RAPPORT D'ÉVALUATION DES PROTOCOLES D'ÉCHANGE DE DONNÉES FÉDÉRÉ

Projet SMILE, Chantier SEN1 – phase POC

Tableau de suivi des versions

Version	1	1.1		
Auteurs	Jaxom, Gautier	Ruby		
Date	12/03/19	04/04/19		
Commentaires	Version initiale	forme		

Contact : pro _@_ consometers.org

TABLE DES MATIÈRES

Introduction

Ce document présente la première partie du travail réalisé dans le cadre d'une preuve de concept (proof of concept - POC) pour le projet SEN1 – « Sensibilisation aux consommations d'énergie ». L'objectif final de ce projet est de faciliter les échanges de données entre acteurs de l'affichage des consommations.

Pour réaliser les objectifs de SEN1, et en accord avec le Demandeur, nous avons souhaité mettre en place une **fédération technique**, qui permet un échange de données entre serveurs pairs, sans lien de subordination entre l'un et l'autre. Ce paradigme s'appuie sur une spécification publique du protocole d'échange, simplifiant ainsi l'entrée de nouveaux acteurs dans le dispositif d'échange. De plus, l'implémentation logicielle correspondante doit être codée sous licence libre, permettant donc sa réutilisation directe par d'autres acteurs, publics comme privés, souhaitant s'intégrer à la fédération.

Ce document expose la démarche suivie pour retenir, parmi l'ensemble des solutions logicielles libres disponibles, la solution qui implémente le protocole d'échange de données fédéré qui conviendra le mieux à notre POC.

Conformément à la charte « PrestaLibre », nous avons effectué ce travail de manière ouverte et documentée, en lien avec le collectif « Consometers ». Les documents sont disponibles sur le site https://www.consometers.org et diffusés selon la licence EUPL v1.2 permettant une libre réutilisation.

Définitions

Le(s) Prestataire(s) est(sont) une(des) Société de Services en Informatique. A ce titre, les Demandeurs mettent à disposition du Commanditaire le résultat d'un travail. Ce travail est cadré par un planning prévisionnel, le détail des tâches, son chiffrage est justifié par des devis et des factures. Ce travail est réalisé par les Prestataires, appuyés par le collectif bénévole des Consometers

Demandeur : ALOEN et l'ALEC du Pays de rennes

Commanditaire : Région Bretagne

Collectif des Consometers : Collectif bénévole qui promeut l'utilisation des logiciels libres autour des besoins d'affichage et de maîtrise de la consommation d'énergie, et l'acculturation sur les logiciels libres, voir : https://cloud.consometers.org/index.php/s/3665sZsz5xz8FS3

Documentation : Manuels techniques et d'information afférents aux résultats de Prestations et mis à disposition du Demandeur par les Prestataires.

Internet : réseau de réseaux permettant l'échange d'informations à partir d'un protocole dénommé IP. Les données sont acheminées à travers des réseaux de natures différentes qui sont capables de transmettre les messages selon cette norme technique. Chaque élément de ce réseau appartient à des organismes privés ou publics qui les exploitent en coopération sans nécessairement impliquer une obligation bilatérale de qualité.

DÉMARCHE DE SÉLECTION

Après la définition des critères et la constitution d'une matrice de choix pondérées, 15 solutions ont été évaluées sur la base de leur documentation disponible publiquement. Enfin, les deux protocoles les mieux notés ont été testés concrètement, pour retenir celui le plus apte à la réalisation d'une preuve de concept rapide.

CRITÈRES D'ÉVALUATION

L'objectif de cette tâche était de définir un ensemble de critères pouvant être évalués objectivement, avec par conséquent une définition et un score (de 0 à 5) explicites. La pondération a été déterminée subjectivement par l'ensemble des auteurs, puis cumulée.

Les critères d'évaluation des protocoles ont été organisés en trois catégories : les critères intrinsèques à la solution, ceux répondant aux besoins à court-terme pour la phase de POC, et ceux répondant aux besoins à long-terme pour le projet SEN1.

Critères internes

Ces critères se rapportent à la nature de la solution, sur le fond comme sur la forme.

Communauté

- Activité du code : le code logiciel, disponible publiquement, est fréquemment mis à jour (correctifs, ajouts de fonctionnalité). Cela assure de la vivacité du projet (suivi, implications, durée dans le temps).
- Diffusion d'usage : les utilisateurs finaux de la solution sont nombreux et variés, ce qui souligne la pertinence de la solution et sa versatilité.
- Roadmap : présence d'une planification des fonctions et correctifs à venir. Ce critère jauge la maturité d'un projet et son organisation.

- Taille : le nombre de développeurs et de commentateurs actifs, sur le dépôt du logiciel et dans ses outils communautaires, assure de la vivacité et de la résilience du projet.

Implémentations logicielles

- Disponibilité : le fait d'avoir une mise en œuvre, codée et mise en production, garantit que le standard est bien défini.
- Langages : l'existence d'implémentations dans différents langages informatiques facilite la prise en main et le codage ultérieur.
- Licence : le code déjà existant doit être disponible sous une licence libre, condition indispensable à sa réutilisation dans de nouveaux contextes (y compris commerciaux).

Qualité du code existant

- Commentaires : un code bien commenté améliore la facilité à retravailler sur le code du projet, et à en étendre les fonctionnalités.
- Documentation : pour les mêmes raisons, une documentation d'usage, d'installation et d'administration est un point fort.
- Variables : au sein du code, des noms de variables clairs et explicites sont un avantage pour reprendre le code par la suite.

Dépendance à un acteur centralisé

- Critère unique, qui assure l'indépendance du projet, et donc la possibilité de le répliquer sans être lié à un acteur tiers qui pourrait exercer un contrôle sur nos activités.

Standard

- Évalue la bonne définition du standard, selon les bonnes pratiques observées dans les communautés informatiques (origine, transparence...).

Structure porteuse du protocole

- La nature de la structure (association, fondation, à but non-lucratif, entreprise) offre plus ou moins de garanties sur l'orientation du travail collectif de développement, qui peut ainsi se rapprocher d'un Commun.

Critères court-terme

Ces critères évaluent la pertinence de la solution pour la réalisation d'une « preuve de concept » rapide et opérationnelle à court-terme, à petite échelle.

Architecture de la solution

- Sa complexité informatique (divers composants, liens internes) est un obstacle à l'adoption par une diversité d'acteurs, ainsi qu'à son empreinte énergétique.

Authentification

- L'intégration d'une authentification permet de sécuriser intrinsèquement l'accès aux données.
- L'utilisation d'un standard, bonne pratique indispensable dans ce domaine, souligne la prise en compte de la sécurité par le protocole.

Autorisation

- L'intégration d'une gestion des autorisations permet d'aller plus rapidement au stade opérationnel, avec une distinction des rôles (lecteur, auteur, administrateur des données...).

Cas d'usage

- Si l'utilisation du protocole prévue en conception est proche de l'objectif de SEN1, cela permet d'aller plus vite vers la mise en place du POC (échanges de données au sens large, pas limité à des messages courts par exemple).

Format et Type de données gérées

- Ce critère évalue les limitations imposées par le protocole sur les données que l'on peut échanger.

Gestion des échanges - Annuaire

- La présence d'un annuaire, dans un contexte fédéré, apporte une grande simplicité d'usage.

Compatible RGPD

- Ce critère relève d'une approche non-experte, qualitative et déclarative à ce stade. Cela sera recoupé avec une autre tâche menée en parallèle dans le projet de POC. Le jugement se base essentiellement sur la documentation des solutions au sujet du RGPD.

