

Proyecto fin de curso

pon farr

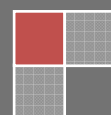


I.E.S Claudio Moyano

TUTOR: Agustín Folgado

ALUMNO: Antonio Santiago Ramos

FECHA: 01/12/2016



Me gustaría dedicar este proyecto por completo a mis padres, por toda su ayuda incondicional, sin la cual, no sé si habría sido capaz de llegar al final y a mi colega Adrián Román, ambos, dos de los trescientos valientes que plantaron cara.

Contenido

2 ESTUDIO DEL PROBLEMA Y ANÁLISIS DEL SISTEMA.....	7
2.1 Introducción.....	7
2.2 Descripción de la aplicación.....	7
2.3 Objetivos.....	7
2.3.1 Objetivos generales.....	8
2.3.2 Objetivos específicos.....	8
2.3.3 Objetivos deseados	9
3 RECURSOS NECESARIOS PARA SU DESARROLLO.....	10
3.1 Recursos humanos	10
3.2 Recursos hardware.....	10
3.3 Recursos software.....	10
4 PLANIFICACIÓN	11
4.1 Metodología.....	11
4.2 Secuencia de desarrollo del proyecto	11
4.3 Tabla de actividades, cálculo de tiempos y diagrama de Gantt.....	14
5 DESARROLLO DEL PROYECTO.....	15
5.1 Secuencia real del desarrollo del proyecto	15
5.2 Diagrama de relacional.....	16
5.3 Interacción con el usuario.....	24
5.3.1 Casos de uso/ historias de usuario e interacciones (más representativos).....	24
Parte de trabajo	24
Cuaderno de bitácora	25
Crear incidencia.....	25
Modificar incidencia.....	26
Cerrar incidencia.....	27
Cliente.....	28
Crear cliente.....	28
Filtrar cliente.....	29
Modificar cliente.....	30
Proveedor.....	31
Crear proveedor	31
Filtrar proveedor	32

5.3.3 Diseño de la interfaz	42
5.3.4 Diagrama de Gantt.....	44
5.4 CÓDIGO FUENTE. PROCESOS MÁS REPRESENTATIVOS.	45
5.4.1 Métodos deshabilitar/ habilitar	45
5.4.2 Método isEmpty	45
5.4.3 Método que vacía los campos de texto	47
5.4.4 Método conectar/desconectar de base de datos	47
5.4.5 Métodos para el cliente.....	48
5.4.6 Stored procedure insertar teléfonos	51
5.4.7 Stored procedure insertar(se escoge el de cliente, el resto son iguales)	51
6 FASE DE PRUEBAS	52
6.1 Pruebas realizadas.....	52
7 CONCLUSIONES FINALES	54
7.1 Reflexión	54
7.2 Grado de cumplimiento de los objetivos fijados	54
7.3 Propuesta de modificaciones y/o ampliaciones	55
8 DIFICULTADES Y PROBLEMAS FUNDAMENTALMENTE ENCONTRADOS	56
10 BIBLIOGRAFÍA	56
10.1 Fuentes en papel	56
10.2 Fuentes digitales	56

LISTA DE IMÁGENES

IMAGEN 1: Pantalla principal de la aplicación, prototipo	12
IMAGEN 2: Muestra de datos con filtro, prototipo	12
IMAGEN 3: inserción cliente, prototipo	13
IMAGEN 4: base de datos, prototipo	13
IMAGEN 5: tabla de tiempos estimada.....	14
IMAGEN 6: diagrama de GANTT estimado	14
IMAGEN 7: tabla de tiempos real	15
IMAGEN 8: base de datos real.....	16
IMAGEN 9: diagrama de clases	17
IMAGEN 10: clase Base de Datos	18

IMAGEN 11: clase conexión a la base de datos	18
IMAGEN 12: clase ControladorCliente	18
IMAGEN 13: clase ControladorCuaderno	19
IMAGEN 14: clase ControladorIncidencia	20
IMAGEN 15: clase ControladorLogin	20
IMAGEN 16: clase ControladorMaterial	20
IMAGEN 17: clase ControladorProveedor	21
IMAGEN 18: clase ControladorProveedor	21
IMAGEN 19: clase ControladorPuesto	22
IMAGEN 20: clase ControladorTelefono	22
IMAGEN 21: clase ControladorUsuario	23
IMAGEN 22: caso de uso parte de trabajo.....	24
IMAGEN 23: diagrama de secuencia crear incidencia	25
IMAGEN 24: Diagrama de secuencia modificar incidencia	26
IMAGEN 25: diagrama de secuencia cerrar incidencia	27
IMAGEN 26: caso de uso cliente	28
IMAGEN27: diagrama de secuencia crear cliente	29
IMAGEN 28: diagrama de secuencia filtrar cliente	29
IMAGEN 29: diagrama de secuencia modificar cliente.....	30
IMAGEN 30: caso de uso proveedor	31
IMAGEN 31: diagrama de secuencia crear proveedor.....	31
IMAGEN 32: diagrama de secuencia filtrar proveedor.....	32
IMAGEN 33: diagrama de secuencia modificar proveedor	33
IMAGEN 34: caso de uso usuarios	34
IMAGEN 35: diagrama de secuencia eliminar usuario.....	34
IMAGEN 36: diagrama de secuencia modificar usuario.....	35
IMAGNE 37: diagrama de secuencia alta usuario	36
IMAGEN 38: caso de uso material	37
IMAGEN 39: caso de uso alta material.....	37
IMAGEN 40: caso de uso eliminar material	38
IMAGEN 41: caso de uso modificar material.....	39
IMAGEN 42: caso de uso login	40
IMAGEN 43: caso de uso correo	41

IMAGEN 44: caso de uso red	41
IMAGEN 45 : Pantalla principal de la aplicación.....	42
IMAGEN 46 : Pantalla principal de la aplicación desplegada	42
IMAGEN 47 : Pantalla principal de clientes, pantalla tipo para el resto de la aplicación	43
IMAGEN 48: diagrama de gannt real.....	44

2 ESTUDIO DEL PROBLEMA Y ANÁLISIS DEL SISTEMA

2.1 Introducción

Con el fin de elaborar el proyecto, se han mantenido conversaciones con el director del mismo, así como con el cliente, ya que en este proyecto se pretende crear una aplicación para un negocio que existe físicamente. Por este motivo, es necesario conocer la actividad de Redes Zamora S.L.

Redes Zamora S.L., es una empresa dedicada a la venta y montaje de ordenadores en su mayor parte, pero también trabajan con otros productos como móviles, impresoras, proyectores y más componentes electrónicos. Hace 5 años dos socios pusieron el negocio en marcha y usando unas simples hojas de cálculo, les era más que suficiente para controlar el volumen de trabajo, actualmente el crecimiento del negocio, que ha derivado en el aumento de obras simultaneas les ha hecho ver la necesidad de tener una aplicación informática para controlar con más facilidad su negocio y poder seguir prosperando adecuadamente.

La idea de los socios de Redes Zamora S.L., es que el programa se adapte a su manera de trabajar, ya que a pesar de la juventud de su empresa, ellos llevan más de veinte años dedicándose profesionalmente en este sector, por este motivo una de las cosas más importantes en mi proyecto será entender correctamente su estructura de trabajo para desarrollar una aplicación ágil efectiva y cómoda.

Analizando el estado actual de Redes Zamora S.L., nace este proyecto final de ciclo formativo de grado superior, donde se diseñará una aplicación interna en la que se podrá gestionar cliente, trabajadores, partes de trabajo, proveedores, redes y correos; abarcando cada una de las necesidades de los clientes de la empresa.

Los propietarios de la empresa y amigos del autor detectan la necesidad de controlar informáticamente la gestión de su negocio. En diferentes conversaciones con el director de la empresa se plantea la posibilidad de realizar una aplicación .Net con la finalidad de cubrir sus objetivos.

2.2 Descripción de la aplicación

Mediante la creación de esta aplicación se pretende conseguir un conocimiento más profundo en la programación .Net así como el modelo cliente servidor de base de datos externas, incluyendo adquirir un conocimiento más profundo en la implementación de Stored Procedures.

2.3 Objetivos

Este epígrafe contará con tres subepígrafes en los que se desglosarán los objetivos del proyecto, objetivos generales, objetivos específicos y objetivos deseados.

2.3.1 Objetivos generales

La aplicación persigue abarcar la gestión de una PYME (Pequeña y Mediana Empresa) dedicada a la informática de sistemas, y su gestión en el día a día dentro de la misma empresa. Las personas implicadas en el uso de este proyecto serían todos los componentes (empleados-jefes) de la PYME

- ❖ Organizar las reparaciones de cualquier componente informático y electrónico
- ❖ Organizar las llamadas telefónicas dedicadas a distintos tipos de empleados de la empresa, problemas, incidencias y soluciones.
- ❖ Acarrear la gestión de redes LAN; MAN; WAN y su mantenimiento para el cliente que lo desee.
- ❖ -Gestionar cuentas de correo electrónico y dominios.

2.3.2 Objetivos específicos

- ❖ Diseñar una aplicación de aspecto minimalista y agradable tanto al uso como a la vista del usuario.
- ❖ La aplicación se desarrollará bajo la estructura de MVP (Modelo Vista Controlador) y con base de datos a no embebida, mediante conexiones a servidor MySQLServer usando la aplicación gratuita XAMPP.
- ❖ Llevará a cabo toda la gestión de todos los proveedores que tenga la PYME (altas, modificaciones y consultas).
- ❖ Gestionar así mismo todos los clientes presentes y futuros que tenga la empresa (altas, modificaciones y consultas).
- ❖ Desarrollar los partes de trabajo a realizar ante la entrada de cualquier equipo o componente averiado, impresión del mismo parte y su posterior

reimpresión con la solución, el tiempo dedicado a la reparación, los componentes utilizados y el técnico encargado de resolver el problema.

- ❖ Desarrollar un cuaderno de bitácora en el que se anotarán todas las incidencias habidas separadas específicamente para un técnico / trabajador de la empresa; por medio de llamadas telefónicas diarias intentando servir de organizador de trabajo o bien citas con los clientes para una incidencia fuera de las instalaciones de la empresa.
- ❖ Llevar la gestión de las posibles redes LAN; MAN; WAN que puedan tener los clientes a tal empresa, haciendo para ello anotaciones al uso de el servidor, router, puestos de trabajo, ubicación así como direcciones ip y configuraciones. Estas configuraciones se podrán imprimir para mayor facilidad del técnico a la hora de resolver una incidencia relacionada con el ámbito net.
- ❖ Gestionar los dominios de páginas web y correos electrónicos del cliente que así lo desee, llevando a cabo anotaciones dependiendo de si el correo es gratuito o tiene dominio y dentro de ese dominio cuántas direcciones de correo electrónico hay que configurar y establecer en los puestos de los trabajadores de un cliente determinado, así como anotando a quién corresponden y la configuración completa de correo para los distintos tipos de gestores de e-mail. Asimismo se podrán imprimir todas las hojas de configuraciones para facilitar la tarea al técnico en caso de encontrarse fuera de sus instalaciones.
- ❖ Gestionar la base de datos relativa al almacén de piezas para utilizar en las reparaciones o vender así como precios al cliente y los precios del proveedor.

2.3.3 Objetivos deseados

- ❖ La aplicación se desarrolla en monopuesto en un principio, uno de los objetivos deseados es desarrollar la aplicación con el tipo de modelo Cliente-Servidor, e instalarla en todos los ordenadores al uso que se puedan encontrar en la empresa.
- ❖ Utilización de sockets para el modelo cliente servidor

- ❖ Intentar llevar la cantidad de piezas del almacén y las advertencias a la hora de que no haya stock o que la cantidad de piezas necesarias sea superior al stock del almacén.

3 RECURSOS NECESARIOS PARA SU DESARROLLO

Los recursos utilizados para este proyecto se van a dividir en tres subapartados; recursos humanos, recursos software y recursos hardware

3.1 Recursos humanos

Este proyecto será desarrollado en su totalidad por una única persona encargada de las tareas de creación e implementación de bases de datos, creación de toda la programación interna y desarrollo de toda la interfaz visual.

Aunque este valor puede ser estimado, comparándolo con el mercado real, si contemplamos el gasto por hora que supondría el hecho de contratar un analista, un diseñador y un programador. El análisis de costes se realizará contemplando los precios de mercado, aunque todas las partes estarán desarrolladas por su autor, se calculará el precio por el tipo de trabajo y quien lo realizaría en el mundo laboral real.

3.2 Recursos hardware

Para la realización del siguiente proyecto se contará con los siguientes elementos hardware:

- ❖ Ordenador PC con procesador intel pentium 7 965 extreme edition
- ❖ 24 gb de memoria RAM
- ❖ Tarjeta gráfica nvidia gtx 280
- ❖ Monitor TFT de 22"

Requisitos software en los equipos de los usuarios:

- ❖ Windows Vista o superior
- ❖ Xampp Server
- ❖ Conexión a internet
- ❖ Net framework 3.0 o superior

3.3 Recursos software

- ❖ Microsoft Windows 7 Ultimate
- ❖ Microsoft Visual Studio
- ❖ XAMPP
- ❖ VMware Workstation

4 PLANIFICACIÓN

A continuación se describirá una estimación de lo que podría llevar la elaboración de todo el proyecto, siendo probablemente un tanto diferente a la estimación real y vista desde una visión más general y no tan detallada como será el desarrollo real del proyecto.

4.1 Metodología

Se propone separar conceptos como el de creación de base de datos, creación de vistas modelos y controladores y a su vez paralelamente ir creando el código correspondiente al back-end en la base de datos.

El proyecto está elaborado en base a la programación orientada a objetos (POO) , arquitectura cliente, servidor y modelo vista controlador (MVC)

4.2 Secuencia de desarrollo del proyecto

Lo primero que se creará será la base de datos, asumiendo variaciones a lo largo de la elaboración del proyecto.

Una vez creada la base de datos tipo, a partir de ella se desarrollarán todos los modelos que se estimen oportunos y paralelamente las vistas para esos modelos.

Una vez diseñado todo lo anterior, se usará como base modificable para crear el comportamiento (los controladores) que ha de llevar cada vista y paralelamente se irá diseñando el código correspondiente a la parte de servidor de base de datos (Stored Procedures).

Teniendo realizado todo lo anterior, se propone el siguiente prototipo de aplicación, modificable, representado por varias capturas de pantalla, ventana principal y distintas vistas de edición , creación y muestreo de los datos.



IMAGEN 1: Pantalla principal de la aplicación, prototipo

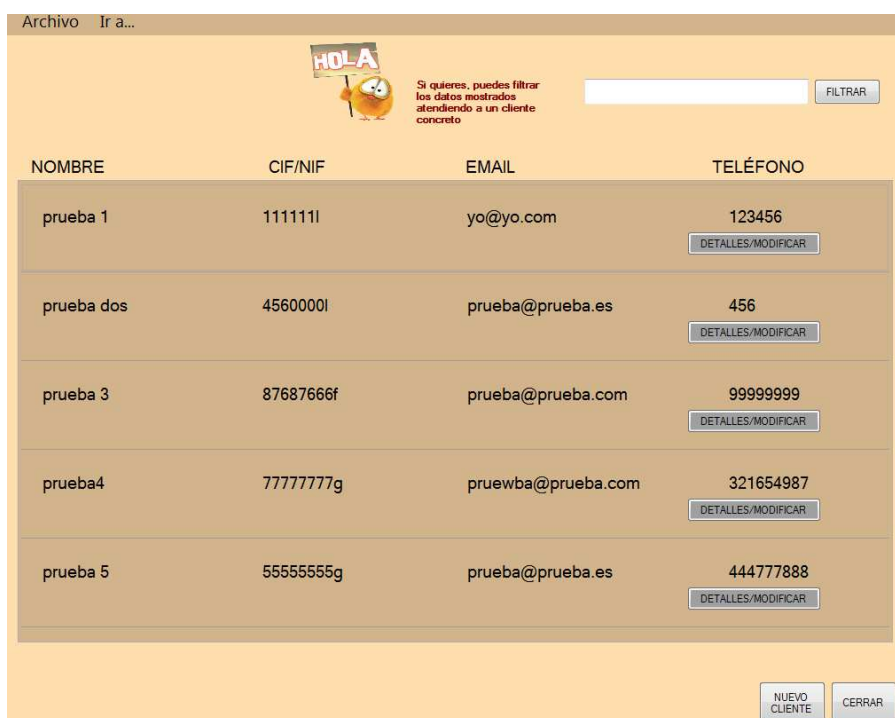


IMAGEN 2: Muestra de datos con filtro, prototipo

Archivo Ir a...

