REGARDS,
un framework complet pour la gestion pérenne de nos jeux de données spatiales

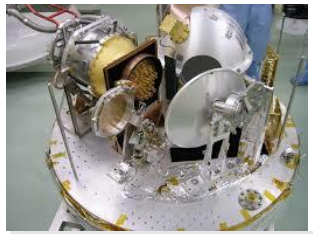
Premiers regards ;)

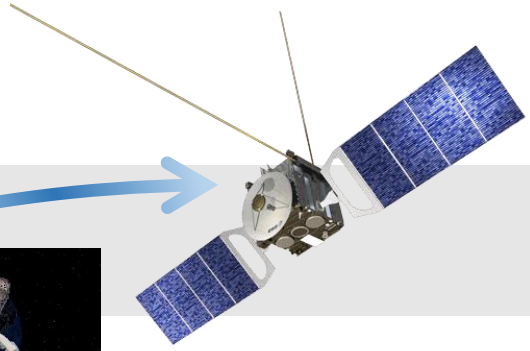
Hervé Ballans & Marc Dexet
Institut d'Astrophysique Spatiale

L'Institut d'Astrophysique Spatiale

- **5 équipes de recherche**
  - Astrochimie et Origines
  - Astrophysique de la Matière Interstellaire
  - Cosmologie et eXtragalactique
  - Physique Solaire et Stellaire
  - Système Solaire et Systèmes Planétaires
- L'IAS fait partie de **l'Observatoire des Sciences de l'Univers de l'Université Paris-Sud** (OSUPS)
- Equipes techniques pour la **conception et la réalisation d'instruments spatiaux**
- **Plateforme technique IDOC**
- Partenaires : autres laboratoires spatiaux, agences spatiales, industriels,…

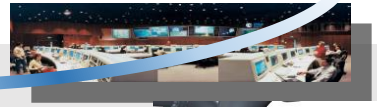
IDOC et domaines d'activités





Avant le lancement

0) Conception et tests de l'instrument



1) Instrument Operations

Segments sols

Autres fournisseurs de données

2) Pipelines

Gestion des données

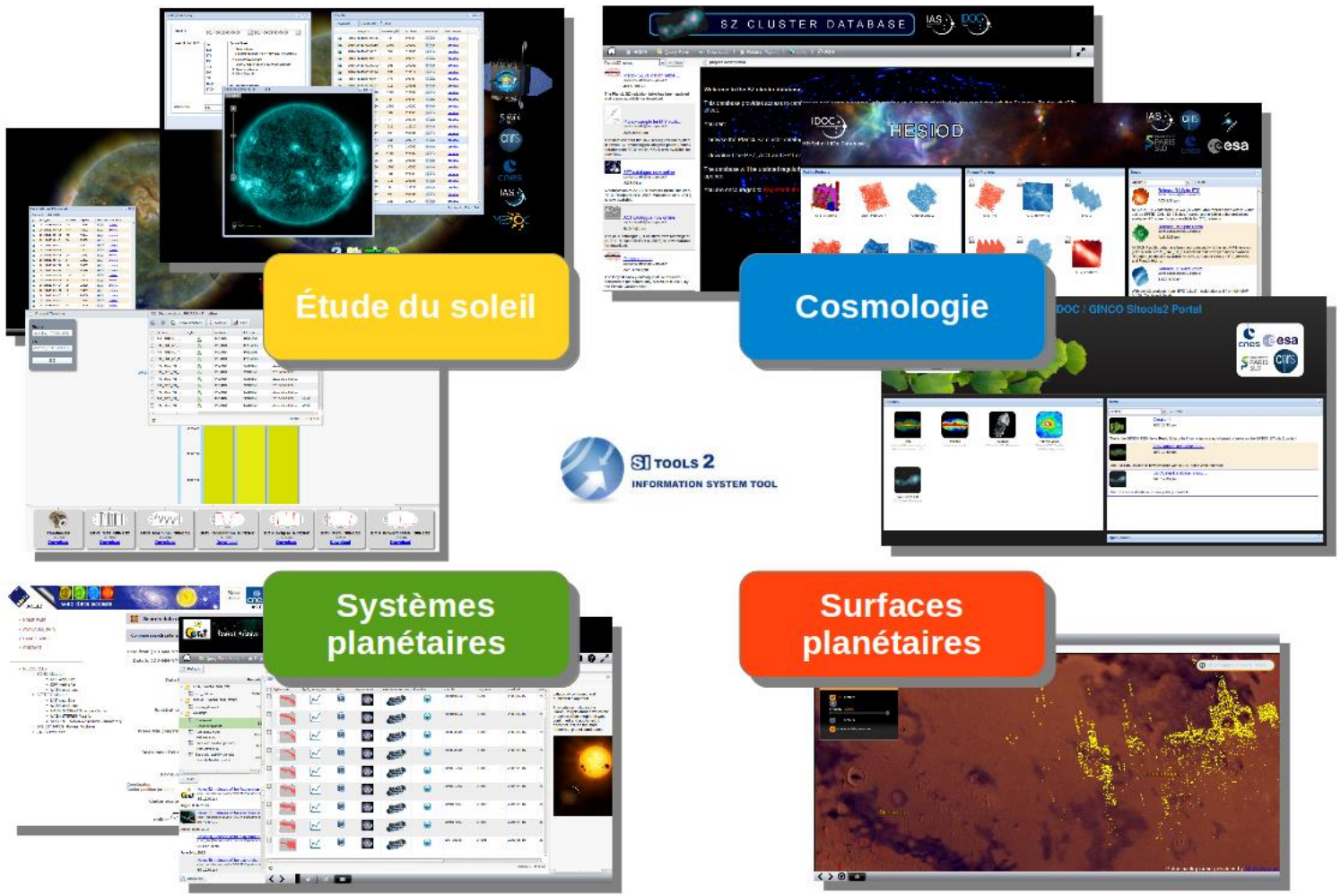
3) Jeux de données, interfaces, outils de traitement, observatoires virtuels

4) Archivage moyen et long terme

Communautés scientifiques

Grand Public

Les interfaces d'accès aux données spatiales



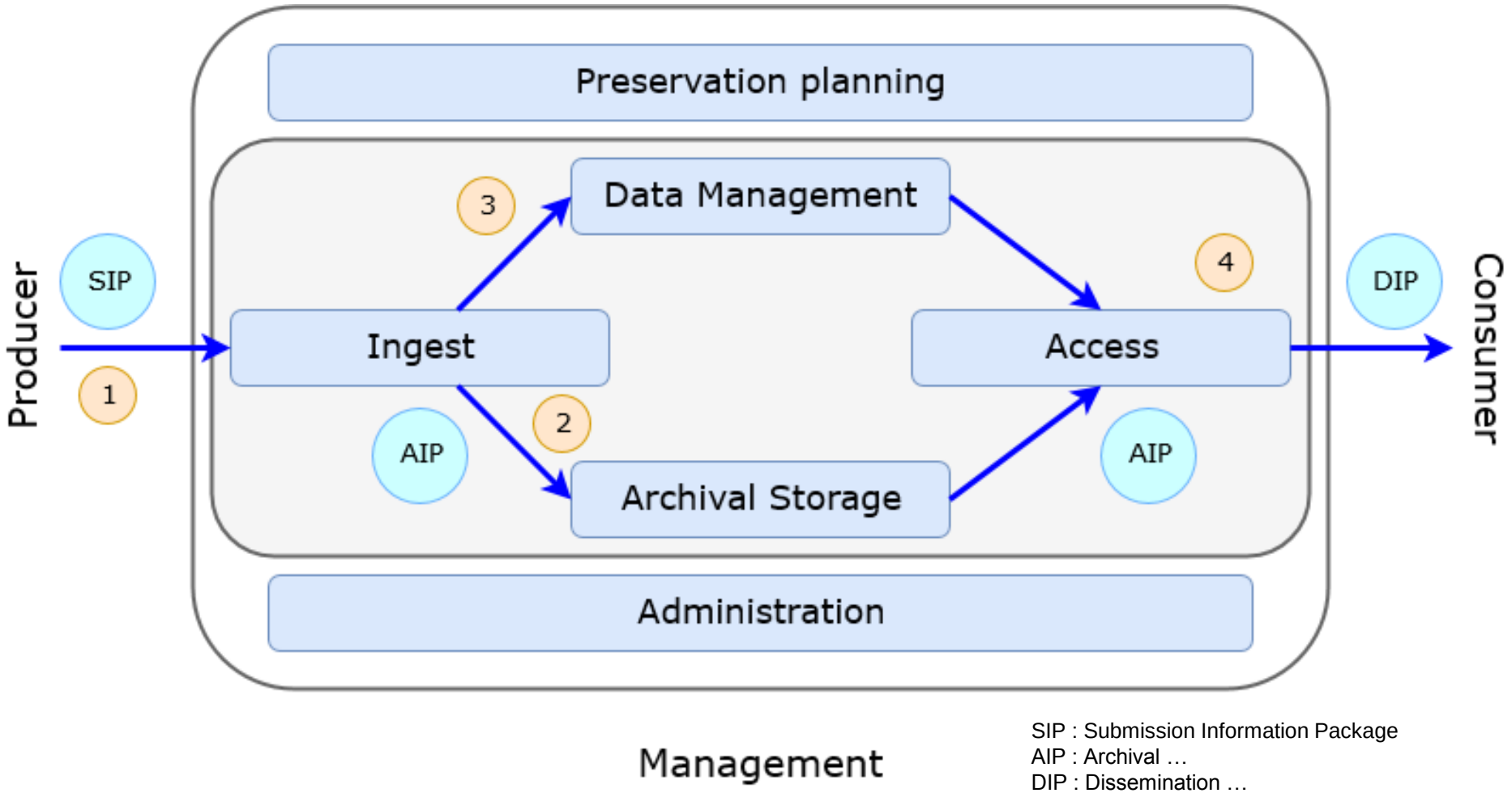
Outil actuel : SITools2

- Outil générique, adaptable, qui répond très bien au besoin de mise à disposition des données
- Développement et intégration de plugins (VO ConeSearch, CubeExplorer,…)
- Limitations techniques :
  - Uniquement compatible avec Java7
  - Librairies clientes devenues obsolètes (Sencha ExtJS 4)
  - Recherche uniquement par formulaire ou exploration de jeux de données
  - Fin de maintenance prévue pour 2019
- Mais aussi limitations fonctionnelles face aux nouvelles exigences de nos partenaires

La pérennisation des données : apporter une réponse standardisée

- Exemple avec la mission solaire PICARD (2010/2014)
  - Exigence du CNES pour la pérennisation des données hébergées à IDOC
  - Document comportant 60 exigences CNES et les réponses IDOC
  - Rédaction d'un « Data Management Plan »
- Réflexion sur la mise en œuvre de documents génériques, adaptables au contexte de la mission et aux exigences
  - Rédaction de documents génériques de méthodologie et de stratégie
  - Questionnaires pour préciser let a mise en oeuvre pour un projet
  - Rédaction de documents spécifique au projet / jeu de données
- Et l'outil ?

