

PROPUESTA MICROSERVICIOS

PROYECTOS INTERNOS

ATENEA, QUALITY TOOLS, OTROS

VERSIÓN 1.0

07/08/2024

Tabla de contenido

Arquitectura de microservicios	1
Ventajas	1
Desventajas	2
Escenario actual	3
Propuesta de micro servicios.	5
Sugerencia de lenguaje para Servicios.....	6
Propuesta alternativa: API rest	7
Análisis para recursos de servidor.	8
Resumen	10
Análisis de Recursos y Carga Actual	10
Consideraciones para Arquitectura de Microservicios	10
Análisis de Viabilidad.....	11

Arquitectura de microservicios

La arquitectura de microservicios es un enfoque de diseño que se ha vuelto muy popular, especialmente en aplicaciones empresariales complejas. A diferencia de las arquitecturas monolíticas tradicionales, donde todas las funcionalidades de una aplicación están empaquetadas en un único sistema, los microservicios dividen una aplicación en un conjunto de servicios pequeños e independientes. Cada uno de estos servicios está diseñado para realizar una función específica y se comunica con otros servicios a través de interfaces bien definidas, generalmente API RESTful.

Ventajas

- Escalabilidad Independiente:

Cada microservicio puede escalarse de manera independiente según sus necesidades de carga. Esto permite un uso más eficiente de los recursos y una mejor capacidad de respuesta a las demandas fluctuantes.

- Despliegue Rápido y Autónomo:

Los equipos de desarrollo pueden implementar cambios en un servicio sin afectar a los demás, lo que reduce el tiempo de entrega de nuevas funcionalidades y correcciones.

- Mejor Mantenimiento y Comprensibilidad:

Al estar divididos en servicios pequeños, el código es más fácil de mantener y entender. Esto simplifica el proceso de actualización y depuración.

- Flexibilidad Tecnológica:

Cada microservicio puede ser desarrollado usando diferentes tecnologías y lenguajes de programación que sean más adecuados para sus requisitos específicos, permitiendo a los equipos elegir las mejores herramientas para cada tarea.

- Resiliencia Mejorada:

La falla de un microservicio no necesariamente compromete el funcionamiento de toda la aplicación, lo que mejora la resistencia general del sistema.

Desventajas

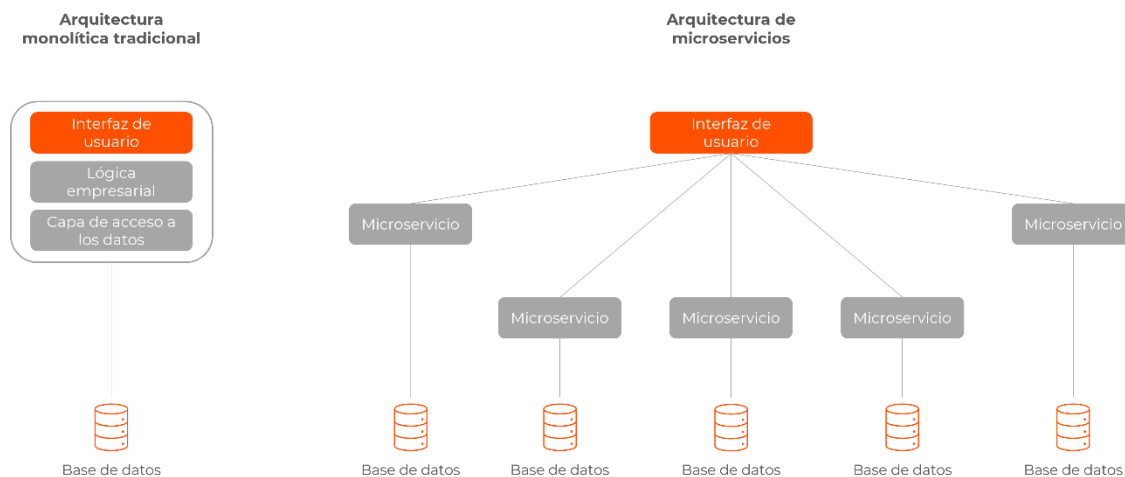
- Tiempo de desarrollo:

Al ser un desarrollo nuevo siempre toma su respectivo tiempo, sin embargo, en este caso puntual al tener ya sistemas monolitos se debe tomar mayor tiempo para analizar y desarrollar la solución para no afectar las funciones actuales de los sistemas monolitos, pero que de apoco se puedan ir migrando.

- Posibles costos en servidores

Al no tener una sola app, si no que múltiples mas pequeñas cabe la posibilidad de alojar las apps en diferentes servidores y eso causar un costo extra, sin embargo, con una buena configuración y pruebas adecuadas podrían funcionar en un servidor en conjunto.

