SACC - Architecture

Boulat Pierre-Antoine
Masia Sylvain
Montoya Damien
Peres Richard
Rigaut François

2020 – 2021

Architecture

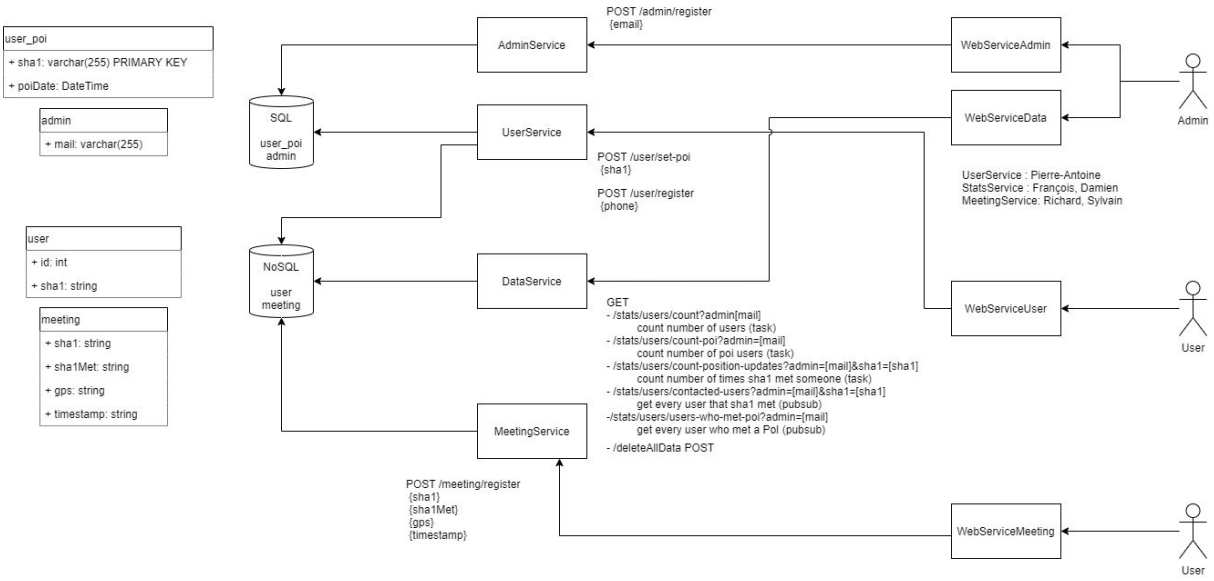

Figure 1 : Architecture du projet (Voir en grand)

Par rapport à l'architecture du projet nous avons effectué divers choix.

Concernant ceux des bases de données, nous avons choisi d'utiliser deux bases de données. La première est une base PostgreSQL qui permet de stocker la liste des administrateurs ainsi que la liste des "Person of Interest", aussi appelés "utilisateurs POI" dans notre architecture. La seconde base de données est un Datastore NoSQL qui permet d'enregistrer tous les utilisateurs ainsi que les meetings, qui sont les points de rencontre en chaque utilisateur. Nous avons choisi de stocker les administrateurs dans une base de données SQL car nous les avons jugées comme sensibles. En effet, dans un cadre relationnel il est facile de vérifier qu'il n'y a qu'un seul administrateur avec un même mail. De plus, il n'y aura pas beaucoup d'administrateurs dans le système, ce qui nous permet d'utiliser une base de données moins flexible que du NoSQL. Ce raisonnement est également valable pour les utilisateurs POI. En effet, nous voulons nous assurer d'une certaine cohérence des données qui s'avère plus pratique avec du relationnel. Également, nous avons estimé que par rapport au nombre d'utilisateurs non POI, il y aura beaucoup moins d'utilisateurs POI. C'est pourquoi nous pensons que la scalabilité de la base de données PostgreSQL pourrait suffire à stocker les données. Enfin, ces dernières ne seront utilisées que lors du calcul de statistiques qui ne se fera pas de manière excessive . Bien entendu, si le système venait à accueillir beaucoup plus d'utilisateurs POI, on pourrait les stocker dans un Datastore pour plus de performances. Concernant la deuxième base de données, nous avons choisi d'y stocker les utilisateurs non POI ainsi que leur meeting. En effet, il risque d'y avoir beaucoup d'utilisateurs ainsi que beaucoup de meeting. Il faudra donc une base de données scalable pour pouvoir répondre à toutes les requêtes.

Au niveau des services nous avons donc séparé le projet en plusieurs couches. La partie WebService permet de récupérer les données du client qu'il transmet ensuite à un autre service, soit directement pour un traitement rapide, soit en utilisant une Cloud Task pour un traitement un peu plus long ou enfin un Pub/Sub pour un traitement lourd. Le

WebServiceData est le seul à utiliser les CloudTask ou les Pub/Sub. En effet, les tâches d'enregistrement d'utilisateur, d'administrateur et de meeting nécessitent une réponse directe à l'utilisateur. En effet, le calcul des statistiques pouvant être long peut être différé. Nous avons décidé d'utiliser des CloudTask seulement pour les statistiques retournant un nombre. Pour les statistiques retournant les entités, ce qui peut mettre plus de temps à être traiter, nous utilisons dans ce cas des Pub/Sub.

Fonctionnalités

AdminService

POST /admin/register - body : {email}
    -> permet d'ajouter un administrateur à la base de données (SQL)
GET /admin/find-all
    -> permet de récupérer la liste de tous les administrateurs

UserService

POST /user/set-poi - body : {sha1}
    -> permet d'ajouter un utilisateur POI (SQL)
POST /user/register - body: {phone}
    -> permet d'ajouter un utilisateur (NoSQL)
GET /user/find-all
    -> permet de récupérer la liste de tous les utilisateurs
GET /user/find-all-poi
    -> permet de récupérer la liste de tous les utilisateurs POI

MeetingService

POST /meeting/register - body : {sha1, sha1Met, gps, timestamp}
    -> permet d'ajouter un meeting (NoSQL)
GET /meeting/find-all
    -> permet de récupérer la liste de tous les meetings

DataService

GET /stats/users/count?admin[mail]
    -> compte le nombre d'utilisateurs (Cloud Task)
GET /stats/users/count-poi?admin=[mail]
    -> compte le nombre d'utilisateurs POI (Cloud Task)
GET /stats/users/count-position-updates?admin=[mail]&sha1=[sha1]
    -> compte le nombre de fois où l'utilisateur sha1 a rencontré quelqu'un (Cloud Task)
GET /stats/users/contacted-users?admin=[mail]&sha1=[sha1]
    -> donne les utilisateurs rencontrés par l'utilisateur sha1 (Pub/Sub)

GET /stats/users/users-who-met-poi?admin=[mail]

-> donne les utilisateurs ayant rencontré PoI (Pub/Sub)

DELETE /stats/users/delete-all

-> seulement pour le rendu, permet de supprimer toutes les données (admins y compis)