Les fonctions de la norme OAIS



SIP : Submission Information Package
AIP : Archival …
DIP : Dissemination …

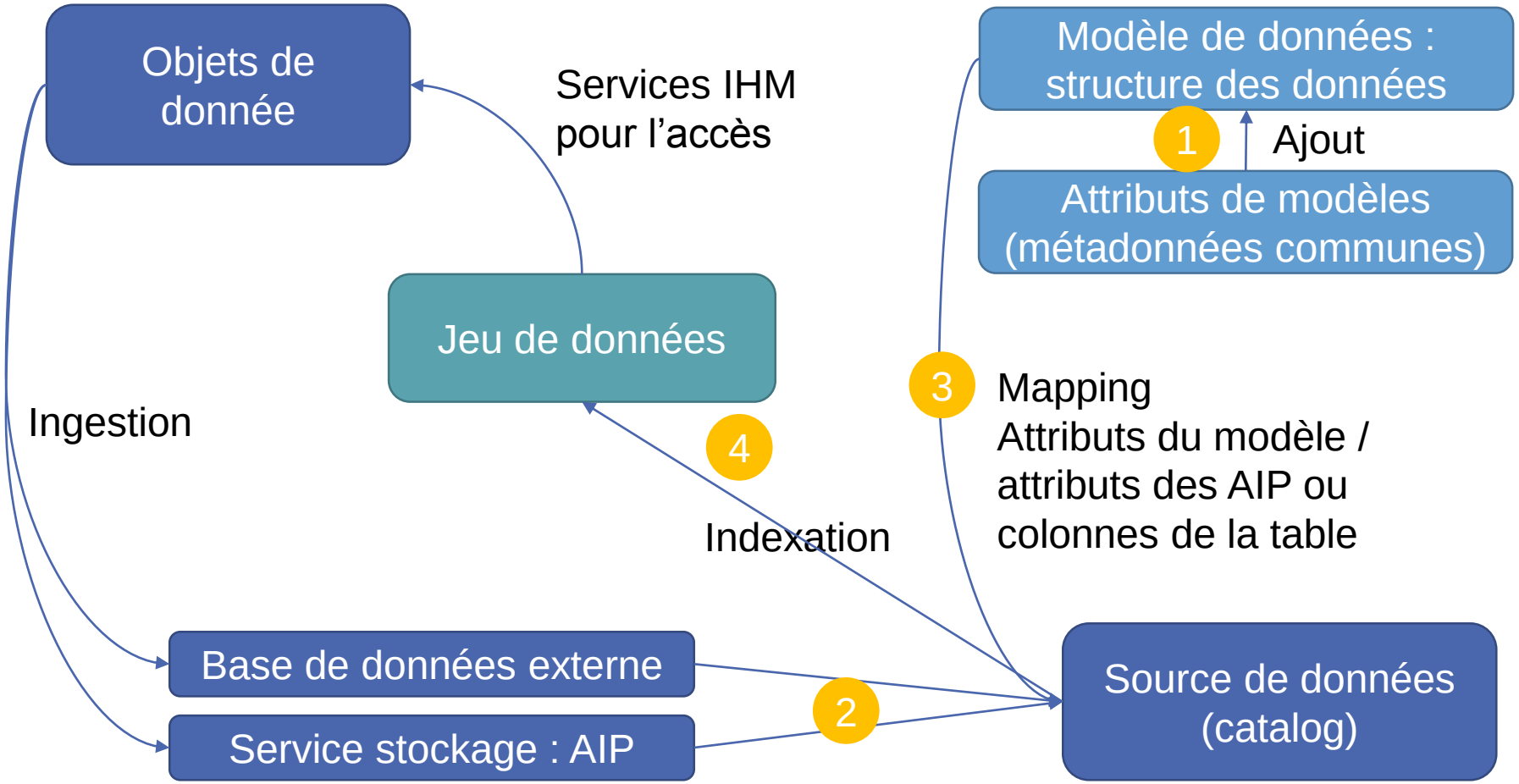
Implémentation OAIS : REGARDS

- Objectifs et apports pour le centre de données
  - Réflexion depuis 2014
  - participation aux spécifications techniques (pas forcément les mêmes critères/besoins au CNES et dans les laboratoires spatiaux)
- Prérequis IDOC :
  - maintenable, évolutif, adaptable
- Maîtrise d'ouvrage (CNES) / maitrise d'œuvre (CS-SI)
  - License GPLv3 Open Source
- Scalable (microservices, NoSQL,…)
- Responsive (nouveaux usages mobilité)
- Implémentation des 5 fonctions

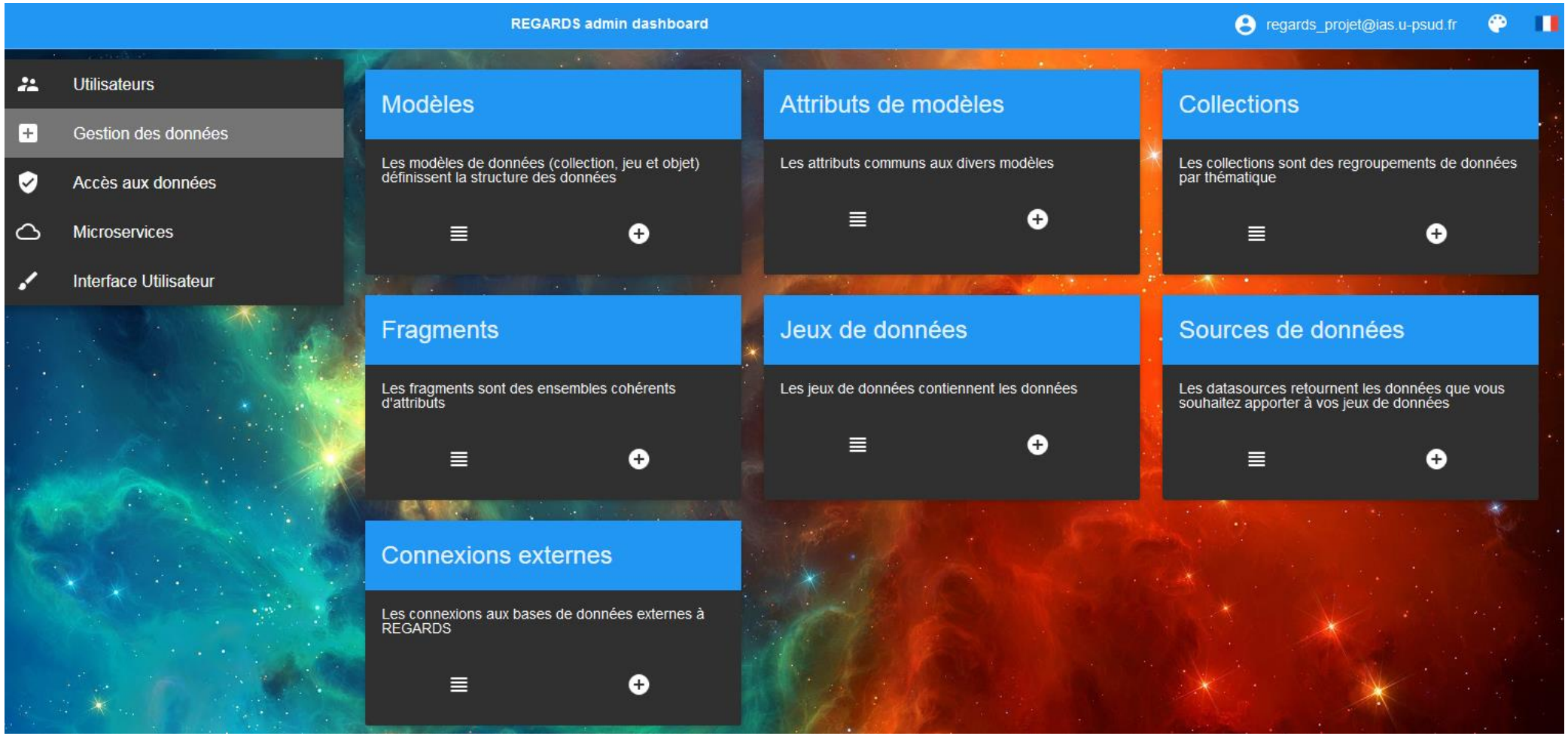
Etude de la fonction Accès de REGARDS

- Il faut s'approprier les concepts et le vocabulaire propres à REGARDS
- Exemples :
  - Pour la donnée d'observation, on parle de « objet de données » ou « data object » ou « produit »
    - En Astronomie : un fichier FITS
    - En Observation de la terre : un « data file » et son « leader file »
  - Pour un groupement d'objet de données on parle de « jeu de données » ou « dataset » ou « type de produit »
    - Exemple, les produits de niveau 2 de l'instrument Spire de la mission spatiale cosmologique Herschel

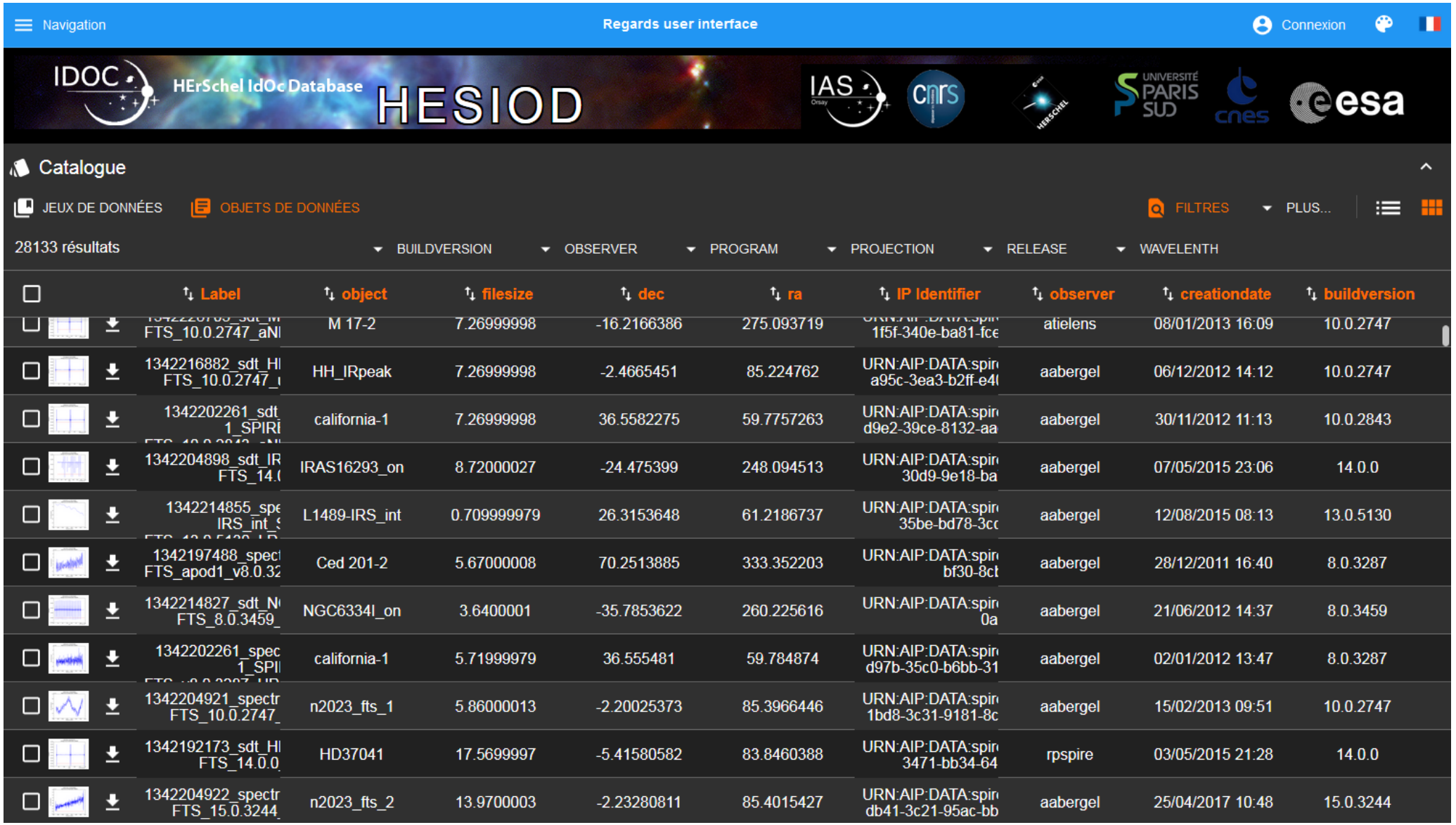
Scénario pour l'ajout d'un jeu de données



