FASES y TAREAS TÍPICAS DE UN PROYECTO SOFTWARE

El presente documento tiene como finalidad servir de soporte a las prácticas de GPR. En él se enumeran las fases típicas de desarrollo del software, así como los entregables de los que suele constar un proyecto informático. Se pretende que esta información resulte de utilidad al alumno para la realización de las diferentes actividades propuestas en las sesiones prácticas de la GPR: declaración de alcance, elaboración de la EDT (primer bloque prácticas), determinación de tareas para las fases de planificación y estimación (segundo bloque de prácticas).

Esta información es meramente orientativa, ya que se pretende dar prioridad a que el alumno tenga libertad para aplicar sus propios conocimientos, y preferencias, a la hora de organizar el desglose del trabajo. También se valorará especialmente que el desglose del trabajo se adapte al caso de estudio en cuestión.

Los contenidos de este documento se organizan de la siguiente forma:

1. Fases del Desarrollo del Software

2. Tareas y Entregables de un Proyecto Software

3. Ejemplos de Estructuras de Desglose del Trabajo (EDT)

En el siguiente apartado se enumeran las fases típicas en el desarrollo de un proyecto software, así como los entregables y tareas más habituales.

1. Fases del Desarrollo del Software

Seguidamente se muestran las fases típicas del desarrollo del software:

1. Estudio de viabilidad
2. Análisis y Especificación de requisitos
3. Diseño
4. Implementación
5. Verificación y pruebas
6. Despliegue (o instalación)
7. Mantenimiento

Generalmente, estas fases (que se describen en más detalle a continuación) o parte de ellas, constituyen el proceso o ciclo de vida del desarrollo del software. **Desde el punto de vista de la EDT (orientada a proceso), estas fases permiten determinar el primer nivel de desglose.**

1. Estudio de viabilidad

El **estudio de la viabilidad** es el estudio que dispone el éxito o fracaso de un proyecto a partir de una serie de datos base de naturaleza empírica: medio ambiente del proyecto, rentabilidad, necesidades de mercado, factibilidad política, aceptación cultural, legislación aplicable, medio físico, flujo de caja de la operación, haciendo un énfasis en viabilidad financiera y de mercado. Es por lo tanto un estudio dirigido a realizar una proyección del éxito o fracaso de un proyecto.

2. Análisis y especificación de requisitos

El objetivo principal de esta etapa es extraer los requisitos del producto de software. En esta etapa la habilidad y experiencia en la ingeniería del software es crítica para reconocer requisitos incompletos, ambiguos o contradictorios. Usualmente el cliente/usuario tiene una visión incompleta/inexacta de lo que necesita y es necesario ayudarle para obtener la visión completa de los requerimientos. El contenido de comunicación en esta etapa es muy intenso ya que el objetivo es eliminar la ambigüedad en la medida de lo posible. Se trata de describir detalladamente el software a ser escrito, de una forma rigurosa. Se describe el comportamiento esperado del software y su interacción con los usuarios y/o otros sistemas.

A su vez esta fase se puede descomponer en varias fases:

a) **Elicitación de requisistos:** descubrir y extraer requisitos.
b) **Análisis de requisistos:** razonar sobre la información obtenida resolviendo inconsistencias y coordinando los requisitos (se pueden utilizar diagramas UML).
c) **Validación de requisitos:** Para evaluar la calidad de los requisitos, lo que puede reducir significativamente los costes del proyecto.
d) **Gestión de requisitos:** indica las acciones que se realizarán cuando se determine que un requisito del sistema cambia durante el transcurso del proyecto.

3. Diseño

Determinar cómo funcionará de forma general, sin entrar en detalles, incorporando consideraciones de la implementación tecnológica, como el hardware, la red, etc. Consiste en el diseño de los componentes del sistema que dan respuesta a las funcionalidades descritas. Generalmente se realiza en base a diagramas que permitan describir las interacciones entre las entidades y su secuenciado.

Capas y niveles

En el **diseño de sistemas informáticos actual** se suelen usar las **arquitecturas multinivel** o Programación por capas. En dichas arquitecturas a cada nivel se le confía una misión simple, lo que permite el diseño de arquitecturas escalables (que pueden ampliarse con facilidad en caso de que las necesidades aumenten), y que se ajustan a las necesidades de gran parte de aplicaciones en la nube, o que necesitan ser accedidas por un gran número de usuarios a través de la red.

