ARQUITECTURA

Ingenieria de Software IIC2143

CONTENIDOS

1

Arquitectura

2

Patrones de arquitectura

3

Ejercicios

QUE ES UNA AQUITECTURA DE SOFTWARE?

- Estructura mas abstracta del software a desarrollar
- Es la Descomposición del sistema en sus grandes componentes y la forma que interactúan estos
- Generalmente el diseños de estas toma en cuenta los requerimientos minimos del sistema

Factores que la influencian

Performance
Security
Availability
Maintainability

MODELO C4

Este modelo posee 4 estructuras bases:

Contexto: Describe qué y quienes interactúan en el sistema

Contenedores: Define los grandes subsistemas de la plataforma

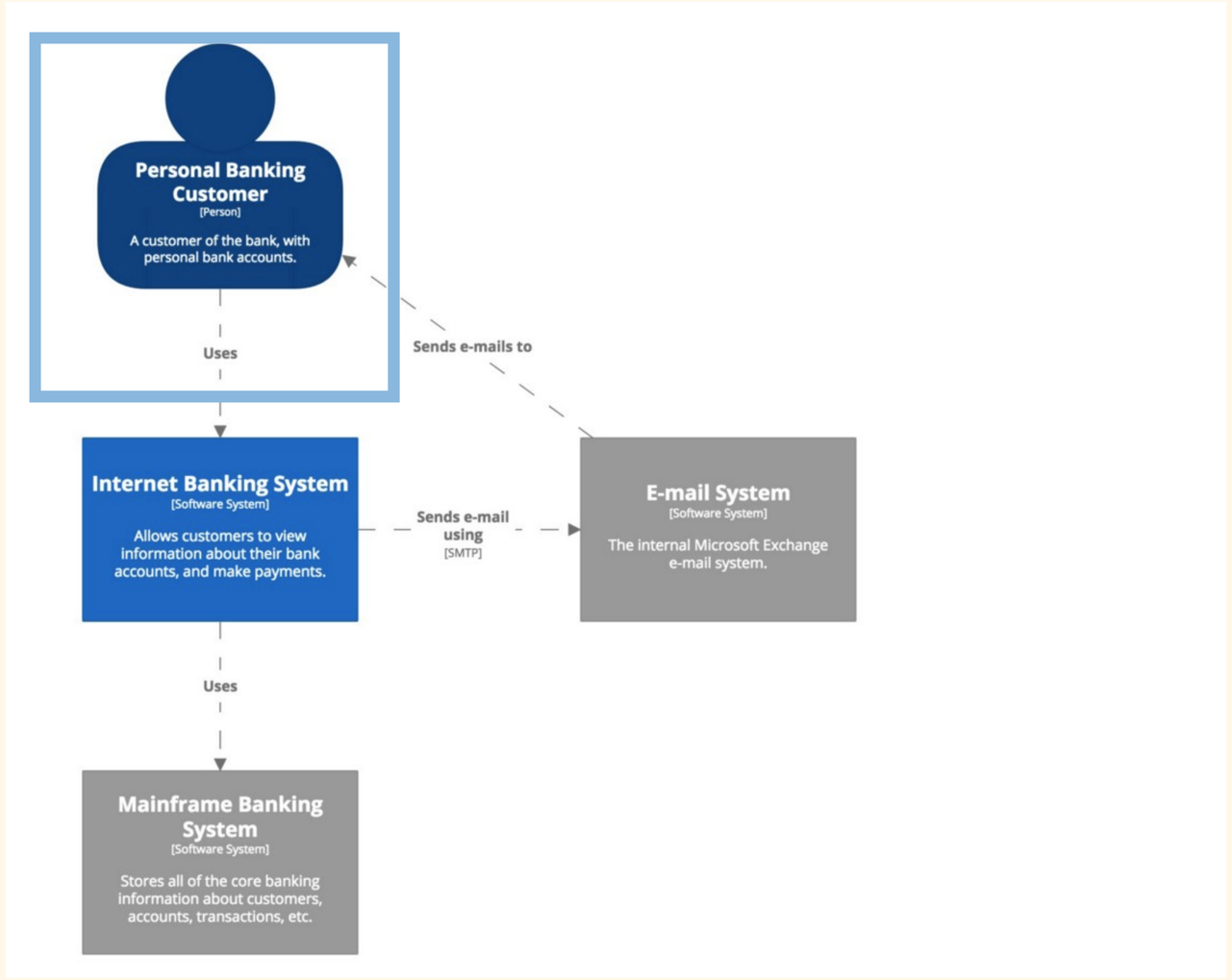
Componentes: componentes lógicos que interactuan en un contenedor

Código: Cómo se implementa cada componente

CONTEXTO

Cosas a preguntarse

¿Quiénes lo usarén?

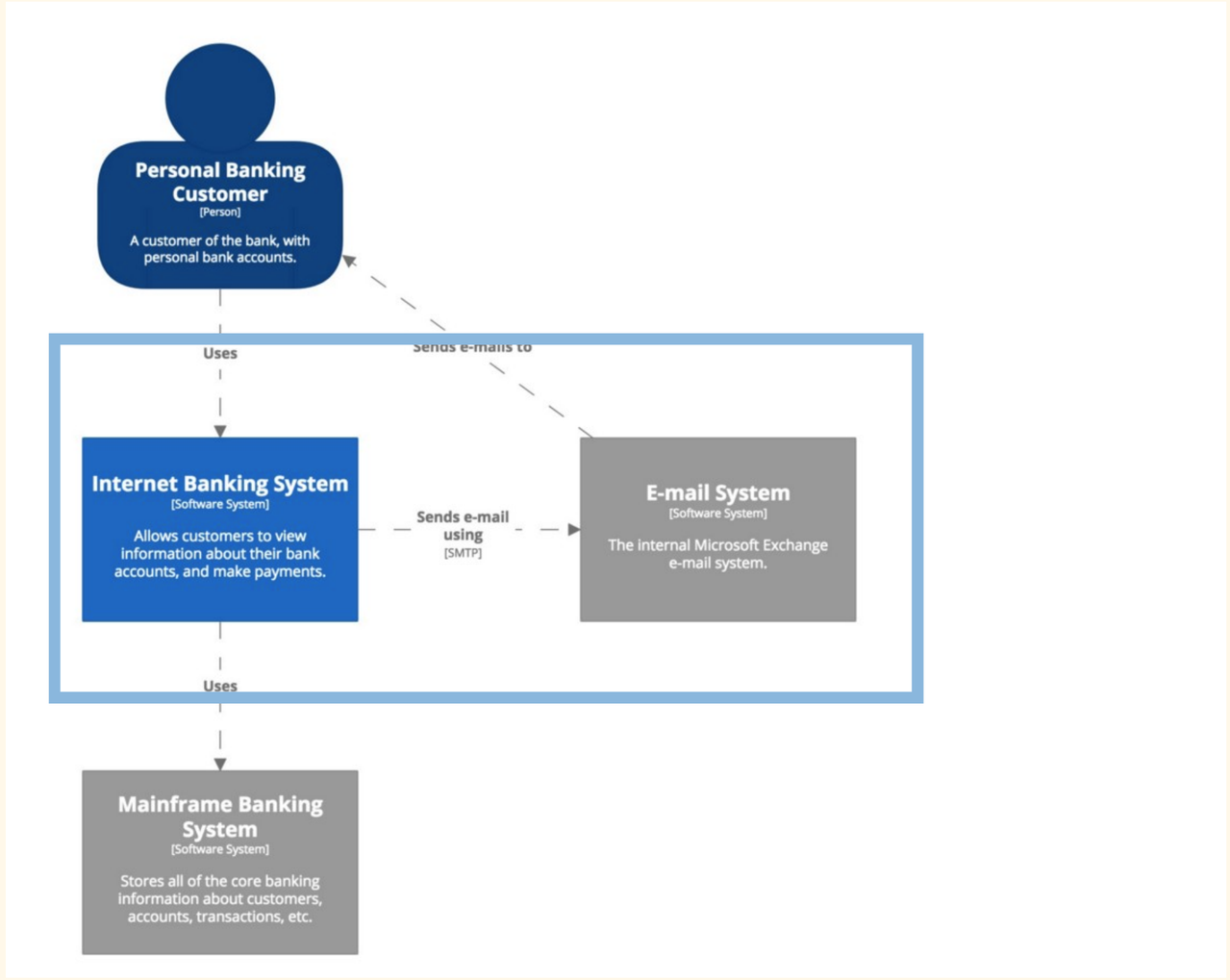


CONTEXTO

Cosas a preguntarse

¿Quiénes lo usarén?

¿Qué estamos construyendo?
¿Cuáles son nuestros
sistemas pares?



