THE NINJ/PROJECT

Microservices 101

Raimundo Alegría

THE NINJA PROJECT

El objetivo de este learning path es recorrer de una forma práctica algunos los aspectos de los aspectos más relevantes que influyen tener éxito a la hora de desarrollar una arquitectura de microservicios

THE NINJA PROJECT

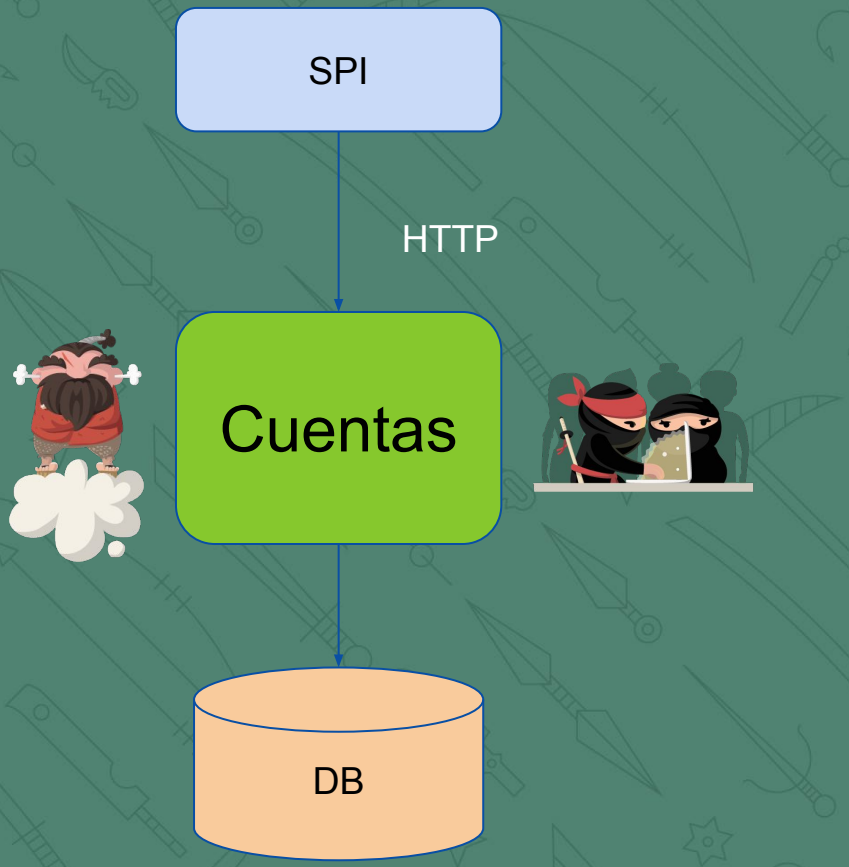
Talleres previstos. Microservices 101

- Conceptos básicos
- ATDD
- TDD
- Containers. Docker
- Infraestructura. Kubernetes
- Continuous delivery
- Seguridad
- Bases de datos
- Event sourcing

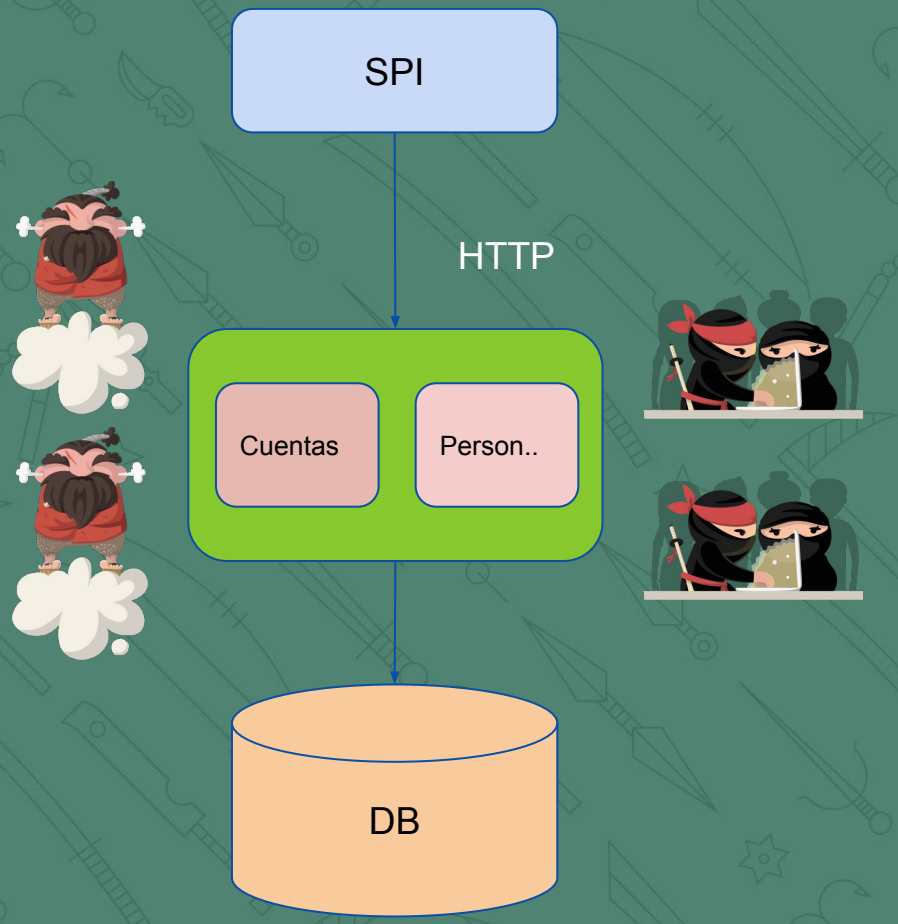
THE NINJA PROJECT

¿Por qué?

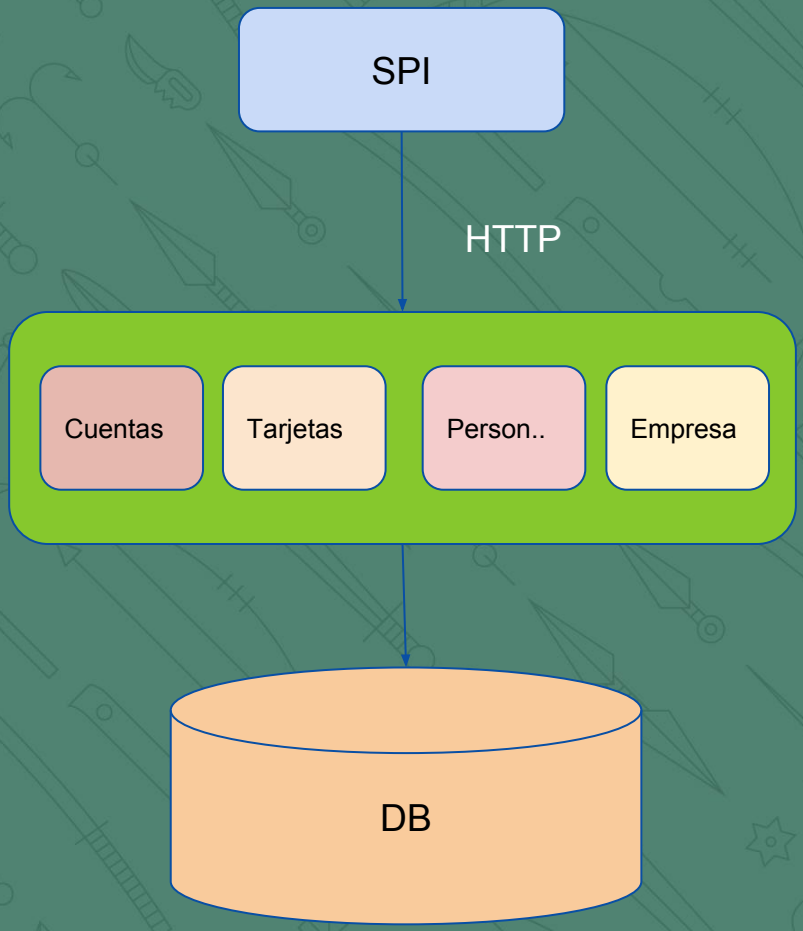
Los comienzos suelen ser sencillos...



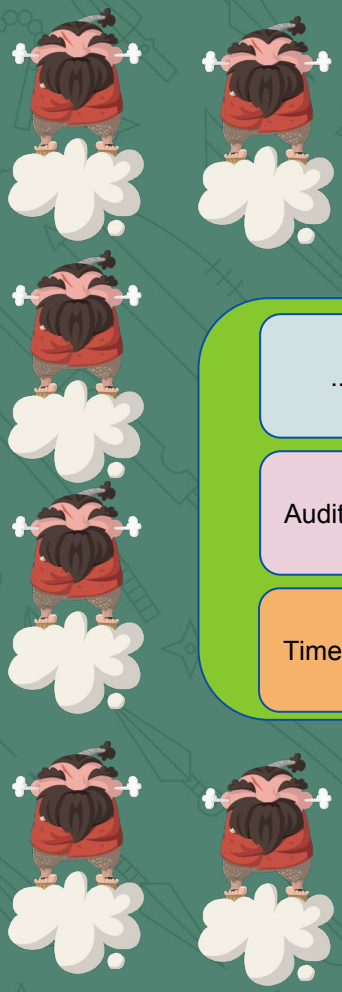
..pero según pasa el tiempo...

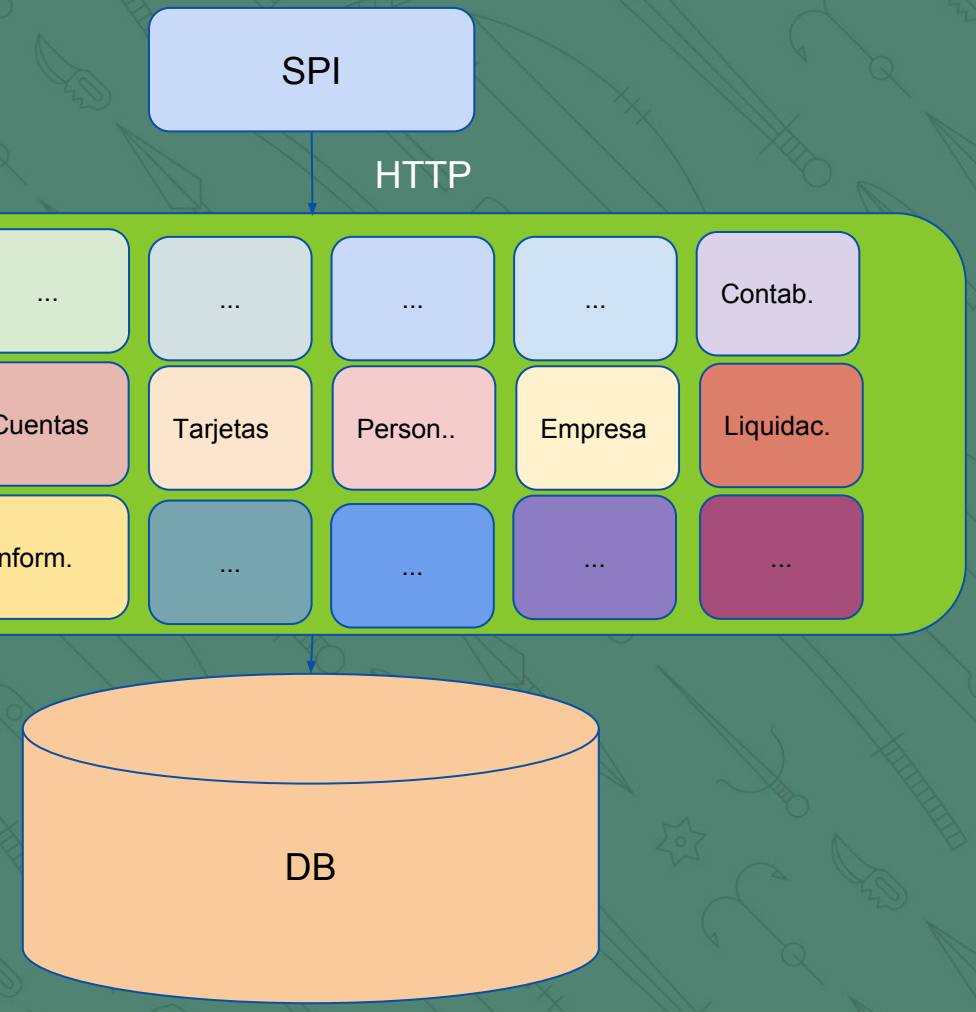


.. las funcionalidades ..