El más utilizado actualmente es el diseño en tres niveles (o en tres capas), aunque estas capas se pueden integrar en dos capas (en este caso se suele llamar **frontend** a la capa que

interacciona con el ususario, y **backend** la capa que implementa la lógica de las operaciones). De esta forma, podemos decidir utilizar esta estrategia para diseñar nuestra aplicación o una parte de ella. Generalmente, siempre que tengamos una aplicación web con un número elevado de usuarios se utiliza este tipo arquitectura software. La interacción entre las diferentes capas se suele realizar a través del esquema cliente-servidor.

a) **Capa de presentación:** la que ve el usuario (también se la denomina "capa de usuario"), presenta el sistema al usuario, le comunica la información y captura la información del usuario en un mínimo de proceso (realiza un filtrado previo para comprobar que no hay errores de formato). También es conocida como interfaz gráfica y debe tener la característica de ser "amigable" (entendible y fácil de usar) para el usuario. Esta capa se comunica únicamente con la capa de negocio.
b) **Capa de negocio:** es donde residen los programas que se ejecutan, se reciben las peticiones del usuario y se envían las respuestas tras el proceso. Se denomina capa de negocio (e incluso de lógica del negocio) porque es aquí donde se establecen todas las reglas que deben cumplirse. Esta capa se comunica con la capa de presentación, para recibir las solicitudes y presentar los resultados, y con la capa de datos, para solicitar al gestor de base de datos almacenar o recuperar datos de él. También se consideran aquí los programas de aplicación.
c) **Capa de datos:** es donde residen los datos y es la encargada de acceder a los mismos. Está formada por uno o más gestores de bases de datos que realizan todo el almacenamiento de datos, reciben solicitudes de almacenamiento o recuperación de información desde la capa de negocio.

4. Implementación

Se traduce el diseño a código. Es la parte más obvia del trabajo de ingeniería de software y la primera en que se obtienen resultados “tangibles”. No necesariamente es la etapa más larga ni la más compleja aunque una especificación o diseño incompletos/ambiguos pueden exigir que, tareas propias de las etapas anteriores se tengan que realizarse en esta.

5. Verificación y pruebas

Consiste en comprobar que el software responda/realice correctamente las tareas indicadas en la especificación. Es una buena praxis realizar pruebas a distintos niveles (por ejemplo primero a nivel unitario y después de forma integrada de cada componente) y por equipos diferenciados del de desarrollo (pruebas cruzadas entre los programadores o realizadas por un área de test independiente).

Las pruebas de software son las investigaciones empíricas y técnicas cuyo objetivo es proporcionar información objetiva e independiente sobre la calidad del producto a la parte interesada o stakeholder. Es una actividad más en el proceso de control de calidad.

6. Despliegue e implantación

En la actualidad, debido a que muchas de las aplicaciones informáticas son multicomponentes (y distribuidas), cada vez es más importante automatizar el despliegue de los diferentes componentes de una aplicación entre un conjunto de máquinas, o para una infraestructura o conjunto de infraestructuras.

El proceso de despliegue de una aplicación (que tradicionalmente se denominada instalación) consta de un conjunto interrelacionado de actividades con interacciones entre ellas. Del correcto despliegue de una aplicación (con el número adecuado de instancias de cada componente y la infraestructura hardware adecuada) depende que se satisfagan determinados **requisitos no funcionales** de la aplicación.

7. Mantenimiento

En esta etapa se realizan un mantenimiento correctivo (resolver errores) y un mantenimiento evolutivo (mejorar la funcionalidades y/o dar respuesta a nuevos requisitos).

Los problemas claves de mantenimiento de software son administrativos y técnicos. Problemas clave de administración son: alineación con las prioridades del cliente, dotación de personal, cuál organización hace mantenimiento, estimación de costos. Son cuestiones técnicas claves: limitado entendimiento, análisis de impacto, pruebas (testing), medición de mantenibilidad.

2. Tareas y Entregables usuales en un proyecto Software

Dado que el objetivo final del proyecto es la entrega de un subsistema informático (entregable) veamos algunas definiciones y utilidades de los entregables. Los entregables los definiremos como "Productos que, en un cierto estado, se intercambian entre los clientes y los desarrolladores a lo largo de la ejecución del proyecto informático". Dichos entregables suelen ser el resultado de un conjunto de tareas o actividades asociadas a cada fase de un proyecto software.

Los entregables los clasificamos como relativos al objetivo y relativos a la gestión del proyecto. Son entregables relativos al objetivo todos aquellos documentos que hacen referencia exclusivamente al sistema de información y al subsistema informático en desarrollo. Pertenecen a este conjunto los requisitos del sistema, la especificación del sistema, la documentación del diseño, él código fuente, los programas ejecutables, los manuales de usuario, etc. La gran mayoría de los entregables que se enumeran en este documento son relativos al objetivo, y no tanto a la gestión del proyecto.

