DOCUMENT TECHNIQUE

API iDCuvePro

Produit :	API iDCuvePro	Référence :	RD-FOR-003 V1.0
Document	Document technique API		
Version :	2.1		

Rédacteur	Cédric SIMON		
Diffusion			

Historique des mises à jour

Date	Objet de la modification	N° version
07/06/2018	Création du document	1.0
23/02/2022	Mise à jour de la procédure d'installation Mise à jour pour API en .Net 5.0 Suppression des configurations obsolètes Ajustement pour la configuration des accès aux bases	2.0
28/05/2025	Nouveaux paramètres API Partie sur l'exposition Nouveaux paramètres pour tracer les requêtes Journalisation effectuée avec NLog .Net 9 Documentation mise en cluster Documentation cache Nouveaux paramètres pour les clés d'API	2.1

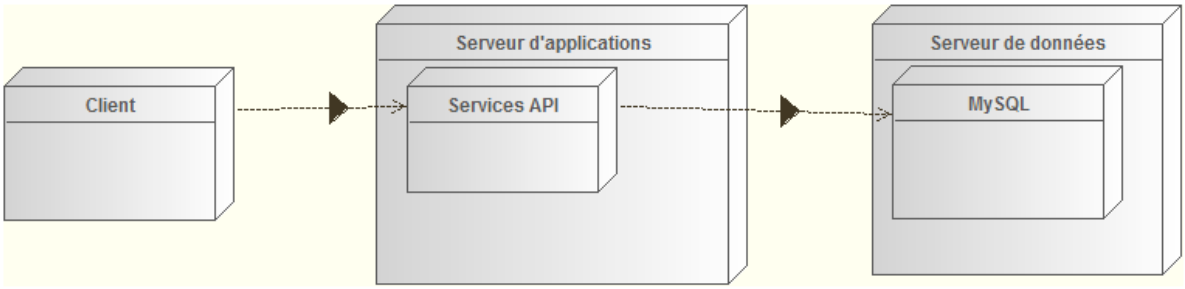
Systemes SAS au capital de 530 250 € - RCS Bordeaux – Siret 343 280 665 00039
Siège social : 18 av. de Chavailles, CS20227, 33525 Bruges Cedex, France – Tél. : +33(0)5 56 11 28 28 – Fax : +33 (0)5 56 11 28 29
Mail : info@idsystemes.com – Web : www.idsystemes.com

SOMMAIRE

1. INFRASTRUCTURE

1.1. Schéma

Le schéma ci-dessous a pour but d'indiquer les éléments minimums nécessaires pour le déploiement des API pour iDCuvePro.



La solution est composée d'une WebApp qui doit être hébergée dans un IIS. Cette webapp doit accéder à la base de données iDCuvePro (sous MariaDB ou MySQL).

1.2. Flux

Origine	Destination	Protocole (Port)	Description
<Extérieur>	Serveur IIS (Services)	HTTPS(TCP/443) ou HTTP (TCP/80)	Services exposés sous forme d'API REST
Serveur IIS (Services)	Serveur BD	MariaDB (TCP/3306)	Accès à la base de données

Les ports indiqués ici sont les ports standards et peuvent varier selon la configuration des logiciels. Concernant les ports utilisés par IIS, ils sont mis à titre indicatif, mais peuvent être paramétrés différemment selon le déploiement souhaité.

2. PREPARATION DE L'ENVIRONNEMENT

2.1. Eléments nécessaires

Logiciel	Version minimale
IIS	7
.Net (+ hosting asp.net)	9.0
WebDeploy (uniquement pour l'installation)	3.6

2.2. OS supportés

Les OS supportés sont les versions de Windows pouvant exécuter le runtime .Net 9.0. Le tableau suivant est donné à titre indicatif, la documentation officielle Microsoft est à utiliser comme référence (https://github.com/dotnet/core/blob/main/release-notes/9.0/supported-os.md).

OS	**Version minimale**
Windows 11	Version 22000 +
Windows 10	Version 1607 +
Windows Server 2016	
Windows Server 2019	
Windows Server 2022	

Pour les versions Windows les plus ancienne, il peut être nécessaire d'installer des composants complémentaires, notamment le runtime Visual C++ 2015-2019 (https://aka.ms/vs/16/release/vc_redist.x64.exe pour la version 64 bits et https://aka.ms/vs/16/release/vc_redist.x86.exe pour la version 32 bits).
Les informations complètes peuvent être trouvées sur le site de Microsoft : https://learn.microsoft.com/fr-fr/dotnet/core/install/windows?tabs=net90

2.3. IIS

Installer IIS : dans le cas d'une version Server de Windows, il s'agit d'ajouter le rôle « Serveur Web (IIS) ». Les services de rôle suivant doivent être intégrés :

- Fonctionnalités HTTP communes :
  - Contenu statique
  - Document par défaut
  - Erreurs http
- Développements d'application
  - ASP.Net
  - Extensibilité .Net
  - Extensions ISAPI
  - Filtres ISAPI
- Outils de gestion
  - Console de gestion IIS

Les autres éléments peuvent être ajoutés et supprimés librement (pas d'obligation ni contre-indication)

2.4. .Net

La version du runtime .Net 9.0 doit être installée. Il faut également qu'il y ait le runtime « ASP.Net Core Runtime 9.0 ». Un package permet d'installer les 2 en même temps. A l'adresse https://dotnet.microsoft.com/en-us/download/dotnet/9.0, il suffit de choisir « Hosting bundle » dans

la partie « Asp.net Core Runtime »

2.5. WebDeploy

WebDeploy est l'outil Microsoft permettant l'installation simplifiée des packages de sites web dans IIS. Cet outil peut être téléchargé via ce lien : https://www.iis.net/downloads/microsoft/web-deploy

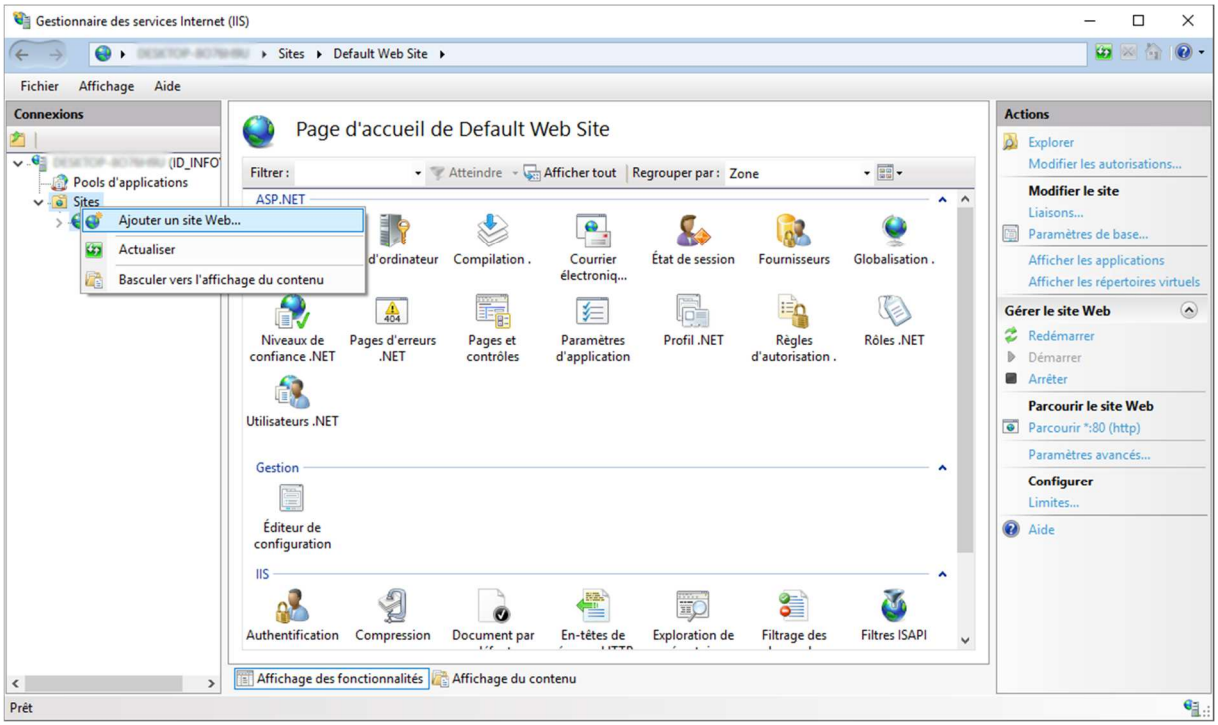
3. DEPLOIEMENT

3.1. Création du site IIS

Cette partie présente des actions communes qui doivent être réalisées par la suite.