.. crecen ..




SPI
HTTP
...
...
...
...
...
Contab.
Auditoría
Cuentas
Tarjetas
Person..
Empresa
Liquidac.
Timeline
Inform.
...
...
...
...
DB

THE NINJA PROJECT

.. Y esto genera algunos problemas ..

- Los ritmos de evolución son distintos en cada uno de los módulos.
- Hay muchos equipos de desarrollo trabajando a la vez sobre el mismo código.
  - Requiere coordinación a la hora de poner en producción.
  - Aparecen ventanas de tiempo para puesta en producción.
- Es difícil modificar y ampliar el código.
- Existe gran dificultad para evolucionar la tecnología.

THE NINJA PROJECT

.. Y esto genera algunos problemas ..

- La base de datos crece en complejidad.
  - Tablas grandes. Muchas columnas.
  - Consultas cada vez mas complicadas.
  - Datos en las tablas que significan cosas distintas para partes distintas.
- Diferentes requisitos no funcionales en cada módulo.
  - Seguridad.
  - Escalabilidad.
  - Tiempos de respuesta.

THE NINJA PROJECT

.. Y sobre todo ..

Cada vez se tarda más en poner software en producción

THE NINJA PROJECT

¿Y si usamos microservicios?

THE NINJA PROJECT

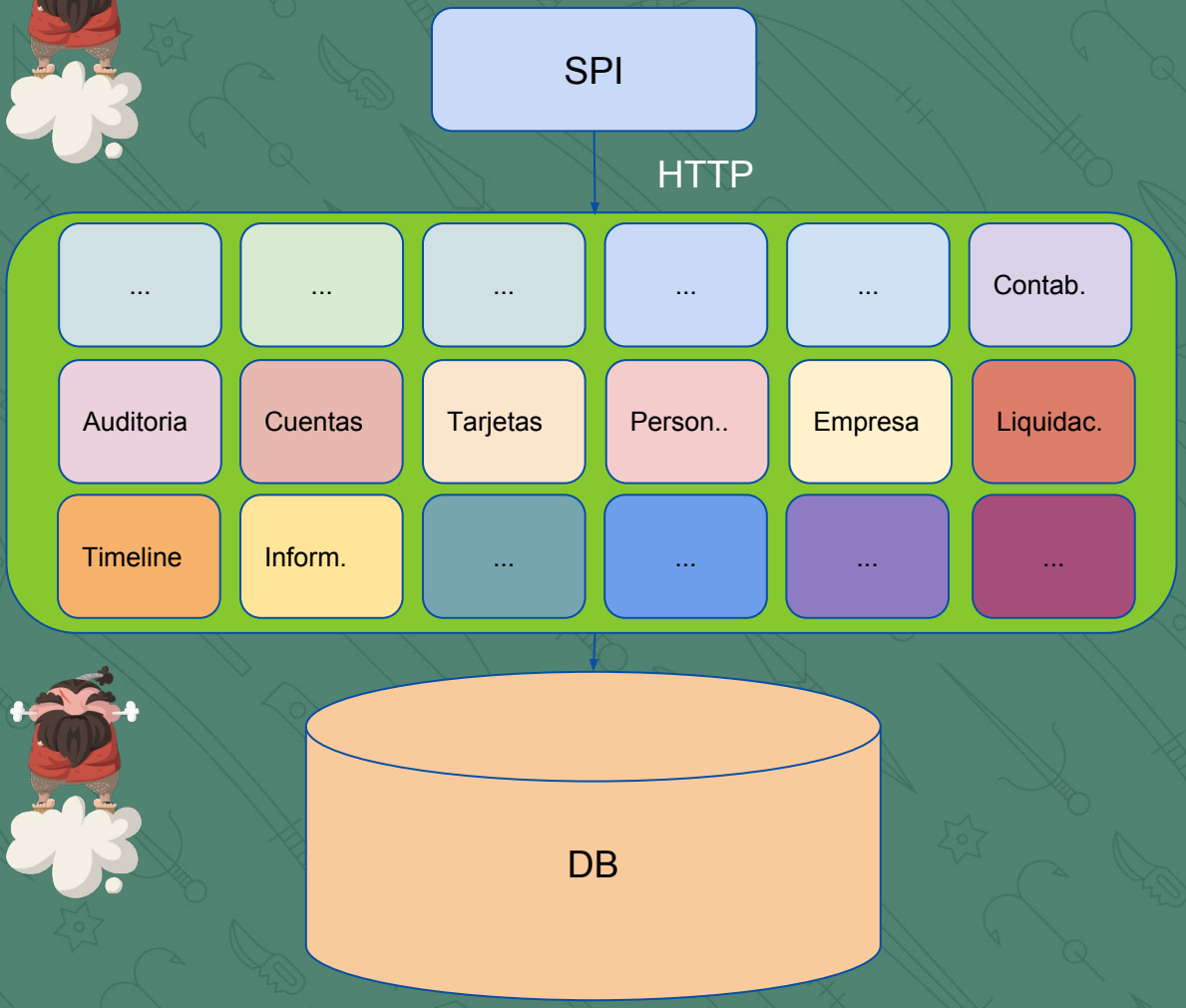
¿Qué es eso?

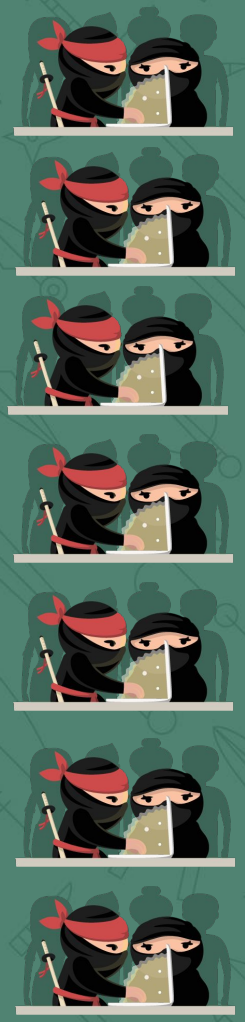
¿En qué consiste una arquitectura de microservicios?

Consiste en desarrollar una aplicación como un conjunto de servicios **independientes**, cada uno de ellos funcionando en su propio proceso y comunicándose con mecanismos ligeros, a menudo API's REST.

THE NINJA PROJECT

¿Pero cómo empezamos?





THE NINJA PROJECT

Con un servicio...

THE NINJA PROJECT

¿Cómo elegimos ese servicio?

- Los microservicios están organizados alrededor de capacidades de negocio.
- Las técnicas de Domain Driven Design pueden ayudar:
  - Ubiquitous language
  - Subdominio
  - Bounded context

THE NINJA PROJECT

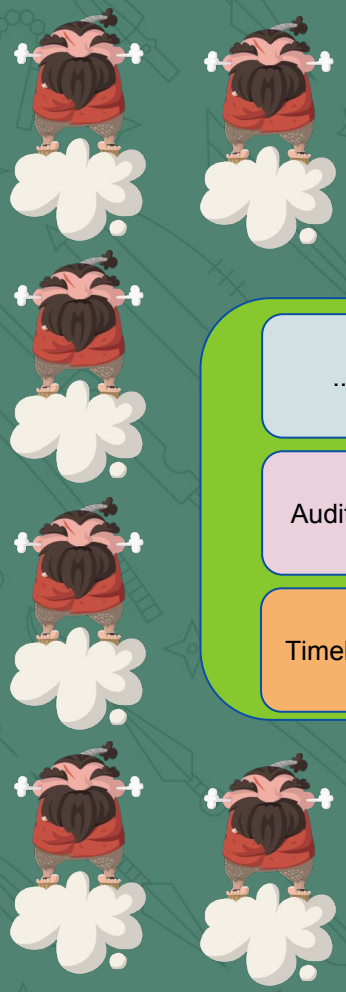
¿Cómo elegimos ese servicio?

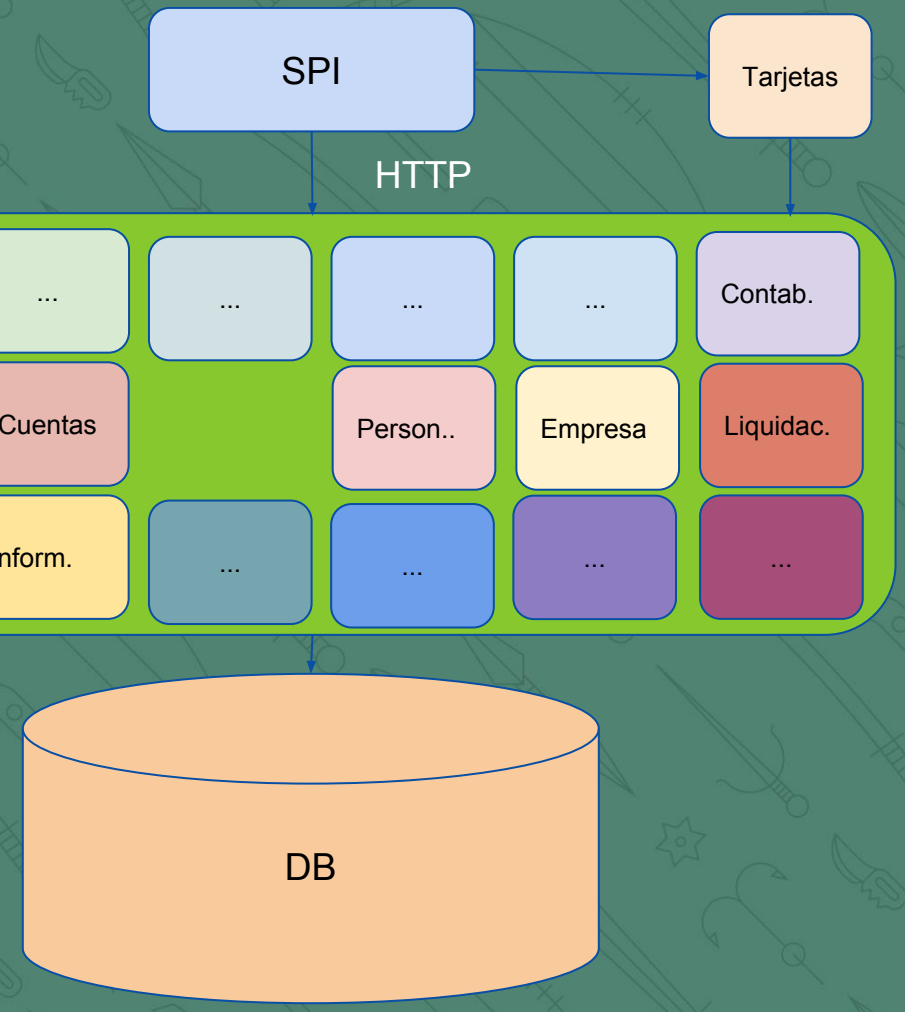
- Ritmo de cambio.
- Estructura del equipo.
- Requisitos no funcionales
  - Seguridad.
  - Escalabilidad.
  - Tiempos de respuesta.
- Requisitos tecnológicos.
- Patrones de uso.

