SAE 3-01

Collecte automatisée de données web

Cavory--Dupuis Antonin et Phuc Anh Dang
26/01/2026

Table des matières

1. Présentation de l'outil 2
2. Environnement, périmètre, hypothèses 2
3. Collecte des données (API / fichiers) 2
3.1 Hôtels (OpenDataSoft — OSM tourism accommodation) 2
3.2 Restaurants (OpenDataSoft — OSM food service) 2
3.3 Lieux historiques (OpenDataSoft — OSM historic) 3
3.4 Trajet (OpenRouteService) 3
3.5 Météo actuelle (Open-Meteo) 3
4. Développement : modules et classes 3
4.1 Fiches classes (attributs & méthodes) 3
HotelClients (hotels_rendu.py) 3
RestaurantClients (restaurants_rendu.py) 4
MuseeClients (musees_rendu.py) 5
CartesFoliumClients (cartes.py) 5
MeteoClients (route_rendu.py) 6
RouteClients (route_rendu.py) 7
5. Diagramme de classes (texte UML) 8
6. Résultats et démonstration (outputs) 9
8. Conclusion et perspectives 10

1. Présentation de l'outil

L'application « MonVoyage » est un notebook Jupyter qui génère un séjour à partir d'une ville de départ, d'une ville d'arrivée, d'une date de départ, d'une durée et d'un nombre de personnes. Elle agrège des données externes (API) et des données en fichiers CSV, puis produit :

- un itinéraire
- la météo actuelle départ et arrivée
- un hôtel unique pour tout le séjour
- des restaurants et des activités par jour
- des cartes Folium
- une estimation du budget

2. Environnement, périmètre, hypothèses

Environnement : Python (Jupyter Notebook), bibliothèques requests, csv, folium, pandas.

Périmètre : villes françaises (filtre « FR » côté géocodage). Les données POI sont collectées via OpenDataSoft (OSM). Le trajet est calculé via OpenRouteService. La météo est fournie par Open-Meteo.

Hypothèses et contraintes :

- Les fichiers CSV sont nommés par ville (ex : hotels_lille.csv) et stockés dans le dossier du projet.
- Les API imposent des limites (pagination par lots de 100 sur OpenDataSoft).
- L'estimation budgétaire repose sur des moyennes nationales (repas, nuit d'hôtel) et une estimation simplifiée des péages.
- La génération du planning est aléatoire (sélection d'un hôtel et choix d'un ensemble de restaurants et activités).

3. Collecte des données (API / fichiers)

3.1 Hôtels (OpenDataSoft — OSM tourism accommodation)

Source : dataset osm-france-tourism-accommodation. Filtre : refine meta_name_com:<ville>.
Pagination : limit<=100, offset.
Champs exportés (CSV) : nom, type, categorie, site_web, telephone, commune, departement, region, latitude, longitude.

3.2 Restaurants (OpenDataSoft — OSM food service)

Source : dataset osm-france-food-service. Filtre : refine meta_name_com:<ville>.
Pagination : limit<=100, offset.
Champs exportés : nom, type, cuisine, enseigne, vegetarien, vegan, livraison, a_emporter, etoiles_michelin, capacite, telephone, site_web, commune, departement, region, latitude, longitude.

3.3 Lieux historiques (OpenDataSoft — OSM historic)

Source : dataset osm-france-historic. Filtre : refine meta_name_com:<ville>. Pagination : limit<=100, offset.
Champs exportés : nom, type, description, date_construction, heritage, wikipedia, religion, denomination, ref_mhs, commune, departement, region, latitude, longitude.

3.4 Trajet (OpenRouteService)

Deux usages :
• Geocoding : ville -> coordonnées [lon,lat] et libellé
• Directions : itinéraire sous forme GeoJSON (distance, durée, géométrie)
Sorties : GeoJSON du trajet et CSV récapitulatif (distance, durée, coûts estimés, météo départ/arrivée).

3.5 Météo actuelle (Open-Meteo)

API gratuite sans clé. Requête sur latitude/longitude. Récupération : température, pluie, vent, weather_code, timestamp.

