



Benemérita Universidad Autónoma de Puebla

Facultad de Ciencias de la Computación

SISTEMA DE ADMINISTRACIÓN Y MONITOREO REMOTO VÍA INTERNET DE UNA ESTACIÓN METEOROLÓGICA

Tesis sometida como requisito para obtener el título de:
Licenciado en Ciencias de la Computación

Presenta:

Jesús Contreras Rodríguez

Asesor:

M.C. Miguel Rodríguez Hernández

**Puebla, Pue.
Marzo 2014**

Índice de contenido

INTRODUCCIÓN	1
Capítulo 1 MARCO TEÓRICO	3
1.1. LA ESTACIÓN METEOROLÓGICA	4
2.1.1. Variables meteorológicas	4
2.1.2. Sistema Solus	4
2.1.3. Configuración de los instrumentos de medición dentro del sistema Solus	5
2.1.4. Ecuaciones utilizadas por el sistema Solus	6
2.1.5. Comandos del sistema Solus	7
2.2. INGENIERÍA DE SOFTWARE	9
2.3. UP: Proceso Unificado de desarrollo de software	10
2.3.1. Características esenciales	10
2.3.2. Ciclo de vida	12
2.4. UML: LENGUAJE UNIFICADO DE MODELADO	14
2.4.1. Historia	14
2.4.2. Definición	15
2.4.3. Modelos en el UML	15
2.5. MODELOS DE PROCESO DE SOFTWARE	17
2.5.1. <i>Modelo lineal o de cascada</i>	17
2.5.2. Modelo en espiral	18
2.5.3. Modelo de prototipos	19
2.5.4. Modelo incremental	20
2.6. BASES DE DATOS	22
2.6.1. Sistema gestor de bases de datos	22
2.6.2. Usuarios de las bases de datos	23
2.7. MODELO DE BASES DE DATOS	23
2.7.1. Modelo Entidad-Relación (E-R)	25
2.7.2. Modelo Relacional	27
2.7.3. Llaves primarias y foráneas	28
2.8. NORMALIZACIÓN	29
2.8.1. Formas normales	29
2.8.2. Tercera Forma Normal (3FN)	31
2.9. ARQUITECTURA CLIENTE-SERVIDOR	33
2.9.1. Código abierto (Open source)	34
2.9.2. Servidor Apache Tomcat	34
2.9.3. La tecnología Java	35
2.9.4. MySQL	36
Capítulo 2 ANÁLISIS Y DISEÑO	37
2.1. INTRODUCCIÓN	38
2.2. PLANTEAMIENTO DEL PROBLEMA	38
2.3. OBJETIVOS	39

2.4.	REQUERIMIENTOS FUNCIONALES Y NO FUNCIONALES DEL SISTEMA _____	39
2.5.	ALCANCES _____	41
2.6.	METODOLOGÍA DE DESARROLLO Y ANALISIS DEL SISTEMA _____	41
2.6.1.	Glosario de términos _____	41
2.6.2.	Diagrama de casos de uso _____	43
2.6.3.	Descripción de los actores del sistema _____	44
2.6.4.	Especificación de los casos de uso _____	46
2.6.5.	Escenarios _____	55
2.6.6.	Diagrama de clases _____	61
2.7.	DISEÑO DEL SISTEMA _____	63
2.7.1.	Diagramas VOPC (Vista de Clases Participantes) _____	63
2.7.2.	Diagramas de secuencia _____	66
2.7.3.	Diagramas de colaboración _____	72
Capítulo 3	DISEÑO DE LA BASE DE DATOS _____	78
3.1.	DISEÑO CONCEPTUAL DE LA BASE DE DATOS _____	79
3.1.1.	Modelo Entidad-Relación _____	80
3.1.2.	Descripción de entidades, relaciones y atributos _____	81
3.1.3.	Diseño lógico de la base de datos _____	82
3.1.4.	Normalización de la base de datos _____	84
3.1.5.	Implementación del modelo relacional usando MySQL Workbench _____	87
Capítulo 4	IMPLEMENTACIÓN Y PRUEBAS DEL SISTEMA _____	89
4.1.	IMPLEMENTACIÓN _____	90
4.1.1.	Implementación de los procesos del servidor _____	90
4.1.2.	Implementación de los procesos del cliente _____	91
4.1.3.	Software utilizado para la implementación del sistema _____	91
4.2.	PANTALLAS DEL SISTEMA _____	96
4.2.1.	Página web principal del sistema _____	96
4.2.2.	Página web de la opción Gráficas _____	97
4.2.3.	Página web de la opción Archivo _____	98
4.3.	PRUEBAS DEL SISTEMA _____	101
4.3.1.	Pruebas a la página web principal del sistema y consulta de gráficas del día actual	101
4.3.2.	Pruebas al formulario para la consulta de gráficas de datos y estadísticas ocurridos en días anteriores _____	104
4.3.3.	Seguridad del sistema _____	109
4.3.4.	Respaldo de datos _____	110
CONCLUSIONES	_____	113
TRABAJOS A FUTURO	_____	114
REFERENCIAS BIBLIOGRÁFICAS	_____	115
REFERENCIAS ELECTRÓNICAS	_____	119

Índice de ilustraciones

<i>Ilustración 1.1: Elementos del sistema Solus</i>	4
<i>Ilustración 1.2: Ciclo de vida del UP</i>	12
<i>Ilustración 1.3: Fases del UP</i>	13
<i>Ilustración 1.4: Línea cronológica del UML</i>	15
<i>Ilustración 1.5: Organigrama de los diagramas en el UML</i>	16
<i>Ilustración 1.6: Etapas del modelo lineal o de cascada</i>	17
<i>Ilustración 1.7: Etapas del modelo en espiral</i>	19
<i>Ilustración 1.8: Etapas del modelo de prototipos</i>	20
<i>Ilustración 1.9: Etapas del modelo incremental</i>	20
<i>Ilustración 1.10: Desarrollo de modelos de datos [22]</i>	24
<i>Ilustración 1.11: Diagrama E-R con los diferentes tipos de relación [22]</i>	25
<i>Ilustración 1.12: Diagrama E-R usando el modelo "pata de gallo"[22]</i>	26
<i>Ilustración 1.13: Estructura básica usada en el modelo relacional</i>	27
<i>Ilustración 1.14: Ejemplo del uso de llaves primarias y foráneas dentro de una base de datos</i>	28
<i>Ilustración 1.15: Relación de ejemplo no normalizada</i>	30
<i>Ilustración 1.16: Relaciones en 1FN.</i>	30
<i>Ilustración 1.17: Relación de ejemplo para normalizar a 2NF</i>	31
<i>Ilustración 1.18: Relaciones en 2NF</i>	31
<i>Ilustración 1.19: Relación de ejemplo en 2FN</i>	32
<i>Ilustración 1.20: Relaciones en 3FN</i>	32
<i>Ilustración 1.21: Esquema de la arquitectura Cliente-Servidor</i>	33
<i>Ilustración 1.22: Una visión general del proceso de desarrollo de software en Java</i>	35
<i>Ilustración 1.23: Java VM, la misma aplicación es capaz de ejecutarse sobre múltiples plataformas.</i>	36
<i>Ilustración 2.1: Diagrama general de casos de uso del sistema</i>	43
<i>Ilustración 2.2: Diagrama de clases básico del sistema</i>	61
<i>Ilustración 2.3: Diagrama VOPC Caso de uso CU-1</i>	63
<i>Ilustración 2.4: Diagrama VOPC Caso de uso CU-2</i>	64
<i>Ilustración 2.5: Diagrama VOPC Caso de uso CU-3</i>	64
<i>Ilustración 2.6: Diagrama VOPC Caso de uso CU-4</i>	65
<i>Ilustración 2.7: Diagrama VOPC Caso de uso CU-5</i>	65
<i>Ilustración 2.8: Diagrama de secuencia del CU-1</i>	66
<i>Ilustración 2.9: Diagrama de secuencia del CU-2</i>	67
<i>Ilustración 2.10: Diagrama de secuencia del CU-3</i>	68
<i>Ilustración 2.11: Diagrama de secuencia del CU-4</i>	69
<i>Ilustración 2.12: Diagrama de secuencia CU-5.1</i>	70
<i>Ilustración 2.13: Diagrama de secuencia del CU-5.2</i>	71
<i>Ilustración 2.14: Diagrama de colaboración del CU-1</i>	72
<i>Ilustración 2.15: Diagrama de colaboración CU-2</i>	73

<i>Ilustración 2.16: Diagrama de colaboración del CU-3</i>	74
<i>Ilustración 2.17: Diagrama de secuencia del CU-4</i>	75
<i>Ilustración 2.18: Diagrama de colaboración del CU-5.1</i>	76
<i>Ilustración 2.19: Diagrama de colaboración del CU-5.2</i>	77
<i>Ilustración 3.1: Diagrama Entidad-Relación para el sistema</i>	80
<i>Ilustración 3.2: Modelo relacional para el sistema realizado en MySQL Workbench</i>	87
<i>Ilustración 4.1: Esquema básico del funcionamiento de un servlet</i>	90
<i>Ilustración 4.2: Pantalla de bienvenida del servidor web Apache Tomcat</i>	93
<i>Ilustración 4.3: Tablas de la base de datos del sistema</i>	95
<i>Ilustración 4.4: Página web principal del sistema</i>	96
<i>Ilustración 4.5: Página web para la opción "Gráficas"</i>	97
<i>Ilustración 4.6: Página web para la opción "Archivo"</i>	98
<i>Ilustración 4.7: Ejemplo de una respuesta a una consulta de datos de una fecha pasada</i>	99
<i>Ilustración 4.8: Ejemplo de una respuesta a una consulta de estadística de una fecha pasada</i>	100
<i>Ilustración 4.9: Definición del puerto serie en la base de datos</i>	101
<i>Ilustración 4.10: Definición de los instrumentos de medición en la base de datos</i>	101
<i>Ilustración 4.11: Pantalla Conectado con el servidor</i>	102
<i>Ilustración 4.12: Pantalla Conexión exitosa entre el cliente y el servidor</i>	102
<i>Ilustración 4.13: Pantalla Servidor fuera de servicio</i>	103
<i>Ilustración 4.14: Pantalla Gráficas del día actual no disponibles</i>	103
<i>Ilustración 4.15: Pantalla: Formulario para consulta de gráficas y estadísticas de días anteriores</i>	104
<i>Ilustración 4.16: Mensaje de error para una fecha muy antigua</i>	105
<i>Ilustración 4.17: Mensaje de error para una fecha que aún no ocurre</i>	105
<i>Ilustración 4.18: Mensaje de error de datos no disponibles</i>	105
<i>Ilustración 4.19: Introducción de un periodo de tiempo usando un plugin</i>	106
<i>Ilustración 4.20: Ejemplo de una consulta de gráficas de datos con un periodo de tiempo de una hora</i>	107
<i>Ilustración 4.21: Definición de la contraseña para el Administrador en el servidor Apache Tomcat</i>	109
<i>Ilustración 4.22: Definición de privilegios a los usuarios de la base de datos</i>	110
<i>Ilustración 4.23: Programación en Windows de la tarea para ejecutar el script que realiza el respaldo de la base de datos</i>	112

Índice de tablas

<i>Tabla 1.1: Tipo de puertos usados por los instrumentos de medición en el sistema Solus</i>	<i>5</i>
<i>Tabla 1.2: Lista de parámetros para configurar un instrumento en el sistema Solus</i>	<i>6</i>
<i>Tabla 2.1: Detalles del actor "Reloj CPU"</i>	<i>44</i>
<i>Tabla 2.2: Detalles del actor "Usuario web"</i>	<i>44</i>
<i>Tabla 2.3: Detalles del actor "Datalogger"</i>	<i>45</i>
<i>Tabla 2.4: Detalles del actor "Base de datos"</i>	<i>45</i>
<i>Tabla 2.5: Especificación del caso de uso CU-1</i>	<i>46</i>
<i>Tabla 2.6: Especificación del caso de uso CU-2</i>	<i>47</i>
<i>Tabla 2.7: Especificación del caso de uso CU-3</i>	<i>48</i>
<i>Tabla 2.8: Especificación del caso de uso CU-4</i>	<i>49</i>
<i>Tabla 2.9: Especificación del caso de uso CU-5 (parte 1 de 2)</i>	<i>50</i>
<i>Tabla 2.10: Especificación del caso de uso CU-5 (parte 2 de 2)</i>	<i>51</i>
<i>Tabla 2.11: Especificación del caso de uso CU-6 (parte 1 de 3)</i>	<i>52</i>
<i>Tabla 2.12: Especificación del caso de uso CU-6 (parte 2 de 3)</i>	<i>53</i>
<i>Tabla 2.13: Especificación del caso de uso CU-6 (parte 3 de 3)</i>	<i>54</i>
<i>Tabla 2.14: Ejemplo de un escenario para el CU-1</i>	<i>55</i>
<i>Tabla 2.15: Ejemplo de un escenario para el CU-2</i>	<i>56</i>
<i>Tabla 2.16: Ejemplo de un escenario para el CU-3</i>	<i>57</i>
<i>Tabla 2.17: Ejemplo de un escenario para el CU-4</i>	<i>58</i>
<i>Tabla 2.18: Ejemplo de un escenario para el CU-5 (parte 1 de 2)</i>	<i>59</i>
<i>Tabla 2.19: Ejemplo de un escenario para el CU-5 (parte 2 de 2)</i>	<i>60</i>
<i>Tabla 2.20: Diccionario de clases del sistema (parte 1 de 2)</i>	<i>61</i>
<i>Tabla 2.21: Diccionario de clases del sistema (parte 2 de 2)</i>	<i>62</i>
<i>Tabla 3.1: Normalización de la tabla Datalogger</i>	<i>84</i>
<i>Tabla 3.2: Normalización de la tabla Instrumento</i>	<i>85</i>
<i>Tabla 3.3: Normalización de la tabla Temperatura</i>	<i>85</i>
<i>Tabla 3.4: Normalización de la tabla EstadísticaMeteorología</i>	<i>86</i>
<i>Tabla 4.1: Muestra del contenido de la bitácora del sistema</i>	<i>108</i>
<i>Tabla 4.2: Muestra del contenido de un archivo de texto con información respaldada</i>	<i>110</i>
<i>Tabla 4.3: Contenido del script que realiza el respaldo de la base de datos</i>	<i>111</i>
<i>Tabla 4.4: Comandos para recuperar la base de datos desde un archivo de respaldo</i>	<i>112</i>

Índice de ecuaciones

<i>Ecuación 1.1: Cálculo del voltaje de entrada</i>	<u>6</u>
<i>Ecuación 1.2: Cálculo de la variable meteorológica de tipo analógico</i>	<u>7</u>
<i>Ecuación 1.3: Cálculo de la variable meteorológica de tipo digital</i>	<u>7</u>

INTRODUCCIÓN

Actualmente el diseño de aplicaciones que utilizan Internet y World Wide Web es muy popular y por consecuencia la necesidad de estas soluciones informáticas es muy solicitada hoy en día. El uso de estas aplicaciones permite tener acceso a información que se encuentra almacenada en otras partes del mundo. Ahora las interfaces son diseñadas para ser utilizadas a través de un navegador web y ofrecen ambientes gráficos agradables e intuitivos que el usuario llega a entender y a manipular rápidamente. Existen muchas empresas e instituciones que ocupan estas aplicaciones para ofrecer a los usuarios acceso a noticias, eventos, recursos multimedia, etc. Una de las más comunes es la relacionada a la publicación de las condiciones meteorológicas de algún lugar en especial.

El uso de las estaciones meteorológicas permite medir las condiciones ambientales que están sucediendo en ese momento, tales como temperatura, humedad, velocidad del viento, etc. Esta información es utilizada para hacer predicciones o pronósticos que resultan ser importantes para la toma de decisiones y lo son aún más cuando se realizan actividades donde es crítico saber constantemente el valor de una variable meteorológica. Las estaciones meteorológicas modernas cuentan con módulos electrónicos para que puedan ser integradas a Internet y ofrecen software tales como Weather Station Software [E1], Virtual Weather Station [E2], Weather Display [E3] para su administración remota y la publicación de datos meteorológicos en sitios web especializados, por ejemplo The Weather Channel [E4] o Weather Underground [E5]; entre otros más.

Son muchos los fabricantes de estaciones meteorológicas como Davis Weather [E6], R. M. Young [E7] o Texas Weather Instruments [E8] que ofrecen estas prestaciones en sus productos, aunque la mayoría las coloca como opciones y es necesario pagar licencias y/o suscripciones para tener acceso completo a los servicios web que dan soporte a sus estaciones. También el software que ofrecen suele tener tantas opciones que no se llega a explotar del todo o no son del todo útiles para los usuarios. En esta parte, el usuario tiene la oportunidad de desarrollar su propio software que se ajuste a sus necesidades y requerimientos. Existe gran cantidad de recursos disponibles (Java, PHP, Apache HTTP Server, Apache Tomcat, MySQL, etc.) para realizar esa tarea y muchos pueden ser utilizados sin pagar licencia alguna, pues se ubican en la categoría de software libre o de código abierto.

En la presente tesina se describe el diseño y desarrollo de procedimientos para comunicar remotamente a través de Internet una estación meteorológica. Se desarrolló un sistema informático para el intercambio seguro de datos entre un servidor y un cliente a través de Internet.

La solución ofrece el beneficio de tener la información de manera remota de las condiciones meteorológicas que adquieren los instrumentos de la estación meteorológica.

Esta característica ayuda a reducir la pérdida de información, así como también a tomar decisiones rápidas y eficaces si se observan comportamientos no deseados. Otras de sus ventajas es la de almacenar en un servidor web y en formatos convenientes toda los datos meteorológicos obtenidos y permite que cualquier persona pueda consultarla con solo tener una computadora con acceso a Internet.

CAPÍTULO 1

MARCO TEÓRICO

1.1. LA ESTACIÓN METEOROLÓGICA

Una estación meteorológica es una instalación dedicada a medir y registrar regularmente diversas variables meteorológicas. Estos datos se utilizan tanto para la proveer información del clima que ocurre en el sitio donde se encuentra instalada, la elaboración de predicciones meteorológicas a partir de modelos matemáticos, así como para estudios climáticos.

En la actualidad, la mayor parte de las estaciones meteorológicas se hallan automatizadas, de forma que puedan registrar la información durante todo el tiempo y solo se requiera de un mantenimiento ocasional.

2.1.1. Variables meteorológicas

Normalmente una estación meteorológica incluye una serie de instrumentos que permiten medir diferentes variables meteorológicas. La estación meteorológica involucrada en este proyecto registra las siguientes:

1. *Temperatura*: medida en grados Celsius ($^{\circ}\text{C}$)
2. *Humedad relativa*: medida en tanto por ciento (%RH)
3. *Presión Barométrica*: medida en milibares (mb)
4. *Intensidad de la luz solar*: medida en potencia incidente por unidad de superficie (W/m^2)
5. *Velocidad del viento*: medida en kilómetros por hora (km/h)
6. *Dirección del viento*: medida en grados sexagesimales (Deg)
7. *Precipitación pluvial*: medida en centímetros cúbicos (cm^3)

2.1.2. Sistema Solus

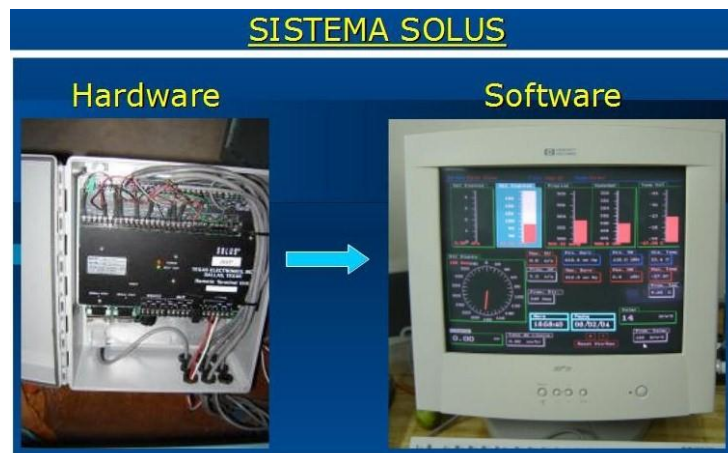


Ilustración 1.1: Elementos del sistema Solus

En la ilustración 1.1 se muestra el sistema Solus que se compone de una computadora de propósito específico equipada con un conjunto de puertos analógicos y digitales de entrada/salida, contadores y acumuladores; que puede detectar y hacer el monitoreo de eventos y registrar en su memoria los datos adquiridos.

El sistema Solus se puede comunicar con una PC por medio del puerto de comunicaciones serial (RS-232) y puede ser controlado usando el conjunto de comandos incluidos en su sistema operativo. También cuenta con software proporcionado por el fabricante.

2.1.3. Configuración de los instrumentos de medición dentro del sistema Solus

Cada uno de los instrumentos que conforman la estación meteorológica debe estar configurado correctamente dentro del sistema Solus para que la información obtenida sea la correcta. Los instrumentos envían señales provenientes de sus propios sensores y son recibidas por el sistema Solus a través de sus múltiples puertos. La señal recibida se le da un valor numérico calculado por una serie de ecuaciones. Cada instrumento cuenta con un conjunto de parámetros que determinan los valores de operación del instrumento. Estos parámetros son utilizados en las ecuaciones para poder obtener el valor de la variable meteorológica correspondiente.

En la tabla 1.1 se señala el tipo de puerto que ocupa cada instrumento dentro del sistema Solus.

Instrumento	Tipo de puerto
Medidor de Temperatura	<i>Analógico</i>
Medidor de Humedad relativa	<i>Analógico</i>
Medidor de Presión barométrica	<i>Analógico</i>
Medidor de Intensidad de la luz solar	<i>Analógico</i>
Medidor de Velocidad del viento #1	<i>Contador Digital</i>
Medidor de Dirección del viento #1	<i>Analógico</i>
Medidor de Velocidad del viento #2	<i>Analógico</i>
Medidor de Dirección del viento #2	<i>Analógico</i>
Medidor de Precipitación pluvial	<i>Acumulador</i>

Tabla 1.1: Tipo de puertos usados por los instrumentos de medición en el sistema Solus

Para configurar un instrumento dentro del sistema Solus es necesario proporcionar un conjunto de parámetros, los cuales se describen en la tabla 1.2.

Parámetros	Descripción
Valor mínimo (Value <i>min</i>) y Valor máximo (Value <i>max</i>)	<i>Estos datos definen los límites del intervalo de salida del instrumento.</i>
Voltaje mínimo (Volt <i>min</i>) y Voltaje máximo (Volt <i>max</i>)	<i>Son los voltajes que definen los límites de los intervalos de salida del instrumento.</i>
Compensación (Offset)	<i>Este dato ajusta el nivel de voltaje continuo (DC) de la señal eléctrica enviada por el instrumento.</i>
Ganancia (Gain)	<i>Amplifica la amplitud de la señal eléctrica enviada por el instrumento. Los valores aceptados son: 1x, 2x, 5x ó 10x</i>

Tabla 1.2: Lista de parámetros para configurar un instrumento en el sistema Solus

2.1.4. Ecuaciones utilizadas por el sistema Solus

Las siguientes ecuaciones son usadas para determinar los valores obtenidos por los sensores de los instrumentos usando los parámetros definidos dentro del sistema Solus.

- **Voltaje de entrada (VI):** es el resultado del ajuste que hace el sistema Solus a la señal recibida en el puerto correspondiente. La ecuación 1.1 solo se aplica a los puertos de tipo analógico.

$$VI = \frac{\left[\left(\frac{10}{1023}\right) * NS\right]}{gain}$$

Ecuación 1.1: Cálculo del voltaje de entrada

Donde:

NS es el valor de la señal que envía el instrumento.

gain es la ganancia asignada al instrumento.

- **Valor de la variable meteorológica (SV):** proporciona el valor de la variable meteorológica del instrumento conectado al sistema Solus. Para los puertos de tipo analógico se ocupa la ecuación 1.2.

$$SV = \left[\left(\frac{Value_R}{Volt_R} \right) * Volt_{offset} \right] + Value_{min} + offset$$

Ecuación 1.2: Cálculo de la variable meteorológica de tipo analógico

Para los de tipo acumulador y contador digital se usa la ecuación 1.3:

$$SV = \frac{(NS * Value_{max})}{Volt_{max}}$$

Ecuación 1.3: Cálculo de la variable meteorológica de tipo digital

Donde:

Value_R = **Value_{max}** – **Value_{min}**

Volt_R = **Volt_{max}** – **Volt_{min}**

Volt_{offset} = **VI** – **Volt_{min}**

offset es la compensación asignada al instrumento

2.1.5. Comandos del sistema Solus

Los comandos del sistema Solus son un medio eficiente de comunicación con el sistema y la PC usando el puerto de comunicaciones serial (RS-232). Los comandos proveen una concisa forma de consultar y controlar la computadora del sistema Solus. El sistema soporta una gran cantidad de comandos, pero solo mencionaremos tres de los más utilizados en el sistema.

- **A: Comando Acumulador**

El acumulador interrumpe el microprocesador de la computadora Solus cada vez que se reconoce un cambio de estado en su entrada. El acumulador cuenta todos los pulsos que su entrada recibe y lo suma al valor almacenado. El retardo puede ser definido para detener el conteo de pulsos por un periodo de tiempo y después activar de nuevo el acumulador.

Sintaxis:

Limpiar el acumulador: **AC**

Leer el acumulador: **ARdddd**

Donde:

dddd es un retardo en mili-segundos en notación hexadecimal.

- ***C: Lectura del Contador Digital***

Este comando activa por un periodo de tiempo el contador especificado. Durante ese tiempo el contador registra el número de pulsos de un flujo digital, tales como revoluciones por minuto (RPM) o millas por hora (MPH).

Sintaxis:

Cccddd

Donde:

cc el número de contador digital (01, 02, 03 ó 04)

ddd duración en milisegundos en notación hexadecimal

- ***N: Lectura del Canal Analógico***

Este comando devuelve el valor de la señal eléctrica que recibe directamente la entrada del canal analógico y aplica la ganancia especificada. Los valores se miden en términos de voltaje, corriente o resistencia.

Sintaxis:

Ncc0g

Donde:

cc el valor asignado al número interno del canal análogo

g el valor asignado a la ganancia especificada para el instrumento (0 para 10x, 1 para 5x, 2 para 2x y 3 para 1x)

2.2. INGENIERÍA DE SOFTWARE

Desde los años 40's las aplicaciones y usos de las computadoras han crecido de forma constante. Hoy en día el software juega un papel central en casi todos los aspectos de la vida cotidiana: gobierno, finanzas, educación, transporte, medicina, etc. De forma paralela, la complejidad de los sistemas ha crecido de forma dramática, y anualmente se gastan miles de millones de dólares en cuestiones de desarrollo de software.

A pesar de que los sistemas de software nos han permitido ser más eficientes y productivos existen serios problemas relacionados con el desarrollo de software. Estos problemas tienen que ver con el tiempo, costo y calidad de los sistemas que se crean. Los productos de software se ubican entre los sistemas más complejos realizados por el hombre y su naturaleza intangible complica más el problema. Las técnicas de desarrollo que funcionan de forma individual o para equipos pequeños no escalan adecuadamente para el desarrollo de sistemas complejos, la velocidad de cambio tecnológico es muy elevada y los clientes tienen nuevas expectativas. Todo ello pone presión que limita la habilidad de crear software de calidad en tiempos aceptables. Ante este panorama la ingeniería de software busca dar soluciones. [1]

Aunque no hay consenso, el origen del término “*Ingeniería del Software*” se atribuye a dos congresos organizados por la OTAN en 1967 y 1968. Ambas conferencias sirvieron para tratar la llamada “crisis del software”, concepto nacido de la sensación de que los grandes programas no llegaban nunca a poderse entregar, además de ser altamente ineficientes, tener un gran número de errores y llevar asociados unos costes impredecibles.

Una definición precisa aún no ha sido contemplada, sin embargo se pueden citar las enunciadas por algunos de los más prestigiosos autores:

1. Ingeniería de software es el estudio de los principios y metodologías para el desarrollo y mantenimiento de sistemas software. [2]
2. Ingeniería de software es la aplicación práctica del conocimiento científico al diseño y construcción de programas de computadora y a la documentación asociada requerida para desarrollar, operar y mantenerlos. Se conoce también como Desarrollo de Software o Producción de Software. [3]
3. Ingeniería de software trata del establecimiento de los principios y métodos de la ingeniería a fin de obtener software de modo rentable, que sea fiable y trabaje en máquinas reales. [4]
4. Ingeniería de software es la aplicación de un enfoque sistemático, disciplinado y cuantificable al desarrollo, operación y mantenimiento del software; es decir, la aplicación de la ingeniería al software. [5]
5. Ingeniería de software como la disciplina preocupada por la aplicación de la teoría, conocimiento y práctica para la construcción eficiente y efectiva de sistemas software que satisfagan los requisitos de usuarios y clientes. [6]

Por otra parte, se indica que “el trabajo del ingeniero del software consiste en construir productos de alta calidad, con un determinado presupuesto y antes de una determinada fecha”. Por todo ello, se podría establecer como una definición apropiada de Ingeniería de software:

“Conjunto de técnicas, metodologías y herramientas que ayudan a la producción de un software de alta calidad, con un determinado presupuesto y antes de una determinada fecha”

Entendiendo por:

- **Técnica (Método):** Procedimiento formal para obtener resultados utilizando alguna notación bien especificada.
- **Metodología:** Colección de métodos aplicados a lo largo del ciclo de vida del software y unificados mediante alguna aproximación filosófica genérica.
- **Herramienta:** Instrumento, o sistema automatizado, utilizado para poner en práctica un método.
- **Software de calidad:** Aquel software que sirve para lo que fue concebido, es decir, que hace lo que se acordó con el cliente. Conviene aclarar, además, que el término calidad no hace referencia a la excelencia del software. Por ejemplo, dados los riesgos y los recursos disponibles, se puede pedir el cumplimiento de determinados parámetros de servicio: fallos leves y recuperables. Sin embargo, en un proyecto crítico (por ejemplo, software aeroespacial) puede resultar imprescindible el que no haya fallos, dadas sus consecuencias. [7]

2.3. UP: Proceso Unificado de desarrollo de software

El Proceso Unificado de desarrollo de software o simplemente Proceso Unificado (*UP: Unified Process*), creado por *James Rumbaugh*, *Grady Booch* e *Ivar Jacobson*, es un proceso de realización o de evolución de software basado en UML y se caracteriza por estar dirigido por casos de uso, centrado en la arquitectura y por ser iterativo e incremental. Está constituido por un conjunto de directivas que permiten producir software a partir del documento de requisitos. Cada directiva define *quién* hace *qué* y en *qué momento*. Existe un refinamiento particular del UP llamado *Proceso Unificado de Rational (RUP: Rational Unified Process)* y es un producto comercial suministrado a los clientes que han comprado una licencia a *IBM Rational Software*. Los principios de RUP proceden del *Proceso Unificado*. La diferencia entre RUP y el UP es el gran número de modelos disponibles que ofrecen un ahorro de tiempo al usar RUP. Además, RUP puede ampliarse para adaptarlo a necesidades específicas gracias a una serie de agregados (*plugins*) disponibles en el mercado. [9][E17]

2.3.1. Características esenciales

Su meta principal es asegurar la producción de software de alta calidad que cumpla con las necesidades de los usuarios, con una planeación y presupuesto predecible.

El UP identifica las 6 mejores prácticas de la Ingeniería de Software, con las que define una forma efectiva de trabajar para los equipos de desarrollo de software: [8]

1. Desarrollo de software de forma iterativa
2. Gestión de los requerimientos
3. Utilizar arquitecturas basadas en componentes
4. Modelar el software visualmente
5. Verificar la calidad del software
6. Controlar los cambios del software

El proceso de software propuesto por UP tiene tres características esenciales [9] [17]:

1. Proceso dirigido por los casos de uso

En UP los casos de uso son una herramienta que sirve para especificar los requisitos del sistema y guían su diseño, implementación y prueba. Los casos de uso constituyen un elemento integrador y una guía.

Los casos de uso inician el proceso de desarrollo y proporcionan un hilo conductor, permitiendo establecer un vínculo entre los artefactos que son generados en las diferentes actividades del proceso de desarrollo.

Basándose en los casos de uso se crean los modelos de análisis y diseño, luego la implementación que los lleva a cabo, y se verifica que efectivamente el producto implemente adecuadamente cada caso de uso. Todos los modelos deben estar sincronizados con el modelo de casos de uso.

2. Proceso centrado en la arquitectura

La arquitectura de un sistema es la organización o estructura de sus partes más relevantes, lo que permite tener una visión común entre todos los involucrados (desarrolladores y usuarios) y una perspectiva clara del sistema completo, necesaria para controlar el desarrollo.

RUP establece refinamientos sucesivos de una arquitectura ejecutable, construida como un prototipo evolutivo. En el caso de RUP además de utilizar los Casos de Uso para guiar el proceso se presta especial atención al establecimiento temprano de una buena arquitectura que no se vea fuertemente impactada ante cambios posteriores durante la construcción y el mantenimiento.

3. Proceso iterativo e incremental

El ciclo de vida iterativo se basa en la evolución de prototipos ejecutables que se muestran a los usuarios y clientes.

En el ciclo de vida iterativo a cada iteración se reproduce el ciclo de vida en cascada a menor escala.

Los objetivos de una iteración se establecen en función de la evaluación de las iteraciones precedentes.

Las actividades se encadenan en una mini-cascada con un alcance limitado por los objetivos de la iteración. Cada iteración comprende:

- Planificar la iteración (estudio de riesgos).

- Análisis de los casos de uso y escenarios.
- Diseño de opciones arquitectónicas.
- Codificación y pruebas. La integración del nuevo código con el existente de iteraciones anteriores se hace gradualmente durante la construcción.
- Evaluación de la entrega ejecutable (evaluación del prototipo en función de las pruebas y de los criterios definidos).
- Preparación de la entrega (documentación e instalación del prototipo)

2.3.2. Ciclo de vida

Basándose en la ilustración 1.2, se pueden distinguir las dos dimensiones del UP. El eje horizontal representa el tiempo y es considerado el eje de los aspectos dinámicos del proceso. El eje vertical representa el aspecto estático del proceso.

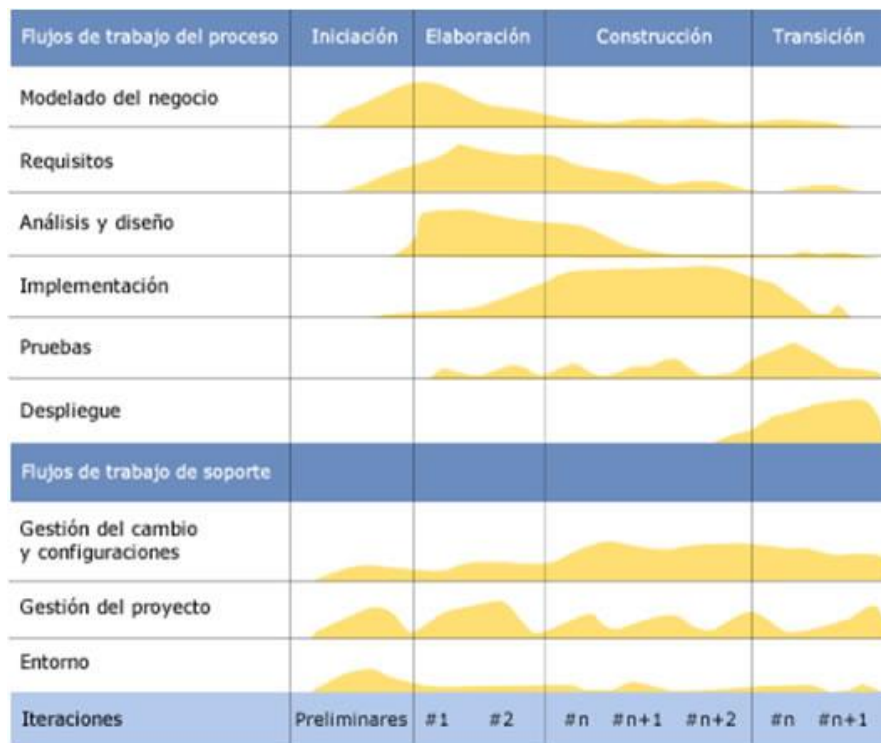


Ilustración 1.2: Ciclo de vida del UP

2.3.2.1. Aspecto dinámico del proceso

El UP se repite a lo largo de una serie de ciclos que constituyen la vida de un producto. El UP divide el proceso de desarrollo en ciclos, teniendo un producto al final de cada ciclo. En la ilustración 1.3 se puede apreciar cada ciclo que consta de cuatro fases: *Inicio*, *Elaboración*, *Construcción* y *Transición*. Cada fase concluye con un hito bien definido donde deben tomarse ciertas decisiones.

1- Fase de Inicio

Se establece la viabilidad del sistema mediante la definición de las funciones principales, la arquitectura inicial, el plan del proyecto, la estimación de costos del sistema y la identificación de los riesgos críticos que pudieran amenazar el planteamiento del proyecto.

