[*TÍTULO DEL PROYECTO*]

**[*Nombre y Apellidos del Alumno*]**

C.F.G.S. DAM.

Curso [*Año académico*]

**[*Nombre y Apellidos del Tutor*]**

# Resumen.

[*Escribir un pequeño resumen de qué va a ser el proyecto, y las fases en las que lo habéis dividido. Esto se realiza, por regla general, al final del proyecto*]

# Índice de contenidos.

[Resumen.](#h.itvcfy97ybkq)

[Índice de contenidos.](#h.ijj4kg86unu7)

[1. Introducción.](#h.r5wwy31ql9wm)

[1.1. Presentación del proyecto.](#h.hu6xecrih243)

[1.2. Justificación y contexto en el que se desarrolla el Proyecto Integrado.](#h.atnl9xd8euq5)

[1.3. Objetivos del Proyecto Integrado.](#h.hklqdzmf4imv)

[Objetivos para la empresa [nombre de la empresa real o ficticia]](#h.x6xjsi2jusun)

[Objetivos para el alumno](#h.u3ys7qhp1v35)

[1.4. Enfoque y método seguido.](#h.re3nghcivex0)

[1.5. Planificación del proyecto.](#h.5ta5rqld1x30)

[1.6. Recursos del proyecto.](#h.5j8t4uq94vqr)

[2. Análisis de requisitos.](#h.dm8v77iqyqo)

[2.1. Descripción de actores.](#h.60c4bv7k2xdy)

[2.2. Características del sistema.](#h.ah6w55z9sxbe)

[2.3. Especificación de requerimientos funcionales.](#h.gw4tp4augvm)

[2.4. Diagrama de caso de uso general.](#h.su5i9cw7ncwu)

[2.5. Diseño de prototipo.](#h.oefcqc19h973)

[2.6. Requerimientos no funcionales del sistema.](#h.1xbov726cim7)

[3. Diseño del sistema.](#h.ua5r1pb3w2h7)

[3.1. Diseño preliminar.](#h.z1bajlvgtxj1)

[3.1.1. Diagramas de Casos de Uso del sistema.](#h.86bp29nx169s)

[3.1.2. Diagramas de Secuencia.](#h.7m0h2418ruqy)

[3.1.3. Diagramas de Colaboración](#h.fbg2xff2oeqv)

[3.1.4. Diagrama de Clases](#h.v21hi8nbf2gn)

[3.1.4.1. Identificación de Clases](#h.g8lcgjb561tz)

[3.1.3.2. Atributos y Métodos de las Clases.](#h.onck7saj65it)

[3.1.3.3. Relaciones entre Clases.](#h.abx8brobd64a)

[3.1.3.4. Diagrama de Clases completo.](#h.tc5n8ohj6d4o)

[3.2. Diseño Detallado.](#h.rmg8eojt9tpv)

[3.2.1. Diagrama de Componentes.](#h.sjo0fvdpbcov)

[3.2.2. Diagrama de Despliegue.](#h.6kc3jo9047io)

[3.3. Diseño de la Base de Datos.](#h.bdtzn2rfvo7u)

[3.3.1. Nivel lógico.](#h.crwrjjxatqv5)

[3.3.2. Nivel Físico.](#h.n826w464gc7o)

[3.3. Implantación de la Base de Datos.](#h.opemhlhrs8tn)

[4. Codificación de la solución.](#h.p80kk6ujzfci)

[4.1. Planteamiento de la estrategia de programación.](#h.ls6rszfcgxyf)

[4.2. Especificaciones técnicas para el desarrollo.](#h.5wqtumfmkh5l)

[4.2.1. Requerimientos técnicos de Hardware.](#h.a3o6022wqfhm)

[4.2.2. Requerimientos técnicos de Software.](#h.e3r54lzgze42)

[4.3. Descripción general de la solución creada.](#h.6e3zer4tec6j)

[5. Pruebas de unidad.](#h.himvbjzgvmb4)

[6. Pruebas de Integración.](#h.wofn8ohk9b9s)

[7. Mantenimiento.](#h.7q4kgzwe8r0q)

[8. Futuras ampliaciones.](#h.2b9j35myzsem)

[9. Valoración económica.](#h.a6qhwmuq9jt1)

[10. Conclusiones.](#h.b2ovhfmevt2x)

# 1. Introducción.

## 1.1. Presentación del proyecto.

[*En este apartado tenéis que poner la descripción del proyecto que me habéis presentado. Pasar a leer el punto 3.1.4, ya que es muy importante que el enunciado sea coherente en todo momento.*]

## 1.2. Justificación y contexto en el que se desarrolla el Proyecto Integrado.

[*En este apartado tenéis que justificar la razón por la que hacéis dicho proyecto (es para una empresa, para vuestro uso personal, etc), así como justificar la solución que se va a proporcionar, que se va a utilizar en la implementación (lenguajes de programación, herramientas, bases de datos, tecnologías WEB, etc)*]

## 1.3. Objetivos del Proyecto Integrado.

Los objetivos del proyecto serán divididos en dos tipos distintos: objetivos a conseguir con el proyecto para la empresa [*nombre de la empresa real o ficticia*] y objetivos propios para el alumno.

### Objetivos para la empresa [*nombre de la empresa real o ficticia*]

* [*Objetivo 1*]
* [*Objetivo 2*]
* …

### Objetivos para el alumno

* [*Objetivo 1*]
* [*Objetivo 2*]
* …

## 1.4. Enfoque y método seguido.

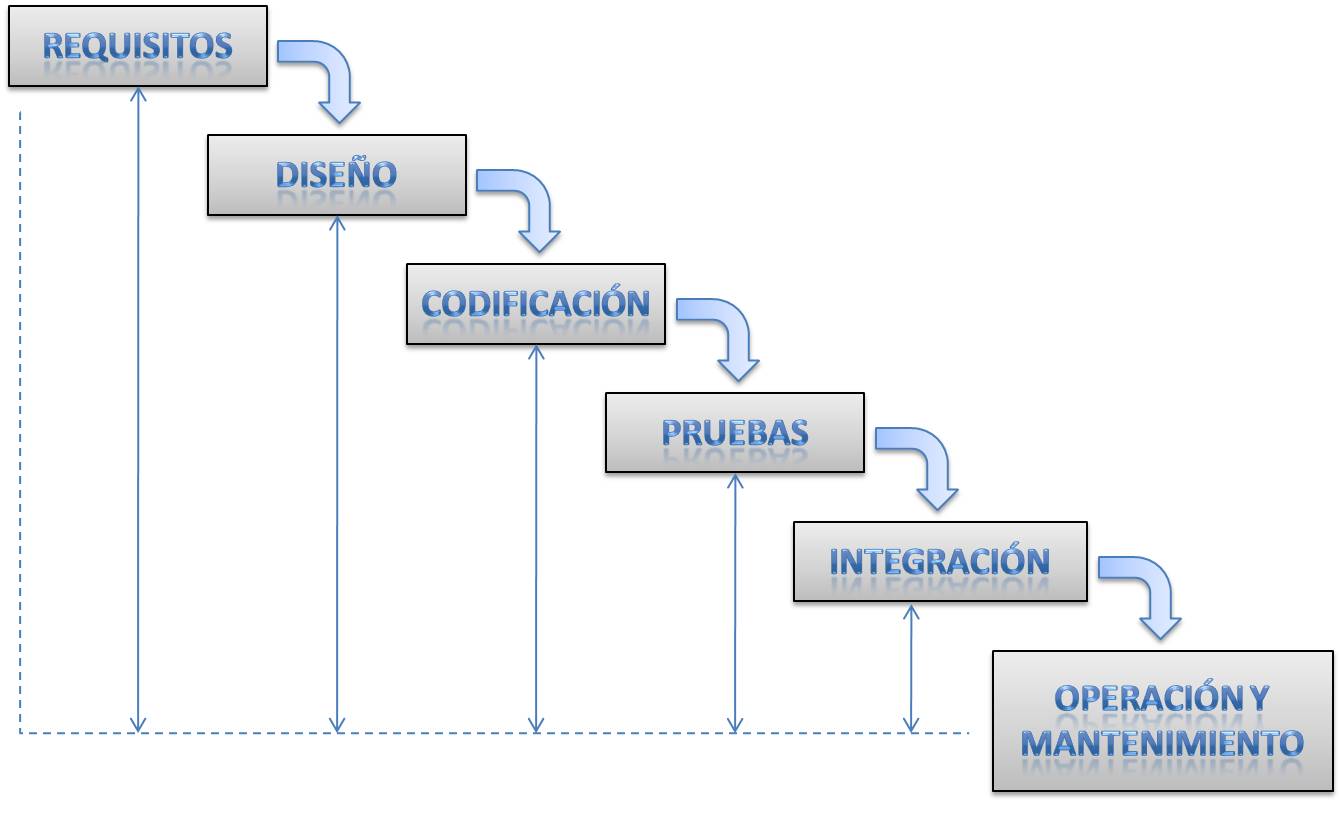
El objetivo principal del proyecto es crear una solución viable al problema planteado en el punto 1.1. Para llevar a cabo dicha tarea haremos uso de los conocimientos adquiridos en otras asignaturas a lo largo de nuestra vida de estudiante del ciclo formativo.

Para ello, el primer paso para afrontar el proyecto será realizar un análisis iniciar con diversas lecturas del enunciado, donde obtendremos los requisitos del proyecto, y con ello, los requerimientos que necesitará nuestra solución, que serán planificados de una forma lo más realizar y ajustada posible. El tutor se ha encargado de realizar un ajuste de dichos requerimientos, ya que es posible que no puedan finalizarse todos.

Para ello se ha optado por una metodología clásica, como es el modelo en cascada, siendo este un método que ordena rigurosamente las etapas del desarrollo de software, de tal forma que para iniciar una etapa de desarrollo tiene que haberse terminado la anterior, de tal manera que si existiera un error en la fase de pruebas llevaría el rediseño y programación nuevamente del programa afectado.

Las etapas del desarrollo en cascadas son

1. Análisis de requisitos
2. Diseño del sistema
3. Codificación
4. Pruebas
5. Integración
6. Mantenimiento



Cada una de estas partes serán desglosadas en este documento, justificando de esta forma el desarrollo del proyecto.

