

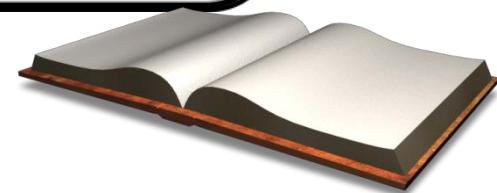
Arquitecturas en Big Data

Agenda

- Diseño de arquitecturas
- Requerimientos
- Diseño de Modelos
- Herramientas
- Arquitecturas de Referencia

¿Definición?

Arquitectura de IT: *La arquitectura de un Sistema describe su ‘estructura estática y su comportamiento dinámico’. Modela los elementos del sistema (software, hardware y usuarios), las ‘propiedades externas que muestran’ esos elementos y las relaciones estáticas y dinámicas entre ellos.*



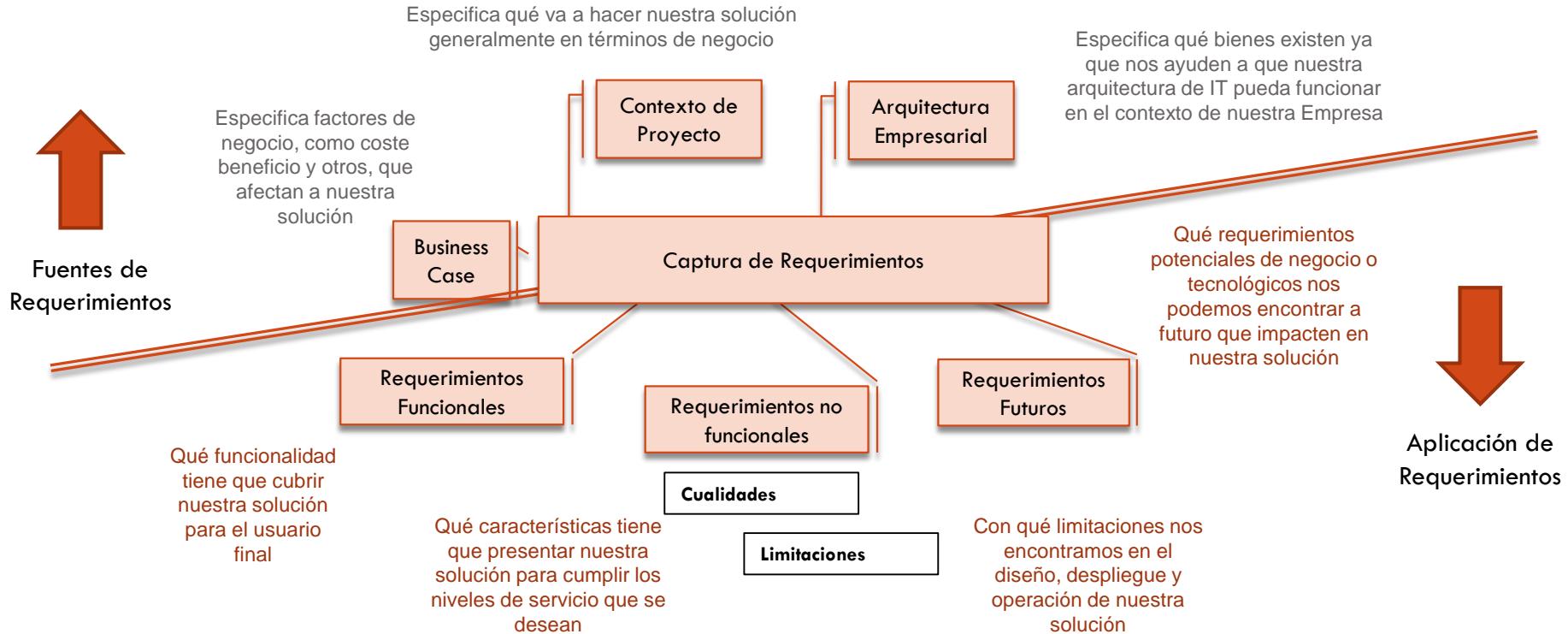
¿Elementos y Sistemas?

- En general la arquitectura puede estar relacionada con la solución a cualquier escala
- Entendamos los conceptos “estructura”, “comportamiento” y “elementos”:
 - ¿Qué es un Sistema?
 - ¿Qué es un Elemento?
- Casi es todo una cuestión de “escala”:
 - Los elementos de un Sistema de Alta Fidelidad son el sintonizador, amplificador, altavoces, cables y controles remotos.
 - Los elementos de un amplificador son su caja, la fuente de alimentación y un circuito impreso.
 - Los elementos del circuito impreso serán resistencias, fusibles y circuitos integrados
 - Los elementos de un circuito integrado son transistores y otros componentes



¿Por dónde empezamos?

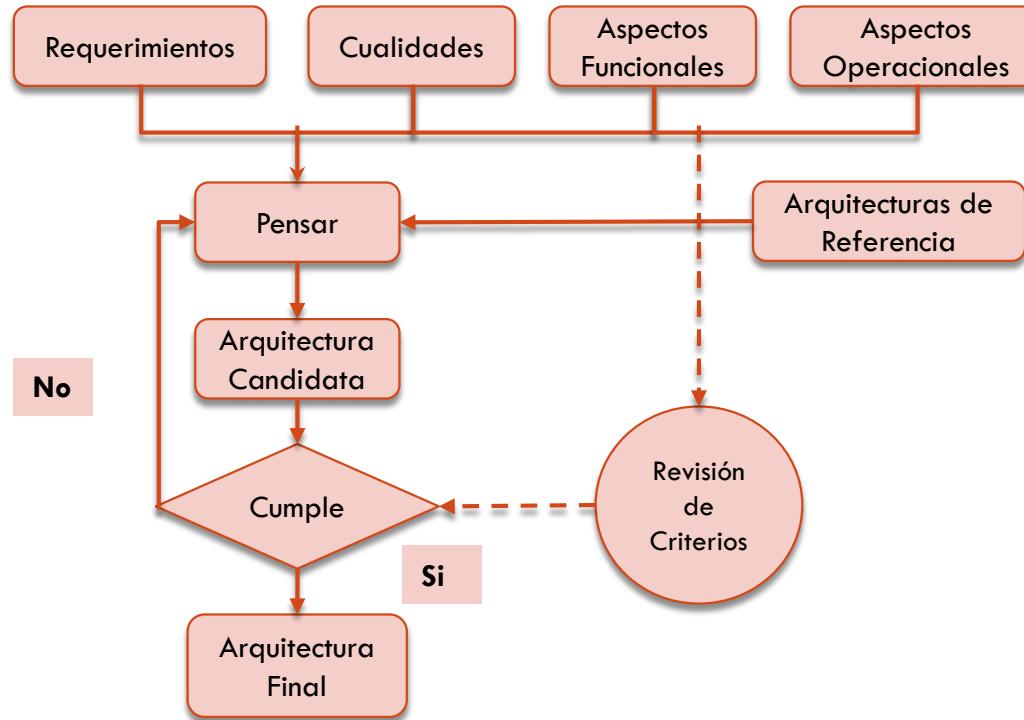
Identificamos una oportunidad de mejora, un proyecto, un nuevo objetivo



Vamos a practicar. Esta información es...

- “Queremos recibir la información de las ventas en tiempo real”
- “Financieramente no podemos plantearnos otra plataforma que cloud”
- “Necesitaremos que soporte 140TPS”
- “Esperamos un crecimiento compuesto del 15% anual”
- “En mi empresa abunda el desarrollo en JAVA en plataforma Linux con servidor de aplicaciones Tomcat”
- “Tenemos 400K€ disponibles para invertir”

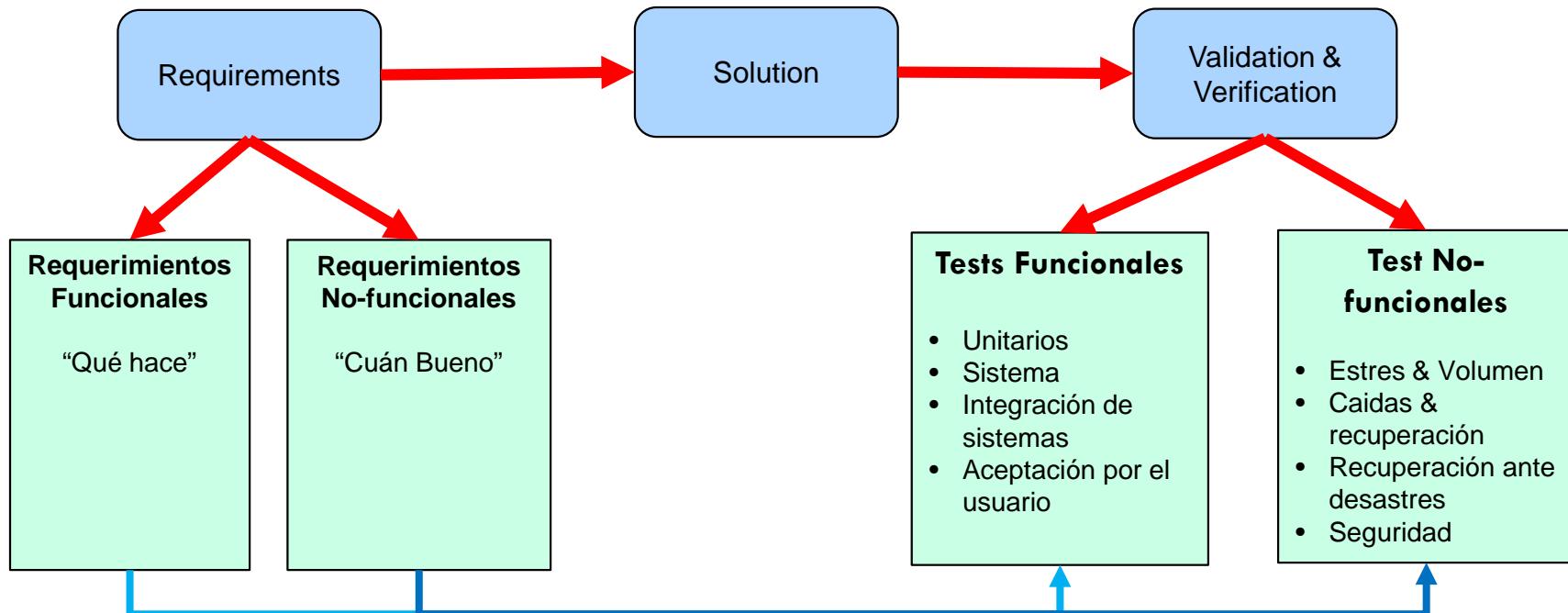
¿Y cómo diseñamos una solución?



El diseño de arquitecturas es un proceso cíclico siempre adaptado a una problemática de negocio en particular, con unos requerimientos, características y unos aspectos funcionales y operacionales a cumplir

Sigamos...

- ¿Por qué son tan importantes los requerimientos



Clasificación

Requerimientos Funcionales

- Capacidades necesarias por los usuarios de la solución para cumplir con el objetivo
- Responde a “qué” tiene que hacer la solución (pero NO como se alcanza)

Requerimientos NO-Funcionales

- **Cualidades:**
 - Define las expectativas y características que debe cumplir el Sistema IT
 - Pueden ser Runtime (en tiempo de ejecución, por ejemplo rendimiento o disponibilidad) o no de tiempo de ejecución (por ejemplo escalabilidad o mantenibilidad)
- **Limitaciones:**
 - Son aquellos aspectos que no pueden ser cambiados dentro del alcance y el tiempo de vida del Proyecto (por ejemplo la infraestructura actual)
 - Tambien pueden ser otros factores como skills disponibles, presupuesto o limitaciones operacionales
 - Tambien pueden actuar como tales las Guías de Arquitectura y tecnologías estándares (tal y como pueden haber sido declaradas en la Arquitectura de empresa)

Requerimientos Futuros

- Podrán ser tanto funcionales como no-funcionales

¿Y cómo son los requerimientos?

De forma individual tienen que ser:

- **Specific:** Sin ambigüedades, coherentes y con el nivel de detalle adecuado
- **Measurable:** Es posible verificar que se ha cumplido un requisito, así que podremos definir los criterios de éxito/cumplimiento.
- **Attainable:** Alcanzable técnicamente y dentro de lo posible
- **Realizable:** Realista dadas todas las limitaciones (p.e. recursos, skills, tiempo, infraestructura)
- **Traceable:** Posibilidad de seguir su evolución y cumplimiento

En grupo, deberán ser:

- Priorizables
- Que no provoquen conflictos unos con otros

Qué otras características debe tener

1. Debe estar definido en una sentencia completa
2. Evitemos utilizar abreviaciones a menos que luego las definamos
3. Debe hacerse un uso consistente de verbos como “debe”, “puede”, “podría”...
4. Debe indicarse los criterios de éxito y ser medibles y testeables.
5. Deberán tener un identificador único: NFR-PERF-001.
6. Deberemos evitar:
 - Ambigüedad (seamos claros y explícitos)
 - Largas frases con perífrasis
 - Deseos (“100% de confianza”, “totalmente seguro”, etc.)

Ejemplo. ¿Cuál es el origen del siguiente requerimiento?

“Este Proyecto depende de que el Proyecto X pase a producción”

- Enterprise architecture
- Business case
- Project context

Ejemplo. ¿Cuál es el origen del siguiente requerimiento?

“La solución propuesta debe tener un período de amortización de 3 años”

- Enterprise architecture
- Business case
- Project context

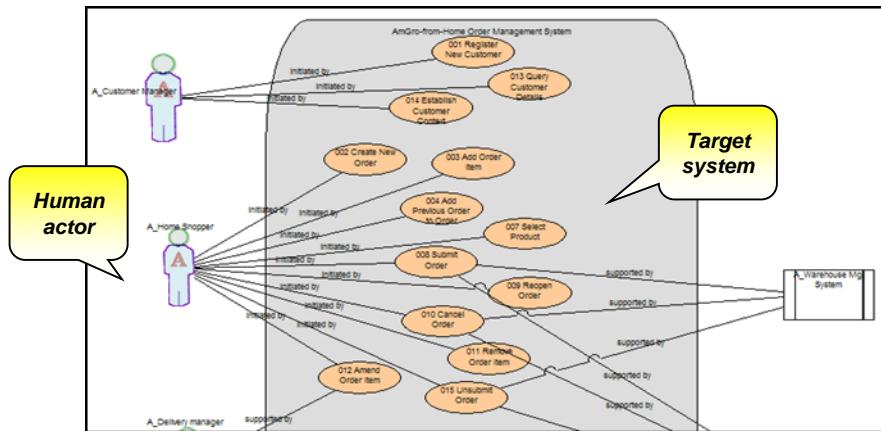
A comprobar

¿Es éste un ejemplo de un requerimiento SMART?

- “***La solución debe ser mejor que el Sistema actual***”

Requerimientos Funcionales

- La estructura, descripción y documentación son muy variadas y algunas de ellas se apoyan en una solución industrial específica
- Una de las metodologías es la de Modelización a través de Casos de Uso



Por ejemplo

➤ Escenario principal

- El Sistema solicita los detalles a la persona.
- El usuario suministra la dirección y el número de teléfono.
- El Sistema valida la dirección y el número de teléfono
- El caso de uso finaliza correctamente.

La representación del Caso de Uso debe ser paso a paso: Actor dice, Solución dice...

No confundamos cada paso (lo que hace cada parte) con el mensaje que fluye entre ellos en cada paso

En la Metodología de Caso de Uso se pueden heredar Casos de Uso, reutilizarlos (encadenarlos), etc.

➤ Alternativas

- 3a. El nombre de la ciudad suministrado en el punto 2 no es correcto
- 3a1. El Sistema suministra una lista de nombres y/o códigos postales que encajan con los aportados (para resolver ambigüedad)
- 3a2. El usuario selecciona una entrada en la lista.
- 3a3. El Sistema aporta la dirección y el número de teléfono.
- 3a4. El caso de uso finaliza correctamente.

