vector ITC
a Softtek Company

AaaS: Architecture as a Service.

Editorial

La transformación digital de las empresas ha dejado de ser una necesidad para ser una realidad. El cliente o usuario como consumidor final marca una dirección esencial en todo el proceso, con nuevas necesidades, requisitos y tendencias. La tecnología, y por ende los proveedores de tecnología, deben ofrecer lo que los clientes demandan, y en la manera en la que lo demandan. Incluso, adelantarse y ofrecer soluciones a necesidades que el cliente ni siquiera ha detectado, podría ser determinante.

Tecnológicamente hablando, los sistemas que se necesitan para dar servicio al usuario son cada vez más complejos, heterogéneos e interconectados, lo que hace de la especialización técnica de las personas un "must" en este tipo de compañías, abarcando un amplio abanico de tendencias y necesidades. Y precisamente en esos conocimientos técnicos reside uno de los activos más importantes de los proveedores tecnológicos: contar con profesionales altamente cualificados, con total conocimiento del mercado y las necesidades actuales, y adaptados a trabajar en el marco de las nuevas metodologías de desarrollo, con una clara orientación al cliente y entendiendo que el objetivo es la prestación de un servicio de una alta calidad. Esto se constituye como un hecho diferencial que los clientes aprecian de manera notable.

La base de una buena plataforma tecnológica es su arquitectura de software, con una complejidad creciente en cuanto a tecnología y alcance. Además, es uno de los pilares sobre los que se debe sustentar la evolución de las empresas en cuanto a los futuros desarrollos de sus soluciones internas y de mercado. Esta complejidad hace aún más difícil el desarrollo de esta pieza clave, unido a que durante dichos desarrollos se debe hacer frente a diversas situaciones que hacen de la agilidad un concepto necesario durante el ciclo de vida de estos proyectos. Entonces, ¿por qué no orientar también el diseño, definición e implementación de las arquitecturas de software a una modalidad de servicio, que aúne todas las bondades que nos puede ofrecer un servicio de estas características?

Especialización tecnológica, conocimiento del mercado, talento, adaptabilidad a los cambios, desarrollo ágil, eficiencia económica y conocimiento compartido, son tendencias que definen la AaaS (Architecture As A Service), y que constituyen la aproximación al mercado de Vector ITC para poner al servicio de nuestros clientes a los mejores profesionales con todos los conocimientos y experiencia necesarios para contar con la mejor solución en el menor tiempo posible.

Iván Lastra Quintana
(AaaS Manager, Vector ITC).

vector ITC
a Softtek Company

Índice

01 **04 - 25**

La evolución de las arquitecturas.

02 **26 - 28**

Architecture as a Service (AaaS).

03 **29 - 31**

Nuestro enfoque AaaS.

01. La evolución de las arquitecturas.

Desde los inicios de la ingeniería del software, allá por los años 60 del pasado siglo, se han ido creando una serie de guías y patrones para resolver los diferentes retos que presentaba la construcción de software. Pero no fue hasta los años 90 cuando definitivamente se implantó el concepto de arquitectura del software para denominar esas **bases que describen la estructura, el funcionamiento y las interacciones a seguir a la hora de construir software**. Por tanto, la arquitectura de un sistema va "más allá de los algoritmos y estructuras de datos de la computación; el diseño y especificación de la estructura global del sistema es un nuevo tipo de problema".

A lo largo de la historia tanto la infraestructura como los requerimientos del software han ido cambiando, aumentando la complejidad de los sistemas a medida que se disponía de mejores prestaciones en la infraestructura. La disponibilidad de mayores recursos y más rápidos, además de la experiencia y la innovación en la construcción de sistemas han ido aumentando la complejidad de las arquitecturas que partieron de modelos simples con estructuras sencillas para llegar, hoy en día, a sistemas distribuidos altamente complejos con cientos de elementos y miles de interacciones entre estos.