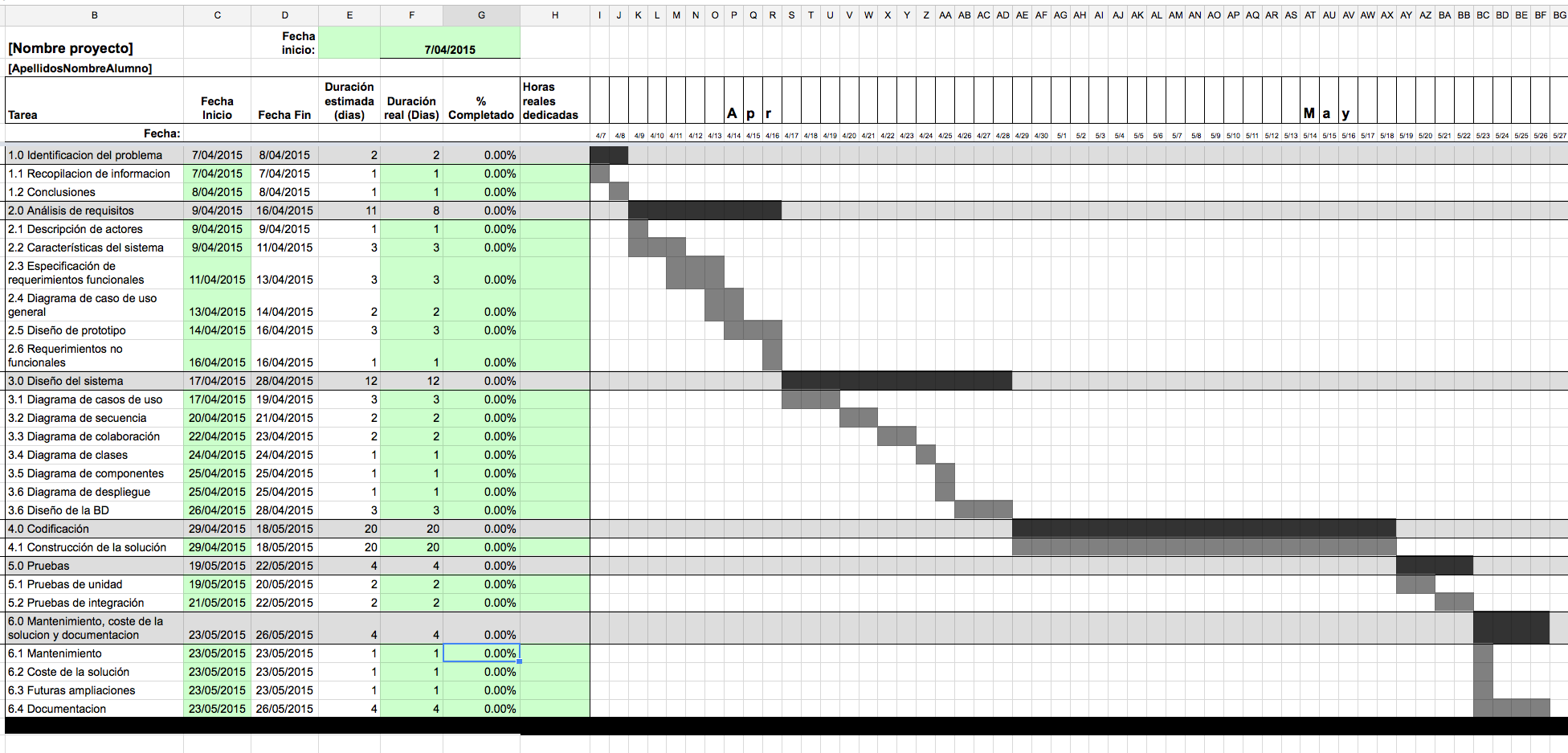
Vamos a pasar a comentar el trabajo a desarrollar en cada uno de los puntos:

1. Análisis de requisitos: Básicamente, durante esta fase, se adquieren, reúnen y especifican las características funcionales y no funcionales que deberá cumplir sistema a desarrollar. Esta tarea se va a subdividir en varios apartados:
   1. Descripción de actores: Son los usuarios que participarán en el sistema. No se utilizará ningún programa, sino una plantilla propia.
   2. Características del sistema: Se describirán, por módulos del sistema y características operativas internas, cada uno de los módulos existentes.
   3. Especificación de requerimientos funcionales: se describirán los distintos requisitos existentes en el sistema.
   4. Diagrama de caso de uso general: Se realizará un diagrama de caso de uso general de la aplicación a solucionar. Se utilizará el programa **ARGOUML** para ello, ya que está orientado al diseño con UML, que podemos descargar de la siguiente dirección: [http://argouml.tigris.org](http://www.google.com/url?q=http%3A%2F%2Fargouml.tigris.org&sa=D&sntz=1&usg=AFQjCNFgPkuJN6yRfB-fEWkpWtOswOOhiQ)
   5. Diseño de prototipo: Diseñaremos las distintas ventanas de nuestra aplicación con la aplicación **Pencil Project**, que podemos descargarla de la siguiente dirección: [http://pencil.evolus.vn/Default.html](http://www.google.com/url?q=http%3A%2F%2Fpencil.evolus.vn%2FDefault.html&sa=D&sntz=1&usg=AFQjCNFZ-CHggedxfUfRboMyna3QFLS5HA)
   6. Requerimientos no funcionales del sistema: Hará referencia a requisitos que no dependen del hardware/software. Encontraremos de 2 tipos: requerimientos de desempeño y requerimientos de seguridad.
2. Diseño del sistema: En este apartado se presenta un resumen de los procesos existentes en el problema, y se pondrán en relieve las necesidades del mismo. Con ello se esbozará las opciones desarrolladas en el análisis de requisitos y se presentarán las recomendaciones. Comenzaremos realizando un diseño preliminar de la solución continuando hacia un diseño más detallado de cada parte de la misma. Es necesario destacar que realizaremos, también, un diseño de la base de datos necesaria. Para ello el diseño de la solución será dividida en las siguientes partes:
   1. Diseño preliminar.
      1. Diagramas de casos de uso de las características del sistema (los módulos existentes): Se diseñan, dibujan y especifican cada uno de los casos de uso existentes en el sistema. Se utilizará el programa **ARGOUML**, ya comentado con anterioridad.
      2. Diagramas de secuencia: Se realizan los diagramas de secuencia de cada uno de los casos de uso existentes. Se utilizará el programa **ARGOUML**, ya comentado con anterioridad.
      3. Diagramas de colaboración: Se realizan los diagramas de colaboración entre las clases, actividades, ventanas bases de datos, etc.
      4. Diagrama de clases: Se identifican las clases existentes pertenecientes al problema y se realiza el correspondiente diagrama UML (ojo, una actividad que da de alta un alumno no es una clase perteneciente al problema, pero la clase Alumno, que representa a los objetos alumnos, sí que lo es).
   2. Diseño detallado.
      1. Diagrama de Componentes: Este diagrama permite establecer correspondencias entre las clases y los componentes de implementación. Estos componentes son las vistas físicas de la aplicación, o lo que es lo mismo, el diagrama de las distintas pantallas que tiene la aplicación, así como sus relaciones. Se utilizará el programa **ARGOUML**, ya comentado con anterioridad.
      2. Diagrama de despliegue: muestra la disposición física de los recursos de ejecución computacionales, tales como computadores (o sistemas móviles) y sus interconexiones. Se utilizará el programa **ARGOUML**, ya comentado con anterioridad.
   3. Diseño de la base de datos.
      1. Nivel lógico: Generaremos el diagrama entidad/relación. Se utilizará el programa DIA, ya que no dependerá de esta forma el diagrama entidad/relación de ninguna base de datos. Lo podemos descargar de la siguiente dirección: [http://dia-installer.de/index.html.en](http://www.google.com/url?q=http%3A%2F%2Fdia-installer.de%2Findex.html.en&sa=D&sntz=1&usg=AFQjCNG7xak4P2Y7Bw5Ftw4CKD1c3LIR2w)
      2. Nivel físico: Generaremos el diagrama de las estructuras necesarias existentes en el diagrama lógico. No se utilizará ningún programa, sino una plantilla propia.
      3. Implantación de la Base de Datos: Crearemos las estructuras de la BD. Se utilizará el programa propio del sistema gestor de base de datos que nos permita realizar las operaciones necesarias. Dicho programa lo seleccionará el alumno, y en este apartado indicará cual es y la dirección de descarga.
3. Codificación de la solución: Se pasará a codificar la solución software al problema planteado. En este punto es necesario definir los siguientes puntos:
   1. Planteamiento de la estrategia de programación: Debéis definir qué estrategia de programación habéis seguido: se implementa un módulo y se realiza un conjunto de pruebas sobre el mismo, o se implementa gran parte de la funcionalidad y luego se realizan pruebas de todo el conjunto, etc.
   2. Especificaciones técnicas para el desarrollo: En este apartado hablaremos de dos puntos esenciales:
      1. Requerimientos técnicos de Hardware: HW necesario para que el sistema funcione.
      2. Requerimientos técnicos de Software: SW necesario para que el sistema funcione.
   3. Descripción general de la solución creada: Se describirán toda la solución generada, tocando todos los módulos y realizando capturas de pantallas de los mismos y comentando lo que se hace en cada uno de ellos.
4. Pruebas de unidad: Se realiza, y documenta, un conjunto de pruebas unitarias sobre la solución que verifica su correcto funcionamiento. Es una forma de comprobar el correcto funcionamiento de un módulo de código. Esto sirve para asegurar que cada uno de los módulos funcione correctamente por separado.
5. Integración (o pruebas de integración): Sirven para garantizar que los diferentes módulos se integren con la aplicación. Éste es el propósito de la *prueba de integración* que está cuidadosamente documentada.
6. Mantenimiento: Se propondrá un plan de mantenimiento del software.

Aparte de todos los puntos desarrollados con anterioridad, tendremos un apartado de conclusiones, donde expondremos las conclusiones del trabajo realizado, así como un apartado de valoración económica del proyecto, donde definiremos aproximádamente cual es el coste de la solución propuesta y futuras ampliaciones, donde indicaremos donde puede crecer la solución y apartados que no hemos ampliado.

## 1.5. Planificación del proyecto.

Se ha realizado una planificación inicial del proyecto de forma que este comenzará el día 07/04/2015 y terminará el 26/05/2015. El planning inicial del mismo es el siguiente:



El planning final tras la finalización del proyecto es el siguiente:

[*Aquí irá el planning final cuando acabemos el proyecto*]

## 1.6. Recursos del proyecto.

Para realizar dicho proyecto el alumno ha usado los siguiente recursos Hardware y Software:

[*lista de recursos utilizados*]

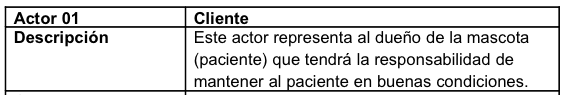
# 2. Análisis de requisitos.

## 2.1. Descripción de actores.

[*En este apartado tenéis que describir los actores que participan en el sistema utilizando esta plantilla*]

|  |  |
| --- | --- |
| **Actor** |  |
| **Descripción** |  |

**/\*\*\*\*\*\*\*\*\*\*\*\*\* EJEMPLO \*\*\*\*\*\*\*\*\*\*\*\***



**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

## 2.2. Características del sistema.

Se describirán, por módulos del sistema y características operativas internas, cada uno de los módulos existentes.

