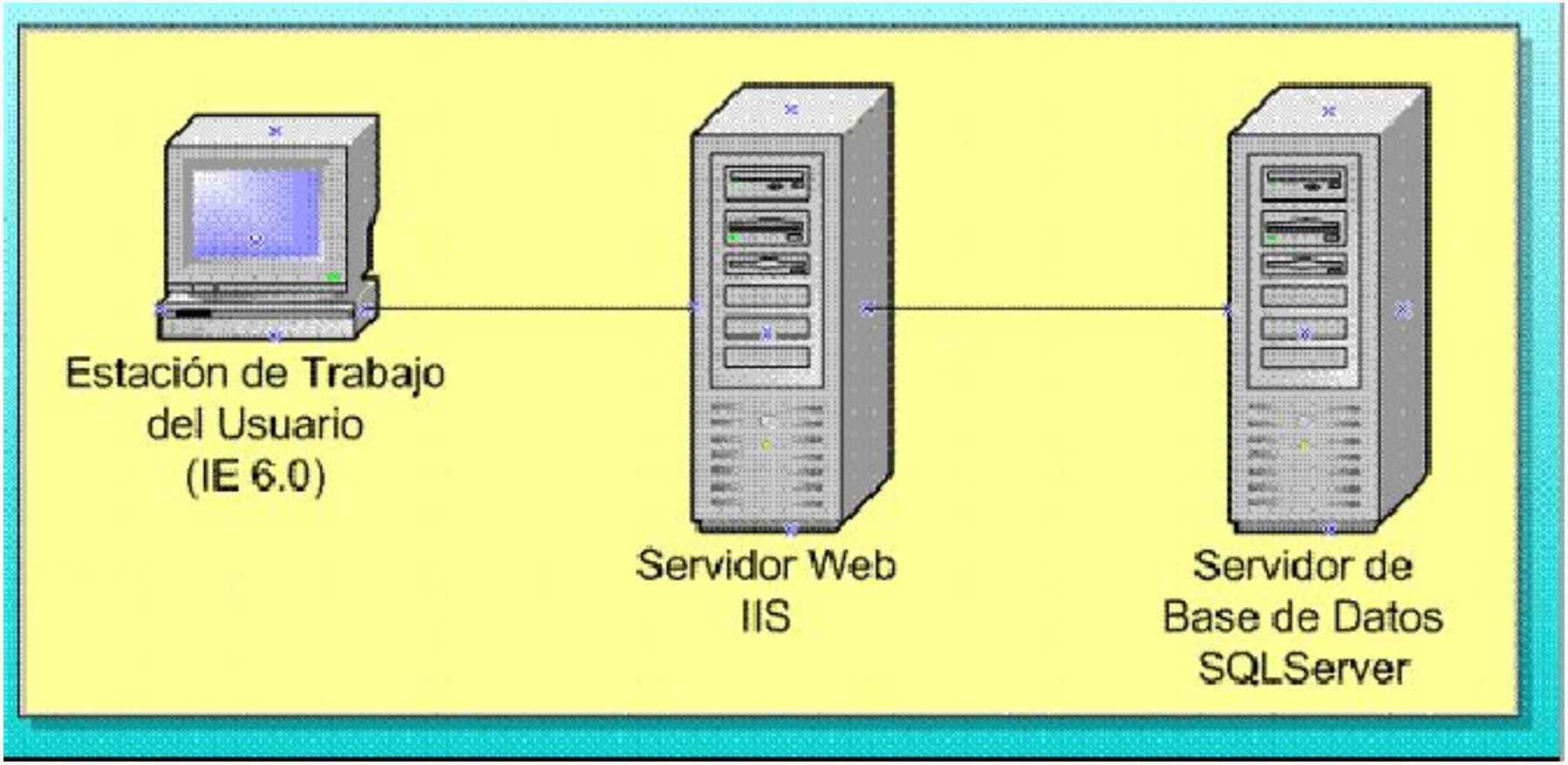
Ejercicio

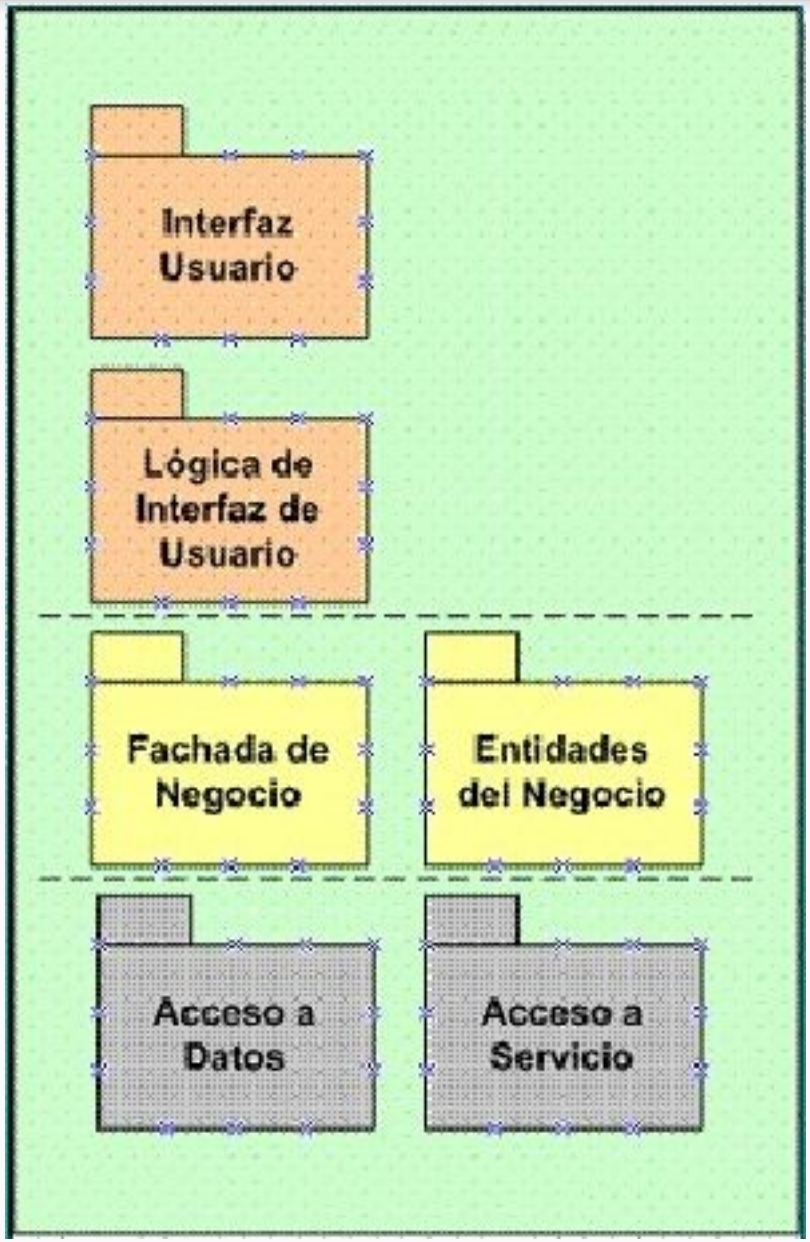
Si nos piden contar y describir la arquitectura del último sistema en que hemos participado, que haríamos?

Podemos encontrarnos con cosas como:

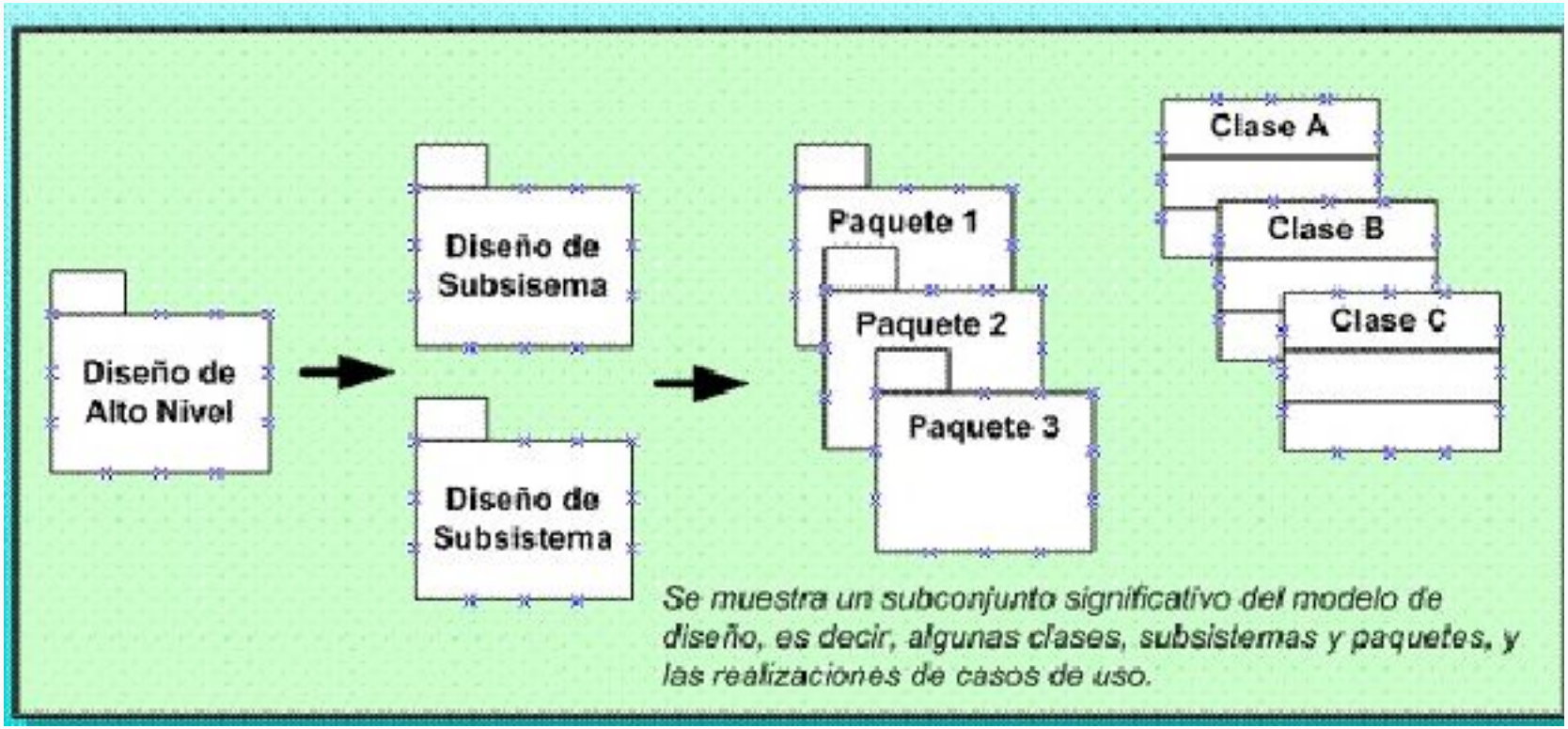


El sistema tendría una Arquitectura Web basada en plataforma Microsoft, o sea, un cliente usando un Internet Explorer 6.0, un servidor corriendo IIS y los componentes