Primer ejercicio

- Diferenciar cuál serían los posibles microservicios de un sistema monolítico.
- Elegir cúal sería el primer servicio con el que iniciar el proceso de partición.
- Equipos de 3 personas
- 15 min

Extraemos nuestro primer servicio



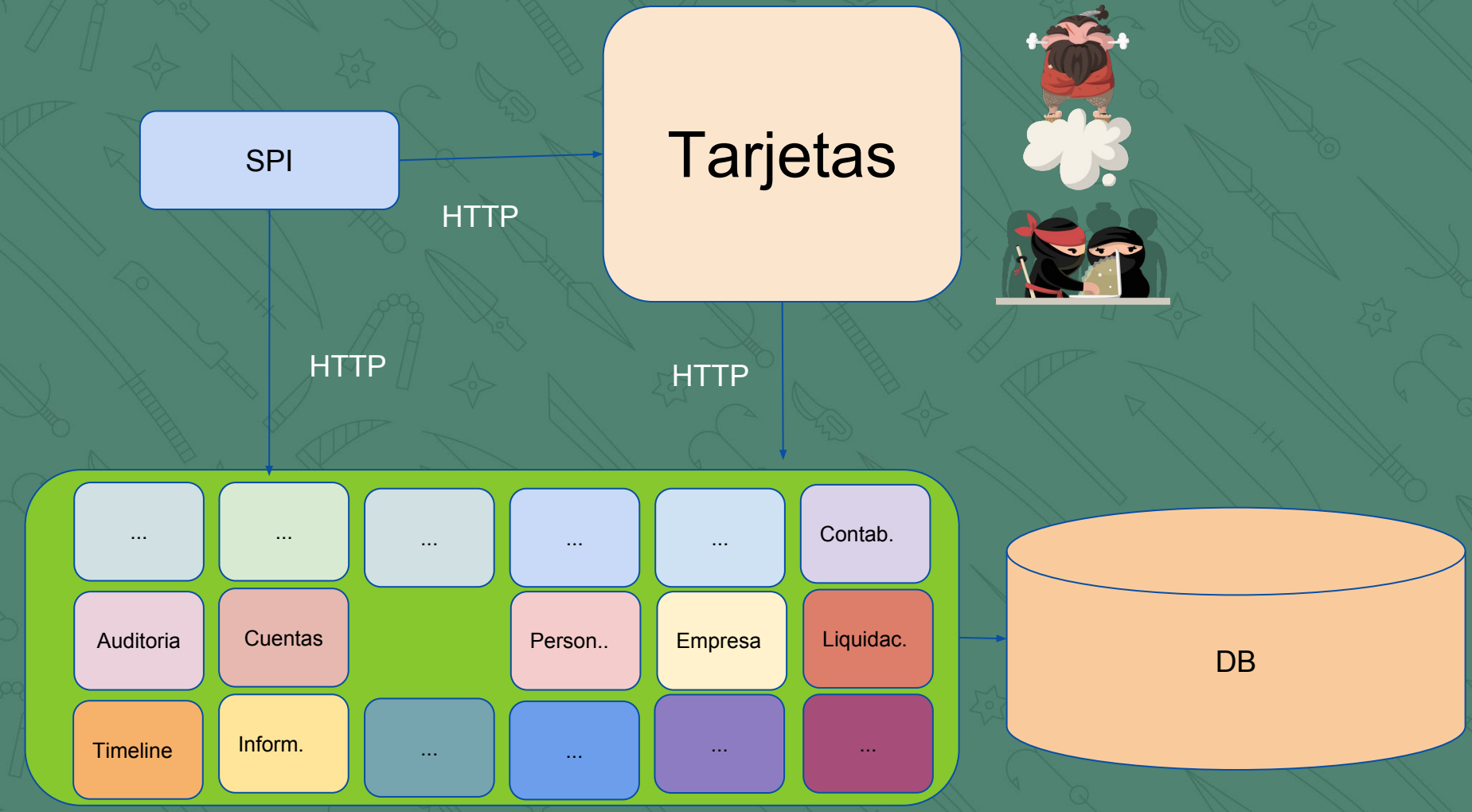


¿Quién es responsable del servicio?

- Un único equipo:
  - Pequeño (Two pizza team).
  - Multidisciplinar. Incluye a las personas de negocio, desarrolladores, expertos en infraestructura, etc.
  - Que decide sobre todos los aspectos de cómo se implementa el servicio (Gobierno descentralizado).
  - Puede desarrollar varios microservicios.
  - DevOps
    - Desarrolla y opera los servicios de los que es propietario.

THE NINJA PROJECT

¿Quién es responsable del servicio?



¿Cómo es su ciclo de vida?

- El servicio es tratado como un producto no como un proyecto.
  - No tiene un principio y un fin definidos
  - Lo mantiene un equipo estable.
  - Debe satisfacer las necesidades de sus clientes. Si no las satisface el servicio desaparece.

THE NINJA PROJECT

¿Cómo se comunica el servicio?

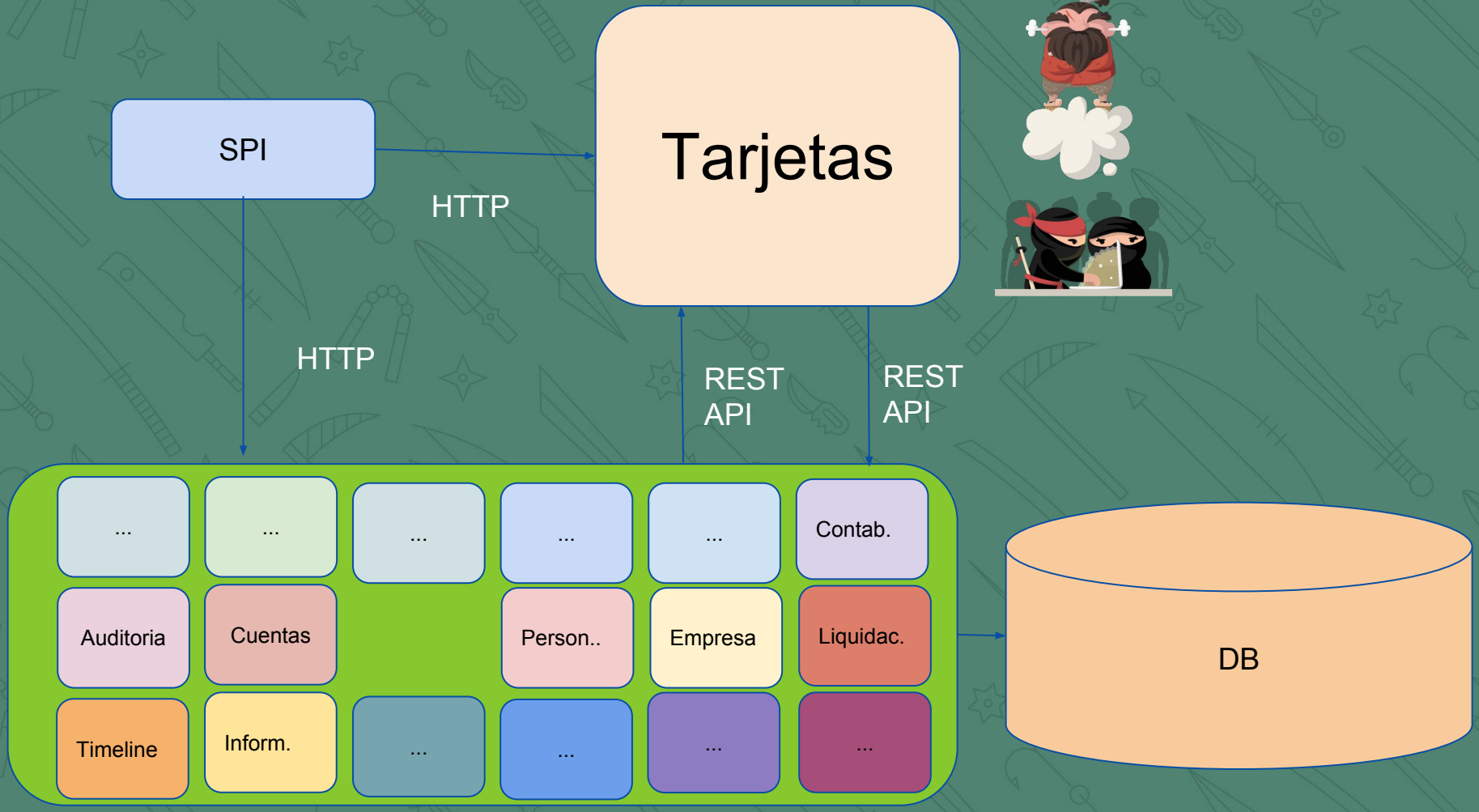
- Rest
  - Hypermedia
- Servicios de mensajería simples
  - No tienen lógica dentro del bus
  - No ESB's
  - Normalmente de forma asíncrona (publish/subscribe)
- Protocolos
  - json
  - xml
  - binarios
    - avro
    - protocol buffers
    - GRPC

¿Cómo se comunica el servicio?