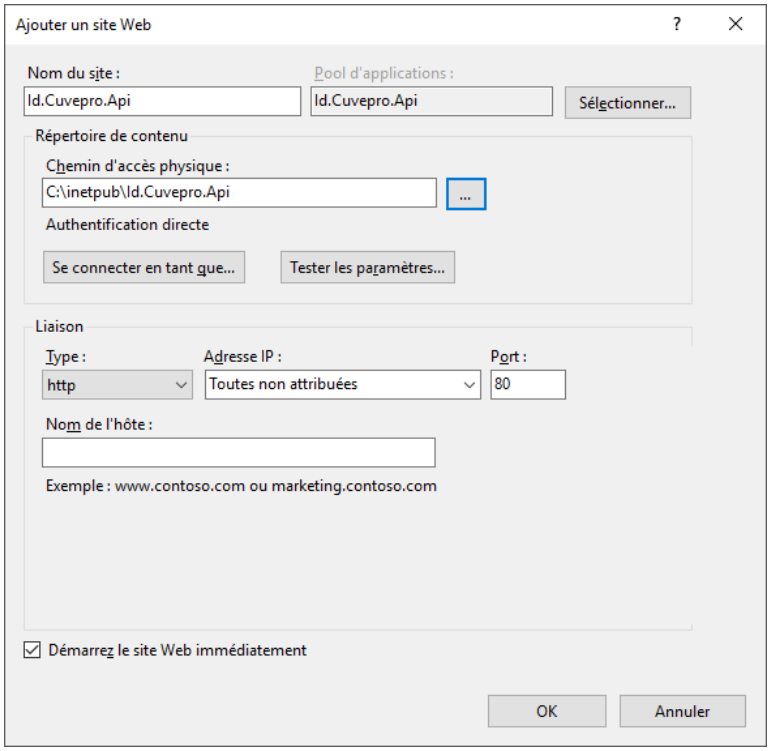
3.1.1. Création de sites sous IIS

Dans le gestionnaire des services internet (IIS)[1], faire un clic droit sur « Sites » et choisir « Ajouter un site Web »



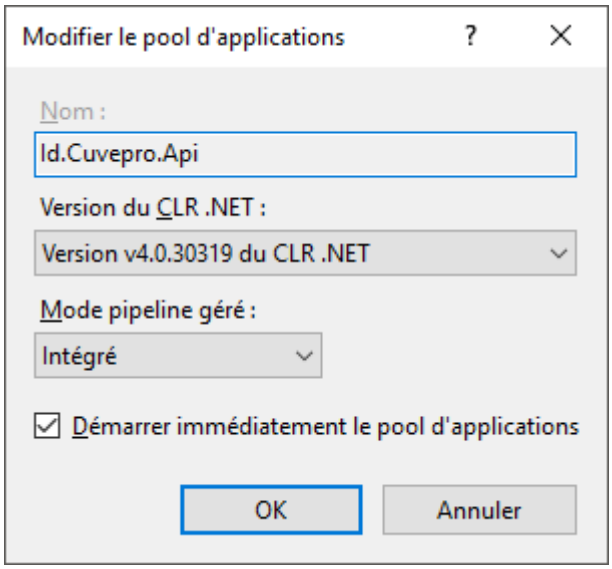
Spécifier le nom du site, le chemin physique

[1] Accessible dans la gestion du serveur sous les versions Windows Server ou dans les outils d'administration de Windows



3.1.2. Configuration des pools d'applications

Dans le gestionnaire des services internet (IIS), aller dans l'option « Pools d'applications » et double cliquer sur le pool à configurer. Choisir la version du framework .Net[2] et le mode de pipeline



3.2. Déploiement

[2] Dans le cas où l'on sélectionne la version 4.0 et que la version 4.5 est installée, c'est la version 4.5 qui sera utilisée par IIS.

Pour déployer les API dans un site en spécifiant son nom, il faut modifier le fichier Package.SetParameters.xml et la ligne IIS WebApplication Name. Par exemple :

Exemple

```
<?xml version="1.0" encoding="utf-8"?>
<parameters>
  <setParameter name="IIS Web Application Name"
value="Id.Cuvepro.Recette.Api" />
</parameters>
```

Déployer le site via le package :

- Ouvrir une invite de commande en mode administrateur
- Se placer dans le répertoire contenant le package d'installation
- Exécuter la commande suivante :
  `Id.Cuverie.WebApi.Net.deploy.cmd /T`
  et corriger les erreurs éventuelles indiquées par l'outil
- Si la commande précédente indique que l'environnement est prêt, exécuter la commande suivante :
  `Id.Cuverie.WebApi.Net.deploy.cmd /Y`

3.3. Mises à jour

Pour une mise à jour d'une webapp existante, il suffit d'utiliser la procédure d'installation décrite en 3.2

4. CONFIGURATIONS

Les fichiers de configuration se trouvent dans le répertoire App_Data\Settings de la webapp. Ces fichiers ne sont ni écrasés ni modifiés lors d'une mise à jour. Pour les modifier, il est nécessaire de disposer des droits suffisants. Il est donc conseillé d'ouvrir ces fichiers avec les droits administrateur.

4.1. Bases de données

Cette section présente la configuration de base pour les chaînes de connexion aux bases de données.
Les chaînes de connexion se situent dans le fichier App_Data\Settings\connections.config.
Il suffit d'adapter la chaîne de connexion pour la base iDCuvePro

Exemple

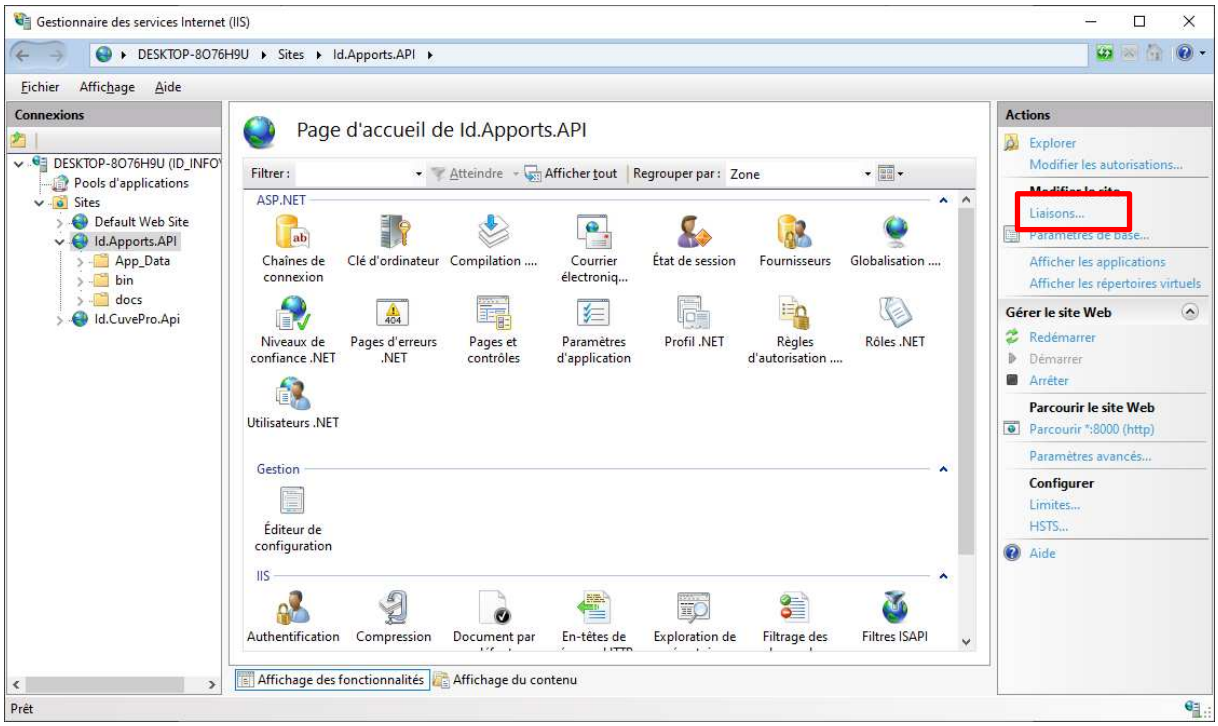
```
<connectionStrings>
  <add name="Entities"
connectionString="Server=localhost;Port=3308;Database=cderlcvu_champagne_prod;…" />
</connectionStrings>
```

La gestion du multi tenant est présentée dans le § 4.5.

