Universidad Andrés Bello
Facultad de Ingeniería
Ingeniería en Computación e Informática

Metodología de Desarrollo de Software
Informe de proyecto

El presente documento tiene como objetivo establecer la estructura del informe final del proyecto, el cual describirá el proceso de implementación del producto de software generado por el equipo de desarrollo. Para esta etapa final de implementación se utilizan los artefactos y resultados intermedios obtenidos de las actividades realizadas para el modelamiento del análisis y diseño orientado a objetos.

Finalmente, el propósito de la presentación final es demostrar la capacidad de análisis, diseño, e implementación de las técnicas y métodos utilizados conforme a las directrices propuestas por la Ingeniería de Software en general. Además, con la culminación de esta etapa, el alumno debe demostrar en forma eficiente la correcta planificación y desarrollo de sistemas informático/computacionales.

1. Portada

Deberá contener las siguientes leyendas en la forma que a continuación se indica:

- En la parte superior y centrado: <Nombre de la institución>
- En la mitad y centrado: Metodología de Desarrollo de Software
- Más abajo y centrado: Informe N°1. “Presentación y Planificación del Proyecto”
- Más abajo y centrado: <Nombre del Proyecto>
- En la parte inferior derecha, con letra más pequeña: <Datos Informe>

<`Nombre de la institución`>: Nombres de la universidad, Facultad y Escuela.

<`Nombre del proyecto`>: Frase corta, bién redactada. Describe en términos precisos una síntesis conceptual del proyecto a realizar.

<`Datos Informe`>: Nombre alumnos, Nombre Profesor, Nombre Ayudantes, Fecha de entrega.

2. Contenido

Deberán indicarse con claridad y precisión los siguientes puntos: Agradecimientos Resumen Tabla de contenido

Capítulo 1. Presentación y planificación del proyecto

1.1 Introducción

1.2 Objetivos del proyecto

1.2.1 Objetivo general

1.2.2 Objetivos específicos

Capítulo 3. Programacion orientada a objetos

3.1 Introducción

3.2 Clases Capa Modelo

3.3 Clases Capa Vista

3.4 Clases Capa Controlador

3.5 Conclusiones del capítulo

Capítulo 4. Conclusiones del proyecto

Referencias bibliográficas

Apéndice a. Manual de usuario

Apéndice b. Manual de explotación

3. Descripción del contenido del informe

Agradecimientos

Es una sección optativa y se podrá incluir en el caso de que los integrantes del equipo de desarrollo deseen agradecer a personas u organizaciones que le facilitaron ayuda en el desarrollo de su proyecto de curso.

Resumen

El resumen presenta en forma breve lo sustantivo del proyecto realizado; su extensión no deberá ser mayor de una página. Deberá dar una idea completa del trabajo, presentando su propósito u objetivos, el alcance, lo más destacado de su desarrollo, los resultados obtenidos y el carácter general de las conclusiones o recomendaciones más relevantes. El resumen no es un introducción a un problema, ni tampoco un relato del índice. El resumen no debe llevar citas bibliográficas. El resumen es un síntesis del contenido, por lo tanto, no debe aparecer nada que no aparezca en el cuerpo del documento. Al pie de la página deberá escribirse "PALABRAS CLAVES: ɾ un conjunto de entre 3 y 8 palabras descriptoras del trabajo.

Tabla de contenido

Índice de contenidos. Corresponde a la enumeración ordenada de las materias contenidas en el informe, con los mismos títulos y con indicaciones de la página en que se encuentran. También deberá incluirse un índice separado para los títulos de las tablas y uno para los títulos de las figuras contenidas en el informe final.

Capítulo 1. Presentación y planificación del proyecto

1. Introducción

Constituirá la primera sección del informe de avance. La introducción debe ofrece una visión clara y preliminar del informe desarrollado. En él deberán establecerse las ideas que faciliten la compresión del proyecto que se realizará durante el semestre, explicando brevemente su contenido.

2. Objetivos del proyecto

2.1. Objetivo General

Define en forma precisa el fin último para el cual se desarrolla el proyecto. Típicamente corresponde al resultado esperado del proyecto, por ejemplo, un sistema a desarrollar que hace X. No explica ni justifica el logro del objetivo.