- Los endpoints son inteligentes para interpretar los mensajes
  - Coreografía
  - Orquestación
- Ejemplos de sistemas de mensajería
  - RabbitMQ
  - Kafka
  - ActiveMQ
  - ZeroMQ

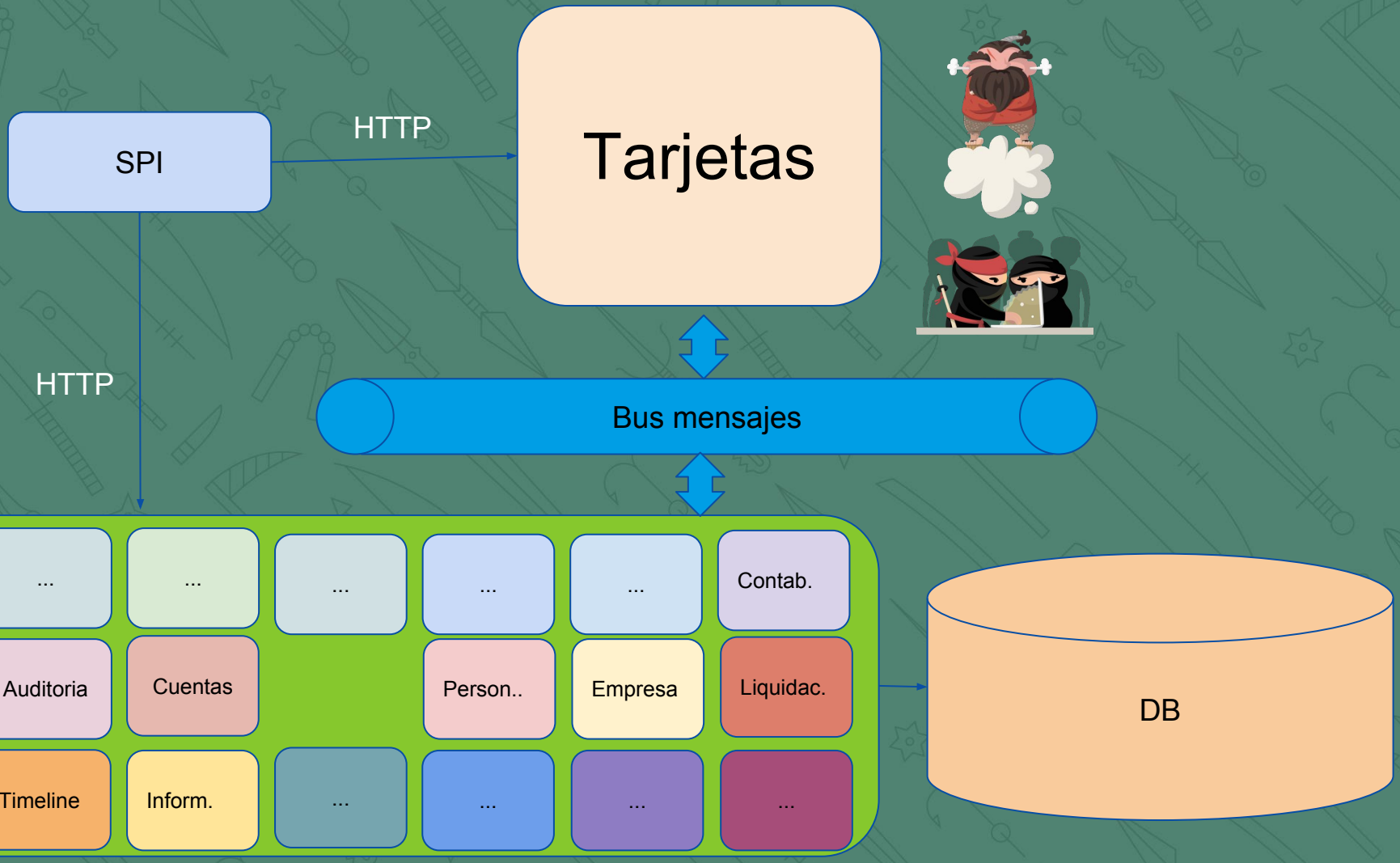
THE NINJA PROJECT

¿Cómo se comunica el servicio?



THE NINJA PROJECT

¿Cómo se comunica el servicio?



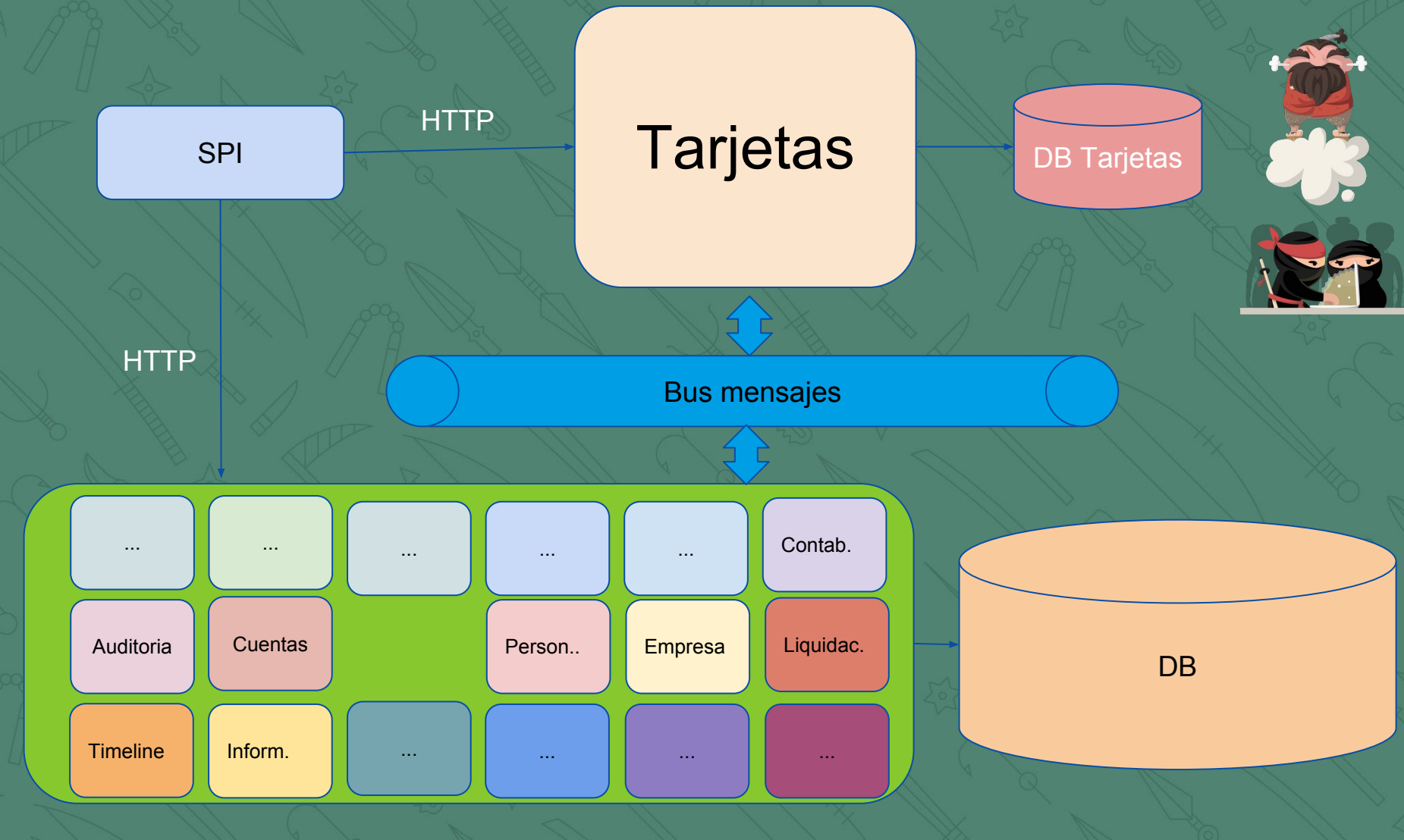
THE NINJA PROJECT

¿Cómo guarda los datos el servicio?

- Cada servicio tiene su propia base de datos.
- No se usa la base de datos como mecanismo de integración.
- La gestión de los datos es descentralizada
  - El equipo elige:
    - La tecnología en la que se guarda la información
    - El esquema (o no esquema) de los datos.

THE NINJA PROJECT

¿Cómo guarda los datos el servicio?



Segundo ejercicio

- Elegir la mejor forma de comunicación entre el servicio y el resto.
  - Api que presenta hacia el exterior
  - Principales eventos que publica.
- Equipos de 3 personas
- 15 min

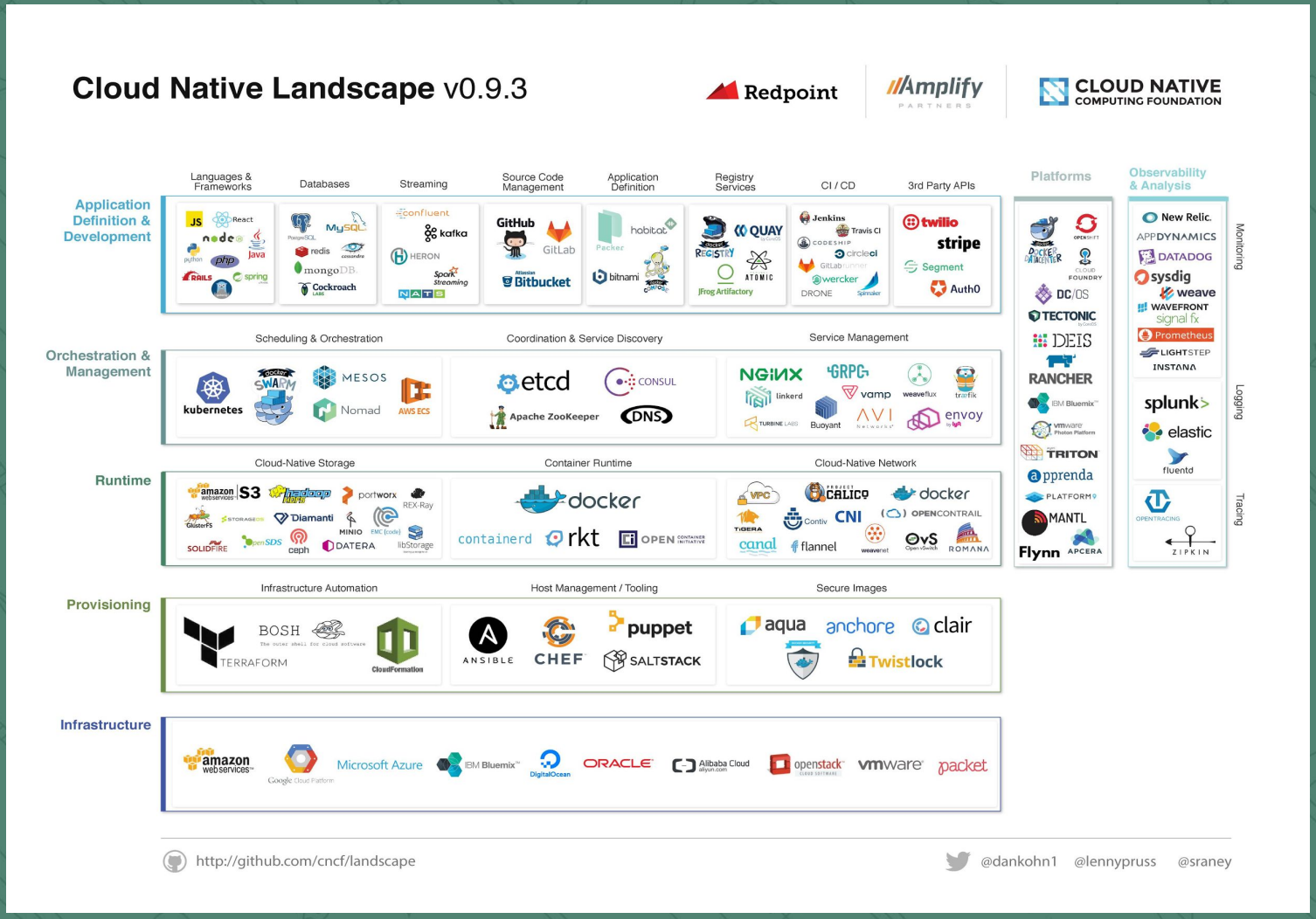
THE NINJA PROJECT

¿Que uso para desplegar el servicio?

