Marco Teórico

# Introducción

En el presente capitulo se desarrollarán de manera resumida las tecnologías, herramientas y metodologías más relevantes que se utilizarán para el desarrollo del proyecto, a fin de introducir en el tema y lograr una mayor comprensión al desarrollo integral de todo el proyecto.

Se iniciará con la descripción de tabletas siendo la principal herramienta para el funcionamiento del sistema, se representarán las ventajas y desventajas de dicha tecnología, luego se describirá el Sistema Operativo Android sobre el cual funcionará la aplicación. También se definirá la metodología Ideal para el estudio de viabilidad del sistema. También se definirá Protégé que se utilizará para el desarrollo de las ontologías y la metodología de construcción de prototipos para el ciclo de vida de desarrollo del sistema y por último la herramienta GitHub para poder realizar un control de versiones.

# Tablet

# Definición:

Es una computadora portátil de mayor tamaño que un teléfono inteligente integrada en una pantalla táctil con la que se interactúa primariamente con los dedos, sin necesidad de teclado físico ni ratón.

# Utilidades:

* Lectura de libros electrónicos.
* Lectura sin conexión de páginas web.
* Consulta y edición de documentos de suites ofimáticas.
* Navegación web (mediante Wi-Fi, USB o 3G interno).
* Llamadas telefónicas, si son 3G, sustituyendo así al teléfono móvil; se suele utilizar un manos libre bluetooth.
* GPS.
* Reproducción de música.
* Visualización de vídeos y películas, cargadas desde la memoria interna, memoria o disco duro USB.
* Cámara fotográfica y de vídeo HD
* Videoconferencia.

## Ventajas

* Su facilidad de uso en entornos donde resulta complicado un teclado y un ratón, como en la cama, de pie, o el manejo con una sola mano.
* Su peso ligero. Los modelos de menor potencia pueden funcionar de manera similar a los dispositivos de lectura.
* El entorno táctil hace que en ciertos contextos —como en la manipulación de imágenes, música o juegos— el trabajo sea más fácil que con el uso de un teclado y un ratón.
* Facilita la realización de dibujos digitales y edición de imágenes pues resulta más preciso e intuitivo que pintar o dibujar con el ratón.
* Facilita y agiliza la posibilidad de agregar signos matemáticos, diagramas y símbolos.
* Permite (con el software adecuado) la interacción con diferentes teclados sin importar su ubicación.
* Para algunos usuarios resulta más interactivo y agradable usar un lápiz o el dedo para apuntar y pulsar sobre la pantalla, en lugar de utilizar un ratón o un touchpad.
* La duración de la batería es mucho mayor que la de una computadora portátil.

## Desventajas

* Velocidad de interacción: la escritura a mano sobre la pantalla, o escribir en un teclado virtual, puede ser significativamente más lento que la velocidad de escritura en un teclado convencional, que puede llegar hasta las 50 a 150 palabras por minuto. Algunos dispositivos también soportan teclados externos.
* Comodidad (ergonomía): una tableta no ofrece espacio para el descanso de la muñeca, el usuario tendrá que mover su brazo constantemente mientras escribe.
* Menor capacidad de vídeo: la mayoría de las tabletas están equipadas con procesadores gráficos incorporados en lugar de tarjetas de vídeo.
* Riesgos en la pantalla: las pantallas de las tabletas se manipulan más que las de los portátiles convencionales, sin embargo, muchas están fabricadas de manera similar. Además, puesto que las pantallas también sirven como dispositivos de interacción, corren un mayor riesgo de daños debido a los golpes y al mal uso.
* Riesgo en la bisagra: la bisagra de un portátil convertible en tableta usualmente necesita girar sobre dos ejes, a diferencia de la pantalla de un portátil normal, lo cual aumenta las posibilidades de fallos mecánicos o eléctricos (cables de transmisión y de vídeo, antenas WiFi integradas, etc).

