Table des matières

[Lexique 2](#_Toc27694608)

[API vs \*\*Service (macro, mini, micro, nano…) 2](#_Toc27694609)

[Types d’API 3](#_Toc27694610)

[Domaines 7](#_Toc27694611)

[Unité d’affaire 7](#_Toc27694612)

[Domaine d’affaires 7](#_Toc27694613)

[Capacité d’affaire 7](#_Toc27694614)

[Secteur d’affaire 7](#_Toc27694615)

[Contexte d’affaire 8](#_Toc27694616)

[Domaine d’intégration 8](#_Toc27694617)

[Domaine applicatif 8](#_Toc27694618)

[Lignes directrices URL 9](#_Toc27694619)

[Exemple concret : identité d’un client (à partir de différent contexte – *binding context*) 10](#_Toc27694620)

[Zones de sécurité physiques 13](#_Toc27694621)

[Zones de sécurité logiques 14](#_Toc27694622)

[Zones et domaines d’intégration 15](#_Toc27694623)

[Définitions 16](#_Toc27694624)

[Tenants (réseau) 17](#_Toc27694625)

[Namespace (Openshift) 18](#_Toc27694626)

[Patrons d’intégration concrets - Openshift 19](#_Toc27694627)

[API Gateway 23](#_Toc27694628)

# Lexique

## DDD (*Domain Driven Desing*)

**En cours…**

### Unité d’affaire (*business unit*)

Organisation qui possède et exploite un ou plusieurs domaines d’affaires stratégiques:

Exemple d’unité d’affaires de LQ :

* Présidence des opérations loteries (POL)
* Société des établissements de jeux du Québec (SEJQ)
* Société des casinos du Québec (SCQ)
* Technologies Nter (Nter)
* Loto-Québec (LQ)

### Capacité d’affaire (*business capability*)

Ensemble d’activités(ou fonctions) d’affaires (*qu’est-ce que l’entrerprise fait, le quoi*) organisées dans le cadre d’un processus d’affaires permettant d’atteindre les objectifs fixés par une unité d’affaires dans l’exploitation de son ou ses domaines d’affaires.

Une capacité d’affaires peut être de deux types :

* **stratégique** : à compléter
* **métier** : activités d’affaires dont la spécificité est exclusive aux domaines d’affaires de LQ. (ex. JHA, casinos du QC, ALV, etc.)
* **soutien** : activités d’affaires qui soutiennent les activités d’affaires métier. (ex. finances, RH, approvisionnement, services juridiques, TI, communications, etc.)

Les capacités d’affaires peuvent avoir plusieurs niveaux de granularité selon le nombre ou le degré de complexité des activités d’affaires.

### Domaine d’affaires (*business domain*)

Regroupement maître et global de différents processus, entités et secteurs d’affaires de l’entreprise. La différenciation des domaines d’affaires de l’organisation vient des différences dans les processus, entités, secteurs et intervenants d’affaires de chaque domaine de l’organisation.

Exemple de domaine d’affaires de LQ :

* Appareils de loteries vidéo (ALV)
* Casinos terrestres
* Salons de jeux
* Jeux de hasard et d’argent en salles (Bingo, Kinzo, etc.)
* Jeux de hasard et d’argent en ligne (Espacejeux)
* Opérations loteries
* Loto-Québec (domaine transversal par rapport aux autres domaines)

Référence : <http://www.thesaurus.gouv.qc.ca/tag/terme.do?id=MDL448>

### Secteur d’affaire (*business sector*)

Représente normalement un regroupement de capacités d’affaires primaires d’un domaine d’affaires.

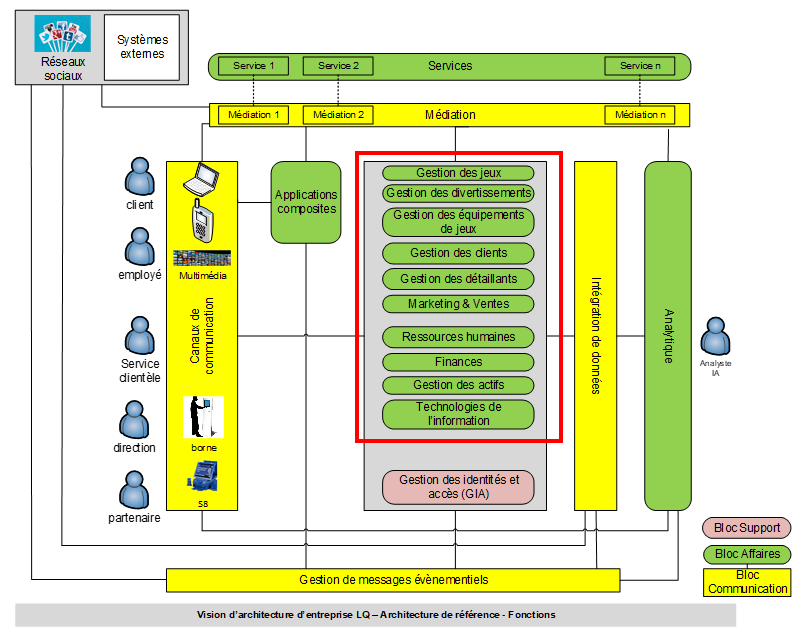
Exemple (non exhaustif et à réviser) des secteurs d’affaires de la SCQ par rapport à ceux de la SEJQ :

|  |  |
| --- | --- |
| **Casinos** | **Appareils de loteries vidéo (ALV)** |
| Établissements | Établissements |
| Produits&Services | Produits&Services |
| Clients | Clients |
| Partenaires | Partenaires |
| Équipements de jeux | Équipements de jeux |
| Marketing&Ventes | Marketing&Ventes |
| OperationsJeux | OperationsJeux |
| Programmes |  |
| Divertissement |  |
|  | Détaillants |

Voir si le secteur « Opérations de jeux » serait à envisager afin de le distinguer des activités d’affaires liées à la définition et au développement des Produits/Jeux.

Référence : <http://www.thesaurus.gouv.qc.ca/tag/terme.do?id=MDL1337>

Les secteurs d’affaires sont normalement arrimés avec les blocs Affaires présents dans l’Architecture d’entreprise.



En fait, les secteurs d’affaires, ce sont l’équivalent des « domaines d’affaires » dans la vision Architecture Affaires.

Par exemple, le secteur d’affaires « Clients » gère les clients de manière différente entre les domaines d’affaires Casinos, LV, PEL ou Loteries. C’est une réalité qui ne remet pas en cause l’objectif de vouloir à terme gérer le client dans une perspective stratégique d’entreprise (domaine LQ). Malgré tout, il y aura très probablement des capacités et fonctions d’affaires du Secteur Clients qui vont demeurer liées spécifiquement au domaine d’affaires.

### Contexte d’affaire (*business context*)

Représente normalement les capacités d’affaires d’un secteur d’affaires.

Exemple de contexte d’affaires pour le secteur Produits du domaine Casinos :

* Tables de jeux (TDJ)
* Poker
* Machines à sous (MAS)
* Keno
* Jeux électroniques (JE)

Voici par exemple quelques capacités d’un contexte d’affaires TDJ :

* Développement des nouveaux produits TDJ
* Piloter et configurer des TDJ
* Planifier le plancher des TDJ
* Opérer les TDJ

Il est aussi possible de décliner un contexte d’affaires en plusieurs sous-contextes si son envergure le justifie.

C’est à ce niveau de granularité (niveau contexte ou sous-contexte, ***bounded-context*** en anglais) que le microservice prend alors son sens.

Références :

<https://martinfowler.com/bliki/BoundedContext.html>

<https://docs.microsoft.com/en-us/dotnet/architecture/microservices/architect-microservice-container-applications/identify-microservice-domain-model-boundaries>

#### Modèle de domaine (domain model)

Dans une architecture orientée DDD (*Domain Driven Design*), le microservice expose et gère les données d’un groupe d’entités présentes dans le contexte d’affaires et qui ont une affinité et une cohésion confirmées.

La granularité de ce microservice dépend du nombre d’entités présentes dans ce groupe. Le choix, la cartographie et topologie des entités sont importantes peu importe le niveau de cette granularité. Elles servent notamment à définir le **modèle de domaine** (son indépendance et sa cohésion) mais aussi l’éventuelle organisation des routes et des ressources dans le service (l’API).

Voici un **exemple** permettant d’illustrer divers niveaux de granularité que pourrait avoir un éventuel microservice devant gérer et exposer des données pour le secteur **Clients** dans le domaine Casinos :

* Si le contexte du microservice est du même niveau que le secteur, le modèle de domaine du microservice pourrait regrouper alors toutes les entités qui décrivent les différents contextes du secteur :
  + Profil du client (informations nominatives, adresses, téléphones, attributs, préférences, segments, etc.)
  + Activités du client (parcours, jeux, achats, remises, etc.)
  + Les inscriptions du client (Jeu en ligne, abonnement loterie, jeu responsable, auto-exclusion, Casino Privilèges, clubs, etc.)
  + Les comptes du client (points, argent virtuel, argent monnayable, etc.)
  + Les services et support offerts au client (requêtes, plaintes, communications, etc.)
  + Etc.

Un microservice qui aurait pour mandat de gérer et d’exposer toutes ces données aurait un modèle de domaine vaste et complexe et serait alors peu granulaire **(granularité +)**.

