POLYTECH NICE-SOPHIA

Université Nice Sophia Antipolis

Rapport final projet SOA

Smartrix Grid

Shifting the energy paradigm

Groupe F

Dina Abakkali - Sylvain Marsili - Thomas Martin - Clement Poueyto - Florian Striebel

Choix de conception du projet Smartrix Grid

Au sein de notre architecture, nous avons cherché à séparer les responsabilités, c'est pourquoi nous l'avons découpée en un ensemble de services répondant chacun à un besoin différent. Nous avons donc choisi une architecture mélangeant l'événementiel et le micro-service.

La partie orientée événement sert à la transmission des informations de consommation et de production d'électricité au sein du réseau. L'avantage est que l'on garde un couplage lâche entre les différents services qui traitent ces informations de manière asynchrone. Cela apporte un gain de résilience en évitant les Single point of Failure (notamment avec Kafka) et un gain de performance pour traiter le grand nombre de données envoyées.

Pour ce qui est de la partie Micro-service, elle permet de réaliser des opérations synchrones côté client comme l'accès aux statistiques, à l'état de la batterie, ou aux factures.

Dans ce projet nous devons prendre en compte les communautés de ménages, pour ce faire nous les avons regroupées sous forme de quartier afin de mieux visualiser les consommations dans une zone donnée.

Une consommation d'un client est caractérisée par une quantité d'énergie, un identifiant du client, une date, une prise dans la maison par laquelle le courant est utilisé ainsi qu'un identifiant de quartier permettant de mieux gérer localement la consommation et la production et enfin une source d'énergie dans laquelle le client puise.
Une production d'énergie peut venir d'un client ou d'un fournisseur et peut être à destination du réseau ou d'une batterie
Un client peut produire et consommer de l'électricité pour sa batterie ou pour le réseau.
On considère que le compteur du client envoie périodiquement des informations au réseau sur la consommation du client, c'est pourquoi nous utilisons des dates et non une période. La valeur correspond à la quantité d'énergie consommée entre deux envoies.

Un client doit avoir un contrat dédié à la consommation pour utiliser de l'électricité et un contrat dédié à la production s'il possède des panneaux solaires et qu'il souhaite revendre son énergie au réseau.
Il peut cependant stocker l'énergie qu'il produit dans sa batterie afin d'atteindre l'autarcie.
Le réseau rachètera toujours l'électricité fournie par un client car il s'agit d'énergie verte et peu chère. Le client verra alors son gain déduit de sa facture à la fin du mois.
Si un client branche sa voiture, et que la batterie est presque remplie alors on reporte la charge plus tard dans la nuit lorsque le pic de consommation aura diminué, dans le cas contraire elle recharge immédiatement.

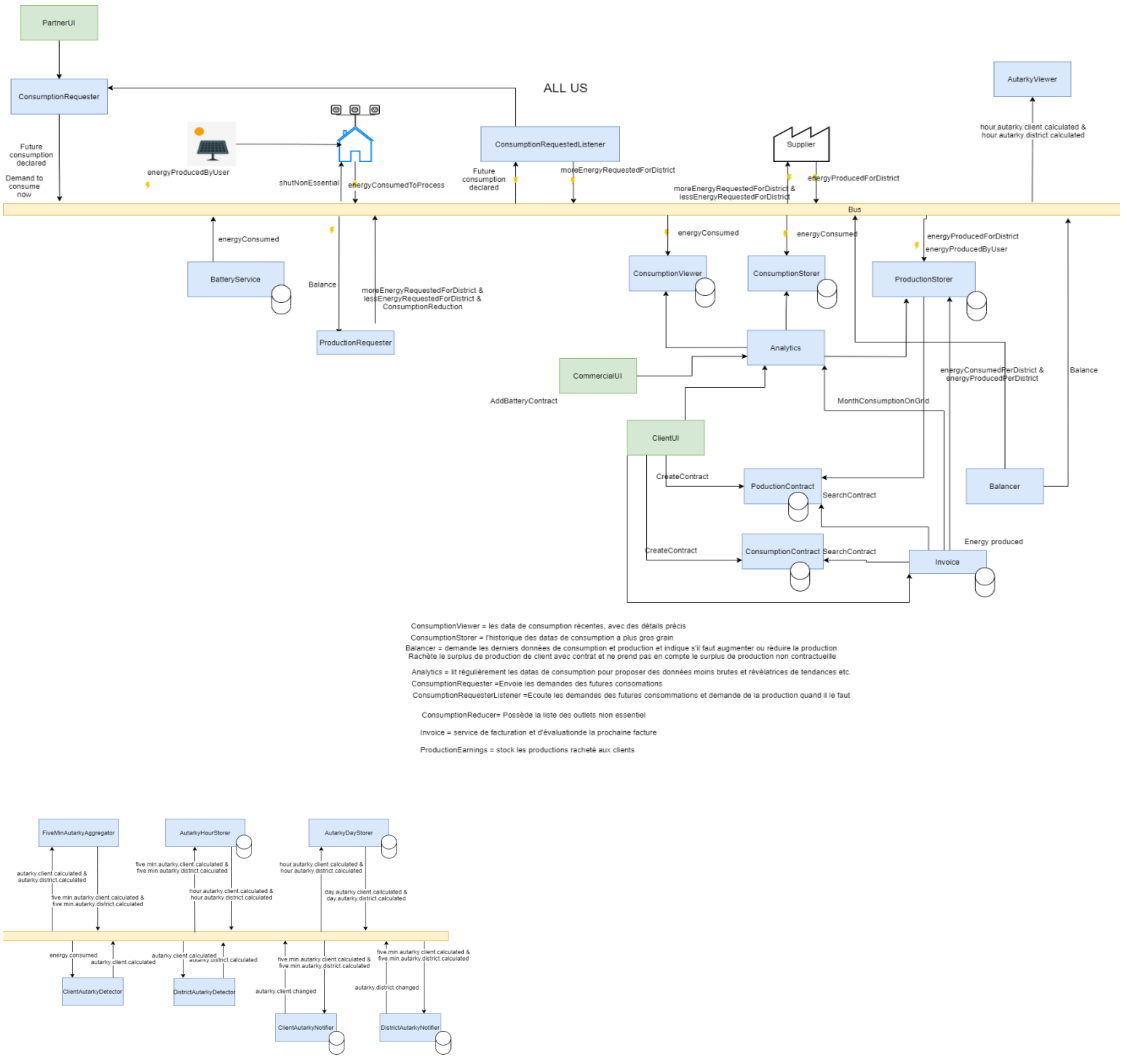
Un partenaire peut prévenir le réseau d'un pic de consommation de sa part à une certaine heure afin de s'assurer que les fournisseurs ont l'énergie nécessaire pour subvenir aux besoins. En cas de surconsommation, il peut être demandé de couper tous les éléments renseignés comme non essentiels car les fournisseurs ont une capacité limitée de production.