Transmission

- Chiffrement : ce critère indispensable souligne la prise en compte de la sécurité de la transmission des données par la solution.
- Modes d'envoi : la prise en charge de plusieurs modes d'envoi des données apporte une simplicité d'usage et d'implémentation, qui favorise par la suite une diversité d'acteurs et de solutions.

Critères long-terme

Cette catégorie de critères s'attache à la vision d'ensemble d'une fédération, avec le passage à l'échelle et la constitution d'un écosystème varié et robuste.

Architecture d'un nœud

- L'architecture d'un nœud permet de « passer à l'échelle » plus ou moins facilement. C'est-à-dire allouer des ressources informatiques nécessaires afin de traiter la charge de travail qui augmente à proportion des utilisateur que ce nœud héberge. Pouvoir passer à l'échelle sur

un seul nœud de fédération permet d'éviter une dilution des identités des serveurs. Autrement le passage à l'échelle serait réalisé par l'adjonction directe de nœuds dans la fédération.

Format des données

- Si le protocole ne nécessite qu'une surcouche légère sur les données à échanger, cela se traduit par des économies de bande passante et d'énergie, ce qui se rapproche de l'objectif du projet SEN1.

Traçabilité non répudiable des autorisations et des transferts de données

- Cette traçabilité, par de systèmes tels que les blockchains, permet de remplacer un tiers de confiance, extérieur au système, par l'infrastructure elle-même. Cela permet par exemple de démontrer de manière certaine et non opposable qu'une personne a donné son consentement pour le traitement de ses données.

Pondération et matrice de choix

L'ensemble des 23 critères a été pondéré par l'attribution de points, selon la méthode suivante : trois professionnels aux compétences complémentaires ont chacun réparti 56 points, avec un minimum de 1 par critère. Les trois scores ont ensuite été additionnés pour obtenir la pondération finale du critère. Le poids minimum est donc de 3, et le critère le plus important atteint le poids de 11.

Un certain nombre de ces critères ont été considérés comme éliminatoires (*par ex. : licence libre, pluralité des développeurs*...). Ils contiennent la valeur '1' dans la colonne « éliminatoire » de la matrice de choix (voir ANNEXE 3 : Matrice d'évaluation finale). Ainsi, si l'évaluation sur un seul de ces critères est à 0, le protocole est éliminé. Cela a permis un premier tri rapide parmi les solutions identifiées.

Protocoles évalués

Une phase de recherche sur différents moteurs de recherche, dans divers corpus et via des échanges avec des experts du secteur, nous ont permis d'identifier les solutions pertinentes à ce jour. Celles-ci devaient à la fois gérer les données elles-mêmes, leur échange, et la fédération entre plusieurs serveurs indépendants. Pour référence, les termes de recherche utilisés sont mentionnés en ANNEXE 1 : Termes de recherche.

Les solutions identifiées et évaluées sont au nombre de 15. Leurs références (sites web officiels, code source public et documentation) sont présentées en ANNEXE 2 : Références des protocoles évalués.

RÉSULTAT D'ÉVALUATION

Chaque protocole a été évalué par un des auteurs sur la base des critères objectifs préalablement définis. Les trois solutions ayant eu la meilleure évaluation (XMPP, Matrix et ActivityPub) ont fait l'objet d'une seconde évaluation indépendante, puis d'une harmonisation. Par ailleurs, la majorité des solutions éliminées ont été passées en revue par au moins deux auteurs afin d'éviter le risque d'une élimination non justifiée.

Ces pratiques nous ont permis d'éliminer les biais subjectifs et les faux négatifs. L'ensemble des évaluations et des résultats est disponible dans la matrice de choix, placée en ANNEXE 3 : Matrice d'évaluation finale.

Les deux lauréats issus de cette phase sont **XMPP** et **Matrix**, avec des scores respectifs de 681 et 634 (sur un maximum théorique de 780 points).

RÉSULTAT DES TESTS

Infrastructure et protocole de test

Les deux protocoles ont été installés, en suivant leurs documentations respectives, sur plusieurs serveurs virtuels, ayant chacun une adresse IP publique. Puis les mécanismes d'échanges de données ont été testés en utilisant les codes informatiques préexistants (librairies) pour se connecter aux serveurs fédérés en tant que client. L'évaluation prend ainsi en compte la facilité d'installation de la solution (administration du système informatique) et sa facilité d'utilisation par un programme.

Les tests réalisés par des programmes simples ont été faits selon le scénario suivant :

- 2 programmes connectés en tant que clients, chacun en tant qu'un utilisateur d'un domaine de test (2 programmes clients, 2 comptes utilisateurs, 2 domaines, 2 serveurs)
- envoi d'un message du programme 1 vers le programme 2
- transit de l'information : programme 1 → serveur du domaine 1 → serveur du domaine 2 → programme 2

Test de Matrix

L'infrastructure à mettre en œuvre est assez complexe pour l'implémentation de référence : serveur Synapse, serveur posgresql, reverse proxy. La procédure suivie utilise les playbooks Ansible prévus à cet effet (ce qui nécessite un autre pan infrastructurel) et instancie des images docker pour lancer les composants de l'infrastructure (voir ANNEXE 4 : Procédure d'installation de Synapse

(matrix.org)). La documentation n'était pas toujours très claire, et le projet en v0.99 se trouve dans une phase de changement au niveau infrastructure (utilisation des champs DNS SRV vs .well-known sur le serveur HTTP), ce qui implique une double-configuration pour l'une des fonctionnalités (découverte du serveur lié à un domaine). Notons que plusieurs documentations sont disponibles et que les possibilités d'installation sont nombreuses. Notons également qu'il n'y a pas d'interface graphique de gestion du serveur et que les opérations de création d'utilisateur par exemple se font en ligne de commande (register_new_matrix_user).

A défaut de trouver une librairie Java satisfaisante, les tests ont été réalisés avec la librairie « Matrix SDK JS » (https://github.com/matrix-org/matrix-js-sdk) sur un environnement NodeJS. L'installation s'est faite rapidement depuis un paquet npm. Une classe façade est présente et permet d'utiliser simplement toutes les fonctionnalités (connexion, envoi et réception message). Cependant, cette librairie n'est pas aisément interfaçable avec l'outil BMHS, du fait d'un environnement de développement différent.

Test de XMPP

L'infrastructure à mettre en œuvre est simple pour une installation de base de l'implémentation de référence (serveur ejabberd seul). Globalement, il s'agit d'un unique programme à installer, à configurer et à lancer (voir : ANNEXE 5 : Procédure d'installation d'ejabberd (XMPP)). La documentation est assez claire. Le serveur ejabberd possède sa propre interface web d'administration, ce qui simplifie la création d'utilisateurs par un administrateur.

Les tests ont été effectués avec la librairie « Java Babbler » (https://sco0ter.bitbucket.io/babbler/). Cette librairie est une implémentation à jour car elle est compatible avec Java 8 et 9. L'intégration de cette librairie se fait aisément depuis un dépôt Maven. Cette librairie est facile d'utilisation car elle dispose d'une classe façade avec toutes les fonctionnalités (connexion, envoi et réception des messages). De nombreuses extensions du protocole XMPP sont également supportées. Le fait que cette librairie utilise le même environnement de développement que BMHS est également un point fort pour l'avancée du POC, gage de simplicité et d'efficacité.

Conclusion

A la suite des tests, il s'est avéré que le protocole XMPP était plus adapté à l'usage de ce POC (meilleure note dans la matrice de tests, facilité d'installation, présence de nombreuses librairies matures, facilité d'administration). C'est donc XMPP le protocole retenu, qui sera mis en œuvre dans la suite du POC-SEN1.

