

7.1 Árbol de directorios

Cada archivo tiene un lugar en el árbol de directorios para que sea sencillo encontrarlos.

7.1.1 Directorio Conexión

Aquí se encuentra la clase Conexión, que incluye los métodos necesarios para conectar y desconectar la base de datos así como los métodos para llamar a procedimientos o funciones (también pueden ejecutar consultas directamente desde aquí)

7.1.2 Directorio Controladores

En este directorio se encuentra el controlador principal, así como los subdirectorios.

7.1.3 Directorio Modelo

Todas las clases del modelo pueden encontrarse aquí

7.1.4 Clases externas

El framework simpletest y la clase PhpMailer se encuentran colgando de la raíz directamente cada uno en su respectivo directorio

7.1.5 Directorio Tests

Todas las pruebas programadas con simpletest se encuentran en este directorio. Para ejecutarlas es necesario llamarlas poniendo toda la ruta hasta ellas. Por ejemplo: http://host/nombre_aplicacion/Tests/usuario_test.php

7.1.6 Directorio Vistas

Este directorio contiene otros subdirectorios que tienen relación con la vista

Subdirectorio app: aquí se encuentran las vistas asociadas a los casos de uso.
Subdirectorio css: contiene las css para cargar los estilos a las vistas
Subdirectorio estructura_vista: contiene el head, el foot y los submenús
Subdirectorio img: contiene todas las imágenes utilizadas.

7.2 Clase PHP Mailer

Para enviar las invitaciones al sistema se utiliza la clase externa PHPMailer que sobre las librerías que ya dispone PHP para enviar email añade nuevas funcionaliades.
Su uso es sencillo. Se configuran los datos para nuestra cuenta específica de correo (se ha creado una en GMAIL) .

```
$mail = new PHPMailer();
$mail->IsSMTP();
$mail->SMTPAuth = true;
$mail->SMTPSecure = "ssl";
$mail->Host = "smtp.gmail.com";
```

```
$mail->Port = 465;
$mail->Username = "bnet.pfc@gmail.com";
$mail->Password = "*******";
$mail->From = "bnet.pfc@gmail.com";

  hasta aquí los datos de la cuenta de correo y del servidor

$mail->FromName = "Bnet";
  nombre del emisor
$mail->Subject = "Invitacion a la red de empresas Bnet";
  asunto del email
$mail->AltBody = "Invitación a Bnet";
  por si el receptor no tiene habilitado HTML
$mail->MsgHTML($cuerpo);
  cuerpo del mensaje
$mail->AddAddress($email);
  dirección del receptor
$mail->IsHTML(true);
  si se quiere enviar en formato HTML
 if(!$mail->Send()) {
  envío del email
    return(-1);
 }
```

7.3 Balance por fechas

La fecha guardada en la base de datos tiene formato datetime y por lo tanto para poder buscar entre fechas es necesario la siguiente modificación para que el procedimiento almacenado pueda trabajar sin problemas.

```
if($mes==0){
Si no se elige un mes, se crean dos fechas-horas la
primera de ese año y la última
      $fecha1=$anio."-01-01 00:00:00";
      $fecha2=$anio."-12-31 23:59:59";
}
else{
```

```
cuando se selecciona un mes hay que comprobar que la fecha
de la compra sea mayor que el 1 de ese mes a las 0:0:0 y
que sea menor que el primer dia del mes siguiente.Hay que
tratar de una manera especial el mes de diciembre ya que
la fecha tope es el primer día del año siguiente
      if($mes!=12){
            $fecha1=$anio."-".$mes."-01 00:00:00";
            $fecha2=$anio."-".($mes+1)."-01 00:00:00";
      }else{
            $fecha1=$anio."-".$mes."-01 00:00:00";
            $fecha2=($anio+1)."-01-01 00:00:00";
      }
}
$parametros="'".$this->id."','".$fecha1."','".$fecha2."'";
 $resultado=$conexion ->
 llamarProcedimiento('obtener_balances',
 $parametros,0);
```

CAPÍTULO 8. PRUEBAS

Existen distintos tipos de pruebas que se pueden aplicar en el desarrollo de un proyecto para dotarlo de la mayor robustez posible. Las pruebas que se han realizado en la aplicación Bnet son las mostradas a continuación.

8.1 Pruebas de caja blanca. Pruebas unitarias

Las pruebas de caja blanca pretenden descubrir errores en la codificación del programa. Su función no es probar la funcionalidad del sistema sino su estructura. Las pruebas unitarias están englobadas en este tipo de pruebas.

Según el libro "Técnicas cuantitativas para la gestión en la ingeniería del software" las pruebas unitarias constituyen:

"el primer paso para detectar errores en el código, pues se centran en la menor unidad de diseño del software: el módulo -por ejemplo, un método de una clase o una clase-. El objetivo principal de estas pruebas es detectar errores en cada uno de los módulos del software al ser ejecutados independientemente del resto de componentes."