- Máquina virtual
- Containers (Docker)
  - En este caso hay que tener en cuenta:
    - Sistemas operativos minimalistas para los containers
      - Alpine
    - Para los hosts
      - CoreOs
      - RancherOs
      - Atomic
      - Protón

THE NINJA PROJECT

¿Que uso para desplegar el servicio?

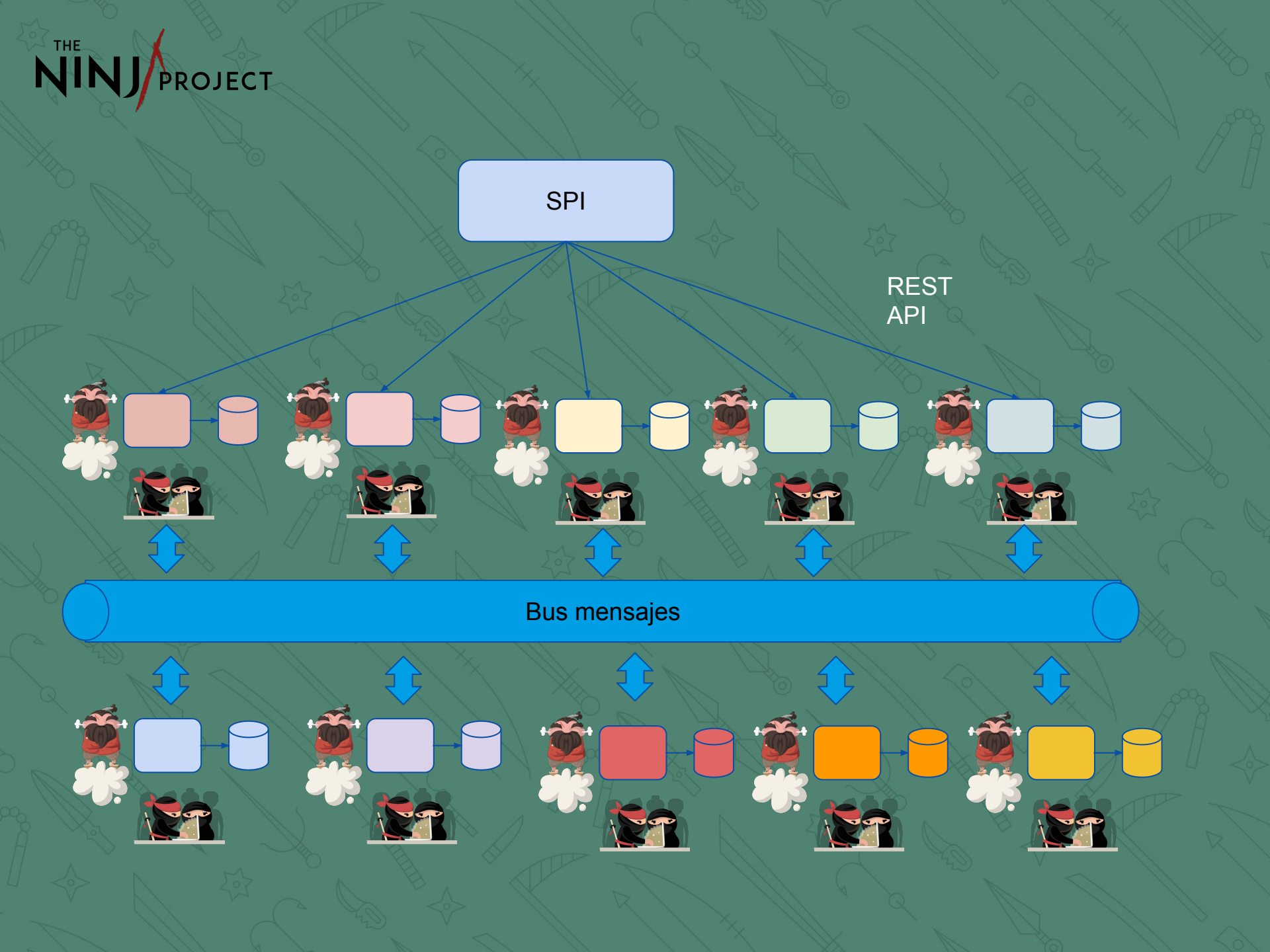


THE NINJA PROJECT

El primer servicio fue un éxito...

THE NINJA PROJECT

Repitamos el módelo con todos...


THE NINJA PROJECT
SPI
REST API
Bus mensajes

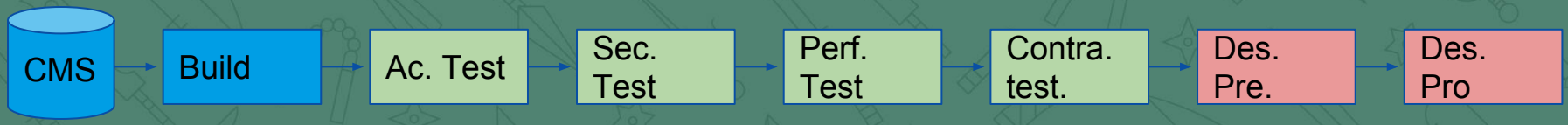
¿Y cómo desplegamos todo esto?

- Automatizando todo.
  - La construcción de los artefactos.
  - Las pruebas que se deben realizar para asegurarse que el sistema se comporte de la manera que se espera.
    - Funcionales.
    - Rendimiento.
    - Contrato.
    - Smoke tests.
  - La definición de la infraestructura.
  - La seguridad.
  - El proceso de despliegue.
  - La migración de los datos.

THE NINJA PROJECT

¿Y cómo desplegamos todo esto?

Todos estos procesos se materializan en lo que se llama un deployment pipelines

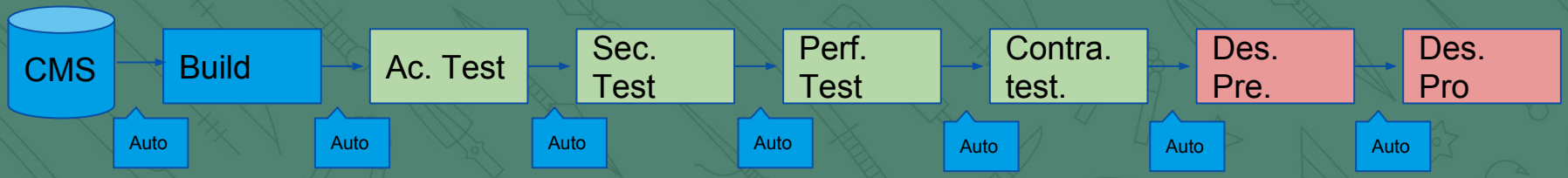


Un pipeline con consiste en una serie de etapas automatizadas que llevan el software a producción.

THE NINJA PROJECT

¿Y cómo desplegamos todo esto?

Continuous deployment

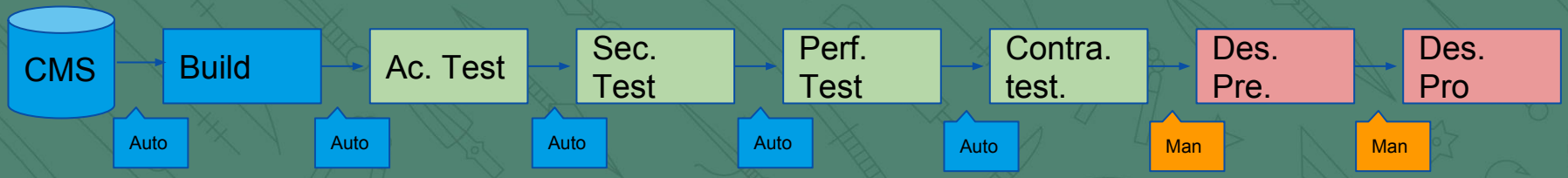


Cada cambio que ocurre en el control de versiones transita hasta producción automáticamente.

THE NINJA PROJECT

¿Y cómo desplegamos todo esto?

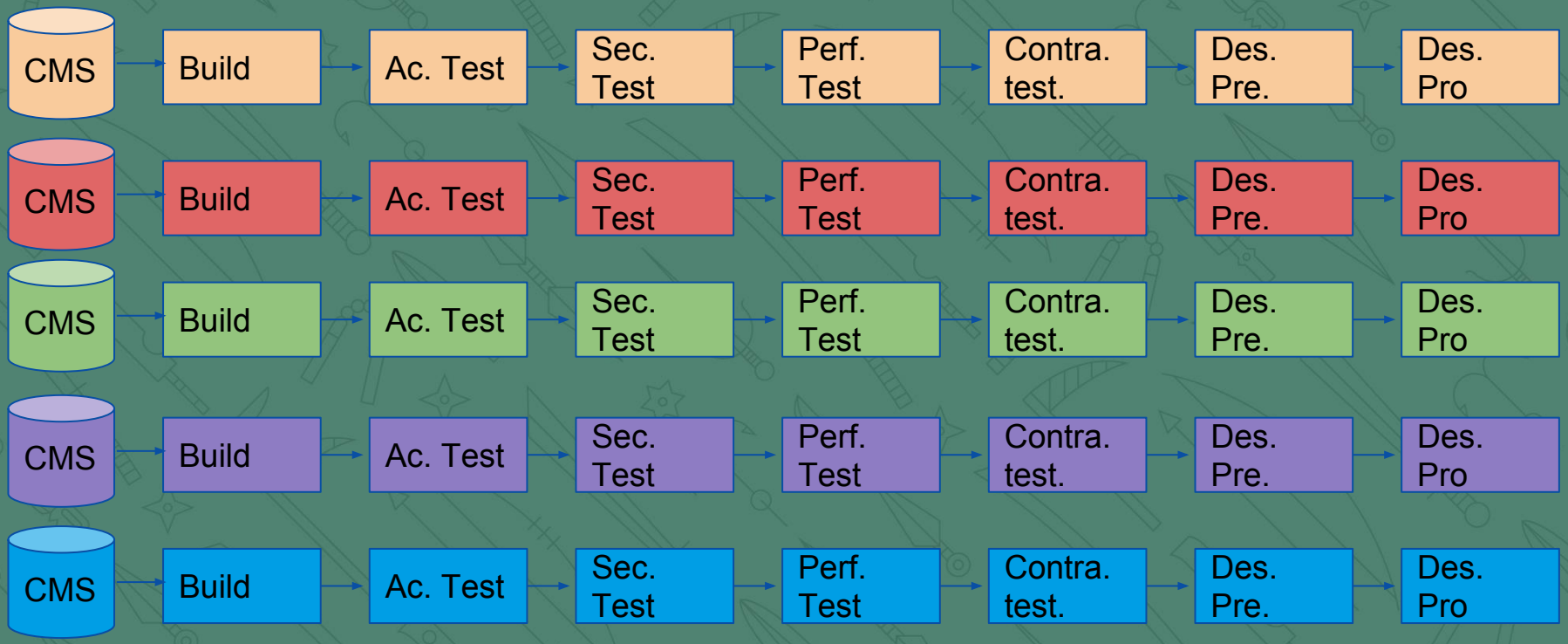
Continuous delivery



Cada cambio que ocurre en el control de versiones transita con paso automáticos y aprobaciones manuales hasta producción.

