Corhuila

Sesión 5

Ago 2024

Visión de Arquitectura

Juan Carlos Ramírez
Snr Manager, Technology Operations and Support

Agosto 27 de 2024

Agenda

01 **Contexto (10 min)**

- Arquitectura
- TOGAF - Frameworks de Arquitectura

02 **Visión de Arquitectura (10 min)**

03 **Diagramas de Arquitectura (15 min)**

04 **Preguntas (15 min)**

01

Contexto

¿Qué es Arquitectura de Software?

"La arquitectura de software es el conjunto de estructuras para razonar acerca del sistema"[1]

Software Architecture in Practice: (By Len Bass, Paul Clements, Rick Kazman)

"La arquitectura de software es el conjunto de decisiones de diseño importantes para organizar el software, y promover los atributos de calidad deseados"[2]

Design it! From programmer to architect (Michael Keeling)

"La arquitectura de Software trata de las cosas importantes. Sea lo que sea"[3].

Design Patterns (Ralph Johnson)

[1] Ref.: https://www.amazon.com/Software-Architecture-Practice-3rd-Engineering/dp/0321815734

[2] Ref.: https://www.amazon.com/Design-Programmer-Architect-Pragmatic-Programmers/dp/1680502093

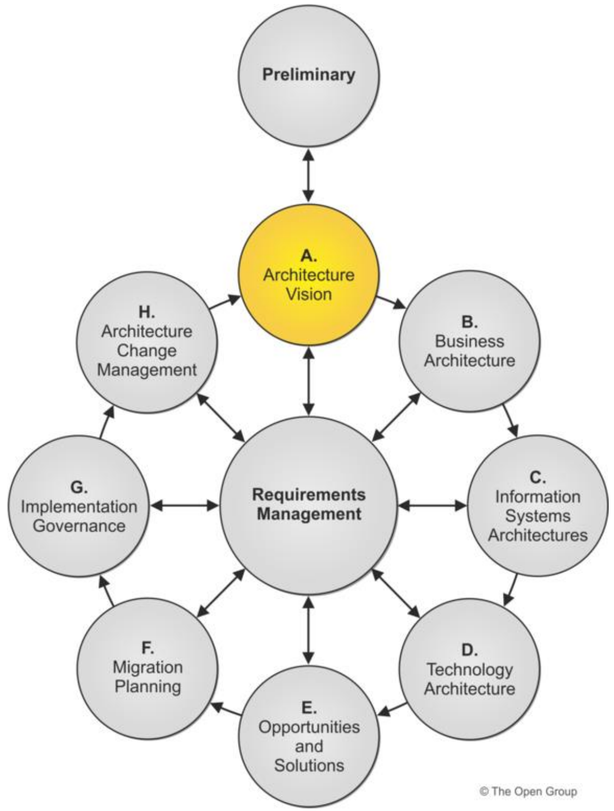
[3] Ref.: https://www.amazon.com/Design-Patterns-Elements-Reusable-Object-Oriented/dp/0201633612/

¿Qué son los Frameworks de Arquitectura?

¿Qué es TOGAF?

Marco de Referencia (Framework): Es un conjunto de buenas prácticas que proveen una estructura conceptual usada para **desarrollar**, **implementar** y **mantener** una arquitectura, sin importar la industria a la que aplique.

TOFAG: Es un Framework de Arquitectura que se centra en los objetivos del negocio modelando un ciclo de vida de procesos, documentación, diagramas y de buenas prácticas.