Nota Bene

Cependant, il faut noter que plusieurs protocoles identifiés dans la phase de recherche sont très intéressants, mais se trouvent encore en phase de recherche et développement. Ainsi, le résultat actuel de nos travaux ne présage pas de la situation dans un à deux ans, quand ces projets auront publié leurs premières implémentations prêtes pour mise en production et réellement utilisées. Un travail de veille est donc à poursuivre dans ce secteur, qui pourrait apporter de meilleures solutions aux défis soulevés par le projet SEN1. Certains projets sont prometteurs, par exemple : Ocean Protocol, mainframe, streamr. Notons également l'existence du projet européen DECODE (financement H2020), qui tente de résoudre peu ou prou les mêmes problématiques d'échanges de données.

A titre d'historisation, et afin de ne perdre aucune information accumulée lors de la recherche d'un protocole fédéré, nous proposons en ANNEXE 6 : Fichier de travail l'ensemble de nos notes de recherches.

ANNEXE 1 : TERMES DE RECHERCHE

La phase de recherche des solutions a utilisé les termes, moteurs et domaines de recherche suivants (avec mention de certains résultats *en italique*) :

- Liste des RFC, sur le site de l'IETF
- Qwant :
  - protocole architecture fédérée
- Duckduckgo :
  - federated exchange protocol -> *rabbitmq*
  - federated messenging -> *mastodon*
  - decentralised protocol -> *ocean protocol*
  - federated data sharing
  - (examples of) federated data exchange
  - how federated exchange/sharing works
  - ddbms standard
  - decentralised data exchange protocol -> *dock.io*
  - decentralised data exchange -> *bigchaindb*, *gxs*, *hatdex*, *safenetwork*
  - decentralised data exchange "open source" iot -currency -coins -cryptocurrencies -> *json-ld*, *web of things*, *DECODE*, *streembit*
  - Standard federation W3C/IETF
- https://www.bortzmeyer.org/search?pattern=fédéré
- http://citeseerx.ist.psu.edu/search?q=federated+protocol

ANNEXE 2 : RÉFÉRENCES DES PROTOCOLES ÉVALUÉS

1 XMPP – ejabberd

https://xmpp.org/rfcs/rfc6121.html

https://wiki.xmpp.org/web/

https://docs.ejabberd.im/

2 RabbitMQ/AMPQ/MQTT

https://www.rabbitmq.com/

https://github.com/rabbitmq/rabbitmq-server

http://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html

3 ActivityPub – Mastodon

https://docs.joinmastodon.org/

https://github.com/tootsuite/mastodon

https://w3c.github.io/activitypub/

4 Ocean Protocol

https://oceanprotocol.com/faq/

5 Matrix

https://github.com/matrix-org/synapse/

https://matrix.org/docs/spec/

6 IRC

https://tools.ietf.org/html/rfc2813

http://www.irchelp.org/

7 Mainframe

https://mainframe.com/

8 Dock.io

https://dock.io/

https://medium.com/dock/our-roadmap-b3c7c6921c8e

9 HatDex

https://hatdex.org/

https://www.hubofallthings.com/

10 Datax.io

https://datax.io/

11 SafeNetwork

https://safenetwork.tech/

https://github.com/maidsafe/

12 StreemBit

https://streembit.github.io/

https://github.com/streembit/streembit-cli

13 DECODE

https://decodeproject.eu/

https://github.com/decodeproject

14 Streamr

https://medium.com/streamrblog

https://github.com/streamr-dev

15 OM2M

http://wiki.onem2m.org/index.php?title=Open_Source

https://projects.eclipse.org/projects/technology.om2m

http://www.onem2m.org/

ANNEXE 3 : MATRICE D'ÉVALUATION FINALE

Cf. Tableur joint.

La page suivante propose un extrait de la matrice, pour les 4 protocoles ayant obtenu le meilleur score.

						XMPP			matrix.org			ActivityPub - GHU			rabbitmq/ampq		
Type	**Critère**	**Sous-critère**	**Définition du score**	**Commentaire**	**Coef.**	Eval.	Note	Commentaire	Eval.	Note	Commentaire	Eval.	Note	Commentaire	Eval.	Note	Commentaire
Interne	Communauté	Activité code	0-aucun patch en 12 mois 3-un patch par mois 5-patch quotidiens	Assure de la vivacité du projet	9	5	45		5	45		5	45		5	45	
Interne	Communauté	Diffusion d'usage	Echelle libre : 0-pas utilisé 5- dans plus de 50 structures à l'international	Evalue la pertinence de la solution et la diversité des utilisateurs	5	5	25		5	25		5	25		5	25	rabbitmq très ut
Interne	Communauté	Roadmap	Echelle libre : 1- rien 5- détaillée	Jauge la maturité d'un projet et son organisation	3	3	9	https://support	1	3		1	3	rien trouvé	1	3	rien vu
Interne	Communauté	Taille	0- dev seul, pas de comm. 3- plusieurs devs, des échanges 5- plus de 50 dev et d'inputs externes	Assure de la vivacité et la résilience du projet	10	5	50		5	50		5	50	mastodon jugé	5	50	
Interne	Implémentations	Disponibilité	0-aucune 2-une 5- plusieurs	Garantit que le standard est bien défini	6	5	30		5	30	Server / client /	5	30		2	12	1 seule pour le
Interne	Implémentations	Langages	1- Un seul 5-plusieurs	facilité de codage	3	5	15		5	15	java, python, c-	5	15	plusieurs implé	1	3	coeur dans 1 se
Interne	Implémentations	Licence	0-Commerciale 3-Libre 5-Copyleft		9	5	45	GPL2	3	27	Apache Licence	5	45		3	27	plugin fédératio
Interne	Qualité Code	Commentaires	très peu 1 ; fonctions commentées ou documentées 2 ; en plus variables documentées 3; en plus codes complexes documentés 5	Evalue la facilité à retravailler sur le code du projet	5	2	10	(sur au moins 3	5	25		1	5	pas de commer	5	25	
Interne	Qualité Code	Documentation	documentation d'installation 1 ; + doc d'utilisation 3 ; + doc d'architecture générale 4 ; + doc technique détaillée 5	idem	7	5	35	https://docs.eja	5	35		3	21		5	35	
Interne	Qualité Code	Variables	variables au nom générique 1 ; variables aux noms abrégés 3 ; variables self-explanatory 5	idem	3	5	15		5	15		5	15		5	15	
Interne	Service tiers	Dépendance à un acteur centralisé	0-Obligatoire 3-remplaçable 5-facultatif/aucun	Assure l'indépendance du projet, et donc la possibilité de le répliquer	6	5	30		5	30		5	30		5	30	
Interne	Standard		1-specs publiques 3-rédaction commune 5-passé par un organisme international	Evalue la bonne définition du standard	8	5	40		3	24	Formulaire prop	5	40	W3C	3	24	ampq standardi
Interne	Structure porteuse		1-entreprise 3-asso 5-Fondation	garanties sur l'orientation du travail de la communauté	4	1	4		1	4		3	12	Copyright (C) 2	1	4	
Adapt POC	Architecture solution	Complexité architecture	Nombre de composants à installer et configurer pour une installation minimale. Cf échelle : #Archi	Simplicité d'usage	6	4	24		4	24	Instance homes	3	18		3	18	
Adapt POC	Authentification	Intégré	1- rien 3-une manière 5-multi-manières		6	5	30	LDAP / oauth v	5	30	https://matrix.o	5	30		5	30	via plugins
Adapt POC	Authentification	Standard	0-non 5-oui	Souligne la prise en compte de la sécurité par la solution	4	5	20	SASL	1	4		5	20	oauth2 possible	5	20	plugin iauth 2
Adapt POC	Autorisation	Intégré	1- Non géré 5-géré		4	3	12	modération	5	20	Notion room et	5	20		5	20	
Adapt POC	Cas d'usage	Prévu en conception	1-Autre usage initial 3-usage proche 5-Echange de données	Permet d'aller plus vite pour notre cas d'usage SEN1	9	5	45		5	45	conversation, w	1	9	Micro-blogging	3	27	messaging prot
Adapt POC	Format des données	Type de données	Si formalisme imposé et format limité 0. si formalisme imposé et format des données OK mais lourd à gérer 3 ; si formalisme imposé et format des données OK ou si non imposé 4 ; si bien documenté et outillé en + : 5	Evalue les limitations imposées par le protocole	11	5	55	https://gitlab	5	55	Format : JSON Formalisme : ev	5	55	Json-ld	4	44	AMPQ
Adapt POC	Gestion des échanges	Annuaire	1-aucun 3-basique 5-étendu	Simplicité d'usage	3	3	9		3	9		3	9		3	9	
Adapt POC	RGPD	Compatible	0 : pas compatible ; 3 : à priori compatible ; 5 : intégré et dans la doc	Approche non-experte, qualitative (à recouper avec autre tâche RGPD)	8	3	24	https://wiki.x	3	24		3	24	en discussion c	3	24	pas d'infos
Adapt POC	Transmission	Chiffrement	0-non 3- Possible via reverse proxy web 5-géré standard	Souligne la prise en compte de la sécurité par la solution	6	5	30		5	30	End-to-End enc	5	30	TLS	5	30	TLS possible
Adapt POC	Transmission	Mode envoi	1-un seul 3-deux modes 5-pull, push, event/queue	Simplicité d'usage	7	5	35	pub/sub, poll (B	3	21	pull par défaut v	3	21	publish subscri	1	7	event queue
Adapt LT	Architecture d'un nœud	Ressources	1 : pas de scalabilité architecturale, 3 : des possibilités mais lourdes, 5 : prévu	Facilité de diffusion d'usage	6	5	30	via server2serv	5	30		1	6		3	18	clustering
Adapt LT	Format des données	Surcouche	Encapsulation "overhead" . Echelle libre : 1-lourde 5- aucune	Economie de bande passante et d'énergie	3	3	9		3	9		5	15		3	9	
Adapt LT	Traçabilité non répudiable des autorisations et des transferts de données		1 : non prévu ; 5 : prévu	Permet de remplacer un tiers de confiance, par l'infrastructure elle-même	5	1	5		1	5		1	5		1	5	
							681			634			598			559	