de sistema desarrollados en ASP.NET accediendo a otro servidor que contendría los datos dentro de un SQL Server 2008

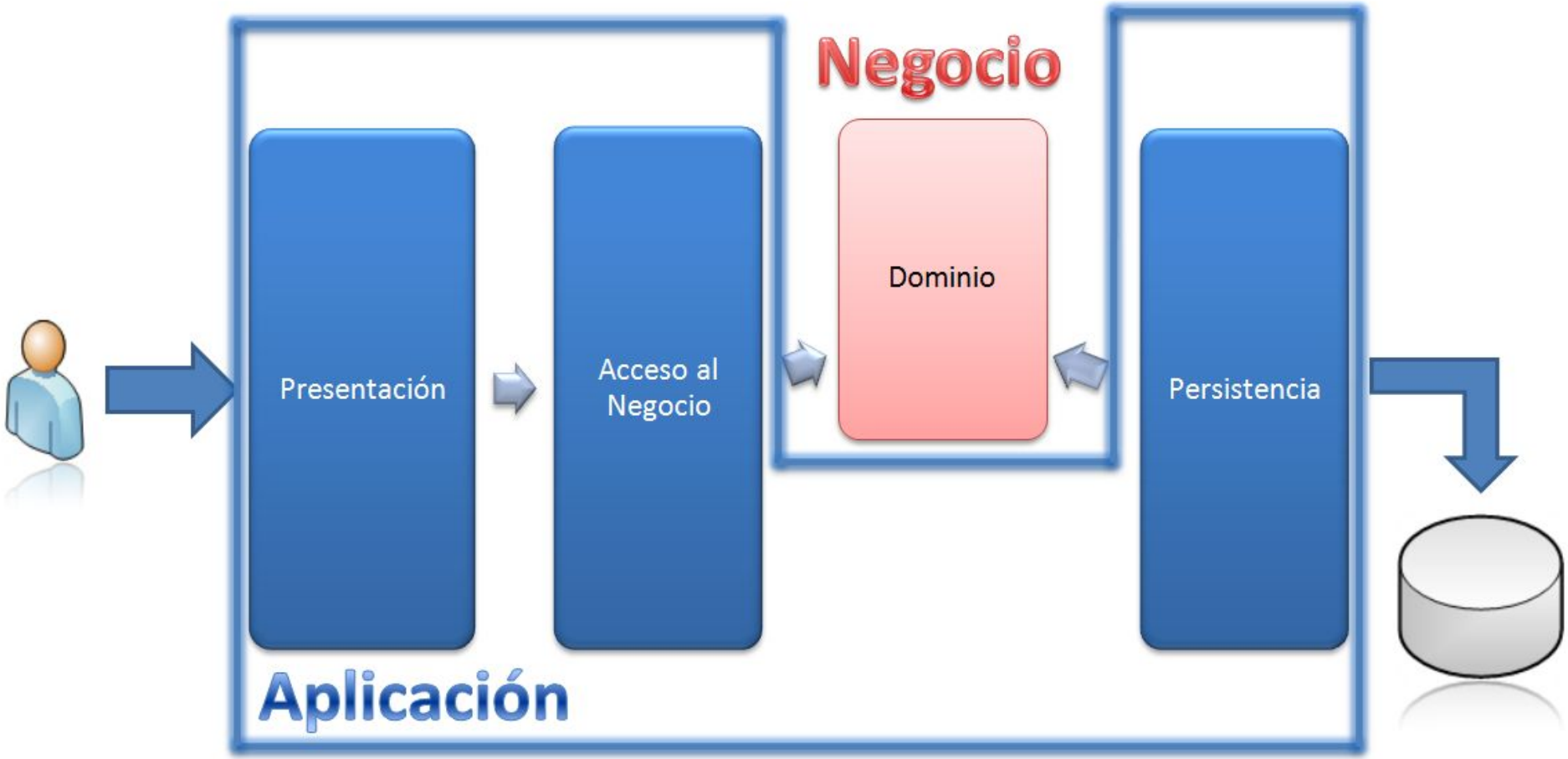
Podemos encontrarnos con cosas como:



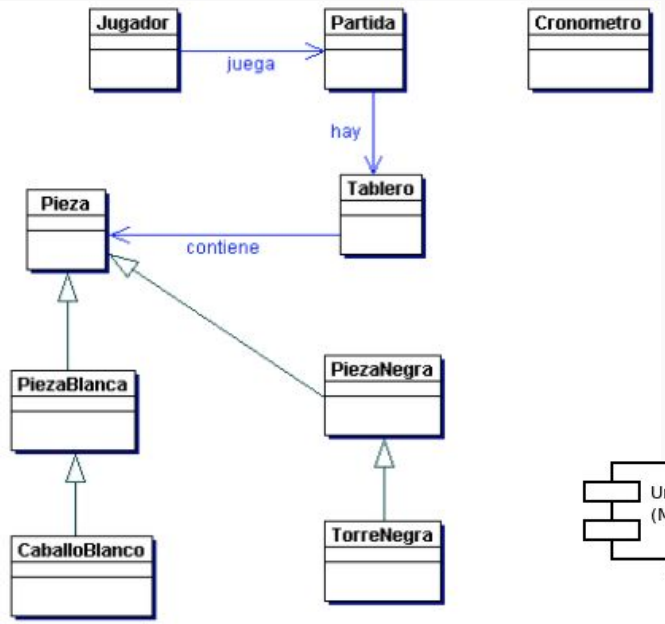
Podemos encontrarnos con cosas como:

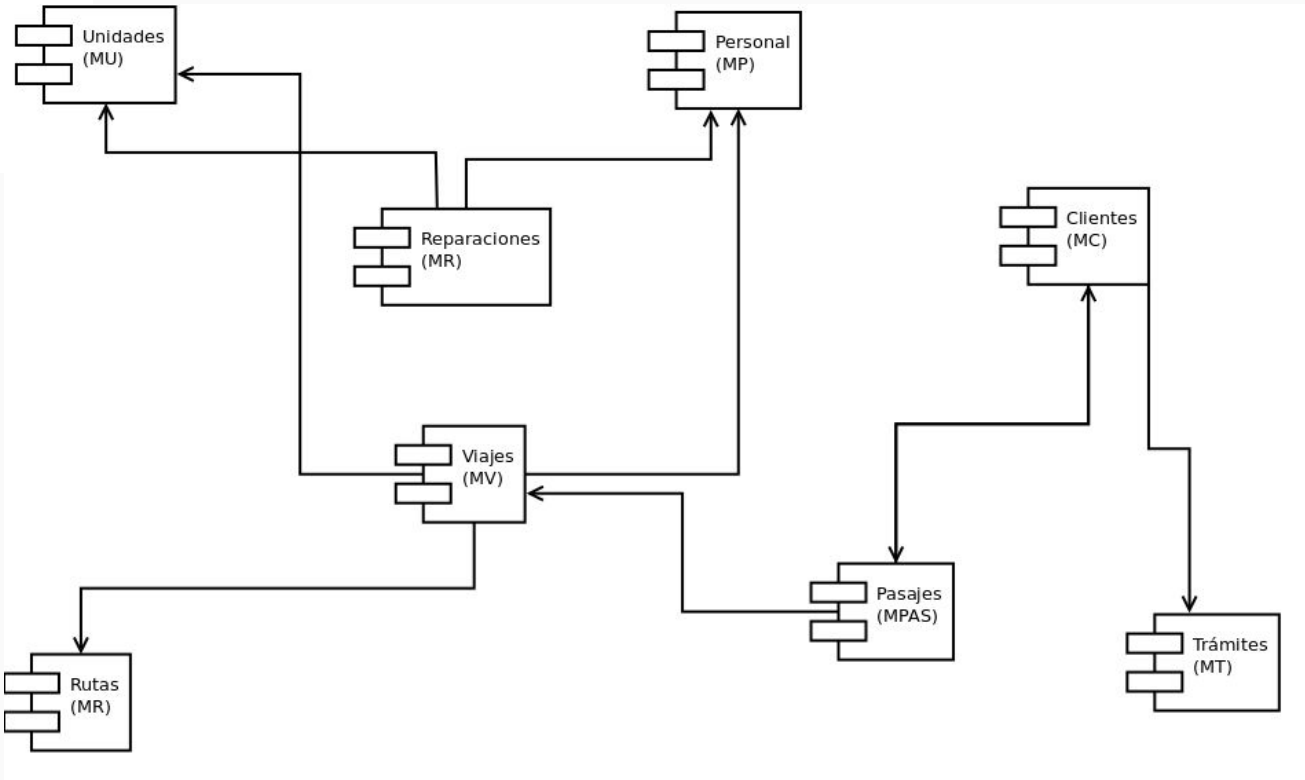


Podemos encontrarnos con cosas como:



Podemos encontrarnos con cosas como:





Arquitectura de Software

Arquitectura

ieee-1471 defining-architecture

- Conceptos o propiedades fundamentales de un sistema en su entorno encarnado en sus elementos, relaciones y en los principios de su diseño y evolución

- La arquitectura de software representa la estructura o las estructuras del sistema, que consta de componentes de software, las propiedades visibles externamente y las relaciones entre ellas.

Arquitectura

La arquitectura de software son aquellas decisiones que son importantes y difíciles de cambiar.

Martin Fowler

Arquitectura

- ¿Cómo contamos la arquitectura de un sistema?
- ¿Cómo diseñamos una posible arquitectura para un sistema como para hacer una prueba de concepto?
- ¿Cómo documentamos la arquitectura de un sistema?

Un paréntesis..

- ¿Qué debemos tener en cuenta cuando escribimos un documento, un informe o queremos transmitir alguna información?

Un paréntesis..

Entonces, ¿a quién está dirigido un Documento de Arquitectura?

Arquitectura

Dentro de un equipo de trabajo podemos encontrar:

- Clientes
- Usuarios
- Líderes de proyectos
- Product owners
- Arquitectos
- Analistas
- Scrum masters
- Desarrolladores
- Testers
- Administradores de bases de datos
- Gente de infraestructura, de despliegue

- etc.

Arquitectura

En realidad existen muchos interesados o stakeholders de un documento de arquitectura, e incluso estos pueden tener distintos intereses y conocimientos.

Entonces, ¿cómo escribir un único documento o diagrama que pueda explicar la arquitectura propuesta para resolver cierto problema?

Modelo de Vistas 4+1

Modelo de Vistas 4+1 - Kruchten

VISTA LOGICA

La vista lógica apoya principalmente los requisitos funcionales –lo que el sistema debe brindar en términos de servicios a sus usuarios.

Aquí se aplican los principios de abstracción, encapsulamiento y herencia. Esta descomposición no sólo se hace para potenciar el análisis funcional, sino también sirve para identificar mecanismos y elementos de diseño comunes a diversas partes del sistema

Modelo de Vistas 4+1 - Kruchten

VISTA DE PROCESOS

La vista de procesos toma en cuenta algunos requisitos no funcionales tales como el rendimiento y la disponibilidad.

Se enfoca en asuntos de concurrencia y distribución, integridad del sistema, de tolerancia a fallas.

Un proceso es una agrupación de tareas que forman una unidad ejecutable. Los procesos representan el nivel al que la arquitectura de procesos puede ser controlada tácticamente (i.e., comenzar, recuperar, reconfigurar, y detener). Además, los procesos pueden replicarse para aumentar la distribución de la carga de procesamiento, o para mejorar la disponibilidad.