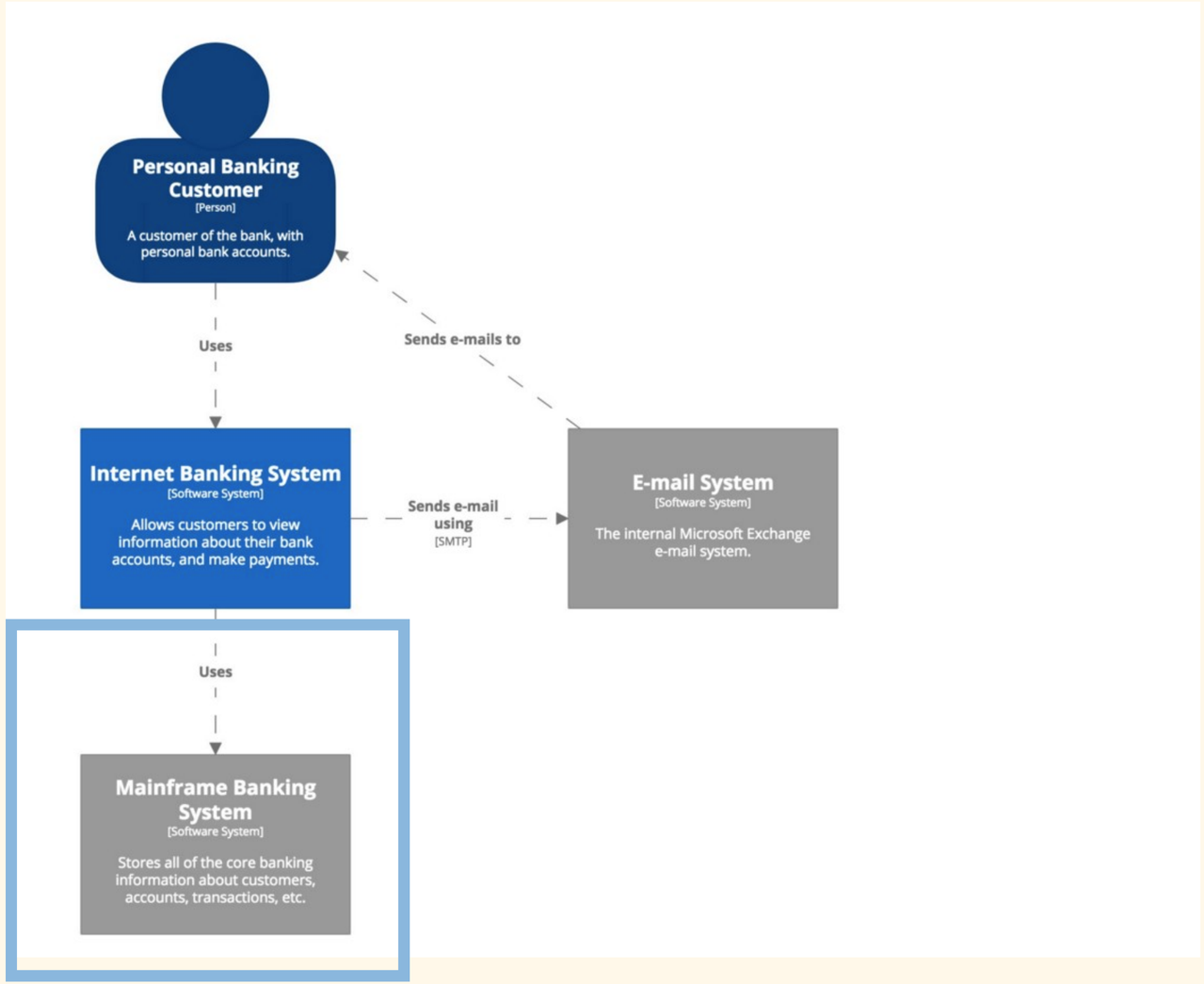
CONTEXTO

Cosas a preguntarse

¿Quiénes lo usarén?

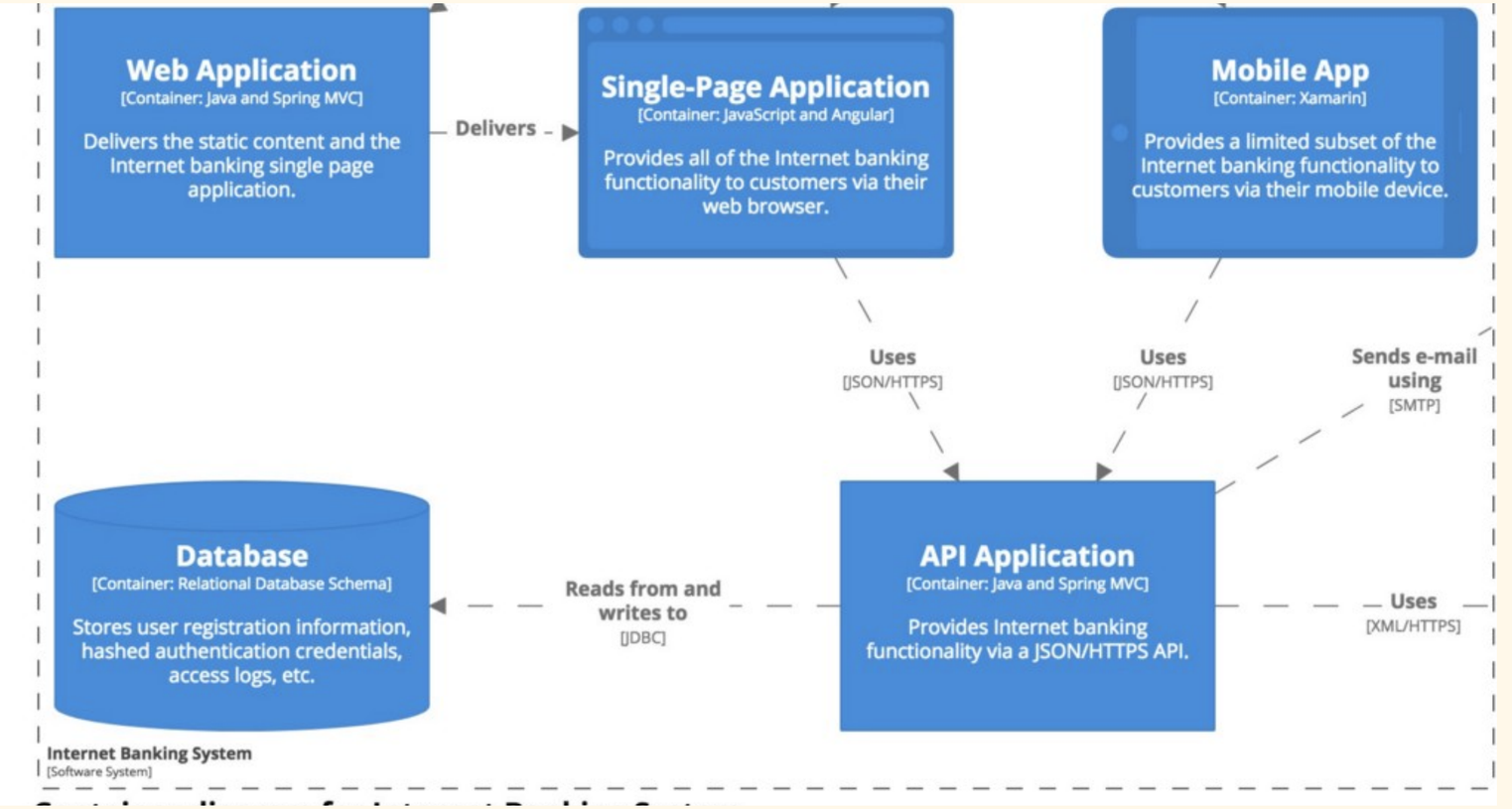
¿Qué estamos construyendo?
¿Cuáles son nuestros sistemas pares?

¿Cómo encaja en la arquitectura existente?



