Interreg
MARITTIMO-IT FR-MARITIME
Fondo Europeo di Sviluppo Regionale
Fonds Européen de Développement Régional

UNIONE EUROPEA
UNION EUROPÉENNE

smart DESTINATION

Rapport: Architecture de l'écosystème numérique transfrontalier (Livrable T2.2.1)

Composant T2 -Conception de l'architecture générale de l'écosystème numérique intégré de SMART DESTINATION

La coopération au coeur de la Méditerranée
La cooperazione al cuore del Mediterraneo

Regione Toscana
Regione Autònoma de Sardigna
Regione Autonoma della Sardegna
Regione Liguria
Métropole Nice Côte d'Azur
ente assistenza tecnica
fondazione sistema toscana
CCI Provence Alpes Côte d'Azur

INDICE

TABLEAU DES ILLUSTRATIONS

TABLEAU GRAPHIQUE

ACRONYMES

Abbreviazione	Description
API (REST)	Application Programming Interface (REpresentational State Transfer)
AWS	Amazon Web Services
BE	Back End (parte gestionale di un'applicazione)
CARF	Communauté d'Agglomération de la Riviera Française
CEF	Connecting Europe Facility
DMS	Document Management System
DMO	Destination Management Organisation
EH	Easy Holiday
FE	Front End (parte pubblica di un'applicazione)
FTP	File Transfer Protocol
GSMA	GSM Association
HIS	Hyperlocal Infopoint Sardegna
IIS	Internet Information Services (Microsoft)
IRPET	Istituto Regionale Programmazione Economica Toscana
JSON	JavaScript Object Notation
MINT	Muoversi IN Toscana
MNCA	Métropole Nice Côte d'Azur
NTIC	Nouvelles Technologies de l'Information de la Communication
OASC	Open & Agile Smart Cities
OT/OTI	Office du Tourisme/Office du Tourisme Intermunicipale
OTCN	Office du Tourisme et des Congrès de Nice
OTM	Office du Tourisme Métropolitain
OTMNCA	Office de Tourisme Métropolitain Nice Côte d'Azur
OVH	Web hosting company francese (fornitore di MNCA)
PA	Public Administration
PME	Piccole e Medie Imprese
POI	Point of Interest

RAS	Regione Autonoma della Sardegna
RL	Regione Liguria
RT	Regione Toscana
SDK	Software Development Kit
SQL	Structured Query Language
SUAPE	Sportello Unico delle Attività Produttive e dell'Edilizia (RAS)
TIX	Tuscany Internet eXchange
TOB	Toscana Ovunque Bella
UL Protocol	Ultralight Protocol
URL	Uniform Resource Locator
VT	Visit Tuscany
XML	Extensible Markup Language

I. INTRODUCTION

A. Contexte du projet

Le projet Smart Destination vise à soutenir et relancer la compétitivité des filières transnationales du tourisme. L'objectif est de mettre en œuvre un processus d'intégration et d'interopérabilité des flux d'information et des bases de données actuellement à disposition du système public-privé en cohérence avec l'offre touristique territoriale de chaque pays / région.

Pour atteindre cet objectif, il est impératif de définir et partager une architecture informatique commune incluant des interfaces de programmation (API) s'appuyant sur des modèles de données standardisés.

Basé sur un système existant qui s'affranchira des frontières, les API permettront de rendre accessible les données de chaque territoire aux acteurs privés, aux acteurs institutionnels et aux touriste.

Cette disponibilité partagée a pour objectif de faciliter la commercialisation par des entreprises, des produits transfrontaliers et de permettre aux usagers d'utiliser un système qui fonctionne au-delà des frontières.

B. Périmètre du document

Ce document fait partie de la composante T2 - Conception de l'architecture générale de l'écosystème numérique intégré de Smart Destination, et plus particulièrement de l'activité préliminaire T2.2 - Définition de l'architecture et des fonctionnalités générales, des protocoles d'interopérabilité et des interfaces.

La composante T2 est un enjeu majeur pour le projet par son doubleobjectif de:

- Concevoir l'architecture technologique du concept SMART DESTINATION;
- Mettre en place les bases de la conclusion de futurs accords transfrontaliers spécifiques et des feuilles de route pour la mise en œuvre de cette stratégie.

La composante T2 est constituée de cinq tâches dont le tableau page suivante rappelle les grandes lignes. Le présent document ne traite que de la tâche T2.2 et du livrable associé T2.2.1.

Ce document est le premier de l'activité T2.2. Il décrit l'architecture de l'écosystème numérique transfrontalier. Il s'appuie sur le rapport T2.1.3: Interopérabilité et gouvernance des systèmes d'information.

Le rapport T2.1.3 a mis en avant le développement d'un SDK commun qui fournit les

fonctionnalités de base partagées entre tous afin que les utilisateurs voient d'une manière identique la plateforme Smart Destination comme une entité unique définie par des composants logiciels et des formats et modalités d'échange de données standardisés.

Deux autres rapports viendront compléter la composante T2.2:

- Rapport T2.2.2: Spécifications fonctionnelles des composants de l'écosystème numérique transfrontalier
- Rapport T2.2.3: Définition des protocoles et interfaces d'interopérabilité

Le tableau ci-dessous rappelle les cinq tâches de la composante T2 et effectue un zoom sur la tâche T2.2 et de ses trois livrables.

<table>
<tr><th>Tâches</th><th colspan="2">Description</th></tr>
<tr><td>T2.1</td><td>Conception et architecture d'écosystèmes numériques transfrontaliers
Analyse des systèmesrégionaux (portails, applications et systèmes de gouvernance) et l'identification des meilleures pratiques et sources de données</td><td rowspan="4"></td></tr>
<tr><td>T2.2</td><td>Définition de l'architecture et des fonctionnalités générales, des protocoles d'interopérabilité et des interfaces
Définition de l'architecture et des fonctionnalités générales, des protocoles d'interopérabilité et des interfaces de l'écosystème numérique ouvert et distribué permettant aux applications de service d'accéder, à l'aide d'un format et d'un protocole unifiés, aux SI touristique des différents acteurs territoriaux publics et privés.
<table>
<tr><th>Livrable</th><th>Description</th></tr>
<tr><td>T2.2.1</td><td>Document
Architecture de l'écosystème numérique transfrontalier</td></tr>
<tr><td>T2.2.2</td><td>Rapport
Spécifications fonctionnelles des composants de l'écosystème numérique transfrontalier</td></tr>
<tr><td>T2.2.3</td><td>Rapport
Définition des protocoles et interfaces d'interopérabilité</td></tr>
</table></td></tr>
<tr><td>T2.3</td><td>Accord stratégique transfrontalier et plan d'action.
Définition de l'état de l'art et des accords sur la base des API sélectionnées mis encommun.</td></tr>
<tr><td>T2.4</td><td>Chemins d'achèvement du système régional
Activitéparallèle sur chaqueterritoireafin de définir les itinérairesspécifiques de réorientation de son écosystèmetouristiquerégional.</td><td>// à T2.1 à T2.3</td></tr>
<tr><td>T2.5</td><td>Rapport - achèvement des fonctionnalités à mettre en œuvre
Description des itinéraires de chaqueécosystèmetouristiquerégional dans la mise enœuvre des orientations techniques et caractéristiquesdéfinies.</td><td>Unitaire</td></tr>
</table>

Tableau 1 - Rappel des activités de la composante T2

II. SPÉCIFICATIONS FONCTIONNELLES DE L'ARCHITECTURE OUVERTE

A. Les attentes au niveau de l'architecture

Le chapitre présente les différentes exigences fonctionnelles retenues pour la mise en œuvre de l'architecture de la plateforme.

Cette architecture doit tenir compte de l'existant, à savoir «les systèmes d'information (SI) déjà en place dans les différentes régions pour promouvoir les produits touristiques locaux», et être construite sur un modèle de plateforme distribuée avec des interfaces de programmation d'applications (API) communes compatibles avec les normes européennes.

La mise en place de ce modèle d'architecture doit faciliter l'interopérabilité des différents SI régionaux et proposer une continuité transrégionale des services offerts au touriste, continuité actuellement inexistante.

Le public cible de la plateforme sont tous les acteurs du monde du tourisme: PA et DMO, PME et entreprises privées et producteurs de services en général, touristes. Chacun de ces acteurs interagira avec la plateforme Smart Destination par le biais d'applications logicielles existantes ou nouvellement développées, convenablement interfacées.*(Cf. T2.1.3, §5 5. Intégration de systèmes et flux de données)*.

B. Les exigences identifiées

3 types d'exigences sont identifiées:

- Fonctionnelles
- Données;
- Performance et sécurité.

1. Exigences fonctionnelles

Le schéma d'architecture, qui sera détaillé dans la phase de conception, fourniten tant que premier bloc fonctionnel fondamental, un SDK commun dont l'objectif est de:

- Fournir les fonctionnalités de base qui seront partagées entre tous;
- Exiger de chaque partenaire le développement des API internes pour s'adapter à cette couche intermédiaire.

Ce SDK permet de surmonter les problèmes liés aux logiciels, bases de données et formats de données non homogènes. Les utilisateurs verront, d'une manière identique, la plateforme Smart Destination comme une entité unique définie par des composants logiciels et des formats et modalités d'échange de données standardisés *(Cf. T2.1.3, §3 Aspect technologique).*

L'architecture sera basée sur un modèle de plateforme distribuée avec des interfaces de programmation d'applications (API) communes compatibles avec les normes européennes. Cette architecture informatique ouverte et standardisée permet de:

1. Regrouper les informations touristiques fragmentées sur de nombreuses plateformes de données;
2. Permettre un accès unifié aux différentes applications et systèmes de gestion pour les acteurs publics et privés opérant dans le domaine du tourisme;
3. Construire une offre intégrée de services personnalisés pour les touristes en mobilité;
4. Développer des capacités avancées d'analyse des données volumineuses ou «big data» (analyse des sentiments, prévision des flux, etc.).

(Cf. Présentation du projet Smart Destination, §50 Description du composant T2).

L'architecture de la plateforme appelée «Smart City SDK» nécessite la conception et le développement d'une infrastructure distribuée et décentralisée basée sur le paradigme d'interopérabilité afin d'assurer un système ouvert et partagé qui permettra à chaque région de gérer ses propres données et contexte. La plateforme pourra ainsi s'étendre au fur et à mesure de l'intégration de nouvelles régions. Cela implique les exigences suivantes:

1. Tous les composants du «Smart City SDK» sont basés sur des logiciels open source;
2. Tous les composants du «Smart City SDK» sont conformes aux engagements de l'OASC;
3. Les régions utilisent toutes les mêmes composants du «Smart City SDK»;
4. Chaque utilisateur a les mêmes possibilités de se connecter à la plateforme «Smart Destination» et d'utiliser des sources de données quelle que soit la région.

Le «Smart City SDK» et ses composants géreront de grands volumes de données et d'informations provenant de solutions verticales, de plates-formes touristiques et de sources de données externes. Le «Smart City SDK» constituera uneplateforme ouverte, réutilisable et fiable pour le partage de données. Il permettra l'homogénéisation et la normalisation des données afin de mettre en œuvre des applications intelligentes tout en réduisant les coûts de développement et d'intégration.

L'intégration du «Smart City SDK» dans chaque région se concentrera sur les modèles de données harmonisés et les APIcommune.

Le public cible de la plateforme est formé de tous les acteurs du monde du tourisme : PA et DMO, PME et entreprises privées et fournisseurs de services en général, touristes. Chacun de ces acteurs interagira avec la plateforme Smart Destination par le biais d'applications logiciellesexistantes ou nouvellement développées, convenablement interfacées. *(Cf. T2.1.3, §5 5. Intégration de systèmes et flux de données).*

Les exigences fonctionnelles sont regroupées dans le tableau suivant:

	Description
RF-01	Architecture basée sur un modèle de plateforme distribuée avec des interfaces de programmation d'applications (API) communes compatibles avec les normes européennes
RF-02	Lesinformations touristiques fragmentées sur de nombreuses plateformes de données sont regroupées dans la plateforme
RF-03	Offrir un accès unifié aux différentes applications et systèmes de gestion
RF-04	Tous les composants du «Smart City SDK» sont basés sur des logiciels open source

Tableau 2 - Exigences fonctionnelles

2. Exigences en matière de données

La gestion des données est l'un des enjeux les plus critiques du «Smart City SDK».

Le «Smart City SDK» rassemblera l'ensemble des données fournies par tous les producteurs de données («Legacy System»).

Il doit prendre en charge de multiples sources/formats de données de manière homogénéisée et normalisée.

L'évolutivité et la fiabilité sont également des atouts essentiels pour les solutions intégrées. Le «Smart City SDK» offre un soutien aux besoins actuels et futurs des solutions intégrées. Il doit permettre la capacité de stocker et d'analyser une large variété et une large quantité de données, de façon fiable.

Les exigences en matière de données sont regroupées dans le tableau suivant:

	Description
RD-01	Prise en charge de multiples sources/formats de données
RD-02	Les données sont homogènes et normalisées
RD-03	Capacité de prise en charge d'un important volume de données
RD-04	Evolutivité et fiabilité des données
RD-05	Développer des capacités d'analyse des données volumineuses

Tableau 3 - Exigences en matière de données

3. Exigences de performance etde sécurité

L'évolutivité et la fiabilité sont des atouts essentiels pour les solutions intégrées. De même la sécurité et les fonctionnalités de contrôle d'accès sont essentielles pour une utilisation appropriée de différentes catégories de données.

Le «Smart City SDK» doit répondre aux exigences en matière de disponibilité, d'efficacité, de confidentialité,de maintenance et de traçabilité.

La haute-disponibilité de la plateforme est un aspect essentiel. Si un serveur ouservice s'arrête, celui-ci ne doit pas perturber le bon fonctionnement de l'architecture.