2.2. Objetivos específicos

Estos objetivos determinan resultados parciales o metas intermedias necesarias para el logro del objetivo general. Cada objetivo logrado se debe poder evaluar (los resultados son medibles). Estos objetivos no deben ser confundidos con los objetivos (o propósitos) del producto del proyecto de desarrollo.

3. Descripción de la problemática

3.1. Motivación

Esta sección presenta el ámbito o la disciplina de investigación donde se aborda el problema del proyecto (seguridad en redes, ciencias de la computación, sistemas de información, etc.). Describe el contexto donde surge el problema. Caracteriza en términos muy simples el problema que se pretende abordar. Se motiva el problema, desribiendo la importancia que tiene desarrollar una solución al problema y su impacto, o las consecuencias más relevantes, que la solución tendría en alguna disciplina del conocimiento, en las personas, en las empresas, etc. Los argumentos aquípresentados, en relación con la importania del desarrollo de una solución, son antecedentes necesarios para justificar la realización del proyecto de software.

3.2. Definición del problema

Presenta resumidamente los principales conceptos del problema, necesarios para entender el problema. Básicamente describe el problema a resolver.

3.3. Estado del arte

Se entregan antecedentes generales de la historia o evolución del problema. Se describe brevemente el estado en que se encuentr el desarrollo de lasoluciones relacionadas con el problema.

4. Descripción de la solución propuesta

4.1. Características de la solución

Se propone una solución en función del problema planteado Se indican las principales características o requisitos que debe cumpmlir la solución.

4.2. Propósito de la solución

Se indica el propósito u objetivo de los resultados, o del producto, a obtener. Indica en qué medida se mejoraría la situación actual del problema.

4.3. Alcances y limitaciones de la solución

Se deja establecido con claridad qué debe considerarse como parte del proyecto y que no, y el grado de satisfacción esperado. Junto con el punto anterior, este punto permite saber si el grupo de trabajo no hará menos de lo que se requiere, o más de lo que se debe.

5. Gestión de riesgos, metodología, herramientas y ambiente de desarrollo

5.1 Gestión de riesgos

Debe identificar y clasificar los posibles riesgos, en términos de su impacto en el proyecto y la probabilidad de ocurrencia, las posibles razones por las cuales el proyecto podría retrasarse o fracasar y establecer las acciones de mitigación.

5.2 Metodología a usar

Informa sobre la (o las) metodología(s) a utilizar (ágiles, monumentales, etc.), sus autores, los fundamentos conceptuales, las principales actividades que ella define y las relaciones entre ellas (modelo de proceso de software). Para cada actividad describe las técnicas o métodos que se aplican y los productos que ella genera Se entregan las razones de su adopción.

5.3 Herramientas de desarrollo

Se describen las características principales de las herramientas de hardware y software que serán utilizadas en el desarrollo de la solución utilizando metodologías de desarrollo orientadas a objetos.

5.4 Ambiente de desarrollo

Se indican las características del entorno de trabajo donde se ha desarrollar la solución, los recursos materiales y humanos disponibles para trabajar, informar si depende de otros para disponer de datos de entrada, etc. No debe plantearse en término de queja sino de hechos objetivos que forman parte del problema que se ha de abordar.

6. Plan de trabajo

6.1. Roles y responsabilidades

Desecribir cada uno de los roles y responsabilidades de los integrantes del equipo de trabajo.

6.2. Planificación de entrevistas

Se debe realizar una programación preliminar de las entrevistas que el equipo realizará tanto con clientes y usuarios finales como también con los especialistas de áreas afines al proyecto de software a realizar (bases de datos, resdes y telecomunicaciones, sistemas operativos, etc.) y que el equipo considere necesario consultar. Esta programación de entrevistas debe estar calendarizada.

6.3. Planificación del proyecto

debe establecerse una programación simple del proyecto dividido en tres etapas principales:

- Planificación del proyecto.
- Análisis y diseño OO.
- Implementación.