ANNEXE 4 : PROCÉDURE D'INSTALLATION DE SYNAPSE (MATRIX.ORG)

```
Test de matrix.org

Implémentation choisie : synapse
Installation via playbook ansible
Domaines de test :
   lognact.fr : 51.158.23.146
   lognact.com : 51.158.23.165

Doc d'install : https://github.com/spantaleev/matrix-docker-ansible-deploy/blob/master/
docs/README.md

Installer synapse

DNS :
_matrix-identity._tcp 1800 IN SRV 10 0 443 matrix.lognact.fr.
_matrix._tcp 1800 IN SRV 10 0 8448 matrix.lognact.fr.
matrix 1800 IN A 51.158.23.146
riot 1800 IN CNAME matrix.lognact.fr.

Sur le serveur physique, là ou il y a kikin installé, se logguer en tant que ghusson
cd /home/ghusson/
mkdir tmp
cd tmp
git clone https://github.com/spantaleev/matrix-docker-ansible-deploy.git
cd matrix-docker-ansible-deploy/
mkdir inventory/host_vars/matrix.lognact.fr
cp examples/host-vars.yml inventory/host_vars/matrix.lognact.fr/
apt-get install pwgen

Générer des mots de passes pour les secrets partagés :
pwgen -s 64 1
pwgen -s 64 1

vi inventory/host_vars/matrix.lognact.fr/host-vars.yml
renseigner les variables
Exemple :
host_specific_matrix_ssl_lets_encrypt_support_email:
admin_matrix.lognact.fr@liberasys.com
host_specific_hostname_identity: lognact.fr
matrix_coturn_turn_static_auth_secret:
"NXR8fOQTBwVkTsr08emY5cZ2rPTDGiQiUdEA4UmmkwqfMd1OCWOiibOn2DqbVNHj"
matrix_synapse_macaroon_secret_key:
"RVuQHZnHftmeg8cOfFPNoY90X5uY4LtLHK2r32N9etgXDySNoEpbtrpPv8yVhjUF"

Fichier inventaire/hosts pour ansible
cp examples/hosts inventory/hosts
vi inventory/hosts
Exemple :
[matrix-servers]
==> matrix.lognact.fr ansible_host="srv3.consometers.org" ansible_port: "22000"

Lancer le setup
ansible-playbook --become -i inventory/hosts setup.yml --tags=setup-all
Recommencer jusqu'à ce que tout soit vert

démarrer les services :
ansible-playbook -i inventory/hosts setup.yml --tags=start

Installer les .wellknown
```

```
Installer apache/certbot
Sur un serveur (avec ports 80 et 443 de libres), installer apache + certificat
letsencrypt, ici sur srv1.consometers.org
DNS : le champ @A doit pointer vers l'IP du serveur qui sera utilisé. Exemple : @ 10800
IN A 51.158.23.118
apt-get install certbot python-certbot-apache -t stretch-backports
certbot --apache -d "lognact.fr"
renseigner les questions, activer la redirection HTTP -> HTTPS

Modifier la configuration apache générée pour gérer l'autre domaine (lognact.com) pour
HTTP :
vi /etc/apache2/sites-enabled/000-default.conf
<VirtualHost lognact.fr:80>
        ServerAdmin webmaster@localhost
        DocumentRoot /var/www/html
        ErrorLog ${APACHE_LOG_DIR}/error.log
        CustomLog ${APACHE_LOG_DIR}/access.log combined
        RewriteEngine on
        RewriteCond %{SERVER_NAME} =lognact.fr
        RewriteRule ^ https://%{SERVER_NAME}%{REQUEST_URI} [END,NE,R=permanent]
</VirtualHost>
<VirtualHost lognact.com:80>
        ServerAdmin webmaster@localhost
        DocumentRoot /var/www/html
        ErrorLog ${APACHE_LOG_DIR}/error.log
        CustomLog ${APACHE_LOG_DIR}/access.log combined
        RewriteEngine on
        RewriteCond %{SERVER_NAME} =lognact.com
        RewriteRule ^ https://%{SERVER_NAME}%{REQUEST_URI} [END,NE,R=permanent]
</VirtualHost>

Modifier la configuration apache générée pour gérer l'autre domaine (lognact.com) pour
HTTPS et activer la proxyfication inverse :
vi /etc/apache2/sites-enabled/000-default-le-ssl.conf
<IfModule mod_ssl.c>
<VirtualHost lognact.fr:443>
        ServerAdmin webmaster@localhost
        DocumentRoot /var/www/html
        ErrorLog ${APACHE_LOG_DIR}/error.log
        CustomLog ${APACHE_LOG_DIR}/access.log combined
        ServerName lognact.fr
        SSLCertificateFile /etc/letsencrypt/live/lognact.fr/fullchain.pem
        SSLCertificateKeyFile /etc/letsencrypt/live/lognact.fr/privkey.pem
        Include /etc/letsencrypt/options-ssl-apache.conf
        SSLProxyEngine on
        <Location /.well-known/matrix>
                ProxyPass "https://matrix.lognact.fr/.well-known/matrix"
        </Location>
</VirtualHost>
<VirtualHost lognact.com:443>
        ServerAdmin webmaster@localhost
        DocumentRoot /var/www/html
        ErrorLog ${APACHE_LOG_DIR}/error.log
        CustomLog ${APACHE_LOG_DIR}/access.log combined
        ServerName lognact.com
        SSLCertificateFile /etc/letsencrypt/live/lognact.com/fullchain.pem
        SSLCertificateKeyFile /etc/letsencrypt/live/lognact.com/privkey.pem
        Include /etc/letsencrypt/options-ssl-apache.conf
        SSLProxyEngine on
        <Location /.well-known/matrix>
                ProxyPass "https://matrix.lognact.com/.well-known/matrix"
        </Location>
</VirtualHost>
</IfModule>

Générer le certificat SSL pour le 2eme domaine
certbot --apache -d "lognact.com"
a2enmod proxy
```

```
sudo a2enmod proxy_http
service apache2 stop
service apache2 start

Vérification de bon fonctionnement :
sur le compte ansible (ghusson sur serveur physique)

    sudo apt-get install python-dnspython

    ansible-playbook -i inventory/hosts setup.yml --tags=self-check


Créer les users admin :
ansible-playbook --become -i inventory/hosts setup.yml --extra-vars='username=admin
password=admin@56270 admin=yes' --tags=register-user
ansible-playbook --become -i inventory/hosts setup.yml --extra-vars='username=gautier
password=gautier@56270 admin=yes' --tags=register-user
ansible-playbook --become -i inventory/hosts setup.yml --extra-vars='username=gautier2
password=gautier2@56270 admin=no' --tags=register-user
ansible-playbook --become -i inventory/hosts setup.yml --extra-vars='username=gregory
password=gregory@56270 admin=no' --tags=register-user

Tester :
    https://riot.lognact.fr
    https://riot.lognact.com

Docs



Clients

    https://riot.im/desktop.html


Librairies



Exemples / tutos
```

