Contenido

[Microservices: Theory and Application 3](#_Toc497950818)

[1. Arquitectura y comunicación 3](#_Toc497950819)

[2. Microservicios 3](#_Toc497950820)

[2.1. Escalabilidad 3](#_Toc497950821)

[2.2. Filosofía 3](#_Toc497950822)

[3. Enfoques de éxito 3](#_Toc497950823)

[3.1. Éxito del negocio 3](#_Toc497950824)

[3.2. Éxito para nosotros 3](#_Toc497950825)

[4. Tamaño del equipo 4](#_Toc497950826)

[5. Tecnologías 4](#_Toc497950827)

[5.1. CQRS 4](#_Toc497950828)

[5.2. Event Sourcing 4](#_Toc497950829)

[5.3. API Gateway 4](#_Toc497950830)

[What I Wish I Had Known Before Scaling Uber to 1000 Services 5](#_Toc497950831)

# Microservices: Theory and Application

## Arquitectura y comunicación

Los patrones de diseño determinan la forma en que se comunican los objetos en un programa. En nivel de calidad de la comunicación dependerá también de cuán estandarizados estén los métodos de desarrollo, mientras más consenso exista, mayor será el nivel de comunicación.

## Microservicios

Son pequeños servicios que permiten separar las funciones de negocio, la interacción entre ellos no depende de ninguna tecnología y su migración es fácil.

### Escalabilidad

Otra de las ventajas de los microservicios es la escalabilidad, al estar las funciones del negocio separadas en distintos microservicios, estas se pueden escalar de manera individual sin necesidad de afectar a las demás.

### Filosofía

El expositor llama la atención en que es necesario entender la filosofía detrás de los microservicios, esto para evitar caer en “guías” o “recetas”. La filosofía de los microservicios se resume en dividir las cosas hasta lograr servicios bien enfocados, es decir servicios que sólo hagan una coas y la hagan bien.

## Enfoques de éxito

### Éxito del negocio

El éxito del negocio depende en gran medida del entendimiento nel negocio, la estructura organizacional y los objetivos. La estrategia tecnológica debe estar alineada con estos.

### Éxito para nosotros

Depende del rol de cada persona, para algunos puede ser realizar las tareas a tiempo, otros respetando el presupuesto, entendiendo a los usuarios,etc.

## Tamaño del equipo

El tamaño del equipo de desarrollo debe ser pequeño, aproximadamente 10 personas. Sin embargo, también juega un rol muy importante la confianza entre los miembros del equipo.

## Tecnologías

### CQRS

Permite separar las operaciones de lectura y escritura, en este caso, un message broker direcciona la interacción a un microservicio ya sea de escritura o de lectura, el mismo que se conecta a la base de datos.

### Event Sourcing

Permite almacenar eventos altamente transaccionales, su aplicación es recomendada en aplicaciones de banca y finanzas, debido a que ofrece una amplia trazabilidad de las transacciones realizadas

### API Gateway

Su uso es ideal cuando los microservicios cambian de manera dinámica, en este caso el API Gateway sirve como un bs que redireccionas las llamadas de la aplicación cliente a los respectivos microservicios.

# What I Wish I Had Known Before Scaling Uber to 1000 Services

El número de servicios web con los que cuenta UBER se ha ido incrementando, esto hasta que la tendencia actual es de tener aproximadamente 1000 servicios.

## ¿Por qué microservicios?

### Costos

El costo de mantenimiento de los microservicios es más bajo que el de una aplicación monolítica, al estar las funcionalidades separadas por servicios es más fácil solucionar posibles errores. Asimismo, se listan algunas ventajas extra, por ejemplo, la facilidad de reemplazar algún servicio fallido en lugar de perder tiempo reparándolo o la posibilidad de evitar tiempo en coordinaciones para desarrollos. De la misma manera, se menciona que los microservicios no dependen de ninguna tecnología o lenguaje de programación.

### Número de repositorios

Actualmente UBER tiene aproximadamente 7000 repositorios, el expositor explica que el escenario ideal sería contar con un solo repositorio que facilite la navegación en el código. Sin embargo, un solo repositorio en la práctica sería imposible de usar

### Operación de los microservicios

En el aspecto operacional, el manejo de errores puede resultar dificultosos, esto debido a que existen dependencias entre microservicios, por lo tanto, para que un equipo pueda realizar pruebas, debe coordinar con los demás que puedan estar relacionados.

### Rendimiento

Dado que los microservicios muchas veces se programan en distintos lenguajes, monitorear el rendimiento puede significar una dificultad, esto se debe a que no todos os lenguajes cuentan con herramientas para esta labor, un caso Odría ser el de java.

### Tracing

No siempre se encuentra una traza óptima, algunas veces se debe sacrificar el rendimiento de las bases de datos para favorecer el de los servicios.

