## Glosario

A continuación se presentan todos los términos manejados a lo largo del proyecto de desarrollo del sistema FM System de la empresa **Floreria Maria’z**

## 2.1Contraseña: Es una forma de [autentificación](http://es.wikipedia.org/wiki/Autentificaci%C3%B3n) para el usuario, que utiliza

## [información](http://es.wikipedia.org/wiki/Informaci%C3%B3n) secreta para controlar el acceso al software.

## 2.2Diagrama Actividades

Un diagrama de Actividad demuestra la serie de actividades que deben ser realizadas en un uso-caso, así como las distintas rutas que pueden irse desencadenando en el uso-caso.

**2.3Diagrama Caso de uso**

Un diagrama Uso-Caso describe lo que hace un sistema desde el punto de vista de un observador externo.

**2.4Diagrama Clases**

Un diagrama de Clases representa las clases que serán utilizadas dentro del

Sistema y las relaciones que existen entre ellas. Los diagramas de Clases por definición son estáticos, esto es, representan que partes interactúan entre sí, no lo que ocurre cuando.

**2.5Diagrama Componentes**

Los Diagramas de Componentes ilustran las piezas del software, controladores embebidos, etc. Que conformarán un sistema. Un diagrama de Componentes tiene un nivel más alto de abstracción que un diagrama de clase, usualmente un componente se implementa por una o más clases (u objetos) en tiempo de ejecución. Estos son bloques de construcción, como eventualmente un componente puede comprender una gran porción de un sistema.

**2.6Diagrama Deployment**

Es aquel que muestra las relaciones físicas entre los componentes de software y de hardware en el sistema entregado. Así, el diagrama de emplazamiento es un buen sitio para mostrar cómo se enrutan y se mueven los componentes y los objetos, dentro de un sistema distribuido.

**2.7Diagrama Estado**

Un Diagrama de Estados muestra la secuencia de estados por los que pasa bien un caso de uso, bien un objeto a lo largo de su vida, o bien todo el sistema. En él se indican qué eventos hacen que se pase de un estado a otro y cuáles son las respuestas y acciones que genera.

**2.8Diagrama Paquete**

Un diagrama de paquetes muestra como un sistema está dividido en agrupaciones lógicas mostrando las dependencias entre esas agrupaciones. Dado que normalmente un paquete está pensado como un directorio, los diagramas de paquetes suministran una descomposición de la jerarquía lógica de un sistema.

**2.9Diagrama Secuencia**

Un diagrama de secuencia es una forma de diagrama de interacción que muestra los objetos como líneas de vida a lo largo de la página y con sus interacciones en el tiempo representadas como mensajes dibujados como flechas desde la línea de vida origen hasta la línea de vida destino. Los diagramas de secuencia son buenos para mostrar qué objetos se comunican con qué otros objetos y qué mensajes disparan esas comunicaciones. Los diagramas de secuencia no están pensados para mostrar lógicas de procedimientos

Complejos.

**2.10Florista:**Persona encargada en realizar todo tipo de arreglos dentro de la florería.

**2.11Inventario:** es cuando la empresa hace, un margen adicional de pedido para la florería.

**2.12Propietaria:**Dueña de la florería y cliente principal.

**2.13RUP**

El Proceso Unificado de Rational (*Rational Unified Process* en inglés, habitualmente resumido como RUP) es un proceso de desarrollo de software y junto con el Lenguaje Unificado de Modelado [UML](http://es.wikipedia.org/wiki/UML), constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos.

**2.14Rational Rose**

El Proceso Unificado Racional (Rational Unified Process en inglés, habitualmente resumido como RUP) es un proceso de desarrollo de software y junto con el Lenguaje Unificado de Modelado [UML](http://es.wikipedia.org/wiki/UML), constituye la metodología estándar más

## 2.15UML

## Lenguaje Unificado de Modelado (LUM o UML, por sus siglas en inglés, *Unified Modeling Language*) es el lenguaje de [modelado](http://es.wikipedia.org/wiki/Modelado) de sistemas de [software](http://es.wikipedia.org/wiki/Software) más conocido y utilizado en la actualidad; Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocio, funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes reutilizables.

## 2.16Usuario

## Es la persona que cuenta con Información requerida para autentificarse y utilizar el sistema.

**Descriptor** : una función miembro que modifique o devuelve el valor de un campo. También conocido como un modificador de acceso. Ver [Getter y Setter](file:///E:\C:Archivos%20de%20programaRationalRationalUnifiedProcessmanualsjavajava.htm#Getter%23Getter) [E:\C:Archivos de programaRationalRationalUnifiedProcessmanualsjavajava.htm - Setter%23Setter](file:///E:\C:Archivos%20de%20programaRationalRationalUnifiedProcessmanualsjavajava.htm#Setter%23Setter).