4. Développement : modules et classes

Le projet est structuré autour de plusieurs scripts Python et d'un notebook Jupyter qui orchestre l'ensemble.
Modules principaux :
• hotels_rendu.py : classe HotelClients
• restaurants_rendu.py : classe RestaurantClients
• musees_rendu.py : classe MuseeClients
• cartes.py : classe CartesFoliumClients
• route_rendu.py : classes RouteClients et MeteoClients
• rendu_final.ipynb : scénario utilisateur et restitution (planning, cartes, budget).

4.1 Fiches classes (attributs & méthodes)

HotelClients (hotels_rendu.py)

Rôle : collecter les hébergements via OpenDataSoft et produire un CSV.

Attributs :

• ville: str

• timeout: int

• URL_BASE: str (constante)

Méthodes :

• recuperer_hotels(limite=100, decalage=0) -> dict

• recuperer_tous_les_hotels(taille_lot=100) -> list

• prendre_premier_champ_non_vide(enregistrement, cles)

• extraire_lat_lon(enregistrement) -> (lat, lon)

• nettoyer_hotels(resultats_bruts) -> list[dict]

• exporter_csv(nom_fichier, lignes)

• executer()

RestaurantClients (restaurants_rendu.py)

Rôle : collecter les restaurants via OpenDataSoft et exporter un CSV.

Attributs :

• ville: str

• timeout: int

• URL_BASE_RESTO: str (constante)

Méthodes :

• recuperer_restaurants(limite=100, decalage=0) -> dict

• recuperer_tous_les_restaurants(taille_lot=100) -> list

• prendre_premier_champ(enregistrement, cles)

• extraire_lat_lon(enregistrement) -> (lat, lon)

• nettoyer_restaurants(bruts) -> list[dict]

• exporter_csv(nom_fichier, lignes)

• executer()

MuseeClients (musees_rendu.py)

Rôle : collecter les lieux historiques via OpenDataSoft et exporter un CSV.

Attributs :

• ville: str

• timeout: int

• URL_BASE_HISTO: str (constante)

Méthodes :

• recuperer_lieux_historiques(limite=100, decalage=0) -> dict

• recuperer_tous_les_lieux_historiques(taille_lot=100) -> list

• prendre_premier_champ(enregistrement, cles)

• extraire_lat_lon(enregistrement) -> (lat, lon)

• nettoyer_lieux_historiques(bruts) -> list[dict]

• exporter_csv(nom_fichier, lignes)

• executer()

CartesFoliumClients (cartes.py)

Rôle : lire les CSV et générer des cartes Folium HTML.

Attributs :

• ville: str

• dossier_csv: str

• dossier_sortie: str

• csv_hotels: str

• csv_restaurants: str

• csv_lieux: str

Méthodes :

• lire_csv(path) -> list[dict]

• to_float(val) -> float|None

• centre_moyen(lignes) -> (lat, lon)

• popup_hotel(r) -> folium.Popup

• popup_restaurant(r) -> folium.Popup

• popup_lieu(r) -> folium.Popup

• carte_hotels()

• carte_restaurants()

• carte_lieux_historiques()

• carte_globale()

• executer()

MeteoClients (route_rendu.py)

Rôle : récupérer la météo actuelle (Open-Meteo).

Attributs :

• timeout: int

• timezone: str

• BASE_URL: str (constante)

Méthodes :

• get_meteo_actuelle(lat, lon) -> dict

• resume_meteo(m) -> str

RouteClients (route_rendu.py)

Rôle : géocoder, calculer l'itinéraire, estimer le coût, récupérer la météo, exporter GeoJSON/CSV.

Attributs :

• timeout: int

• cache_dir: str

• meteo_client: MeteoClients

• ORS_BASE: str (constante)

• API_KEY: str

• PROFILS: dict

• PRIX_CARBURANT_EUR_L: float

• CONSO_MOY_L_100KM: float

• PEAGE_EUR_100KM: float

Méthodes :

• _get(url, params=None) -> dict

• _post(url, body=None) -> dict

• _cache_path(prefix, key) -> str

• _cache_load(prefix, key, max_age_seconds) -> dict

• _cache_save(prefix, key, data)

• geocoder_ville(ville, country='FR') -> (coords, label)

• calculer_itineraire(coords_from, coords_to, profil='driving-car') -> GeoJSON

• extraire_distance_duree(route_geojson) -> (distance_m, duration_s)