# Android:

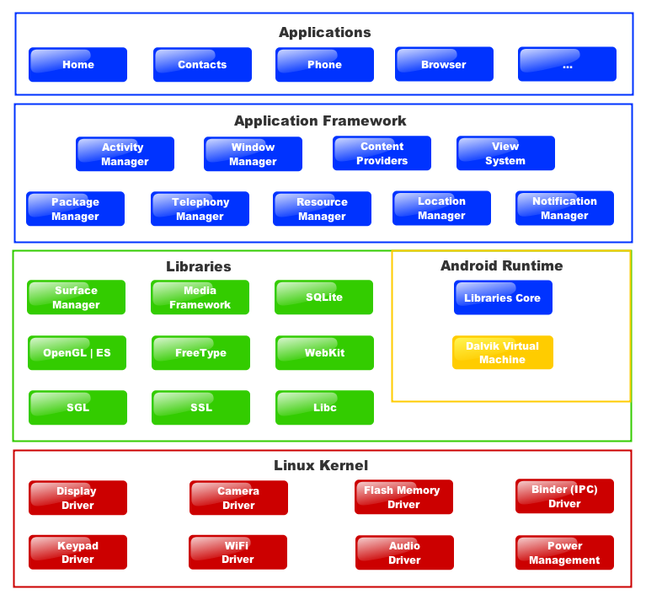
Es un [sistema operativo](http://es.wikipedia.org/wiki/Sistema_operativo) basado en el [núcleo Linux](http://es.wikipedia.org/wiki/N%C3%BAcleo_Linux), posee un núcleo de sistema operativo libre, gratuito y multiplataforma.

El sistema operativo proporciona todas las interfaces necesarias para desarrollar aplicaciones que accedan a las funciones del teléfono (como el GPS, las llamadas, la agenda, etc.) de una forma muy sencilla en un lenguaje de programación muy conocido como es Java.

# Arquitectura:

Los componentes principales del sistema operativo de Android son:

* **Aplicaciones:** las aplicaciones base incluyen un cliente de [correo electrónico](http://es.wikipedia.org/wiki/Correo_electr%C3%B3nico), programa de [SMS](http://es.wikipedia.org/wiki/Servicio_de_mensajes_cortos), calendario, mapas, navegador, contactos y otros. Todas las aplicaciones están escritas en lenguaje de programación Java.
* **Marco de trabajo de aplicaciones:** los desarrolladores tienen acceso completo a los mismos APIs del framework usados por las aplicaciones base. La arquitectura está diseñada para simplificar la reutilización de componentes; cualquier aplicación puede publicar sus capacidades y cualquier otra aplicación puede luego hacer uso de esas capacidades (sujeto a reglas de seguridad del framework). Este mismo mecanismo permite que los componentes sean reemplazados por el usuario.
* **Bibliotecas:** Android incluye un conjunto de bibliotecas de [C](http://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n_C)/[C++](http://es.wikipedia.org/wiki/C%2B%2B) usadas por varios componentes del sistema. Estas características se exponen a los desarrolladores a través del marco de trabajo de aplicaciones de Android; algunas son: System C library (implementación biblioteca C estándar), bibliotecas de medios, bibliotecas de gráficos, 3D y SQLite, entre otras.
* **Runtime de Android:** Android incluye un set de bibliotecas base que proporcionan la mayor parte de las funciones disponibles en las bibliotecas base del lenguaje Java. Cada aplicación Android corre su propio proceso, con su propia instancia de la máquina virtual Dalvik. Dalvik ha sido escrito de forma que un dispositivo puede correr múltiples máquinas virtuales de forma eficiente. Dalvik ejecuta archivos en el formato Dalvik Executable (.dex), el cual está optimizado para memoria mínima. La Máquina Virtual está basada en registros y corre clases compiladas por el compilador de Java que han sido transformadas al formato.dex por la herramienta incluida "dx".
* **Núcleo Linux:** Android depende de [Linux](http://es.wikipedia.org/wiki/Linux_(n%C3%BAcleo)) para los servicios base del sistema como seguridad, gestión de memoria, gestión de procesos, pila de red y modelo de controladores. El [núcleo](http://es.wikipedia.org/wiki/N%C3%BAcleo_(inform%C3%A1tica)) también actúa como una capa de abstracción entre el hardware y el resto de la pila de software.



