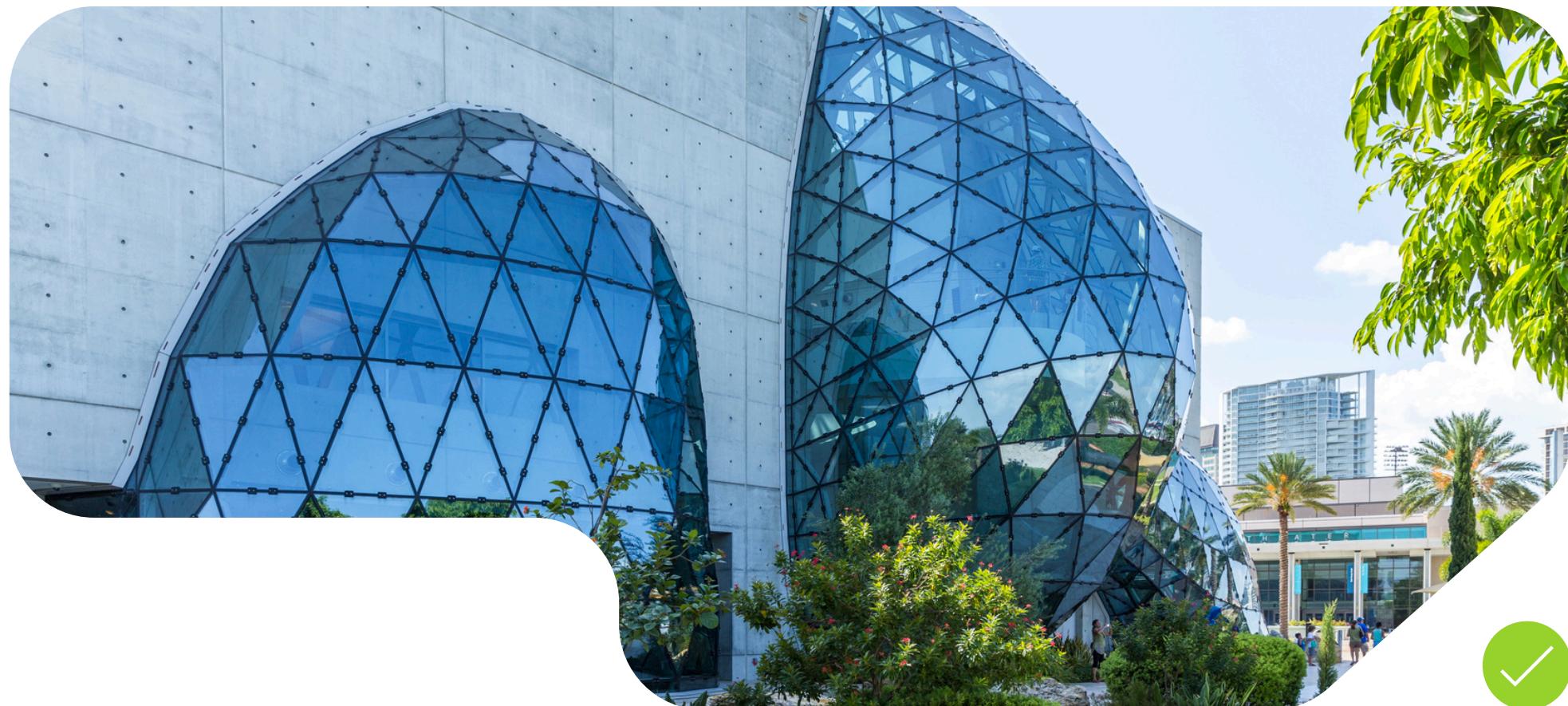


• • •



Developed by
Thunder

smart city



• • •

Notre équipe :



Yassine Mannai



Wael Marouani



Kenza Ben
Slimane



Aymen Jallouli



Nassim Khaldi



Plan

- 1 - Introduction**
- 2- Modélisation des Classes Principales**
- 3- Relations entre les Classes**
- 4- Restrictions OWL**
- 5- Classes Disjointes**
- 6- Types de Propriétés**
- 7- Exemple de Données**
- 8- Conclusion**



Introduction



Contexte du projet

- Objectif : modéliser les interactions entre citoyens, transports et événements de circulation dans une ville intelligente.
- Outil : Protégé (OWL) pour la modélisation ontologique.
- Thème : **Smart City et Mobilité durable**.

Pourquoi une ontologie ?