Requerimientos No Funcionales. Cualidades

Las de tipo “runtime” tienen valor para el usuario del sistema

- Pueden ser testeadas empíricamente (por ejemplo monitorizando el Sistema)
- Algunas se establecen en características del nivel de servicio (y serán las bases del SLA)
- También se les llaman cualidades observables

Las de “nonruntime” tienen valor para el operador del sistema

- También se conocen como propiedades no-observables o en tiempo-de-desarrollo
- Algunas no pueden ser medidas fácilmente y pueden tener que apoyarse en otras métricas

Clasificación de NFR

Categoría	Descripción o ejemplos
Volumetría: <ul style="list-style-type: none">• Estática• Dinámica	Volumetría para las entidades de datos que estén dentro del Sistema que, aunque sean relativamente estáticos probablemente tengan un efecto significativo en el dimensionamiento del sistema P.e. Número de clientes, pólizas, suscriptores...
Rendimiento	(Transacciones por Segundo), tiempo de respuesta
Reliability; Horas de disponibilidad	P.e.: 99.5% 8:00 to 19:00
Escalabilidad	Vertical u Horizontal
Seguridad	P.e. Autenticación, Autorización, Confidencialidad, Integridad, No-repudio en origen, Envío
Calidad de Servicio	E.g. Entrega garantizada, envío único (no duplicidades)
Programación en el tiempo y Temporalidad	Limitaciones en el tiempo debido a procesos de negocio (cierre de tiendas, fin de transacciones). Programación en el tiempo de eventos (sobre todo batch)
Monitorización e instrumentación	Requerimientos para facilidades específicas, funciones, herramientas, etc. para operar y gestionar el sistema

Clasificación de NFR

NFR Category	Descripción o ejemplos
Logging y timestamp	Qué se registra, y qué timestamps se utilizan para ser capaces de recrear la secuencia de eventos
Cumplimiento y auditabilidad	Datos a ser registrados para soportar las auditorías
Regulaciones y legal	Funciones y parámetros operacionales que necesitan cumplirse en términos regulatorios y legales
Archivado, backup y recuperación	¿Qué dato debe ser mantenido activo y cual se archiva?, ¿cuál es el período de retención? ¿Cómo se puede inspeccionar los datos archivados? ¿Cuantos y qué datos se deben hacer salvaguardas, cual es el periodo de retención del backup?
Recuperación ante desastres	Recovery Point Objective (RPO), Recovery Time Objective (RTO)
Soporte, Mantenimiento	Requerimientos para funciones específicas y herramientas para operar el Sistema y dar soporte a los usuarios
Lenguajes	Lenguajes y configuraciones culturales soportadas
Usabilidad	Parámetros para definir cuán fácil es utilizar el Sistema (curva de formación requerida)
Regulación medioambiental	Cumplimiento con la regulación medioambiental

A practicar

- ¿Qué tipo de NFR lo definimos como “capacidad para responder a un número mayor de usuarios aumentando los recursos”?
 - A. Disponibilidad
 - B. Mantenibilidad
 - C. Rendimiento
 - D. Escalabilidad
 - E. Seguridad

Aspectos Funcionales y Operacionales

➤ El Modelo Funcional

- Describe la estructura de los componentes de software (de todos los tipos)
- Su comportamiento dinámico (colaboración)
- Y sus interfaces

➤ El Modelo Operacional:

- Incluye la topología de red (nodos de ejecución/hardware, localización, etc.)
- Describe qué se ejecuta en ellos (colocaremos los componentes)
- Asegura que se alcanza los niveles de servicios requeridos (rendimiento, disponibilidad, etc.)
- Y describe la gestión y operación del Sistema IT

Un poco de terminología para ayudarnos

Modelo Funcional

➤ Componente IT

Módulo con una funcionalidad y datos determinados al que se accede por una o más interfaces.

➤ Interface

A través de la cual un componente se comunica a través de una o más operaciones que utilizarán la funcionalidad y los datos del componente.

➤ Colaboración

Intercambio de mensajes entre varios componentes en un escenario en particular.

➤ Interacción

Intercambio de mensajes entre dos componentes en una colaboración, incluyendo la secuencia de esos mensajes.

Modelo Operacional

➤ Nodo de IT

Colección de componentes de hardware y software con una responsabilidad y niveles de servicio específicos

➤ Conexión

Capacidad de conectividad entre los nodos

➤ Características de los nodos

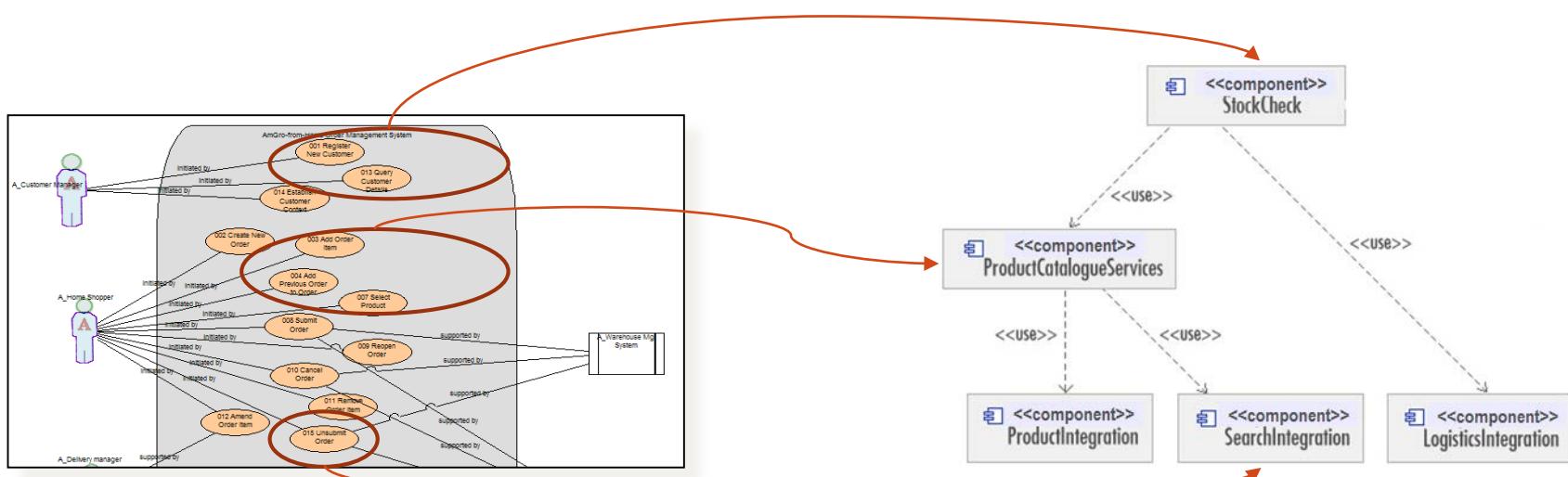
Es una abstracción del componente que utilizamos para modelar cada nodo con respecto a su capacidad de ejecución, presentación y datos

➤ Localización

Dónde se localizan los nodos (y sus componentes)

Empecemos con los Componentes

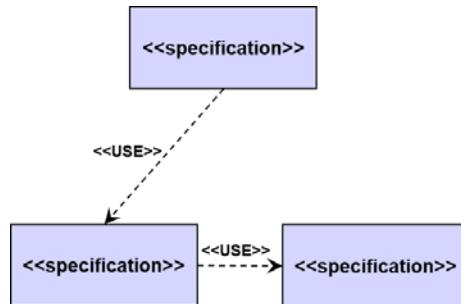
- Con los casos de uso que se han preparado (que es como hemos reflejado los requerimientos funcionales) hacemos una primera identificación de componentes



Representación de componentes

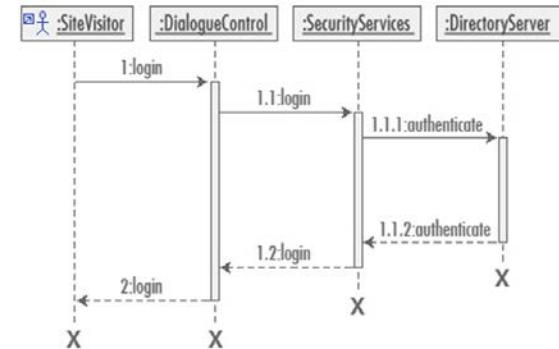
- UML está especialmente diseñado para poder representar los componentes

UML Class Diagram



Visión Estática

UML Sequence Diagram



Visión Dinámica

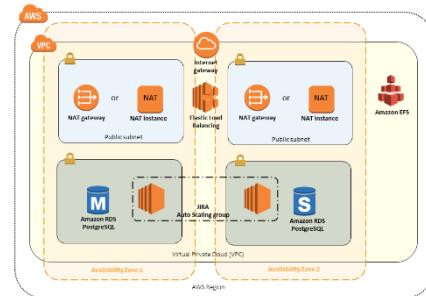
Modelo Operacional y Requerimientos No Funcionales

- Hasta ahora tenemos una lista de requerimientos no funcionales pero necesitamos aplicarlos o considerarlos de forma controlada
- Los requerimientos no funcionales hacen referencia a la solución de forma general, en su totalidad (no específicamente componente a componente que es lo que hemos definido posiblemente hasta ahora)
- La aproximación de la solución, sobre todo en arquitecturas en cloud, comienza, en muchos casos, repasando la oferta actual del cloud del proveedor que estudiemos

¿Y cómo los aplicamos?

- Queremos considerar los Requerimientos No-Funcionales de una manera que nos permita asegurarnos que los consideramos todos (con cierta prioridad)
- La lista que nos encontramos (o recopilamos) habitualmente puede ser extensa y necesitamos tratarla de forma que nos sea factible su aplicación a los componentes...

Categoría	Descripción o ejemplos
Volumetría:	Volumetría para las entidades de datos que estén dentro del Sistema que, aunque sean relativamente estáticos probablemente tengan un efecto significativo en él
• Estática	
• Dinámica	
NFR Category	Descripción o ejemplos
Logging y timestamp	Qué se registra, y qué timestamps se utilizan para ser capaces de recrear la secuencia de eventos
Reliability; Horas de disponibilidad	Complimiento y auditableidad Datos a ser registrados para soportar los auditórios
Regulaciones y legal	Funciones y parámetros operacionales que necesitan cumplirse en términos regulatorios y legales
Escalabilidad	Archivado, backup y recuperación ¿Qué dato debe ser materializado y cuál se archiva?, cuál es el periodo de retención? ¿Cómo se puede inspeccionar los datos archivados?
Seguridad	¿Cuántos y qué datos se deben hacer solvengardos, cuál es el periodo de retención del backup?
Calidad de Servicio	
Programación en el tiempo y Temporalidad	Recovery Point Objective (RPO), Recovery Time Objective (RTO)
Soporte, Mantenimiento	Requerimientos para funciones especiales y herramientas para operar el Sistema y dar soporte a los usuarios
Lenguajes	Lenguajes y configuraciones culturales soportados
Usabilidad	Parámetros para definir cuán fácil es utilizar el Sistema (curva de formación requerida)
Regulación medioambiental	Cumplimiento con la regulación medioambiental



Cualificación de los componentes

- La experiencia ha mostrado que cualificar un componente se debe hacer teniendo en cuenta las siguientes **características de despliegue**:

Presentación

Punto de entrada visual (o acceso) utilizado por actores para acceder a la funcionalidad de los componentes

Datos

Los datos necesarios por el componente (perfil de usuario, películas, etc.)

Ejecución

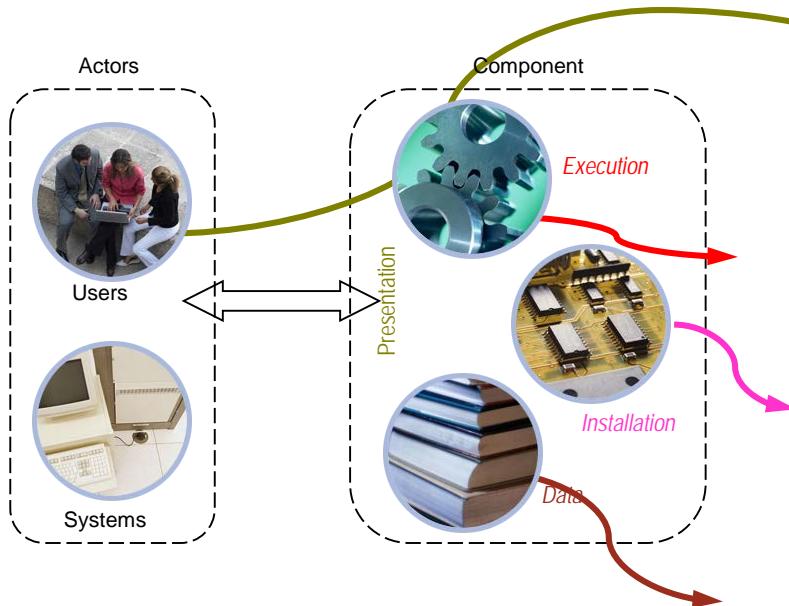
El entorno de ejecución de un componente (por ejemplo, tres instancias de un procesador de textos en el mismo puesto de trabajo)

Instalación

Ficheros requeridos para instalar el componente (ejecutables, configuración y datos)

- En algunos entornos se considera otra característica específicamente técnica respondiendo a componentes de hardware con microcódigo específico tipo tarjetas de cifrado, routers, etc.

Avancemos en esa definición. Volcamos los NFR



Presentación:

Naturaleza del Pto. De acceso	Actor	Localización	Horas requeridas	Concurrentes activos
-------------------------------	-------	--------------	------------------	----------------------

Ejecución:

Tipo de Txn	Ventana de uso	Número por usuario	95% tiempo de respuesta
-------------	----------------	--------------------	-------------------------

Instalación:

Tamaño	Frecuencia de cambios
--------	-----------------------

Datos:

Tipo	Unidad/Tamaño de registro	Número de registros	Volatilidad	Currency
------	---------------------------	---------------------	-------------	----------

Veamos un ejemplo

¿Cómo desplegarías una solución de Ofimática tipo Procesador de Textos (tipo MS-Word o OpenOffice) en una empresa?

(Inicialmente partimos de que puede haber un único componente aunque enseguida veremos que la implementación me va a suponer descomponer su funcionalidad en otros componentes)

Todo en el PC

Opción 1



Presentación
Ejecución
Data
Instalación



Procesador de textos
Ejecutándose en un único punto

Opción 2



Presentación
Ejecución



Data
Instalación

El Procesador de Textos se ejecuta en local con Datos
Suministrados por un servidor de ficheros

Opción 3



Presentación
Data



Ejecución
Instalación

Procesador de textos ejecutándose en Citrix con datos en local

O nada en
el PC

Opción 4



Presentación



Ejecución
Data
Instalación

Se accede al procesador desde un thin client.

Y qué identificamos

Cualificación

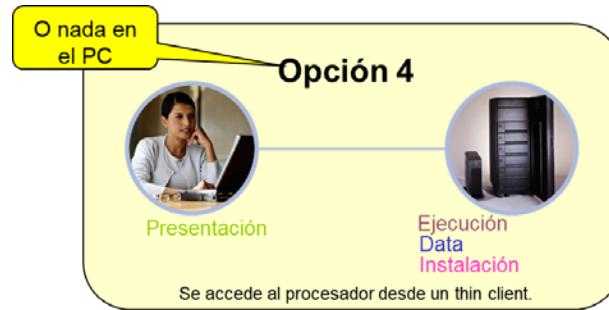
Procesador de Textos

Presentación	Datos	Ejecución
U_offline	D_Plantillas_corporativas	E_word_processor
U_online	D_Plantillas_datos_sensibles	
	D_Perfil_de_usuario	
	d_Plantillas_corporativas	Instalación
	d_Plantillas_datos_sensibles	I_word_processor

La D (mayúscula) indica copia original y la d (minúscula) copia del original

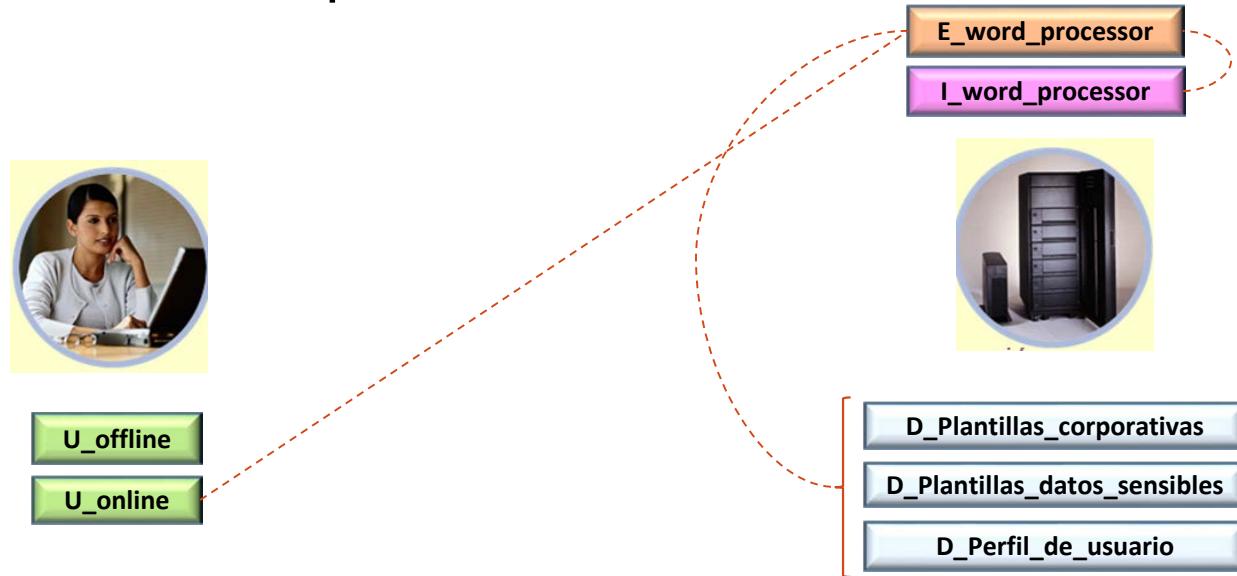
El siguiente paso es...

- Asignar esos requerimientos no funcionales a cada uno de los componentes de la arquitectura que he diseñado y observar su cumplimiento
- Es posible que, inicialmente, no se pueda conocer el nivel de cumplimiento. En ese caso hay que evaluar si es conveniente hacer una Prueba de Concepto o de Tecnología



Despliegue de características

- El despliegue de estas características permite identificar referencias cruzadas entre ellas, referencias que ponen de manifiesto su interdependencia



¿Qué otros documentos respaldan mi solución?