Para realizar las pruebas unitarias se ha utilizado un framework llamado SimpleTest. Su funcionamiento es del estilo de Junit de Java.

Se programan las pruebas usando un juego de pruebas (una base de datos con datos de prueba) establecido para así saber cuales deben ser los resultados esperados.

8.2 Pruebas de caja negra

Las pruebas de caja negra se abstraen de la codificación del software y se basan en la funcionalidad de éste basándose únicamente en las entradas y salidas del apartado que se prueba.

En nuestro caso probamos que los casos de uso cumplieran con la funcionalidad esperada. Se realizaron las siguientes pruebas:

8.2.1 Iniciar Sesión

- No introducir el usuario
- No introducir la contraseña
- No introducir ningún campo
- Introducir los dos bien
- Introducir los dos mal
- Introducir usuario bien y contraseña mal
- Introducir usuario mal y contraseña bien

8.2.2 Invitaciones

- Enviar invitación sin datos
- Enviar invitación solo con empresa
- Enviar invitación solo con email
- Enviar invitación con todos los campos

8.2.3 Alta Usuario

- No rellenar algún campo

- Realizar el registro y e intentar realizar un registro nuevo recargando la página

8.2.4 Foot de la página

- Comprobar que muestra si no existen compras o ventas
- Comprobar que muestra si no existen peticiones de amistad pendientes

8.2.5 Ver mis contactos

- Comprobar que muestra si no hay ningún contacto

8.2.6 Contactos pendientes

- Comprobar que muestra si no hay ninguna petición de amistad pendiente
- Realizar una peticion de amistad y aceptar con el otro usuario
- Realizar una peticion de amistad y recharla con el otro usuario

8.2.7 Añadir contacto

- Intentar buscar la empresa usuaria como contacto
- Realizar una búsqueda que no coincida con ningún resultado
- Realizar una búsqueda con resultados
- Añadir un contacto que está pendiente de aceptacion

8.2.8 Registrar venta

- Buscar usuario inexistente
- No introducir un valor en la compra
- Introducir un valor no válido

8.2.9 Valorar compra

- No valorarla
- Valorar una compra

8.2.10 Balances

Comprobar que los balances y las ventas coinciden con la bbdd

8.2.11 Valoraciones ajenas

- Buscar empresa inexistente
- Buscar la empresa usuaria
- Comprobar la coherencia de los datos con la realidad

8.3 Pruebas de integración

Las pruebas de integración procuran descubrir errores que se producen al interactuar dos o más módulos cuyo funcionamiento es correcto individualmente según las pruebas anteriores pero que generan mal-funcionamientos al trabajar juntos.

Estas pruebas has sido muy sencillas de probar en el proyecto gracias al diseño homogéneo de la aplicación. Su función sólo ha sido tratar de comprobar el funcionamiento general del sistema y la coherencia de las secciones y distintos usuarios.

8.4 Pruebas de validación

Las pruebas de validación buscan comprobar que la funcionalidad del sistema se corresponde con los requisitos iniciales.

A lo largo de todo el proyecto se han ido realizando reuniones para comprobar que realmente lo desarrollado era lo que se pretendía al principio.

CAPÍTULO 9. IMPLANTACIÓN

La aplicación está implantada en el servidor de la empresa. Puede visitarse en http://aplicaciones.grupogisma.com/bnet

El servidor es un equipo con el sistema GNU/Linux, en su distribución de Ubuntu.

En este equipo existen otras webs por lo que la instalación de la aplicación fue muy sencilla. Copiar la carpeta de archivos a la carpeta www, importar la base de datos y cambiar los datos específicos de la base de datos. Todo este proceso se hizo siguiendo el manual de instalación (Ver apéndice A).

CAPÍTULO 10. GESTIÓN

Incidencias relevantes

El mayor problema surgido a lo largo de la realización del proyecto ha tenido que ver con el convenio entre la empresa, universidad y alumno. Debido a problemas con la aplicación Proiekges (fue necesario abrir dos incidencias), no se pudo firmar el convenio hasta febrero de 2010. Por este motivo fue necesario que se redactaran dos planificaciones para el DOP. La primera fue presentada a la empresa como plan de proyecto y la segunda se hizo teniendo en cuenta la demora inicial.

Además la formación necesaria para aprender un nuevo lenguaje de programación, fue mayor que la planificada e hizo que se retrasara un poco más el comienzo del diseño e implementación de la aplicación.

10.1 Partes de horas

Se resumen aquí las horas dedicadas por el alumno a cada tarea por cada mes.