SmartGrid s'assure plusieurs fois par heure que la consommation et la production d'électricité sont équilibrées dans tous les quartiers. En cas de problème, il est possible d'envoyer des demandes d'augmentation ou de réduction de la production aux fournisseurs.

Pour un client, l'autarcie est atteinte lorsqu'il ne consomme pas d'énergie provenant de la grille. Quand on calcule l'autarcie d'un client sur une période donnée, une seule donnée indiquant une consommation depuis la grille suffit à invalider l'autarcie du client. De cette manière si un client a consommé depuis la grille pendant 5 secondes sur les 5 dernières minutes alors il n'a pas été en autarcie durant ces 5 dernières minutes.
Un district atteint l'autarcie lorsqu'aucun des clients y résidant ne consomme d'énergie provenant de la grille. De la même manière que pour un client, s'il existe une seule donnée durant une période de temps indiquant qu'un client résidant dans le district n'a pas été en autarcie, alors c'est le district entier qui n'est pas compté comme étant en autarcie pour la période de temps calculée.

Le temps qui passe est simulé dans notre projet. Nous avons des services qui ont deux modes de fonctionnement : real time et mock. En mode real time ces services ont des tâches répétitives avec un certain pas de temps (exemple: le balancer calcule la balance toutes les 5 secondes). Pour pouvoir simuler le temps nous avons donc mis en place un mode Mock pour ces services qui vont alors garder en mémoire une date, un timestamp. Ce timestamp va servir de repère au service. Lors de l'exécution de nos scripts de mock de données, on va faire avancer dans le temps ces services de manière manuelle via une requête http, les services vont alors comparer la date qu'ils ont en mémoire et celle qu'ils viennent de recevoir et vont effectuer des actions en conséquences (par exemple si la nouvelle date reçu est supérieur de plus de 5 secondes de celle qui est stocké, alors calculer la balancer). De cette manière, notre système va produire des événements comme s'il avait réellement tourné durant la période de temps mockée.

Diagramme d’architecture



Notre architecture est centrée autour d’un bus d'événements qui permet de transmettre les informations de consommations et de production d'énergie à l’ensemble des services à travers différents topics.

House et Supplier

Au sein du réseau les clients envoient l'information de leur consommation et de production à travers le service *House* tandis que les fournisseurs d'électricité envoient la quantité d'énergie produite à travers le service *Supplier*.
Ces services permettent de mocker de nombreuses données dans le temps pour se rapprocher d'un système en conditions réelles.

Consumption Viewer

Lorsqu'un événement de consommation est émis, les données sont enregistrées de manière brute. Ce service permet d'obtenir des informations de consommations récentes et précises sur une journée en fonction d'un client ou d'un quartier.

Consumption Storer & Consumption day storer

Consumption Storer réalise des agrégats de données de consommations d'un client pour une heure et Consumption day storer utilise les agrégats par heure pour réaliser des agrégats par jour. Cela évite d'aller chercher l'ensemble des consommations et de réaliser des calculs sur ces dernières à chaque appel.
Les services de consumption Viewer et Storer ont été séparés car l'utilisation de leurs données est différente.

Five min consumption aggregator

Ce service réalise des agrégations pour cinq minutes qui seront utilisées par la suite par *Consumption Storer.*

Analytics

Le service Analytics permet de regrouper des données par utilisateur ou par quartier pour proposer une vue d'ensemble des consommations des utilisateurs.
Il est possible de récupérer les informations heure par heure, par jour, ou mois pour un utilisateur ou un quartier.
Ce service permet notamment à Charles d'accéder aux consommations des communautés

ProuctionStorer

Ce service regroupe l'ensemble d'informations de production d'électricité des fournisseurs et des clients.
Il réalise des agrégats sur les productions des clients par jour pour le réseau.
Ces informations serviront à déduire de la facture du client l'énergie qu'il a produite.

Consumption Contract & Production Contract

Ces services permettent de recenser l'ensemble des clients du réseau.
Un contrat a une date de début et de fin et un prix de consommation ou de production au kWH.
Le contrat de consommation fixe le prix auquel le client paie l'électricité qu'il utilise.
Si un client a un contrat de production alors il sera en mesure de se faire racheter l'électricité qu'il produit pour le réseau.

Invoice

Les factures des clients sont générées chaque mois et sont sauvegardées pour ne pas avoir à recalculer le montant à chaque demande du client.
La facture prend en compte l'électricité consommée et produite sur le réseau pour un mois.
Ce service permet aussi d'obtenir une estimation de la facture du mois en utilisant la facture de l'année précédente à la même période et en réalisant une moyenne sur la consommation quotidienne du mois.

BatteryService
Un client peut posséder une batterie afin de stocker de l'énergie, il peut la recharger ou l'utiliser et a accès à son état actuel. La batterie a une capacité limitée

Balancer
Ce service permet de garantir l'équilibre entre la production et la consommation en permettant de suivre la balance à chaque instant. Pour ce faire, le service va périodiquement récupérer les dernières données de consommation et de production des différents quartiers, les sommer, et afficher l'équilibre sous forme de pourcentage. De plus.

Consumption Requester :
Nous avons implémenté ce service pour offrir à notre partenaire Elon la possibilité de renseigner les périodes de forte demande de consommation ainsi que leurs valeurs. Ce service reçoit les données renseignées par Elon à travers une cli afin qu'elles puissent être traitées par la suite par le service *Consumption Requested Listener*. Le partenaire est amené à renseigner son identifiant client , l'identifiant de son quartier, la quantité de consommation demandée et l'heure de début et de fin de la consommation. Nous avons considéré que les demandes des partenaires se font une seule fois mais elles sont traitées quotidiennement. Il envoie également les consommations des partenaires sur le bus, pour qu'ils soient par la suite traités par le consumption storer.
Le partenaire dans notre cas , est un client qui gère un business.
Ce service se charge aussi des autorisations pour les "super chargeurs" , ça bloque ou autorise le chargement des voitures des clients selon la situation.

Consumption Requested Listener :
Ce service reçoit les messages émis par Consumption Requester , et une heure avant la consommation prévue d'Elon, le service émet un message au supplier pour qu'il prenne en compte la future consommation et augmente sa production.

CarCharger:
Pour que la recharge de véhicule puisse être faite dans un période où la consommation est basse, nous obligeons les superchargeurs à nous demander l'autorisation de charge. Il peut nous la demander de 2 façons : soit une consommation directe ou soit une consommation future. Si par exemple l'utilisateur à une voiture chargé à 80% mais que le réseau est tendu lors de la demande et que le producteur arrive à sa limite de production, le service d'autorisation refusera et une heure a partir de laquelle il sera autorisé à charger la voiture lui sera envoyé

Production requester

Pour éviter la surcharge et le blackout de la grille, ce service reçoit les données de la part du *Balancer* par zone et envoie des demandes d'augmentation de production au supplier.