- Cualquiera de los escenarios vistos en el caso del despliegue de la solución de procesador de textos es tecnológicamente válido, sin embargo posiblemente no debamos proponer todos ellos al cliente o nosotros tengamos que seleccionar alguno o algunos de ellos
- Para ello nos ayudarán:
 - Decisiones de Arquitectura
 - Principios, Políticas y Guías de Diseño

Decisiones de Arquitectura

- Permiten al arquitecto formalizar con un documento las elecciones críticas que tiene que tomar en la creación de las soluciones.
- Las decisiones de arquitectura resuelven elecciones de diferente índole como:
 - Si la estructura será centralizada o distribuida
 - Asignar alguna funcionalidad a un componente y no a otro
 - Seguir un estándar u otro o ninguno
 - Cumplir esta calidad o esa limitación
- Las decisiones de arquitectura:
 - Hacen explícito el razonamiento y la justificaciones de las decisiones hechas
 - Ayudan a asegurar la mantenibilidad y ampliación del sistema
 - Evita un trabajo innecesario de reincidir sobre temas dentro del ciclo de vida de la solución

Qué incluye el documento

- Deberá describir en un lenguaje entendible por el equipo técnico y todos los colaboradores (incluidos los patrocinadores) por qué la solución es cómo es
- Deberán ser documentadas correctamente utilizando algún tipo de plantilla que indique, como mínimo:
 - Título de la decisión de arquitectura
 - Identificador único
 - Problema o inconveniente
 - Consideraciones
 - Alternativas
 - Decisión
 - Justificación
- Un ejemplo...

En documento

2.2 AD02: Choice of Point of Sale (PoS) Client

Subject Area	PoS Client Technology	Topic	Client
Design Decision	What should be the underlying technology to develop the PoS Client?	Id.	AD02
Issue or Problem Statement	<ul style="list-style-type: none">Client infrastructure constraint includes the continued use of the legacy PoS. Legacy PoS models have limited system resources e.g. 32 MB of RAM.The current PoS Client is in VC++ and is tightly coupled with other component.		
Assumptions	<ul style="list-style-type: none">Legacy PoS model 4614-111 will not be supported in the future.All other legacy PoS models can be upgraded to standard configuration		
Motivation	<ul style="list-style-type: none"><i>Performance:</i> PoS client technology should support standard and faster communication between the PoS and Application Server.<i>Integration:</i> PoS client should be able to integrate with the transport.		
Alternatives	<ol style="list-style-type: none">Thick Client (Custom Development)<ul style="list-style-type: none">Java Swing - Common technology for the development of PoS clients<ul style="list-style-type: none">System Requirements: Runtime environment requires 32MB RAM.Uses RMI protocol that will enable fast communicationVC++ - Expedites development & runs efficiently on legacy PoS models<ul style="list-style-type: none">Enables realization of a rich GUI experience.Support has been withdrawn by ISV Microsoft in the year 2005..NET Based (C#) - Fast development technology and Rich GUI<ul style="list-style-type: none">Integrates with transport (WMQ) and via Web ServicesIBM Lotus Expeditor<ul style="list-style-type: none">Will expedite development of the PoS client however has high system configuration needs.Thin Client (Browser)<ul style="list-style-type: none">Provides a thin interface to PoS operations and simplest client to maintain.OS constraints limit leveraging recent browser versions that could support AJAX. Client side validations will be restricted to Javascript and will be cumbersome to maintain.Integration with peripherals would need to be verified.		
Decision	Alternative 1 - .NET Based (C#) client for all PoS models.		
Justification	<ul style="list-style-type: none">Thin browser based clients will not support the usability and speed that the PoS operators are used to. Additionally, does not provide constructs to customize the PoS keysIBM Lotus Expeditor will require higher system resources to perform, which will not be feasible in the legacy PoS that can support a maximum of 128 MB of RAM		
Implications	System Resources on legacy PoS will have to be upgraded to standard configuration.		
Parties Agreeing to the Decision	Person 1 (Representing the solution architecture team) Person 2 (representing the ARB) Person 3 (Representing Enterprise Architecture Standards team)		

Descripción

- **Decisión de arquitectura:** Se describe como una declaración. Por ejemplo: “El Sistema seguirá el patrón de integración por mensajería para la comunicación con la aplicación XYZ”
- **Identificador Único:** Para identificarla sin duda. Por ejemplo: “AD-065.”
- **Problema:** Una descripción de la situación debida a la cual se hace la necesidad de tomar una decisión. Por ejemplo: “¿Cómo se debe integrar la aplicación XYZ?”
- **Consideraciones:** Lo que creemos que es verdad sobre el contexto del problema, limitaciones, etc. Por ejemplo: “La aplicación XYZ soporta interfaz de mensajería”.
- **Alternativas:** Si no hay alternativas no puede haber una decisión de arquitectura. Si no, se deberán recoger una lista de alternativas y su explicación: “1 – Transferencia de ficheros, 2 – Mensajería.”
- **Decisión:** La decisión tomada. Por ejemplo: “Se elige la alternativa 2.”
- **Justificación:** Por qué se ha tomado esa decisión. Por ejemplo: “La Mensajería es más segura cuando se realiza una comunicación asíncrona entre programas”

Además hay políticas, principios y guías (I)

Políticas, principios y guías establecen un marco para poder llevar a cabo el diseño.

Políticas

- Una declaración a nivel empresarial de como se deben hacer las cosas. A menudo asociado con opciones legales impuestas desde fuera de la empresa. Su no cumplimiento puede conllevar multas financieras por ejemplo.
- Relativas a protección de datos, salud, seguridad, regulaciones de la industria
- No se pueden conceder excepciones: hay que encontrar una alternativa para garantizar el cumplimiento de la política.

Principios

- Una regla empresarial que utilizará una organización para tomar decisiones de arquitectura. Por ejemplo: Distribuciones de datos que pueden afectar a las organizaciones, “Comprar en vez de desarrollar” en el entorno de Desarrollo de Aplicaciones, Single Sign-On que impacte en la gestión de seguridad
- Se pueden conceder excepciones: Aunque normalmente se escribe “Se debe...”, los principios pueden entrar en conflicto o estar en desacuerdo con los requisitos.

Además hay principios, políticas y guías (II)

Guías

- Rara vez vinculante. Es una declaración de mejores prácticas para toda la empresa, que ofrece a los arquitectos de soluciones la oportunidad de adoptar enfoques coherentes con los retos arquitectónicos.
- Podrían ser los mismos ejemplos que los principios, pero faltarían motivaciones explícitas y declaraciones de implicación.
- No se requieren excepciones formales.

¿Y como coexisten?

- Las decisiones y los principios tienen muchas veces el mismo efecto. Ambos representan alguna condición o limitación del diseño que nuestra solución debe cumplir.
- El origen, sin embargo, es muy diferente:
 - Las decisiones de arquitectura se toman dentro del contexto de desarrollo de la arquitectura de la solución.
 - Los principios de arquitectura suelen ser impuestos desde fuera del contexto de la solución de la arquitectura. A menudo forman parte de la Arquitectura de Empresa.
- En general ambas tienen el mismo peso. Pero puede ocurrir que entren en conflicto con lo que será necesario recurrir a procesos de gobierno del proceso para tomar la decisión correcta
- Los principios son definidos a nivel empresa e su totalidad.
- Si la empresa reconoce que una decisión de arquitectura tomada en un proyecto tiene la suficiente relevancia la puede adoptar como un principio de arquitectura.

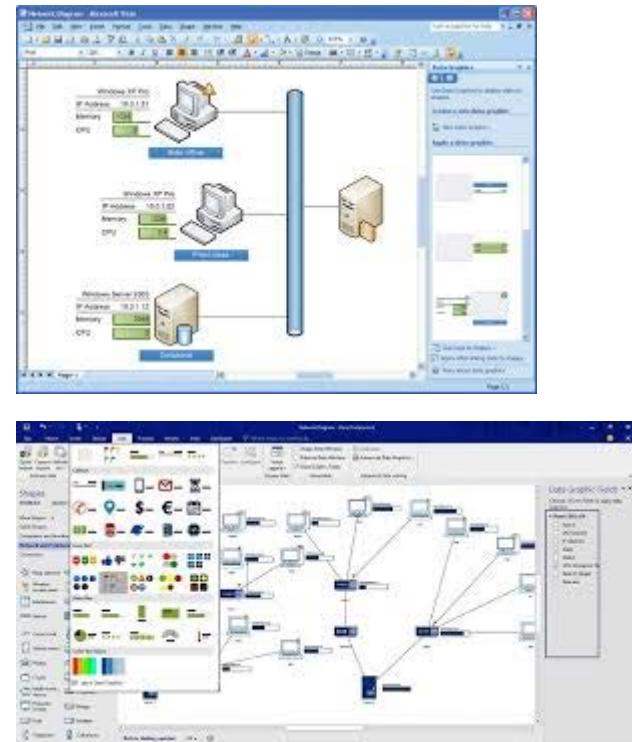
Documentación ejemplo que respalda el diseño

- Diagrama de Arquitectura o Visión general de Arquitectura
- Diagrama de Contexto de Sistema o Solución
- Casos de Uso para representar los Requerimientos Funcionales de la solución
- Modelo Funcional (representado habitualmente con UML)
 - Diagrama de Componentes
 - Diagrama de Interacción de componentes
- Modelo Operacional
 - Esquemas de la implantación final de acuerdo a los puntos de vista que debamos presentar: red, escalabilidad, seguridad, operación, etc.

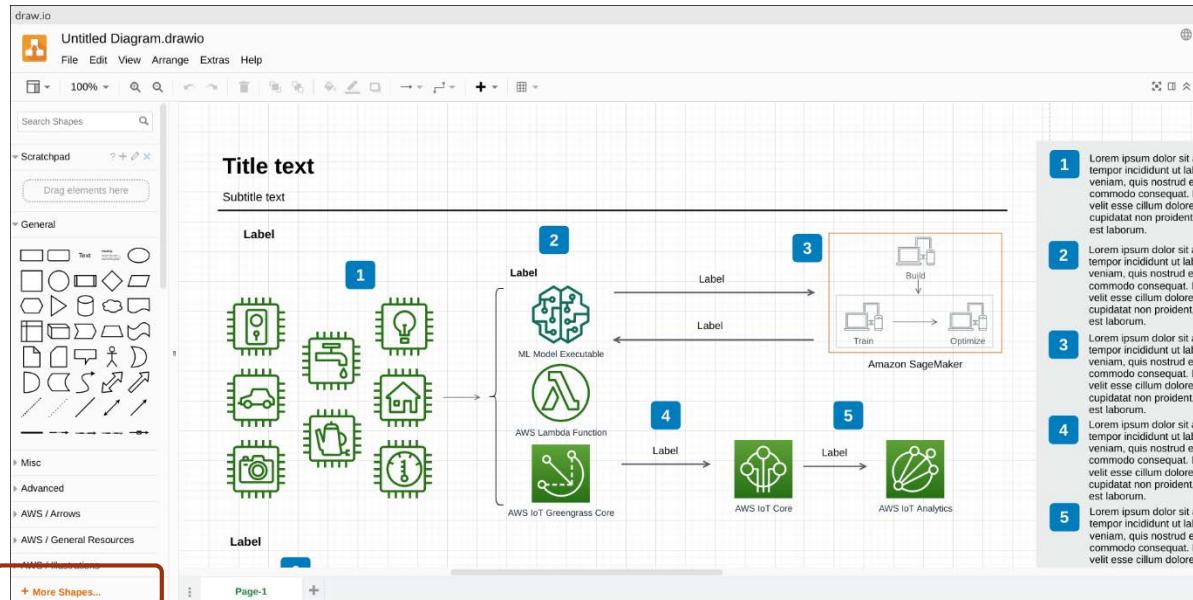
Ver Apéndice y el apartado de Arquitecturas de Referencia para algunos ejemplos

Herramientas para esquemas de arquitectura

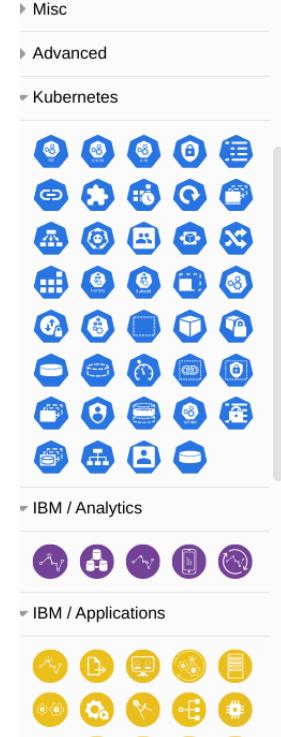
- Herramientas de propósito general:
 - PowerPoint, LibreOffice
- Específicas:
 - MS Visio



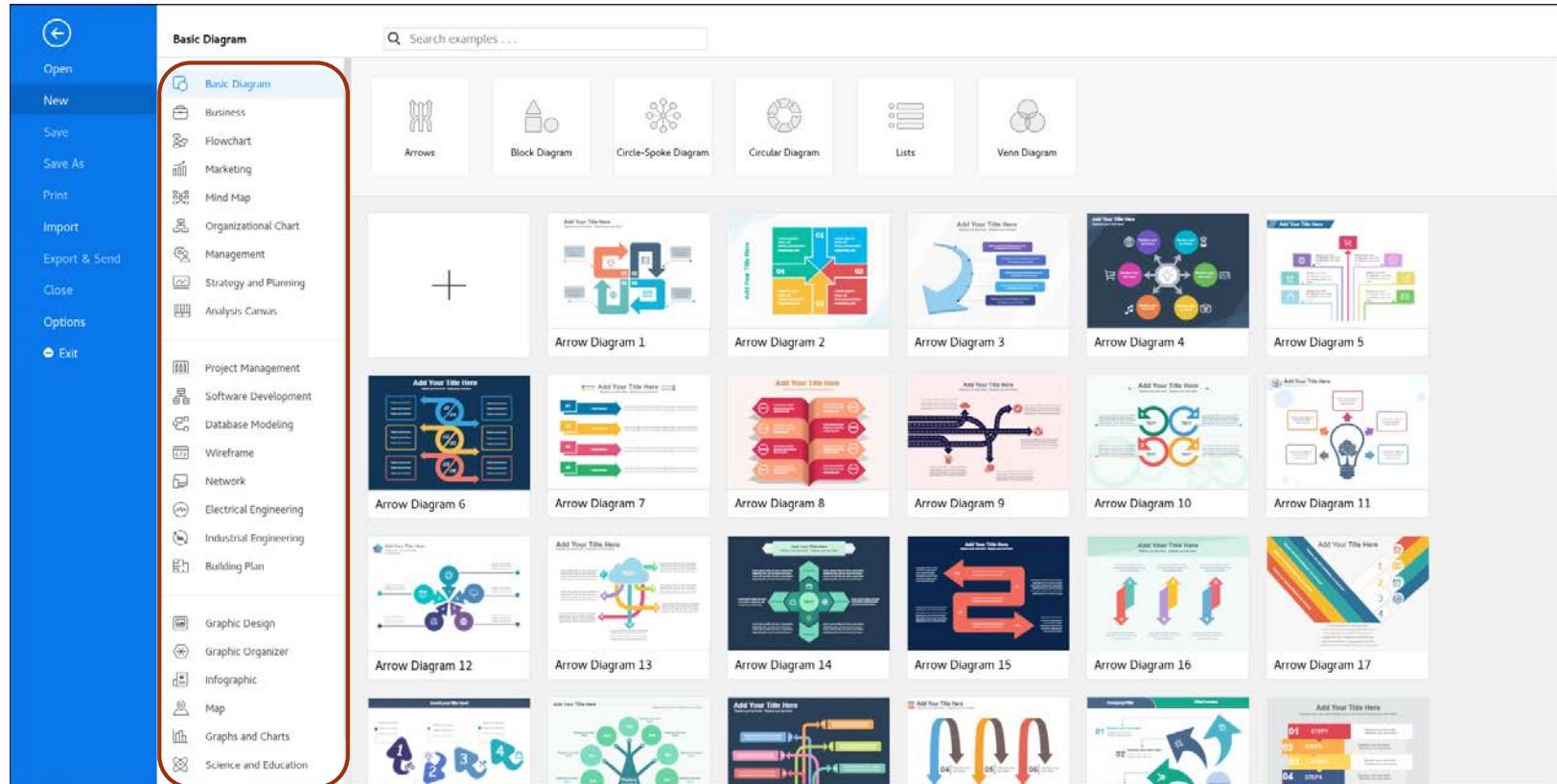
Draw.io



Multitud de colecciones de figuras a poder añadir: Software, Networking, Business, Generales, etc.