2- Fase de Elaboración

Comprende la especificación de los casos de uso del producto y la arquitectura del sistema. Además, es necesario analizar los riesgos que pudieran afectar el sistema.

3- Fase de Construcción

Abarca la creación del producto, el desarrollo del software y su integración a la arquitectura diseñada. El objetivo principal es que el producto vaya alcanzando la funcionalidad planteada inicialmente para los usuarios, lo cual se logra mediante iteraciones que incrementen el producto en cada entrega.

4- Fase de Transición

Durante esta fase, el producto se convierte en una versión de prueba donde un número específico de usuarios hace una evaluación del producto de donde es posible que surjan problemas no identificados inicialmente y se puedan corregir.

[33]

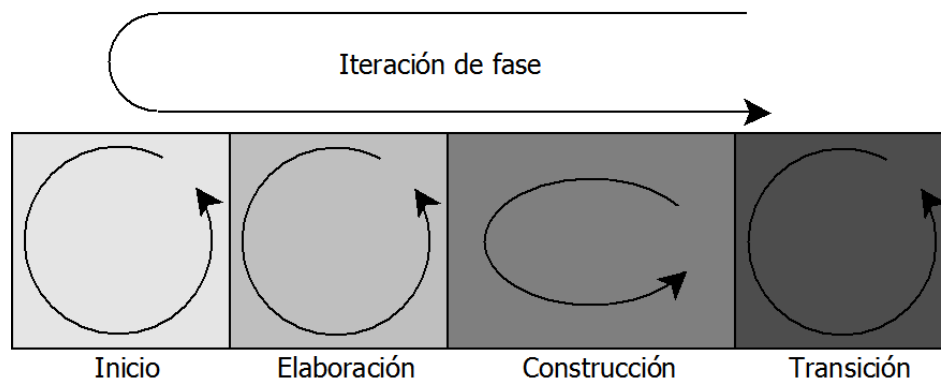


Ilustración 1.3: Fases del UP

2.3.2.2. Aspecto estático del proceso (Disciplinas de UP)

El aspecto estático del UP se centra en las actividades que tienen lugar durante el proceso de desarrollo. Éstas se denominan flujos de trabajo en la descripción del UP. Existen seis principales flujos de trabajo del proceso identificados en el proceso y tres principales flujos de trabajo de soporte.

1- Modelado de negocios

Entiende los problemas e identifica mejoras potenciales, asegura que los participantes en este modelo tengan el entendimiento del problema, deriva los requerimientos del software.

2- Requerimientos

Mantiene a los interesados sobre lo que el proyecto debe realizar, define los límites y requerimientos, se enfoca en las necesidades del usuario y hace una base de costos.

3- Análisis y diseño

Transforma los requerimientos al diseño y su arquitectura robusta y lo adapta para corresponder al ambiente de implementación y ajustarla para un desempeño esperado.

4- Implementación

Define el código, convierte el diseño en archivos ejecutables, prueba los componentes desarrollados como unidades, integra esas unidades en un sistema ejecutable.

5- Pruebas

Se enfoca en la evaluación de la calidad del producto, encuentra las fallas y las documenta, valida los requerimientos planteados y el buen funcionamiento.

6- Transición

Describe las actividades entre el aseguramiento de la entrega y disponibilidad del producto hacia el usuario final, hay un énfasis entre probar el software en el sitio de desarrollo.

7- Administración y configuración del cambio

Consiste en controlar los cambios y mantiene la integridad de los productos que incluye el proyecto.

8- Administración de proyectos

Provee un marco de trabajo para administrar los proyectos, guías para la planeación, soporte y ejecución, un marco de trabajo para administrar los riesgos.

9- Ambiente

Se enfoca en las actividades para configurar el proceso del proyecto, describe las actividades requeridas para apoyar el proyecto, su propósito para proveer a las organizaciones de desarrollo de software del ambiente necesario. [8]

2.4. UML: LENGUAJE UNIFICADO DE MODELADO

El UML (*Unified Modeling Language*) es un lenguaje de modelado estandarizado de propósito general y en la actualidad es administrado por la OMG (Object Management Group) [E9]

2.4.1. Historia

A mediados de 1990, James Rumbaugh y Grady Booch; empleados de la Rational Software Corporation publicaron la versión 0.8 bajo el nombre de Método Unificado (Unified

Method). Ivar Jacobson se unió al equipo y se publicó, en 1996, la versión 0.9 del Método Unificado. Con la unión de otras compañías se formó el Consorcio UML (UML Consortium) y se publicó la versión 1.0 bajo el nombre de UML y fue entregada a la OMG. Las versiones subsecuentes del UML fueron desarrolladas y publicadas por la OMG. En el año 2004 fue publicada la versión 2.0 del UML. La ilustración 1.4 se muestra la línea cronológica del UML. [11]

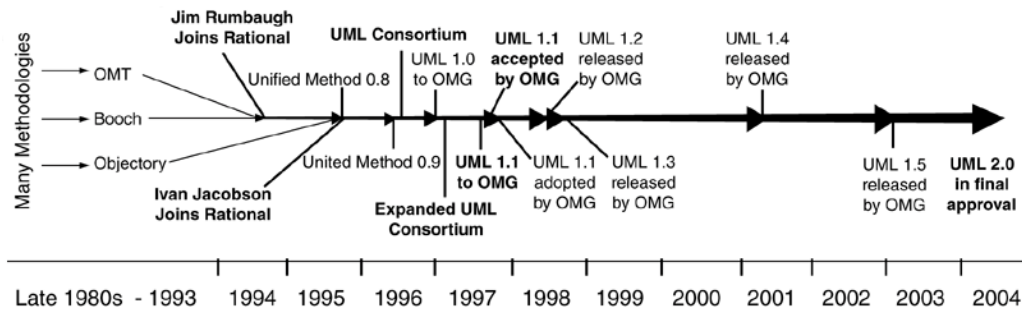


Ilustración 1.4: Línea cronológica del UML

2.4.2. Definición

El UML es usado para especificar, visualizar, modificar, construir y documentar un sistema. Ofrece una manera estándar para visualizar la arquitectura de un sistema e incluye un conjunto de elementos tales como actividades, actores, procesos de negocio, esquemas de bases de datos, componentes, declaraciones de lenguajes de programación y componentes de software re-utilizables. [11]

El UML combina técnicas del modelado de datos (diagramas de entidad relación), modelado de negocios (flujos de trabajo), modelado de objetos y modelado de componentes. Estos pueden ser usados en todos los procesos, en todo el ciclo de vida del desarrollo de software y a través de diferentes tecnologías de información. [12]

2.4.3. Modelos en el UML

Es importante distinguir entre un modelo UML y un conjunto de diagramas del sistema. Un diagrama es una representación parcial del modelo del sistema. El modelo contiene documentación que maneja los elementos del modelo y diagramas. [13]

Los diagramas UML representan dos diferentes vistas de un modelo del sistema.

- **Estático (o estructural):** enfatiza la estructura estática del sistema con el uso de objetos, atributos, operaciones y relaciones. La vista estructural incluye diagramas de clases y diagramas de estructura compuesta.
- **Dinámicos (o de comportamiento):** enfatiza el comportamiento dinámico del sistema mostrando las colaboraciones entre los objetos y los cambios internos de los objetos. Esta vista incluye diagramas de secuencia, diagramas de actividades y diagramas de máquinas de estado.

Dicha división no significa dividir los modelos en dos paquetes para acto seguido crear elementos y diagramas en cada uno de estos; muy por el contrario, la presentación de la parte dinámica y estática se hace para cada modelo del sistema es decir, ya sea que se esté creando un modelo de requisitos, uno de análisis o presentando el modelo físico, en todos los casos se tendrán diagramas que visualicen tanto el comportamiento dinámico como la estructura estática del sistema. Es más habitual presentar los sistemas como una colección de puntos de vista, y para cada uno de estos, mezclar diagramas dinámicos y estáticos.

2.4.4. Los diagramas en el UML

En la ilustración 1.5 se muestra la organización de los diagramas en el UML. Los primeros seis representan la información estructural. Los otros siete representan tipos generales de comportamiento, incluyendo cuatro que representan diferentes aspectos de interacción.

En la actualidad el UML está dividido en 2 categorías. [14]

- **Diagramas de estructura**

Estos diagramas enfatizan las cosas que deben estar presentes en el sistema que está siendo modelado. Al representar una estructura, se utilizan ampliamente en la documentación de la arquitectura del sistema.

- **Diagramas de comportamiento**

Estos diagramas enfatizan en lo que debe de suceder en el sistema modelado y son usados ampliamente para describir las funcionalidades que ofrece el sistema desarrollado.

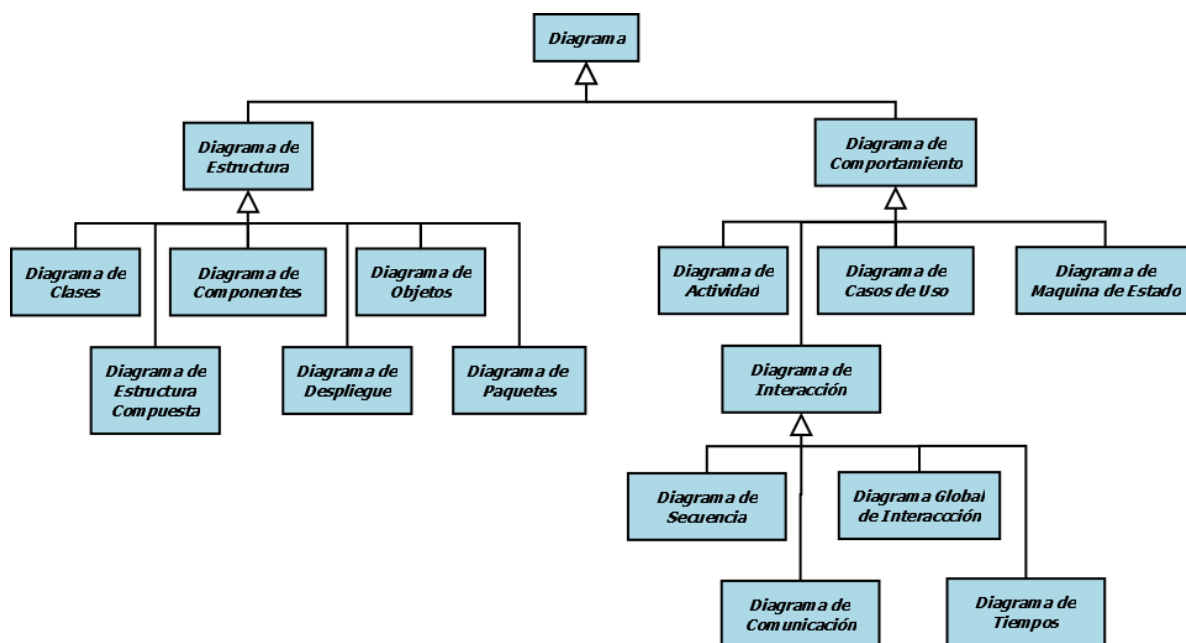


Ilustración 1.5: Organigrama de los diagramas en el UML

2.5. MODELOS DE PROCESO DE SOFTWARE

Un proceso define *quién* hace *qué*, *cuándo* y *cómo* para alcanzar cierto objetivo. En el caso del desarrollo de software, se requieren procesos especializados que abarquen desde la creación hasta la administración de un sistema de software. Para administrar la complejidad de tales sistemas, es necesario contar con modelos de proceso y tecnologías de software apropiadas.

Un modelo de proceso de software define como solucionar la problemática del desarrollo de sistemas de software. Para desarrollar el software se requiere resolver ciertas fases de su proceso, las cuales se conocen como el *ciclo de vida del desarrollo del software*. Un modelo de proceso debe de considerar una variedad de aspectos como el conjunto de personas, estructuras organizacionales, reglas, políticas, componentes de software, metodologías y herramientas utilizadas.

Los modelos de proceso de software dependen de las opiniones o creencias de las personas involucradas en un proyecto. Por ejemplo, algunas de estas creencias son:

- Es necesario comprender el problema antes de desarrollar la solución.
- El proceso para resolver el problema debe de dar un resultado predecible, sin importar qué individuo hace el trabajo.
- Es indispensable planear y calcular el proceso con gran precisión.
- Para que el proceso tenga éxito, es importante evaluar y administrar el riesgo.
- La entrega de etapas intermedia bien definidas aumentan la confianza que se tiene en el resultado finalidad [15]

2.5.1. Modelo lineal o de cascada

Este modelo indica que el producto de cada una de las fases es la entrada de la siguiente fase. Tiene una evolución lineal y acumulativa.

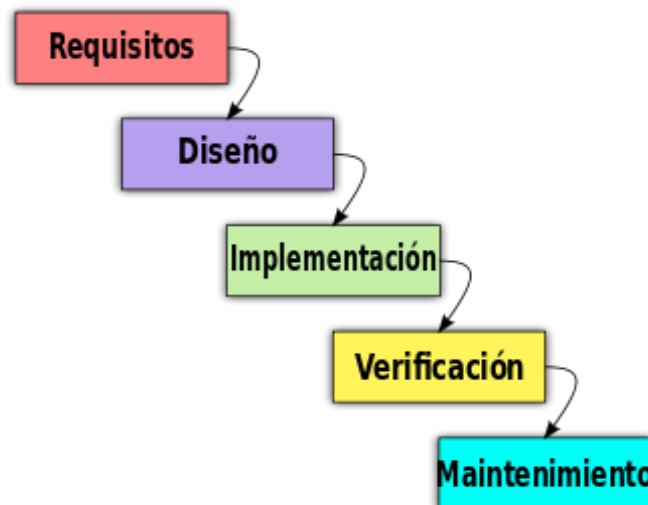


Ilustración 1.6: Etapas del modelo lineal o de cascada

En la ilustración 1.6 se esquematiza el modelo, el cual consta de las siguientes etapas:

Requisitos: Los servicios, restricciones y objetivos son establecidos con los usuarios para los que va destinado el sistema. Se busca hacer esta definición lo más detallado posible.

Diseño: Se divide el sistema en sistemas de software o hardware. Se establece la arquitectura total del sistema. Se identifican, se describen las abstracciones y relaciones de los componentes del sistema.

Implementación: Construcción de los módulos y unidades de software. Se realizan pruebas de cada unidad.

Verificación: Se integran todas las unidades. Se prueban en conjunto. Se entrega el conjunto probado al cliente.

Mantenimiento: Generalmente es la fase más larga. El sistema es puesto en marcha y se realiza la corrección de errores descubiertos. Se realizan mejoras de implementación. Se identifican nuevos requisitos.

Este modelo es el más tradicional, sin embargo, hoy en día tiende ser obsoleto por las siguientes razones:

- a) Casi ningún proyecto de la vida real sigue el patrón lineal.
- b) Si un error no es detectado en el inicio será muy costoso, pues solo al final del proyecto será evidente el error.
- c) Este modelo se basa en las tecnologías de antes, las cuales forzaban una especialización en el campo del análisis, del diseño, etc.

Sin embargo, para proyectos poco complejos que siguen este patrón de desarrollo lineal, puede ser muy útil y adoptarse sin problema. [16]

2.5.2. Modelo en espiral

Este modelo busca pasar de manera controlada y cíclica, por cuatro actividades en el desarrollo del software. En la ilustración 1.7 se señalan estas actividades.

- a) **Planificación:** se determinan los objetivos, alternativas de solución y restricciones que se dan para el proyecto. Se debe de encontrar el *qué* del software a desarrollar, así como aspectos de administración de proyectos y de los recursos necesarios por utilizar.
- b) **Análisis de riesgos:** identificación y resolución de riesgos que pueda tener el proyecto. Los riesgos pueden ser técnicos, administrativos, humanos, políticos, de seguridad, etc.
- c) **Ingeniería:** es la construcción del software. Aquí se combina el *cómo* (aspectos de diseño) y parte de la implementación (programación, pruebas, puesta en producción)
- d) **Evaluación del cliente:** aquí se cumple otra parte de la implementación (pruebas del cliente). Es importante esta actividad debido al control de calidad y a la participación activa de los clientes en la construcción del software.



Ilustración 1.7: Etapas del modelo en espiral

Este modelo ha sido muy utilizado en proyectos en los cuales la experiencia en el desarrollo del software y la tecnología resulta muy nueva para el equipo de trabajo u organización.

Este paradigma es el más “realista” para la producción de software, sobre todo en proyectos de complejidad considerable. Sin embargo, pueden notarse algunos problemas:

- Se requiere una gran habilidad para la identificación de riesgos. Si un riesgo no se detecta, el conjunto del proyecto puede verse afectado.
- Al utilizar un proceso de software evolutivo, tal vez sea difícil a los clientes visualizar el final del proyecto.

Al brindar actividades bien definidas, este modelo puede adecuarse bien, por su enfoque evolutivo, a proyectos complejos o donde se tenga poca experiencia. [16]

2.5.3. Modelo de prototipos

Los prototipos nacieron como un método para acelerar la definición de requisitos del software a construir. Un prototipo es un programa. La idea principal es hacer un modelo de la aplicación y presentársela al cliente, sobre todo a nivel de interfaces y otras salidas (consultas, reportes). El cliente hará sus observaciones acerca de lo que ve en ese modelo y el programador modificará ese modelo de acuerdo a dichas observaciones. El proceso se repite hasta alcanzar cubrir todos los requerimientos del producto en un software, pasando luego a construir la aplicación (muy probablemente, aprovechando la programación efectuada en el prototipo). En la ilustración 1.8 se esquematiza cada una de las etapas con las que consta el modelo de prototipos.

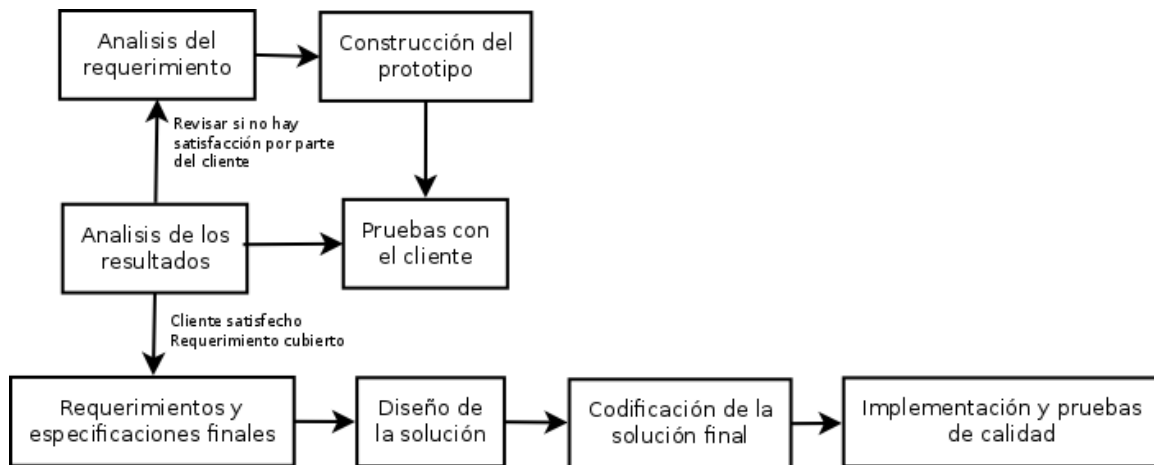


Ilustración 1.8: Etapas del modelo de prototipos

Los problemas detectados en este modelo son:

- a) El prototipo es un software en “borrador” no un producto de ingeniería como tal. Sin embargo, puede causar falsas expectativas al cliente en el sentido de que no es un producto consolidado. En el momento en que los ingenieros tardan en hacer el software apropiado, podría darse un efecto de ansiedad en el cliente y repercutir en el proyecto.
- b) De no establecerse con claridad los alcances del proyecto, en cada revisión se podrían agregar más requisitos, el cual haría tender a un proyecto sin fin.

Para aplicaciones no muy complejas, bien definidas y claras en sus procesamientos, orientadas a presentar muchas salidas, este modelo podría funcionar muy bien. [16]

2.5.4. Modelo incremental

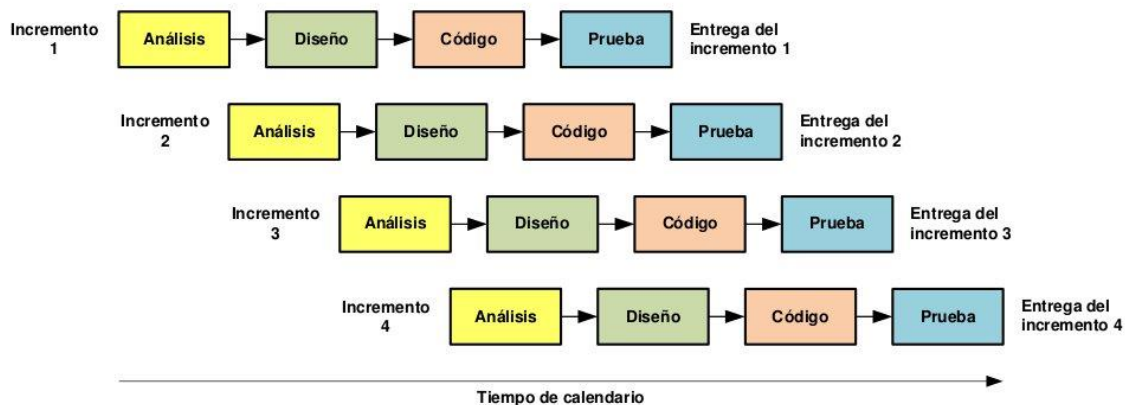


Ilustración 1.9: Etapas del modelo incremental

El principio del modelo incremental es el diseño de un subconjunto inicial de componentes o funcionalidades, a las que el equipo de desarrollo agrega más componentes o funcionalidades (el incremento) hasta que el desarrollo del producto se complete.

Este modelo, representado en la ilustración 1.9, tiene algunas fases que están presentes en el modelo lineal o de cascada. Sus etapas son análisis, diseño, codificación y prueba. **[18]**
Sus características son:

- Se realiza una serie de entregas al cliente. Estas entregas son llamadas incrementos. Más y más funcionalidades o componentes son integradas con cada incremento.
- El primer incremento es llamado producto núcleo. En esta entrega los requerimientos básicos son implementados y entonces en incrementos subsecuentes se suman nuevos requerimientos.

El uso de este modelo se sugiere:

1. Cuando los requerimientos están razonablemente bien definidos,
2. Cuando el alcance general del proyecto sugiere un esfuerzo puramente lineal.
3. Cuando un conjunto limitado de funcionalidades del software se requieran desarrollar con rapidez.

Sus ventajas o méritos de este modelo son:

1. Puede ser adoptado cuando el número de personas involucradas en el proyecto es pequeño.
2. Los riesgos técnicos pueden ser manejados en cada incremento.
3. En un lapso breve de tiempo, al menos, el producto núcleo puede ser entregado al cliente.

2.6. BASES DE DATOS

La aparición del concepto de *bases de datos* (*data base*) se produce a comienzos de la década de 1960, concretamente en el simposio que se celebró en *Santa Mónica, California EUA.*, y cuyo título era “*Development and Management of a Computer-Centered Data Base*”.

Una *base de datos* es un conjunto, colección o depósito de datos almacenados en un soporte informático de acceso directo. Los datos deben estar relacionados y estructurados de acuerdo con un modelo capaz de recoger el contenido semántico de los datos almacenados. Dada la importancia que tienen en el mundo real las relaciones entre los datos, es importante e imprescindible que la base de datos sea capaz de almacenar estas interrelaciones.

Una base de datos debe cumplir una serie de requisitos:

- No debe de existir redundancia lógica de datos, aunque es admisible cierta redundancia física por motivos de eficiencia.
- La base de datos han de dar soporte a múltiples usuarios y a diferentes aplicaciones de manera simultánea.
- En las bases de datos debe existir una independencia tanto física como lógica entre datos y su proceso.
- La definición y descripción del conjunto de datos contenidos en la base de datos deberán ser únicas y estar integradas con los mismos datos.
- La base de datos debe asegurar la integridad, seguridad y confidencialidad de sus datos cuando éstos se actualizan y recuperan.

El termino base de datos hace referencia a una colección específica de datos, aunque es habitual usarlo, de manera errónea, como sinónimo del software que gestiona dicha colección de datos. En la bibliografía, a este software se le conoce como Sistema gestor de bases de datos (SGBD) [19]

2.6.1. Sistema gestor de bases de datos

Un sistema gestor de bases de datos proporciona los mecanismos para almacenar y organizar datos de una manera consistente con el formato de la base de datos. Los SGBD permiten el acceso y almacenamiento de datos sin necesidad de preocuparse por la representación interna [20]

Los SGBD proporcionan un conjunto coordinado de programas, procedimientos y lenguajes que permiten a los distintos usuarios realizar tareas habituales con los datos, garantizando además la seguridad de los mismos.

El éxito de los SGBD reside en mantener la seguridad e integridad de los datos. Por lógica, tiene que brindar herramientas a distintos tipos de usuarios. Entre las herramientas que proporcionan están: [21]

- Herramientas para la creación y especificación de los datos, así como la estructura de la base de datos.
- Herramientas para administrar y crear la estructura física requerida en las unidades de almacenamiento.
- Herramientas para la manipulación de datos de las bases de datos; ya sea añadir, modificar, suprimir o consultar.
- Herramienta de recuperación en caso de desastre.
- Herramientas para la creación de copias de seguridad.
- Herramientas para la gestión de la comunicación de la base de datos.
- Herramientas para la creación de aplicaciones que utilicen esquemas externos de los datos.
- Herramientas para la instalación de la base de datos.
- Herramientas para la exportación e importación de datos.

2.6.2. Usuarios de las bases de datos

En los SGBD se pueden identificar cinco tipos de usuario que pueden realizar tanto funciones únicas como complementarias. [22]

1. Los *administradores de sistemas* supervisan las operaciones generales del sistema de bases de datos.
2. Los *administradores de bases de datos*, administran el SGBD y garantizan que las bases de datos estén funcionando apropiadamente.
3. Los *diseñadores de bases de datos*, son los que diseñan la estructura de ésta.
4. Los *analistas de sistemas y programadores*, diseñan y ejecutan los programas de aplicación, así como los filtros de entrada de datos, reportes y procedimientos mediante los cuales los usuarios finales tengan acceso para manipular los datos almacenados en las bases de datos.
5. Los *usuarios finales*, son las personas que usan los programas de aplicación para realizar las operaciones diarias de la organización. Los usuarios pueden clasificados en expertos, habituales u ocasionales.

2.7. MODELO DE BASES DE DATOS

El diseño de bases de datos se simplifica cuando se utilizan modelos. Los *modelos* son abstracciones simplificadas de eventos y condiciones del mundo real. Tales abstracciones permiten explorar las características de entidades y las relaciones que se pueden dar entre ellas.

Un *modelo de bases de datos* es un conjunto de ideas lógicas utilizadas para representar la estructura de datos y las relaciones entre ellos dentro de la base de datos. Estos modelos se pueden agrupar en dos categorías: [22]

1. **El modelo conceptual** se enfoca en la naturaleza lógica de la representación de datos. Por consiguiente, este modelo está comprometido con lo *que* está representado en la base de datos y no en *cómo* está representado. Los modelos conceptuales incluyen el modelo *Entidad-Relación* (E-R) y el modelo *Orientado a objetos*
2. **El modelo de ejecución** hace énfasis en *cómo* los datos están representados en la base de datos o en *cómo* se ejecutan las estructuras de datos para representar lo que está modelado. Los modelos de ejecución incluyen el modelo de *base de datos jerárquico*, el de *base de datos de red*, el modelo de *base de datos relacional*.

En la ilustración 1.10 se incluye un resumen de los modelos de datos más reconocidos.

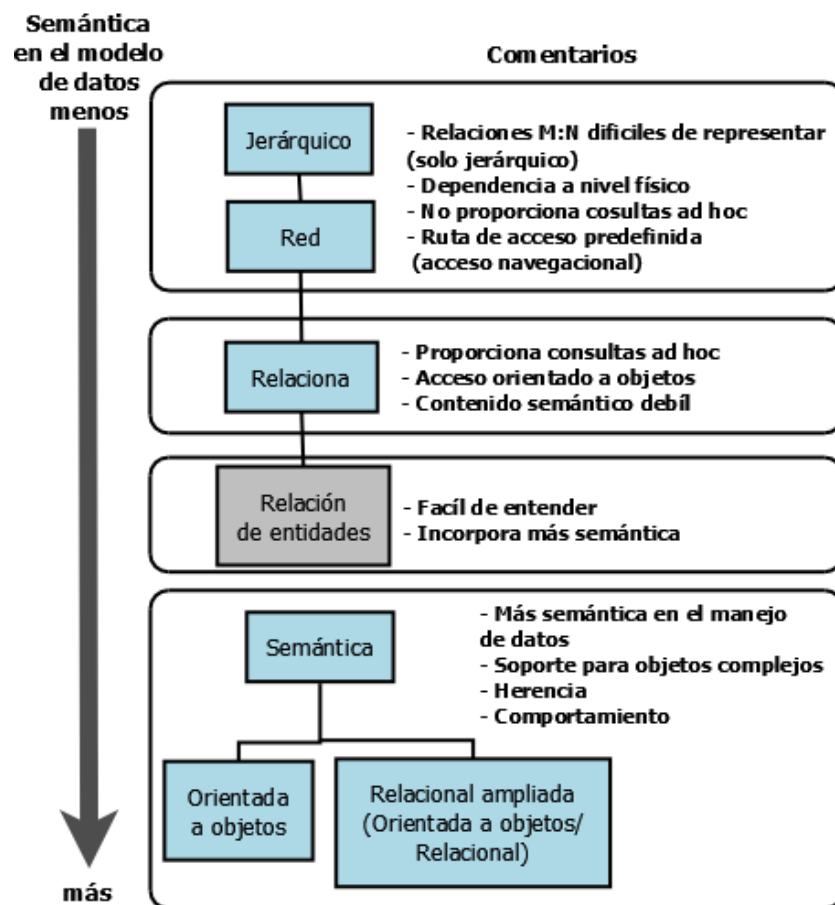


Ilustración 1.10: Desarrollo de modelos de datos [22]

Cuando se analizan los modelos de datos, se puede determinar algunas características que éstos deben de tener para que sean ampliamente aceptados:

1. Un modelo de datos debe mostrar un cierto grado de simplicidad conceptual sin que se comprometa el contenido semántico de la base de datos. No tiene sentido tener un modelo de datos que sea más difícil de conceptualizar que el mundo real.
2. Un modelo de datos debe de representar tan fiel como sea posible el mundo real. Este objetivo se alcanza más fácilmente mediante la adición de semántica a la

representación de datos del modelo (La semántica tiene que ver con el comportamiento dinámico de los datos, mientras que la representación de datos constituye el aspecto estático del escenario en el mundo real)

3. La representación de las transformaciones en el mundo real (comportamiento) debe de ajustarse a las características de consistencia e integridad de cualquier modelo de datos.

2.7.1. Modelo Entidad-Relación (E-R)

El modelo E-R es la herramienta más aceptada y adaptada de modelo de datos en el ambiente de producción de bases de datos. *Peter Chen* introdujo por primera vez el modelo E-R en 1976 en su artículo *"The entity-relationship model: Toward a unified view of data"*.

El modelo de relación de entidades produjo una representación gráfica de las entidades y sus relaciones en una estructura de base de datos. Es precisamente esta representación gráfica la que popularizó el uso de diagramas E-R como herramienta de modelado de datos a nivel conceptual. Es más, el modelo E-R complementó los conceptos del modelo de datos relacional, con lo que se establecieron las bases para un ambiente de diseño de bases de datos bien estructuradas que garantizan el diseño apropiado de bases de datos relacionales. En la ilustración 1.11 se muestran ejemplos de uso de los diagramas E-R.

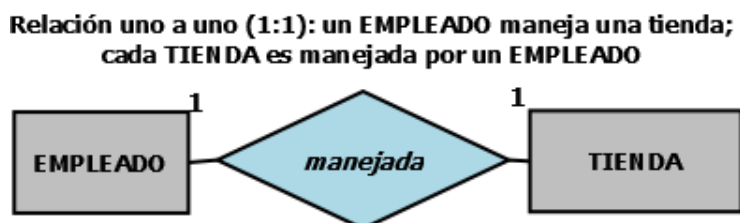
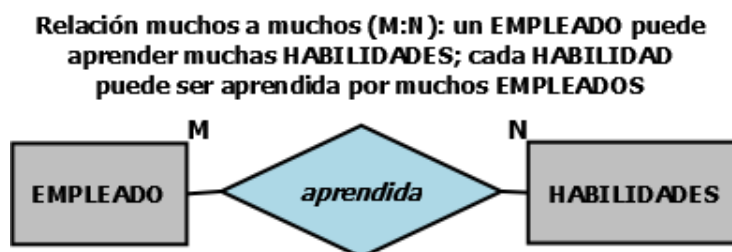
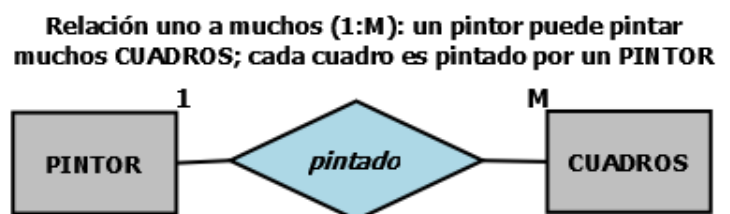


Ilustración 1.11: Diagrama E-R con los diferentes tipos de relación [22]

El modelo E-R básico se basa en los siguientes componentes: [22]

- **LA ENTIDAD** se define como una persona, lugar o cosa sobre la cual se tienen que reunir y guardar datos. Se representa por medio de un rectángulo. Su nombre, un sustantivo, se escribe en el centro del rectángulo. El nombre de la entidad se escribe en mayúsculas y en singular. Normalmente, cuando se aplica el diagrama E-R al modelo relacional, una entidad se correlaciona con una tabla relacional. Cada fila en la tabla relacional se conoce como *instancia de la entidad* u *ocurrencia de entidad* en el modelo E-R. Cada entidad se describe mediante un conjunto de atributos. Un atributo describe una característica en particular de la entidad.
- **UNA RELACIÓN** describe una asociación entre datos. La mayoría de las relaciones describen asociaciones entre dos entidades. Cuando se introdujeron los modelos de bases de datos, se ilustraron tres tipos de relaciones posibles entre los datos: *uno a muchos (1:M)*, *muchos a muchos (M:N)* y *uno a uno (1:1)*. En el modelado se usa el término **conectividad** para designar las clasificaciones de relación. Las conectividades se escriben junto a cada cuadro de entidad) Las relaciones se representan por medio de un rombo conectado a las entidades relacionadas. El nombre de la relación, un verbo activo o pasivo, se escribe adentro del rombo.

Otra versión bien establecida del diagrama Entidad-Relación, es el modelo de **pata de gallo**. La designación de *pata de gallo* se deriva del símbolo de tres puntas utilizado para representar el lado “muchos” de la relación. Unos ejemplos se muestran en la figura 1.12.

Relación uno a muchos (1:M): un pintor puede pintar muchos CUADROS; cada cuadro es pintado por un PINTOR



Relación muchos a muchos (M:N): un EMPLEADO puede aprender muchas HABILIDADES; cada HABILIDAD puede ser aprendida por muchos EMPLEADOS



Relación uno a uno (1:1): un EMPLEADO maneja una tienda; cada TIENDA es manejada por un EMPLEADO



Ilustración 1.12: Diagrama E-R usando el modelo "pata de gallo"[22]

2.7.2. Modelo Relacional

Los principios del modelo de datos relacional fueron establecidos por *E. F. Codd* en los años 1969 y 1970. Fue hasta la década de los 80 que se empezaron a comercializar los primeros SGBD relacionales con rendimientos aceptables.

El principal objetivo de éste modelo es facilitar que la base de datos sea percibida o vista por el usuario como una estructura lógica que consiste en un conjunto de relaciones y no como una estructura física de implementación. Esto ayuda a conseguir un alto grado de independencia de los datos.

Un objetivo adicional del modelo es conseguir que esta estructura lógica con la que se percibe la base de datos sea simple y uniforme. Para proporcionar simplicidad y uniformidad, toda la información se representa de una única manera: mediante valores explícitos que contienen las relaciones. Con el mismo propósito, todos los valores de datos son atómicos, es decir, no es posible descomponerlos. [23] En la ilustración 1.13 se muestra la estructura básica usada en el modelo relacional.