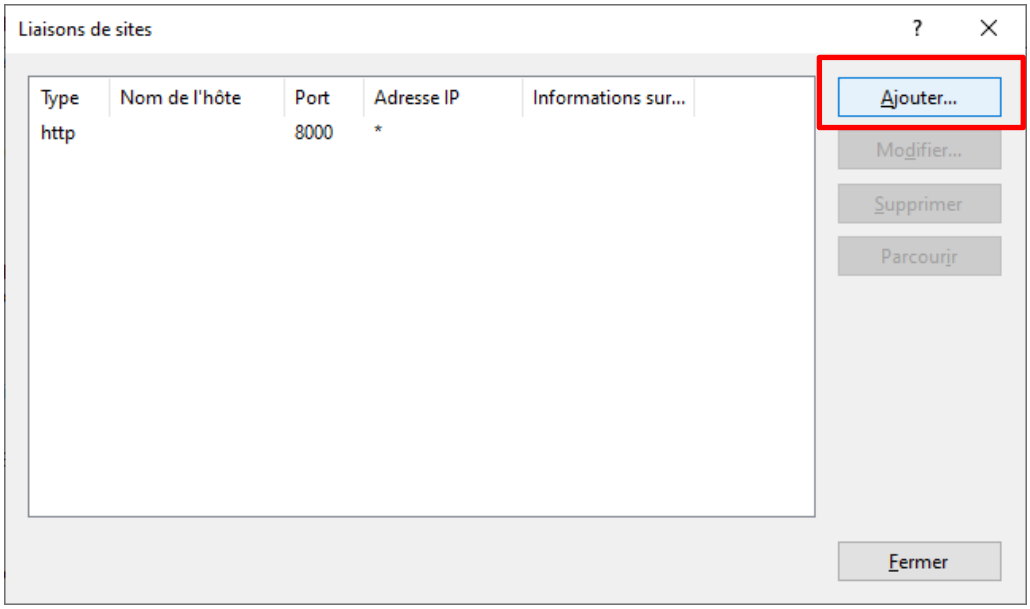
L'accès à la base de données **doit** se faire avec un compte utilisateur dédié avec les privilèges adaptés et restreints aux seuls besoins de l'API. Les privilèges suivants sont nécessaires au bon fonctionnement des API : EXECUTE, SELECT, SHOW DATABASES, DELETE, INSERT, UPDATE, LOCK TABLES.
Tout autre privilège accordé est inutile et pourrait conduire à des problèmes de sécurité.

4.2. Configuration HTTPS

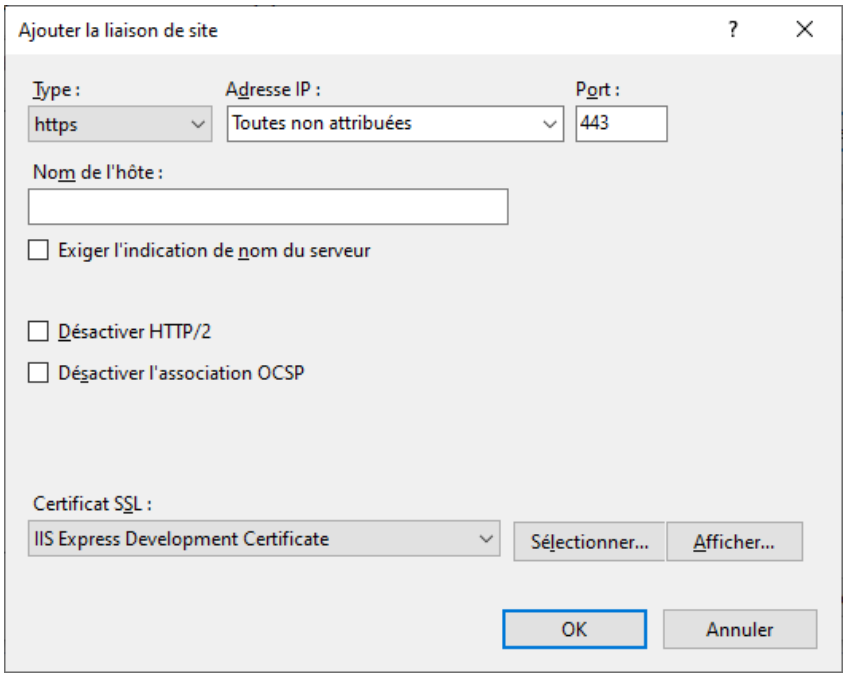
Aucune configuration particulière n'est requise, il suffit de créer les liaisons pour le site dans IIS. Pour cela, aller dans la configuration du site, puis cliquer sur « Liaisons » dans le panneau d'actions



La fenêtre suivante présente les liaisons actuellement configurées. Cliquer sur « Ajouter »



Paramétrer les informations avec le type et le port.



4.3. Authentification

Pour utiliser les API, il est nécessaire de fournir une identité à chaque méthode de l'API (sauf pour les services de connexion). 2 méthodes existent pour cela :

- Utiliser le mécanisme d'authentification pour obtenir une clé unique (valable un certain délai ou jusqu'à l'appel à la méthode de déconnexion)
- Utiliser les clés d'API qui sont fixes et toujours valides

4.3.1. Processus d'identification

L'identification d'un utilisateur peut se faire selon plusieurs méthodes. Cette méthode est paramétrable dans le fichier de configuration App_Data\Settings\appsettings.config en modifiant la valeur du paramètre LoginType :

- **Internal** : les utilisateurs propres à iDCuvePro sont utilisés
- **Domain** : les utilisateurs du domaine sont utilisés pour l'authentification. iDCuvePro doit être paramétré pour utiliser le SSO avec les comptes Windows. De plus, le serveur hébergeant les API doit être intégré au domaine
- **Ldap** : un serveur LDAP est interrogé pour l'identification. Il est alors nécessaire de renseigner les paramètres complémentaires :
    - **LdapServer** : IP ou nom du serveur LDAP à utiliser
    - **LdapPath** : chemin dans l'arborescence LDAP
    - **LdapUseSsl** : indique si la connexion au serveur LDAP doit se faire over SSL ou non
    - **LdapAuthenticationType :** type d'identification utilisé par le serveur LDAP. Une des valeurs suivantes peut être utilisée :
        - 0 - Anonymous
        - 1 - Basic
        - 2 - Negotiate
        - 3 - NTLM
        - 4 - Digest
        - 5 - Sicily (negotiate MSN, DPA or NTLM - LDAPv2)
        - 6 - DPA
        - 7 - MSN
        - 8 - External
        - 9 – Kerberos
    - **LdapRetrieveUserInformations** : ce paramètre est optionnel et permet d'indiquer que l'on ne récupère pas certaines données auprès du serveur LDAP (permet de résoudre certains rares cas d'erreurs)
    - **LdapForceDomain** : si le serveur LDAP ne permet pas de récupérer le nom du domaine, ou s'il en gère plusieurs, le nom du domaine à utiliser peut être forcé par ce paramètre
- **Saml :** les utilisateurs se connectent au travers d'un portail gérant une connexion SAML. Dans ce cas, il faut utiliser le site Id.Auth.WebModule pour gérer la connexion SAML à proprement parlé.

Note : il est possible de spécifier plusieurs méthodes en les séparant par « , » (virgule). On va alors tenter la connexion avec chacune des méthodes, dans l'ordre spécifié, jusqu'à une réponse positive.

4.3.2. Stockage des tokens de connexion

Une fois l'identification faite, un token (jeton) de connexion est fourni, et celui-ci est stocké sur le serveur. Plusieurs méthodes de stockage sont possibles, et modifiable via le paramètre **LoginProviderType** dans le fichier de configuration App_Data\Settings\appsettings.config. Les valeurs

possibles sont les suivantes :

- **Simple** (valeur par défaut) : les données sont enregistrées directement dans l'espace mémoire de la webapp. Ce cas convient parfaitement aux déploiements mono instance.
- **Redis** : utilise un serveur Redis (base de données clé-valeur). L'adresse du serveur Redis se paramètre via **LoginProvider.Redis.Connection**[3]. Ce paramétrage permet une installation multi instances des API avec partage des informations de connexion (permettant une distribution de charge)

4.3.3. Clés d'API

4.3.3.1. Généralités

Les clés d'API fournissent une identité via une clé toujours valide. Le fichier de clé associe une clé à un utilisateur iDApports. Les clés sont à générer via l'outil présenté ci-dessous.
Par défaut, les clés sont recherchées dans le fichier App_Data\keys.idk de la webapp. Ceci peut être paramétré en modifiant le paramètre **apiKeys** dans le fichier App_Data\Settings\appsettings.config.