* Si le contexte du microservice est à un niveau inférieur, par exemple au niveau du **profil du client**, le modèle de domaine pourrait alors regrouper typiquement l’information suivante :
  + Le nominatif du client ; ce qui est typiquement non spécifiques au contexte de LQ comme le nom, prénom, date de naissance, sexe, langue parlée, langue de communication, etc.
  + Les coordonnées du client (ex. adresses, téléphones, courriels, numéros en cas d’urgence, etc.)
  + Les attributs du client qui sont spécifiques au contexte de LQ (ex. accès interdit, sous surveillance, fusionné, etc.)
  + Les préférences du client (type de jeu préféré, casino préféré, breuvage ou restaurant préféré, courriel confidentiel, etc.)
  + La segmentation du client
  + Etc.

Un microservice qui aurait pour mandat de gérer et d’exposer toutes ces données aurait un modèle de domaine de moindre envergure que le précédent et serait alors plus granulaire **(granularité ++)**.

* Finalement, si le contexte du microservice est à un niveau encore plus précis, par exemple au niveau des **coordonnées du profil du client**, le modèle de domaine pourrait alors regrouper typiquement l’information suivante :
  + Les adresses du client
  + Les téléphones du client
  + Les courriels du client
  + Les courriels de récupération de compte
  + Les numéros de téléphones en cas d’urgence
  + Etc.

Un microservice qui aurait pour mandat de gérer et d’exposer toutes ces données aurait un modèle de domaine ayant une envergure encore moins importante que le précédent et serait alors encore plus granulaire **(granularité +++)**

Les groupes d’entités en orange et souligné ci-dessus sont donc tous des contextes pris à divers degré de granularité.

Le défi est de trouver la bonne granularité pour le modèle de domaine associé au microservice. C’est-à-dire le bon niveau et la meilleure affinité possible pour les entités de ce contexte d’affaires. Et que ce contexte d’affaires et ce modèle de domaine soit idéalement arrimés avec une capacité d’affaires unique pour le secteur.

### Domaine d’intégration (*integration domain*)

Le domaine d’intégration est le regroupement :

* d’une ou plusieurs capacités d’affaires métier ; typiquement en lien avec les domaines d’affaires.

ou (exclusif)

* d’une ou plusieurs capacités d’affaires de soutien.

### Domaine applicatif (*application domain*)

Le domaine applicatif peut se définir comme l’implémentation concrète, dans un composant logiciel, d’une ou plusieurs capacités d’affaires.

**Exemple :**

Bijou(SLC)

Tout comme la plupart des systèmes patrimoines (*legacy*), ce système possède son propre domaine applicatif. Il gère à la fois :

* une partie des capacités d’affaires Clients (gestion des membres du programme Casino Privilèges) ;
* une partie des capacités d’affaires Produits (transactions et activités consolidées de jeux tables et Keno provenant des systèmes SIJT et KenoXpert) ;
* une partie des capacités d’affaires Marketing en lien avec ses modules de reconnaissance (Calcul de points associés aux jeux tables et Keno, Promotions, Gratuités, Offres personnalisées)

Lorsque l’on parle des systèmes patrimoines (généralement des « *systems of records* – SOR »), il est plus juste de parler de domaines applicatifs.

Un microservice possède également son domaine applicatif. La particularité est qu’il implémente habituellement les capacités d’affaires liées à un contexte bien précis. Si le service adresse plusieurs capacités pour des contextes différents et de même niveau, il devient alors un mini ou un macroservice.

Puisque Bijou adresse plusieurs capacités d'affaires pour des contextes différents, l’API de domaine qui expose les données de Bijou serait alors soit de type macro ou mini selon que l’on ait respectivement un seul ou plusieurs API de domaine pour Bijou.

## API vs \*\*Service (macro, mini, micro, nano…)

|  |  |
| --- | --- |
| **Terme** | **Définition, caractéristiques et cas d’usage** |
| Service | Un service est une représentation logique d’une activité répétitive (ex. fonction d’affaire) qui possède un résultat spécifique. **C’est un artéfact totalement découplé de ses consommateurs.**  Domicile des règles d’affaires d’un domaine ou sous-domaine d’entreprise.  Hautement réutilisable et testable.  Il y a trois types de service :   * Domaine * Applicatif * Affaire   Le service implémente des patrons de conception (ex. BFF) |
| API | Interface qui expose un **service** (ou plusieurs services) à un consommateur par le biais d’un contrat.  Dans une architecture orientée service où le service est exposé par le biais du protocole HTTP à un consommateur, on parle en fait d’une **WEBAPI**. Par souci de simplifier le langage, le terme API est utilisé.  <https://en.wikipedia.org/wiki/Web_API>  Il y a six types d’API (voir section suivante) :   * Domaine * Applicatif * Affaire * ~~BFF~~ * Externe * Public   Un **service** exposé par un API est de même type que cet API.  Exemple : Un API de domaine expose un service de domaine. |
| Application | Logiciel (avec ou sans interface utilisateur) utilisé par un utilisateur, un service ou une autre application et qui peut optionnellement faire usage d’un ou plusieurs **services** (directement ou par l’entremise d’une **API**) pour réaliser une ou plusieurs fonctions d’affaires. |
| Macroservice | Composant logiciel combinant à la fois une **API** et un **Service** de domaine ayant les caractéristiques suivantes :   * Déploiement monolithique * Typiquement couplé à une source de données partagée avec d’autres services. * Ne possède pas normalement les droits exclusifs de gestion de ses données dans la source de données partagée. |
| Miniservice | Composant logiciel combinant à la fois une **API** et un **Service** de domaine ayant les caractéristiques suivantes :   * Déploiement indépendant * Typiquement couplé à une source de données partagée avec d’autres services. * Possède normalement les droits exclusifs de gestion de ses données dans la source de données partagée. |
| Microservice | Composant logiciel combinant à la fois une **API** et un **Service** de domaine ayant les caractéristiques suivantes :   * Gère ses propres données ; couplé uniquement à ses propres données * Déploiement indépendant |
| Nanoservice | Composant logiciel combinant à la fois une **API** et un **Service** de domaine ayant les caractéristiques suivantes :   * Réalise une seule est unique tâche. Habituellement une portion (et non la totalité) d’une fonction d’affaires * Offre un seul et unique point d’accès (*end-point*, route) * Déploiement indépendant et totalement découplé. * Peut utiliser l’agrégation, la composition ou encore l’orchestration dans la réalisation de sa tâche.   Autres termes : *serverless functionality*, *function as a service* (FaaS)  <https://peterdaugaardrasmussen.com/2019/08/14/what-is-a-nano-service/> |

