# Base de données voluptuaria

## Nos besoins au niveau de la base de données

- Avoir des profils utilisateurs
- Pouvoir à partir de cette base de données proposer des recommandations et parcours de visite
- Avoir des commentaires propres à l'application sur des lieux
- Permettre aux utilisateurs d'avoir des wishlist de lieux
- Pouvoir enregistrer des références de lieu et les différencier
- Pour le moment afin de récupérer nos lieux nous nous basons sur l'API de Google, mais l'application et donc la base de donnée est réflechie pour pouvoir fonctionner à partir de différentes sources de données

## Cas précis d'enregistrement d'un lieu

> Etant donné que l'objectif n'est pas d'extraire la base de données des APIS utilisées actuellement et autres sources possibles dans le futur, nous enregistrons le moins possibles les lieux. Quand nous le faisons nous enregistrons uniquement dans un format JSON défini au niveau du back end les données nécessaires à la récupération du lieu.

- Un utilisateur a visité ce lieu (permet également le commentaire sur le lieu)
- Un utilisateur a mis ce lieu dans sa wishlist

## Critères pour la recommandation (de parcours comme de lieu simple)

```
"types": [
    string
],
```

### Pour un utilisateur non authentifié

- On se base sur la catégorie des lieux les plus recherché

### Pour un utilisateur authentifié

Un lien entre :

- Les catégories des lieux aimés (les recherches fréquentes sont considérés comme lieux aimés)
- Les catégories des lieux non aimés

- Les catégories de lieux récupérés à partir des réseaux sociaux
- Les catégories de lieux visités ou souhaités
- Un lien fait avec les autres utilisateurs à partir de ces critères pour proposer les lieux que l'utilisateur courant n'a pas encore visité

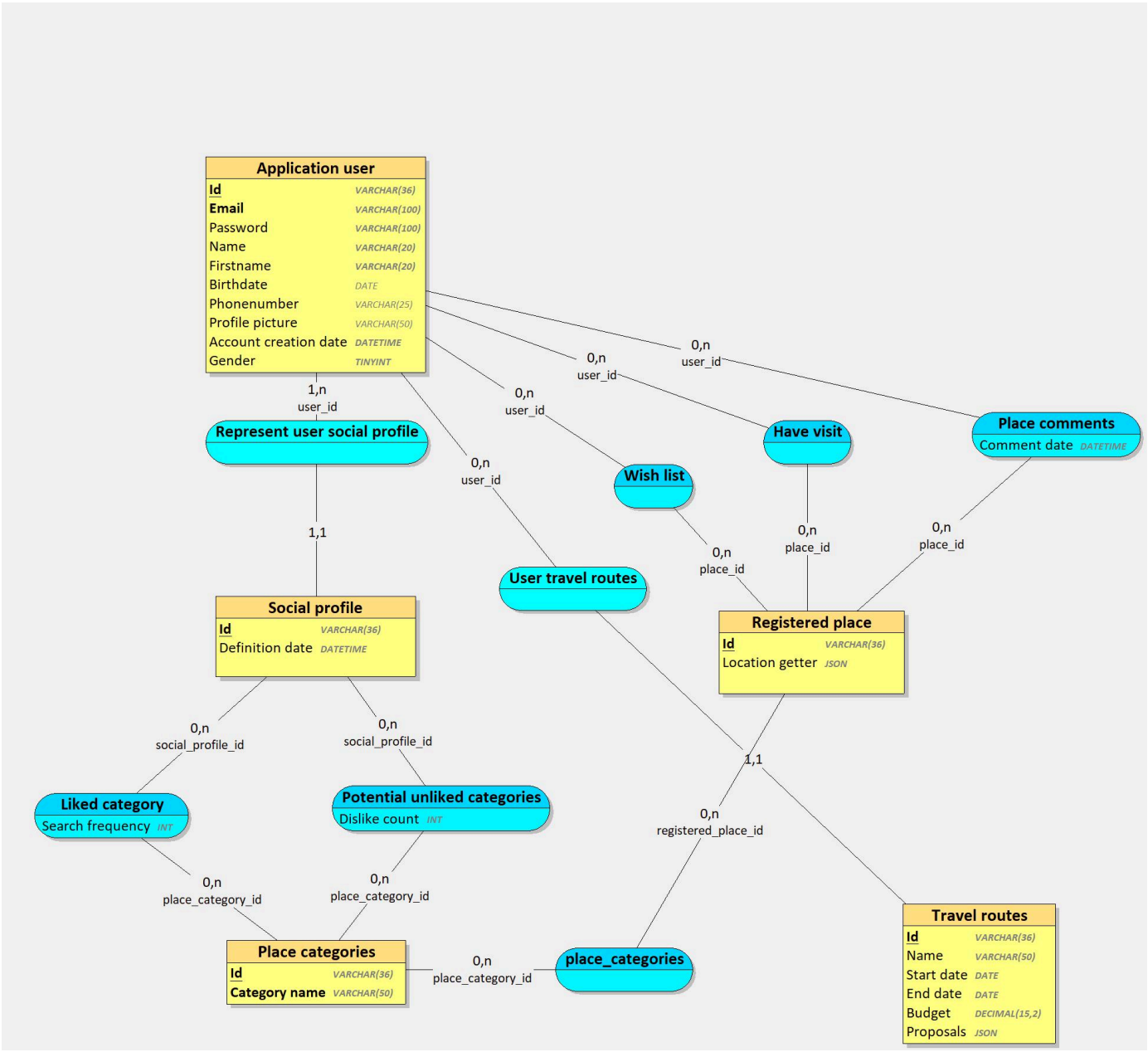
# Choix du système

> Nous étions à la base parti sur du NoSql pensant au volume de données potentiel au fil du temps, mais basé sur les critères cités qui nécessitent des jointures entre les données d'utilisateurs et le fait que les bases de données SQL classiques gèrent bien également le volume de données, nous sommes partis sur un système SQL classique.

## Pourquoi MYSQL

- Suffit pour nos besoins en jointure
- Permet la gestion de données potentiellement volumineuse tout en conservant de bonnes performances via des partitions, peut être utile précisément sur les
  - Lieux enregistrés
  - Les parcours de visites
- En terme de déploiement derrière , étant donné que nous sommes pauvres, MYSQL étant OpenSource et gratuit d'utilisation nous n'aurons pas de coûts supplémentaires

# Schéma tiré des critères


Application user
Id VARCHAR(36)
Email VARCHAR(100)
Password VARCHAR(100)
Name VARCHAR(20)
Firstname VARCHAR(20)
Birthdate DATE
Phonenumber VARCHAR(25)
Profile picture VARCHAR(50)
Account creation date DATETIME
Gender TINYINT
1,n user_id
0,n user_id
0,n user_id
0,n user_id
0,n user_id
Represent user social profile
Have visit
Place comments
Comment date DATETIME
Wish list
1,1
0,n place_id
0,n place_id
0,n place_id
User travel routes
Social profile
Id VARCHAR(36)
Definition date DATETIME
Registered place
Id VARCHAR(36)
Location getter JSON
0,n social_profile_id
0,n social_profile_id
1,1
Liked category
Search frequency INT
Potential unliked categories
Dislike count INT
0,n registered_place_id
0,n place_category_id
0,n place_category_id
Travel routes
Id VARCHAR(36)
Name VARCHAR(50)
Start date DATE
End date DATE
Budget DECIMAL(15,2)
Proposals JSON
Place categories
Id VARCHAR(36)
Category name VARCHAR(50)
0,n place_category_id
place_categories