4.3.3.2. Format du fichier

Le fichier est au format JSON qui fournit un tableau d'objets, chacun ayant 4 entrées possibles :

- **Key** (obligatoire) pour la clé d'API. Ce sont généralement des GUID, avec l'impératif que toutes les clés doivent être différentes
- **UserName** (obligatoire) correspondant à l'utilisateur logiciel
- **TenantName** (optionnel) pour indiquer le nom du tenant. En absence de valeurs, la clé d'API sera associée au tenant par défaut
- **AuthorizedOrigins** (optionnel) pour indiquer une liste blanche d'hôtes pouvant utiliser cette API (toute connexion depuis un autre hôte avec cette clé sera impossible)
- **RestrictToRoutes** (optionnel) pour indiquer une liste blanche de routes utilisables par l'API (toutes les autres seront interdites)

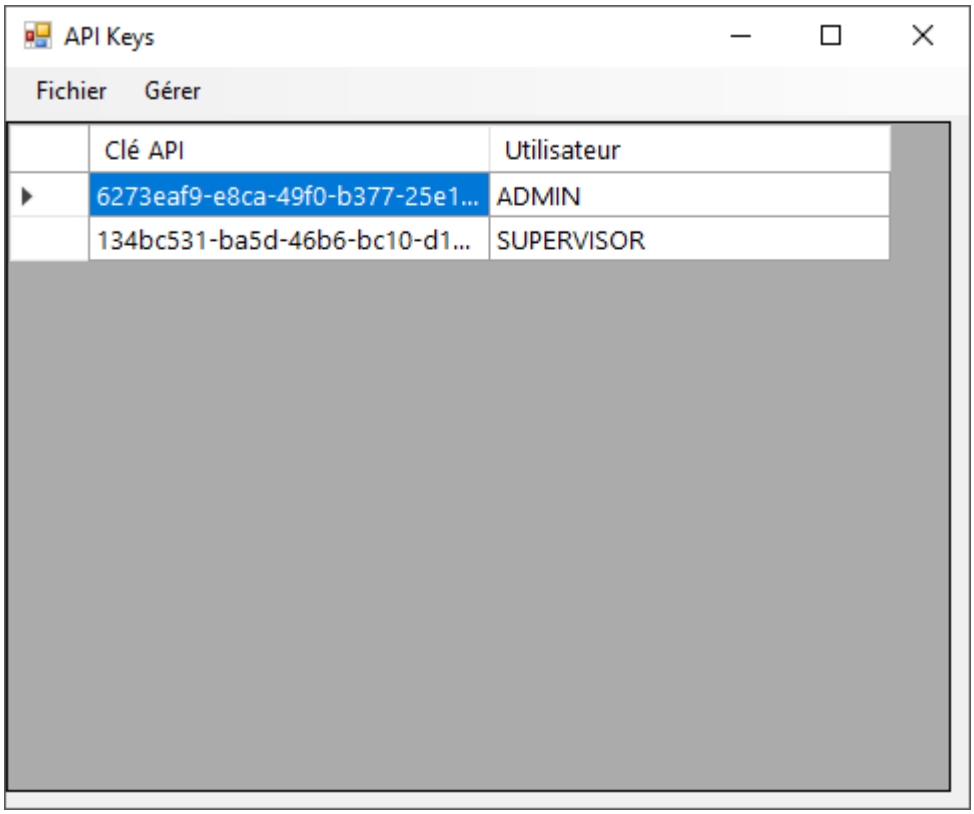
[3] Voir https://github.com/ServiceStack/ServiceStack.Redis pour le paramétrage des URL de connexion

Exemple

```
[
  {
    "Key": "6273eaf9-e8ca-49f0-b377-25e1cf618109",
    "UserName": "USER"
  },
  {
    "Key": "134bc531-ba5d-46b6-bc10-d12afe3849cd",
    "UserName": "API_USER",
    "AuthorizedHosts": [ "localhost", "srv-webapp1" ]
  },
  {
    "Key": "40dc93eb-d7da-43d0-b9cb-8e79f5e42229",
    "UserName": "API_USER",
    "TenantName": "Tenant1",
    "AuthorizedHosts": [ "srv-external" ],
    "RestrictToRoutes": [ "IoMovementServices", "WineTankServices" ]
  }
]
```

4.3.3.1. Outil de génération

Un outil permet de gérer les clés d'API (cet outil ne gère que le champ clé et le nom d'utilisateur, pour les autres champs pour lesquels il faut éditer manuellement le fichier) :



Le menu « Fichier » permet d'ouvrir un fichier et l'enregistrer. Le menu « Gérer » permet de créer ou

supprimer une clé (les clés sont générées, il faut ensuite renseigner l'utilisateur).
Attention l'outil devrait être exécuté en administrateur pour ouvrir et enregistrer le fichier de la webapp.

4.4. Autres configurations

4.4.1. Journalisation

La journalisation est configurable via le fichier nlog.config, et est réalisée via le framework NLog. Voir le site de NLog pour les détails de configuration : https://nlog-project.org/.
Par défaut, la configuration fournie va journaliser les données dans le répertoire App_Data\logs :

- Logs généraux : roulement sur 5 fichiers de 50 Mo maximum. Le fichier api_logs.log est le fichier actif, contenant les dernières entrées. Puis le fichier api_logs_1.log est l'archive la plus récente et api_logs_5.log l'archive la plus ancienne.
- Logs liés aux interfaces : roulement sur 2 fichiers de 50 Mo maximum. Le fichier interfaces_logs.log est le fichier actif, contenant les dernières entrées. Puis le fichier interfaces_logs_1.log est l'archive la plus récente et interfaces_logs_2.log l'archive la plus ancienne.
- Logs liés à la journalisation des requêtes (cf § 4.4.2) : roulement sur 4 fichiers de 50 Mo maximum. Le fichier http_logging.log est le fichier actif, contenant les dernières entrées. Puis le fichier http_logging_1.log est l'archive la plus récente et http_logging_4.log l'archive la plus ancienne.

4.4.2. Journalisation des requêtes

Il est possible de configurer la trace de tous les messages échangés entre le serveur et les clients. Cette trace permet essentiellement les mises au point.
Deux niveaux sont disponibles, le premier permet une trace simple de l'API appelé avec un second message avec le code retour. Ceci se fait au travers du paramètre **TraceMessages** du fichier App_Data\Settings\appsettings.config

Le 2nd niveau plus complet permet une trace complète des flux HTTP (requête / réponse). Cette trace devrait être réservée pour les mises au point car elle engendre un surcoût d'exécution. De plus, le contenu des messages peut être tracé (sur option) et donc des informations sensibles pourraient se retrouver dans les fichiers de logs. Cette journalisation se gère au travers des paramètres suivants :

- **HttpLogging.Enabled** : active / désactive globalement la journalisation. Ce paramètre peut valoir *true* (pour activer) ou *false* (pour désactiver, valeur par défaut)
- **HttpLogging.Fields** : permet de définir les éléments à journaliser. Cela peut être une ou plusieurs des valeurs suivantes (si plusieurs, il faut les séparer par une virgule). Les différentes valeurs possibles sont celles consultables dans la documentation : https://learn.microsoft.com/en-us/dotnet/api/microsoft.aspnetcore.httplogging.httploggingfields?view=aspnetcore-7.0

- **HttpLogging.RequestHeaders** : par défaut seuls les headers standards sont journalisés pour la requête (les autres sont indiqués avec une valeur masqués). Ce paramètre permet d'indiquer les headers supplémentaires pour lesquels on veut journaliser la valeur. Pour indiquer plusieurs paramètres, il faut les séparer par une virgule
- **HttpLogging.ResponseHeaders** : par défaut seuls les headers standards sont journalisés pour la réponse (les autres sont indiqués avec une valeur masqués). Ce paramètre permet d'indiquer les headers supplémentaires pour lesquels on veut journaliser la valeur. Pour indiquer plusieurs paramètres, il faut les séparer par une virgule
- ***HttpLogging.RequestBodyLimit :*** permet d'indiquer la taille limite (en octets) de la journalisation du corps de la requête (par défaut 32 ko)
- ***HttpLogging.ResponseBodyLimit :*** permet d'indiquer la taille limite (en octets) de la journalisation du corps de la réponse (par défaut 32 ko)
- ***HttpLogging.TextMediaTypes*** : permet d'indiquer les types de réponse texte à journaliser (par défaut les contenus json sont inclus)
- **HttpLogging.BinaryMediaTypes** : permet d'indiquer les types de réponse binaire à journaliser (par défaut aucun n'est journalisé)

4.4.3. Sécurité

Par défaut, les API sont livrées pour fonctionner en https (en priorité) et http. Si les 2 endpoints sont configurés, une redirection https est automatiquement faite, et des headers http sont ajoutés dans les échanges pour améliorer la sécurité.
Pour désactiver ces comportements (en cas d'utilisation http seule où les headers bloquent l'utilisation), il est possible de spécifier les paramètres suivants :

- **RequiredHttps** (par défaut true) : permet d'activer ou désactiver (valeur false) l'utilisation privilégiée du protocole HTTPS ou l'utilisation de HSTS
- **EnforcedSecurity** (par défaut true) : permet d'activer ou désactiver (valeur false) l'ajout des headers http liés à l'utilisation sécurisée des API[4]