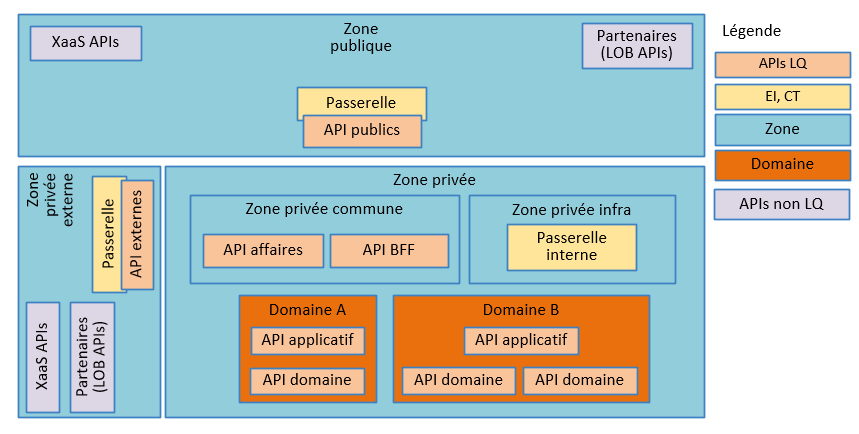
## Types d’API

|  |  |  |
| --- | --- | --- |
| **Type** | **Définition et responsabilité** | **Zone ou domaine d’implantation** |
| API domaine | **Contient la logique d'affaires** représentant une capacité d’affaires ou une fonction d’affaires associée à un domaine de l’entreprise.  Intervient seulement comme participant à un processus d'affaires ou applicatif. N’est pas le processus.  Les API de domaine de type microservice qui englobent une capacité d’affaires atomique et qui n’ont qu’une seule responsabilité (fonction d’affaires), peuvent être considérées comme des API d’affaires.  Peuvent faire de l’agrégation dans un contexte EDA (ex. microservice d’agrégation)  Les différents sous-types d’API de domaine sont :   * Macro (service) * Mini (service) * Micro (service) * Nano (service)   Macro/Mini/Micro : <https://www.gartner.com/document/3953730?ref=solrAll&refval=236026567>  \*\*Les macro et mini services ne se conforment pas aux normes MSA (microservices). | Domaine d’intégration |
| API applicatif | Héberge les règles spécifiques à un **contexte ou processus applicatif intra-domaine** (au lieu d’être embarqué dans l’application consommatrice).  Peut réaliser de l’**agrégation, de la composition ou de l’orchestration d’API de domaine**  Héberge les règles de transformation/normalisation vers le **domaine applicatif**.  Peut servir à **intégrer les APIs externes** qui ne sont pas d'origine « LQ » (SaaS, PaaS, systèmes de fournisseurs on-prem, etc.)  Peut héberger les **services applicatifs bidirectionnels** (websockets, service workers, etc.) qui requiert typiquement un canal de communication permanent (sticky session) comme les applications de clavardage, les applications de jeux en salle, etc.  Lieu possible d’implémentation des **services de requêtes** (ex. oData, graphQL)  Ne se conforme pas aux normes MSA (microservices).  Ce type d’API suit le **patron de façade**. <https://en.wikipedia.org/wiki/Facade_pattern> | Domaine d’intégration |
| API affaires | Héberge les règles spécifiques à un **contexte ou processus d’affaires inter-domaine**.  Peut réaliser de l’**agrégation, de la composition ou de l’orchestration d’API de domaine et applicatif**  Héberge les règles de transformation/normalisation vers le **domaine affaires (modèle canonique)**.  Réutilisable au niveau affaires au sein de l’entreprise (valeur ajoutée). Contribuent à la réalisation des capacités d’affaires intersectorielles qui génèrent de la valeur ajoutée pour l’entreprise  Ne se conforme généralement pas aux normes MSA (à moins d’un API de domaine atomique qui répond à une seule capacité d’affaires)  Ce type d’API suit le **patron de façade**. <https://en.wikipedia.org/wiki/Facade_pattern> | Zone commune |
| API BFF | L’API BFF est requise lorsque **l’information doit être présentée sur plusieurs canaux** (web, mobile, assistant, desktop, IoT, etc.). Elle est alors le lieu de la logique BFF (« BFF : Back-end for front-end »)  Elle est frontale aux API d’affaires, applicatif et de domaine.  Ne se conforme pas aux normes MSA.  Ce type d’API suit le **patron de façade**. <https://en.wikipedia.org/wiki/Facade_pattern> | Zone commune |
| API externe | Assure une **séparation des préoccupations entre la zone privée et le zone privée externe**.  La zone privée externe est une extension de la zone privée interne. Ce n’est pas l’internet public.  Sert à exposer pour les systèmes présents dans la zone privée externe, les API de la zone privée. Ou à l’inverse, de rendre accessibles pour les systèmes de la zone privée interne, les API présentes dans la zone privée externe.  L’implémentation se réalise généralement par une inscription dans la passerelle privée externe.  Ce type d’API suit le **patron de médiateur**. <https://en.wikipedia.org/wiki/Mediator_pattern> | Zone privée externe |
| API public  (entreprise) | Assure une **séparation des préoccupations entre la zone privée et le zone publique** (l’internet public).  Sert à exposer pour les systèmes présents dans la zone publique, les API de la zone privée. Ou à l’inverse, de rendre accessibles pour les systèmes de la zone privée interne, les API présentes dans la zone publique.  L’implémentation se réalise généralement par une inscription dans la passerelle publique.  Ce type d’API suit le **patron de médiateur**. <https://en.wikipedia.org/wiki/Mediator_pattern> | Zone publique |

Référence :

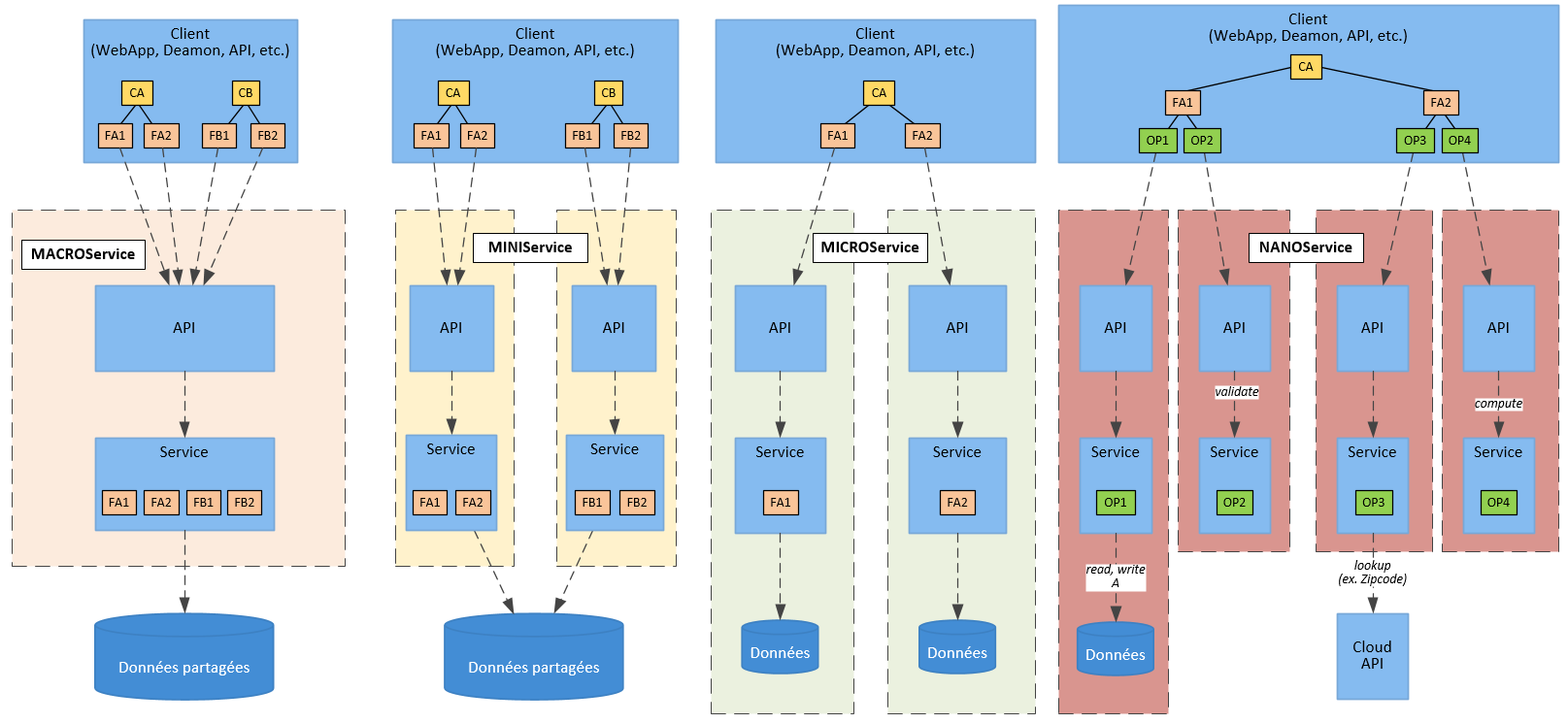
Stratégie d’intégration : <https://portailvpcti.loto-quebec.com/arch/ent/DevOps/Integration/Logimethods/Strate%CC%81gie%20d'inte%CC%81gration%20A2A%20et%20B2B%20v4.0_Final.docx>

BFF : (Back-end for front-end) : <https://docs.microsoft.com/en-us/azure/architecture/patterns/backends-for-frontends>



### Nomenclature pour le nom des services déployés

<système-information>.<produit-service>.<version>





## Lignes directrices URL

À aligner avec les travaux du CSI et architectures réseaux (AWS, rezonage, etc.)

À réviser avec DCAA avant adoption.

Passerelles

|  |  |  |  |
| --- | --- | --- | --- |
| **Uri** | **Description** | **Zone** | **Flux entrant/sortant** |
| web-gw.lotoquebec.com | Passerelle de la zone publique (DMZ) destinée aux applications web | Publique | Entrant |
| mobile-gw.lotoquebec.com | Passerelle de la zone publique (DMZ) destinée aux applications mobiles natives | Publique | Entrant |
| public-gw.lotoquebec.com | Passerelle de la zone publique (DMZ) destinée aux applications partenaires | Publique | Entrant |
| private-gw.loto-quebec.com | Passerelle de la zone privée externe (DMZ) | Privée | Sortant |
| gateway.lotoquebec.com | Passerelle de la zone privée interne (NDMZ) | Privée | Entrant et sortant |

APIs

|  |  |
| --- | --- |
| **Uri** | **Description** |
| api.loto-quebec.com | API de domaine |
| apiapp.loto-quebec.com | API applicative (voir impact DNS avec existant) |
| apiscq.loto-quebec.com | API d’affaires spécifiquement SCQ |
| apisejq.loto-quebec.com | API d’affaires spécifiquement SEJQ |
| apilq.loto-quebec.com | API d’affaires spécifiquement LQ |
| apiaff.loto-quebec.com | API d’affaires corporative Loto-Québec |
| apipub.loto-quebec.com | API public (entreprise) Loto-Québec |
| ~~bff.loto-quebec.com~~ | ~~API de canal, incluant les API bidirectionnels (Websockets, service workers, etc.)~~ |
| eda.loto-quebec.com | API applicative événementielle (voir si requis. Est-ce qu’on ne pourrait pas utiliser apiapp.loto-quebec.com ?) |
| webapp.loto-quebec.com | Applications web |
| mobileapp.loto-quebec.com | Applications mobiles |

Référence :

Conventions de nommage et meilleures pratiques de conception d’un API

* <https://docs.microsoft.com/en-us/azure/architecture/best-practices/api-design>
* <https://cloud.google.com/apis/design/>
* <https://restfulapi.net/resource-naming/>