Client autarky detector

Ce service écoute les consommations du client, et émet, toutes les 5 secondes, l'état d'autarcie de sa consommation. Si le client consomme de la grille il n'est pas en autarcie sinon il l'est. Cette donnée est envoyée sur le bus.

District autarky detector

Ce service écoute les calculs bruts d'autarcie des clients, et emet toutes les 5 secondes l'état d'autarcie du district. Si il y a une seule donnée client du district qui n'est pas en autarcie alors, le district n'est pas en autarcie.

Five Min Autarky Aggregator

Ce service écoute les données d'autarcie envoyées par les détectors et les garde en mémoire pendant 5 mins avant d'émettre l'état de l'autarcie des clients et des districts pour les 5 dernières minutes. Si une donnée indique une dépendance à la grille, alors les 5 mins ne sont pas considérés comme du temps passé en autarcie.

Client Autarky Notifier & District Autarky Notifier

Ces services stockent en base de données le dernier état connu pour les clients / les districts en écoutant l'autarcie des 5 dernières minutes. Si, lors de la réception d'une nouvelle donnée, l'état reçu est différent de l'état stocké, alors ils émettent un message dans le bus indiquant un changement d'état d'autarcie permettant donc d'être notifié lorsqu'un client ou un district atteint l'autarcie ou redevient dépendant de la grille.

Autarky hour storer & Autarky day storer

Ces deux services vont stocker en base de données les données d'autarcie des clients et des districts. Le premier va agréger les données sur une heure et émettre sur le bus l'autarcie des clients / districts sur la dernière heure. Le second va écouter ces données par heure et les stocker sous forme de données par jour en mettant à jour les données à la réception de nouvelles heures.

Seule la dernière heure est stockée dans hour storer, lorsqu'une donnée est reçue, s'il s'agit d'une nouvelle heure alors on émet sur le bus la donnée de l'heure précédente et on la supprime.

Autarky Viewer

Ce service garde en RAM les données d'autarcie par heure des dernières 24 heures pour les clients et les districts en écoutant les données envoyées par Autarky Hour Storer.

Consumption authorizer

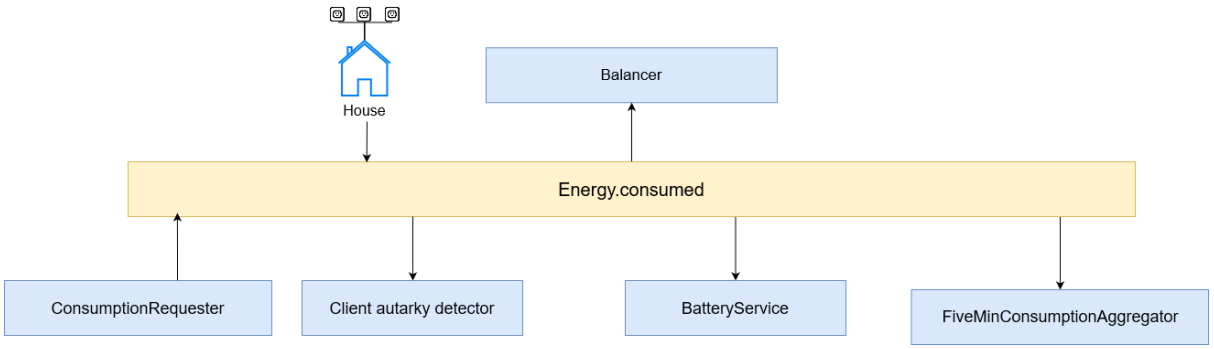
Pour s'assurer qu'on arrivera pas au blackout, on vérifie que la consommation ne dépasse pas les 80% de la production max d'une zone, avant d'autoriser le chargement des voitures .

Topics

Pour mieux gérer nos événements , on a découpé nos événements par topics.
On a fini par avoir 22 topics, dont 8 sont utilisés pour envoyer des agrégats :

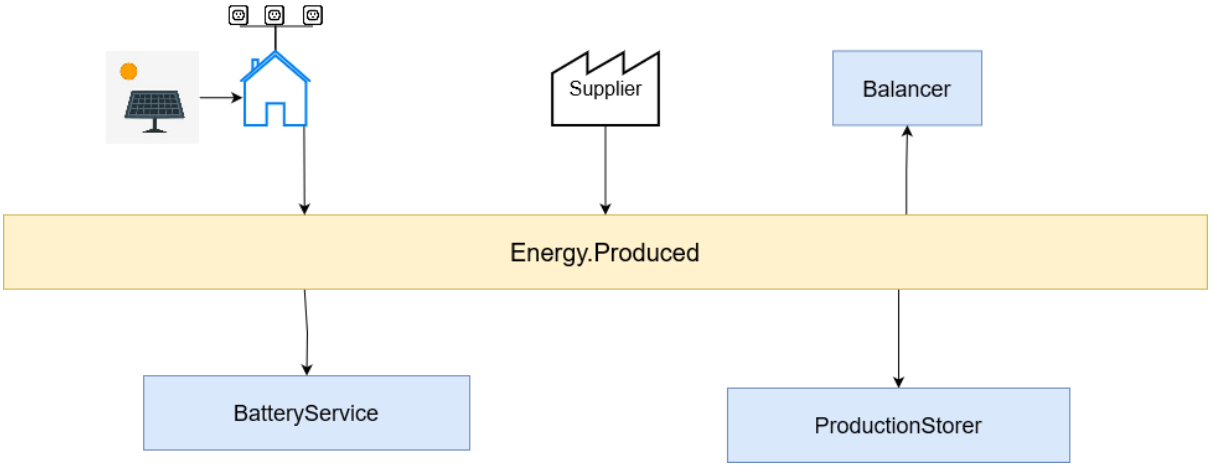
- five.min.energy.consumed
- hour.energy.consumed
- five.min.autarky.client.calculated
- hour.autarky.client.calculated
- day.autarky.client.calculated
- five.min.autarky.district.calculated
- hour.autarky.district.calculated
- day.autarky.district.calculated

Energy.consumed
Comme son nom l'indique, ce topic sert pour envoyer et recevoir les informations sur les consommations.