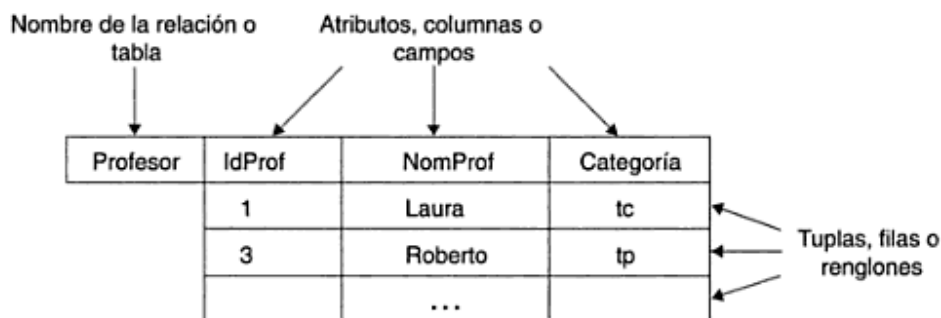


Ilustración 1.13: Estructura básica usada en el modelo relacional

Los rasgos fundamentales de este modelo son: [24]

- Los elementos estructurales son las relaciones, los atributos y las tuplas (tablas, columnas y filas). Las relaciones agrupan una serie de atributos que denotan el esquema (o intensión) de la relación. El conjunto de tuplas de cada relación constituye la extensión de la relación.
- Sobre estos elementos se define un conjunto de operaciones del álgebra relacional. Las principales operaciones son la *unión*, *intersección*, *diferencia*, *producto cartesiano*, *selección*, *proyección* y *combinación*. Las cuatro primeras pertenecen a la álgebra de conjuntos; el resto son propias del modelo.
- Se proporciona un lenguaje de alto nivel, basado en el cálculo de predicados, el lenguaje **SQL**. Este lenguaje incluye un subconjunto de sentencias que permiten crear y manipular las bases de datos relacionales.

El modelo relacional también proporciona mecanismos para definir reglas de integridad, ya sean inherentes al modelo como las propias de los usuarios.

- a) Cada relación tiene un atributo o conjunto de atributos, llamado *clave o llave primaria*.
- b) Los atributos no pueden tomar más de un valor en una tupla.

- c) El orden de la tuplas de la relación y el orden de los atributos dentro de la tupla son irrelevantes.
- d) La interrelación entre dos relaciones se lleva a cabo mediante *claves o llaves foráneas*.

2.7.3. Llaves primarias y foráneas

Las llaves (o claves) son campos especiales que juegan roles muy específicos dentro de una relación (tabla) y el tipo de llave determina su propósito dentro de la relación. Hay varios tipos de llaves que una relación puede contener, aunque las más significantes son la *llave primaria* y la *llave foránea*. Un ejemplo se muestra en la ilustración 1.14

Una llave primaria es un atributo (campo) o grupo de atributos que únicamente identifica un registro dentro de una relación; una llave primaria compuesta de dos o más atributos es conocida como llave primaria compuesta. [25]

- El valor de una llave primaria identifica una tupla en específico en toda la base de datos.
- El atributo de la llave primaria identifica a una relación dada en toda la base de datos.
- La llave primaria impone integridad a la relación y ayuda a establecer interrelaciones con otras relaciones.
- Cada relación en la base de datos debe tener una llave primaria y su valor no puede ser desconocido o nulo.

Cuando se decide que dos relaciones (R y T) deben tener interrelación, se establece a través de una llave foránea. La llave foránea es un atributo o conjunto de atributos en una relación (por ejemplo, T), de manera que los valores que toman para una tupla determinada coinciden con el valor de la clave primaria de una tupla en la otra relación (en este caso R). Esta igualdad de valores permite pasar de una relación a otra o viceversa. [24] Un ejemplo del uso de llaves primarias y foráneas se muestra en la ilustración 1.14.

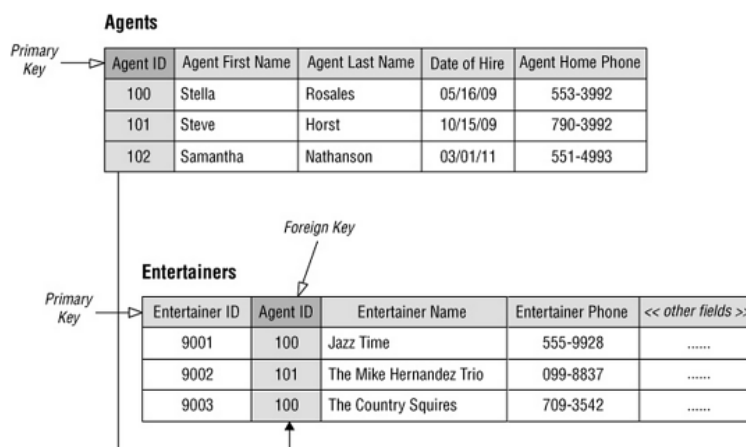


Ilustración 1.14: Ejemplo del uso de llaves primarias y foráneas dentro de una base de datos

2.8. NORMALIZACIÓN

La normalización se podría definir como el proceso de organizar y estructurar los datos de forma que se minimice la redundancia. Este proceso se basa en un conjunto de pautas, llamadas reglas de normalización, que disminuyen el riesgo de tener un diseño de bases de datos defectuoso.

Estas reglas se aplican al modelo relacional, obtenido a partir del modelo Entidad-Relación. Los objetivos que persigue el proceso de normalización son:

- Evitar la redundancia de datos.
- Proteger y dar un mejor soporte a la integridad de los datos.
- Eliminar anomalías en los datos, tanto en las actualizaciones como en las inserciones y las eliminaciones.
- Reducir, en la medida de lo posible, el re-diseño de la base de datos cuando ésta se amplía.
- Hacer más comprensible el modelo de datos a quienes vayan a utilizarlo, puesto que se modela mejor la realidad (el dominio del problema)

Hay que destacar que la normalización se puede hacer a nivel completo de la base de datos, o a nivel de tablas o esquemas. La técnica es la misma: analizar el conjunto de campos y en base a eso designar una clave inicial que identifique a un grupo de datos.

2.8.1. Formas normales

Edgar F. Codd, el padre del modelo relacional definió el concepto de normalización y las tres primeras formas normales (**1NF, 2NF y 3NF**). Las formas normales proporcionan el criterio necesario para determinar el grado de vulnerabilidad e inconsistencias lógicas y anomalías en los datos. Cuando más alto sea el número de la forma normal, menos vulnerable serán las tablas que conforman la base de datos.

2.8.1.1. Primera Forma Normal (1FN)

Se dice que una relación está en 1NF si y solo si cada uno de los atributos tiene un único valor para un registro determinado, es decir, los dominios de esta relación tienen valores únicos.

Si una relación no está en 1NF hay que eliminar de ella los atributos que tienen múltiples valores para cada tupla de la relación. Éstos se eliminan con meterlos en otra relación y hacer una interrelación mediante la clave primaria de la relación en la que se encontraban. [26]

En la ilustración 1.15 se tiene la siguiente relación:

Código	Nombre	Asignaturas
1	María	1.303
2	Eva	1.321,1402
3	Inés	2.701,3271

Ilustración 1.15: Relación de ejemplo no normalizada

Se nota que no cumple con la 1NF ya que los registros 2 y 3 de la relación tienen dominios con varios valores. La solución es separar estos valores mediante otra relación.

Código	Nombre
1	María
2	Eva
3	Inés

Código	Asignaturas
1	1.303
2	1.321
2	1.402
3	2.701
3	3.271

Ilustración 1.16: Relaciones en 1FN.

De esta forma tenemos las relaciones en 1NF, tal como lo muestra la ilustración 1.16.

2.8.1.2. Segunda Forma Normal (2FN)

Esta forma se basa en el concepto de dependencia funcional total o completa para todos y cada uno de los atributos de la relación con la clave primaria. Si todos los atributos dependen directamente de esta clave se dice que la relación está en 2FN. La 2FN se aplica a aquellas relaciones que tienen claves primarias compuestas por dos o más atributos.

Si una relación no está en 2FN hay que eliminar las dependencias parciales de la clave primaria. Para ello se eliminan los atributos que son funcionalmente dependientes y se colocan en una nueva relación con una copia de su determinante (los atributos de la clave primaria de los que dependen) [26]

En la ilustración 1.17 se tiene la siguiente relación:

Cod_Prof	Cod_asig	Nombre	Asignatura	Créditos
1	1.303	Juan Pérez	Fundamentos Informática	3
2	1.321	Alberto Ruiz	Criptografía	2
3	1.402	Carlos Vilas	Modelado	3
4	1.321	Ana Paz	Criptografía	2
2	1.303	Alberto Ruiz	Fundamentos Informática	3

Ilustración 1.17: Relación de ejemplo para normalizar a 2NF

La clave primaria está formada por el atributo **Cod_Prof** y **Cod_asig**. Podemos decir que está en 1NF pero vamos a evaluar si está en 2NF:

- El atributo **Nombre** no depende de **Cod_asig**
- El atributo **Asignatura** no depende de **Cod_Prof**
- El atributo **Créditos** si depende de **Cod_Prof** y **Cod_asig**

Por lo tanto, al no depender todos los atributos de la totalidad de la clave primaria se dice que la relación no está en 2NF. La ilustración 1.18 muestra la solución.

Cod_asig	Asignatura
1.303	Fundamentos Informática
1.321	Criptografía
1.402	Modelado

Cod_Prof	Nombre
1	Juan Pérez
2	Alberto Ruiz
3	Carlos Vilas
4	Ana Paz

Cod_Prof	Cod_asig	Créditos
1	1.303	3
2	1.321	2
3	1.402	3
4	1.321	2
2	1.303	3

Ilustración 1.18: Relaciones en 2NF

Podemos observar que ahora las tres relaciones se encuentran 2NF.

2.8.2. Tercera Forma Normal (3FN)

Esta forma normal se basa en el concepto de dependencia transitiva (ocurre cuando un atributo es funcionalmente dependiente de otro y ninguno de las dos formas parte de la clave) y se dice que una relación están en 3NF si y solo si los atributos de la relación

dependen únicamente de la clave primaria, es decir, los atributos de la tabla no dependen unos de otros.

Para pasar una relación 2NF a 3NF deben de suprimir las posibles dependencias transitivas, eliminando los atributos que dependen transitivamente y se colocan en una nueva relación con una copia del atributo o atributos no clave de los que dependen. [26]

En la ilustración 1.19 tenemos la siguiente relación:

Código	Nombre	Asignatura	Aula
1	María	1.303	10
2	Eva	1.321	5
3	Inés	2.701	8

Ilustración 1.19: Relación de ejemplo en 2FN

Se procede a estudiar la dependencia de cada atributo con el campo clave **Código**:

- **Nombre** depende directamente de **Código**
- **Asignatura** depende de igual modo.
- **Aula**, aunque es parte depende de alumno, está más ligada a **Asignatura** que **Nombre**

Por esta última razón se dice que la relación no está en 3NF, la solución se muestra en la ilustración 1.20.

Asignatura	Aula	Código	Nombre	Asignatura
1.303	10	1	María	1.303
1.321	5	2	Eva	1.321
2.701	8	3	Inés	2.701

Ilustración 1.20: Relaciones en 3FN

Se puede seguir profundizando en el proceso de normalización de las relaciones de la base de datos (Forma normal de Boyce-Codd, 4NF, 5NF, Forma normal de Dominio/Clave y 6NF), a la hora de realizar este proceso se tiene que llegar a un compromiso entre el gasto de recursos, de tiempo, la complejidad del diseño y los problemas que se desean evitar en la base de datos.

2.9. ARQUITECTURA CLIENTE-SERVIDOR

La arquitectura cliente-servidor es una arquitectura de red en la que cada computadora o proceso en la red es cliente o servidor (aunque una misma computadora puede ser cliente y servidor de manera simultánea). Los servidores son computadoras potentes que están dedicados a gestionar unidades de almacenamiento, sistemas de impresión, tráfico de red (servidor de red), datos y aplicaciones. Mientras que los clientes son computadoras menos potentes (no necesariamente) y usan los recursos que ofrecen los servidores. [27]

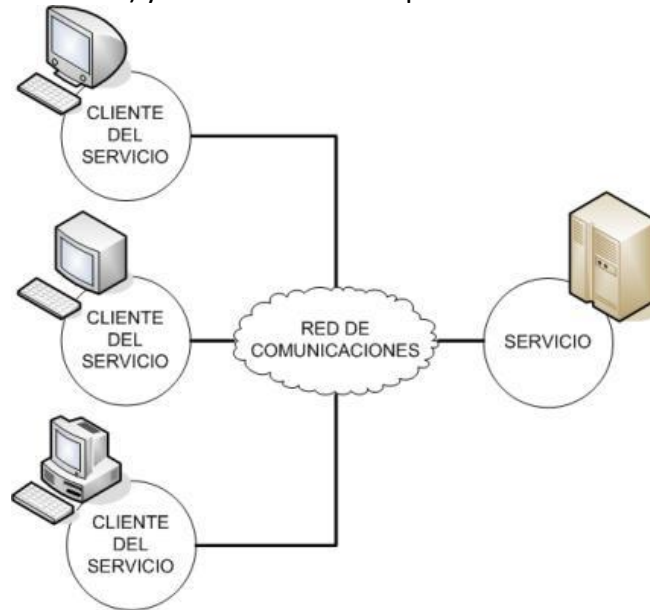


Ilustración 1.21: Esquema de la arquitectura Cliente-Servidor

Basándose en la ilustración 1.21, los clientes interaccionan con el usuario. Frecuentemente se comunican con procesos auxiliares que se encargan de establecer conexión con el servidor, enviar el pedido, recibir la respuesta, manejar las fallas y realizar actividades de sincronización y de seguridad.

Los servidores proporcionan un servicio al cliente y devuelven los resultados. En algunos casos existen procesos auxiliares que se encargan de recibir las solicitudes del cliente, verificar la protección, activar un proceso servidor para satisfacer el pedido, recibir su respuesta y enviarla al cliente.

Para que los clientes y los servidores puedan comunicarse se requiere una infraestructura de comunicaciones, la cual proporciona los mecanismos básicos de direccionamiento y transporte. La mayoría de los sistemas cliente - servidor actuales se basan en redes locales que cuentan con características adecuadas de desempeño, confiabilidad, transparencia y administración.

Uno de los aspectos que más ha promovido el uso de sistemas cliente - servidor es la existencia de plataformas de hardware cada vez más baratas. Esta constituye a su vez una de las más palpables ventajas de este esquema, la posibilidad de utilizar máquinas considerablemente más baratas que las requeridas por una solución centralizada, basada en sistemas grandes. Además de lo anterior, se pueden utilizar componentes, tanto de

hardware como de software, de varios fabricantes, lo cual contribuye considerablemente a la reducción de costos y favorece la flexibilidad en la implantación y actualización de soluciones.

2.9.1. Código abierto (Open source)

Este término es utilizado comúnmente para referirse al *software libre*. Sin embargo, sus criterios son algo menos determinantes. El código abierto acepta algunas restricciones que el software libre ha rechazado.

El significado de código abierto se refiere a tener acceso al código fuente, aunque pueden aplicarse algunas licencias restrictivas sobre ese código.

El término se comenzó a utilizar en 1998 y es promovido por la Iniciativa de Código Abierto (OSI: Open Source Initiative) y rápidamente fue asociado a una filosofía, a unos valores e incluso a unos criterios [28]

Según la Iniciativa de Código Abierto, el código abierto no sólo es acceso al código fuente; sino que debe cumplir los siguientes criterios: [E10]

1. Debe permitirse la libre redistribución del software.
2. El código fuente debe estar siempre disponible.
3. Debe permitirse la modificación del software y la creación de programas derivados.
4. Debe garantizarse la integridad del programa original. Esto puede hacerse exigiendo que la distribución de cualquier modificación se haga de forma separada, o que cualquier modificación o programa derivado sea distribuido con un nombre o versión diferente.
5. No se debe discriminar a ninguna persona o grupo de personas.
6. Debe permitirse el uso del software para cualquier fin.
7. La licencia debe ser distribuida junto con el software. La licencia debe aplicarse por igual a todos los que utilizan el programa.
8. La licencia deberá ser siempre la misma, sin importar si el software es incluido dentro de una distribución o paquete específico.
9. La licencia no debe aplicar restricciones sobre otros programas.
10. La licencia debe ser tecnológicamente neutral.

2.9.2. Servidor Apache Tomcat

Apache Tomcat es un software de código abierto, desarrollado por la Apache Software Foundation, y sirve para la implementación de tecnologías relacionadas con los Java Servlets y las páginas JSP (JavaServer Pages), éstas a su vez se especifican y desarrollan por la Java Community Process. [E11]

El objetivo es proporcionar soluciones de servidores con calidad comercial, basados en la plataforma Java, que se desarrollen en forma abierta y cooperativa. A la parte relacionada con el servidor web se conoce como Apache Tomcat y al contenedor de servlets se le llama Catalina, ambos componentes vienen incluidos en el mismo paquete de instalación.

Tomcat funciona en cualquier sistema operativo que soporte Java. La instalación y administración se realiza modificando directamente los archivos de configuración organizados en una estructura de directorios bien definidos, esto permite que la instalación y ejecución de los programas sean fáciles de realizar.

2.9.3. La tecnología Java

La tecnología Java consiste de un lenguaje y de una plataforma. [E12]. En la ilustración 1.24 se muestra el proceso de desarrollo de los programas en este lenguaje y consiste en escribir el código fuente en texto plano y guardarlo en un archivo con extensión *.java*. Estos archivos son entonces compilados en archivos *.class* por el compilador *javac*. Un archivo *.class* no contiene código que algún procesador pueda entender, solo lo entiende la Máquina Virtual Java (Java VM). Una herramienta llamada *java launcher* carga la aplicación dentro de la máquina virtual Java para ejecutar nuestra aplicación.

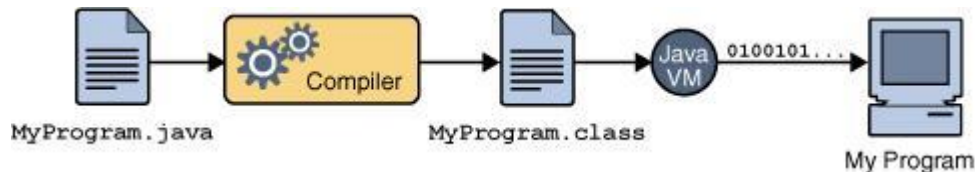


Ilustración 1.22: Una visión general del proceso de desarrollo de software en Java

Una plataforma es el hardware o software de entorno en el que un programa se ejecuta. La mayoría de las plataformas se puede describir como una combinación del sistema operativo y el hardware subyacente. La plataforma Java difiere de la mayoría de las otras plataformas, pues se trata únicamente de una plataforma de software que se ejecuta sobre otras plataformas basadas en hardware.

La plataforma Java tiene dos componentes:

- La Máquina virtual de Java (Java VM)
- La interfaz de programación de aplicaciones Java (API Java)

La Java VM es una de las piezas fundamentales de la plataforma Java. Básicamente se sitúa en un nivel superior al hardware del sistema sobre el que se pretende ejecutar la aplicación, y este actúa como un puente que entiende tanto el *bytecode*, como el sistema sobre el que se pretende ejecutar. Así, cuando se escribe una aplicación Java, se hace pensando que será ejecutada en una máquina virtual Java en concreto, siendo ésta la que en última instancia convierte de *código bytecode* a código nativo del dispositivo final.

La gran ventaja de la Java VM es su portabilidad al lenguaje de manera que se han creado diferentes máquinas virtuales Java para diferentes arquitecturas y así un programa *.class* escrito en un Windows puede ser interpretado en un entorno Linux, Solaris o Mac. Tan solo es necesario disponer de dicha máquina virtual para dichos entornos. De ahí el

famoso axioma que sigue a Java, "escribelo una vez, ejecútalo en cualquier parte". Éste concepto se resume en la ilustración 1.25.

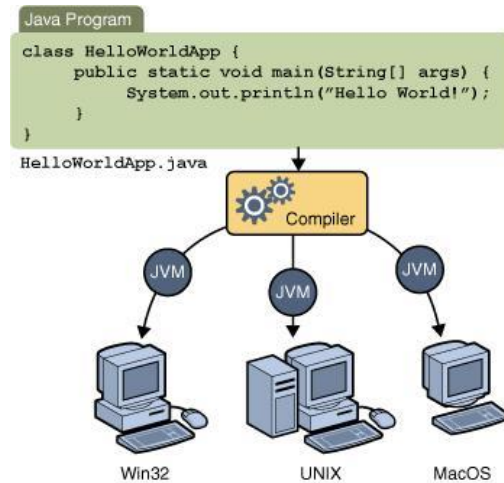


Ilustración 1.23: Java VM, la misma aplicación es capaz de ejecutarse sobre múltiples plataformas.

El API de Java es una gran colección de archivos de componentes de software que ofrecen muchas posibilidades. Se agrupan en bibliotecas de clases e interfaces; estas bibliotecas se conocen como paquetes, las cuales permiten efectuar toda clase de tareas necesarias dentro de un programa.

2.9.4. MySQL

MySQL es un sistema de administración de bases de datos relacionales (SGBDR) de código abierto, rápido, robusto y fácil de usar. Se adapta bien a la administración de datos en un entorno de red, especialmente en arquitecturas cliente-servidor. Proporciona muchas herramientas y es compatible con muchos lenguajes de programación. La manipulación de datos se puede hacer a través del *Lenguaje de consulta estructurado (Structured Query Language – SQL)* [29]

Los procesos de desarrollo de MySQL se enfocan en ofrecer una eficiente implementación de las características que más usa la gente a la hora de desarrollar aplicaciones con bases de datos relacionales.

MySQL es popular debido a los siguientes factores: [30]

- **Tamaño y velocidad**
- **Fácil instalación**
- **Enfocado a estándares**
- **Atención a la comunidad**
- **Fácil Interacción con otro software**

CAPÍTULO 2

ANÁLISIS Y DISEÑO

2.1. INTRODUCCIÓN

Hoy en día, en el mundo de la computación e informática, existe una gran cantidad de herramientas (software) para crear sistemas para redes de computadoras e Internet. Algunas son comerciales y es necesario pagar una licencia por su uso, pero también las hay libres o de código abierto, es decir no implica pago alguno para su uso. Con la finalidad de aprovechar estas herramientas se ha decidido realizar un sistema que ayude a resolver el problema descrito en este capítulo. Se describe el proceso de análisis y diseño, que resulta ser la base en la que se apoya un buen desarrollo e implementación. Esta base toma forma con el análisis orientado objetos y el uso de un modelo de proceso de software. También se hace uso del lenguaje de modelado UML y de algunos de sus diagramas para visualizar, especificar y documentar cada una de las partes que comprende el desarrollo del sistema.

2.2. PLANTEAMIENTO DEL PROBLEMA

La estación meteorológica se compone de una computadora de uso específico (datalogger) con un conjunto de puertos de entrada/salida (analógicos y digitales) y una memoria de almacenamiento. En los puertos se conectan los instrumentos para medir diferentes variables meteorológicas (temperatura, humedad, presión, dirección y velocidad del viento, etc.).

La energía para la estación es suministrada por un arreglo de acumuladores que a su vez recibe la energía que generan un grupo de celdas solares. La información que obtiene cada instrumento de medición es almacenada en un formato especial en la memoria del datalogger. Estos datos son recuperados en archivos de texto (ASCII) y almacenados en una computadora personal (PC). La transferencia de la información se hace mediante la comunicación entre puertos seriales (RS-232) que se controla con un software proporcionado por el fabricante del datalogger.

El sistema es, en lo general, eficiente; pero aun así presenta algunos aspectos negativos que impiden un mejor desempeño de la estación meteorológica. Uno de estos aspectos es la capacidad de almacenamiento de la memoria del datalogger que solo permite guardar los datos meteorológicos de un par de semanas. Esta situación obliga hacer viajes continuos al sitio de la estación para transferir los datos a una computadora personal. En ciertas temporadas del año, las condiciones meteorológicas son más extremas y se caracterizan por fuertes vientos, nevadas y constantes descargas atmosféricas. Estas condiciones llegan afectar de manera seria a la estación meteorológica, por ejemplo, el arreglo de acumuladores no se recarga lo suficiente porque los paneles solares se cubren de nieve, las descargas atmosféricas se inducen en el cableado eléctrico y provocan daños a los componentes electrónicos de los instrumentos y del datalogger. Como consecuencia de estos hechos, la información obtenida llega a presentar alteraciones, errores y en el peor de los casos la pérdida total. Lamentablemente el datalogger no cuenta con sistema de respaldo de datos.

En la actualidad, la infraestructura del sitio de la estación se ha enriquecido con la mejora y ampliación del servicio de suministro eléctrico, la instalación de pararrayos y la instalación de una red de datos con acceso a Internet. Ante este nuevo panorama, es importante

buscar nuevos métodos para la captura de los datos meteorológicos, almacenar los datos en dispositivos de almacenamiento masivo y realizar respaldos de información. También es importante aprovechar la red de datos para, de alguna forma, observar el comportamiento de las variables meteorológicas que se registran y tener acceso a los datos sin necesidad de ir al sitio donde se ubica la estación.

2.3. OBJETIVOS

Proporcionar un conjunto de programas informáticos que permitan la administración y el monitoreo remoto, a través de Internet, de las condiciones meteorológicas obtenidas por una estación meteorológica.

Como objetivos específicos, se establecen los siguientes:

- Establecer una comunicación entre el datalogger de la estación y una computadora (PC) para obtener los datos meteorológicos adquiridos por los instrumentos de medición.
- Procesar los datos adquiridos (aplicar formatos, filtrar datos, hacer conversión de unidades de medición y de bases numéricas)
- Almacenar la información en una base de datos.
- Implementar procedimientos para realizar consultas a la base de datos, generar reportes estadísticos y gráficas a través de una página web.
- Implementar procedimientos para la publicación, en una página de Internet, de los datos meteorológicos más recientes ocurridos en el sitio de la estación meteorológica.

2.4. REQUERIMIENTOS FUNCIONALES Y NO FUNCIONALES DEL SISTEMA

Los requerimientos funcionales hacen referencia a la descripción de las actividades y servicios que un sistema debe de proveer. Normalmente este tipo de requerimientos están vinculados con las entradas, las salidas, los procesos y los datos a almacenar en el sistema.

[31]

Los requerimientos funcionales con los que debe cumplir el sistema son:

- RF 1.** El sistema deberá establecer una comunicación, a través del puerto serie, con el datalogger para tener acceso a los datos que almacena en su memoria.
- RF 2.** El sistema solicitará al datalogger, cada minuto, los datos meteorológicos que ocurren en ese momento.
- RF 3.** Los datos de cada variable meteorológica serán deberán ser validados y procesados para poder ser almacenados en una base de datos.
- RF 4.** Los datos de cada variable meteorológica deberán ser almacenados en la base de datos junto con su hora y fecha de ocurrencia.

- RF 5.** El sistema deberá almacenar en la base de datos, al final de cada día, la estadística de cada variable meteorológica (valor mínimo, valor máximo y promedio).
- RF 6.** El sistema deberá publicar en una página web las condiciones meteorológicas actuales que suceden en el sitio y deberán ser actualizadas de manera constante y automática en periodos de tiempo no mayor a dos minutos.
- RF 7.** El sistema deberá ofrecer al cliente una interfaz gráfica agradable en donde se muestren los datos meteorológicos actuales.
- RF 8.** El sistema deberá ofrecer al usuario la opción de consultar las gráficas de todas las variables meteorológicas capturas en el transcurso de día.
- RF 9.** El sistema deberá ofrecer al usuario la opción de consultar las gráficas de todas las variables meteorológicas capturadas en una fecha anterior.

Los requerimientos no funcionales describen otras prestaciones, características y limitaciones que debe tener el sistema para alcanzar el éxito. Los requerimientos no funcionales engloban características como rendimiento, facilidad de uso, presupuestos, tiempo de entrega, documentación, seguridad y auditorías externas. **[31]**

Los requerimientos no funcionales con los que debe cumplir el sistema son:

- RNF 1.** Las consultas deberá hacerse desde diferentes computadores usando diferentes sistemas operativos.
- RNF 2.** La interfaz gráfica del cliente para consultar los últimos datos meteorológicos adquiridos deberá desplegarse en un navegador web que pueda ejecutar aplicaciones Java.
- RNF 3.** Los computadores deben tener acceso a Internet para conectarse el servidor en donde se ejecuta la aplicación servidor.
- RNF 4.** El servidor debe soportar entre 200 y 300 conexiones simultáneas sin causar la caída del servidor ni la pérdida de datos.
- RNF 5.** El sistema debe tener un nivel de disponibilidad mínimo del 99%
- RNF 6.** El equipo de cómputo deberá cumplir con los requisitos mínimos para la instalación del software del servidor y el SGBD.
- RNF 7.** El servidor deberá ejecutar aplicaciones Java.
- RNF 8.** El servidor deberá usar siempre la última versión de Java para garantizar la seguridad y estabilidad de las aplicaciones del servidor.
- RNF 9.** La administración del servidor y del SGBD será a través de los mismos sistemas de administración incluidos en el software.
- RNF 10.** Los respaldos de los datos guardados en la base de datos serán realizados por los propios sistemas de respaldo del SGBD.
- RNF 11.** El equipo de cómputo deberá soportar el uso de puertos seriales, ya sea nativos o convertidores USB-SERIAL, para comunicarse con el datalogger de la estación meteorológica.
- RNF 12.** El acceso a las consultas deberá ser para todo el público y sin necesidad de registro ni solicitud de algún dato personal en especial.

2.5. ALCANCES

- Desarrollo de un sistema informático multiplataforma, programado en lenguaje Java y basado en el modelo Cliente-Servidor que automatizará el proceso de administración de datos meteorológicos obtenidos por la estación meteorológica.
- Creación de una base de datos que guardará la información de los datos meteorológicos adquiridos por la estación meteorológica.
- Uso de la base de datos para generar reportes estadísticos y gráficas.
- Acceso, por parte de los usuarios, a los datos meteorológicos, reportes estadísticos y gráficas a través de una página web.
- Presentación de los datos meteorológicos recién adquiridos por la estación mediante el uso de una interfaz gráfica agradable para el usuario.
- Elaboración de la documentación referente al diseño, desarrollo e implementación del sistema.

2.6. METODOLOGÍA DE DESARROLLO Y ANALISIS DEL SISTEMA

El desarrollo del sistema se realizó aplicando una metodología basada en UP, empleando casos de uso, un desarrollo iterativo e incremental y empleando UML como lenguaje de modelado.

2.6.1. Glosario de términos

A continuación se listan las definiciones de los términos más utilizados en el análisis y diseño del sistema. Todas las definiciones están adaptadas al ambiente del sistema.

- **Estación meteorológica**
Una estación meteorológica es el lugar donde se realizan mediciones y observaciones puntuales de las diferentes variables meteorológicas utilizando los instrumentos adecuados para así poder establecer el comportamiento atmosférico o clima.
- **Variable meteorológica**
Es un elemento que describe un rasgo fundamental del clima. Los más representativos son la temperatura, humedad, presión atmosférica, radiación solar, precipitación pluvial, velocidad y dirección del viento.
- **Datos meteorológicos**
Es el conjunto de registros de variables meteorológicas que representan el estado del clima en un periodo de tiempo.
- **Datalogger**
Es un dispositivo electrónico con procesador y memoria que registra mediciones ordenadas en el tiempo provenientes de diferentes sensores. A través de una interfaz se pueden comunicar con una PC para ser programados y leer la información almacenada.
- **Puerto serie o serial**

Es una interfaz de comunicaciones de datos digitales usada frecuentemente por computadoras y dispositivos electrónicos.

- **Comunicación serial**

Es el intercambio de datos entre puertos serie. Consiste en el envío de un bit de información de manera secuencial, es decir, un bit a la vez y a un ritmo acordado entre el emisor y el receptor.

- **Reloj CPU**

Es un elemento de hardware que consulta la aplicación servidor para saber si ya ha transcurrido el lapso de tiempo definido para realizar una captura de datos meteorológicos.

- **Base de datos**

Es el lugar de almacenamiento de los datos meteorológicos capturados. El control lo realiza un gestor de bases de datos encargado de realizar todas las transacciones que le son solicitadas por el sistema.

- **Cliente web**

Es un usuario del sistema que usa un navegador web para consultar los datos obtenidos por la estación meteorológica.

2.6.2. Diagrama de casos de uso

Los requerimientos del sistema se encuentran expresados en el siguiente diagrama UML llamado *Diagrama general de casos de uso*. El símbolo de una persona (*stickman*) designa un *actor* y representan las entidades externas que interactúan con el sistema. Los óvalos representan los casos de uso, es decir, una funcionalidad que puede realizar un actor.

El diagrama de casos de uso actúa como una manera clave de comunicación entre el cliente y los creadores del sistema y ayuda a asegurar que se cumplan todas las necesidades de los clientes. La meta del diagrama de casos de uso es mostrar los tipos de interacciones que tienen los usuarios con el sistema, sin proporcionar detalles de dichas interacciones. [20]

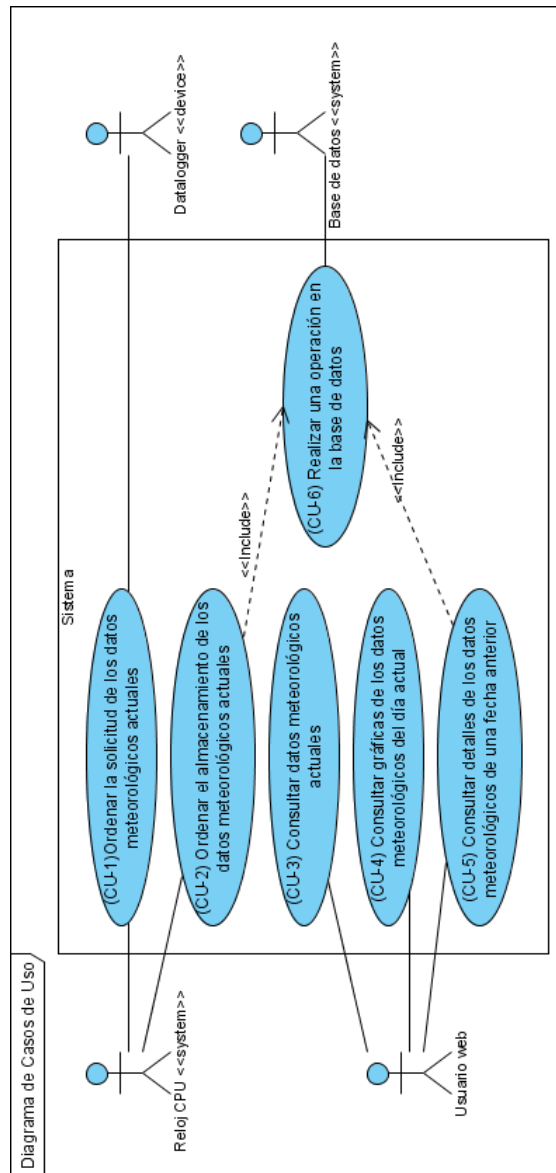


Ilustración 2.1: Diagrama general de casos de uso del sistema

2.6.3. Descripción de los actores del sistema

Se describen algunos detalles de los actores que interaccionan con el sistema. La tabla 2.1 muestra los relacionados al actor *Reloj CPU*.

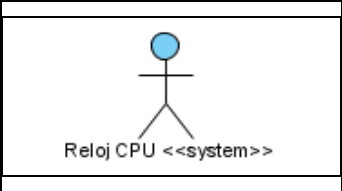
Símbolo UML del actor	 Reloj CPU <<system>>
Actor	<i>Reloj CPU</i>
Casos de uso	• (CU-1) Ordenar la solicitud de los datos meteorológicos actuales. • (CU-2) Ordenar el almacenamiento de los datos meteorológicos actuales. • (CU-6) Realizar una operación en la base de datos
Tipo	Primario
Descripción	Este actor representa a un sistema externo que ordena al sistema, cada minuto, la solicitud de los datos meteorológicos y posteriormente ordena al su almacenamiento.

Tabla 2.1: Detalles del actor "Reloj CPU"

La descripción para el actor Usuario web se muestra en la tabla 2.2.

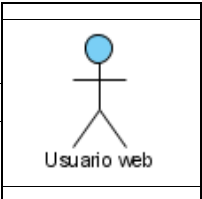
Símbolo UML del actor	 Usuario web
Actor	<i>Usuario web</i>
Casos de uso	• (CU-3) Consultar datos meteorológicos actuales. • (CU-4) Consultar gráficas de los datos meteorológicos del día actual. • (CU-5) Consultar detalles de los datos meteorológicos de una fecha anterior. • (CU-6) Realizar una operación en la base de datos
Tipo	Primario
Descripción	Este actor representa a una persona que accede al sistema para hacer uso de sus servicios a través de un navegador web.

Tabla 2.2: Detalles del actor "Usuario web"

De igual manera, la tabla 2.3 muestra los detalles para el actor *Datalogger*.

Símbolo UML del actor	 Datalogger <<device>>
Actor	<i>Datalogger</i>
Casos de uso	(CU-1) Ordenar la solicitud de los datos meteorológicos actuales.
Tipo	Secundario
Descripción	Este actor representa a un dispositivo que entrega los datos meteorológicos actuales cuando el sistema los solicita.

Tabla 2.3: Detalles del actor "Datalogger"

Finalmente, la descripción del actor Base de datos se muestra en la tabla 2.4.