Tarea	Enero	Febrero	Marzo	Abril	Mayo	Total
DOP	8	4				12
Captura de Requisitos		4	8			12
Análisis		4	8			12
Diseño		6	16			22
Implementación		20	28	24	32	104
Pruebas		4	8	8	12	32
Implantación			4			4
Interfaz gráfica			8	24	4	36
Formación	52			20		72
Reuniones y documentación de estas	8	12	8	4	4	36
Otros		20	8		24	52
Total	68	74	96	80	76	**394 horas**

Tabla 24. Horas mensuales por tarea

Enero: Como se puede apreciar la mayor parte de las horas de formación se realizarion en este mes, las horas de formación incluyen:

- Instalación de las aplicaciones necesarias y aprendizaje de las mismas, apache, php, mysql, eclipse, fireftp, firebug
- Formación en PHP, HTML, CSS y javascript

Febrero: En febrero se comenzó el diseño de la aplicación. También se dedicaron horas a realizar esquemas de las pantallas que tendría la aplicación y debatirlas en las reuniones.

Marzo: Cabe destacar que en este mes se comenzó a usar el servidor en el que quedaría finalmente alojada la aplicación.

También hubo algunos problemas en el diseño que obligaron a realizar correcciones como queda plasmado en el apartado "otros".

Abril: Este mes fue uno de los más difíciles para el alumno ya que nunca había tenido que realizar una interfaz gráfica desde el inicio.

Mayo: Este mes se dedicó a la puesta a punto de la aplicación y a reunir la documentación desarrollada a lo largo del proyecto para redactar la memoria.

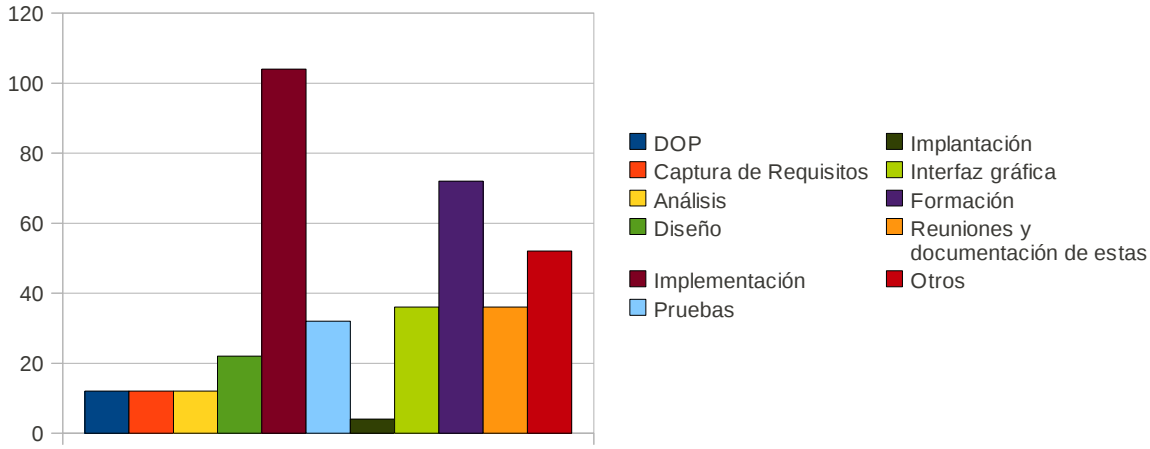


Figura 26. Tiempo dedicado a cada tarea

Total

Las horas totales dedicadas al proyecto que quedan reflejadas en los partes son 394 horas. Coinciden bastante bien con las horas planificadas. Sin embargo cabe destacar que algunas horas no se apuntaron en su día (las que salían del horario fijado por el convenio) y por lo tanto no son exactamente las reales aunque se acercan bastante.

