Rapport - Description de notre modèle : Web de données

Yann Martin d'Escrienne, Yohann Tognetti

I. Les classes de notre modèle

Les classes de notre modèle sont les suivantes :

TripType (et toutes les classes qui en hérite) :
Cette classe et ses descendants représentent tous les types de voyage de l'agence à gros grains. Elle représente les catégories de voyage. Par exemple "Snow" pour tous les voyages en rapport avec la neige, "Bike" pour tous les voyages en vélo etc…
Elle possède comme propriétés : *name* et *hasActivity.*

Activity :
Cette classe représente une activité du site de trekking. Elle est la classe la plus centrale de notre modèle. Les activités représentent des voyages plus précisément.
Elle possède les propriétés suivantes : *name, description, evaluation*, *price, hasDestination* et *hasTravelGroup.*

TravelGroup :
Cette classe représente un groupe de voyage avec des personnes pour une activité. Les groupes se composent généralement d'un guide.
Elle possède les propriétés suivantes : *participantsMaxNumber, departureDate, hasGuide*, *hasClient.*

Country:
Cette classe représente une destination de voyage d'une activité. Dans notre modèle, il s'agit du pays de destination.
Elle possède les propriétés suivantes :*name, description.*

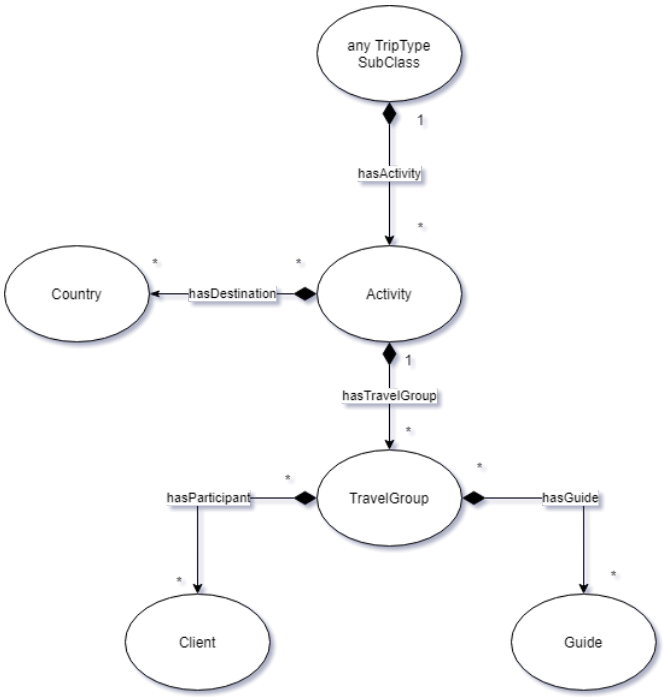
Guide:
Cette classe représente un guide de voyage qui est partenaire du site de trekking. Elle possède les propriétés suivantes :*name, age, familyName.*

Client:
Cette classe représente un client participant à un voyage du site de trekking. Elle possède les propriétés suivantes :*name, age, familyName.*

Person:
Cette classe représente une personne interagissant avec le site de trekking.

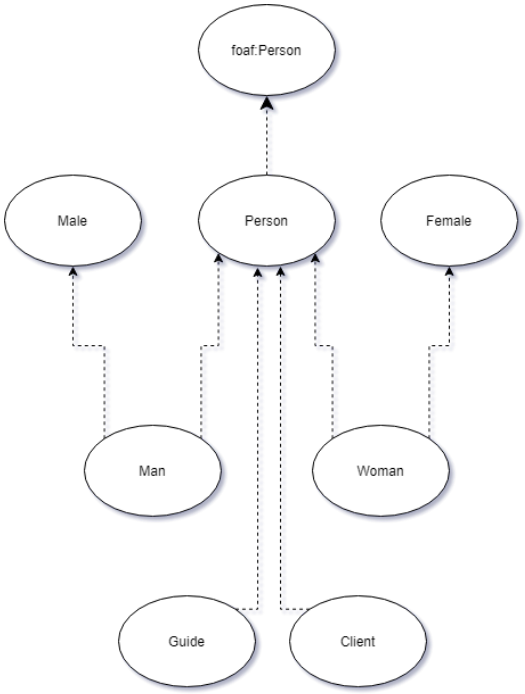
Male et Female:
Ces classes représentent le genre d'une espèce (ici, utilisé seulement pour une personne).

Man et Woman:
Ces classes représentent un homme et une femme interagissant avec le site de trekking.