NOMBRE

NIF/CIF DOMICILIO


E-MAIL POBLACION

TELEF.01 PROVINCIA CP

TELEF.02

PERSONA DE CONTACTO

OBSERVACIONES

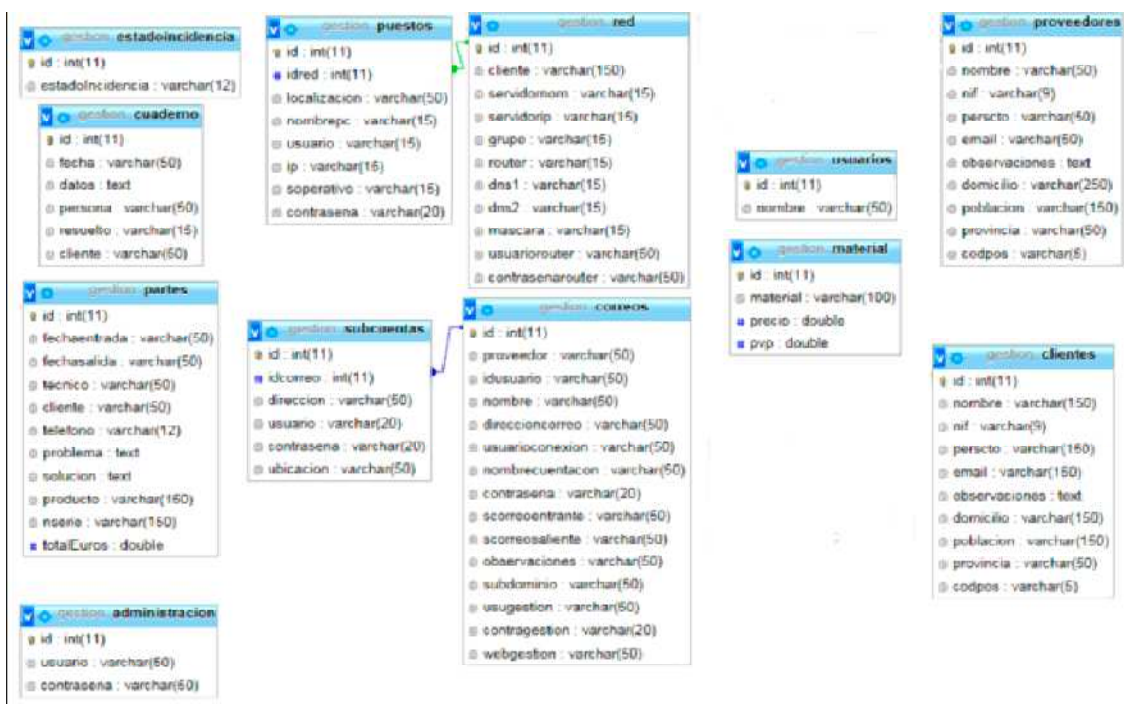


Tienes campos obligatorios de rellenar:

- Nombre.
- NIF/CIF.
- Telef.01.
- Domicilio.
- Población.
- Provincia.
- Código postal.

IMAGEN 3: inserción cliente, prototipo

IMAGEN 4: base de datos, prototipo



4.3 Tabla de actividades, cálculo de tiempos y diagrama de Gantt.

A continuación se describe por medio de un gráfico y una tabla, los posibles tiempos que, de forma general, puede que lleve el desarrollo completo de la aplicación.

Actividades	inicio	Duración(horas)	Fin
Diseño de la base de datos	22/09/2016	8	30/09/2016
Diseño de los modelos	24/09/2016	40	05/10/2016
Diseño de las vistas	24/09/2016	70	25/11/2016
Diseño de los controladores	24/09/2016	80	25/11/2016
Diseño de los controladores de base de datos	24/09/2016	80	25/11/2016
Fase de pruebas	25/11/2016	20	30/11/2016
Memoria	25/09/2016	100	30/11/2016

IMAGEN 5: tabla de tiempos estimada

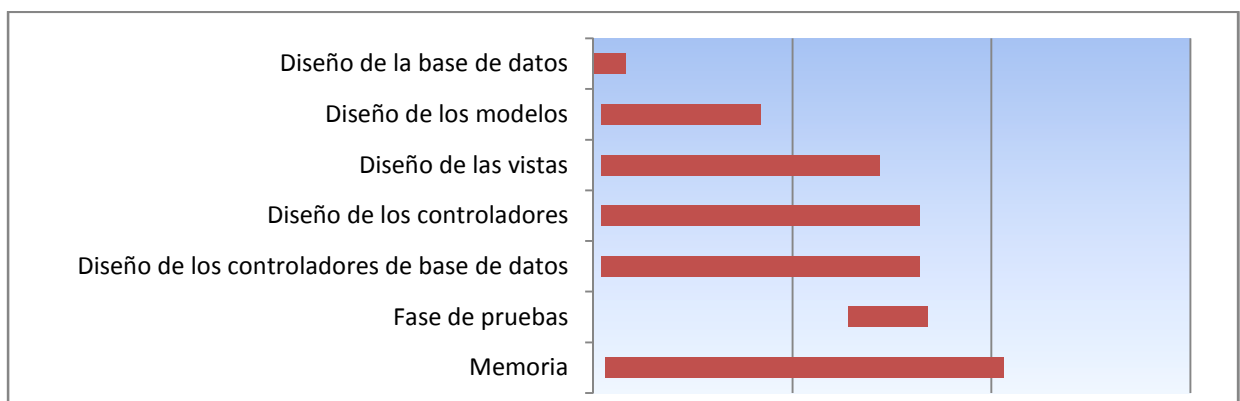


IMAGEN 6: diagrama de GANTT estimado

5 DESARROLLO DEL PROYECTO

A continuación se van a detallar todos los tiempos y diagramas reales de los que ha constado el proyecto en su totalidad.

5.1 Secuencia real del desarrollo del proyecto

Actividades	inicio	Duración(horas)	Fin
Diseño de la base de datos	22/09/2016	8	23/09/2016
Diseño modelo Administrador	24/09/2016	1	24/09/2016
Diseño modelo Cliente	24/09/2016	1	24/09/2016
Diseño modelo Correo	24/09/2016	1	24/09/2016
Diseño modelo Cuaderno	24/09/2016	1	24/09/2016
Diseño modelo Incidencia	24/09/2016	1	24/09/2016
Diseño modelo Material	25/09/2016	1	25/09/2016
Diseño modelo ParteTrabajo	25/09/2016	1	25/09/2016
Diseño modelo Proveedor	25/09/2016	1	25/09/2016
Diseño modelo Puesto	25/09/2016	1	25/09/2016
Diseño modelo Red	26/09/2016	1	26/09/2016
Diseño modelo Subcuenta	26/09/2016	1	26/09/2016
Diseño modelo Telefono	26/09/2016	1	26/09/2016
Diseño modelo Usuario	26/09/2016	1	26/09/2016
Diseño de la conexión de la base de datos	27/09/2016	4	27/09/2016
Diseño del controlador Cliente	28/09/2016	6	28/09/2016
Diseño del controlador Cuaderno	29/09/2016	6	29/09/2016
Diseño del controlador Incidencia	30/09/2016	6	30/09/2016
Diseño del controlador Login	01/10/2016	3	01/10/2016
Diseño del controlador Material	02/10/2016	4	02/10/2016
Diseño del controlador Parte	03/10/2016	7	03/10/2016
Diseño del controlador Proveedor	04/10/2016	5	04/10/2016
Diseño del controlador Puesto	05/10/2016	8	05/10/2016
Diseño del controlador Red	06/10/2016	7	06/10/2016
Diseño del controlador Telefono	07/10/2016	8	07/10/2016
Diseño del controlador de Correo	08/10/2016	7	08/10/2016
Diseño del controlador de Subcuentas	09/10/2016	8	09/10/2016
Diseño del controlador Usuario	10/10/2016	4	10/10/2016
Diseño del controlador Metodos Globales	27/09/2016	70	31/11/2016
Diseño de la interfaz de usuario	27/09/2016	40	20/11/2016
Realización de las pruebas	27/09/2016	40	30/11/2016
Documentación total	27/09/2016	60	30/11/2016

IMAGEN 7: tabla de tiempos real

5.2 Diagrama de relacional

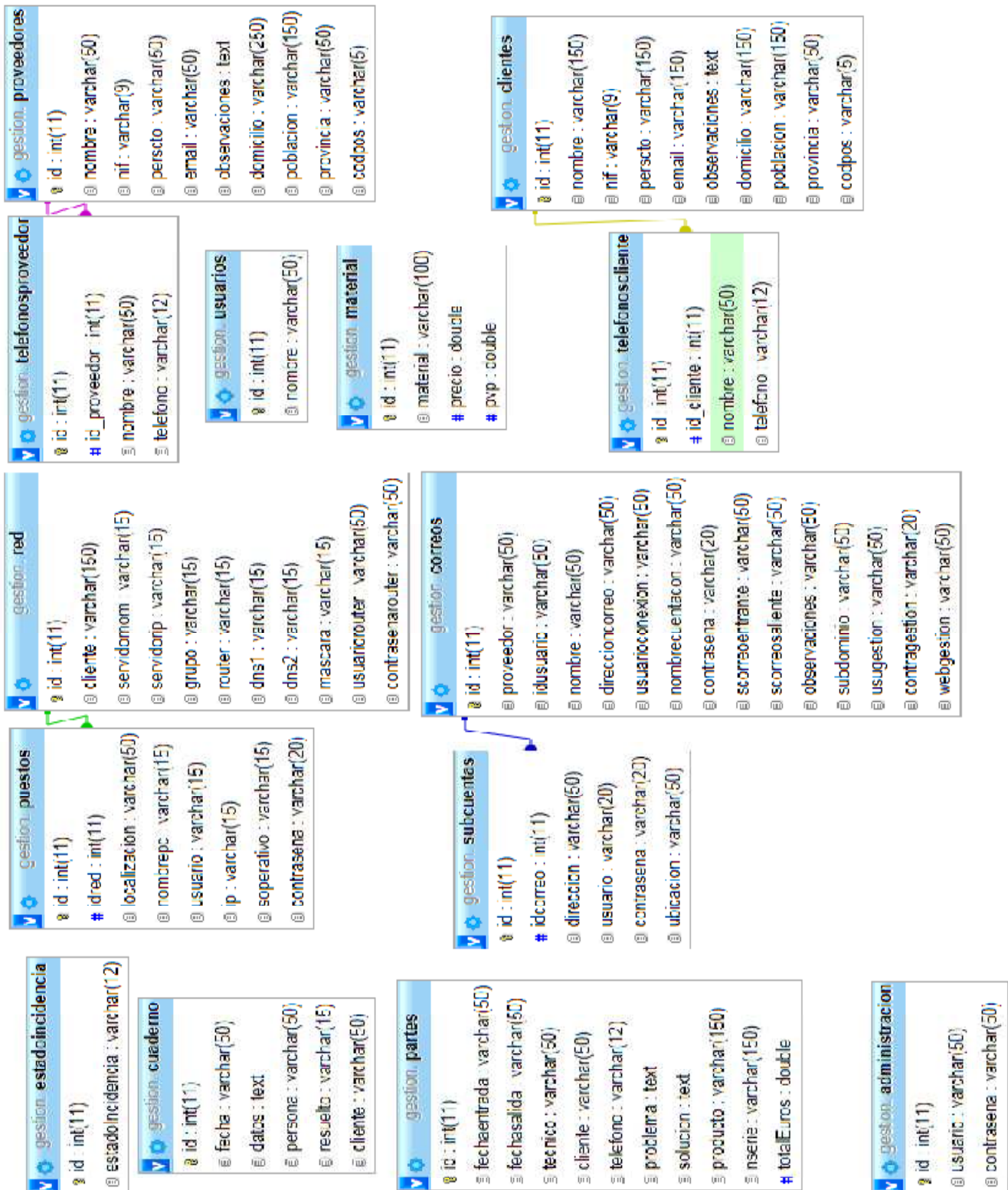


IMAGEN 8: base de datos real

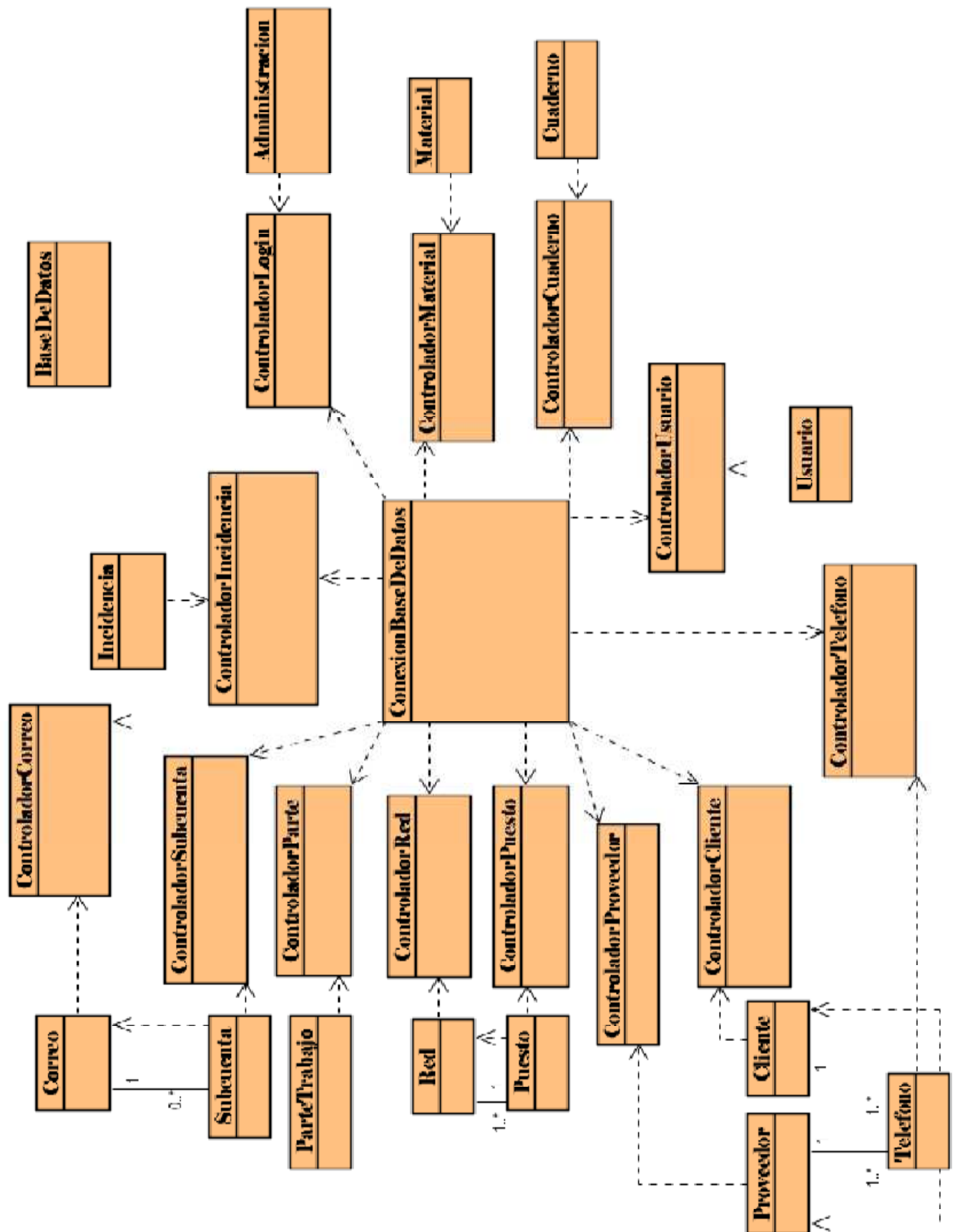


IMAGEN 9: diagrama de clases

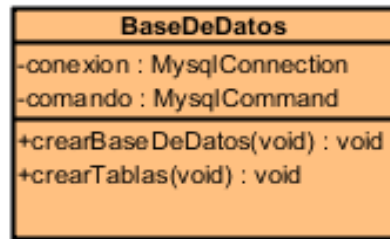


IMAGEN 10: clase Base de Datos

- ❖ crearBaseDeDatos , se conecta al servidor de la base de datos y crea la base de datos, una vez creada , llama a la función crearTablas y crea todas las tablas pertinentes para la base de datos dentro de las directrices del modelo relacional.