Para cada etapa es necesario asignar ciertos periodos de tiempo a las actividades que la componen, a través de una carta Gantt. En forma mínima debe especificar las fechas de inicio y término globales. La planificación debe considerar como fecha de inicio el día 26 de Marzo y finalización el 18 de Junio (fecha correspondiente a la realización de la presentación final del proyecto).

7. Conclusiones

Esta sección presenta las conclusiones del equipo de trabajo respecto al informe/trabajo realizado.

Capítulo 2. Análisis y diseño orientado a objetos

1. Introducción

Costituirá la primera sección del segundo capítulo del informe. La introducción debe ofrecer una visión clara y preliminar del informa realizado. En él deberán establecerse las ideas que faciliten la comprensión del análisis y diseño orientado a objetos desarrollado, explicando brevemente su contenido.

2. Captura de requerimientos

La captura de requerimientos es la primera etapa de OMT++ que permite detectar las necesidades de los clientes/usuarios para llegar a una especificación precisa, no ambigua, consistente y completa de los requisitos funcionales, no funcionales, y de implementación del sistema. Con el propósito de obtener un mayor orden y facilidad de entendimiento en la organización de los requerimientso, se suguiere realizar su documentación por módulos o procesos principales.

2.1 Requerimientos funcionales

Se desribe en detalle los requerimientos funcionales (lógica) del sistema. Los requerimientos funcionales se refieren a las operaciones que el sistema debe efectuar, describiendo las transformaciones que el sistema realiza sobre las entradas para producir salidas específicas.

2.2. Requerimientos no funcionales

Se describen en detalle los requerimientos no funcionales (propiedades emergentes) del sistema. Los requerimientos no funcionales son aquellos que no se involucran con el funcionamiento en sí del sistema, estos se refieresn al as características y atributos del sistema.

2.3. Requerimienso de implementación

Se describe en detalle los requerimientos de implementación del sistema. Los requerimientos de implementación son necesidades del cliente que restringen la implementación (por ejemplo, lenguaje de programación, plataforma de hardware, servidor de página web, hoja de estilo, etc.).

2.4. Diagrama de casos de uso

Se realizan los diagramas de caos de uso (representación gráfica de un caso de uso) para describir los requerimientos funcionales del sistema, con el propósito de entregar una visión más explicativa de la interacción que el usuario tendrá con el sistema. En la figura 1 se puede visualizar un diagrama de casos de uso, en el cual los actores se representan en forma de pequeñas personas y los casos de uso se representan por elipses contenidas dentro de un rectángulo que representa el sistema. La participación de los actores en los casos de uso se indica por una flecha entre el actor y el caso de uso que apunta en la dirección en la que fluye la información.

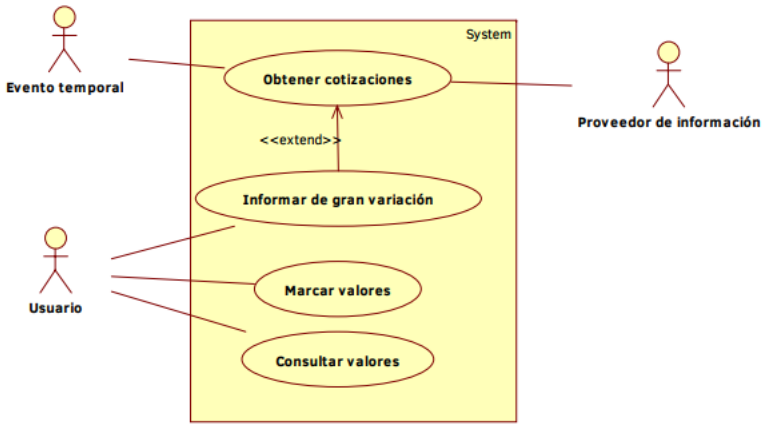

Figura 1: Diagrama de caso de uso.

2.5. Especificación de casos de uso

Una vez que los requerimientos son capturados, y representados gráficamente con diagramas de casos de uso, se procede a la especificación de cada uno de ellos. Los casos de uso describen la forma como el sistema responde a la funcionalidad requerida junto con la participación dle usuario. Ésta descripción es de forma abstracta, esto quiere decir, que el caso de uso debe ser desrito de manera tal, que cualquier persona sea capaz de entender su contenido. En lo posible, la explicación no debe contener términos técnicos. En la tabla 1

Tabla 1: Estructura (especificación) de un Caso de uso.