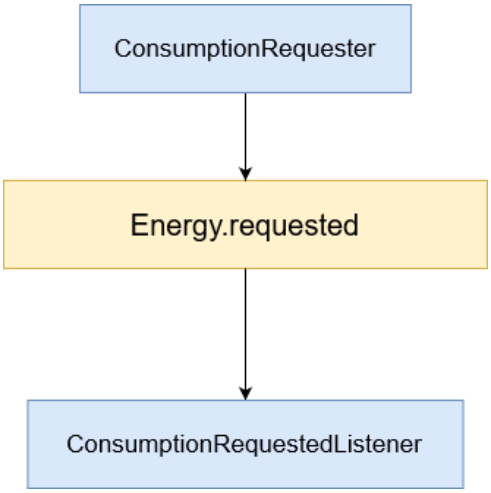
Energy.produced

Ce topic sert à échanger les données de production entre le supplier et house qui produisent l'énergie, et Balancer, BatteryService et ProductionStorer qui reçoivent les données et les traitent par la suite.



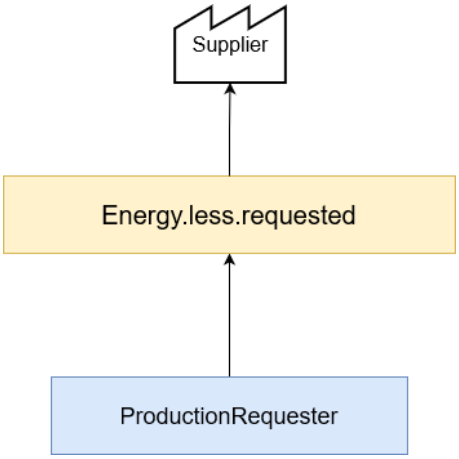
Energy.requested

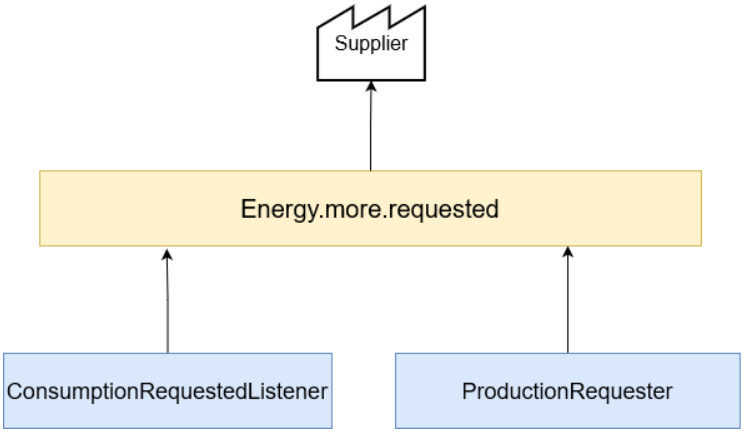
Afin d'envoyer des informations sur les consommations futures on a créé ce topics pour éviter la confusion entre les messages envoyés pour demander de la production et les messages envoyés pour planifier les productions.



Energy.less.requested & Energy.more.requested

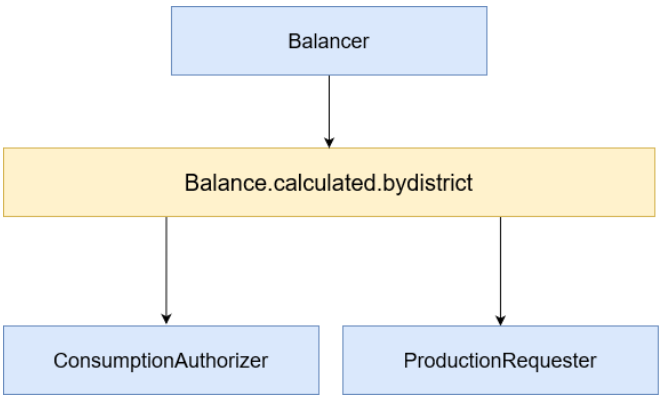
Afin de réaliser la balance entre les consommations et les productions, on envoie des informations au supplier grâce à ces deux topics.

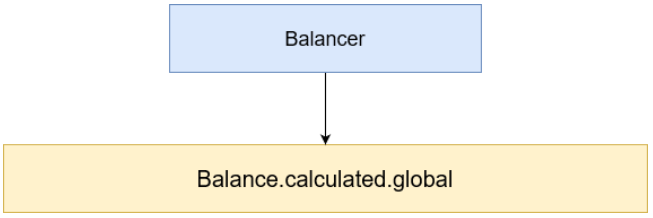




Balance.calculated.bydistrict & Balace.calculated.global

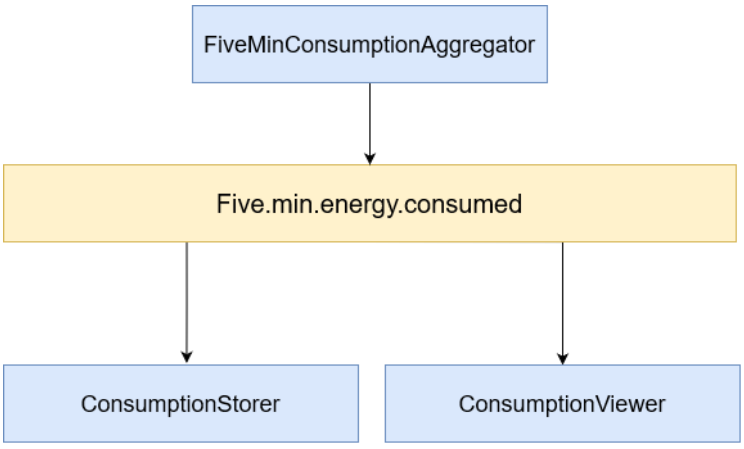
Une fois la balance (par district ou global) calculer, on envoi les informations du systèmes