- Permet de structurer la connaissance sur la mobilité urbaine.
- Facilite l'interopérabilité entre systèmes (bus, capteurs, trajets...).
- Base pour des applications intelligentes : planification, analyse du trafic, etc.



Modélisation des Classes Principales :

Transport

Bus

Métro

Vélo

VoiturePartagée

Trottinette

Utilisateur

Citoyen

Touriste

Station

StationMétro

StationVélo

Parking

ZoneUrbaine

CentreVille

Banlieue

ZoneIndustrielle

EvenementDeCirculation

CentreVille

Accident

Capteur

CapteurTrafic

CapteurQualiteAir

Horaire

HoraireBus

HoraireMetro

Ticket

TicketSimple

AbonnementMensuel

Energie

Electrique

Diesel

Hybride

Trajet

Relations entre les Classes :

utiliseTransport (Utilisateur → Transport)

circuleDans (Transport → ZoneUrbaine)

partDe / arriveA (Trajet → Station)

impacte (EvenementDeCirculation → Trajet)

mesure (Capteur → EvenementDeCirculation / ZoneUrbaine)

aHoraire (Transport → Horaire)

aTicket (Utilisateur → Ticket)

alimentePar (Transport → Energie)

connecteA (Station → Station)

organiseDans (EvenementDeCirculation → ZoneUrbaine)

Restrictions OWL :

1. Utilisateur

- Chaque utilisateur utilise au moins un transport
- Chaque utilisateur possède au moins un ticket

2. Transport

- Chaque transport circule dans au moins une zone urbaine
- Chaque transport est alimenté uniquement par une énergie

3. Bus (sous-classe de Transport)

- Chaque bus a au moins un horaire
- Chaque bus est alimenté par une énergie (par ex. Diesel ou Électrique)

4. Trajet

- Chaque trajet commence par une station
- Chaque trajet arrive à au moins une station

5. EvenementDeCirculation

- Chaque événement impacte au moins un trajet
- Chaque événement est organisé dans une zone urbaine

6. Capteur

- Chaque capteur mesure au moins un événement
- Chaque capteur mesure uniquement des événements de circulation

Classes Disjointes :

1. Moyens de transport

[Bus](#), [Metro](#), [Velo](#), [VoiturePartagee](#), [Trottinette](#)

- Ils doivent être disjoints, car un individu ne peut pas être à la fois un Bus et un Métro.

2. Types d'utilisateurs

[Citoyen](#), [Touriste](#)

- Un utilisateur est soit un citoyen, soit un touriste, mais pas les deux.

3. Éléments urbains

[Station](#), [ZoneUrbaine](#), [Capteur](#), [EvenementDeCirculation](#)

- Une Station n'est pas une ZoneUrbaine, ni un Capteur, ni un Événement.

4. Ressources

[Ticket](#), [Horaire](#), [Energie](#)

- Ce sont des concepts différents et doivent rester disjoints.

Types de Propriétés :

1. connecteA (Station → Station)

Type : Symmetric + Transitive

Justification : si une Station A est connectée à une Station B, alors B est connectée à A. Et si A est connectée à B, et B à C → A est connectée à C.

2. partDe (Trajet → Station)

Type : Functional

Justification : un Trajet part d'une seule station de départ.

3. arriveA (Trajet → Station)

Type : Functional (ou max 1)

Justification : un Trajet arrive dans une seule station d'arrivée.

4. utiliseTransport (Utilisateur → Transport)

Type : InverseOf de estUtilisePar (Transport → Utilisateur).

Justification : si un utilisateur utilise un bus, ce bus est utilisé par l'utilisateur.

5. circuleDans (Transport → ZoneUrbaine)

Type : Non transitive (car un bus qui circule dans une zone A et A incluse dans B ≠ circule automatiquement dans B).

Types de Propriétés :

6. alimentePar (Transport → Energie)

Type : Functional

Justification : un bus est alimenté par une seule énergie principale (exemple : diesel ou électrique).

7. aHoraire (Transport → Horaire)

Type : Non fonctionnelle (un transport peut avoir plusieurs horaires).

8. aTicket (Utilisateur → Ticket)

Type : Non fonctionnelle (un utilisateur peut avoir plusieurs tickets).

9. impacte (EvenementDeCirculation → Trajet)

Type : Non transitive (un accident impacte un trajet précis, mais pas indirectement tous les trajets).

10. mesure (Capteur → EvenementDeCirculation)

Type : Functional (un capteur mesure un seul événement à la fois).