4.4.4. Gestion des caches

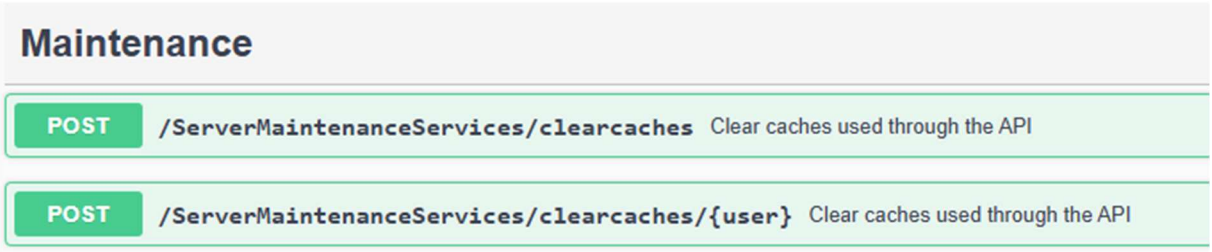
Un certain nombre de données peuvent être mises en cache pour de meilleures performances. Par défaut, un cache mémoire, local à l'instance IIS est utilisé. Il est possible d'utiliser un cache L1+L2 en spécifiant le type de cache dans le fichier de configuration App_Data\Settings\appsettings.config avec le paramètre **CacheType** :

- **InMemory** (valeur par défaut) : le cache mémoire est utilisé
- **Redis** : un cache Redis est utilisé en plus du cache mémoire. L'adresse du serveur se paramètre via le paramètre **Cache.Redis.Connection**

[4] Les headers suivants sont positionnés lors que le paramètre « EnforcedSecurity » est activé : "referrer-policy" : "strict-origin-when-cross-origin", "x-content-type-options" : "nosniff", "x-frame-options" : "DENY", "x-xss-protection" : "1; mode=block", "Content-Security-Policy" : "default-src https: "

En complément, et dans le cas de mise en cluster, il est possible d'utiliser le cache redis pour avoir un backplane et assurer la synchronisation des caches des différentes instances. Pour cela, il faut positionner le paramètre **CacheBackplane** à la valeur true.

2 API permettent de vider les caches si cela est nécessaire :



La 1ère permet de vider l'intégralité des caches, la 2nde permet de vider uniquement les caches de l'utilisateur uniquement.

4.5. Configuration Cross-Domain

Dans le cas où les API sont utilisées par une application Web (webapp), il faut modifier le fichier App_Data\Settings\services_deployment.xml pour ajouter les adresses d'accès du site API dans la partie « authorizedHosts ».
La configuration peut être faite au niveau global (valable pour toutes les configurations) en modifiant les données dans le nœud deployment/crossDomainConfiguration/authorizedHosts et/ou spécialisée par liaison en modifiant le paramétrage dans les nœuds endpoint.
Ceci n'est pas utile lorsque les API ne sont utilisées que par des applications tierces non web.

4.6. Gestion du multi tenant

4.6.1. Définition des tenants

Les tenants sont définis via les chaînes de connexion (fichier App_Settings\connections.config) par une convention de nommage des différentes connexions. Par exemple :

Exemple

```
<connectionStrings>
  <add name="Entities" connectionString="Server=localhost;Port=3308;Database=cderlcuv_champagne;…" />

  <add name="Tenant1_Entities" connectionString="Server=localhost;Port=3308;Database=cderlcuv_tenant1;…" />

  <add name="Tenant2_Entities" connectionString="Server=localhost;Port=3308;Database=cderlcuv_tenant2;…" />
</connectionStrings>
```

Dans cet exemple, 3 tenants sont définis : le tenant par défaut (optionnel), et 2 tenants nommés (Tenant1 et Tenant2). Il est impératif de définir les connexions pour les 3 bases pour chaque tenant, mais une base peut être partagée par plusieurs tenants (notamment la base d'utilisateurs)

4.6.2. Utilisation des tenants

La sélection du tenant se fait à la connexion et ne peut plus être changée par la suite (il faut une nouvelle connexion pour changer de tenant). Lors de l'appel au service /UserConnectionServices/signin, le header optionnel « x-idx-tenant-name » permet d'indiquer le tenant à utiliser. En l'absence de ce header, le tenant par défaut est utilisé.

Concernant les clés d'API, chaque clé est associée à un et un seul tenant. Pour indiquer le tenant associé à la clé d'API, il faut utiliser l'élément « TenantName » dans la structure json (en l'absence de ce paramètre, le tenant par défaut est associé à la clé d'API).

Exemple

```
[
  {
    "Key": "73598507-5f4f-4f61-9872-3b8ba14987c3",
    "UserName": "API_USER"
  },
  {
    "Key": "332ac87c-4b45-401a-96db-74cb51b4538b",
    "UserName": "ANOTHER_API_USER",
    "TenantName": "Tenant1"
  }
]
```

4.7. Health check

Un service de vérification de certains composants est disponible à l'URL <URLBase>/health. Ce service retourne un code 200 si tout est dans un bon état ou un état acceptable. Par contre, il retourne un code 503 en cas de défaillance d'un élément. Le corps du message indique le détail des différentes vérifications. Ce détail peut être contrôlé par le paramètre **DetailedHealthChecksReport** (true par défaut) pour activer / désactiver le rapport détaillé.
Par défaut, ce service n'est accessible qu'en local, il est possible d'indiquer un ou plusieurs hôtes pouvant y accéder avec le paramètre **HealthCheckRequiredHosts**. Si plusieurs hôtes sont spécifiés, il faut les séparer par une virgule ou un point-virgule. Si ce paramètre n'est pas spécifié ou avec une valeur vide, le service n'est accessible qu'en local. Enfin, il est possible d'utiliser des caractères génériques (exemple : *.domain.com, *:5000, *, …)

Dans le cas où plusieurs tenants sont définis, le service health check va vérifier le tenant par défaut. Pour spécifier un tenant particulier, il faut ajouter « ?tenant=<nom_du_tenant> » à la fin de l'URL.

Il existe également une API /HealthCheckServices conservée pour compatibilité (le tenant vérifié est celui associé à la clé d'API utilisée).

5. DOCUMENTATIONS

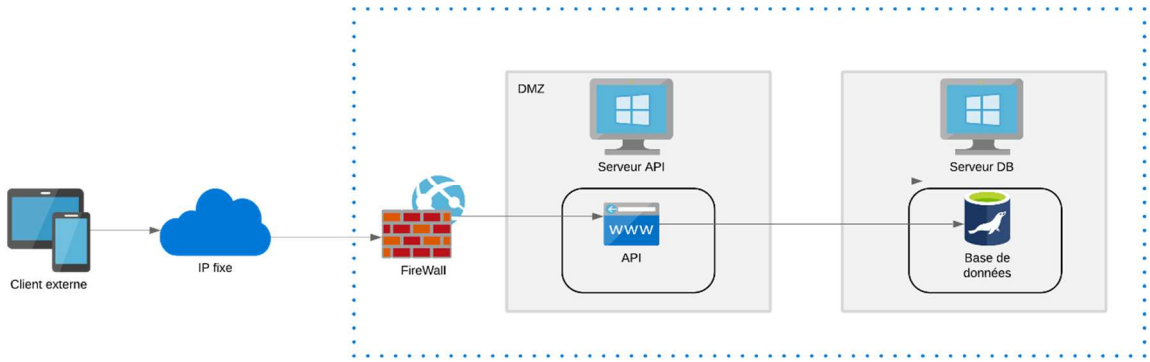
La documentation swagger est intégrée aux API et accessible à l'URL <UrlBase>/api-docs. Il est possible

d'accéder directement au json à l'URL <UrlBase>/api-docs/v1/swagger.json.
Cette documentation suit la spécification OpenAPI 3.0

6. EXPOSITION

Dans le cas d'une utilisation par une ressource externe au réseau local, il est nécessaire d'exposer les API. Cette partie décrit les conditions d'accès les prérequis techniques complémentaires à l'installation décrite précédemment.
Dans le cadre d'une utilisation externe, et donc une exposition sur internet, une utilisation du protocole HTTPS est indispensable (voir § **Erreur ! Source du renvoi introuvable.** pour la configuration).

6.1. Schéma

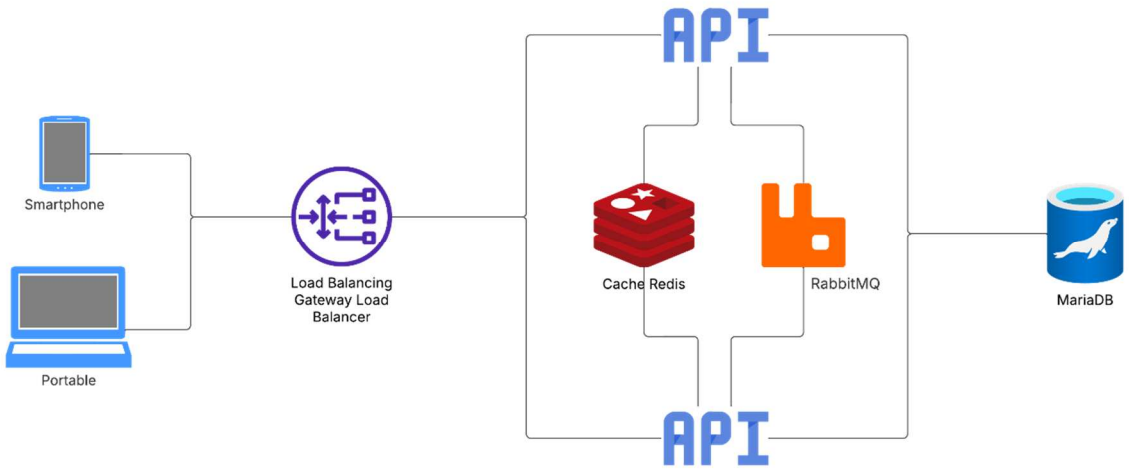


Seul le serveur API doit être exposé. Le port utilisé pour la liaison HTTPS (TCP/443 par défaut) doit être ouvert et accessible en externe.