02

Visión de Arquitectura

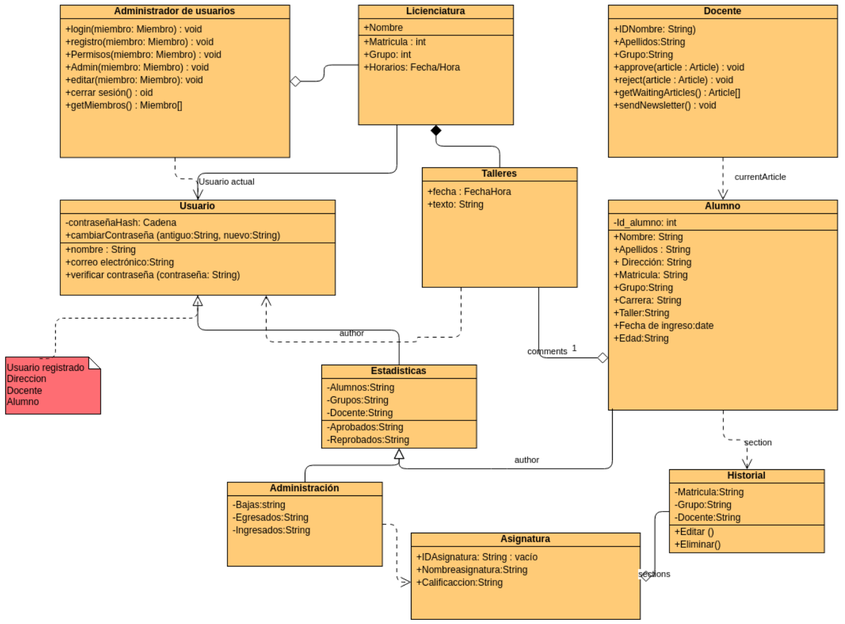
Vista de Arquitectura

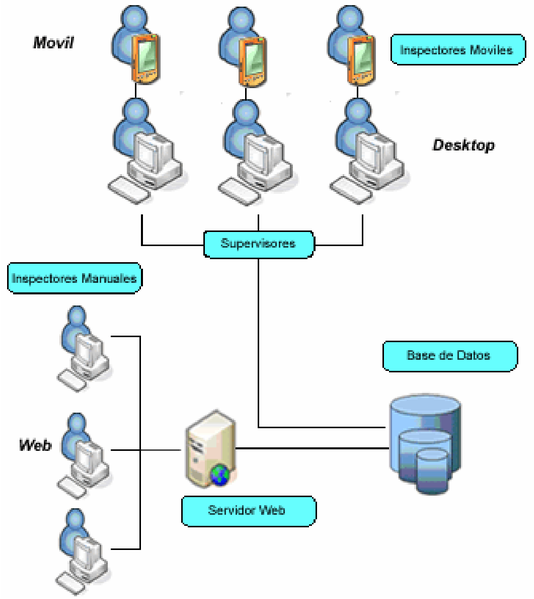
"Una vista es una representación de un **conjunto** de **elementos** de un sistema y las **relaciones entre ellos**"

| **Software Architecture in Practice: (By Len Bass, Paul Clements, Rick Kazman)**

Las vistas existen debido a que no es sencillo explicar un sistema complejo en una sola dimensión/diagrama. Ejemplo:

Una **vista de desarrollo** muestra elementos del código y las relaciones con las clases, pero no muestra los **modelos de datos** o los **diagramas de servidores**.



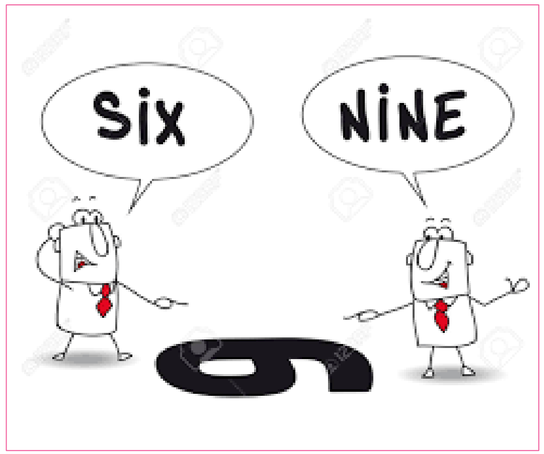


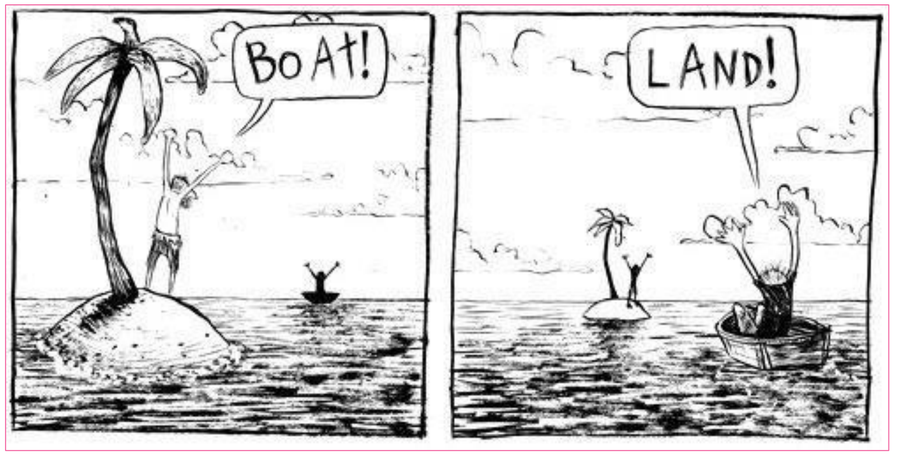
Punto de Vista de Arquitectura

Un punto de vista es un conjunto de reglas y directrices que definen cómo construir y utilizar una vista.

