Estrategia de multitenancy a nivel de aislamiento de datos

Context and Problem Statement

Queremos aislar la información entre tenants por cumplimiento legal y compartir la infraestructura para optimización de costos.

Decision Drivers

- El SaaS está construido bajo una arquitectura orientada a microservicios.
- Integridad de datos, una empresa no debe ver los datos de otra empresa nunca.
- Debe tener una forma fácil de medir el consumo de recursos por tenant.
- Debe garantizar que la multitud de gente trabajando en un tenant no afecte a los demás.
- Debe estar en la capacidad de dar backup por tenant.
- Cada bounded context tiene recursos independientes.

Considered Options

- Motor de base de datos por tenant.
- Motor de base de datos por bounded context, con una sola base de datos lógica compartida.
- Motor de base de datos por bounded context, con base de datos lógica por tenant.

Decision Outcome

Opción escogida: La opción que más nos conviene es ***Motor de base de datos por bounded context, con bases de datos lógicas compartidas*** con la variación de tener varios motores de bases de datos, de ser necesario. Al igual que en el caso donde un tenant nos obligue a tener los datos aislados a nivel lógico, se pueda crear una base de datos lógica "compartida" pero dedicada a un solo tenant.

Consequences

- Al momento de crear un tenant debe propagarse la creación en todos los bounded context que apliquen.
- Cada bounded context debe administrar su motor de base de datos.
- El aislamiento de los datos debe garantizarse a nivel de aplicación.

Pros and Cons of the Options

Motor de base de datos por tenant

En este esquema cada tenant tiene su propio motor de base de datos y la base de datos lógica es compartida por todos los bounded context.

- Bueno, independizar el costo del tenant.
- Bueno, no se comparten recursos por tenant.
- Bueno, escalamiento vertical por tenant.
- Bueno, aislamiento de datos a nivel físico por tenant.
- Bueno, se puede tener backup por tenant independiente.
- Bueno, sacar información que cruce datos de varios microservicios es fácil.
- Neutral, se necesita descubrir la cadena de conexión en cada petición.
- Malo, las migraciones, se tienen que correr las migraciones en más lugares y esto implica mayor riesgo a daños.
- Malo, administración de las bases de datos se vuelve extenso (ej permisos).
- Malo, el costo de infraestructura es mayor.
- Malo, no hay aislamiento de datos entre bounded contexts.

Motor de base de datos por bounded context, con bases de datos lógicas compartidas

En este esquema cada bounded context tiene su motor de base de datos independiente donde se alojan los datos de todos los tenant en una sola base de datos lógica.

- Bueno, se garantiza el aislamiento de datos entre bounded contexts.
- Bueno, en caso de recuperación de información, solo habría que recuperar la información de un bounded context.
- Neutral, administración de las bases de datos se hace por bounded context.
- Malo, la reportería que implique cruce de información entre bounded context se tiene que solucionar por aplicación.

Motor de base de datos por bounded context, con base de datos lógica por tenant.

En este esquema cada bounded context tiene un motor de base de datos independiente y cada tenant tiene una base de datos lógica.

- Bueno, es el mejor aislamiento posible, tanto de bounded context como por tenant.
- Bueno, no hay cuellos de botella ocasionados entre bounded context.
- Bueno, backup por bounded context y tenant.
- Bueno, medición de recursos por tenant y bounded context.

- Bueno, administración de permisos granular por tenant y bounded context.
- Malo, descubrimiento de cadenas de conexión por tenant.
- Malo, la administración de las bases de datos es inmensa.
- Malo, los costos se incrementan considerablemente.

More Information

- https://medium.com/@seetharamugn/complete-guide-to-multi-tenant-architecture-d69b24b518d6
- https://learn.microsoft.com/en-us/azure/architecture/guide/saas-multitenant-solution-architecture/
- https://learn.microsoft.com/en-us/azure/architecture/guide/multitenant/approaches/overview