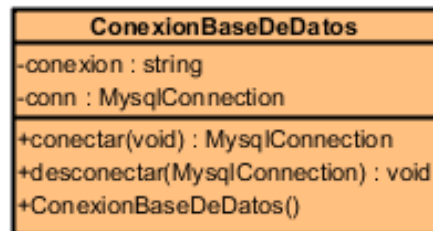


IMAGEN 11: clase conexión a la base de datos

- ❖ Conectar , se conecta a la base de datos en cuestión y devuelve a variable de tipo MySqlConnection para poder operar contra la base de datos.
- ❖ Desconectar recibe la variable de tipo conexion de la función conectar cuando corresponda, cerrando la conexión y limpiando recursos.

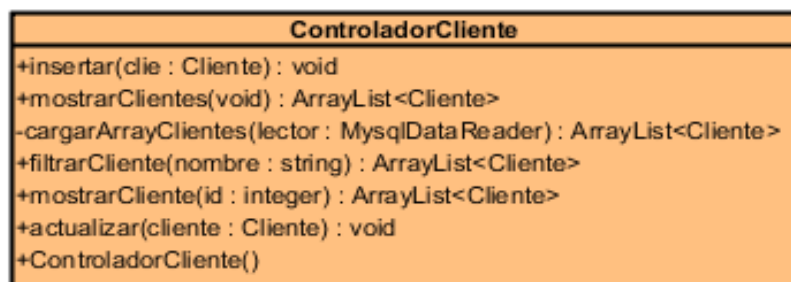


IMAGEN 12: clase ControladorCliente

- ❖ Insertar recibe un objeto de tipo cliente y hace la inserción de sus valores en la base de datos por medio de una Stored Procedure (procedimiento almacenado).
- ❖ MostrarClientes muestra al usuario todos los clientes

- ❖ cargarArrayClientes es un método interno solo accesible desde la clase y recibe el parámetro de tipo msqldatareader de todas las funciones de mostrar y filtrar, devolviendo un arraylist a las anteriores, evita que se repita código.
- ❖ FiltrarCliente, filtra por el nombre de un cliente, mostrando los similares al nombre pasado por el usuario.
- ❖ MostrarCliente devuelve un único cliente procedente de una consulta msql en Stored Procedure
- ❖ actualizarCliente envía a la base de datos una sentencia de actualización de los datos de un cliente en concreto por medio de un Stored Procedure.

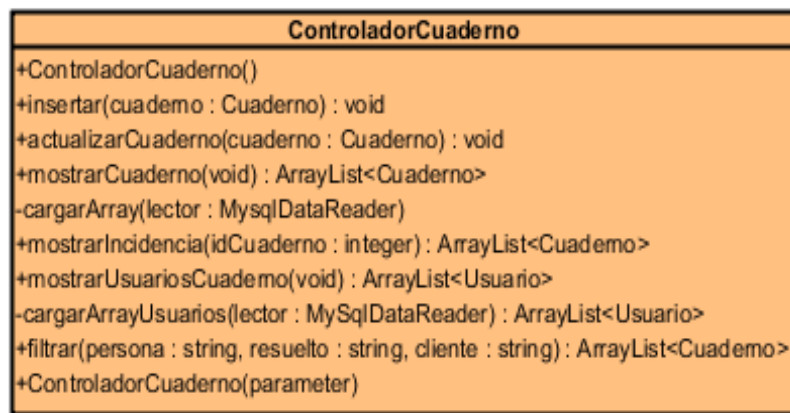


IMAGEN 13: clase ControladorCuaderno

- ❖ Insertar, inserta una incidencia completa en la base de datos por Stored Procedure
- ❖ Actualizar actualiza una incidencia en concreto en la base de datos por Stored Procedure
- ❖ mostrarCuaderno muestra todos los apuntes del cuaderno contenidos en la base de datos
- ❖ mostrarIncidencias recibe un arrayList de tipo cuaderno pero muestra una incidencia en concreto, se usa para reutilizar código
- ❖ mostrarUsuarioCuaderno muestra todos los usuarios disponibles a la hora de un nuevo apunte para poder seleccionar el correspondiente.
- ❖ cargarArrayUsuarios devuelve un array de usuarios para mostrarlos, es una función interna que se usa solamente para la clase, llamada por las funciones de mostrar e insertar, se usa para reutilizar código

- ❖ filtrar filtra por un determinado parámetro y devuelve de la base de datos los resultados de la consulta correspondiente.

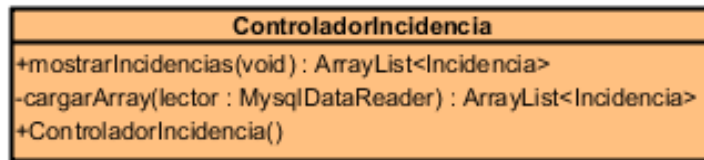


IMAGEN 14: clase ControladorIncidencia

- ❖ Devuelve un arraylist de todos los tipos de incidencias de la tabla correspondiente de la base de datos
- ❖ cargarArray se encarga de cargar el array de forma interna y devolverlo a mostrarIncidencias.

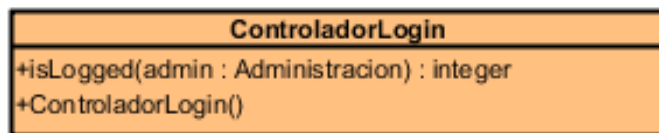


IMAGEN 15: clase ControladorLogin

- ❖ Recibe un objeto de tipo administración y comprueba si es correcto o no el login de usuario y contraseña, devolviendo un entero que se interpreta de forma interna como un verdadero o un falso.

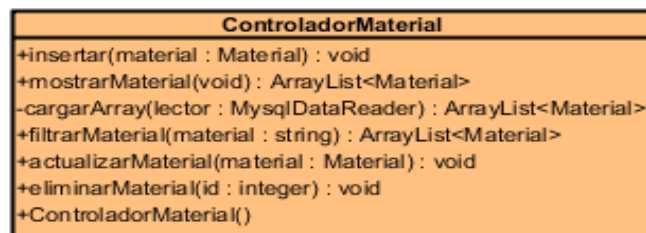


IMAGEN 16: clase ControladorMaterial

- ❖ insertar, inserta un material nuevo en la tabla correspondiente de la base de datos.
- ❖ mostrarMaterial, carga todos los materiales disponibles en la tabla.
- ❖ cargarArray carga de forma interna todos los objetos de tipo material y los devuelve en forma de arraylist a las funciones de mostrar y filtrar

- ❖ `filtrarMaterial` filtra todos los materiales dejando aquellos con cierta semejanza a un parámetro dado por el usuario
- ❖ `actualizarMaterial` actualiza los datos de un material en concreto.
- ❖ `eliminarMaterial`, elimina el material seleccionado de la tabla y por consiguiente, de la base de datos.

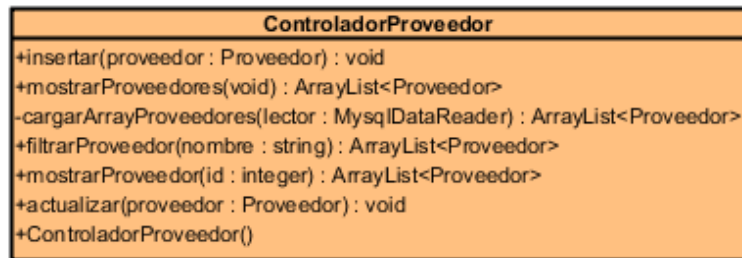


IMAGEN 17: clase ControladorProveedor

- ❖ `insertarProveedor` recibe un objeto de tipo proveedor, toma los valores con los métodos correspondientes a la clase proveedor y los inserta, por medio de un Stored Procedure en la tabla correspondiente de la base de datos.
- ❖ `mostrarProveedores`, muestra absolutamente todos los proveedores de la base de datos
- ❖ `cargarArrayProveedores`, es una función interna de la clase, que se usa para reutilizar código y devuelve un array con los proveedores que convenga, dependiendo de a qué método(`mostrarProveedores`, `filtrarProveedor`, `mostrarProveedor`) se le haya hecho la llamada al uso.

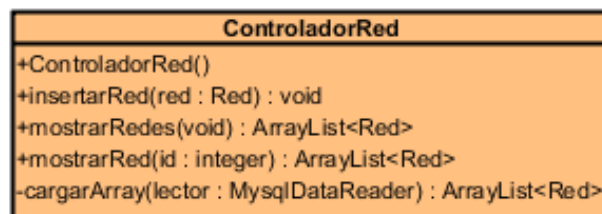


IMAGEN 18: clase ControladorProveedor

- ❖ `insertarRed` recibe un objeto de tipo red y lo inserta dentro de la tabla correspondiente pero comprobando si uno de sus atributos (es un arraylist de tipo puesto) está vacío o lleno, en caso de que esté lleno, se lo manda a la función `controladorPuesto` para que lo agregue en su lugar de la base de datos.

- ❖ mostrarRedes, devuelve en un arraylist de tipo red, todas las redes guardadas en la base de datos
- ❖ mostrarRed muestra una red en concreto
- ❖ cargarArray se utiliza de forma interna para cargar el array con los datos respectivos a la red, en cada caso, es una función implementada para reutilizar código.

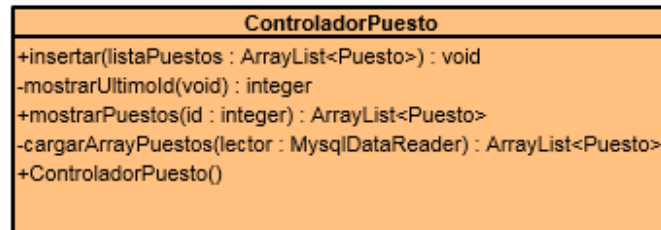


IMAGEN 19: clase ControladorPuesto

- ❖ ControladorPuesto forma parte de una relación con ControladorRed, recibe; si lo hay, un arraylist de tipo puesto para insertar una serie de puestos de trabajo relacionados con la red lan de los mismos puestos.
- ❖ mostrarUltimoId es un método interno de la clase que se usa para tomar la última id de la tabla "red" y así hacer el ingreso en la tabla datos con la clave tomada de la tabla red, usada en la misma como clave principal y almacenada en la tabla de puesto como clave foránea.
- ❖ mostrarPuestos, muestra todos los puestos relativos a una determinada red.
- ❖ CargarArrayPuestos, carga el array de tipo puesto para luego ser mostrado por el método mostrarPuestos, es un método interno que se usa para reutilizar el código.

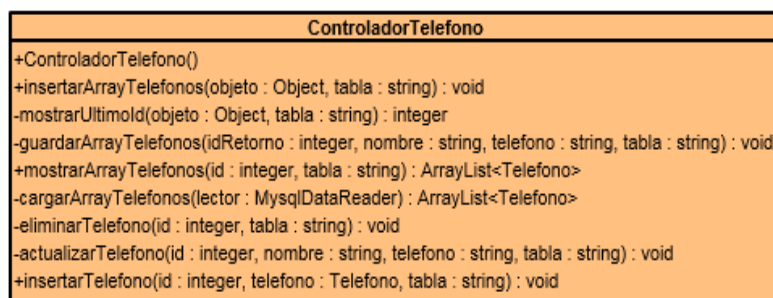


IMAGEN 20: clase ControladorTelefono

- ❖ ControladorTeléfono recibe los teléfonos tanto de clientes como de proveedores y los discrimina en sus respectivas tablas al uso, mediante el método insertarArray.

- ❖ `mostrarUltimoId` es un método análogo al existente en la clase `ControladorPuesto`.
- ❖ `cargarArray`, carga el array a mostrar por el método `mostrarArrayTelefonos`.
- ❖ `eliminarTelefono`, elimina un teléfono en concreto de la base de datos
- ❖ `actualizarTelefono` actualiza los datos existentes de un teléfono en concreto.
- ❖ `insertarTelefono`, inserta un teléfono nuevo en un cliente o proveedor determinado, se usa en tiempo de edición de un cliente o un proveedor, junto con el método `eliminarTelefono`.

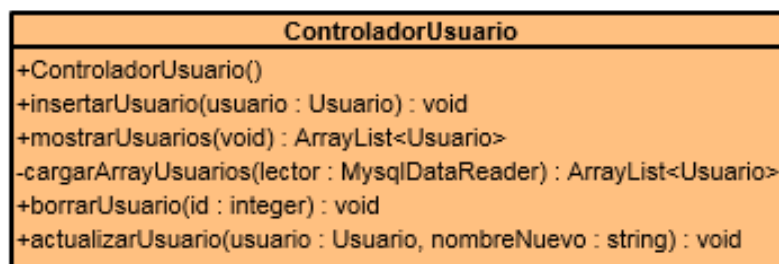


IMAGEN 21: clase ControladorUsuario

- ❖ `insertarUsuario`, inserta un nuevo usuario con el rol más bajo en la base de datos.
- ❖ `mostrarUsuarios`, muestra todos los usuarios de la aplicación existentes en la base de datos.
- ❖ `cargarArrayUsuarios`, es un método interno, se usa para cargar un array de usuarios y reutilizar código, este array se devuelve a `mostrarUsuarios`.
- ❖ `borrarUsuario`, elimina el usuario de la base de datos.
- ❖ `actualizarUsuario`, actualiza los datos de un usuario concreto, existente en la base de datos.

5.3 Interacción con el usuario

A continuación se procederá a comentar mediante textos cortos y diagramas toda la interacción con el usuario final.

5.3.1 Casos de uso/ historias de usuario e interacciones (más representativos)

Parte de trabajo

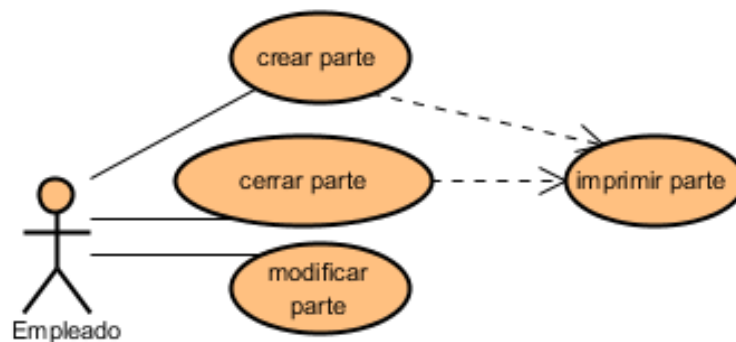
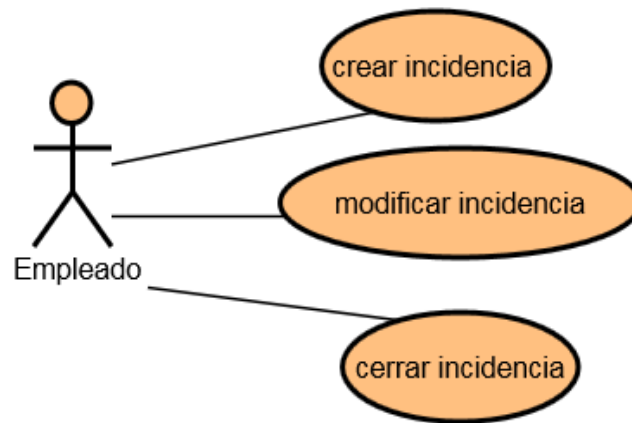


IMAGEN 22: caso de uso parte de trabajo

Actores involucrados	Empleado, jefe, administrador
Descripción	Acceder a partes de trabajo
Flujo básico	El actor accede a partes de trabajo y desde ahí puede crear un parte de trabajo nuevo, modificar un parte ya existente o bien cerrar un parte existente, Cuando crea o cierra el parte, tiene la opción de imprimirlo

Cuaderno de bitácora



Actores involucrados	Empleado, jefe , administrador
Descripción	Acceder al módulo de cuaderno de bitácora
Flujo básico	El actor accede al cuaderno de bitácora, puede crear una incidencia, modificarla o bien cerrarla, cuando ya esté solucionada, usando filtros de búsqueda para tales efectos

Crear incidencia

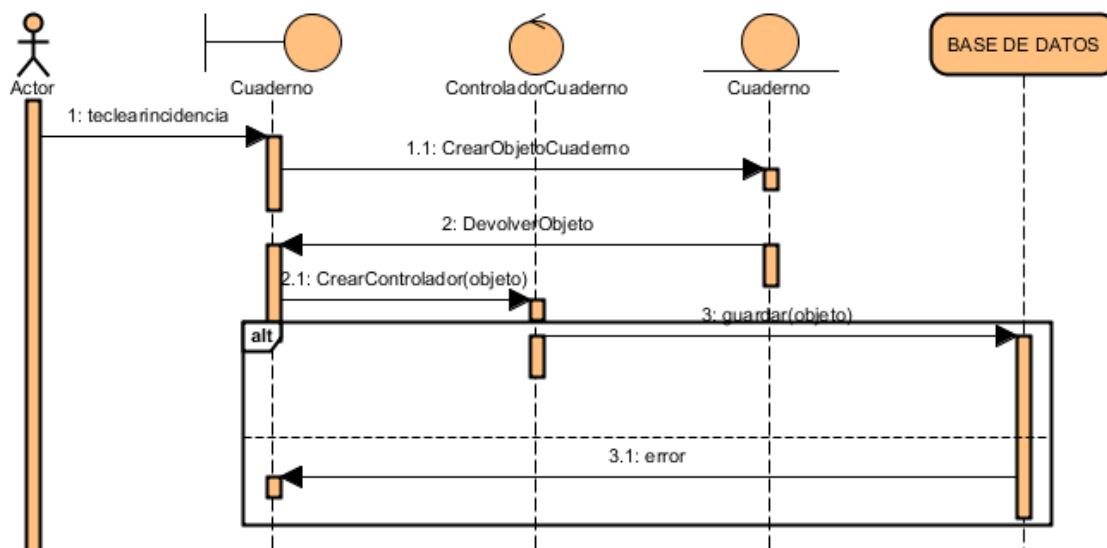


IMAGEN 23: diagrama de secuencia crear incidencia

- ❖ El usuario teclea una incidencia nueva y pincha en aceptar.