[*Debéis crear una lista de características del sistema utilizando esta plantilla*]

|  |  |
| --- | --- |
| **Módulo** |  |
| **Operatividad** |  |
| **Descripción** |  |
| **Secuencia** |  |

**/\*\*\*\*\*\*\*\*\*\*\*\*\* EJEMPLO \*\*\*\*\*\*\*\*\*\*\*\***

|  |  |
| --- | --- |
| **Módulo** | **Módulo de Acceso al Sistema** |
| **Operatividad** | **Acceso al sistema** |
| **Descripción** | Es una sección donde se realiza el control de ingreso al sistema determinado por los privilegios de uso establecidos por el sistema si es administrador o personal médico. |
| **Secuencia** | Para estar en condiciones de iniciar una sesión de trabajo en el sistema, se deberá contar con una cuenta de acceso valida. El usuario deberá proporcionar sus datos de usuario( login y password) de ahí derivara si ingresa al sistema o no. |

|  |  |
| --- | --- |
| **Módulo** | **Módulo de consulta médica** |
| **Operatividad** | **Gestionar Historia Clínica** |
| **Descripción** | El medico veterinario consultara,añadira o modficara la informacion de la historia clínica de la mascota. |
| **Secuencia** | Durante el proceso de atencion clinica sera necesario consultar los antecedentes de la mascota los cuales estaran almacenados en la historia clinica asi como tambien añadir el diagnostico como conclusiones obtenidas durante la consulta medica. |

|  |  |
| --- | --- |
| **Módulo** | **Módulo de consulta médica** |
| **Operatividad** | **Gestión de la documentación** |
| **Descripción** | Se realiza la emisión de un documento para el cliente y el paciente donde se detallará el tratamiento en la receta médica o se emitirá una constancia de vacunación. |
| **Secuencia** | Cuando se tiene un diagnóstico médico sobre el estado de salud de la mascota, el médico veterinario estará en la necesidad de brindar un recetario donde especifique el tratamiento a seguir o detalle algunas recomendaciones a seguir por el cliente para el bienestar de la mascota. Si la mascota ha sido vacunada el medico esta en la obligación de emitir una constancia de vacunación al cliente. |

|  |  |
| --- | --- |
| **Módulo** | **Módulo de consulta médica** |
| **Operatividad** | **Planificación de citas** |
| **Descripción** | Aquí se gestiona la posibilidad de una nueva cita para el cliente y su mascota. |
| **Secuencia** | Si el médico veterinario cree necesario una próxima cita para Se anexara a una agenda un recordatorio de la existencia de una próxima cita con el cliente. El médico veterinario tendrá la opción de ver las citas del dia de manera que estará listo para realizar la atención. |

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

## 2.3. Especificación de requerimientos funcionales.

Pasamos a indicar los requerimientos funcionales del sistema:

[*Indicamos la lista de requerimientos funcionales utilizando la siguiente plantilla para ello*]

* **R.1 - Requerimiento 1**
  + R.1.1 - Subrequerimiento 1.1
  + R.1.2 - Subrequerimiento 1.2
    - R.1.2.1 - Subrequerimiento 1.2.1
    - R.1.2.2 - Subrequerimiento 1.2.2
* **R.2 - Requerimiento 2**
  + R.2.1 - Subrequerimiento 2.1
  + R.2.2 - Subrequerimiento 2.2
* ….

**/\*\*\*\*\*\*\*\*\*\*\*\*\* EJEMPLO \*\*\*\*\*\*\*\*\*\*\*\***

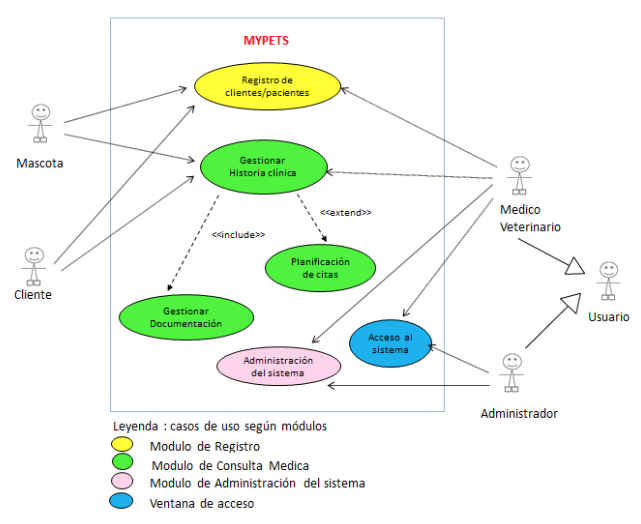
* **R.1 - Requerimientos generales del Acceso al Sistema:**
  + R.1.1 - Especificar el acceso al sistema en niveles mediante un control de acceso a usuario de médico y administrador.
  + R.1.2 - Contar con una cuenta de usuario valida gestionada por el nivel de administrador para poder iniciar sesión.
* **R.2 - Requerimientos generales de la Gestión de Historias clínicas.**
  + R.2.1 - Ejecutable cuando exista una historia clínica asignada a una mascota
  + R.2.2 - Consultar las historias clínicas
  + R.2.3 - Actualizar las historias clínicas con nuevos datos.
  + R.2.4 - Las historias clínicas deben además poder consultarse y listarse por fechas.
* **R.3 - Requerimientos generales de la Gestion de Documentacion** 
  + R.3.1 - Requisitos para la emisión del recetario
    - R.3.1.1 - Cada recetario tendrá un codigo y fecha de emisión además debe consignar los datos de la mascota y cliente; donde se describirá el tratamiento para determinado e diagnóstico.
    - R.3.1.2 - El nombre del emisor del recetario médico es decir el médico veterinario.
    - R.3.1.3 - El recetario solo será emitido cuando exista un tratamiento a aplicar.
  + R.3.2 - Requisitos para la emisión de la constancia de vacunación.
    - R.3.2.1 - Haberse realizado la vacunación.
    - R.3.2.2 - Para la emisión de una constancia de vacunación esta tendrá un codigo y fecha de emisión, además de consignar los datos de la mascota y cliente.
    - R.3.3 - Añadir los resultados de los documentos emitidos a la historia clínica de la mascota.
* **R.4 - Requerimientos generales de la planificación de citas.**
  + R.4.1 - Una agenda para controlar las próximas citas.
  + R.4.2 - Listado de las citas del día para el médico veterinario usuario del sistema
  + R.4.3 - Funciones básicas de añadir y eliminar citas.

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

## 2.4. Diagrama de caso de uso general.

*[Debéis dibujar el diagrama de caso de uso general con* ***ARGOUML*** *y pegarlo aquí*]

**/\*\*\*\*\*\*\*\*\*\*\*\*\* EJEMPLO \*\*\*\*\*\*\*\*\*\*\*\***



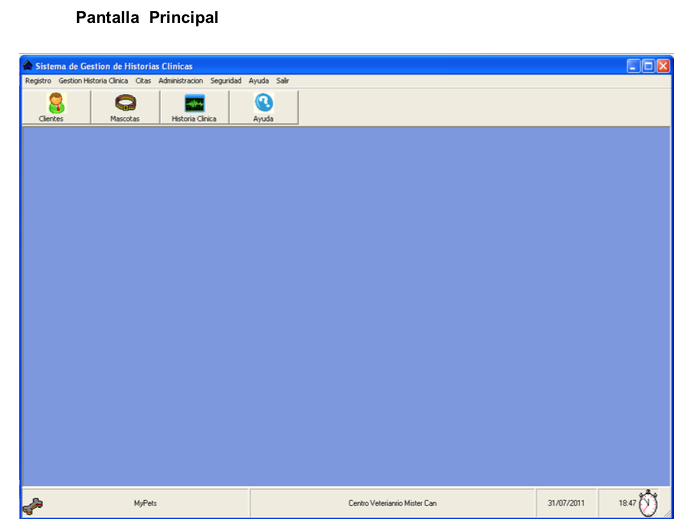
**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

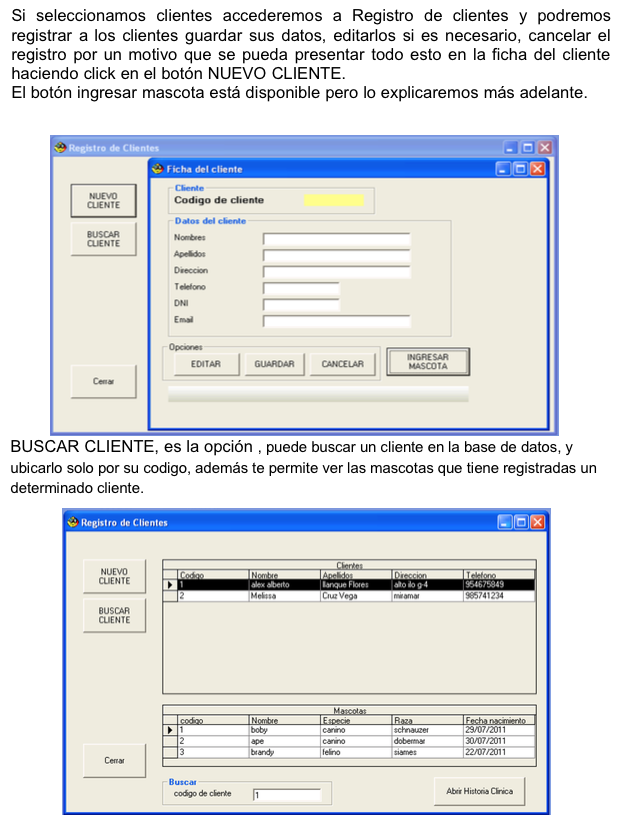
## 2.5. Diseño de prototipo.

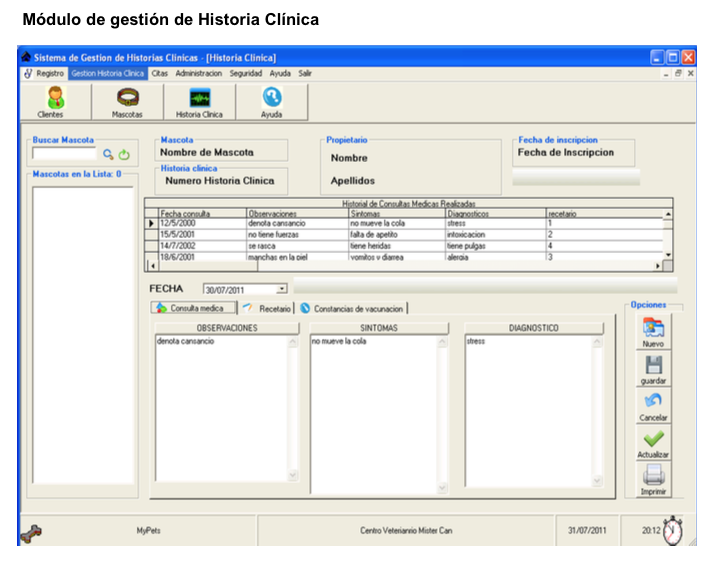
[*Debéis describir cada pantalla que tenéis que va a permitir (realizar una descripción muy generalista) y dibujar el prototipo con* ***Pencil Project*** *a continuación*]

**/\*\*\*\*\*\*\*\*\*\*\*\*\* EJEMPLO \*\*\*\*\*\*\*\*\*\*\*\***









**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* FIN EJEMPLO \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

## 2.6. Requerimientos no funcionales del sistema.

[*Debéis poner una lista de requerimientos no funcionales del sistema. Seguid el ejemplo como plantilla*]

**/\*\*\*\*\*\*\*\*\*\*\*\*\* EJEMPLO \*\*\*\*\*\*\*\*\*\*\*\***

* Requerimientos de desempeño
  + Interfaz del Usuario:
    - Se utilizan imágenes, íconos y/o nombres descriptivos para cada elemento de la interfaz para eliminar así confusiones en el uso de la herramienta.
    - Según sea el perfil de usuario, se debe habilitar / deshabilitar las opciones correspondientes.
    - Las interfaces no contienen más de 20 campos por pantalla para evitar la confusión del usuario
    - En caso de presentase lo anterior se agrupa la funcionalidad a través de tabuladores intuitivos y fáciles de usar.
  + Rendimiento y soporte
    - El sistema procesa la información en un tiempo razonable para el levantamiento de un contenido o el envío del mismo. El tiempo de respuesta a los usuarios debe ser menor a 5 segundos.
    - El sistema debe contar con un instalador, para su distribución en diferentes puntos.
  + Usabilidad y Disponibilidad
    - El sistema debe ser fácil de usar y cuenta con ayudas contextuales, mensajes de estado, mensajes de error y cualquier ayuda necesaria para facilidad del usuario
    - El sistema debe estar disponible 99% del tiempo en cualquier momento que el usuario lo requiera.
* Requerimientos de seguridad
  + Se debe restringir la información a ser mostrada a los usuarios según su rol. Se debe restringir las operaciones según el tipo de rol que posea el usuario que ingrese al sistema.
  + Se debe contar con. una copia de seguridad de la base de datos (backup) para mantener la integridad de la información.