Este aumento de complejidad requiere necesariamente de una mayor inversión de tiempo y recursos en el diseño de la arquitectura de los sistemas, lo que deriva en **un diseño mucho más estructurado que asegura la mantenibilidad y la escalabilidad**. El incremento en el número de piezas del sistema ha derivado igualmente en la especialización de los profesionales TI que ya no pueden abarcar el conocimiento de la inmensa cantidad de tecnologías que implican un sistema de software complejo.

APLICACIONES MONOLÍTICAS

APLICACIONES CLIENTE-SERVIDOR

SISTEMAS MULTICAPA

ARQUITECTURAS ORIENTADAS A SERVCIOS

ARQUITECTURAS SERVERLESS

ARQUITECTURAS DIRIGIDAS POR EVENTOS

MICROSERVICIOS

ARQUITECTURAS HEXAGONALES

APLICACIONES MONOLÍTICAS

Durante mucho tiempo, las arquitecturas monolíticas fueron populares, a menudo debido al coste de los recursos físicos y la velocidad –lenta– en la que las aplicaciones se desarrollaban y desplegaban. Este tipo de arquitectura encajaba bien con sistemas mainframe, siendo posible encontrar, hoy en día, muchas aplicaciones que se ejecutan bajo el marco de una arquitectura monolítica.

Se puede definir como un gran sistema de software, con todas las capas fuertemente acopladas, que **sólo se puede implementar de una pieza**. Tiene que pasar, en su totalidad, a través de todas las fases del *continuous delivery*: desarrollo, test *stages* y *release*. Debido a su tamaño, estos procesos tardan más tiempo en desplegarse que otros sistemas más pequeños; reduciendo la flexibilidad y aumentando los costes del proceso. Internamente, su desarrollo puede tener una estructura modular; sin embargo, todos los módulos de los cuales se compone deben de desplegarse en producción simultáneamente.

APLICACIÓN CLIENTE-SERVIDOR (FAT-CLIENT)

Dentro del marco de las diferentes arquitecturas, un avance importante fue la **separación entre cliente y servidor**, siendo prácticamente la primera arquitectura distribuida. El cliente –pesado– pasó a ser el responsable de gestionar tanto la interfaz gráfica de la aplicación como la capa de negocio. El servidor, sin embargo, quedó relegado a la administración de la base de datos, utilizando conexiones dedicadas. La comunicación entre ambos, cliente y servidor, se basa en protocolos pesados.

La distribución de software, a través de red o puesto por puesto en los clientes, se caracteriza por ser un proceso complejo y propenso a errores.

APLICACIÓN CLIENTE-SERVIDOR MEJORADA (THIN-CLIENT)

En este modelo, tanto la lógica de la aplicación como la gestión de la base de datos es llevada a cabo por el servidor. El cliente –ligero–, simplemente, es responsable de la capa de presentación.

ARQUITECTURA DE TRES CAPAS

La arquitectura de tres capas establece la organización del modelo de diseño en capas, que pueden estar físicamente distribuidas. Los componentes de cada capa solo hacen referencia a los de la capa inmediatamente inferior. De esta manera se simplifica la organización y la comprensión del desarrollo de sistemas complejos, reduciendo las relaciones de forma que las capas más bajas no requieren detalles o interfaz de las superiores.

El sistema **se divide en tres capas lógicas diferentes**, cada una con un conjunto de interfaces definido. La primera es la capa de presentación, consiste principalmente en una interfaz gráfica de usuario. La capa del medio, la capa de negocio, gestiona la lógica de empresa, y la tercera, es la capa de persistencia, que contiene los datos necesarios para la aplicación. En resumen, la capa de negocio es el código al que recurre la capa de presentación para acceder a los datos necesarios y posteriormente presentarlos.

La separación entre la interfaz de usuario y la lógica de aplicación aporta gran flexibilidad al diseño de la aplicación. Permite la construcción y despliegue de múltiples interfaces sin cambiar en absoluto la lógica de aplicación.