Interface administrateur : gestion des données

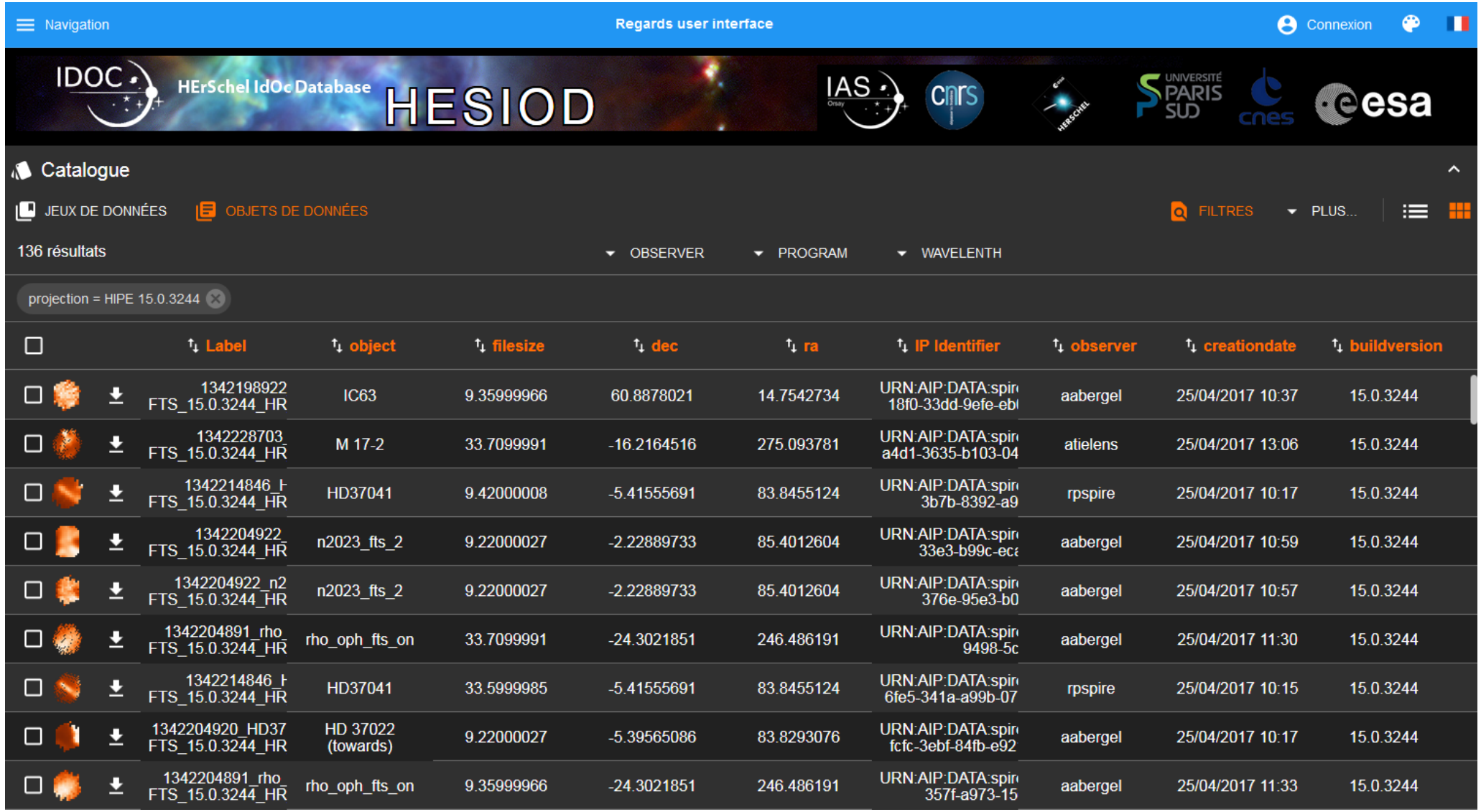


Interface utilisateur : multidataset



Label	object	filesize	dec	ra	IP Identifier	observer	creationdate	buildversion
FTS_10.0.2747_aN	M 17-2	7.26999998	-16.2166386	275.093719	1f5f-340e-ba81-fce	atielens	08/01/2013 16:09	10.0.2747
1342216882_sdt_HI FTS_10.0.2747_	HH_IRpeak	7.26999998	-2.4665451	85.224762	URN:AIP:DATA:spir a95c-3ea3-b2ff-e4	aabergel	06/12/2012 14:12	10.0.2747
1342202261_sdt 1_SPIRI	california-1	7.26999998	36.5582275	59.7757263	URN:AIP:DATA:spir d9e2-39ce-8132-aa	aabergel	30/11/2012 11:13	10.0.2843
1342204898_sdt_IR FTS_14.(	IRAS16293_on	8.72000027	-24.475399	248.094513	URN:AIP:DATA:spir 30d9-9e18-ba	aabergel	07/05/2015 23:06	14.0.0
1342214855_spe IRS_int_S	L1489-IRS_int	0.709999979	26.3153648	61.2186737	URN:AIP:DATA:spir 35be-bd78-3cc	aabergel	12/08/2015 08:13	13.0.5130
1342197488_spect FTS_apod1_v8.0.32	Ced 201-2	5.67000008	70.2513885	333.352203	URN:AIP:DATA:spir bf30-8ct	aabergel	28/12/2011 16:40	8.0.3287
1342214827_sdt_N FTS_8.0.3459_	NGC6334I_on	3.6400001	-35.7853622	260.225616	URN:AIP:DATA:spir 0a	aabergel	21/06/2012 14:37	8.0.3459
1342202261_spec 1_SPI	california-1	5.71999979	36.555481	59.784874	URN:AIP:DATA:spir d97b-35c0-b6bb-31	aabergel	02/01/2012 13:47	8.0.3287
1342204921_spectr FTS_10.0.2747_	n2023_fts_1	5.86000013	-2.20025373	85.3966446	URN:AIP:DATA:spir 1bd8-3c31-9181-8c	aabergel	15/02/2013 09:51	10.0.2747
1342192173_sdt_HI FTS_14.0.0_	HD37041	17.5699997	-5.41580582	83.8460388	URN:AIP:DATA:spir 3471-bb34-64	rpspire	03/05/2015 21:28	14.0.0
1342204922_spectr FTS_15.0.3244_	n2023_fts_2	13.9700003	-2.23280811	85.4015427	URN:AIP:DATA:spir db41-3c21-95ac-bb	aabergel	25/04/2017 10:48	15.0.3244

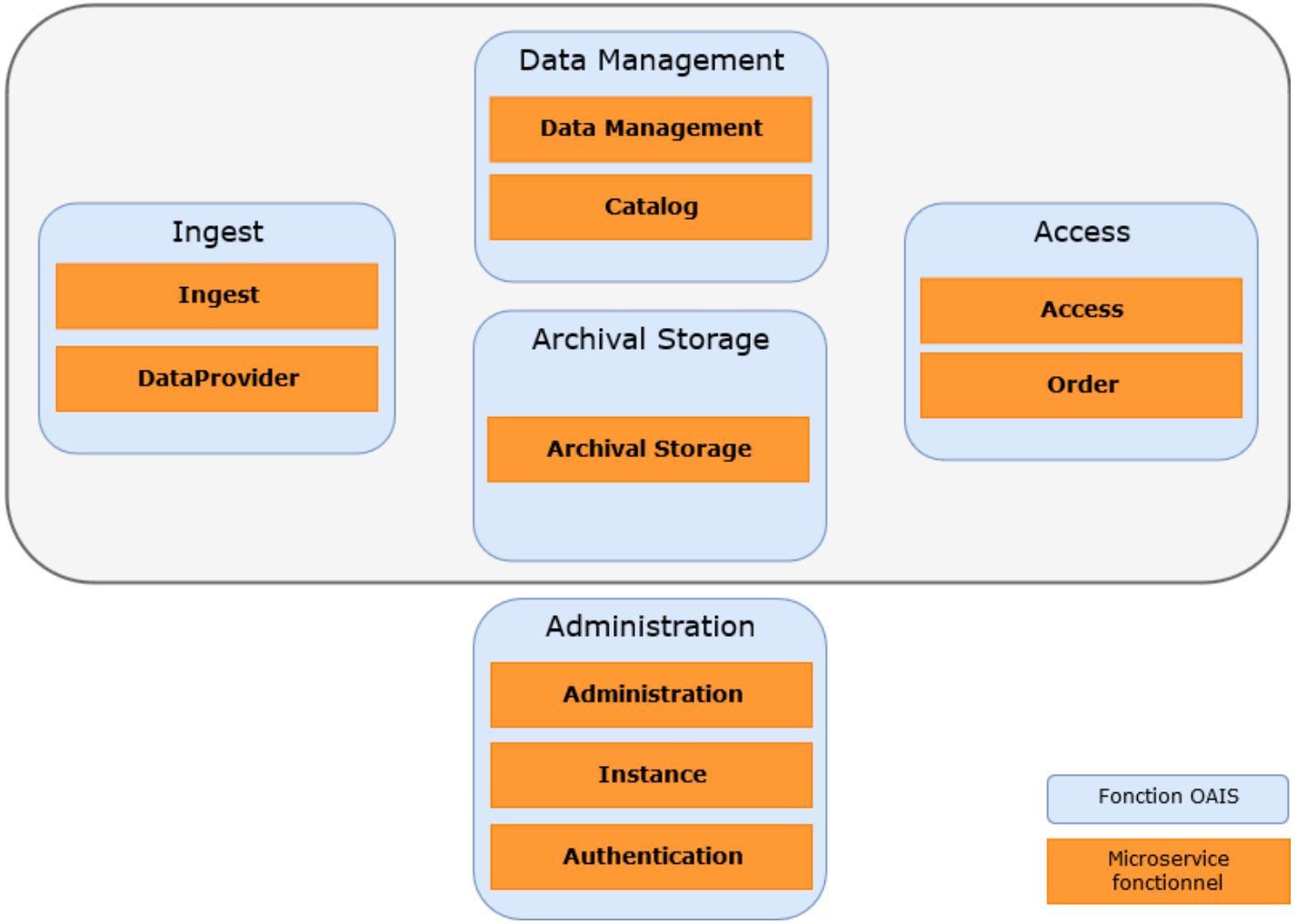
Interface utilisateur : filtres à facettes



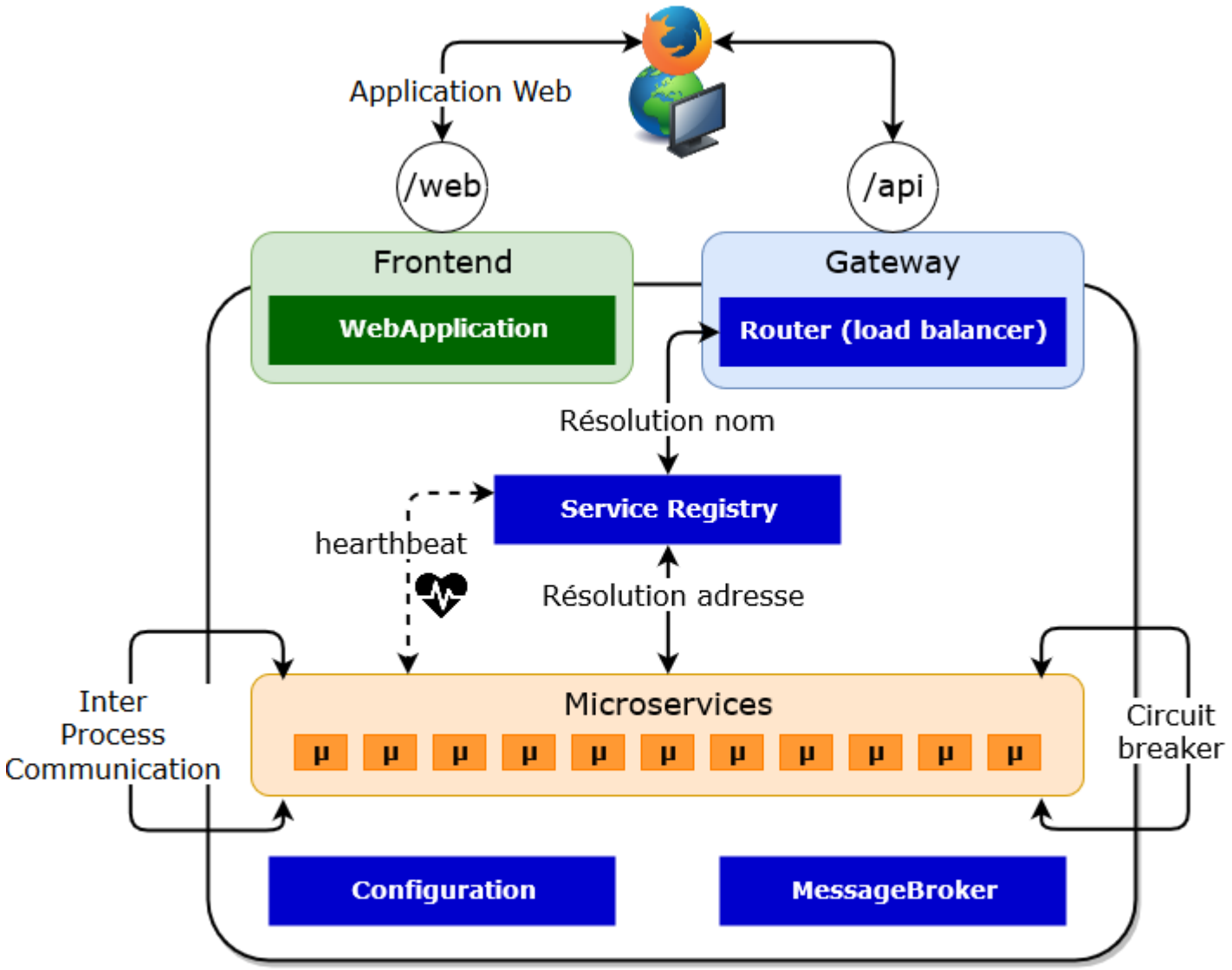
Comprendre l'architecture technique

- Pourquoi chercher à comprendre ?
  - Evaluer le potentiel à moyen terme
    - Complexité, l'évolutivité et la maintenabilité
  - Evaluer la facilité à bâtir
    - Ouvertures, points d'ancrage, extension
    - Courbe d'apprentissage et recrutements laboratoires
- S'approprier la solution
  - Construire une expertise opérationnelle
  - Dialoguer avec le CNES en connaissance de cause

Microservices fonctionnels
Correspondance avec OAIS



Architecture Cloud microservices



C’est quoi un microservice?