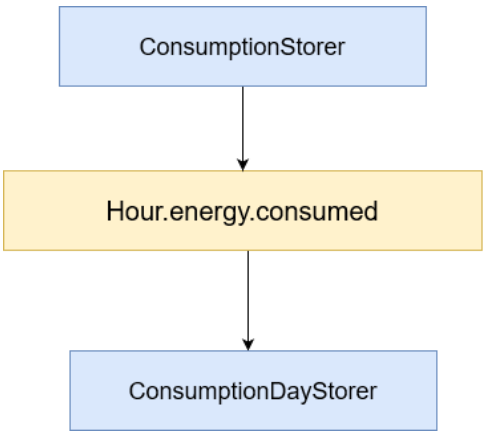




Five.min.energy.consumed & Hour.energy.consumed

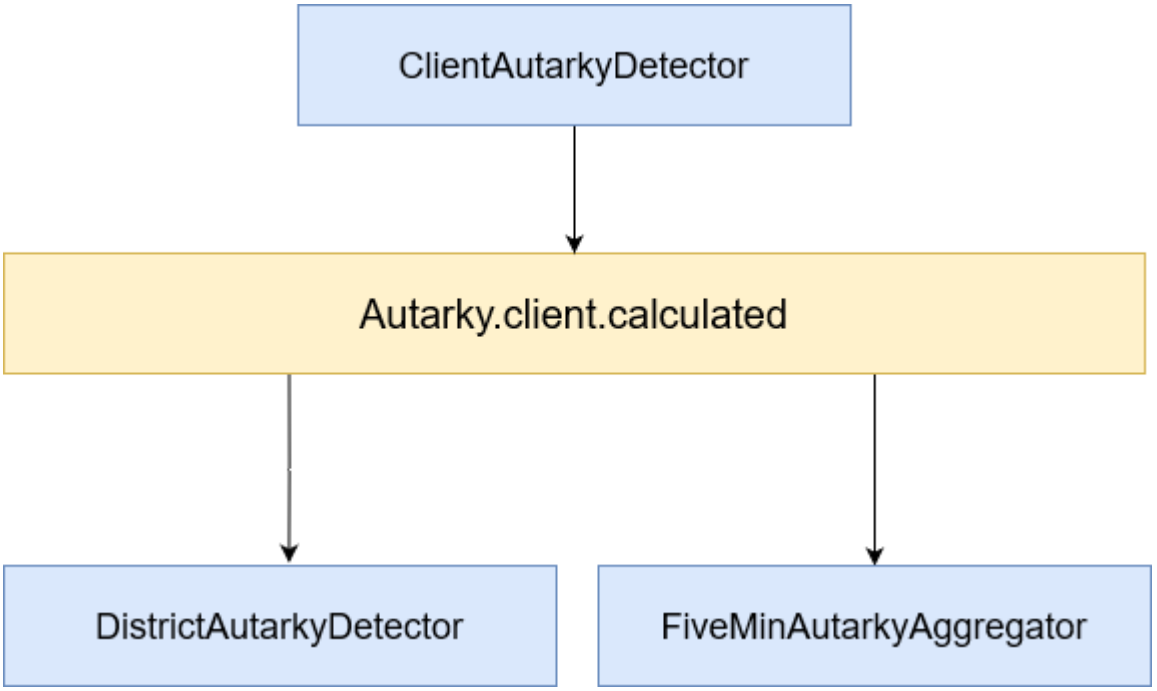
Sur le premier topic on envoie les agrégats de données sur un pas de 5 minutes. Sur le deuxième les agrégats de données sur 1 heure en fonction des données reçues sur 5 minutes.





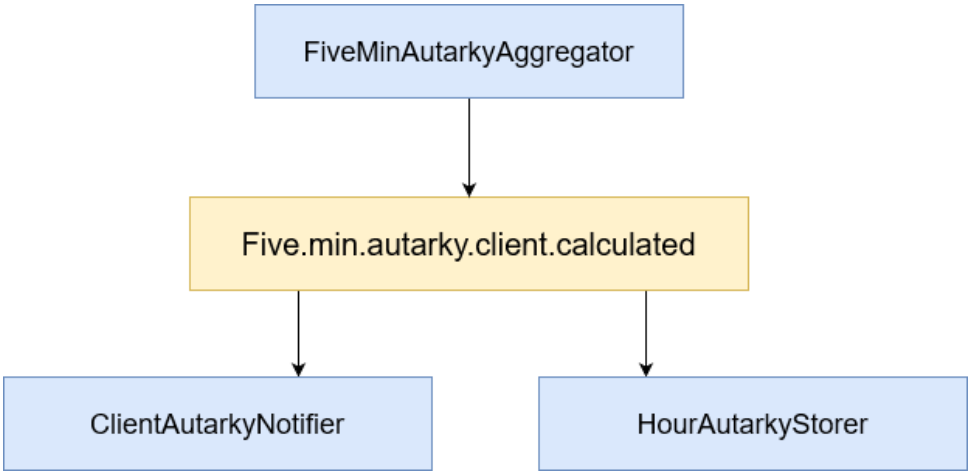
Autarky.client.calculated

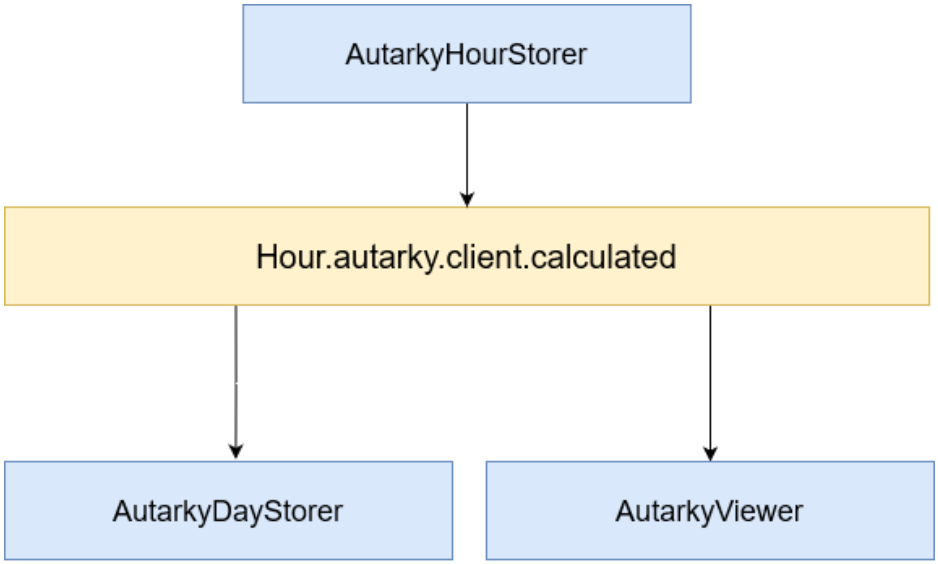
On envoie l'état d'autarcie pour chaque client

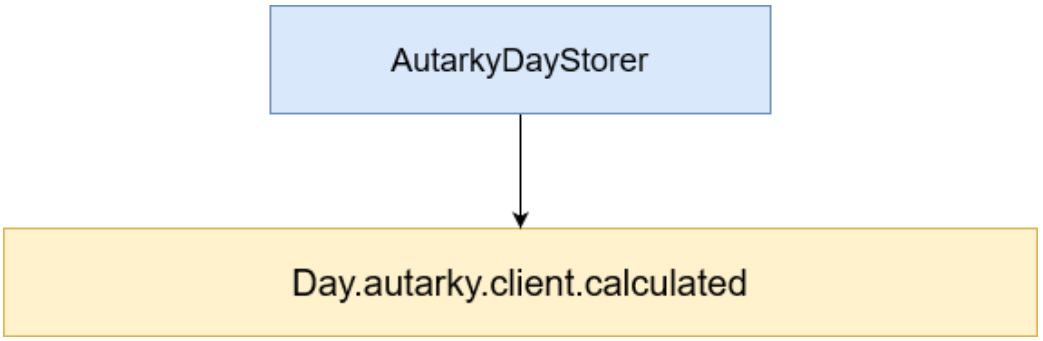


Five.min.autarky.client.calculated, Hour.autarky.client.calculated & Day.autarky.client.calculated