De plus le «Smart City SDK» doit pouvoir répondre àun nombre croissant de requêtes, pour cela il doit pouvoir s'adapter par unescalabilité horizontale.

Sécurité et confidentialité: le «Smart City SDK» prendra en charge la sécurité et la

protection de la vie privée à plusieurs niveaux(composants matériels et logiciels) et se conformera au RGPD/GDPR. L'un des sujets importants au sein du RGPD/GDPR est la protection de la vie privée par conception. Tous les échanges seront ainsi tracés.

Les exigences en matière de performance et de sécurité sont regroupées dans le tableau suivant:

	Description
RPS-01	L'architecture est hautement disponible
RPS-02	L'architecture permet facilement le passage à l'échelle (scalabilité horizontale)
RPS-03	Les temps de réponse permettent l'échange de flux touristiques en temps réel
RPS-04	Traçabilité des échanges
RPS-05	Chiffrement et sécurité des échanges
RPS-06	Gestion des politiques d'accès, de production et de mise à jour
RPS-07	Respect de la vie privée

Tableau 4 - Exigences de performance et de sécurité

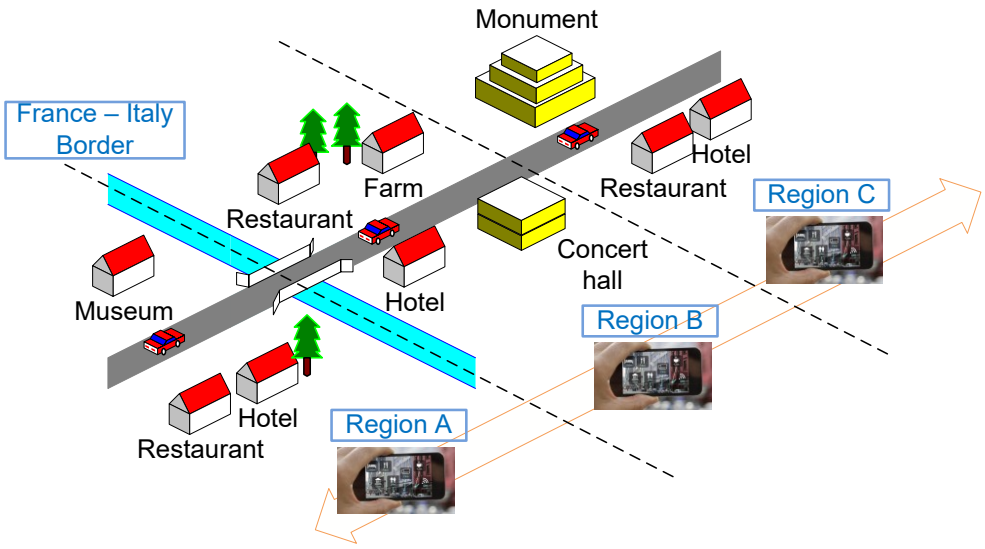
III. ARCHITETURE DE L'ÉCOSYSTÈME NUMÉRIQUE TRANSFRONTALIER

A. Présentation des principes

Le projet vise à mettre en place une infrastructure numérique communicante entre les territoires transfrontaliers.

L'objectif est de faciliter:

- a un utilisateur l'accès à toutes informations touristiques, quel que soit sa position géographique en proposant une seule interface des deux côtés de la frontière sans l'obligation de changer d'application ou de se connecter avec des identifiants et des modalités de contrôle différents;
- Aux entreprises de pouvoir développer des applications touristiques transfrontalières.

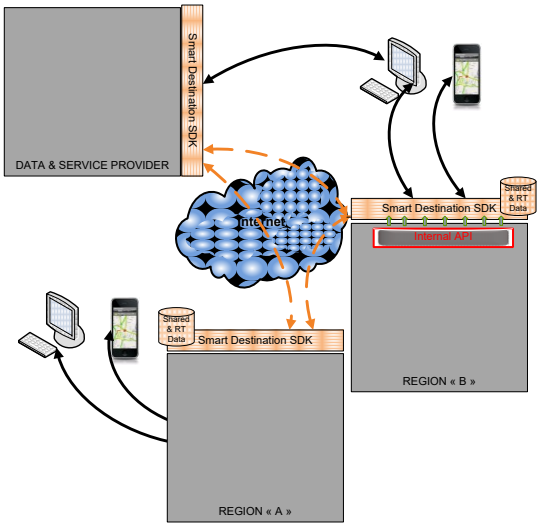


Graphique 1 - Les territoires transfrontaliers

Chaque territoire implémente le «Smart City SDK». Celui-ci permet d'héberger les données touristiques relatives au territoire et de les rendre disponibles aux applications clientes (application web, application mobile …).

Les «Legacy System» ont la responsabilité de la mise à jour régulière des données en communiquant au «Smart City SDK» toutes les modifications apportées aux données.

Dans certains cas un utilisateur peut se trouver à proximité de plusieurs territoires, un mécanisme appelé «Discovery Service» permettra à l'application d'obtenir l'ensemble des informations qui peuvent répondre aux demandes de l'utilisateur même si ces informations sont hébergées sur plusieurs territoires et donc plusieurs plateformes «Smart City DSK».

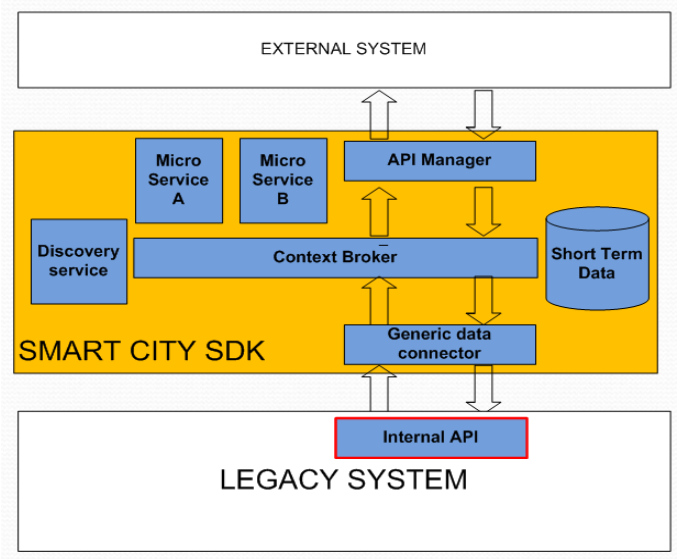


Graphique 2 - Implémentation du SDK Smart City

B. Concept de l'architectureinformatique Smart Destination

Le «Smart City SDK» rassemble l'ensemble des données fournies par tous les fournisseurs de données («Legacy System») et les rend disponible au travers d'API aux clients («External system»).

Le schéma ci-dessous (*Graphique 3 - Architecture Smart Destination*) représente les différents composants du «Smart City SDK» qui seront implémentés lors de la tâche relative aux spécifications fonctionnelles des composants de l'écosystème numérique transfrontalier.



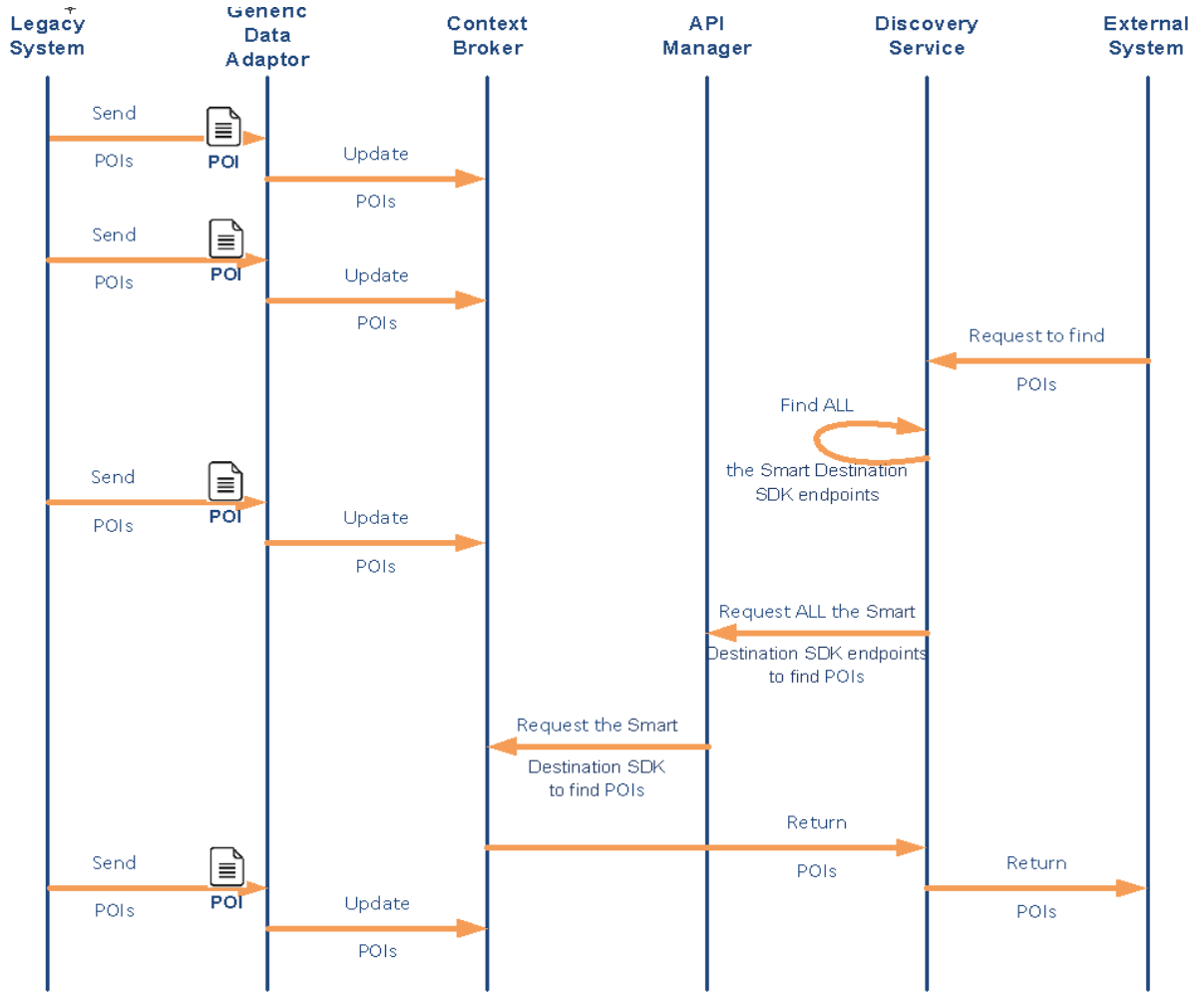
Graphique 3 - Architecture Smart Destination

Le schéma suivant (*Graphique 4 - Interaction entre les différents composants*) présente le principe d'interaction entre les different composants du «Smart City SDK».

Les fournisseurs de données («Legacy System») transfèrent régulièrement les informations relatives à leurs données touristiques (POI, Hébergements, Restauration, Evènements …). Pour cela les fournisseurs de données utilisent l'API du «Generic Data Connector» pour transmettre leurs données dans un format normalisé (Normalized Data Model). Le «Generic Data Connector» propose plusieurs protocoles d'échange afin de faciliter l'interconnexion et les échanges avec les producteurs de données. Le «Generic Data Connector» aura la charge de transmettre au «Context Broker» les données reçues. Ce dernier aura la charge d'enregistrer les données transmises après vérification des autorisations d'accès et des habilitations.

Les applications clientes («External System») qui désirent consommer des données touristiques doivent s'adresser aux «Smart City SDK» qui héberge les données désirées. Afin de connaître l'adresse du «Smart City SDK» l'application utilise le service de découvertes appelés «Discovery Service». Ce service renvoi l'adresse des «API Manager» qui permettent d'accéder à l'information recherchée.

Une fois l'adresse des «API Manager» connues, l'application utilise ses adresses afin d'accéder aux informations recherchées. Les «API Manager» ont la charge de vérifier les autorisations et les habilitations et de communiquer avec le «Context Broker».



Graphique 4 - Interaction entre les différents composants

L'ensemble de ces fonctionnalités sont détaillées ci-après.

1. CEF Context Broker

Afin de supporter la collecte, le stockage et l'approvisionnement des données en temps réel, un Context Broker sera implémenté.

Il fait partie des «Connecting Europe Facility building blocks» promu par l'Europe dont le lien est:

ec.europa.eu/cefdigital/wiki/display/CEFDIGITAL/Orion+Context+Broker).

Ce composant permet de gérer les informations de contexte de manière

décentralisée et à grande échelle. Il met en œuvre l'API FIWARE NGSI v2. C'est une API simple et puissante, qui permet d'effectuer des requêtes de mise à jour et des requêtes de consultation.

Il met en œuvre une fonction essentielle permettant de répondre à la nécessité de gérer les informations de contexte, d'effectuer des mises à jour et d'apporter un accès au contexte.

La technologie FIWARE/CEF Context Broker a été adoptée par plusieurs organismes significatifs tels que GSMA, TMForum ou des initiatives telles que OASC (Open and Agile Smart Cities). Il s'est imposé comme un de facto standard ouvert pour la gestion des informations de contexte, permettant un développement plus rapide et plus facile de solutions intelligentes ainsi que la matérialisation d'une architecture de «système de systèmes» dans de multiples domaines d'application, en particulier dans les domaines des villes intelligentes.