Caso de uso N°	Nombre del caso de uso.
Resumen	Breve resumen de la descripción del caso de uso.
Frecuencia	Frecuencia con que se realiza el caso de uso en el sistema.
Actores	Usuarios que participan en el caso de uso.
Precondiciones	Condiciones preestablecidas al realizar el caso de uso.
Descripción	Descripción paso por paso, y de forma básica y clara, del desarrollo del caso de uso.
Excepciones	Acciones que se presentan durante el desarrollo del caso de uso que conlleva a un error.
Poscondiciones	Condiciones que se presentan después del desarrollo del caso de uso.

3. Análisis OO

Se descrbie la etapa de análisis orientado a objetos (AOO) de la metodología OMT++, aplicada en el desarrollo del sistema. La realización de esta etapa está basada en los documentos obtenidos en la etapa de captura de requerimientos. En el AOO se realizan las siguientes actividades: "*Análisis de objetos*", "*Análisis de comportamiento*" y "*Especificación de componentes de interfaz de usuario*".

3.1. Análisis de objetos

Esta actividad consiste en el análisis de los requerimientos y casos de uso especificados en la sección anterior, lo que deriva en el establecimiento de los conceptos del dominio del problema y sus asociaciones.

3.1.1. Modelo de objetos del análisis

Se realiza el modelo conceptual del análisis (modelo de objetos del análisis = diagrama de clases del análisis). Para el diagrama se debe describir cada una de las clases y sus atributos a través de un diccionario de datos. El diccionario se representa por una tabla con tres columnas que describen el nombre de la clase, la descripción de la clase y sus atributos, respectivamente. En la tabla 2

Tabla 2: Ejemplo de diccionario de datos del modelo de objeto del análisis.

Objeto/Clase	Descripción	Atributo
UNIDAD	Contiene información de los centros académicos existentes en la Universidad, encargadas de impartir carreras	`uni_id`: identificador (código) de la unidad. `uni_nombre`: nombre de la unidad. `uni_umayor`: unidad mayor de la cual depende jerárquicamente.
TIPO_UNIDAD	Clasificación que se asigna a una unidad académica o centro (Facultades, Departamentos, Escuelas, etc.)	`Tuni_id`: identificador (código) del tipo de unidad. `Tuni_nombre`: nombre del tipo.

3.2. Análisis del comportamiento

Luego del análisis de objetos se realiza el análisis de comportamiento, cuyo objetivo es definir las operaciones realizadas por el usuario para la manipulación de la información, sin especificar detalles de la interfaz.

3.2.1. Especificación de operaciones

El análisis de comportamiento modela el sistema como una caja negra, es decir, solo la funcionalidad externa, y produce una lista de operaciones. El sistema final debe soportar la ejecución de todas las operaciones de dicha lista. En la Tabla 3 se describe la forma en que se debe listar las operaciones.

Tabla 3: Ejemplo de lista de operaciones.

N°	**Operación**	**N° de Caso de Uso**
1	Configurar parámetros	5
2	Configurar menús	7
3	Crear perfil de usuario	3

3.3. Especificación de la interfaz de usuario

La última actividad de la etapa de análisis de objetos es la especificación y visualización de la interfaz de usuario. La interfaz de usuario es una entidad intermediaria entre el usuario final y la aplicación, por la cual la interfaz de usuario debe ser capaz de comunicarse tanto con el usuario final como con la aplicación. La especificación de operaciones define requerimientos de funcionalidad y el modelo de objetos del análisis determina los datos que se van a incluir en la interfaz de usuario. Dentro de las actividades que componen la especificación de interfaz de usuario, está la construcción de diagramas de diálogo, y la especificación de sus componentes.

3.3.1. Análisis de operaciones

Antes de construir los diálogos y especificar sus componentes se debe analizar cada operación obtenida para dividirlas en tareas. Esto forma parte del análisis necesario para la especificación de la interfaz de usuario. En la Tabla 4 se describe la forma en que se debe listar las tareas para cada operación del sistema.