Se deberá definir de forma clara el conjunto mínimo de entregables necesarios para dar por terminada cada fase de desarrollo. Aunque algunos entregables se desarrollan a lo largo de varias tareas. Los entregables nos proveen de:

1. Un conjunto de componentes que formarán el producto una vez finalizado el desarrollo.
2. Los medios para medir el progreso y la calidad del producto en desarrollo.
3. Los materiales necesarios para la siguiente etapa.

Seguidamente se enumeran las tareas y entregables más usuales asociadas a cada una de las fases de desarrollo del software. Esta información pretende ser una guía flexible y tampoco pretende ser exhaustiva, es meramente orientativa de las necesidades que requieren para llevar a cabo un proyecto informático.

1. Estudio de viabilidad:

El estudio de viabilidad permite decidir si el sistema propuesto es conveniente. Es un estudio rápido y orientado a conocer:

- Si el sistema contribuye a los objetivos de la organización.
- Si el sistema se puede desarrollar con la tecnología actual y con el tiempo y el coste previsto.
- Si el sistema puede integrarse con otros existentes.

Posibles Entregables

Descripción breve del sistema propuesto y sus características.
Descripción breve de las necesidades del negocio en el sistema propuesto.
Propuesta de organización del equipo de desarrollo y definición de responsabilidades.
Estudio de los costes, que contendrán estimaciones groseras de la planificación y fechas, tentativas, de entrega de los productos.
Estudio de los beneficios que producirá el sistema.

2. Análisis y especificación de requisitos:

Tal y como se comentó en el apartado 1 de este documento, podemos descomponer esta fase en:

a) **Elicitación:** fase inicial en la que se trata de descubrir (elicitar) los requisitos. El personal técnico trabaja con los clientes y usuarios para descubrir el dominio de la aplicación, los servicios que se deben proporcionar y las restricciones. Puede implicar a los usuarios finales, encargados, ingenieros implicados en el mantenimiento, expertos de dominio, etc. (Son los llamados participantes, interesados o **stakeholders**). El objetivo es que estos descubran, comprendan y revelen los requisitos que desean. Algunas de las tareas a realizar en esta fase inicial son:

- Realización de entrevistas a los diferentes stakeholders del sistema.
- Preparar y realizar reuniones de elicitación/negociación.
- Obtener información sobre el dominio y el sistema actual.
- Observación del sistema.
- Identificar los objetivos del sistema y priorizarlos.

b) **Análisis:** El siguiente paso consiste en obtener recopilar los requisitos del sistema razonando (análisis) sobre la información obtenida. Detectando y resolviendo posibles inconsistencias o conflictos, y coordinando los diferentes requisitos entre sí. Algunas de las tareas asociadas pueden ser:

- Identificar los requisitos del sistema: de información, funcionales, no funcionales.
- Priorizar los requisitos.
- Descripción detallada de los requisitos
- Realización de diagramas UML y diagramas de casos de uso.

c) **Validación:** Seguidamente se realiza la validación de los requisitos. Consiste en demostrar que los requisitos definen el sistema que el cliente desea. Esto permitirá ahorrar esfuerzo y coste en futuras fases del proyecto. Se trata de asegurar que los requisitos satisfacen ciertos criterios de calidad como: validez, completitud, consistencia, verificabilidad, trazabilidad, adaptabilidad, etc.

d) **Gestión:** indica las acciones que se realizarán cuando se determine que un requisito del sistema cambia durante el transcurso del proyecto.

Posibles Entregables

Resumen de Entrevistas
Documento de Visión
Alcance del proyecto
Participantes del proyecto
Visión global de la arquitectura del sistema
Glosario de términos
Prototipos
Catálogo de requisitos del sistema
Requisitos Funcionales
-Diagramas de casos de uso
-Definición de actores
-Casos de uso del sistema
Requisitos No funcionales
Requisitos de Información
Reglas de negocio
Modelo del dominio
Documento de validación de requisitos
Documento de gestión de requisitos

3. Diseño:

Las tareas propias de la fase de diseño de un proyecto informático pueden dividirse de la siguiente forma:

a) **Selección de técnicas de implementación recomendadas:** estas tareas consisten en la determinación de los estándares de programación y de diseño de programas que se recomiendan para el desarrollo del proyecto.

b) **Diseño de la Interfaz de usuario del sistema:** Forma parte de la capa de presentación (si escogemos una arquitectura por capas) y determina cómo será la interacción del sistema con los usuarios del sistema. Algunas de las tareas a realizar pueden ser:

- Prototipos de Interfaz de usuario.
- Diseño HTML (para aplicación web).
- Validación de los datos (diseña estrategias para validar datos introducidos por el usuario y de esta forma reducir errores).
- Evaluación de la calidad (en función de determinados criterios de calidad en uso, o siguiendo guías de la organización o criterios establecidos por la industria).

c) **Diseño de Datos:** Es el proceso de producir un modelo de datos detallados para el almacenamiento de datos de nuestra aplicación. Algunas de las tareas a realizar pueden ser:

- Determinar los datos que serán almacenados.
- Determinar las relaciones entre los datos.

d) **Diseño de la arquitectura:** Tiene como objetivo determinar los componentes e instancias que tendrá el sistema con el objetivo de poder satisfacer todos los requerimientos solicitados (principalmente atañe a los requerimientos no funcionales). También como serán las relaciones entre los diferentes componentes. Algunas de las tareas asociadas pueden ser:

- Realizar diagramas que representen la arquitectura.
- Determinar los requerimientos hardware y software de cada componente, y de cada capa (en caso de arquitectura por capas). Como por ejemplo: que tipo de servidor web utilizaremos, o que tipo de base de datos necesitamos.
- Diseñar la comunicación entre los diferentes componentes o capas (en caso de arquitectura por capas).
- Diseñar la seguridad del sistema.
- Validación de la arquitectura (demostrar que permite satisfacer los requisitos del sistema, lo que está muy relacionado con la calidad del sistema).

e) **Diseño de módulos:** Incluye diagramas de los diferentes módulos e interfaces soportadas por cada uno de ellos. Atañe principalmente a la capa de "lógica de negocio", que es donde se implementa la lógica de la aplicación.

Posibles Entregables

Documento de estándares de programación y diseño de programas recomendados
Documento con necesidades externas: compra de paquetes y contratación externa
Diseño de interfaz de usuario (para los diferentes módulos)
Prototipos de interfaz de usuario
Diseño HTML
Estrategias de validación de datos
Validación de la interfaz de usuario (satisfaciendo los requisitos de calidad)
Diseño de datos
Diseño de la arquitectura
Diagramas
Requerimientos hardware y software
Comunicación
Seguridad
Validación de la arquitectura (satisfaciendo los requisitos de calidad)
Diseños particulares de cada módulo
Diagramas
Interfaces

4. Implementación:

Como hemos comentado anteriormente, no tiene por qué ser la fase más costosa del proyecto. Aquí es donde el software a ser desarrollado se codifica. Dependiendo del tamaño del proyecto, la programación puede ser distribuida entre distintos programadores o grupos de programadores. Cada uno se concentrará en la construcción y prueba de una parte del software, a menudo un subsistema. Las pruebas, en general, tiene por objetivo asegurar que todas las funciones están correctamente implementadas dentro del sistema.

Posteriormente, todos los subsistemas codificados independientemente se juntan. Cada sección es enlazada con otra y, entonces, probada. Este proceso se repite hasta que se han agregado todos los módulos y el sistema se prueba como un todo.

Posibles Entregables

Documento de evaluación y definición de la herramienta de Desarrollo
Documentos finales de implementación (para cada componente)
Documentos del diseño final.
Descripción detallada de la lógica del componente.
Listado de los módulos y componentes, conteniendo comentarios.
Cadenas de ejecución si es necesario (scripts, etc.).
Documento de pruebas
Resultado de las pruebas de cada unidad.
Resultado de las pruebas de cada componente.
Resultado de las pruebas de la integración.
Guía para los operadores del sistema
Manual de usuario del sistema
Programa de entrenamiento de los operadores
Programa de entrenamiento de los usuarios del sistema

5. Verificación y pruebas:

Una vez que el sistema ha sido integrado, comienza esta etapa. Es donde es probado para verificar que el sistema es consistente con la definición de requisitos y la especificación funcional. Por otro lado, la verificación consiste en una serie de actividades que aseguran que el software implementa correctamente una función específica. Al finalizar esta etapa, el sistema ya puede ser desplegado en ambiente de explotación.

Posibles Entregables

Plan de Verificación
Casos de Prueba
Procedimientos de Prueba
Pruebas de la funcionalidad
Pruebas de comunicación
Pruebas de seguridad
Informes de Verificación de Documentos

6. Despliegue e Implantación:

El despliegue de una aplicación incluye todas las actividades necesarias para ensamblar el sistema y transferirlo a una infraestructura donde pueda ser utilizado por los usuarios del mismo. Por tanto también debe determinar los recursos y operaciones que son necesarios por parte de los futuros administradores del sistema.