Ejemplo



Escenario actual

Actualmente se poseen los siguientes sistemas: Atenea, Quality Tools, Bodeguita, RRHH,

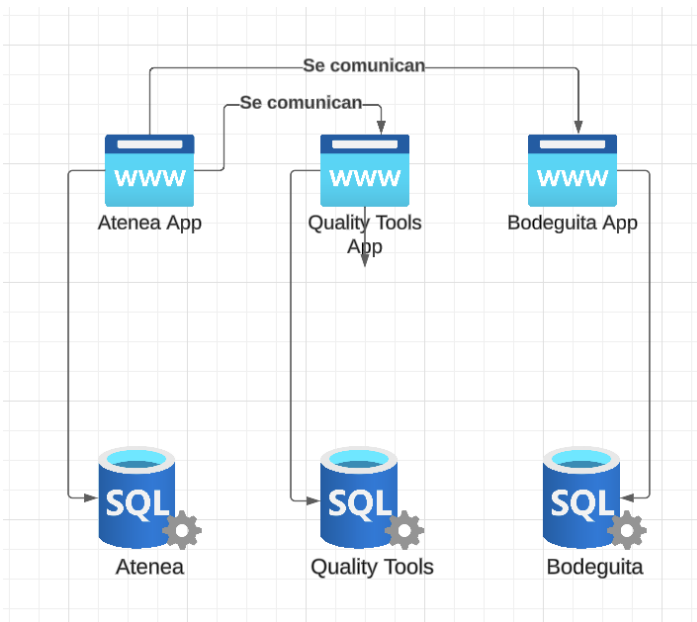
Los cuales cada uno de ellos posee su respectiva base de datos, sin embargo, hay información que se comparten los diferentes sistemas, haciendo más énfasis entre Atenea y Quality Tools que comparten información como lo son clientes, fabricantes, proyecto, oportunidades entre otras.

Para poder comunicarse entre sí hay diferentes técnicas las cuales serán listadas:

Servicios REST: Se realizan servicios en los mismos proyectos, dependiendo la necesidad, esto conlleva a modificar código en el proyecto correspondiente y esperar a que pueda salir a producción o a QA junto con las demás funcionalidades ajenas a los servicios realizados para poder ocuparlos.

Anclaje entre base de datos: En un procedimiento almacenado de una base de datos A, se manda a llamar a una base de datos B, y de tal forma podemos obtener la información de B en A y luego consumir ese procedimiento desde nuestra aplicación. Una limitante es que las dos bases de datos deben estar en el mismo servidor, si no se tendría que hacer un Link en SQL Server, y si aun así se resuelve con eso, en el caso de existir otra base de datos que no sea SQL server ya seria mas complicado hacer el link.

Diagrama actual



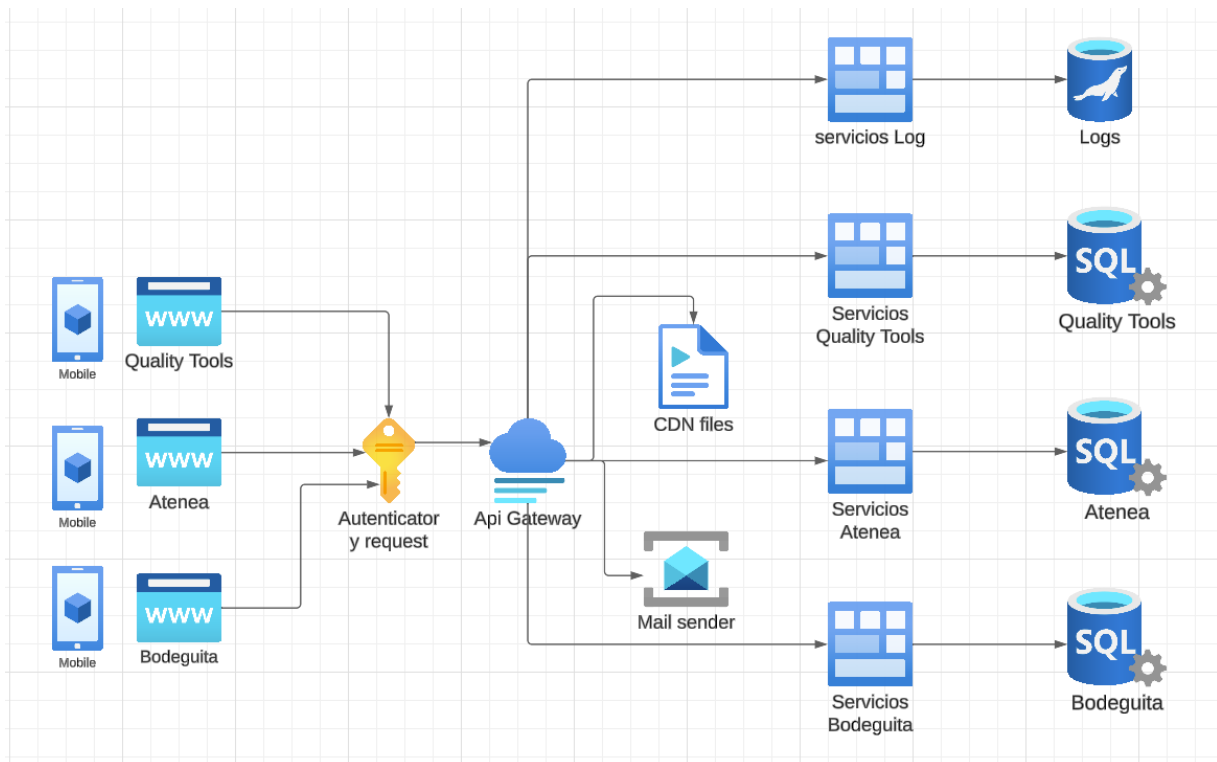
En el diagrama se observa que todo esta en un mismo servidor, dado que en algunas situaciones la información de las diferentes bases de datos se comunica, y se hace directo en la base anfitriona, las bases de datos no pueden ser movidas, es decir no podemos tener las bases en diferentes servidores, a no ser que implementemos una solución para ello.