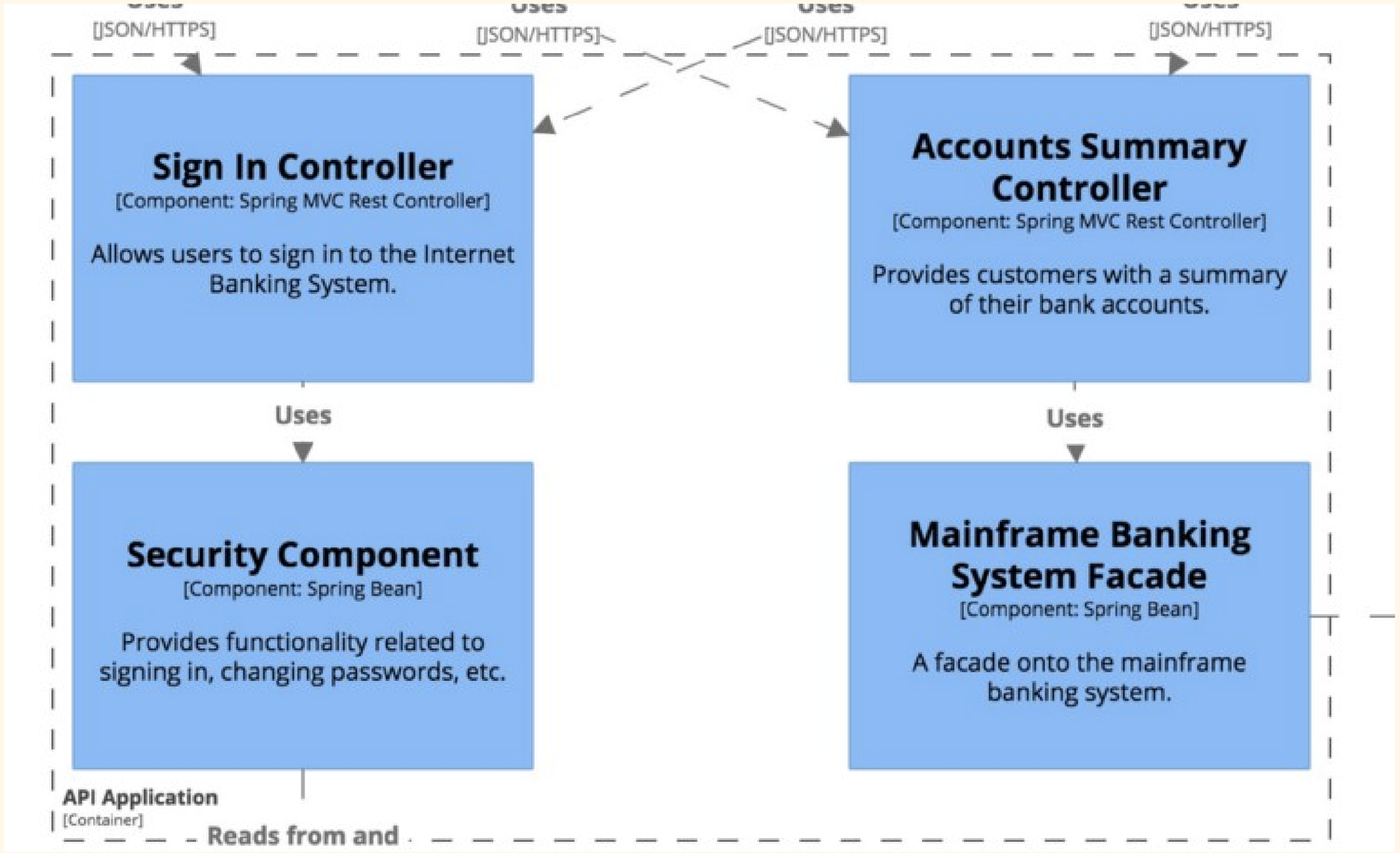
CONTENEDORES

En la arquitectura vienen a ser grandes subsistemas



CONTENEDORES

Cada componente del contenedor



PATRONES DE AQUITECTURA

Arquitectura de software

N-tier architecture

Arquitectura de capas

Service oriented architecture

Microservicios

N-TIER ARCHITECTURE

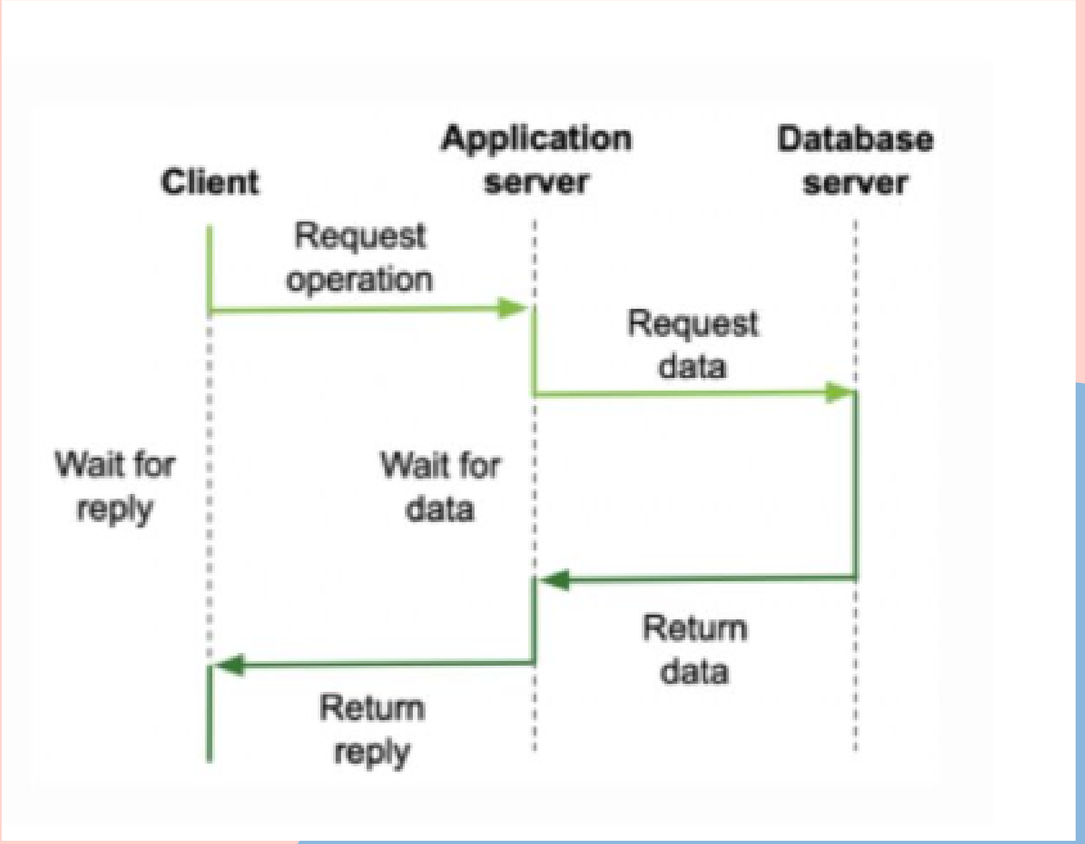
2 Tier

Cliente - Servidor

Servidor recibe y procesa solicitudes de clientes

3 Tier

Client Application Server



ARQUITECTURA DE CAPAS

Una capa es una colección lógica de componentes

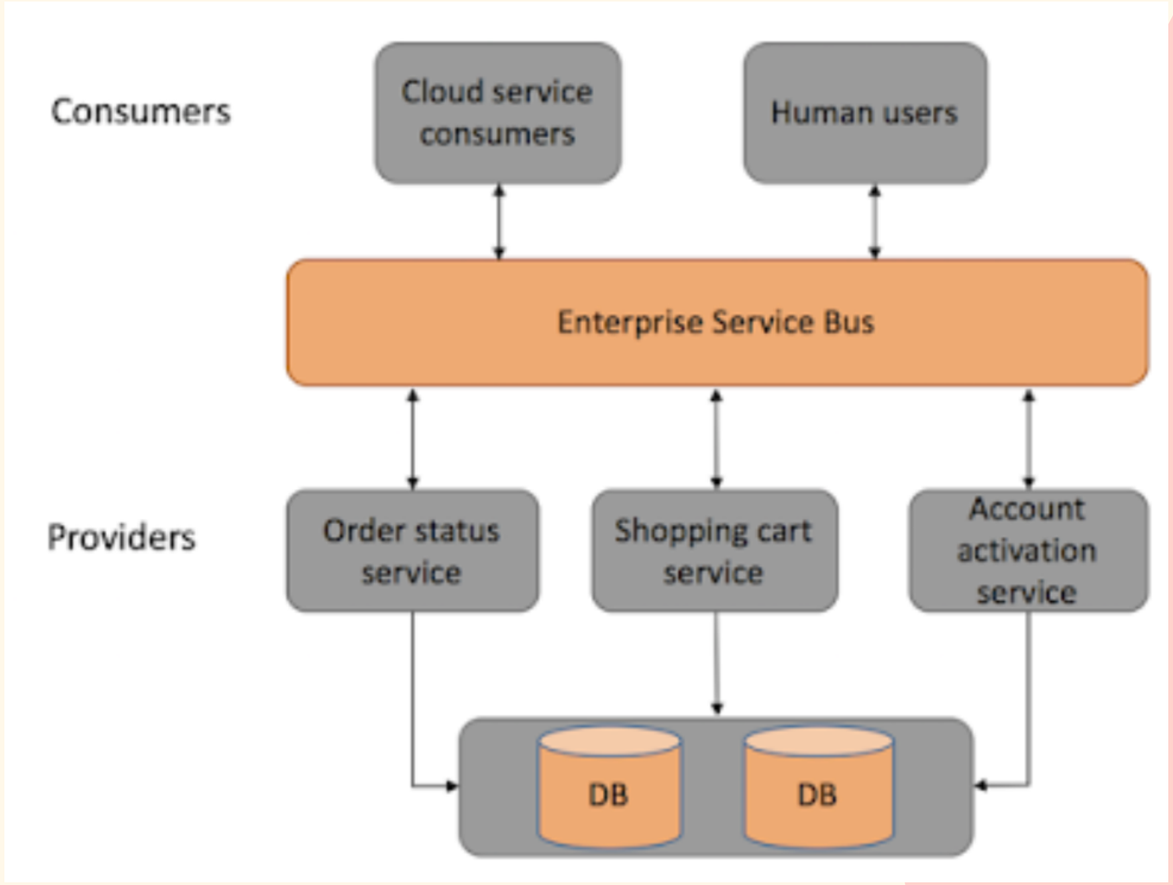
Usa y es usada por a lo más una capa

Permite que haya desacoplamiento, no hay dependencia entre niveles distintos

ARQUITECTURA ORIENTADA A SERVICIOS

La SOA, o arquitectura orientada a servicios, define una manera de hacer que los componentes de software sean reutilizables a través de interfaces de servicio. Estas interfaces utilizan estándares de comunicación comunes entre sí, de tal manera que pueden incorporarse rápidamente a nuevas aplicaciones sin tener que realizar una integración profunda cada vez.