ANNEXE 5 : PROCÉDURE D'INSTALLATION D'EJABBERD (XMPP)

```
Implémentation sélectionnée : Ejabberd https://www.ejabberd.im/
Doc principale : https://docs.ejabberd.im/get-started/

Serveur : installation

Général

https://docs.ejabberd.im/admin/installation/#install-on-linux
https://github.com/processone/ejabberd
https://docs.ejabberd.im/admin/installation/

Création dossier général :
    mkdir /opt/xmpp
    cd /opt/xmpp

Téléchargement dernière version installateur debian/ubuntu :
    wget https://www.process-one.net/downloads/downloads-action.php?file=/ejabberd/
18.12.1/ejabberd-18.12.1-linux-x64.run
    mv downloads-action.php?file=%2Fejabberd%2F18.12.1%2Fejabberd-18.12.1-linux-x64.run
ejabberd-18.12.1-linux-x64.run
    chmod +x ejabberd-18.12.1-linux-x64.run

Lancer l'installateur

    ./ejabberd-18.12.1-linux-x64.run

Choisir la langue : [2] Francais
Accepter la licence
Choisir le dossier d'installation : /opt/xmpp/ejabberd-18.12.1
Nom de domaine : consometers.org
admin username : ejbdadmin
adlmin password : adminejbd!2019
cluster : N

Création des certificats (wildcard)
  mkdir /opt/xmpp/certificates
  apt-get install certbot -t stretch-backports
  certbot certonly --manual --preferred-challenges dns --server https://acme-
v02.api.letsencrypt.org/directory -d '*.consometers.org'
  certbot certonly --manual --preferred-challenges dns --server https://acme-
v02.api.letsencrypt.org/directory -d '*.consometers.org' -d 'consometers.org'
  cp /etc/letsencrypt/live/consometers.org/*.pem /opt/xmpp/certificates
openssl x509 -in /opt/xmpp/certificates/cert.pem -text -noout

Changer les certificats dans la configuration ejabbed
  nano /opt/xmpp/ejabberd/conf/ejabberd.yml
certfiles:
  - "/opt/xmpp/certificates/fullchain.pem"
  - "/opt/xmpp/certificates/privkey.pem"
##  - "/etc/letsencrypt/live/localhost/fullchain.pem"
##  - "/etc/letsencrypt/live/localhost/privkey.pem"
## ca_file: "/opt/xmpp/ejabberd-18.12.1/conf/cacert.pem"



Démarrer le noeud pour vérifier son fonctionnement :
    /opt/xmpp/ejabberd-18.12.1/bin/ejabberdctl live
    # taper (Ctrl + G)  Ctrl + C puis "a" pour quitter le test
```

```
Exemple :
21:20:05.817 [notice] Changed loghwm of /opt/xmpp/ejabberd/logs/ejabberd.log to 100

(ejabberd@localhost)1> 21:20:06.184 [info] Loading configuration from /opt/xmpp/ejabberd/
conf/ejabberd.yml
21:20:09.153 [info] Loading modules for consometers.org
21:20:09.174 [warning] Module 'mod_mam' is recommended for module 'mod_muc' but is not
found in the config
21:20:09.734 [info] Waiting for Mnesia synchronization to complete
21:20:09.853 [warning] No certificate found matching 'consometers.org': strictly
configured clients or servers will reject connections with this host; obtain a
certificate for this (sub)domain from any trusted CA such as Let's Encrypt
(www.letsencrypt.org)
21:20:09.854 [info] ejabberd 18.12.1 is started in the node ejabberd@localhost in 4.50s
21:20:09.862 [info] Start accepting TCP connections at [::]:5280 for ejabberd_http
21:20:09.863 [info] Start accepting TCP connections at [::]:5269 for ejabberd_s2s_in
21:20:09.863 [info] Start accepting TCP connections at [::]:5443 for ejabberd_http
21:20:09.863 [info] Start accepting TCP connections at [::]:5222 for ejabberd_c2s
21:20:09.863 [info] Start accepting TCP connections at 127.0.0.1:7777 for
mod_proxy65_stream



Configurer le démarrage auto (systemd) :
    cp /opt/xmpp/ejabberd-18.12.1/bin/ejabberd.service /etc/systemd/system
    systemctl daemon-reload
    systemctl enable ejabberd.service

Si IPv4
sed -i "s/::/0.0.0.0/g" /opt/xmpp/ejabberd/conf/ejabberd.yml

Démarrer le service

    systemctl start ejabberd

      journalctl -xl
     systemctl status ejabberd

Console admin : https://srv1.consometers.org:5443/admin
login : admin@consometers.org
password : admin


DNS : mettre les hosts du domaine en entrée type CNAME vers le serveur correpondant :
    upload
    conference
    proxy
    pubsub
    xmpp
Exemple :
upload 1800 IN CNAME srv1.consometers.org.
conference 1800 IN CNAME srv1.consometers.org.
proxy 1800 IN CNAME srv1.consometers.org.
pubsub 1800 IN CNAME srv1.consometers.org.
xmpp 1800 IN CNAME srv1.consometers.org.

DNS : mettre les entrées pour XMPP via les champs srv, ex :
_xmpp-client._tcp 1800 IN SRV 0 5 5222 srv1.consometers.org.
_xmpp-server._tcp 1800 IN SRV 0 5 5269 srv1.consometers.org.


Tester les accès :
    https://xmpp.net/
    a faire en mode c2s ET s2s !

Management mode console :
    /opt/xmpp/ejabberd-18.12.1/bin/ejabberdctl help
```

```
Pour tester XMPP :
    https://conversejs.org/fullscreen.html



Spam protection
If you start having problems with spammers sending messages or subscription requests to
your users, you can whitelist the servers that are allowed to federate with yours by
putting the following in /etc/ejabberd/ejabberd.yml:

acl:
  trusted_servers:
    server:
      - "cheogram.com"
      - "conference.soprani.ca"
      - "conversations.im"

access:
  s2s:
    trusted_servers: allow
    all: deny



Client

converse.js : https://conversejs.org/
XMPP-IOT http://www.xmpp-iot.org/basics/
http://www.xmpp-iot.org/
Read-Write IoT devices with converse.js : http://www.xmpp-iot.org/tutorials/read-and-
write-to-devices-with-converse-js/
libs : https://xmpp.org/software/libraries.html
lib simple : https://github.com/ltg-uic/simple-java-xmpp-client



Docs

Généralités : https://tiriboy.blogspot.com/2015/08/xmppjabber-for-iot-devices-part-1.html
tuto client/serveur IOT : http://tiriboy.blogspot.com/2015/08/xmppjabber-for-iot-devices-
part-2.html
Fonctionnement avec IOT : https://blog.securitycompass.com/xmpp-swiss-army-knife-for-
internet-of-things-iot-9eff783c44ba

IoT Broker : https://waher.se/Broker.md
IEEE XMPP IoT Interfaces : https://gitlab.com/IEEE-SA/XMPPI/IoT
Internet of Things (IoT) - Communication Patterns :
https://xmpp.org/uses/iot/patterns.html

Formalisme ?? : XEP-0323: Internet of Things - Sensor Data ; https://xmpp.org/extensions/
xep-0323.html
https://www.slideshare.net/peterwaher/xmpp-iot-sensor-data-xep0323

XEP-0325: Internet of Things - Control : https://xmpp.org/extensions/xep-0325.html
Proxy ? XEP-0100: Gateway Interaction : https://xmpp.org/extensions/xep-0100.html

tutoriels : http://www.xmpp-iot.org/gsoc/
```