Símbolo UML del actor	 Base de datos <<system>>
Actor	<i>Base de datos</i>
Casos de uso	(CU-2) Ordenar el almacenamiento de los datos meteorológicos actuales. (CU-5) Consultar detalles de los datos meteorológicos de una fecha anterior. (CU-6) Realizar una operación en la base de datos
Tipo	Secundario
Descripción	Este actor representa a un sistema externo que realiza las operaciones de almacenamiento y recuperación de datos meteorológicos cuando el sistema lo solicita.

Tabla 2.4: Detalles del actor "Base de datos"

2.6.4. Especificación de los casos de uso

La especificación de los casos de uso consisten en una descripción, en forma textual, de un conjunto pequeño de interacciones típicas, tales como flujo principal y flujos alternativos o excepcionales.

En la tabla 2.5 se muestran los detalles del *Caso de uso CU-1*.

Nombre del caso de uso	<i>Ordenar la solicitud de los datos meteorológicos actuales.</i>
Actores	<i>Reloj CPU, Datalogger</i>
Tipo	Básico
Descripción	El actor Reloj CPU realiza, cada minuto, este caso de uso para que el sistema solicite los datos meteorológicos al actor Datalogger.
Pre-condiciones	El sistema debe tener abierto un enlace de comunicación con el actor Datalogger.
Flujo normal	1. El sistema solicita al actor Datalogger los datos meteorológicos actuales. 2. El actor Datalogger recibe la solicitud. 3. El actor Datalogger responde con los datos meteorológicos actuales. 4. El sistema recibe y procesa los datos meteorológicos actuales. 5. El caso de uso finaliza con éxito.
Post-condiciones	El actor Reloj CPU puede realizar el caso de uso CU-2 y el sistema queda estable.
Flujos de excepción	• En el paso 1, Error: el sistema no pudo enviar la solicitud al actor Datalogger. • En el paso 3, Error: el actor Datalogger no respondió a la solicitud. • En el paso 4, Error: los datos recibidos no se pueden procesar.

Tabla 2.5: Especificación del caso de uso CU-1

El caso de uso CU-2: Ordenar el almacenamiento de los datos meteorológicos actuales, se describe en la tabla 2.6.

Nombre del caso de uso	<i>Ordenar el almacenamiento de los datos meteorológicos actuales.</i>
Actores	<i>Reloj CPU, Base de datos</i>
Tipo	Básico
Descripción	El actor Reloj CPU realiza este caso de uso para que el sistema envíe los datos meteorológicos actuales al actor Base de datos para su almacenamiento.
Pre-condiciones	El actor Reloj CPU debe de realizar el caso de uso CU-1 con éxito.
Flujo normal	1. El sistema inicia el caso de uso CU-6 para realizar la operación ALMACENAR. 2. El caso de uso finaliza con éxito.
Post-condiciones	El sistema queda estable.
Flujos de excepción	• En el paso 1, no se terminó con éxito el caso de uso CU-6.

Tabla 2.6: Especificación del caso de uso CU-2

En la tabla 2.7 se describe la especificación del caso de uso CU-3: *Consultar datos meteorológicos actuales*.

Nombre del caso de uso	<i>Consultar datos meteorológicos actuales.</i>
Actores	<i>Usuario web</i>
Tipo	Básico
Descripción	En este caso de uso, el actor Usuario web consulta al sistema para obtener los datos meteorológicos actuales y visualizarlos a través de un navegador web.
Pre-condiciones	No tiene.
Flujo normal	1. El actor Usuario web solicita la consulta de los datos meteorológicos actuales. 2. El sistema recibe la solicitud de consulta del actor Usuario web. 3. El sistema obtiene los datos meteorológicos actuales. 4. El sistema elabora una respuesta con los datos meteorológicos actuales y la envía. 5. El usuario web visualiza la respuesta. 6. El caso de uso finaliza con éxito.
Post-condiciones	El sistema queda estable.
Flujos de excepción	• En el paso 2, el sistema no reconoce la solicitud o no existe el recurso solicitado. Se elabora una respuesta explicando el error. Se continúa en el paso 5. • En el paso 3, el sistema no cuenta con los datos meteorológicos actuales. Se elabora una respuesta explicando el error y se continúa en el paso 5.

Tabla 2.7: Especificación del caso de uso CU-3

El caso de uso *Consultar gráficas de los datos meteorológicos del día actual*, declarado como Caso de uso No. 4, se muestra en la tabla 2.8

Nombre del caso de uso	<i>Consultar gráficas de los datos meteorológicos del día actual.</i>
Actores	<i>Usuario</i>
Tipo	Básico
Descripción	En este caso de uso, el actor Usuario web consulta al sistema para obtener las gráficas de los datos meteorológicos que han ocurrido en el día actual.
Pre-condiciones	El actor Usuario debe de realizar el caso de uso CU-3 con éxito.
Flujo normal	1. El actor Usuario web usa el menú para seleccionar la opción “Consultar gráficas del día actual”. 2. El sistema recibe del actor Usuario web la solicitud. 3. El sistema obtiene todos los datos meteorológicos del día actual. 4. El sistema elabora las gráficas de todos los datos meteorológicos del día actual. 5. El sistema elabora una respuesta con las gráficas elaboradas y la envía. 6. El usuario web visualiza la respuesta. 7. El caso de uso finaliza con éxito.
Post-condiciones	El sistema queda estable.
Flujos de excepción	• En el paso 2, Error: el sistema no reconoce la solicitud o no existe el recurso solicitado. Se elabora una respuesta explicando el error. Se continúa en el paso 6. • En el paso 5A-2, Error: el sistema no pudo elaborar las gráficas de todos los datos meteorológicos del día actual. Se elabora una respuesta explicando el error y se continúa en el paso 6. • En el paso 5B-1, Error: el sistema no pudo obtener todos los datos meteorológicos del día actual. Se elabora una respuesta explicando el error y se continúa en el paso 6.

Tabla 2.8: Especificación del caso de uso CU-4

El caso de uso No. 5, *Consultar detalles de los datos meteorológicos de una fecha anterior* se presenta en las tablas 2.9 y 2.10.

Nombre del caso de uso	<i>Consultar detalles de los datos meteorológicos de una fecha anterior.</i>
Actores	<i>Usuario, Base de datos</i>
Tipo	Básico
Descripción	En este caso de uso, el actor Usuario web consulta al sistema para obtener detalles sobre los datos meteorológicos que han ocurrido en una fecha anterior.
Pre-condiciones	El actor Usuario debe de realizar el caso de uso CU-3 con éxito.
Flujo normal	1. El actor Usuario web usa el menú para seleccionar la opción <i>“Consultar detalles de una fecha anterior”</i>. 2. El sistema muestra al actor Usuario Web un calendario. 3. El actor selecciona la fecha que quiere consultar. 4. El actor selecciona del sub-menú el tipo de detalle que quiere consultar. 5. El sistema recibe del actor Usuario web la solicitud con la fecha y el detalle a consultar. 6. Si la opción seleccionada por el actor Usuario web fue: • GRAFICAS se ejecuta el flujo alternativo A <i>“Consultar las gráficas de una fecha anterior”</i>. • ESTADISTICA se ejecuta el flujo alternativo B <i>“Consultar la estadística de una fecha anterior”</i>. 7. El sistema envía la respuesta. 8. El usuario web visualiza la respuesta. 9. El caso de uso finaliza con éxito.

Tabla 2.9: Especificación del caso de uso CU-5 (parte 1 de 2)

Flujos alternativos	5A. <i>Consultar las gráficas de una fecha anterior.</i> 5A-1. Se realiza el caso de uso CU-6 para realizar la operación CONSULTAR DATOS. 5A-2. El sistema elabora las gráficas de todos los datos meteorológicos correspondientes a la fecha elegida por el actor Usuario web 5A-3 El sistema elabora una respuesta con las gráficas elaboradas. 5A-4. El caso de uso prosigue en el paso 8 del flujo normal. 6B. <i>Consultar la estadística de una fecha anterior.</i> 5B-1. Se realiza el caso de uso CU-6 para realizar la operación CONSULTAR ESTADÍSTICA. 5B-2. El sistema elabora una respuesta con la estadística correspondiente a la fecha y al mes que pertenece la fecha. 5B-3. El caso de uso prosigue en el paso 8 del flujo normal.
Post-condiciones	El sistema queda estable.
Flujos de excepción	• En el paso 6, Error: el sistema no reconoce la solicitud, no existe el recurso solicitado o la fecha es igual o superior al día actual. Se elabora una respuesta explicando el error y se continúa en el paso 8. • En el paso 5A-2, Error: el sistema no pudo elaborar las gráficas de todos los datos meteorológicos correspondientes a la fecha elegida por el actor Usuario web. Se elabora una respuesta explicando el error y se continúa en el paso 8. • En el paso 5C-2, Error: el sistema no pudo obtener la estadística correspondiente a la fecha elegida por el actor Usuario web. Se elabora una respuesta explicando el error y se continúa en el paso 8.

Tabla 2.10: Especificación del caso de uso CU-5 (parte 2 de 2)

Finalmente, el caso de uso CU-6 se muestra en las tablas 2.11, 2.12 y 2.13.

Nombre del caso de uso	<i>Realizar una operación en la base de datos</i>
Actores	<i>Reloj CPU, Usuario web, Base de datos</i>
Tipo	Include
Descripción	En este caso, el sistema realiza una operación sobre los datos meteorológicos administrados por el Actor Base.
Pre-condiciones	No tiene.
Flujo normal	1. El sistema abre un enlace de comunicación con el actor Base de datos. 2. Según el tipo operación: • ALMACENAR se ejecuta el flujo alternativo A <i>“Almacenar datos meteorológicos actuales”</i>. • CONSULTAR DATOS se ejecuta el flujo alternativo B <i>“Obtener datos meteorológicos de una fecha anterior”</i>. • CONSULTAR ESTADÍSTICA se ejecuta el flujo alternativo C <i>“Obtener la estadística de una fecha anterior”</i> 3. El sistema cierra el enlace de comunicación con el actor Base de datos. 4. El caso de uso finaliza con éxito.

Tabla 2.11: Especificación del caso de uso CU-6 (parte 1 de 3)

Flujos alternativos	2A. <i>Almacenar datos meteorológicos actuales.</i> 2A-1. El sistema envía los datos meteorológicos actuales al actor Base de datos para su almacenamiento. 2A-2. El actor Base de datos recibe los datos meteorológicos actuales y los almacena. 2A-3. El caso de uso prosigue en el paso 3 del flujo normal. 2B. <i>Obtener datos meteorológicos de una fecha anterior.</i> 2B-1. El sistema envía la solicitud al actor Base de datos para obtener todos los datos meteorológicos correspondientes a la fecha que se especifica en la solicitud. 2B-2. El actor Base de datos recibe la solicitud del sistema. 2B-3. El actor Base de datos obtiene todos los datos meteorológicos correspondientes a la fecha que se especifica en la solicitud. 2B-4. El actor Base de datos envía en un mensaje el resultado de la solicitud al sistema. 2B-5 El sistema recibe el mensaje. 2B-6. El caso de uso prosigue en el paso 3 del flujo normal. 2C. <i>Obtener la estadística de una fecha anterior.</i> 2C-1. El sistema envía la solicitud al actor Base de datos para obtener la estadística correspondiente a la fecha y al mes que pertenece la fecha. 2C-2. El actor Base de datos recibe la solicitud del sistema. 2C-3. El actor Base de datos obtiene la estadística de correspondientes a la fecha que se especifica en la solicitud. 2C-4. El actor Base de datos envía en un mensaje el resultado de la solicitud al sistema. 2C-5 El sistema recibe el mensaje. 2C-6. El caso de uso prosigue en el paso 3 del flujo normal.
Post-condiciones	El sistema queda estable.

Tabla 2.12: Especificación del caso de uso CU-6 (parte 2 de 3)

Flujos de excepción	• En el paso 1, Error: el sistema no pudo abrir un enlace de comunicación con el actor Base de datos. El caso de uso finaliza con una excepción. • En el paso 2A-1, Error: el sistema no pudo enviar los datos meteorológicos actuales al actor Base de datos. El sistema cierra el enlace de comunicación con el actor Base de datos con una excepción. El caso de uso finaliza con una excepción. • En el paso 2A-2, Error: el actor Base de datos no pudo almacenar los datos meteorológicos actuales y envía un mensaje de operación fallida al sistema. El caso de uso continúa en el paso 3. • En el paso 2B-1, Error: el sistema no pudo enviar a solicitud al actor Base de datos para obtener todos los datos meteorológicos correspondientes a la fecha que se especifica en la solicitud. El sistema cierra el enlace de comunicación con el actor Base de datos con una excepción. El caso de uso finaliza con una excepción. • En el paso 2B-3, Error: el actor Base de datos no pudo obtener todos los datos meteorológicos correspondientes a la fecha que se especifica en la solicitud, ya sea por error en la búsqueda o no existen datos meteorológicos que cumplan la solicitud. Se envía un mensaje de operación fallida o aviso al sistema y se continúa en el paso 3. • En el paso 2C-1, Error: el sistema no pudo enviar a solicitud al actor Base de datos para obtener la estadística correspondiente a la fecha que se especifica en la solicitud. El sistema cierra el enlace de comunicación con el actor Base de datos con una excepción. El caso de uso finaliza con una excepción. • En el paso 2C-3, Error: el actor Base de datos no pudo obtener la estadística correspondientes a la fecha que se especifica en la solicitud, ya sea por error en la búsqueda o no existen datos meteorológicos que cumplan la solicitud. Se envía un mensaje de operación fallida al sistema o aviso y se continúa en el paso 3.
-----------------------------------	---

Tabla 2.13: Especificación del caso de uso CU-6 (parte 3 de 3)

2.6.5. Escenarios

Los escenarios son descripciones secuenciales de los pasos que se toman para llevar a cabo el caso de uso. Pueden incluir escenarios múltiples, para satisfacer circunstancias excepcionales y caminos de proceso alternativos.

Se presenta un escenario exitoso (*"Happy day path"*) y un escenario de excepción cada caso de uso. La tabla 2.14 contiene un ejemplo de escenario para el caso de uso *Ordenar la solicitud de los datos meteorológicos actuales*.

Escenarios para el caso de uso CU-1
ESCENARIO #1 (Happy day path)
1. A las 13:47:00 del martes 12 de junio se inicia el caso de uso. 2. El sistema envía el comando Q a través del enlace de comunicación <i>serial por el puerto COM8</i> para obtener las <i>mediciones actuales</i> de <i>temperatura, humedad, presión atmosférica, radiación solar, precipitación pluvial, velocidad y dirección del viento</i>. 3. El Datalogger TWI recibe el comando y verifica su validez. 4. El Datalogger TWI responde con el siguiente mensaje: 12/06/2012 13:47:00 139 001.9K/H 0217K 030.6C 029% 0805.1mb 0000.1mm 5. El sistema recibe el conjunto de valores y los procesa para obtener: 2012/06/12 13:47:00 139 1.9 217 30.6 29 805.1 0.1 6. El caso de uso CU-1 finaliza con éxito.
ESCENARIO #2 (Excepción)
1. A las 13:48:00. 100 del martes 12 de junio se inicia el caso de uso. 2. El sistema envía el comando a través del enlace de comunicación <i>serial por el puerto COM8</i> para obtener las <i>mediciones actuales</i> de <i>temperatura, humedad, presión atmosférica, radiación solar, precipitación pluvial, velocidad y dirección del viento</i>. 3. El Datalogger TWI recibe el comando y verifica su validez. 4. El Datalogger TWI responde con el siguiente mensaje: 12/06/2012 13:48:00 139 001.9K/H 0217K 030.6C 029% 0805.1mb 0000.1mm 5. El sistema recibe el mensaje y lo procesa para obtener: >>? 6. ERROR: "El sistema no puede procesar el mensaje >>?" 7. El caso de uso CU-1 finaliza con una excepción.

Tabla 2.14: Ejemplo de un escenario para el CU-1

La tabla 2.15 se presenta un ejemplo de escenario para el caso de uso *Ordenar el almacenamiento de los datos meteorológicos actuales*.

Escenarios para el caso de uso CU-2
ESCENARIO #1 (Happy day path)
1. A las 13:47:00.150 del martes 12 de junio se realizó el caso de uso CU-1 con éxito. 2. El sistema inicia el caso CU-6 para realizar la operación ALMACENAR con los datos obtenidos en el caso de uso CU-1. 3. El sistema abre un enlace de comunicación con <i>el sistema administrador de Base de Datos</i>. 4. Se inicia el flujo alternativo <i>Almacenar datos meteorológicos actuales</i>. 5. El sistema envía los datos obtenidos en el caso de uso CU-1 con la <i>orden</i> para almacenar los datos: 2012/06/12,13:47:00,139,1.9,217,30.6,29,805.1,0.1 6. El <i>sistema administrador de Base de Datos</i> recibe la <i>orden</i> y la ejecuta. 7. El sistema cierra el enlace de comunicación con el <i>sistema administrador de Base de Datos</i>. 8. El caso de uso CU-6 finaliza con éxito. 9. El caso de uso CU-2 finaliza con éxito.
ESCENARIO #2 (Excepción)
1. A las 13:47:00.150 del martes 12 de junio se realizó el caso de uso CU-1 con éxito. 2. El sistema inicia el caso CU-6 para realizar la operación ALMACENAR con los datos obtenidos en el caso de uso CU-1. 3. El sistema abre un enlace de comunicación con el <i>sistema administrador de Base de Datos</i>. 4. Se inicia el flujo alternativo <i>Almacenar datos meteorológicos actuales</i>. 5. El sistema envía los datos obtenidos en el caso de uso CU-1 con la <i>orden</i> para almacenar los datos: 2012/06/12,13:47:00,139,1.9,217,30.6,29,805.1,0.1 6. El <i>sistema administrador de Base de Datos</i> recibe la orden con los datos a almacenar. 7. ERROR: Se envía al sistema un mensaje de operación fallida al ejecutar la <i>orden</i>. 8. El sistema recibe el mensaje de operación fallida. 9. El sistema cierra el enlace de comunicación con el <i>sistema administrador de Base de Datos</i> con una excepción. 10. El caso de uso CU-6 finaliza con una excepción. 11. El caso de uso CU-2 finaliza con una excepción.

Tabla 2.15: Ejemplo de un escenario para el CU-2

Un ejemplo de escenario para el caso de uso *Consultar datos meteorológicos actuales*, se muestra en la tabla 2.16.

Escenarios para el caso de uso CU-3
ESCENARIO #1 (Happy day path)
1. <i>Jesús</i> ingresa en el navegador web la dirección URL para solicitar la consulta de los datos meteorológicos que están sucediendo el día de hoy, <i>12 de junio de 2012 a las 13:47:00</i>. 2. <i>Jesús</i> establece una comunicación con el sistema a través del navegador web. 3. El sistema recibe la solicitud de consulta de <i>Jesús</i>. 4. El sistema elabora una respuesta con los datos: /06/12 13:47:00 139 1.9 217 30.6 29 805.1 0.1 5. El sistema envía la respuesta para que el navegador web la despliegue en una ventana de navegación. 6. <i>Jesús</i> visualiza la respuesta en una ventana del navegador web. 7. El caso de uso CU-3 finaliza con éxito.
ESCENARIO #2 (Excepción)
1. <i>Adolfo</i> ingresa en el navegador web la dirección URL para solicitar la consulta de los datos meteorológicos que están sucediendo el día de hoy, <i>12 de junio de 2012 a las 13:49:00</i>. 2. <i>Adolfo</i> establece una comunicación con el sistema a través del navegador web. 3. ERROR: El sistema no encontró el recurso solicitado. 4. El sistema elabora una respuesta con el error <i>“El recurso requerido no está disponible”</i>. 5. El sistema envía la respuesta para que el navegador web la despliegue en una ventana de navegación. 6. <i>Adolfo</i> visualiza la respuesta en una ventana del navegador web. 7. El caso de uso finaliza con éxito.

Tabla 2.16: Ejemplo de un escenario para el CU-3

Para el caso de uso *Consultar gráficas de los datos meteorológicos del día actual*, se ha elaborado el siguiente ejemplo contenido en la tabla 2.17.

Escenarios para el caso de uso CU-4	
ESCENARIO #1 (Happy day path)	
1.	Jesús realizó el caso de éxito CU-3 con éxito.
2.	Jesús selecciona la opción <i>“Consultar gráficas del día actual”</i> .
3.	El sistema recibe de Jesús la solicitud para consultar las gráficas del día actual.
4.	El sistema obtiene todos los datos meteorológicos del día actual.
5.	El sistema elabora las gráficas del día actual para la <i>temperatura, humedad, presión atmosférica, radiación solar, precipitación pluvial, velocidad y dirección del viento</i> .
6.	El sistema elabora una respuesta con las gráficas elaboradas.
7.	El sistema envía la respuesta para que el navegador web la despliegue en una nueva de navegación.
8.	Jesús visualiza la respuesta en una nueva ventana de navegación del navegador web.
9.	El caso de uso CU-4 finaliza con éxito.
ESCENARIO #2 (Excepción)	
1.	Jesús realizó el caso de éxito CU-3 con éxito.
2.	Jesús usa el menú para seleccionar la opción <i>“Consultar gráficas del día actual”</i> .
3.	El sistema recibe de Jesús la solicitud para consultar las estadísticas del día actual.
4.	ERROR: El sistema no pudo obtener las estadísticas del día actual.
5.	El sistema elabora una respuesta con el error <i>“Las gráficas del día actual no están disponibles”</i> .
6.	El sistema envía la respuesta para que el navegador web la despliegue en una ventana de navegación.
7.	Jesús visualiza la respuesta en una nueva ventana de navegación del navegador web.
8.	El caso de uso CU-4 finaliza con éxito.

Tabla 2.17: Ejemplo de un escenario para el CU-4

El caso de uso CU-5 se ejemplifica un escenario en las tablas 2.18 y 2.19.

Escenarios para el caso de uso CU-5	
ESCENARIO #1 (Happy day path)	
1.	<i>Jesús realizó el caso de éxito CU-3 con éxito.</i>
2.	<i>Jesús usa el menú para seleccionar la opción “Consultar detalles de una fecha anterior”.</i>
3.	<i>El sistema muestra a Jesús un calendario.</i>
4.	<i>Jesús selecciona la fecha 10 de mayo de 2012.</i>
5.	<i>El sistema muestra a Jesús C. R. un sub-menú.</i>
6.	<i>Jesús selecciona del sub-menú DATOS.</i>
7.	<i>El sistema recibe de Jesús C. R. la solicitud para consultar los datos del día 10 de mayo de 2012.</i>
8.	<i>Se realiza el caso de uso CU-6 para realizar la operación CONSULTAR DATOS para la fecha 10 de mayo de 2012.</i>
9.	<i>El sistema abre un enlace de comunicación con el <i>sistema administrador de Base de Datos</i>.</i>
10.	<i>El sistema envía la <i>orden</i> para obtener los datos meteorológicos del 10 de mayo del 2012.</i>
11.	<i>El <i>sistema administrador de Base de Datos</i> recibe la <i>orden</i> y la ejecuta.</i>
12.	<i>El <i>sistema administrador de Base de Datos</i> obtiene los datos meteorológicos pertenecientes al 10 de mayo del 2012.</i>
13.	<i>El <i>sistema administrador de Base de Datos</i> envía la respuesta a la <i>orden</i> del sistema.</i>
14.	<i>El sistema recibe la respuesta.</i>
15.	<i>El sistema cierra el enlace de comunicación con el <i>sistema administrador de Base de Datos</i>.</i>
16.	<i>El caso de uso CU-6 finaliza con éxito.</i>
17.	<i>El sistema elabora una respuesta con los valores de <i>temperatura, humedad, presión atmosférica, radiación solar, precipitación pluvial, velocidad y dirección del viento</i> del día 10 de mayo del 2012.</i>
18.	<i>El sistema envía la respuesta para que el navegador web la despliegue en una nueva de navegación.</i>
19.	<i>Jesús visualiza la respuesta en una nueva ventana de navegación del navegador web.</i>
20.	<i>El caso de uso CU-5 finaliza con éxito.</i>

Tabla 2.18: Ejemplo de un escenario para el CU-5 (parte 1 de 2)

ESCENARIO #2 (Excepción)
1. <i>Adolfo</i> realizó el caso de éxito CU-3 con éxito. 2. <i>Adolfo</i> usa el menú para seleccionar la opción “Consultar detalles de una fecha anterior”. 3. El sistema muestra a <i>Adolfo</i> un calendario. 4. <i>Adolfo</i> selecciona la fecha <i>10 de mayo de 2014</i>. 5. El sistema muestra a <i>Adolfo</i> un sub-menú. 6. <i>Adolfo</i> selecciona del sub-menú DATOS. 7. El sistema recibe de <i>Adolfo</i> la solicitud para consultar los datos del día <i>10 de mayo de 2014</i>. 8. ERROR: La fecha es superior al día actual. 9. El sistema elabora una respuesta con el error “<i>La fecha proporcionada es invalida porque es superior al día de hoy</i>”. 10. El sistema envía la respuesta para que el navegador web la despliegue en una ventana de navegación. 11. <i>Adolfo</i> visualiza la respuesta en una nueva ventana de navegación del navegador web. 12. El caso de uso CU-5 finaliza con éxito.

Tabla 2.19: Ejemplo de un escenario para el CU-5 (parte 2 de 2)

2.6.6. Diagrama de clases

Un diagrama de clases permite modelar conceptos puros o abstracciones, sin importar como sean estas clases. Un diagrama hecho desde esta perspectiva suelen ser muy sencillo y no tienen detalle alguno más que para nombrar las clases.

En la ilustración 2.2 se muestran las clases que se hallaron en el análisis del sistema.

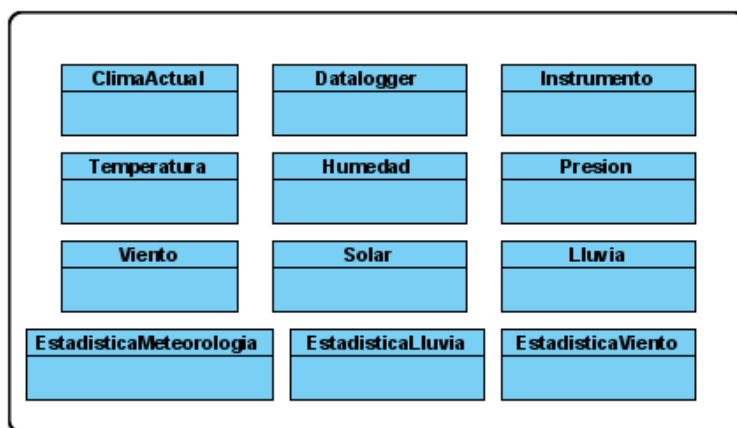


Ilustración 2.2: Diagrama de clases básico del sistema

El diccionario de clases se describe en las tablas 2.20 y 2.21 donde se especifica el uso de cada término y función dentro del sistema.

Nombre de clase	Definición de trabajo
<i>ClimaActual</i>	Fecha, hora y valor de las variables meteorológicas capturadas por el sistema en el transcurso del día.
<i>Datalogger</i>	Nombre del dispositivo electrónico que administra los valores de las variables meteorológicas y el nombre del puerto serie que usa para comunicarse con el sistema.
<i>Instrumento</i>	Información de los instrumentos que miden las diferentes variables meteorológicas.
<i>Temperatura</i>	Fecha, hora y valor de la variable meteorológica Temperatura.
<i>Humedad</i>	Fecha, hora y valor de la variable meteorológica Humedad.
<i>Presión</i>	Fecha, hora y valor de la variable meteorológica Presión Atmosférica.
<i>Viento</i>	Fecha, hora y valor de la variable meteorológica Viento, tanto dirección como velocidad.

Tabla 2.20: Diccionario de clases del sistema (parte 1 de 2)

<i>Solar</i>	Fecha, hora y valor de la variable meteorológica Radiación Solar.
<i>Lluvia</i>	Fecha, hora y valor de la variable meteorológica Precipitación Pluvial.
<i>EstadisticaMeteorologia</i>	Promedio del día; fecha y hora de ocurrencia, en el día, de los valores máximos y mínimos de las variables meteorológicas temperatura, humedad, presión atmosférica y radiación solar.
<i>EstadisticaLluvia</i>	Registro de la fecha y el valor de la precipitación pluvial acumulada en un día.
<i>EstadisticaViento</i>	Promedio del día; fecha y hora de ocurrencia de los valores máximos y mínimos del viento.

Tabla 2.21: Diccionario de clases del sistema (parte 2 de 2)

2.7. DISEÑO DEL SISTEMA

En las siguientes secciones se presentan los diferentes diagramas UML que representan los casos de uso hallados en la etapa de análisis del sistema.

2.7.1. Diagramas VOPC (Vista de Clases Participantes)

Un diagrama VOPC (*View Of Participating Classes*) muestra como los objetos del sistema trabajan juntos en la realización de un caso de uso. Las clases usadas son de tipo abstracto y se clasifican en:

- **Clases Limite** (<<Boundary>>) modelan la interacción del sistema con los actores.
- **Clases Control** (<<Control>>) capturan el control lógico de uno o más casos de uso.
- **Clase Entidad** (<<Entity>>) representan los conceptos de negocio manipulados por el sistema.

La ilustración 2.3 muestra el diagrama VOPC para cada caso de uso CU-1 del sistema, junto con sus respectivas clases de tipo abstracto.

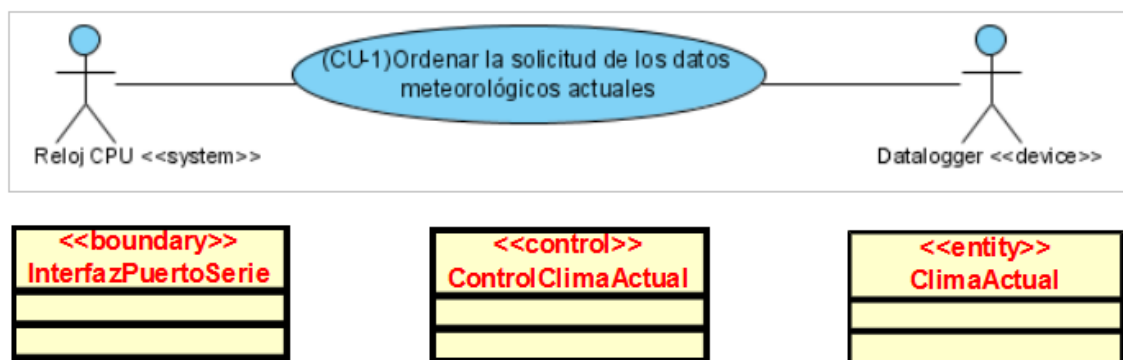


Ilustración 2.3: Diagrama VOPC Caso de uso CU-1

El diagrama VOPC del caso de uso CU-2 se presenta en la ilustración 2.4.

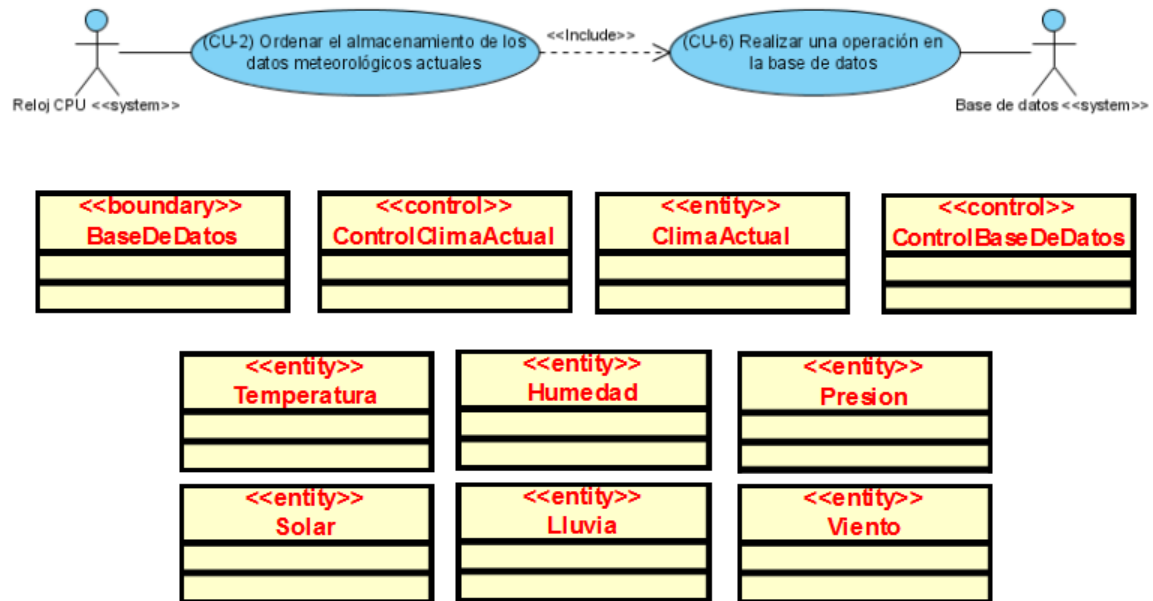


Ilustración 2.4: Diagrama VOPC Caso de uso CU-2

En la ilustración 2.5 se muestra el diagrama VOPC para el caso de uso *Consultar datos meteorológicos actuales*.

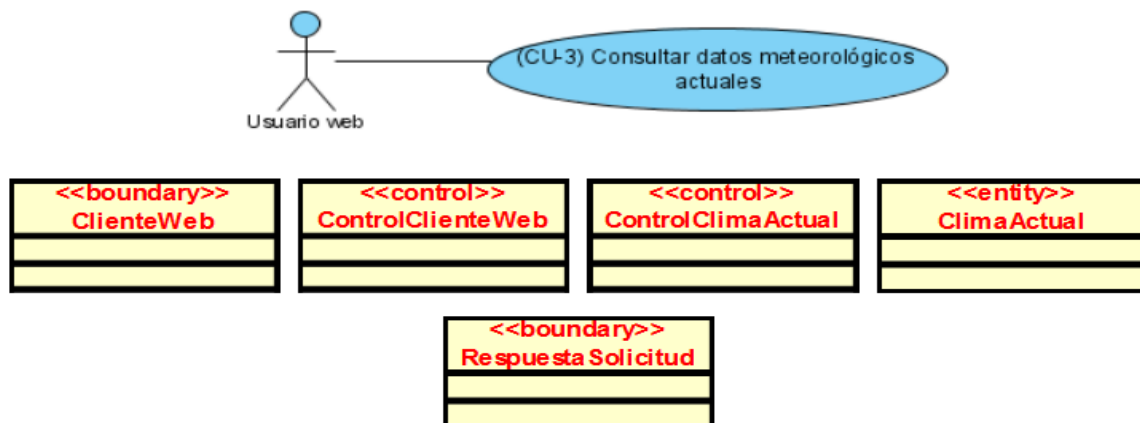


Ilustración 2.5: Diagrama VOPC Caso de uso CU-3

Los diagramas VOPC para los casos de uso CU-4 y CU-5 se muestran en las ilustraciones 2.6 y 2.87, respectivamente.