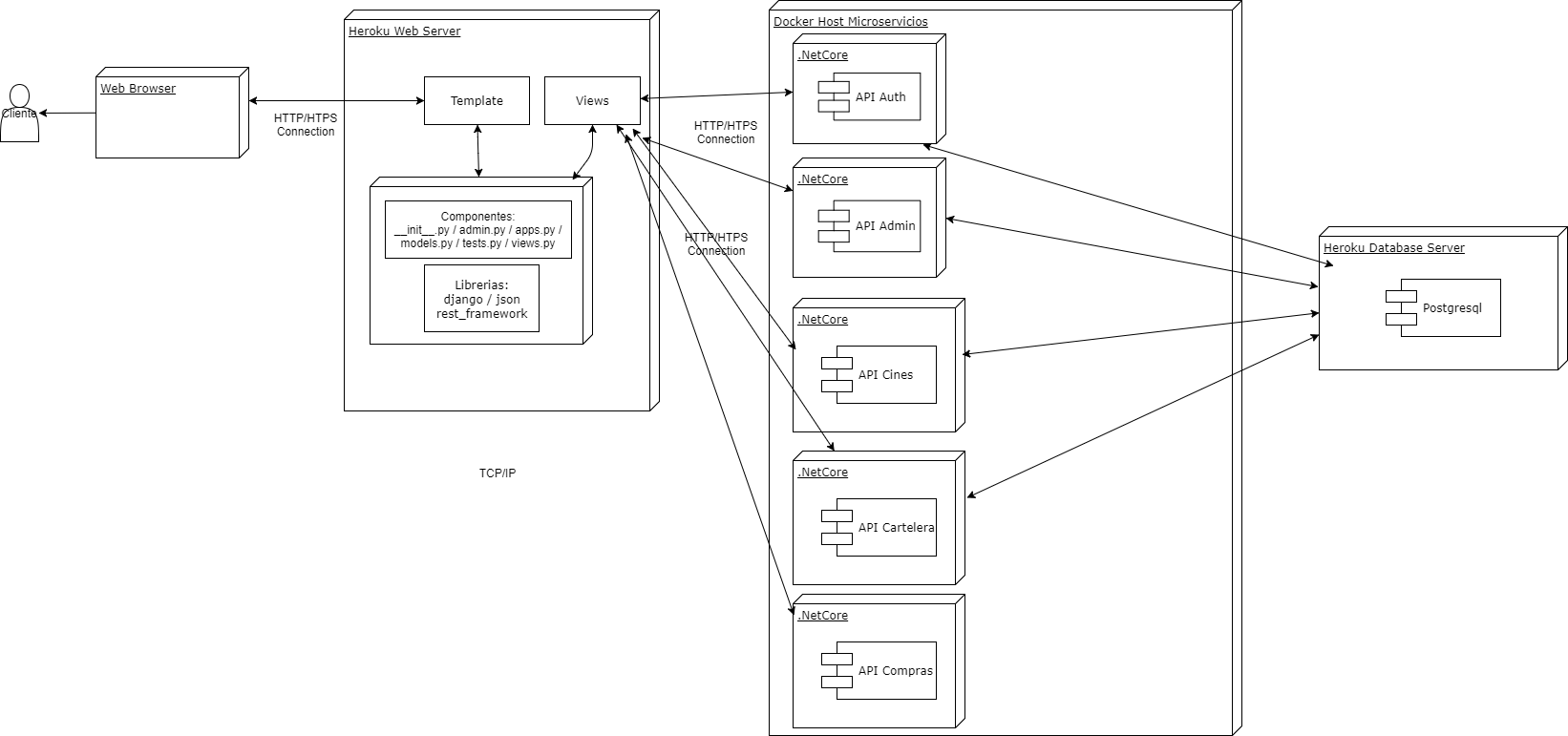
### Logging

El expositor recomienda para este caso, definir un estándar. Muchos desarrolladores tienden a realizar sus propios logs de acuerdo a sus estilo, sin embargo, dado la cantidad y heterogeneidad de los servicios, lo recomendable es manejar un estándar dado que un mal logging puede agudizar los problemas en lugar de mostrar posibles soluciones.

### Load testing

Para obtener una perspectiva real son realizados en los ambientes de producción, esto en las horas de menor tráfico. Sin embargo, esto requiere de una buena coordinación dado que puede generar malentendidos y confusión.

.



## Explicación del diagrama de despliegue

1. Web Browser: Navegador web de preferencia del usuario, se decidió usar tecnología web dado que el usuario no específico ninguna tecnología en particular. Por ello, se asumió la creación de una web responsive.
2. Heroku web server: Servicio proveído en Cloud por Heroku, se eligió desarrolar una aplicación en Python dado que el framework DJANGO incluyen elementos como la gestión de accesos y facilidad para desarrollar servicios web en un corto tiempo.
3. Docker host Microservicios:

Se decidió usar un contenedor Docker dado que permite alta disponibilidad, cada microservicio fue desarrollado usando .NET Core, el cual está optimizado para este tipo de desarrollos y presenta un rendimiento 10 veces superior al .NET convencional, de la misma manera, su rendimiento es superior al de Python o Java.

Documentación oficial:

https://docs.microsoft.com/en-us/dotnet/standard/microservices-architecture/net-core-net-framework-containers/net-core-container-scenarios