Capa de presentación

Mediante esta capa se interactúa con el usuario; le muestra el sistema, le presenta la información y obtiene información del usuario. Es conocida como interfaz gráfica y debe ser amigable, es decir, entendible y fácil de usar para el usuario. Esta capa se comunica únicamente con la capa intermedia o de negocio.

Capa de negocio

En esta capa se encuentran las funciones que se ejecutan, se recogen las peticiones del usuario, la información se procesa y tras ello se envían las respuestas. Se denomina capa de lógica del negocio o capa de negocio, ya que en ella se establecen las reglas a efectuar. Esta capa se comunica con la de presentación, para recibir las peticiones y mostrar los resultados, y con la capa de acceso a datos, para solicitar al gestor de base de datos almacenar o recuperar datos.

Capa de acceso a datos

Esta es la capa donde se almacenan los datos de los usuarios y del sistema. Su cometido es almacenar los datos y devolverlos a la capa de negocio, aunque para ello en ocasiones es necesario el almacenamiento de procedimientos y funciones en la misma capa. En esta modalidad de arquitectura, esta es la única capa que puede acceder a los datos. Está compuesta por uno o varios sistemas gestores de bases de datos, localizados en uno o varios servidores.

VENTAJAS CON RESPECTO A ARQUITECTURAS PREVIAS

- Aumenta la reutilización de lógica de negocio para diferentes clientes o sistemas.
- Aumenta la independencia de la base de datos.
- Mejora la flexibilidad.
- Mejora la escalabilidad.

ARQUITECTURA MULTICAPA

La evolución de la arquitectura de tres niveles es la arquitectura multicapa (n-layer), esta **separa la lógica de negocio en capas de un sistema software**. Una capa es simplemente un grupo de clases, paquetes o subsistemas con responsabilidades relacionadas dentro del funcionamiento del sistema. Las capas están organizadas jerárquicamente, de modo que cada capa depende exclusivamente de las capas inferiores. Lo más común en backend suele ser: servicio – negocio – acceso a datos, aunque también puede haber una capa de presentación o controladores REST.

- CAPA DE PRESENTACIÓN
- SERVICIOS DE APLICACIÓN
- DOMINIO DE NEGOCIO
- ACCESO A DATOS

> La gran desventaja de este modelo es que la capa final es la de acceso a datos. Esto supone que todas las demás capas, incluyendo la interfaz de usuario, dependen de ella de forma directa o hereditaria.

En este caso el centro de toda la aplicación será la persistencia de los datos. Es cierto que para algunos sistemas puede ser válida o no ser una desventaja muy importante, pero para sistemas más grandes puede llegar a ser un problema y por eso surgieron otras arquitecturas que intentaban solucionarlo, aunque no dejan de ser una versión ligeramente modificada para aplicar correctamente el principio de inversión de dependencia (DIP). Este es uno de los principios SOLID, que dice que se debe depender de abstracciones y no de clases concretas.

Siguiendo este concepto, se crearon arquitecturas multicapa que sí seguían el principio DIP y que, por tanto, daban solución a la dependencia de la capa de datos. Esta es la forma en la que se suele representar la **arquitectura onion**.

Básicamente la arquitectura onion es una arquitectura multicapa que está construida alrededor de un modelo de dominio, el código depende de las capas centrales y no de las externas.

Las capas más cercanas al núcleo son las interfaces de repositorio, estas definen el comportamiento del almacenamiento de datos, sin implementarlo. La siguiente capa es la lógica de negocio, la cual utiliza las interfaces y realiza las implementaciones apropiadas durante la ejecución.

En este modelo puede haber un número distinto de capas, pero siempre teniendo en cuenta que las interfaces deben estar más cerca del núcleo que las clases que las utilizan.

Finalmente, en la capa exterior se encuentran la infraestructura, la interfaz de usuario y los tests de integración. Las clases de esta capa implementan las interfaces de las capas inferiores. De esta manera el cambio de implementaciones que dependen de tecnología no afecta a capas inferiores.