- Composant à responsabilité unique
  - Un domaine fonctionnel, une fonction spécifique, …
- Autonome
  - Contexte de fonctionnement isolé
    - Persistance, librairies
- Si possible de taille réduite
- Faible couplage
  - API REST (synchrones)
  - Bus d’évènement (asynchrones)
- Sans état (stateless)
  - Instances de µ-service interchangeables

Intérêt architecture microservice
côté développement

- Autonomie de développement
  - Architecture agnostique
    - Aucune adhérence technologique tant que HTTP possible
  - Développement faiblement couplé
    - Périmètre fonctionnel réduit et isolé
  - Cycles de livraison asynchrones
    - Réalisation, tests, déploiement
- Modestie vs Ambition
    - Modestie du périmètre des développements
    - Ambition du maintien de la cohérence

- La clef c'est l'API REST

Pourquoi il faut utiliser des API REST

- Standard
  - Les API REST web reposent sur des standards très répandus
    - HTTP, JSON, XML, RFC,
- Interopérables
  - Repose sur l'architecture d'internet
    - Serveur web, serveurs cache, protocoles
  - Prise en main rapide et peu couteuse
    - SOAP ?
  - Utilisable par tous
    - Tous les langages (Python, R ☺)
    - Scripts (curl, wget), simple navigateur web
  - Pas de problème réseau
- Navigable avec HATEOAS

GET /api/v1/rs-dam/models/attributes/1


```
{
 "id": 1,
 "name": "dec",
 "description": "dec",
 "defaultValue": null,
 "type": "DOUBLE",
 "jsonPath": "properties.dec",
 "identifiable": true,
...
 "_links": {
    "self": {
        "href": "https://idoc-regards.ias.u-psud.fr:80/api/v1/rs-dam/models/attributes/1"
    },
    "update": {
        "href": "http://idoc-regards.ias.u-psud.fr:80/api/v1/rs-dam/models/attributes/1"
    },
    "list": {
        "href": "http://idoc-regards.ias.u-psud.fr:80/api/v1/rs-dam/models/attributes"
    }
 }
}
```


GET

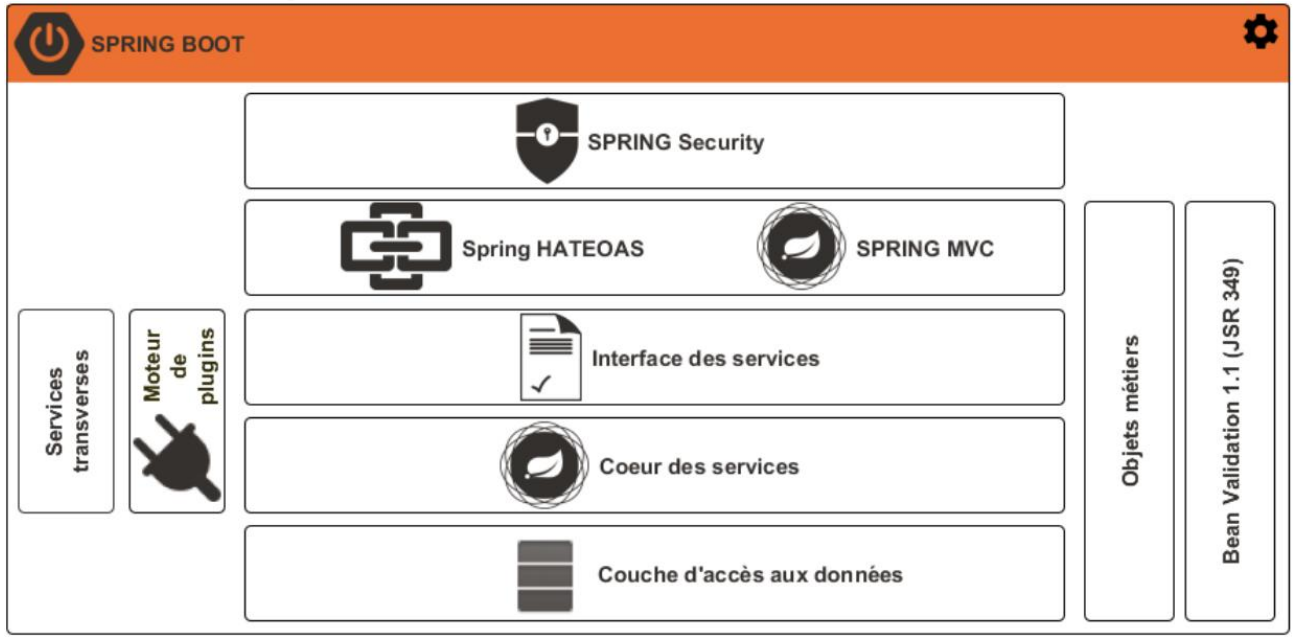
PUT

REGARDS
côté développement

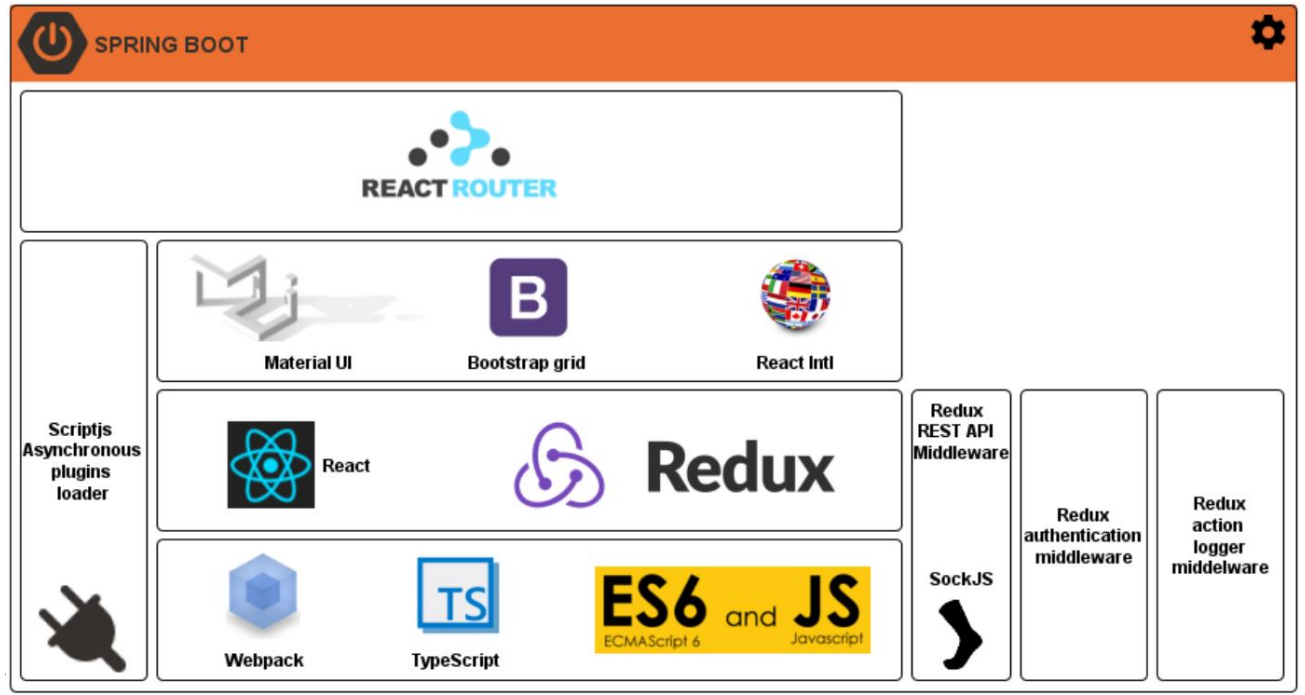
- Développement avec des gages de sérieux
  - Utilisation de frameworks établis
    - Ecosystème Spring & Spring Cloud Netflix
    - Maven
    - Ecosystème React, typeScripts, webpack
  - Modularité, Tests unitaires, Intégration continue, …
- Documentation
  - Faiblesse de la V1,
  - Gros efforts en cours avec la V2
  - http://regardsoss.github.io/
    - API REST
    - Documentation frontend complète
    - Jobs, catalog

Compositions

Côté backend



Côté frontend



Elasticsearch

- En cours d'exploration
- Moteur d'indexation et de recherche distribué RESTful
  - Indexation des métadonnées
  - Analyse et recherche
- Utilisation des facettes
  - Agrégation de données pré-calculés pour faciliter l'exploration des données
- Interrogeable par Api REST (encore)
  - Utilisation déportée possible outre REGARDS

Possibilités d'adaptation

- Ne pas modifier le cœur
  - Mécanisme d'extensions prévus
    - Java : archetypes maven Microservice et Module
    - Frontend : cf. documentation V2 complète https://regardsoss.github.io/frontend/
  - Ajout de Plugin
    - Interfaces Java (points d'extension)
      - Authentification, calcul d'attributs, stockage, lecture SIP, génération SIP
  - Extension microservice
    - Ajout d'un module
    - Création d'un nouveau microservice
- Système REST
  - Possibilité de mettre sa propre application en frontal