ARQUITECTURA ORIENTADA A SERVICIOS



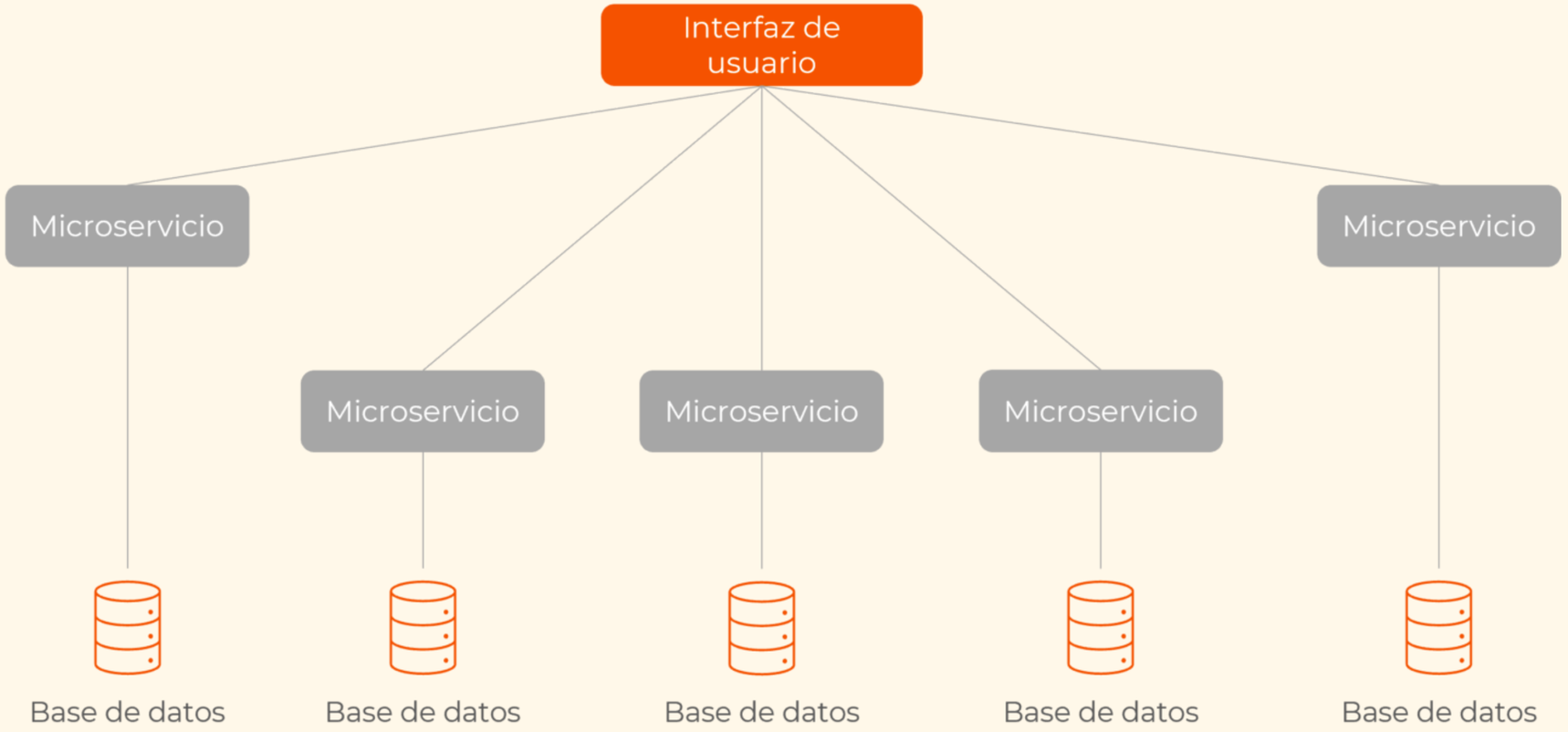
ARQUITECTURA DE MICROSERVICIOS

Método de desarrollo de software que funciona como un conjunto de pequeños servicios que se ejecutan de manera independiente, lo que proporciona una funcionalidad completa. Los microservicios se comunican entre sí a través de APIs, y cuentan con sistemas de almacenamiento propios, lo que evita la sobrecarga y caída de la aplicación.

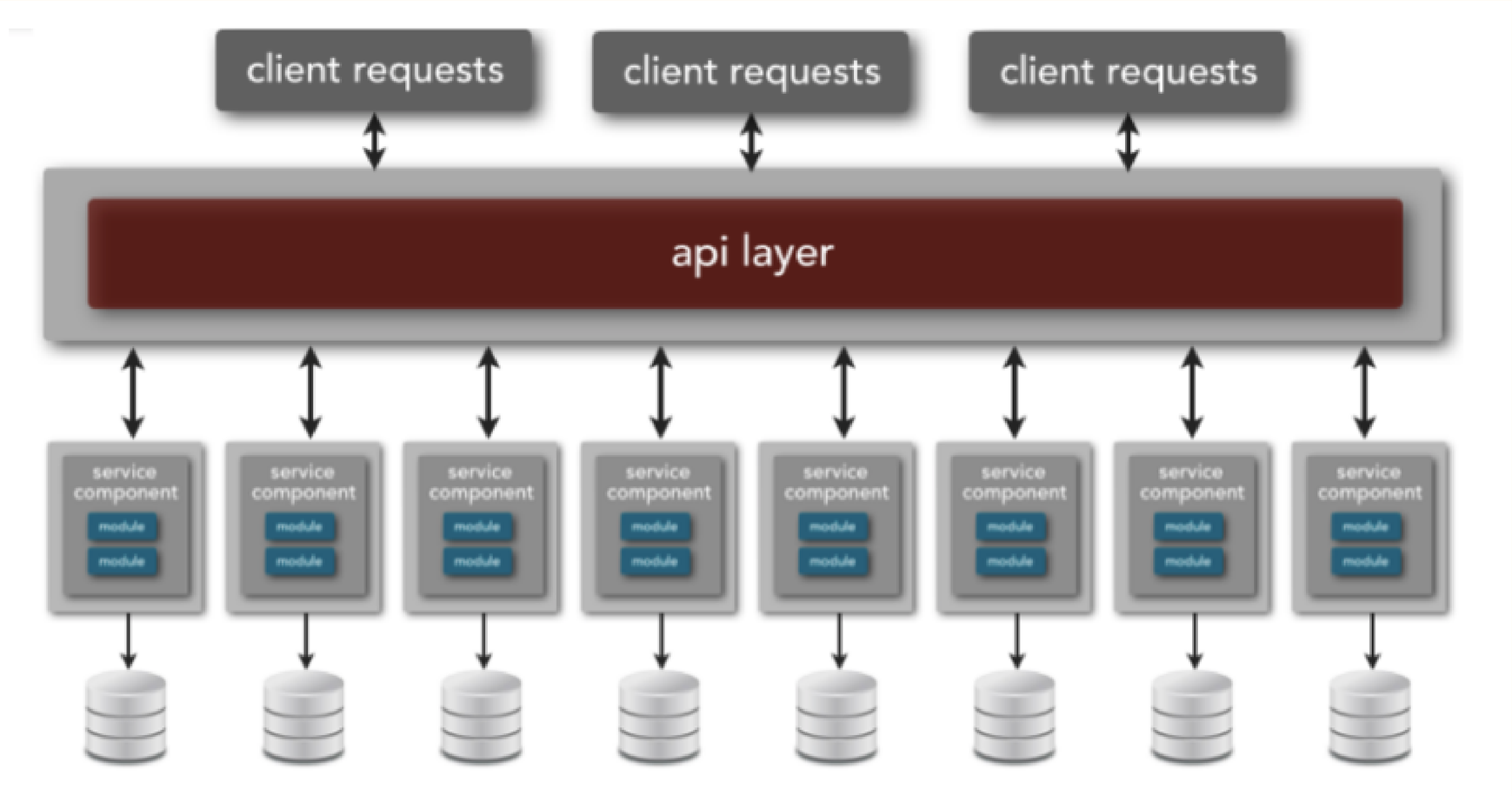
ARQUITECTURA DE MICROSERVICIOS

Arquitectura monolítica tradicional

Arquitectura de microservicios



ARQUITECTURA DE MICROSERVICIOS



ARQUITECTURA DE MICROSERVICIOS

Pros	Contras
Desarrollo independiente (proyectos pequeños)	Puede haber pérdidas de desempeño general
Escalabilidad es más sencilla	Es más difícil aislar y entender
Descentralización de los datos	El versionamiento es complejo
Flexibilidad tecnológica	Mayor esfuerzo de desarrollo
	Difícil asegurar transacciones o consistencia