Que a futuro podría suceder pues el crecimiento de las bases de datos va siendo exponencial y de pronto una base de datos llamada “atenea” crecerá tanto que deberá dividirse para poder darle la escalabilidad necesaria, o en caso que viene a corto plazo, tener mas aplicaciones con sus respectivas bases de datos.

Si bien actualmente la arquitectura monolítica sigue siendo soportada y no genera mucho problema entre la comunicación de los sistemas se comienzan a dar casos en los que viene siendo necesario una estructura un poco más sofisticada.

Se nos presentan dos nuevas aplicaciones las cuales serán **Finanzas y licitaciones**, las cuales ambas tendrán sus propias bases de datos porque aparte de ocupar información de otros sistemas, tendrán la propia suya. Eso nos da un total de 5 base de datos y que por escalabilidad para poder comunicarse entre si debemos implementar otro tipo de arquitectura pues en un futuro a mediano plazo, nos generar mayores problemas que serán mas complicados en tiempo poder darle la solución hasta el punto de colapsarlos.

Propuesta de micro servicios.



La propuesta sería implementar una especie de arquitectura de micro servicios donde cada base de datos de cada sistema tendría su respectivo clúster de servicios necesarios para que sea consumida por los demás clústeres gracias a la API Gateway que será la encargada de gestionar la comunicación entre todos. Por ahorita se ha dado la idea de tener clúster por base de datos sin embargo va quedar a criterio de la implementación si se realizan clústeres por modulo de cada base de datos.

A nivel de base de datos no cambiaría nada, si no en la manera de consultar, pues ahora utilizaríamos los respectivos ORM de los frameworks de los lenguajes que implementemos o a consultas SQL. Tratando la manera de remover el uso de los procedimientos almacenados pues en las bases de datos se están realizando muchos y es muy complicado comenzar a darles seguimiento a ya más de 50 procedimientos, sin embargo, la utilidad de los procedimientos puede mantenerse en casos especiales como consultas para reportes, procesos internos, procesos que desencadenen el llenado de muchas tablas, pero se intentaría remover el uso para procesos sencillos e intermedios como los GET, PUT, POST, DELETE.

En el diagrama se aprecian dos ítems los cuales son el sender de mail y el cdn de archivos, la idea de ellos es poder separar la lógica de envío e correos para los diferentes sistemas y embeberlos en un sitio, para poder administrar de una mejor manera los envíos.

Así mismo el manejo de archivos, el poder tener una especie de CDN o sitio donde estén alojados (el mismo servidor podría ser) pero que los demás sistemas puedan tener accesos a ellos con sus respectivos permisos, los cual ya sucede con Quality y Atenea.