- ❖ Se crea un objeto de tipo cuaderno y a su vez se crea un controlador al que se le pasa este objeto.
- ❖ El controlador con el objeto usa el método de insertar e inserta los valores en la base de datos.
- ❖ Si ha habido un error, se muestra por pantalla.

Modificar incidencia

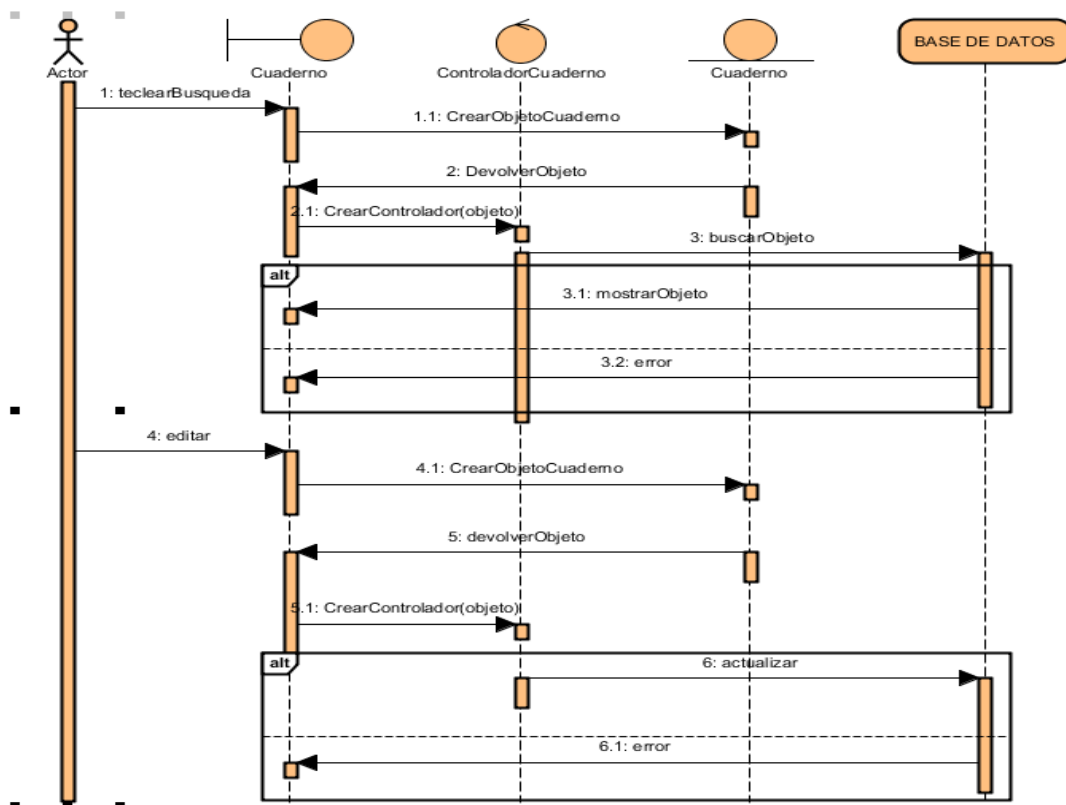


IMAGEN 24: Diagrama de secuencia modificar incidencia

- ❖ Se busca una incidencia
- ❖ Se crea el objeto de tipo Cuaderno
- ❖ Se crea el controlador y se le pasa el objeto cuaderno
- ❖ El controlador crea el método con el objeto para llamar al procedimiento de buscar de la base de datos pasándole lo necesario
- ❖ la base de datos devuelve el resultado y se muestra por pantalla, o bien devuelve un error

- ❖ Si ha ido todo bien, se actualizan los datos.
- ❖ Se crea un nuevo objeto y se pasa al controlador que a su vez crea el método para enviarlo a la base de datos.
- ❖ Se actualiza la información en la base de datos, si ha habido un error, se muestra por pantalla y la información no se actualiza

Cerrar incidencia

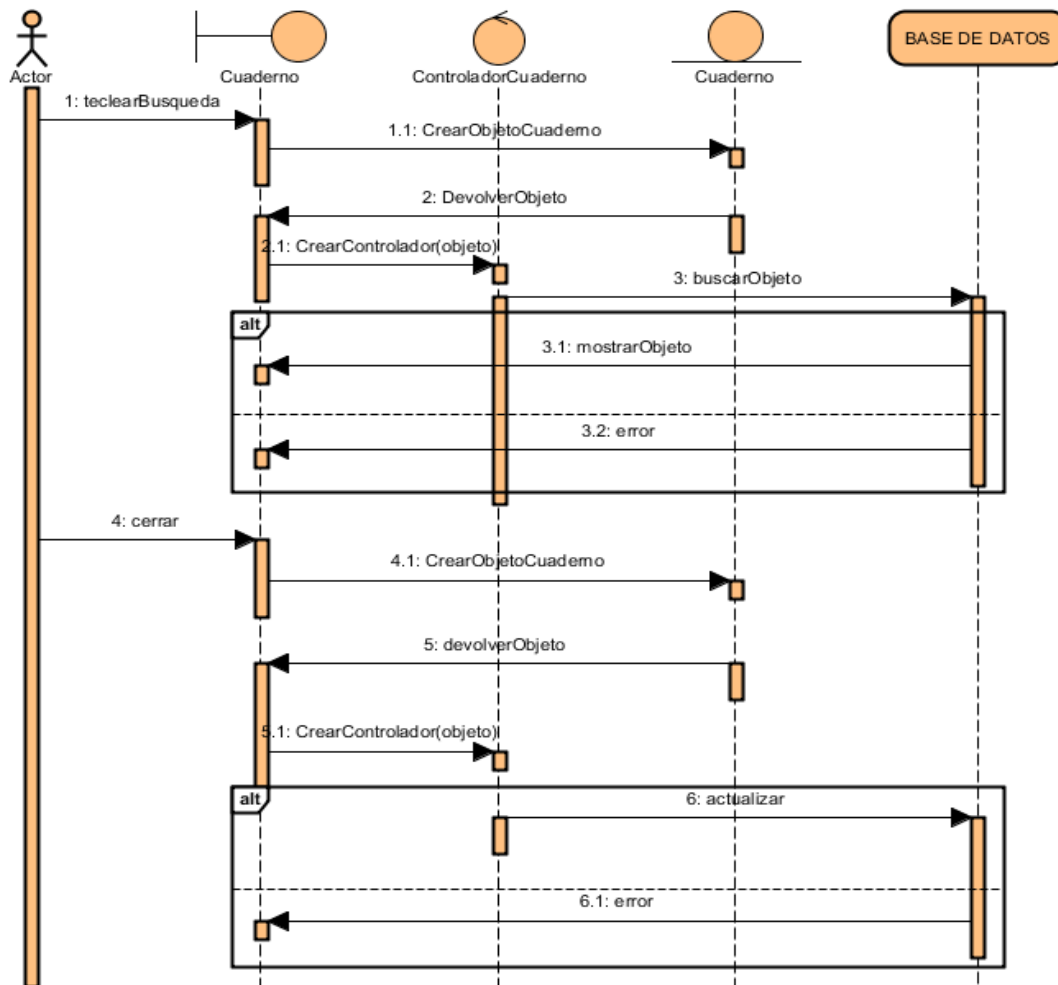


IMAGEN 25: diabrama de secuencia cerrar incidencia

- ❖ El usuario teclea la incidencia aproximada o entera a buscar.
- ❖ Se crea el objeto de tipo cuaderno con los datos del usuario y a su vez se crea el controlador al que se le pasa dicho objeto usando el método de buscar.
- ❖ Se busca el objeto o los objetos en la base de datos y se devuelven en formato de array que se mostrará por pantalla al usuario.

- ❖ El usuario escoge el deseado y lo modifica cerrando la incidencia marcándola como resuelta.
- ❖ Se crea otro objeto Cuaderno y un controlador para tal objeto y se llama al método de acutalizar pasandole dicho objeto, se hace la llamada a la base de datos pasando los valores del objeto y los actualizará en la misna.
- ❖ Si ha habido un error , se mostrará por pantalla.

Cliente

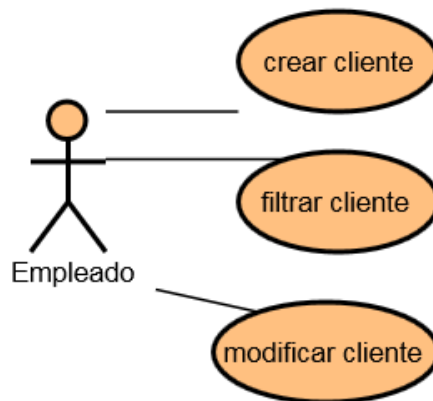


IMAGEN 26: caso de uso cliente

Actores involucrados	Jefe, Administrador
Descripción	Acceder al módulo de gestión de clientes
Flujo básico	El actor accede al módulo de clientes y puede crear un cliente nuevo, modificar un cliente existente o bien, filtrar para buscar un cliente y modificarlo a posteriori

Crear cliente

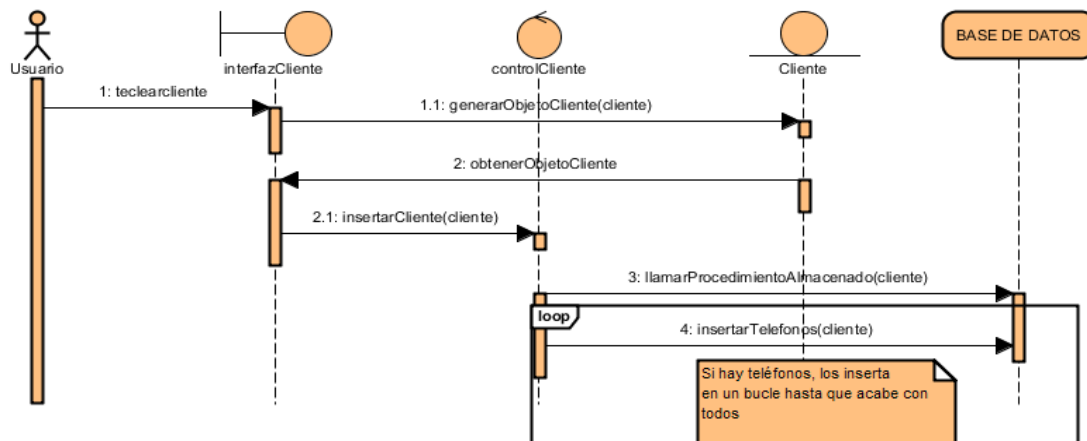


IMAGEN27: diagrama de secuencia crear cliente

- ❖ Se rellenan los datos de un cliente que puede ser un particular o una empresa.
- ❖ Se crea el objeto de tipo cliente y un controlador al uso para almacenarlo en la base de datos
- ❖ Se guarda el cliente en la base de datos y se recorre un array de teléfonos que se guardarán en una tabla de teléfonos relacionada con el cliente en cuestión.

Filtrar cliente

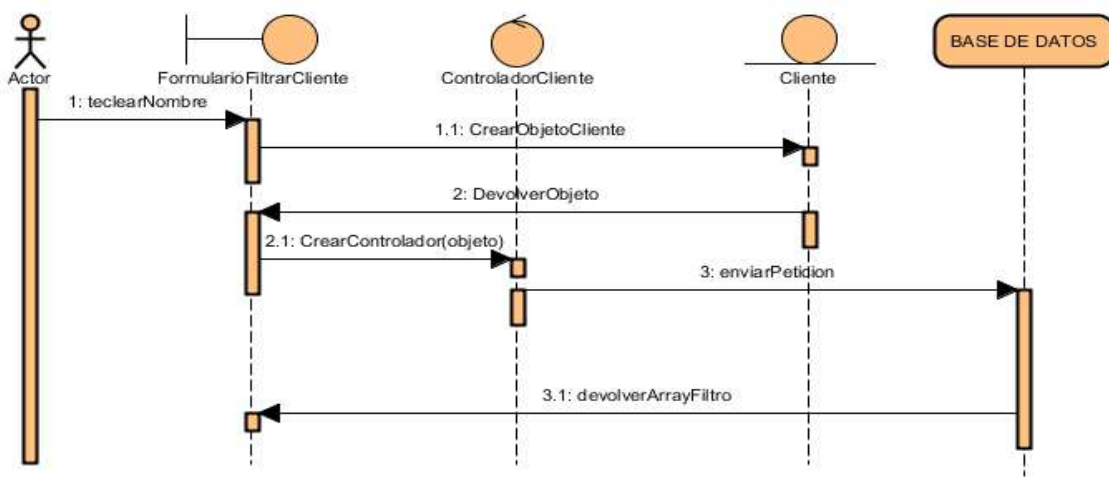


IMAGEN 28: diagrama de secuencia filtrar cliente

- ❖ Se teclea el nombre del cliente a buscar o un nombre aproximado.
- ❖ Se crean los objetos y los métodos apropiados para enviar sus valores a la base de datos y hacer una consulta de selección.
- ❖ La consulta se almacena en un array y se muestra al usuario para que escoja el que sea oportuno.

Modificar cliente

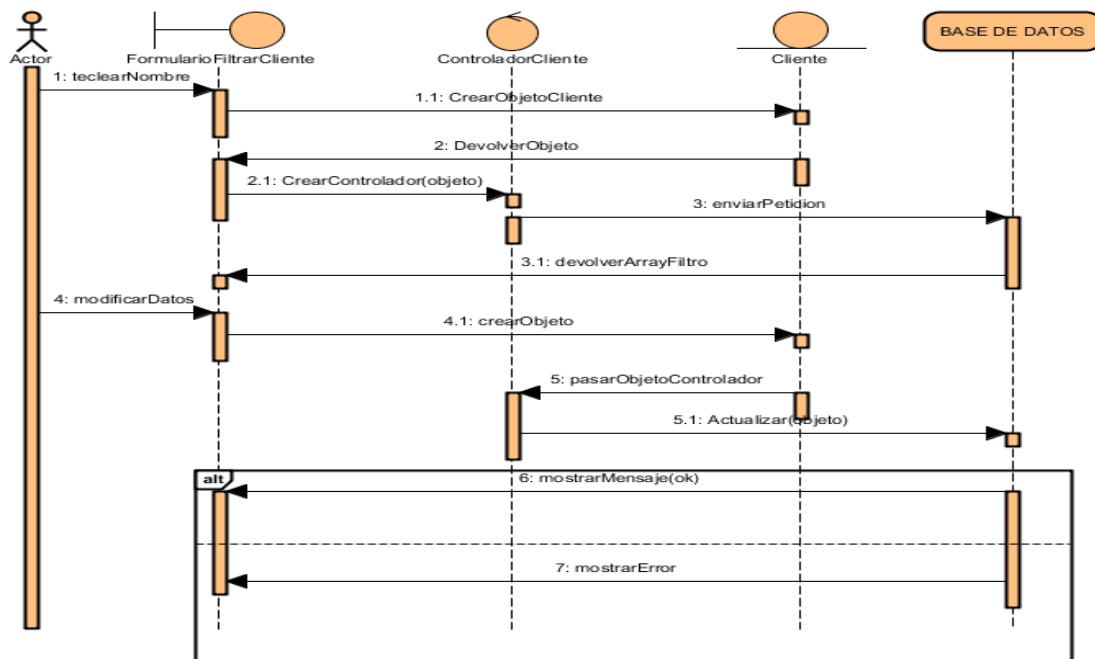


IMAGEN 29: diagrama de secuencia modificar cliente

- ❖ Se teclea un nombre o un nombre aproximado.
- ❖ Se crea el objeto de tipo cliente y el método apropiado para enviar la información a la base de datos
- ❖ La base de datos ejecuta la operación de selección y devuelve los resultados que se almacenan en un array y se mostrarán al usuario.
- ❖ El usuario escoge uno específico para modificar su contenido, lo edita .
- ❖ Al actualizar se crea el objeto y el controlador al uso, se llama al método para que actualice la información pasando el objeto como parámetro.
- ❖ La base de datos recibe los valores y actualiza su información.