# SISTEMA BASADO EN CONOCIMIENTO:

Se puede definir como un sistema que resuelve problemas utilizando una representación simbólica del conocimiento humano. Sistemas basados en conocimientos son un software que imita el comportamiento de un experto humano en la solución de un problema. Pueden almacenar conocimientos de expertos para un campo determinado y solucionar un problema mediante deducción lógica de conclusiones.

## Base de Conocimiento

Es la parte del sistema que contiene los conocimientos acumulados sobre el dominio, ademas contienen hechos que describen el area del problema y tecnicas de representación de conocimientos que describen como los hechos se embonan de forma logica. El conocimiento del experto se codifica en la base de conocimientos. Una forma clásica de representar el conocimiento en un sistema experto son las reglas. Una regla es una [estructura](http://www.monografias.com/trabajos15/todorov/todorov.shtml#INTRO) condicional que relaciona lógicamente la información contenida en la parte del antecedente con otra información contenida en la parte del consecuente.

## Motor de Inferencia

El motor de inferencia selecciona, decide, interpreta y aplica el conocimiento de la Base de Conocimientos sobre los datos que le introduce el usuario con el fin de obtener la solución más adecuada. El motor de inferencia controla todo el proceso de un sistema basado en conocimiento. Realiza dos funciones básicas: la inferencia y el control. La primera se relaciona con el examen de las reglas y la ejecución de comparaciones de patrones, mientras que la segunda gestiona el orden en el que se examinan estas reglas.

El motor de inferencia utiliza los datos proporcionados por el usuario para recorrer la base de conocimientos hasta alcanzar una conclusión. El motor de inferencia, ante una situación específica, detecta los conocimientos de la Base de Conocimientos del Sistema que interesan en la resolución, los utiliza, los encadena y construye un método de resolución o razonamiento independiente del dominio del caso actual.

## Justificación:

La construcción del sistema basado en conocimiento se justificara mediante el estudio de viabilidad realizado con la metodología Ideal.

# IDEAL

# CONSTRUCCION DE PROTOTIPOS:

## Justificación

Se optara por utilizar la metodología construcción de prototipo ya que permite un diseño rápido que se centra en una representación de aquellos aspectos del software que serán visibles para el cliente o el usuario final. Este diseño conduce a la construcción de un prototipo, el cual es evaluado por el cliente para una retroalimentación; gracias a ésta se refinan los requisitos del software que se desarrollará. Con la construcción de prototipos se permite que el desarrollador entienda mejor lo que se debe hacer y el cliente vea resultados a corto plazo. Gracias a esta metodología se puede observar las reacciones del cliente y sugerencias del mismo.

## Metodologia

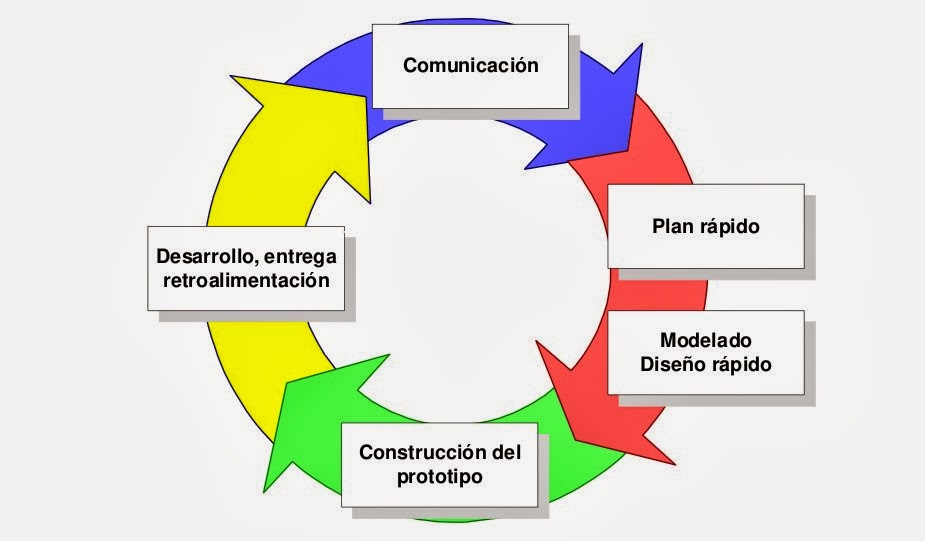
**Kenneth C. Laudon** [2004:395] define un prototipo como ”una versión operativa preliminar de un sistema de información para fines de demostración y evaluación”. Como su definición lo indica el prototipo es una versión preliminar, el objetivo es mejorarla con el paso del tiempo hasta que se acomode a las necesidades de los usuarios, por medio de un proceso iterativo que consiste en probar y afinar constantemente el prototipo hasta que cumpla su objetivo.