Lo que se consigue con esto es una arquitectura basada en cómo está montado el sistema, y no en los terceros que se comunican con él.

VENTAJAS CON RESPECTO A ARQUITECTURAS PREVIAS

- Disminuye el coste de administración de clientes.
- Se incrementa la independencia con la base de datos.
- Alta disponibilidad y tolerancia a fallos.
- Alta flexibilidad.
- Alta escalabilidad.
- Alta accesibilidad.

ARQUITECTURA SOA

La SOA, Arquitectura Orientada a Servicios, deriva de la computación distribuida, que está basada en el modelo de pregunta-respuesta para aplicaciones sincrónicas y asincrónicas. Con SOA, la lógica de negocio o **las funciones individuales son modulares y son presentadas como servicios para las aplicaciones**. La clave se encuentra en separar la lógica de integración de negocio, de la implementación.

La estrategia es centrarse en la creación de servicios, en vez de aplicaciones, aunque estos deben definir correctamente la funcionalidad de la aplicación que la requiera. Esta arquitectura plantea una aplicación final para que dirija la ejecución del conjunto de servicios, agregue una lógica propia y se lo presente al usuario final mediante una interfaz.

Al exponer procesos de negocio como servicios se aporta una gran flexibilidad a la arquitectura, ya que permite que piezas de funcionalidad puedan utilizar otros servicios. Por ello, el sistema evoluciona con su mejora continua y mediante la agregación de nuevos servicios, que evolucionan de forma independiente.

Una solución SOA definirá los servicios que componen el sistema, sus interacciones y las tecnologías utilizadas para su implementación.

No son desarrolladas grandes aplicaciones complejas, sino componentes reutilizables (servicios) fáciles de mantener y testear. Para la creación de aplicaciones, se diseña un proceso que interactúa con los servicios, en el que los componentes se reutilizan y solo son desarrollados los componentes no implementados.

El denominador común de todos los sistemas de la organización de la Arquitectura Orientada a Servicios es la necesidad de una infraestructura de comunicación que sea escalable y segura. Esta es conocida como Enterprise Service Bus.

El ESB proporciona una integración de aplicaciones rápida, directa y fundada en patrones. Es un conjunto de productos que no dependen de la infraestructura y permite facilitar el procesado, el enrutamiento, la transformación de datos y la organización de procesos utilizando Servicios Web.

Las aplicaciones orientadas a servicios son multicapa, característica de las aplicaciones distribuidas. En el caso de SOA **la clave está en la capa de servicios y la capa de procesos de negocio**.

Capa de servicios

Los servicios son similares a componentes EJBs u objetos Java de arquitecturas orientadas a objetos tradicionales. Sin embargo, los servicios son autocontenidos, con estado y poco acople. Al analizar los requisitos de negocio, hay que tener en cuenta los servicios existentes y cuáles son los componentes que corresponde ensamblar a partir de ellos. Por lo general los servicios deben tener una funcionalidad significativa.

Capa de procesos de negocio

Una de las principales ventajas de esta arquitectura es la orquestación de servicios, es decir, la estandarización de la configuración de procesos de negocio. Puede crearse una capa de servicios web fundada en la abstracción de sistemas legacy y utilizarse en el ensamble de procesos de negocio.

VENTAJAS CON RESPECTO A ARQUITECTURAS PREVIAS

- Administración y monitoreo de procesos, servicios e infraestructura.
- Manejabilidad de procesos de negocio.
- Aumenta en gran medida la escalabilidad.
- Distribución.
- Disponibilidad.
- Heterogeneidad.

ARQUITECTURA DE MICROSERVICIOS

En base a los principios SOA, la Arquitectura de Microservicios aparece con la intención de dotar de una mayor autonomía y responsabilidad propia a los servicios y ofrecer de esta forma: un menor acoplamiento y alta cohesión entre módulos, un sistema mucho más descentralizado, un aumento del grado de escalabilidad horizontal y, en consecuencia, disminuir la complejidad que ofrecía el Enterprise Service Bus de SOA.