Se definen qué elementos, propiedades y relaciones incluir, cómo organizarlos y visualizarlos, y qué preguntas responder con la vista.

Por ejemplo, un punto de vista puede basarse en una notación estándar, como UML, o en una personalizada, como una tabla o un diagrama.



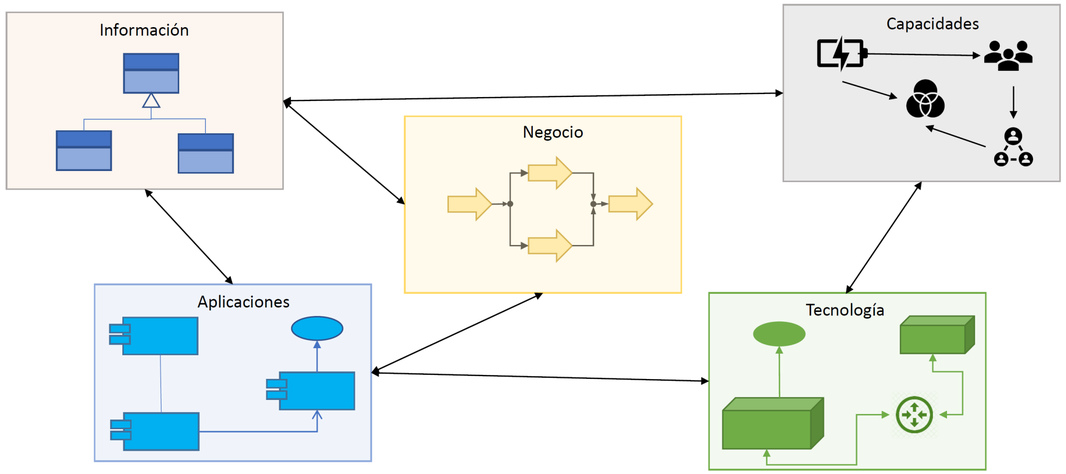


Vista y Punto de Vista de Arquitectura

- La arquitectura de un sistema es un conjunto de decisiones importantes que se deben tomar para satisfacer diferentes requisitos Funcionales y No Funcionales.

- El arquitecto necesita ver la realidad que va a mostrar desde diferentes vistas.

- El Arquitecto debe contar con herramientas que le permitan mostrar una arquitectura a los diferentes "Stakeholders":
  - Junta Directiva (Patrocinadores económicos)
  - Jefes de Departamento
  - Usuarios Técnicos
  - Usuarios Finales