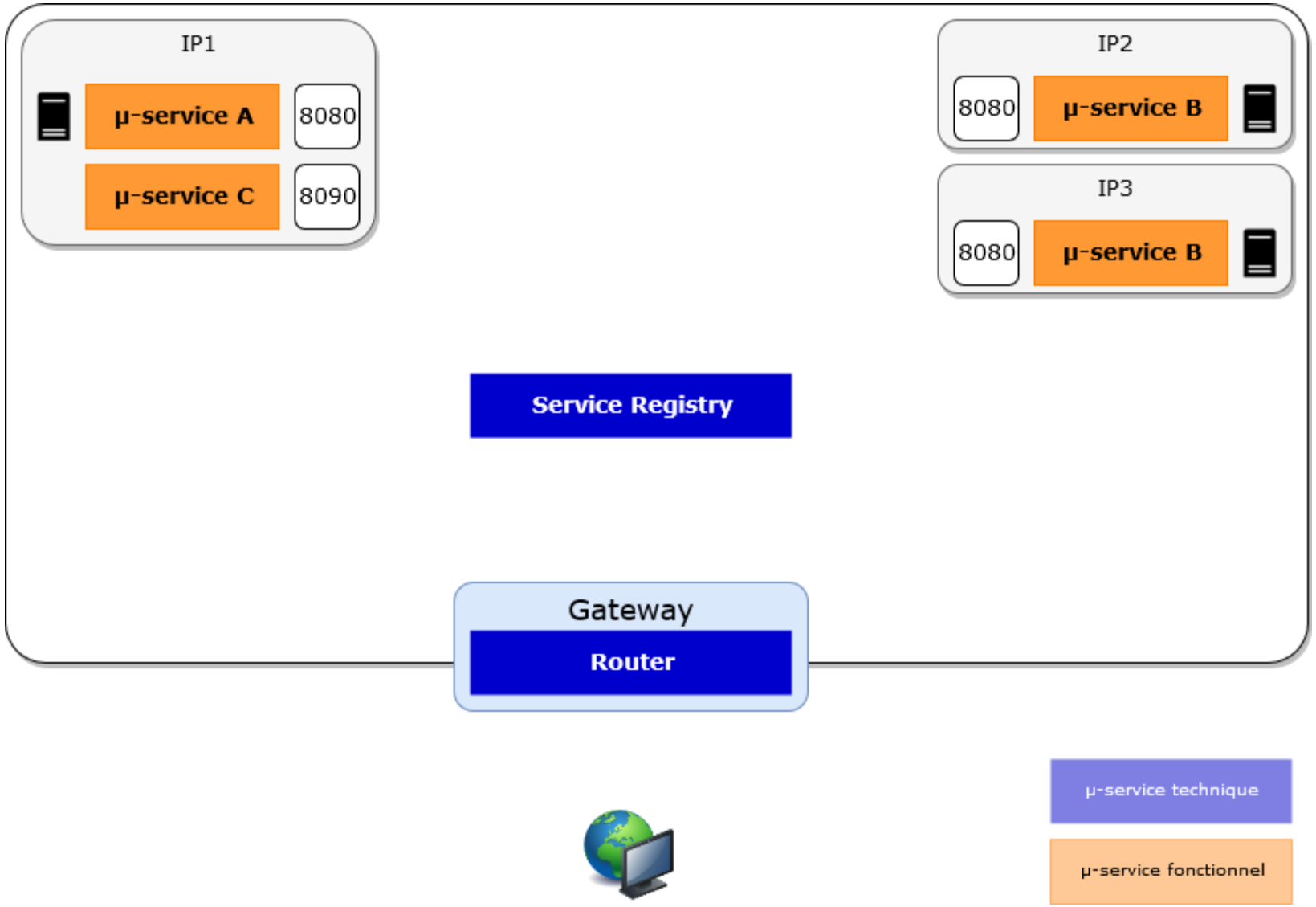
Intérêt architecture microservice
côté exploitation

- Scalability et load balancing
  - Déploiement de plusieurs instances
  - Résistance à la montée en charge
  - Robustesse et tolérance aux pannes
- Déploiements plus modulaires
  - Sans arrêt de service, à la demande ou dynamiquement
- Au prix d'une architecture un peu intimidante…

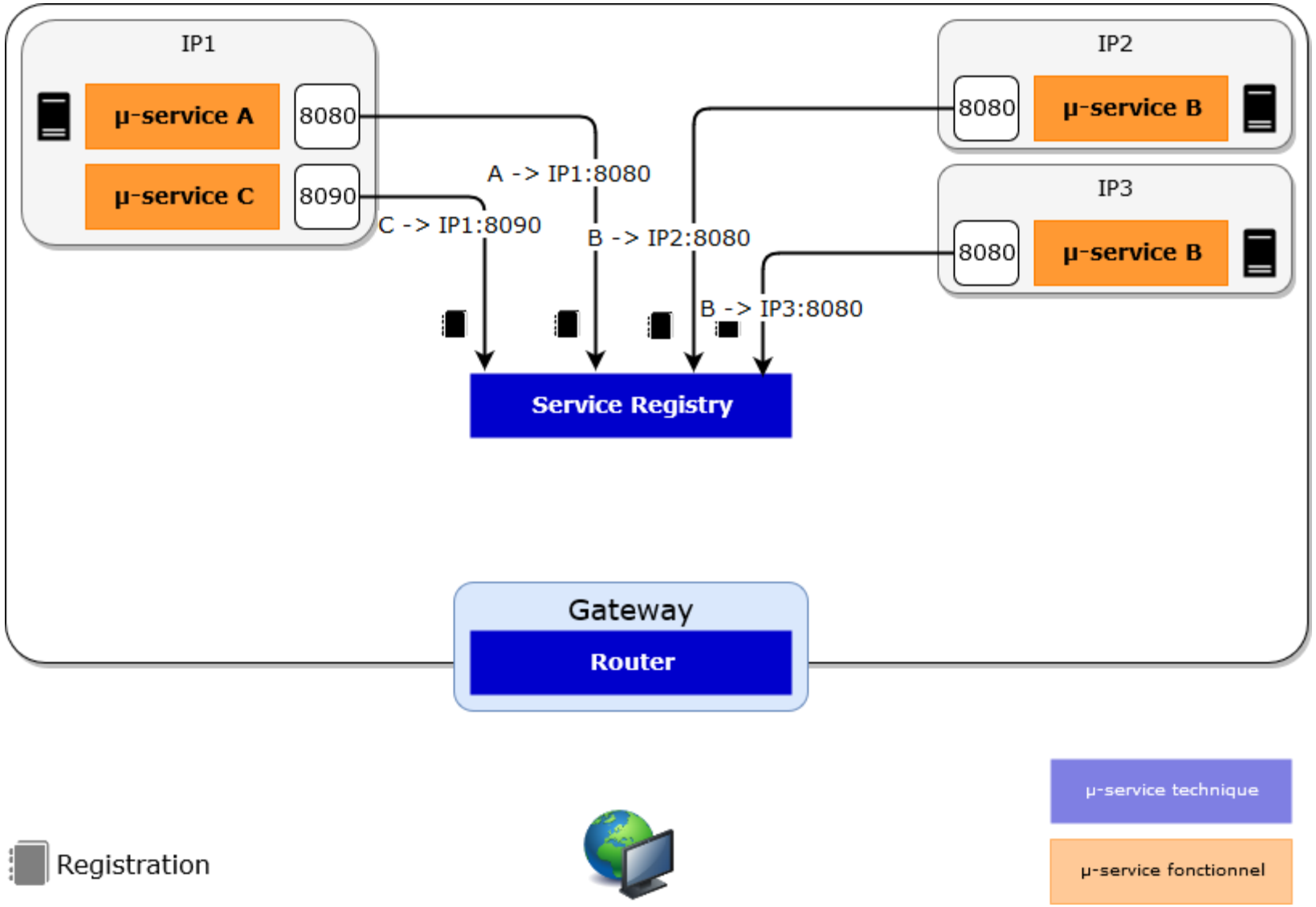
Patterns d'architectures microservices

- Service Registry et API gateway
  - Découvrir les microservices disponibles
  - Uniformiser et faciliter l'accès
- Heartbeat et REST client load balancer
  - Maintenir une cartographie du système
  - Laisser le microservice client sélectionner une instance
- Circuit breaker
  - Court-circuiter les défaillances

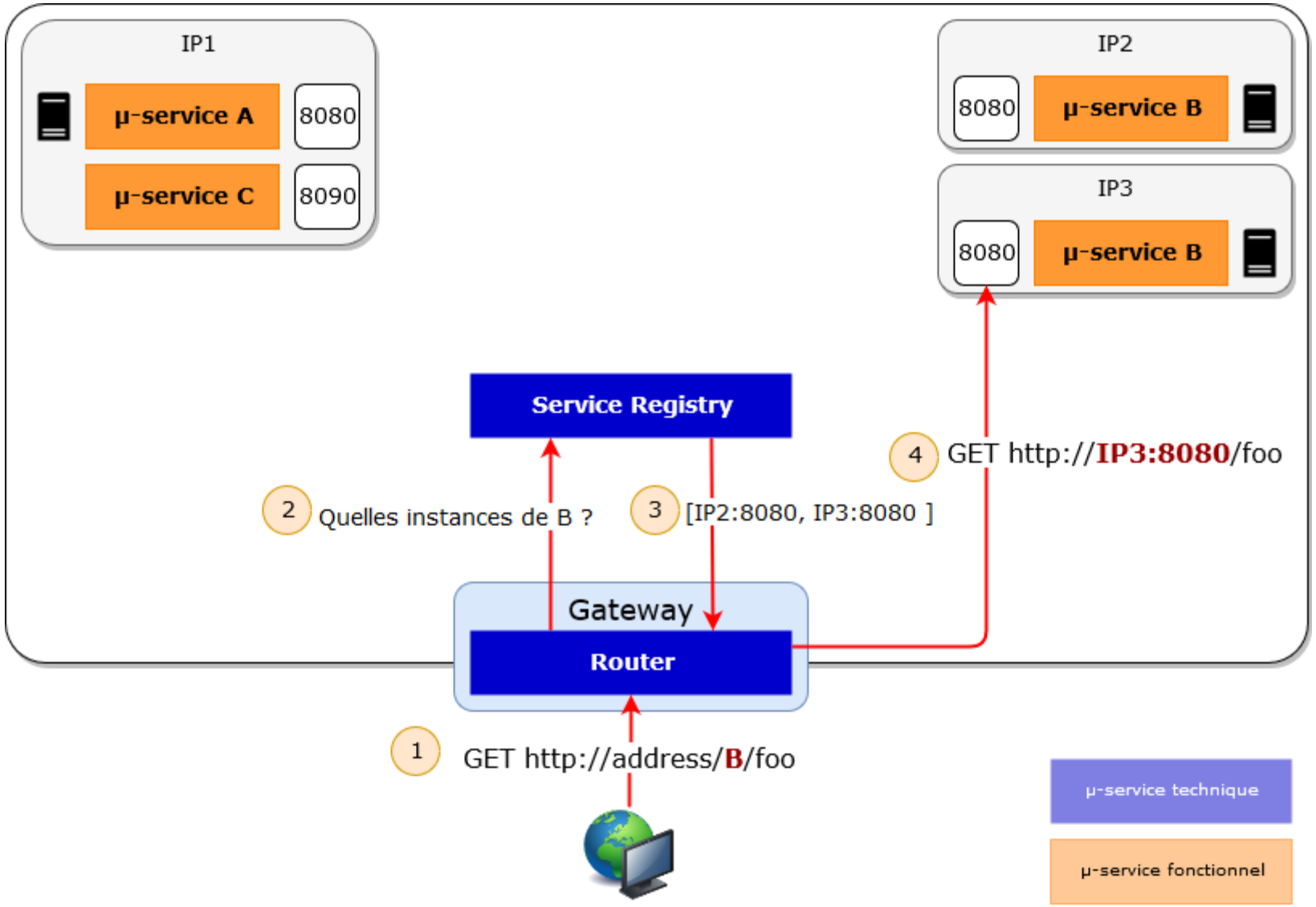
Registry


IP1
µ-service A
8080
µ-service C
8090
IP2
8080
µ-service B
IP3
8080
µ-service B
Service Registry
Gateway
Router
µ-service technique
µ-service fonctionnel

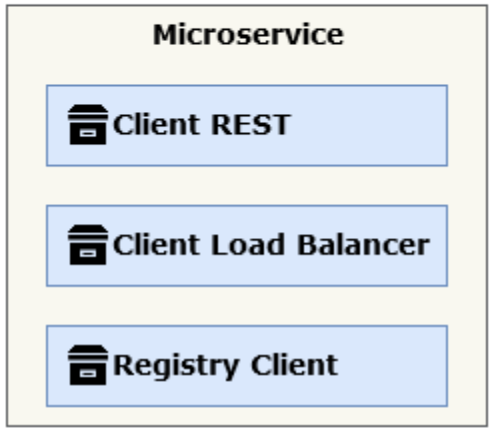
Registry – self-registry

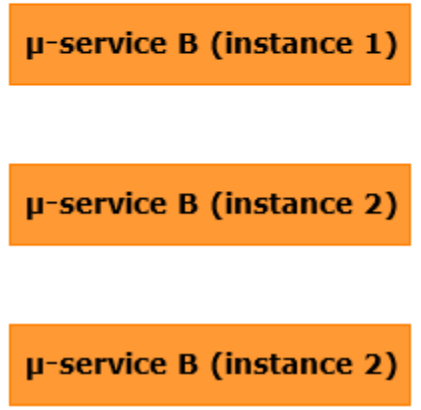


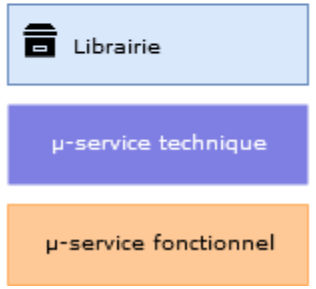
API gateway


IP1
µ-service A
8080
µ-service C
8090
IP2
8080
µ-service B
IP3
8080
µ-service B
Service Registry
2 Quelles instances de B ?
3 [IP2:8080, IP3:8080]
4 GET http://IP3:8080/foo
Gateway
Router
1 GET http://address/B/foo
µ-service technique
µ-service fonctionnel