• format_km(distance_m) -> str

• format_min(duration_s) -> str

• estimer_prix_trajet(distance_m, profil) -> dict

• recommander_mode(mode_nom, profil, meteo_depart, meteo_arrivee) -> dict

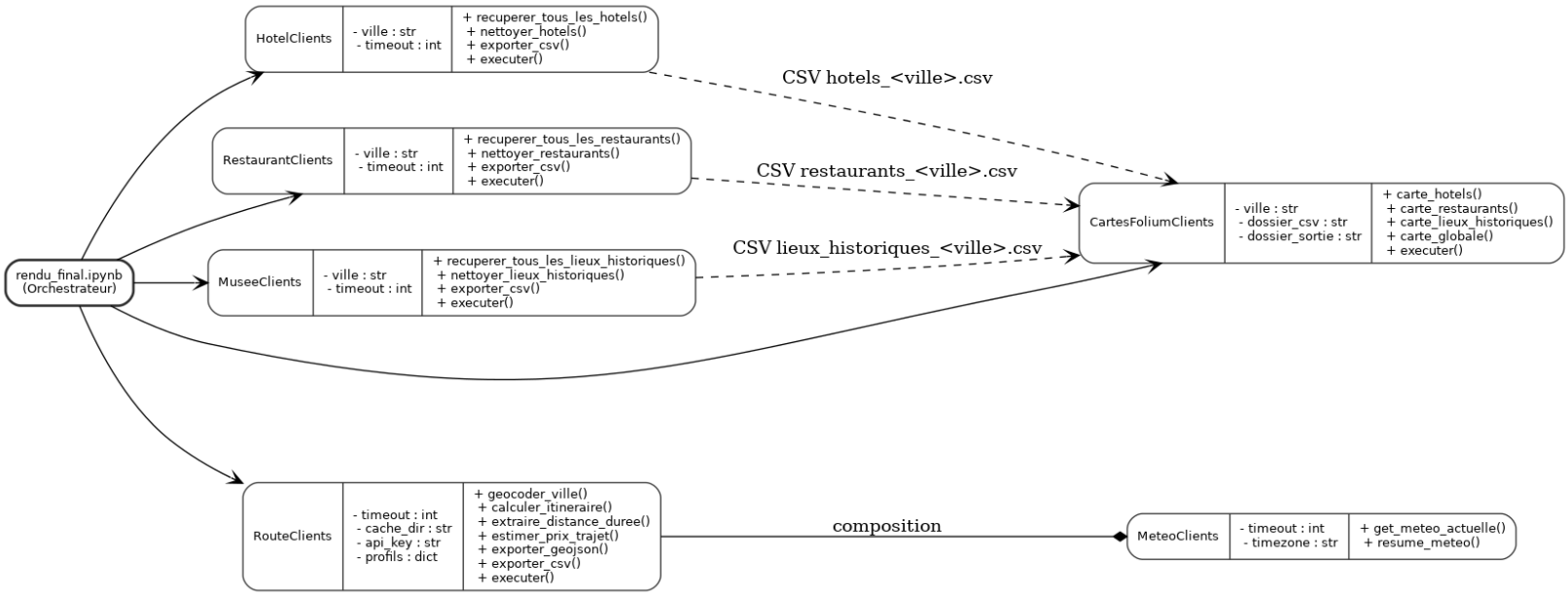
• exporter_geojson(nom_fichier, route_geojson)

• exporter_csv(...)

• executer()