Le CEF Context Broker permet d'atteindre les objectifs suivants:

- Disposer d'un modèle d'informations contextuelles pouvant s'adapter à l'évolution de la connaissance du contexte;
- Disposer d'un modèle d'informations contextuelles capable de modéliser des informations hétérogènes;
- Avoir un modèle d'informations contextuelles qui collecte et fournit des informations au bon moment;
- Disposer d'un modèle d'informations contextuelles ouvert auquel on peut accéder par programme de manière standard;
- Disposer d'un modèle d'informations contextuelles indépendant du domaine d'application ou du secteur professionnel, permettant une interopérabilité entre différents secteurs et / ou différents services d'une même organisation.

Le CEF Context Broker devra respecter au minimum les exigences suivantes:

Exigences Fonctionnelles	
RF-01	Architecture basée sur un modèle de plateforme distribuée avec des interfaces de programmation d'applications (API) communes compatibles avec les normes européennes
RF-02	Les informations touristiques fragmentées sur de nombreuses plateformes de données sont regroupées dans la plateforme « Smart Destination »
RF-04	Tous les composants du « Smart City SDK » sont basés sur des logiciels open source
Exigences sur les Données	
RD-01	Prise en charge de multiples sources/formats de données
RD-02	Les données sont homogènes et normalisées

RD-03	Capacité de prise en charge d'un important volume de données
RD-04	Evolutivité et fiabilité des données
RD-05	Développer des capacités d'analyse des données volumineuses
Exigences de performance et sécurité	
RPS-01	L'architecture est hautement disponible
RPS-02	L'architecture permet facilement le passage à l'échelle (scalabilité horizontale)
RPS-03	Les temps de réponse permettent l'échange de flux touristiques en temps réel
RPS-04	Traçabilité des échanges
RPS-05	Chiffrement et sécurité des échanges

Tableau 5 - Exigences du CEF Context Broker

2. Data Storage

Le stockage des données doit s'appuyer sur un système de gestion de base de données qui manipule des objets structurés sans schéma spécifique.Cette structuration permet de faire évoluer en temps réel la représentation physique des objets sans obligation de reconfigurer la base de données. Ce type de bases de données est classé comme base de données NOSQL.

Le CEF Context Broker s'appuie sur un système de gestion de base de données orienté documents, répartissable sur un nombre quelconque d'ordinateurs et ne nécessitant pas de schéma prédéfini des données

Ce système de gestion de base de données prend en charge la réplication via un modèle master slave à des fins de résistance aux pannes et de répartition de la charge de traitement. Il peut tourner sur plusieurs ordinateurs, en répartissant ou en dupliquant les données. Il est ainsi possible de répartir les données sur plusieurs machines pour répartir la charge de travail, mais il est également possible de dupliquer les données de chaque ordinateur sur un ou plusieurs autres ordinateurs afin de garder le système de base de données opérationnel même en cas d'une panne de l'un d'eux. Le système de gestion de base de données MongoDB rend d'ailleurs ces configurations aisées à mettre en place en les automatisant. De cette façon, il est tout à fait possible d'ajouter à la volée un ou des ordinateurs à une base de données en cours de fonctionnement.

Le stockage des données devra respecter au minimum les exigences suivantes:

Exigences Fonctionnelles	
RF-02	Les informations touristiques fragmentées sur de nombreuses plateformes de données sont regroupées dans la plateforme « Smart Destination »
RF-04	Tous les composants du «Smart City SDK» sont basés sur des logiciels open source
Exigences sur les Données	
RD-01	Prise en charge de multiples sources/formats de données
RD-03	Capacité de prise en charge d'un important volume de données
RD-04	Evolutivité et fiabilité des données
RD-05	Développer des capacités d'analyse des données volumineuses
Exigences de Performance et sécurité	
RPS-01	L'architecture est hautement disponible
RPS-02	L'architecture permet facilement le passage à l'échelle (scalabilité horizontale)
RPS-03	Les temps de réponse permettent l'échange de flux touristiques en temps réel
RPS-04	Traçabilité des échanges
RPS-05	Chiffrement et sécurité des échanges

Tableau 6 - Exigences de Data Storage

3. Generic Data Connector

Le Generic Data Connector est une interface d'adaptation assurant la compatibilité de la plateforme pour son utilisation dans un large éventail de scénarios et de services applicatifs. Il permet de s'adapter aux «Legacy System» supportant différents protocoles de communications.

Le Generic Data Connector traduit les protocoles spécifiques mis en œuvre en protocole d'informations de contexte NGSI, c'est-à-dire le modèle d'échange de données standard FIWARE. Ce composant n'est pas nécessaire si les «Legacy system» prennent en charge de manière native l'API NGSI. Le Generic Data Connector supporter les protocoles suivants:

- Json over HTTP;
- Json over MQTT;
- UL2.0 over HTTP;
- UL2.0 over MQTT.

Le Generic Data Connector devra respecter au minimum les exigences suivantes:

Exigences Fonctionnelles	
RF-04	Tous les composants du «Smart City SDK» sont basés sur des logiciels open source
Exigences sur les Données	
RD-03	Capacité de prise en charge d'un important volume de données
Exigences de Performance et sécurité	
RPS-01	L'architecture est hautement disponible
RPS-02	L'architecture permet facilement le passage à l'échelle (scalabilité horizontale)
RPS-03	Les temps de réponse permettent l'échange de flux touristiques en temps réel
RPS-04	Traçabilité des échanges
RPS-05	Chiffrement et sécurité des échanges
RPS-06	Gestion des politiques d'accès, de production et de mise à jour

Tableau 7 - Exigences de Generic Data Connector

4. API Manager

Le gestionnaire d'API est un proxy placé devant les API d'accès aux composants de la plateforme «Smart Destination» (CEF Context Broker, Micro-Services).

Il doit permettre d'ajouter des fonctionnalités communes telles que la gestion des clés d'API, la limitation de débit et les statistiques d'usage de toutes les API.

La «Limitation de débit» permet de contrôler le nombre de requêtes que chaque utilisateur peut adresser aux API afin de prévenir les abus ou définir des seuils d'utilisation. Des limites variables de taux d'accès peuvent être définies, allant d'une limite par seconde à une limite par jour. Différentes limites peuvent être définies pour différentes API ou pour des utilisateurs spécifiques.

Les statistiques d'usage permettent d'analyser comment les API sont utilisées grâce à une analyse détaillée des requêtes d'API, d'afficher des données récapitulatives et d'explorer les détails de l'utilisation des API.

Le gestionnaire d'API doit intégrer une couche de mise en cache HTTP standard devant les API afind'accélérer l'accès aux API. Il permet également de fournir un seul point d'entrée public à toutes les API et aux Micro-Services.

Le gestionnaire d'API devra respecter au minimum les exigences suivantes:

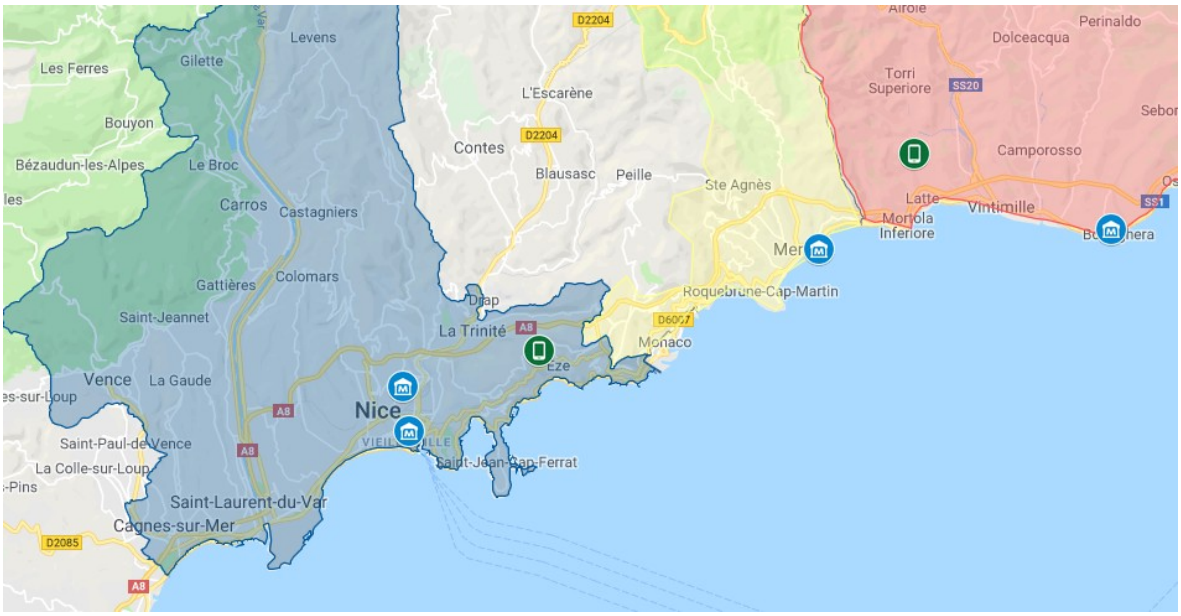
Exigences Fonctionnelles	
RF-03	Offrir un accès unifié aux différentes applications et systèmes de gestion
RF-04	Tous les composants du «Smart City SDK» sont basés sur des logiciels open source
Exigences sur les Données	
RD-03	Capacité de prise en charge d'un important volume de données
Exigences de Performance et sécurité	
RPS-01	L'architecture est hautement disponible
RPS-02	L'architecture permet facilement le passage à l'échelle (scalabilité horizontale)
RPS-03	Les temps de réponse permettent l'échange de flux touristiques en temps réel
RPS-04	Traçabilité des échanges
RPS-05	Chiffrement et sécurité des échanges
RPS-06	Gestion des politiques d'accès, de production et de mise à jour

Tableau 8 - Exigences de API Manager

5. Discovery Service

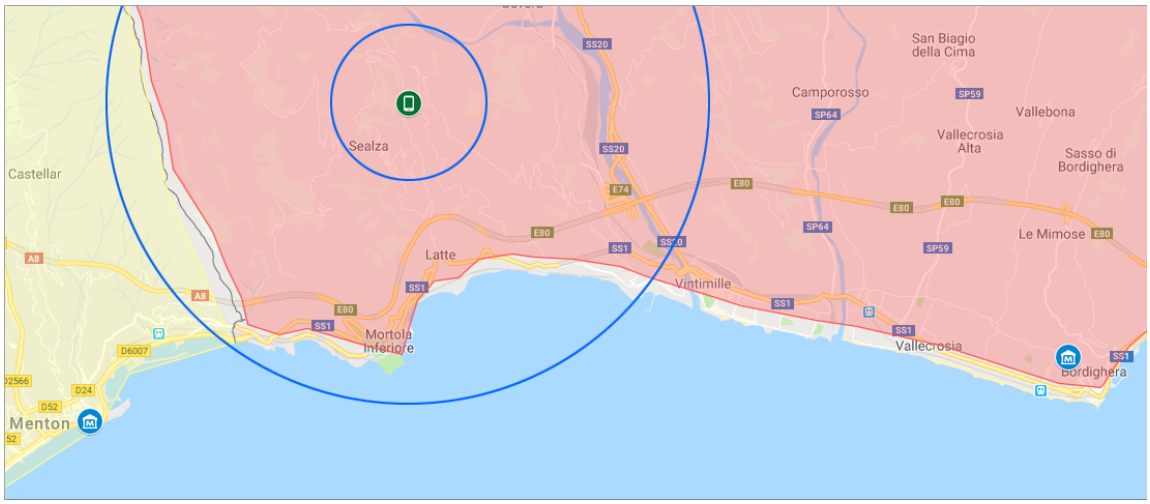
Le «Discovery Service» est un mécanisme qui permet à une application ded'obtenir l'ensembles des informations qui peuvent répondre aux demandes de l'utilisateur même si ces informations sont hébergées sur plusieurs territoires et leur plateforme «Smart City SDK».

Pour rappel, chaque région intégrera la plateforme «Smart Destination» et rassemblera les modèles harmonisés des données touristiques *(Cf. Graphique 4 - Region et Smart City SDK*).



Graphique 5 - Régions e Smart City SDK

Si un utilisateur se trouve à proximité de plusieurs régions, l'application qu'il utilise doit pouvoir obtenir l'ensemble des informations des plateformes « Smart City SDK » succeptibles de délivrer les informations recherchées par l'utilisateur *(Cf. Graphique 6 - Application à proximité de deux régions)*.



Graphique 6 - Application à proximité de deux régions

Pour cela une application doit s'adresser au «Discovery Service » et celui-ci doit pouvoir intérogerl'ensemble des adresses (URL) des services «Smart Destination SDK»de différentes régions afin de retouner l'ensemble des données recherchées par l'utilisateur. Une recherche d'un utilsateur se fera en spécifiant la topologie d'une zone de recherche, c'est-à-dire la forme de référence à utiliser lors de la résolution de la requête (ex: recherche dans un rayon autour de ma position, recherche dans un polygone déterminé, recherche sur un trajet).

Le «Discovery Service» devra respecter au minimum les exigences suivantes:

Exigences Fonctionnelles	
RF-04	Tous les composants du « Smart City SDK » sont basés sur des logiciels open source
Exigences sur les Données	
	Aucune
Exigences de Performance et sécurité	
RPS-01	L'architecture est hautement disponible
RPS-02	L'architecture permet facilement le passage à l'échelle (scalabilité horizontale)
RPS-03	Les temps de réponse permettent l'échange de flux touristiques en temps réel
RPS-04	Traçabilité des échanges
RPS-05	Chiffrement et sécurité des échanges

Tableau 9 - Exigences de Discovery Service

6. Micro-Service

Des Micro-Services pourront être déployés pour répondre aux besoins spécifiques des démonstrateurs. Ces Micro-Services permettront au «Smart City SDK» de proposer des services supplémentaires. La définition et l'étude de ces Micro-Services se fera lorsque les démonstrateurs auront été spécifiés.

