**¿Qué entiende por arquitectura de un sistema de información?**

Una arquitectura de aplicaciones describe los patrones y las técnicas que se utilizan para diseñar y desarrollar aplicaciones. La arquitectura le proporciona un plan y las prácticas recomendadas que se deben seguir para diseñar una aplicación bien estructurada.

**¿Cuál es la importancia de la arquitectura en el sistema de información?**

Por un lado, la arquitectura de software es de especial importancia ya que la manera en que se estructura un sistema tiene un impacto directo sobre la capacidad de este para satisfacer lo que se conoce como los atributos de calidad del sistema. Ejemplos de atributos de calidad son el desempeño (que tiene que ver con el tiempo de respuesta del sistema a las peticiones que se le hacen), la usabilidad (que tiene que ver con qué tan sencillo les resulta a los usuarios realizar operaciones con el sistema), o bien la modificabilidad (que tiene que ver con qué tan simple resulta introducir cambios en el sistema). Los atributos de calidad son parte de los requerimientos no funcionales del sistema y son características que deben expresarse de forma cuantitativa (no tiene sentido, por ejemplo, decir que el sistema debe devolver una petición “de manera rápida”, o presentar una página “ligera”, ya que no es posible evaluar objetivamente si el sistema cubre o no esos requerimientos).

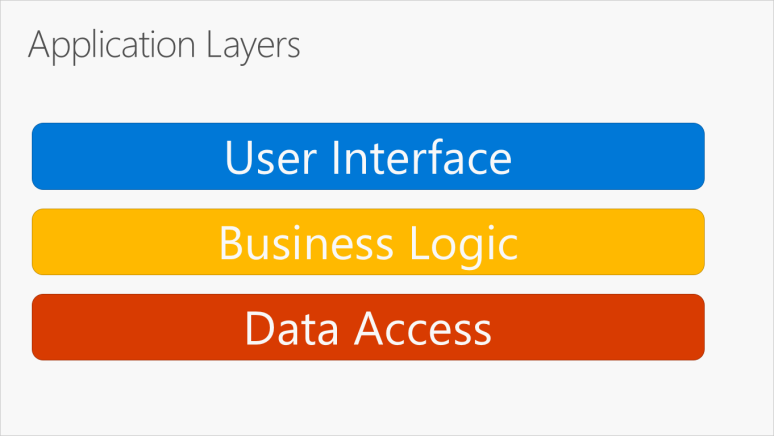
Por el otro, léase el punto siguiente.

**¿Cuáles son las metas de una arquitectura?**

En este sentido, las metas de una arquitectura es casi sinónimo a enunciar sus ventajas:

Cuando aumenta la complejidad de las aplicaciones, una manera de administrarla consiste en dividir la aplicación según sus responsabilidades o intereses. Este enfoque sigue el principio de separación de intereses y puede ayudar a mantener organizado un código base que crece para que los desarrolladores puedan encontrar fácilmente dónde se implementa una función determinada. Pero la arquitectura en capas ofrece una serie de ventajas que van más allá de la simple organización del código. Al organizar el código en capas, la funcionalidad común de bajo nivel se puede reutilizar en toda la aplicación. Esta reutilización es beneficiosa ya que significa escribir menos código y puede permitir que la aplicación se estandarice en una sola implementación, siguiendo el principio Una vez y solo una (DRY). Con una arquitectura en capas, las aplicaciones pueden aplicar restricciones sobre qué capas se pueden comunicar con otras capas. Esta arquitectura permite lograr la encapsulación. Cuando se cambia o reemplaza una capa, solo deberían verse afectadas aquellas capas que funcionan con ella. Mediante la limitación de qué capas dependen de otras, se puede mitigar el impacto de los cambios para que un único cambio no afecte a toda la aplicación. Las capas lógicas son una técnica común para mejorar la organización del código en las aplicaciones de software empresarial, y hay varias formas de organizar el código en capas.