6.2. Pré-requis réseau

Afin d'obtenir les performance optimales, l'infrastructure réseaux et télécom doit disposer d'une bande passante et d'une stabilité suffisante

6.3. Cluster



Pour une mise à l'échelle, l'utilisation d'un cluster est possible. Il faut en complément un cache Redis et une instance RabbitMQ si des fonctionnalités utilisant SignalR sont utilisées.
Les configurations nécessaires (dans le fichier appsettings.config) sont les suivantes :

- Utiliser un cache Redis (voir § 4.4.4) avec l'activation du backplane
- Utiliser RabbitMQ comme intermédiaire pour les communications bidirectionnelles (voir § **Erreur ! Source du renvoi introuvable.**)
- Utiliser Redis comme stockage des tokens de connexion (voir § **Erreur ! Source du renvoi introuvable.**)

En complément, les configurations de connexion et de clés d'API devront être réalisés de manière identique sur chaque instance.

Au niveau du cluster, si des fonctionnalités utilisant SignalR sont utilisée, il faudra veiller à :

- Mettre en place l'affinité de session
- S'assurer que le proxy gère également les WebSockets

7. DIAGNOSTICS ET RESOLUTIONS D'ERREUR

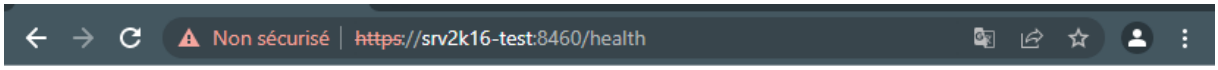
7.1. Tests de fonctionnement

7.1.1. Notes préalables

Les tests devraient être réalisés dans un premier temps sur le serveur hébergeant les API, afin de limiter les impacts des firewalls et autres paramètres réseaux pouvant intervenir. Une fois les tests réalisés en local, les mêmes tests peuvent être répétés sur un autre poste.
Les tests qui suivent se font au travers d'un navigateur. Il est conseillé d'utiliser Chrome, Firefox ou MS Edge. Internet Explorer ne peut pas être utilisé (quelle que soit la version).

7.1.2. Vérification de la santé API

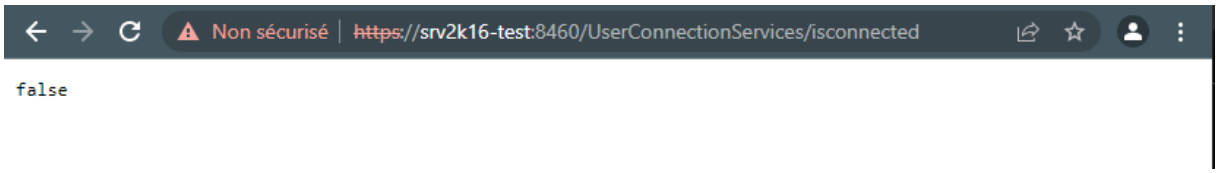
Le service de vérification peut être utilisé (7.1.2) avec l'adresse <UrlBase>/health/. Cela permet de vérifier le bon démarrage des API et les connexions aux bases de données.
Le navigateur devrait proposer le message suivant (dans certains cas, le navigateur peut proposer de télécharger un fichier dont le contenu sera celui indiqué).
La connexion à la base est vérifiée. Si la base a un statut « Degraded », il faut vérifier les chaînes de connexion et le bon accès du serveur API vers le serveur de base de données.




```
{
  "status": "Healthy",
  "results": {
    "cuverie": {
      "status": "Healthy",
      "description": "Reports status for entity context DalCuverieWrapper",
      "data": {}
    },
    "memory_check": {
      "status": "Healthy",
      "description": "Reports degraded status if allocated bytes >= 1073741824 bytes.",
      "data": {
        "AllocatedBytes": 14431832,
        "Gen0Collections": 0,
        "Gen1Collections": 0,
        "Gen2Collections": 0
      }
    }
  }
}
```

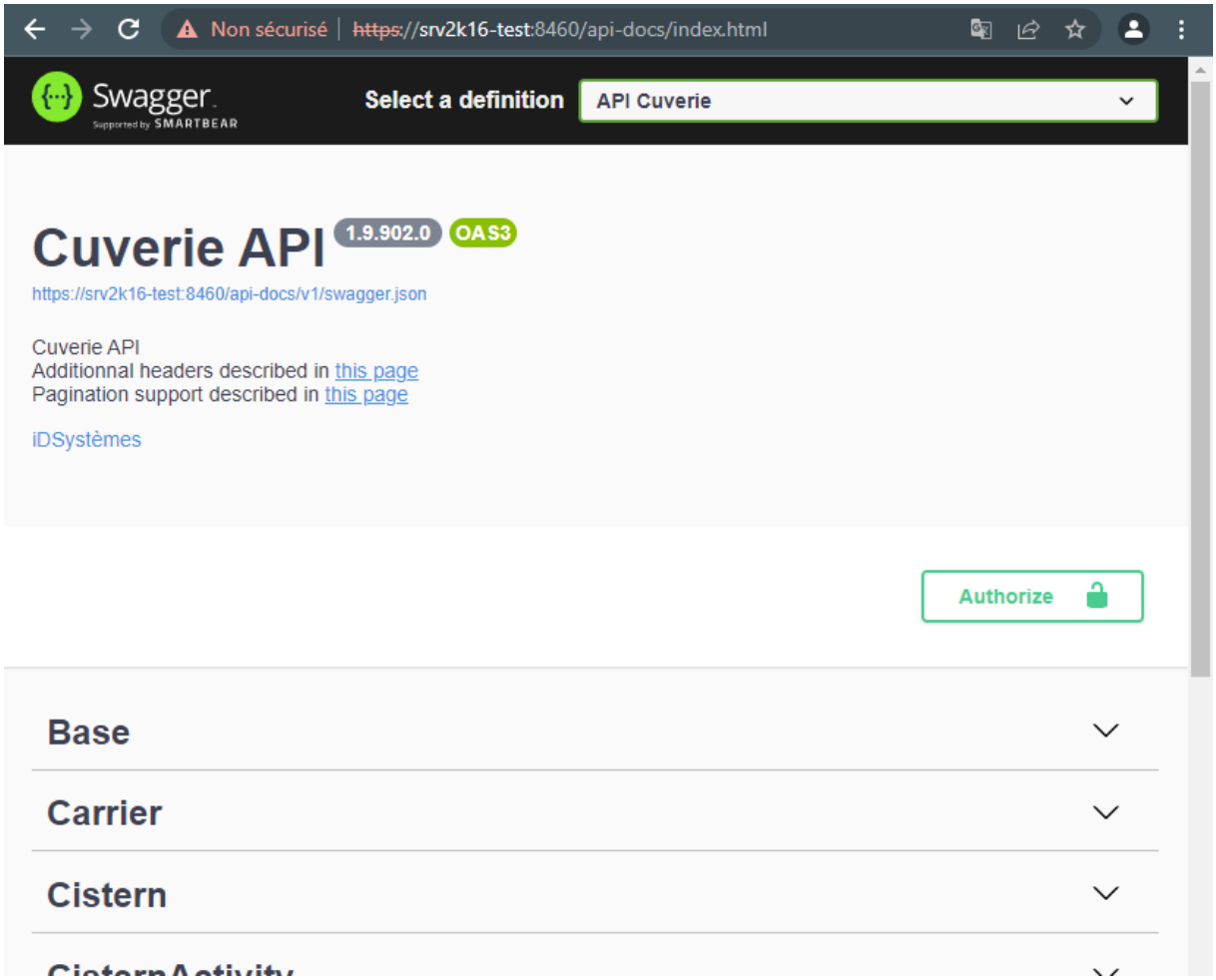

7.1.3. Appels API

L'API de test de connexion peut être utilisée comme premier test pour vérifier que les API répondent correctement. Cette API se situe à l'adresse <UrlBase>/UserConnectionServices/isconnected.
Le navigateur doit afficher « false »



7.1.4. Documentation swagger

L'accès à la page swagger se fait via l'url <UrlBase>/api-docs. Cela doit aboutir à l'affichage de la page swagger. Le numéro de version de l'API est indiqué dans la partie haute