10.2 Planificado vs real

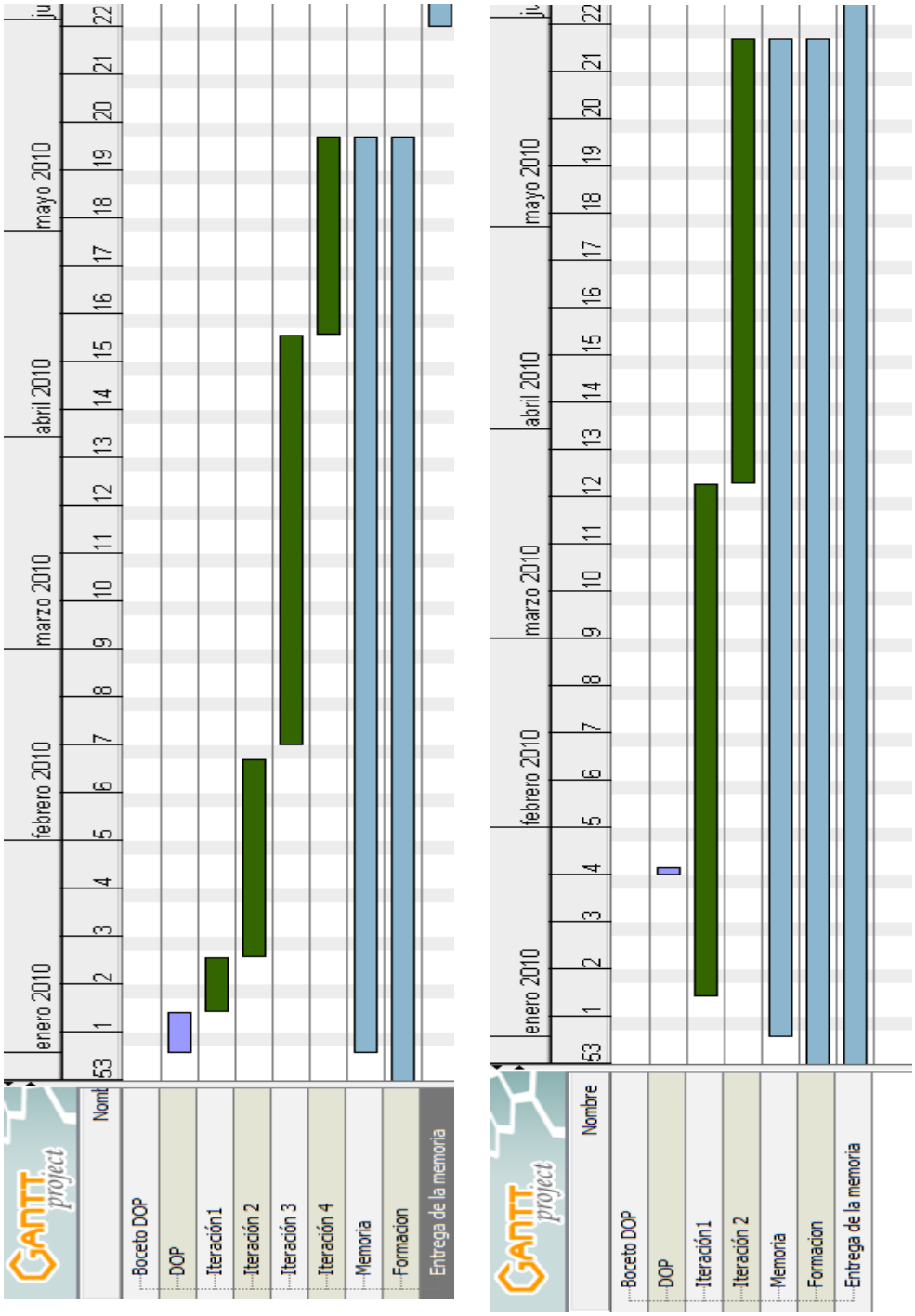


Figura 27 y 28. Diagrama de Gantt planificado y real

Como se puede apreciar en la figura 27 y 28 existen cambios notables entre la planificación temporal y la real. Debido a necesidades cambiantes de la empresa y problemas con los plazos se eliminaron dos iteraciones completas. Sin embargo, esto no ha afectado a las horas totales dedicadas al proyecto como puede verse en la tabla 25.

Tarea	Horas planificadas	Horas reales
DOP	8	12
Captura de Requisitos	30	12
Análisis	50	12
Diseño	80	22
Implementación	100	104
Pruebas	30	32
Implantación	4	4
Interfaz gráfica	20	36
Formación	40	72
Reuniones y documentación de estas	38	36
Otros		52
Total	400 horas	394 horas

Tabla 25. Horas planificadas vs Horas Reales

10.3 Conclusiones sobre la gestión

Si bien al principio la gestión se realizó más o menos adecuadamente, en la segunda iteración no se pudo mantener la planificación. Debido a la reducción de plazos para poder dar a la empresa parte del prototipo para una exposición se tuvo que modificar el número de iteraciones aunque finalmente esto no ha supuesto un gran retraso en la finalización del proyecto

CAPÍTULO 11. CONCLUSIONES

Finalmente, sólo queda comentar la valoración general del proyecto.Es decir, cuáles han sido los objetivos cumplidos y cuales no, qué se ha aprendido, qué se podría mejorar, etc.

11.1 Los objetivos

Repasando los objetivos recogidos en el DOP podemos ver que los objetivos principales se han cumplido

Del DOP: ***"Objetivo 1 – Realización de un prototipo funcional para la empresa."***

Esta meta fue la que se cumplió primero por el ritmo de la empresa. El prototipo final del que es objeto esta memoria no es exactamente el mismo que el ofrecido a la empresa ya que por temas de propiedad prefería mantener dos versiones.

Del DOP: **"Objetivo 2 – Creación de toda la documentación necesaria para la aprobación del proyecto por parte de la Universidad"**

Todas las fases del proyecto han sido documentadas siguiendo la metodología usada anteriormente en la asignatura Ingeniería del Software.