Proveedor

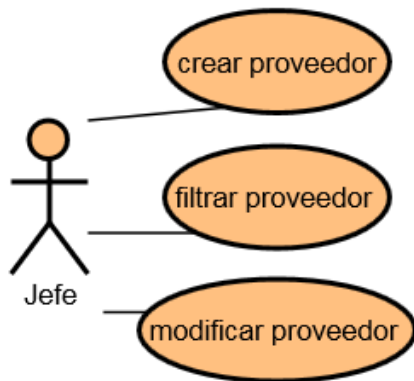


IMAGEN 30: caso de uso proveedor

Actores involucrados	Administrador, jefe
Descripción	Acceder al módulo de proveedores
Flujo básico	Igual que en la parte de clientes, se puede crear un proveedor nuevo, modificar sus datos filtrándolo en las búsquedas

Crear proveedor

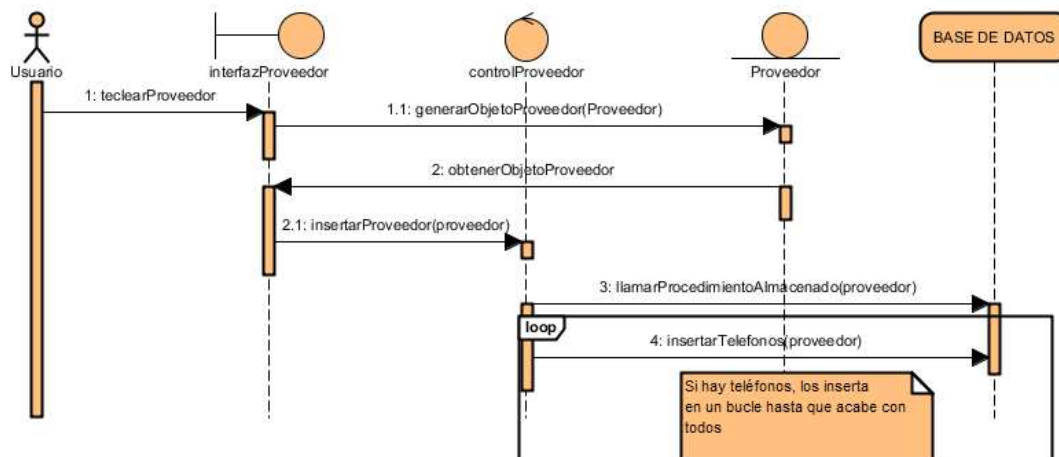


IMAGEN 31: diagrama de secuencia crear proveedor

- ❖ Se rellenan los datos de un proveedor que puede ser un particular o una empresa.
- ❖ Se crea el objeto de tipo proveedor y un controlador al uso para almacenarlo en la base de datos

- ❖ Se guarda el proveedor en la base de datos y se recorre un array de teléfonos que se guardarán en una tabla de teléfonos relacionada con el proveedor en cuestión.

Filtrar proveedor

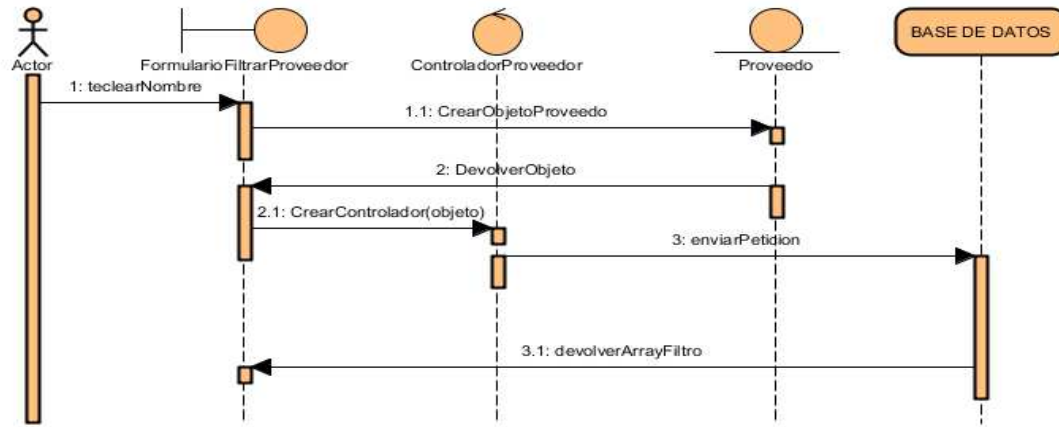


IMAGEN 32: diagrama de secuencia filtrar proveedor

- ❖ Se teclea el nombre del proveedor a buscar o un nombre aproximado.
- ❖ Se crean los objetos y los métodos apropiados para enviar sus valores a la base de datos y hacer una consulta de selección.
- ❖ La consulta se almacena en un array y se muestra al usuario para que escoja el que sea oportuno.

Modificar proveedor

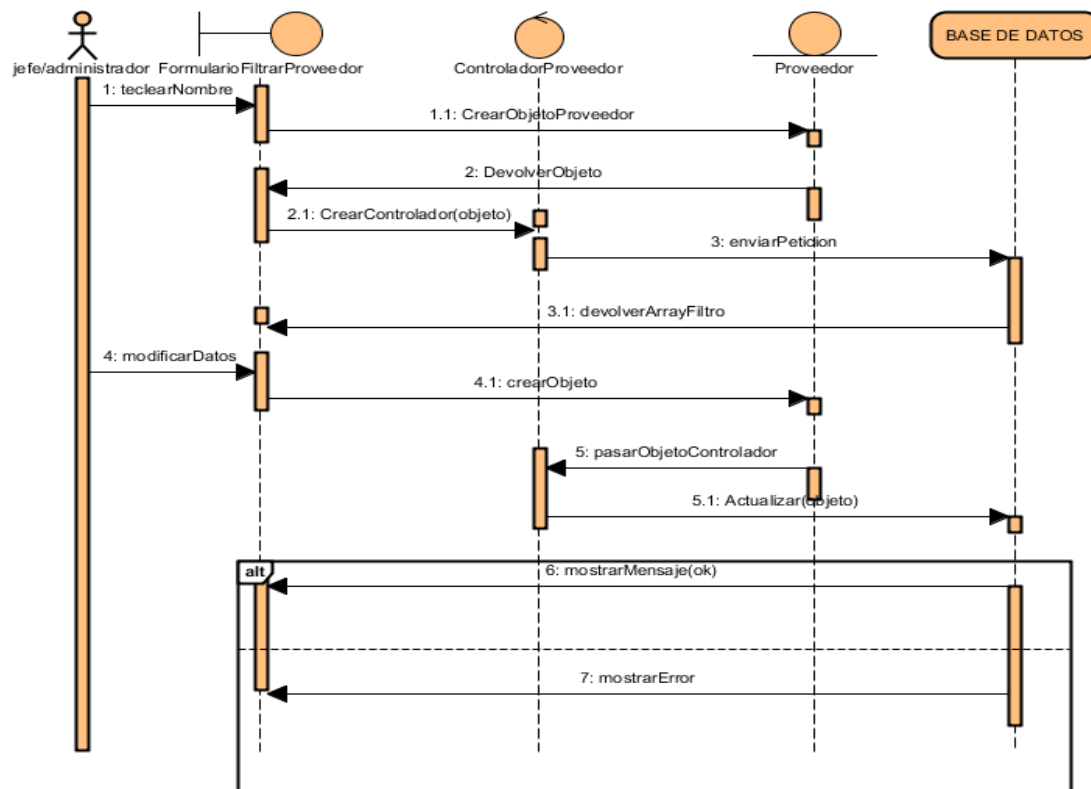


IMAGEN 33: diagrama de secuencia modificar proveedor

- ❖ El actor teclea un nombre completo o aproximado de un proveedor.
- ❖ Se crea el objeto proveedor.
- ❖ Se crea el controlador específico pasándole como parámetro el objeto proveedor
- ❖ Envía la petición de búsqueda a la base de datos.
- ❖ devuelve un array con todos los proveedores que concuerden con la búsqueda
- ❖ El actor escoge el que le interese y modifica los datos que quiera.
- ❖ Se crea otro objeto con los datos actualizados.
- ❖ Se crea otro controlador con el nuevo objeto actualizado como parámetro
- ❖ Se actualiza la información en la base de datos.
- ❖ Si ocurriera un error, se mostraría por pantalla al usuario

Usuarios

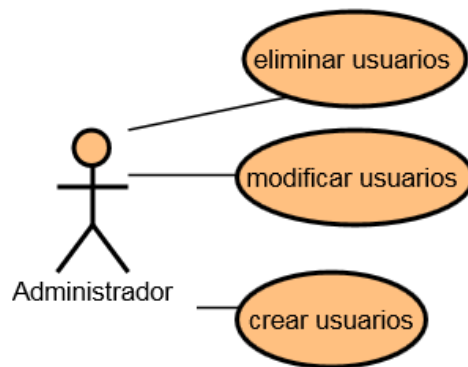


IMAGEN 34: caso de uso usuarios

Actores involucrados	Jefe
Descripción	Acceder al módulo de usuarios
Flujo básico	El actor accede para crear un usuario nuevo, modificar un usuario existente o bien, eliminar un usuario si éste deja de existir

Eliminar usuario

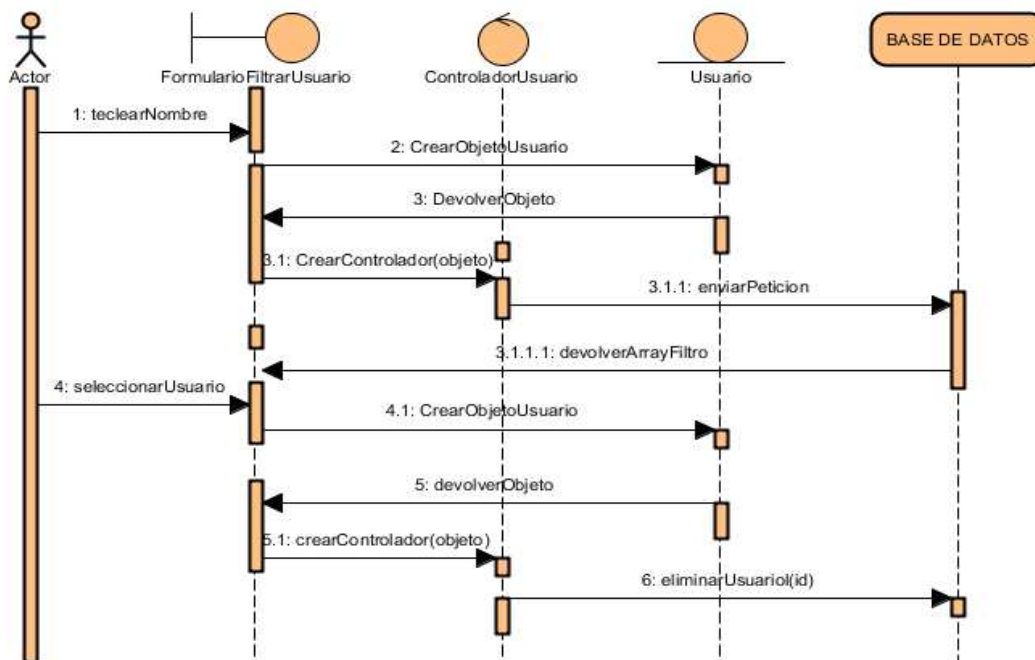


IMAGEN 35: diagrama de secuencia eliminar usuario

- ❖ El usuario teclea un nombre para buscar el usuario o los usuarios relacionados con la búsqueda.

- ❖ Se crea el controlador, se crea el objeto y se llama al método al uso para hacer la consulta del usuario.
- ❖ Se hace la consulta acorde al patrón introducido por el usuario en la base de datos y se devuelve un array de los usuarios que tengan algo que ver.
- ❖ El actor selecciona un usuario específico y se elimina de la base de datos.

Modificar usuario

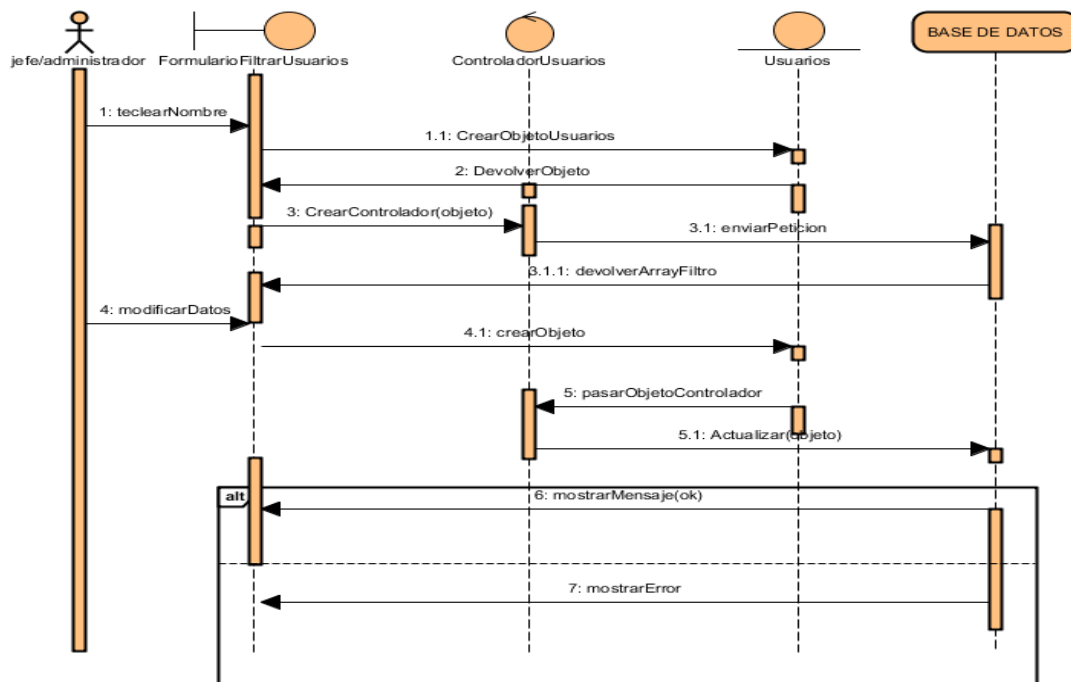
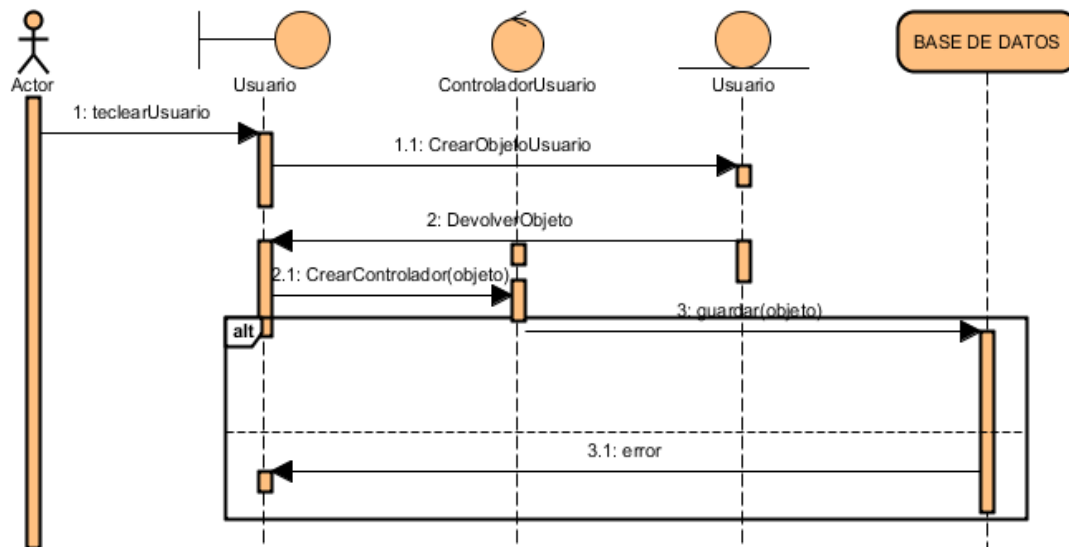


IMAGEN 36: diagrama de secuencia modificar usuario

- ❖ El actor teclea un nombre completo o aproximado de un usuario.
- ❖ Se crea el objeto usuario.
- ❖ Se crea el controlador específico pasándole como parámetro el objeto usuario.
- ❖ Envía la petición de búsqueda a la base de datos.
- ❖ devuelve un array con todos los usuario que concuerden con la búsqueda.
- ❖ El actor escoge el que le interese y modifica los datos que quiera.
- ❖ Se crea otro objeto con los datos actualizados.
- ❖ Se crea otro controlador con el nuevo objeto actualizado como parámetro

- ❖ Se actualiza la información en la base de datos.
- ❖ Si ocurriera un error, se mostraría por pantalla a lusuario

Alta usuario



IMAGNE 37: diagrama de secuencia alta usuario

- ❖ El actor teclea el usuario; se crea un objeto de tipo material.
- ❖ Se crea el controlador y se llama al método correspondiente para agregar el usuario en la base de datos.
- ❖ Si ha habido algún error, se mostrará por pantalla y el usuario no se guardará.