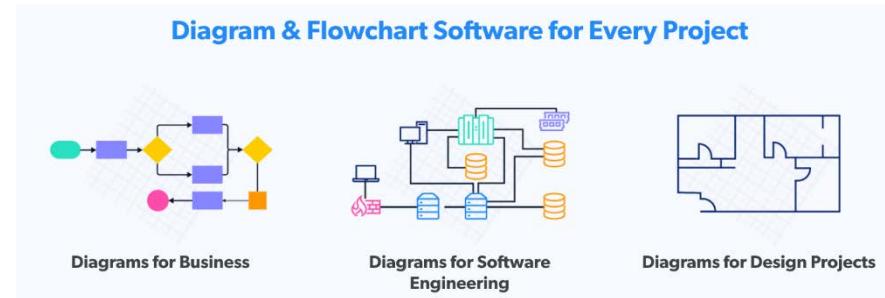


Edrawmax



Otras herramientas (no gratuitas)

➤ Gliffy



➤ LucidChart



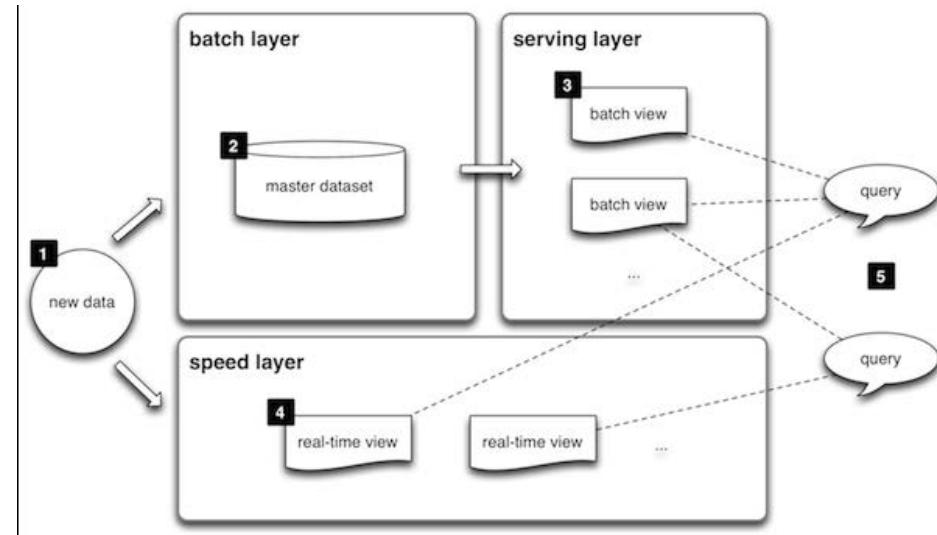
Arquitecturas de Referencia. Características

- Implementaciones basadas en una interpretación funcional muy orientada a plataformas abiertas:
 - Arquitectura Lambda
 - Arquitectura Kappa
- Orientadas a plataforma en un específico hardware o software:
 - IBM Big Data & Analytics Reference Architecture
 - Cloud: AWS, IBM Cloud
 - Lenovo Big Data architecture for Hortonworks

Arquitectura Lambda

<http://lambda-architecture.net/>

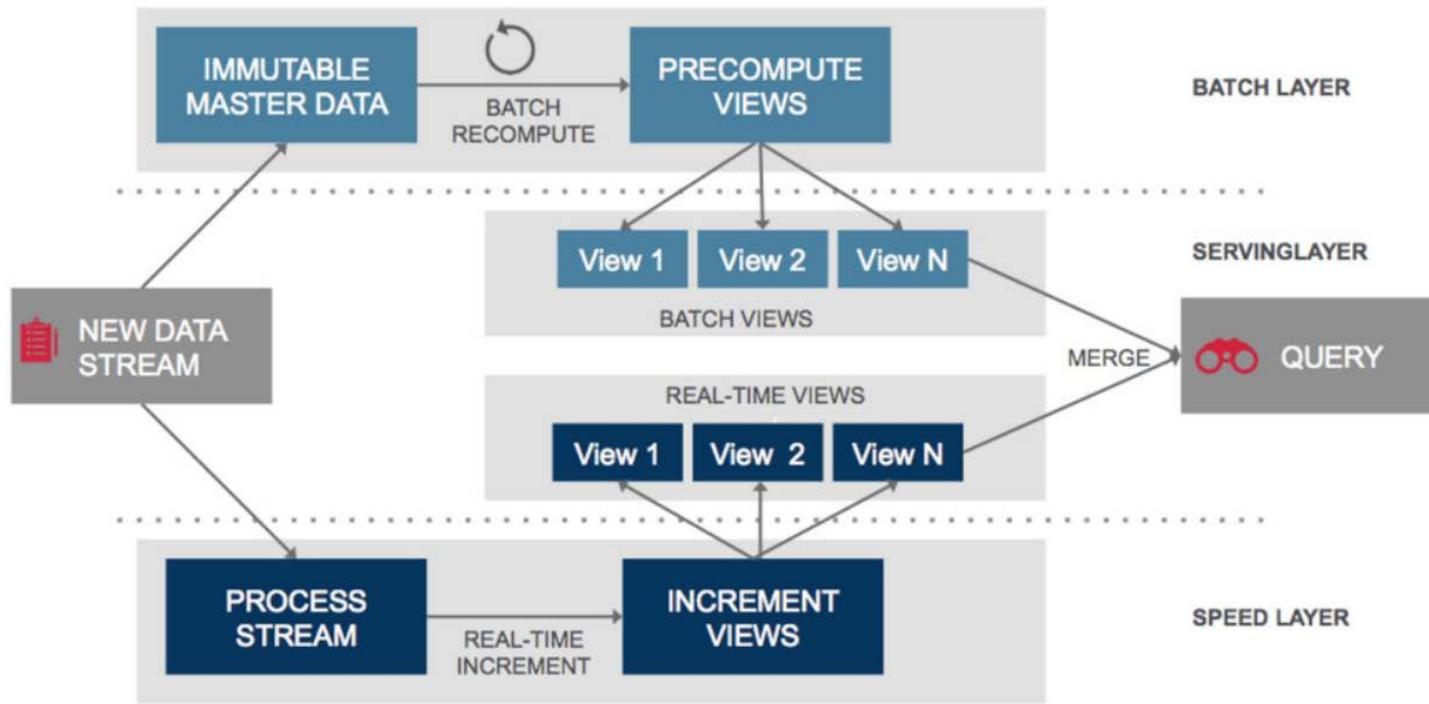
- Los datos introducidos en el Sistema son enviados tanto a la capa de batch como la de velocidad para su procesamiento.
- La capa batch tiene dos funciones: (i) gestionar el juego de datos maestros (un juego de datos inmutable y que no se borra) y (ii) realizar el pre-cálculo de las vistas batch.
- La capa de servicios indexa las vistas batch para que las consultas puedan ser libres y de baja latencia.
- La capa de velocidad solo trata datos recientes
- Cualquier consulta que se haga puede ser respondida juntando resultados tanto de la vista batch como de la vista real-time



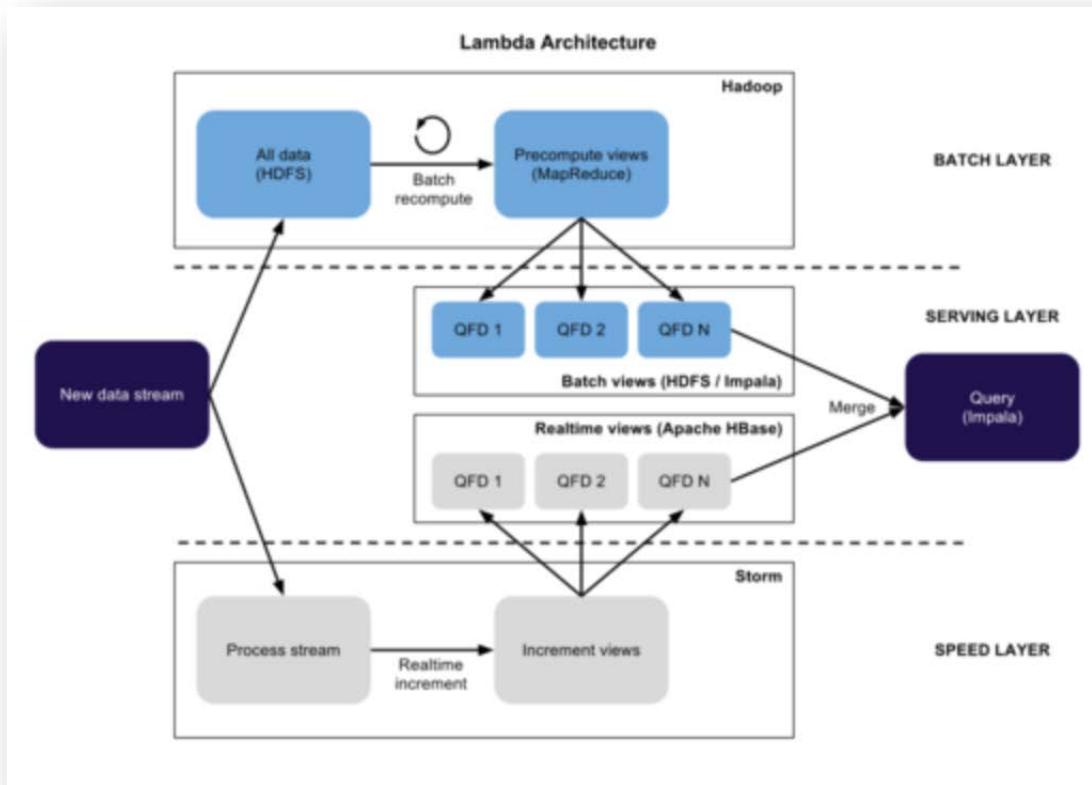
Características

- ¿A qué requerimientos no funcionales presta más atención?
 - Necesidad de un Sistema robusto tolerante a fallos tanto del hardware como errores humanos.
 - Servir a un amplio número de casos de carga de trabajo y casos de uso en los que se requieran lecturas y actualizaciones de baja latencia. El Sistema deberá permitir queries libres.
 - El Sistema deberá ser escalable linealmente de forma horizontal: si se incluyen más sistemas la carga de trabajo se reparte entre ellos
 - El Sistema será ampliable de tal manera que permitirá que se incorporen nuevas funcionalidades fácilmente. Deberá ser también fácilmente debuggable y requerir mantenimiento mínimo.

Otra representación muy habitual



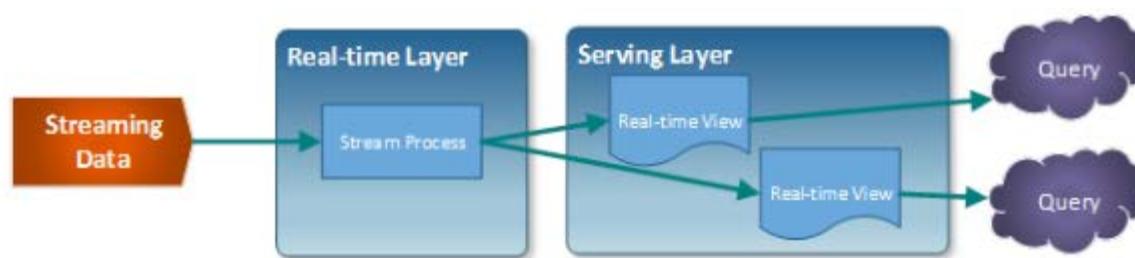
Y esta ya basada en hadoop



<http://jameskinley.tumblr.com/post/37398560534/the-lambda-architecture-principles-for>

Arquitectura Kappa

- Surge en el año 2014, dos años después de la Arquitectura Lambda
- Si bien la arquitectura Lambda reconoce dos flujos de datos: Batch y Tiempo Real, la arquitectura Kappa presenta un solo flujo de datos siempre considerando tiempo real o streaming, es decir el dato se procesa en cuanto llega



Principios de diseño

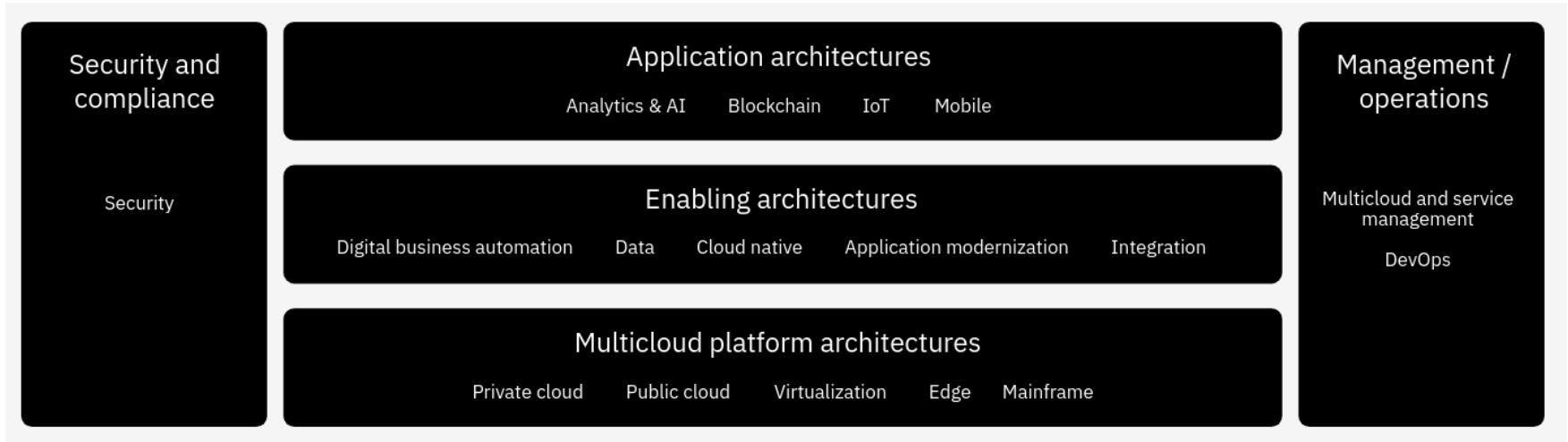
- El proceso batch no es más que un proceso streaming que se lanza en un momento en el tiempo y que tiene un final
- Se busca tener un único tipo de código funcionando en vez de contar con varios lenguajes, plataformas o entornos de desarrollo
- El dato en origen no se modifica con lo que reproducir una vista en un momento en el tiempo se puede recuperar en cualquier momento
- Al partir de los datos originales, sin modificar, se pueden adaptar las vistas a nuevos requerimientos sin ningún problema

¿Cuándo utilizar cual?

- Si tenemos el mismo tipo de proceso en batch que en streaming deberemos pensar en Kappa,
- Si el proceso batch es muy pequeño (o poco relevante) debemos seguir un diseño Kappa
- Si existe una necesidad de acceder a una gran gama de datos sin impactar unos procesos en otros será conveniente seguir una arquitectura Lambda para poder diferenciar las vistas
- Si la lógica a aplicar en alguna de las líneas es muy distinta a la otra (por ejemplo modelos predictivos en la parte de streaming o tiempo real) entonces deberemos seguir una línea Lambda