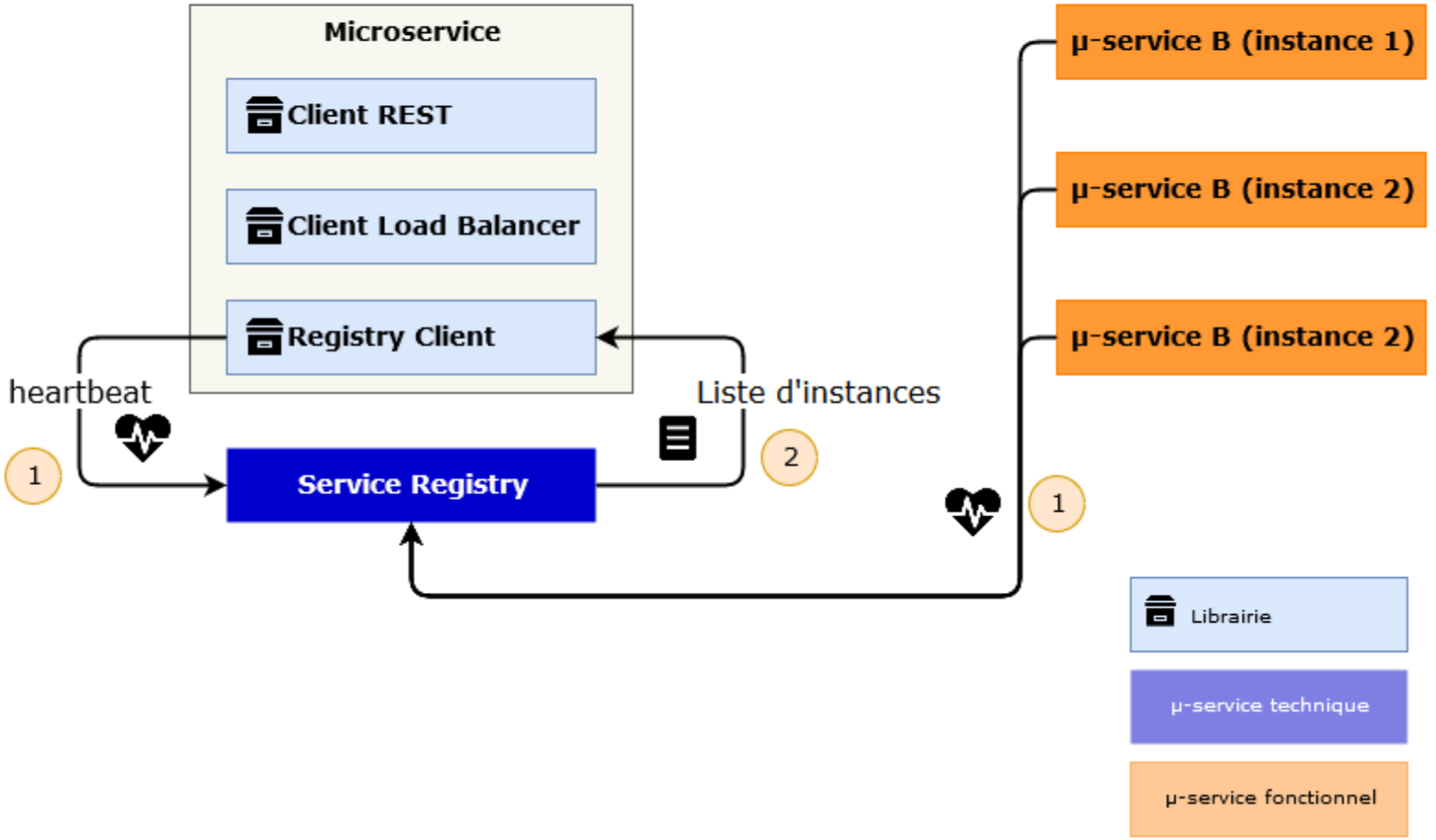
Heartbeat client – composition d'un microservice