Somos conscientes de que hay muchos aspectos que podrían mejorarse a este respecto y si algo se ha aprendido es que ir dejando partes sin documentar a la larga hace perder más tiempo y no permite aprender de lo ya realizado.

El total cumplimiento de este objetivo se verá una vez se haya calificado el proyecto.

Del DOP: ***“ Objetivo 3 – Formación en distintos campos (programación, documentación, metodología de trabajo en el mercado laboral,...) para el alumno ”***

11.2 Las mejoras

Como en cualquier proyecto existen algunos puntos que podrían mejorarse o cambiarse.

11.2.1 Encuestas

Para tener una idea más clara de las posibles mejoras no a nivel técnico sino a nivel del usuario se han realizado una serie de encuestas sencillas a usuarios con distintos niveles de conocimientos en informática (pueden verse en el CD adjunto).

11.2.2 Mejoras en la aplicación

Mejoras sugeridas por los usuarios:
De los resultados de las encuestas pueden deducirse algunas mejoras. Destacan:

- El formulario de alta, es poco usable, si se produce un error, hay que volver a rellenar los datos.
- El color verde con algunos contrastes puede resultar molesto.
- Se echa de menos tecnología tipo JavaScript-Ajax que ayude al usuario con autorrellenados al buscar empresas o usuarios,...

Mejoras sugeridas por el alumno :

Al tratarse de un prototipo existen algunos aspectos que no se han tenido demasiado en cuenta y que deberían mejorarse si llegara a formarse un producto completo. Temas como la seguridad o la accesibilidad no han sido promovidas a lo largo de las iteraciones, pero son puntos clave en un producto real.

El aspecto gráfico también es un punto a tomar en cuenta. El diseño de la interfaz de usuario para la versión de la empresa fue realizada en colaboración con un diseñador. Esto fue una experiencia también importante que ayudó al alumno a realizar su propia interfaz gráfica aplicando lo aprendido.

11.2.3 Mejoras para el alumno

Como se ha mencionado a lo largo de la memoria, las incidencias, plazos y descuidos han hecho que no todas las partes hayan quedado igual de documentadas, por lo tanto dotarse de herramientas adecuadas y desarrollar una metodología de trabajo serán un aspecto importante a tomar en cuenta en el futuro.

11.3 Valoración personal del alumno

11.3.1 Mundo empresarial

Si tengo que quedarme con algo positivo del proyecto, escogería todo lo que he aprendido al realizarlo en una empresa.

Los plazos, las continuas adaptaciones, las situaciones inesperadas y en general las situaciones propias del mundo empresarial me han obligado a aprender nuevas cosas casi a diario y he aprendido cuanto se valora no quedarse “estancado” en el mundo en continuo cambio de la informática.

11.3.2 Formación

El aprendizaje de tecnologías como PHP, CSS y nuevas herramientas como FireBug, simpletest, Mysql GUI Tools, Photoshop.. ha sido otro de los aspectos más positivos que me han ayudado a ampliar mi formación y experiencia.

Considero por lo tanto que he alcanzado la meta propuesta y que realizar el proyecto en una empresa me ha permitido tener una idea mucho más real del proceso de desarrollo de software. He aprendido mucho y he sufrido otro tanto pero siempre desde un punto de vista positivo.

APÉNDICES

Apéndice A. Manual de Implantación

El objeto de este apartado es tratar de dar unas pautas sencillas para la instalación en un servidor de todos los programas necesarios para la puesta en marcha de la aplicación.

Se distinguen claramente dos casos:

Caso 1. Existencia de un servidor web propio o contratado (hosting)

La aplicación Bnet al funcionar como un servicio web puede instalarse en cualquier servidor web que trabaje con Mysql y PHP.

Si el equipo en el que se va a instalar la aplicación ya dispone de estos servicios lo único necesario es importar la base de datos y copiar el directorio de la aplicación en el directorio de webs del servidor.

Caso 2. Implantación de la aplicación en un equipo sin servidor web

Caso 2.1

Si, en cambio, el equipo no dispone de alguno de los servicios requeridos será necesario instalarlos teniendo en cuenta el sistema operativo.

Una opción sencilla es descargarse un paquete como XAMPP o LAMPP (según el s.o) que permite instalar todos los servicios de un modo sencillo.
Puede descargarse de la página oficial:
http://www.apachefriends.org/en/xampp.html

Una vez instalado el paquete solo es necesario importar la base de datos y copiar el arbol de directorios en la carpeta pública web (htdocs o www, según el s.o)

Caso 2.2

Otra opción, que ha sido la realizada a lo largo del desarrollo del proyecto es implantar la aplicación en un equipo GNU/Linux.

A continuación se mostrarán los pasos para instalar todos los programas necesarios en un equipo con una distribución de GNU/Linux.