Un microservicio es un **componente software que puede ser desplegado, evolucionado y actualizado de forma independiente**. Para ello debe encapsular la lógica necesaria para cubrir una funcionalidad completa, es decir, debe ser capaz de autogestionarse.

En general, estas funcionalidades deben ser pequeñas. La arquitectura se basará en tener un gran número de componentes comunicándose unos con otros mediante algún protocolo ligero (HTTP, generalmente).

CARACTERÍSTICAS DE LOS MICROSERVICIOS:

Alta cohesión y bajo acoplamiento entre módulos.

Como cada *microservicio* es un proceso distinto es posible desplegarlos en máquinas diferentes, consiguiendo un alto grado de **escalabilidad horizontal**.

Una vez definidas las interfaces de comunicación, **facilitan el desarrollo en paralelo y la optimización de recursos**.

Por aplicación del principio de responsabilidad única los microservicios son relativamente **sencillos de modificar y mantener**.

Dada la independencia de cada microservicio, **pueden desarrollarse con tecnologías distintas**.

El software construido en base a microservicios se podrá descomponer en **módulos funcionales autónomos**. Cada microservicio contará con los recursos necesarios para operar de forma independiente. Por ejemplo, con su propio contexto o base de datos.

Debe ser un **sistema descentralizado** donde cada módulo podrá ser detenido, modificado y redesplegado sin comprometer los otros aspectos funcionales de la aplicación.

PRINCIPIOS ARQUITECTÓNICOS:

Bajo acoplamiento. Los cambios llevados a cabo en un servicio no deberían suponer cambios en otros. Funcionalidad expuesta a través de interfaces independientes de la implementación interna.

Debe existir un **mecanismo de aviso y actuación en caso de que alguno de los módulos falle**, filtrando la información destinada a él y favoreciendo la gestión del resto de recursos.

Los servicios deben exponerse como **recursos fáciles de localizar y utilizar en la arquitectura empresarial**. Su propósito y capacidades deben ser fácilmente interpretables, tanto manual como automáticamente.

Principio de **responsabilidad única**. Cada componente debe tener responsabilidad sobre una sola parte de la funcionalidad del sistema al que pertenece. Dicha funcionalidad debe estar encapsulada en su totalidad.

EJEMPLO DE ARQUITECTURA DE MICROSERVICIOS

VENTAJAS CON RESPECTO A ARQUITECTURAS PREVIAS

- Alta cohesión.
- Escalabilidad horizontal.
- Heterogeneidad de tecnologías de implementación.
- Disminución considerable del acoplamiento.
- Sencillez para modificar y mantener componentes debido a su gran independencia.
- Simplificación de la complejidad que incluía el ESB de SOA.

ARQUITECTURA DIRIGIDA POR EVENTOS (EDA)

Una arquitectura dirigida por eventos (en inglés, Event-driven architecture – EDA) es un **patrón de diseño que permite a un conjunto de sistemas comunicarse entre sí de forma reactiva**, mediante la publicación y el consumo de eventos.

Los eventos son sucesos o cambios en el estado de un elemento del sistema, y pueden originarse por estímulos internos (cuando se ha terminado un procesamiento) o externos (la actividad de un usuario). No hay que confundir un evento y su notificación, que se puede entender como un mensaje entre los distintos elementos del sistema para comunicar que se produjo un evento.

Es una arquitectura con un bajo nivel de acoplamiento, lo que la convierte en una buena opción para arquitecturas de aplicaciones distribuidas modernas. Este patrón convierte en una realidad la creación de arquitecturas con respuesta en tiempo real.

Una arquitectura dirigida por eventos está compuesta por consumidores y productores de eventos, y la relación entre ellos puede estar basada tanto en modelo publ/sub como en un modelo de event streaming (flujo de eventos).

MODELO PUB/SUB:

Este modelo se basa en suscripciones a un flujo de eventos. Cada vez que se publica un evento en el canal, se envía a los suscriptores de dicho evento.