El modelo de prototipos permite que todo el sistema, o algunos de sus partes, se construyan rápidamente para comprender con facilidad y aclarar ciertos aspectos en los que se aseguren que el desarrollador, el usuario,y el cliente estén de acuerdo en lo que se necesita así como también la solución que se propone para dicha necesidad y de esta forma minimizar el riesgo y la incertidumbre en el desarrollo, este modelo se encarga del desarrollo de diseños para que estos sean analizados y prescindir de ellos a medida que se adhieran nuevas especificaciones.

Siguiendo la clasificación de **Kendall y Kendall** [Analisis y diseño de sistemas, 2005: 156] se establecio realizar un PROTOTIPO DE CARACTERÍSTICAS SELECTAS que incluye la creación de un modelo operacional que sólo implementa algunas características del sistema final. La retroalimentación de los usuarios ayuda a comprender lo que funciona y lo que no. También puede ayudar con las sugerencias sobre cuáles pueden ser las siguientes características a agregar. Mediante este tipo de creación de prototipos los usuarios evaluaron positivamente las características presentadas, se pueden incorporar al sistema final sin tener que trabajar mucho para interconectar los módulos. Y Los prototipos que se realizan de esta manera forman parte del sistema actual.

## Etapas

* Plan rápido
* Modelado, diseño rápido
* Construcción del Prototipo
* Desarrollo, entrega y retroalimentación
* Comunicación



## Pasos

Los pasos necesarios para la construcción de prototipos son los siguientes:

**PASO 1.**

Evaluar la petición del software y determinar si el programa a desarrollar es un buen candidato para construir un prototipo.

**PASO 2.**

Dado un proyecto candidato aceptable, el analista desarrolla una representación abreviada de los requerimientos.Antes de que pueda comenzar la construcción de un prototipo, el analista debe representar los dominios funcionales y de información del programa y desarrollar un método razonable de partición. La aplicación de estos principios de análisis fundamentales, pueden realizarse mediante los métodos de análisis de requerimientos.

**PASO 3.**

Después de que se haya revisado la representación de los requerimientos, se crea un conjunto de especificaciones de diseño abreviadas para el prototipo.El diseño debe ocurrir antes de que comience la construcción del prototipo. Sin embargo, el diseño de un prototipo se enfoca normalmente hacia la arquitectura a nivel superior y a los aspectos de diseño de datos, en vez de hacia el diseño procedimental detallado.

**PASO 4.**

El software del prototipo se crea, prueba y refina idealmente, los bloques de construcción de software preexisten se utilizan para crear el prototipo de una forma rápida. Desafortunadamente, tales bloques construidos raramente existen.Incluso si la implementación de un prototipo que funcione es impracticable, el escenario de construcción de prototipos puede aun aplicarse. Para las aplicaciones interactivas con el hombre, es posible frecuentemente crear un prototipo en papel que describa la interacción hombre-maquina usando una serie de hojas de historia.

**PASO 5.**

Una vez que el prototipo ha sido probado, se presenta al cliente, el cual "conduce la prueba" de la aplicación y sugiere modificaciones.Este paso es el núcleo del método de construcción de prototipo. Es aquí donde el cliente puede examinar una representación implementada de los requerimientos del programa, sugerir modificaciones que harán al programa cumplir mejor las necesidades reales.

**PASO 6.**

Los pasos 4 y 5 se repiten iterativamente hasta que todos los requerimientos estén formalizados o hasta que el prototipo haya evolucionado hacia un sistema de producción. El paradigma de construcción del prototipo puede ser conducido con uno o dos objetivos en mente.