Exemple de Données :

[Utilisateur_1](#)

→ aNom = “Ali”, aAge = 25, aEmail = “ali@gmail.com”

[Bus_101](#)

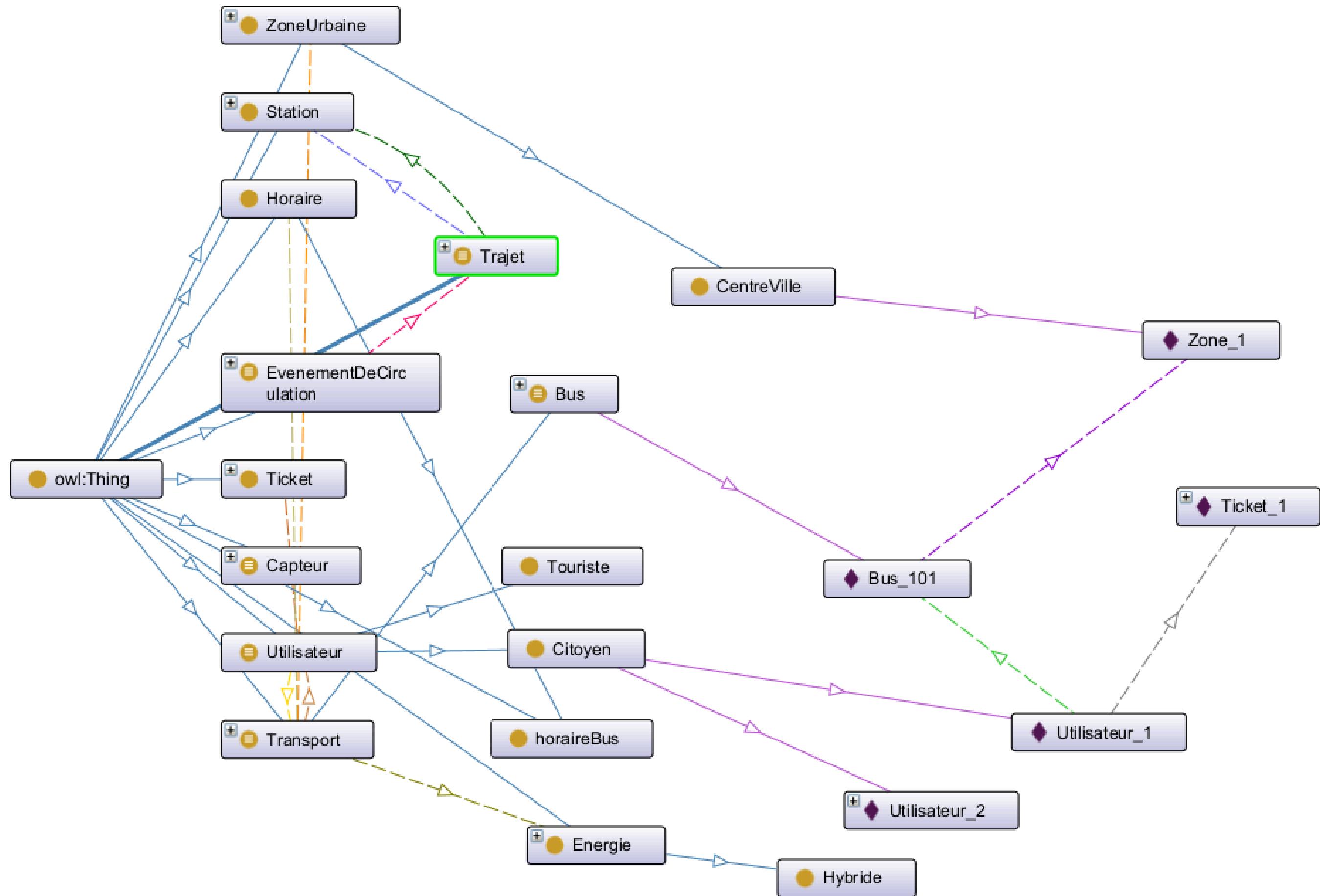
→ aCapacite = 50, almmatriculation = “BUS-101-TN”, estElectrique = false

[Station_BabElBhar](#)

→ aNomStation = “Bab El Bhar”, aLatitude = 36.80, aLongitude = 10.18

[Ticket_1](#)

→ aPrixTicket = 1.2, aDateEmission = “2025-09-27T08:00:00”, aValide = true



conclusion



En conclusion, cette ontologie offre une représentation claire et cohérente des interactions entre utilisateurs, moyens de transport et zones urbaines. Elle constitue une base réutilisable pour les systèmes de Smart Mobility, en facilitant l'intégration des données et une mobilité plus fluide et durable.

Merci