IBM. Arquitecturas cloud de referencia

<https://www.ibm.com/cloud/garage/architectures>



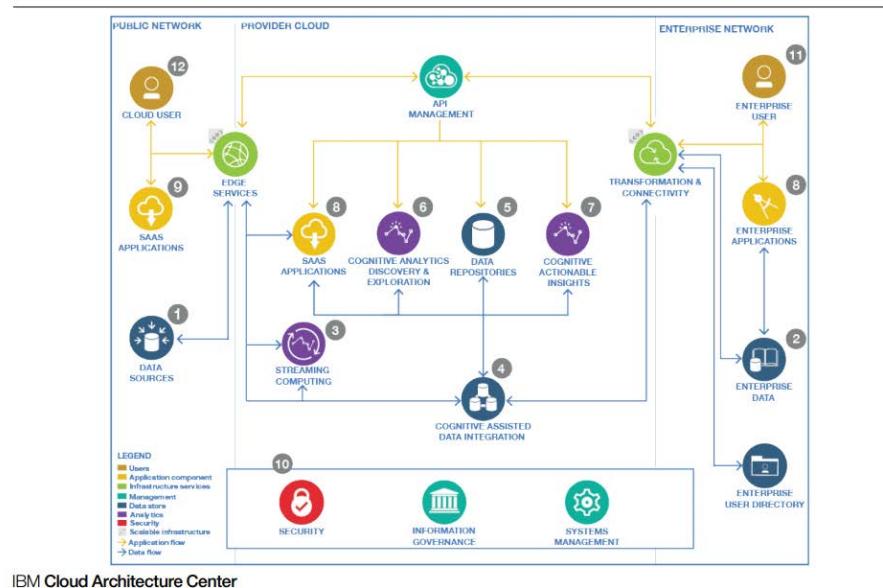
IBM Data Architecture

<https://www.ibm.com/cloud/architecture/architectures/dataArchitecture>



IBM Cloud Architecture Center

Data and analytics reference architecture

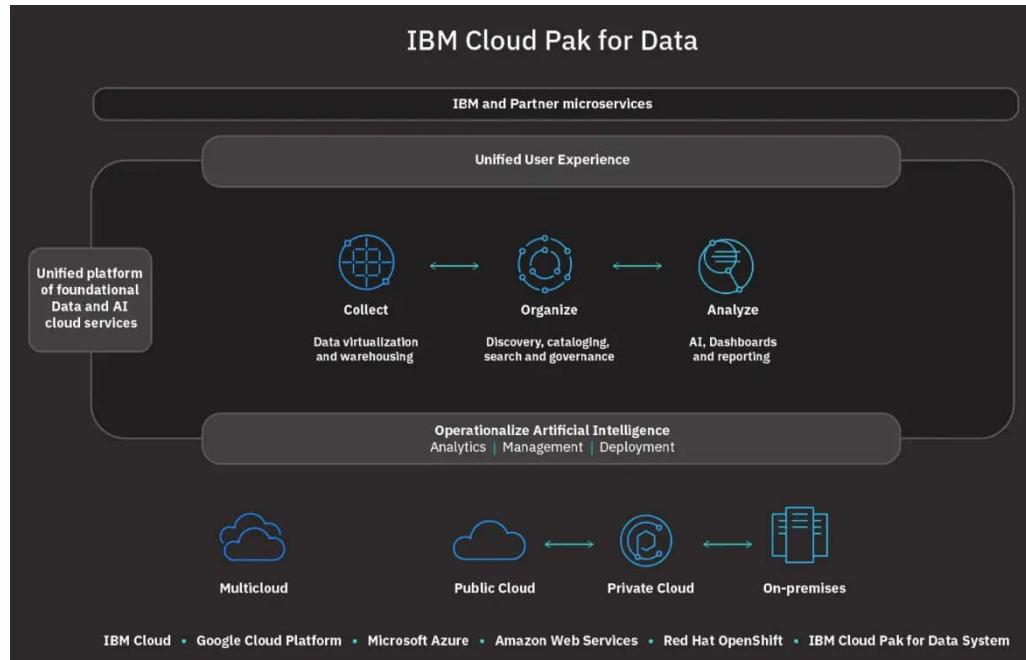


1

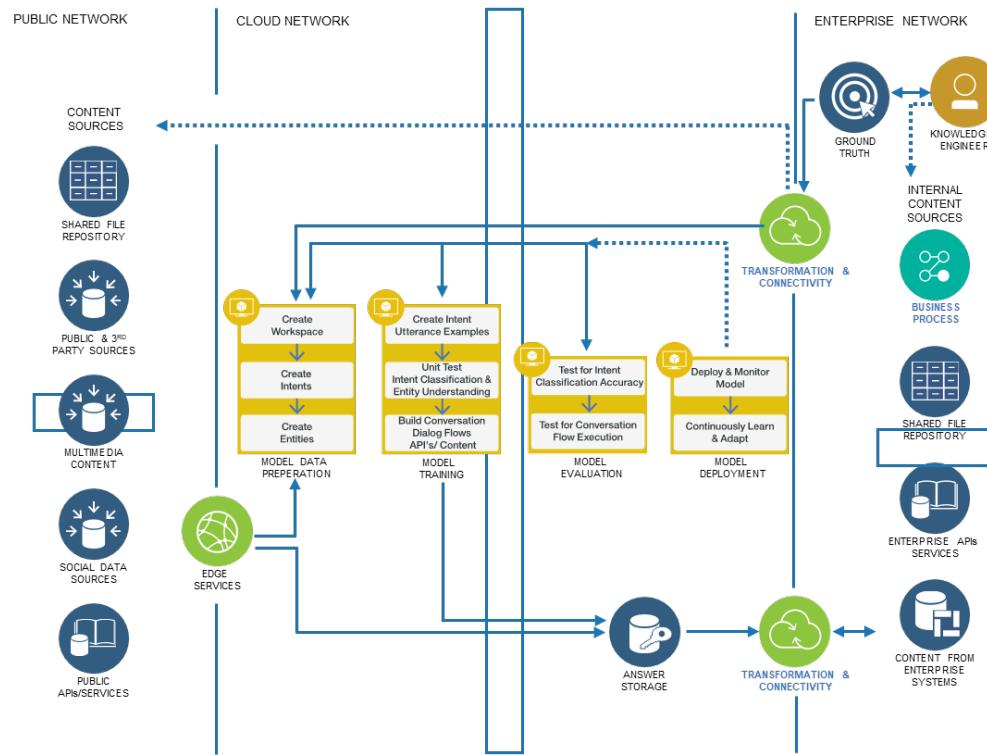
<https://www.ibm.com/cloud/garage/architectures/dataAnalyticsArchitecture/product-guidance-table>

IBM Analytics & AI Architecture

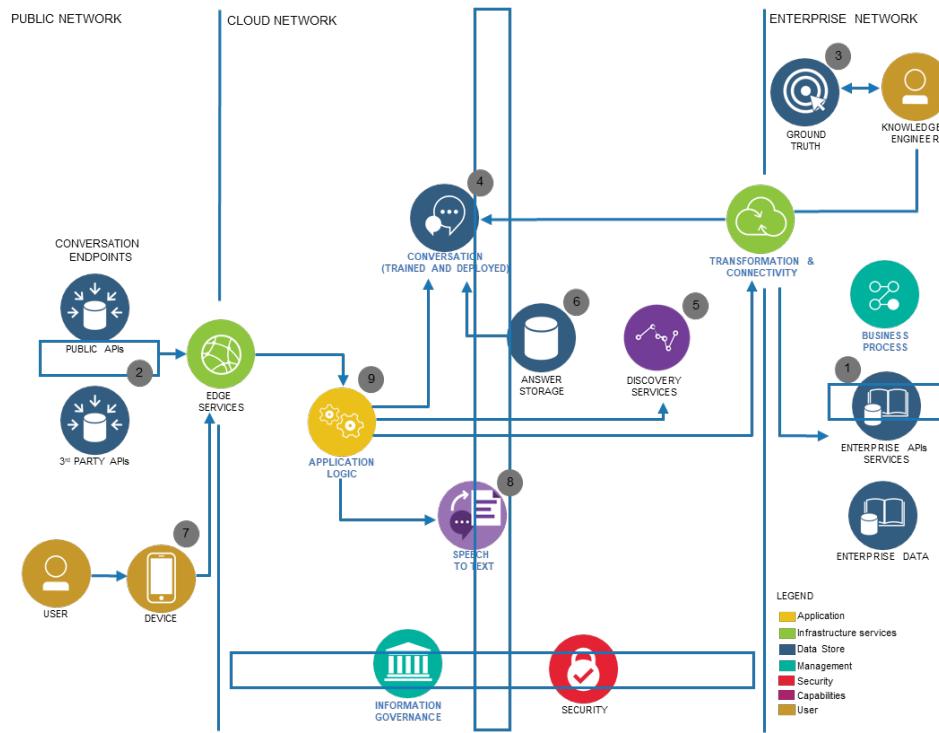
<https://www.ibm.com/cloud/architecture/architectures/aiAnalyticsArchitecture>



Cognitive Architecture. Soluciones de conversación



Cognitive Architecture. Coreografía de servicios



Centro de Arquitectura de AWS

<https://aws.amazon.com/es/architecture/>

The screenshot shows the AWS Architecture Center homepage. At the top, there's a navigation bar with links for Products, Soluciones, Precios, Documentación, Aprender, Red de socios, AWS Marketplace, Habilitación para clientes, Eventos, Explorar más, Contacte con nosotros, Support, Español, Mi cuenta, and a 'Cree una cuenta de AWS' button. Below the navigation is a secondary menu with links for Centro de arquitectura de AWS, Categorías tecnológicas, Sectores, Serie de videos, AWS Well-Architected, Bibliotecas, and Más recursos.

Centro de arquitectura de AWS

[Iniciar sesión y comenzar a crear](#)

El centro de arquitectura de AWS proporciona diagramas de arquitectura de referencia, soluciones de arquitectura autorizadas, prácticas recomendadas de Well-Architected, patrones, íconos, etc. Los expertos de arquitectura de la nube de AWS contribuyeron a esta guía de expertos, incluidos arquitectos de soluciones de AWS, consultores de servicios profesionales y socios.

AWS Well-Architected

- Marco de Buena Arquitectura de AWS
- Herramienta de buena arquitectura de AWS
- Enfoques de buena arquitectura de AWS
- Laboratorios de buena arquitectura de AWS

Arquitecturas de referencia

- Análisis y big data
- Computación y HPC
- Bases de datos
- Machine learning

[Explorar bibliotecas](#)

Diagramas de arquitectura de referencia de AWS | Documentos técnicos y guías de AWS | Biblioteca de soluciones de AWS | Íconos de arquitectura de AWS | Esta es mi arquitectura | Volver a lo básico | Centro para desarrolladores de AWS

AWS. Arquitecturas de Referencia

El centro de arquitectura de AWS proporciona diagramas de arquitectura de referencia, soluciones de arquitectura autorizadas, prácticas recomendadas de Well-Architected, patrones, iconos, etc. Los expertos de arquitectura de la nube de AWS contribuyeron a esta guía de expertos, incluidos arquitectos de soluciones de AWS, consultores de servicios profesionales y socios.

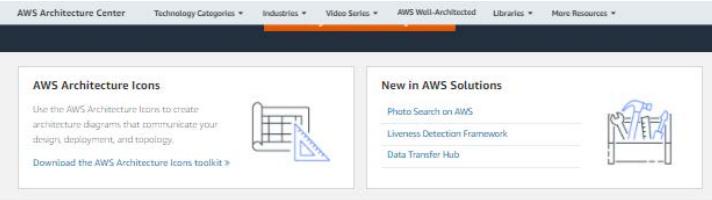
AWS Well-Architected

- Marco de Buena Arquitectura de AWS
- Herramienta de buena arquitectura de AWS
- Enfoques de buena arquitectura de AWS
- Laboratorios de buena arquitectura de AWS

Arquitecturas de referencia

- Análisis y big data
- Computación y HPC
- Bases de datos
- Machine learning





AWS Architecture Icons

Use the AWS Architecture Icons to create architecture diagrams that communicate your design, deployment, and topology.

[Download the AWS Architecture Icons toolkit](#)

Did this page help you?

Yes No [Feedback](#)

Filter by:

[Clear filters](#)

Technology Categories

- Analytics & Big Data
- Application Integration
- Cloud Financial Management
- Compute
- Containers
- Databases
- Developer Tools
- End-User Computing
- Front-End Web & Mobile
- Internet of Things (IoT)
- Machine Learning & AI
- Management & Governance
- Media Services
- Migration
- Networking & Content Delivery
- Security, Identity, & Compliance
- Spatial Computing
- Storage

Industries

- Aerospace
- Agriculture
- Automotive
- Consumer Packaged Goods
- Defense
- Education
- Electronics
- Financial Services
- Game Tech
- Healthcare
- Hospitality
- Life Sciences
- Manufacturing
- Manufacturing (Semiconductors)
- Media & Entertainment
- Power & Utilities
- Retail & Wholesale
- Telecommunications
- Transportation & Logistics
- Travel

New in AWS Solutions

[Photo Search on AWS](#)

[Liveness Detection Framework](#)

[Data Transfer Hub](#)

Search AWS Reference Architecture Diagrams:

1-9 (253)

REFERENCE ARCHITECTURE DIAGRAM UPDATED

MLOps Workload Orchestrator

This reference architecture is an extendable framework that provides a standard interface for managing ML pipelines for AWS ML services and third-party services.

[PDF](#)

January 2022

REFERENCE ARCHITECTURE DIAGRAM NEW

Centralized Alarms and Notifications

Deployment of a serverless monitoring and alarm system configured to send workload-specific notifications.

[PDF](#)

January 2022

REFERENCE ARCHITECTURE DIAGRAM NEW

Financial Advisor Chat Assistant

Financial advisors often work outside of normal business hours, when head office teams might not be available to answer questions. Use Amazon Web Services (AWS) to create a sophisticated conversational chatbot to help financial advisors get answers to their queries.

[PDF](#)

January 2022

REFERENCE ARCHITECTURE DIAGRAM NEW

Electric Vehicle Charging Station Management...

Build an OCPP-based charging station management software for electric vehicles.

[PDF](#)

Automotive | Power & Utilities

January 2022

REFERENCE ARCHITECTURE DIAGRAM NEW

Moodle for High Availability on AWS

Moodle is an open source learning management system (LMS) that supports distributed online learning. When implemented on AWS, Moodle can scale flexibly to optimize cost and maximize availability.

[PDF](#)

Analytics | Databases

December 2021

REFERENCE ARCHITECTURE DIAGRAM NEW

Couchbase on AWS Local Zones for Low Latency...

An architecture overview of Couchbase Server and Couchbase Sync Gateway deployment on AWS Local Zones for low latency edge use case.

[PDF](#)

Networking & Content Delivery

December 2021

REFERENCE ARCHITECTURE DIAGRAM UPDATED

IPv6 Reference Architectures for AWS...

Reference architecture series for dual stack IPv6 AWS and hybrid networks.

[PDF](#)

Networking & Content Delivery

60

AWS. Well-Architected

El centro de arquitectura de AWS proporciona diagramas de arquitectura de referencia, soluciones de arquitectura autorizadas, prácticas recomendadas de Well-Architected, patrones, iconos, etc. Los expertos de arquitectura de la nube de AWS contribuyeron a esta guía de expertos, incluidos arquitectos de soluciones de AWS, consultores de servicios profesionales y socios.

The screenshot shows the AWS Well-Architected landing page. It features three main sections: 'AWS Well-Architected' with a hexagonal icon, 'Arquitecturas de referencia' with a briefcase icon, and a video player titled 'THIS IS MY ARCHITECTURE' from the 'EMIRATES MARS MISSION'. The 'AWS Well-Architected' section lists five items: Marco de Buena Arquitectura de AWS, Herramienta de buena arquitectura de AWS, Enfoques de buena arquitectura de AWS, and Laboratorios de buena arquitectura de AWS.

AWS Well-Architected ayuda a los arquitectos de la nube a crear una infraestructura segura, de alto rendimiento, resistente y eficiente para una variedad de aplicaciones y cargas de trabajo. Este marco, creado en torno a seis pilares (excelencia operativa, seguridad, fiabilidad, eficiencia de rendimiento, optimización de costos y sostenibilidad), ofrece un enfoque coherente para que los clientes y los socios evalúen las arquitecturas e implementen diseños escalables.

AWS Well-Architected Framework incluye enfoques de dominios específicos, laboratorios prácticos y AWS Well-Architected Tool. AWS Well-Architected Tool, disponible sin costo alguno en la consola de administración de AWS, proporciona un mecanismo para evaluar regularmente las cargas de trabajo, identificar los problemas de alto riesgo y registrar las mejoras.

Además, AWS brinda acceso a un ecosistema formado por cientos de miembros del programa para socios de AWS Well-Architected. Póngase en contacto con un socio en su zona para que lo ayude a analizar y revisar las aplicaciones.



AWS Well-Architected y los seis pilares

Vista general del marco

AWS Well-Architected Framework describe los conceptos clave, los principios de diseño y las prácticas recomendadas de arquitectura para diseñar y ejecutar cargas de trabajo en la nube. Responda un conjunto de preguntas básicas para descubrir hasta qué punto su arquitectura está en consonancia con las prácticas recomendadas en la nube y obtenga orientación para mejorarla.

[HTML](#) | [Kindle](#) | [Laboratorios](#)



Pilar de excelencia operativa

El pilar de la excelencia operativa se concentra en ejecutar y monitorear los sistemas y en mejorar constantemente los procesos y los procedimientos. Entre los temas clave se incluyen la automatización de cambios, la respuesta a eventos y la definición de estándares para administrar las operaciones diarias.

[HTML](#) | [Kindle](#) | [Laboratorios](#)

Pilar de seguridad

El pilar de la seguridad se concentra en proteger la información y los sistemas. Entre los temas clave se incluyen la confidencialidad y la integridad de los datos, la administración de los permisos de usuarios y el establecimiento de controles para detectar eventos de seguridad.

[HTML](#) | [Kindle](#) | [Laboratorios](#)

Pilar de fiabilidad

El pilar de fiabilidad se centra en las cargas de trabajo que realizan las funciones previstas y en cómo recuperarse rápidamente de los errores para cumplir con las demandas. Entre los temas clave se incluyen el diseño de sistemas distribuidos, la planificación de la recuperación y cómo adaptarse a los requisitos cambiantes.

[HTML](#) | [Kindle](#) | [Laboratorios](#)

Pilar de eficacia del rendimiento

El pilar de eficacia del rendimiento se centra en la asignación estructurada y simplificada de TI y en los recursos informáticos. Entre los temas clave se incluyen la comprensión del tiempo dedicado y el control de la asignación de fondos, la selección de recursos para el tipo y la cantidad adecuados y el escalado para cumplir con las necesidades de la empresa sin gastos excesivos.

[HTML](#) | [Kindle](#) | [Laboratorios](#)

Pilar de optimización de costos

El pilar de optimización de costos se centra en evitar gastos innecesarios. Entre los temas clave se incluyen la comprensión del tiempo dedicado y el control de la asignación de fondos, la selección de recursos para el tipo y la cantidad adecuados y el escalado para cumplir con las necesidades de la empresa sin gastos excesivos.

[HTML](#) | [Kindle](#)

Pilar de sostenibilidad

El pilar de sostenibilidad se centra en minimizar los impactos ambientales de ejecutar cargas de trabajo en la nube. Entre los temas clave se incluyen un modelo de responsabilidad compartida para la sostenibilidad, la comprensión del impacto y la maximización del uso para minimizar los recursos necesarios y reducir los impactos posteriores.

[HTML](#) | [Kindle](#)

AWS. Documentos técnicos y guías de AWS

Centro de arquitectura de AWS Categorías tecnológicas ▾ Sectores ▾ Serie de videos ▾ AWS Well-Architected Bibliotecas ▾ Más recursos ▾

Borrar filtros

▼ Tipos de contenidos

- Documento técnico
- Guía técnica
- Guía de conformidad
- Diagrama de la arquitectura de referencia

¿Le ha servido de ayuda esta página?