En la actualidad, donde muchas aplicaciones son desplegadas sobre en la nube, o sobre entornos virtualizados, es muy importante automatizar el despliegue para que pueda funcionar sobre cualquier tipo de infraestructura. Para ello son muy importantes los sistemas de "contenedorización", los cuales permiten la ejecución de los componentes e instalación del sistema sobre cualquier tipo de infraestructura. Estos sistemas facilitarán considerablemente las tareas de mantenimiento del sistema por parte de los operadores.

Algunas de las tareas típicas para realizar el despliegue de una aplicación, utilizando la filosofía que acabamos de comentar, son construir:

- **Descriptores de componentes:** Indican todo lo que necesita (dependencias software y hardware) cada componente del sistema para poder ser ejecutado.
- **Descriptores de servicio:** Contienen una descripción de todos los componentes que constituyen la aplicación, así como las relaciones y dependencias entre ellos. Esto facilitará significativamente las tareas de mantenimiento del sistema a desarrollar, facilitando la sustitución de un componente por otro, así como la gestión de operaciones básicas (parada y reinicialización del sistema, etc.).
- **Descriptores de despliegue:** Indica cuántas instancias de cada componente serán utilizadas para desplegar la aplicación sobre una determinada infraestructura.

Una vez el sistema sea desplegado y esté operativo para los usuarios la implantación puede finalizarse realizando la formación adecuada de los usuarios del sistema y evaluando el grado de asimilación y aceptación. Algunas las tareas a realizar en este sentido pueden ser:

- **Elaborar cursos de formación de usuarios**
- **Elaborar cursos de formación de operadores**
- **Impartir los cursos de formación de usuarios y operadores**
- **Evaluar el grado de aceptación del producto**

Posibles Entregables

Descriptores
Descriptores de componentes (contiene dependencias software)
Descriptores de Servicio (contiene conexiones y dependencias entre componentes)
Descriptores de Despliegue (indica cuantas instancias de cada componente se lanzan)
Material de cursos de Formación
Informe de evaluación de la aceptación del producto

7. Mantenimiento:

El mantenimiento de software es una actividad muy amplia que incluye la corrección de errores, mejoras de las capacidades, eliminación de funciones obsoletas y optimización. Debido a que el cambio es inevitable, se debe desarrollar mecanismos para la evaluación, controlar y hacer modificaciones.

Posibles Entregables

Plan de revisión post-instalación
Listado de fallos detectados en el sistema
Listado de mejoras solicitadas por los usuarios (si no dan lugar a nuevos proyectos)
Traza detallada de los cambios realizados en el sistema
Actas de las revisiones regulares del sistema y aceptación de los niveles de soporte

3. Ejemplos de Estructuras de Desglose del Trabajo (EDT)

3.1 ¿Que es una EDT?

La Estructura de Desglose del Trabajo (EDT) equivale al término Work Breakdown Structure (WBS) y representa un esquema jerárquico del trabajo a realizar. El objetivo de la EDT es desglosar el trabajo de un proyecto en elementos más manejables y fáciles de definir, de modo que con cada nivel que desciende la descripción de la EDT, se incrementa la precisión en la definición del trabajo, hasta alcanzar el nivel máximo de detalle, formado por los paquetes de trabajo. La estructura de desglose del trabajo es el principal esquema de trabajo del proyecto. Concretamente, esta herramienta aporta las siguientes ventajas:

- Permite identificar posibles tareas omitidas u olvidadas, y proporciona una estructura para verificar rápidamente si existe al menos una tarea para cada requisito y objetivo definidos en la planificación.
- Facilita la estimación de la duración y coste del trabajo, pues una vez se han identificado todas las tareas a realizar, la elaboración del cronograma y el presupuesto se basa en la agregación de duración y coste estimados para todas las tareas
- Proporciona una estructura para detallar el trabajo, en porciones comprensibles y manejables, con suficiente nivel de detalle.

Los paquetes de trabajo deben definirse de forma que puedan planificarse temporal y económicamente, así como realizar su seguimiento y control durante la ejecución del proyecto. El nivel de detalle en su definición dependerá del área de aplicación, e igualmente del tamaño y complejidad del proyecto. Una vez definidos los paquetes de trabajo, pueden agruparse de forma lógica, en fases o funciones, o siguiendo algún tipo de división organizativa. La descomposición puede hacerse en función de las fases del ciclo de vida, los entregables, e incluso a nivel de subproyectos contratados de forma externa. Lo más importante es que el conjunto de los paquetes de trabajo de nivel inferior, permitan completar los entregables correspondientes de nivel superior. Es recomendable numerar jerárquicamente cada uno de los paquetes de trabajo para facilitar la lectura.