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* FIN EJEMPLO \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

# 3. Diseño del sistema.

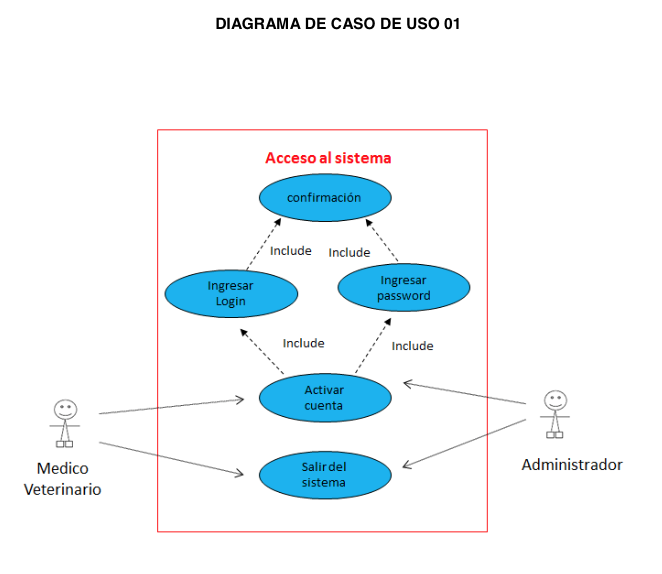
## 3.1. Diseño preliminar.

### 3.1.1. Diagramas de Casos de Uso del sistema.

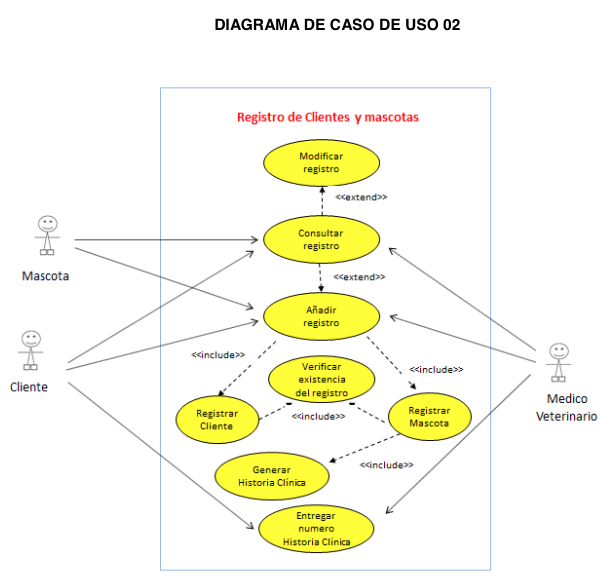
[*Diseñaremos los distintos diagramas de casos de uso y rellenaremos las plantillas de documentación del mismo*]

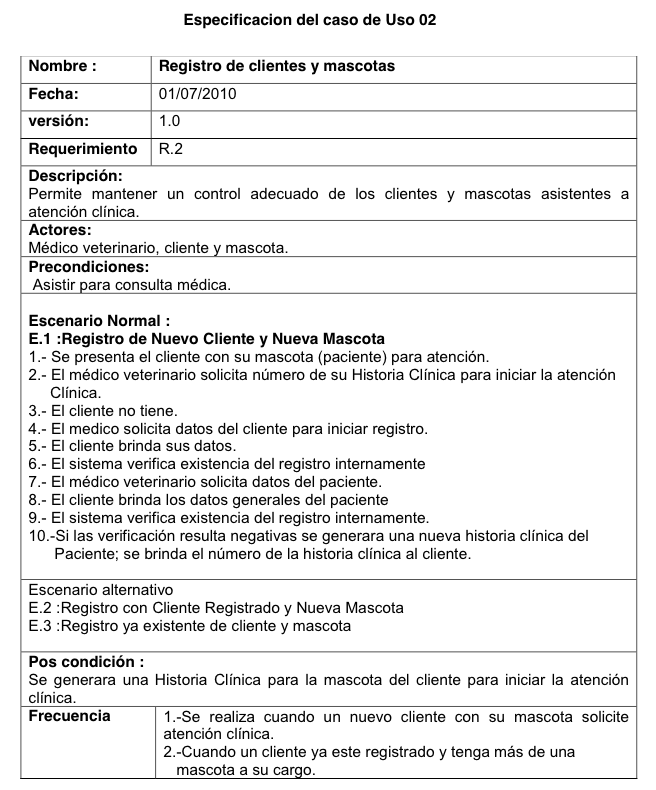
|  |  |
| --- | --- |
| **Nombre** |  |
| **Fecha** |  |
| **Versión** |  |
| **Requerimiento** |  |
| **Descripción** |  |
| **Actores** |  |
| **Precondiciones** |  |
| **Escenario Normal** |  |
| **Escenario Alternativo** |  |
| **Pos Condición** |  |
| **Frecuencia de uso** |  |

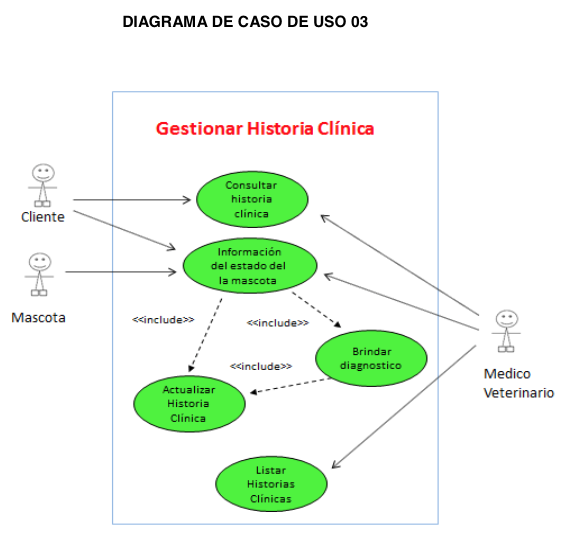
**/\*\*\*\*\*\*\*\*\*\*\*\*\* EJEMPLO \*\*\*\*\*\*\*\*\*\*\*\***

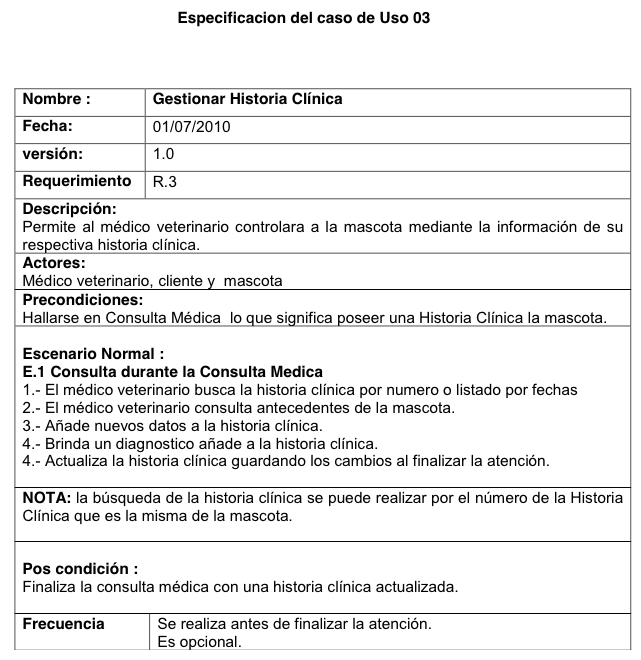










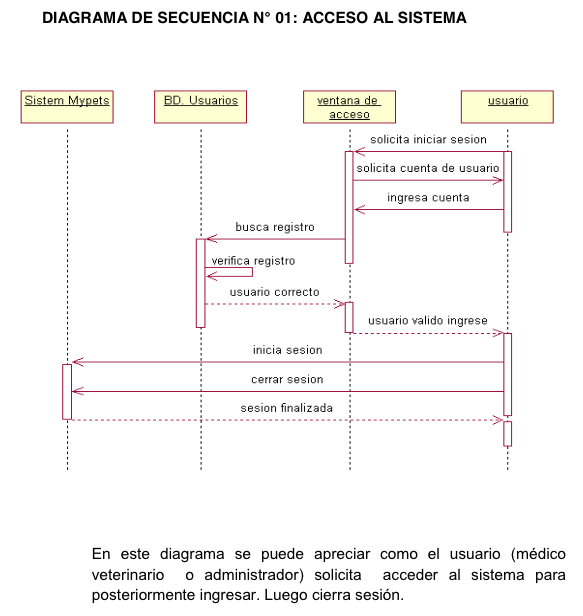


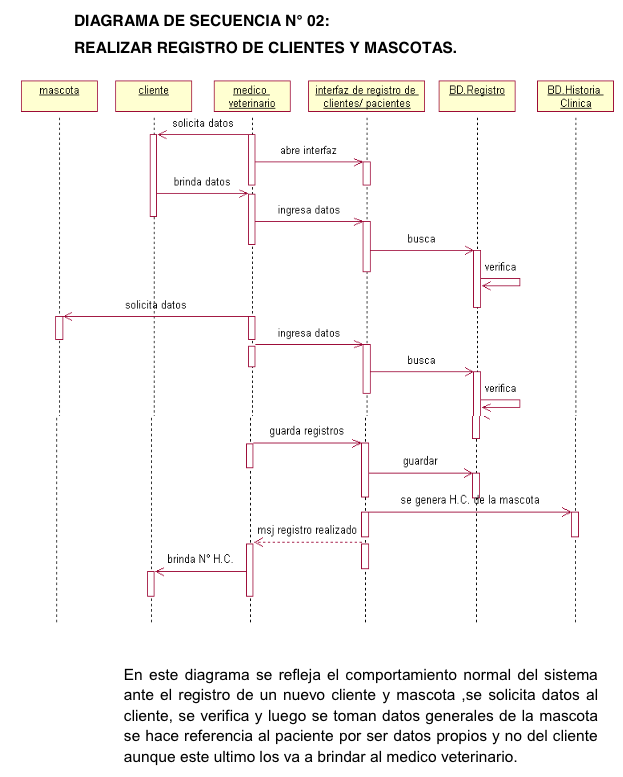
**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* FIN EJEMPLO \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

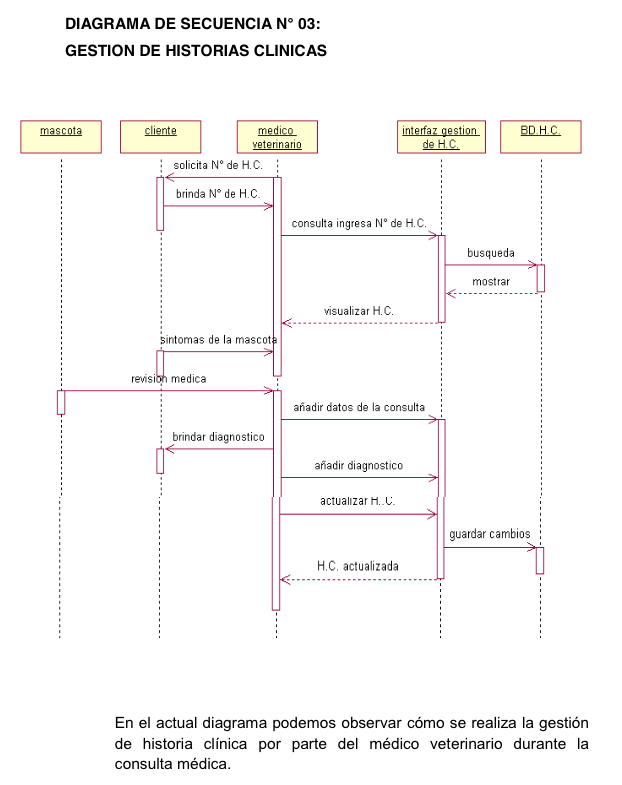
### 3.1.2. Diagramas de Secuencia.