Modelo de Vistas 4+1 - Kruchten

VISTA DE DESARROLLO (o de **COMPONENTES**)

La vista de desarrollo se centra en la organización real de los módulos de software en el ambiente de desarrollo del software.

El software se empaqueta en partes pequeñas –bibliotecas de programas o subsistemas– que pueden ser desarrollados por uno o un grupo pequeño de desarrolladores.

La vista de desarrollo tiene en cuenta los requisitos internos relativos a la facilidad de desarrollo, administración del software, reutilización y elementos comunes, y restricciones impuestas por las herramientas o el lenguaje de programación que se use.

Modelo de Vistas 4+1 - Kruchten

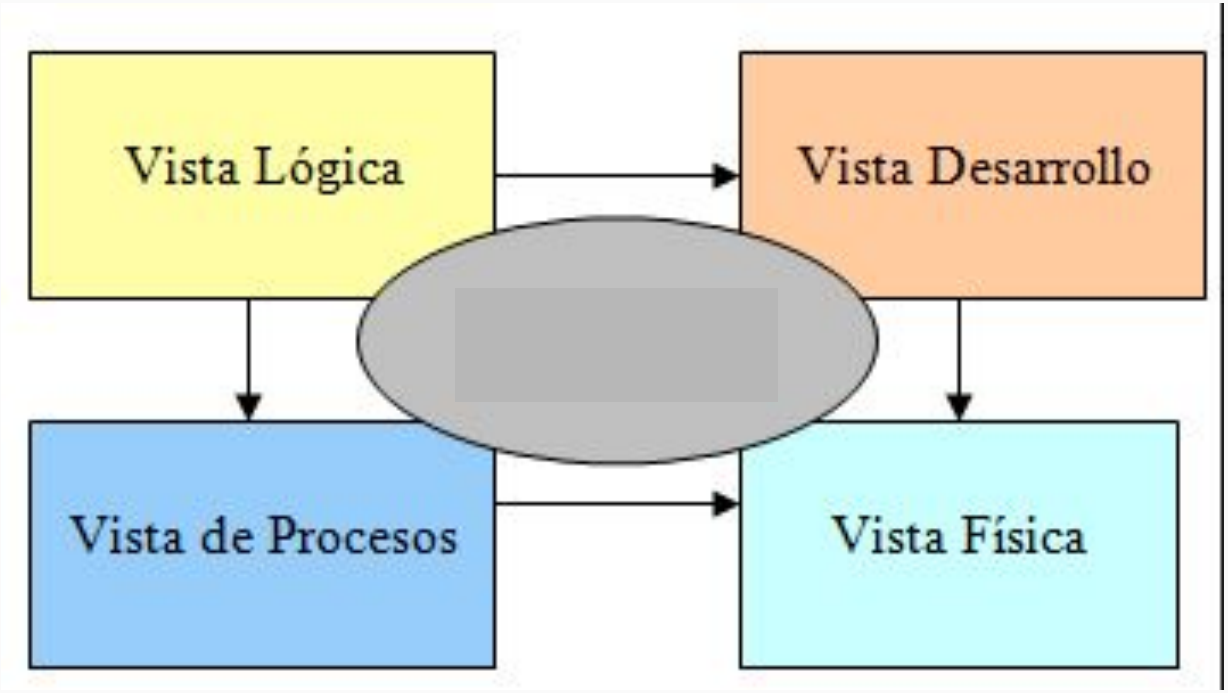
VISTA FISICA (o de **DESPLIEGUE**)

La vista física toma en cuenta primeramente los requisitos no funcionales del sistema tales como la disponibilidad, confiabilidad (tolerancia a fallas), rendimiento (throughput), y escalabilidad.