Ces topics servent à envoyer les informations sur les états d'autarcie des clients sur les 5 dernières minutes, sur la dernière heure et sur le dernier jour

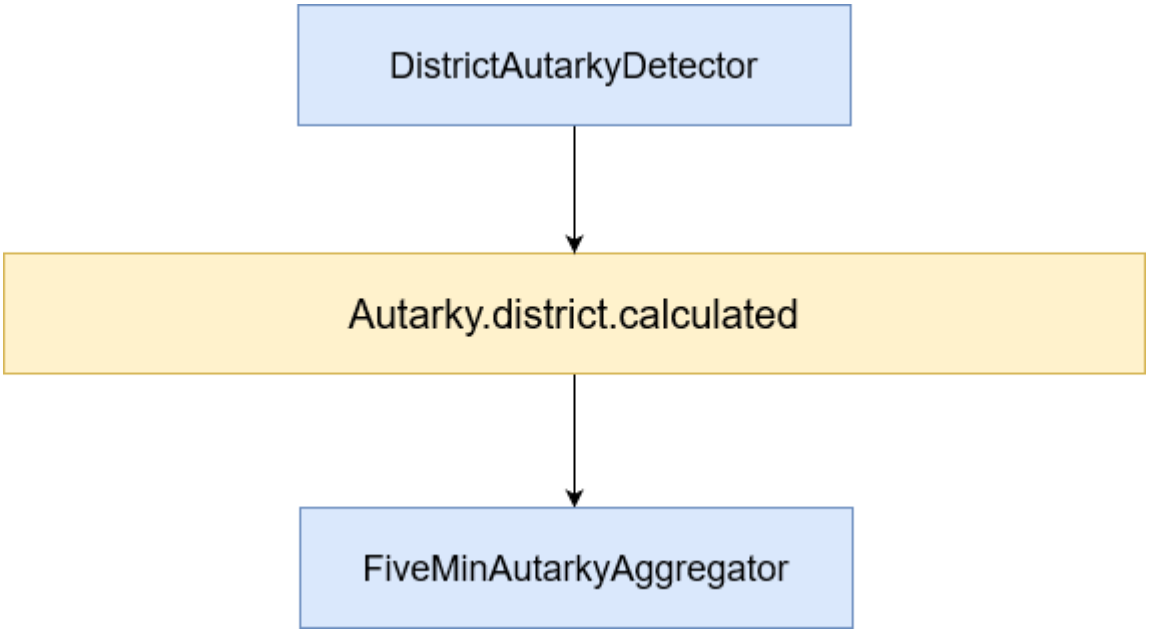






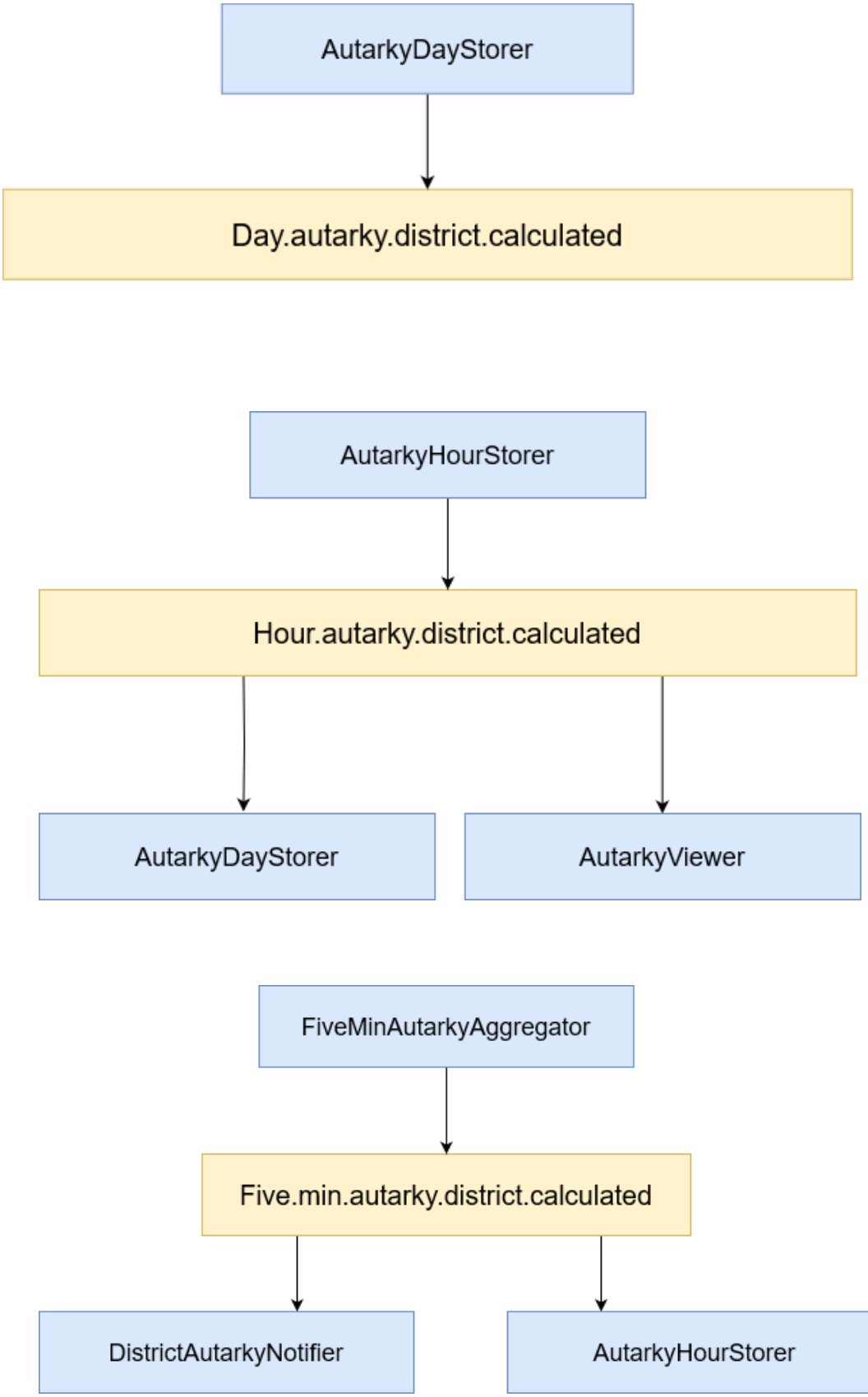
Autarky.district.calculated

On envoie l'état d'autarcie pour chaque district.