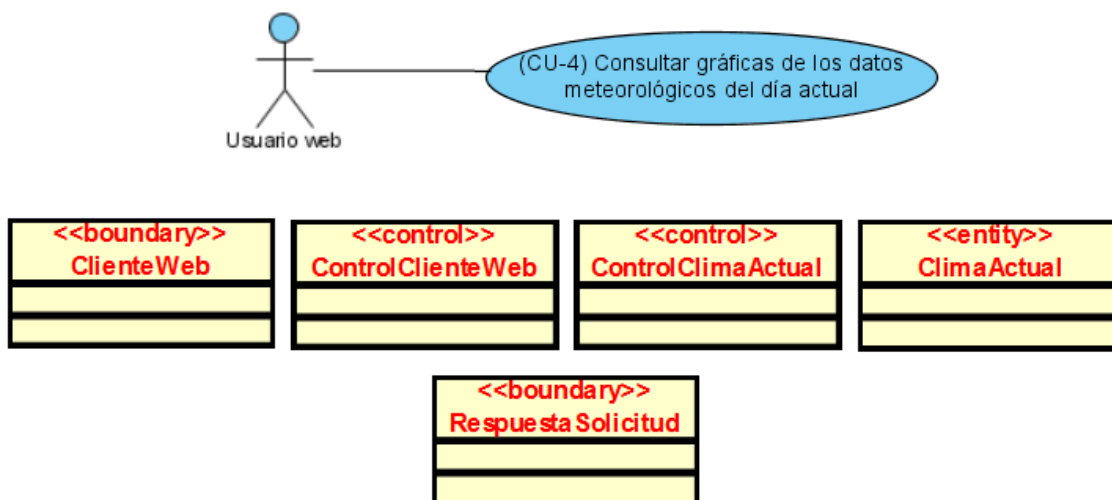


Ilustración 2.6: Diagrama VOPC Caso de uso CU-4

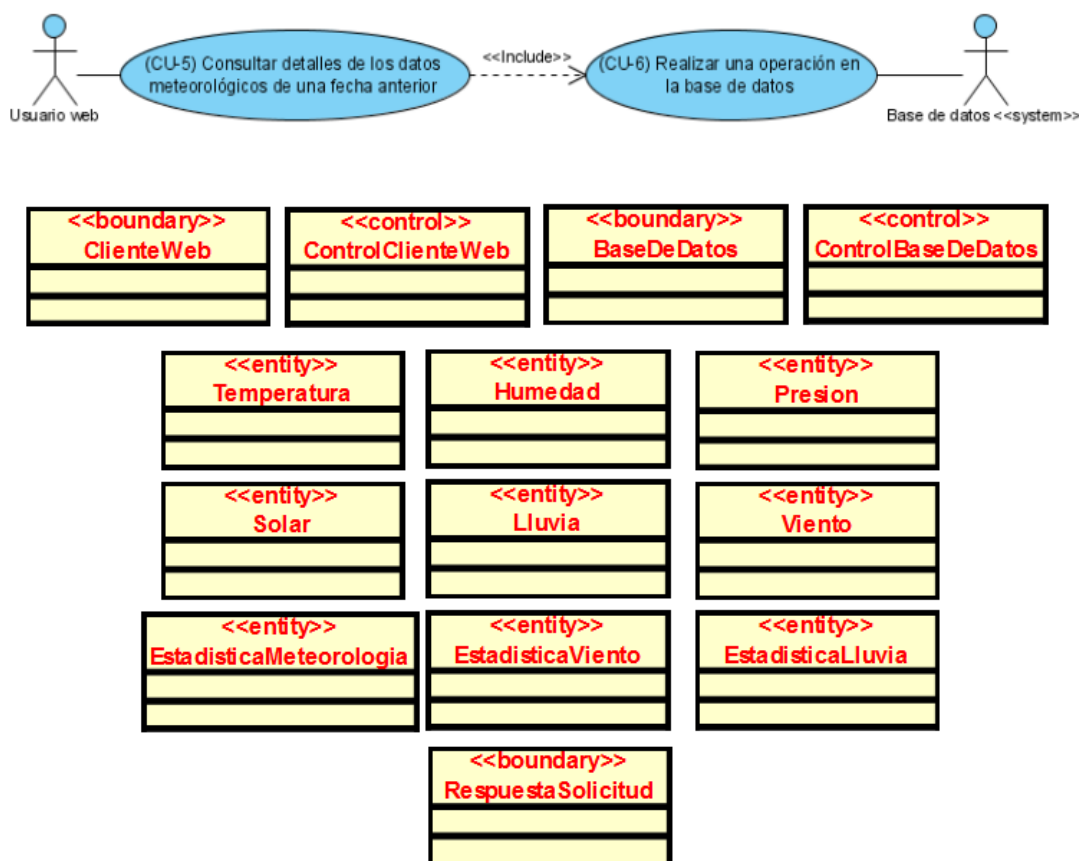


Ilustración 2.7: Diagrama VOPC Caso de uso CU-5

2.7.2. Diagramas de secuencia

Un diagrama de secuencia describe las interacciones entre un grupo de objetos mostrando de forma secuencial los envíos de mensajes entre objetos. El diagrama puede asimismo mostrar los flujos de datos intercambiados durante el envío de mensajes. [32]

El diagrama de secuencia CU-1 de la ilustración 2.8 muestra las interacciones de los actores involucrados en el caso de uso *Ordenar la solicitud de los datos meteorológicos actuales*.

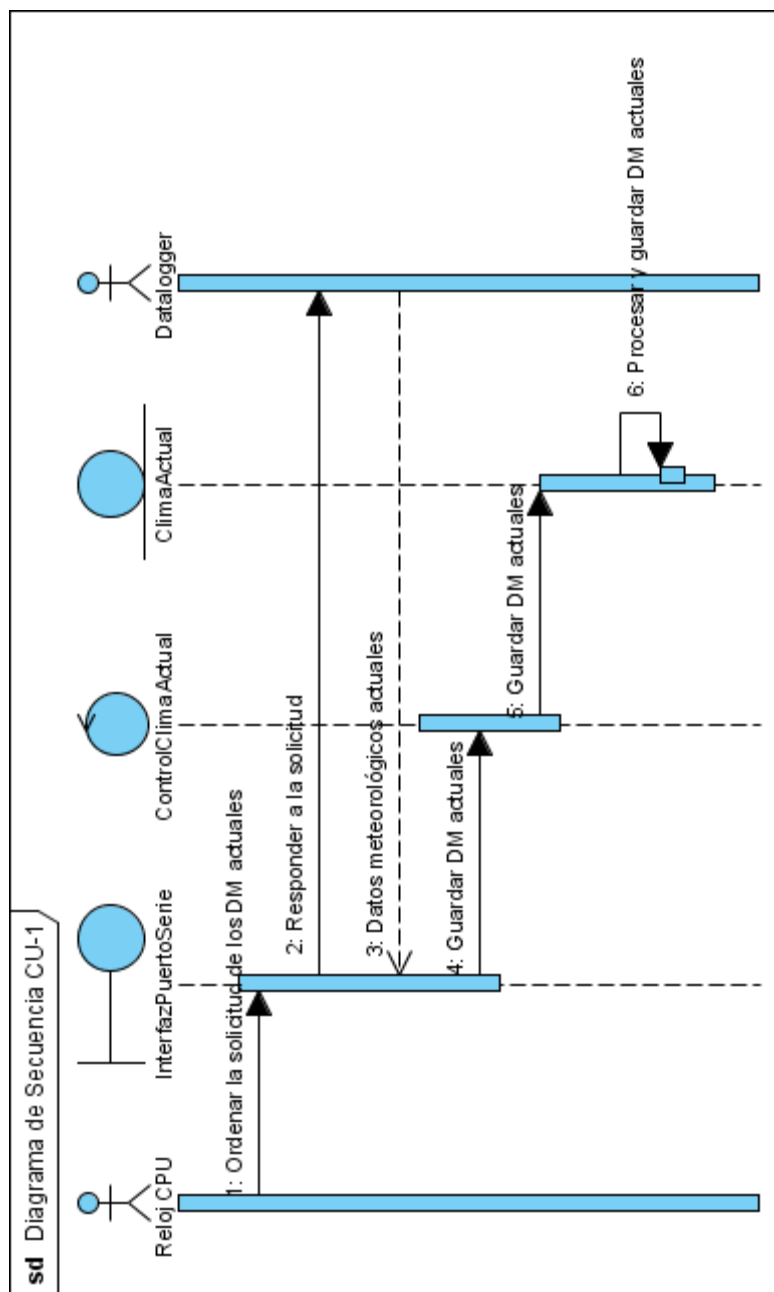


Ilustración 2.8: Diagrama de secuencia del CU-1

El diagrama de secuencia CU-2 de la ilustración 2.9 muestra todas las interacciones del actor *Reloj CPU* con las diversas clases abstractas definidas en el sistema para realizar el caso de uso *Ordenar el almacenamiento de los datos meteorológicos actuales*.

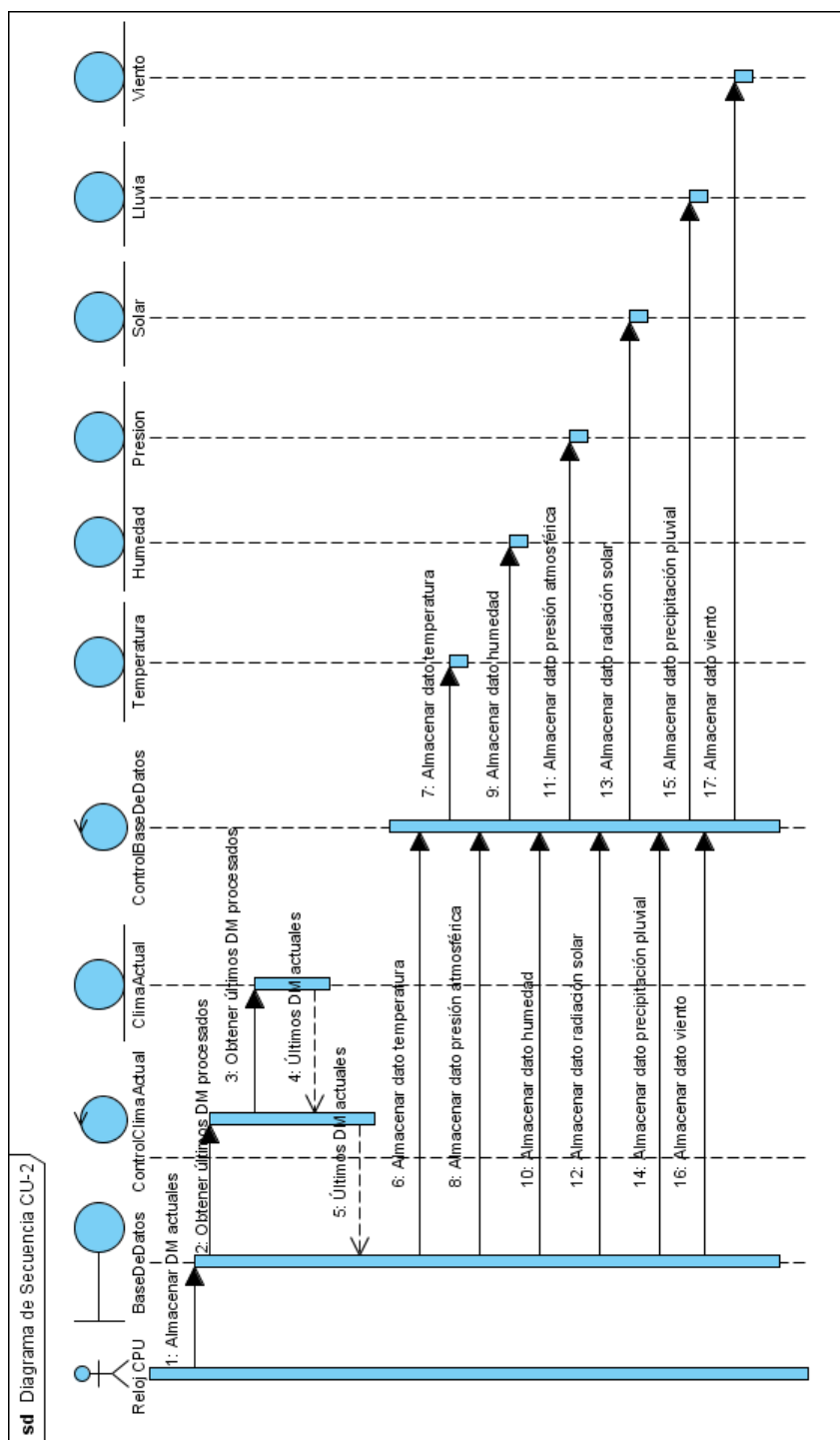


Ilustración 2.9: Diagrama de secuencia del CU-2

La secuencia del caso de uso *Consultar datos meteorológicos actuales* se encuentra representado en el diagrama de secuencia de la ilustración 2.10.

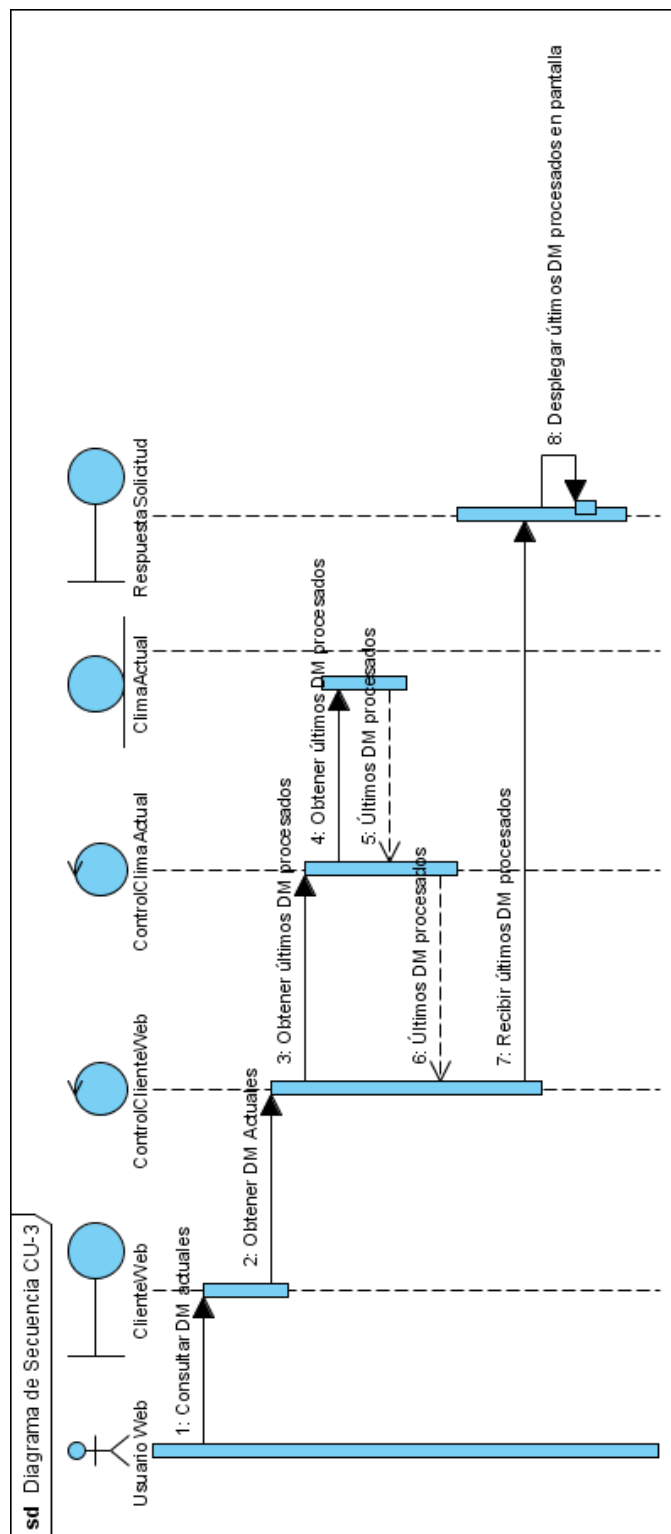


Ilustración 2.10: Diagrama de secuencia del CU-3

Las interacciones del caso de uso *Consultar gráficas de los datos meteorológicos del día actual* se presentan en el diagrama de secuencia de la ilustración 2.11.

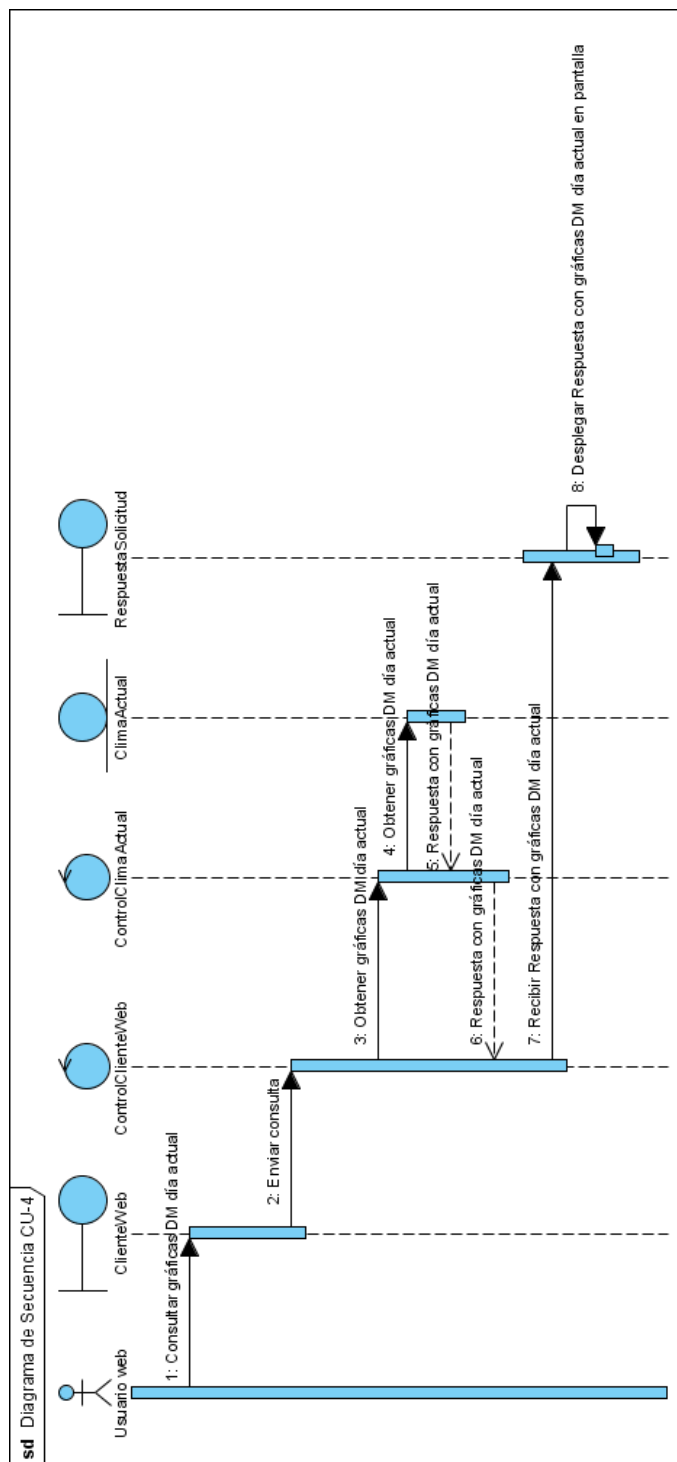


Ilustración 2.11: Diagrama de secuencia del CU-4

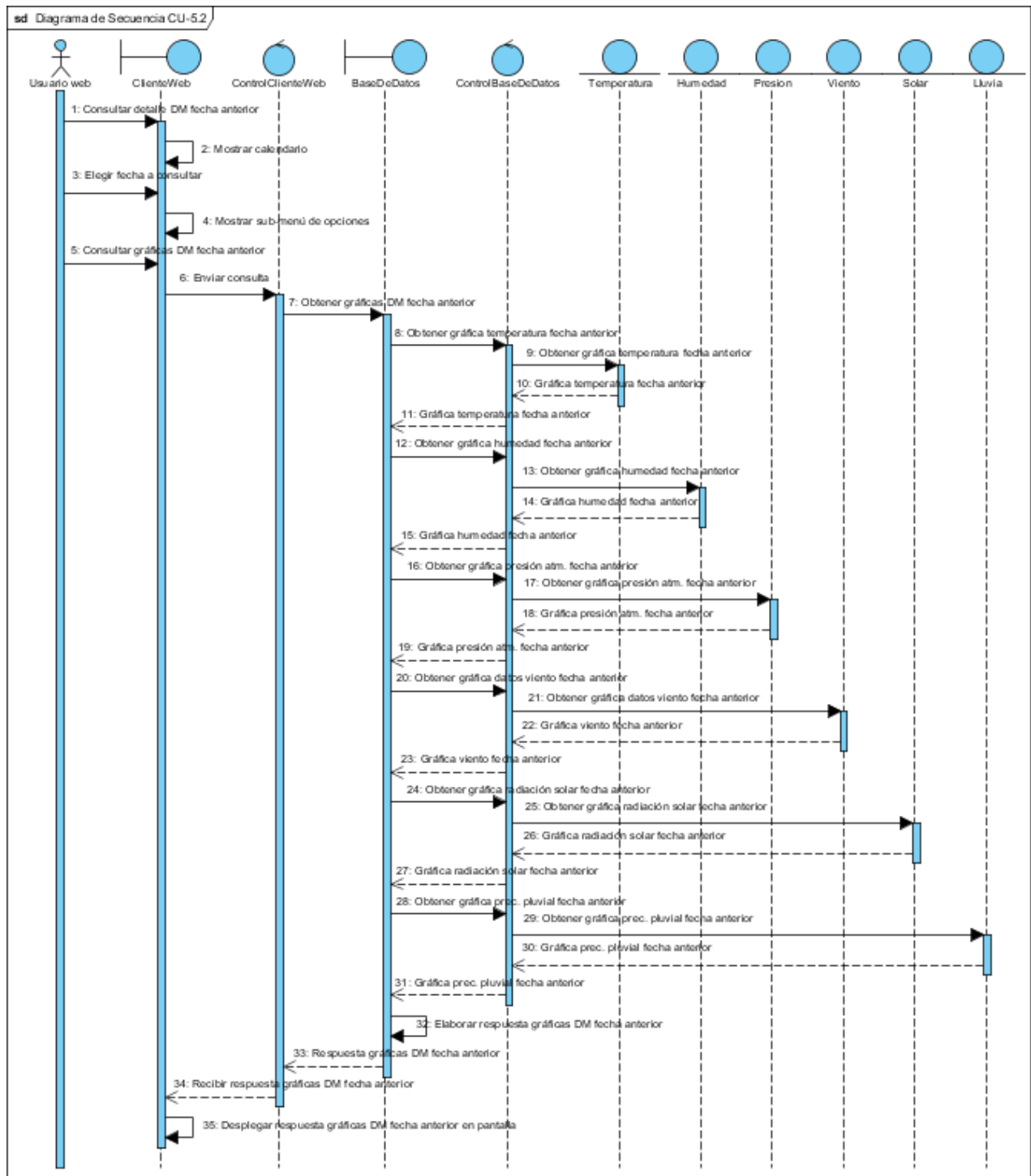


Ilustración 2.12: Diagrama de secuencia CU-5.1

El caso de uso *Consultar detalles de los datos meteorológicos de una fecha anterior* presenta flujos alternos. En el diagrama de secuencia de la ilustración 2.12 muestra cuando el actor *Usuario web* quiere consultar las gráficas de una fecha anterior.

El diagrama de secuencia mostrado en la ilustración 5.2 pertenece al caso de uso *Consultar detalles de los datos meteorológicos de una fecha anterior* en su flujo alterno para cuando el actor Usuario web quiere consultar las estadísticas de una fecha anterior.

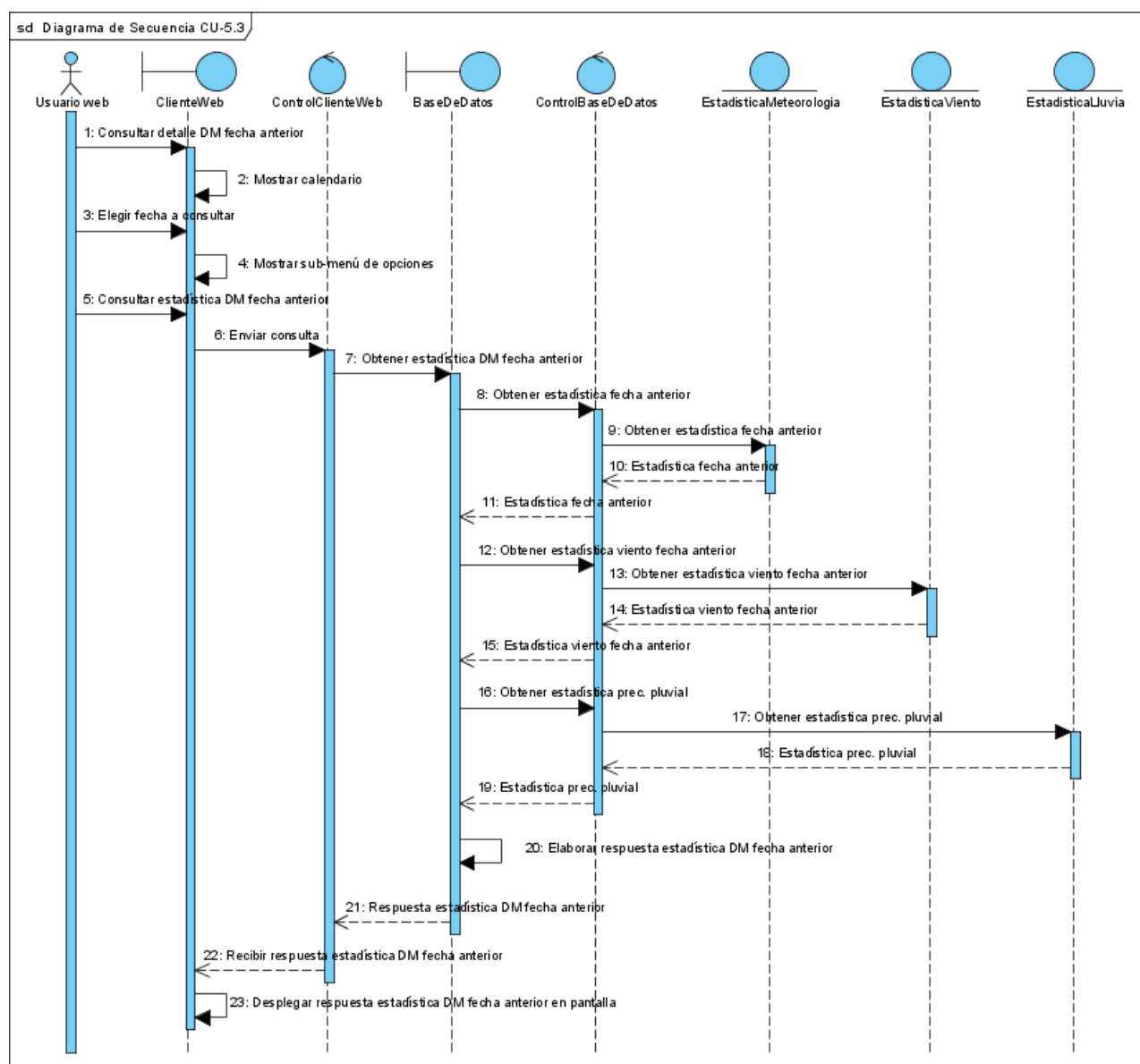


Ilustración 2.13: Diagrama de secuencia del CU-5.2

2.7.3. Diagramas de colaboración

Un diagrama de colaboración es un diagrama de interacción que resalta la organización estructural de los objetos que envían y reciben los mensajes. Este tipo de diagrama muestra un conjunto de objetos, enlaces entre ellos y los mensajes que intercambian. Un diagrama de colaboración es un grafo, donde los nodos son los objetos y los arcos son los enlaces. Un enlace es una instancia de una asociación o una dependencia entre clases. Se representa con una línea continua que une los dos objetos.

El caso de uso *Ordenar la solicitud de los datos meteorológicos actuales* cuenta con el diagrama de colaboración mostrado en la ilustración 2.14.

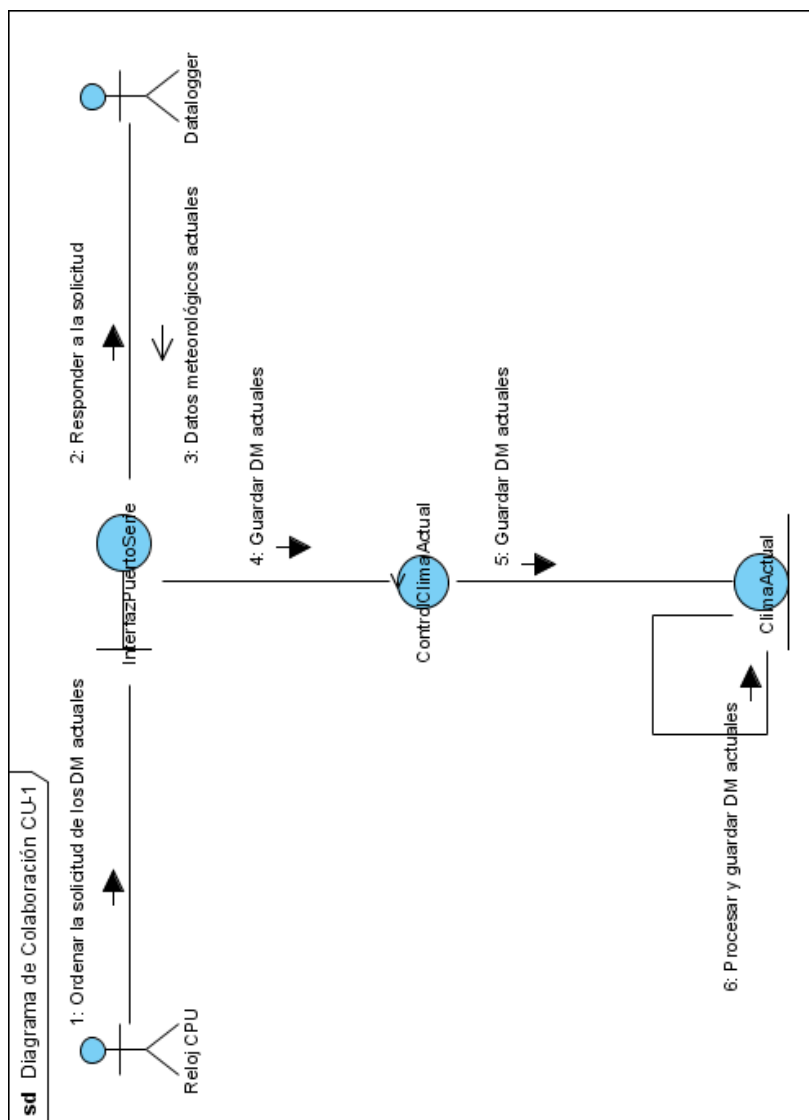


Ilustración 2.14: Diagrama de colaboración del CU-1

En la ilustración 2.15 se muestra el diagrama de colaboración pertenece al caso de uso *Ordenar el almacenamiento de los datos meteorológicos actuales*.

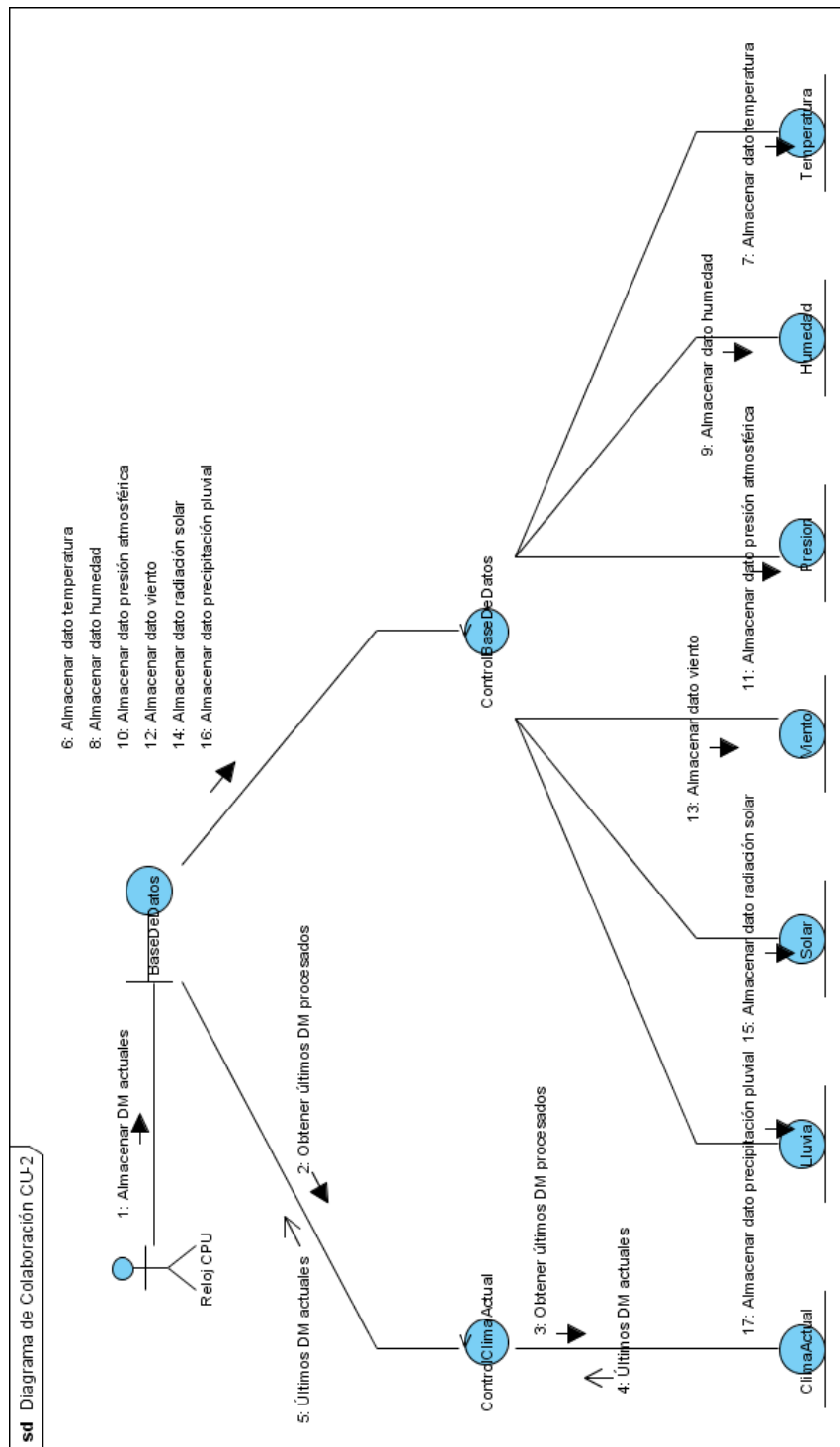


Ilustración 2.15: Diagrama de colaboración CU-2

Para el caso de uso *Consultar datos meteorológicos actuales* se tiene el siguiente diagrama de colaboración mostrado en la ilustración 2.16.