Material

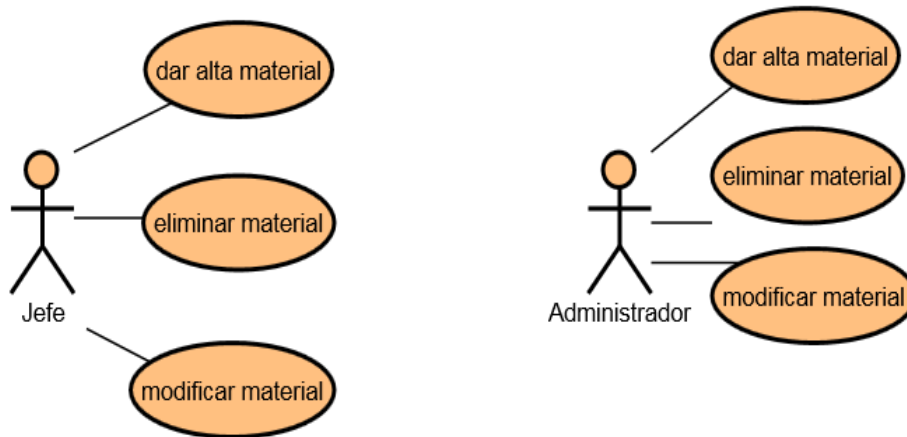


IMAGEN 38: caso de uso material

Alta material

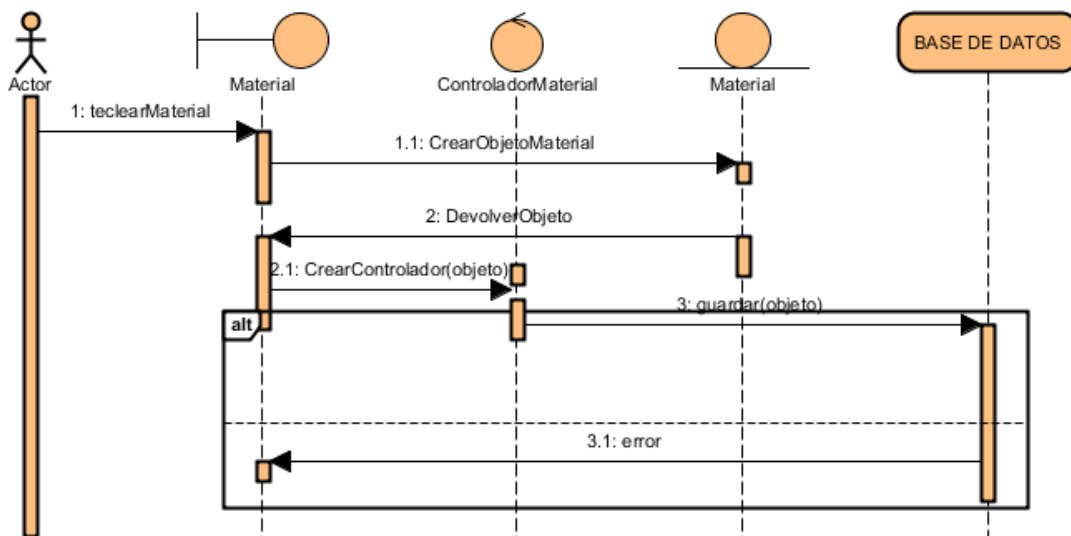


IMAGEN 39: caso de uso alta material

- ❖ El actor teclea el material y cuando acabe, se crea un objeto de tipo material.
- ❖ Se crea el controlador y se llama al método correspondiente para agregar el material en la base de datos.

- ❖ Se agrega el material en la base de datos.
- ❖ Si ha habido algún error, se mostrará por pantalla y el material no se guardará.

Eliminar material

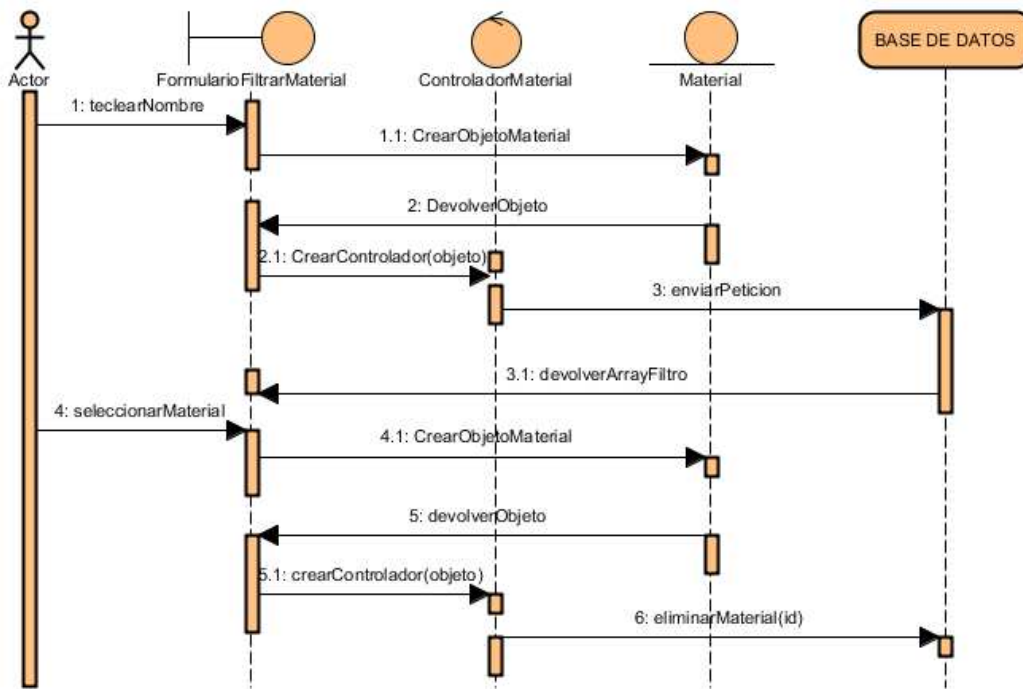


IMAGEN 40: caso de uso eliminar material

- ❖ El usuario teclea un nombre para buscar el material o los materiales relacionados con la búsqueda.
- ❖ Se crea el controlador, se crea el objeto y se llama al método al uso para hacer la consulta del material.
- ❖ Se hace la consulta acorde al patrón introducido por el usuario en la base de datos y se devuelve un array de los materiales que tengan algo que ver.
- ❖ El usuario selecciona un material específico y se elimina de la base de datos.

Modificar material

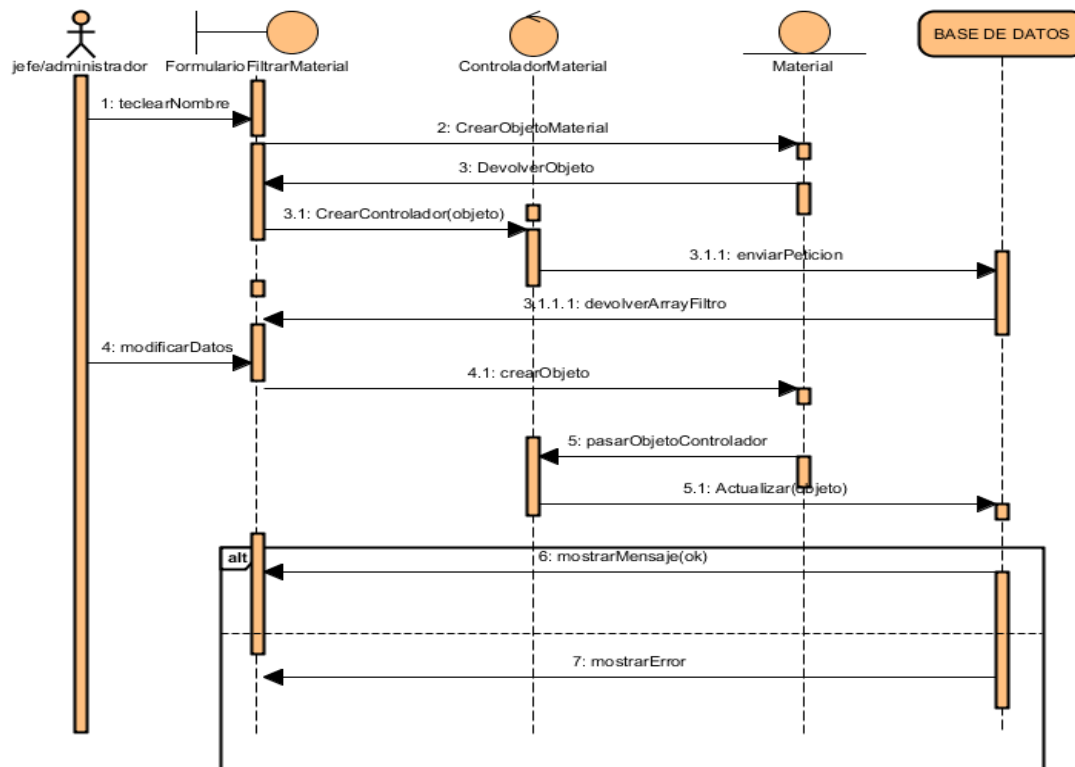
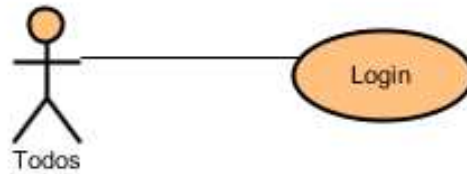


IMAGEN 41: caso de uso modificar material

- ❖ El actor teclea un nombre completo o aproximado de un material.
- ❖ Se crea el objeto material.
- ❖ Se crea el controlador específico pasándole como parámetro el objeto material.
- ❖ Envía la petición de búsqueda a la base de datos.
- ❖ devuelve un array con todos los materiales que concuerden con la búsqueda.
- ❖ El actor escoge el que le interese y modifica los datos que quiera.
- ❖ Se crea otro objeto con los datos actualizados.
- ❖ Se crea otro controlador con el nuevo objeto actualizado como parámetro
- ❖ Se actualiza la información en la base de datos.
- ❖ Si ocurriera un error, se mostraría por pantalla a lusuario

Login



Actores involucrados	Jefe, administrador
Descripción	Acceder al módulo de login para gestionar la aplicación
Flujo básico	Se accede mediante usuario y contraseña, se comprueban los roles y dependiendo de cuales sean, se asignan unas características u otras

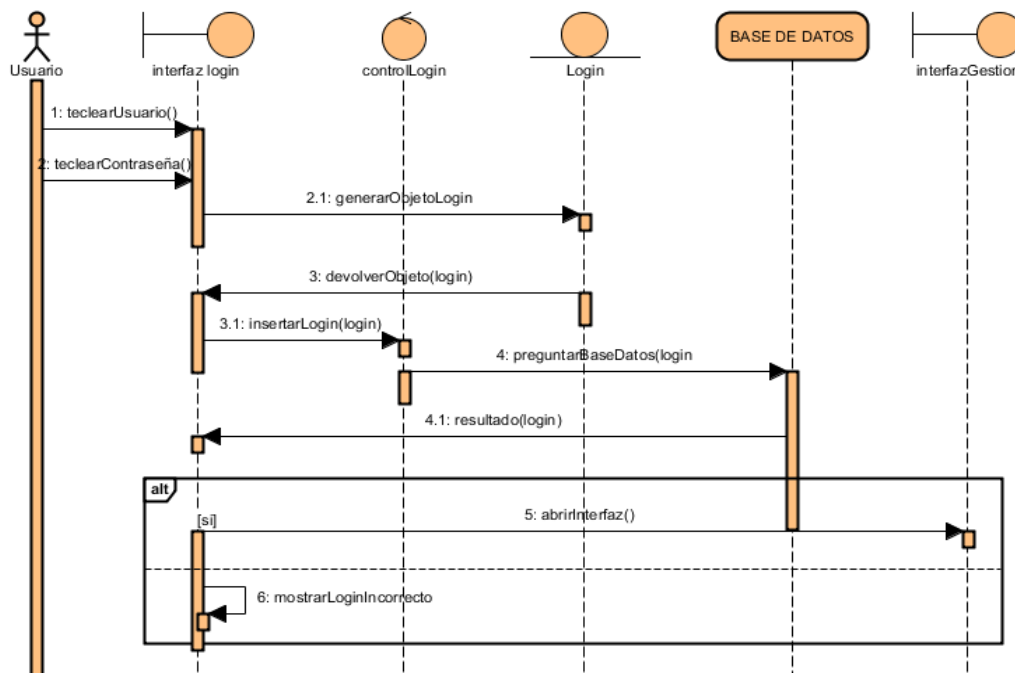


IMAGEN 42: caso de uso login

- ❖ El usuario teclea un nombre de usuario y una contraseña.
- ❖ Se crea un objeto de tipo login desde la interfaz.
- ❖ Desde la interfaz también se crea on objeto de tipo insertarLogin pasándole el objeto de tipo login.
- ❖ Hace la pregunta en la base de datos y devuelve un resultado positivo o negativo.
- ❖ Si el resultado es positivo, se abre la interfaz de gestión de la aplicación.

- ❖ Si es negativo muestra que el login es incorrecto.

Correo

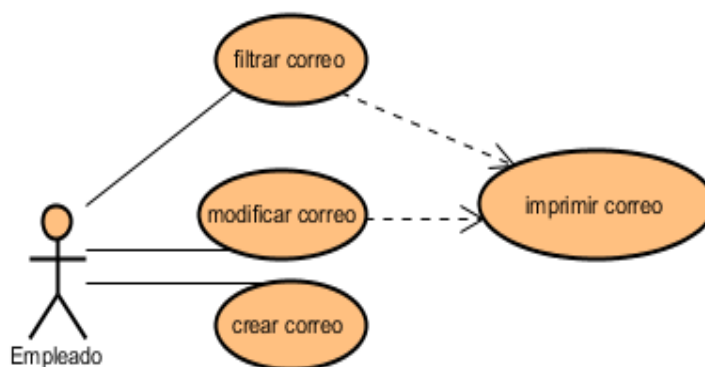


IMAGEN 43: caso de uso correo

Actores involucrados	Jefe, administrador, usuario
Descripción	Acceder al módulo de correos
Flujo básico	Acceder al módulo de correos para crear un correo nuevo, o bien modificarlo, o bien; agregar más cuentas para ese cliente. Con opción a imprimir

Red

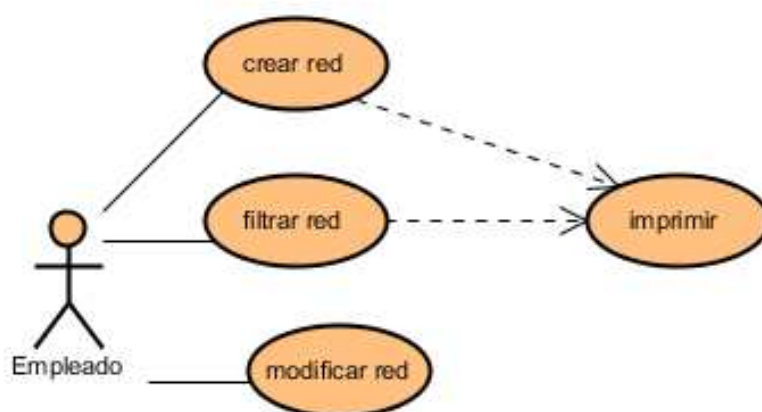


IMAGEN 44: caso de uso red

(análogo al de correos)

5.3.3 Diseño de la interfaz

La interfaz final ha sido diseñada careciendo de texto, se ha intentado un diseño con imágenes minimalista, las cuales acompaña la descripción propia de cada una mediante el uso de tooltips, los cuales se despliegan al hacer un situar el puntero del ratón encima de la imagen.