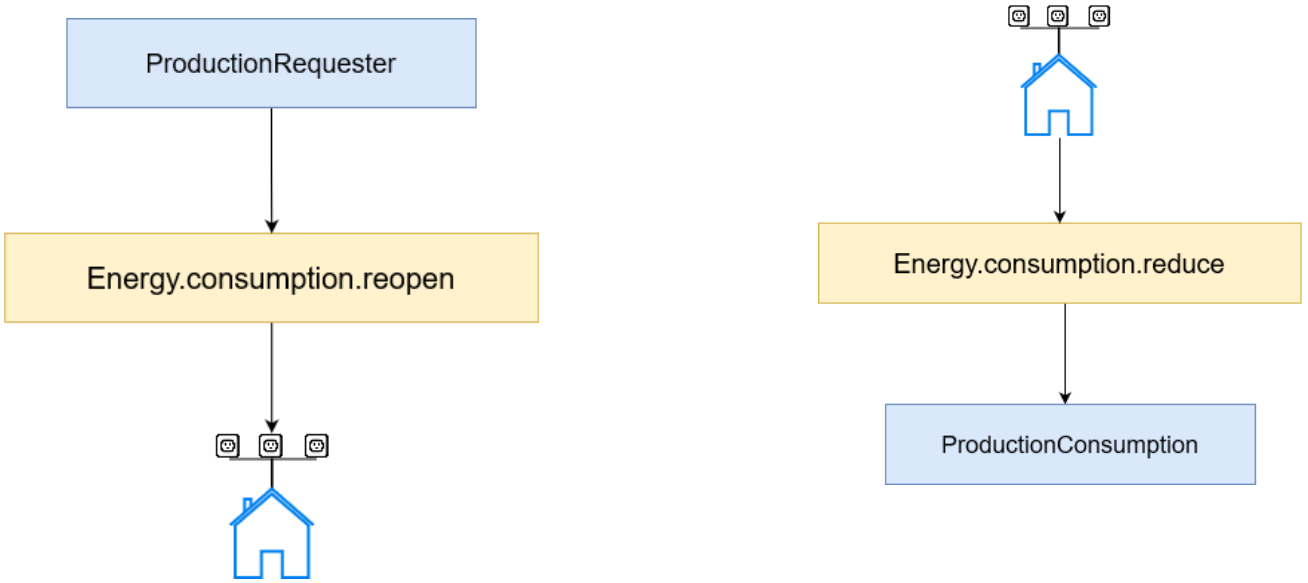


Day.autarky.district.calculated, Hour.autarky.district.calculated & Five.min.autarky.district.calculated

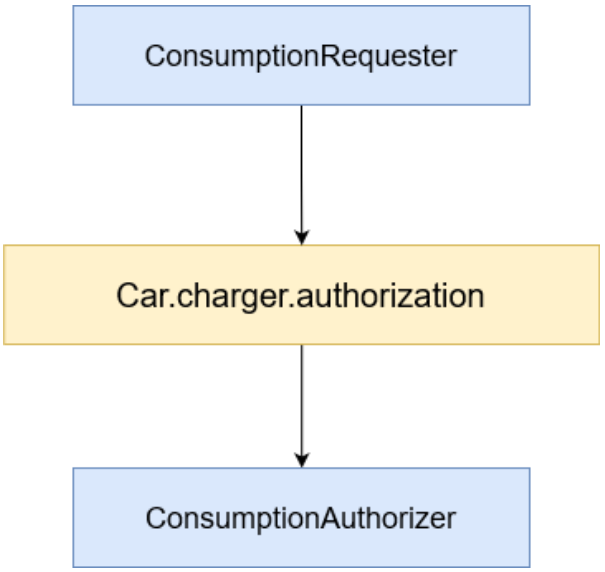
Ces topics servent à envoyer les informations sur les états d'autarcie des districts sur les 5 dernières minutes, sur la dernière heure et sur le dernier jour



Energy.consumption.reopen & Energy.consumption.reduce
Envoie un événement afin d' arrêter ou respectivement de relancer une consommation dite non-nécessaire

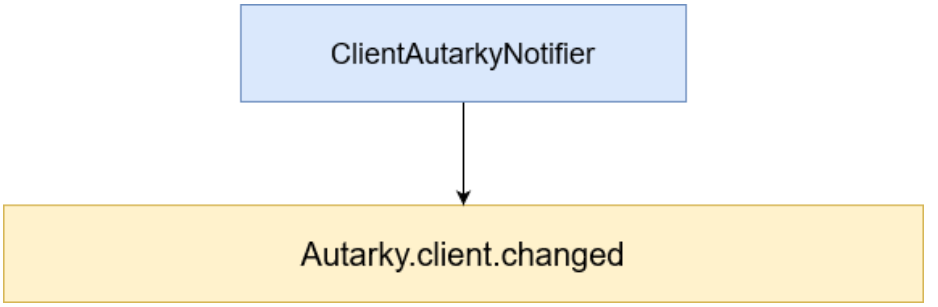


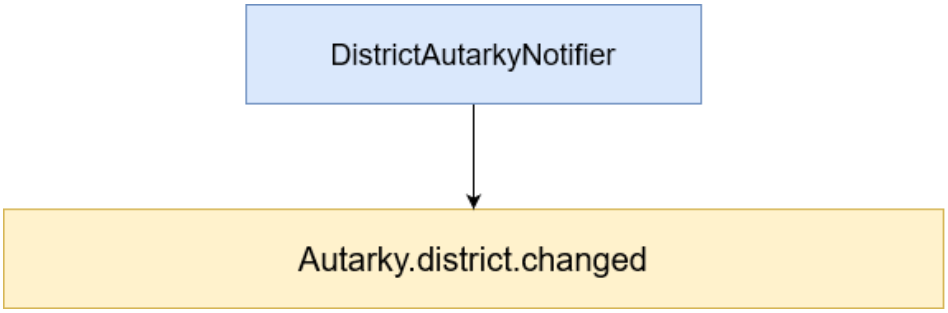
Car.charger.authorization
Événement permettant de d'autoriser ou de programmer le re-chargement d'une voiture électrique.