### Exemple concret : identité d’un client (à partir de différent contexte – *binding context*)

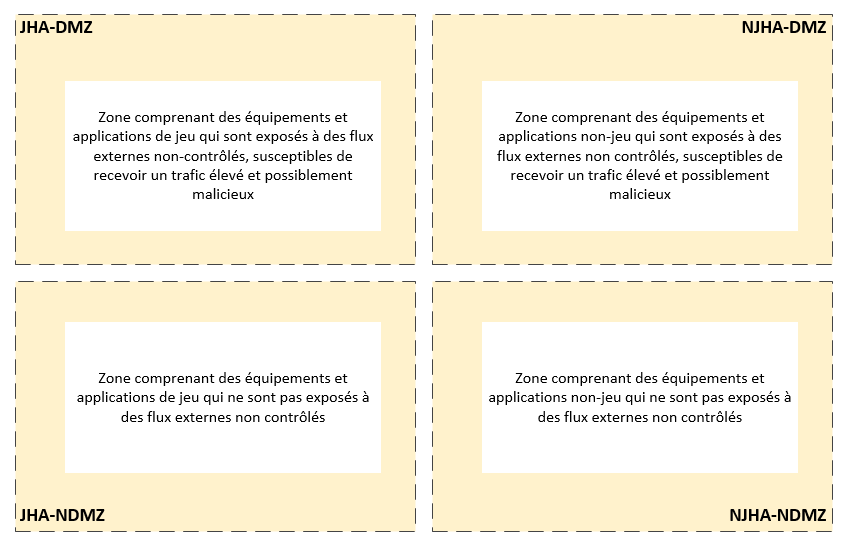
Le tableau suivant présente différentes routes exposant l’information touchant l’identité d’un client Loto-Québec.

Chaque route expose l’accès à l’identité du client en fonction du modèle/contexte qui est propre à son domaine ou sous-domaine.

Les exemples décrits sont fictifs mais permettre néanmoins d’illustrer l’approche proposée.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **URL (à titre indicatif seulement ; sert à illustrer le propos)** | **Détail** | **Zone**  Privée  Publique  Privée externe | **API**  Domaine  Applicatif  Affaire  BFF  Médiateur | **Sous-type**  REST  oData  GraphQL |
| **api**.loto-quebec.com/casino/slc/clients/profile/… | Domaine Clients  Sous-domaine = Bijou(SLC)  Exemple : SLC.API  Autre option : **api**.loto-quebec.com/slc/players puisque le jargon « SLC » parle plus de joueurs que de clients.  Le fait de mettre clients dans la route permet de se rapprocher du jargon entreprise et de la capacité d’affaires. | Privée | Domaine (miniservice) | REST |
| **api**.loto-quebec.com/casino/sijt/clients/profile/… | Domaine Clients  Sous-domaine = SIJT.CLI  Exemple : SIJT.CLI.API  Note :  SIJT possède 3 sous-domaines : TGM, SJT, CLI.  SJT et TGM sont des sous-domaines qui pourraient être liés à une capacité d’affaire Jeux. Alors que CLI est plus lié à une capacité Clients.  Autre option : **api**.loto-quebec.com/sijt/cli/players. Même explication que pour SLC ci-dessus. | Privée | Domaine (miniservice) | REST |
| **api**.loto-quebec.com/pel/openbet/clients/profile/… | Domaine Jeux  Sous-domaine = OpenBet  Exemple : OpenBet.Client.Account.API (LQAccount)  Du fait que l’on inscrit de la logique d’affaire propre à OpenBet dans l’API, il devient service de domaine au lieu d’être un service applicatif.  Autre option : **api**.loto-quebec.com/openbet/clients/account. | Privée | Domaine (miniservice) | REST |
| **api**.loto-quebec.com/entreprise/rcu/clients/profile/… | Domaine Clients  Sous-domaine = RCU  Exemple : RCU.Clients.Profile.API  API de domaine de type miniservice qui se nourrit des API de facade développées par Blueway pour le RCU.  L’API de domaine que LQ développe en façade au RCU permet d’assurer un découplage entre le RCU et les intégrations internes de LQ. Ce qui est conforme à la stratégie d’intégration (section 4.7.2, note G10).  À la différence de l’API d’affaires actuelle (FounirInfoProfilClient), cet API de domaine n’est pas orchestrateur.  Puisque le RCU contient déjà une information normalisée (canonique) l’intégration peut alors se faire via   * les services d’affaires (lieu d’orchestration si le processus est par exemple synchrone) qui appelle alors cet API de domaine * ou encore directement par cet API de domaine (qui peut alors émettre un événement) pour assurer une cohérence dans les autres systèmes (événement alors consommé par les services des autres systèmes). | Privée | Domaine (miniservice) | REST |
| **api**.loto-quebec.com/entreprise/clients/profile/identities | Domaine = Clients  Sous-domaine N1 = Profil  Sous-domaine N2 = Identité  Exemple : Clients.Profile.Identities.API  Cet exemple suppose un véritable microservice de domaine dont le seul mandat est la gestion de l’identité du profil d’un client Loto-Québec. Donc, que sa persistance n’est pas nécessairement celle du RCU (idéalement) mais une persistance exclusive au microservice.  Un microservice qui pourrait recevoir toute activité touchant l’identité du client via une architecture EDA (ex. MOM) et pourrait diffuser la mise à jour par principe de cohésion éventuelle (*eventual consistency*) vers les autres systèmes qui conservent l’identité du client (en fonction de leur sous-domaine respectif). Ces systèmes s’abonnent alors à l’événement via un service applicatif qui normalise l’information à leur sous-domaine et met ainsi à jour l’identité du client dans le système en question. Le RCU étant typiquement l’un de ces systèmes. | Privée | Domaine (microservice) | REST |
| **apiaff**.loto-quebec.com/clients/profile | Domaine = Clients  Sous-domaine N1 = Profil  Service d’affaires qui retourne l’information normalisée dans le modèle canonique client (ex. FournirProfilClient). Fait la composition de plusieurs informations liées au client (partie prenante individu, adresses, téléphones, segments, activités, programmes, informations de sites, etc.).  Peut faire usage des microservices du sous-domaine Client N2 ou plus (en dessous du sous-domaine Clients.Profil) ou encore faire appel au service de façade (miniservice) du RCU. | Privée | Affaires | REST |
| **public-gw**.lotoquebec.com/dialog-insight/clients/profile | Exposition externe d’une API privée. Passage par l’API Gateway public. Réécriture SSL. Redirection vers l’URL interne correspondante (ex. via service de découverte).  Authentification avec protocole autorisé (ex. SAML, OAuth ou OIDC), clé secrète et/ou filtrage IP.  Combinée à un flux OAuth-OBO ou OAuth-CCF au moment de la réécriture SSL et redirection.  Exemple fictif :  Un système comme Dialog Insight qui appellerait un service d’affaires pour mettre à jour (dans le RCU et autres systèmes par orchestration) le profil d’un client suite à un exercice de segmentation…  Redirection typique vers  **soe**.loto-quebec.com/dialog-insight/clients/profile  ou  **soa**.loto-quebec.com/clients/profile | Publique | Médiateur | REST |
| **public-gw**.lotoquebec.com/openbet/clients/account/payment | Idem.  Exemple fictif de Paysafe qui appellerait l’API de OpenBet dans la zone publique JHA-DMZ de LQ.  Redirection typique vers  **apiapp**.loto-quebec.com/openbet/clients/account/payment  Voir avec Louis-Julien si cohérent | Publique | Médiateur | REST |
| **apiapp**.loto-quebec.com/clients/profile/identities/graph/ | Rend possible l’utilisation des mécanismes de requêtes « GraphQL » vers le service de domaine sous-jacent. | Privée | Applicatif | GraphQL |
| **apiapp**.loto-quebec.com/clients/profile/identities/odata/ | Rend possible l’utilisation des mécanismes de requêtes « oData » vers le service de domaine sous-jacent. | Privée | Applicatif | oData |
| **eda**.loto-quebec.com/clients/profile/identities  ou encore  apiapp.loto-quebec.com/client/profile/identities/updateevent | Permet d’émettre un message vers une médiation (ex. MOM) lié à un événement s’étant déroulé sur l’identité du profil d’un client LQ. | Privée | Applicatif | REST |

## Zones de sécurité physiques

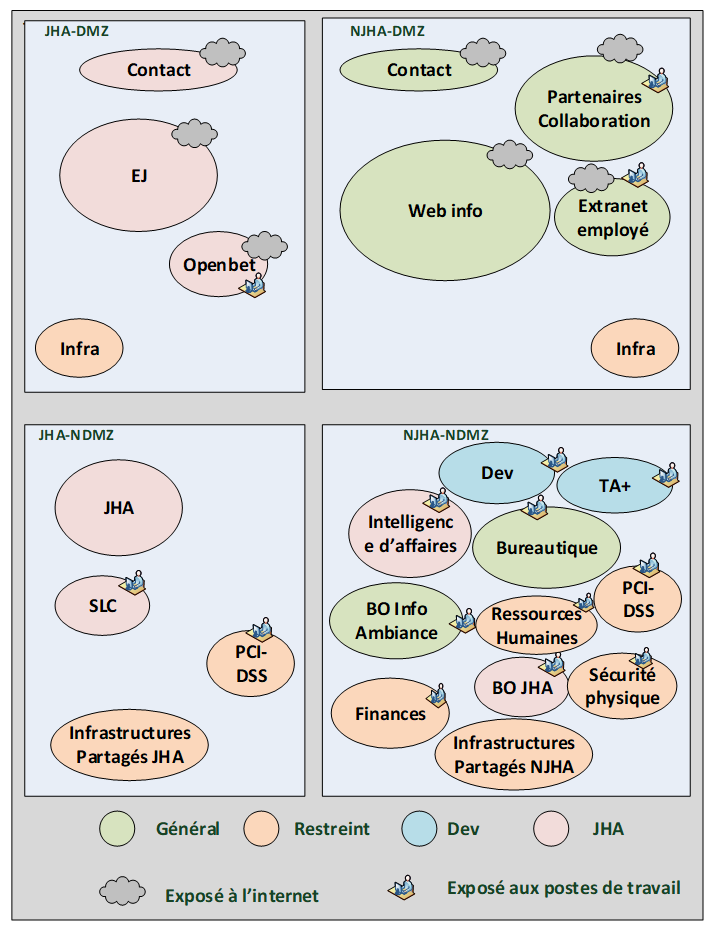


Références :