Tabla 4: Ejemplo de tareas de operación "Configurar Parámetros".

N°	**Operación**	N°	**Tarea**
1	Configurar parámetros	1	Seleccionar la opción para configurar parámetros.
		2	Seleccionar parámetro.
		3	Ingresar valor.
		4	Modificar descripción.
		5	Guardar configuración del parámetros.

3.3.2. Diagramas de diálogo

La especificación de las operaciones y sus tareas ayudan a identificar los diagramas de diálogos necesarios para el sistema. Para esto se organizan las tareas para formar diálogos, los cuales ayudan a entender cómo el usuario se comunica con el sistema a través de las tareas especificadas en la etapa anterior. Es necesario precisar que los números que identifican a las tareas en los diálogos, corresponden a aquellos que enumeran a cada tarea del sistema según la Tabla 4.

3.3.3 Especificación de componentes

Cada diagrama de diálogo está compuesto de tareas que el usuario debe realizar a través de la interfaz. Pensando en la interfaz de usuario, cada diálogo está formado por componentes y cada componente está constituido de herramientas, tales como botones o campos de texto, entre otras. Estas herramientas pueden ser de manipulación o de retroalimentación o una combinación de ambas. Los componentes se ilustran mediante un rectángulo redondeado de color blanco; las herramientas se ilustran mediante una elipse cuando se trata de una herramienta de manipulación, un rectángulo cuando se trata de una herramientas de retroalimentación y una elipse rectangular cuando se trata de una herramienta combinada.

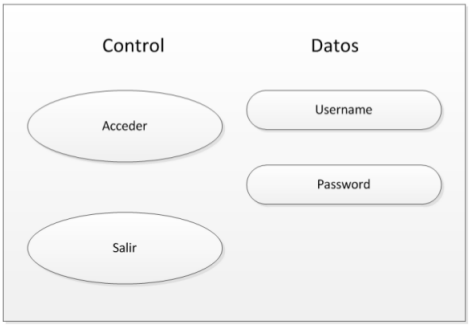

Figura 2: Diagrama de componente.

Para cada diálogo se debe realizar su especificación (representación gráfica) de componentes, y la descripción del comportamiento de cada componente (en forma breve) a través de una tabla. En la Figura 2 se observa un diagrama de componentes para un típico ejemplo de iniciar sesión y en la Tabla 5, su correspondiente descripción del comportamiento del componente.

Componente	**Manipulación**	Retroalimentación
Iniciar sesión	Aceptar	Username
	Salir	Contraseña

2.4 Diseño OO

Se describe la etapa de diseño orientado a objetos de la metodología OMT++, aplicada en el desarrollo del sistema. El desarrollo de esta etapa está basado en la documentación obtenida en el análisis orientado a objetos, al realizar el modelo de objetos, la especificación de operaciones, los diagramas de diálogos y la especificación de componentes. En el diseño orientado a objetos se realizan las siguientes actividades: "Diseño de objetos" y "Diseño de comportamiento".

2.4.1 Diseño de objetos.

En esta etapa actividad se realiza el modelo de clases del diseño aplicando el paradigma Modelo-Vista-Controlador (MVC) al modelo conceptual de objetos del análisis. El modelo de clases resultante se compone de tres capas; la capa del modelo, cuyas clases se extraen del modelo de objetos del análisis; la capa de la vista, cuyas clases se obtienen de los diagramas de diálogos de la especificación de la interfaz de usuario; y por último, la capa controlador; la cual controla la interacción entre la capa vista y la capa del modelo.

2.4.2 Diseño del comportamiento.

El diseño de comportamiento permite identificar los métodos de las clases definidas en los modelos de diseño MVC para cada módulo, mediante la creación de diagramas de secuencia o trazas de eventos. Para definir estos métodos, se realizan diagramas de secuencia para cada operación del sistema, definidas en la especificación de operaciones de la etapa de análisis orientado a objetos.