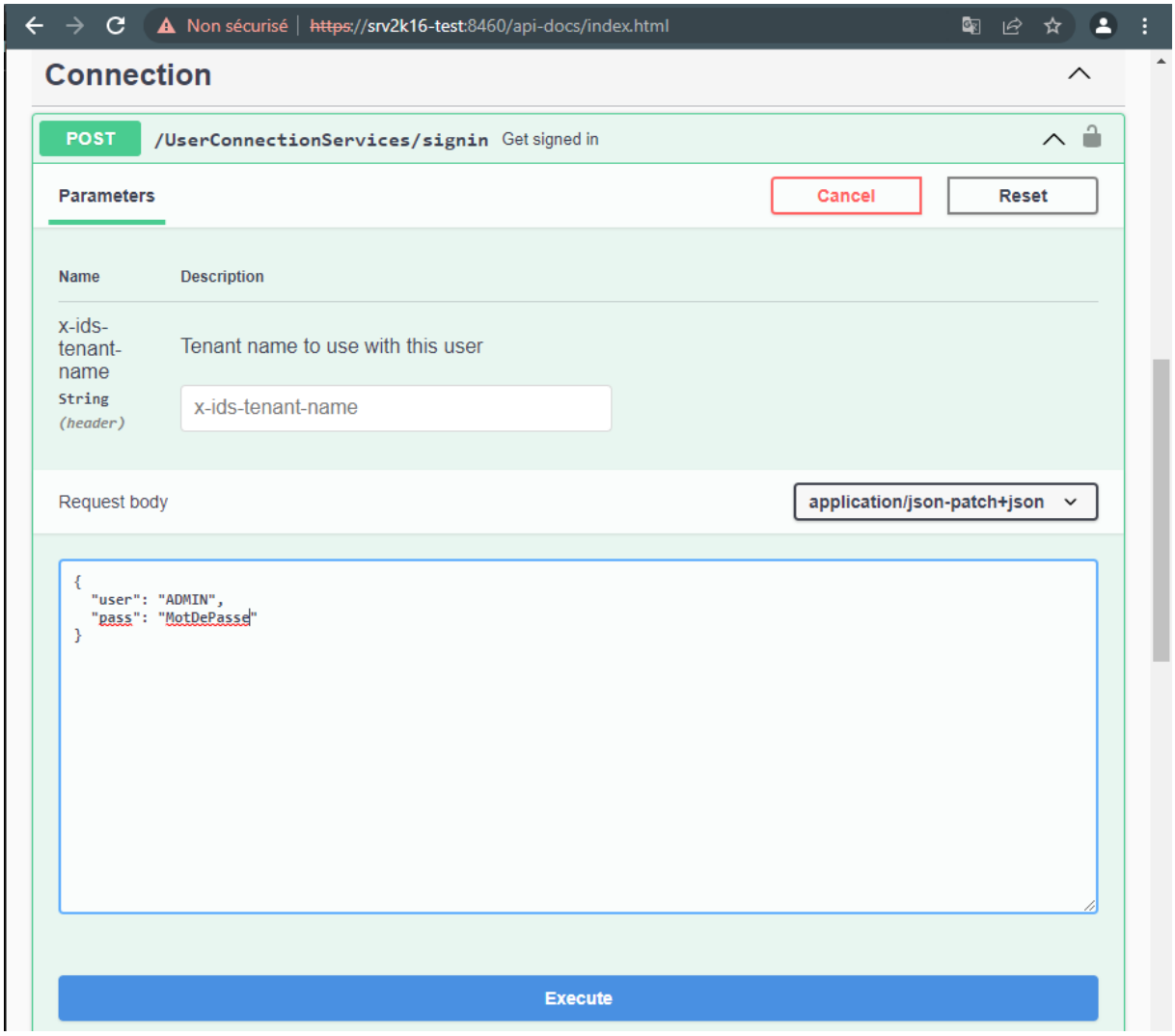


7.1.5. Appels d'API

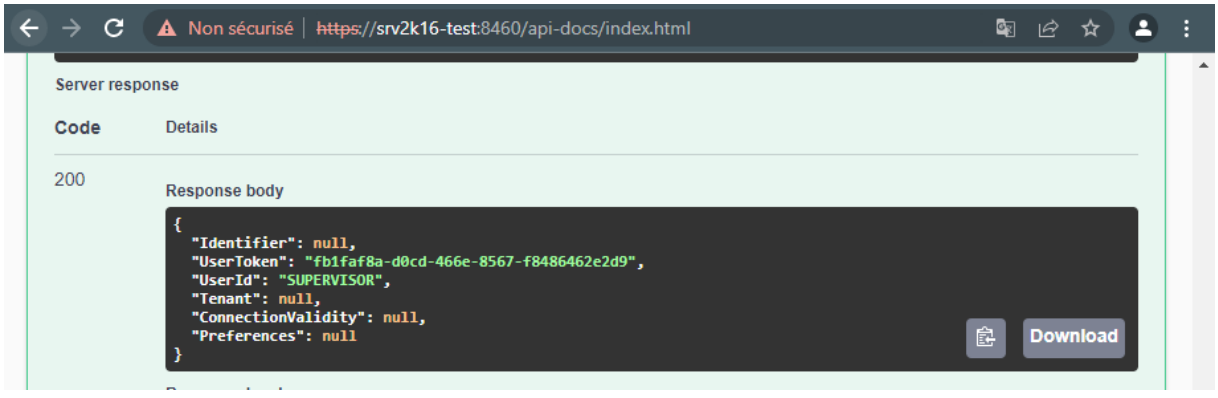
Swagger permet de réaliser des appels API. Il est préconisé de réaliser des appels via Swagger UI pour valider le bon fonctionnement. Un enchaînement conseillé est le suivant :

Connexion

Dans la section *Connection*, utiliser le service de connexion UserConnectionServices/signin.
Cliquer sur « Try it out » et remplir le nom d'utilisateur et mot de passe dans la requête



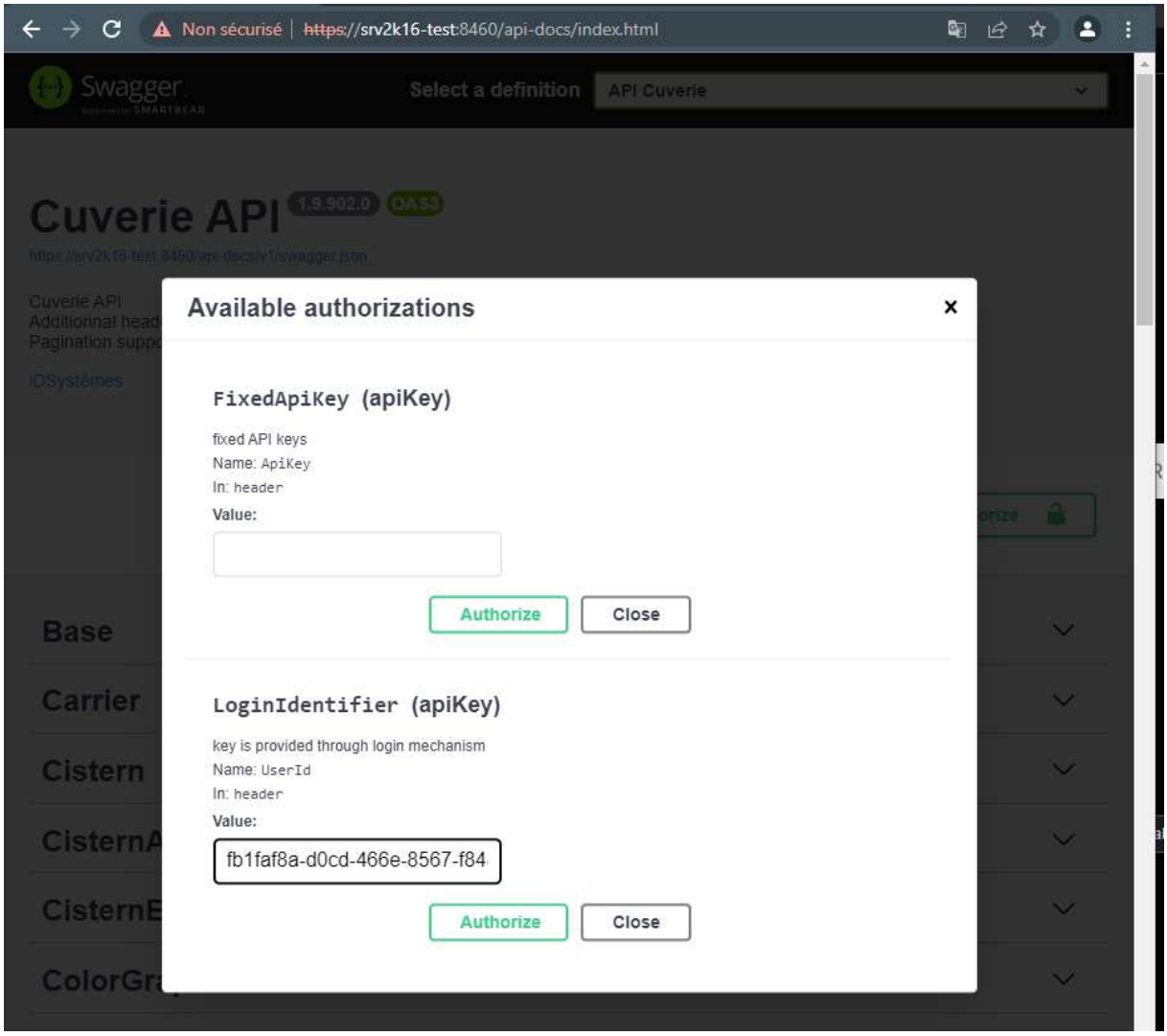
Après avoir cliqué sur Execute, la partie réponse doit présenter une réponse 200 avec un message similaire à celui-ci-dessous :