Architecture de sécurité 2017-2018 :

<https://portailvpcti.loto-quebec.com/prj/archec17/ar/Architecture%20de%20s%C3%A9curit%C3%A9%20Loto-Qu%C3%A9bec%20-%20Phase%20Syst%C3%A8me%20-%202%20-%20Flux%20r%C3%A9seau.docx>

## Zones de sécurité logiques

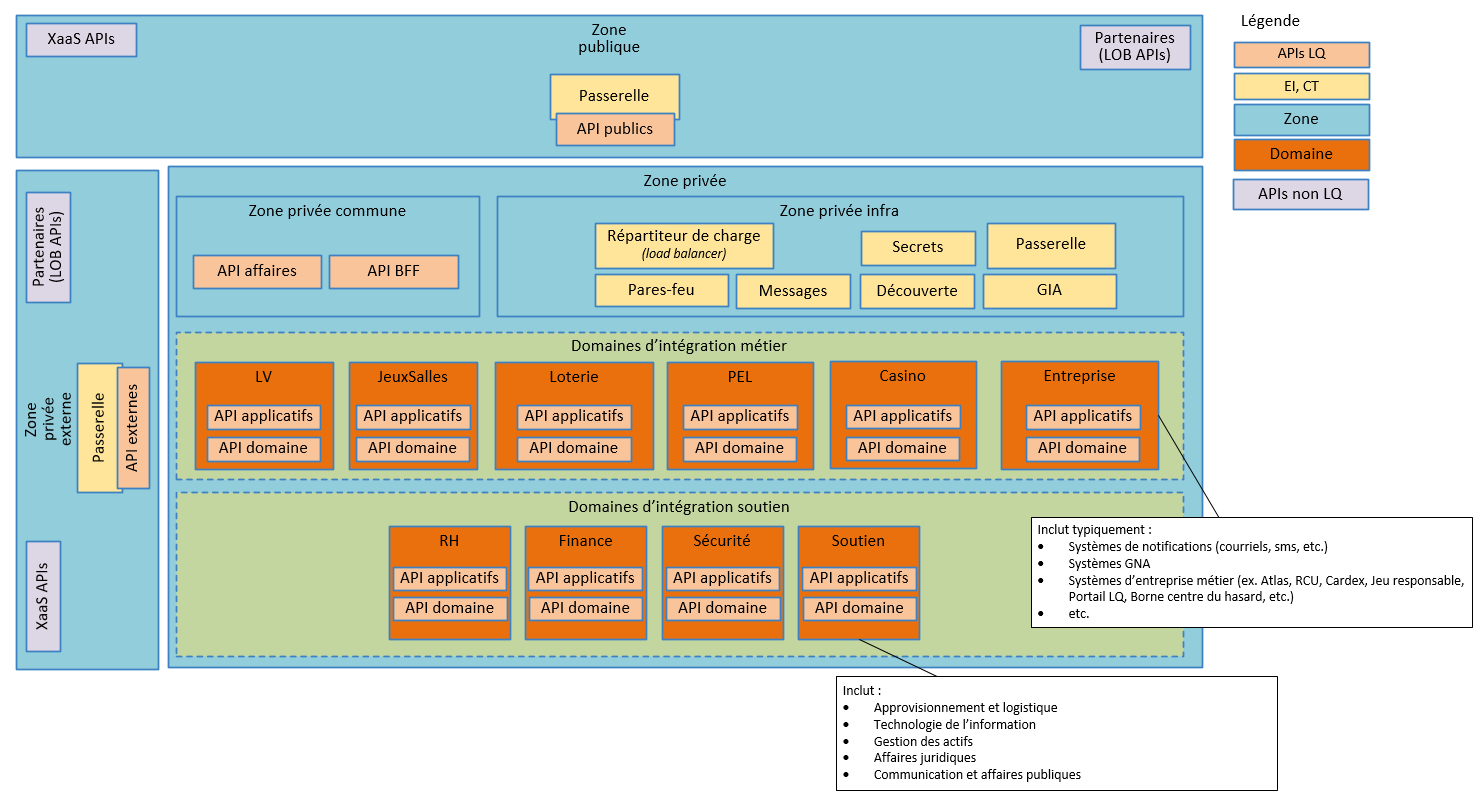


Références :

Architecture de sécurité 2017-2018 :

<https://portailvpcti.loto-quebec.com/prj/archec17/ar/Architecture%20de%20s%C3%A9curit%C3%A9%20Loto-Qu%C3%A9bec%20-%20Phase%20Syst%C3%A8me%20-%202%20-%20Flux%20r%C3%A9seau.docx>

## Zones et domaines d’intégration



|  |
| --- |
|  |
| Schématisation simplifiée |

### Définitions

|  |  |
| --- | --- |
| **Zone** | **Définition** |
| publique | Zone exposée au trafic de l’internet public |
| privée | Zone interne à LQ |
| Privée externe | Zone qui est une extension de la zone interne de LQ. Considérée comme faisant partie de LQ mais ayant une exposition à l’externe de la zone privée. |
| Privée commune | Zone interne à LQ réservée aux composants ayant une portée applicative transversale (inter‑domaines d’intégration) |
| Privée infra | Zone interne à LQ réservée aux composants techniques d’infrastructure |

|  |  |
| --- | --- |
| **Domaine** | **Définition** |
| LV | Domaine d’affaires métier des loteries vidéo (LV) |
| JeuxSalles | Domaine d’affaires métier des jeux en salles (ex. Bingo, Kinzo, BLP, etc.) |
| Loteries | Domaine d’affaires métier des loteries |
| PEL | Domaine d’affaires métier des produits offerts exclusivement en ligne |
| Casinos | Domaine d’affaires métier des casinos terrestres |
| Entreprise | Domaine d’affaires métier de Loto-Québec (systèmes d’entreprise, intelligence d’affaires, analytique, référentiels de données, entrepôts, etc.) |
| RH | Domaine d’affaires en soutien aux domaines d’affaires métier touchant les capacités d’affaires en ressources humaines. |
| Finances | Domaine d’affaires en soutien aux domaines d’affaires métier touchant les capacités d’affaires en finances. |
| Sécurité | Domaine d’affaires en soutien aux domaines d’affaires métier touchant les capacités d’affaires de sécurité. |
| Soutien | Domaine d’affaires en soutien aux domaines d’affaires métier touchant les capacités d’affaires « administratives » telles que les affaires juridique, l’approvisionnement et la logistique, les TI, etc. |

## Tenants (réseau)

#### Définition

Regroupement logique (*scope*) d’infrastructure matérielle partagée (ex. FW, VLAN, tables de routage, LB) et utilisé par un ensemble de produits (applications/progiciels). Le partage de l’infrastructure peut s’établir au niveau logique mais aussi physique mais à la base, le regroupement demeure logique. Les caractéristiques de ce regroupement d’infra partagé répondent aux besoins de l’ensemble des produits qui évoluent dans ce Tenant et sont soumis aux mêmes règles.

