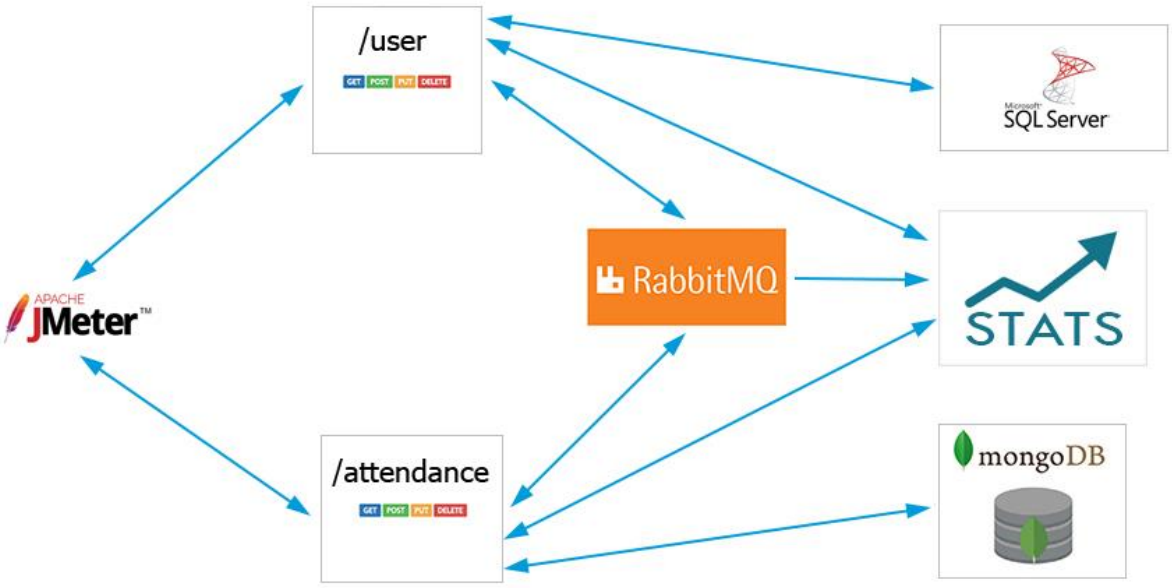
Attendance



- Se tienen 2 servicios (Rest APIs)
- El cliente es un simple consumidor del Rest API (NO UI)
- Para el API de usuarios se tiene la siguiente funcionalidad:

  - Crear usuario (id, nickname, nombre real, etc.)
  - Eliminar usuario (por por id)
  - Listado simple de usuarios (id, nickname, **incluyendo la cantidad de asistencia**)
  - Búsqueda de usuarios (por nickname, y nombre)
  - Detalle completo un de usuario :id, nickname, nombre real, **incluyendo la lista de todas las asistencias.**

- Para el api de asistencia se tiene la siguiente funcionalidad

  - Crear asistencia (id, hora inicio, hora fin, fecha, notas)
  - Eliminar asistencia (por id, solo lo puede hacer el usuario que creo la asistencia)
  - Listado simple de asistencias

- Los servicios pueden hablar entre sí, pero **NO pueden acceder a los repositorios (DB) del otro.**
- La DB para “usuarios” deber una DB Relacional SQL, y para los “asistencia” debe ser una DB NoSQL.
- El “listado de usuarios” es uno de los features más requeridos, y observa que la información necesaria implica consultas a las 2 DBs, definiendo que por motivos de rendimiento la DB de usuarios exista un “**campo pre calculado**” “TotalAttendance” que refleja la cantidad de asistencia de un usuario. Este campo debe ser actualizado cuando existan cambios en la asistencia.

- Un usuario puede ser eliminado, al igual que la asistencia, debiendo mantenerse la consistencia en ambas tablas.
- Para el valor precalculado de “TotalAttendance” debe existir un servicio API interno “Stats”, el cual puede realizar la sincronización y actualización de tablas, sugiriendo además utilizar una COLA de eventos para dicho fin.

NOTA:
Pueden conectarse a una DB SQL y NO SQL de su elección (uds. eligen).
Pueden usar la tecnología y patron de abstraccion que gusten (ORM, Mapper, DAO, etc.).
El trabajo es individual y debe ser publicado en GIT y defendido/ilustrado por cada alumno.