Una aproximación para el desarrollo de la EDT para un proyecto software, orientada al proceso, consiste en establecer el primer nivel a partir de las diferentes fases del desarrollo del software. Las siguientes subdivisiones se pueden establecer a partir de los entregables o de los diferentes módulos o paquetes que tenga el proyecto.

3.2 Segundo nivel de desglose de la EDT

Las tareas o entregables descritos en el apartado 2 de este documento pueden servirnos como orientación a la hora de realizar el **segundo nivel de desglose de la EDT** (recordemos que el primer nivel de desglose se corresponde con las fases del desarrollo del software enumeradas anteriormente) o incluso los **paquetes de trabajo**, ya que nos dan una idea del trabajo que debe ser completado en cada una de las fases del proyecto. También podemos utilizar los **diferentes paquetes (o entregables principales) de los que conste la aplicación software a desarrollar como segundo nivel de desglose de la EDT**.

En el siguiente ejemplo se muestran diferentes opciones para construir una EDT. Con este ejemplo se pretende que el alumno comprenda que la construcción de la EDT es un proceso abierto a muchas posibilidades. **Todo el trabajo debe ser realizado, pero nosotros elegimos como organizarlo.**

3.2 Ejemplo (aplicación de venta on-line):

Para explicar estas dos opciones, con mayor detalle, podemos imaginar el siguiente escenario:

Se pretende: desarrollar una aplicación para la gestión de ventas on-line a través de la web.

Esta aplicación podríamos tener tres módulos claramente diferenciados:

- **Aplicación web accesible para los usuarios** (a través del cual los usuarios hacen las compras).
- **Módulo de gestión** (a través del cual se gestiona la plataforma: productos, ofertas, contabilidad, etc.),
- **Módulo especial para los transportistas** (que facilitaría que los productos lleguen de forma ágil al comprador).

Seguidamente se muestran los fragmentos de tres posibles EDT's (centrándonos en la fase de diseño). En los dos primeros utilizamos como segundo nivel de desglose los 3 entregables principales (o paquetes) en los que dividimos el proyecto (**Figuras 1 y 2**), en el tercero utilizamos como segundo nivel de desglose las capas de diseño (en caso de que utilicemos un diseño por capas, **Figura 3**).