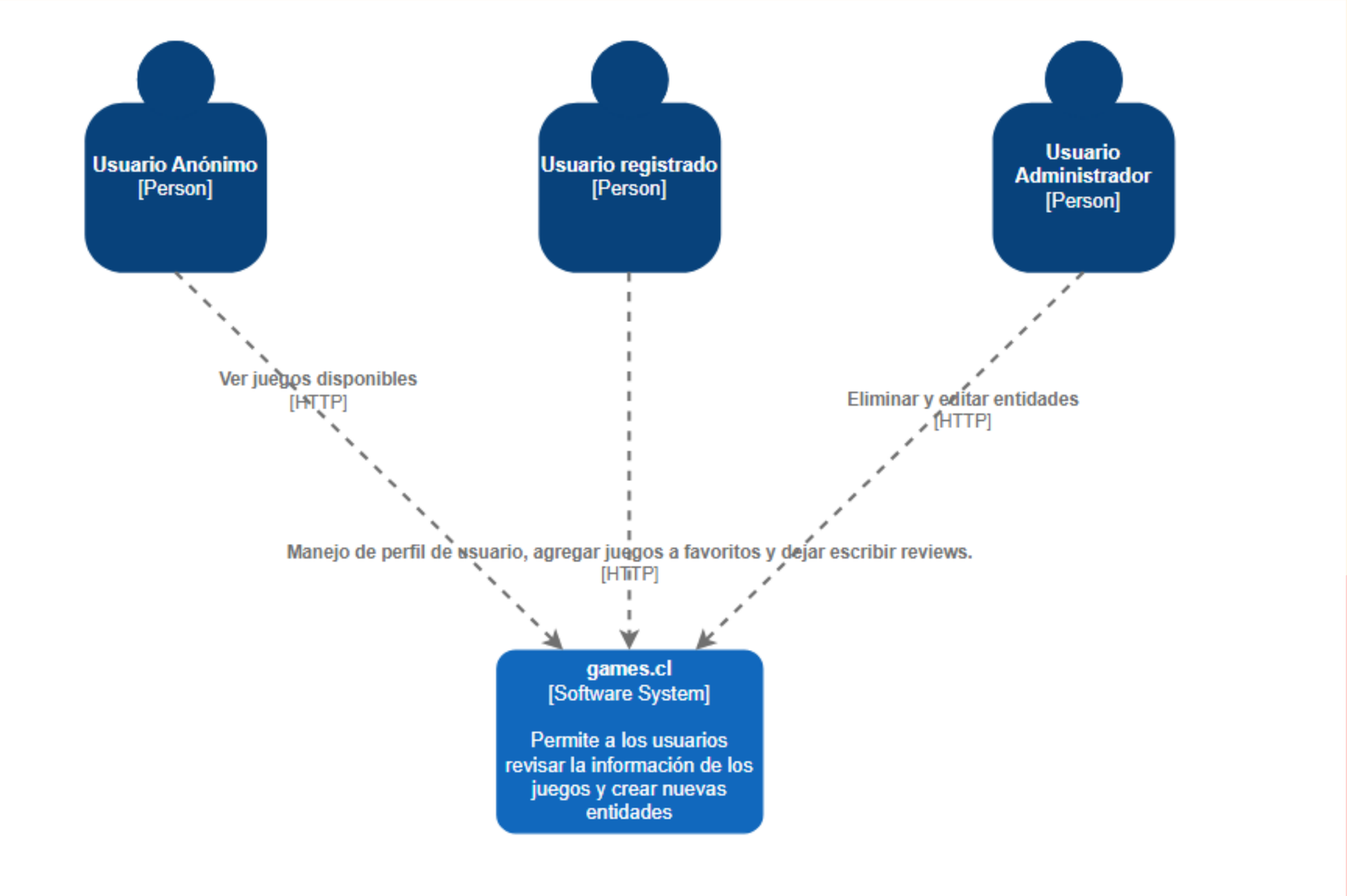
ARQUITECTURA DE MICROSERVICIOS DESAFIOS

- necesidad de mantener y monitorear cientos de microservicios
- problema de descubrimiento, se debe manejar un inventario y proveer formas de consultar
- como asegurar SLAs cuando hay servicios que dependen de otros
- no es suficiente asegurar correctitud de los servicios
- muchas opciones de escalamiento (flexible pero mas complejo)
- decisiones de upgrading mas complejas
- asegurar seguridad es mas complejo