Algunos casos de uso de este modelo son:

- Actualización de cachés distribuidas.
- Distribución de notificaciones de eventos.
- Implementación de flujos de trabajo asíncronos.
- Balanceo de cargas de trabajo en clústeres de red.
- Transmisión de datos de varios procesos o dispositivos.

MODELO DE FLUJO DE EVENTOS:

En este modelo los eventos se escriben en un registro, y son los consumidores los que pueden leerlo o unirse a él en cualquier momento.

VENTAJAS CON RESPECTO A ARQUITECTURAS PREVIAS

Es asíncrona, y por tanto sin bloqueos y con alto rendimiento.

Los servicios están desacoplados, no necesitan saber o depender de otros servicios.

Una EDA con una cola puede recuperar trabajo perdido al "repetir" eventos del pasado.

El escalado resulta muy sencillo, ya que cada procesador de eventos puede escalar por separado.

ARQUITECTURA HEXAGONAL

La arquitectura hexagonal (dada a conocer por Alistair Cockburn) y también conocida como arquitectura de puertos y adaptadores, se pensó para garantizar un alto grado de abstracción entre sus componentes añadiendo capas para este fin. Además, garantiza calidad, rendimiento y escalabilidad del software a lo largo del tiempo, y es una de las denominadas "**Clean Architecture**" por el alto grado de abstracción entre sus componentes.

VENTAJAS CON RESPECTO A ARQUITECTURAS PREVIAS

Construcción de software de calidad gracias a sus pautas.

Facilita la prueba de componentes de forma unitaria.

Permite que no esté acoplado al método de entrega, facilitando que el caso de uso funcione en una aplicación móvil, web tradicional, API Rest, web SPA por API Rest, etc.

Gracias al uso del patrón SOLID se fomenta el uso de patrones de diseño altamente efectivos.

Obliga a que la lógica de negocios, o el core business, esté desacoplada de la tecnología o framework.

ARQUITECTURA SERVERLESS

Por último, cabe destacar la gran tendencia hacia Arquitecturas Serverless. El concepto Serverless hace referencia a una organización, un framework y una metodología. El patrón de infraestructura como servicio nace con la intención de reemplazar las máquinas virtuales, servidores físicos y contenedores sólo con código.

Es una metodología relativa a proveedores. El framework Serverless es una **herramienta multiproveedor que mediante la línea de comandos realiza tareas de forma automática**. Una gran ventaja es que en los servicios cloud no importa el hardware a la hora de desarrollar aplicaciones.

Este servicio permite alojar una aplicación que, al desarrollarse ciertos eventos, será ejecutada. No es necesaria la administración de servidores y escala de forma automática para ajustarse a la demanda.

El coste será en función de los recursos consumidos, sin preocuparnos por insuficiencia de servicio (picos de demanda) por recursos inicialmente mal definidos o un exceso de estos recursos con los consiguientes gastos innecesarios.

Actualmente, existen varios proveedores cloud que ofrecen estos servicios, los más conocidos son Azure Functions, Amazon Lambda y Google Cloud Functions.

Amazon Lambda

Google Cloud Functions

Azure Functions

Además, hay gran cantidad de **proyectos open source** relacionados con arquitecturas serverless como Kubeless, OpenFaaS, Fission o Riff. Estos marcos de trabajo se han diseñado para ser desarrollados sobre clusters de Kubernetes, sobre Swarm o sobre Docker. Ofrecen un entorno, tanto en nube pública como privada, donde ejecutar fragmentos de código y permiten gran cantidad de lenguajes como Node.js, Python, Java, Ruby, etc. Se integran con los principales proveedores, admiten sistemas de mensajería como Kafka y pueden adaptarse a herramientas de monitorización como Prometheus.

VENTAJAS CON RESPECTO A ARQUITECTURAS PREVIAS

Pago por consumo.

Enorme flexibilidad y casi infinita escalabilidad.

Elimina la infraestructura física y el mantenimiento que conlleva.