5. Diagramme de classes (texte UML)

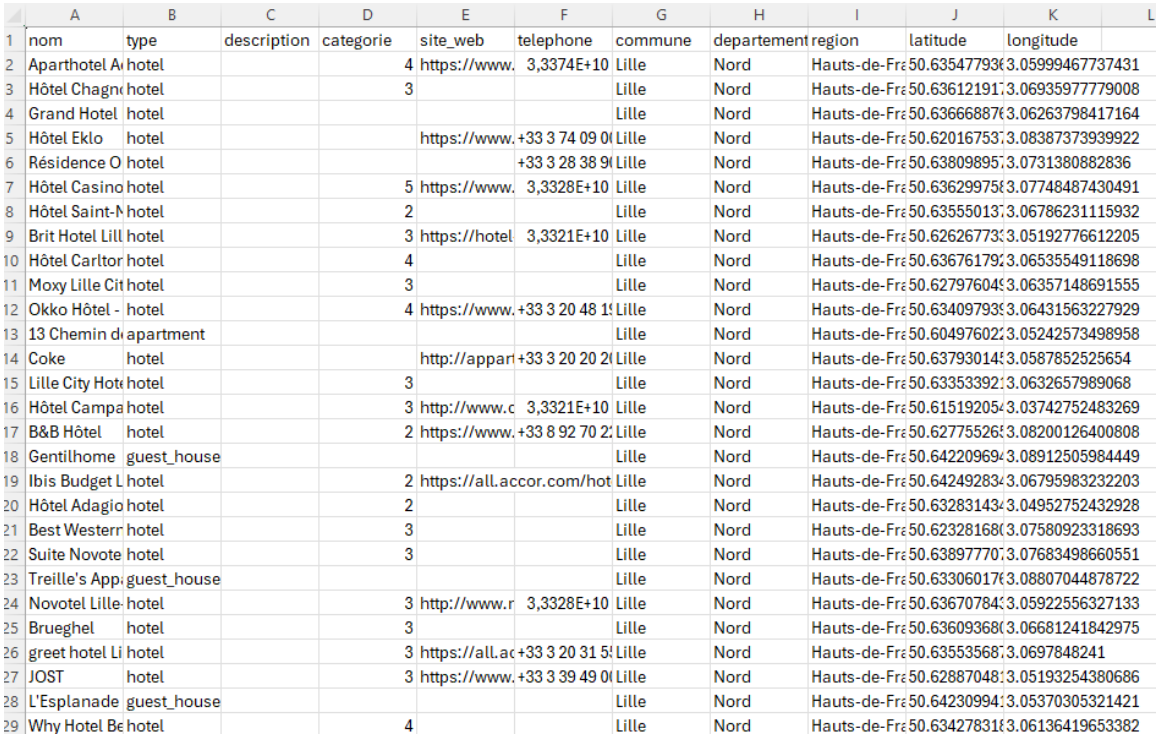
Il représente les relations principales : le notebook orchestre les clients d'API ; RouteClients dépend de MeteoClients ; CartesFoliumClients consomme les CSV produits par HotelClients/RestaurantClients/MuseeClients.



6. Résultats et démonstration (outputs)

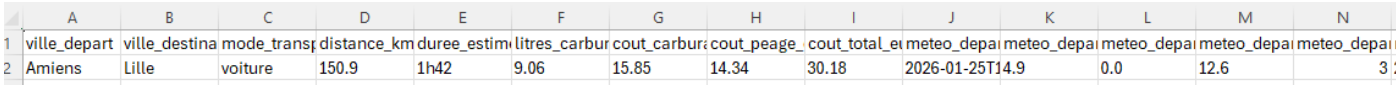
Fichiers produits (exemples) :