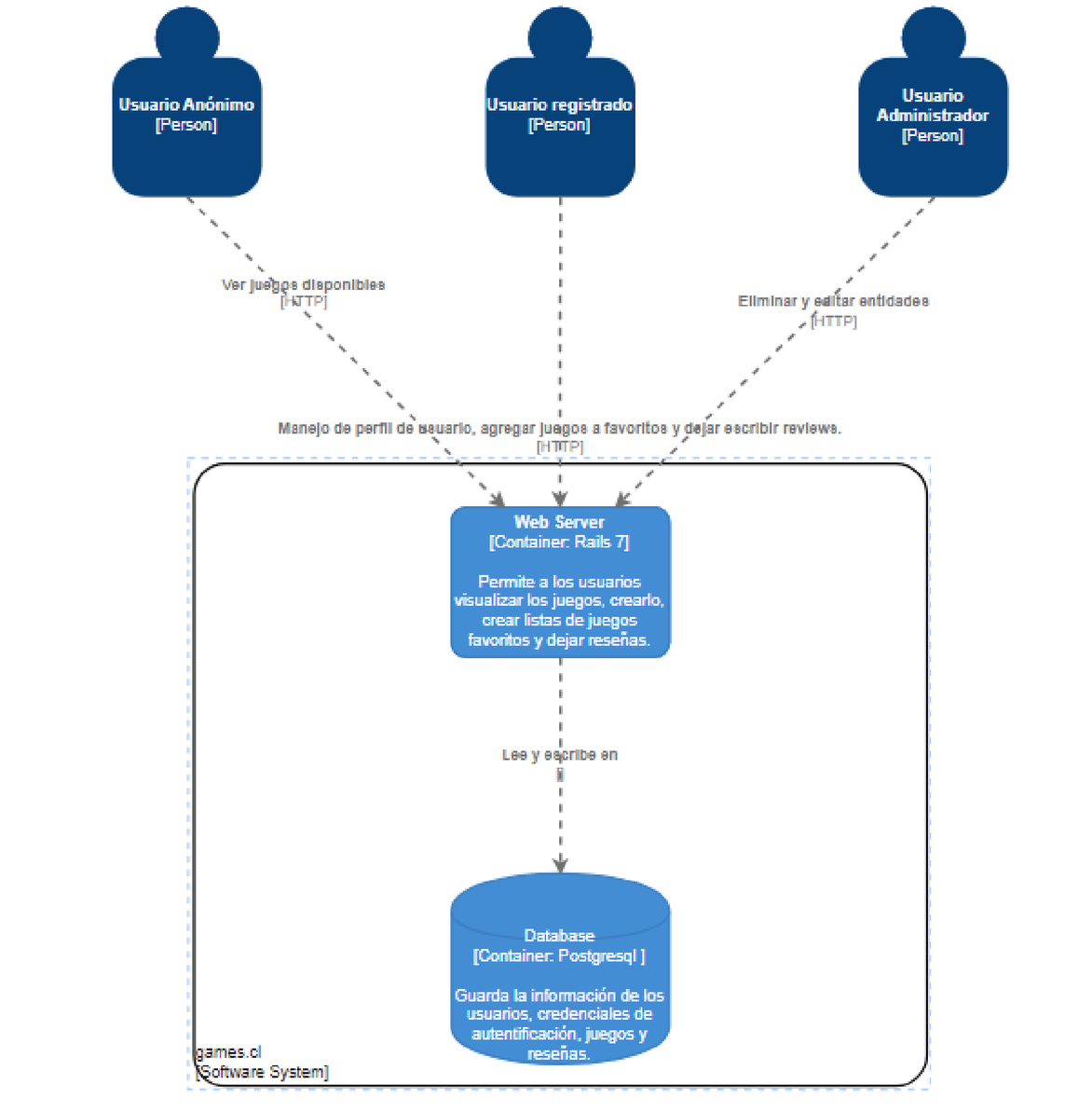
EJERCICIOS

REVISEMOS EL PROYECTO DE EJEMPLO

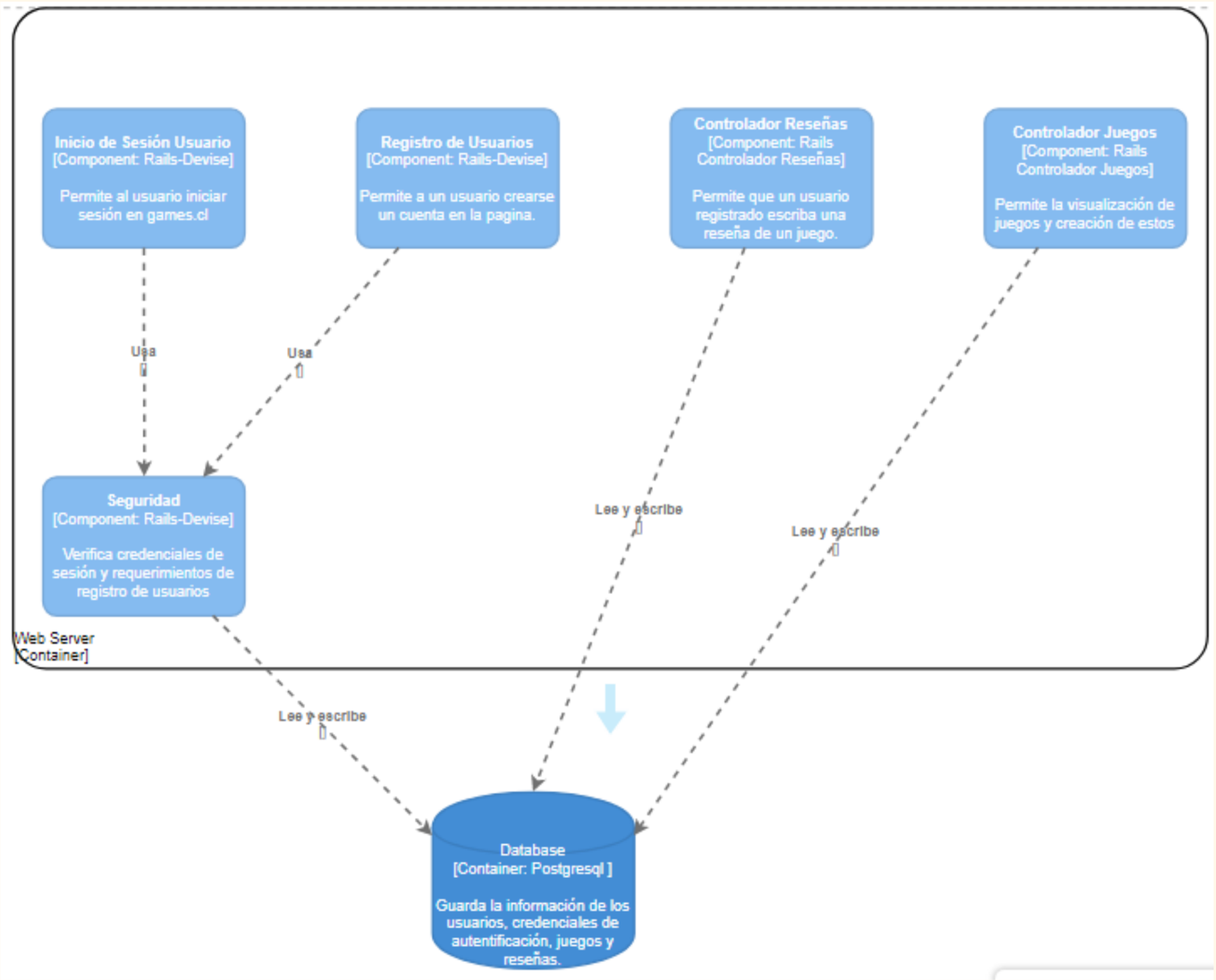
PROYECTO EJEMPLO



PROYECTO EJEMPLO



PROYECTO EJEMPLO



EJERCICIOS TIPO IES

La empresa *ComoNuevos* corresponde a una cadena de venta de automóviles usados que mantiene muchos lugares de venta de autos en las distintas ciudades de Chile. Cada uno de estos *dealer*s locales funciona en forma mas o menos independiente y maneja su propio sistema que llamaremos *SistemaLocal* que es operado por los vendedores de cada tienda de ventas de autos, pero si es necesario también pueden consultar sobre autos disponibles en otros lugares.

A nivel de gestión de *ComoNuevos* se requiere construir un sistema que mantenga la información de la totalidad de autos disponibles para la venta en las distintas ciudades del país, que pueda relacionarse con los sistemas locales de los *dealers* y también poder conectarse con el Registro Civil donde se mantiene la información de la vida de cada auto (esto es para comprobar la propiedad, si hay deudas, etc.)

El sistema de *ComoNuevos* se visualiza con un *front-end* orientado a interactuar con el administrador, y un *backend* que se encarga de manejar la información de los autos usados disponibles. El administrador del sistema, aparte de poder acceder a la información de los autos, es responsable de manejar todos los datos de configuración que se almacenan en el sistema. El *backend* está además encargado de interactuar con el Registro Civil. Los distintos sistemas locales se conectarán también con el *backend* de *ComoNuevos* ya sea para extraer nueva información o para actualizar información cuando, por ejemplo, se vende un automóvil.

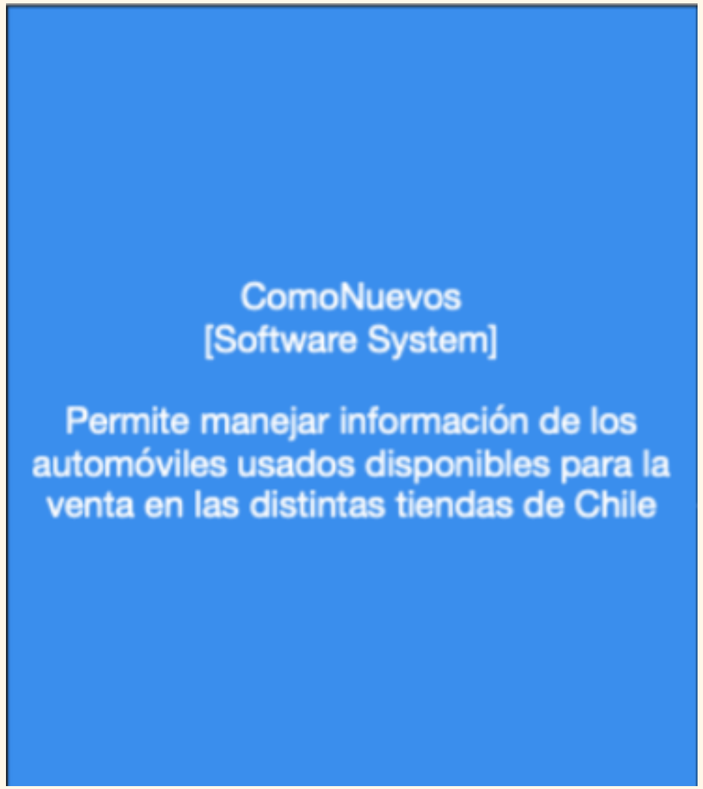
EJERCICIOS TIPO IES

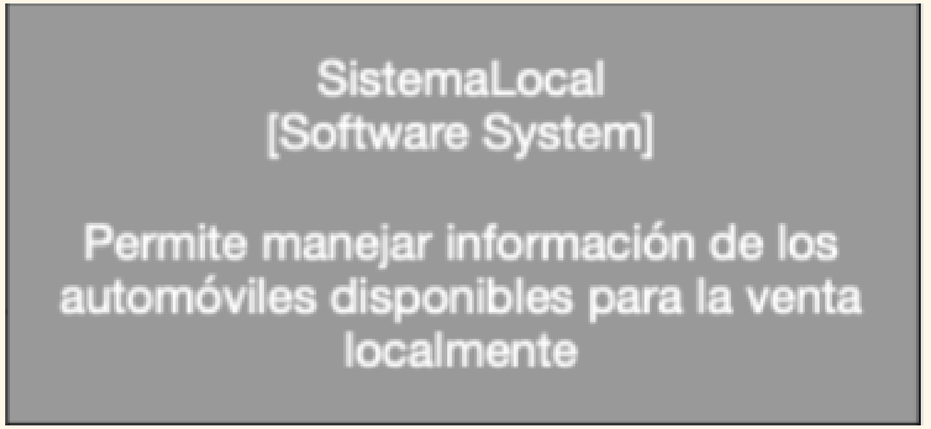
a) (14 puntos) Dibuja un diagrama de contexto para la arquitectura del sistema *ComoNuevos*. Identifica claramente usuarios y subsistemas externos.

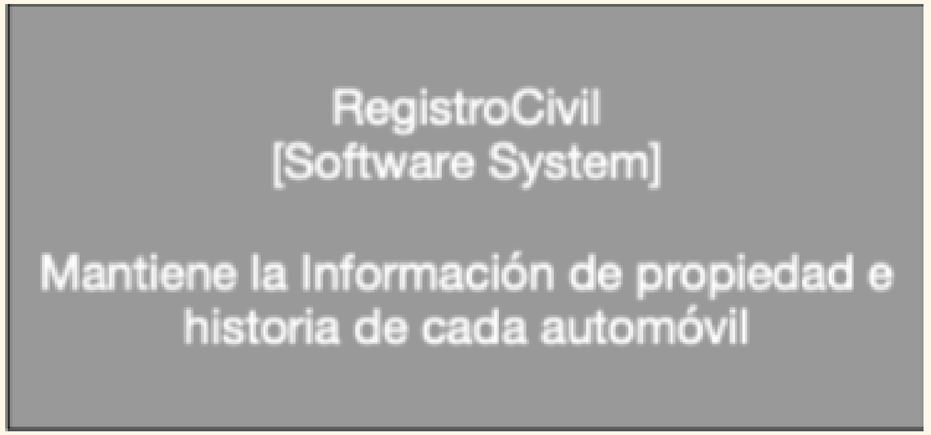
EJERCICIOS TIPO IES

a) (14 puntos) Dibuja un diagrama de contexto para la arquitectura del sistema *ComoNuevos*. Identifica claramente usuarios y subsistemas externos.