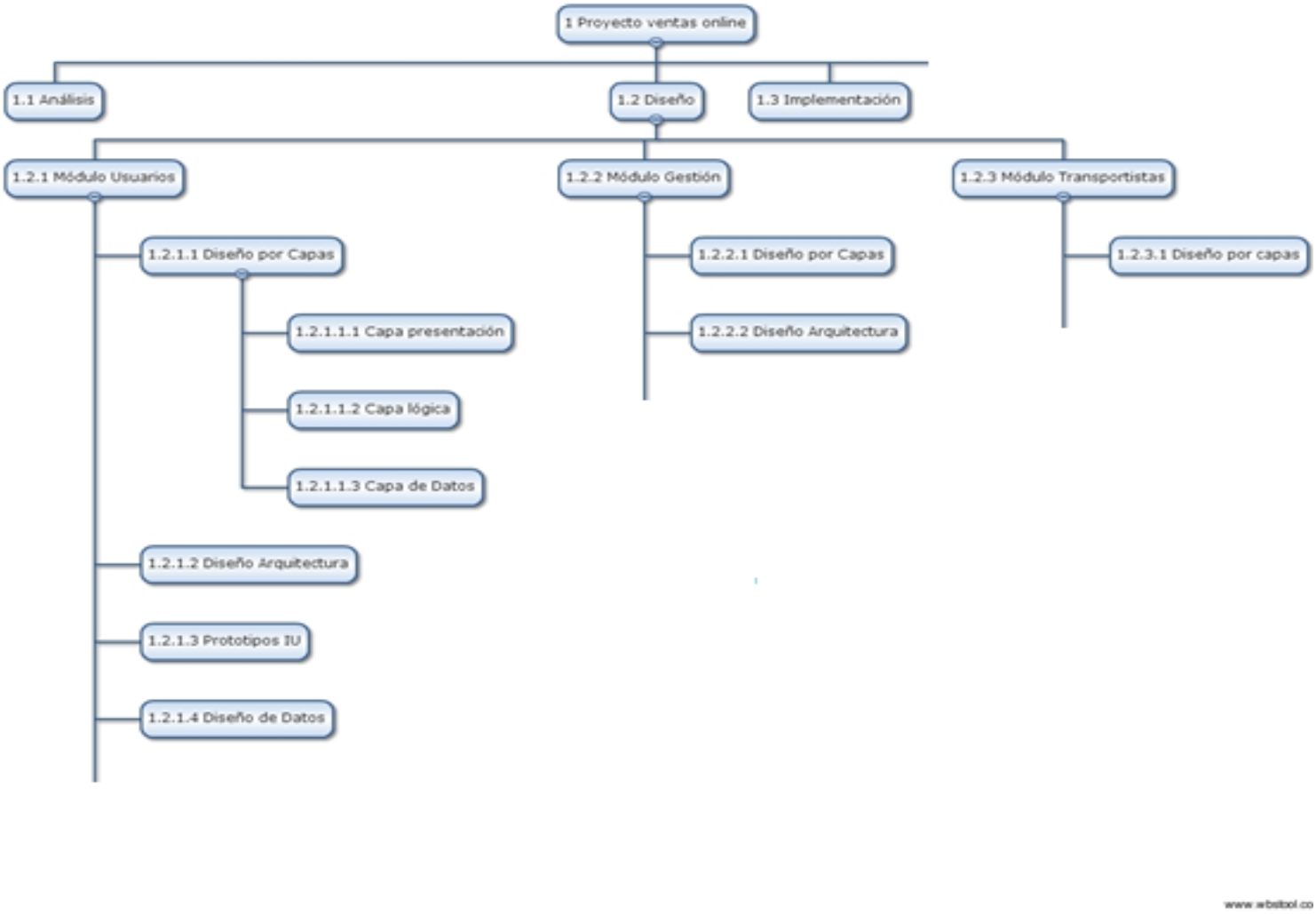

Figura 1: Segundo nivel de desglose (fase de diseño) en base a módulos de la aplicación.

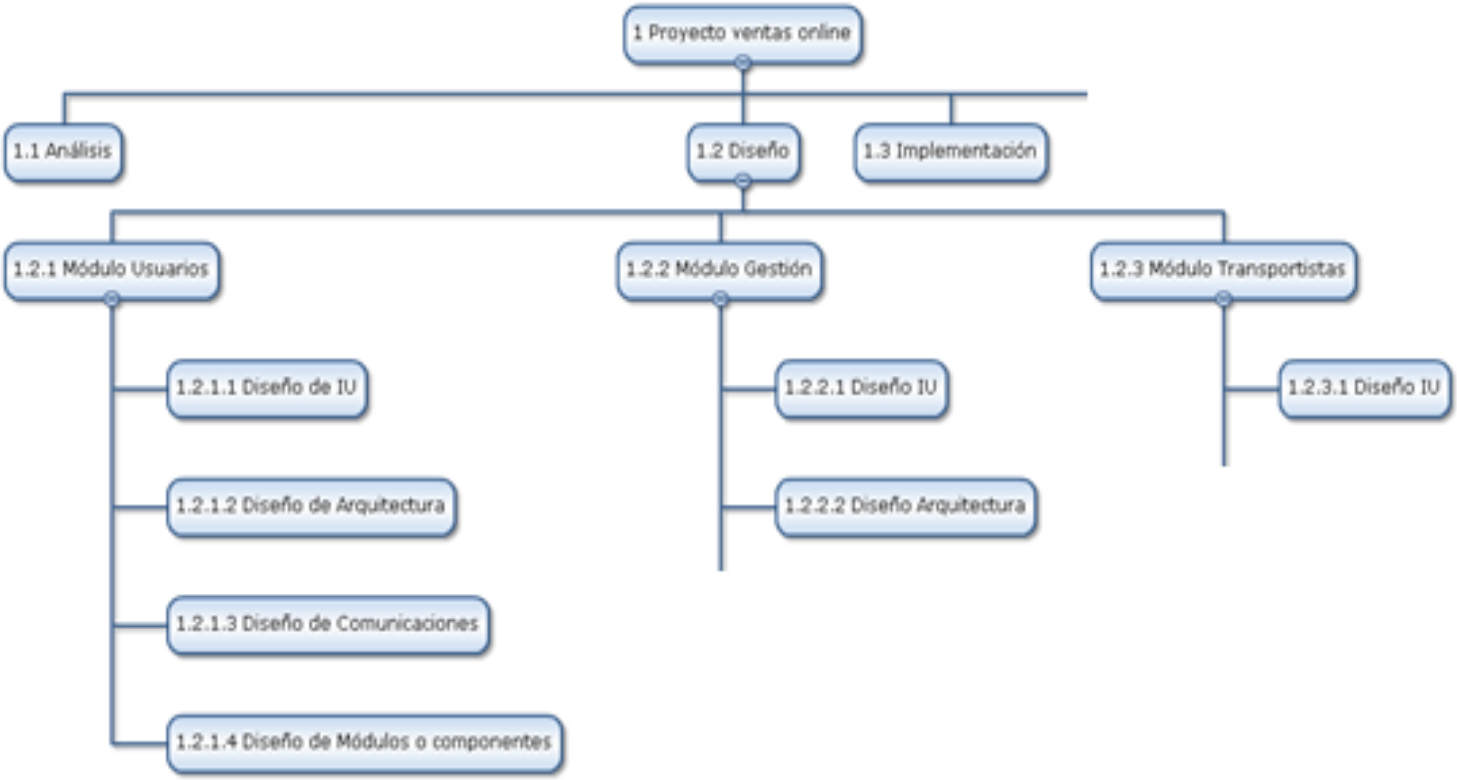

Figura 2: Segundo nivel de desglose (fase de diseño) en base a módulos de la aplicación.