#### Caractéristiques d’un Tenant ; qui justifient normalement la différenciation d’un Tenant par rapport à un autre

* Doit avoir un seul responsable qui possède le pouvoir décisionnel sur son évolution et sa maintenance.
* Niveau de service (SLA) infra identique pour l’ensemble des produits évoluant dans ce Tenant
* Même règles de pare-feu (*Firewall*), VIPs (*load balancer*), tables de routage, sous-réseau local (*vlan*), etc. Donc, une affinité certaine en termes d’infrastructure.
* Déployables dans toutes les zones (500/pdc/dmz/interne) mais pas nécessairement déployés dans toutes les zones.
* Déployables autant en JHA qu’en NJHA.
* Doit être définissable pour un (ou plusieurs) environnement(s) d’exécution : DE, TI, TS, TA, FO, PP, PO.
* Doit contenir une envergure justifiée (masse critique de nœuds *node/host*)) qui ne soit ni trop vaste ni trop petite.
* Doit être redondant et préférablement en mode actif-actif
* Un Tenant est similaire à VPC (*Virtual Private Cloud*) sur AWS

#### Notes explicatives détaillées de Jean-Christophe 🡪 fichier texte

* Un tenant doit avoir un (et un seul) (propriétaire/) responsable technique avec pouvoir décisionnel sur la gestion (MACD) de celui-ci.
* Un tenant doit rassembler des systèmes avec une certaine affinité
* Un tenant doit être définissable pour un scope d'environnement précis (DE, TI, TA, PP, PO) et permet de reproduire individuellement et facilement la configuration d'infrastructure de celui-ci.
* Un tenant déployé en JHA doit absolument avoir en contrepartie un tenant de test en NJHA qui sert à valider que
  + les changements au tenant ne causent pas de problèmes à l'infrastructure et aux autres tenants
  + les changements à l'infrastructure ne causent pas de problèmes aux tenants
* Tous les tenants sont déployables dans toutes les zones (500/pdc/dmz/interne) mais ne sont pas nécessairement déployés dans toutes les zones
* Rezonage phase 1 (partitionnement réseau physique) avec les équipements en place nous permet 30 tenants avec zones internes (limite du répartiteur de charge A10) et 8 tenants zones DMZ (limite du pare-feu DMZ) par environnement JHA/NJHA. Augmenter ces limites exige une acquisition de licences et/ou matériel.
* Rezonage phase 2 (virtualisation réseau avec NSX) nous donnera plus de marge de manœuvre.
* Il devrait y avoir une certaine masse critique de nœuds (*node/host*) déployés dans un tenant pour justifier celui-ci et *l'overhead* en ressources (particulièrement en phase 1).
* Chaque tenant possède une politique de firewall distincte, appliquée par l'infra réseau mais dont le propriétaire/responsable technique est responsable des règles (sous les politiques établies par l'architecture de sécurité). Si un système dans le tenant A doit parler à un système dans le tenant B, les politiques de pare-feu (*firewall*) de chaque tenant doit être ajusté par leur responsable technique respectif.
* L'infrastructure étant mutualisée, une forme de *showback* ou *chargeback* avec les consommateurs/propriétaires de tenants devrait être mise en place pour pouvoir justifier les coûts d'opérations et assurer l'évolution de celle-ci.
* Le déploiement dans un tenant devrait se faire en mode automatisé/devops autant que possible de façon à pouvoir reproduire et tester celui-ci avec certitude.
* L'architecture des systèmes déployés dans l'infrastructure mutualisée doit tenir compte des niveaux de services offerts par l'infrastructure pour atteindre le niveau de service requis par son système.
* L'utilisateur / propriétaire d'un tenant/système déployé dans un tenant ne peut pas \*exiger\* un gel de toute l'infrastructure mutualisée sous-jacente et/ou de ces tenants voisins;

## Namespace (Openshift, Kubernetes)

### Définition d’un namespace

Regroupement logique de produits (applications/progiciels/services) qui partage les mêmes caractéristiques.

### Caractéristiques d’un namespace qui justifient la différenciation (création) d’un namespace par rapport à un autre

* Détient un responsable unique (*Product owner*)
* Lié à un ensemble de produits ayant la même affinité ; i.e.
  + évoluant dans le même domaine applicatif ou évoluant dans le même secteur d’affaires
  + évoluant dans le même domaine d’intégration
  + les produits peuvent être des applications (web, natives, desktop, etc.), des services autonomes (*deamon*), des services web, etc.
* Ne diffère pas selon l’environnement (DE, TI, TA, PP, PO) dans lequel il est implanté si ce n’est que le nom ou les configurations (applicatif, sécurité) qui lui sont appliqué.
  + Autrement dit, le contenu est normalement le même d’un environnement à l’autre. Dans la mesure où l’environnement est applicable pour les produits évoluant dans le namespace.
* Possède une portée (envergure) en liaison avec la complexité des produits qui y évoluent
  + Le namespace n’est pas démesurément trop vaste ou trop petit. Par exemple, le namespace ne contient pas un seul produit à moins que la complexité, l’envergure ou certaines dispositions (ex. sécurité) le justifie.
* Peut contenir des produits appartenant à des zones logiques de sécurité distinctes (ex. EJ vs OpenBet)
* Ne contient pas des produits appartenant à des zones physiques de sécurité distinctes (JHA vs NJHA)

### Nomenclature de nom d’un namespace

* Le nom d’un namespace (<namespace-name>) se décline ainsi :
  + ns-<integration-domain>-<business-sector ou application-domain*>[-<organization-name>]*
  + *[-<organization-name>]* est optionnel et sert à préciser l’unité d’affaire responsable du groupe de produits ou du domaine applicatif.
    - Utile lorsque les produits ou domaines applicatifs ont la même appellation dans différentes unités d’affaires ou domaines d’intégration tels que DIF et GDN.
  + Exemples : ns-casinos-slc, ns-casinos-dif-scq
* Le nom d’un produit (<product-name>) se décline ainsi :
  + <systeme-information>-<product-type>*[-<product-name>][-<version>]*
  + *[-<product-name>]* est optionnel et sert à préciser le nom du produit lorsque le type n’est pas suffisant.
  + Exemples pour le système d’information Casino Privilèges (acronyme = SLC) :
    - slc.app.bijou
    - slc.app.datacard
    - slc.app.acp-console
    - slc.webapp.acp
    - slc.api
    - slc-svc-casinoprivileges-v14
    - slc-svc-casinoprivileges-v28
* Le nom d’un environnement (<environment-acronym>) se décline ainsi :

|  |  |  |  |
| --- | --- | --- | --- |
| Environnement | Acronyme | Environnement | Acronyme |
| Production | po | Système | ts |
| Pré-production | pp | Intégration | ti |
| Formation | fo | Développement | de |
| Acceptation | ta | Laboratoire | la |

* Le nom du cluster Openshift (<cluster-name>) se décline ainsi :
  + < openshift-product-acronym>-cluster-<environment-acronym>
  + Exemple : okd-cluster-po
* Le nom complet d’un produit dans un namespace se décline donc ainsi :
  + <product-name>.<namespace-name>-<environment-acronym>.<cluster-name>
  + Exemples
    - slc-app-bijou.ns-casinos-slc-po.okd-cluster-po
    - slc-svc-casinoprivileges-v28. ns-casinos-slc-po.okd-cluster-po
    - tdjsue-api.ns-casinos-products-tdj.osd-cluster-de

### Sécurité du namespace

**À approfondir et à valider (voir courriels et documentation fournies par Louis-Julien. Voir aussi les travaux de nos collègues ayant participés au projet de conteneurisation)**

#### Considération administrative

La création du namespace et les autorisations d’accès à ces namespaces afin d’y permettre le déploiement des pods sont gérées avec un principe RBAC. Donc, via des groupes de sécurité que l’on associe à la fois aux namespaces et aussi aux comptes (utilisateurs ou services) responsables de cette administration.

L’administration peut être réalisée soit :

* manuellement par un ou des utilisateurs (équipe réduite), par exemple via une console ou par ligne de commande, ou
* automatiquement par un compte de service via l’usine logicielle (CD…*Release Management*).

#### Considération opérationnelle

La gestion de l’accès logique par les applicatifs aux namespaces est plutôt adressée via un libellé (tag) sur les pods déployés dans le namespace et par des règles de sécurité de réseau (*network policies*).

Un pod est alors associé obligatoirement à une zone logique de sécurité ainsi qu’à un domaine d’intégration.

Les pods appartenant à la même zone logique de sécurité peuvent communiquer entre eux sans que des règles de sécurité réseau ne soient impliquées.

Un pod A désirant communiquer avec un service externe au cluster de conteneurisation ou encore communiquer avec un pod B appartenant à une autre zone de sécurité logique doit alors respecter les règles de sécurité de réseau « sortantes » liées à ce pod A.