[*De cada caso de uso existente debéis diseñar los diagramas de secuencia y comentarlos. Dichos diagramas de secuencia nos permitirán entender como van a funcionar las operaciones en nuestro programa. Utilizad ARGOUML para diseñar los diagramas de secuencia.*]

**/\*\*\*\*\*\*\*\*\*\*\*\*\* INICIO EJEMPLO \*\*\*\*\*\*\*\*\*\*\*\***





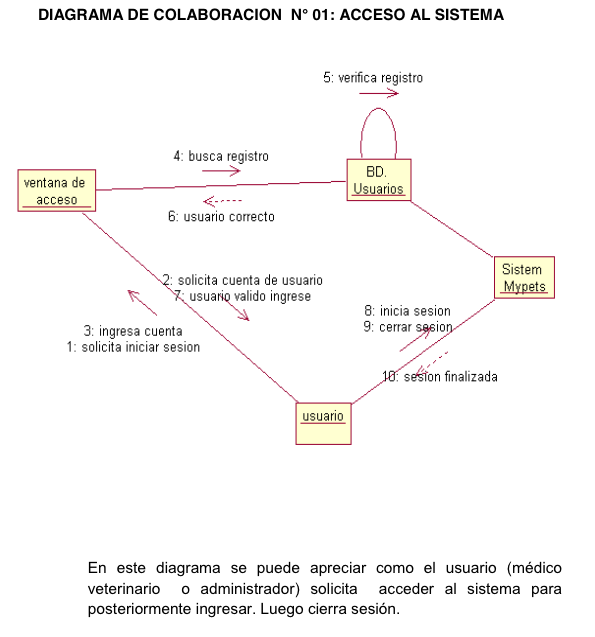


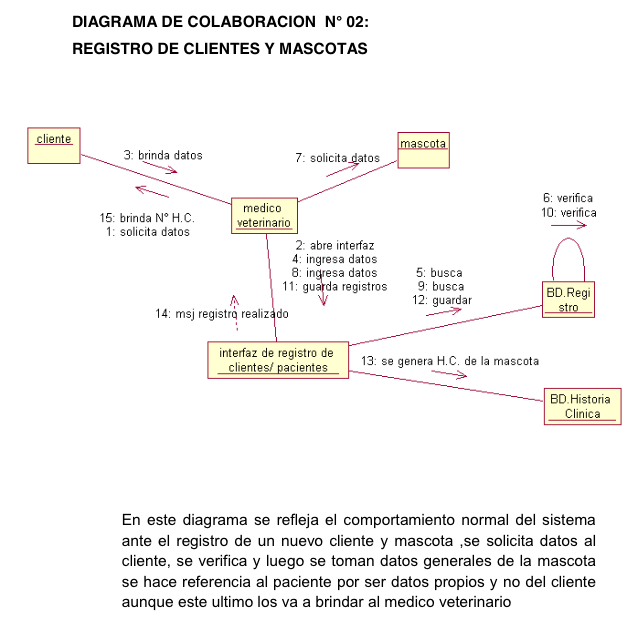
**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* FIN EJEMPLO \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

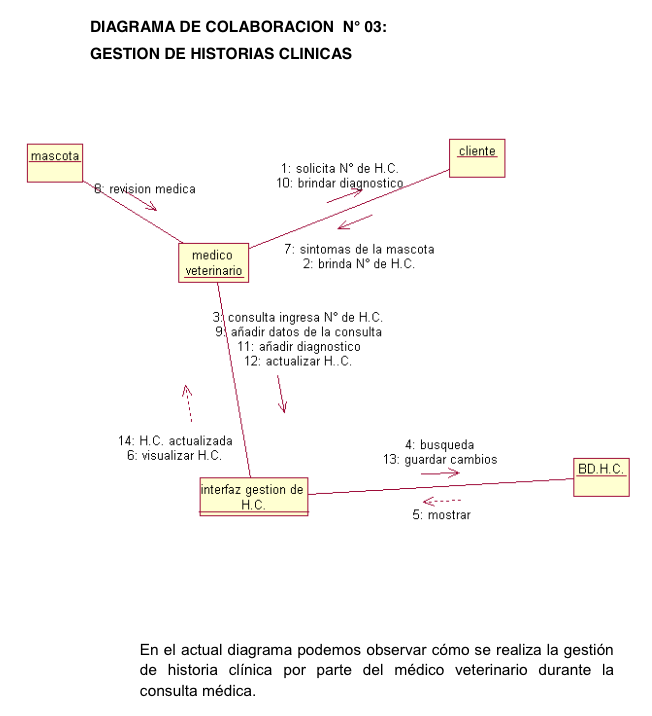
### 3.1.3. Diagramas de Colaboración

[*A partir de caso de uso existente y de su diagrama de secuencia debéis diseñar los diagramas de colaboración y comentarlos. Dichos diagramas de colaboración nos permitirán entender como se van comunicando los formularios, clases, bases de datos, etc. Utilizar ARGOUML para diseñar los diagramas de colaboración. Son muy similares a los diagramas de secuencia, pero más legibles.*]

**/\*\*\*\*\*\*\*\*\*\*\*\*\* INICIO EJEMPLO \*\*\*\*\*\*\*\*\*\*\*\***







**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* FIN EJEMPLO \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

### 3.1.4. Diagrama de Clases

[*Para obtener el diagrama de clases es necesario estudiar a fondo la descripción del problema y realizar un análisis profundo del mismo, con el fin de obtener las clases, atributos y relaciones. Por ello debéis escribir el enunciado del problema de forma correcta, que nos servirá, tanto para ponerlo de nuevo en este apartado. Es el momento de modificar el punto 1.1 y escribir un enunciado coherente con lo que deseamos obtener. Tras ello pasar a obtener las clases, sus atributos y las relaciones entre ellos.*]

#### 3.1.4.1. Identificación de Clases

[*Simplemente pondremos una lista con el nombre de las clases con las que trabajaremos.*]

**/\*\*\*\*\*\*\*\*\*\*\*\*\* INICIO EJEMPLO \*\*\*\*\*\*\*\*\*\*\*\***

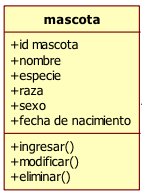
* Usuario
* Médico veterinario
* Administrador
* Cita
* Cliente
* Mascota
* Consulta medica
* Historia clínica
* Formato
* Recetario
* Vacunas

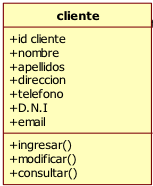
**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* FIN EJEMPLO \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

#### 3.1.3.2. Atributos y Métodos de las Clases.

[*Se identificarán los atributos y los métodos de las clases y se realizarán las correspondientes fichas de las mismas a partir del programa ARGOUML*]

**/\*\*\*\*\*\*\*\*\*\*\*\*\* INICIO EJEMPLO \*\*\*\*\*\*\*\*\*\*\*\***



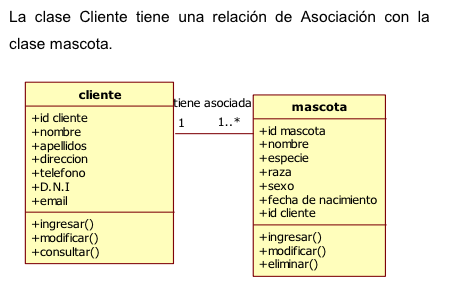


**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* FIN EJEMPLO \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

#### 3.1.3.3. Relaciones entre Clases.

[*Pasamos a describir las relaciones/asociaciones que tendrán las distintas clases entre ellas, de forma que al realizar este apartado estamos haciendo el diagrama UML por trozos. Diseñamos dichas relaciones en el ARGOUML y las pegamos aquí, junto con una pequeña explicación de a qué hace referencia.*]

**/\*\*\*\*\*\*\*\*\*\*\*\*\* INICIO EJEMPLO \*\*\*\*\*\*\*\*\*\*\*\***

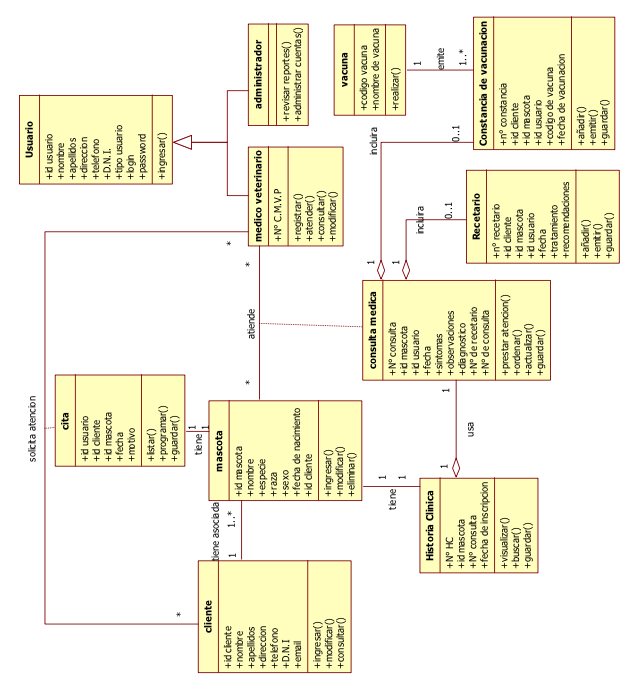


**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* FIN EJEMPLO \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