1. Instalar Apache

Abrimos una terminal e introducimos los siguientes comandos:

Primero instalamos el servidor web Apache.

apt-get install apache2

2. Instalar Mysql

Después instalamos el gestor de bases de datos Mysql

apt-get install mysql-server-5.0

Por defecto Mysql viene sin contraseña así que le damos una

/usr/bin/mysqladmin -u root pass otravezpass

3. Instalar PHP

Instalamos PHP5

apt-get install php5

4. Permitir el funcionamiento conjunto

Para que estos tres programas puedan funcionar conjuntamente es necesario introducir lo siguiente:

apt-get install libapache2-mod-auth-mysql

apt-get install php5-mysql

5. Importación de la base de datos Bnet

Importamos la base de datos

mysql -u root -p

Introducimos la contraseña y quedamos conectados a mysql. Creamos una nueva bbdd

create database nombrebbdd

exit

Importamos el fichero de bbdd

mysql -u root -p nombrebbdd < rutafichero

6. Copiar la aplicación

Copiamos los ficheros de la aplicación en la carpeta

/var/www/nombre_aplicacion

7 . Edición de los datos del servidor

Es necesario adaptar la aplicación, para ello hay que cambiar estos dos archivos con los datos específicos de la aplicacion

- Conexion/conexion.php

linea 12: $this->link=mysql_connect(“localhost”,”usuario”,”password”,””);
linea 13: mysql_select_db(“nombrebbdd”);

linea 17:
$this → link=mysql_connect(“localhost”,”usuario”,”password”,”nombrebbdd”);

- *linea 117: <a href='http://direccionaplicacion/alta.php?.....*

8 . Introducción del usuario inicial

Debido a que Bnet es un sistema cerrado basado en invitaciones es necesario incluir manualmente el primer usuario de todos. Para ello es necesario introducir los datos en la tabla usuarios y empresas. (Recuérdese que la contraseña tiene que guardarse en formato md5 de 32 caracteres)

Si todo funciona correctamente se podrá acceder a la aplicación a través de localhost/nombre_aplicacion

Apéndice B. Manual de usuario

Para poder acceder a cualquier función de la aplicación es necesario iniciar sesión para que el sistema pueda comprobar si el usuario es una empresa registrada.

Página inicial de Bnet

Si el sistema válida correctamente los datos se dará acceso a la página de bienvenida y a los menús de acceso a las distintas funciones de la aplicación

En la parte superior de la página puede verse debajo del logotipo el menú principal, en la parte superior derecha hay disponible un enlace para desconectarse de la aplicación cuando ya no se desee navegar más por esta.

En la parte inferior existen dos subsecciones a modo de acceso directo.

- La sección 'Últimas acciones realizadas'. Muestra las dos últimas ventas y las dos últimas compras
- La sección 'Peticiones de amistad'. Muestra el número de solicitudes de amistad que le han hecho al usuario y están sin aceptar.

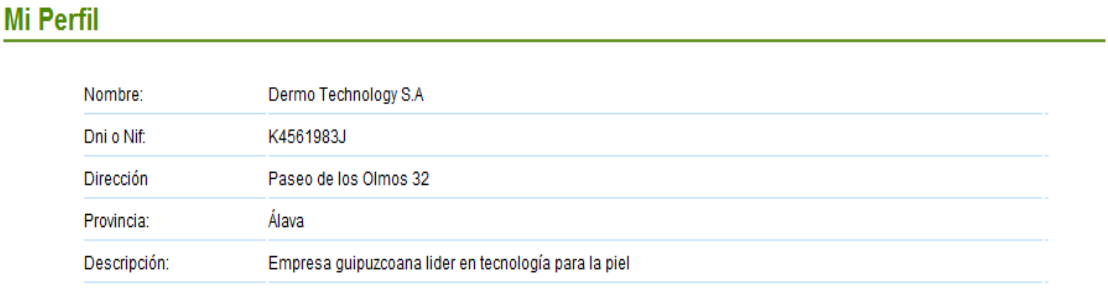
Menú Perfil

Al pulsar sobre el menú perfil se cargará el submenú asociado

también se cargará por defecto la opción 'mi perfil'.