THE NINJA PROJECT

¿Y cómo desplegamos todo esto?

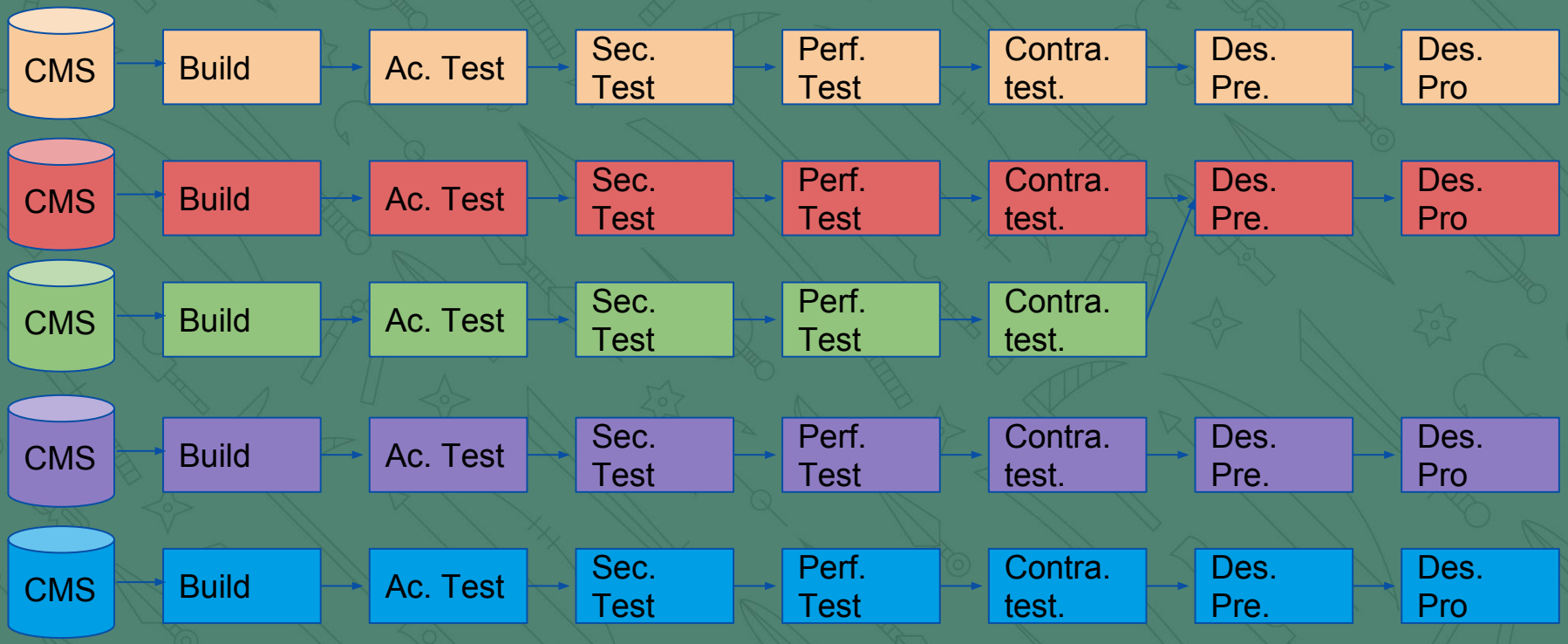


THE NINJA PROJECT

Los microservicios deben poder desplegarse siempre que sea posible de manera independiente.