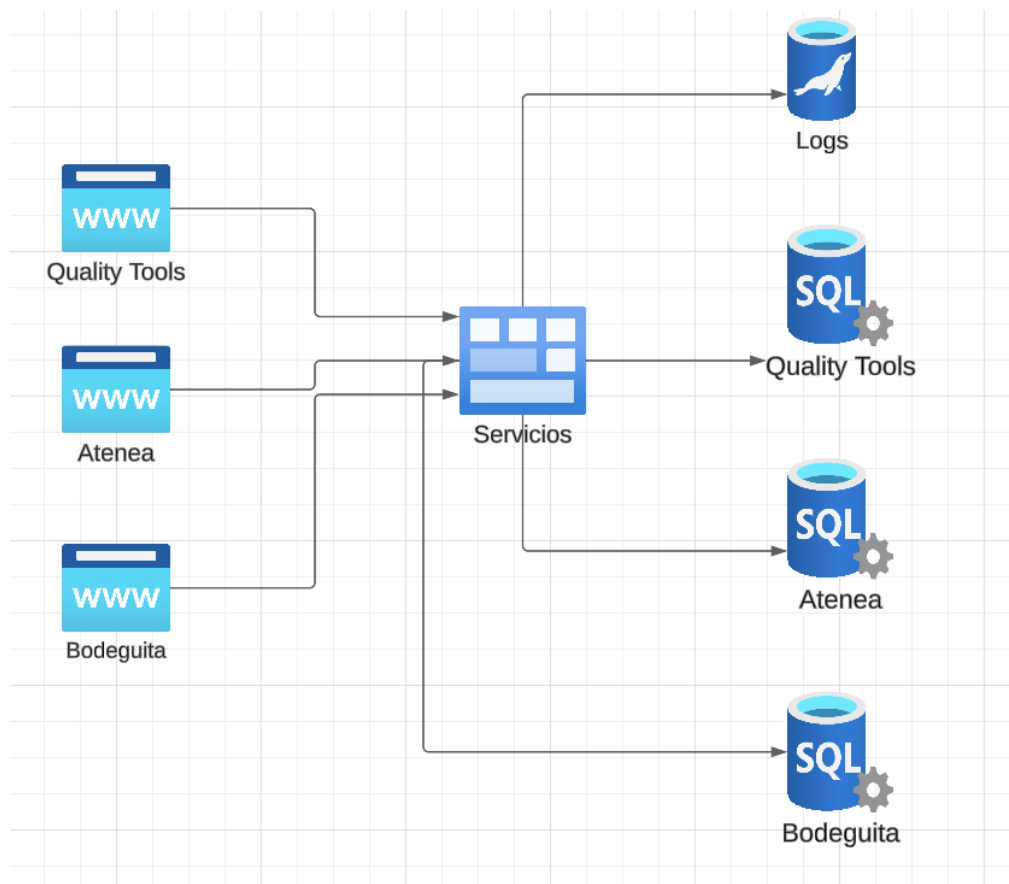
La idea es ir enfascando cada funcionalidad a razón que si una de ellas no llega a funcionar de la mejor manera no perdemos la funcionalidad total de los sistemas.

Sugerencia de lenguaje para Servicios

Se sugiere como lenguaje base PHP con laravel, pues es un lenguaje bastante ligero y posee uno de los frameworks mas robustos a nivel de estructura y escalabilidad, lo cual nos va permitir hacer consultas mas libres y manipular la información de una forma más sencilla de cómo está sucediendo en .net

Propuesta alternativa: API rest

En una propuesta alternativa a microservicios sería implementar un api rest que se conecte a todas las bases de datos y recolecte la información para poder manipularla y así hacer entradas y salidas.



Esta es una estructura mas sencilla donde todos los servicios son hechos en el clúster Servicios lo cual tendremos del detalle que deberá conectarse a todas las bases de datos, es decir realizando múltiples conexiones, a nivel de estructura y tiempo de implementación es mas sencillo sin embargo tendremos el detalle de las múltiples conexiones que a la larga será complicado de mantener




Análisis para recursos de servidor.

Para iniciar hablaremos sobre la cantidad de personas y la probabilidad concurrente que puede existir al visitar los sitios.









Se estima que del lado del sistema de Quality Tools existen alrededor de 60 usuarios registrados, de los cuales no todos tienen el mismo acceso, ni acceso a todos los módulos, alrededor de 4 personas tienen acceso total y no acceden con mucha concurrencia pues son usuarios administradores que solo acceden si existe algún inconveniente. En la mayoría de módulos no es necesario una asistencia constante, lo mas probable es que 1 vez al día por usuario, a revisar algún proceso o llenado de información.