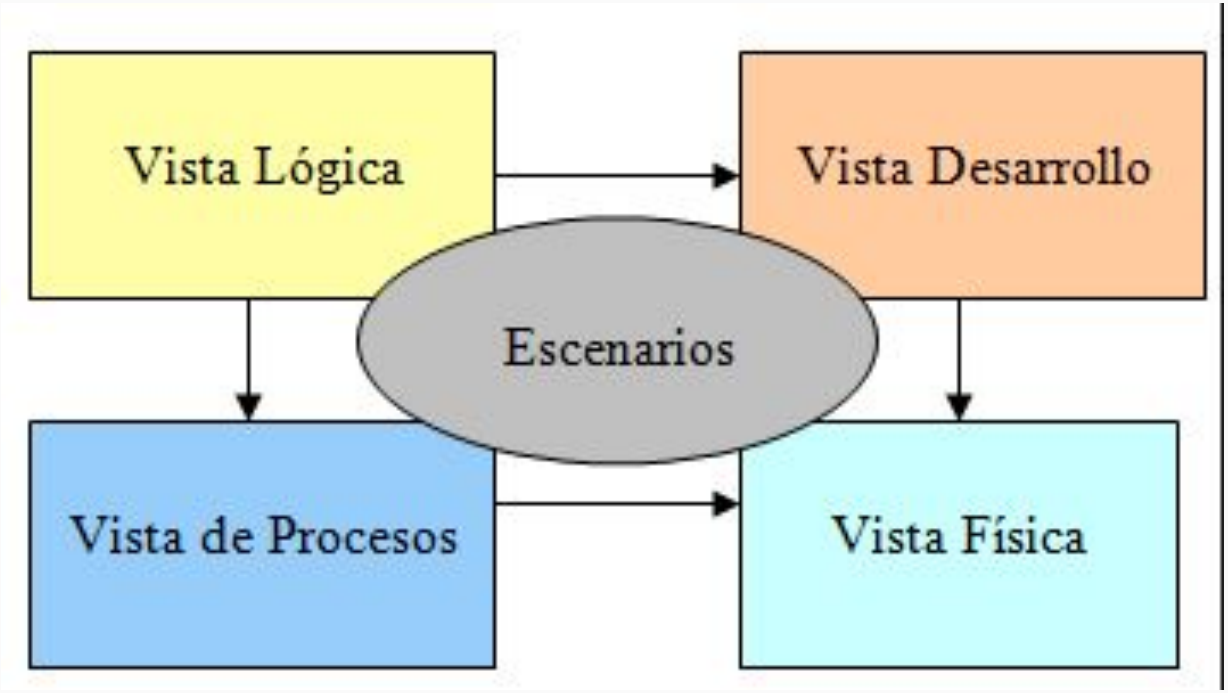
El software se ejecuta sobre una red de computadores o nodos de procesamiento. Los variados elementos identificados –redes, procesos, tareas y objetos– requieren ser mapeados sobre los nodos.

Esperamos que diferentes configuraciones puedan usarse: algunas para desarrollo y pruebas, otras para mostrar el sistema en varios sitios para distintos usuarios. Por lo tanto, la relación del software en los nodos debe ser altamente flexible y tener un impacto mínimo sobre el código fuente.

Modelo de Vistas 4+1 - Kruchten



Modelo de Vistas 4+1 - Kruchten



Modelo de Vistas 4+1 - Kruchten

VISTA DE ESCENARIOS

Los elementos de las cuatro vistas trabajan conjuntamente en forma natural mediante el uso de un conjunto pequeño de escenarios relevantes.

Los escenarios son de alguna manera una abstracción de los requisitos más importantes.

Sirve a dos propósitos principales:

• Como una guía para descubrir elementos arquitectónicos durante el diseño de arquitectura

• Como un rol de validación e ilustración después de completar el diseño de arquitectura, en el papel y como punto de partido de las pruebas de un prototipo de la arquitectura.

Modelo de Vistas 4+1 - Kruchten

CONCLUSIONES

- Permite a través de diferentes vistas analizar distintas perspectivas del problema, focalizándose en el problema en cuestión

- Concentra en un único documento las principales decisiones tomadas sobre el sistema

- Permite a nuevos integrantes del equipo entender la arquitectura del sistema y ubicarse dentro de la solución

- Permite discutir con todos los stakeholders las distintas decisiones y validarlas en una etapa temprana

Decisiones de Arquitectura

Decisiones de Arquitectura son decisiones de diseño que abordan requisitos significativos desde el punto de vista arquitectónico; se perciben como difíciles de hacer y/o costosos de cambiar.

Estas decisiones son importantes de guardarlas y conocerlas. Conocer el motivo por el cual se tomó dicha decisión en la arquitectura o diseño.

Por ejemplo, para el envío de emails se usa un SMTP y realiza el envío programáticamente gestionado por nuestro backend o se delega a un servicio de *email provider* de terceros como ser Sendgrid, Mailgun, etc.

- Estado (propuesta, aceptada, deprecada, sustituida)
- Contexto
- Decisión
- Consecuencias

Decisiones de Arquitectura

10. Scalable Queue Messaging Platform

Status

proposed

Context

Our current queue messaging platform (RabbitMQ) is a self-hosted service. And already became a headache for the infrastructure team to manage. We're in need to handle up to 150K msgs/s for internal usage in Tumorang Finance.