Sugerencia: Partir por distinguir los sistemás y subsistemas



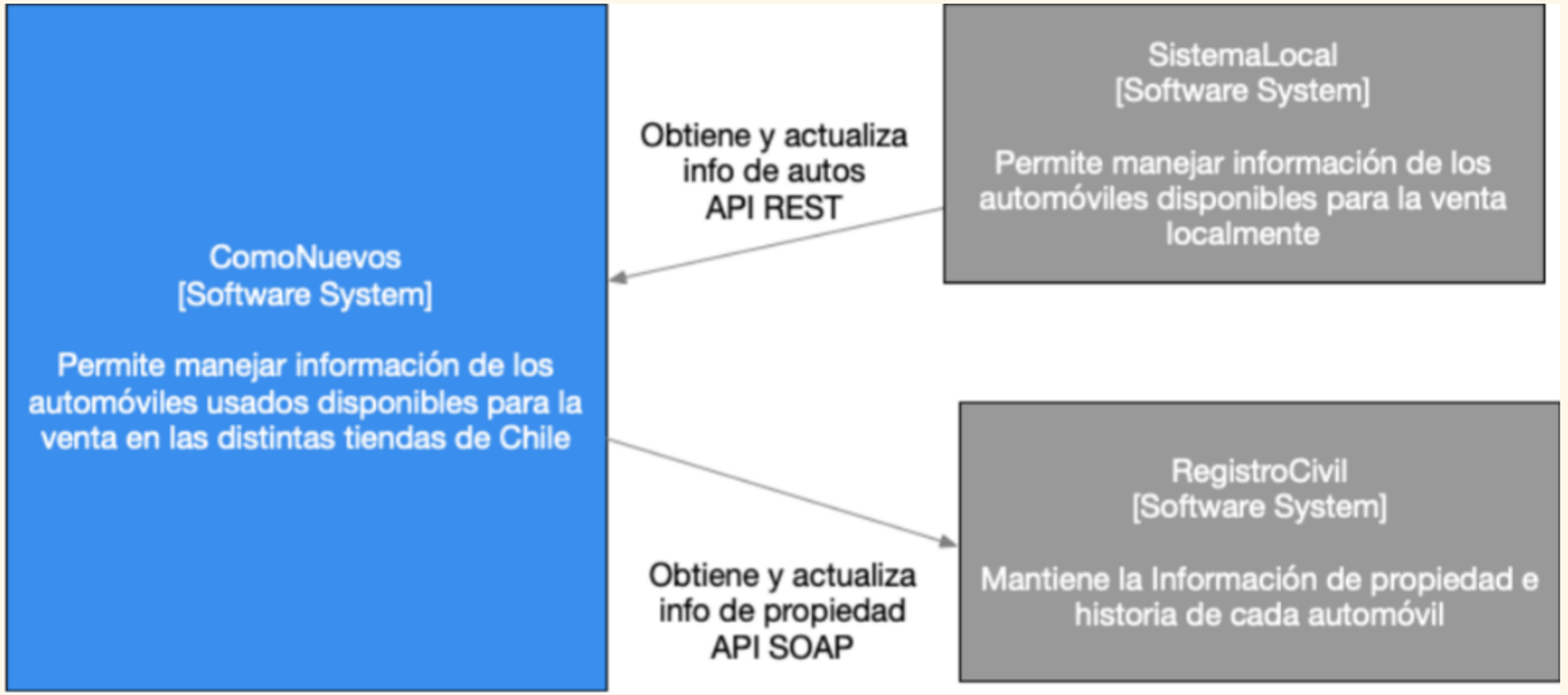




EJERCICIOS TIPO IES

a) (14 puntos) Dibuja un diagrama de contexto para la arquitectura del sistema *ComoNuevos*. Identifica claramente usuarios y subsistemas externos.

¿Cómo se conectan?



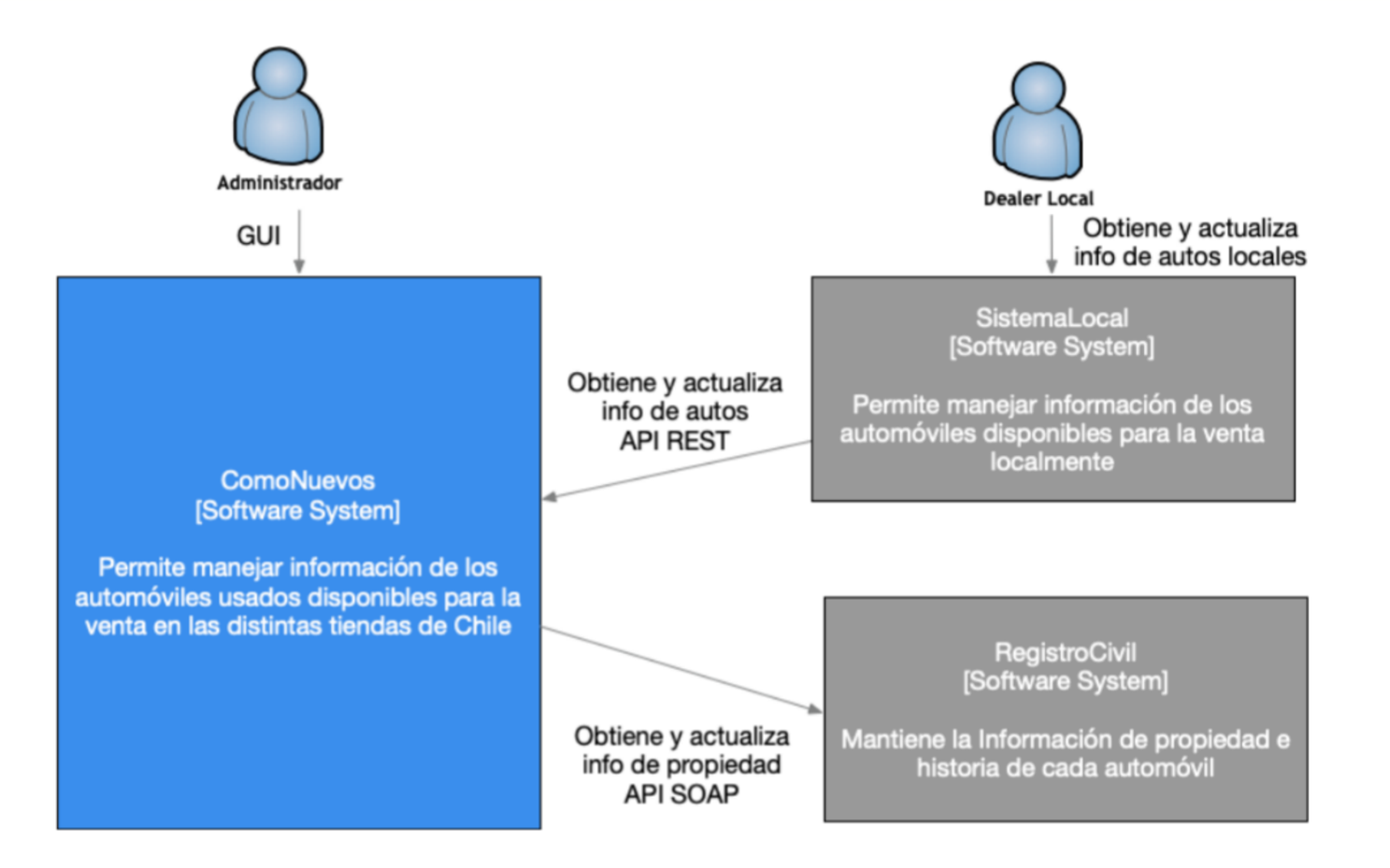
EJERCICIOS TIPO IES

a) (14 puntos) Dibuja un diagrama de contexto para la arquitectura del sistema *ComoNuevos*. Identifica claramente usuarios y subsistemas externos.

¿Cuáles son nuestro usuarios?

EJERCICIOS TIPO IES

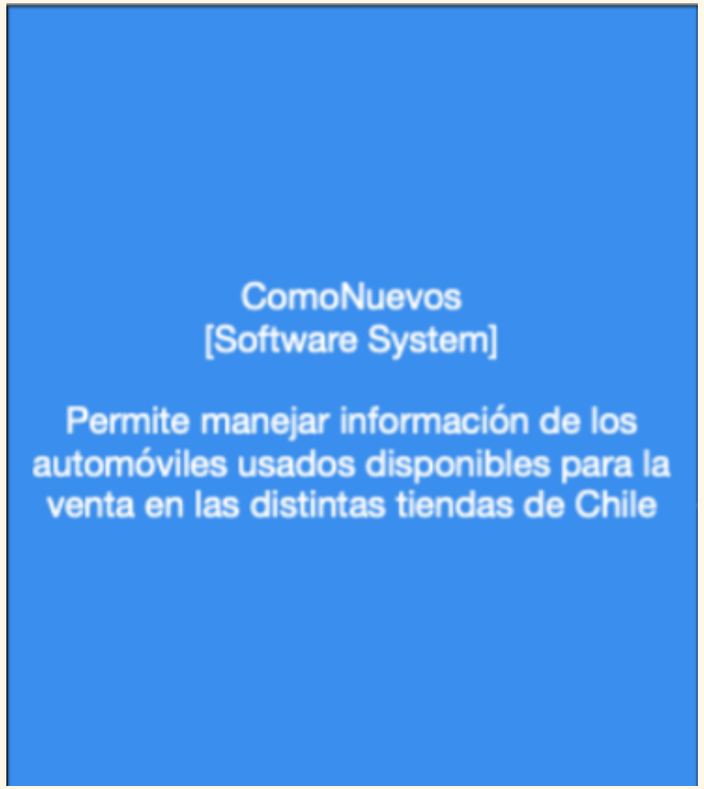
a) (14 puntos) Dibuja un diagrama de contexto para la arquitectura del sistema *ComoNuevos*. Identifica claramente usuarios y subsistemas externos.