#### 3.1.3.4. Diagrama de Clases completo.

**[***En este apartado pegamos todo el diagrama de clases que hayamos obtenido con el ARGOUML.*]

**/\*\*\*\*\*\*\*\*\*\*\*\*\* INICIO EJEMPLO \*\*\*\*\*\*\*\*\*\*\*\***



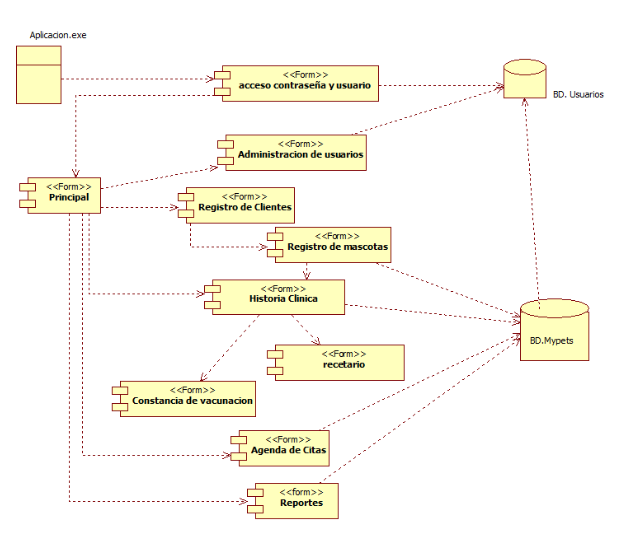
**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* FIN EJEMPLO \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

## 3.2. Diseño Detallado.

### 3.2.1. Diagrama de Componentes.

[*En este apartado diseñaremos el diagrama de componentes, o lo que es lo mismo los distintos formularios o ventanas (como queráis llamarlos) existente en el sistema, así como las relaciones que existen entre ellos. Para ello hacer uso del programa ARGOUML. Nos dará un mapa de todo nuestro programa y su navegación.*]

**/\*\*\*\*\*\*\*\*\*\*\*\*\* INICIO EJEMPLO \*\*\*\*\*\*\*\*\*\*\*\***

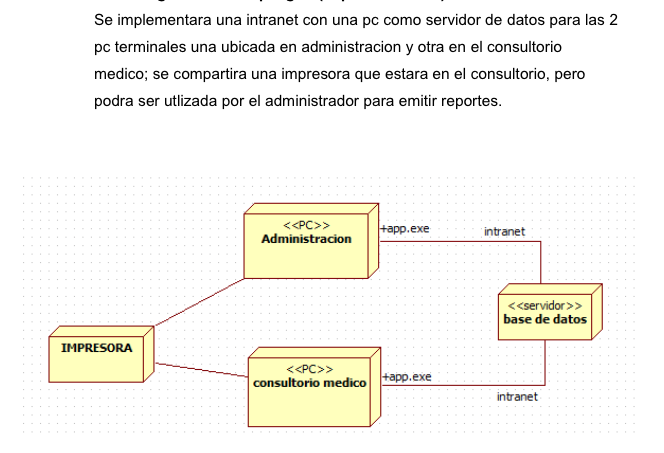


**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* FIN EJEMPLO \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

### 3.2.2. Diagrama de Despliegue.

[*En este apartado describiremos como va a ser desplegado nuestra aplicación. En el ejemplo que vais a ver a continuación lo tendréis muy claro.*]

**/\*\*\*\*\*\*\*\*\*\*\*\*\* INICIO EJEMPLO \*\*\*\*\*\*\*\*\*\*\*\***



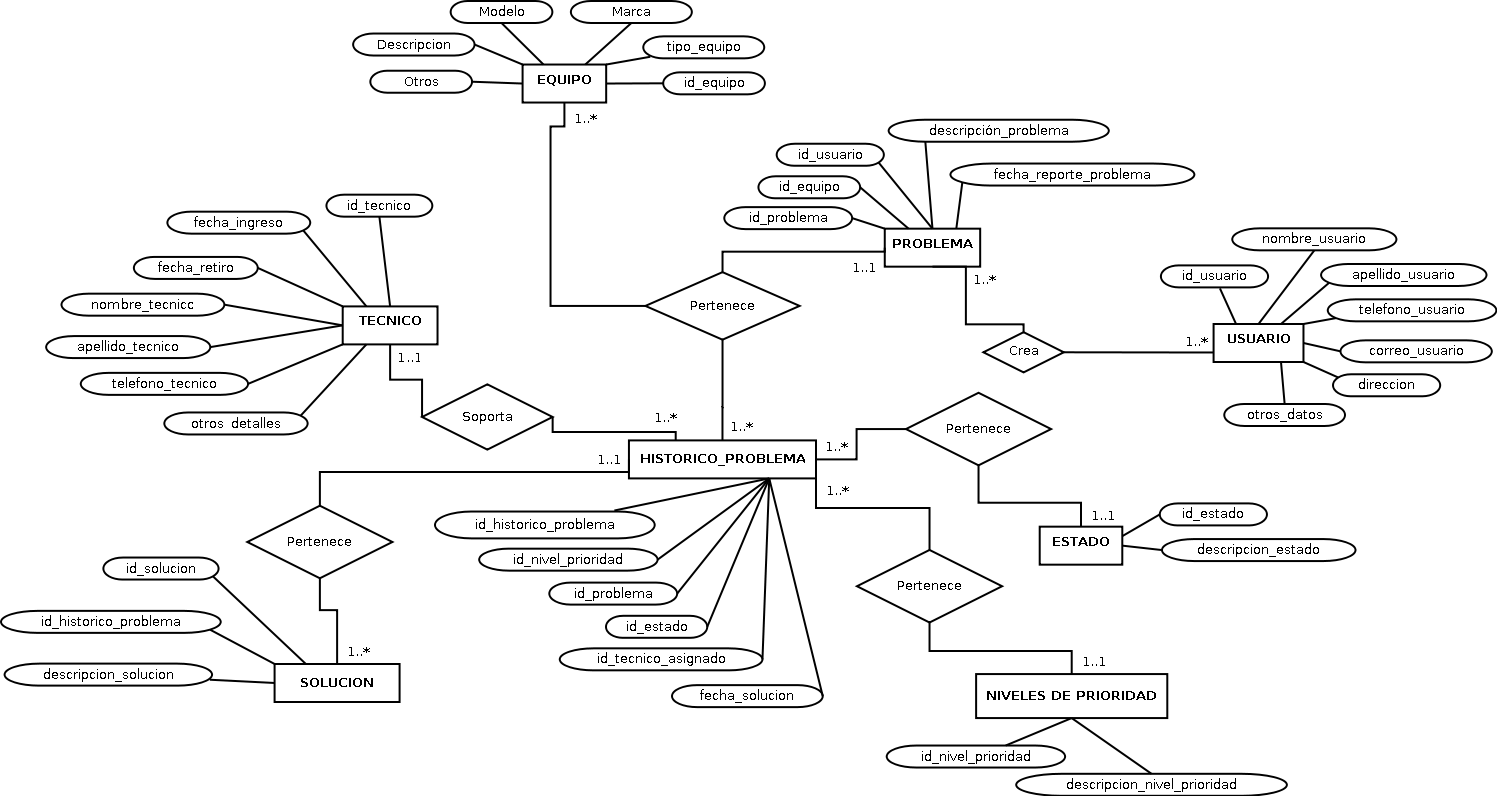
**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* FIN EJEMPLO \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

## 3.3. Diseño de la Base de Datos.

### 3.3.1. Nivel lógico.

[*A partir del diagrama UML podemos obtener el diagrama entidad/relación, por lo que usaremos el programa DIA para poder generarlo. Tener en cuenta que es posible que algún proyecto no haga uso de la base de datos, por lo que no será necesario este apartado no los siguientes pertenecientes al 3.3, simplemente indicaremos que no hace uso de bases de datos.*]

**/\*\*\*\*\*\*\*\*\*\*\*\*\* INICIO EJEMPLO \*\*\*\*\*\*\*\*\*\*\*\***



**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* FIN EJEMPLO \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

### 3.3.2. Nivel Físico.

[*De cada tabla existente en el nivel lógico realizaremos una implementación física usando la siguiente plantilla.*]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tabla** | | **[*Nombre de Tabla*]** | | |
| **Atributo** | **Descripción**  **Tipo de Dato** | | **Tipo de Dato** | **¿Es clave o índice?** |
| [*Atributo1*] | [*Descripción del atributo 1*] | | [*Tipo de dato con longitud, si tiene*] | [*Vacio, Clave primaria ó índice*] |
| [*Atributo2*] | [*Descripción del atributo 2*] | | [*Tipo de dato con longitud, si tiene*] | [*Vacio, Clave primaria ó índice*] |
| …. | …. | | ….. | …. |

**/\*\*\*\*\*\*\*\*\*\*\*\*\* INICIO EJEMPLO \*\*\*\*\*\*\*\*\*\*\*\***

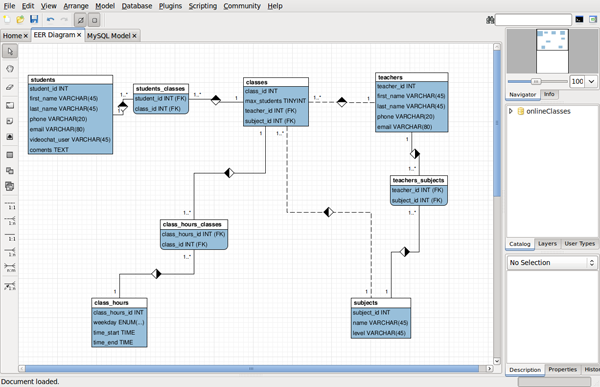
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tabla** | | **Equipo** | | |
| **Atributo** | **Descripción**  **Tipo de Dato** | | **Tipo de Dato** | **¿Es clave o índice?** |
| id\_equipo | Identificador del equipo | | INT | Clave Primaria |
| descripcion | Descripción del equipo | | VARCHAR(100) | Índice |
| marca | Marca del equipo | | VARCHAR(100) |  |
| modelo | Modelo del equipo | | VARCHAR(100) |  |
| tipo\_equipo | Tipo de equipo al que pertenece el mismo | | INT |  |
| otros | Observaciones | | VARCHAR(255) |  |
| fecha\_alta | Fecha de alta en el sistema | | DATE |  |