Ces Micro-Services devront respecter au minimum les exigences suivantes:

Exigences Fonctionnelles	
RF-01	Architecture basée sur un modèle de plateforme distribuée avec des interfaces de programmation d'applications (API) communes compatibles avec les normes européennes
RF-04	Tous les composants du « Smart City SDK » sont basés sur des logiciels open source

Exigences sur les Données	
	Aucune
Exigences de Performance et sécurité	
RPS-01	L'architecture est hautement disponible
RPS-02	L'architecture permet facilement le passage à l'échelle (scalabilité horizontale)
RPS-03	Les temps de réponse permettent l'échange de flux touristiques en temps réel
RPS-04	Traçabilité des échanges
RPS-05	Chiffrement et sécurité des échanges
RPS-07	Respect de la vie privée

Tableau 10 - Exigences de Micro Service

7. Modèles de données

Le rapport T2.1.3a mis en avant le développement d'un SDK commun qui fournit les fonctionnalités de base partagées entre tous afin que les utilisateurs voient d'une manière homogène, la plateforme Smart Destination comme une entité unique bien définie à la fois en tant que composants logiciels et en tant que formats et modalités d'échange de données.

Il précise au chapitre 4 *(§4 Aspect qualitatif, quantitatif et sémantique)* que la conception du SDK doit inclure:

- La définition d'un format de représentation unique pour chaque attribut des différentes entités.
  Lors du développement de la couche d'adaptation entre le système régional et le SDK, il sera de la responsabilité de chaque partenaire de respecter le format imposé.

- La définition de la liste des entités pouvant être représentées et des attributs relatifs.
  Dans ce cas, chaque partenaire, ayant une connaissance maximale de son système régional, doit proposer la meilleure cartographie entre son système régional et les données requises par le SDK.

- Sur la fréquence des mises à jour et des modalités de saisie des données, en particulier une analyse approfondie par l'ensemble des partenaires sur ces deux aspects est nécessaire afin d'assurer la cohérence des résultats fournis par la plateforme Smart Destination.

Il existe peu de modèles de données associés spécifiquement au tourisme pour le moment et nous aurons par conséquent l'opportunité de proposer de nouveaux modèles pour répondre aux enjeux de «Smart Destination».

La Métropole Nice Côte d'Azur a été sélectionnée comme territoire pilote d'un nouveau programme de coopération mondiale entre villes intelligentes intitulé «Front-runner Smart Cities».

Ce programme a été lancé mi-novembre 2018 lors du Smart City Expo World Congress *(www.smartcityexpo.com/en)* à Barcelone par le TM Forum et la FiwareFoundation. Il a l'ambition de définir un nouveau standard de référence pour les données relevant de la smart city. Les villes qui accompagneront Nice à travers ce programme de coopération sont Vienne (Autriche), Nice, Saint Quentin (France), Gênes (Italie), Utrecht (Netherlands), Porto (Portugal) Santander, Valence (Espagne), Gothenburg (Suède), La Plata (Argentine) et Montevideo (Uruguay).

Ce programme dans lequel la Métropole de Nice est engagée, permettra de soumettre les modèles de données que nous aurons définis dans le cadre de Smart Destination et de les voir ainsi reconnu comme standard de référence pour les données de la smart city.

Afin de modéliser les données touristiques nous pouvons nous appuyer sur le référentiel SynchroniCity qui contient un ensemble de modèles de données partagées OASC pour les domaines Smart City, adoptés et développés et du guide pratique qui donne les lignes directrices pour la définition de nouveaux modèles de données.

Référentiel: https://gitlab.com/synchronicity-iot/synchronicity-data-models

Guide pratique: https://gitlab.com/synchronicity-iot/synchronicity-data-models/blob/master/guidelines.md

C. L'architecture vis-à-vis des exigences

Le tableau suivant permet de visualiserles exigences définies au §II.B des différents composants de l'architecture de l'écosystème numérique transfrontallier.

Exigence	Context Broker	Storage	Generic Data Connector	API Manager	Directory Service	Micro Service
RF-01	X					X
RF-02	X	X				
RF-03				X		
RF-04	X	X	X	X	X	X
RD-01	X	X				
RD-02	X					
RD-03	X	X	X	X		
RD-04	X	X				
RD-05	X	X				
RPS-01	X	X	X	X	X	X
RPS-02	X	X	X	X	X	X
RPS-03	X	X	X	X	X	X
RPS-04	X	X	X	X	X	X
RPS-05	X	X	X	X	X	X
RPS-06			X	X		
RPS-07						X

Tableau 11 - Conformité architecturale

Interreg
MARITTIMO–IT FR–MARITIME
Fondo Europeo di Sviluppo Regionale
Fonds Européen de Développement Régional

UNIONE EUROPEA
UNION EUROPÉENNE

smart DESTINATION

Rapport: Spécifications fonctionnelles des composants de l'écosystème numérique transfrontalier (Livrable T2.2.2)

Composant T2 - Conception de l'architecture générale de l'écosystème numérique intégré de SMART DESTINATION

La coopération au coeur de la Méditerranée
La cooperazione al cuore del Mediterraneo

Regione Toscana
Regione Autònoma de Sardigna
Regione Autonoma della Sardegna
Regione Liguria
Métropole Nice Côte d'Azur
entro assistenza tecnica
fondazione sistema toscana
CCI Provence Alpes Côte d'Azur

INDICE

TABLEAU DES ILLUSTRATIONS

TABLEAU GRAPHIQUE

ACRONYMES

Abbreviazione	Description
API (REST)	Application Programming Interface (REpresentational State Transfer)
AWS	Amazon Web Services
BE	Back End (parte gestionale di un'applicazione)
CARF	Communauté d'Agglomération de la Riviera Française
CEF	Connecting Europe Facility
DMS	Document Management System
DMO	Destination Management Organisation
EH	Easy Holiday
FE	Front End (parte pubblica di un'applicazione)
FTP	File Transfer Protocol
GSMA	GSM Association
HIS	Hyperlocal Infopoint Sardegna
IIS	Internet Information Services (Microsoft)
IRPET	Istituto Regionale Programmazione Economica Toscana
JSON	JavaScript Object Notation
MINT	Muoversi IN Toscana
MNCA	Métropole Nice Côte d'Azur
NTIC	Nouvelles Technologies de l'Information de la Communication
OASC	Open & Agile Smart Cities
OT/OTI	Office du Tourisme/Office du Tourisme Intermunicipale
OTCN	Office du Tourisme et des Congrès de Nice
OTM	Office du Tourisme Métropolitain
OTMNCA	Office de Tourisme Métropolitain Nice Côte d'Azur
OVH	Web hosting company francese (fornitore di MNCA)
PA	Public Administration
PME	Piccole e Medie Imprese

POI	Point of Interest
RAS	Regione Autonoma della Sardegna
RL	Regione Liguria
RT	Regione Toscana
SDK	Software Development Kit
SQL	Structured Query Language
SUAPE	Sportello Unico delle Attività Produttive e dell'Edilizia (RAS)
TIX	Tuscany Internet eXchange
TOB	Toscana Ovunque Bella
UL Protocol	Ultralight Protocol
URL	Uniform Resource Locator
VT	Visit Tuscany
XML	Extensible Markup Language

I. INTRODUCTION

A. Contexte du projet

Le projet Smart Destination vise à soutenir et relancer la compétitivité des filières transnationales du tourisme. L'objectif est de mettre en œuvre un processus d'intégration et d'interopérabilité des flux d'information et des bases de données actuellement à disposition du système public-privé en cohérence avec l'offre touristique territoriale de chaque pays/région.

Pour atteindre cet objectif, il est impératif de définir et partager une architecture informatique commune incluant des interfaces de programmation (API).

Basé sur un système existant qui s'affranchira des frontières, les API permettront de fournir des données de chaque territoire et la rendre accessible aux acteurs privés, aux institutions publiques et aux touristes.

Cette disponibilité partagée a pour objectif de faciliter la commercialisation par des entreprises, des produits transfrontaliers et de permettre aux usagers d'utiliser un système qui fonctionne au-delà des frontières.

B. Périmètre du document

Ce document fait partie de la composante T2 - Conception de l'architecture générale de l'écosystème numérique intégré de Smart Destination, et plus particulièrement de l'activité préliminaire T2.2 - Définition de l'architecture et des fonctionnalités générales, des protocoles d'interopérabilité et des interfaces.

La composante T2 est un enjeu majeur pour le projet par son double objectif de:

- Concevoir l'architecture technologique du concept SMART DESTINATION;
- Mettre en place les bases de la conclusion de futurs accords transfrontaliers spécifiques et des feuilles de route pour la mise en œuvre de cette stratégie.

La composante T2 est constituée de 5 tâches dont le tableau page suivante rappelle les grandes lignes. Le document ne traite que de la tâche T2.2 et du livrable T2.2.2.

Ce document est le deuxième volet de l'activité T2.2. Il donne les spécifications fonctionnelles des composants de l'écosystème numérique transfrontalier. Il s'appuie sur le rapport T2.1.3: Interopérabilité et gouvernance des systèmes d'information et

le document T.2.2.1: Architecture de l'écosystème numérique transfrontalier.

Le rapport T2.1.3 a mis en avant le développement d'un SDK commun qui fournit les fonctionnalités de base partagées entre tous afin que les utilisateurs voient d'une manière transparente, la plate-forme Smart Destination comme une entité unique bien définie à la fois en tant que composants logiciels et en tant que formats et modalités d'échange de données. Le document T.2.2.1 a permis d'une part de définir les différentes exigences fonctionnelles retenues pour la mise en œuvre de l'architecture de la plateforme et d'autre part de définir l'architecture différents composants de la plateforme «Smart Destination».

Un dernier rapport viendra compléter la composante T2.2:

- Rapport T2.2.3: Définition des protocoles et interfaces d'interopérabilité.

Le tableau ci-dessous rappelle les 5 tâches de la composante T2 et effectue un zoom sur la tâche T2.2 et de de ses 3 livrables.

<table>
<tr><th>Tâches</th><th colspan="2">Description</th></tr>
<tr><td>T2.1</td><td colspan="2">Conception et architecture d'écosystèmes numériques transfrontaliers
Analyse des systèmes régionaux (portails, applications et systèmes de gouvernance) et l'identification des meilleures pratiques et sources de données</td></tr>
<tr><td>T2.2</td><td colspan="2">Définition de l'architecture et des fonctionnalités générales, des protocoles d'interopérabilité et des interfaces
Définition de l'architecture et des fonctionnalités générales, des protocoles d'interopérabilité et des interfaces de l'écosystème numérique ouvert et distribué permettant aux applications de service d'accéder, à l'aide d'un format et d'un protocole unifiés, aux SI touristique des différents acteurs territoriaux publics et privés.
<table>
<tr><th>Livrable</th><th>Description</th></tr>
<tr><td>T2.2.1</td><td>Document
Architecture de l'écosystème numérique transfrontalier</td></tr>
<tr><td>T2.2.2</td><td>Rapport
Spécifications fonctionnelles des composants de l'écosystème numérique transfrontalier</td></tr>
<tr><td>T2.2.3</td><td>Rapport
Définition des protocoles et interfaces d'interopérabilité</td></tr>
</table></td></tr>
<tr><td>T2.3</td><td colspan="2">Accord stratégique transfrontalier et plan d'action.
Définition de l'état de l'art et des accords sur la base des API sélectionnées mis en commun.</td></tr>
<tr><td>T2.4</td><td>Chemins d'achèvement du système régional
Activité parallèle sur chaque territoire afin de définir les itinéraires spécifiques de réorientation de son écosystème touristique régional.</td><td>// à T2.1 à T2.3</td></tr>
<tr><td>T2.5</td><td>Rapport - achèvement des fonctionnalités à mettre en œuvre
Description des itinéraires de chaque écosystème touristique régional dans la mise en œuvre des orientations techniques et caractéristiques définies.</td><td>Unitaire</td></tr>
</table>

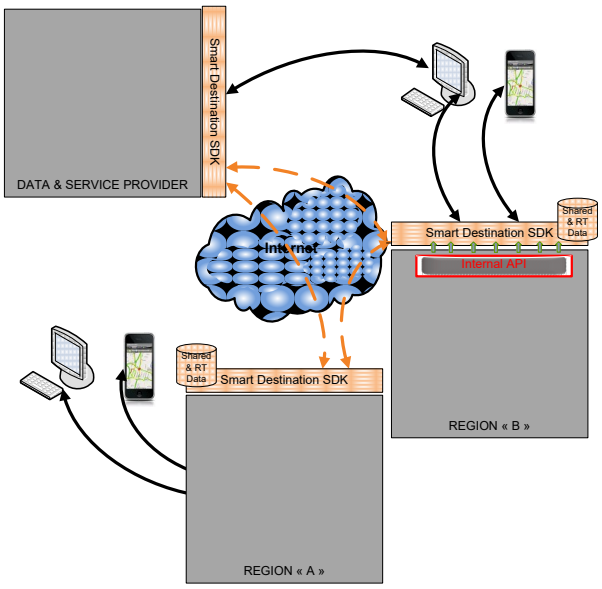
Tableau 1 - Rappel des activités de la composante T2

II. SPÉCIFICATIONS FONCTIONNELLES DES COMPOSANTS DE L'ÉCOSYSTÈME NUMÉRIQUE TRANSFRONTALIER

Chaque territoire implémente le «Smart City SDK».

Celui-ci permet d'héberger les données touristiques relatives au territoire et de les rendre disponibles aux applications clientes (application web, application mobile …).