03

Diagramas de Arquitectura

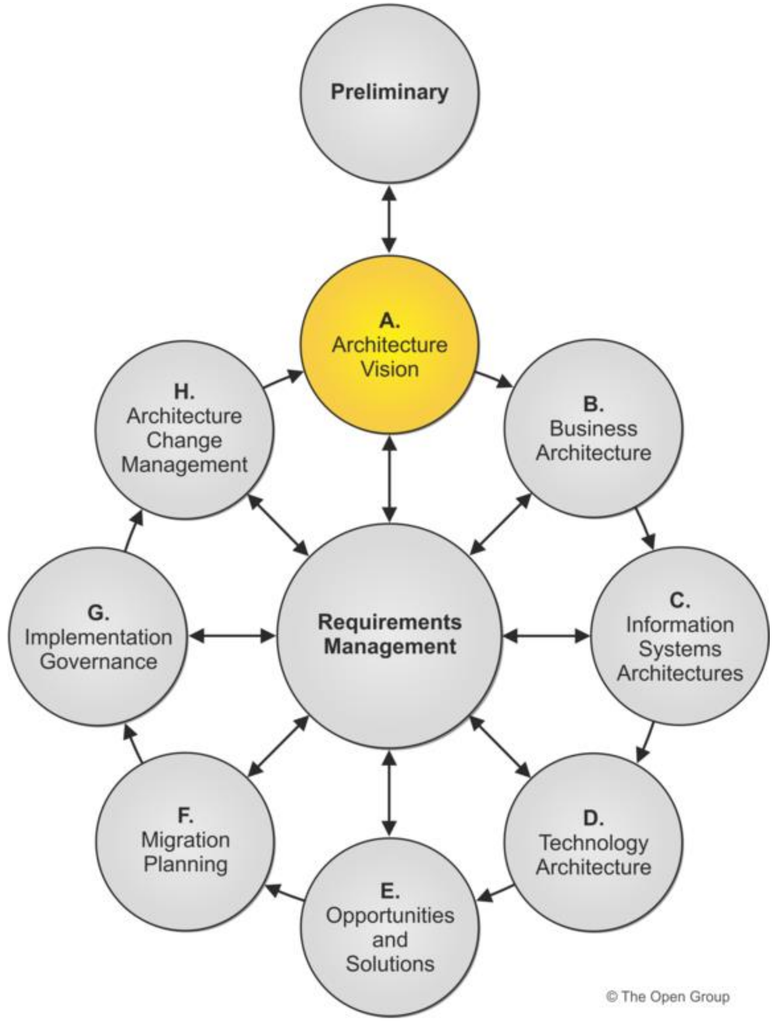
Visión de Arquitectura - TOGAF

Cuando se establece el alcance, restricciones y expectativas para un proyecto… se crea la visión de Arquitectura.

- Establecer las restricciones y expectativas de Arquitectura.
- Identificamos los KPI como elemento de Gobierno de TI
- Debemos definir los objetivos de negocio, Vistas de Negocio.
- Definimos a partir de los objetivos de Negocio: Roadmap, Arquitectura As-Is, To-Be.

Nombre	**Continuidad del negocio**
Referencia	PN-02
Declaración	Las operaciones empresariales se mantienen a pesar de las interrupciones del sistema.
Justificación	A medida que las operaciones del sistema se vuelven más generalizadas, nos volvemos más dependientes de ellas; por lo tanto, debemos considerar la confiabilidad de dichos sistemas en todo su diseño y uso.
Implicaciones	La dependencia de las aplicaciones del sistema exige que los riesgos de interrupción del negocio se gestionen y establezcan con anticipación.

Tabla 5: Principio de continuidad del negocio. (Fuente propia - Adaptación TOGAF).



Modelo de Vistas 4+1

El Modelo **4+1** es un modelo diseñado por Philippe Kruchten para "**describir** la **arquitectura** de sistemas software, **basado en** el uso **de múltiples vistas** concurrentes".

Las vistas suelen describir el sistema desde el punto de vista de diferentes interesados, tales como usuarios finales, desarrolladores o directores de proyecto.

Las cuatro vistas del modelo son:

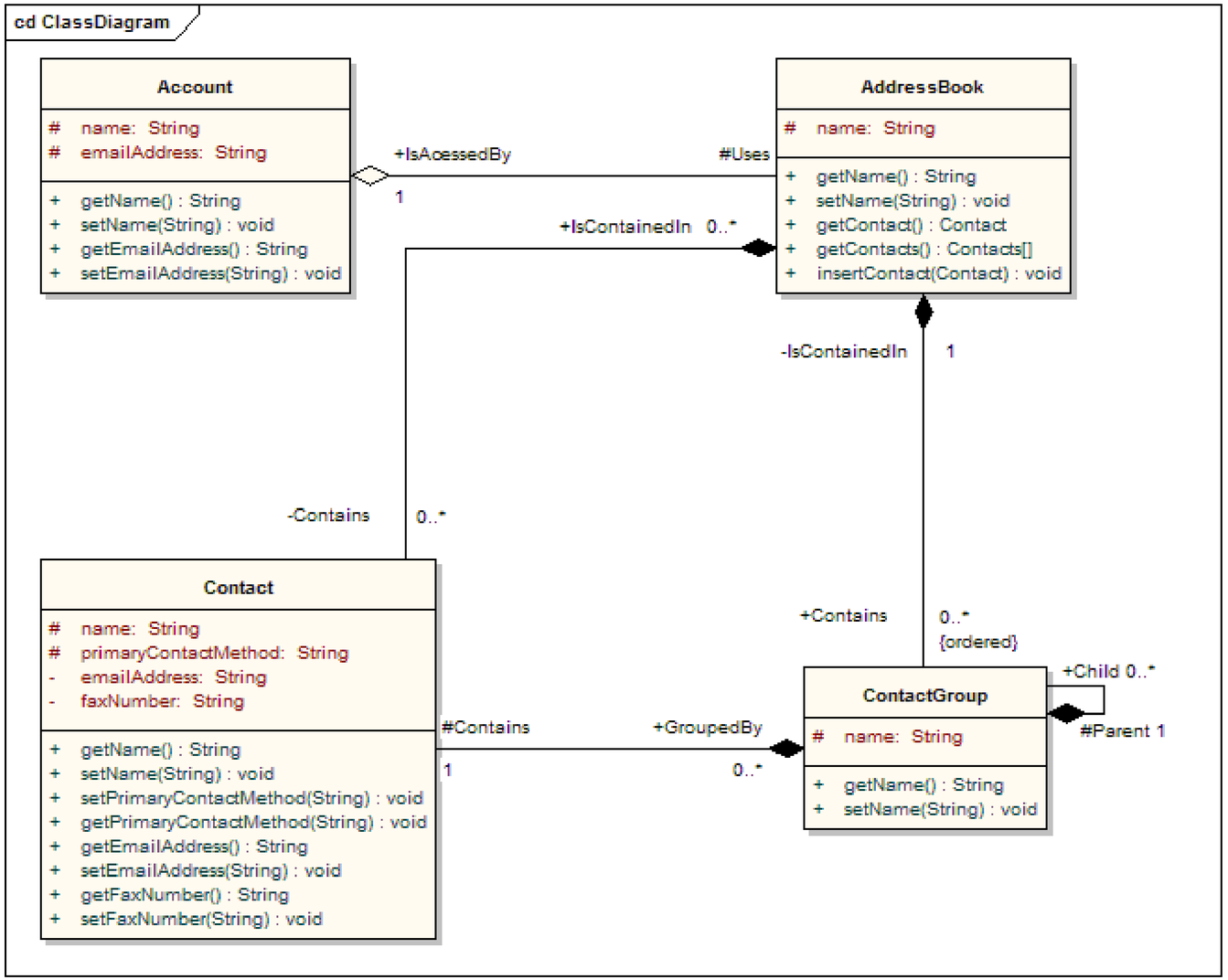
- Vista lógica
- Vista de desarrollo
- Vista de proceso
- Vista física

➢ La selección de casos de uso o escenarios suele utilizarse para ilustrar la arquitectura sirviendo como una vista más.

Modelo de Vistas 4+1

Vista Lógica

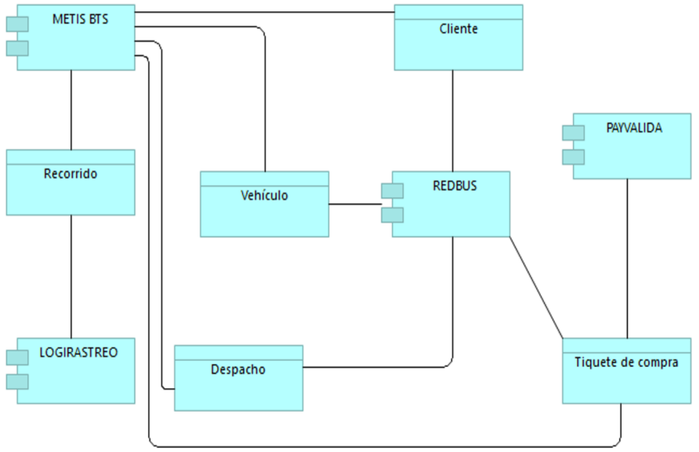
La vista lógica está enfocada en describir la estructura y funcionalidad del sistema. Los diagramas UML que se utilizan para representar esta vista son los Diagrama de Clase, Diagrama de Comunicación:

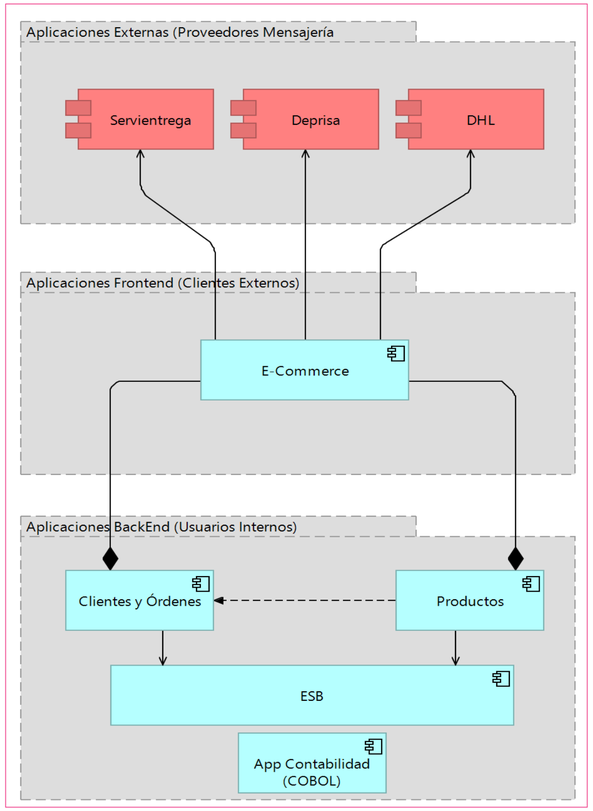


Modelo de Vistas 4+1

Vista de Desarrollo

La vista de desarrollo ilustra el sistema de la perspectiva del programador y está enfocado en la administración de los artefactos de software. Esta vista también se conoce como vista de implementación. Utiliza el Diagrama de Componentes UML para describir los componentes de sistema. Otro diagrama UML que se utiliza en la vista de desarrollo es el Diagrama de Paquetes:

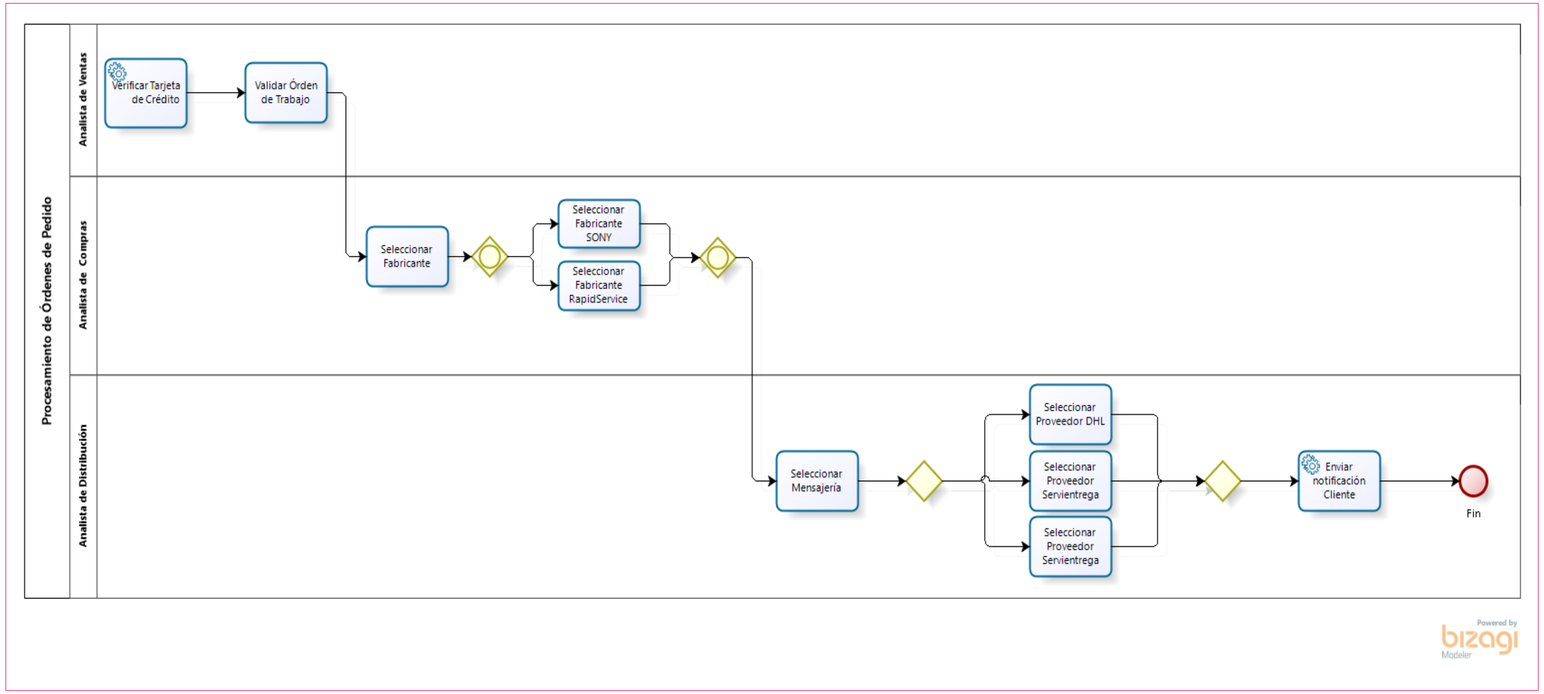




Modelo de Vistas 4+1

Vista de Procesos

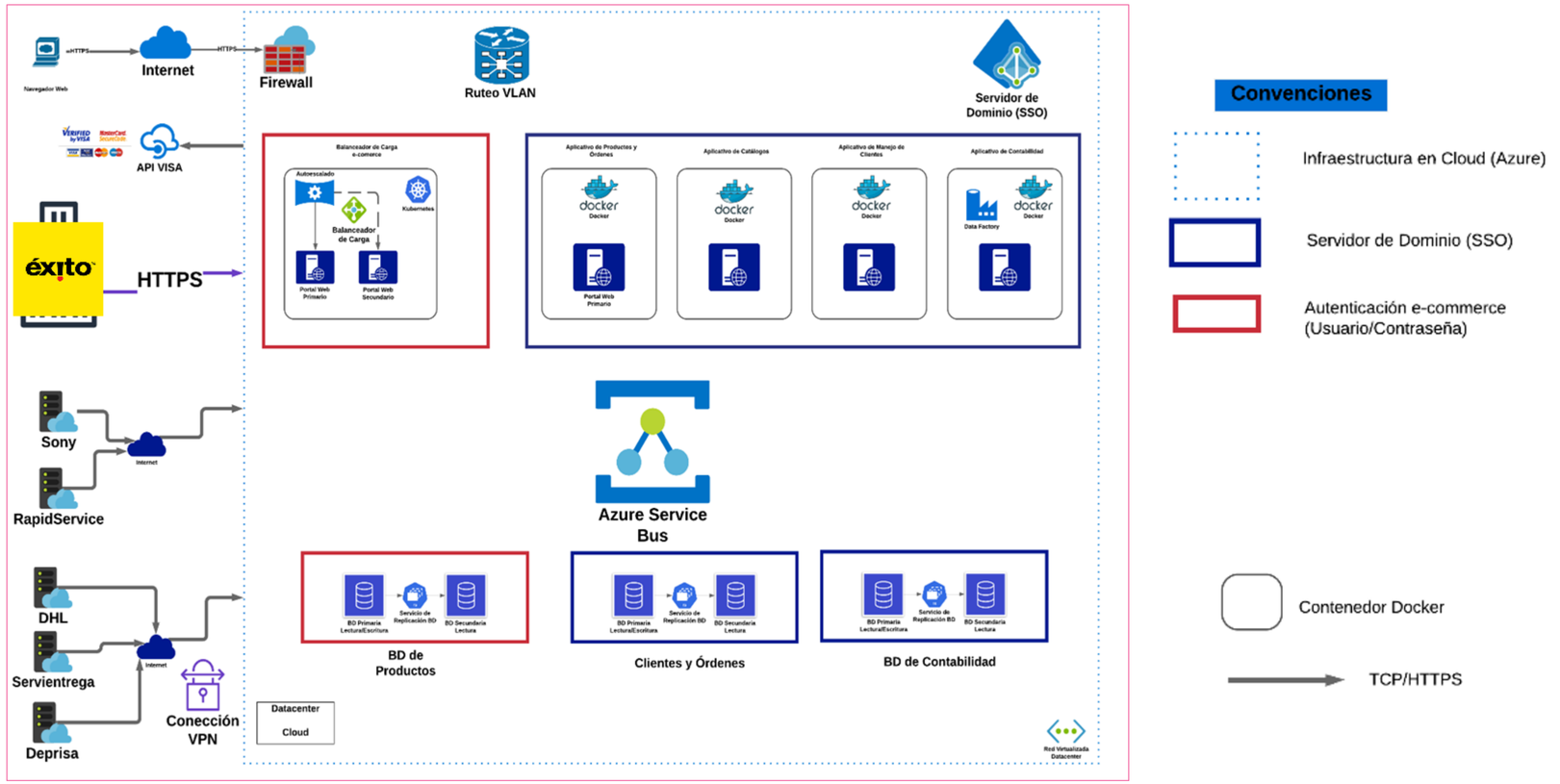
La vista de proceso trata los aspectos dinámicos del sistema, explica los procesos de sistema y cómo se comunican. Se enfoca en el comportamiento del sistema en tiempo de ejecución. La vista considera aspectos de concurrencia, distribución, rendimiento, escalabilidad, etc. En UML se utiliza el Diagrama de Actividad o Diagrama de Secuencia para representar esta vista:



Modelo de Vistas 4+1

Vista Física

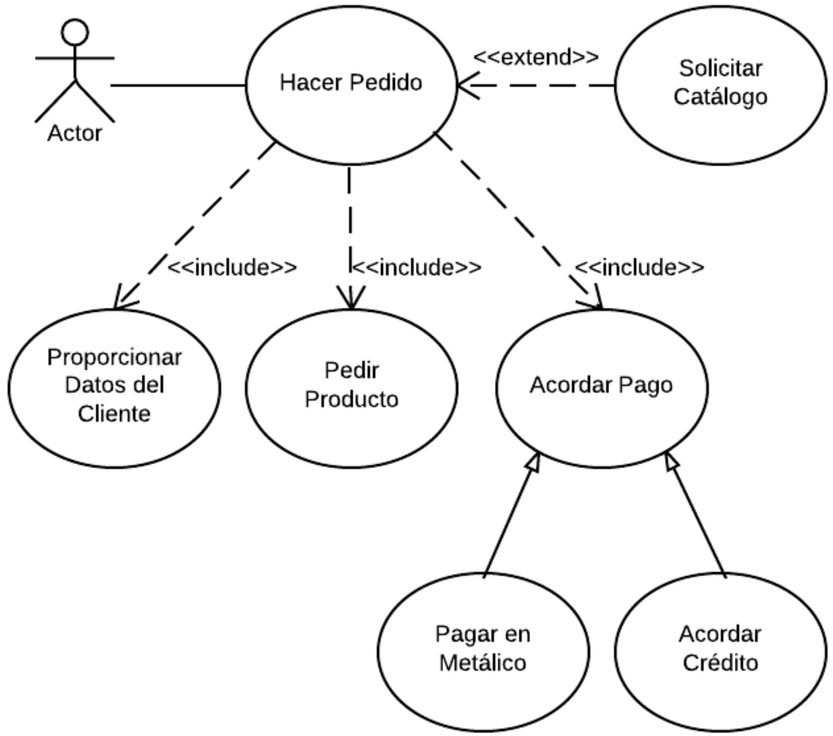
La vista física describe el sistema desde el punto de vista de un ingeniero de sistemas. Está relacionada con la topología de componentes de software en la capa física, así como las conexiones físicas entre estos componentes. Esta vista también se conoce como vista de despliegue. En UML se utiliza el Diagrama de Despliegue para representar esta vista:

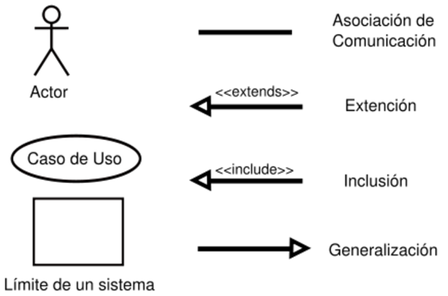


Modelo de Vistas 4+1

Vista Escenarios

La descripción de la arquitectura se ilustra utilizando un conjunto de casos de uso, o escenarios lo que genera una quinta vista. Los escenarios describen secuencias de interacciones entre objetos, y entre procesos. Se utilizan para identificar y validar el diseño de arquitectura. Esta vista es también conocida como vista de casos de uso.





05

Preguntas...

Técnica priorización: MoSCow

Must Have – Debe Tener

- No negociable.
- Producto mínimo viable.
- Incapaz de entregar el producto final sin esto.
- Inseguro sin esto.
- Incumplimiento sin esto.
- Proyecto no viable sin esto.

Should Have – Debería Tener

- Importante pero no vital.
- Se puede abordar con una solución alternativa (workaround).
- Puede ser “doloroso” no hacerlo, pero la solución sigue siendo viable.

Could Have – Podría Tener

- Deseable pero no tan importante como un “Should Have”.
- Solo se hace si se cuenta con recurso disponible (tiempo y presupuesto).

Won’t Have – No Debe Tener

- Fuera de alcance (Presupuesto y/o tiempo).
- Sería bueno tenerlo, pero no genera ningún impacto real.

Priorización MoSCoW

Must Have – Debe Tener

Should Have – Debería Tener

Could Have – Podría Tener

Won't Have – No Debe Tener

Gracias!