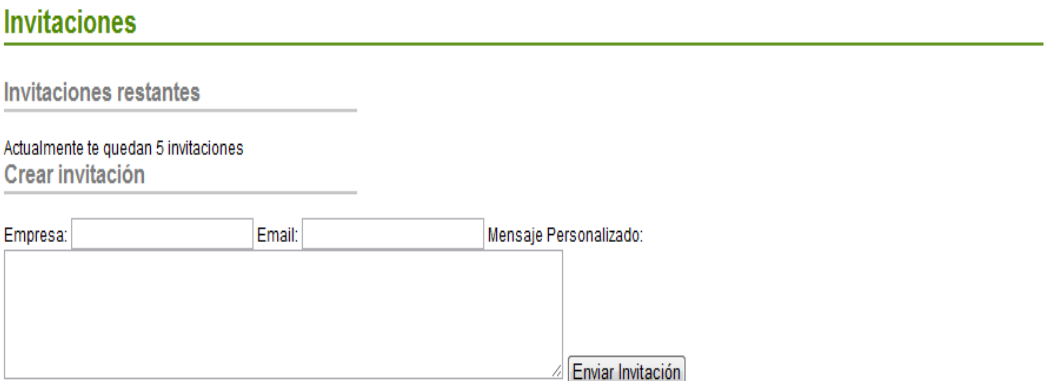
Mi Perfil

En esta sección se pueden ver los datos asociados a la empresa usuaria



Invitaciones

Bnet es un sistema cerrado y por lo tanto sólo puede accederse a él a través de invitaciones de los usuarios ya registrados. A través de esta opción, un usuario puede comprobar el número de invitaciones que posee y enviar invitaciones a empresas amigas, rellenando un formulario.

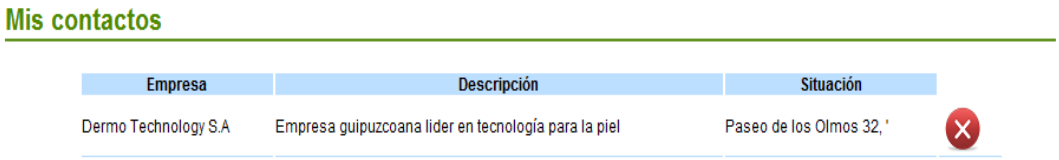


Menú Contactos

Al entrar en el menú de los contactos aparecerán 3 nuevas opciones, la opcion mis contactos es cargada por defecto

Mis Contactos

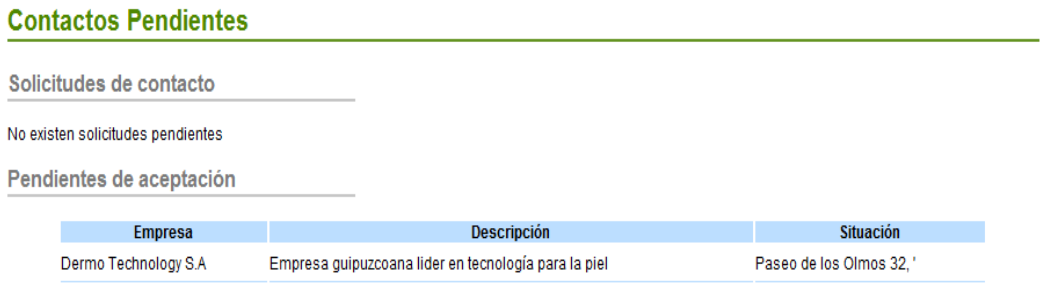
Esta sección incluye los datos básicos sobre los contactos de la empresa usuaria.



Es posible eliminar un contacto pulsando el botón rojo. Ese contacto estará de nuevo disponible en la sección de añadir contacto

Contactos Pendientes

Esta otra sección muestra primero las empresas que nos han realizado una petición de amistad y después las empresas a las que hemos pedido ser sus contactos.

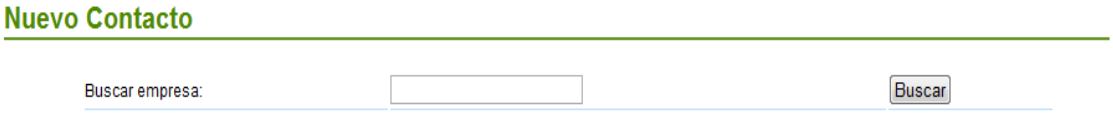


Para aceptar una petición de contacto es necesario pulsar el botón verde, en cambio, para rechazarla es necesario pulsar el botón rojo. En este caso, el otro usuario ya no podrá volver a realizar una petición de contacto. Por otro lado, cuando una de nuestras peticiones acabe siendo aceptada se borrará de la sección 'Pendientes de aceptación' y pasará a estar en 'Mis contactos'

En caso de que en alguna de las subsecciones no existan solicitudes pendientes se mostrará un mensaje de información

Añadir contacto

Para poder añadir a una empresa como contacto primero es necesario utilizar el buscador para encontrar la empresa a añadir.



Se mostrará una lista de empresas que coincidan con el patrón de búsqueda.

Para realizar una petición de amistad a alguna de las empresas de la lista es necesario pulsar el botón aceptar.