En el caso de Atenea se cuentan con 40 usuarios aproximadamente, quienes acceden concurrentemente son los PM pues visualizan información de los proyectos constantemente

Estado actual del servidor

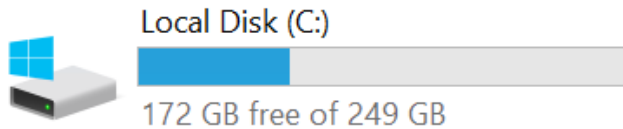
Processes Performance Users Details Services			
Name	Status	24% CPU	49% Memory
 .NET Host		0%	1,150.6 MB
 .NET Host		0%	1,054.8 MB
>  IIS Manager		0%	236.1 MB

Proyectos actuales

-  Sites
 - >  AprobacionesProduction
 - >  AprobacionesTesting
 - >  AteneaBeta
 - >  AteneaProduction
 - >  AteneaTesting
 - >  BodeguitaProduction
 - >  BodeguitaTesting

Disco duro

✓ Devices and drives (2)



Especificaciones

Windows edition

Windows Server 2019 Standard

© 2018 Microsoft Corporation. All rights reserved.

Windows Server® 2019

System

Processor:	Intel(R) Xeon(R) CPU E5-2609 v4 @ 1.70GHz 1.70 GHz (2 processors)
Installed memory (RAM):	12.0 GB
System type:	64-bit Operating System, x64-based processor
Pen and Touch:	Pen and Touch Support with 10 Touch Points

Para poder optimizar un poco más podríamos retirar los proyectos de testing y moverlos completamente a el servidor de pruebas, para dejar este complemente con los proyectos de producción actuales y los que se irán sumando. Con eso podemos reducir el movimiento no necesario en el servidor que dará cabida a utilizarse con los proyectos nuevos de producción que en este caso seria los proyectos de los servicios que se van a elaborar.

Referente a la base de datos no sufriría cambios, actualmente se encuentran las bases de producción y las QA entonces igual podríamos migrar las de QA para liberar espacio y rendimiento a nivel de base, en este caso no agregaríamos más bases pues ocuparíamos las existentes de producción para obtener la data de los diferentes servicios implementados.

Resumen

Análisis de Recursos y Carga Actual

1. Usuarios y Concurrency:

- **Quality Tools:** 60 usuarios registrados, con poca concurrencia. Accesos esporádicos, mayormente administrativos.
- **Atenea:** 40 usuarios, con PMs accediendo frecuentemente para visualizar información de proyectos.

2. Estado Actual del Servidor:

- **CPU:** 24% de uso.
- **RAM:** 49% de uso.
- **Disco Duro:** 249 GB total, 172 GB libres.
- **Especificaciones:** 12 GB RAM, Intel Xeon CPU E5-2906 v4 @ 1.70 GHz, 2 procesadores.

3. Proyectos Actuales:

- 7 proyectos en IIS, 2 pausados.
- Se recomienda mover proyectos de testing a un servidor de pruebas.

Consideraciones para Arquitectura de Microservicios

1. Recursos Adicionales Necesarios

• Microservicios:

- Cada microservicio puede consumir recursos adicionales de CPU y RAM.
- Espera un aumento en el uso de red debido a la comunicación entre microservicios.

• API Gateway:

- La API Gateway añadirá un nivel de abstracción y manejará todas las solicitudes, lo que puede aumentar la carga en CPU y RAM.

2. Base de Datos

- Tendrás 3 bases de datos principales (esto puede crecer).
- La API Gateway manejará las conexiones e intercomunicaciones.
- Considera la posibilidad de migrar las bases de datos de QA a un servidor separado para liberar recursos.

Análisis de Viabilidad

1. Capacidad del Servidor Actual

- **CPU y RAM:**
 - Uso actual de CPU es 24%, lo cual es relativamente bajo.
 - Uso de RAM es 49%, con 12 GB disponibles. Alrededor de 6 GB están en uso.
- **Disco Duro:**
 - 172 GB libres, lo cual es suficiente para alojar nuevas bases de datos y microservicios a corto plazo y mediano plazo.

2. Impacto de Migrar a Microservicios

- **Aumento en la Carga de Red:** La comunicación entre microservicios a través de la API Gateway puede aumentar el uso de red.
- **Aumento en Uso de CPU y RAM:** Cada microservicio y la API Gateway añadirán carga adicional a la CPU y RAM.

Recomendación: Considera la posibilidad de escalar horizontalmente (agregar más servidores) si el uso de recursos se vuelve lento, esto no afecta el desarrollo ni la organización del esquema de microservicios.

Como conclusión actualmente el servidor es suficiente para la cantidad de proyectos y en el caso de agregar más, sin embargo si la dimensión de los proyectos y cantidad aumenta exponencialmente, es recomendable tener un servidor similar para poder ir alojando lo nuevo para no ir saturando hasta el punto de colapsar, pero por el momento se considera que soporta la estructura a corto y mediano plazo.