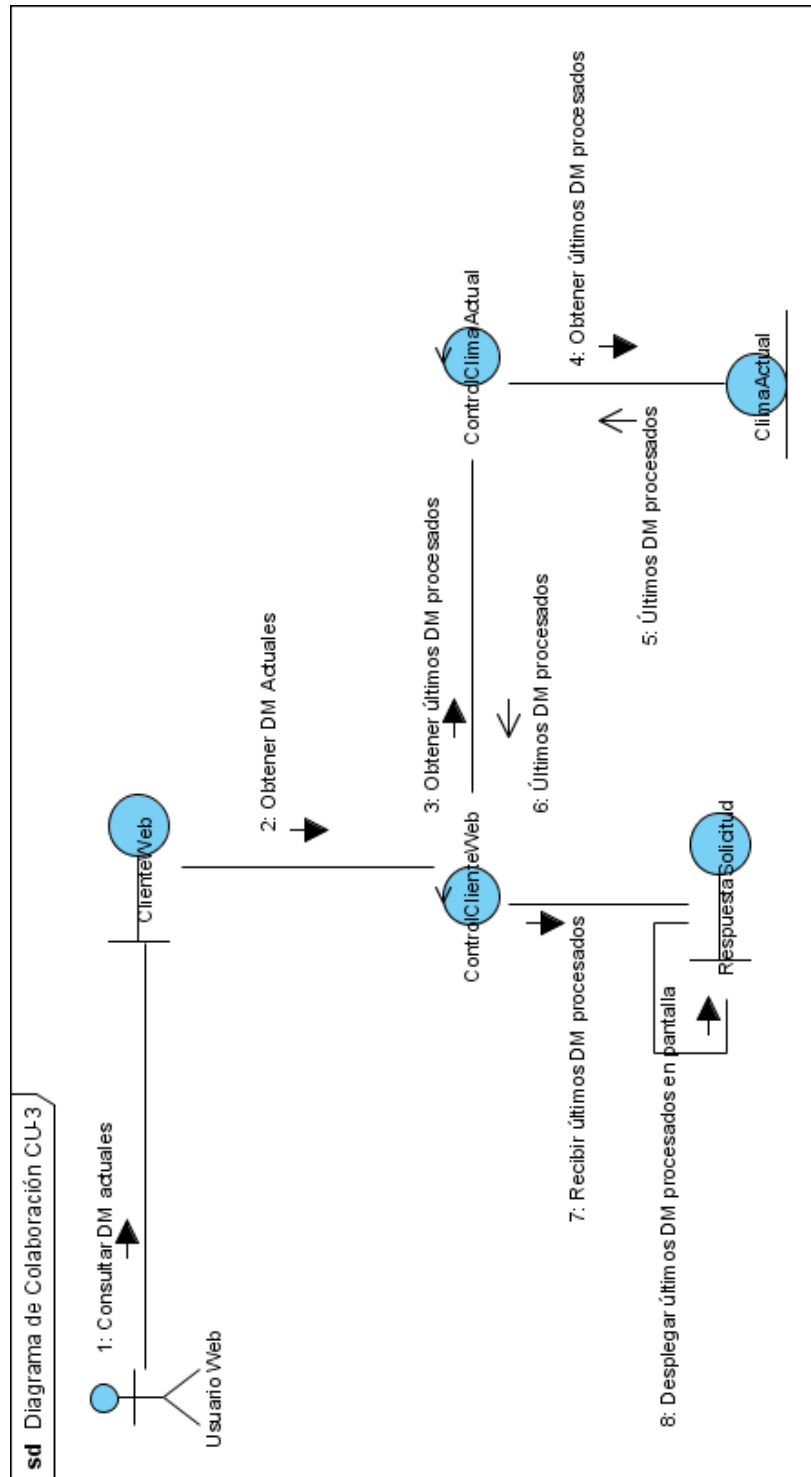


Ilustración 2.16: Diagrama de colaboración del CU-3

El diagrama de colaboración que se muestra en la ilustración 2.17 pertenece al caso de uso *Consultar gráficas de los datos meteorológicos del día actual*

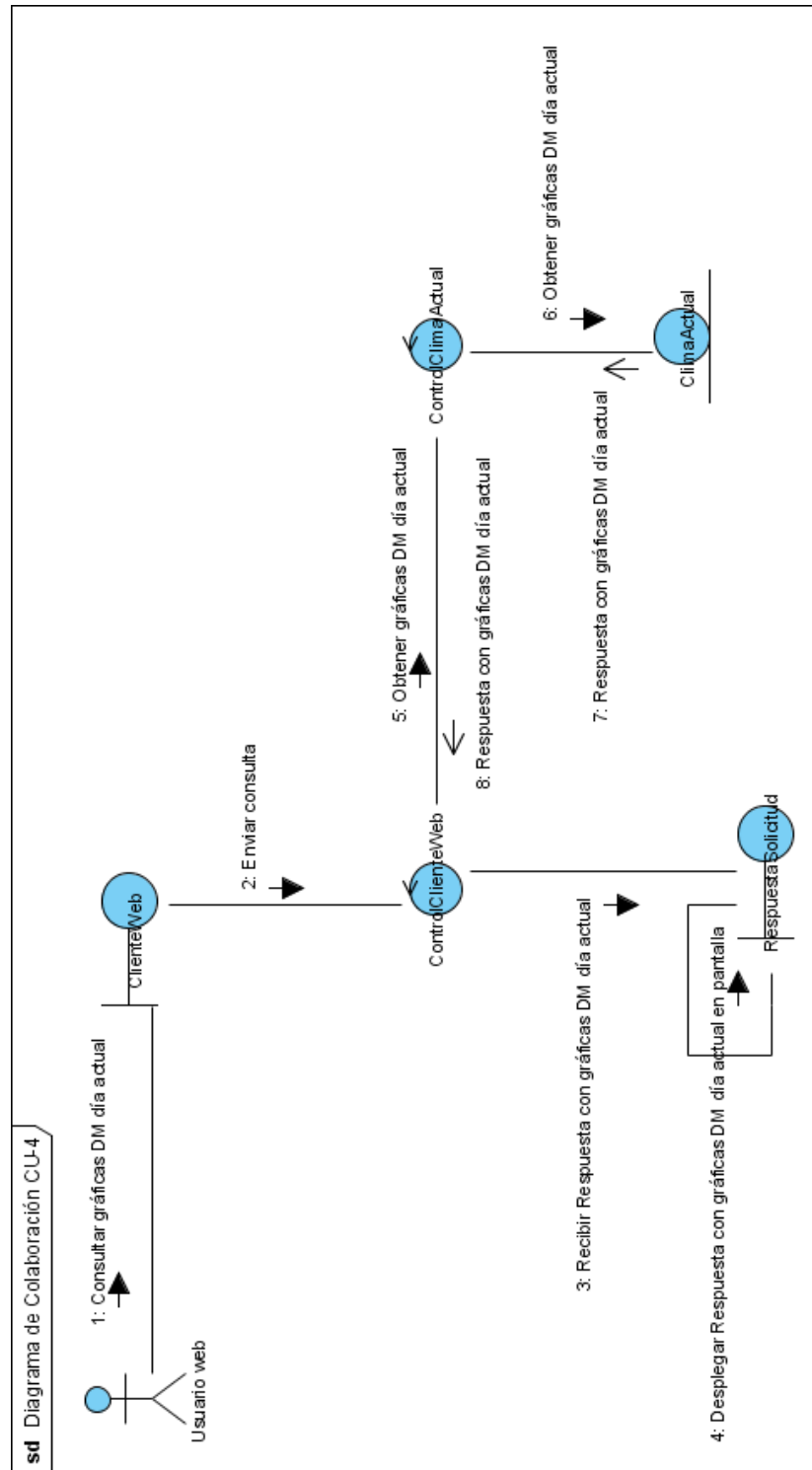


Ilustración 2.17: Diagrama de secuencia del CU-4

El diagrama de colaboración que muestra el caso de uso *Consultar detalles de los datos meteorológicos de una fecha anterior* para la opción de consultar gráficas de una fecha anterior, se muestra en la ilustración 2.18.

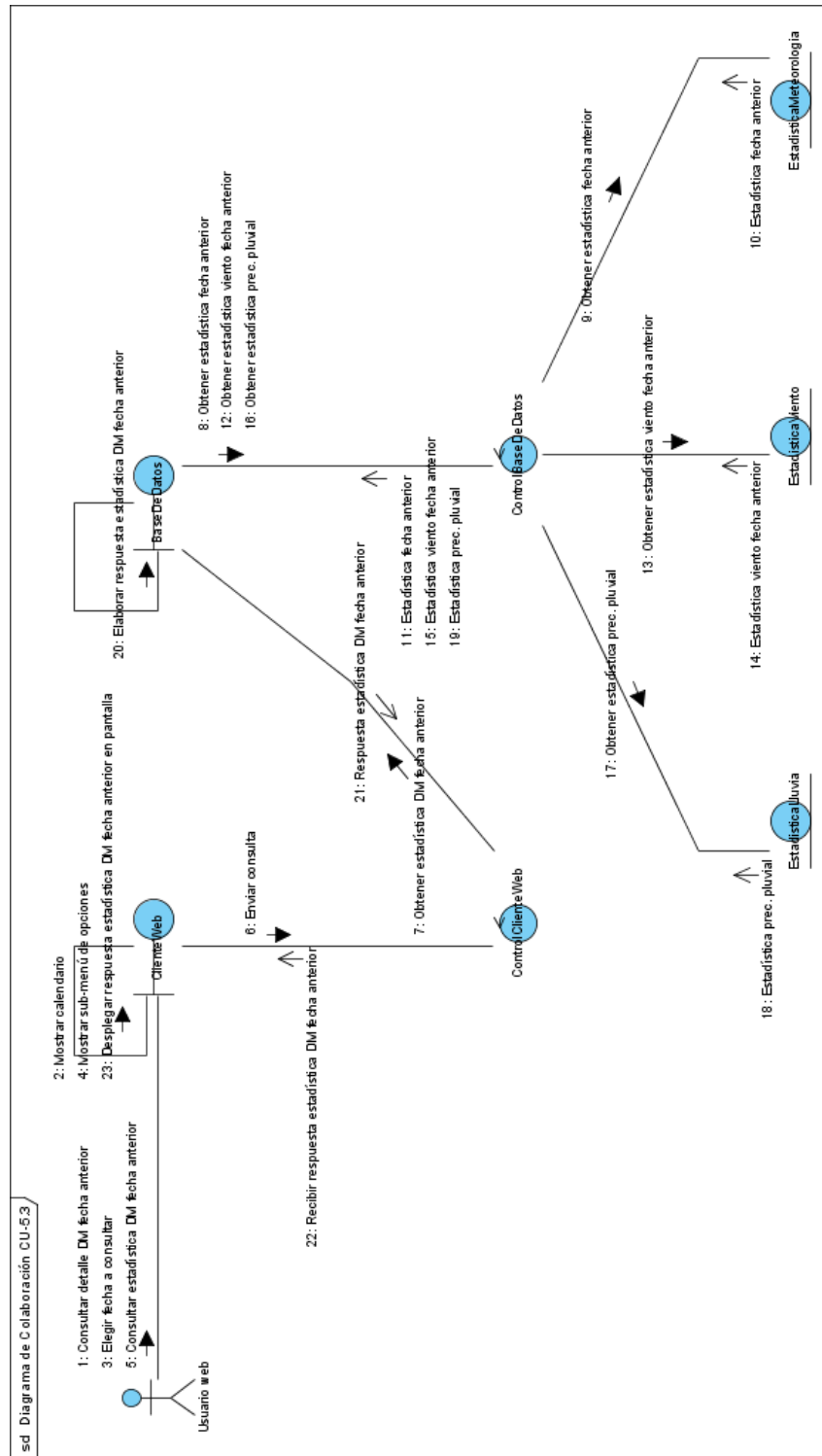


Ilustración 2.19: Diagrama de colaboración del CU-5.2

CAPÍTULO 3

DISEÑO DE LA BASE DE DATOS

3.1. DISEÑO CONCEPTUAL DE LA BASE DE DATOS

Cuando se trabaja bajo el análisis conceptual de una situación, nos referimos a la abstracción de hechos reales de los cuales se emite un concepto o es posible hacer una idea de ello. Para poder realizar la abstracción de un tema en un área específica, es necesario tener los requerimientos formulados por los usuarios. Estos requerimientos contienen el conjunto de hechos y reglas que dan pauta a la creación del esquema conceptual y se podrá realizar una descripción de alto nivel de la futura base de datos. Para manipular este esquema se utiliza un modelo conceptual que proporciona un lenguaje que permite utilizar un conjunto de símbolos (estándares) para la creación de la base de datos.

El diseño conceptual se hace independiente al sistema gestor de base de datos (SGBD) que se utilice para la implementación.

Para modelar conceptualmente es posible utilizar varios modelos de datos. Un modelo práctico para ilustrar el diseño conceptual es el modelo Entidad-Relación.

Durante la fase de análisis y diseño del sistema se logró la identificación de varias entidades y sus respectivos atributos, esta tarea resultó sencilla debido a que el objetivo principal del sistema es de almacenar una gran cantidad de datos de las diversas variables meteorológicas que capturan los instrumentos de la estación. A parte del almacenamiento directo de cada variable meteorológica, se realiza también el almacenamiento de una estadística básica al final del día. Esta tarea implica también un proceso de almacenamiento. La ilustración 3.1 muestra el diagrama E-R obtenido que modela la base de datos del sistema.

En la siguiente sección se describirán cada uno de los elementos que forman parte del diseño conceptual y del diagrama Entidad-Relación.

3.1.1. Modelo Entidad-Relación

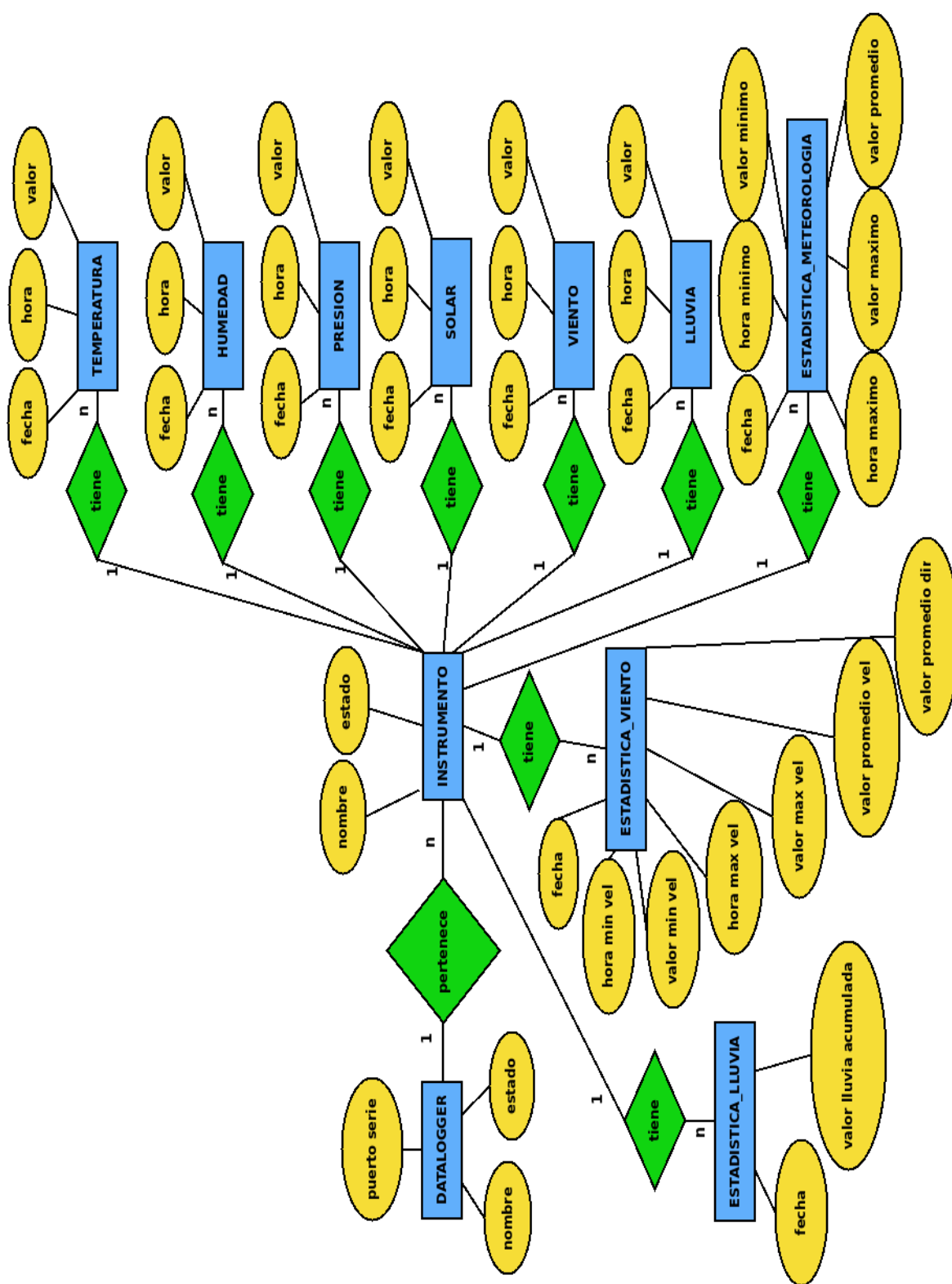


Ilustración 3.1: Diagrama Entidad-Relación para el sistema

3.1.2. Descripción de entidades, relaciones y atributos

Las entidades y atributos identificados son las siguientes;

1. **DATALOGGER:** es el dispositivo al que se conectan los diferentes instrumentos de medición. La computadora se conecta al datalogger a través del puerto serial para intercambiar la información meteorológica. Los atributos de esta entidad son: *nombre, puerto serie y estado*.
2. **INSTRUMENTO:** es el dispositivo que mide una variable meteorológica. Un instrumento tiene definido un conjunto de características para su funcionamiento correcto. Un instrumento se conecta siempre a un datalogger. Los atributos de esta entidad son: *nombre y estado*.
3. **TEMPERATURA:** es el valor que registra un instrumento que mide la temperatura ambiente. Cada valor se asocia con su fecha y hora de captura. Los atributos de esta entidad son: *fecha, hora y valor de temperatura*.
4. **HUMEDAD:** es el valor que registra un instrumento que mide la humedad relativa. Cada valor se asocia con su fecha y hora de captura. Los atributos de esta entidad son: *fecha, hora y valor de humedad relativa*.
5. **PRESION:** es el valor que registra un instrumento que mide la presión atmosférica. Cada valor se asocia con su fecha y hora de captura. Los atributos de esta entidad son: *fecha, hora y valor de presión atmosférica*.
6. **SOLAR:** es el valor que registra un instrumento que mide la radiación solar. Cada valor se asocia con su fecha y hora de captura. Los atributos de esta entidad son: *fecha, hora y valor de radiación solar*.
7. **VIENTO:** son los valores de velocidad y dirección que mide un sensor del viento (anemómetro). Ambos valores se asocian con su fecha y hora de captura. Los atributos de esta entidad son: *fecha, hora, valor de velocidad y valor de dirección*.
8. **LLUVIA:** es el valor que registra un instrumento que mide la precipitación pluvial. Cada valor se asocia con su fecha y hora de captura. Los atributos de esta entidad son: *fecha, hora y valor de precipitación pluvial*.
9. **ESTADISTICA_METEOROLOGIA:** son los valores estadísticos obtenidos a partir de todas las mediciones hechas por un instrumento durante un día. Los valores son el máximo, mínimo y promedio. Cada valor se asocia con su fecha y hora de ocurrencia. Los atributos de esta entidad son: *fecha, hora del mínimo, valor del mínimo, hora del máximo, valor del máximo y valor del promedio*.
10. **ESTADISTICA_VIENTO:** son los valores estadísticos obtenidos a partir de todas las mediciones hechas por el instrumento que mide el viento (anemómetro) durante un día. Los valores son el máximo, mínimo y promedio para la velocidad y la dirección del viento. Cada valor se asocia con su fecha y hora de ocurrencia. Los atributos para esta entidad son: *fecha, hora del mínimo de velocidad del viento, valor del mínimo de la velocidad del viento, hora del máximo de velocidad del viento, valor del máximo de la velocidad del viento, promedio de la velocidad del viento y promedio de la dirección del viento*.

11. **ESTADISTICA_LLUVIA**: es el valor del total de precipitación pluvial que se registró durante un día. Cada valor se asocia con su fecha de ocurrencia. Los atributos de esta entidad son: *fecha* y *valor de la lluvia acumulada durante el día*.

Las relaciones identificadas se describen a continuación:

1. Un DATALOGGER le **pertenece uno o más** INSTRUMENTO(s). Cada INSTRUMENTO **tiene a uno y solo un** DATALOGGER.
2. Un INSTRUMENTO le **pertenece uno o más** (mediciones de) TEMPERATURA. Cada (medición de) TEMPERATURA **tiene uno y solo un** INSTRUMENTO.
3. Un INSTRUMENTO le **pertenece uno o más** (mediciones de) HUMEDAD. Cada (medición de) HUMEDAD **tiene uno y solo un** INSTRUMENTO.
4. Un INSTRUMENTO le **pertenece uno o más** (mediciones de) PRESION. Cada (medición de) PRESION **tiene uno y solo un** INSTRUMENTO.
5. Un INSTRUMENTO le **pertenece uno o más** (mediciones de) SOLAR. Cada (medición de) SOLAR **tiene uno y solo un** INSTRUMENTO.
6. Un INSTRUMENTO le **pertenece uno o más** (mediciones de) VIENTO. Cada (medición de) VIENTO **tiene uno y solo un** INSTRUMENTO.
7. Un INSTRUMENTO le **pertenece uno o más** (mediciones de) LLUVIA. Cada (medición de) LLUVIA **tiene uno y solo un** INSTRUMENTO.
8. Un <INSTRUMENTO le **pertenece uno o más** (mediciones de) ESTADISTICA_METEORLOGIA. Cada (medición de) ESTADISTICA_METEORLOGIA **tiene uno y solo un** INSTRUMENTO.
9. Un INSTRUMENTO le **pertenece uno o más** (mediciones de) ESTADISTICA_VIENTO. Cada (medición de) ESTADISTICA_VIENTO **tiene uno y solo un** INSTRUMENTO.
10. Un INSTRUMENTO le **pertenece uno o más** (mediciones de) ESTADISTICA_LLUVIA. Cada (medición de) ESTADISTICA_LLUVIA **tiene uno y solo un** INSTRUMENTO.

3.1.3. Diseño lógico de la base de datos

Una vez establecido el modelo conceptual del problema o situación, el diseño lógico de los datos permite que estos se puedan representar usando de manera eficiente posibles recursos para estructurar datos y modelar restricciones disponibles en el modelo lógico. El objetivo es convertir el esquema conceptual de datos en un esquema lógico que se ajuste al gestor de la base de datos que va a ser utilizado (SGBD). Para escenificar esta situación se tomará el Modelo Relacional cuyo esquema relacional es trabajado por muchos SGBD comerciales.

Según el modelo relacional el elemento fundamental se le conoce como **relación**. Hay que tener en cuenta la diferencia de la palabra **relación** en ambos modelos, en el Modelo Relacional es una tabla mientras que en el modelo Entidad-Relación es la asociación que se produce entre dos entidades.

Una relación en el Modelo Relacional se define de la siguiente forma:

<nombre_relacion> (<atributo 1>, <atributo 2>, ...)

Donde el atributo clave principal aparece subrayado y donde también se señala, de alguna forma, cuáles son claves foráneas (por ejemplo, con un *)

Las relaciones para el sistema son:

1. <Datalogger> (id_dl, nombre_dl, puerto_dl, ES_ACTIVADO_DL)
2. <Instrumento> (*id_dl, id_inst, nombre_inst, ES_ACTIVADO_INST)
3. <Temperatura> (*id_inst, id_temp, fecha_temp, hora_temp, valor_temp)
4. <Humedad> (*id_inst, id_hum, fecha_hum, hora_hum, valor_hum)
5. <Presion> (*id_inst, id_presion, fecha_presion, hora_presion, valor_presion)
6. <Solar> (*id_inst, id_solar, fecha_solar, hora_solar, valor_solar)
7. <Viento> (*id_inst, id_viento, fecha_viento, hora_viento, valor_vel, valor_dir)
8. <Lluvia> (*id_inst, id_lluvia, fecha_lluvia, hora_lluvia, valor_lluvia)
9. <EstadisticaMeteorologia> (*id_inst, id_estmet, fecha_estmet, hora_min, valor_min, hora_max, valor_max, promedio)
10. <EstadisticaViento> (*id_inst, id_est_viento, fecha_est_viento, hora_vel_min, valor_vel_min, hora_vel_max, valor_vel_max, promedio_vel, promedio_dir)
11. <EstadisticaLluvia> (*id_inst, id_est_lluvia, fecha_est_lluvia, valor_acum_lluvia)

3.1.4. Normalización de la base de datos

En el desarrollo del diseño lógico se obtiene una serie de tablas finales que son las candidatas a formar la base de datos. Sin embargo, las tablas han sido obtenidas a partir de un diseño conceptual elaborado sin ningún tipo de reglas, por lo que se puede obtener un diseño de tablas más o menos heterogéneo.

La teoría de la normalización consiste en un conjunto de reglas formales que permiten asegurar que un diseño lógico cumple una serie de propiedades, corrigiendo la estructura de los datos de las tablas y evitando una serie de problemas como:

- Incapacidad de almacenar ciertos hechos.
- Redundancias y posibilidad de inconsistencias.
- Ambigüedades.
- Pérdida de información.
- Anomalías de inserción, borrado y modificación.

Las reglas formales de la teoría de la normalización son conocidas como **Formas Normales**. Existen seis formas normales, pero generalmente solo se aplican sobre las bases de datos las tres primeras, considerando que una base de datos que está en 3FN es una base de datos correctamente diseñada.

Las 3 reglas de la normalización básica son:

- **1FN:** La tabla contiene una clave primaria y los atributos son atómicos.
- **2FN:** Esta en 1FN y todos los atributos (no claves) dependen por completo de la clave primaria (dependencia funcional).
- **3FN:** Esta en 2FN y cada atributo que no forma parte de ninguna clave, depende directamente y no transitivamente, de la clave primaria.

Estas formas normales fueron aplicadas al conjunto de tablas pertenecientes al diseño lógico que se obtuvo en la sección anterior. En la tabla 3.1 se muestra el resumen de la aplicación de las 3 reglas de normalización a la tabla *Datalogger*.

		Datalogger			
		♦ <u>id_dl</u>	<u>int</u>		
		• nombre_dl	varchar		
		• puerto_dl	varchar		
		• ES ACTIVADO DL	boolean		
1FN		2FN		3FN	
Clave primaria	SI	1FN	SI	2FN	SI
Valores atómicos	SI	Dependencia Funcional	SI	Dependencias Transitivas	NO

Tabla 3.1: Normalización de la tabla Datalogger

De manera similar a la tabla *Datalogger*, en la tabla 3.2 se muestra el resumen de la normalización para la tabla *Instrumento*.

		Instrumento			
		<u>•id_inst</u> int •nombre_inst varchar •ES ACTIVADO INST boolean			
1FN		2FN		3FN	
Clave primaria	SI	1FN	SI	2FN	SI
Valores atómicos	SI	Dependencia Funcional	SI	Dependencias Transitivas	NO

Tabla 3.2: Normalización de la tabla Instrumento

Se puede observar que las tablas *Temperatura*, *Humedad*, *Presion*, *Solar*, *Viento* y *Lluvia* tiene una estructura similar y la misma situación se observa en las tablas *EstadisticaMeteorologia*, *EstadisticaViento* y *EstadisticaLluvia*. Bajo esta situación, se puede deducir que si alguna de las tablas cumple con las formas normales, las demás también las cumplirán de manera automática.

Aplicando las deducciones anteriores, en la tabla 3.3 y 3.4 se muestra el resumen de la normalización para las tablas *Temperatura* y *EstadisticaMeteorología*.

		Temperatura			
		<u>•id_temp</u> int •fecha_temp DATE •hora_temp TIME •valor temp DOUBLE			
1FN		2FN		3FN	
Clave primaria	SI	1FN	SI	2FN	SI
Valores atómicos	SI	Dependencia Funcional	SI	Dependencias Transitivas	NO

Tabla 3.3: Normalización de la tabla Temperatura

		EstadisticaMeteorologia ♦ <u>id_estmet</u> int • fecha_estmet DATE • hora_min TIME ° valor_min DOUBLE ° hora_max TIME ° valor_max DOUBLE ° promedio DOUBLE			
1FN		2FN		3FN	
Clave primaria	SI	1FN	SI	2FN	SI
Valores atómicos	SI	Dependencia Funcional	SI	Dependencias Transitivas	NO

Tabla 3.4: Normalización de la tabla EstadisticaMeteorología

3.1.5. Implementación del modelo relacional usando MySQL Workbench

MySQL Workbench es una aplicación para el diseño y documentación de bases de datos con el sistema de gestión de bases de datos MySQL. Existen dos versiones del producto, una es open source y la otra es una versión comercial. La versión comercial proporciona algunas funcionalidades que pueden resultar de interés en algún ámbito, aunque la versión open source es más que suficiente para implementar el modelo relacional de una base de datos.

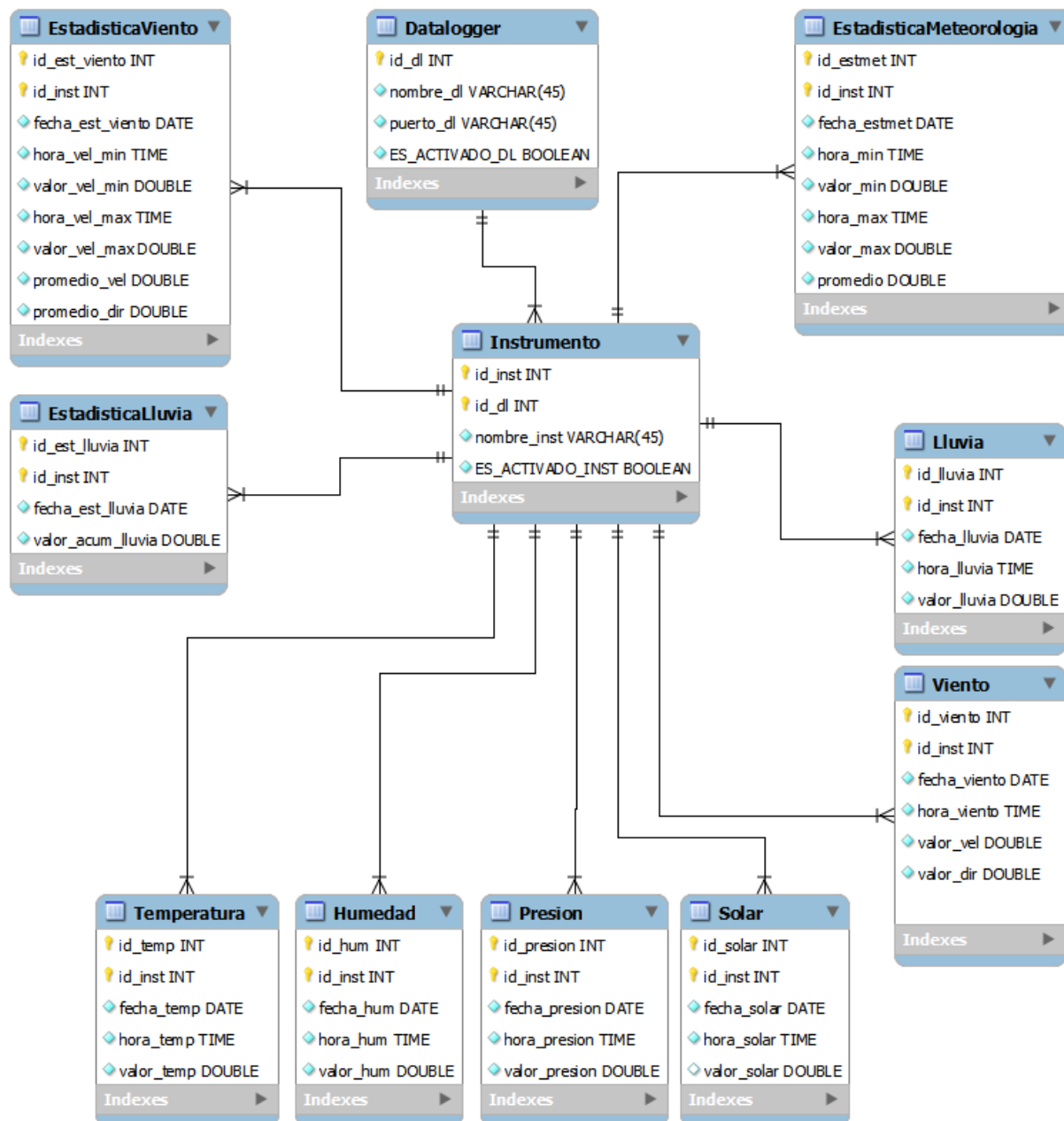


Ilustración 3.2: Modelo relacional para el sistema realizado en MySQL Workbench

Con esta herramienta se puede elaborar una representación visual de las tablas, vistas, procedimientos almacenados y claves foráneas de la base de datos. Además, es capaz de sincronizar el modelo en desarrollo con la base de datos real. Se puede realizar una ingeniería directa e ingeniería inversa para exportar e importar el esquema de una base de datos ya existente. MySQL Workbench puede generar también el script necesario para crear la base de datos que se ha representado en el modelo relacional. Con el uso de la herramienta MySQL Workbench se han implementado las relaciones del modelo relacional perteneciente al sistema desarrollado. En la ilustración 3.2 se muestra el diagrama que representa el diseño lógico de la base de datos del sistema.

CAPÍTULO 4

IMPLEMENTACIÓN Y

PRUEBAS DEL

SISTEMA

4.1. IMPLEMENTACIÓN

En este capítulo se presentara la implementación del sistema basándose en lo obtenido en la etapa del análisis y diseño tanto del sistema como de la base de datos.

El lenguaje de programación base para codificar los procesos del servidor fue Java. Con este lenguaje se implementaron los procesos de adquisición, procesamiento y almacenamiento de los datos meteorológicos. También se utilizó para implementar las operaciones en la base de datos y los procesos para recibir las solicitudes del cliente y elaborar su respectiva respuesta.

4.1.1. Implementación de los procesos del servidor

Los procesos pertenecientes al servidor se implementaron a través del uso de servlets, que es un tipo de aplicación Java que aprovecha la comunicación vía protocolo HTTP, el mismo protocolo que usan las páginas web que hay en la Internet.

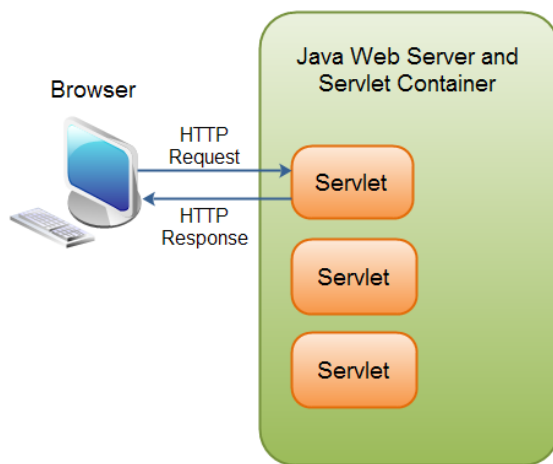


Ilustración 4.1: Esquema básico del funcionamiento de un servlet

En la ilustración 4.1 se muestra el esquema básico de un servlet. Los servlets son componentes del servidor, se instalan en un contenedor de servlets y pueden ser ejecutados en cualquier plataforma debido a la tecnología Java que se usa para implementarlos. Los servlets incrementan la funcionalidad de una aplicación web. Se cargan de forma dinámica por el entorno de ejecución Java del servidor cuando se necesitan. Cuando se recibe una petición del cliente, el contenedor/servidor web inicia el servlet requerido. El servlet procesa la petición del cliente y envía la respuesta de vuelta al contenedor/servidor, y éste la redirige al cliente.

Los servlets realizan procesamientos, que puede incluir la interacción con una base de datos u otros componentes del lado del servidor. Los resultados devueltos al cliente se hacen comúnmente en forma de documentos HTML y que pueden ser mostrados en un

navegador web, aunque también pueden devolver otros formatos, como imágenes y datos binarios.

4.1.2. Implementación de los procesos del cliente

La implementación de los procesos para el cliente se realizó en el lenguaje Java, principalmente para la presentación de los datos meteorológicos recién adquiridos por la estación meteorológica.

Los servlets son aplicaciones muy poderosas; pero carecen de una interfaz propia para interaccionar con el cliente (*front-end*). Para resolver esta carencia se hizo uso de un tipo de aplicación Java llamada *applet*.

El applet es un programa que puede incrustarse en un documento HTML; es decir en una página web. Cuando un navegador carga una página web que contiene un applet, éste se descarga en el navegador web y comienza a ejecutarse. El applet es una buena alternativa para presentar contenidos de forma atractiva al cliente, ya que permite el uso de imágenes, animaciones y/o video.

Se utilizó el lenguaje HTML para la creación de las páginas web y el formulario utilizado para realizar la consulta de datos meteorológicos adquiridos en días anteriores.

El formulario HTML también es una buena alternativa para enviar información a un servlet, ya que permiten el uso de una gran variedad de componentes llamados **controles** (etiquetas de texto, casillas de verificación y selección, menús desplegables, casillas de captura de texto, botones, etc.) para recolectar datos o valores a través de una página web. Los valores son enviados al servlet cuando el usuario presiona el botón de **Enviar** (*Submit*). El servlet recibe los valores, los procesa y elabora una respuesta, que suele ser comúnmente un documento HTML.

4.1.3. Software utilizado para la implementación del sistema

a) Ambiente de desarrollo integrado Netbeans

Para la codificación de los procesos en lenguaje Java se utilizó **Netbeans** [E13], que es un ambiente de desarrollo integrado o **IDE** (sigla en inglés de *Integrated Development Environment*) que ofrece un amplio conjunto de módulos para la elaboración eficiente de proyectos de software en diferentes lenguajes de programación (*Java, HTML, PHP, C*, etc).

b) API's de terceros utilizadas

Una de las ventajas que ofrece el lenguaje Java es la existencia de las *API* (del inglés *Application Programming Interface*). La interfaz de programación de aplicaciones es el conjunto de funciones y procedimientos para efectuar toda clase de tareas necesarias dentro de un programa. En la implementación del sistema se hace uso de dos API's de terceros para realizar dos procesos importantes: la comunicación vía puerto serial y la elaboración de las gráficas para representar los datos meteorológicos.

- **Giovinet Driver [E14]** es una API programada en Java que contiene las funciones y procedimientos necesarios para establecer una comunicación vía puerto serial entre un PC y otros dispositivos. Esta API trabaja en sistemas Windows y Linux y se ofrece una versión libre para uso no comercial.
- **JFreeChart [E15]** es una API libre y gratuita hecha en Java que permite de manera fácil la creación de gráficas de datos con calidad profesional. Ofrece una amplia variedad de tipos de gráficas y formatos gráficos para exportar las gráficas. Se integra bien a las aplicaciones Java tanto del lado del servidor como del cliente.

c) Contenedor de servlets Apache Tomcat

Los servlets desarrollados requieren de un contenedor para que puedan ser ejecutados, para la implementación del sistema se hace uso del servidor web Apache Tomcat.

La instalación del servidor web Apache Tomcat varía según el sistema operativo, en este caso se utiliza el sistema operativo *Windows*. En el sitio oficial de Apache Tomcat [E11] es posible descargar el software de instalación de Apache Tomcat para Windows.

Para el desarrollo del sistema se han elegido las características por omisión que se ofrecen en los procesos de instalación y solo se definió la contraseña de seguridad para la administración del servidor. Finalmente, se debe comprobar que el servidor web esté en funcionamiento. Una manera de hacerlo es usando un navegador web e ingresar la URL:

http://localhost:8080

El navegador deberá mostrar un contenido similar al que se muestra en la ilustración 4.2.