# Protégé

## Justificación

Se optará por utilizar la herramienta Protégé ya que nos permitirá la creación y edición de las ontologías del dominio y asistirá en la construcción de la base de conocimiento [Gennari, 2003]. Además provee mecanismos de visualización alternativa, permitiendo administrar múltiples ontologías, el uso de motores de inferencia y métodos de resolución de problemas. A partir de las ontologías definidas la herramienta posibilitará construir automáticamente un sistema grafico de adquisición de conocimiento que concederá a los especialistas de la aplicación ingresar el conocimiento requerido para las aplicaciones especificas.

## Herramienta

Las aplicaciones Protégé [Puerta et al., 1992; Eriksson et al., 1995; Tu et al., 1995] son un conjunto de herramientas qué han ido evolucionando durante una década, pasando de un simple programa para la construcción de bases de conocimiento, a convenirse en una herramienta para construir y mantener bases de conocimiento de propósito general. El objetivo principal que guió el desarrollo del proyecto Protégé fue el de reducir el cuello de botella de la adquisición de conocimiento minimizando el papel del ingeniero del conocimiento, al proporcionar herramientas que interactúan directamente con los expertos.

PROTÉGÉ-II proporciona un entorno para ingeniería del conocimiento en el que el ingeniero del conocimiento puede especificar tareas y seleccionar métodos de resolución de una librería de métodos reutilizables. Para permitir el modelado de conocimiento, se distinguen los siguientes elementos:

* **Tareas:** Es un problema en el mundo real, localizado dentro de una organización, o una especificación abstracta de objetivos independientes del dominio que tienen que ser alcanzados (clases de tareas, o tarea). La descripción de una tarea es una especificación abstracta del problema que tiene que ser resuelto por el método. Debe incluir definiciones abstractas de los datos disponibles (entradas) y de la solución (salida), y las relaciones que se deben mantener entre ellos. Aunque en la especificación de una tarea no se indica cómo se resuelve, ésta condiciona la selección del método para su resolución.
* **Método:** Un método es un modelo independiente del dominio, que indica cómo se resuelve el problema especificado por la tarea. Además de la especificación de la entrada y la salida de la tarea, el método posee una definición abstracta de los roles que juega el conocimiento del dominio en la resolución del problema. La selección de un método depende de factores que van más allá de la especificación de la tarea, como la disponibilidad de la experiencia, requisitos de espacio y tiempo de computación, y la compatibilidad con el resto de los métodos que cooperan en la resolución del problema. Los métodos pueden delegar problemas a subtareas que, a su vez, pueden ser resueltas por otros métodos, formando un árbol de tareas-métodos-subtareas.
* **Mecanismos:** Los mecanismos se pueden considerar como métodos primitivos que no pueden ser descompuestos en otras subtareas. Por lo tanto, pueden ser considerados como cajas negras que no admiten descomposición.

Las tareas pueden ser resueltas por diferentes métodos, que a su vez pueden delegar en otras sub-tareas que pueden ser descompuestas. Esta descomposición termina cuando se alcanzan los mecanismos. Esta estrategia de descomposición ayuda a la reutilización de componentes de dos formas: por un lado, los mecanismos, al desarrollar funciones muy definidas, pueden ser aplicables a varios dominios : y por otro, la existencia de métodos descomponibles hace más flexible la reutilización, ya que permiten el uso de métodos y mecanismos alternativos para la resolución de sus subtareas.

Además de estos elementos relacionados con la estructura de tareas-métodos, hay que definir los elementos de conocimiento que van a intervenir en dicha estructura de tarea. Estos elementos se describen a través de dos ontologías:

• La ontología del dominio, que incluye una descripción declarativa del dominio, y. representa la conceptualización del dominio de la aplicación (por ejemplo, si estamos diseñando un SBC en el dominio médico tendremos conceptos como: hallazgos, signos, test de laboratorio...).

• La ontología de los métodos, qué define los conceptos y relaciones que utilizan los métodos, y sirve de marco para especificar sus entradas y salidas. La ontología de los métodos se corresponde con la terminología genérica utilizada por el conjunto de roles del método (por ejemplo, en una tarea de diagnóstico tendríamos conceptos como manifestaciones e hipótesis).