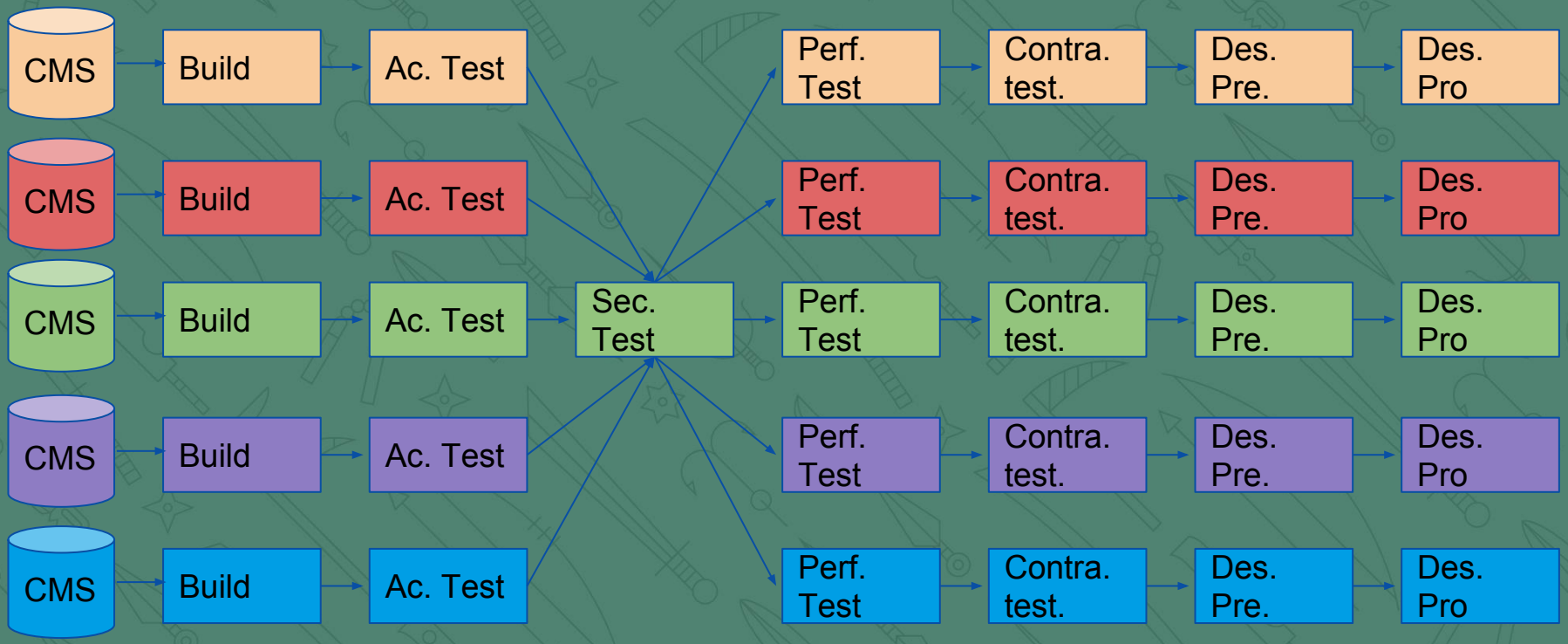
THE NINJA PROJECT

Dependencia entre dos servicios



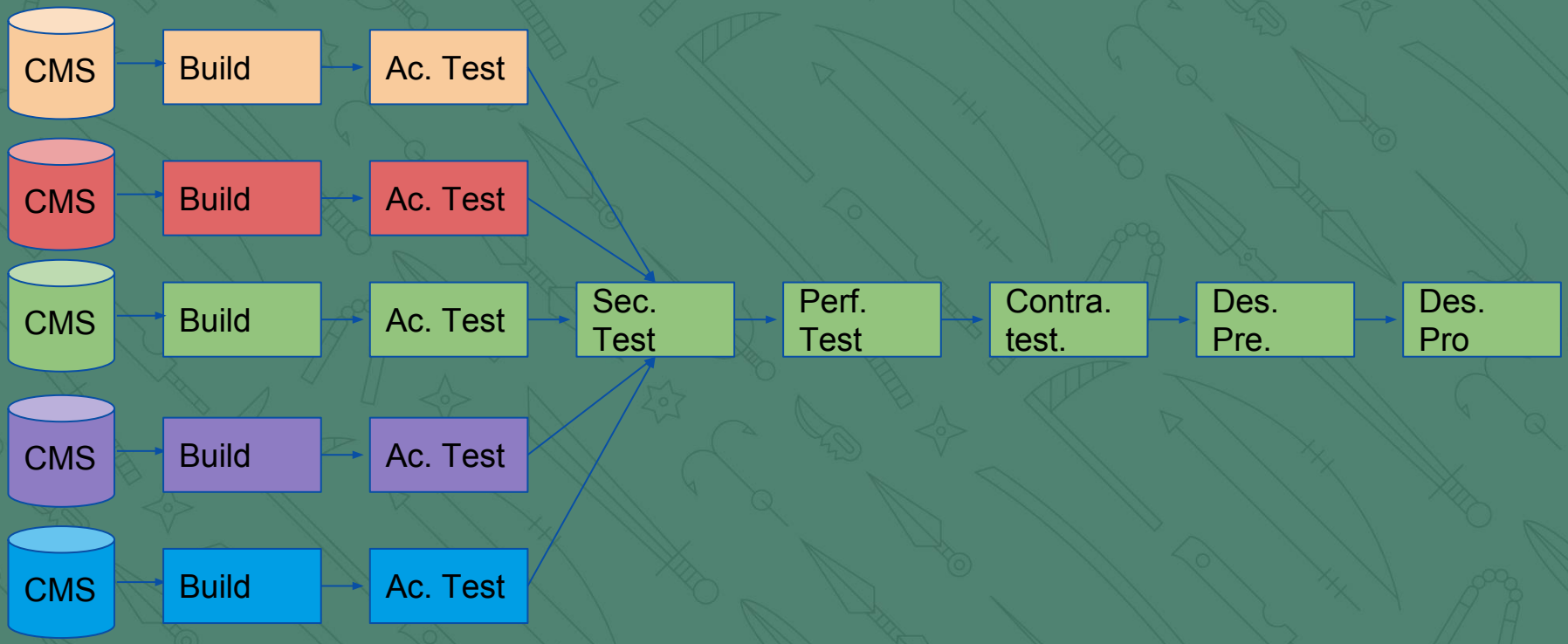
THE NINJA PROJECT

Punto central de validación



THE NINJA PROJECT

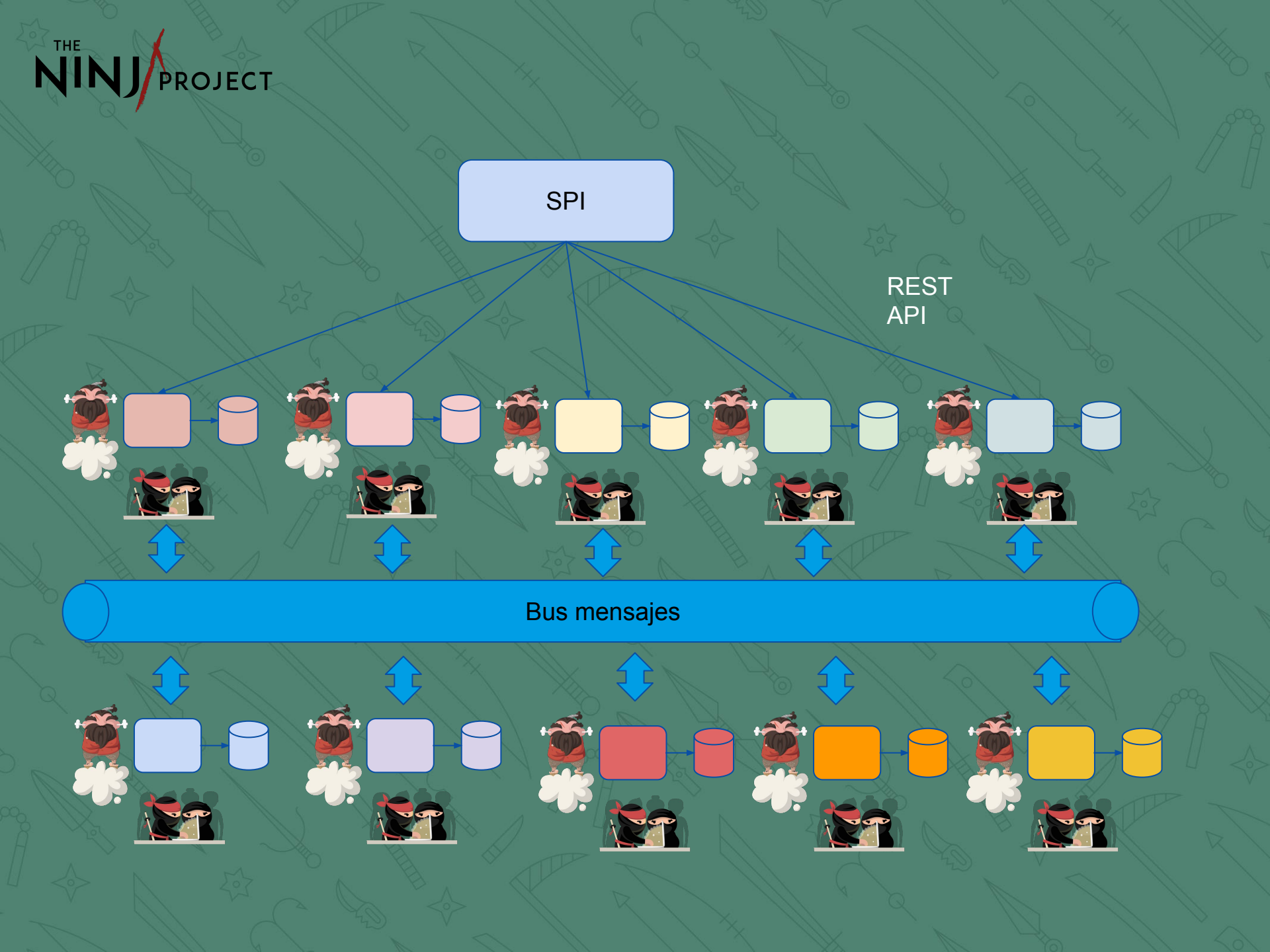
Entorno central de pruebas

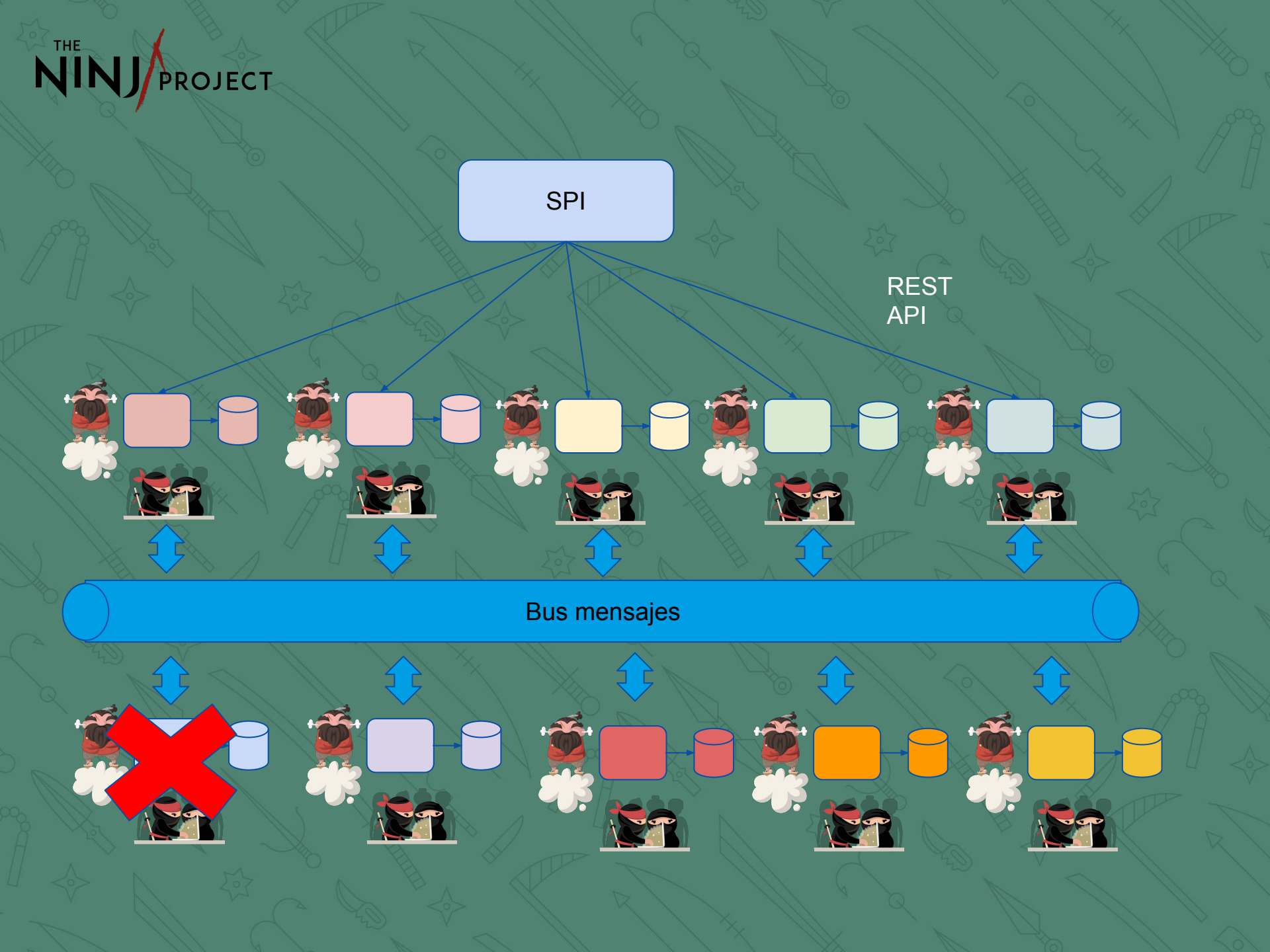


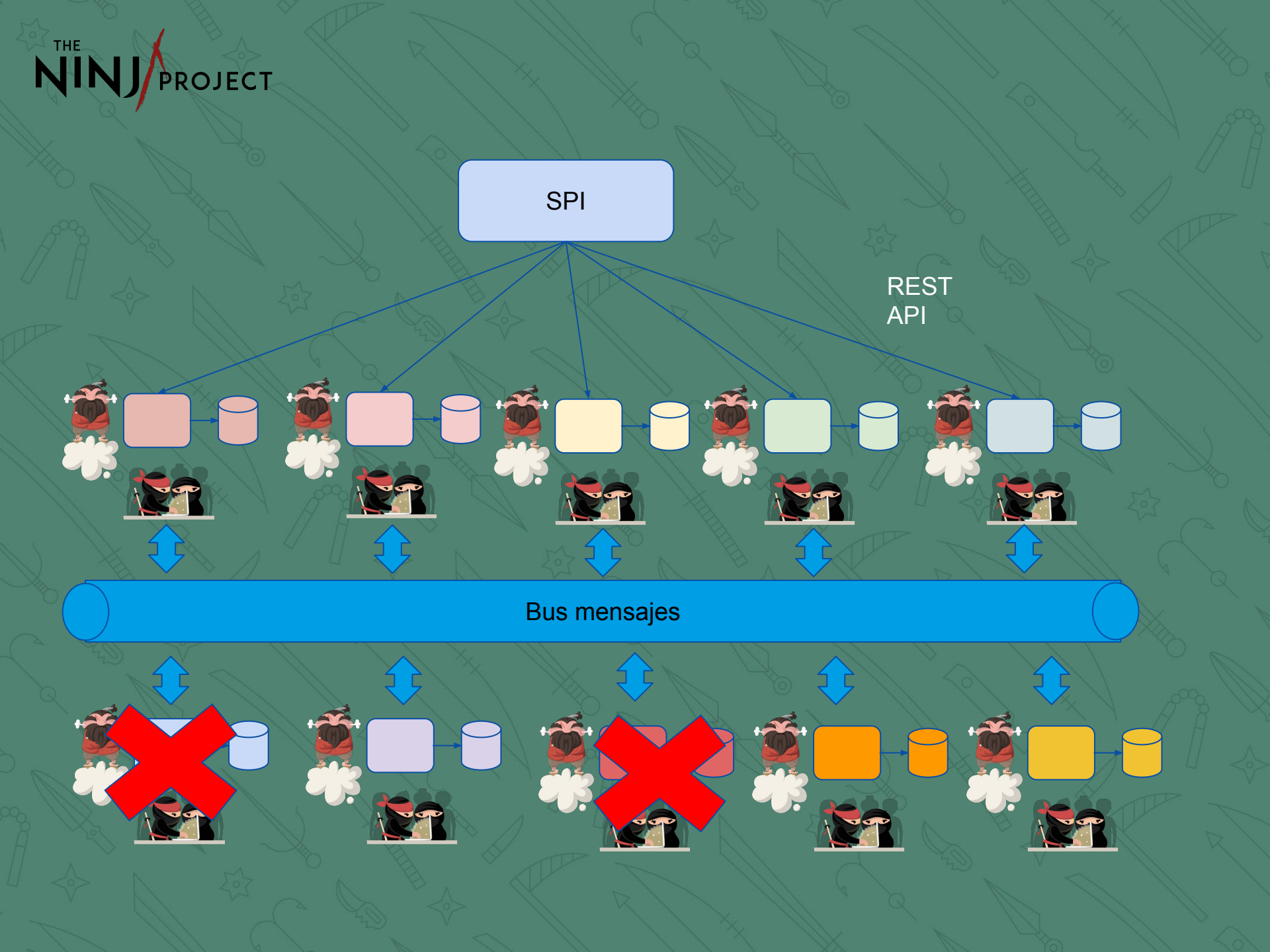
THE NINJA PROJECT

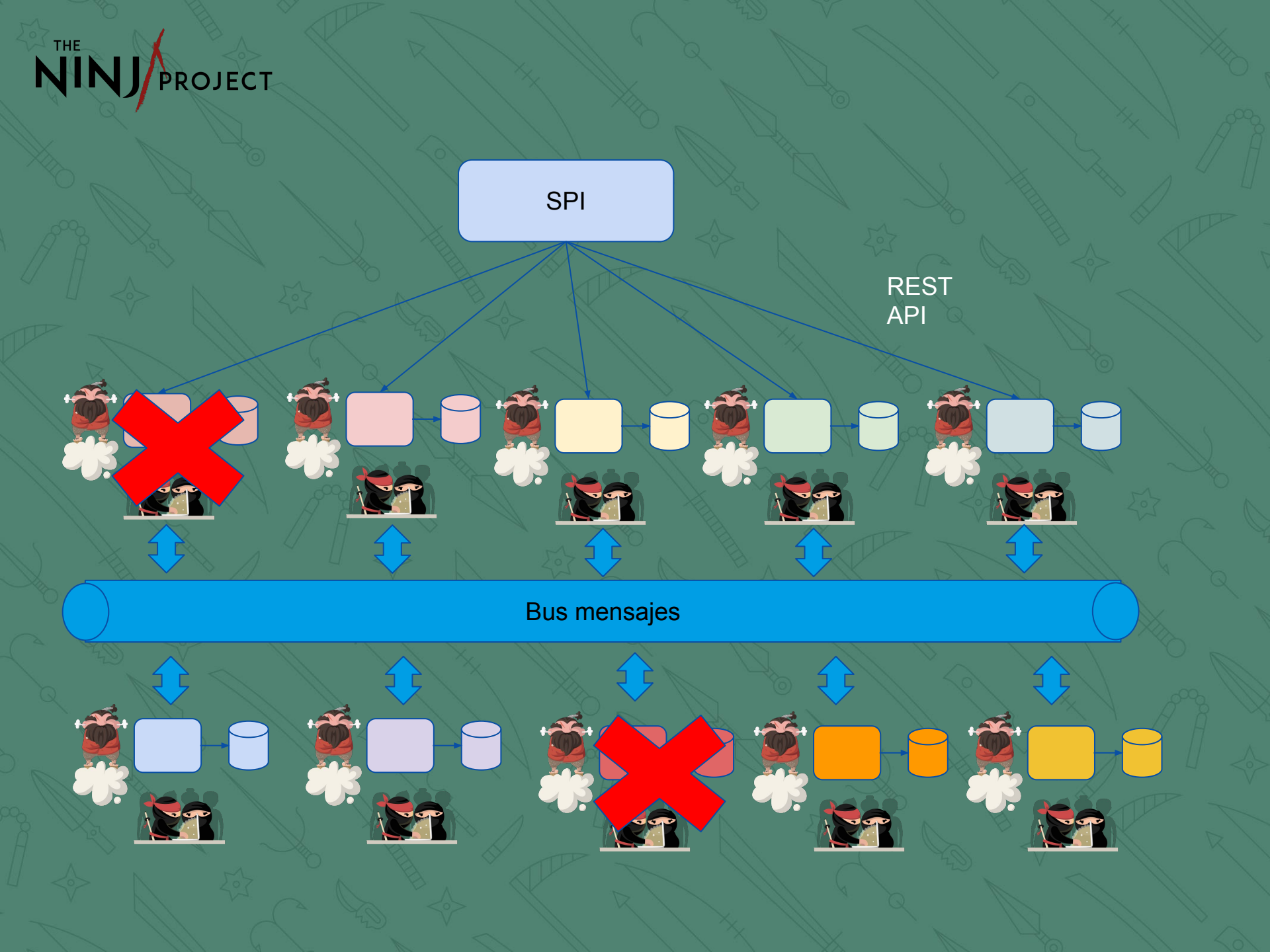
Conway's law

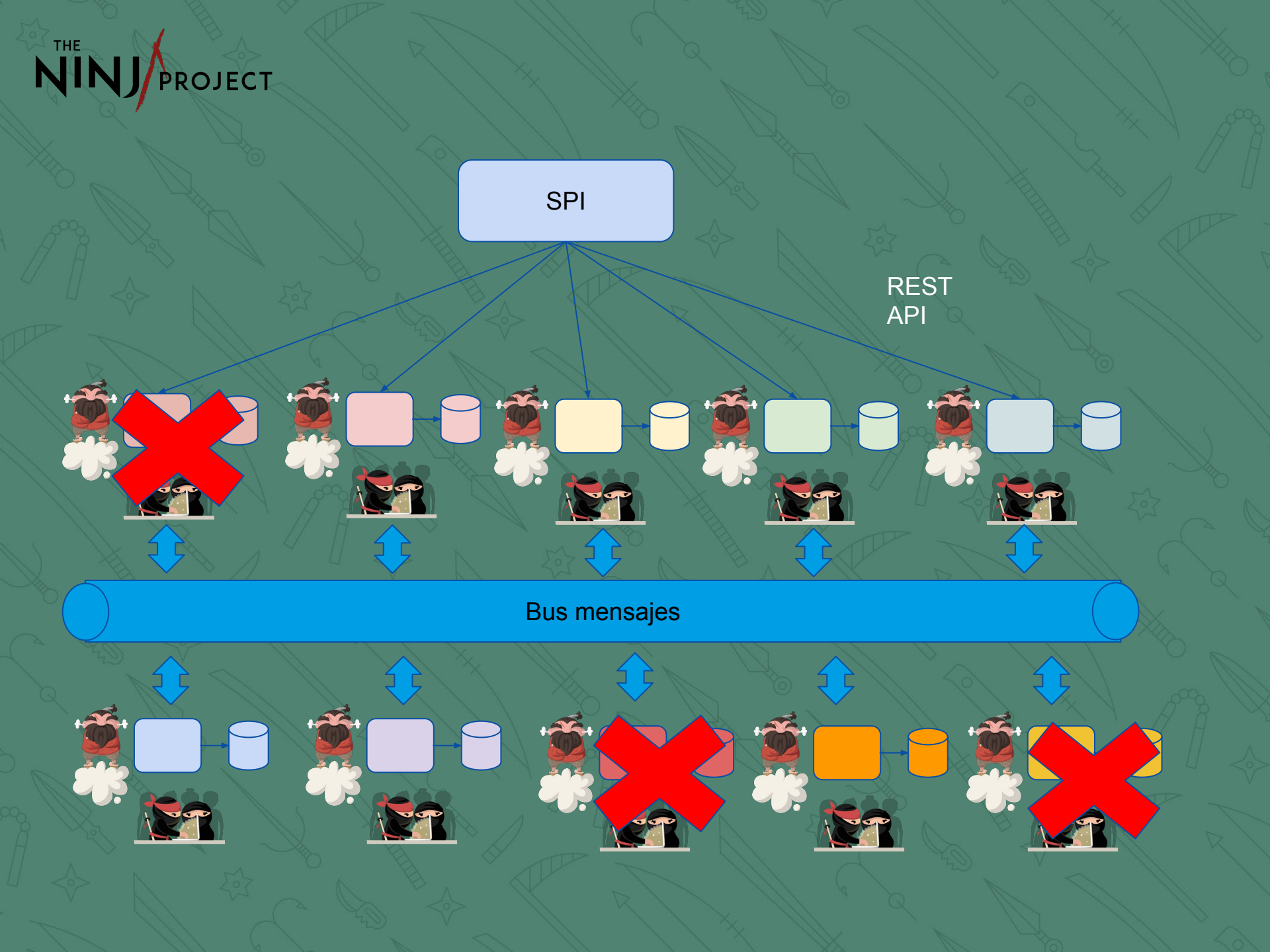
Organizations which design systems ... are constrained to produce designs which are copies of the communication structures of these organizations


THE NINJA PROJECT
SPI
REST API
Bus mensajes


THE NINJA PROJECT
SPI
REST API
Bus mensajes


THE NINJA PROJECT
SPI
REST API
Bus mensajes


THE NINJA PROJECT
SPI
REST API
Bus mensajes


THE NINJA PROJECT
SPI
REST API
Bus mensajes

THE NINJA PROJECT

Un sistema de microservicios normalmente no falla al 100%, tiene diferentes niveles de fallo.

Es básico diseñar pensando que ese fallo puede ocurrir.

THE NINJA PROJECT

¿Cómo detecto un fallo?

- Sistemas de logs centralizados
  - ELK(Elastic logstash kibana)
  - Splunk
  - Logtrust
- Sistemas de métricas
  - InfluxDB+ Grafana
  - Ruxit
  - App dynamics
- Semantic monitoring

THE NINJA PROJECT

¿Cuáles son los beneficios?

- Permiten diseño evolutivo.
- Generan límites de los módulos definidos.
- Habilitan el despliegue independiente.
- Pueden operarse de manera independiente.