Para una mejor comprensión, se sugiere exponer los diagramas (trazas) en forma agrupada en los módulos que componen el sistema junto con los usuarios que interactúan en cada módulo. En la representación de las trazas de eventos, cada columna representa una clase correspondiente a una clase del modelo, de la vista o una clase del controlador, y cada flecha horizontal representa una comunicación entre las clases. En la Figura 3 se muestra un ejemplo de traza de eventos de la operación.

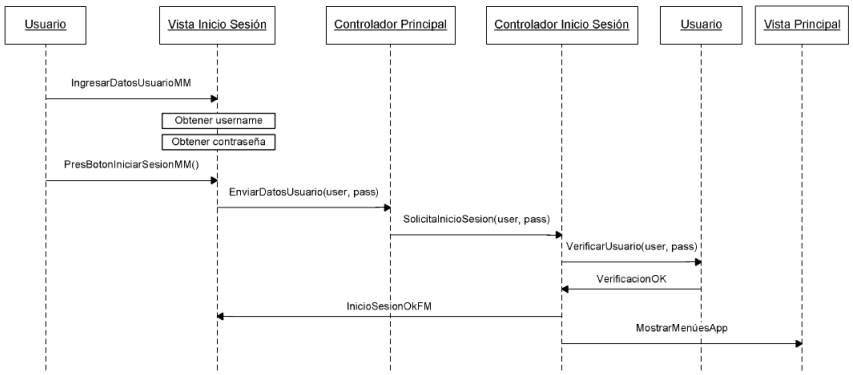

Figura 3: Traza de eventos operación "Iniciar sesión".

2.5 Conclusiones del capítulo

Esta sección presenta las conclusiones del equipo de trabajo respecto al informe/trabajo realizado.

Capítulo 3. Programacion orientada a objetos

En este capítulo se describe la última fase de la programación orientada a objetos, en donde se especifica e implementa cada una de las clases definidas en la etapa de diseño (MVC++). Se describe además, los atributos y los métodos de las clases a través de los cuales se comunican los objetos para la capa modelo, controlador y vista. La especificación de clases de cada capa se debe realizar a través de tablas, por ejemplo:

Tabla 5: Multi-row table

	Unidad	Descripción
Atributos	`uni_id`	Almacena el identificador de la unidad.
	`uni_nombre`	Almacena el nombre de la unidad.
	`uni_tipo`	Almacena el tipo de unidad.
	`uni_umayor`	Indica la unidad mayor de la cual depende jerárquicamente la unidad.
Métodos	`RegistrarUnidad()`	Guarda una unidad en el sistema.
	`ExtraerDatosUnidad()`	Extrae los valores de los atributos de una unidad.
	`ModificarUnidad()`	Actualiza los datos modificados de la unidad.
	`EliminarTipoUnidad()`	Elimina una unidad del sistema.
	`BuscarPorTipoUnidad()`	Busca las unidades de un tipo determinado.
	`BuscarPorUmayorUnidad()`	Busca las unidades que dependen de una unidad mayor.

En este capítulo **NO** se debe incluir el código fuente de las clases implementadas.

Las secciones del capítulo son las siguientes:

3.1. Introducción

3.2. Clases Capa Modelo

En esta sección se especifican las clases que forman parte de la capa del modelo del diseño de objetos, indicando sus atributos y métodos.

3.3. Clases Capa Vista

En esta sección se especifican las clases que forman parte de la capa vista del diseño de objetos, indicando los atributos y métodos necesarios para manejar los datos y peticiones entre el usuario y el controlador.

3.4. Clases Capa Controlador

En esta sección se especifican las clases que forman parte de la capa controlador del diseño de objetos, indicando los atributos y métodos necesarios para manejar los datos y peticiones entre la vista y el modelo. Estas clases permiten recuperar instancias de objetos del modelo del diseño para presentar los datos manejados por el sistema a las clases de la vista, además de controlar y dirigir peticiones hacia algún recurso del sistema

3.5. Conclusiones del capítulo

Esta sección presenta las conclusiones del capítulo.

Capítulo 4. Conclusiones del proyecto

Este capítulo presenta las conclusiones finales del proyecto de software realizado. Las conclusiones deben ser el resultado de un análisis cuidadoso del trabajo. Ejemplos de conclusiones posibles: nivel de logro de los objetivos, análisis crítico del proceso de desarrollo realizado, consecuencias o impacto observado de los resultados obtenidos con la metodología, recomendaciones, proyecciones futuras, etc.