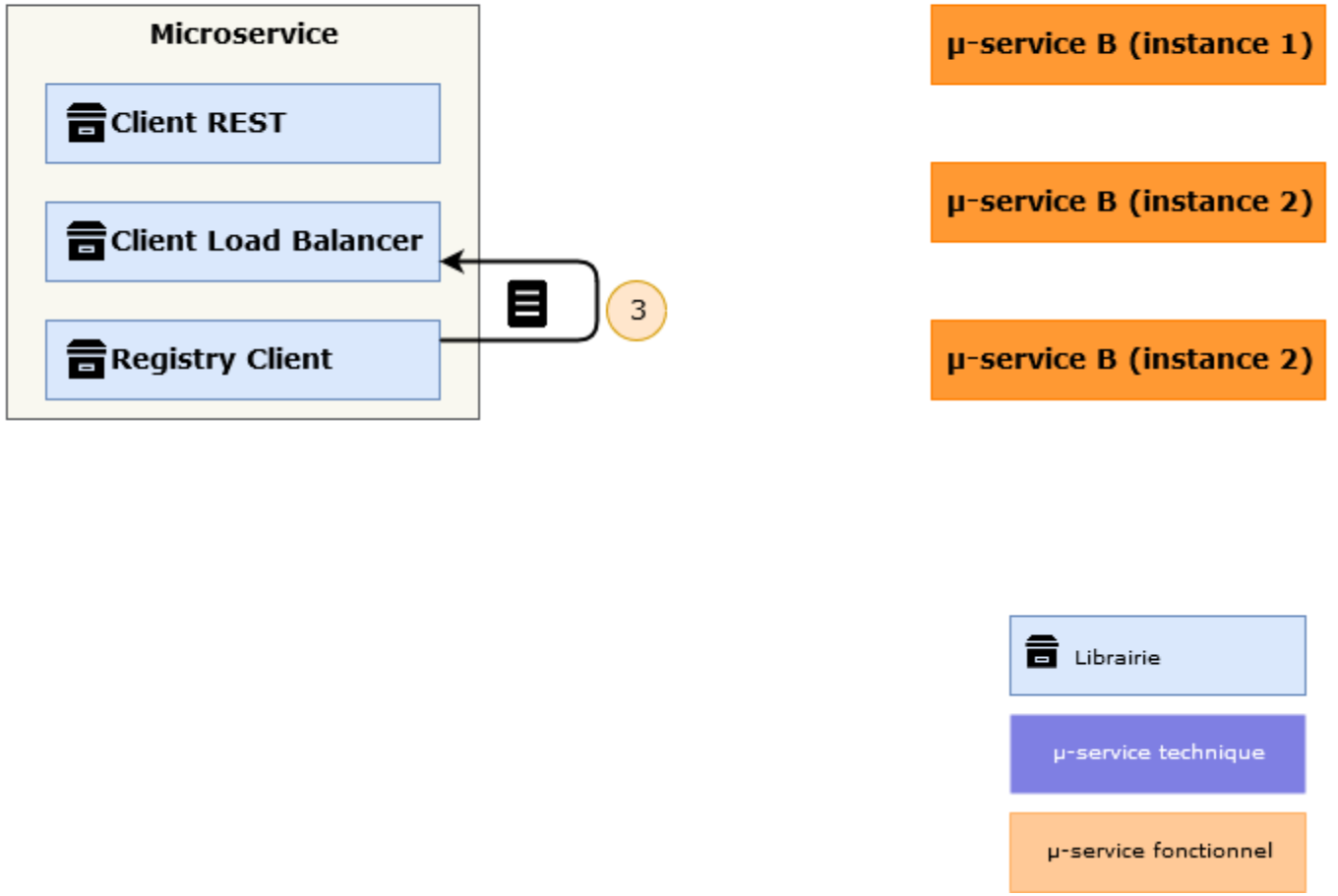




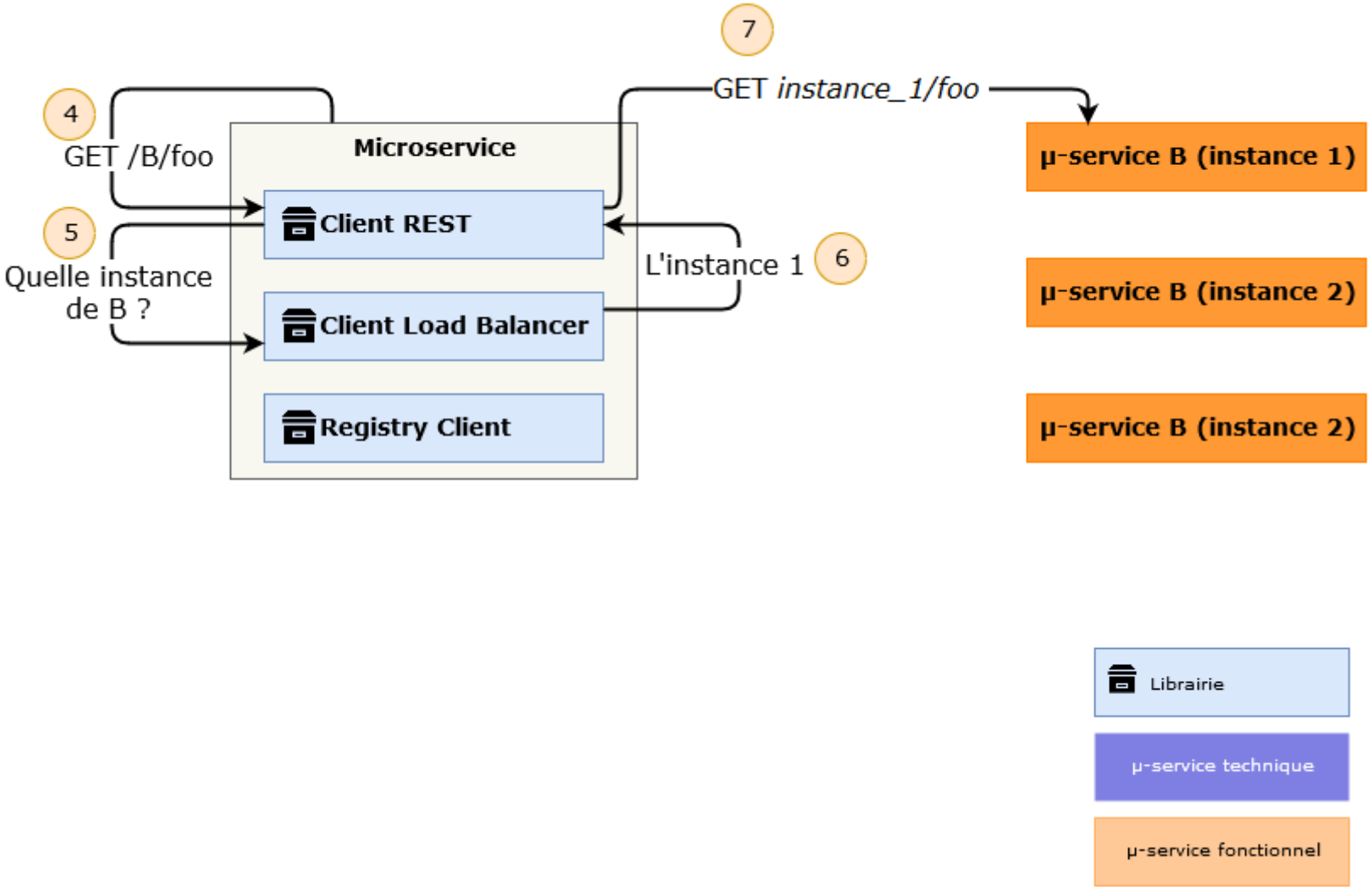
Heartbeat client – renouvellement



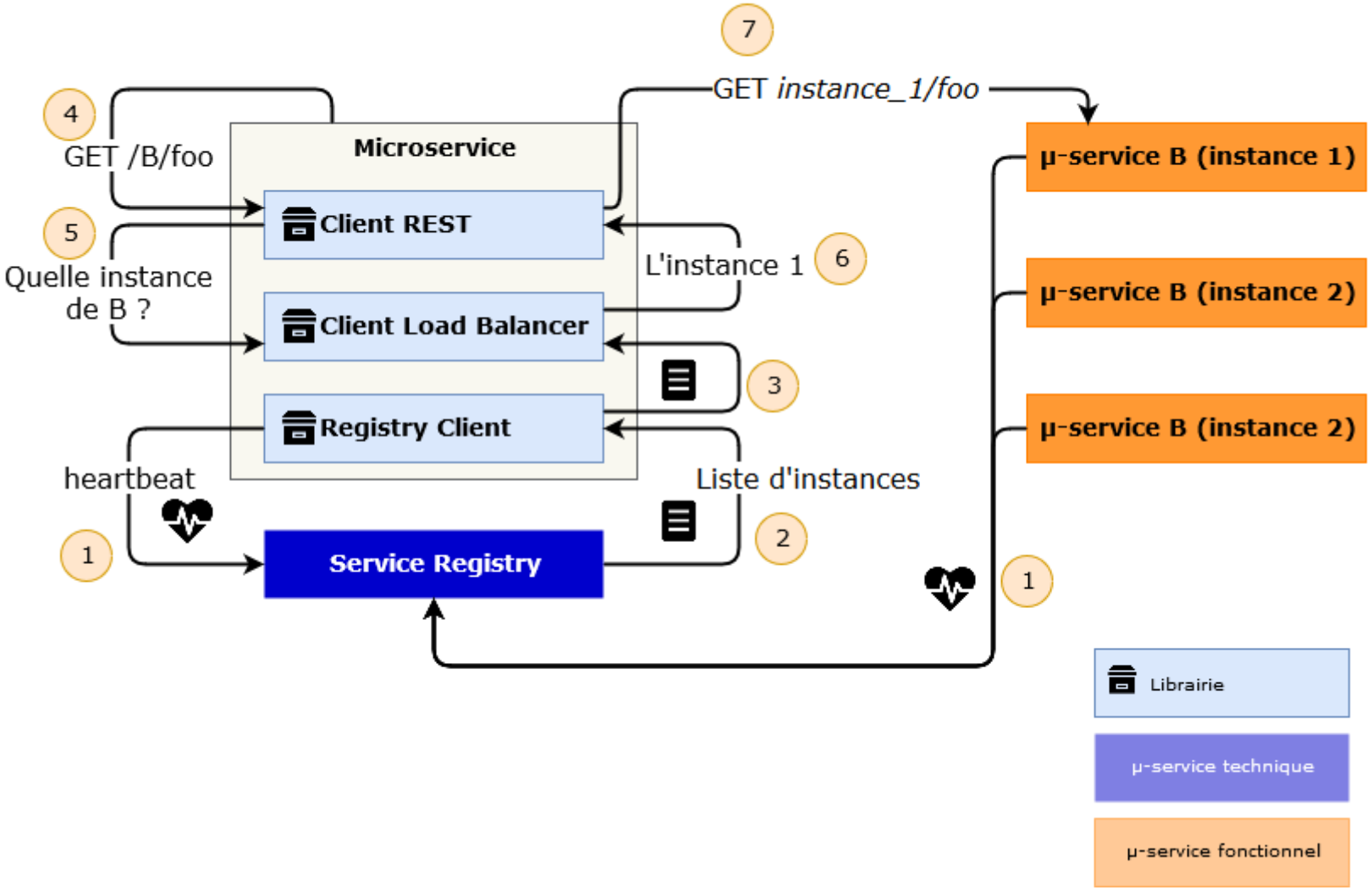
Heartbeat client – Mise en cache (client load balancer)



Heartbeat Client – Invocation avec Client Load Balancer

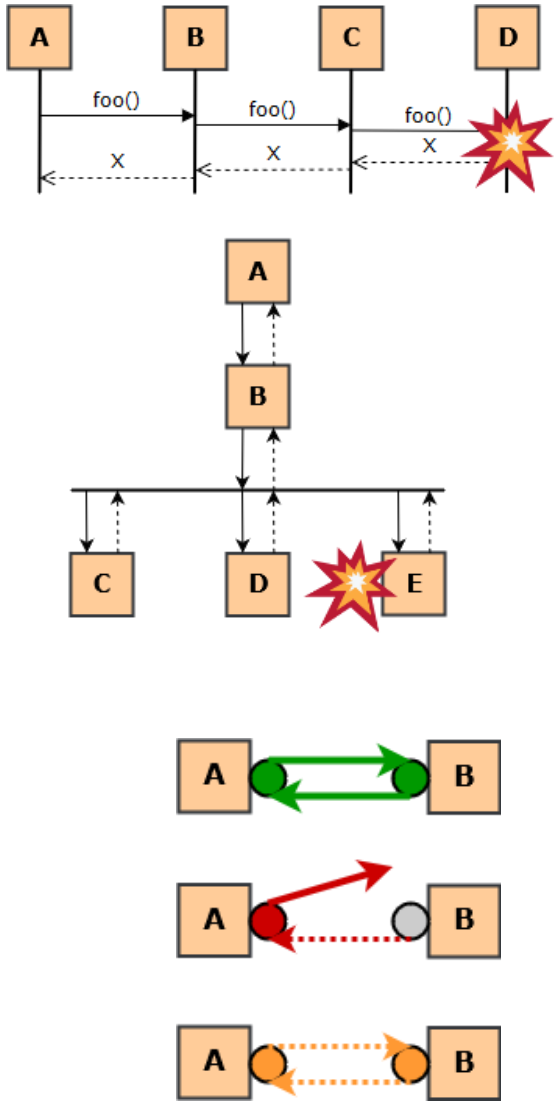


Heartbeat Client

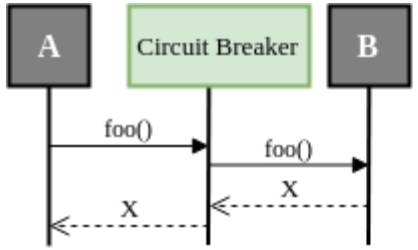


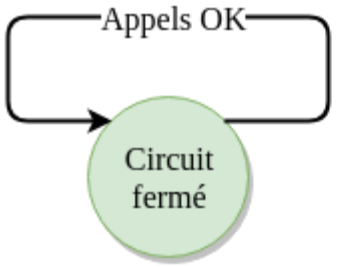
Circuit Breaker

- Défaillance des dépendances
  - Timeout, taux d'erreurs, mauvaises performances
- Avalanches
  - Lourds traitements en erreur *à la fin*
  - Cascade sur enchainement d'appels
- Circuit Breaker
  - Dépendance comme un circuit
    - Circuit fermé, ouvert, semi-ouvert
  - Fail Fast
    - Solution de contournement *Gracefull degradation*
  - Auto stabilisation
    - Redémarrage de service (solution tierce)