Two options exist

1. Use managed RabbitMQ vendors (CloudAMQP.com)
   Identified issue to address:
   - **internal application changes**: there are no changes on the application side because we will only be rerouting the request from self-hosted to managed RabbitMQ in CloudAMQP.com.
   - **performances:** there will be a significant impact on the performance side because in CloudAMQP we can set multi-region clusters and we will also be able to provision a GCP instance that reduces network call latency. Based on the pricing plan, it will be able to handle up to 200K msgs/s which is covered our needs of 150K msgs/s
   - **pricing Cost:** it costs us $ 17.5K flat monthly for the multi-AZ cluster in CloudAMQP for a max of 200K msgs/s.
2. Migrate to Google PubSub
   Identified issue to address
   - **internal application changes**: There will be breaking changes and a big migrations processes from RabbitMQ (AMQP) protocol to Google PubSub. Considering the total services that are connected to RabbitMQ is around 50 repositories, with an estimation 1 sprint/repository for migration effort, it may take 50 sprint to finally migrate to Google PubSub
   - **performances:** With Google PubSub high scalability it's able to handle up to 4 GB/s throughput
   - **pricing cost:** it costs us up to $10K per month depending on how many messages are being published/consumed in Google PubSub.

Decision

We will migrate to Google PubSub

Consequences

- The internal applications will be required to be refactored to follow Google Pub/Sub contract
- The development and the migration process may take 50 Sprint (with 2 weeks sprint) to be finished.
- A clear RFC and constant reviews of the RFC will be required until the migrations are finished.

Decisiones de Arquitectura

10. Scalable Queue Messaging Platform

Status

accepted

Context

Our current queue messaging platform (RabbitMQ) is a self-hosted service. And already became a headache for the infrastructure team to manage. We're in need to handle up to 150K msgs/s for internal usage in Tumorang Finance.

Two options exist

1. Use managed RabbitMQ vendors (CloudAMQP.com)
   Identified issue to address:
   - **internal application changes:** there are no changes on the application side because we will only be rerouting the request from self-hosted to managed RabbitMQ in CloudAMQP.com.
   - **performances:** there will be a significant impact on the performance side because in CloudAMQP we can set multi-region clusters and we will also be able to provision a GCP instance that reduces network call latency. Based on the pricing plan, it will be able to handle up to 200K msgs/s which is covered our needs of 150K msgs/s
   - **pricing Cost:** it costs us $ 17.5K flat monthly for the multi-AZ cluster in CloudAMQP for a max of 200K msgs/s.
2. Migrate to Google PubSub
   Identified issue to address
   - **internal application changes**: There will be breaking changes and a big migrations processes from RabbitMQ (AMQP) protocol to Google PubSub. Considering the total services that are connected to RabbitMQ is around 50 repositories, with an estimation 1 sprint/repository for migration effort, it may take 50 sprint to finally migrate to Google PubSub
   - **performances:** With Google PubSub high scalability it's able to handle up to 4 GB/s throughput
   - **pricing cost:** it costs us up to $10K per month depending on how many messages are being published/consumed in Google PubSub.

Decision

We will migrate to Google PubSub

Consequences

- The internal applications will be required to be refactored to follow Google Pub/Sub contract
- The development and the migration process may take 50 Sprint (with 2 weeks sprint) to be finished.
- A clear RFC and constant reviews of the RFC will be required until the migrations are finished.

1. Using RabbitMQ over SQS

Status

superseded by ADR-10

Context

The infrastructure cloud platform that we use is Google Cloud Platform (GCP). Our request is around 120 requests/seconds. We need a queue that will be used by our system to handle the queueing logic for user.

There are 2 options that we have in mind.

- **Option 1:** Use self-hosted RabbitMQ on GCP and utilize the RabbitMQ feature (AMQP).
- **Option 2:** Use SQS simple queueing platform from AWS

Decision

We will use RabbitMQ in GCP

We currently have low requests (120 requests/seconds). And anticipate this will remain stable for the foreseeable future. Therefore we don't see any performance issue with the option we choose.

We also need complex logic like dead-letter-queueing from RabbitMQ, the broadcasting (fan-out) feature that AMQP offer for development needs. So RabbitMQ is a preferable choice in this case compared to SQS.

Using Amazon MQ might be a good choice for fully managed RabbitMQ. But since most systems are in GCP. To avoid lots of network calls between 2 vendors, owning a self-hosted RabbitMQ is the most accepted solution.

Consequences

- Maintaining the self-hosted RabbitMQ will be an extra job for Infra team

Bibliografía

- 4+1view-architecture-Krutchen.pdf
- Making Architecture Matter - Martin Fowler Keynote
- http://www.iso-architecture.org/ieee-1471/defining-architecture.html
- https://github.com/7510-tecnicas-de-disenio/material-clases/tree/master/lecturas-recomendadas