• hotels_<ville>.csv, restaurants_<ville>.csv, lieux_historiques_<ville>.csv



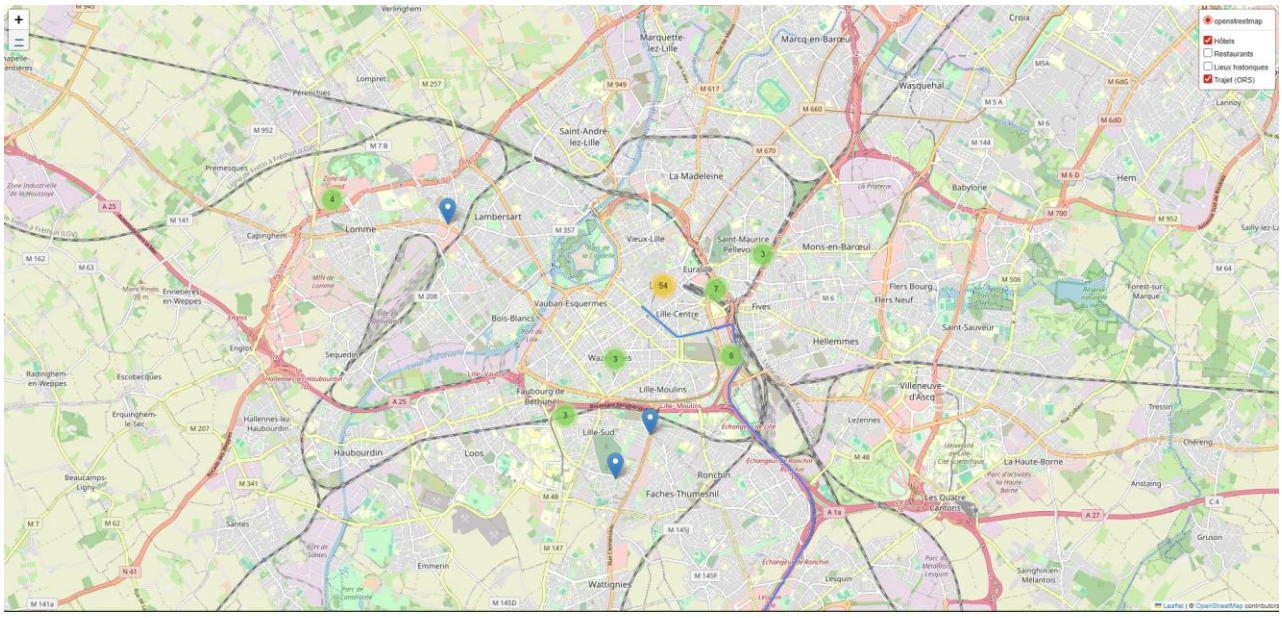
	A	B	C	D	E	F	G	H	I	J	K	L
1	nom	type	description	categorie	site_web	telephone	commune	departement	region	latitude	longitude	
2	Aparthotel A	hotel		4	https://www.	3,3374E+10	Lille	Nord	Hauts-de-Fra	50.63547793	3.05999467737431	
3	Hôtel Chagn	hotel		3			Lille	Nord	Hauts-de-Fra	50.63612191	3.06935977779008	
4	Grand Hotel	hotel					Lille	Nord	Hauts-de-Fra	50.63666887	3.06263798417164	
5	Hôtel Eklo	hotel			https://www.	+33 3 74 09 0	Lille	Nord	Hauts-de-Fra	50.62016753	3.08387373939922	
6	Résidence O	hotel				+33 3 28 38 9	Lille	Nord	Hauts-de-Fra	50.63809895	3.0731380882836	
7	Hôtel Casino	hotel		5	https://www.	3,3328E+10	Lille	Nord	Hauts-de-Fra	50.63629975	3.07748487430491	
8	Hôtel Saint-M	hotel		2			Lille	Nord	Hauts-de-Fra	50.63555013	3.06786231115932	
9	Brit Hotel Lill	hotel		3	https://hotel	3,3321E+10	Lille	Nord	Hauts-de-Fra	50.62626773	3.05192776612205	
10	Hôtel Carlton	hotel		4			Lille	Nord	Hauts-de-Fra	50.63676179	3.06535549118698	
11	Moxy Lille Cit	hotel		3			Lille	Nord	Hauts-de-Fra	50.62797604	3.06357148691555	
12	Okko Hôtel -	hotel		4	https://www.	+33 3 20 48 1	Lille	Nord	Hauts-de-Fra	50.63409793	3.06431563227929	
13	13 Chemin d	apartment					Lille	Nord	Hauts-de-Fra	50.60497602	3.05242573498958	
14	Coke	hotel			http://appar	+33 3 20 20 2	Lille	Nord	Hauts-de-Fra	50.63793014	3.0587852525654	
15	Lille City Hot	hotel		3			Lille	Nord	Hauts-de-Fra	50.63353392	3.0632657989068	
16	Hôtel Campa	hotel		3	http://www.c	3,3321E+10	Lille	Nord	Hauts-de-Fra	50.61519205	3.03742752483269	
17	B&B Hôtel	hotel		2	https://www.	+33 8 92 70 2	Lille	Nord	Hauts-de-Fra	50.62775526	3.08200126400808	
18	Gentilhome	guest_house					Lille	Nord	Hauts-de-Fra	50.64220969	3.08912505984449	
19	Ibis Budget L	hotel		2	https://all.accor.com/hot		Lille	Nord	Hauts-de-Fra	50.64249283	3.06795983232203	
20	Hôtel Adagio	hotel		2			Lille	Nord	Hauts-de-Fra	50.63283143	3.04952752432928	
21	Best Western	hotel		3			Lille	Nord	Hauts-de-Fra	50.62328168	3.07580923318693	
22	Suite Novote	hotel		3			Lille	Nord	Hauts-de-Fra	50.63897770	3.07683498660551	
23	Treille's App	guest_house					Lille	Nord	Hauts-de-Fra	50.63306017	3.08807044878722	
24	Novotel Lille	hotel		3	http://www.r	3,3328E+10	Lille	Nord	Hauts-de-Fra	50.63670784	3.05922556327133	
25	Brueghel	hotel		3			Lille	Nord	Hauts-de-Fra	50.63609368	3.06681241842975	
26	greet hotel Li	hotel		3	https://all.ac	+33 3 20 31 5	Lille	Nord	Hauts-de-Fra	50.63553568	3.0697848241	
27	JOST	hotel		3	https://www.	+33 3 39 49 0	Lille	Nord	Hauts-de-Fra	50.62887048	3.05193254380686	
28	L'Esplanade	guest_house					Lille	Nord	Hauts-de-Fra	50.64230994	3.05370305321421	
29	Why Hotel Be	hotel		4			Lille	Nord	Hauts-de-Fra	50.63427831	3.06136419653382	