Renseignement des informations d'authentification

Pour appeler d'autres API, il est nécessaire d'indiquer les éléments d'authentification. Pour cela, il faut copier la valeur de « UserToken » dans la réponse précédente, puis cliquer sur le bouton « Authorize » présent tout en haut de la page.

La valeur copiée est à mettre dans le champ LoginIdentifier



Il suffit ensuite de cliquer sur Authorize puis Close

Appel d'API de consultations

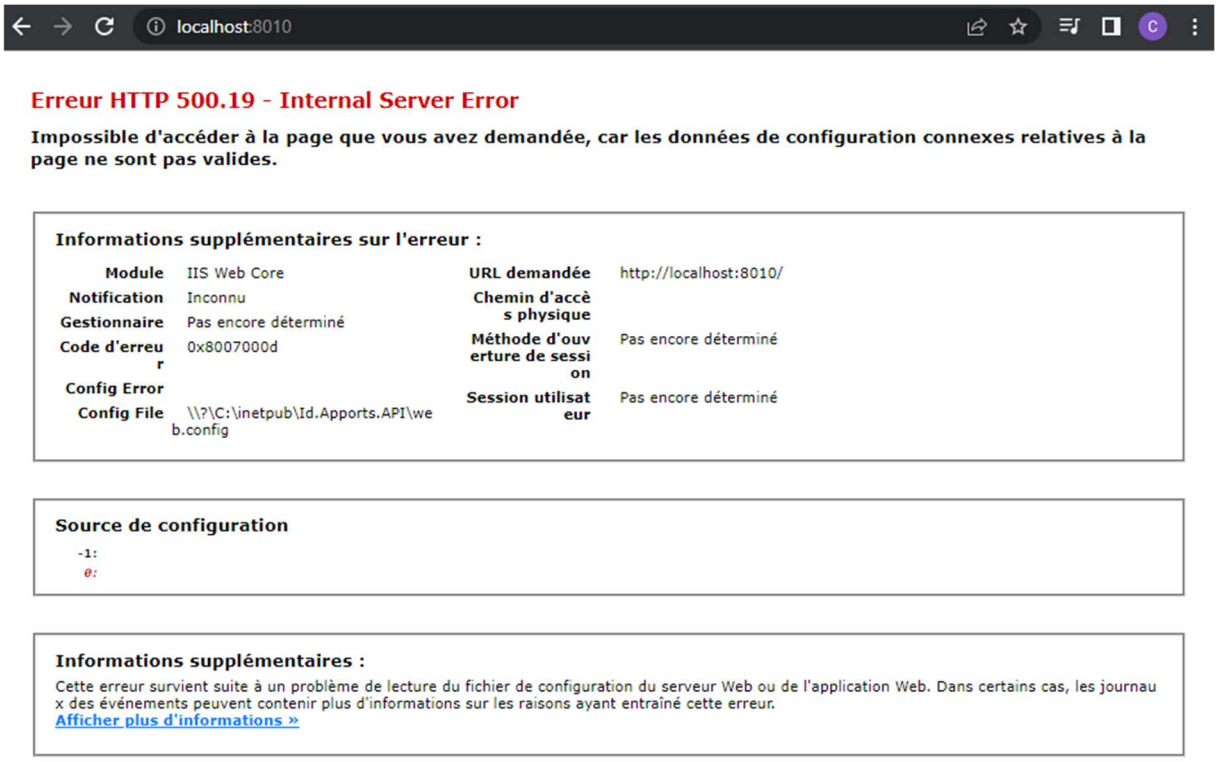
Une fois l'authentification réalisée, il est possible de faire des appels API via swagger. Les méthodes GET peuvent être utilisées pour vérifier la bonne réponse des API. Par exemple, dans la section *Cistern* avec le service /CisternServices/cistern qui retourne la liste de toutes les citernes de la base iDCuvePro.

7.2. Résolutions de problèmes courants

7.3. Résolutions de problèmes courants

7.3.1. Erreur 500.19 / IIS Web Core

Symptômes : l'accès aux API indique une erreur 500.19 sur le module IIS Web Core



Causes possibles :

- Le runtime .Net 9 n'est pas installé
- Le runtime ASP.Net core 9 n'est pas installé
- Le module IIS ASP.Net core n'est pas installé

Diagnostics et solutions

Runtime .Net 9.0

Exécuter la ligne de commande

`dotnet --list-runtimes`

La sortie devrait être semblable à celle présentée ci-dessous (avec plus ou moins de lignes et de versions). Il doit y avoir des lignes Microsoft.AspNetCore.App 9.0 et Microsoft.NETCore.App 9.0.

Si ces lignes ne sont pas présentes, il faut installer le runtime .Net 9.0

```
Microsoft.AspNetCore.App 5.0.17 [C:\Program Files\dotnet\shared\Microsoft.AspNetCore.App]
Microsoft.AspNetCore.App 6.0.36 [C:\Program Files\dotnet\shared\Microsoft.AspNetCore.App]
Microsoft.AspNetCore.App 7.0.20 [C:\Program Files\dotnet\shared\Microsoft.AspNetCore.App]
Microsoft.AspNetCore.App 8.0.11 [C:\Program Files\dotnet\shared\Microsoft.AspNetCore.App]
Microsoft.AspNetCore.App 8.0.13 [C:\Program Files\dotnet\shared\Microsoft.AspNetCore.App]
Microsoft.AspNetCore.App 9.0.0 [C:\Program Files\dotnet\shared\Microsoft.AspNetCore.App]
Microsoft.AspNetCore.App 9.0.2 [C:\Program Files\dotnet\shared\Microsoft.AspNetCore.App]
Microsoft.NETCore.App 5.0.17 [C:\Program Files\dotnet\shared\Microsoft.NETCore.App]
Microsoft.NETCore.App 6.0.36 [C:\Program Files\dotnet\shared\Microsoft.NETCore.App]
Microsoft.NETCore.App 7.0.20 [C:\Program Files\dotnet\shared\Microsoft.NETCore.App]
Microsoft.NETCore.App 8.0.11 [C:\Program Files\dotnet\shared\Microsoft.NETCore.App]
Microsoft.NETCore.App 8.0.13 [C:\Program Files\dotnet\shared\Microsoft.NETCore.App]
Microsoft.NETCore.App 9.0.0 [C:\Program Files\dotnet\shared\Microsoft.NETCore.App]
Microsoft.NETCore.App 9.0.2 [C:\Program Files\dotnet\shared\Microsoft.NETCore.App]
Microsoft.WindowsDesktop.App 5.0.17 [C:\Program Files\dotnet\shared\Microsoft.WindowsDesktop.App]
Microsoft.WindowsDesktop.App 6.0.36 [C:\Program Files\dotnet\shared\Microsoft.WindowsDesktop.App]
Microsoft.WindowsDesktop.App 7.0.20 [C:\Program Files\dotnet\shared\Microsoft.WindowsDesktop.App]
Microsoft.WindowsDesktop.App 8.0.11 [C:\Program Files\dotnet\shared\Microsoft.WindowsDesktop.App]
Microsoft.WindowsDesktop.App 8.0.13 [C:\Program Files\dotnet\shared\Microsoft.WindowsDesktop.App]
Microsoft.WindowsDesktop.App 9.0.0 [C:\Program Files\dotnet\shared\Microsoft.WindowsDesktop.App]
Microsoft.WindowsDesktop.App 9.0.2 [C:\Program Files\dotnet\shared\Microsoft.WindowsDesktop.App]
```

Module IIS

Ouvrir l'explorateur et aller dans le répertoire %ProgramFiles%\IIS. Il doit y avoir un répertoire « Asp.Net Core Module\V2 » contenant le fichier « aspnetcorev2.dll » et un ou plusieurs sous répertoires
Si ces éléments sont absents, il faut réinstaller le bundle .Net 9 (§ 2.4)