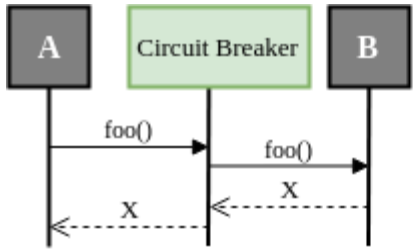


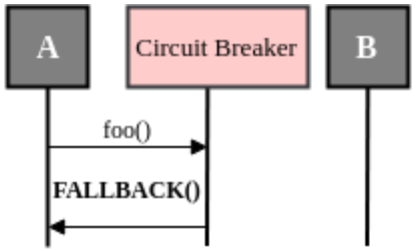
Circuit Breaker – Cas nominal

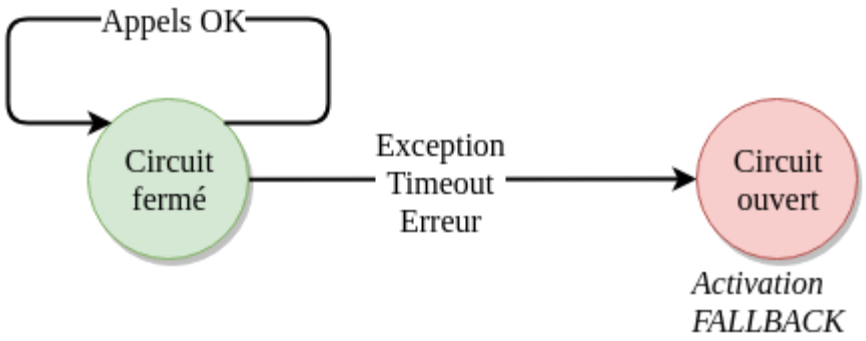




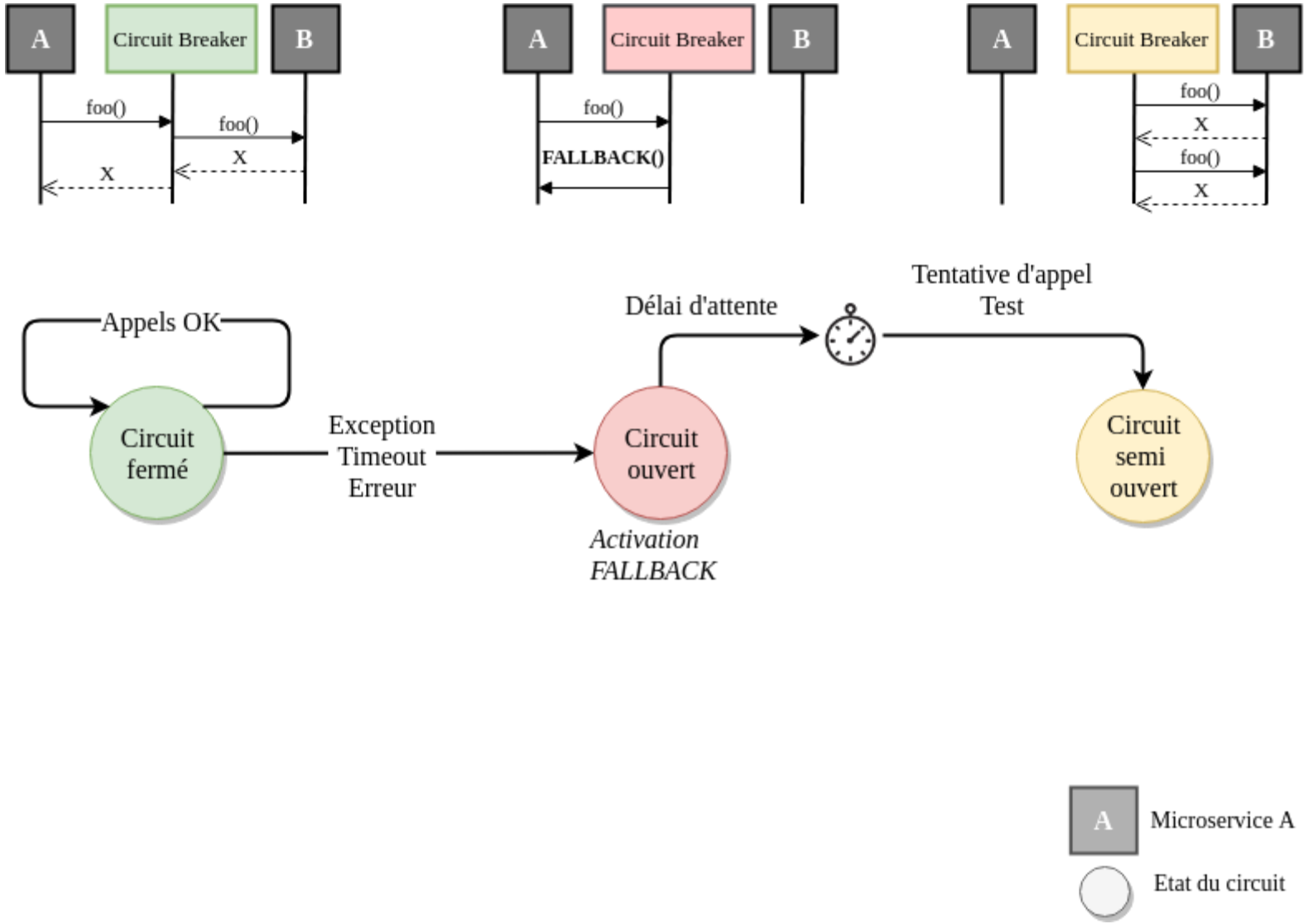
Circuit Breaker – Problèmes constatés



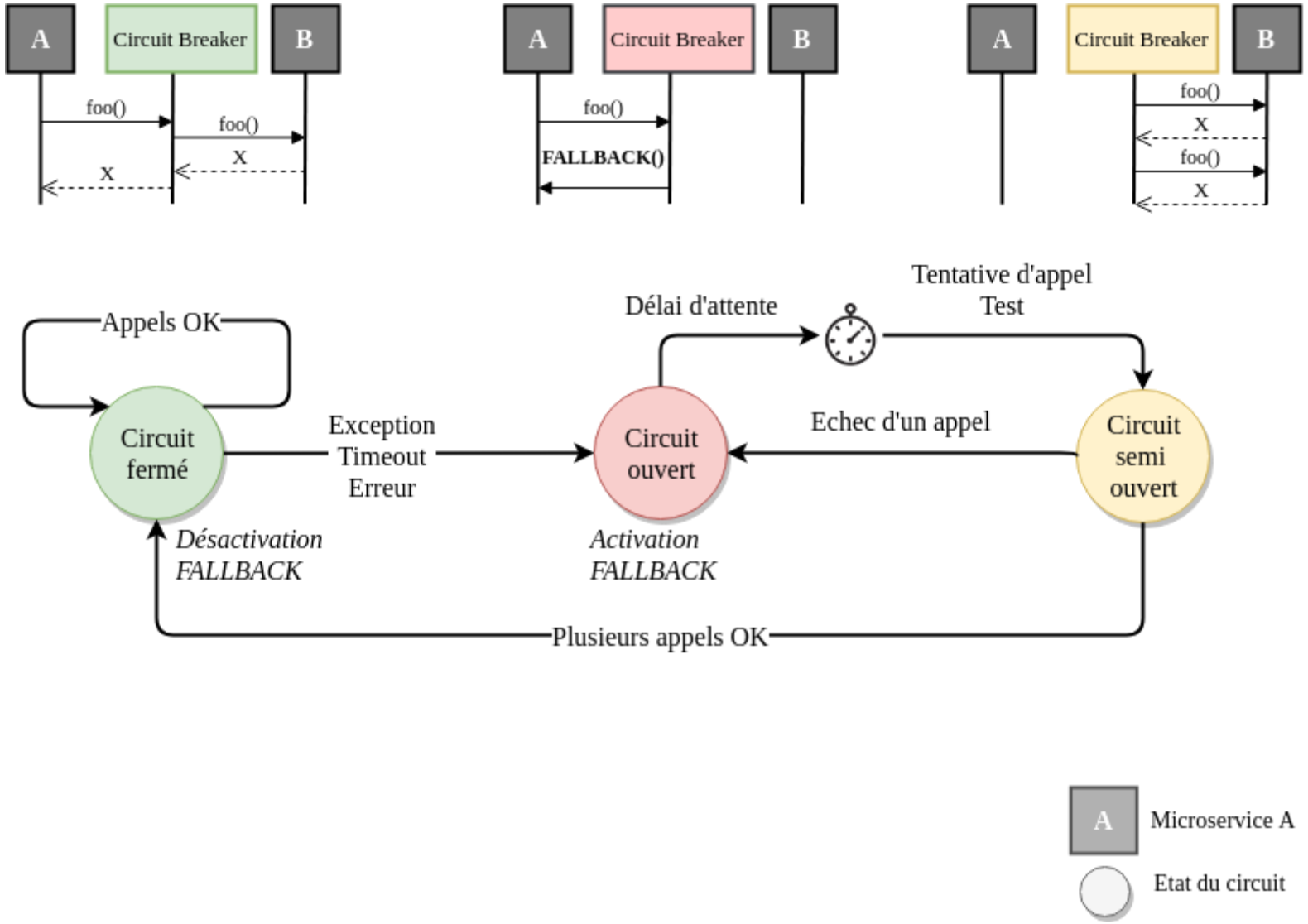




Circuit Breaker – Attendre et évaluer



Circuit Breaker – Rétablissement



Et demain ?

- Roadmap
  - V2, V2.1 anomalies V2 (sept.), V3 (oct.), V4 (début 2019)
- Prochainement en V2
  - Ajout de 4 microservices liés à la gestion des SIP et des AIP
    - **dataprovider** : Acquisition de fichiers par création des SIP associés aux fichiers à acquérir.
    - **order** : Commandes de fichiers
    - **ingest** : Création des AIPs depuis les SIP soumis
    - **storage** : Stockage des AIPs.
  - Prochains tests sur notre plateforme expérimentale
    - Manipulation SIP/ AIP et des métadonnées associées

Conditions du succès

- Investissement continu sur l'outil
  - Le CNES ne peut plus se permettre d'abandonner
- Gouvernance du projet
  - Quel modèle de gouvernance ?
  - Quelle écoute et quelle place pour les labos ?
  - Comment contribuer ?
- Couverture des besoins / mise en œuvre
  - Capacité à couvrir des besoins spécifiques
- Capitalisation sur la technologie
  - Ambitieuse mais accessible
  - S'approprier la solution

Conclusion

- Bon potentiel
  - Architecture convaincante
    - haute-disponibilité & aux grosses volumétries
    - Applicable à tous types de données (décrites par des métadonnées)
  - Développement technique de qualité
- Très bonnes relations CNES et prestataire
  - Merci aux développeurs et chefs de projet pour leur réactivité et leur patience ;)
- Et pourquoi pas :
  - créer une communauté d'utilisateurs / de développeurs...

Best regards !

Questions & Liens

- Merci
  - Si vous avez des questions…

- Liens
  - Documentation V2 (API Rest, Frontend)
    - http://regardsoss.github.io/
  - Sources :
    - https://github.com/RegardsOss
  - REGARDS-Community
    - (tests, retours d'expérience et documentation interne)
    - https://idoc-projets.ias.u-psud.fr/redmine/projects/regards-community
  - Spring-cloud:
    - https://cloud.spring.io/spring-cloud-netflix/single/spring-cloud-netflix.html
  - Understanding eureka client server communication
    - https://github.com/Netflix/eureka/wiki/Understanding-eureka-client-server-communication

Annexes

API REST

- Parlez-vous REpresentational State Transfer?
  - Orienté ressource et représentation
    - Ressource : une donnée, un laboratoire, une collections de chat
    - Représentation: un contenu XML, JSON, texte
  - Actions sur des ressources
    - Pas invocation de fonction distante (Remote Procedure Call)
- Quelques concepts clefs
  - Aucune hypothèse sur le client ou le serveur .
  - Aucune gestion état côté serveur. Responsabilité du client.
  - « À interface uniforme »
    - Ressource identifiable
    - Représentation complète et autosuffisante
    - Hypermédia comme moteur d'état de l'application (HATEOAS)

HATEOAS

- **H**ypermedia **A**s the **E**ngine of **A**pplication **S**tate
  - Utiliser les hyperliens comme moyen de découvrir l'API
    - Comme un humain parcourant un site
  - Autosuffisance
    - Le premier document suffit à dérouler
  - Intelligence
    - Pas de codage en dur
  - Standardisation en cours – HAL
    - Hypertext Application Language
    - http://stateless.co/hal_specification.html
- Dans chaque réponse
  - Fournir les étapes suivantes

API Gateway et self-registry

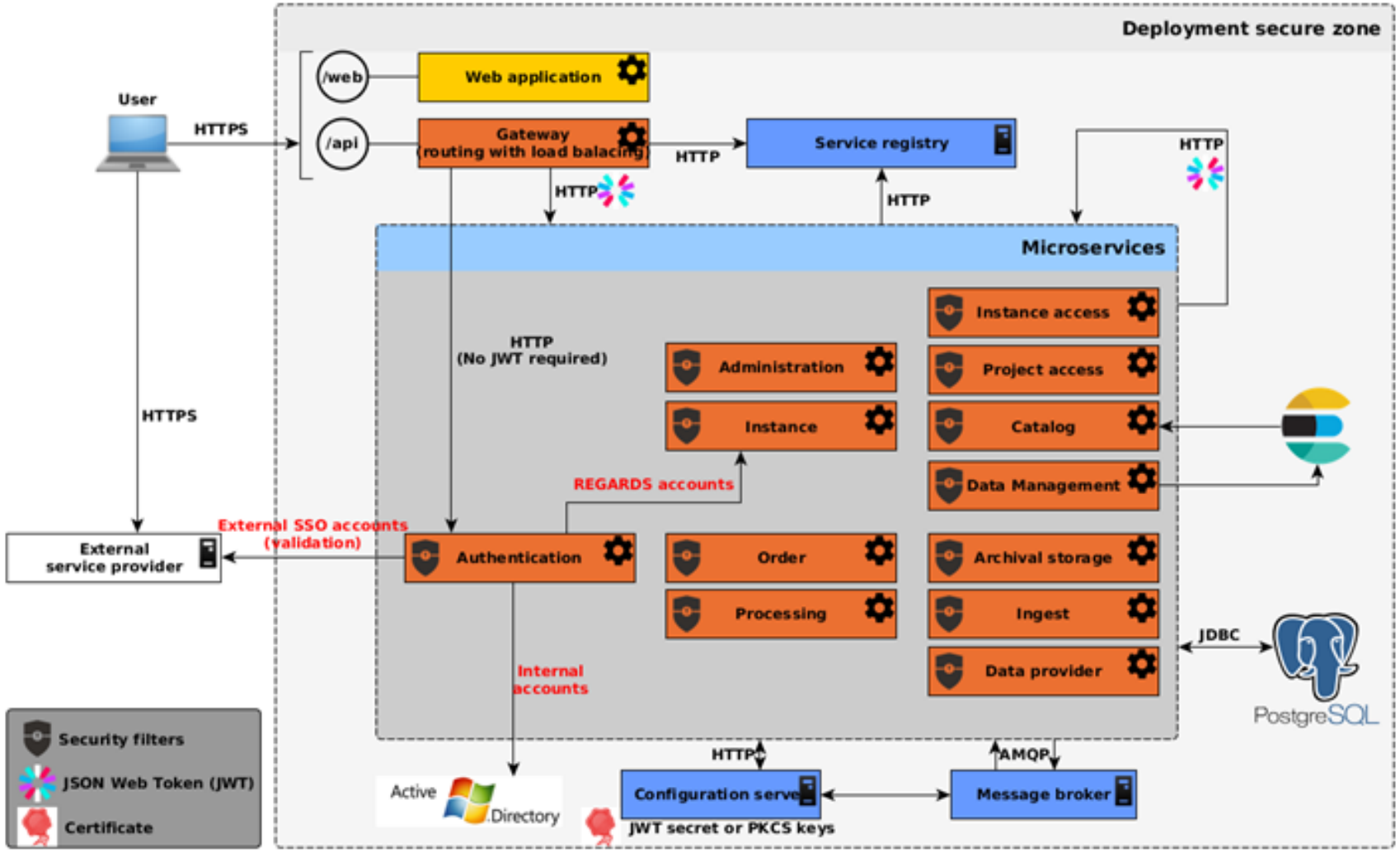
- Faciliter les accès aux microservices
  - Cacher la multiplicité des instances
  - Accéder par nom logique (*catalog*, *admin*)
    - Et non par adresse:port
- Connaître les instances déployées
  - Informations
    - Nom µ-service, adresse, port, état, machine virtuel
- Patterns
  - API Gateway
    - Translation nom logique vers adresse:port
  - Registry
    - Les clients s'enregistrent auprès d'un service

Heartbeat

- Maintenir une cartographie du système
  - Etat actualisé des instances de µ-service enregistrés

- Pattern Heartbeat
  - Heartbeat client :
    - Le client *pousse* l'information
  - Obtention d'un bail (lease)
    - À renouveller toutes les 30 s, durée de vie 90 s maximum
  - Le service retourne la cartographie des instances
    - Status UP, DOWN, STARTING, OUT_OF_SERVICE
    - Adresse IP , port, healthURL, …

Schéma d'architecture complet



REGARDS : déploiement CNES

- Migration des SIPAD-NG (Oracle)
  - Une dizaine de catalogues, 8000+ jeux de données, 20 millions d'objets, 250+ To de données
- Nouveaux projets
  - Microscope (vérification du principe d'équivalence)
    - Archivage des données
    - Mise à disposition
  - SVOT (altimétrie)
- Utilisation prévue à l'ONERA

Interface administrateur : gestion de l'interface cliente