Referencias bibliográficas

Ver guía para la elaboración de informes publicada en el sitio http://www.unabvirtual.cl de la asignatura.

Apéndice a. Manual de usuario

El manual de usuario es una guía funcional que permite a los usuarios finales del sistema encontrar, conocer y aplicar fácilmente las operaciones principales del producto software. En general, un manual de usuario debe ser entendido de manera fácil por cualquier usuario principiante, como así también ser útil a los usuarios avanzados. Se recomienda desplegar, de manera lógica, las interfaces del sistema para realizar una tarea determinada, utilizando ejemplos que permitan rápidamente utilizar las funciones principales del sistema. Para una mejor comprensión, el manual puede incluir un glosario de términos ad-hoc.

Apéndice b. Manual de explotación

El manual de explotación es un documento técnico que debe presentar de manera detallada todos los pasos a seguir para instalar, configurar, y administrar el producto de software. Se debe especificar claramente la secuencia de pasos para, por ejemplo, instalar servidores web, de correo, bibliotecas (dll, ocx, etc.), opciones del sistema operativo, conexión a la base de datos, etc. Por consiguiente, este manual debe presentar todos los aspectos que utilizará el personal

4. Presentación del informe final

La presentación final se realizará en base a la siguiente estructura:

1. Presentación y Planificación del Proyecto.
2. Análisis y Diseño Orientado a Objetos.
3. Programación Orientada a Objetos.
4. Conclusiones Finales.

Cada punto anterior debe ser expuesto en una Presentación (en formato PDF y debe tener un total de 4 diapositivas). La exposición de los 4 puntos señalados tiene 10 minutos como máximo, y durante ella el equipo de trabajo presentará los aspectos más relevantes de cada una de las etapas desarrolladas (se debe demostrar capacidad de síntesis y claridad en los contenidos de la exposición).

Posteriormente, se debe mostrar el producto de software 100 % operativo, de acuerdo a los objetivos declarados al comienzo del desarrollo del proyecto y a la especificación de requerimientos. Con el propósito de facilitar la demostración del software, es obligatorio poblar la base de datos con suficiente información para explicar adecuadamente las funcionalidades del sistema, como también disponer de un conjunto de datos de prueba previamente validados por el equipo de desarrollo.

El grupo de trabajo deberá realizar la presentación del informe final y del sistema de software operativo en la fecha definida para ello, según el sorteo de las presentaciones de los proyectos de curso. La duración total de la presentación final y del sistema de software funcionando no puede exceder los 30 minutos. En caso de exceder el tiempo máximo, se dará por finalizada la presentación (con la consiguiente penalización en la calificación del proyecto).

Finalmente, cada equipo de trabajo debe entregar en 1 CD-ROM (correctamente etiquetado con el nombre del producto de software, y los integrantes del equipo de trabajo), los siguientes directorios:

```
\DOCUMENTACION
\PRESENTACION
\SOFTWARE
\SCRIPT_BD
```

La carpeta `DOCUMENTACION`, debe contener el informe final, en formato PDF. La carpeta `PRESENTACION`, debe contener las 3 presentaciones realizadas durante el semestre, en formato PDF. La carpeta `SOFTWARE`, debe contener todos los programas fuentes estructurados en un forma lógica y coherente con el diseño orientado a objetos del sistema. La carpeta `SCRIPT_BD`, debe contener todos los script de creación de los diversos objetos de la base de datos (tablas, vistas, etc.), y los scripts de creación de usuarios, aplicación de permisos (grant), entre otros.

Es preciso señalar que es requisito fundamental la entrega del informe final impreso, debidamente compaginado y anillado (o empastado), como también del CD-ROM del proyecto, justo antes de iniciar la presentación respectiva. La no entrega de cualquiera de éstos, inhabilitará al equipo de trabajo para realizar su presentación final, siendo calificados con la nota mínima. NO se aceptarán informes o CD- ROM posteriores a la hora de inicio de la presentación programada para el grupo de trabajo.