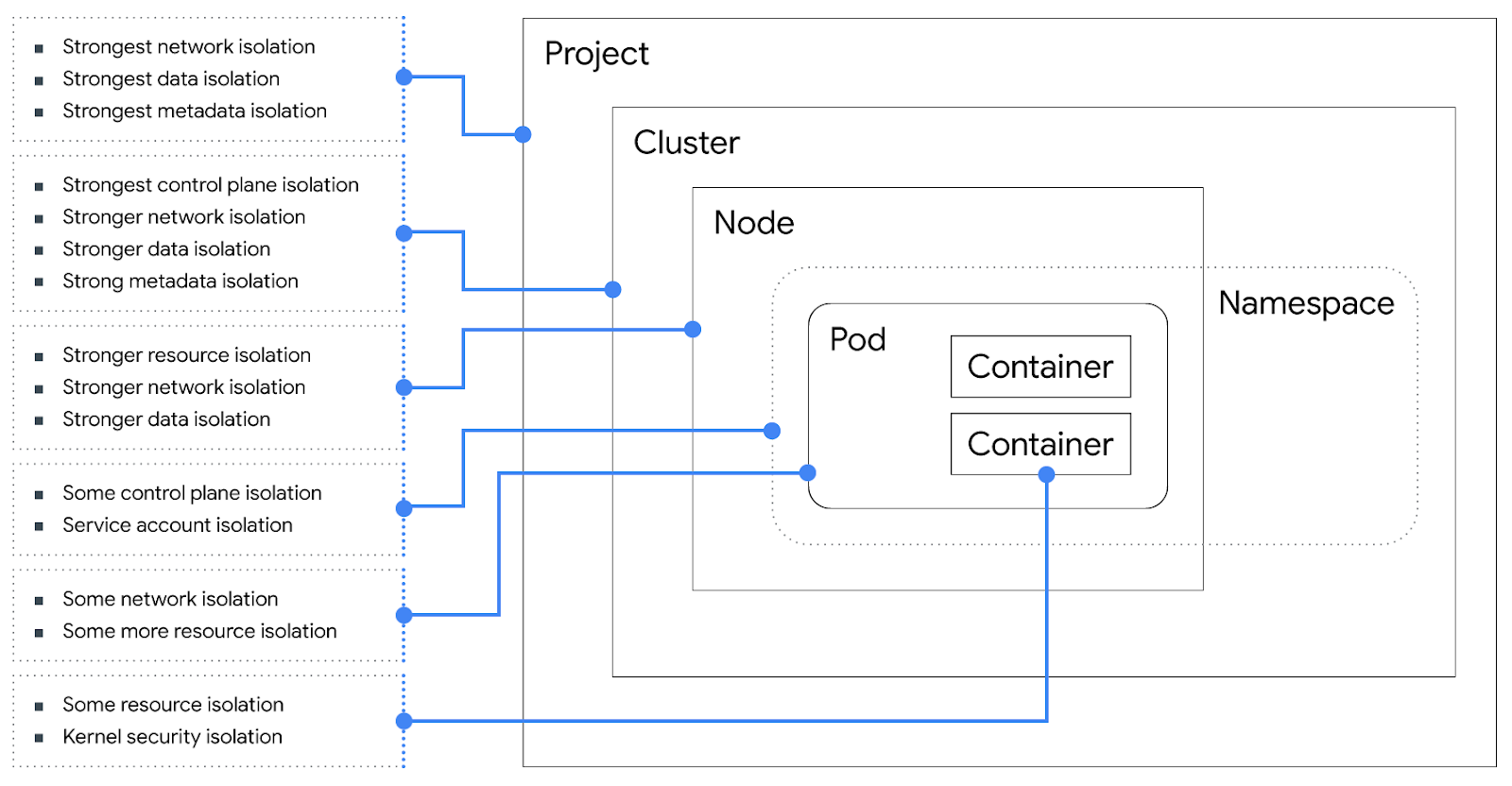
et au même domaine d’qui lui sont associée. Ces règles sont différentes des règles administratives.

Securité

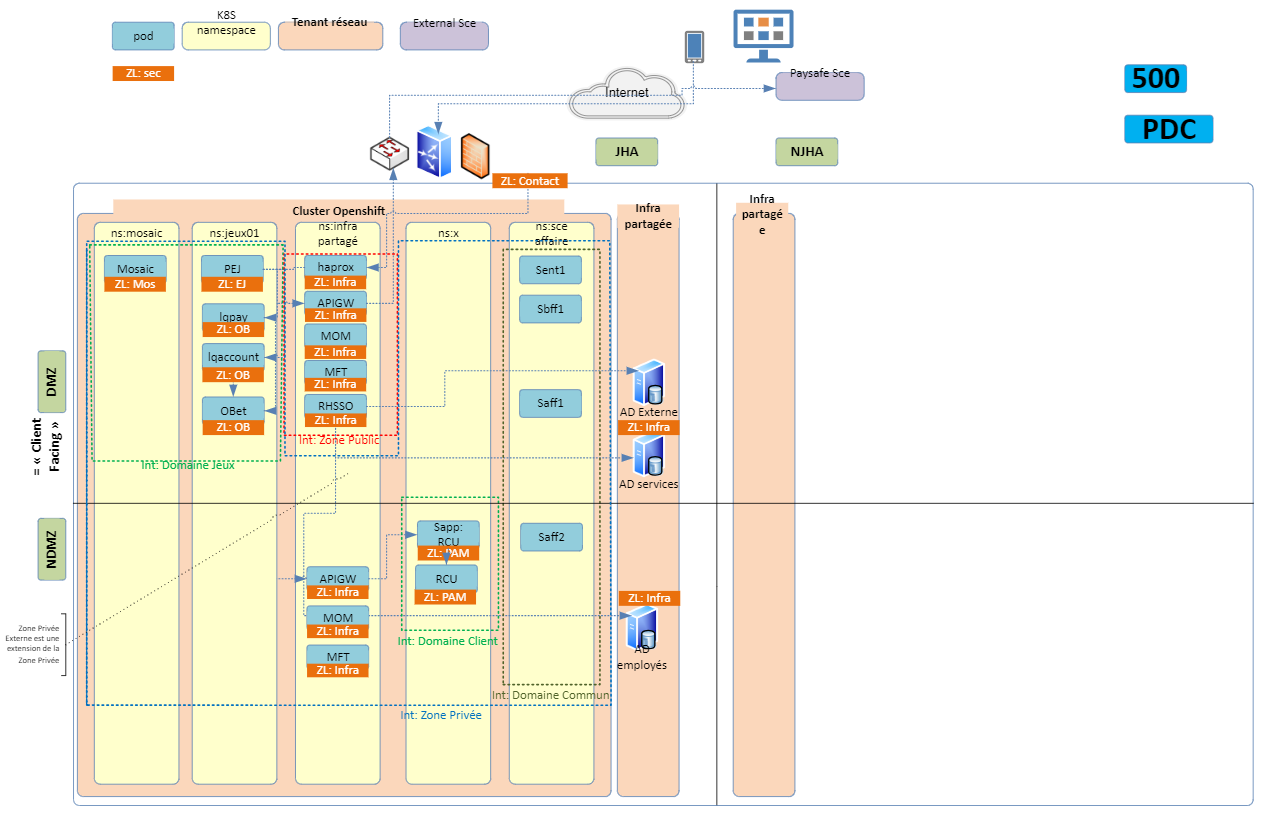
Contrôle souhaité : les pods doivent avoir leur Tag. Un pod qui n’a pas de tag ne pourrait théoriquement pas « parler » à d’autres pods.

Voir applicabilité de la network policy vs domaine d’intégration vs namespace.

Équipe réduite ou usine logicielle à définir les network policies lors du déploiement des pods.