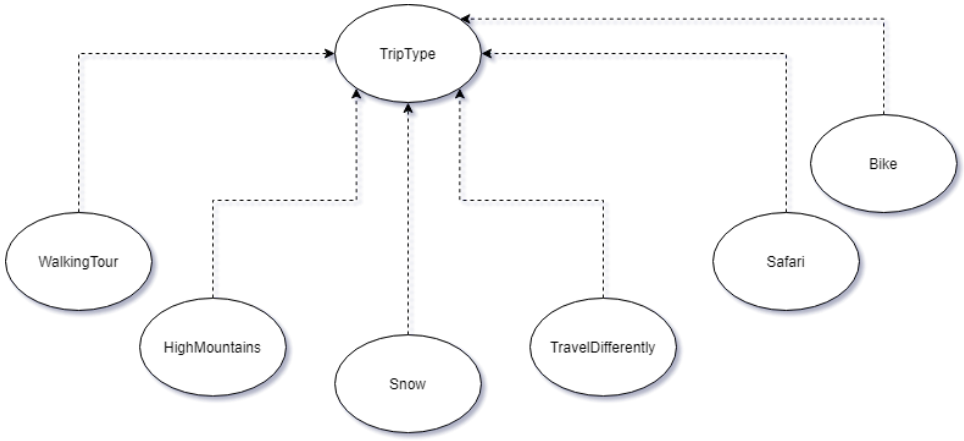
Le diagramme ci-dessus montre les intéractions entre nos classes principales. Il n'y apparaît pas les différents attributs qui ont pour range des valeurs brutes et les classes non principales. On voit donc que les catégories de voyages héritant de **TripType** peuvent avoir plusieurs activités. Que chaque activité à une ou plusieurs destinations et plusieurs groupes qui s'y rattachent. Enfin les groupes sont composés de guides et de clients.

II. Héritage de nos classes



Pour représenter les différents acteurs de notre site de trekking nous nous basons sur ce diagramme de parenté ci-dessus.

Nous partons de la classe **Person** qui hérite de **Friend of a Friend Person** que nous complétons par les classe **Male** et **Female**. Nous créons alors les classes **Man** et **Woman** utilisant les classes précédentes comme parents. Enfin les classes **Guide** et **Client** représente nos acteurs, elles héritent forcément de Person et héritent ensuite de façon optionnelle de **Man** et **Woman**.



Pour représenter les différents voyages de notre site de trekking nous nous basons sur ce diagramme de parenté ci-dessus.

Nous créons tout d'abord une classe mère **TripType** sur laquelle vont se baser toutes les sous-classes ci-dessus. Elle représente chacun une catégorie de voyage dans laquelle viendra s'ajouter un voyage d'un type particulier qui sera une ressource dans notre modèle.

III. Les propriétés de notre modèle

Les propriétés de notre modèle sont les suivantes :

hasTravelGroup : Activity -> TravelGroup
Cette propriété représente l'appartenance d'un groupe de voyage à une activité.

participantsMaxNumber : TravelGroup-> integer
Cette propriété représente le nombre de participants maximum à groupe.

departureDate : TravelGroup -> date
Cette propriété représente la date de départ d'un groupe de voyage.

hasParticipant : TravelGroup -> Person
Cette propriété représente la participation d'une personne à un groupe de voyage.

hasGuide : TravelGroup -> Guide
Cette propriété représente la présence d'un guide dans un groupe de voyage.

hasClient : TravelGroup -> Client
Cette propriété représente la présence d'un client du site dans un groupe de voyage.

hasDestination : Activity -> Country
Cette propriété représente le pays de destination d'une activité.

hasActivity : TripType -> Activity
Cette propriété représente l'appartenance d'une activité à une catégorie de voyage.

evaluation : -> double
Cette propriété représente la note sur 5 d'une activité.

price : -> double
Cette propriété représente le prix d'une activité.

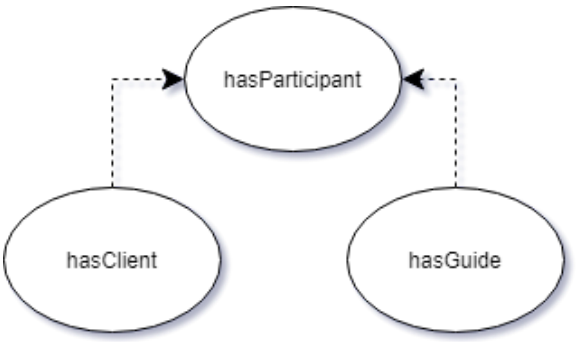
name *hérité de foaf* **: -> string**
Cette propriété représente le nom de la ressource auquelle elle se rattache.

description : -> string
Cette propriété représente la description de la ressource auquelle elle se rattache.

familyName *hérité de foaf* **: -> string**
Cette propriété représente le nom de famille d'une personne.

age *hérité de foaf* **: -> integer**
Cette propriété représente l'âge d'une personne.

IV. Héritage de nos propriétés



Concernant l'héritage de propriétés nous l'utilisons pour *hasParticipant* qui représente le fait qu'on personne soit dans un groupe de voyage. Nous affinons avec les propriétés *hasGuide* et *hasClient* en héritant et qui détermine le rôle d'une personne dans le groupe.

D'autres propriété comme *name*, *age* et *familyName* hérite de leur homonymes respectifs de FOAF.

V. Les requêtes créées pour notre modèle

Voir le fichier SPARQL.txt (ou request.sparql) et ses commentaires.