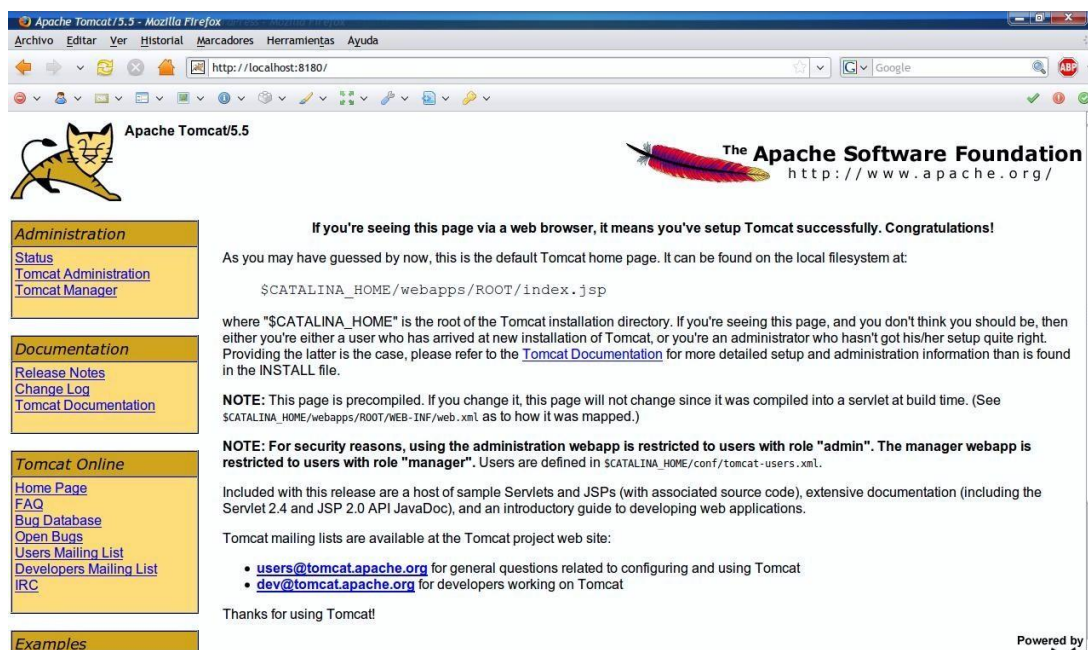


Ilustración 4.2: Pantalla de bienvenida del servidor web Apache Tomcat

d) El SGBD MySQL y la creación de la base de datos

El sistema gestor de base de datos elegidos para el sistema fue MySQL. El instalador del SGBD MySQL para el sistema operativo Windows se obtuvo de la página oficial de MySQL [E16]. Para el desarrollo del sistema se ha elegido las características por omisión que ofrece el proceso de instalación y solo se ha ingresado el *password de seguridad* para la administración del SGBD.

Para la creación de la base de datos del sistema se ha aprovechado la herramienta *MySQL Workbench* para obtener el script a partir del diagrama del modelo relacional del sistema. El script contiene las instrucciones para construir la base de datos y las tablas que la conforman.

A continuación se muestra el contenido del script.

```
SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='TRADITIONAL';

CREATE SCHEMA IF NOT EXISTS `mydb_estmet` DEFAULT CHARACTER SET
latin1 COLLATE latin1_spanish_ci ;
USE `mydb_estmet` ;

-----
-- Table `mydb_estmet`.`Datalogger`
-----
CREATE TABLE IF NOT EXISTS `mydb_estmet`.`Datalogger` (
  `id_dl` INT NOT NULL AUTO_INCREMENT,
  `nombre_dl` VARCHAR(45) NOT NULL,
  `puerto_dl` VARCHAR(45) NOT NULL,
  `ES_ACTIVADO_DL` TINYINT(1) NOT NULL,
  PRIMARY KEY (`id_dl`)
ENGINE = InnoDB;

-----
-- Table `mydb_estmet`.`Instrumento`
-----
CREATE TABLE IF NOT EXISTS `mydb_estmet`.`Instrumento` (
  `id_inst` INT NOT NULL AUTO_INCREMENT,
  `id_dl` INT NOT NULL,
  `nombre_inst` VARCHAR(45) NOT NULL,
  `ES_ACTIVADO_INST` TINYINT(1) NOT NULL,
  PRIMARY KEY (`id_inst`, `id_dl`),
  INDEX `fk_instrumento_datalogger` (`id_dl` ASC),
  CONSTRAINT `fk_instrumento_datalogger`
    FOREIGN KEY (`id_dl`)
      REFERENCES `mydb_estmet`.`Datalogger` (`id_dl`)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION)
ENGINE = InnoDB;

-----
-- Table `mydb_estmet`.`Temperatura`
-----
CREATE TABLE IF NOT EXISTS `mydb_estmet`.`Temperatura` (
  `id_temp` INT NOT NULL AUTO_INCREMENT,
  `id_inst` INT NOT NULL,
  `fecha_temp` DATE NOT NULL,
  `hora_temp` TIME NOT NULL,
  `valor_temp` DOUBLE NOT NULL,
  PRIMARY KEY (`id_temp`, `id_inst`),
  INDEX `fk_temperatura_instrumento1` (`id_inst` ASC),
  CONSTRAINT `fk_temperatura_instrumento1`
    FOREIGN KEY (`id_inst`)
      REFERENCES `mydb_estmet`.`Instrumento` (`id_inst`)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION)
ENGINE = InnoDB;

-----
-- Table `mydb_estmet`.`Humedad`
-----
CREATE TABLE IF NOT EXISTS `mydb_estmet`.`Humedad` (
  `id_hum` INT NOT NULL AUTO_INCREMENT,
  `id_inst` INT NOT NULL,
  `fecha_hum` DATE NOT NULL,
  `hora_hum` TIME NOT NULL,
  `valor_hum` DOUBLE NOT NULL,
  PRIMARY KEY (`id_hum`, `id_inst`),
  INDEX `fk_humedad_instrumento1` (`id_inst` ASC),
  CONSTRAINT `fk_humedad_instrumento1`
    FOREIGN KEY (`id_inst`)
      REFERENCES `mydb_estmet`.`Instrumento` (`id_inst`)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION)
ENGINE = InnoDB;

-----
-- Table `mydb_estmet`.`Presion`
-----
CREATE TABLE IF NOT EXISTS `mydb_estmet`.`Presion` (
  `id_presion` INT NOT NULL AUTO_INCREMENT,
  `id_inst` INT NOT NULL,
  `fecha_presion` DATE NOT NULL,
  `hora_presion` TIME NOT NULL,
  `valor_presion` DOUBLE NOT NULL,
  PRIMARY KEY (`id_presion`, `id_inst`),
  INDEX `fk_presion_instrumento1` (`id_inst` ASC),
  CONSTRAINT `fk_presion_instrumento1`
    FOREIGN KEY (`id_inst`)
      REFERENCES `mydb_estmet`.`Instrumento` (`id_inst`))
```

```

ON DELETE NO ACTION
ON UPDATE NO ACTION)
ENGINE = InnoDB;

-----
-- Table `mydb_estmet`.`Solar`
-----
CREATE TABLE IF NOT EXISTS `mydb_estmet`.`Solar` (
  `id_solar` INT NOT NULL AUTO_INCREMENT ,
  `id_inst` INT NOT NULL ,
  `fecha_solar` DATE NOT NULL ,
  `hora_solar` TIME NOT NULL ,
  `valor_solar` DOUBLE NULL ,
  PRIMARY KEY (`id_solar`, `id_inst`),
  INDEX `fk_solar_instrumento1` (`id_inst` ASC),
  CONSTRAINT `fk_solar_instrumento1`
    FOREIGN KEY (`id_inst` )
    REFERENCES `mydb_estmet`.`Instrumento` (`id_inst` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

-----
-- Table `mydb_estmet`.`Viento`
-----
CREATE TABLE IF NOT EXISTS `mydb_estmet`.`Viento` (
  `id_viento` INT NOT NULL AUTO_INCREMENT ,
  `id_inst` INT NOT NULL ,
  `fecha_viento` DATE NOT NULL ,
  `hora_viento` TIME NOT NULL ,
  `valor_vel` DOUBLE NOT NULL ,
  `valor_dir` DOUBLE NOT NULL ,
  PRIMARY KEY (`id_viento`, `id_inst`),
  INDEX `fk_viento_instrumento1` (`id_inst` ASC),
  CONSTRAINT `fk_viento_instrumento1`
    FOREIGN KEY (`id_inst` )
    REFERENCES `mydb_estmet`.`Instrumento` (`id_inst` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

-----
-- Table `mydb_estmet`.`Lluvia`
-----
CREATE TABLE IF NOT EXISTS `mydb_estmet`.`Lluvia` (
  `id_lluvia` INT NOT NULL AUTO_INCREMENT ,
  `id_inst` INT NOT NULL ,
  `fecha_lluvia` DATE NOT NULL ,
  `hora_lluvia` TIME NOT NULL ,
  `valor_lluvia` DOUBLE NOT NULL ,
  PRIMARY KEY (`id_lluvia`, `id_inst`),
  INDEX `fk_lluvia_instrumento1` (`id_inst` ASC),
  CONSTRAINT `fk_lluvia_instrumento1`
    FOREIGN KEY (`id_inst` )
    REFERENCES `mydb_estmet`.`Instrumento` (`id_inst` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

-----
-- Table `mydb_estmet`.`EstadisticaMeteorologia`
-----
CREATE TABLE IF NOT EXISTS `mydb_estmet`.`EstadisticaMeteorologia` (
  `id_estmet` INT NOT NULL AUTO_INCREMENT ,
  `id_inst` INT NOT NULL ,
  `fecha_estmet` DATE NOT NULL ,
  `hora_min` TIME NOT NULL ,
  `valor_min` DOUBLE NOT NULL ,
  `hora_max` TIME NOT NULL ,
  `valor_max` DOUBLE NOT NULL ,
  `promedio` DOUBLE NOT NULL ,
  PRIMARY KEY (`id_estmet`, `id_inst`),
  INDEX `fk_estadisticacomun_instrumento1` (`id_inst` ASC),
  CONSTRAINT `fk_estadisticacomun_instrumento1`
    FOREIGN KEY (`id_inst` )
    REFERENCES `mydb_estmet`.`Instrumento` (`id_inst` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

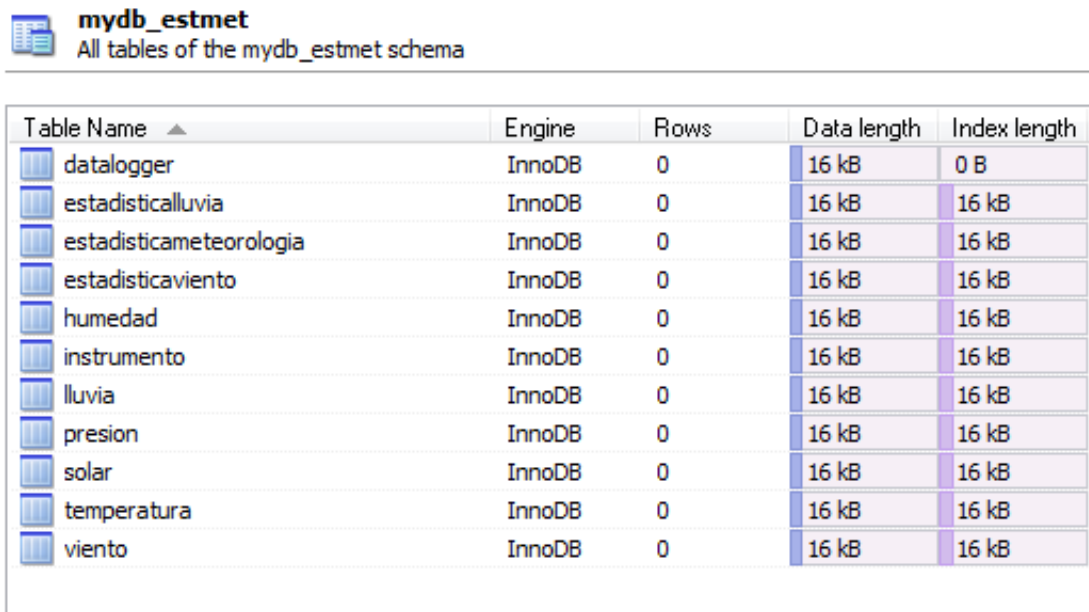
-----
-- Table `mydb_estmet`.`EstadisticaViento`
-----
CREATE TABLE IF NOT EXISTS `mydb_estmet`.`EstadisticaViento` (
  `id_est_viento` INT NOT NULL AUTO_INCREMENT ,
  `id_inst` INT NOT NULL ,
  `fecha_est_viento` DATE NOT NULL ,
  `hora_vel_min` TIME NOT NULL ,
  `valor_vel_min` DOUBLE NOT NULL ,
  `hora_vel_max` TIME NOT NULL ,
  `valor_vel_max` DOUBLE NOT NULL ,
  `promedio_vel` DOUBLE NOT NULL ,
  `promedio_dir` DOUBLE NOT NULL ,
  PRIMARY KEY (`id_est_viento`, `id_inst`),
  INDEX `fk_estadisticaviento_instrumento1` (`id_inst` ASC),
  CONSTRAINT `fk_estadisticaviento_instrumento1`
    FOREIGN KEY (`id_inst` )
    REFERENCES `mydb_estmet`.`Instrumento` (`id_inst` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

-----
-- Table `mydb_estmet`.`EstadisticaLluvia`
-----
CREATE TABLE IF NOT EXISTS `mydb_estmet`.`EstadisticaLluvia` (
  `id_est_lluvia` INT NOT NULL AUTO_INCREMENT ,
  `id_inst` INT NOT NULL ,
  `fecha_est_lluvia` DATE NOT NULL ,
  `valor_acum_lluvia` DOUBLE NOT NULL ,
  PRIMARY KEY (`id_est_lluvia`, `id_inst`),
  INDEX `fk_estadisticalluvia_instrumento1` (`id_inst` ASC),
  CONSTRAINT `fk_estadisticalluvia_instrumento1`
    FOREIGN KEY (`id_inst` )
    REFERENCES `mydb_estmet`.`Instrumento` (`id_inst` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;

```

La ejecución del script se realiza a través del uso de la consola de MySQL y del comando *source*. El parámetro que requiere el comando es la ubicación donde se almacena el script. La ilustración 4.3 muestra las tablas producidas al momento de ejecutar el script.



The screenshot displays the MySQL Workbench interface for the 'mydb_estmet' database. The title bar reads 'mydb_estmet All tables of the mydb_estmet schema'. Below the title bar is a table listing the database's tables. The table has five columns: 'Table Name', 'Engine', 'Rows', 'Data length', and 'Index length'. There are 12 rows of data, each representing a table in the database. All tables are using the 'InnoDB' engine and currently contain 0 rows. Most tables have a data length of 16 kB and an index length of 16 kB, while the 'datalogger' table has an index length of 0 B.

Table Name ▲	Engine	Rows	Data length	Index length
datalogger	InnoDB	0	16 kB	0 B
estadisticalluvia	InnoDB	0	16 kB	16 kB
estadisticameteorologia	InnoDB	0	16 kB	16 kB
estadisticaviento	InnoDB	0	16 kB	16 kB
humedad	InnoDB	0	16 kB	16 kB
instrumento	InnoDB	0	16 kB	16 kB
lluvia	InnoDB	0	16 kB	16 kB
presion	InnoDB	0	16 kB	16 kB
solar	InnoDB	0	16 kB	16 kB
temperatura	InnoDB	0	16 kB	16 kB
viento	InnoDB	0	16 kB	16 kB

Ilustración 4.3: Tablas de la base de datos del sistema

4.2. PANTALLAS DEL SISTEMA

4.2.1. Página web principal del sistema

A continuación se presentan las pantallas que muestra el sistema al cliente que se conecta al servidor web Apache Tomcat. La ilustración 4.4 se muestra la página principal del sistema y es en ésta donde se aprecia la publicación de los datos más recientes que ha obtenido la estación meteorológica. Esta página web está programada en lenguaje HTML y su diseño se apoya del uso de una hoja de estilos en cascada (*Cascading Style Sheets*, o sus siglas **CSS**) para brindar una interfaz gráfica agradable al usuario.

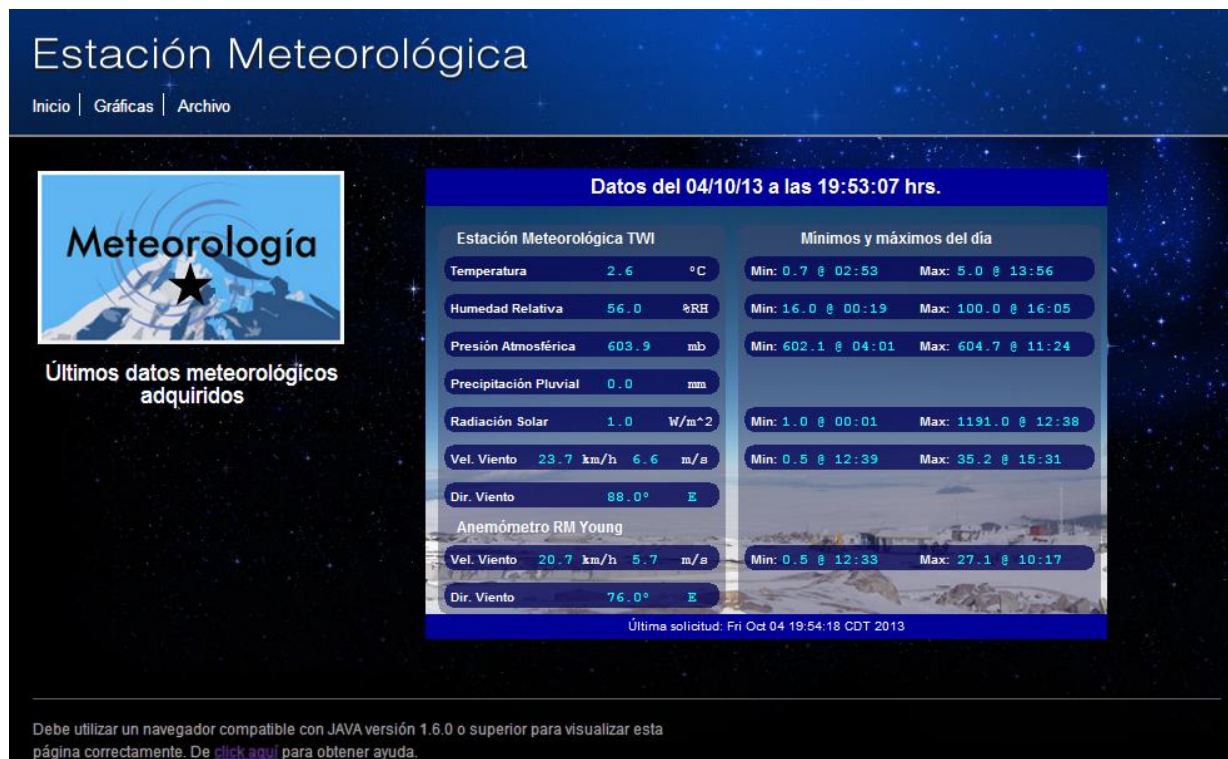


Ilustración 4.4: Página web principal del sistema

En la parte superior se ha colocado un menú para tener acceso a los diferentes servicios que ofrece el sistema. La opción *Inicio* da acceso a la página principal, la opción *Gráficas* da acceso a la consulta de las representaciones gráficas de todos los datos capturados en el día actual para cada una de las variables meteorológicas. La opción *Archivo* da acceso al formulario para consultar las representaciones gráficas de los datos capturados en días pasados y su respectiva estadística. En el cuerpo principal de la página web está colocado el applet desarrollado para mostrar los datos meteorológicos recientes; así como también los valores mínimos y máximos de algunas variables meteorológicas importantes. En la parte superior del applet se muestra la fecha y hora de captura de los datos meteorológicos. En la parte inferior se muestra la fecha y hora de la última solicitud enviada al servlet. El applet ha sido programado para realizar una solicitud de datos cada 10 segundos.

En la parte inferior de la página web se muestra un aviso para el usuario donde se le recuerda que es necesario tener instalado el plugin de la máquina virtual Java (en inglés *Java Virtual Machine, JVM*) en su navegador web y así poder ejecutar el applet. Finalmente, se da un acceso directo a la página <http://www.java.com> donde el usuario podrá obtener más información para descargar e instalar el plugin que le corresponde a su navegador.

4.2.2. Página web de la opción Gráficas

La ilustración 4.5 muestra un ejemplo del resultado de cuando el usuario elige del menú la opción *Gráficas*. El sistema mostrará las representaciones gráficas (gráficas de línea) de todas las variables meteorológicas que captura la estación. La página web cuenta con el mismo menú en la parte superior para tener siempre acceso a las demás opciones que ofrece el sistema.

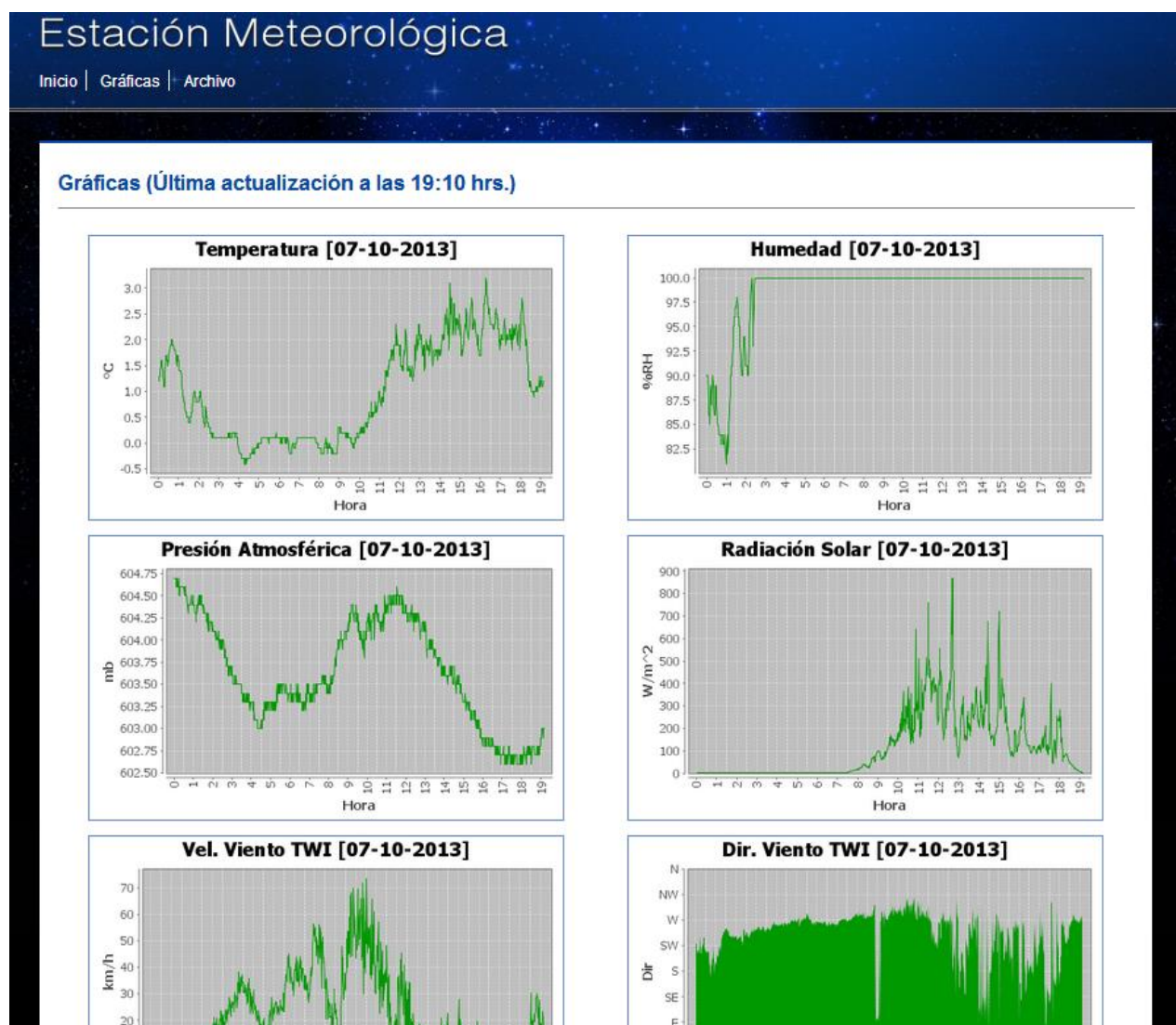


Ilustración 4.5: Página web para la opción "Gráficas"

La intención de esta página web es la de mostrar al usuario, de manera gráfica, el comportamiento actual del clima en el lugar donde se ubica la estación. La información mostrada es la más actual que ha registrado el sistema en el día y se está actualizando cada minuto. La página mostrará siempre la hora de la última actualización para que el usuario tenga una referencia más precisa sobre los datos que está consultando.

4.2.3. Página web de la opción Archivo

En la ilustración 4.6 se muestra la página web para la opción *Archivo* contiene un formulario HTML donde el usuario podrá ingresar los parámetros para realizar la consulta, de manera gráfica, de los datos meteorológicos y estadísticas que pertenecen a fechas pasadas.

El formulario ofrece al usuario el tipo de gráfica, ya sea de datos o estadística, después deberá definir la fecha que desea consultar. Finalmente, si la gráfica elegida es de datos, el usuario podrá definir un intervalo de tiempo (hora inicial y hora final) del día que desea consultar (los valores por omisión son de la 00:00 horas a las 23:59 horas). En esta parte del formulario solo se le permite al usuario establecer un intervalo no mínimo de 5 minutos. Una vez definidos los parámetros de la consulta, el usuario deberá proceder a pulsar el botón de *Enviar*. Los datos del formulario serán enviados a un servlet desarrollado para esta opción y éste le devolverá una respuesta al cliente.

The screenshot shows a web application titled "Estación Meteorológica" with a navigation bar containing "Inicio", "Gráficas", and "Archivo". On the left, there is a logo with the word "Meteorología" and a star, with the text "Gráficas de datos y estadísticas de días anteriores" below it. The main content area is a form titled "Elegir el tipo de gráfica" with two radio buttons: "Datos" (selected) and "Estadísticas". Below this is a section "Definir la fecha" with a date picker showing "Oct 7 2013". Another section "Definir el periodo de tiempo" includes a note "(Aplica solo a gráficas de datos)" and two input fields: "Hora Inicial" with "0:00" and "Hora Final" with "23:59". At the bottom of the form is an "Enviar" button.

Ilustración 4.6: Página web para la opción "Archivo"

4.2.3.1. Páginas web de respuesta de la opción *Gráficas*

En las ilustraciones 4.7 y 4.8 se muestran unos ejemplos de los resultados que puede obtener el usuario al realizar una consulta, tanto de datos como de estadística, de una fecha pasada.

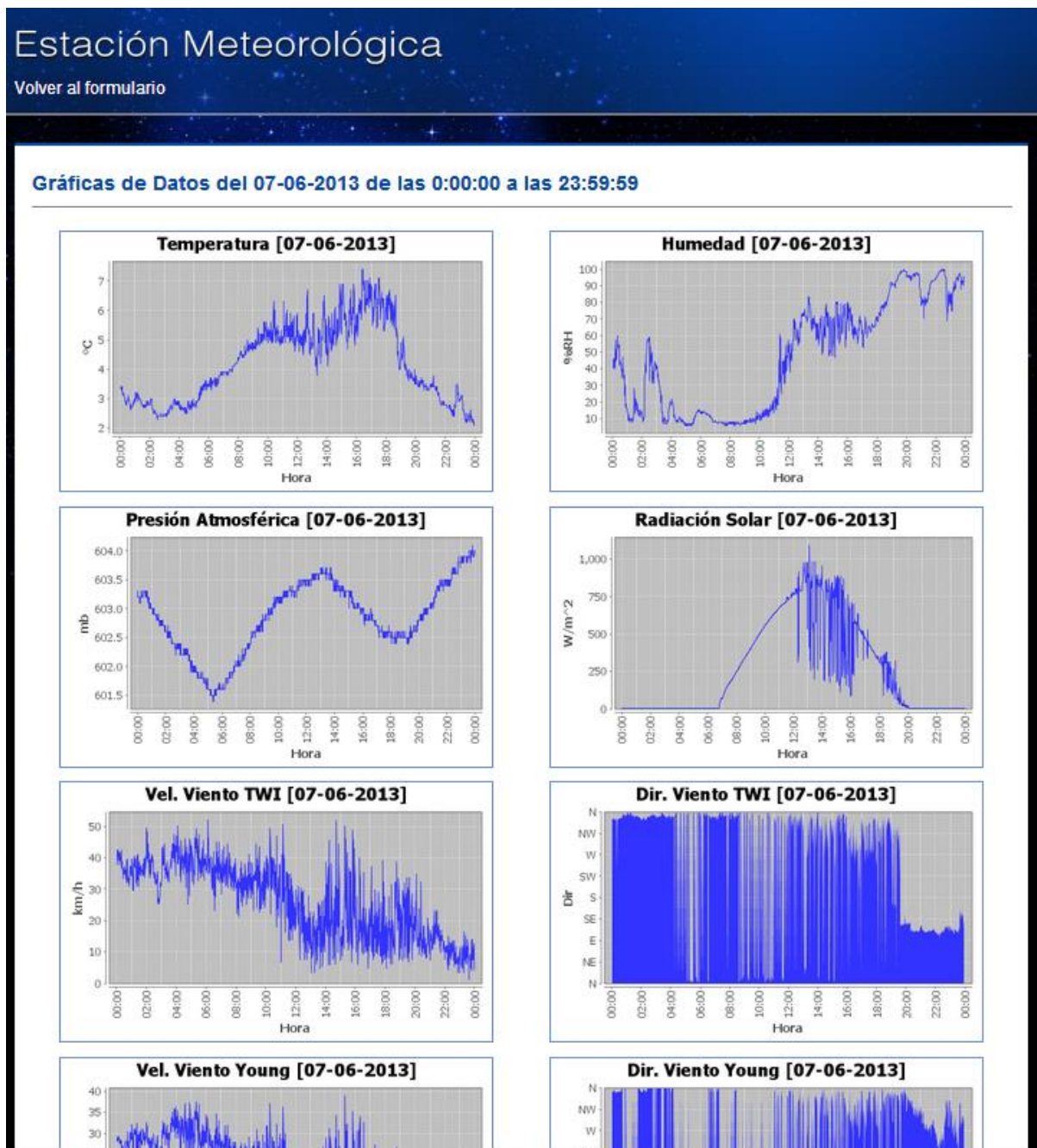


Ilustración 4.7: Ejemplo de una respuesta a una consulta de datos de una fecha pasada

Las respuestas a las consultas de datos de fechas pasadas incluyen un conjunto de gráficas de línea para cada variable meteorológica, además se agrega información sobre la fecha y el intervalo de tiempo elegido.

Estación Meteorológica

[Volver al formulario](#)

Gráficas de Estadística del 07-06-2013

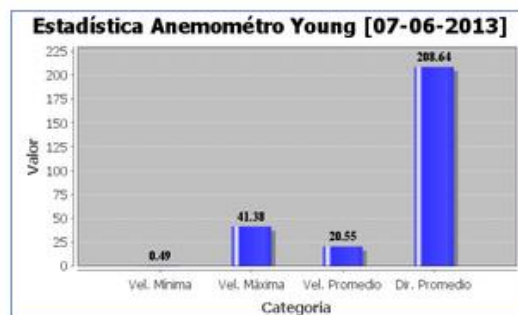
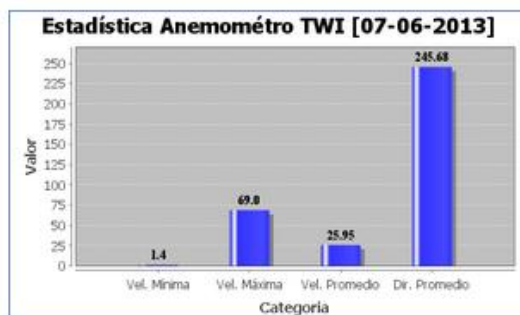
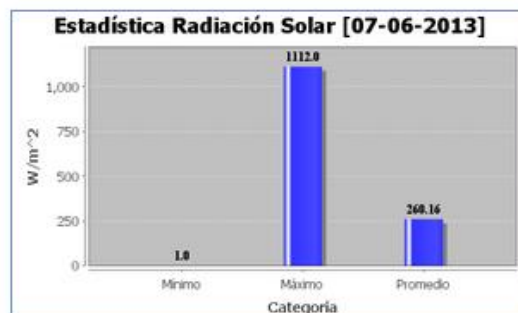
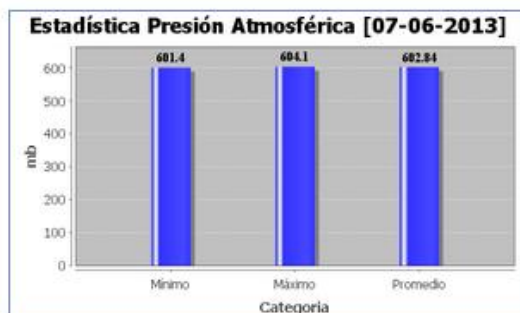
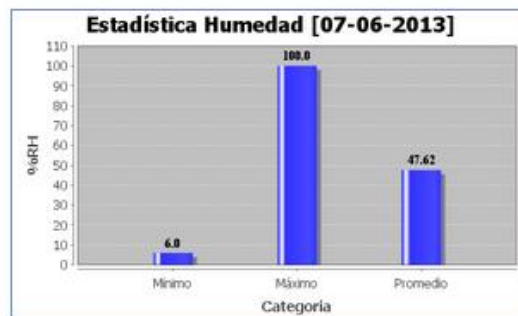
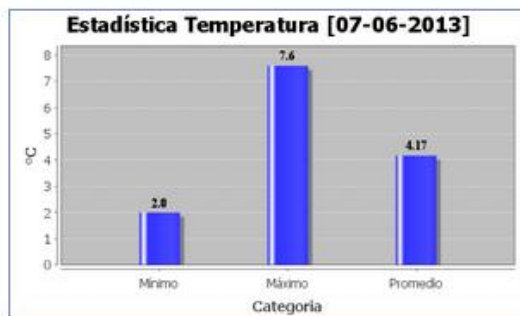


Ilustración 4.8: Ejemplo de una respuesta a una consulta de estadística de una fecha pasada

Las respuestas a las consultas de estadística de fechas pasadas incluyen un conjunto de gráficas de barras que muestran el mínimo, máximo y promedio de cada una de las variables meteorológicas.

4.3. PRUEBAS DEL SISTEMA

Las pruebas realizadas al sistema sirvieron para comprobar que cada uno de los módulos que conforman el sistema cumple con las especificaciones definidas en la etapa de diseño. A la vez, estas pruebas sirvieron para verificar el funcionamiento correcto de los métodos de manejo de errores implementados en la codificación del sistema.

Las pruebas se realizaron usando dos computadoras conectadas a una red local. Esta configuración permitió establecer, de manera mínima, la arquitectura cliente-servidor. En la computadora cliente se instaló y configuró el software necesario para ejecutar el sistema desarrollado. Para completar el esquema definido para las pruebas, se conectó vía puerto serie, el datalogger de la estación meteorológica. El puerto serie utilizado fue el COM4, el cual fue definido en la tabla **Datalogger** de la base de datos. El registro de esta información es utilizada por el sistema para abrir la comunicación entre la PC y el datalogger. La ilustración 4.9 muestra el resultado del proceso antes descrito.

id_dl	nombre_dl	puerto_dl	ES_ACTIVADO_DL
1	Datalogger Solus	COM4	1

Ilustración 4.9: Definición del puerto serie en la base de datos

Se observa en la ilustración 4.10 que los instrumentos de medición conectados al datalogger se definieron en la tabla **Instrumento**. Esta información es usada para establecer que valores de las variables meteorológicas serán registradas en la base de datos. En la ilustración 56 se muestra que instrumentos fueron habilitados para realizar las pruebas. La columna **ES_ACTIVADO_INST** se indica que instrumentos están habilitados, el valor 1 equivale a “activado” y el valor 0 a “desactivado”.

id_inst	id_dl	nombre_inst	ES_ACTIVADO_INST
1	1	Temperatura	1
2	1	Humedad Relativa	1
3	1	Presión Atmosférica	1
4	1	Precipitación Pluvial	1
5	1	Radiación Solar	1
6	1	Anemómetro TWI	1
7	1	Anemómetro Young	0

Ilustración 4.10: Definición de los instrumentos de medición en la base de datos

4.3.1. Pruebas a la página web principal del sistema y consulta de gráficas del día actual

Una vez definidos la forma de operar del sistema, se procedió a realizar las pruebas en la computadora destinada a actuar como cliente. En primer lugar, se realizó a visitar la página web principal del sistema, la cual presenta los últimos datos meteorológicos adquiridos por la estación.

En la ilustración 4.11 se muestra como el cliente establece una conexión con el servidor.



Ilustración 4.11: Pantalla Conectado con el servidor

La ilustración 4.12 se muestra el resultado de una conexión exitosa entre el cliente y el servidor. En esta prueba se verificó que la activación y desactivación de los instrumentos se cumpliera.



Ilustración 4.12: Pantalla Conexión exitosa entre el cliente y el servidor

Otra prueba consistió en detener las operaciones del servidor por un momento para verificar que se mostrará el mensaje donde se informa al cliente que los servicios dados por el servidor no están disponibles. El resultado de la prueba se muestra en la figura 4.13.