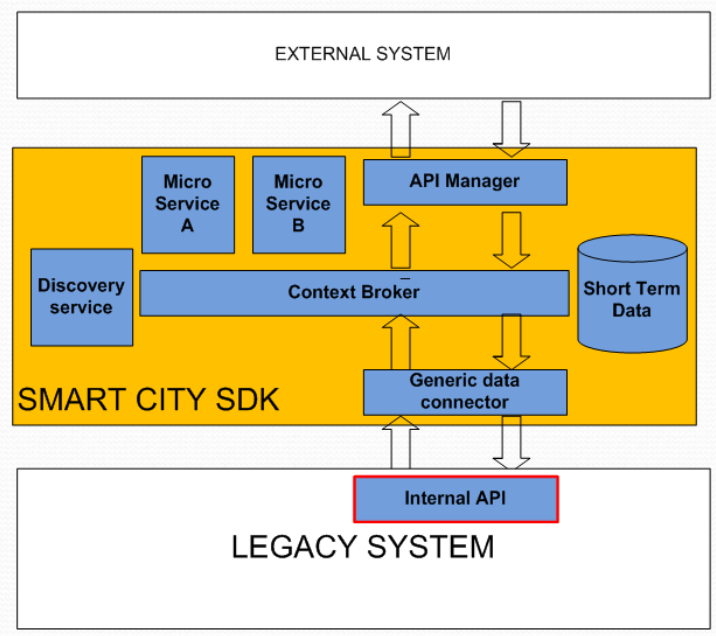
Dans certains cas un utilisateur peut se trouver à proximité de deux territoires, un mécanisme appelé «Discovery Service» permettra à l'application de connaître les plates-formes «Smart City SDK» qui peuvent répondre aux demandes de l'utilisateur.



Graphique 1 - Implémentation du SDK Smart City

Le «Smart City SDK» rassemble l'ensemble des données fournies par tous les fournisseurs de données («Legacy System») et les rend disponible au travers d'API de système externes.

Le schéma ci-dessous (Architettura Smart Destination) représente les différents composants du «Smart City SDK» qui seront implémentés lors de la tâche relative aux spécifications fonctionnelles des composants de l'écosystème numérique transfrontalier.



Graphique 2 - Architecture Smart Destination

A. CEF Context Broker

Afin de supporter la collecte, le stockage et l'approvisionnement des données en temps réel, un Context Broker est implémenté.

orion FIWARE **FIWARE-ORION**	Orion Context Broker permet de gérer l'ensemble du cycle de vie des informations contextuelles, y compris les mises à jour, les requêtes, les inscriptions et les abonnements. Il s'agit d'une implémentation de serveur NGSIv2 pour gérer les informations contextuelles et leur disponibilité. https://fiware-orion.readthedocs.io/en/master/
Informations techniques	*Installation* https://fiware-orion/master/admin/install *Implémentation* • Admin Guide https://fiware-orion/master/admin • User & Programmers Guide https://fiware-orion/master/user
Software/Hardware Requirements	Orion context broker est développé en C++. La taille de l'image du conteneur Orion est de 260 MB. *Host* • 2 CPU cores • 4 GB RAM *Operating system* • CentOS 7.4.1708 but it should work also in any later CentOS • RedHat 7.x version. *Database* • MongoDB 3.6 est nécessaire pour s'exécuter soit dans le même hôte où Orion Context Broker doit être installé ou dans un hôte différent accessible via le réseau.

Pour rappel le document T2.2.1 Architecture de l'écosystème numérique transfrontalier a défini les exigences suivantes:

Exigences Fonctionnelles	
RF-01	Architecture basée sur un modèle de plateforme distribuée avec des interfaces de programmation d'applications (API) communes compatibles avec les normes européennes
RF-02	Les informations touristiques fragmentées sur de nombreuses plateformes de données sont regroupées dans la plateforme « Smart Destination »
RF-04	Tous les composants du « Smart City SDK » sont basés sur des logiciels open source
Exigences sur les Données	
RD-01	Prise en charge de multiples sources/formats de données
RD-02	Les données sont homogènes et normalisées
RD-03	Capacité de prise en charge d'un important volume de données
RD-04	Evolutivité et fiabilité des données
RD-05	Développer des capacités d'analyse des données volumineuses
Exigences de performance et sécurité	
RPS-01	L'architecture est hautement disponible
RPS-02	L'architecture permet facilement le passage à l'échelle (scalabilité horizontale)
RPS-03	Les temps de réponse permettent l'échange de flux touristiques en temps réel
RPS-04	Traçabilité des échanges
RPS-05	Chiffrement et sécurité des échanges

Tableau 2 - Exigences de CEF Context Broker

Le Context Broker Orion est basé sur un logiciel Open Source (RF-04) et met en œuvre l'API FIWARE NGSI v2 (RF-01, RF-02, RD-02).

Afin de fournir une disponibilité et une évolutivité élevées, le composant Orion est déployé en tant que service à travers un fichier de pile Docker, plusieurs instances de conteneur s'exécuteront pour fournir une haute disponibilité et une grande évolutivité grâce à une scalabilité horizontale (RPS-01, RPS-02, RPS-03). Tous les échanges sont sécurisés et chiffrés par l'utilisation de certificats, et également tracés

(RPS-04, RPS-05).

Pour le stockage des données un système de gestion de base de données NoSQL est utilisé, il manipule des objets structurés sans schéma spécifique (RD-01). Il est évolutif en multipliant les nœuds/conteneurs exécutant le logiciel et en reproduisant les données sur les nœuds (RD-02, RD-03, RD-04, RPS-01, RPS-02, RPS-03).

B. Data storage

MongoDB est un système de gestion de base de données orienté documents, répartissable sur un nombre quelconque d'ordinateurs et ne nécessitant pas de schéma prédéfini des données.

mongoDB	MongoDb est un système de gestion de base de données NoSQL qui manipule des objets structurés sans schéma spécifique, évolutif en multipliant les nœuds/conteneurs exécutant le logiciel et en reproduisant les données sur les nœuds. www.mongodb.com
How to install the technical component	*Installation* https://mongodb.com/community *Implémentation* • Admin Guide / User Guide https://fiware-draco/ngsi_mongo_sink
Software/Hardware Requirements	• MongoDB version 3.6 Taille de la mémoire RAM du container : 4 GB Storage: • Docker image 361 MB • Les données doivent être stockées sur un volume Docker spécifique sur l'hôte du conteneur, chaque conteneur exécutant MongoDB doit avoir son propre volume sur le nœud hôte • La taille disponible pour le stockage de données, doit être d'au moins 300 Go pour chaque volume de stockage

Avec MongoDB, nous parlons de "Documents" qui sont stockés dans "Collections". Une "Collection" contient un certain nombre de "Documents". Un

"Document" peut contenir un nombre différent de "Champs", et peuvent varier pour la même "Collection".

Comparaison avec une base relationnelle:

- Collections = Tables,
- Documents = Records.

Pour rappel le document T2.2.1 Architecture de l'écosystème numérique transfrontalier a défini les exigences suivantes:

Exigences Fonctionnelles	
RF-02	Les informations touristiques fragmentées sur de nombreuses plateformes de données sont regroupées dans la plateforme « Smart Destination »
RF-04	Tous les composants du « Smart City SDK » sont basés sur des logiciels open source
Exigences sur les Données	
RD-01	Prise en charge de multiples sources/formats de données
RD-03	Capacité de prise en charge d'un important volume de données
RD-04	Evolutivité et fiabilité des données
RD-05	Développer des capacités d'analyse des données volumineuses
Exigences de Performance et sécurité	
RPS-01	L'architecture est hautement disponible
RPS-02	L'architecture permet facilement le passage à l'échelle (scalabilité horizontale)
RPS-03	Les temps de réponse permettent l'échange de flux touristiques en temps réel
RPS-04	Traçabilité des échanges
RPS-05	Chiffrement et sécurité des échanges

Tableau 3 - Exigences de Data Storage

MongoDB est un logiciel libre qui permet le stockage des informations touristiques, il est orienté documents et répartissable sur un nombre quelconque d'ordinateurs (RF-02, RF-04, RPS-01, RPS-02, RPS-03).

Un "Document" peut contenir un nombre différent de "Champs", et peuvent varier pour la même "Collection" (RD01- RD-03, RD-04). Il permet le stockage et le

traitement d'un important volume de données de façon sécurisé (RD-05, RPS-04, RPS-05)

C. Generic Data Connector

Le Generic Data Connector est une interface d'adaptation assurant la compatibilité de la plateforme pour son utilisation dans un large éventail de scénarios et de services applicatifs. Il permet de s'adapter aux Legacy System supportant différents protocoles de communications.

Pour rappel le document T2.2.1 Architecture de l'écosystème numérique transfrontalier a défini les exigences suivantes:

Exigences Fonctionnelles	
RF-04	Tous les composants du « Smart City SDK » sont basés sur des logiciels open source
Exigences sur les Données	
RD-03	Capacité de prise en charge d'un important volume de données
Exigences de Performance et sécurité	
RPS-01	L'architecture est hautement disponible
RPS-02	L'architecture permet facilement le passage à l'échelle (scalabilité horizontale)
RPS-03	Les temps de réponse permettent l'échange de flux touristiques en temps réel
RPS-04	Traçabilité des échanges
RPS-05	Chiffrement et sécurité des échanges
RPS-06	Gestion des politiques d'accès, de production et de mise à jour

Tableau 4 - Exigences de Generic Data Connector

Le Generic Data Connector est un ensemble d'API mis à disposition des Legacy System. Il est développé en NodeJS et est réparti sur plusieurs containeurs assurant ainsi la performance, la haute disponibilité et scalabilité (RD-03, RPS-01. RPS-02.RPS-03, RPS-04). En frontal l'API Manager permet d'assurer la sécurité et la confidentialité des échanges (RPS-05, RPS-06).

D. API Manager

Le gestionnaire d'API est un proxy placé devant les API d'accès aux composants de la plateforme Smart Destination (CEF Context Broker, Micro-Services).

API Umbrella	Umbrella est une plate-forme de gestion d'API open source pour les services Web. https://apiumbrella.io/
How to install the technical component	*Installation* https://apiumbrella.io/install/ https://api-umbrella.readthedocs.io/en/latest/
Software/Hardware Requirements	Configuration minimum recommandée de mémoire 1.5 GB L'API Umbrella s'exécute sur le componant suivant "Docker / Debian / Enterprise Linux / Umbuntu / à partir du code source"

API Umbrella est développé avec Ruby on Rails, il utilise MongoDB pour la persistance des données, et Elasticsearch pour l'analyse, l'interface est une API RESTfull

Pour rappel le document T2.2.1 Architecture de l'écosystème numérique transfrontalier a défini les exigences suivantes:

Exigences Fonctionnelles	
RF-03	Offrir un accès unifié aux différentes applications et systèmes de gestion
RF-04	Tous les composants du « Smart City SDK » sont basés sur des logiciels open source
Exigences sur les Données	
RD-03	Capacité de prise en charge d'un important volume de données
Exigences de Performance et sécurité	
RPS-01	L'architecture est hautement disponible
RPS-02	L'architecture permet facilement le passage à l'échelle (scalabilité horizontale)
RPS-03	Les temps de réponse permettent l'échange de flux touristiques en temps réel
RPS-04	Traçabilité des échanges

RPS-05	Chiffrement et sécurité des échanges
RPS-06	Gestion des politiques d'accès, de production et de mise à jour

Tableau 5 - Exigences de API Manager

API Umbrella est un logiciel open source permettant l'accès unifié aux différentes applications et systèmes de gestion (RF-03, RF-04)

API Umbrella est réparti sur plusieurs containeurs assurant ainsi la performance, la haute disponibilité et scalabilité (RD-03, RPS-01. RPS-02.RPS-03, RPS-04). L'API Manager permet d'assurer la sécurité et la confidentialité des échanges (RPS-05, RPS-06).

E. Discovery Service

Le Discovery Service est un mécanisme qui permet à une application de connaître les plateformes destinations intelligientes qui peuvent répondre aux demandes de l'utilisateur.