Sí No Comentarios

Buscar documentos técnicos y guías de AWS

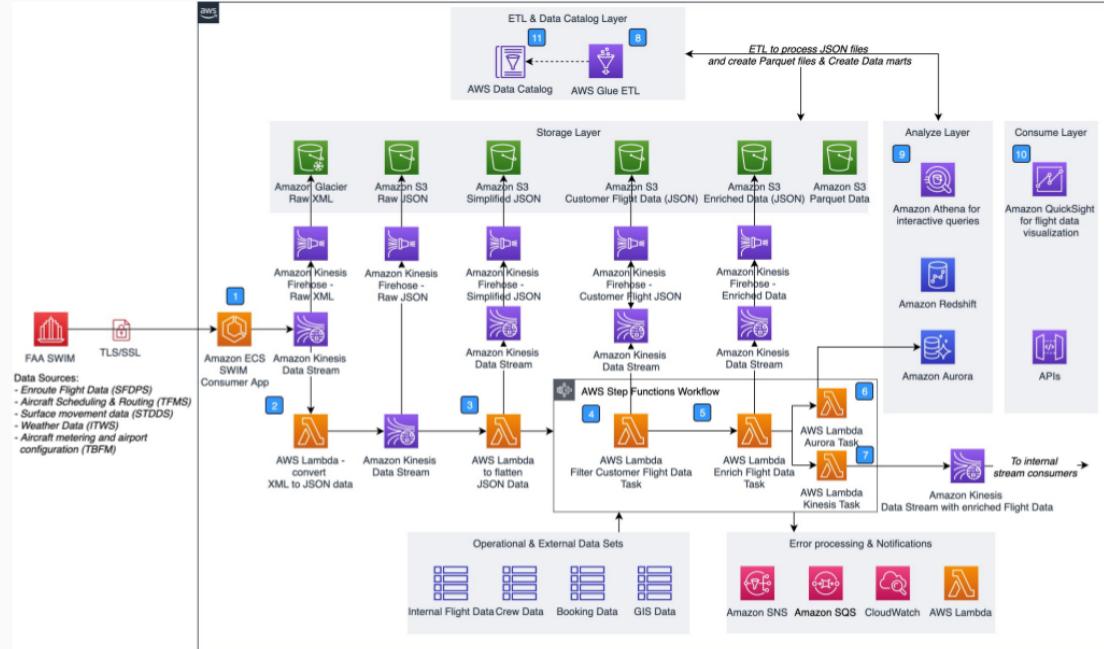
1-15 (561) Ordenar por: Fecha (de más reciente a menos reciente) ▾

DOCUMENTO TÉCNICO NOVEDADES Sistemas reactivos en AWS Este documento técnico describe las prácticas recomendadas para diseñar un sistema basado en principios reactivos mediante la plataforma en la nube de AWS y ofrece una arquitectura de referencia para orientar a las organizaciones en la entrega de estos sistemas. Noviembre de 2021	DIAGRAMA DE LA ARQUITECTURA DE... NOVEDADES Análisis de contenido de AWS Esta arquitectura de referencia implementa una solución que utiliza los servicios de IA de AWS para analizar los archivos multimedia y generar información significativa a través de metadatos generados por el machine learning. Octubre de 2021	DOCUMENTO TÉCNICO DESTACADO Enfoque en el análisis de datos: AWS... En este documento se describe el enfoque en análisis de datos de AWS Well-Architected, una colección de prácticas recomendadas probadas por los clientes para el diseño de cargas de trabajo de análisis bien diseñadas. HTML Kindle Octubre de 2021
DOCUMENTO TÉCNICO ACTUALIZADO	DOCUMENTO TÉCNICO ACTUALIZADO	DOCUMENTO TÉCNICO DESTACADO

FAA SWIM Data Lake

Federal Aviation Administration SWIM Data Lake

The Federal Aviation Administration (FAA) System Wide Information Management (SWIM) Data Lake enables real-time delivery of relevant aeronautical flight and weather information to help airline companies optimize operations.



© 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.

AWS Reference Architecture

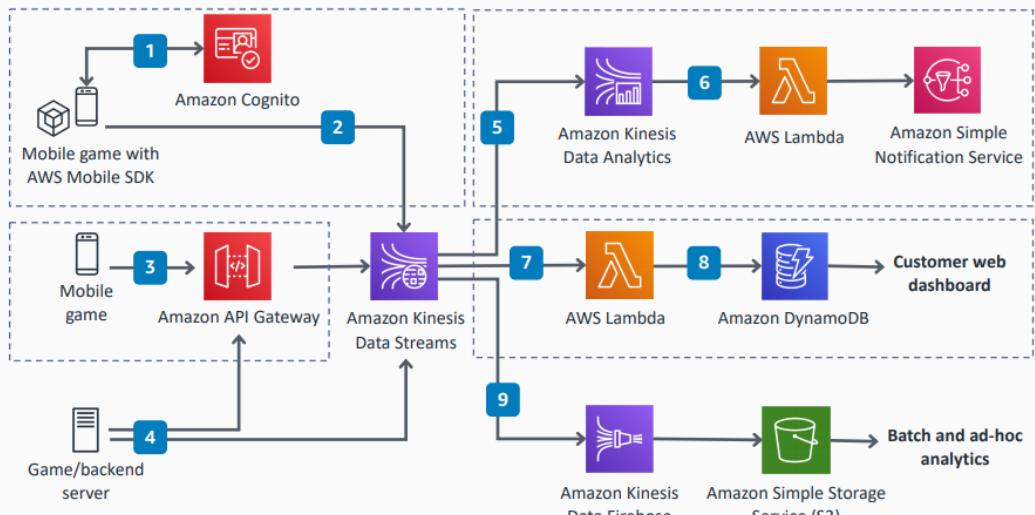
- 1 The **SWIM Consumer App** consumes SWIM data and sends it to the **Amazon Kinesis Data Stream**.
- 2 **AWS Lambda** converts XML data to JSON data.
- 3 **AWS Lambda** simplifies the JSON data.
- 4 The **AWS Lambda Task** filters customer flight data. It writes one copy of the data to **Amazon Kinesis Data Stream** and a second copy to the subsequent **AWS Lambda Task**.
- 5 The **AWS Lambda Task** enriches customer flight messages with one or more operational or external datasets.
- 6 The **AWS Lambda Task** writes enriched flight data to **Amazon Aurora**.
- 7 The **AWS Lambda Task** writes enriched flight data to the **Amazon Kinesis Data Stream**.
- 8 **AWS Glue ETL** jobs transform enriched flight data from JSON files to Parquet files with appropriate partitions. The jobs register Parquet files as external tables in **Glue Data Catalog**.
- 9 Transformed and enriched flight data is analyzed by services such as **Amazon Athena**, **Amazon Redshift**, and **Amazon Aurora**.
- 10 Analyzed data is consumed by various services such as **Amazon QuickSight**, **custom applications**, and **APIs**.
- 11 The **Glue Data Catalog** with databases and tables representing raw, transformed, enriched, and analyzed datasets is stored in **Amazon S3**, **Amazon Aurora**, and **Amazon Redshift**.

Serverless Real-Time Analytics for Mobile Gaming

Serverless Real-Time Analytics for Mobile Gaming

Build Serverless Real-Time Analytics Pipelines for Mobile Gaming

This serverless architecture example collects events from mobile games, analyzes them in real-time, and stores them for processing. It can be leveraged to provide real-time flash offers, monitor user acquisition campaigns, detect abusive players, find deficiencies and anomalies during A/B testing of game balance changes, and build online dashboards for metrics.



© 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.

AWS Reference Architecture

Collecting events (option with AWS Mobile SDK):

- 1 A mobile game receives temporary AWS credentials from the Amazon Cognito Identity Pool to access Amazon Kinesis Data Stream.
- 2 The game leverages AWS Mobile SDK to submit events in JSON directly to the Kinesis Data Stream.

Collecting events (option without AWS Mobile SDK):

- 3 A mobile game submits events via the REST API to the Amazon API Gateway, which integrates into the Kinesis stream through AWS Service integration.

Submitting events:

- 4 Game/backend servers submit events directly to Kinesis using AWS SDK, or through the API gateway.

Analyzing events (option with aggregation):

- 5 Amazon Kinesis Data Analytics consumes the Kinesis stream.
- 6 AWS Lambda delivers output data from Kinesis Data Analytics to Amazon DynamoDB, Amazon SNS, or Amazon SQS to perform actions.

Analyzing events (option without aggregation):

- 7 An AWS Lambda function consumes events from the Kinesis Data Stream.
- 8 Metrics (like CCUs and crashes) are stored in Amazon DynamoDB and displayed on a customer web dashboard.

Storing events for analysis:

- 9 Amazon Kinesis Data Firehose consumes the data stream, converts the data, and stores the data in Amazon S3.

Otras arquitecturas de referencia. Lenovo.

ARCHITECTURE BRIEF

Big Data

Lenovo Big Data Reference Architectures

Easy-to-implement hardware, software and services for analyzing data at rest and data in motion



Big data is here—in a big way. Mind-boggling volumes of data are created every second of every day. What's more, the number and types of sources generating data is growing exponentially from both people and electronic devices. More data should lead to better decisions. But because data growth is so fast, many enterprises are analyzing a smaller percentage of data than before. So what is the answer? You don't have the time, resources or capital to experiment with what works in big data analytics and what doesn't. The challenge you face is to manage the volume, variety and velocity of data, apply analytics for better insight and use this insight for making better business decisions faster than the competition.

Lenovo reference architecture solutions for big data analytics are easy to order and easy to implement. They include hardware, software and services along with a standardized blueprint. You can quickly deploy these cost-effective solutions to analyze stored data at rest such as databases. Additionally, these solutions may be used to analyze data in motion, like streaming data. The solutions are built around powerful, affordable, scalable Lenovo M5 servers and Lenovo networking options so you can deploy proven solutions quickly.

WWW.LENOVO.COM

Lenovo

Lenovo™

cloudera

Hortonworks®
POWERING THE FUTURE OF DATA™

IBM

MAPR®

hadoop

APACHE
Spark™

Why Lenovo

Lenovo is a leading provider of x86 servers for the data center. The server portfolio includes rack, tower, blade, dense and converged systems, and provides excellent performance, reliability and security. Lenovo also offers a full range of networking, storage, software, solutions, and comprehensive services supporting business needs throughout the IT lifecycle (such as planning, deployment, and support). Lenovo offers expertise and services needed to deliver better service-level agreements, and generate greater end-user satisfaction.

Download the [Lenovo Big Data Reference Architecture for Cloudera Distribution for Hadoop](#)

Download the [Lenovo Big Data Reference Architecture for Hortonworks Data Platform](#)

Download the [Lenovo Big Data Reference Architecture for IBM BigInsights brings the power of Apache Hadoop](#)

Download the [Lenovo Big Data Reference Architecture for MapR Converged Data Platform](#)

Download the [Lenovo Big Data Configuration for Apache Spark](#)

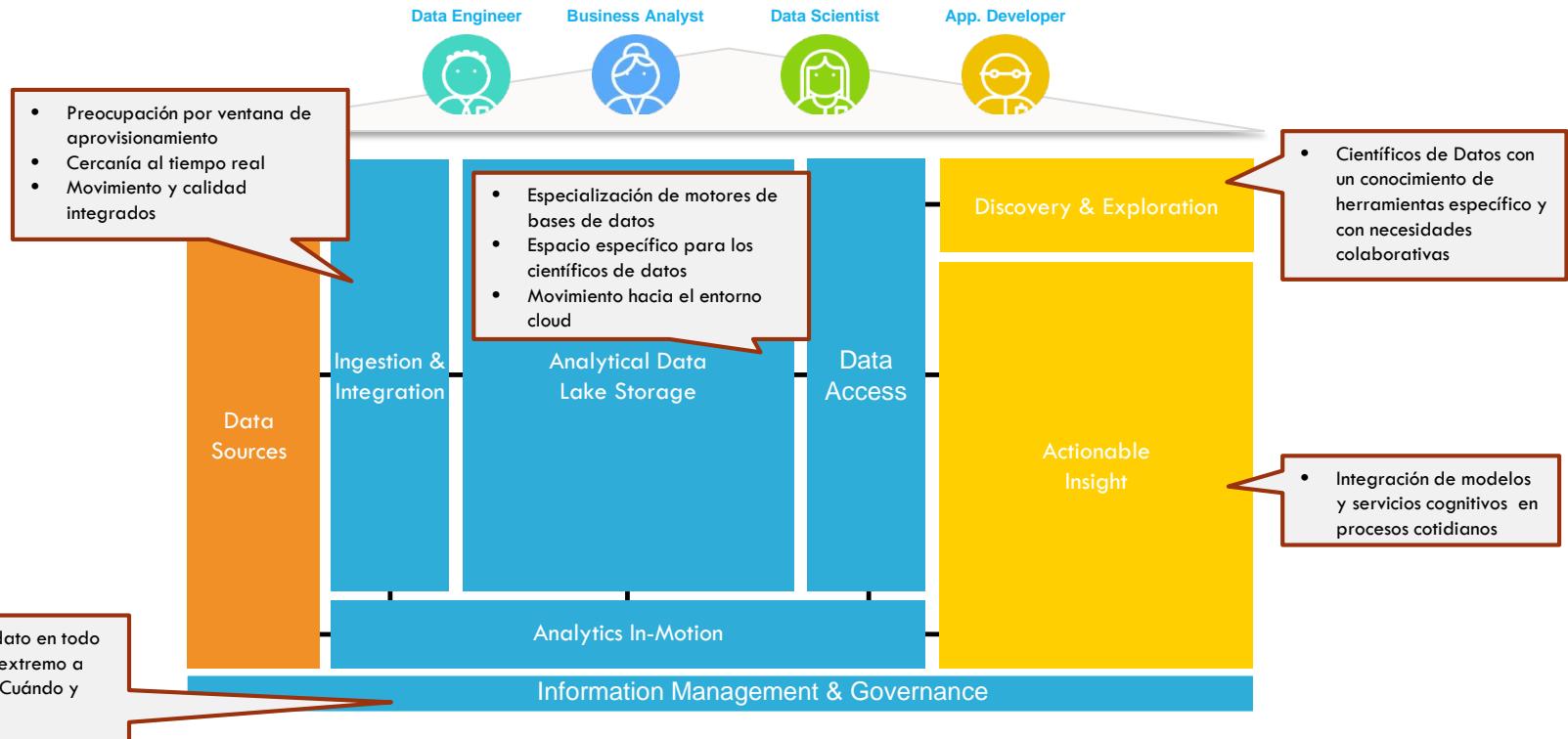
For More Information

To learn more about the Lenovo Big Data Reference Architectures, contact your Lenovo Business Partner or visit:

<http://shop.lenovo.com/us/en/systems/solutions/big-data/>

<https://www.lenovo.com/us/en/resources/data-center-solutions/reference-architecture/lenovo-big-data-reference-architectures/>

Tendencias en las arquitecturas de Big Data



APÉNDICE

Primer esquema:

- Diagrama o Visión General de Arquitectura
- Propósito:
 - Comunicar al patrocinador del Proyecto un entendimiento conceptual del Sistema IT que proponemos
 - Suministrar una vision de alto nivel de la arquitectura y alcance de la solución propuesta a los equipos de desarrollo o cualquier otro equipo
 - Explorar y evaluar alternativas arquitecturales
 - Reconocer y validar las implicaciones del enfoque arquitectural que proponemos
 - Facilitar una comunicación efectiva entre las distintas comunidades que estén involucradas
 - Facilitar la incorporación de nuevos miembros del equipo de proyecto

En otras palabras

Este artefacto proporciona una visión general de los "principales elementos conceptuales y relaciones" de una arquitectura, que pueden incluir subsistemas, componentes, nodos, conexiones, almacenes de datos, usuarios y sistemas externos. Como tal, representa las ideas directoras y los "*ladrillos* (*) candidatos" de la arquitectura.

(*) Building Blocks

¿En qué consiste?

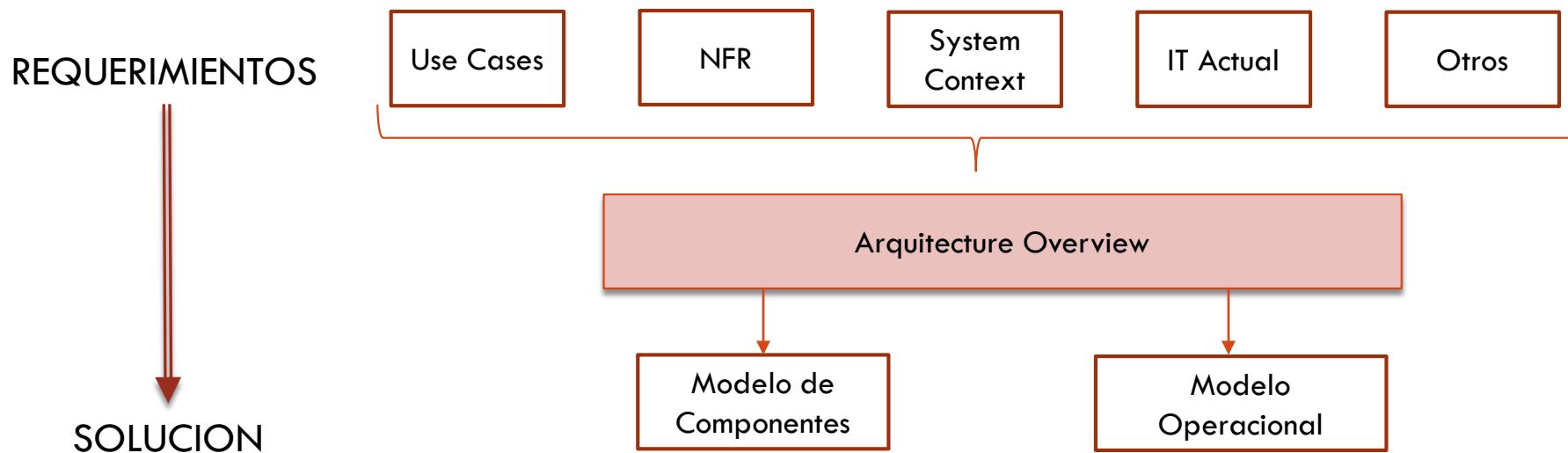
- Está formado por uno o más diagramas ‘informales’ (sin una notación formal que pudiera ser obligada por herramientas especiales de modelación) junto con un texto que soporta la descripción de los conceptos más importantes como por ejemplo:
 - Diferentes medios de visualización: terminals, kioskos, móviles, etc.
 - Diferenciación de funciones
 - Modelos de arquitectura, como en capas de tres niveles, cuatro...
 - Uso de hardware específico como servidores distribuidos, estaciones de trabajo, portátiles, etc.
 - Acceso a sistemas legacy

Más características

- Los diagramas son generalmente “estáticos” y muestran las relaciones entre los componentes. Sin embargo, si tiene sentido, debemos incluir algún efecto dinámico que muestra las relaciones. Los diagramas pueden mostrar vistas funcionales, operacionales o una combinación de ambas.
- Cuando haya varias alternativas a mostrar se podrán hacer distintos diagramas dependiendo del promotor de la solución.