**Análisis de** modelos: un modelo que describe una solución a un problema o dominio.

**Antipattern** : Un enfoque para la solución de un problema común, que con el tiempo resulta ser incorrecto o altamente ineficaz.

**Argumento** : Ver el parámetro [E:\C:Archivos de programaRationalRationalUnifiedProcessmanualsjavajava.htm - Parameter%23Parameter](file:///E:\C:Archivos%20de%20programaRationalRationalUnifiedProcessmanualsjavajava.htm#Parameter%23Parameter).

**BDK** : Frijol Kit de desarrollo

**Bloque** : una colección de cero o más instrucciones (llaves) entre llaves.

**Entre corchetes:** Los caracteres **{ y } ,** conocido como una llave de apertura y una llave de cierre, respectivamente, se utilizan para definir el comienzo y el final de un bloque.

**Clase:** una definición, o una plantilla, de la que se crean instancias de objetos.

**Prueba Clase** : El acto de asegurar que una clase y sus instancias (objetos) realizar tal como se define.

**CMVC** : Gestión de la configuración y el Control de la versión

**Unidad de compilación** : un archivo de código fuente, ya sea a nivel físico en el disco o "virtual" se almacena en una base de datos, en el que las clases e interfaces son declarados.

**Componente:** Una interfaz widget como una lista, botón, o ventana.

**Getter constante** : un captador función miembro que devuelve el valor de una "constante", que, a su vez, puede ser codificado o calcular si es necesario.

**Constructor:** una función miembro que realiza la inicialización necesaria cuando se crea un objeto.

**Contención** : Un objeto que contiene otros objetos que colabora con sus comportamientos. Esto puede llevarse a cabo mediante el uso de las clases internas (JDK 1.1 +) o la agregación de instancias de otras clases dentro de un objeto (JDK 1.0 + ).

**CPU:** unidad central de procesamiento

**C-style comentarios:** Java formato de comentario, / \* y \* /, aprobado en el lenguaje C/C++ que se pueden utilizar para crear múltiples comentarios. Comúnmente se utiliza para "documento" innecesarios o líneas de código no deseados durante la prueba.

**Patrón de Diseño** : un modelo patrón que describe una solución a un problema de diseño.

**Destructor** : una clase de C++ función miembro que se utiliza para eliminar un objeto de la memoria una vez que ya no la necesita. Porque Java maneja su propia memoria, este tipo de función miembro no es necesario. Java, sin embargo, apoyar una función miembro que es similar en el concepto llamado **finalize() .**

**Comentarios de documentación** : Java formato de comentario, / \*\* y \* /, que pueden ser procesados por *javadoc* para proporcionar documentación externa para un archivo de clase. La documentación principal para las interfaces, clases, funciones miembro, y los campos deberían estar escritas con comentarios de documentación.

**Campo:** una variable, un valor literal tipo de datos u otro objeto, que describe una clase o una instancia de una clase. Campos de Instancia describir los objetos (instancias) y los campos estáticos describir las clases. Campos también se las conoce como campos, variables de campo y las propiedades.

**Finalize()** : una función miembro que se invoca automáticamente durante la recolección de elementos no utilizados antes de que un objeto se elimina de la memoria. El objetivo de esta función miembro es hacer una limpieza general, como es el caso del cierre archivos abiertos.

**Recolección de basura** : La gestión automática de la memoria, donde los objetos que ya no se hace referencia se eliminan automáticamente de la memoria.

**Getter** : un tipo de descriptor miembro función que devuelve el valor de un campo. Un candidato puede ser utilizada para responder el valor de una constante, que a menudo es preferible a la aplicación de la constante como un campo estático ya que este es un enfoque más flexible.

**HTML** : lenguaje de marcado de hipertexto, un formato estándar en el sector para crear páginas Web.

**Sangría** : Véase [paragraphing](file:///E:\C:Archivos%20de%20programaRationalRationalUnifiedProcessmanualsjavajava.htm#Paragraphing%23Paragraphing) .

**Comentarios En Línea** : El uso de un comentario de línea al documento una línea de código fuente donde el comentario inmediatamente siguiente del código, en la misma línea que el código. Comentarios de una sola línea son generalmente utilizados para esto, aunque de estilo C comentarios también se pueden emplear.