	Node.js est une plateforme logicielle libre en JavaScript orientée vers les applications réseau événementielles hautement concurrentes qui doivent pouvoir monter en charge.
How to install the technical component	*Installation* https://nodejs.org/en/
Software/Hardware Requirements	N/A

Pour rappel le document T2.2.1 Architecture de l'écosystème numérique transfrontalier a défini les exigences suivantes:

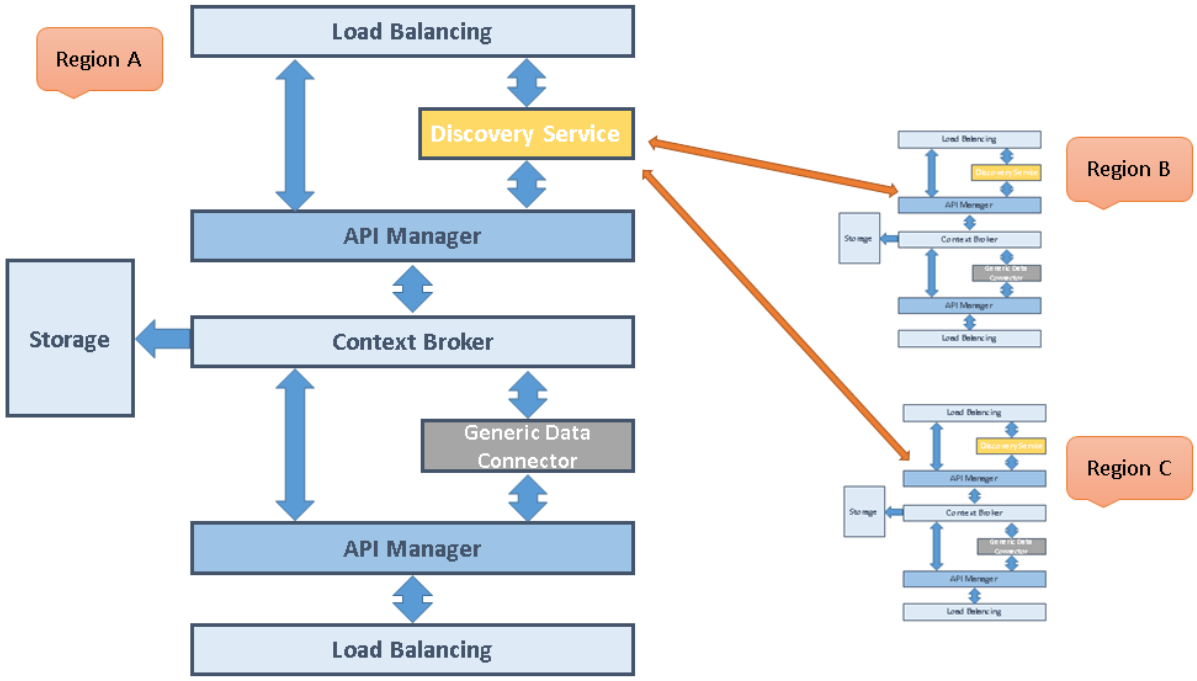
Exigences Fonctionnelles	
RF-04	Tous les composants du « Smart City SDK » sont basés sur des logiciels open source
Exigences sur les Données	
	Aucune
Exigences de Performance et sécurité	
RPS-01	L'architecture est hautement disponible
RPS-02	L'architecture permet facilement le passage à l'échelle (scalabilité horizontale)
RPS-03	Les temps de réponse permettent l'échange de flux touristiques en temps réel
RPS-04	Traçabilité des échanges
RPS-05	Chiffrement et sécurité des échanges

Tableau 6 - Exigences de Discovery Service

Le Discovery Service utilise le Context Broker pour décrire les différents services accessibles sur chacune des plateformes Smart Destination. Pour chacune de ces adresses retournées le Context Broker permet de décrire la «topologie» de la zone couverte par ce service ainsi que la liste des types d'entités accessibles.

Le Discovery Service s'appuie sur le Context Broker (RF-04, RPS-01, RPS-02, RPS-03, RPS-04, RPS-05).

F. Architecture Smart Destination



Graphique 3 - Architecture Smart Destination

L'architecture «Destination Intelligente» doit être hautement disponible et répondre aux exigences de temps de réponse.

Pour cela l'utilisation de Docker est un moyen simple de déployer des services hautement disponibles, en déployant plusieurs conteneurs sur plusieurs nœuds d'un essaim, en utilisant des fichiers de composition de docker. De plus un Load Balancer sera mis en place afin de garantir la disponibilité et la répartition de charge.

Interreg
MARITTIMO-IT FR-MARITIME
Fondo Europeo di Sviluppo Regionale
Fonds Européen de Développement Régional

UNIONE EUROPEA
UNION EUROPÉENNE

smart DESTINATION

Rapport: Définition des protocoles et interfaces d'interopérabilité (Livrable T2.2.3)

Composant T2 - Conception de l'architecture générale de l'écosystème numérique intégré de SMART DESTINATION

La coopération au coeur de la Méditerranée
La cooperazione al cuore del Mediterraneo

Regione Toscana
Regione Autònoma de Sardigna
Regione Autonoma della Sardegna
Regione Liguria
Métropole Nice Côte d'Azur
ente assistenza tecnica
fondazione sistema toscana
CCI Provence Alpes Côte d'Azur

INDICE

TABLEAU DES ILLUSTRATIONS

TABLEAU GRAPHIQUE

ACRONYMES

Abbreviazione	Description
API (REST)	Application Programming Interface (REpresentational State Transfer)
AWS	Amazon Web Services
BE	Back End (parte gestionale di un'applicazione)
CARF	Communauté d'Agglomération de la Riviera Française
CEF	Connecting Europe Facility
DMS	Document Management System
DMO	Destination Management Organisation
EH	Easy Holiday
FE	Front End (parte pubblica di un'applicazione)
FTP	File Transfer Protocol
GSMA	GSM Association
HIS	Hyperlocal Infopoint Sardegna
IIS	Internet Information Services (Microsoft)
IRPET	Istituto Regionale Programmazione Economica Toscana
JSON	JavaScript Object Notation
MINT	Muoversi IN Toscana
MNCA	Métropole Nice Côte d'Azur
NTIC	Nouvelles Technologies de l'Information de la Communication
OASC	Open & Agile Smart Cities
OT/OTI	Office du Tourisme/Office du Tourisme Intermunicipale
OTCN	Office du Tourisme et des Congrès de Nice
OTM	Office du Tourisme Métropolitain
OTMNCA	Office de Tourisme Métropolitain Nice Côte d'Azur
OVH	Web hosting company francese (fornitore di MNCA)
PA	Public Administration
PME	Piccole e Medie Imprese

POI	Point of Interest
RAS	Regione Autonoma della Sardegna
RL	Regione Liguria
RT	Regione Toscana
SDK	Software Development Kit
SQL	Structured Query Language
SUAPE	Sportello Unico delle Attività Produttive e dell'Edilizia (RAS)
TIX	Tuscany Internet eXchange
TOB	Toscana Ovunque Bella
UL Protocol	Ultralight Protocol
URL	Uniform Resource Locator
VT	Visit Tuscany
XML	Extensible Markup Language

I. INTRODUCTION

A. Contexte du projet

Le projet Smart Destination vise à soutenir et développer la compétitivité des filières transnationales du tourisme. L'objectif est de mettre en œuvre un processus d'intégration et d'interopérabilité des flux d'information et des bases de données actuellement à disposition du système public-privé afin de refléter pleinement l'offre touristique territoriale de chaque pays / région.

Pour atteindre cet objectif, il est impératif de définir et partager une architecture informatique commune incluant des interfaces de programmation (API) unifiées.

Basé sur un système qui s'affranchira des frontières, les API permettront de publier des données de chaque territoire et les rendre accessibles aux acteurs privés, aux institutions publiques et aux touristes.

Cette disponibilité partagée a pour objectif de faciliter la commercialisation par des entreprises, des produits transfrontaliers et de permettre aux usagers d'utiliser des applications touristiques qui fonctionnent au-delà des frontières.

B. Périmètre du document

Ce document fait partie de la composante T2 - Conception de l'architecture générale de l'écosystème numérique intégré de Smart Destination, et plus particulièrement de l'activité préliminaire T2.2 - Définition de l'architecture et des fonctionnalités générales, des protocoles d'interopérabilité et des interfaces.

La composante T2 est un enjeu majeur pour le projet par son double objectif de:

- Concevoir l'architecture technologique du concept SMART DESTINATION;
- Mettre en place les bases de la conclusion de futurs accords transfrontaliers spécifiques et des feuilles de route pour la mise en œuvre de cette stratégie.

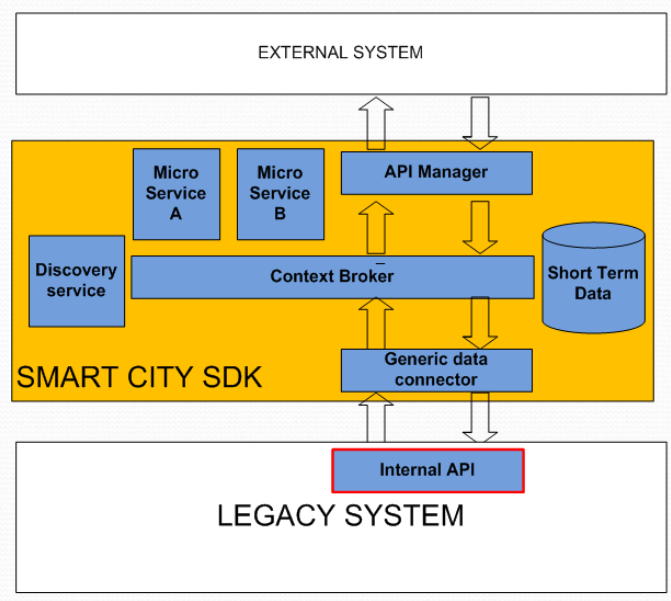
La composante T2 est constituée de 5 tâches dont le tableau page suivante rappelle les grandes lignes. Le document ne traite que de la tâche T2.2 et du livrable T2.2.3.

Ce document est le troisième volet de l'activité T2.2. Il décrit l'architecture de l'écosystème numérique transfrontalier et s'appuie sur le rapport T2.2.2: Spécifications fonctionnelles des composants de l'écosystème numérique transfrontalier.

Le rapport T2.2.2 a décrit les spécifications fonctionnelles des composants de

l'écosystème et plus particulièrement celle du Generic Data Connector (Cf. III.B.3) et de l'API Manager (Cf. III.B.4). Ces deux composats permettent de s'interfacer et d'échanger des données avec la plateforme «Smart Destination».

La plateforme « Smart Destination » rassemble l'ensemble des données fournies par tous les fournisseurs de données («Legacy System») et les rend disponibles par API aux applications clientes («External system»).



Graphique 1 - Architecture Smart Destination

Le tableau ci-dessous rappelle les 5 tâches de la composante T2 et effectue un zoom sur la tâche T2.2 et de de ses 3 livrables.

<table>
<tr><th>Tâches</th><th colspan="2">Description</th></tr>
<tr><td>T2.1</td><td colspan="2">Conception et architecture d'écosystèmes numériques transfrontaliers
Analyse des systèmes régionaux (portails, applications et systèmes de gouvernance) et l'identification des meilleures pratiques et sources de données</td></tr>
<tr><td>T2.2</td><td colspan="2">Définition de l'architecture et des fonctionnalités générales, des protocoles d'interopérabilité et des interfaces
Définition de l'architecture et des fonctionnalités générales, des protocoles d'interopérabilité et des interfaces de l'écosystème numérique ouvert et distribué permettant aux applications de service d'accéder, à l'aide d'un format et d'un protocole unifiés, aux SI touristique des différents acteurs territoriaux publics et privés.
<table>
<tr><th>Livrable</th><th>Description</th></tr>
<tr><td>T2.2.1</td><td>Document
Architecture de l'écosystème numérique transfrontalier</td></tr>
<tr><td>T2.2.2</td><td>Rapport
Spécifications fonctionnelles des composants de l'écosystème numérique transfrontalier</td></tr>
<tr><td>T2.2.3</td><td>Rapport
Définition des protocoles et interfaces d'interopérabilité</td></tr>
</table></td></tr>
<tr><td>T2.3</td><td>Accord stratégique transfrontalier et plan d'action.
Définition de l'état de l'art et des accords sur la base des API sélectionnées mis en commun.</td><td rowspan="3">// à T2.1 à T2.3</td></tr>
<tr><td>T2.4</td><td>Chemins d'achèvement du système régional
Activité parallèle sur chaque territoire afin de définir les itinéraires spécifiques de réorientation de son écosystème touristique régional.</td></tr>
<tr><td>T2.5</td><td>Rapport - achèvement des fonctionnalités à mettre en œuvre
Description des itinéraires de chaque écosystème touristique régional dans la mise en œuvre des orientations techniques et caractéristiques définies.</td><td>Unitaire</td></tr>
</table>

Tableau 1 - Rappel des activités de la composante T2

II. ENSEMBLE D'INTERFACES DE PROGRAMMATION D'APPLICATIONS (API)

A. Les modèles de données

Le rapport T2.1.3 a mis en avant au chapitre 4 *(§4 Aspetto qualitativo, quantitativo e semantico)* que la conception du SDK permettant de construire la plateforme Smart Destination doit inclure:

- La définition d'un format de représentation unique pour chaque attribut des différentes entités. Lors du développement de la couche d'adaptation entre le système régional et le SDK, il sera de la responsabilité de chaque partenaire de respecter le format imposé.
- La définition de la liste des entités pouvant être représentées et des attributs relatifs. Dans ce cas, chaque partenaire, ayant une connaissance maximale de son système régional, doit proposer la meilleure cartographie entre son système régional et les données requises par le SDK.
- Sur la fréquence des mises à jour et des modalités de saisie des données, en particulier une analyse approfondie par l'ensemble des partenaires sur ces deux aspects est nécessaire afin d'assurer la cohérence des résultats fournis par la plate-forme Smart Destination.

Le format de représentation unique des données respecte le modèle d'échange de données standard FIWARE et met en œuvre les modèles de données NGSI FIWARE. FIWARE propose des modèles de données harmonisés.

Les modèles de données FIWARE sont consultables sur le site FIWARE (https://fiware-datamodels.readthedocs.io/en/test_next_version/index.html).

A ce jour quatre modèles de données concernent le tourisme :

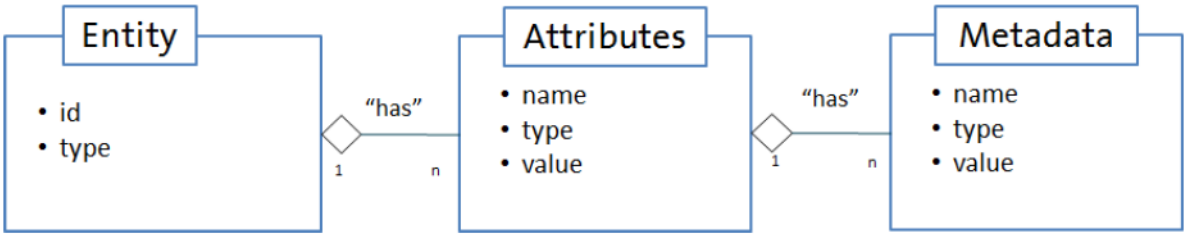
- Les points d'intérêt (https://fiware-datamodels.readthedocs.io/en/test_next_version/PointOfInterest/PointOfInterest/doc/spec/index.html);
- Les musées (https://fiware-datamodels.readthedocs.io/en/test_next_version/PointOfInterest/Museum/doc/spec/index.html);
- Les plages (https://fiware-datamodels.readthedocs.io/en/test_next_version/PointOfInterest/Beach/doc/spec/index.html);
- Les parkings (https://fiware-datamodels.readthedocs.io/en/test_next_version/Parking/doc/introduction/inde

x.html);

Afin de répondre aux enjeux de « Smart Destination » de nouveaux modèles de données seront proposés à l'issu de la définition des démonstrateurs.