ANNEXE 6 : FICHIER DE TRAVAIL

```
*--- Etape 1 : Recherches de solutions ---
*Requêtes de recherche :
	* RFC / IETF
	* qwant : protocole architecture fédérée -> trop de faux positifs
	* quant : protocole architecture fédérée -> RAS
	* duckduckgo : federated exchange protocol -> rabbitmq
	* duckduckgo : federated messenging -> mastodon
	* duckduckgo : decentralised protocol -> ocean protocol
	* DDG : federated data sharing
	* DDG : (examples of) federated data exchange
	* https://www.bortzmeyer.org/search?pattern=fédéré -> chou blanc
	* how federated exchange/sharing works
	* ddbms standard. ->
	* http://citeseerx.ist.psu.edu/search?q=federated+protocol
	* Standard federation W3C/IETF
	* DDG : decentralised data exchange protocol -> dock
	* DDG : decentralised data exchange : bigchaindb, gxs, hatdex, safenetwork,
	* ==> DDG : decentralised data exchange "open source" iot -currency -coins -
cryptocurrencies : json-ld, web of things, DECODE, streembit,


*Protocoles A EVALUER :
	* GEL : v XMPP : https://xmpp.org/
	* GHU: v rabbitmq / federation plugin : https://www.rabbitmq.com/federated-
exchanges.html  // AMPQ :  https://www.amqp.org/
	* GHU: v ActivityPub : https://www.w3.org/TR/activitypub/#Overview -
https://activitypub.rocks/
		* Et ses limites selon diaspora*
https://github.com/diaspora/diaspora/issues/7422
		* https://schub.io/blog/2018/02/01/activitypub-one-protocol-to-rule-them-
all.html
		* @todo Jaxom : Partager avec Ronan (hors travail) pour un retour technique
	* GHU: v-KO Ocean protocol : en R&D, mais intéressant à suivre
	* GEL: v Matrix : https://matrix.org/blog/home/
(https://en.wikipedia.org/wiki/Matrix_(protocol) https://matrix.org/blog/home/     et :
https://matrix.org/blog/posts/?s=iot  et : http://opensourcebridge.org/proposals/1616
	* GHU: v-KO IRC : https://tools.ietf.org/html/rfc1459
	* GEL: EXCLU AtomPub ? -> pour moi KO, c'est juste un protocole de push de post,
pas d'infra de fédération
	* GHU: v-KO Mainframe ? https://mainframe.com/ : en R&D mais intéressant à suivre
bien que complexe
	* GEL: EXCLU Ethereum ? 0x ? système de smart contract distribué, 0x a un
protocole pour broadcaster des commandes, mais pas de format de message adapté
	* GEL: EXCLU API HTTP / REST / HATEOAS / JSON (l'existant, pas "ce qu'on pourrait
faire avec"). => pas adapté, pas une infra de fédération, protocoles "nus"
	* GHU: v-KO https://dock.io/protocol (sur Ethereum)
	* GEL: EXCLU https://www.bigchaindb.com/ => BDD répartie, ne gère pas les contrats
ni l'accounting mais dispose d'un modèle de permissions
	* GHU : v-KO IPFS : système de fichiers répartit, ne gère pas la partie contrat ni
de droits
	* GEL : EXCLU https://gxs.gxb.io/en/ ==> doc tech en chinois, ça va pas être
possible...
https://github.com/gxchain/Technical-Documents/blob/master/gxb_contract_api.md
	* GHU : v-KO https://hatdex.org/the-technology
	* GHU : v-KO https://datax.io/  - voir sous quelle license ça va être
distribué ???
	* GHU : v-KO https://safenetwork.tech/how-it-works/ - internet en mode P2P,
concept à suivre !
	* GHU: v-KO streembit : https://blog.valbonne-consulting.com/2016/06/08/streembit-
a-decentralised-peer-2-peer-messaging-platform-for-the-iot/
	* DECODE : v-KO https://github.com/DECODEproject/decode-os
https://www.decodeproject.eu/ => ouverts à customisation selon les besoins !!!
```

	* GHU: v-KO MAIS A SUIVRE, très prometteur : streamr - prochaine version P2P : https://www.streamr.com/ https://medium.com/streamrblog/news-ruuvi-streamr-partner-to-create-worlds-largest-monetised-open-source-sensor-community-606f166c9384
	* GHU: MQTT : KO : trop bas niveau, pas prévu pour de la décentralisation
	* GHU: AMPQ / 0MQ : juste la partie transmission en queue, pas de gestion de contrat

29/01/19 :
    Suggestion Ronan : https://www.eclipse.org/om2m/ => Evaluation par Gautier
          https://github.com/mattconsto/onem2m-federation