Ilustración 4.13: Pantalla Servidor fuera de servicio

También se comprobó que la página web que muestra las gráficas de día actual mostrará el respectivo mensaje cuando el servidor esta fuera de servicio. La pantalla se aprecia en la figura 4.14



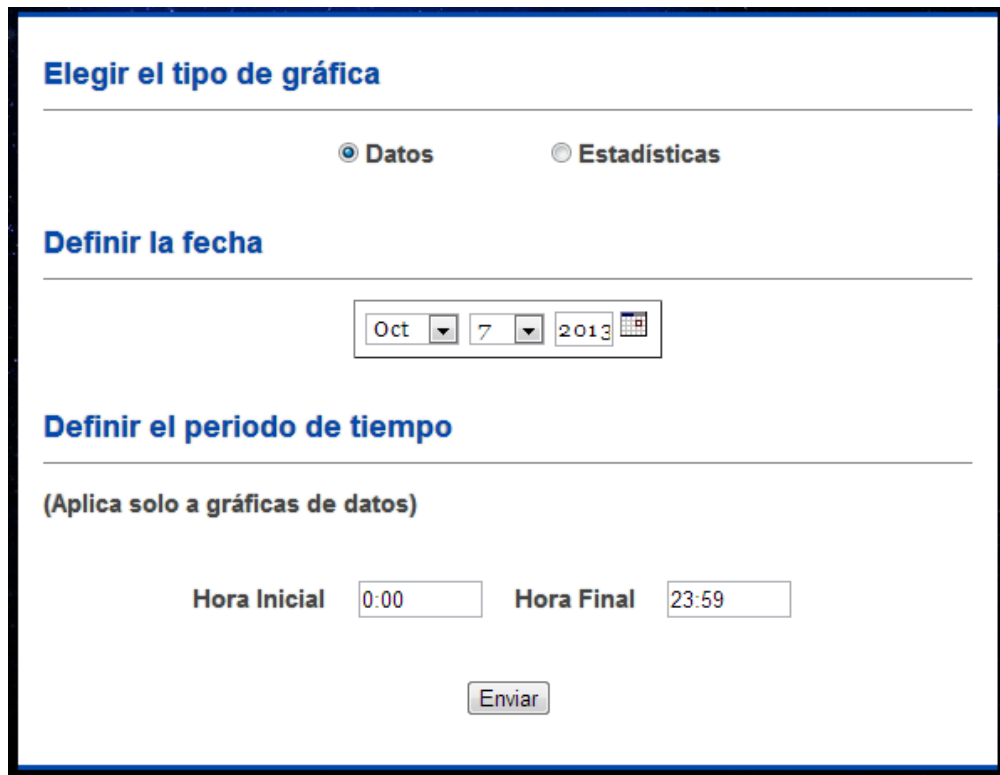
Ilustración 4.14: Pantalla Gráficas del día actual no disponibles

En los que respecta a las pruebas aplicadas a la presentación de los últimos datos meteorológicos adquiridos y la consulta de las gráficas del día actual, se pudo comprobar que el sistema se comportó de la manera en que se esperaba.

4.3.2. Pruebas al formulario para la consulta de gráficas de datos y estadísticas ocurridos en días anteriores

A continuación se presentan los resultados obtenidos de las pruebas aplicadas a la página web que se utiliza para consultar las gráficas de datos y estadísticas de los datos meteorológicos pertenecientes a fechas pasadas.

En la ilustración 4.15 se muestra el formulario que el usuario debe de usar para realizar una consulta. Las pruebas aplicadas al formulario se centran al manejo de errores y excepciones que puedan ocurrir al momento de introducir una fecha y la definición de un periodo de tiempo.



Elegir el tipo de gráfica

☒ Datos ☐ Estadísticas

Definir la fecha

Oct 7 2013

Definir el periodo de tiempo

(Aplica solo a gráficas de datos)

Hora Inicial 0:00 Hora Final 23:59

Enviar

Ilustración 4.15: Pantalla: Formulario para consulta de gráficas y estadísticas de días anteriores

Al momento de definir una fecha, ya sea para consultar gráficas de datos o de estadística, pueden ocurrir 3 casos:

Caso 1. Introducir una fecha muy antigua.

Si ocurre este caso, se presenta al usuario el mensaje de error que muestra la ilustración 4.16



Ilustración 4.16: Mensaje de error para una fecha muy antigua

Caso 2. Introducir una fecha que aún no ocurre.

El mensaje de error que se maneja para este caso se muestra en la ilustración 4.17.



Ilustración 4.17: Mensaje de error para una fecha que aún no ocurre

Caso 3. Introducir una fecha que no tenga datos registrados.

El mensaje de error que se maneja para este caso se muestra en la ilustración 4.18.



Ilustración 4.18: Mensaje de error de datos no disponibles

En todos los mensajes de error presentados al usuario, siempre se ejecuta una re-dirección a la página web que contiene el formulario de consulta.

En lo que concierne a la introducción del periodo de tiempo para la consulta, se ha implementado un plugin hecho en JavaScript, el cual fue desarrollado por un tercero y está disponible para su uso libre y gratuito. El plugin permite evitar los errores más comunes a la hora de introducir un periodo de tiempo en un formulario, además de que es altamente configurable. Para los fines de uso en el formulario, el plugin ha sido configurado para introducir periodos de tiempo no menores de 5 minutos. La ilustración 4.19 muestra el uso del plugin implementado en la página web.

The image displays two screenshots of a web form interface for selecting a time period. The form includes two input fields: 'Hora Inicial' (Initial Time) and 'Hora Final' (Final Time). Below these fields is a time selection grid with two main sections: 'Hora' (Hour) and 'Minuto' (Minute). The grid is divided into AM and PM rows. The 'Hora' section has columns for hours 0-5 (AM) and 12-23 (PM). The 'Minuto' section has columns for minutes 00, 05, 10, 15, 20, 25, 30, 35, 40, 45, 50, and 55. In the first screenshot, 'Hora Inicial' is 8:00 and 'Hora Final' is 23:15. In the second screenshot, 'Hora Final' is updated to 8:05, and an 'Enviar' button is visible below the grid.

Hora							Minuto		
AM	0	1	2	3	4	5	00	05	10
	6	7	8	9	10	11	15	20	25
PM	12	13	14	15	16	17	30	35	40
	18	19	20	21	22	23	45	50	55

Enviar

Ilustración 4.19: Introducción de un periodo de tiempo usando un plugin

En la ilustración 4.20 se muestra el resultado de una consulta de gráficas de datos del 17 de mayo de 2013 y con un periodo de tiempo comprendido entre las 7 y las 8 horas.

Estación Meteorológica

[Volver al formulario](#)

Gráficas de Datos del 17-05-2013 de las 7:00:00 a las 8:00:59

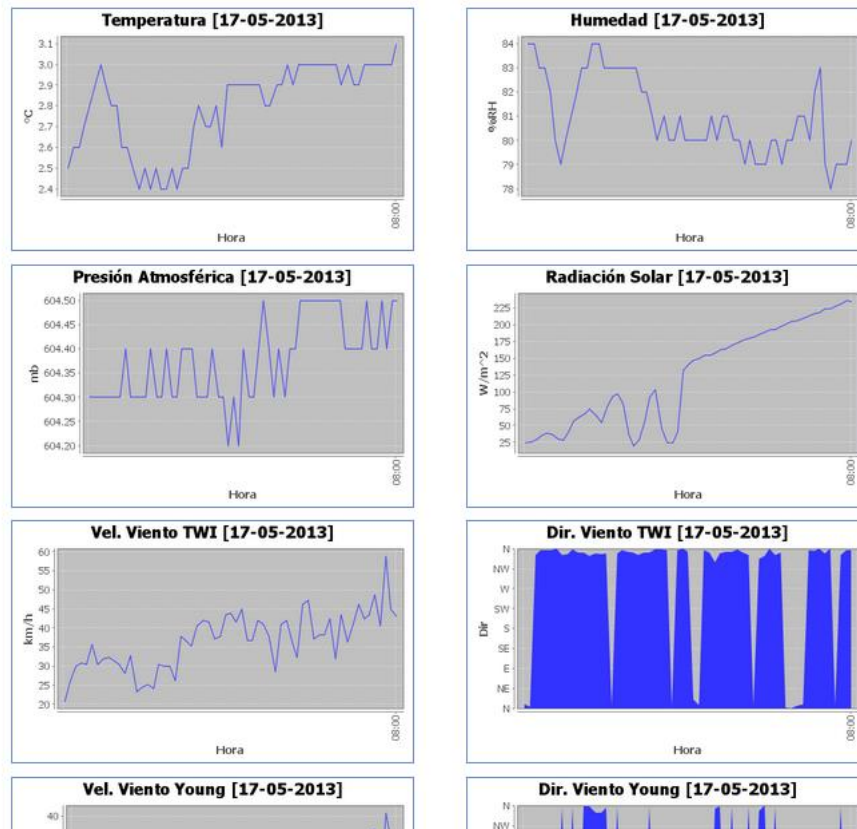


Ilustración 4.20: Ejemplo de una consulta de gráficas de datos con un periodo de tiempo de una hora

Las pruebas aplicadas al formulario de consulta fueron satisfactorias y presentaron los resultados esperados, tanto en las respuestas como en el manejo de errores y excepciones.

Para complementar las pruebas al sistema, se implementó un módulo encargado de elaborar una bitácora donde se registra la fecha, hora y descripción de los diversos eventos, errores y excepciones que ocurren durante la ejecución de las tareas que realiza el servidor. Esta bitácora resulta de bastante utilidad, ya que permite verificar el comportamiento del sistema; así como también de dar solución a los errores no contemplado en la etapa de diseño del sistema. La tabla 4.1 muestra un ejemplo del archivo que guarda la bitácora.

```

[Wed May 15 18:14:23 CDT 2013] SolusServlet inició el Wed May 15 18:14:23 CDT 2013
[Wed May 15 18:14:23 CDT 2013] Cargando parametros de inicio de SolusServlet
Wed May 15 18:14:23 CDT 2013
C:\Tomcat7\webapps\SolusServlet\datos\TW150513.txt (El sistema no puede encontrar el archivo
especificado) No se encuentra el archivo:C:\Tomcat7\webapps\SolusServlet\datos\TW150513.txt
|CLASE| |ARCHIVO| |LINEA| |METODO|
java.io.FileInputStream->null->-2->open
java.io.FileInputStream->null->-1-><init>
java.io.FileInputStream->null->-1-><init>
java.io.FileReader-> null->-1-><init>
SolusServletpackage.SolusServlet-> SolusServlet.java-> 834->TwiCargarDatos_hoy
SolusServletpackage.SolusServlet-> SolusServlet.java-> 136->init
org.apache.catalina.core.StandardWrapper-> StandardWrapper.java-> 1266->initServlet
org.apache.catalina.core.StandardWrapper-> StandardWrapper.java-> 1185->loadServlet
org.apache.catalina.core.StandardWrapper-> StandardWrapper.java-> 1080->load
org.apache.catalina.core.StandardContext-> StandardContext.java-> 5027->loadOnStartup
org.apache.catalina.core.StandardContext-> StandardContext.java-> 5314->startInternal
org.apache.catalina.util.LifecycleBase-> LifecycleBase.java-> 150->start
org.apache.catalina.core.ContainerBase-> ContainerBase.java-> 901->addChildInternal
org.apache.catalina.core.ContainerBase-> ContainerBase.java-> 877->addChild
org.apache.catalina.core.StandardHost-> StandardHost.java-> 618->addChild
org.apache.catalina.startup.HostConfig-> HostConfig.java-> 1100->deployDirectory
org.apache.catalina.startup.HostConfig$DeployDirectory-> HostConfig.java-> 1618->run
java.util.concurrent.Executors$RunnableAdapter->null->-1->call
java.util.concurrent.FutureTask$Sync-> null->-1->innerRun
java.util.concurrent.FutureTask-> null->-1->run
java.util.concurrent.ThreadPoolExecutor-> null->-1->runWorker
java.util.concurrent.ThreadPoolExecutor$Worker->null->-1->run
java.lang.Thread-> null->-1->run

[Wed May 15 18:14:23 CDT 2013] No existe el
archivo:C:\Tomcat7\webapps\SolusServlet\datos\TW150513.txt
[Wed May 15 18:14:27 CDT 2013] Carga correcta de parámetros de inicio para SolusServlet
[Wed May 15 18:14:27 CDT 2013] TWI conectado al Puerto COM4 usando GiovynetSerialPort a 9600
Bauds
[Wed May 15 18:14:27 CDT 2013] Se creó el proceso concurrente para SolusServlet el Wed May
15 18:14:27 CDT 2013
[Wed May 15 18:14:27 CDT 2013] Capturando datos meteorológicos en el archivo TW150513.txt
[Wed May 15 18:22:22 CDT 2013] SolusServlet finalizó el Wed May 15 18:22:22 CDT 2013

```

Tabla 4.1: Muestra del contenido de la bitácora del sistema

4.3.3. Seguridad del sistema

Los aspectos relacionados a la seguridad del sistema, tanto para el servidor como para el cliente, se ha limitado solo a usar los que ofrece por omisión cada software utilizado en el desarrollo e implementación del sistema. Del lado del servidor, se hizo uso de los esquemas de seguridad que ofrece el servidor Apache Tomcat, que básicamente consistió en definir la contraseña para el administrador (**admin**) (ver la ilustración 4.21). Tomando en cuenta las tareas que realiza el servidor, se considera suficiente para que el sistema funcione de manera correcta.

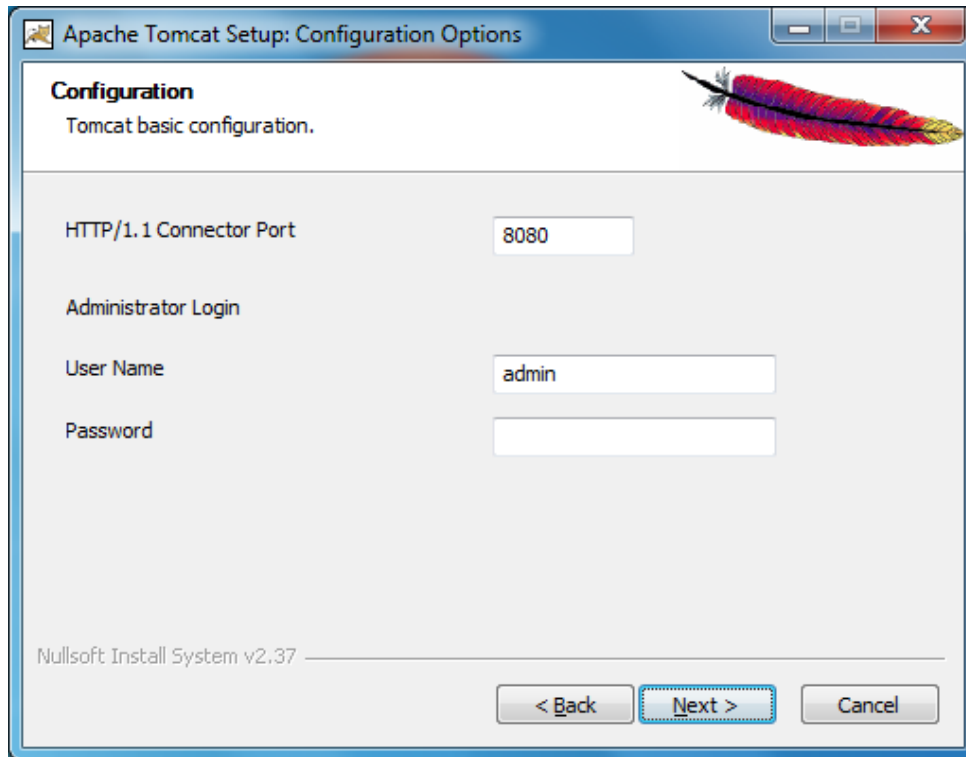


Ilustración 4.21: Definición de la contraseña para el Administrador en el servidor Apache Tomcat

En cuanto a la base de datos, el *SGBD MySQL* debe definirse la contraseña para el administrador (**root**). El sistema no usa la cuenta de administrador para realizar las operaciones en la base de datos, en su lugar se ocupan cuentas de usuarios con privilegios restringidos que solo permiten hacer operaciones relacionadas a la lectura y escritura; más no se les permiten operaciones tales como crear, modificar y eliminar las tablas; borrar y/o modificar registros. Con la definición de este tipo de usuarios se garantiza la seguridad e integridad de la base de datos que se usa en el sistema.

En la ilustración 4.22 se muestra la definición de privilegios para los diferentes usuarios de la base de datos a través del uso del software **MySQL Workbench**.

Ningún esquema de seguridad se implementó para el lado del cliente; ya que por el momento el sistema no hace uso de información proporcionada por el cliente que resulte del tipo personal sensible.

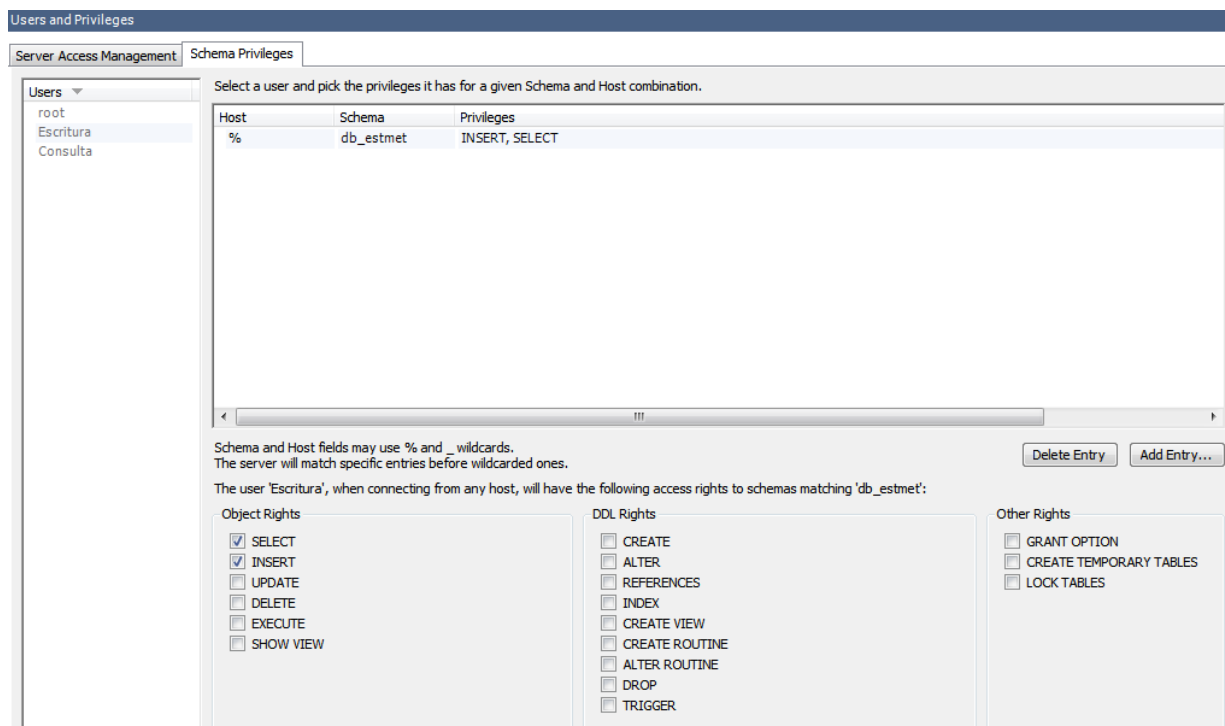


Ilustración 4.22: Definición de privilegios a los usuarios de la base de datos

4.3.4. Respaldo de datos

Debido a que el sistema realiza el almacenamiento de una gran cantidad de datos, se implementaron dos métodos sencillos para realizar el respaldo de los datos meteorológicos que se adquieren.

El primer método consiste en un archivo de texto que contiene una copia de los datos meteorológicos adquiridos durante el día. El sistema crea uno por día y le da por nombre la fecha de su creación con la extensión **txt**. Como ejemplo, los datos mostrados en la tabla 4.2 pertenecen al archivo llamado **150513.txt**, es decir, los datos pertenecen a la fecha del *15 de mayo de 2013*. Cada fila guarda la fecha y hora en que se capturaron las diferentes variables meteorológicas.

LOCAL-F	LOCAL-H	WD-Dg	WS-km/hr	S-W/m^2	T-°C	H-%RH	B-mb	R-mm
15/05/13	18:14:37	289	16.9	180	5.7	83	605.1	0
15/05/13	18:15:13	321	10.1	177	5.6	83	605.1	0
15/05/13	18:16:12	281	10.6	198	5.5	82	605.1	0
15/05/13	18:17:11	276	13.5	349	5.6	83	605.1	0
15/05/13	18:18:10	287	11.1	196	5.4	83	605.1	0
15/05/13	18:19:23	285	9.2	224	5	82	605.1	0

Tabla 4.2: Muestra del contenido de un archivo de texto con información respaldada

El segundo método consiste en crear un respaldo de la base de datos al final del día. El método hace uso del **Programador de tareas** del sistema operativo **Windows**. Esta herramienta se utiliza para programar la ejecución de un script a las 0 horas de cada día. El script contiene un conjunto de instrucciones que permiten hacer una copia del estado actual de la base de datos.

En la tabla 4.3 se muestran las instrucciones que contiene el script. Las primeras 6 líneas construyen el nombre que llevará el archivo de respaldo. La forma del nombre es **estmet_backup_DDMMYYYYY.sql**, donde **DD** es el día, **MM** es el mes y **YYYY** es el año. El archivo de respaldo se almacena en la carpeta **db_bkp** ubicada en la unidad **C**.

El comando **mysqldump** es usado para realizar el respaldo de la base de datos dentro de un archivo de texto. Los parámetros que requiere el comando son el nombre de usuario, contraseña de usuario y el nombre del archivo que guardará el respaldo. El usuario debe tener los suficientes privilegios para realizar la copia de la base de datos.

```
set FECHA= %date%
set FECHA=%FECHA:/=%
set FECHA=%FECHA:=%
set FECHA=%FECHA::=%
set FECHA=%FECHA:,%=%
set NOMARCH=C:\db_bkp\estmet_backup_%FECHA%.sql
mysqldump -p'contraseña' -u 'usuario' db_estmet > %NOMARCH%
```

Tabla 4.3: Contenido del script que realiza el respaldo de la base de datos

La ejecución del script se realiza diariamente a las 0 horas a través del uso de una tarea programada en el sistema operativo Windows. En la ilustración 4.23 se muestra el resumen de la programación para realizar el respaldo de la base de datos.

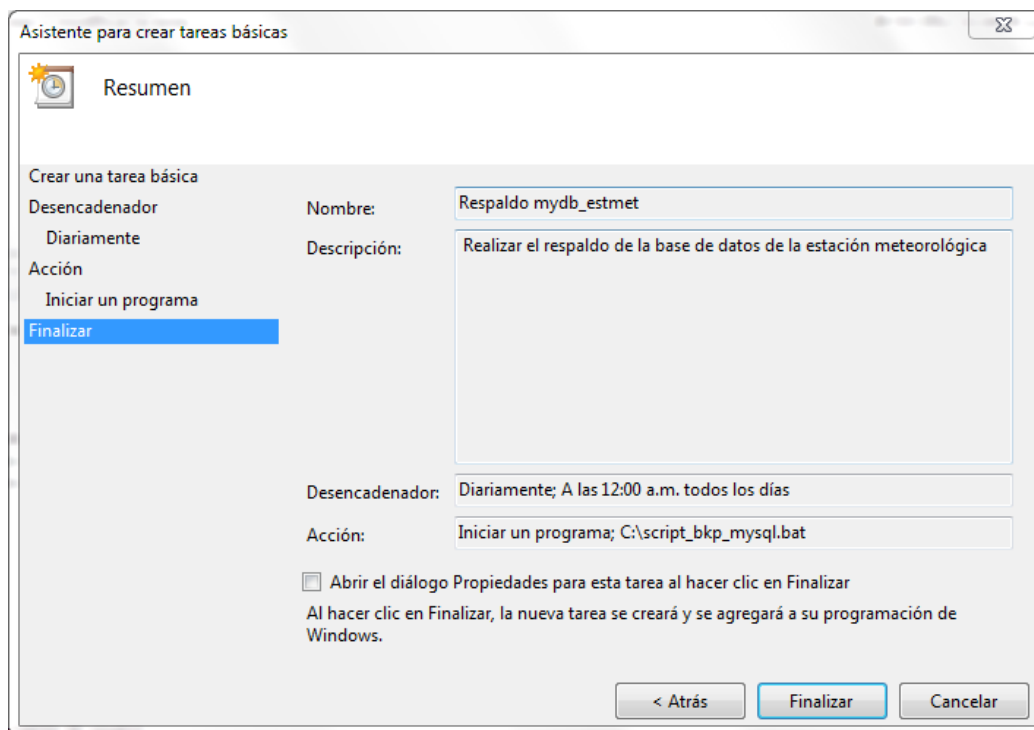


Ilustración 4.23: Programación en Windows de la tarea para ejecutar el script que realiza el respaldo de la base de datos

En caso de que se necesite recuperar la base de datos, el archivo de respaldo resultante podrá ser utilizado para reconstruir de nuevo la base de datos al estado de un día anterior. El procedimiento de recuperación se hace mediante la consola de MySQL donde se ejecutan los comandos de la tabla 4.4.

```
mysql> create database mydb_estmet;
mysql> use mydb_estmet;
mysql> source c:/db_bkp/estmet_backup_DDMMYYYYY.sql
```

Tabla 4.4: Comandos para recuperar la base de datos desde un archivo de respaldo

El primer comando crea una base de datos vacía, el segundo comando indica que se usará la base de datos recién creada y el último comando ejecuta todas las instrucciones que contiene el archivo de respaldo. Estos comandos son suficientes para recuperar la base de datos del sistema, en caso de que sea necesario.

CONCLUSIONES

En este capítulo se hace una recopilación de todas las conclusiones obtenidas en cada una de las etapas de desarrollo del sistema.

Anteriormente la estación meteorológica era un sistema aislado sin comunicación con el exterior. Toda la información que recopilaba era almacenada en una memoria limitada de forma local e implicaba hacer visitas constantes al lugar para obtener los datos. Posteriormente la infraestructura de comunicaciones se incrementó con la instalación de una red de datos con acceso a Internet y la mejora del servicio del suministro eléctrico. Estos factores permitieron buscar una solución al problema y que aprovechara los recursos disponibles, dando origen al *SISTEMA DE ADMINISTRACIÓN Y MONITOREO REMOTO VÍA INTERNET DE UNA ESTACIÓN METEOROLÓGICA*.

El sistema desarrollado permite el beneficio de tener la información en tiempo real de los datos meteorológicos generados por los instrumentos de la estación meteorológica. El tener la información en un servidor hace que cualquier persona en el mundo pueda acceder a ellos solamente teniendo una computadora con acceso a Internet y visualizar la información en diferentes formatos, ya sea para el día actual o de los que ocurrieron en días pasados.

Este sistema toma ventaja de las herramientas de software disponibles en la actualidad tales como *Java*, *Apache Tomcat* y *MySQL*. Estas herramientas cuentan con mucho arraigo dentro del desarrollo de sistemas web porque son libres o de código abierto y evita el pago de licencias.

La elección de la metodología UP, de los casos de uso y del uso UML fueron elecciones cruciales para realizar con éxito la etapa de análisis y diseño, ya que permitió tener una visión concreta y ordenada de cómo debería de ser desarrollado el sistema para cumplir cada una de las especificaciones y objetivos que debía cubrir el proyecto. La experiencia adquirida en esta etapa sirvió para ampliar y renovar los conocimientos sobre las nuevas tendencias y metodologías usadas en la ingeniería de software.

El uso de la programación orientada a objetos, la arquitectura cliente-servidor y del modelo relacional para la base de datos resultaron adecuadas para el desarrollo del sistema. Los beneficios obtenidos se reflejan en un sistema que desarrolla sus múltiples tareas de manera estable, segura y confiable.

La base teórica que respalda el análisis y diseño del sistema y el uso de software libre o de código abierto permitieron cubrir de manera satisfactoria los objetivos planteados en esta tesis. Ahora ya se cuenta con un sistema bien estructurado y multiplataforma que representa una base tecnológica comprobada que da pauta para establecer monitoreo remoto con otros dispositivos que generen otro tipo de información.

TRABAJOS A FUTURO

Considerando que la primera versión del sistema es estable en cuanto a la adquisición, procesamiento y almacenamiento de los datos meteorológicos; que cumple de manera satisfactoria con la arquitectura cliente – servidor y cubre las especificaciones más importantes que están enfocadas al cliente; aún quedan algunas especificaciones que debe cumplir el sistema para que sea más completo y este a la par del software comercial de su tipo.

Del lado del servidor, se plantea los siguientes trabajos a futuro:

1. Creación de un módulo con acceso vía web para la configuración del datalogger y los instrumentos de medición. Actualmente las configuraciones se realizan modificando directamente el código y la base de datos. El módulo deberá funcionar bajo el formato *login/password*.
2. Soporte para conectar más dataloggers e instrumentos de medición.
3. Agregar funciones para generar reportes de estadísticas semanales y mensuales.
4. Elabora un instalador para el sistema.
5. Mejorar los esquemas de manejo de errores y de respaldo de la base de datos.

Del lado del cliente, se proponen los siguientes trabajos a futuro:

1. Buscar una nueva alternativa para presentar los datos recién adquiridos. Si bien los applets de Java ofrecen amplias opciones en cuanto a diseño de interfaces gráficas y de intercambio de información con servidores; hoy en día están casi en desuso debido principalmente a motivos de seguridad.
2. Agregar más opciones para la consulta de datos; tales como consulta de reportes semanales y mensuales.
3. Integración de gráficas dinámicas que muestren la evolución durante el día de cada una de las variables meteorológica.
4. Implementar páginas web optimizadas para dispositivos móviles.
5. Crear un sitio web dedicado a la estación meteorológica.

REFERENCIAS BIBLIOGRÁFICAS

[1] Presentación del curso de Ingeniería de software.

Humberto Cervantes Maceda

Universidad Autónoma Metropolitana

2008

[2] Principles of software engineering and design.

M.V. Zelkovitz, A.C. Shaw, J.D. Gannon

Prentice Halls

1979

[3] Principles of software engineering and design.

B.W. Boehm

IEEE Transactions on Computers. C-25, n.12

1976

[4] Software engineering: Information processing

F. L. Bauer

North Holland Publishing Co.

1972

[5] Standards collection: Software engineering.

IEEE Standard 610.12-1990: IEEE Standard Glossary of Software Engineering Terminology.

IEEE Society Press N.Y.

1993

[6] Computing curricula 2001, Volume I: Overview, Volume II: Computer Science, Volume III: Computer Engineering, Volume IV: Software Engineering, Volume V: Information Systems Technical report, ACM/IEEE-CS/ Association for Information Systems, The Joint Task Force on Software Engineering Education Project (SWEEP)

ACM/IEEE Computer Society Press

2001

[7] Introduction to the personal software process.

W.S. Humphrey

Addison-Wesley

1997

[8] Software engineering. 6th edition.

I. Sommerville
Addison Wesley
2001

[9] UML 2 Iniciación, ejemplos y ejercicios corregidos.

Laurent Debrauwer, Fien Van der Heyde
2005

[10] UML for mere mortals.

Robert A. Maksimchuck, Eric J. Naiburg
Addison Wesley Prof.
2004

[11] OMG: Unified Modeling Language specification

Grady Booch, Ivar Jacobson, Jim Rumbaugh
2008

[12] Visual modeling & Unified Modeling Language (UML): Introduction to UML

Satish Mishra
Rational Software Corporation
2008

[13] UML for systems engineering: Watching the wheels (Second Edition)

Jon Holt
Institution of Electrical Engineers.
2004

[14] UML: Superstructure specification (Version 2.2)

OMG: Object Management Group
2009

[15] Ingeniería de software orientada a objetos con UML, Java en Internet

Alfredo Weitzenfeld
Cengage Learning Editores
2005

[16] Introducción al análisis de sistemas y la ingeniería de software

Roberto Cortés Morales
Editorial EUNED
1998

[17] UML 2 and the Unified Process: Practical Object-Oriented analysis and design
Jim Arlow, Ila Neustadt
Pearson Education
2005

[18] Software engineering end quality assurance
Anuradha A. Puntambekar
Technical Publications
2010

[19] Fundamentos de informática y programación en C
Diego Rafael Llanos Ferraris
Editorial Paraninfo
2010

[20] Cómo programar en Java (5ta edición)
Harvey M. Deitel, Paul J. Deitel. Traducido por Alfonso Vidal Romero Elizondo
Pearson Educación
2004

[21] Introducción a las bases de datos relacionales
Ma. Victoria Nevado Cabello
Editorial Visión Libros

[22] Sistemas de bases de datos: diseño, implementación y administración
Peter Rob, Carlos Coronel
Cengage Learning Editores
2004

[23] El modelo relacional y el álgebra relacional
Dolors Costal Costa
Editorial UOC (Universitat Oberta de Catalunya)

[24] Escaneando la informática.
María Jesús Marco Galindo, Josep María Marco Simo, Josep Prieto Blázquez, et al.
Editorial UOC (Universitat Oberta de Catalunya)
2010

[25] Database design for mere mortals: A hands-on guide to relational database design for mere mortals
Michael J. Hernandez
Addison-Wesley
2013

[26] Sistemas de información medioambiental

José Manuel Cotos Yáñez, José Ángel Taboada González

Editorial Netbiblo

2005

[27] Programación de aplicaciones web: Historia, principios básicos y clientes web

Sergio Luján Mora

Editorial Club Universitario

2002

[28] La tecno ciencia y su divulgación: un enfoque transdisciplinar. Volumen 15 de Ciencia, tecnología y sociedad.

Andoni Alonso, Carmen Galán Rodríguez

Anthropos Editorial

2004

[29] MySQL 5: instalación, implementación, administración, programación

Cyril Thibaud

Ediciones ENI

2006

[30] Pro MySQL

Jay Pipes, Michael Kruckenberg

Editorial Apress

2005

[31] Desarrollo de sistemas de información: una metodología basada en el modelado.

Vicenç Fernández Alarcón

Ediciones UPC 8 Univ. Politècnica de Catalunya)

2006

[32] UML 2: Iniciación, ejemplos y ejercicios corregidos

Laurent Debrauwer, Fien van der Heyde

Ediciones ENI

2005

[33] Creatividad y entornos virtuales de aprendizaje

Sonia Cristina Gamboa Sarmiento

Universidad Pedagógica Nacional

2004

REFERENCIAS ELECTRÓNICAS

[E1] Página oficial de Weather View 32

Weather Station Software for monitoring weather stations.

<http://www.weatherview32.com>

[E2] Página oficial de Ambient Weather

AmbientWeather.com: weather stations, weather station equipment, weather radios, and more.

<http://www.ambientweather.com>

[E3] Página oficial de Weather Display

Weather Display - the Weather Station software.

<http://www.weather-display.com>

[E4] Página oficial de The Weather Channel

National and Local Weather Forecast, Hurricane, Radar and Report.

<http://www.weather.com>

[E5] Página oficial de Wunderground

Weather Forecast & Reports - Long Range & Local

<http://www.wunderground.com>

[E6] Página oficial de Davis

Professional quality weather stations and software, automotive tracking, and marine accessories by Davis.

<http://www.davisnet.com>

[E7] Página oficial de Young USA

R. M. Young Company

<http://www.youngusa.com>

[E8] Página oficial de Texas Weather Inst.

Texas Weather Instruments, Inc - Manufacturer of Digital Weather Stations.

<http://www.txwx.com>

[E9] Página oficial de la OMG

Object Management Group

<http://www.omg.org/>

[E10] Página web oficial de la Iniciativa de Código Abierto (OSI)
Open Source Initiative
<http://www.opensource.org>

[E11] Página web oficial del proyecto Apache Tomcat
Apache Tomcat is an open source software implementation of the Java Servlet and JavaServer Pages technologies
<http://tomcat.apache.org>

[E12] Página web "The Java Tutorials"
The Java Tutorials are practical guides for programmers who want to use the Java programming language to create applications.
<http://docs.oracle.com/javase/tutorial/>

[E13] Página oficial del IDE Netbeans
NetBeans IDE, the Smarter and Faster Way to Code
<http://www.netbeans.org>

[E14] Página oficial de Giovynet: Java Electronic Solutions
Giovynet Driver is a framework that lets you use Java to create applications that allows communicate external circuits with a PC
http://www.giovynet.com/giovynetDriver_en.html

[E15] Página oficial del proyecto JFreeChart
JFreeChart is a free 100% Java chart library that makes it easy for developers to display professional quality charts in their applications
<http://www.jfree.org/jfreechart/>

[E16] Página de descarga del software MySQL Server
The world's most popular open source database
<http://www.mysql.com/downloads/>

[E17] Proceso Unificado
Autor: colaboradores de Wikipedia
Editor: Wikipedia, la enciclopedia libre
Fecha de consulta: 21 de febrero de 2014
http://es.wikipedia.org/w/index.php?title=Proceso_Unificado&oldid=66355877