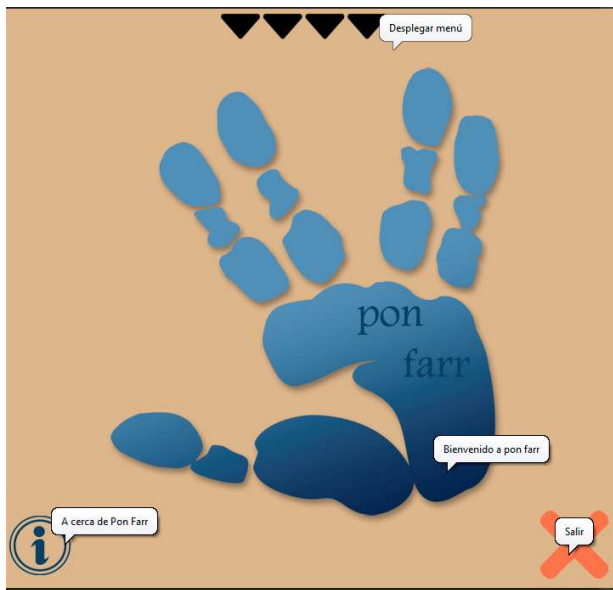


IMAGEN 45 : Pantalla principal de la aplicación

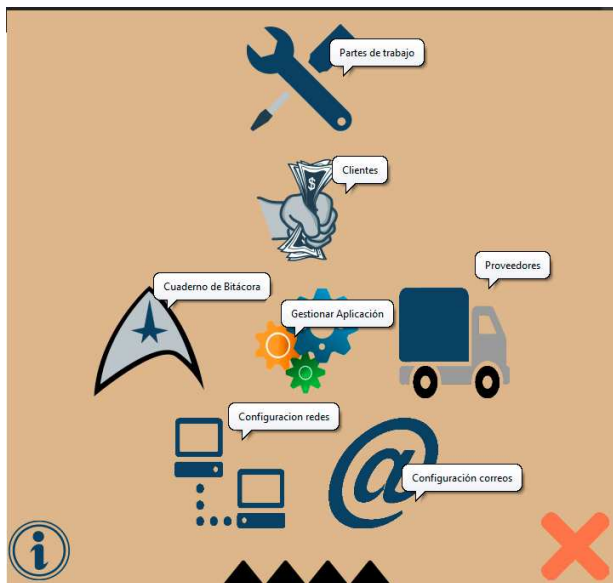


IMAGEN 46 : Pantalla principal de la aplicación desplegada



IMAGEN 47 : Pantalla principal de clientes, pantalla tipo para el resto de la aplicación

5.3.4 Diagrama de Gantt

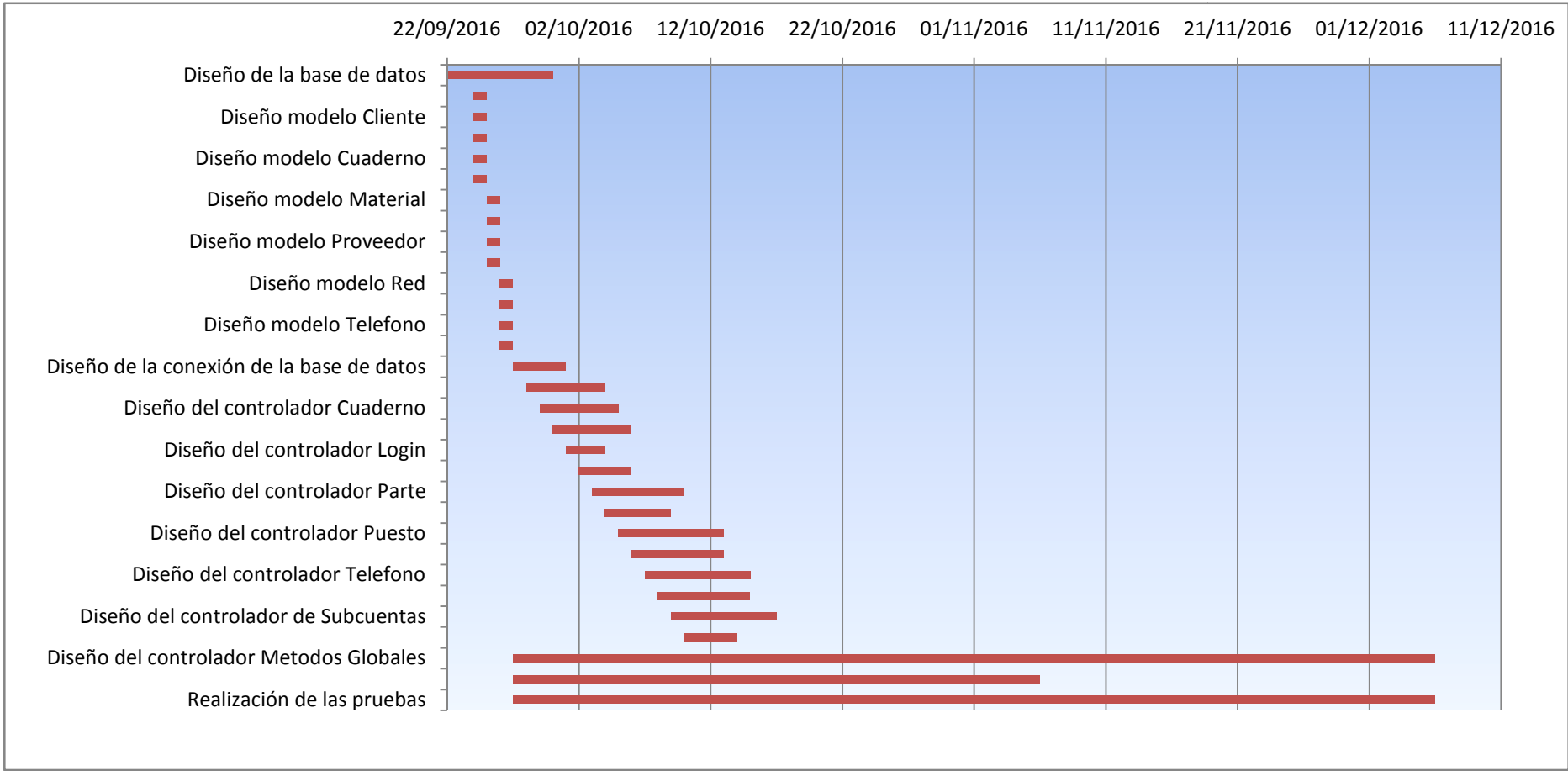


IMAGEN 48: diagrama de gannt real

5.4 CÓDIGO FUENTE. PROCESOS MÁS REPRESENTATIVOS.

5.4.1 Métodos deshabilitar/ habilitar

```
PublicSub deshabilitar()  
Dim miControl AsControl  
ForEach miControl InMy.Application.OpenForms  
If (TypeOf (miControl) IsForm) Then  
CType(miControl, Form).Hide()  
EndIf  
Next  
EndSub
```

```
PublicSub habilitar()  
Dim miControl AsControl  
ForEach miControl InMy.Application.OpenForms  
If (TypeOf (miControl) IsForm) Then  
CType(miControl, Form).Show()  
EndIf  
Next  
EndSub
```

Lo que hace este método es recorrer los formularios que estén abiertos en la aplicación, y minimiza todos menos el formulario que llama a este método.

Al llamar a su contrario, todos los formularios que quedaron ocultos con el método deshabilitar, vuelven a ser visibles.

5.4.2 Método isEmpty

Este método recibe un objeto y va comprobando de qué tipo es el objeto que recibo para ir al case determinado y comprobar si se han rellenado todos los datos o bien hay algún campo vacío, mostrando un mensaje al uso.

```
PublicFunction isEmpty(ByVal objeto AsObject) AsBoolean  
Dim resultado AsBoolean  
Dim nombreControl AsString  
Dim miControl AsControl  
Dim nombreObjeto AsString = objeto.Name  
'la idea de este select case es que recorra el formulario en cuestión que tenga  
para rellenar  
'mediante un select case comprueba qué nombre de formulario recibe y si se ha  
rellenado bien  
'o mal  
SelectCase nombreObjeto  
' comprobación para una nueva incidencia para el cuaderno de bitácora  
Case"frmCuadernoNuevaIncidencia", "frmCuadernoEditarIncidencia"  
ForEach miControl In objeto.Controls  
If (TypeOf (miControl) IsTextBox) Then  
If miControl.Text = ""Then  
mensaje = "Has de rellenarlo todo"  
resultado = True  
frmAdvertencia.Show()  
ExitFor  
Else
```

```

                                resultado = False
EndIf
EndIf
Next
' para las pestañas de la parte de gestion de la aplicación
Case"TabPage1"
ForEach miControl In objeto.controls
If (TypeOf (miControl) IsTextBox) Then
                                nombreControl = miControl.Name
If miControl.Text = ""And nombreControl.CompareTo("txtFiltrar") <> 0 Then
mensaje = "Has de rellenar todos los precios y el tipo de material que es"
                                resultado = True
                                frmAdvertencia.Show()

ExitFor
Else
                                resultado = False

EndIf
EndIf
Next
Case"TabPage3"
ForEach miControl In objeto.controls
If (TypeOf (miControl) IsTextBox) Then
If miControl.Text = ""Then
                                mensaje = "Has de darme un nombre"
                                resultado = True
                                frmAdvertencia.Show()

ExitFor
Else
                                resultado = False

EndIf
EndIf
Next
' comprobación para el caso de un parte nuevo
Case"frmParteNuevo"
ForEach miControl In objeto.Controls
If (TypeOf (miControl) IsTextBox) Then
nombreControl = miControl.Name
If miControl.Text = ""And nombreControl.CompareTo("txtSolucion") <> 0 Then
mensaje = " la solución es el único cuadro que puedes dejar en blanco," +
" junto con el técnico y el material cuando creas un nuevo parte"
                                frmAdvertencia.Show()
                                resultado = True

ExitFor
Else
                                resultado = False

EndIf
EndIf
Next
Case"frmClienteNuevo", "frmProveedorNuevo"
' comprobación para el caso de agregar cliente o agregar proveedor
ForEach miControl In objeto.Controls
If (TypeOf (miControl) IsTextBox) Then
                                nombreControl = miControl.Name
If miControl.Text = ""Then
If nombreControl.CompareTo("txtObservaciones") <> 0 And
nombreControl.CompareTo("txtNom") <> 0 And nombreControl.CompareTo("txtTlf") <>
0 Then
                                mensaje = "Sólo puedes dejar en blanco el cuadro
de observaciones"
                                frmAdvertencia.Show()
                                resultado = True

ExitFor

```

```

Else
                                resultado = False
EndIf
EndIf
EndIf
Next
EndSelect
Return resultado
EndFunction

```

5.4.3 Método que vacía los campos de texto

Este método lo que hace es recibir un objeto donde se contienen todos los campos y el método recorre todos los controles buscando los que sean cuadros de texto, si encuentra alguno, lo vacía.

```

PublicSub vaciar(ByVal objeto AsObject)
Dim miControl AsControl
ForEach miControl In objeto.Controls
If (TypeOf (miControl) IsTextBox) Then
    miControl.Text = ""
EndIf
If (TypeOf miControl IsComboBox) Then
'esta parte no funciona directamente con miControl, porque no tiene el método
selectedIndex
'entonces si el control es un combobox, entonces me crea una variable de tipo
combobox
'que sí tiene el método selected index, igualo el control a la variable de tipo
combobox
'y sólo me queda vaciar el combo
Dim combo AsComboBox
    combo = miControl
    combo.SelectedIndex = -1
EndIf
Next
EndSub

```

5.4.4 Método conectar/desconectar de base de datos

```

' función que conecta con la base de datos y devuelve una variable de tipo
mysqlconnection
PublicFunction conectar() AsMySQLConnection
Try
    conexion = "Server=localhost;database=gestion;Uid=root;Pwd="
    conn = NewMySQLConnection(conexion)
Return conn
Catch ex AsException
    mensaje = ex.Message
    frmAdvertencia.Show()
EndTry
EndFunction
'función que recibe una variable de tipo mysqlconnection y se desconecta de la
base de datos
'después de desconectar, el método dispone libera los posibles residuos que
queden
PublicSub desconectar(ByVal conexion AsMySQLConnection)
Try
    conexion.Close()
Catch ex AsException

```

```

        mensaje = ex.Message
        frmAdvertencia.Show()
Finally
conexion.Dispose()
EndTry
EndSub

```

Se instancia un objeto de esta clase para llamar a la base de datos y hacer las distintas consultas, una vez se ha acabado, se llama al método desconectar para anular y eliminar la conexión a la base de datos

5.4.5 Métodos para el cliente

El resto de métodos son similares, se escoge éste a modo de representación general

```

PublicSub insertar(ByVal clie AsCliente)

Dim con AsNew MySqlConnection
Dim conexion AsNew ConexionBaseDeDatos
Dim comando AsNew MySqlCommand
Try
con = conexion.conectar
    comando.Connection = con
comando.CommandText = "insertarCliente"
    comando.CommandType = CommandType.StoredProcedure
comando.Parameters.AddWithValue("_nombre", clie.Nombre1)
comando.Parameters.AddWithValue("_nif", clie.Nif1)
comando.Parameters.AddWithValue("_personaContacto", clie.PersonaContacto1)
comando.Parameters.AddWithValue("_email", clie.Email1)
comando.Parameters.AddWithValue("_observaciones", clie.Observaciones1)
    comando.Parameters.AddWithValue("_domicilio", clie.Domicilio1)
    comando.Parameters.AddWithValue("_poblacion", clie.Poblacion1)
    comando.Parameters.AddWithValue("_provincia", clie.Provincia1)
    comando.Parameters.AddWithValue("_codpos", clie.Codpos1)
con.Open()
comando.ExecuteNonQuery()
mensaje = "Cliente agregado"
frmAdvertencia.Show()
conexion.desconectar(con)

```

una de las variables de tipo cliente es un arraylist que almacena teléfonos lo que hago es ver si tiene algún dato ingresado, si es así entonces llama a controladorTeléfono para insertar los datos que tenga, pasándole el objeto de tipo cliente y el nombre de la tabla de destino para guardar los objetos

```

If clie.ArrayTelefonos1.Count <> 0 Then
Dim controlador AsNew ControladorTelefono
    controlador.insertarArrayTelefonos(clie, "cliente")
EndIf
Catch ex AsException
    mensaje = ex.Message
    frmAdvertencia.Show()
EndTry
EndSub

```

Función que me muestra todos los clientes de la base de datos y me los guarda en unarraylist


```

PublicFunction mostrarClientes() AsArrayList
Dim arrayClientes AsNewArrayList
Dim con AsNewMySQLConnection
Dim conexion AsNewConexionBaseDeDatos
Dim comando AsNewMySQLCommand
Dim lector AsMySQLDataReader
Try
    con = conexion.conectar
    comando.Connection = con
    comando.CommandText = "mostrarClientes"
    comando.CommandType = CommandType.StoredProcedure
    con.Open()
    lector = comando.ExecuteReader()
arrayClientes = cargarArrayClientes(lector)
    conexion.desconectar(con)

Catch ex AsException
    mensaje = ex.Message
frmAdvertencia.Show()

EndTry
Return arrayClientes
EndFunction