Rapidez y sencillez en preparar/configurar la infraestructura para una aplicación que se reduce a la ejecución de unos comandos.

02. Architecture as a Service (AaaS).

Hoy en día las empresas se encuentran en un mercado cambiante con una competencia feroz y un entorno comercial volátil, que obliga a demostrar una gran rapidez y agilidad a la hora de adaptarse a los requisitos de sus clientes. Los sistemas TI de las empresas deben acompañar este proceso de adaptación con la misma o mayor celeridad para poder mantenerse en este ritmo frenético. Esto lleva en la mayoría de los casos a la **necesidad de modernización de los sistemas** tanto a nivel de infraestructura como a nivel arquitectural.

Para acelerar los desarrollos de aplicaciones, existen dos claras aproximaciones con diferentes tendencias. A un lado se sitúa el uso de nuevas metodologías como las metodologías ágiles, DevOps, agilismo empresarial, etc. Por otro lado, para resolver mejor las necesidades de negocio, actualmente se utilizan diferentes tecnologías, introduciendo **nuevos campos a la construcción de software** como la Ciberseguridad Big Data, Microservicios, Business Intelligence o los despliegues en Cloud, ya sea público, privado o híbrido, utilizando los elementos TI como servicios (IaaS, PaaS y SaaS).

Para la aplicación de estas últimas tendencias tecnológicas las empresas requieren de arquitectos altamente cualificados que revisen las arquitecturas actuales junto con las necesidades del negocio para plantear nuevas arquitecturas más modernas que permitan una **mayor velocidad de desarrollo y características como la mantenibilidad y la escalabilidad**. La necesidad de conocimiento en multitud de nuevas tecnologías y ámbitos de la arquitectura de software lleva a pensar que es casi imposible que una única persona sea capaz de tener conocimientos suficientes en todas las áreas necesarias en el desafío de una modernización arquitectural en la tecnología de las empresas.

Es precisamente por esta necesidad por la que aparece en concepto de **Arquitectura como Servicio (Architecture as a Service o AaaS)**. Este concepto se basa en utilizar los conocimientos de arquitectura como un servicio y no como la contratación de un arquitecto o consultor de arquitectura.

La AaaS se basa en ofrecer no sólo el conocimiento de un arquitecto para evaluar y definir un cambio de arquitectura tecnológica, sino en ofrecer todo **un equipo de arquitectos especialistas en diferentes ámbitos de la tecnología**, representados por uno o varios arquitectos de proyecto que recogerán toda la información del sistema actual, las necesidades y requisitos del negocio y llevarán toda esta información a una mesa de arquitectura, donde los expertos en las diferentes áreas analizaran todos los datos para ofrecer una solución conjunta que resuelva las necesidades planteadas.

Por tanto, la implementación de la AaaS se realizará por un equipo compuesto por uno o varios arquitectos de proyecto, que trabajarán en conjunto con los diferentes expertos de la empresa para establecer cuáles son las necesidades y los requisitos de la evolución. Estos arquitectos de proyecto trasladaran la información a la mesa de arquitectura, coordinada por un Manager de arquitectura, donde se diseñará un planteamiento de evolución y de convivencia con los sistemas actuales aportando los conocimientos de expertos en diferentes ámbitos de la tecnología, **asegurando una propuesta completa e innovadora que cubra las necesidades** planteadas con las mejores soluciones posibles.

La metodología de trabajo se compone habitualmente de **dos fases**:

DEFINICIÓN

Una fase de definición, donde uno o varios de los arquitectos del departamento son asignados para la toma de contacto inicial. Recogerán información sobre el estado inicial del proyecto y establecerán las necesidades específicas. En proyectos que comiencen de cero, la asignación de estas personas será del 100%.

Los arquitectos asignados irán contrastando periódicamente la información recogida con la Mesa de Arquitectura, donde se irán detectando nuevas necesidades y resolviendo las dudas que puedan plantearse, proponiendo la mejor solución que será trasladada y consensuada con el Responsable Técnico del proyecto por parte del cliente. Esta fase tiene como objetivo la definición de la nueva Arquitectura y el Roadmap que seguirá su implementación.