**Interfaz:** La definición de una firma, incluyendo tanto las funciones miembro, y una clase que implementa una interfaz debe apoyar. Promover las Interfaces polimorfismo en su composición.

**E/S** : entrada/salida

**Invariante** : un conjunto de afirmaciones sobre una instancia o de la clase que debe ser cierto en todos los "estable" veces, tales como los períodos antes y después de la invocación de una función miembro sobre el objeto o clase.

**Java** : Un estándar de la industria, orientado a objetos y lenguaje de desarrollo que se adapta muy bien para el desarrollo de aplicaciones para Internet y las aplicaciones que deben funcionar en una amplia variedad de plataformas informáticas.

**Javadoc** : una utilidad incluida en el JDK que procesa un archivo de código fuente Java y produce un documento externo, en formato HTML, describe el contenido del archivo de código fuente basado en los comentarios de documentación en el archivo de código.

**JDK** : Java Development Kit

**Inicialización Diferida** : una técnica en la cual un campo se inicializa en su correspondiente función miembro proponga la primera vez que la necesite. Inicialización diferida se utiliza cuando un campo no es comúnmente necesaria y que requiere una gran cantidad de memoria para almacenar o que debe ser leído a partir de almacenamiento permanente.

**Variable Local** : una variable definida en el ámbito de un bloque, a menudo una función miembro. El alcance de una variable local es el bloque en el que se define.

**Función Miembro** : un trozo de código ejecutable asociado a una clase o las instancias de una clase. Pensar en una función miembro como el de la programación orientada a objetos equivalentes de una función.

**Función Miembro firma:** Ver [firma .](file:///E:\C:Archivos%20de%20programaRationalRationalUnifiedProcessmanualsjavajava.htm#Signature%23Signature)

**Ensayos del Método** : El acto de asegurar que una función miembro (función miembro) realiza tal como se define.

**Nombre ocultar** : Esto se refiere a la práctica de utilizar la misma, o al menos similar, el nombre de un campo, variable o argumento en cuanto a uno de mayor alcance. El más común uso indebido del nombre oculto es el de dar nombre a un variable local la misma como un campo de instancia. Nombre ocultar debe ser evitada, ya que permite que el código más difícil de entender y propenso a errores.

**Sobrecarga :** una función miembro se dice que está sobrecargada cuando es definido más de una vez en la misma clase (o en una subclase); la única diferencia es que la firma de cada definición.

**Anulación** : una función miembro se dice que se anula cuando se redefine en una subclase y tiene la misma firma que la definición original.

**Paquete** : una colección de clases relacionadas.

**Paragraphing** : una técnica donde sangría en el código que se encuentra dentro del ámbito de un bloque de código en una unidad, por lo general un tabulador horizontal, a fin de distinguirlo de la código fuera del bloque de código. Paragraphing ayuda a aumentar la legibilidad del código.

**Parámetro:** Uno de los argumentos pasados a una función miembro, un parámetro puede ser un tipo definido, como una cadena, o un valor de tipo int, o un objeto.

**Postcondition** : una propiedad o afirmación de que será cierto después de una función miembro está terminado.

**Condición previa** : una limitación en virtud de la cual una función miembro funcione correctamente.

**Propiedad** : Véase [campo .](file:///E:\C:Archivos%20de%20programaRationalRationalUnifiedProcessmanualsjavajava.htm#Field%23Field)

**Setter** : Un descriptor función miembro que establece el valor de un campo.

**Firma:** La combinación del tipo de los parámetros, si los hubiere, y el orden que se debe pasar a una función miembro. Esto se llama también la firma de la función miembro.

**Comentarios de una sola línea** : Java formato de comentario, // , adoptado desde el lenguaje C/C++ que se usa habitualmente para la documentación de las funciones miembro interno de la lógica de negocio.

**Tags** : un convenio para marcar secciones especificadas de comentarios de documentación que serán procesadas por *javadoc* de aspecto profesional comentarios. Ejemplos de etiquetas incluyen *@y* *@autor .*

**Cables de prueba** : una colección de funciones miembro para probar su código.

**UML** : lenguaje unificado de modelado, que es un estándar de la industria notación de modelado.

**Visibilidad** : una técnica que se utiliza para indicar el nivel de encapsulación de una clase, función miembro, o el campo. Las palabras clave de públicos, protegidos y privados) pueden utilizarse para definir visibilidad.

**Blanco** : líneas en blanco, espacios o tabuladores añadido al código para aumentar su legibilidad.

**Widget** : Véase [componente .](file:///E:\C:Archivos%20de%20programaRationalRationalUnifiedProcessmanualsjavajava.htm#Component%23Component)