Afin de modéliser les données touristiques FIWARE propose un ensemble de lignes directrices pour la définition de nouveaux modèles de données (https://fiware-datamodels.readthedocs.io/en/test_next_version/guidelines/index.html).

La figure ci-dessous décrit le méta-modèle NGSI. Ce méta-modèle comprend trois éléments principaux, comme le montre la figure ci-dessous: Entités, Attributs et Métadonnées.



Graphique 2 - OMA NGSI meta-model

Une entité est représentée par un objet JSON avec la structure suivante:

- L'ID de l'entité est codée par la propriété id de l'objet.
- Le type d'entité est codé par la propriété type de l'objet
- Les attributs d'entité sont spécifiés par des propriétés supplémentaires, dont les noms sont le nom de l'attribut et dont la représentation est décrite ci-dessous.

Un attribut est représenté par un objet JSON avec la syntaxe suivante:

- La valeur de l'attribut est spécifiée par la propriété value, qui peut être une valeur JSON quelconque.
- Le type de l'attribut NGSI est spécifié par la propriété type.

Les métadonnées d'attribut sont spécifiées par la propriété «metadata». La valeur de la métadonnée est un autre objet JSON qui contient une propriété par élément de métadonnée défini (le nom de la propriété est le nom de l'élément de métadonnée). Chaque élément de métadonnée, à son tour, est représenté par un objet JSON contenant les propriétés suivantes:

- La valeur de la métadonnée est spécifiée par la propriété value, qui peut correspondre à une valeur JSON quelconque.

- Le type de la métadonnée est spécifié par la propriété type.

L'implémentation de référence FIWARE définit certaines restrictions syntaxiques supplémentaires de l'API NGSI sur les identifiants d'entité, les types ou les noms d'attributs / métadonnées (http://fiware.github.io/specifications/ngsiv2/latest/).

L'exemple ci-dessous représente une entité contenant une description géographique harmonisée d'un musée.


```
{
      "id": "Museum:TOURIS-OTMETR-MAMAC",
      "type": "Museum",
      "address": {
         "addressCountry": "Fr",
         "postalCode": "06300",
         "addressLocality": "Nice",
         "streetAddress": "Place Yves Klein"
      },
      "alternateName": "M.AM.A.C.",
      "artPeriod": [
         "contemporary"
      ],
      "description": "Les collections du musée, près de 1300 oeuvres, témoignent du rôle important joué par Nice dans le développement des mouvements novateurs des années 60 et 70, à nos jours : les Nouveaux Réalistes - César, Arman, Niki de Saint-Phalle, le Pop'Art - Andy Warhol, Wesselmann, le groupe Fluxus - Ben Vautier, l%27Art Conceptuel, l%27Ecole de Nice, le groupe Supports-Surfaces, mais aussi des individualités niçoises comme Ernest Pignon-Ernest ou Claude Gilli. Le musée accueille de nombreuses expositions temporaires.",
      "facilities": [
         "shop",
         "cloakRoom",
         "guidedTour"
      ],
      "location": {
         "type": "Point",
         "coordinates": [7.2786, 43.7014]
      },
      "museumType": [
         "fineArts"
      ],
      "name": "MUSÉE D%27ART MODERNE ET D%27ART CONTEMPORAIN M.A.M.A.C",
      "openingHoursSpecification": [
         {
            "dayOfWeek": "Tuesday",
            "closes": "18:00",
```

```
            "opens": "10:00",
            "validFrom": "--06-23",
            "validThrough": "--10-15"
        },
        {
            "dayOfWeek": "Wednesday",
            "closes": "18:00",
            "opens": "10:00",
            "validFrom": "--06-23",
            "validThrough": "--10-15"
        },
        {
            "dayOfWeek": "Thursday",
            "closes": "18:00",
            "opens": "10:00",
            "validFrom": "--06-23",
            "validThrough": "--10-15"
        },
        {
            "dayOfWeek": "Friday",
            "closes": "18:00",
            "opens": "10:00",
            "validFrom": "--06-23",
            "validThrough": "--10-15"
        },
        {
            "dayOfWeek": "Saturday",
            "closes": "18:00",
            "opens": "10:00",
            "validFrom": "--06-23",
            "validThrough": "--10-15"
        },
        {
            "dayOfWeek": "Sunday",
            "closes": "18:00",
            "opens": "10:00",
            "validFrom": "--06-23",
            "validThrough": "--10-15"
        },
        {
            "dayOfWeek": "Tuesday",
            "closes": "18:00",
            "opens": "11:00",
            "validFrom": "--10-16",
            "validThrough": "--06-22"
        },
        {
```

```
            "dayOfWeek": "Wednesday",
            "closes": "18:00",
            "opens": "11:00",
            "validFrom": "--06-23",
            "validThrough": "--10-15"
        },
        {
            "dayOfWeek": "Thursday",
            "closes": "18:00",
            "opens": "11:00",
            "validFrom": "--06-23",
            "validThrough": "--10-15"
        },
        {
            "dayOfWeek": "Friday",
            "closes": "18:00",
            "opens": "11:00",
            "validFrom": "--06-23",
            "validThrough": "--10-15"
        },
        {
            "dayOfWeek": "Saturday",
            "closes": "18:00",
            "opens": "11:00",
            "validFrom": "--06-23",
            "validThrough": "--10-15"
        },
        {
            "dayOfWeek": "Sunday",
            "closes": "18:00",
            "opens": "11:00",
            "validFrom": "--06-23",
            "validThrough": "--10-15"
        }
    ],
    "source": "http://www.mamac-nice.org/",
    "touristArea": "Acropolis"
}
```

B. Les API du Generic Data Connector

Le Generic Data Connector est une interface ouverte destinée à être utilisée dans un large éventail de scénarios et de services d'applications. Il permet d'offrir aux systèmes d'information existants «Legacy System» différents protocoles de communications.

Les fournisseurs de données («Legacy System») publient régulièrement la mise à jour des informations relatives à leurs données touristiques (POI, Hébergements, Restauration, Evènements …). Pour cela les fournisseurs de données utilisent l'API du «Generic Data Connector» pour transmettre leurs données dans un format normalisé (Normalized Data Model). Le «Generic Data Connector» propose plusieurs protocoles d'échange afin de faciliter l'interconnexion et les échanges avec les producteurs de données. Le «Generic Data Connector» aura la charge de transmettre au «Context Broker» les données reçus. Ce dernier aura la charge après vérification des autorisations et habilitations, d'enregistrer les données transmises.

Le Generic Data Connector traduit les protocoles spécifiques mis en œuvre en protocole d'informations de contexte NGSI, c'est-à-dire le modèle d'échange de données standard FIWARE. Ce composant n'est pas nécessaire si les «Legacy system» prennent en charge de manière native l'API NGSI. Le Generic Data Connector supporte les protocoles suivants:

- JSON over HTTP;
- JSON over MQTT;
- UL2.0 over HTTP;
- UL2.0 over MQTT.

1. JSON

JSON (JavaScript Object Notation) est un format d'échange de données léger. C'est un format facile à lire et à écrire. Il est basé sur un sous-ensemble du langage de programmation JavaScript, norme ECMA-262, 3e édition - Décembre 1999. Le format JSON est un format de texte totalement indépendant du langage, qui utilise des conventions bien connues des programmeurs.

JSON est construit sur deux structures (https://JSON.org/):

- Une collection de paires nom / valeur. Dans différentes langues, ceci est réalisé sous forme d'objet, d'enregistrement, de structure, de dictionnaire, de

table de hachage, de liste à clé ou de tableau associatif.
- Une liste ordonnée de valeurs. Dans la plupart des langues, cela est réalisé sous forme de tableau, de vecteur, de liste ou de séquence.

Les modèles de données FIWARE sont basés sur le format JSON.

Les modèles de données peuvent être créés, modifiés ou supprimés en respectant les spécifications FIWARE-NGSI v2.

Toutes les spécifications sont disponibles sur le site FIWARE à l'adresse suivante: http://fiware.github.io/specifications/ngsiv2/stable/.

a) JSON over HTTP

L'API NGSI permet de mettre en œuvre différentes opérations, dont voici les principales:

- Création d'entité

Les requêtes de création utilisent le verbe http POST.

Le corps du message est un objet représentant l'entité à créer. L'objet suit le format de représentation d'entité JSON.

L'opération réussie retourne «201 Created» (si l'option upsert n'est pas utilisée) ou «204 No Content» (si l'option upsert est utilisée). La réponse comprend également un en-tête Location avec l'URL de l'entité créée. Les erreurs utilisent une réponse non 2xx et retournent éventuellement une erreur dans le corps du message.

- Mise à jour des attributs existants d'une entité

Les requêtes de mise à jour utilisent le verbe http PATCH.

Le corps du message est un objet représentant les attributs à mettre à jour. L'objet suit le format de représentation d'entité JSON, sauf que l'id et le type ne sont pas autorisés.

Les attributs d'entité sont mis à jour avec ceux du corps du message. De plus, si un ou plusieurs attributs du corps du message n'existent pas dans l'entité, une erreur est renvoyée.

L'opération réussie retourne «204 No Content». Les erreurs utilisent une réponse non 2xx et retournent éventuellement une erreur dans le corps du message.

- Remplacement des attributs existants d'une entité

Les requêtes de mise à jour utilisent le verbe http PUT.

Le corps du message est un objet représentant les nouveaux attributs d'entité. L'objet suit le format de représentation d'entité JSON, sauf que l'id et le type ne sont pas autorisés.

Les attributs précédemment existants dans l'entité sont supprimés et remplacés par ceux de la demande.

L'opération réussie retourne «204 No Content». Les erreurs utilisent une réponse non 2xx et retournent éventuellement une erreur dans le corps du message.

- Suppression d'entité

Les requêtes de suppression utilisent le verbe http DELETE.

L'opération réussie retourne «204 No Content». Les erreurs utilisent une réponse non 2xx et retournent éventuellement une erreur dans le corps du message.

- Consultation d'entité

Les requêtes de consultation utilisent le verbe http GET

La réponse est un objet représentant l'entité identifiée par l'ID. L'objet suit le format de représentation d'entité JSON.

Cette opération renvoi un seul élément d'entité, mais plusieurs entités peuvent avoir le même ID (par exemple, des entités avec le même ID mais de types différents). Dans ce cas, un message d'erreur est renvoyé, avec le code d'état HTTP défini à «409 Conflict».

L'opération réussie retourne «200 OK». Les erreurs utilisent une réponse non 2xx et retournent éventuellement une erreur dans le corps du message.

b) JSON over MQTT

MQTT (Message Queuing Telemetry Transport) est un protocole de messagerie de type publish-subscribe.

MQTT dispose de plusieurs atouts:

- Son en-tête très léger ne pèse que 2 bytes, il dispose de capacités bidirectionnelles, et repose sur un modèle publish/subscribe permettant de collecter plus de données que les protocoles de type polling tout en

consommant moins de bande passante;
- L'efficience, la scalabilité, la réduction de consommation de la bande passante du réseau.

Dans le cas de JSON over MQTT, les entités au format JSON sont envoyées par le «Legacy System» dans un topic MQTT dédié à ce type d'entité. De son côté le «Generic Data Connector» a la charge de récupérer les données du topic et de les mettre à jour dans la plateforme. Pour cela le «Generic Data Connector» utilise les requêtes de création avec l'option upsert (permet de mettre à jour si l'entité existe déjà).

```
mosquitto_pub -t "Museum" -u username -P p@ssW0rd -h mqtt.domaine.org -p 8883
-d --cafile .\keys\capem.crt -m "{"id":"Museum:TOURIS-OTMETR-MAMAC",
"type":"Museum", "address":{"addressCountry":"Fr", "postalCode":"06300",
"addressLocality":"Nice", "streetAddress":"Place Yves Klein"},
"alternateName":"M.AM.A.C.", "artPeriod":["contemporary"]}"
```

Graphique 3 - JSON Message over MQTT

2. UL2.0

Ultralight 2.0 est un protocole texte léger permettant de réduire le volume de données transférées.

Le corps du message pour les demandes de mise à jour d'informations est composé d'une liste de paires clé-valeur séparées par le symbole « | ».

Dans l'exemple ci-dessous (cf. Grafico 4), deux attributs, l'un nommé « n » avec la valeur «MUSÉE D%27ART MODERNE ET D%27ART CONTEMPORAIN M.A.M.A.C» et l'autre appelé «an» avec la valeur «M.AM.A.C» sont transmis. Les valeurs dans Ultralight 2.0 ne sont pas typées (tout est traité comme une chaîne).

Le «Generic Data Connector» a la charge de récupérer le message et de créer le JSON correspondant à l'entité. C'est ce JSON qui est alors utilisé pour créer ou mettre à jour l'entité. Le «Generic Data Connector» utilise les requêtes de création avec l'option upsert (permet de mettre à jour si l'entité existe déjà).