IMPLEMENTACIÓN

Una fase de implementación de lo definido en la fase anterior, donde los arquitectos asignados realizarán un seguimiento de la implementación, ejerciendo labores de consultoría y replanteando la solución en caso de que surgieran nuevas necesidades.

En este caso el objetivo de la fase será desarrollar la nueva Arquitectura TI.

03. Nuestro enfoque AaaS.

Vector ITC, como proveedor líder de servicios tecnológicos y digitales, ha puesto sus esfuerzos durante los últimos tiempos en contar con un equipo de arquitectos especialistas en las últimas tendencias y tecnologías, lo que ha permitido estructurar un equipo con un compendio de skills necesarios para la atención de las demandas de todos sus clientes.

El equipo de arquitectura formado por Vector ITC cuenta con una estructura que permite abordar con éxito las problemáticas y los proyectos de nuestros clientes, desde la definición, diseño e implementación de arquitecturas desde cero, hasta la evolución tecnológica de las arquitecturas legacy, pasando por la toma de decisiones de la mejor opción a implementar en función de las necesidades estratégicas de las compañías.

La apuesta de Vector ITC por esta modalidad de servicio está completamente interiorizada, ofreciendo ya unos **resultados muy positivos, tanto en nuestros clientes, como en los propios equipos de desarrollo**, que ven en el equipo de arquitectura un grupo en quienes apoyarse para cualquier proyecto.

Este aspecto es algo fundamental en el crecimiento de nuestros profesionales, constituyendo una retroalimentación de capacidades y talento, aprovechables a su vez en nuevos proyectos y clientes.

¿Cómo planteamos nuestro servicio AaaS?

La forma más habitual de llegada a un cliente es, **desde los inicios de los proyectos o incluso en las fases de preventa, contar con uno o varios arquitectos** de soluciones, con gran experiencia y muy buenas capacidades de entendimiento de requisitos y conocimiento del mercado, para identificar las primeras necesidades del cliente, y detectar la orientación de hacia dónde debe ir tecnológicamente y estratégicamente hablando. A partir de este momento, el equipo de soporte, en las mesas de arquitectura, trabajará a fondo cada detalle, cada pieza, cada decisión, para diseñar y definir la arquitectura más adecuada y, una vez ésta sea consensuada y aprobada, dirigirá la implementación de la misma, así como la implantación en la infraestructura seleccionada que más se adapte a los requisitos.

Nunca una única persona en exclusiva es conocedora de un proyecto; la información se comparte en las mesas de arquitectura, con el objetivo de eliminar las dependencias clásicas de según que perfiles, un aspecto que es fundamental cuando hablamos de una figura tan clave como un arquitecto, aportando fiabilidad y confianza a los clientes, así como una tranquilidad de tener detrás a un equipo completamente conocedor de cada uno de los detalles del proyecto.

Finalmente, para la puesta en producción del proyecto, nuestro equipo de arquitectura define y ejecuta las diferentes pruebas de seguridad y de rendimiento necesarias, así como da soporte o realiza directamente la implantación en los entornos de producción si así lo requiere el cliente.

Somos expertos en servicios, productos y soluciones digitales, en innovacion tecnológica y en procesos ágiles de transformación.

Con un equipo humano de **más de 2.500 profesionales**, tenemos presencia a nivel nacional e internacional, con sedes en EE. UU., Perú, Brasil, Chile, México, Colombia, Paraguay, Reino Unido y Alemania.

¿Nuestra misión? Generar el mayor valor a nuestros clientes, a los sectores económicos y al conjunto de la sociedad, mediante el diseño y desarrollo de iniciativas **basadas en tecnología de vanguardia**, con el objetivo de accionar el cambio digital de una manera disruptiva.

www.vectoritcgroup.com

vector ITC
a Softtek Company