¿Cuando lo construimos?

- En las fases iniciales



Aspectos a tener en cuenta...

#1 Debe dejar claro qué valor aporta a través de la solución que proponemos.

#2 Muy orientado al patrocinador. Por ejemplo mostraremos una visión corporativa si el usuario es de negocio.

#3 Pero no nos quedemos ahí, a nivel Empresa: estemos atentos a otras visiones.

#4 Si hay alguna arquitectura de referencia que soporte el diseño Podemos indicarlo.

#5 No incluyamos demasiada información. Tengamos cuidado.

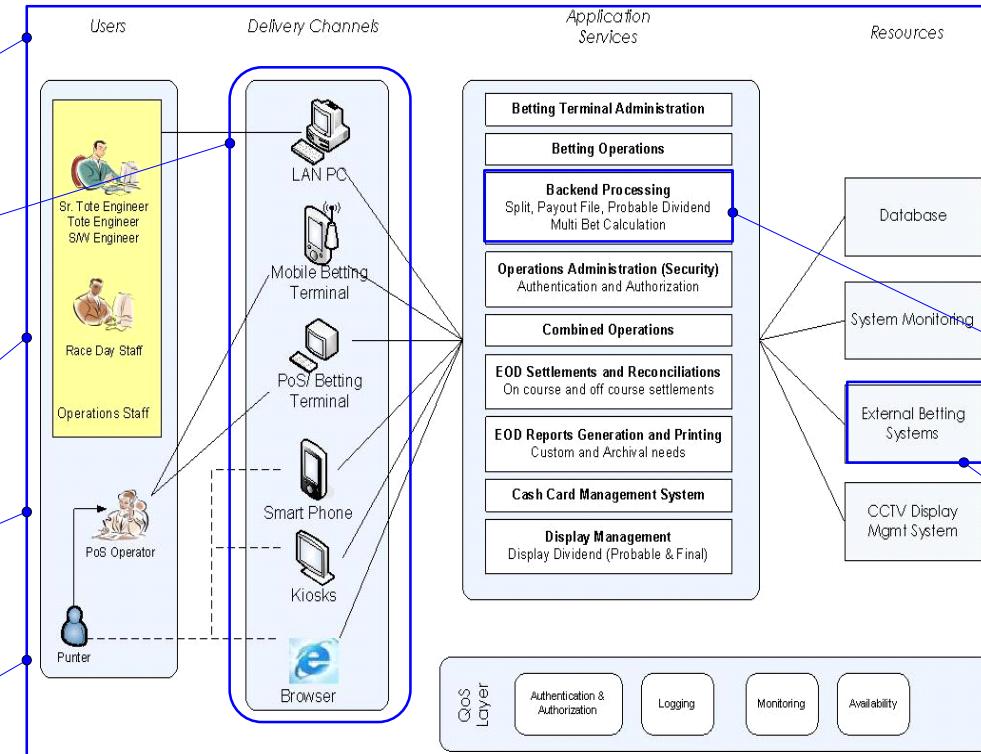
6 Dejemos claro cuáles son los Building blocks de nuestra solución

#7 Demos pie para defender los requerimientos no funcionales.

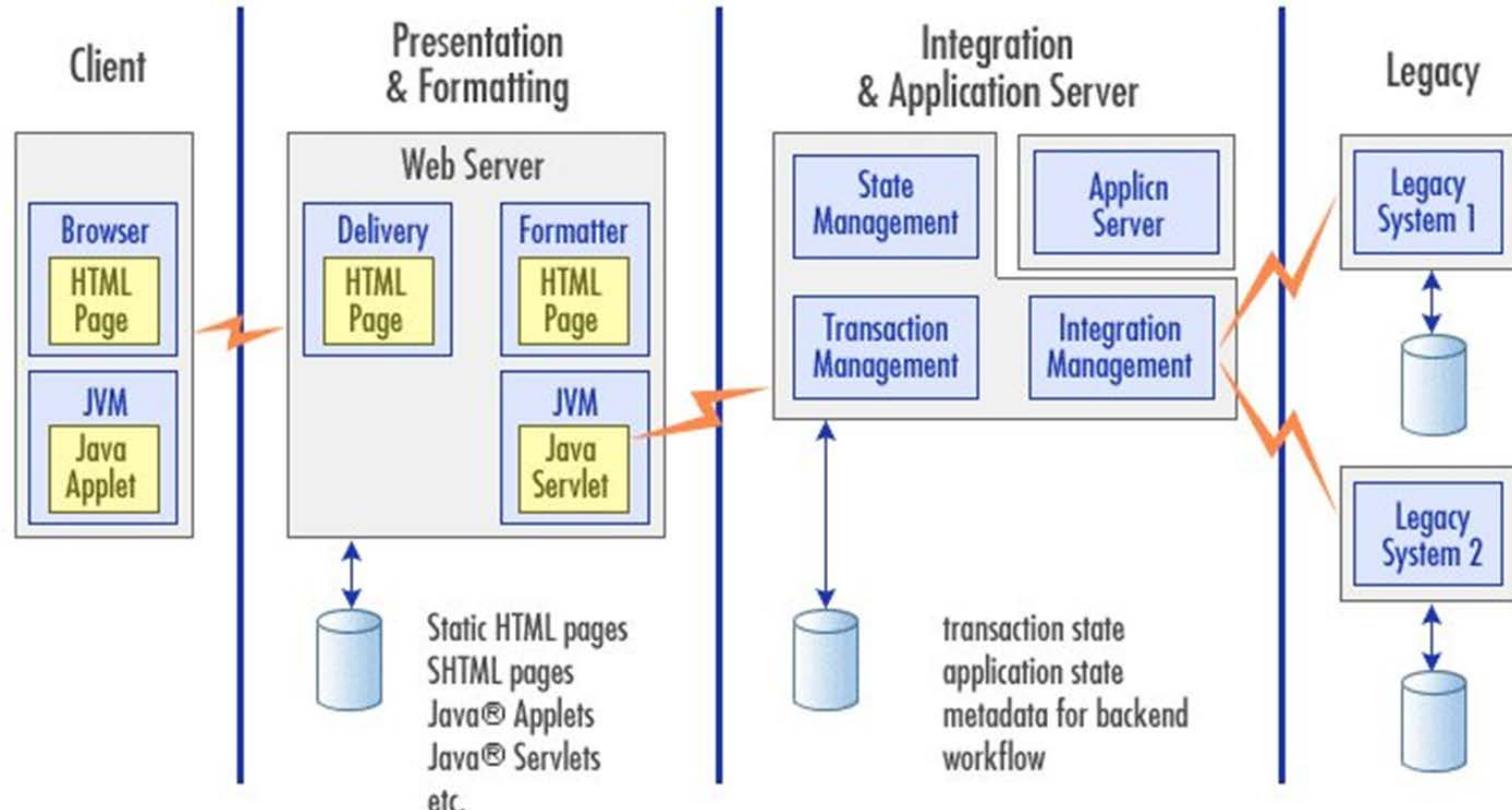
#8 Vayamos a un alto nivel en vez de bajar un nivel muy detallado

#9 Si conocemos nombres específicos utilizemoslos en vez de nombres genéricos.

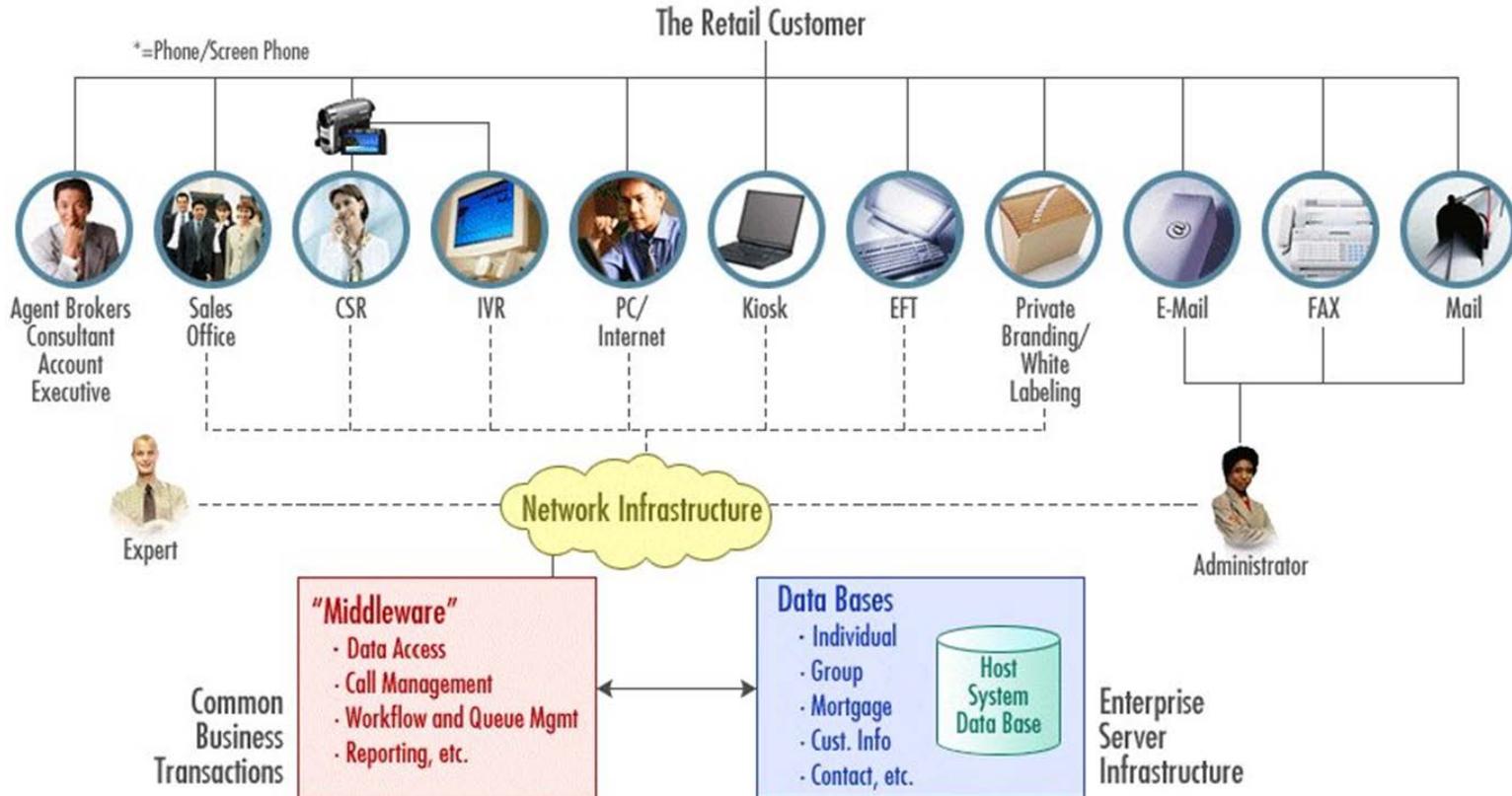
#10 Utilizemos anotaciones cuando sea necesario para aportar claridad



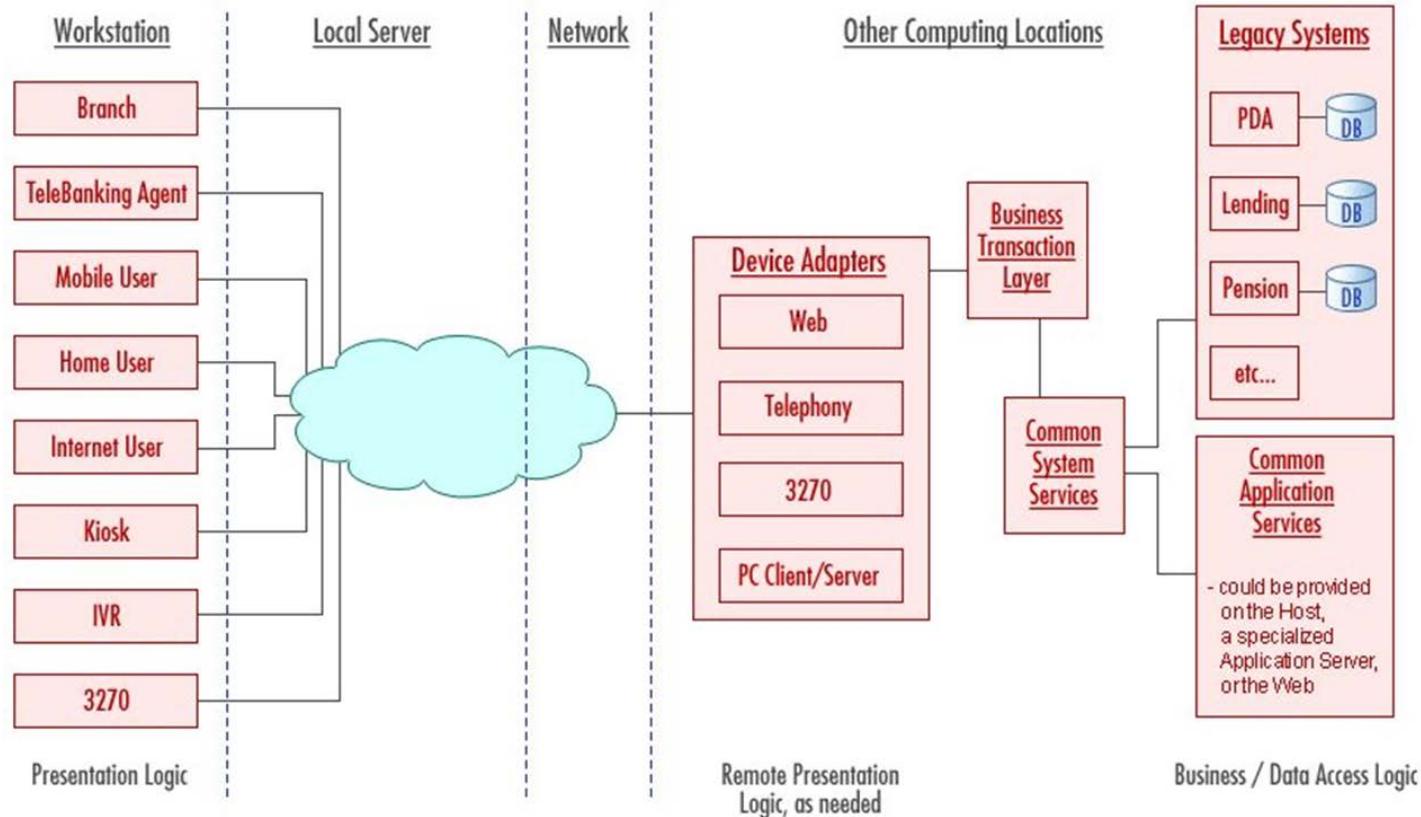
Ejemplo 1. A discutir. ¿Qué opinas?



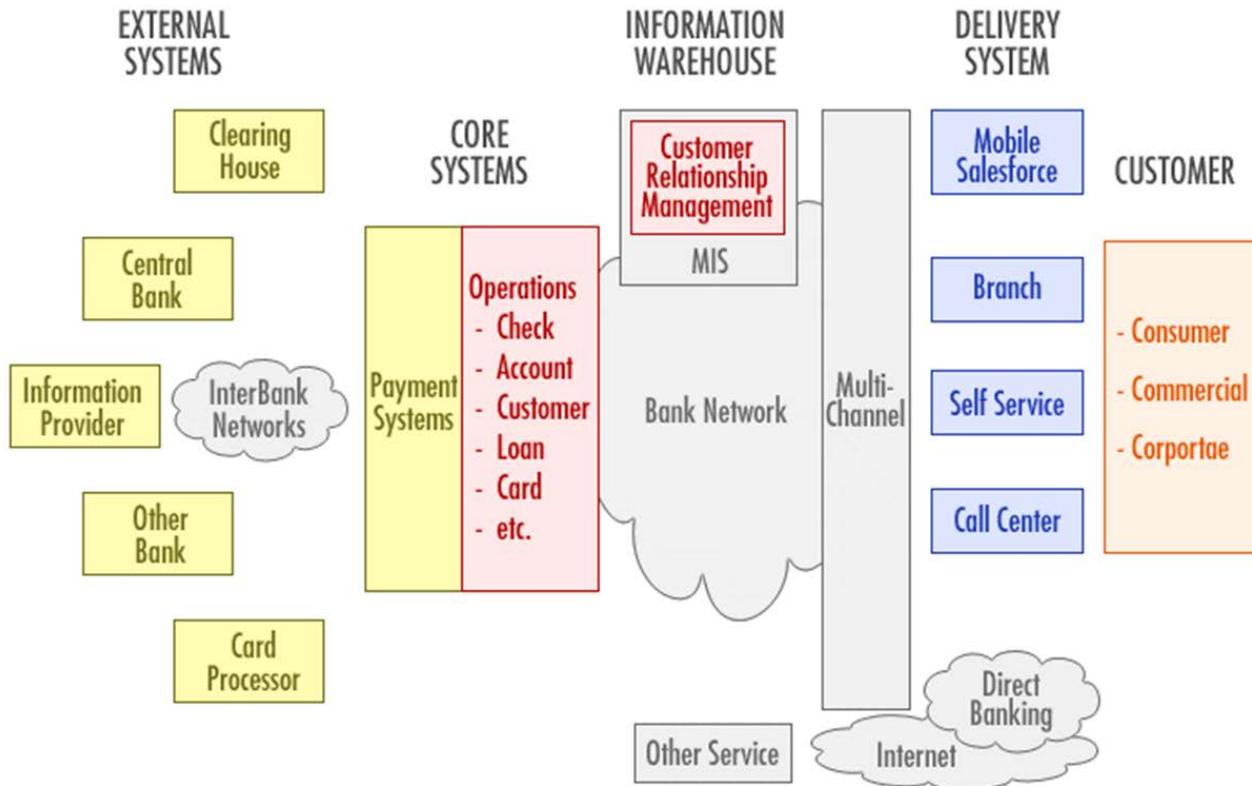
Ejemplo 2. A discutir. ¿Qué opinas?