*Autre - solutions - docs :
	* client XMPP :https://dino.im/ (C'est pas ce qui manque : https://xmpp.org/software/clients.html , https://wiki.jabberfr.org/Clients)
	* Une carte du Fediverse : https://fediverse.party/
	* Cours sur les DDBMS : http://www.inf.unibz.it/dis/teaching/DDB/
	* Identité fédérée : https://www.computerworld.com/article/2558314/security0/steps-for-choosing-the-right-federated-identity-standard-for-your-company.html
		* https://wso2.com/articles/2018/06/what-is-federated-identity-management
		* https://openid.net/specs/openid-connect-federation-1_0.html
		* https://docs.microsoft.com/en-us/previous-versions/dotnet/articles/bb498017(v=msdn.10)
		* https://docs.oasis-open.org/wsfed/federation/v1.2/os/ws-federation-1.2-spec-os.html
	* Blockchain !! https://medium.com/@sbmeunier/blockchain-technology-a-very-special-kind-of-distributed-database-e63d00781118
	* (https://en.wikipedia.org/wiki/Distributed_database && https://en.wikipedia.org/wiki/Distributed_data_store)
	* (Plateforme : http://druid.io/)
	* voir code de Mastodon ? https://joinmastodon.org/
	* voir code de Pleroma ? https://pleroma.social/
	* Gopher (Nan, c'est une blague)
	* https://mainframe.com/protocol/
	* Autorisation (en fédération) : https://en.wikipedia.org/wiki/XACML#XACML_and_other_standards
	* https://www.meetup.com/fr-FR/meetup-group-AzJhoEEn/events/lftxznywqbdb/
	* DATA CHAIN : https://metaquestions.me/2016/07/20/data-chains-what-why-how/
	* https://github.com/dirvine/data_chain/blob/master/docs/0029-data-chains.md
	* WEB of Things : www.w3.org/2015/05/wot-framework.pdf
	* JSON-LD : http://www.w3.org/TR/json-ld/
	* IoT & blockchain : https://www.rs-online.com/designspark/blockchain-and-iot
	* iiot (iot indus) : https://www.linkedin.com/pulse/why-iiot-revolution-happen-edge-cloud-kudzai-manditereza

Technos intéressantes
https://github.com/orbitdb/orbit-db
https://ipfs.io/

Docs intéressantes
https://outlierventures.io/research/introducing-the-convergence-ecosystem/

*Exemples :
	* SIG américain : https://www.esri.com/news/arcnews/fall09articles/pennsylvania-gis.html
	* Données recherche Canada : https://www.globus.org/developing-federated-research-data-repository-canada
	* Génome humain : http://science.sciencemag.org/content/352/6291/1278.full

*Notes Ronan :
    Je pense qu'ActivityPub serait une bonne direction bien que ce ne soit pas le seul protocole qui puisse nous permettre d'implémenter ce que nous souhaitons implémenter.
    Certes c'est un protocole tout jeune (début 2018) mais il est standardisé et pas mal de projets qui reposent sur ce protocole voient le jour (https://lordtoniok.com/posts/142-les-services-du-fediverse-fin-2018/).

Ces services sont tous orientés communication type "réseautage social" mais ActivityPub n'oblige pas à embrasser ce type d'utilisation.
L'avantage d'intégrer directement ce protocole est de pouvoir élargir les fonctionnalités du logiciel vers une orientation "réseautage social" tout en promouvant les services déjà existant et donc par extension intégrer une forme de militantisme pro-indépendance.

Mon second choix serait XMPP car c'est aussi un standard qui fonctionne bien et qui est plus ancien (et qui a fait ses preuves).

Attention cependant à ne pas trop s'orienter "réseautage social". Je ne pense pas que ce soit pertinent à ce stade de s'orienter vers ça bien que l'utilisation d'ActivityPub puisse nous permettre d'intégrer ce genre de fonctionnalités.

Jaxom : et quid du P2P ? -> Il faut mettre à plat notre vocabulaire pour être sûr de chercher la même chose :)

*Comparatifs protocoles/technos greg

*ActivityPub

Plus:
* Structure le format des données avec ActivityStream et la manière d'échanger (HTTP)
* Bonne structure en cas d'évolution vers une partie sociale
* Repose sur des protocoles standard (HTTP) et architecture (REST)
* double API : client-serveur et serveur-serveur
* Implémentation possible avec n'importe quel langage
* protocole nativement "fédéré"
* Gère nativement la publication en temps réel des données sur l'ensemble des serveurs fédérés référencés sur l'acteur

Moins:
* Très orienté réseau social et une majeure partie du protocole ne sera pas forcément utilisée
* La partie authentification/autorisation (sécurité) n'est pas précisée dans la spécification. Recommandation vers OAuth
* Couplage fort avec le vocabulaire (ActivityStream). Cela peut-être éventuellement contraignant sur les objets échangés (à confirmer)
* Surcouche sociale sur REST : est-ce finalement intéressant ?
* Les données publiées sont distribuées sur l'ensemble des serveurs, à voir si ce fonctionnement est désiré... : en mode "mur", sinon publish/subscribe (messages privés, envoyés aux destinataires seulement)

*API HTTP / REST / HATEOAS / JSON

Plus:
* Protocole standard bas niveau mixé avec architecture standard REST.
* API entièrement à définir : grande souplesse sur les choix et évolution
* hateoas : les ressources sont identifiées
* main complète sur la manière de faire en fonction des besoins
* Implémentation possible avec n'importe quel langage
* Maîtrise de la partie sécurité : on peut tout envisager (OAuth, JSON Web Token, etc.)

Moins :
* Ne gère pas le formalisme des données. Le JSON impose un format mais le contenu n'est pas formalisé
* API entièrement à définir : travail contraignant même si le périmètre reste pour l'instant limité
* Notion fédération à implémenter

*AtomPub

Plus :
* Protocole général pour publier/modifier des ressources web
* Gère le formalisme des données

	* Repose aussi sur standard HTTP et architecture REST
	* Semble plus léger et générique que ActivityPub

Moins :
	* Basé sur le format XML, plus lourd que le JSON
	* Notion "fédération" à implémenter

*MainFrame ?

A platform for easily building and running distributed, unstoppable apps.
https://mainframe.com/

*Ethereum ? 0x ?

https://www.ethereum.org/
https://0x.org/

*18/12/18 Restitution commune
Jaxom : pas de graal de standard
Greg : analyse des technos aujd dispos
Gautier : Papiers univs/DDBMS - pas pertinent car DB déjà présentes. AP pas étudié mais reste un fort potentiel. Idem XMPP plus ancien mais pertinent. RabbitMQ ?? Matrix à voir mais proche XMPP. Ocean protocol semble le plus adapté mais est en béta
Jaxom : attention, la notion/définition de fédération dépend des projets, et des fois c'est juste le lien entre serveurs gérés administrativement par la même entité.
Ronan : RàS car le besoin est plus large qu'attendu au départ (comparaison conso). -> On veut assembler des solutions qui ont leur fonctionnement propre !

Conclusion de cette première étape : Liste bouclée pour le moment ! 5-6 solutions à juger selon des critères à définir. C'est l'étape actuelle !

*--- Etape 2 : Critères ---
travail fait par Greg et Jaxom + Ruby

TODO @all : proposer des critères d'évaluation des solutions. Remise en commun courant janvier
S'appuyer sur : https://forge.pallavi.be/projects/qualoss-platform/repository/changes/visualization-tool/trunk/assessment-documents
et https://www.openhub.net/ - https://opensourceprojects.eu/p/osp/osseval/wiki/
https://www.slideshare.net/galoppini/scoring-zarafa-with-sos-open-source

Catégories de critères :
	* Qualité de la solution,
	* adaptation au besoin POC,
	* adaptation au besoin long-terme,
	* Autres ?