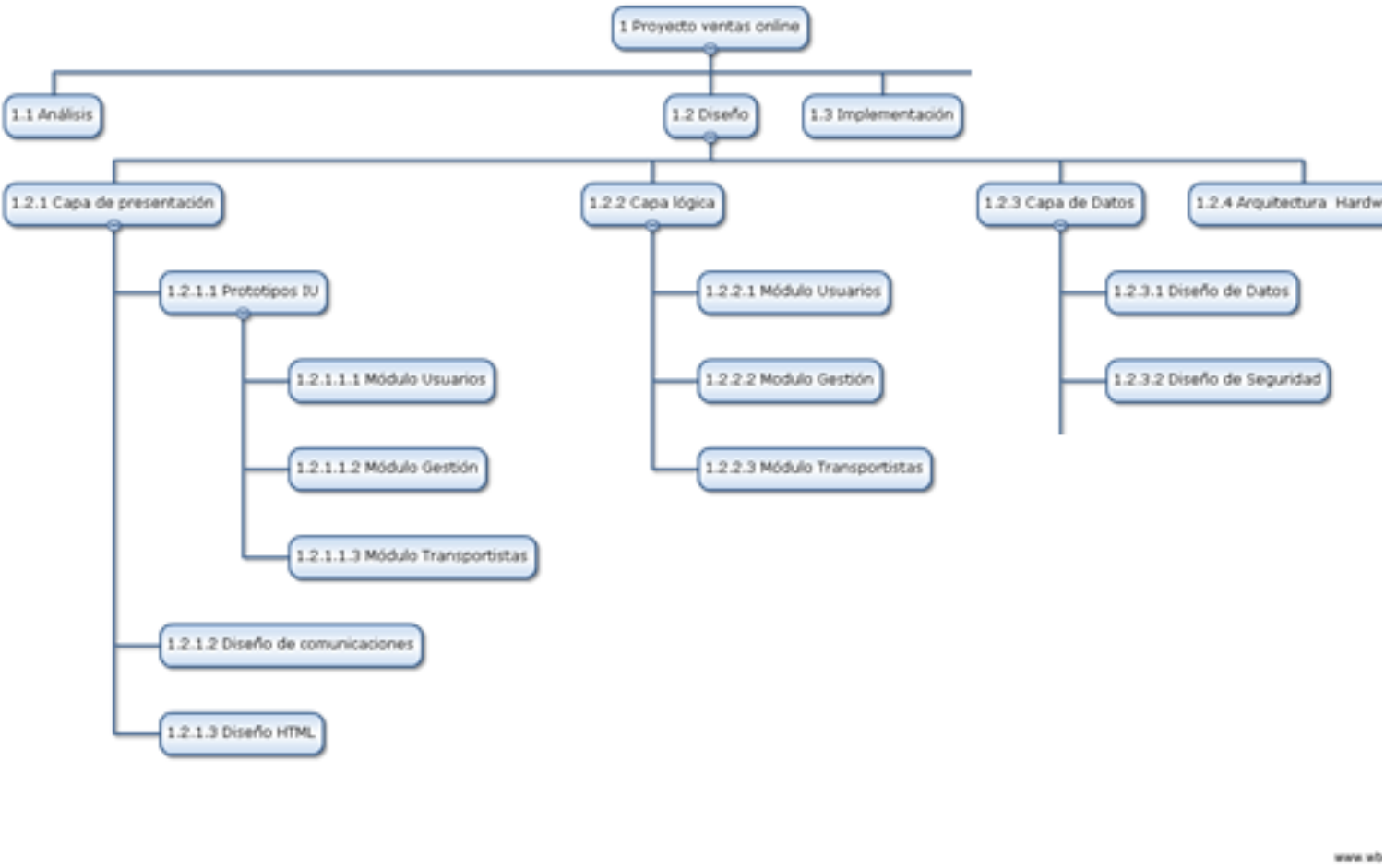

Figura 3: Segundo nivel de desglose (fase de diseño) en base a capas y arquitectura.