En caso de no encontrar ningún resultado coincidente se mostrará un mensaje de error

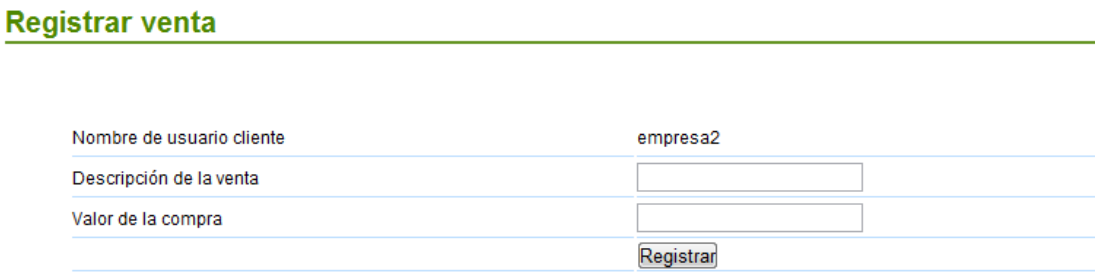
Menú Compras

En esta sección se pueden ver los balances de compra-venta realizados a través del sistema y registrar una venta.

Registrar Venta

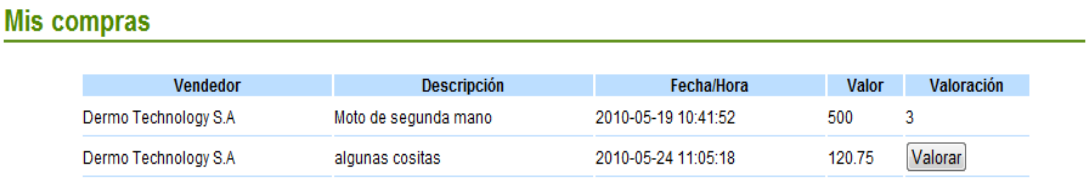
Por defecto esta es la opción cargada al entrar en el menú Compras.

Para que el sistema funcione es necesario registrar las ventas. Esto es tarea del vendedor-proveedor que tiene que introducir en nombre de usuario del comprador. Si el comprador existe le dejará introducir los datos de la venta.



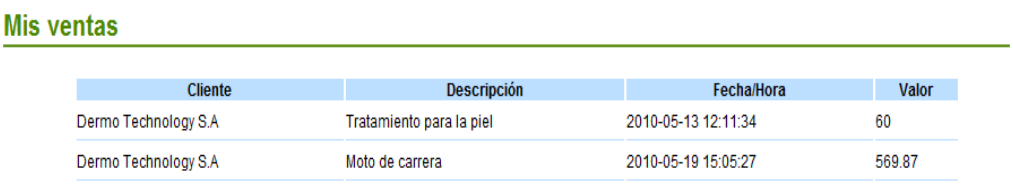
Mis compras

En esta subsección puede verse el historial de compras realizadas. Además se puede valorar cada compra para así mostrar el grado de satisfacción con ese comprador-vendedor de modo que también los contactos podrán verlo



Mis Ventas

Aquí puede verse el historial de ventas.

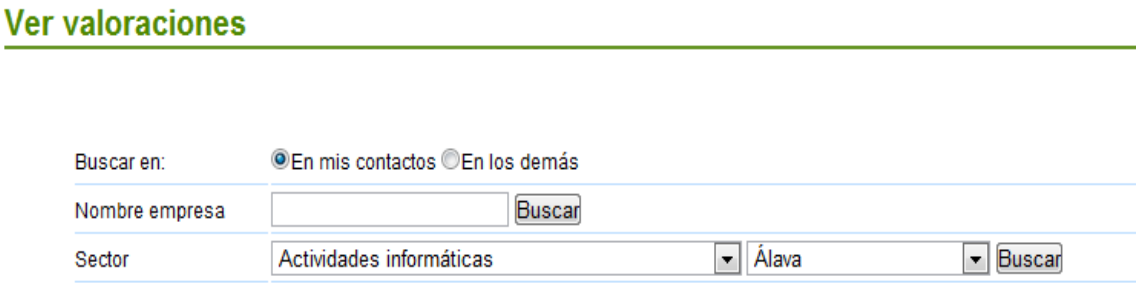


Balance

En esta sección se pueden ver los balances generales

Valoraciones

Este apartado es informativo y busca ser una guía para encontrar nuevos proveedores de calidad por ello el usuario puede buscar aquí una empresa para comprobar la valoración media a una empresa.



Existen dos posibilidades:

- Buscar la valoración media que han dado todos los usuarios del sistema a una empresa. Esto se hace eligiendo la opción "en los demás"
- Buscar la valoración media que han dado tus contactos a esa empresa, pulsando "en mis contactos".

Además puedes buscar las empresas por su nombre o según su sector y provincia

Este sería un posible resultado de buscar en la sección contactos
En el campo valoración se ve la media de valoraciones de tus contactos, si pulsas el botón ver se muestra por cada contacto la valoración dada.