*Critères de Qualité de la solution :
	* Taille et activité de la communauté utilisatrice : forum, support, équipe développeuse - Communauté ? protocole déjà utilisé par de nombreux projets ? nouveau ? communauté active ?
	* Activité du projet : présence de patchs récents, rythme nouvelles versions
	* Nature de la structure porteuse : non-profit, entreprise, aucune
	* Feuille de route : présente ou non
	* Présence d'un standard : oui, validé par un groupe (W3C...), non

	* Disponibilité des implémentations : non, une, plusieurs
		* Implémentations existantes (librairies) ? langages ? si aucune implémentation, tout est à développer
	* Licence des implémentations : libre, open-source, copyleft
	* Appui sur des services tiers : aucun, reproductible, verrouillé

*Critères d'adaptation au besoin POC
	* Adéquation vis à vis des cas d'usages POC (données énergie etc) par rapport à la conception initiale de l'outil
		* à détailler ?? pour quantifier

```
      * Panel fonctionnel (fonctionnalités déjà présentes dans le code du logiciel),
            * prévision d'une diversité d'usages/de fonctions ?? A re-définir

      * Authentification / Autorisation : le protocole définit-il ses propres procédures
pour ces phases.[authentification = métaphore "tu sonnes à ma porte je te reconnais ou
pas" autorisation = une fois que tu es rentré chez moi je t'autorise à faire ci ou ça]
            * oui : complexité mise en oeuvre, rapprochement avec standard (ex : oauth,
token, etc.)
            * non : possibilité de greffer des protocoles standard pour gérer ces phases
(ex : oauth, token, etc.)
      * Gestion des échanges : adresses (trop précis/détaillé), annuaire, préparation de
transmission [ex: Webfinger c'est un outil qui permet de trouver un compte perso, c'est
un genre de protocole d'annuaire, sur un réseau fédéré on peut trouver qui il y a sur
quel instance]
      * Format des données : quel format ?  [format = c'est la forme du message ex :
excel, texte, xml, json, etc....] [encapsulation = métaphore "l'enveloppe de ta lettre"]
            * a manière de le traiter y'a t il des outils pour les traiter facilement
(le format aura des répercussions aussi sur les volumes échangés)
      * Transport des données : par quel protocole réseau transitent les données ? (ex :
http, smtp, ftp, etc.). Cela peut avoir des répercussions sur la configuration sécurité
des serveurs (ex : obligation d'ouvrir des ports supplémentaires). Chiffrement des
données ? Gestion du contrôle de transmission [il y a plusieurs manière d'envoyer les
données exemple protocole web http (le plus simple classique) mais il y en a d'autres,
cela peut avoir des répercussion sur la sécurité par ex]
      * Modes d'envoi des données : le protocole met-il en oeuvre plusieurs modes
[implementation ==code] [comment on décide d'envoyer les données - "tu m'appelles pour
avoir les données ou c'est automatique"]:
            * push (temps réel) : l'entité envoie ses données vers les autres [-->
j'autorise implicitement (ou pas) les autres instances à utiliser mes données]
            * pull / on-demand : l'entité met à disposition ses données pour les autres
            * autre ?
            * Question : est-ce que ça impacte la perception du business model
(utilisation/stockage de données) pour un certain prestataire ?
            * NB : si un seul mode, ca impacte fortement le dimensionnement (ressources)
des instances
      * Compatibilité avec le Règlement Général sur la Protection des Données,

*Critères d'adaptation au besoin long-terme
      * Adéquations vis à vis des cas d'usages LT (nombreux serveurs, données
hétérogènes)
      * Formalisme : formalisme imposé (Ex : ActivityStream, JSON-LD, etc...) ou libre.
Si imposé, évolutif ? souplesse ? Compatibilité avec le formalisme des données
            * [ vulgarisation pour Ruby - formalisme =comment j'écris dans mon fichier ]
      * Capacité de passage à l'échelle : augmentation de taille, load-balancing

*Critère autres


*
*21/12/18 - constitution matrice

Et la blockchain : il faudra évaluer des protocoles blockchain, type
https://www.ethereum.org/
Faut-il noter chacun chaque implémentation et moyenner, ou faire un consensus pour chaque
note ?
TODO @all : parler de la blockchain


*08/01/19 - Reprise matrice
matrice co-éditable sur https://cloud-epr.yourownnet.cloud/s/BjP4HTKYX9ABMyq
Remarque Gautier :
      * il n'y a pas de critère de qualité du code (commentaires, documentation,
factorisation / longueur des fonctions, nommage des variables, ...)
            * commentaire : (très peu 1 ; fonctions commentées ou documentées 2 ; en
plus variables documentées 3; en plus codes complexes documentés 5)
            * taille des fonctions : (1 fonction > 200 ligne 1 ; 1 fonction > 150 lignes
3 ; contions < 100 lignes 5) ==> laissé de côté, trop chronophage
            * documentation (documentation d'installation 1  ; + doc d'utilisation 3 ; +
```

doc d'architecture générale 4 ; + doc technique détaillée 5)
            * nommage des variables (variables au nom générique 1 ; variables aux noms abrégés 3 ; variables self-explanatory 5)
      * beaucoup de critères, peut être pas nécessaire pour le POC, à voir en fonction du budget alloué à l'étude du protocole
            * P-e en basculer en "adaptation LT"
Q : qu'est ce qu'il y a comme protocole standard d'autorisation ?
Critères de formalisme données (l.25 et 35) : redéfinir clairement la notion, et le critère objectif à évaluer. Une pré-définition est-elle bloquante ou au contraire un avantage ?

La note de "0" sur un critère est éliminatoire (ex. licence commerciale).

Pour l'évaluation des solutions avec la matrice (réponse à Grégory) : la matrice sera le plus objective possible, une seule évaluation par l'un d'entre nous suffira.

Élimination des solutions potentielles par la matrice via note éliminatoire sur critère ==> 1er filtre. le reste on passe dans la matrice

*11/01/19 - Mise au point commune

*18/01/19 - Choix des protocoles à répartir
GHU, GEL, JXM
Jaxom préfère (au vu de ses compétences) être en relecture plutôt qu'en première évaluation. Pour ne pas surcharger le travail : 4 relectures max.
Deux prochains RDV : Mercredi 23 vendredi 25/01 à 9h30
@Jaxom : envoyer invits

*23/01/19 - Remplissage matrice
GHU, GEL, JXM
Autres protocoles trouvés par Gautier, à évaluer également. Date finale maintenue à fin janvier pour la sélection du protocole
Matrice à modifier/aménager par Jaxom.

*25/01/19 - Réévaluation croisée
Gautier : Matrix et XMPP
Jaxom : les protocoles éliminés par Gautier + caractériser les formalismes des protocoles en tête.

Todos :
@Jaxom : prévoir événement de restitution aux consometers, à caler avec Ruby.
@Gautier+Grégory : prévoir un "lab de test" jeudi 31/01.

Préparer un rapport de conclusion sur cette étape.
reste : faire le formalisme, STANDBY - une fois que les 3 protos finalistes sont choisis libres de formalisme.

*29/01/19 - Compilation résultats
Rassemblement de toutes les notes, pour déterminer le trio de tête
Mise au point de la notation commune : ActivityPub...
AP : débat sur la nature du cas d'usage initial (éloigné car social) mais échange de contenu tout de même.
      Sera à rappeler lors de la restitution, pour avoir d'autres usages.
OM2M : évalué par Gautier d'ici vendredi.

*01/02/19 - Restitution choix du protocole
In : Cédric, Gautier, Grégory, Jaxom, Ruby, Solenn

Choix aujd : XMPP et Matrix.org . Beaucoup de potentiel pour les protocoles à blockchain, mais pas mature aujourd'hui.

Q. Ruby :
    - Pourquoi ActivityPub arrive 3e ? -> Car très orienté vers l'usage micro blog, alors que les deux autres généralistes
    - Quid sur les protocoles orienté Blockchain ? on attend la mise en place d'échanges P2P, la précision des datachains... Les solutions ne sont pas matures.