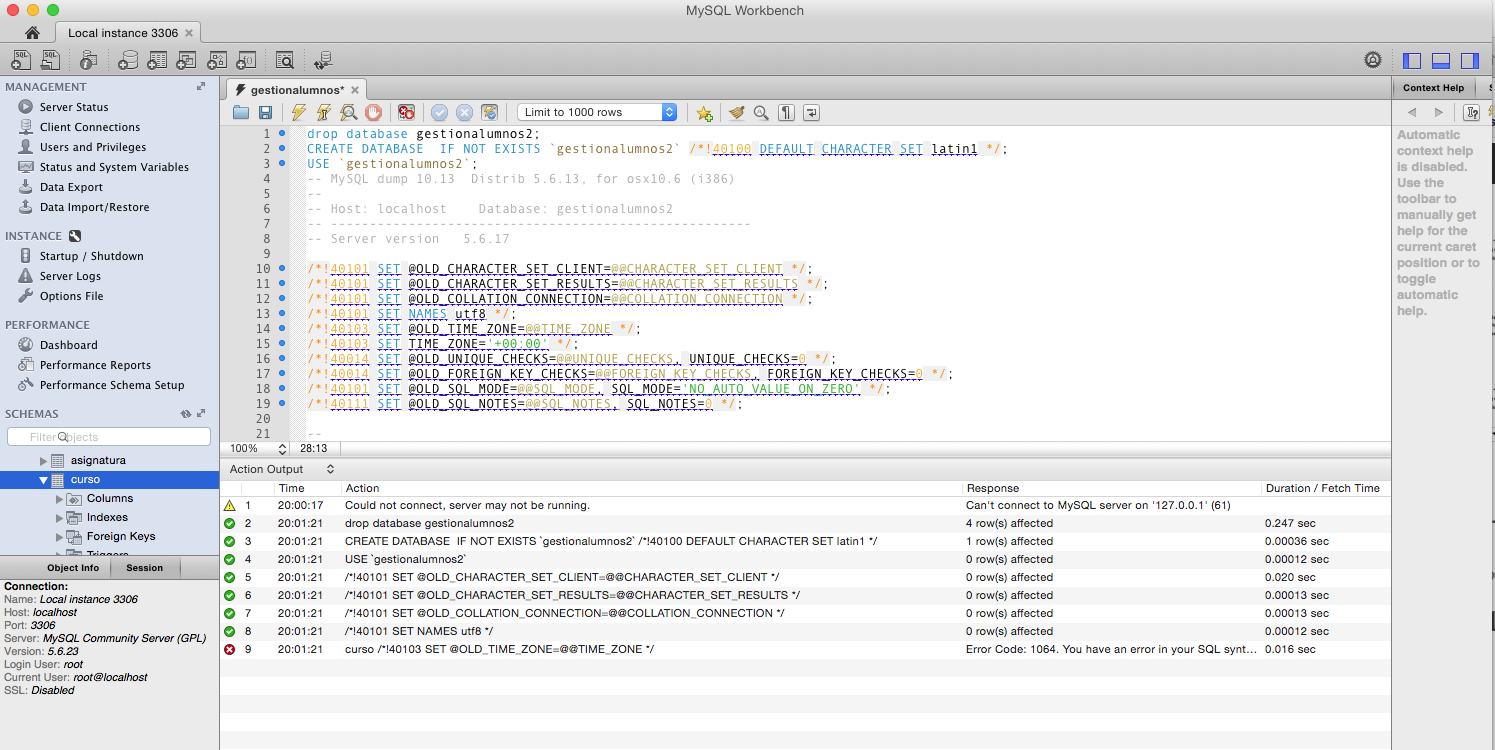
**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* FIN EJEMPLO \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

### 3.3. Implantación de la Base de Datos.

[*En este apartado se hará uso de un motor de bases de datos, así como de una aplicación gestora de la misma. Simplemente tenéis que crear el esquema de la base de datos que vais a crear y realizar las capturas pertinentes. El esquema de la base de datos debéis copiarlo y pegarlo íntegramente en el documento Anexo A que encontraremos en la carpeta de trabajo]*

**/\*\*\*\*\*\*\*\*\*\*\*\*\* INICIO EJEMPLO \*\*\*\*\*\*\*\*\*\*\*\***





**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* FIN EJEMPLO \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

# 4. Codificación de la solución.

## 4.1. Planteamiento de la estrategia de programación.

[*Debéis describir cuál ha sido vuestra estrategia de programación. Pongo algunos ejemplo:*

* *Comenzáis a programar siguiendo toda la documentación generada, y en caso de encontrar un error en alguna de las fases anteriores paráis de programar, arreglais el error en la documentación y demás pasos y continuais programando.*
* *Comenzáis a programar hasta obtener una versión 1.0. Una vez obtenida la versión, y habiendo comprendido los errores de diseño que se ha visualizado paso a modificar los errores de diseño y demás y genero una nueva versión 1.1.*
* *Comenzáis a programar los módulos poco a poco, generando las pruebas unitarias al mismo tiempo y documentándolas, de forma que ya están hechas. Si encuentro un error en las fases previas rectifico el error en ellas, sigo programando de nuevo y reinicio todas las pruebas unitarias afectadas.*
* *Etc….*

*Se generará un fichero comprimido con el proyecto de forma que el formato de entrega será [ApellidosNombreAlumno\_proyecto\_1415.zip, de forma que en el mismo estará solo y exclusivamente el proyecto y el fichero .sql para crear la base de datos correspondiente. Si el proyecto esta compuesto por dos subproyectos (uno cliente y uno servidor) dentro del .zip habrán dos carpetas indicando que es cada cosa.*]

**/\*\*\*\*\*\*\*\*\*\*\*\*\* INICIO EJEMPLO \*\*\*\*\*\*\*\*\*\*\*\***

Para la gestion y desarrollo del presente proyecto de software se decidio aplicar un metodo de desarrollo agil como la programacion extrema que se sitúa en una posición diferente a los ciclos de vida tradicionales. Ya no se trata de requisitos que se cierran en etapas tempranas del desarrollo y que constituyen el contrato de lo que posteriormente se va a desarrollar (y que los usuarios no ven hasta mucho después) sino de requisitos que están vivos y que son modificables a lo largo del ciclo de vida.

Se busca obtener el software en poco tiempo ademas se desarrollara de manera paulatina, la complejidad del código va creciendo de manera proporcional o incluso exponencial en función del tamaño del proyecto, El control de la calidad del código se complementa con otras estrategias como por ejemplo el intento de que el mismo sea lo más autocomentado posible.

Las pruebas unitarias resultan esenciales, revisión del funcionamiento de cada módulo Hay que tener en cuenta también que el código está en continua evolución El objetivo es detectar posibles errores lo antes posible. La detección tardía de errores es más costosa y su corrección además tiende a ensuciar el código, desarrollo iterativo incremental permite que en cada evolución el usuario pueda expresar su opinión sobre los resultados obtenidos y si es necesario corregir algo, planificarlo para la siguiente o próximas iteraciones.

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* FIN EJEMPLO \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

## 4.2. Especificaciones técnicas para el desarrollo.

### 4.2.1. Requerimientos técnicos de Hardware.

[*En este apartado pondréis una lista de requerimientos hardware que se necesitan para el desarrollo del proyecto como ejemplo ponemos un ordenador potente, un dispositivo móvil, un servidor, una impresora, etc.*]

### 4.2.2. Requerimientos técnicos de Software.

[*En este apartado pondréis una lista de requerimientos software que se necesitan para el desarrollo del proyecto como ejemplo un sistema operativo, un entorno de desarrollo, un lenguaje, etc.*]

## 4.3. Descripción general de la solución creada.

[*En este apartado describiremos la solución que hemos creado, de forma que realizaremos capturas de pantalla de las mismas y las explicaremos. No confundir con las capturas del prototipo, ya que son distintas; el prototipo no es funcional, y simplemente muestra como van a ser las pantallas y aquí vamos a pegar las pantallas de nuestro programa ya hecho.*]

# 5. Pruebas de unidad.

[*En este apartado definiremos un conjunto de pruebas unitarias para los módulos que hemos creado. Las pruebas unitarias son un conjunto de pruebas sobre las operaciones de los módulos. Por ejemplo, podemos probar, del módulo usuario, la operación alta, de forma que recibirá todos los datos correctamente, y se espera obtener como resultado que el usuario sea dado de alta en el sistema de forma correcta. Otra prueba sería la misma, pero en este caso podríamos dejar en blanco el campo NIF del usuario, de forma que lo que esperamos del módulo es que se produzca un error y que avise de que no es posible dar de alta al usuario porque se ha dejado el NIF en blanco.*]

|  |  |  |
| --- | --- | --- |
| Id del caso de prueba unitaria | | CDPU*XXX* |
| Objetivo de la prueba unitaria | | [*Describiremos el objetivo de la prueba a realizar*] |
| Módulo | | [*Nombre del módulo al que le realizamos la prueba*] |
| Operación | | [*Nombre de la operación que se va a invocar perteneciente al módulo*] |
| Fecha | | [*Fecha de la prueba*] |
| Versión de la prueba | | [*XX*] |
| Error esperado | | [*Si ha de generar algún error se indica aquí el error esperado.*] |
| Descripción | | [*Descripción de la prueba*] |
| Entrada | | [*Si recibe algún valor de entrada la prueba se pone aquí*] |
| Salida a obtener | | [*Se pone aquí la salida que nosotros debemos obtener*] |
| Requisitos especiales | | [*Indica si la prueba tiene algún requisito especial*] |
| Dependencias con otros casos de prueba | | [*Si la prueba depende antes de otras pruebas se pone aquí el código de las pruebas de las que depende*] |
| Resultado | | [*Se informa del resultado de la prueba, si se ha obtenido lo esperado o por el contrario es un error y se ha obtenido algo no esperado*] |
| Salida | [*Se pone la salida de la consola, siempre que sea necesario*] | |

**/\*\*\*\*\*\*\*\*\*\*\*\*\* INICIO EJEMPLO \*\*\*\*\*\*\*\*\*\*\*\***

|  |  |  |
| --- | --- | --- |
| Id del caso de prueba unitaria | | CDPU001 |
| Objetivo de la prueba unitaria | | Verificar que se puede dar de alta un usuario con todos sus datos. |
| Módulo | | Usuario |
| Operación | | Alta |
| Fecha | | 10/04/2015 |
| Versión de la prueba | | 01 |
| Error esperado | | Ninguno |
| Descripción | | Invocaremos a la operación de alta de usuario, pasando todos los datos correctos, de forma que el sistema los validará de forma correcta, y a continuación creará el nuevo usuario y mostrará un mensaje indicando que se ha dado el usuario de alta correctamente. |
| Entrada | | nombre=”Loren”, apellidos=”Diaz”, NIF=”12345678A”, direccion=”Calle Don Pepito”, poblacion=”Valencia”, provincia=”Valencia”, cp=”46000”, edad=39, fechaNacimiento=”10/03/1976” |
| Salida a obtener | | Un mensaje indicando que el usuario se ha dado de alta. |
| Requisitos especiales | | Ninguno |
| Dependencias con otros casos de prueba | | Ninguna |
| Resultado | | OK, el usuario se ha dado de alta correctamente. |
| Salida | Ninguna. | |

|  |  |  |
| --- | --- | --- |
| Id del caso de prueba unitaria | | CDPU002 |
| Objetivo de la prueba unitaria | | Verificar que no se puede dar de alta un usuario faltando el NIF. |
| Módulo | | Usuario |
| Operación | | Alta |
| Fecha | | 10/04/2015 |
| Versión de la prueba | | 01 |
| Error esperado | | ExceptionUsuario |
| Descripción | | Invocaremos a la operación de alta de usuario, pasando todos los datos correctos excepto el NIF, de forma que el sistema no podrá validar el usuario y generará una excepción propia creada por nosotros indicando que nos falta el NIF del usuario. |
| Entrada | | nombre=”Loren”, apellidos=”Diaz”, NIF=””, direccion=”Calle Don Pepito”, poblacion=”Valencia”, provincia=”Valencia”, cp=”46000”, edad=39, fechaNacimiento=”10/03/1976” |
| Salida a obtener | | Un mensaje de error indicando que nos falta el NIF del usuario. |
| Requisitos especiales | | Ninguno |
| Dependencias con otros casos de prueba | | CDPU001 |
| Resultado | | OK, el sistema ha mostrado el error deseado y no se ha dado de alta el usuario. |
| Salida | Ninguna. | |

|  |  |  |
| --- | --- | --- |
| Id del caso de prueba unitaria | | CDPU003 |
| Objetivo de la prueba unitaria | | Verificar que no se puede dar de alta un usuario si ponemos caracteres alfanuméricos en el campo de la fecha de nacimiento. |
| Módulo | | Usuario |
| Operación | | Alta |
| Fecha | | 10/04/2015 |
| Versión de la prueba | | 01 |
| Error esperado | | ExceptionUsuario |
| Descripción | | Invocaremos a la operación de alta de usuario, pasando todos los datos correctos excepto la fecha de alta, que pondremos “aa/aa/aaaa”, y al ser caracteres alfanuméricos no se podrá dar de alta, y el sistema mostrará un mensaje de error controlado indicando que la fecha no está bien indicada. |
| Entrada | | nombre=”Loren”, apellidos=”Diaz”, NIF=”12345678A”, direccion=”Calle Don Pepito”, poblacion=”Valencia”, provincia=”Valencia”, cp=”46000”, edad=39, fechaNacimiento=”aa/aa/aaaa” |
| Salida a obtener | | Un mensaje de error indicando que la fecha de nacimiento no es correcta. |
| Requisitos especiales | | Ninguno |
| Dependencias con otros casos de prueba | | CDPU001 |
| Resultado | | ERROR, ya que el sistema ha mostrado un mensaje de error que no ha sido tratado por nosotros, lo que indica que este error no está siendo controlado de forma correcta. Se ha de arreglar el error y repetir la prueba y documentarla de nuevo. |
| Salida | java.lang.NumberFormatException: 12  at java.lang.Integer.parseInt(Compiled Code)  at java.lang.Integer.parseInt(Integer.java:390)  at excepcion.ExcepcionApp.main(ExcepcionApp.java:8) | |