**¿Qué es una arquitectura clásica?**



Es una arquitectura organizada en 3 capas. En la figura se muestra la organización más común de la lógica de la aplicación en 3 capas. Estas capas se suelen abreviar como UI (interfaz de usuario), BLL (capa de lógica de negocios) y DAL (capa de acceso a datos). Con esta arquitectura, los usuarios realizan solicitudes a través de la capa de interfaz de usuario, que interactúa con la capa BLL. BLL, a su vez, puede llamar a DAL para las solicitudes de acceso de datos. La capa de interfaz de usuario no debe realizar solicitudes directamente a DAL, ni debe interactuar con la persistencia de forma directa a través de otros medios. Del mismo modo, BLL solo debe interactuar con la persistencia a través de DAL. De este modo, cada capa tiene su propia responsabilidad conocida. Una desventaja de este enfoque de distribución en capas tradicional es que las dependencias de tiempo de compilación se ejecutan desde la parte superior a la inferior. Es decir, la capa de interfaz de usuario depende de BLL, que depende de DAL. Esto significa que BLL, que normalmente contiene la lógica más importante de la aplicación, depende de los detalles de implementación del acceso a datos (y a menudo de la existencia de una base de datos).

**Defina y describa las capas de un modelo de arquitectura de N-capas.**

Una arquitectura de N-capas es un sistema divido en grupos lógicos de componentes (subsistemas) denominados capas. Vale la pena mencionar que se considera que cada capa termina residiendo en un servidor diferente, o como expresa el apunte, la arquitectura de N-capas es la segmentación de una aplicación en partes y la distribución de esas partes a lo largo de una red. ¿Qué significa esto? Esto equivale a decir que uno de los principios básicos de la arquitectura en capas es lograr un bajo nivel de acoplamiento entre componentes.

**Capa de presentación o IU.** Es responsable de manejar la interfaz gráfica de usuario como vistas renderizadas, archivos HTML de una página web, la interfaz de línea de comandos o los formularios de una aplicación de escritorio.

**Capa de negocio o dominio.** Es el componente más importante, es el corazón de la aplicación, codifica las reglas comerciales y el flujo de trabajo del mundo real, que determinan como se pueden crear, almacenar y cambiar los datos de un objeto comercial. Las reglas comerciales describen las operaciones y restricciones que se aplica a una organización. El flujo de trabajo consta de las tareas, los pasos de procedimiento, la información de entrada y salida requerida y las herramientas necesarias para cada paso de ese procedimiento.

**Capa de acceso de datos o persistencia.** Se encarga de almacenar u obtener datos desde un almacén de datos, como una base de datos, un servicio externo o un archivo plano.

En una arquitectura tradicional, estas son las 3 capas principales de aplicación. Hay una cuarta capa de soporte que también estará siempre presente (sobre todo en las aplicaciones web) y que es la **capa corte transversal**, también conocida como la capa común o capa de soporte.

Ahora bien, si resulta que la capa de dominio es muy compleja (es decir, que tiene un montón de métodos con lógica de negocio) podemos agregar la **capa de servicio** o aplicación, que es una capa delgada que sirve como fachada y envuelve la capa de dominio, y que se encarga del flujo de operaciones, coordina todas las transacciones.

En el apunte, para una arquitectura de 5 capas, se señala:

Una **capa de entidades** (probablemente, la misma capa de corte transversal señalada anteriormente) y que se utiliza para la comunicación entre las capas para llevar y traer datos (las propiedades de las clases a través de parámetros).

Otra **capa de mapper** que convierte objetos a datos relacionales (muy probablemente, un ORM, como suele ser más conocido).

**¿Qué debe cumplir una arquitectura?**

El objetivo final de la arquitectura es identificar los requisitos que producen un impacto en la estructura del software y reducir los riesgos asociados con la construcción del mismo. La arquitectura debe soportar los cambios futuros del software, del hardware y de funcionalidad demandada por los clientes (que ocurren muy a menudo). Del mismo modo, es responsabilidad del arquitecto analizar el impacto de sus decisiones de diseño y establecer un compromiso entre los diferentes requisitos de calidad así como entre los compromisos necesarios para satisfacer a los usuarios, al software y los objetivos del negocio.