La construcción de un modelo de conocimiento se puede hacer en tres pasos:

* 1. Formulación de la ontología del método, incluyendo el conjunto de roles de conocimiento que utiliza.
  2. Definición de la ontología de la aplicación independientemente de la del método y reutilizando alguna ontología del dominio previamente definida.
  3. Formulación de las relaciones de correspondencia entre las dos ontologías (aplicación y método).

Aparte de la definición de nuevos modelos por reutilización, una de las características importantes de PROTÉGÉ-II es que facilita la generación de herramientas para la adquisición de conocimiento de forma independiente de los métodos que se vayan a utilizar. La generación de la herramienta de adquisición de conocimiento se hace de manera gráfica y dirigida por la ontología del dominio. En PROTÉGÉ-II se propone un análisis de la adquisición de conocimiento que presenta las siguientes fases:

1. Identificación de los usuarios potenciales de la herramienta de adquisición.
2. Segmentación de la base de conocimiento y determinación de qué parte va a ser adquirida por la herramienta.
3. Definición de un lenguaje en el que expresar el conocimiento (puede ser un lenguaje gráfico).
4. Especificación de la semántica para dicho lenguaje, además de la semántica denotacional que describe la generación de la base de conocimiento.

Como podemos ver, el desarrollo de una herramienta para la adquisición de conocimiento requiere un proceso previo de análisis. Una vez terminado este análisis, PROTÉGÉ-II facilita el proceso de desarrollo de la herramienta de adquisición.

Como se ha descrito, PROTÉGÉ-II define un marco que permite la especificación de nuevos modelos de conocimiento por reutilización de componentes incluidos en una librería. Este nuevo modelo puede ser creado por reconfiguración de alguno de los componentes de la librería o por combinación de varios componentes de la misma. La diferencia principal de esta aproximación con otras basadas en la reutilización, como puede ser la de los componentes de conocimiento [Steels, 1990], es que está orientada a la minimización del coste de programación. Sin embargo, una de las debilidades más importantes es que no aporta la definición de un ciclo de vida completo para el desarrollo de aplicaciones al estar más centrado en la fase de adquisición de conocimiento. Esta falta de definición de un ciclo de vida trae consigo un problema importante, y es que al carecer de un análisis de contexto, se complica el uso de la herramienta para grandes aplicaciones que tengan que trabajar en un entorno determinado.

# UML - Lenguaje del Modelado Unificado

UML son las siglas para Unified Modeling Language, que en castellano quiere decir: Lenguaje de Modelado Unificado. Para comprender qué es el UML, basta con analizar cada una de las palabras que lo componen, por separado.

* **Lenguaje:** el UML es, precisamente, un lenguaje. Lo que implica que éste cuenta con una sintaxis y una semántica. Por lo tanto, al modelar un concepto en UML, existen reglas sobre cómo deben agruparse los elementos del lenguaje y el significado de esta agrupación.
* **Modelado:** el UML es visual. Mediante su sintaxis se modelan distintos aspectos del mundo real, que permiten una mejor interpretación y entendimiento de éste.
* **Unificado**: unifica varias técnicas de modelado en una única.

Ya que el UML proviene de técnicas orientadas a objetos, se crea con la fuerte intención de que este permita un correcto modelado orientado a objetos.

El lenguaje unificado de diagrama o notación (UML) sirve para especificar, visualizar y documentar esquemas de sistemas de software orientado a objetos. UML no es un método de desarrollo, lo que significa que no sirve para determinar qué hacer en primer lugar o cómo diseñar el sistema, sino que simplemente le ayuda a visualizar el diseño y a hacerlo más accesible para otros.

## Diagrama UML

UML se compone de muchos elementos de esquematización que representan las diferentes partes de un sistema de software. Los elementos UML se utilizan para crear diagramas, que representa alguna parte o punto de vista del sistema.

## Diagrama de casos de uso

Los diagramas de casos de uso describen las relaciones y las dependencias entre un grupo de *casos de uso* y los actores participantes en el proceso.

Los diagramas de casos de uso sirven para facilitar la comunicación con los futuros usuarios del sistema, y con el cliente, y resultan especialmente útiles para determinar las características necesarias que tendrá el sistema. En otras palabras, los diagramas de casos de uso describen *qué* es lo que debe hacer el sistema, pero no *cómo*.