De même que de nouveaux modèles de données seront proposés à l'issu de la définition des démonstrateurs, les modèles de mapping seront définis (cf. Graphique 5).

```
i|Museum:TOURIS-OTMETR-MAMAC|n|MUSÉE D%27ART MODERNE ET D%27ART
```

```
CONTEMPORAIN M.A.M.A.C|an|M.AM.A.C.
```

Graphique 4 - UL2.0 message

```
{
   "attributes": [
      {"object_id": "id", "name": "id", "type":"Text"},
      { "object_id": "n", "name": "name", "type": "Text" },
      { "object_id": "an", "name": "alternateName", "type": "Text" }
   ]
}
```

Graphique 5 - Mapping UL2.0

a) UL2.0 over HTTP

Dans le cas de UL2.0 over HTTP, les entités au format UL2.0 sont envoyées par le «Legacy System» en HTTP et le verbe http POST. De son côté le «Generic Data Connector» a la charge de mettre à jour les entités dans la plateforme. Pour cela le «Generic Data Connector» crée le JSON correspondant à l'entité et utilise les requêtes de création avec l'option upsert (permet de mettre à jour si l'entité existe déjà).


```
curl -iX POST \
  'http://localhost /Museum' \
  -H 'Content-Type: text/plain' \
  -d 'i|Museum:TOURIS-OTMETR-MAMAC|n|MUSÉE D%27ART MODERNE ET D%27ART
CONTEMPORAIN M.A.M.A.C|an|M.AM.A.C.'
```

Graphique 6 - Message UL2.0 over HTTP

b) UL2.0 over MQTT

Dans le cas de UL2.0 over MQTT, les entités au format UL2.0 sont envoyées par le «Legacy System» dans un topic MQTT dédié à ce type d'entité. De son côté le «Generic Data Connector» a la charge de récupérer les données du topic et de les mettre à jour dans la plateforme. Pour cela le «Generic Data Connector» crée le JSON correspondant à l'entité et utilise les requêtes de création avec l'option upsert (permet de mettre à jour si l'entité existe déjà).

```
mosquitto_pub -t "Museum" -u username -P p@ssW0rd -h mqtt.domaine.org -p 8883
```

```
-d --cafile .\keys\capem.crt -m "i|Museum:TOURIS-OTMETR-MAMAC|n|MUSÉE
D%27ART MODERNE ET D%27ART CONTEMPORAIN M.A.M.A.C|an|M.AM.A.C. "
```

Graphique 7 - Message UL2.0 over MQTT

C. Les API de l'API Manager

Le gestionnaire d'API est un proxy placé devant les API (CEF Context Broker, Micro-Services).

Il doit permettre d'ajouter des fonctionnalités communes telles que les clés d'API, la limitation de débit et les statistiques d'usage de toutes les API.

Les requêtes de consultation utilisent le verbe http GET. L'API Key doit être passer à chaque requête, elle permet de limiter les droits d'accès. Elle peut être transmise soit dans la requête, soit dans l'en-tête.

- Name: api_key In: query
- Name: X-API-Key In: header

L'API permet:

- D'obtenir en retour l'ensemble des entités d'un domaine (ex : Museum) ;
- D'obtenir en retour une entité spécifique ;
- D'utiliser des filtres pour limiter les réponses.

1. Requêter l'ensemble des entités

Pour obtenir toutes les entités il suffit d'utiliser l'opération GET /

```
curl api.domain.org/ -s -S -H 'Accept: application/json'
```

Dans notre cas, deux musées sont retournés Museum:TOURIS-OTMETR-MAMAC et Museum:TOURIS-OTMETR-MATISSE.

```
[
  {
    "id": "Museum:TOURIS-OTMETR-MAMAC",
    "type": "Museum",
    "name": {
      "type": "Text",
```

```
        "value": "MUSÉE D%27ART MODERNE ET D%27ART CONTEMPORAIN M.A.M.A.C"
      },
      "alternateName": {
         "type": "Text",
         "value": "M.AM.A.C."
      } ,
      "location": {
         "type": "geo:json",
         "value": {
            "type": "Point",
            "coordinates": [
               7.2786,
               43.7014
            ]
         }
      }
   },
   {
      "id": "Museum:TOURIS-OTMETR-MATISSE",
      "type": "Museum",
      "name": {
         "type": "Text",
         "value": "MUSÉE MATISSE"
      } ,
      "location": {
         "type": "geo:json",
         "value": {
            "type": "Point",
            "coordinates": [
               7.27524,
               43.7196
            ]
         }
      }
   }
]
```


Graphique 8 - Obtenir toutes les entités

2. Requêter une entité

Pour obtenir une entité il suffit d'utiliser l'opération GET /{id}

```
curl api.domain.org/Museum:TOURIS-OTMETR-MAMAC -s -S -H 'Accept: application/json'
```

Dans notre cas, un seul musée est retourné Museum:TOURIS-OTMETR-MAMAC.


```
{
      "id": "Museum:TOURIS-OTMETR-MAMAC",
      "type": "Museum",
      "name": {
         "type": "Text",
         "value": "MUSÉE D%27ART MODERNE ET D%27ART CONTEMPORAIN M.A.M.A.C"
      },
      "alternateName": {
         "type": "Text",
         "value": "M.AM.A.C."
      } ,
      "location": {
         "type": "geo:json",
         "value": {
            "type": "Point",
            "coordinates": [
               7.2786,
               43.7014
            ]
         }
      }
}
```


Graphique 9 - Obtenir une entité


```
FiltrerAIN M.A.M.A.C"
      },
      "alternateName": {
         "type": "Text",
         "value": "M.AM.A.C."
      } ,
      "location": {
```

```
        "type": "geo:json",
        "value": {
            "type": "Point",
            "coordinates": [
                7.2786,
                43.7014
            ]
        }
    }
}
```


Graphique 10 - Obtenir une entité

3. Filtrer

a) Option keyValues

L'option keyValues peut être utilisée afin d'obtenir une représentation plus compacte et plus brève.

Pour obtenir une entité compacte il suffit d'utiliser l'opération GET /{id}?options=keyValues

```
curl api.domain.org/Museum:TOURIS-OTMETR-MAMAC?options=keyValues -s -S -H 'Accept: application/json'
```

Dans notre cas, un seul musée est retourné Museum:TOURIS-OTMETR-MAMAC.


```
{
   "id": "Museum:TOURIS-OTMETR-MAMAC",
   "type": "Museum",
   "alternateName": "M.AM.A.C.",
   "name": "MUSÉE D%27ART MODERNE ET D%27ART CONTEMPORAIN M.A.M.A.C",
   "location": {
      "type": "Point",
      "coordinates": [
         7.2786,
         43.7014
      ]
   }
```

```
}
```

Graphique 11 - Obtenir une entité compacte

b) Option values / attrs

Vous pouvez également utiliser l'option values afin d'obtenir une représentation encore plus compacte correspondant à une liste de valeurs d'attributs. Dans ce cas, le paramètre URL attrs doit être utilisé pour spécifier l'ordre. Par exemple, pour obtenir le nom en premier, puis la géolocalisation.

Pour obtenir une entité compacte il suffit d'utiliser l'opération GET /{id}?options=values&attrs=name,location

```
curl api.domain.org/Museum:TOURIS-OTMETR-MAMAC?options=values&attrs=name,location -s -S -H 'Accept: application/json'
```

Dans notre cas, un seul musée est retourné Museum:TOURIS-OTMETR-MAMAC.


```
[
  "MUSÉE D%27ART MODERNE ET D%27ART CONTEMPORAIN M.A.M.A.C",
  {
    "type": "Point",
    "coordinates": [
      7.2786,
      43.7014
    ]
  }
]
```

Graphique 12 - Obtenir des valeurs d'entité compactes

c) Option attrs

Vous pouvez également demander un seul attribut, en utilisant l'opération /{id}/attrs/{attrsName}.

```
curl api.domain.org/Museum:TOURIS-OTMETR-MAMAC/attrs/location -s -S -H 'Accept: application/json'
```

Dans notre cas, un seul musée est retourné Museum:TOURIS-OTMETR-MAMAC.

```
[
   {
      "type": "Point",
      "coordinates": [
         7.2786,
         43.7014
      ]
   }
]
```


Graphique 13 - Obtenir un attribut compact d'entité

d) Option attrs/value

Vous pouvez également demander une seule valeur d'attribut, en utilisant l'opération /{id}/attrs/{attrsName}/value.

```
curl api.domain.org/Museum:TOURIS-OTMETR-MAMAC/attrs/location/value -s -S -H 'Accept: application/json'
```

Dans notre cas, un seul musée est retourné Museum:TOURIS-OTMETR-MAMAC.


```
{
   "type": "Point",
   "coordinates": [
      7.2786,
      43.7014
   ]
}
```


Graphique 14 - Obtenir une valeur d'attribut d'entité compacte

e) Option idPattern

Vous pouvez filtrer à l'aide de modèles d'identité d'entité, à l'aide du paramètre URL idPattern (dont la valeur est une expression régulière). Par exemple, pour obtenir toutes les entités dont l'id commence par Museum:TOURIS-OTMETR-MAM et est suivi de plusieurs lettres comprises entre A et Z (dans ce cas, récupérer Museum:TOURIS-OTMETR-MAMAC), vous pouvez utiliser:

```
curl api.domain.org/?idPattern=^Museum:TOURIS-OTMETR-MAM[A-Z]  -g -s -S -H 'Accept: application/json'
```

Dans notre cas, un seul musée est retourné Museum:TOURIS-OTMETR-MAMAC.


```
{
      "id": "Museum:TOURIS-OTMETR-MAMAC",
      "type": "Museum",
      "name": {
         "type": "Text",
         "value": "MUSÉE D%27ART MODERNE ET D%27ART CONTEMPORAIN M.A.M.A.C"
      },
      "alternateName": {
         "type": "Text",
         "value": "M.AM.A.C."
      } ,
      "location": {
         "type": "geo:json",
         "value": {
            "type": "Point",
            "coordinates": [
               7.2786,
               43.7014
            ]
         }
      }
}
```


Graphique 15 - Filtre à motifs

f) Option q

Vous pouvez filtrer à l'aide de filtres d'attributs, à l'aide du paramètre d'URL q. Pour une description complète, consultez la section "Langage de requête simple" de la spécification NGSv2. Par exemple, pour obtenir toutes les entités dont l'attribut alternateName est égal à « M.AM.A.C. » vous pouvez utiliser:

```
curl api.domain.org/?q=alternateName==M.AM.A.C. -s -S -H 'Accept: application/json'
```

Dans notre cas, un seul musée est retourné Museum:TOURIS-OTMETR-MAMAC.


```
{
      "id": "Museum:TOURIS-OTMETR-MAMAC",
      "type": "Museum",
```

```
        "name": {
            "type": "Text",
            "value": "MUSÉE D%27ART MODERNE ET D%27ART CONTEMPORAIN
M.A.M.A.C"
        },
        "alternateName": {
            "type": "Text",
            "value": "M.AM.A.C."
        } ,
        "location": {
            "type": "geo:json",
            "value": {
                "type": "Point",
                "coordinates": [
                    7.2786,
                    43.7014
                ]
            }
        }
}
```


Graphique 16 - Filtre q

g) Filter entities by distance

Vous pouvez filtrer à l'aide du filtre géographique.

georel: Requêtes géolocalisées.

- near - la relation signifie que les entités correspondantes doivent être situées à une certaine distance seuil par rapport à la géométrie de référence. Il prend en charge les modificateurs suivants.
  - maxDistance expressions, en mètres, la distance maximale à laquelle les entités correspondantes doivent être situées.
  - minDistance expressions, en mètres, la distance minimale à laquelle les entités correspondantes doivent être situées.
- coveredBy - indique que les entités correspondantes sont celles qui existent entièrement dans la géométrie de référence.
- intersects - indique que les entités correspondantes sont celles qui se croisent avec la géométrie de référence.
- equals - la géométrie associée à la position des entités correspondantes et la géométrie de référence doivent être exactement les mêmes.

- disjoint - indique que les entités correspondantes sont celles qui ne se croisent pas avec la géométrie de référence.

geometry : Permet de définir la forme de référence à utiliser lors de la résolution de la requête en utilisant les géométries suivantes:

- point - définit un point sur la surface de la Terre.
- line - définit une ligne polygonale.
- polygon - définit un polygone.
- box - définit une boîte de délimitation.

coords: Doit être une chaîne contenant une liste de paires de coordonnées géographiques séparées par les points-virgules, conformément à la géométrie spécifiée et aux règles prescrites par le format de localisation simple :

- point - coords contient une paire de géo-coordonnées WGS-84.
- line - coords contient une liste de paires de géo-coordonnées WGS-84
- polygon - coords est composé d'au moins quatre paires de géo-coordonnées WGS-84.
- box - coords est composé de deux paires de géo-coordonnées WGS-84.

    - e.g. georel=near;maxDistance:1000&geometry=point&coords=43.69973,7.2832187
    - e.g. georel=coveredBy&geometry=polygon&coords=43.630350111919306,7.12125406077034;43.62437257664505,7.275871196782733;43.79958168232838,7.288983949701983;43.8055782213152,7.133894415565227;43.630350111919306,7.12125406077034

```
curl api.domain.org/?georel=near;maxDistance:1000&geometry=point&coords=43.71271,7.27858 -s -S -H 'Accept: application/json'
```

Dans notre cas, un seul musée est retourné Museum:TOURIS-OTMETR-MAMAC.


```
{
    "id": "Museum:TOURIS-OTMETR-MAMAC",
    "type": "Museum",
    "name": {
        "type": "Text",
        "value": "MUSÉE D%27ART MODERNE ET D%27ART CONTEMPORAIN
```

```
M.A.M.A.C"
        },
        "alternateName": {
            "type": "Text",
            "value": "M.AM.A.C."
        } ,
        "location": {
            "type": "geo:json",
            "value": {
                "type": "Point",
                "coordinates": [
                    7.2786,
                    43.7014
                ]
            }
        }
}
```

Graphique 17 - Filtre qéo