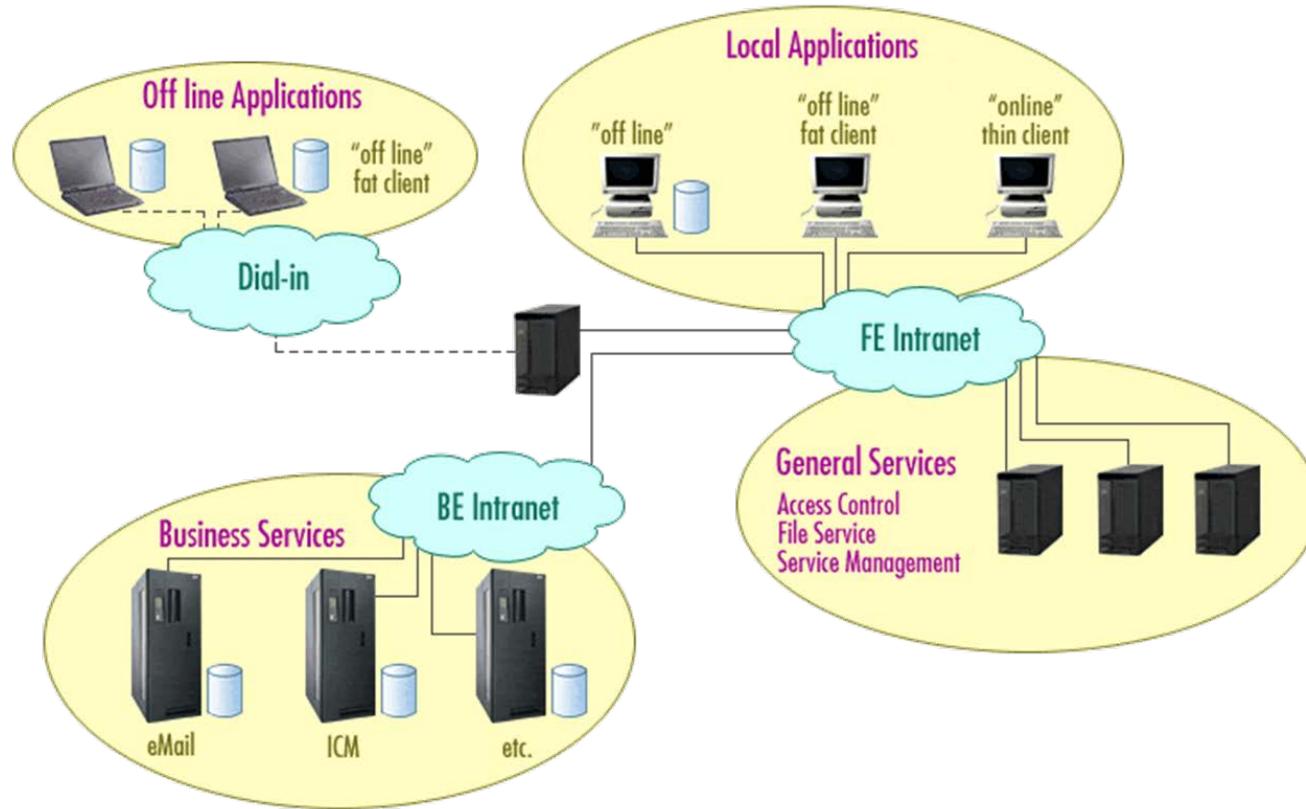
Ejemplo 3. A discutir. ¿Qué opinas?



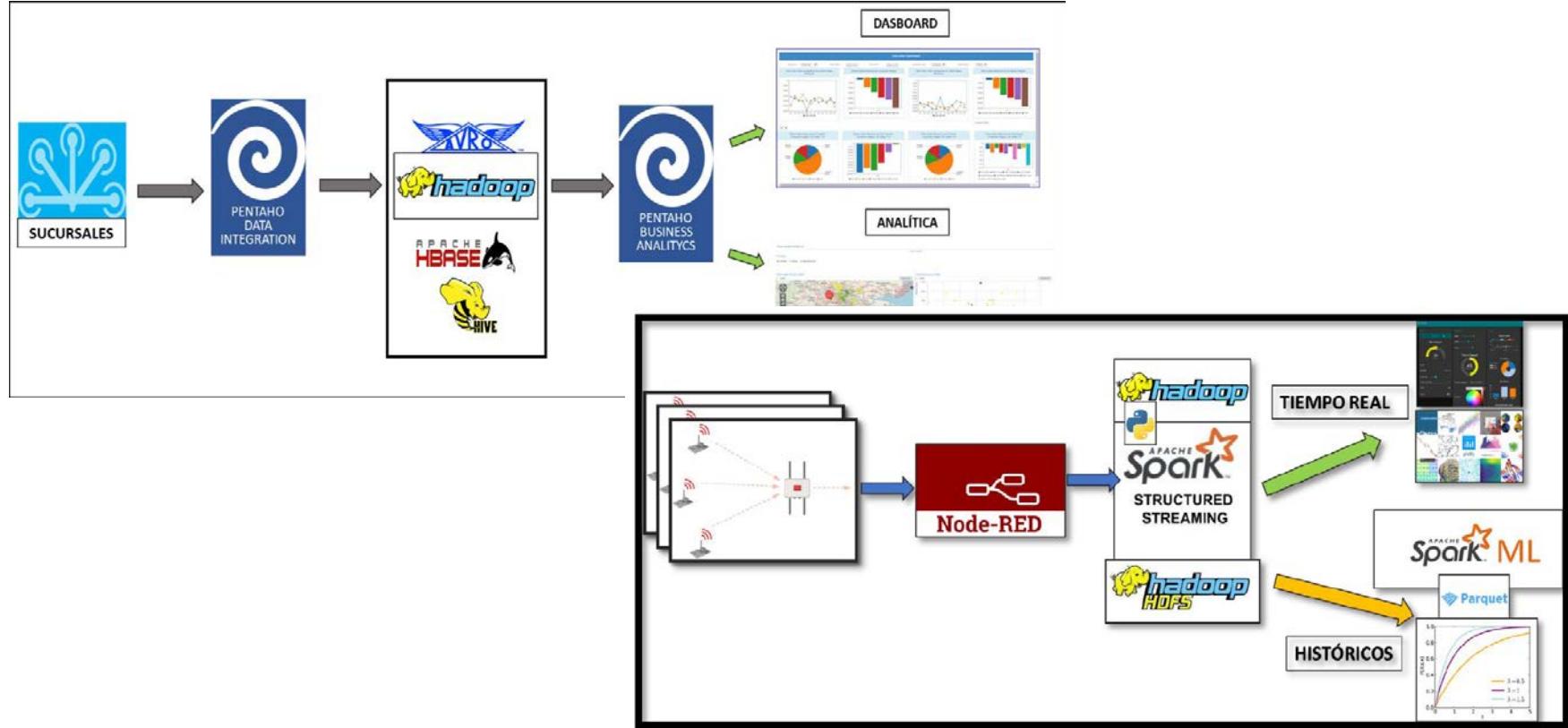
Ejemplo 4. A discutir. ¿Qué opinas?



Ejemplo 5. A discutir. ¿Qué opinas?



Ejemplo 6. A discutir. ¿Qué opinas?



Ejemplo 7. A discutir. ¿Qué opinas?

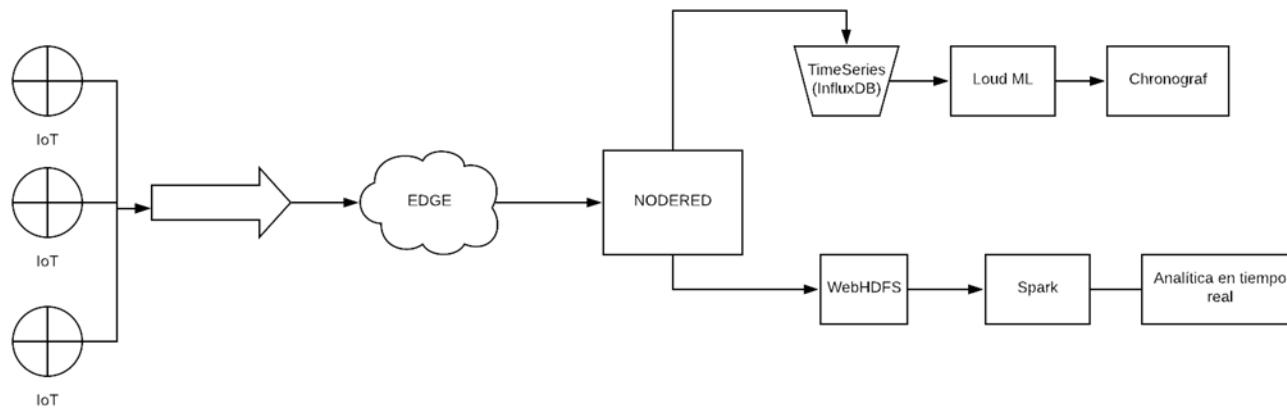
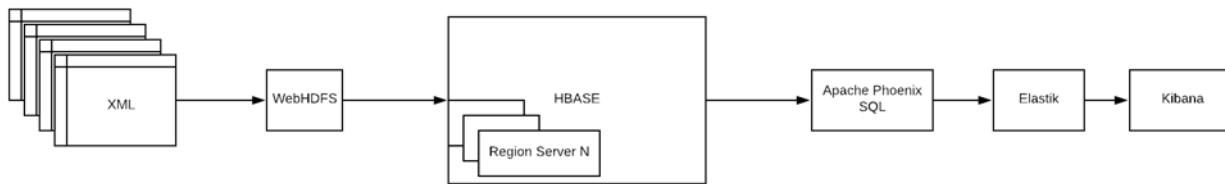
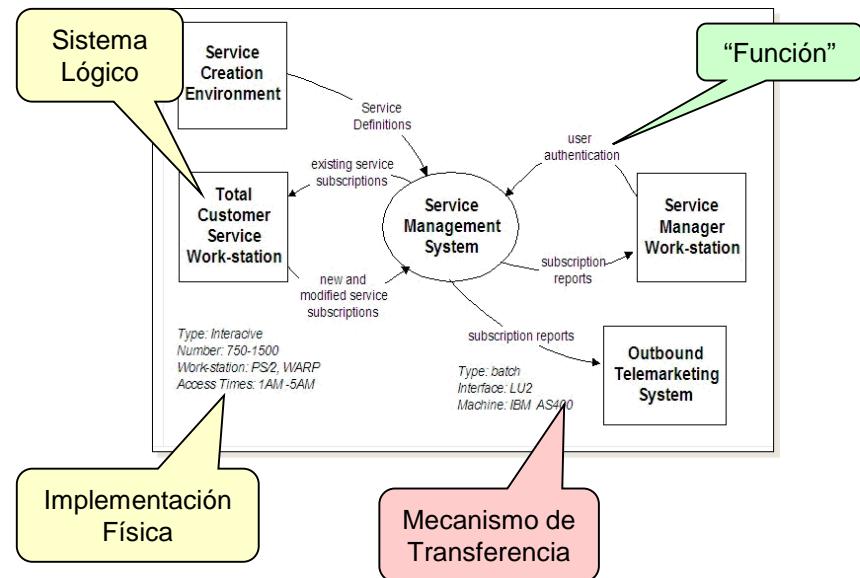
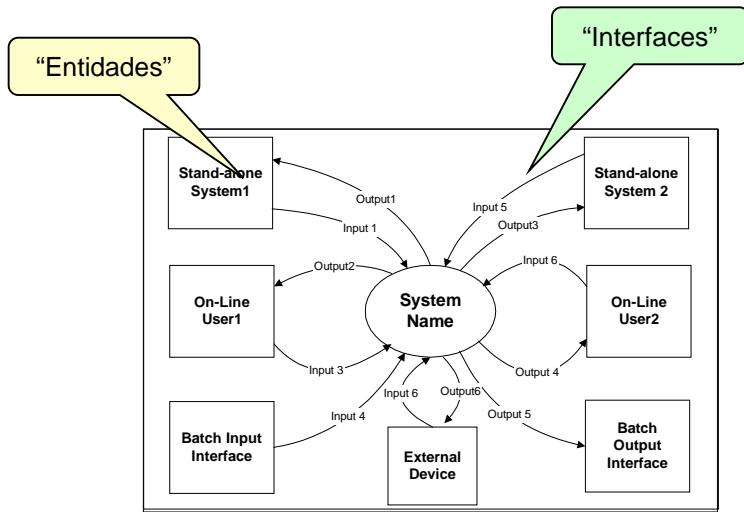


Diagrama de Contexto

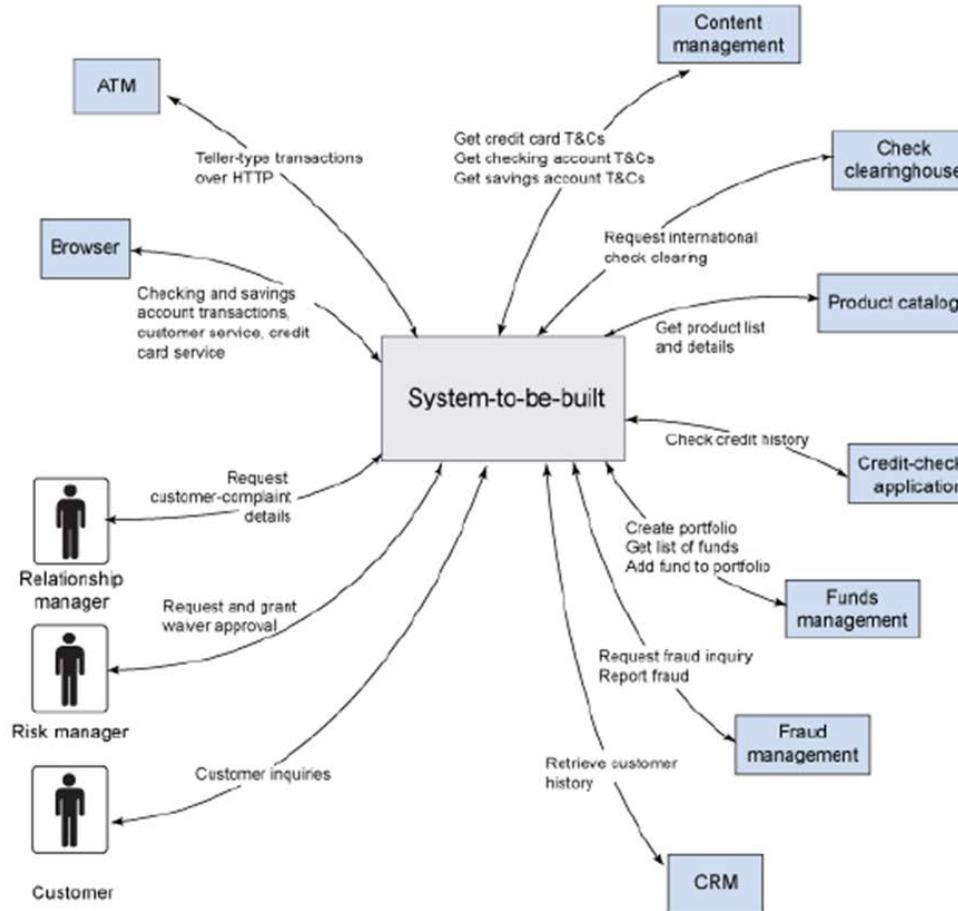
- Documenta cómo el sistema, representado por una “caja negra”, interactúa con entidades externas
- Define la información y flujo entre nuestra solución y los Actores (usuarios/personas y/o sistemas)
- Se usa para aclarar, confirmar y documentar el entorno en el que nuestro sistema va a operar
- La naturaleza de los sistemas externos, sus interfaces, la información y como fluye nos ayudarán en la fase de especificación de los componentes técnicos de nuestra arquitectura

Y por tanto deberá incluir

- Quién o qué interactúa con nuestra solución
- Qué hacen
- Cómo lo van a hacer



Más Ejemplos



Más ejemplos