EJERCICIOS TIPO IES

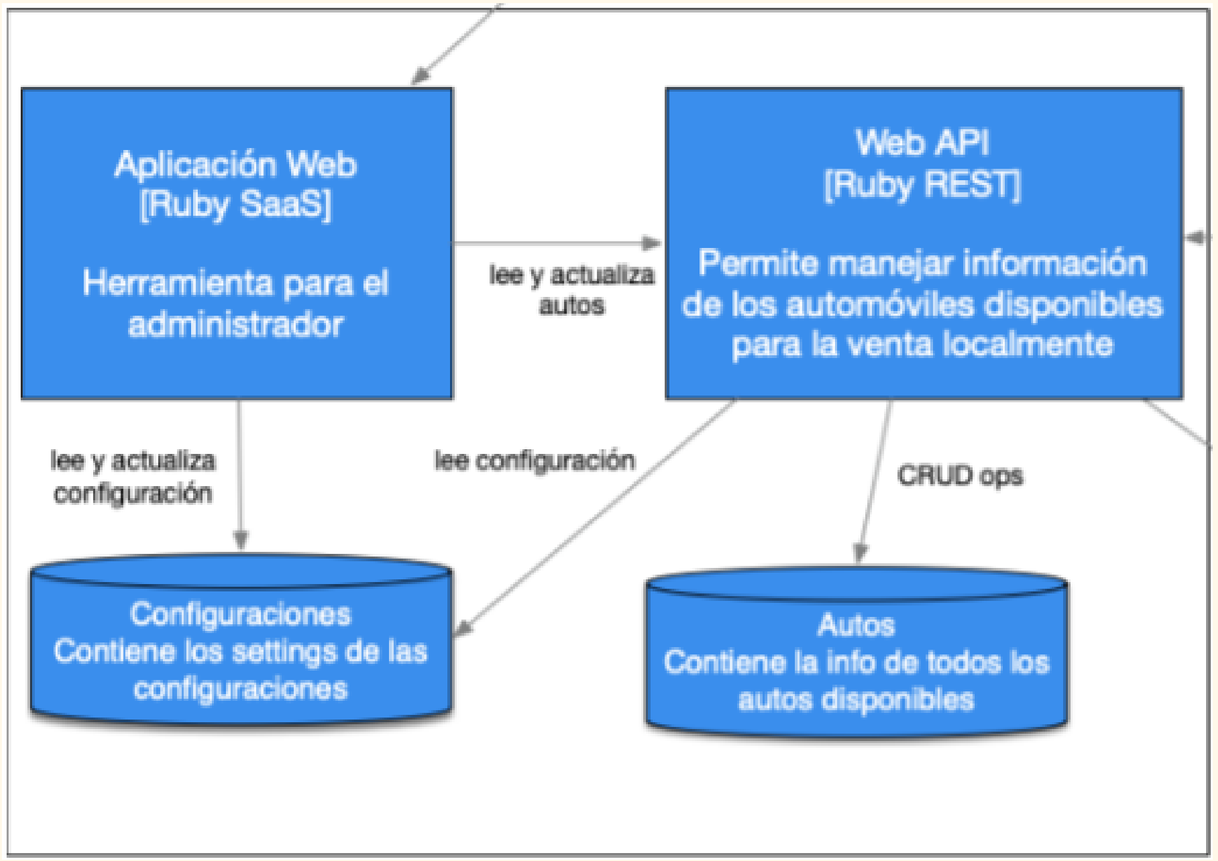
b) (18 puntos) Dibuja un diagrama a nivel de contenedores incluyendo los principales contenedores y a que corresponden los principales intercambios de información entre ellos

¿Qué contenedores tenemos?



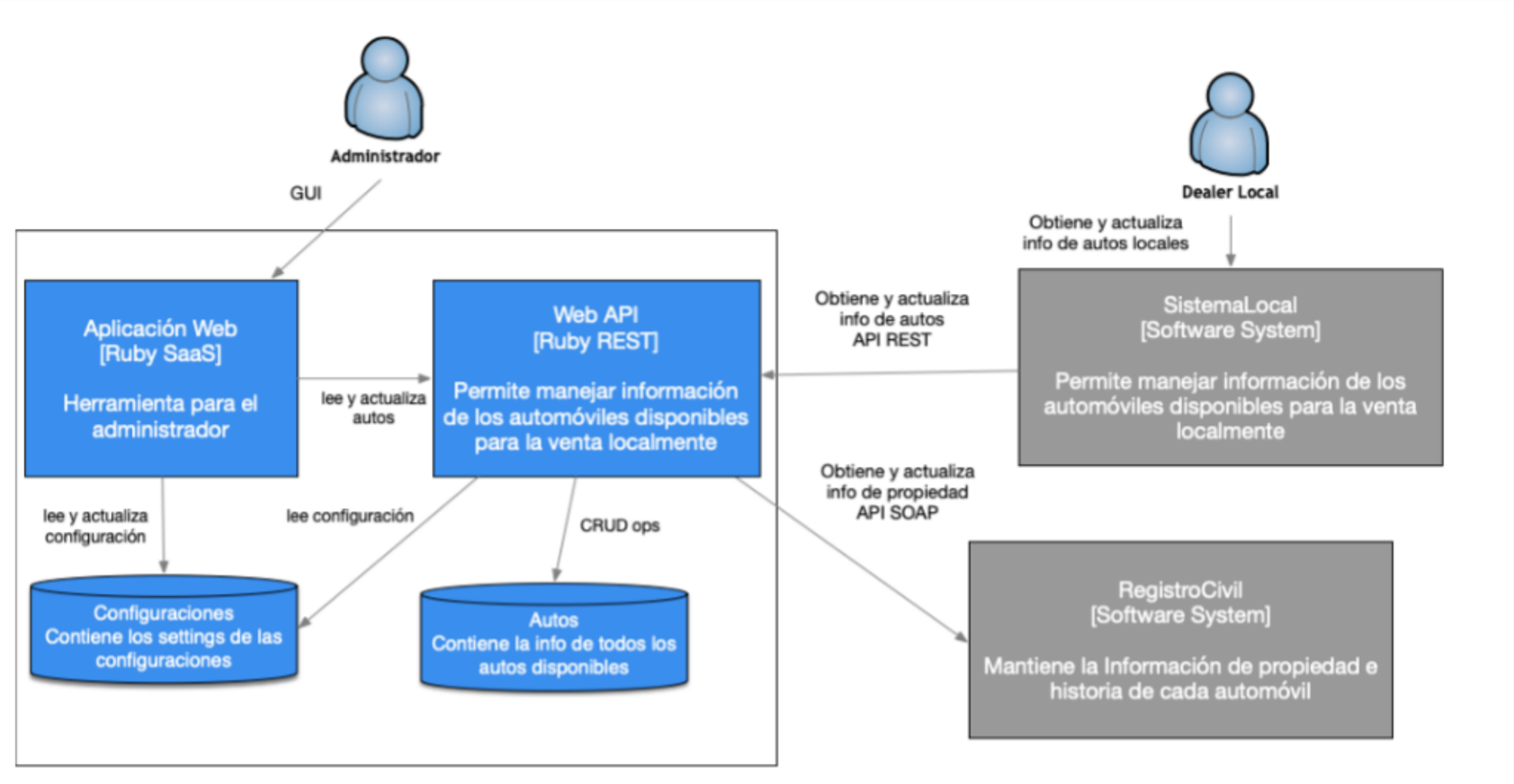
EJERCICIOS TIPO IES

b) (18 puntos) Dibuja un diagrama a nivel de contenedores incluyendo los principales contenedores y a que corresponden los principales intercambios de información entre ellos



EJERCICIOS TIPO IES

b) (18 puntos) Dibuja un diagrama a nivel de contenedores incluyendo los principales contenedores y a que corresponden los principales intercambios de información entre ellos



TEORICAS

1) LA PRINCIPAL DIFERENCIA ENTRE LA ARQUITECTURA DE SERVICIOS CONOCIDA COMO SOA CON LA DE MICROSERVICIOS ES EL TAMAÑO DE LOS SERVICIOS

2)LA ARQUITECTURA DE MICROSERVICIOS TIENDE A PRODUCIR SOLUCIONES DE MEJOR DESEMPEÑO QUE LAS MONOLÍTICAS

3) ES POSIBLE TENER UNA ARQUITECTURA DE VARIOS TIERS QUE CONTIENEN LAYERS Y TAMBIEN UNA DE VARIOS LAYERS QUE CONTIENEN TIERS

TEORICAS

1) LA PRINCIPAL DIFERENCIA ENTRE LA ARQUITECTURA DE SERVICIOS CONOCIDA COMO SOA CON LA DE MICROSERVICIOS ES EL TAMAÑO DE LOS SERVICIOS

R: LA PRINCIPAL DIFERENCIA ES QUE NO SE REQUIERE DE UN MIDDLEWARE QUE ACTÚE COMO UN BUS DE SERVICIOS

2)LA ARQUITECTURA DE MICROSERVICIOS TIENDE A PRODUCIR SOLUCIONES DE MEJOR DESEMPEÑO QUE LAS MONOLÍTICAS

R: POR LO GENERAL NO ES ASÍ PORQUE HAY MUCHO COSTO DE COMUNICACIÓN (HTTP) ENTRE ESTAS COMPONENTES (SERVICIOS) DÉBILMENTE ACOPLADAS

3) ES POSIBLE TENER UNA ARQUITECTURA DE VARIOS TIERS QUE CONTIENEN LAYERS Y TAMBIEN UNA DE VARIOS LAYERS QUE CONTIENEN TIERS

R: QUE UN TIER SE IMPLEMENTE EN LAYER TIENE SENTIDO, EL INVERSO NO LO TIENE