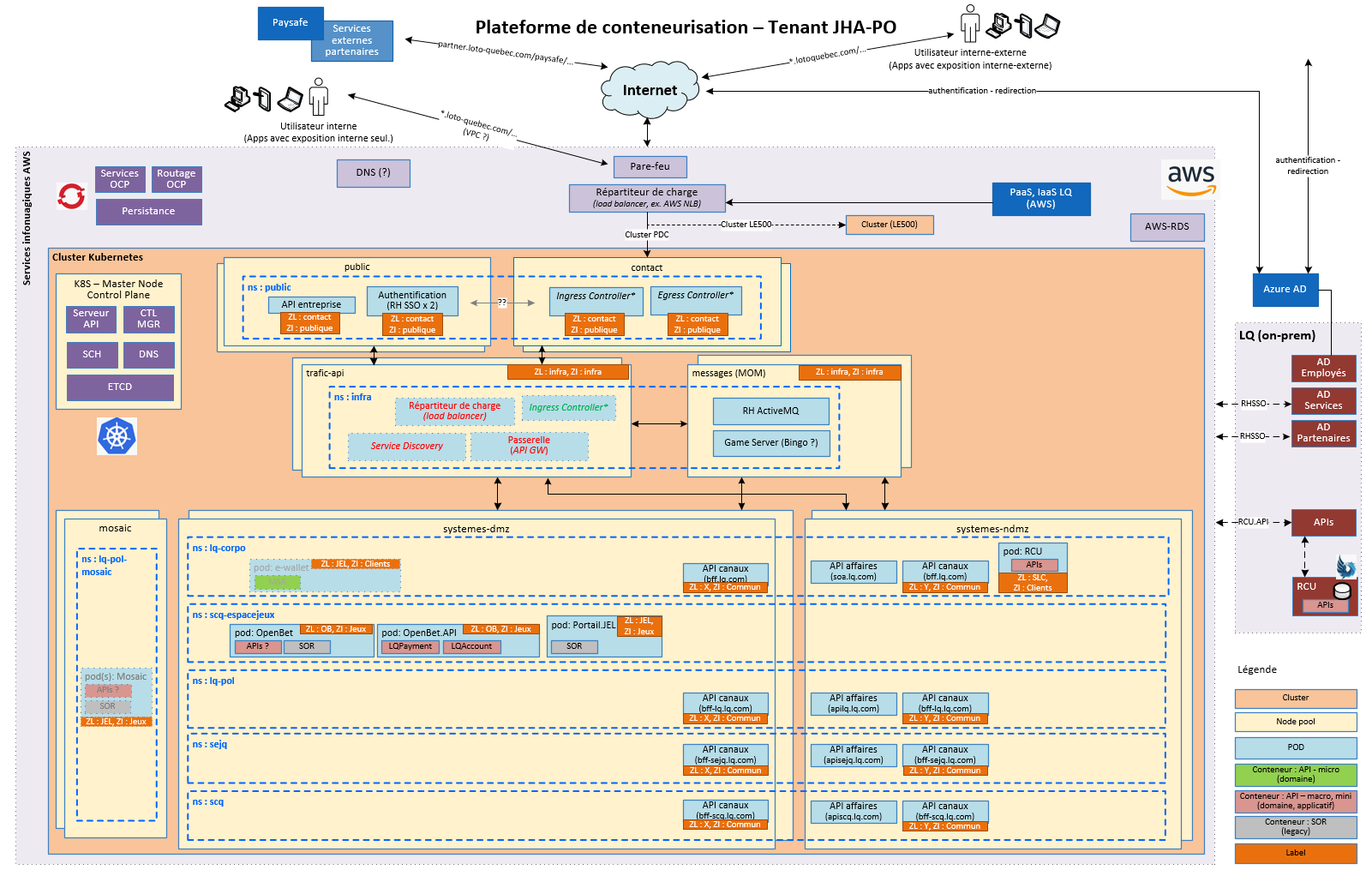


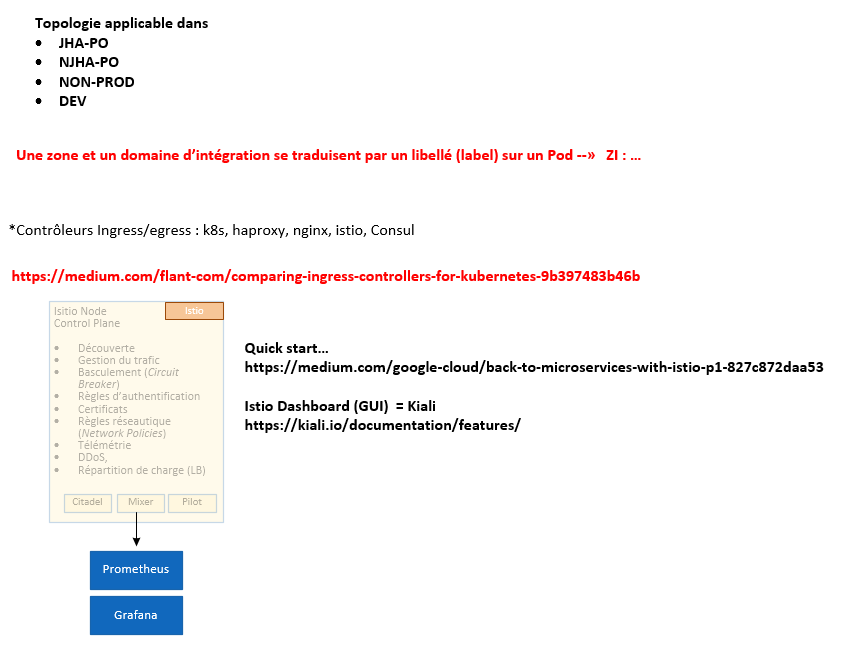
## Patrons d’intégration concrets - Openshift



Notes :

* Une zone sécurité logique est égale à un tag Kubernetes (catégorie).
  + A valider si un système a une relation de 1 à 1 avec un tag, ou si c'est 1 à n. (voir page To Do)
    - 1 système : relation 1-1 avec un tag (selon PH)
    - 1 système : relation 1-n avec un tag (selon JC)
* Une zone de sécurité physique n'est pas égale à un VLAN
* Une zone de sécurité physique n'est pas égale à un Subnet





## API Gateway

Termes « API Gateway » :

* Le patron de conception :
  + <https://docs.microsoft.com/en-us/dotnet/architecture/microservices/architect-microservice-container-applications/direct-client-to-microservice-communication-versus-the-api-gateway-pattern#what-is-the-api-gateway-pattern>
* Le produit acquis :
  + NGINX Plus, Ambassador, Kong, 3Scale, haProxy, etc.
* Le produit développé :
  + Zuul, Ocelot, Spring Cloud Gateway

Quelles seront les responsabilités/capacités déléguées à l’API Gateway (produit acquis) ?

* Routage, filtrage
* Authentification et autorisation réseau (sécurité IP, publique vs privée, network policies vs tags, zone logique, zone intégration)
  + Ex. URL X ne peut parler à URL Y,
* Orchestration
* Agrégation
* Authentification et autorisation applicative : saml2, oauth2, odic, jwt, etc.
* Découverte de service (Service discovery)
* *Response Caching*
* Retry et Circuit Breaker
* Rate Limiting
* Load Balancing
* Logging et auditing (tracing)
* Transformation de requêtes (headers, querystring, uri variables, etc.)
* Transformation de protocole (ex. REST vers SOAP)
* Transformation de contenu (content transformation)
* IP ou URL Whitelisting

En théorie, un microservice (ex. API de domaine) ne doit pas dépendre d’un autre microservice. C’est le principe d’autonomie et d’indépendance des MSA.

Ainsi, seules des Apps (Web, Mobile, Desktop, etc.) et des API applicatifs, affaires, canal ou entreprise ou encore des macro ou mini-services de domaine peuvent typiquement appeler un API de domaine.

Exemple : Parce que la prémisse « actuelle » veut que le point à point soit autorisé entre 2 API du même domaine, un API applicatif du domaine A ne serait pas tenu de passer par le API Gateway pour aller vers l’API de domaine A.

Comment alors s’assurer que les capacités sont quand même mises en œuvre dans les API si l’API Gateway n’est pas impliqué dans les appels ?

Est-ce que l’API Gateway ne serait pas alors un passage obligé ?

N’est-ce pas le principe d’un service-mesh ? Tous les APIs d’un cluster K8S sont gérés par le MESH peu importe le type de l’API… Ils sont tous inscrits au registre peu importe qu’ils soient de type domaine, applicatif, affaire, macro, mini, micro, etc.

Comment concilier les mises à jour successives de l’API Gateway s’il est partagé par plusieurs équipes… ? Qui fait les mises à jour ? Quel stack techno ? etc.

L’appel d’offres parle de 6 instances de l’API Gateway en production.

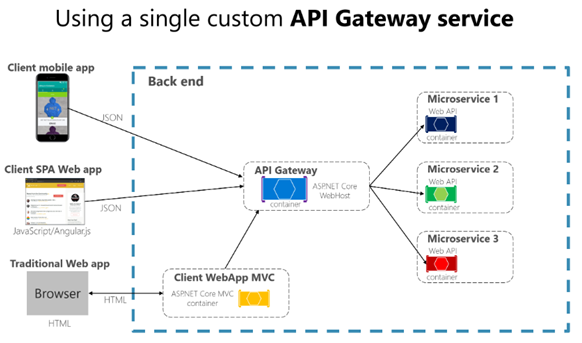
Avait-on alors une vision plus centralisée de l’API Gateway ? 2 ou 3 par cluster (JHA, NJHA) ? + on-prem ??

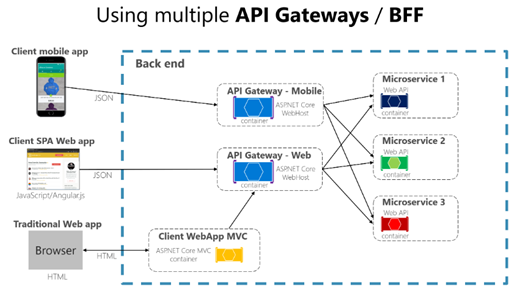
Et donc pas une vue personnalisée ou chaque produit possède son propre API Gateway ? Comme l’approche SPEL ou Paysafe avec Zuul… À valider

Comment concilier par exemple l’orchestration dans la vision (API Gateway Pattern) avec celle d’un service d’affaire traditionnel d’une architecture SOA (service de la couche Affaires)

Patron API Gateway :

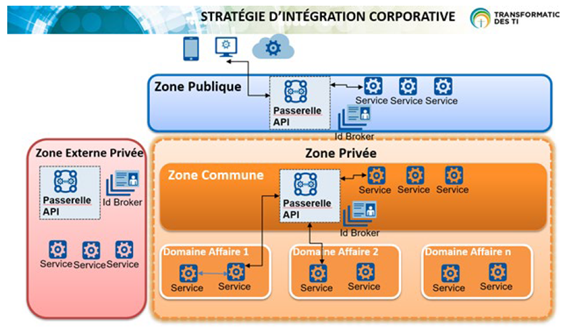
<https://docs.microsoft.com/en-us/dotnet/architecture/microservices/architect-microservice-container-applications/direct-client-to-microservice-communication-versus-the-api-gateway-pattern#what-is-the-api-gateway-pattern>





L’API Gateway est ainsi représenté comme un service qui propose un point d’entrée unique et faisant de l’orchestration ou encore de l’agrégation de services pour un client (un système). Ça commence drôlement à ressembler aux services d’affaires et de canaux (BFF) dont on parle actuellement. À la différence que nos services d’affaires et de canaux actuels ne sont pas alignés (il me semble) pour mettre en œuvre l’ensemble des capacités d’un API Gateway. De mon point de vue, il nous faut clarifier cela.

Versus ce qui est demandé dans l’appel d’offre



Je pense qu’il faut clarifier le rôle que l’on veut donner à nos API Gateways.

**Quelles seront les responsabilités/capacités déléguées à l’API Gateway ?**

• Routage, filtrage

• Authentification et autorisation réseau (sécurité IP, publique vs privée, network policies vs tags, zone logique, zone intégration)

• Orchestration

• Agrégation

• Authentification et autorisation applicative : saml2, oauth2, odic, jwt, etc.

• Découverte de service (Service discovery)

• Response Caching

• Retry et Circuit Breaker

• Rate Limiting

• Load Balancing

• Logging et auditing (tracing)

• Transformation de requêtes (headers, querystring, uri variables, etc.)

• Transformation de protocole (ex. REST vers SOAP)

• Transformation de contenu (content transformation)

• IP ou URL Whitelisting