• trajet_<...>.geojson et trajet_<...>.csv (route + coût + météo)



	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	ville_depart	ville_destina	mode_transp	distance_km	duree_estim	litres_carbur	cout_carbura	cout_peage_	cout_total_e	meteo_depa	meteo_depa	meteo_depa	meteo_depa	meteo_depa
2	Amiens	Lille	voiture	150.9	1h42	9.06	15.85	14.34	30.18	2026-01-25T1	4.9	0.0	12.6	3

• cartes/carte_hotels_<ville>.html, carte_restaurants_<ville>.html, carte_lieux_historiques_<ville>.html, carte_globale_<ville>.html



• rendu_final.ipynb : restitution (planning Jour 1, Jour 2, … + cartes Folium + budget estimé).

8. Conclusion et perspectives

Bilan : l'application répond au besoin principal (planifier un séjour + données externes + restitution cartographique). Le découpage en clients (collecte), stockage CSV, et notebook (orchestration) rend le projet clair et évolutif.

Améliorations possibles :

• Création automatique des CSV si la ville n'existe pas (pipeline si fichiers absents -> lancer collecte).

• Normalisation des noms de ville (espaces, tirets, virgules).

• Ajout de filtres utilisateur (cuisine, type de lieu, budget).

Antonin

La réalisation de cette SAE m'a permis de consolider mes compétences en programmation Python, notamment en programmation orientée objet et en utilisation d'API. J'ai appris à structurer un projet autour de plusieurs modules, à collecter et exploiter des données provenant de sources variées, puis à les restituer de manière claire à l'aide de visualisations interactives.

Ce projet m'a également aidé à développer une méthode de travail plus rigoureuse, en particulier pour la gestion des données, le débogage et l'organisation du code. Enfin, il m'a permis de mieux comprendre l'importance de la présentation des résultats pour un utilisateur final, au-delà de la simple collecte des données.

Phuc Anh

Ce projet m'a permis de mettre en pratique la programmation orientée objet et l'utilisation d'API REST dans une application concrète. Il repose sur plusieurs services externes, comme le transport SNCF, la météo et les points d'intérêt, chacun étant géré par une classe spécifique afin de structurer correctement le code.

L'intégration de l'API SNCF a été la partie la plus complexe du projet. Elle a demandé une bonne compréhension des réponses JSON, la gestion de l'authentification et la prise en compte des cas où certaines informations, comme les prix, ne sont pas toujours disponibles. Pour éviter les erreurs, des contrôles ont été ajoutés et des valeurs par défaut ont été utilisées.

Ce projet m'a également appris l'importance d'organiser le code de manière claire et logique. La séparation des rôles entre les différentes classes rend le programme plus lisible et plus facile à faire évoluer.

En conclusion, ce travail m'a permis de renforcer mes compétences en Python et de mieux comprendre comment concevoir une application qui combine plusieurs sources de données de façon cohérente.