  - Escalado.
  - Seguridad.
- Pueden utilizar diversas tecnologías que permiten decidir cuál es la que mejor se ajusta al problema.

THE NINJA PROJECT

¿Y los costes?

- Eventual consistent
  - No existe transaccionalidad distribuida
- Distribuidos
  - CAP
  - Latencias
- Mayor complejidad en la operación

THE NINJA PROJECT

Muchas gracias

Referencias (Libros)

- Building Microservices: Designing Fine-Grained Systems. Sam Newman.
- Release It!: Design and Deploy Production-Ready Software (Pragmatic Programmers). Michael T. Nygard.
- Domain driven design. Eric Evans.
- Refactoring Databases: Evolutionary Database Design. Scott J Ambler, Pramod J. Sadalage

THE NINJA PROJECT

Referencias (Links)

- https://martinfowler.com/microservices
- https://martinfowler.com/articles/microservices.html
- https://martinfowler.com/bliki/SacrificialArchitecture.html
- http://www.reactivemanifesto.org/
- https://martinfowler.com/articles/microservice-trade-offs.html
- https://martinfowler.com/articles/microservice-testing/
- https://martinfowler.com/bliki/MicroservicePremium.html
- https://github.com/heynickc/awesome-ddd

THE NINJA PROJECT

Referencias (Talks)

- https://www.infoq.com/presentations/microservices-replaceability-consistenc
- https://www.infoq.com/presentations/evolutionary-architecture-microservices-cd
- https://www.thoughtworks.com/insights/blog/microservices-nutshell
- https://www.thoughtworks.com/insights/blog/microservices-lessons-frontline