```

esta función es privada porque sólo la necesita usar esta clase de forma interna no hace falta tenerla de forma pública para acceder desde fuera. recibe una variable lector que es de tipo mysqldatareader, la gestiona leyendo los datos que hay dentro de la variable , los va almacenando en un arraylist y devuelve el arraylist

```

PrivateFunction cargarArrayClientes(ByVal lector AsMySQLDataReader) AsArrayList
Dim arrayClientes AsNewArrayList
Dim controlador AsNewControladorTelefono
Dim listaTelefonos AsNewArrayList
If lector.HasRows Then
Dowhile lector.Read
    por cada vuelta, me creo un objeto de tipo cliente y le añado sus valores
    finalmente almaceno ese nuevo objeto cliente en el arraylist
Dim clie AsNewCliente()
        clie.Id1 = lector.GetInt32(0)
        clie.Nombre1 = lector.GetString(1)
        clie.Nif1 = lector.GetString(2)
        clie.PersonaContacto1 = lector.GetString(3)
        clie.Email1 = lector.GetString(4)
clie.Observaciones1 = lector.GetString(5)
        clie.Domicilio1 = lector.GetString(6)
        clie.Poblacion1 = lector.GetString(7)
        clie.Provincial1 = lector.GetString(8)
        listaTelefonos = controlador.mostrarArrayTelefonos(clie.Id1,
"cliente")
        clie.ArrayTelefonos1 = listaTelefonos
        clie.Codpos1 = lector.GetString(9)
arrayClientes.Add(clie)

Loop
EndIf
Return arrayClientes
EndFunction

```

función que filtra todos los clientes atendiendo a un nombre en concreto o que tenga unos caracteres específicos

```

PublicFunction filtrarCliente(ByVal nombre AsString) AsArrayList

```

```

Dim arrayClientes AsNewArrayList
Dim con AsNewMySQLConnection
Dim conexion AsNewConexionBaseDeDatos
Dim comando AsNewMySQLCommand
Dim lector AsMySQLDataReader
Try
    con = conexion.conectar
    comando.Connection = con
    comando.CommandText = "filtrarClientes"
    comando.CommandType = CommandType.StoredProcedure
    comando.Parameters.AddWithValue("_filtro", nombre)
    con.Open()
    lector = comando.ExecuteReader()
    arrayClientes = cargarArrayClientes(lector)
    conexion.desconectar(con)

Catch ex AsException
    mensaje = ex.Message
    frmAdvertencia.Show()
EndTry
Return arrayClientes
EndFunction

```

función que me buscar un cliente en concreto y me lo muestra, me lo guarda en un arraylist, no hace falta pero así ahorro código para mostrarlo usando la función cargarArrayClientes

```

PublicFunction mostrarCliente(ByVal id AsInt32) AsArrayList
Dim arrayClientes AsNewArrayList
Dim con AsNewMySQLConnection
Dim comando AsNewMySQLCommand
Dim conexion AsNewConexionBaseDeDatos
Dim lector AsMySQLDataReader
Try
    con = conexion.conectar
    comando.Connection = con
    comando.CommandText = "mostrarCliente"
    comando.CommandType = CommandType.StoredProcedure
    comando.Parameters.AddWithValue("_id", id)
    con.Open()
    lector = comando.ExecuteReader
    arrayClientes = cargarArrayClientes(lector)
    conexion.desconectar(con)

Catch ex AsException
    mensaje = ex.Message
    frmAdvertencia.Show()
EndTry
Return arrayClientes
EndFunction

```

función que actualiza un cliente determinado alguno de sus valores

```

PublicSub actualizar(ByVal clie AsCliente)
Dim con AsNewMySQLConnection
Dim comando AsNewMySQLCommand
Dim conexion AsNewConexionBaseDeDatos
Try
    con = conexion.conectar
    comando.Connection = con
    comando.CommandText = "ActualizarCliente"

```

```

        comando.CommandType = CommandType.StoredProcedure
        comando.Parameters.AddWithValue("_id", clie.Id1)
comando.Parameters.AddWithValue("_nombre", clie.Nombre1)
comando.Parameters.AddWithValue("_nif", clie.Nif1)
comando.Parameters.AddWithValue("_personaContacto", clie.PersonaContacto1)
comando.Parameters.AddWithValue("_email", clie.Email1)
comando.Parameters.AddWithValue("_observaciones", clie.Observaciones1)
        comando.Parameters.AddWithValue("_domicilio", clie.Domicilio1)
        comando.Parameters.AddWithValue("_poblacion", clie.Poblacion1)
        comando.Parameters.AddWithValue("_provincia", clie.Provincial)
        comando.Parameters.AddWithValue("_codpos", clie.Codpos1)
        con.Open()
        comando.ExecuteNonQuery()
        mensaje = "Registro actualizado"
        frmAdvertencia.Show()
        conexion.desconectar(con)
Catch ex As Exception
mensaje = ex.Message
        frmAdvertencia.Show()

EndTry
EndSub

```

5.4.6 Stored procedure insertar teléfonos

```

CREATE DEFINER='root'@'localhost' PROCEDURE `insertarTelefonos`(in _idRetorno integer,
in _nombre varchar(15),
in _telefono varchar(12),
in _nombreTabla varchar(15)
)
begin
        case _nombreTabla
        when "cliente" then
insert into telefonoscliente(id_cliente,nombre,telefono) values(_idRetorno, _nombre, _telefono);
        when "proveedor" then
insert into telefonosproveedor (id_proveedor, nombre, telefono) values(_idRetorno, _nombre, _telefono);
        end case;
end$$

```

A este método se le llama reiteradamente hasta que se acaba el array de insertar teléfonos y discrimina en qué tabla insertar el array con la variable de nombreTabla.

5.4.7 Stored procedure insertar(se escoge el de cliente, el resto son iguales)

```

CREATE DEFINER='root'@'localhost' PROCEDURE `insertarCliente`(IN _nombre` VARCHAR(150), IN _nif` VARCHAR(9),
IN _personaContacto` VARCHAR(150), IN _email` VARCHAR(150), IN _observaciones` TEXT, IN _domicilio` VARCHAR(150),
IN _poblacion` VARCHAR(150), IN _provincia` VARCHAR(50), IN _codpos` VARCHAR(5))
begin
        insert into clientes (nombre,nif,perscto,email,observaciones,domicilio,poblacion,provincia,codpos)
values( _nombre, _nif, _personaContacto, _email, _observaciones, _domicilio, _poblacion, _provincia, _codpos);
end$$

```

Ejemplo de stored procedure de inserción, el resto son similares a este y se toma como ejemplo. Recibe unos valores procedentes del código y los inserta en la base de datos.

6 FASE DE PRUEBAS

Se crearán dos tipos de pruebas , unas relacionadas con el código que abarcarán tanto el código .Net como los Stored Procedure de la base de datos y otro serán pruebas de interface relativas al funcionamiento visual.

6.1 Pruebas realizadas

Prueba 1	Test conexión base de datos
Descripción	Probar la conexión y desconexión de la base de datos y el uso de la variable de tipo connection en el código .net
Resultado esperado	Conexión establecida y variable válida
Resultado obtenido	Imposible conectar, error en el string de conexión
Solución	Modificar el string de conexión a la base de datos por estar erróneo "Server=localhost;database=gestion;Uid=root;Pwd="

Prueba 2	Pruebas de los stored procedure de inserción
Descripción	Probar todos los stored procedures de inserción de datos y que se almacenen correctamente
Resultado esperado	Valores insertados correctamente en la base de datos
Resultado obtenido	Se cumple con el resultado esperado
Solución	

Prueba 3	Pruebas de los stored procedure de actualización
Descripción	Probar todos los stored procedures de actualización de datos y que se almacenen correctamente
Resultado esperado	Valores insertados correctamente en la base de datos
Resultado obtenido	Se cumple con el resultado esperado
Solución	

Prueba 4	Pruebas de los stored procedure de eliminación
Descripción	Probar todos los stored procedures de eliminación de datos
Resultado esperado	Valores insertados correctamente en la base de datos
Resultado obtenido	Se cumple con el resultado esperado
Solución	

Prueba 5	Prueba de login
Descripción	Probar la parte de login de la aplicación y que haga lo que se desea tanto si el usuario y la contraseña son correctos como si cualquiera de los dos anteriores son incorrectos
Resultado esperado	Si usuario y/o contraseña incorrectos error. Si usuario y contraseña correctos, logueado.
Resultado obtenido	Se cumple con el resultado esperado
Solución	

Prueba 6	Vaciar campos de texto
Descripción	Método único para vaciar todos los campos de texto ante una nueva inserción de un dato
Resultado esperado	Se espera un vaciado de todos los campos usables por el usuario , del formulario
Resultado obtenido	No funciona con los combo box
Solución	Crear una variable de tipo combobox e igualar al objeto que se recibe en el método si el objeto es tipo combo, para poder vaciarla.

Prueba 7	Creación de la base de datos si no existe
Descripción	La primera vez que se ejecute el programa, detectar que si la base de datos no existe, se cree al completo antes de que el usuario pueda usar la aplicación
Resultado esperado	Creación de la base de datos
Resultado obtenido	Cumple el resultado esperado
Solución	

Prueba 8	Probar que los datos se agregan a la base de datos tal cual
Descripción	Ver si hay cortes en los datos a la hora de agregarlos en la base de datos
Resultado esperado	Inserción de los datos tal cual los introduce el usuario
Resultado obtenido	Algunos datos se cortan en la base de datos
Solución	Restringir el número de caracteres en el formulario, al mismo número de la base de datos

Prueba 9	Restringir entrada de datos a sólo 5 números en el textbox de código postal
Descripción	El cuadro de texto correspondiente al código postal, sólo debe admitir números y un máximo de cinco
Resultado esperado	Se espera que sólo se puedan teclear cinco números
Resultado obtenido	No funciona con los combo box
Solución	Quando el código postal recibe el foco para escribir, borrar el contenido del portapapeles mediante la expresión <code>Clipboard.Clear()</code>

Prueba 10	Filtro de búsqueda de usuarios al teclear caracteres
Descripción	El método <code>on textchange</code> del textbox del usuario hacer que haga una consulta cada vez que se escriba un caracter para mostrar coincidencias
Resultado esperado	Se espera un filtro por nombre de usuarios
Resultado obtenido	Mensaje de error: " la cadena no tiene el formato correcto"
Solución	Corregir el método <code>getString(numero)</code> del <code>mysqldatareader</code> para tomar el campo correspondiente. <code>lector.getString(0)</code>

Prueba 11	Insertar y mostrar nuevo registro pero no todos los campos
Descripción	Crear un nuevo registro sin insertar todos los campos null
Resultado esperado	Insertar y visualizar sin problema
Resultado obtenido	Error con campos null
Solución	Pasar el contenido vacío de todos los campos en el objeto e insertarlos en la tabla para que haya campos vacíos no null

Prueba 12	Reportes
Descripción	Probar que los reportes reciben los parámetros correctos, los imprimen, y que la parte de reportes maestro- detalle, recibe bien la fuente de datos para el detalla
Resultado esperado	Mostrar los reportes correctamente
Resultado obtenido	Los reportes no se muestran correctamente y al imprimir muestra más páginas de la cuenta
Solución	Comprobar y rectificar que los nombres de los parámetros coincidan, para que las referencias en el reporte funcionen. Redimensionar los campos para que encajen con las medidas y márgenes de impresión de un folio tamaño DIN A-4.

7 CONCLUSIONES FINALES

7.1 Reflexión

El aspecto más importante de la realización de este proyecto es el aprendizaje de las fases a seguir y la estructura necesaria para el desarrollo y finalización de un proyecto. Estos conocimientos se podrán aplicar a cualquier aplicación que se tenga que desarrollar en un futuro.

El desarrollo de un proyecto requiere de mucho tiempo, esfuerzo y dedicación; debido básicamente a que se están demostrando los conocimientos adquiridos y aplicando todos los conceptos que se han ido aprendiendo durante los años de estudio del ciclo formativo. Además, la elaboración del proyecto comporta un aprendizaje que quedará reflejado en el mismo proyecto realizado.

Se ha intentado realizar el proyecto de manera profesional y procurando en todo momento realizar una codificación clara y entendedora con tal de facilitar que otro programador que tenga que realizar un módulo de la aplicación, lo pueda llevar a cabo sin gran dificultad.

Aunque siempre existen aspectos que podrían ser mejorados, se considera que los objetivos establecidos inicialmente, se han podido cumplir en gran medida.

7.2 Grado de cumplimiento de los objetivos fijados

Los objetivos que se pretendían conseguir en el desarrollo de este proyecto eran, por un lado, gestionar cada una de las necesidades de REDES ZAMORA S.L. ; mediante un sistema informático que fuera una herramienta útil e intuitiva que facilitara las tareas de gestión de la empresa de manera clara y eficaz.

El primero de los objetivos ha cumplido con las expectativas establecidas. No sólo por el hecho de conseguir realizar de forma completa y individualmente una aplicación, sino por el hecho de ampliar conocimientos, aprovechando el tiempo realizando esta aplicación para enriquecerse profesionalmente.

La aplicación final integra un correcto diseño de la aplicación, manteniendo un estilo sencillo, claro y útil que hace uso de colores para mostrar correctamente la información.

7.3 Propuesta de modificaciones y/o ampliaciones

Todo sistema informático evoluciona y mejora con el tiempo, introduciendo nuevas características y funcionalidades.

Estas modificaciones o ampliaciones se pueden originar, por ejemplo, debido a la aparición de nuevas necesidades dentro de la empresa o aspectos no contemplados anteriormente.

Actualmente, ya se pueden predecir algunas líneas de desarrollo de cara a una futura versión ampliada.

Se podría implementar un módulo para gestionar los trabajos pendientes de los trabajadores, mediante un calendario, que permitiera obtener hojas diarias de tareas.

Otra mejora importante sería implementar la aplicación en aplicación web mediante una intranet utilizando las herramientas al uso y conseguir por ejemplo, mediante un uso "responsive", que la aplicación se pueda visualizar perfectamente desde cualquier tipo de dispositivo (multidispositivo).

Diseñar una aplicación específica para android de algún módulo que se necesite, como por ejemplo el cuaderno de bitácora, para recibir incidencias y llamadas al teléfono móvil y poder gestionarlas desde donde sea que se encuentre el usuario (multiplataforma).

Diseñar la aplicación al completo en modo cliente servidor, donde el servidor recibe las conexiones de todos los clientes por medio de sockets, éste envía la petición a la base de datos, la base de datos devuelve el resultado al servidor y éste, a su vez, se lo devuelve al cliente que lo ha solicitado a través del socket creado al uso.

Los filtros de búsqueda se podrían implementar de tal forma que de una única caja de texto, se buscara todo en la base de datos, y no separarlo en las cajas de texto relativas a lo que se quiera buscar

Ampliar la aplicación con un módulo dedicado a la facturación, generador de hojas de cálculo y gráficos en los que se muestren los ingresos y los gastos así como un generador de nóminas para todos los usuarios

Sería interesante también, barajar la idea de concatenar los permisos, ya que ahora mismo un usuario sólo puede tener un rol, si pudiera tener más de uno concatenado, se podría personalizar más la experiencia con el usuario.

Implementar mejor todo el sistema de validaciones ya que, por falta de tiempo, hay ciertas validaciones que han quedado en el tintero.

Usar la tecnología WPF para diseñar la aplicación como aplicación de escritorio, instalable; tanto en red como en modo stand alone.

8 DIFICULTADES Y PROBLEMAS FUNDAMENTALMENTE ENCONTRADOS

Como resumen de estos, se exponen los principales problemas aparecidos durante el desarrollo de la aplicación.

Por un lado, la creación de la arquitectura MySQL - vb.net ha sido difícil de integrar teniendo en cuenta que la base de datos se maneja en su totalidad con stored procedures, este produjo un retraso en el desarrollo, pero finalmente se puede decir que esta arquitectura ha permitido generar correctamente la aplicación en su totalidad.

Una incidencia importante durante el desarrollo de la aplicación ha sido el solucionar los problemas de codificación de acentos y otros caracteres, ya que eran mostrados de manera incorrecta.

Generar los informes de maestro-detalle atrajo algo de complicación a la hora de realizar el binding con la subclase correspondiente. Siendo varios ensayos hasta dar con la solución correcta.

Por último la más difícil de desarrollar, ha sido la de integrar los reportes correspondientes a la parte de correos y red, ya que todo lo que reciben es por medio de parámetros y no por datagrid, el problema era generar los parámetros dinámicamente porque no se sabe a priori cuántos recibirá el informe.

10 BIBLIOGRAFÍA

10.1 Fuentes en papel

Nombre	Programming with Microsoft Visual Basic
Autores	Diane Zak
Editorial	Cenage Learning
Edición	2012

10.2 Fuentes digitales

<https://www.mysql.com/>

<http://answers.microsoft.com/es-es/?auth=1>