## Caso de uso

Los casos de uso son descriptores de las interacciones típicas entre los usuarios de un sistema y ese mismo sistema. Representan el interfaz externo del sistema y especifican qué requisitos de funcionamiento debe tener este.

## Actor

Un actor es una entidad externa (de fuera del sistema) que interacciona con el sistema participando (y normalmente iniciando) en un caso de uso. Los actores pueden ser gente real (por ejemplo, usuarios del sistema), otros ordenadores o eventos externos.

Los actores no representan a personas *físicas* o a sistemas, sino su *rol*. Esto significa que cuando una persona interactúa con el sistema de diferentes maneras (asumiendo diferentes papeles), estará representado por varios actores

## Descripción de casos de uso

Las descripciones de casos de uso son reseñas textuales del caso de uso. Normalmente tienen el formato de una nota o un documento relacionado de alguna manera con el caso de uso, y explica los procesos o actividades que tienen lugar en el caso de uso.

## Diagrama de clases

Los diagramas de clases muestran las diferentes clases que componen un sistema y cómo se relacionan unas con otras. Los diagramas de clases son diagramas «estáticos» porque muestran las clases, junto con sus métodos y atributos, así como las relaciones estáticas entre ellas: qué clases «conocen» a qué otras clases o qué clases «son parte» de otras clases, pero no muestran los métodos mediante los que se invocan entre ellas.

## Clase

Una clase define los atributos y los métodos de una serie de objetos. Todos los objetos de esta clase (instancias de esa clase) tienen el mismo comportamiento y el mismo conjunto de atributos (cada objeto tiene el suyo propio).

## Diagramas de secuencia

Los diagramas de secuencia muestran el intercambio de mensajes en un momento dado. Los diagramas de secuencia ponen especial énfasis en el orden y el momento en que se envían los mensajes a los objetos.

Un Diagrama de Secuencia consta de objetos, mensajes entre estos objetos y una línea de vida del objeto representada por una línea vertical

## Diagrama de actividad

Los diagramas de actividad describen la secuencia de las actividades en un sistema. Los diagramas de actividad son una forma especial de los diagramas de estado, que únicamente (o mayormente) contienen actividades.

Los diagramas de actividad siempre están asociados a una *clase*, a una *operación* o a un *caso de uso*.

Los diagramas de actividad soportan actividades tanto secuenciales como paralelas.

# Estudio de viabilidad

El estudio de viabilidad permite evaluar si el problema a resolver conviene ser tratado con las técnicas de Ingeniería del Conocimiento para el desarrollo de un Sistema Experto. La evaluación a la que se hace referencia se realiza a través del test de Viabilidad propuesto por la metodología IDEAL (Gómez, Juristo, Montes, & Pazos, 1997) donde se determina si el proyecto es posible, está justificado, es apropiado y si será exitosa su construcción.

## Consideraciones referentes al test de viabilidad

El test de viabilidad consta de varias características que se deben tener en cuenta antes de desarrollar un Sistema Experto. Esas características están divididas en cuatro dimensiones: Plausibilidad, Justificación, Adecuación y Éxito (Gómez, Juristo, Montes, & Pazos, 1997). Para el estudio completo de las características, se analiza el problema y se realiza su valoración. Las características tienen asociadas una serie de propiedades que las representan, y deben ser consideradas en el uso del test de viabilidad: Categoría, Peso, Naturaleza del valor asociado a la característica, Tipo, Umbral y Valor. Finalizada la valoración de las características, se comienza con el cálculo del test de viabilidad. La siguiente fórmula se aplica a las dimensiones de Adecuación, Plausibilidad y Éxito. Para cuantificar cada dimensión el método propone obtener la media armónica y la media aritmética del conjunto de intervalos y luego calcular la media aritmética de los dos valores obtenidos:

 **(1)**

En (1):

: Valor global de la aplicación en una dimensión dada.   
: Valor de la característica k en la dimensión i.   
: Peso de la característica k en la dimensión i.  
: Número de características en la dimensión i.