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* FIN EJEMPLO \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

# 6. Pruebas de Integración.

[*Mientras que las pruebas unitarias solamente tocan una operación de un módulo, las pruebas de integración es una operación que toca a varios módulos, con lo que llama a diversas operaciones de distintos módulos. Utilizaremos la siguiente plantilla para ello.]*

|  |  |  |
| --- | --- | --- |
| Id del caso de prueba de integracion | | CDPI*XXX* |
| Objetivo de la prueba integradora | | [*Describiremos el objetivo de la prueba a realizar*] |
| Módulos | | [*Lista de nombres de los módulos que participan*] |
| Operaciones | | [*Lista de nombre de operaciones que participan. El formato de la lista será:*  *nombreModulo.operacion*  ] |
| Fecha | | [*Fecha de la prueba*] |
| Versión de la prueba | | [*XX*] |
| Error esperado | | [*Si ha de generar algún error se indica aquí el error esperado.*] |
| Descripción | | [*Descripción de la prueba*] |
| Entrada | | [*Si recibe algún valor de entrada la prueba se pone aquí*] |
| Salida a obtener | | [*Se pone aquí la salida que nosotros debemos obtener*] |
| Requisitos especiales | | [*Indica si la prueba tiene algún requisito especial*] |
| Dependencias con otros casos de prueba | | [*Si la prueba depende antes de otras pruebas se pone aquí el código de las pruebas de las que depende*] |
| Resultado | | [*Se informa del resultado de la prueba, si se ha obtenido lo esperado o por el contrario es un error y se ha obtenido algo no esperado*] |
| Salida | [*Se pone la salida de la consola, siempre que sea necesario*] | |

**/\*\*\*\*\*\*\*\*\*\*\*\*\* INICIO EJEMPLO \*\*\*\*\*\*\*\*\*\*\*\***

|  |  |  |
| --- | --- | --- |
| Id del caso de prueba de integracion | | CDPI001 |
| Objetivo de la prueba integradora | | Alta de usuario en el sistema y en la base de datos. |
| Módulos | | UsuarioAltaActivity, Usuario, UsuarioBLL, UsuarioDAO |
| Operaciones | | UsuarioAltaActivity.alta  UsuarioBLL.alta  UsuarioBLL.verificarUsuario  UsuarioDAO.alta  contructor de Usuario |
| Fecha | | 10/04/2015 |
| Versión de la prueba | | 01 |
| Error esperado | | No se espera ningún error. |
| Descripción | | Se mostrará la pantalla asociada a UsuarioAltaActivity, de forma que pondremos todos los datos en los cuadros de texto. Tras ello se pulsará un botón de OK y se creará un nuevo usuario con esos datos invocando al contructor de Usuario, y se invocará al método UsuarioBLL.alta, que verificará si el usuario es correcto o no (invocando este a UsuarioBLL.verificarUsuario). Si el usuario es correcto se invocará a UsuarioDAO.alta pasandole el objeto usuario ya verificado, y permitiendo dar de alta el usuario en la base de datos. A continuación se informará de que el usuario ha sido dado de alta. |
| Entrada | | nombre=”Loren”, apellidos=”Diaz”, NIF=”12345678A”, direccion=”Calle Don Pepito”, poblacion=”Valencia”, provincia=”Valencia”, cp=”46000”, edad=39, fechaNacimiento=”10/04/2015” |
| Salida a obtener | | Un mensaje indicando que el usuario se ha dado de alta, y un registro más en la base de datos. |
| Requisitos especiales | | Ninguno. |
| Dependencias con otros casos de prueba | | Ninguna. |
| Resultado | | OK, el usuario se ha dado de alta correctamente. |
| Salida | Ninguna. | |

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* FIN EJEMPLO \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

# 7. Mantenimiento.

[*En este apartado tenéis que poner las consideraciones oportunas que creáis para que el sistema reciba un mantenimiento adecuado y funcione de forma correcta.*]

**/\*\*\*\*\*\*\*\*\*\*\*\*\* INICIO EJEMPLO \*\*\*\*\*\*\*\*\*\*\*\***

El sistema cuenta con una función de respaldo de la base de datos, un backup para realizar una copia de seguridad de la base de datos del sistema, esta opción puede ser activada por el administrador antes de cerrar sesión.

La evaluación continua de la infraestructura de hardware instalada para evitar posteriores fallas.

Se necesitara digitalizar toda la información de las historia clínicas físicas a al sistema para mantener un control completo.

De parte de los usuarios se les recomienda leer el manual de usuario brindado para conocer las bondades del software y no cometer un uso inadecuado del mismo.

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* FIN EJEMPLO \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

# 8. Futuras ampliaciones.

[*En este apartado tenéis que poner las ampliaciones que recomendais para el proyecto, así como los apartados que no habéis realizado, que se quedarían como ampliaciones.*]

# 9. Valoración económica.

[*En este apartado sólo tenéis que poner las horas que habéis dedicado a cada cosa, que se encuentran en la planificación, y realizar el cálculo total.*]

Para calcular la valoración económica se hará uso de los recursos humanos utilizados y sus costes y de la duración real de tareas.

Los elementos necesarios para la realización de un proyecto de esta envergadura serían los siguientes:

* Analista: Se encarga de la realización del análisis y diseño, junto con la documentación necesaria del proyecto y el diseño las pruebas.
* Administrador de bases de datos: Es la persona responsable de los aspectos ambientales de una base de datos. En general esto incluye lo siguiente:
  + Recuperabilidad: Crear y probar Respaldos
  + Integridad: Verificar o ayudar a la verificación en la integridad de datos
  + Seguridad: Definir o implementar controles de acceso a los datos
  + Disponibilidad: Asegurarse del mayor tiempo de encendido
  + Desempeño: Asegurarse del máximo desempeño incluso con las limitaciones
  + Desarrollo y soporte a pruebas: Ayudar a los programadores e ingenieros a utilizar eficientemente la base de datos.
* Desarrollador: se encarga de escribir y depurar el código fuente necesario para obtener la solución.
* Jefe de proyecto: Se encarga de la conducción de un proyecto informático desde su concepción original hasta el lanzamiento al público. El jefe de proyecto tiene muchas tareas:
  + Definir el proyecto y evaluar sus necesidades.
  + Redactar las especificaciones del proyecto.
  + Calcular el costo del proyecto.
  + Contratar al equipo de producción.
  + Realizar un seguimiento e informes del progreso del proyecto, en términos de calidad, costo y plazos de entrega.

Debido a que solamente se dispone de un solo recurso que realizará los 4 roles existentes anteriormente citados se especifica que el coste por hora de trabajo de dicho recurso (asimilando los 4 roles) es de 50€.

Con el coste por recurso definido se realiza un resumen de las tareas realizadas, con la duración real en horas y el coste total de cada parte:

|  |  |  |
| --- | --- | --- |
| **Tareas** | **Horas** | **Coste** |
| **1.0 Identificacion del problema** | **[*XX*]** | **[*XX*]** |
| **1.1 Recopilacion de informacion** | **[*XX*]** | **[*XX*]** |
| **1.2 Conclusiones** | **[*XX*]** | **[*XX*]** |
| **2.0 Análisis de requisitos** | **[*XX*]** | **[*XX*]** |
| **2.1 Descripción de actores** | **[*XX*]** | **[*XX*]** |
| **2.2 Características del sistema** | **[*XX*]** | **[*XX*]** |
| **2.3 Especificación de requerimientos funcionales** | **[*XX*]** | **[*XX*]** |
| **2.4 Diagrama de caso de uso general** | **[*XX*]** | **[*XX*]** |
| **2.5 Diseño de prototipo** | **[*XX*]** | **[*XX*]** |
| **2.6 Requerimientos no funcionales** | **[*XX*]** | **[*XX*]** |
| **3.0 Diseño del sistema** | **[*XX*]** | **[*XX*]** |
| **3.1 Diagrama de casos de uso** | **[*XX*]** | **[*XX*]** |
| **3.2 Diagrama de secuencia** | **[*XX*]** | **[*XX*]** |
| **3.3 Diagrama de colaboración** | **[*XX*]** | **[*XX*]** |
| **3.4 Diagrama de clases** | **[*XX*]** | **[*XX*]** |
| **3.5 Diagrama de componentes** | **[*XX*]** | **[*XX*]** |
| **3.6 Diagrama de despliegue** | **[*XX*]** | **[*XX*]** |
| **3.6 Diseño de la BD** | **[*XX*]** | **[*XX*]** |
| **4.0 Codificación** | **[*XX*]** | **[*XX*]** |
| **4.1 Construcción de la solución** | **[*XX*]** | **[*XX*]** |
| **5.0 Pruebas** | **[*XX*]** | **[*XX*]** |
| **5.1 Pruebas de unidad** | **[*XX*]** | **[*XX*]** |
| **5.2 Pruebas de integración** | **[*XX*]** | **[*XX*]** |
| **6.0 Mantenimiento, coste de la solución, futuras ampliaciones y documentación** | **[*XX*]** | **[*XX*]** |
| **6.1 Mantenimiento** | **[*XX*]** | **[*XX*]** |
| **6.2 Coste de la solución** | **[*XX*]** | **[*XX*]** |
| **6.3 Futuras ampliaciones** | **[*XX*]** | **[*XX*]** |
| **6.4 Conclusiones** | **[*XX*]** | **[*XX*]** |
| **6.4 Revisión de la documentación** | **[*XX*]** | **[*XX*]** |
| **Total** | **[*XX*]** | **[*XX*] €** |

# 10. Conclusiones.

[*En este apartado indicaremos nuestras conclusiones sobre el proyecto realizado.*]