El estudio de viabilidad del proyecto concluye con el cálculo del valor final, mediante la media aritmética ponderada de los valores obtenidos en cada una de las dimensiones. La fórmula final es la siguiente:

 **(2)**

En (2):

: Valor obtenido en la dimensión j.   
: Peso obtenido en la dimensión j.   
: Valor final

Para que un proyecto sea viable es necesario que el valor final presente un valor mayor o igual a 6.

## Evaluación de viabilidad del proyecto

La información necesaria para realizar esta evaluación se obtuvo de las primeras entrevistas efectuadas al equipo de expertos, lo que permitió analizar y valorar las características de la tarea en estudio. Ver Apéndice: Evaluación de las características del problema. Basados en la métrica propuesta, se llega a los resultados que se presentan en la Tabla 2, en la que se detallan las dimensiones, el peso que le corresponde a cada una de ellas y los valores calculados. Las magnitudes establecidas para los pesos son los recomendados por los autores de la métrica en base a su experiencia.

Tabla 1 - Valor final del estudio de viabilidad

|  |  |  |
| --- | --- | --- |
| Dimensiones | Peso | Valor global de la aplicación en una dimensión |
| Plausibilidad | 8 | VCj = (8.16,8.55,8.97,8.97) |
| Justificación | 3 | VCj = (7.8,8.8,10,10) |
| Adecuación | 8 | VCj = (5.86,7.03,8.2,8.73) |
| Éxito | 5 | VCj = (6.11,7,8.2,8.86) |
| Valor final |  | VF = (6.92,7.75,8.68,8.99) |

Para que un proyecto sea viable con la tecnología de los SSBBCC, el resultado del promedio de los componentes del vector VF, debe ser mayor o igual a 6.

# GitHub

## Justificación

Se optará por utilizar la herramienta Github ya que nos permitirá usar una plataforma de desarrollo colaborativo de software para alojar el proyecto utilizando el sistema de control de versiones [Git](http://es.wikipedia.org/wiki/Git) (GitHub Inc, 2008), además que posee tu repositorio de código y brinda herramientas muy útiles para el trabajo en equipo. El control de versiones es una de las tareas fundamentales para la administración de un proyecto de desarrollo de software. Surge de la necesidad de mantener y llevar control del código que vamos programando, conservando sus distintos estados. Es absolutamente necesario para el trabajo en equipo, pore so se opto por la utilizacion de GitHub.

## Herramienta

Git es un sistema de código abierto y distribuido de control de versiones inventado por Linus Torvalds en 2005. Se utiliza para versionar el kernel de Linux , Git es multiplataforma, por lo que se puede usarlo y crear repositorios locales en todos los sistemas operativos más comunes, Windows, Linux o Mac. Existen multitud de GUIs (Graphical User Interface o Interfaz de Usuario Gráfica) para trabajar con Git.

**Sobre Github:** Es un servicio para alojamiento de repositorios de software gestionados por el sistema de control de versiones Git. Por tanto, Git es algo más general que sirve para controlar el estado de un desarrollo a lo largo del tiempo, mientras que Github es algo más particular: un sitio web que usa Git para ofrecer a la comunidad de desarrolladores repositorios de software. En definitiva, Github es un sitio web pensado para hacer posible el compartir el código de una manera más fácil y al mismo tiempo darle popularidad a la herramienta de control de versiones en sí, que es Git. En Github es gratuito alojar proyectos *Open Source*, lo que ha posibilitado que el número de proyectos no pare de crecer, y en estos momentos haya varios millones de repositorios y usuarios trabajando con la herramienta.

GitHub es mucho más que un servicio de alojamiento de código. Además de éste, se ofrecen varias herramientas útiles para el trabajo en equipo. Entre ellas:

* Una wiki para el mantenimiento de las distintas versiones de las páginas.
* Un sistema de seguimiento de problemas que permiten a los miembros de tu equipo detallar un problema con tu software o una sugerencia que deseen hacer.
* Una herramienta de revisión de código, donde se pueden añadir anotaciones en cualquier punto de un fichero y debatir sobre determinados cambios realizados en un commit específico.
* Un visor de ramas donde se pueden comparar los progresos realizados en las distintas ramas de nuestro repositorio.