Autarky.client.changed & Autarky.district.changed

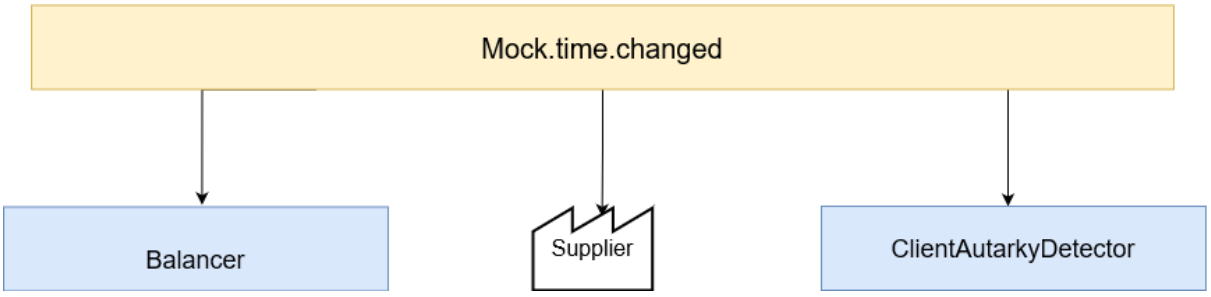
Ces topics servent à envoyer les informations sur les changement de statut d'autarky pour chaque client ainsi que pour chaque district





Mock.time.changed

Afin de faire avancer certain service dans le temps on envoie des messages sur ce topic



Contraintes et limites

Dans le cadre de notre projet, nous avons rencontré des difficultés lors de la migration vers une architecture orientée événement car il a fallu repenser les services et leur utilisation. Nous avons découpé nos services finement pour séparer au mieux les responsabilités, cependant avec ce choix, nous avons aussi amené de la complexité sur certains aspects du métier comme les services de contrats ou les services d'agrégation qui auraient pu être regroupés.
En effet, en multipliant les services on arrive à une architecture "spaghettis" du côté du contexte du client ce qui peut alourdir la compréhension du code lors du développement.
Si le projet venait à se complexifier nous aurions pu rajouter un deuxième bus d'événements centré autour de la gestion globale des communautés pour offrir plus de valeurs aux données de Charles en produisant plusieurs services de statistiques plus poussés.
Ces services viendraient se greffer à la place du service Analytics et empêcherait le Single Point of Failure sur ce service qui est actuellement présent dans notre architecture.

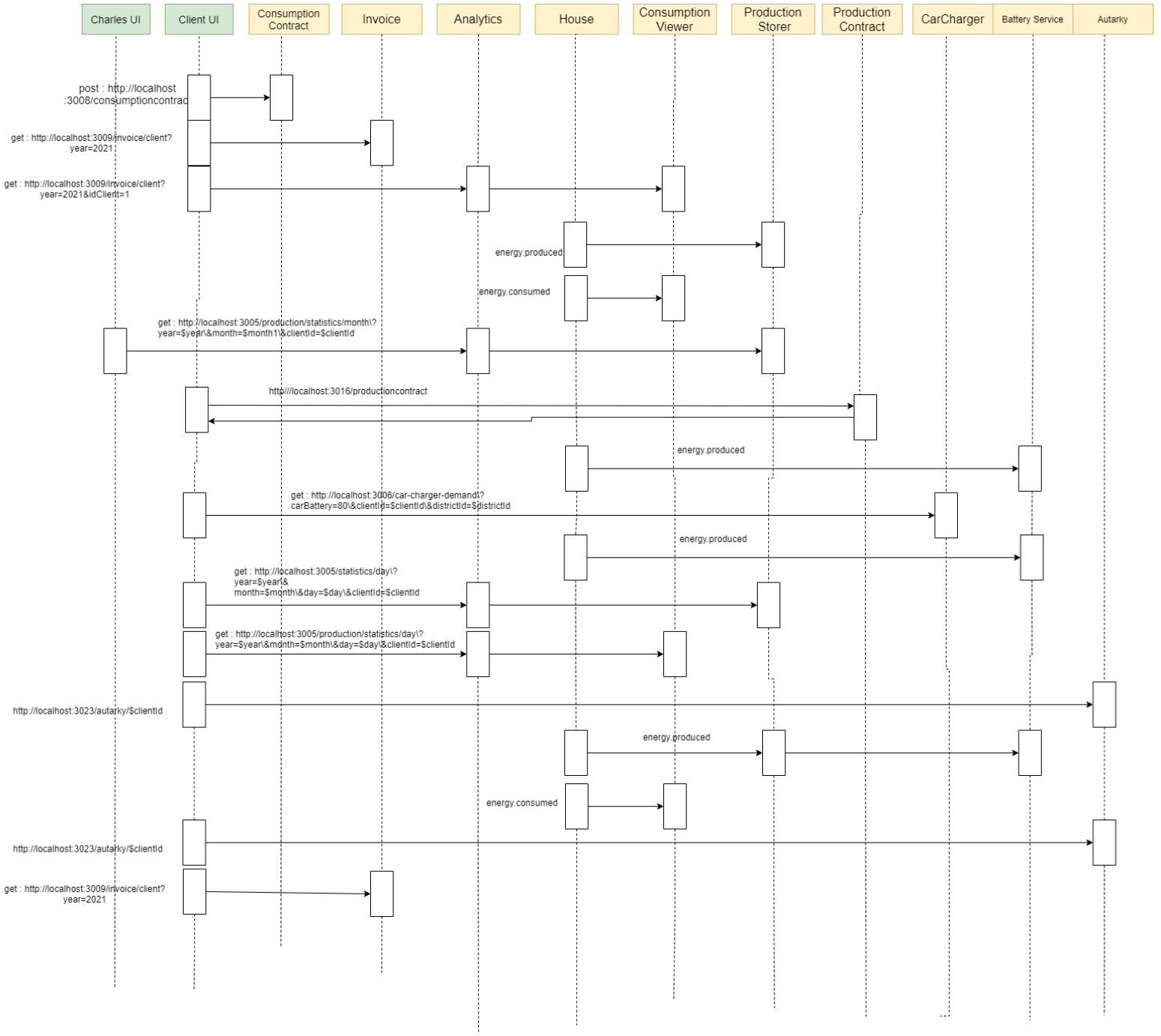
La configuration de notre bus kafka est minimal mais offre les fonctionnalités voulues. Mais l'utilisation conjointe avec NestJS nous a posé quelques problèmes. Nous ne pouvions pas utiliser toutes les fonctionnalités comme les Ktable ou les Topic.Compressed . Ceci nous a contraint à ajouter des services commes des aggregators, qui sont plus sensibles à de fortes charges.
Le mock de données nous a aussi forcé à repenser une partie de l'architecture dans les dernières semaines du projet. Rendant le système plus instable, face à des problèmes aléatoires.(synchronisation, charge importante de données, longueur du mock …)

Scénarios

Scénario 1) Pierre & Marie adorent la technologie et n'hésitent jamais à rajouter de nouveaux devices chez eux.

- Ils consultent leurs factures trimestrielles, en se connectant sur la UI du client grâce à leurs Id Client et ils remarquent que leur consommation a trop augmenté. (US12 et US1)
- Ils décident d'installer des panneaux solaires. (US8) Après un mois, Charles remarque que P&M ont un excès de production
- ils leurs propose un contrat de production (US3 et US9)
- P&M décident d'acheter une voiture électrique. Après avoir fait des courses, Ils rentrent chez eux à 19h et ils branchent leur voiture sur le « super chargeur ».
- Charles qui surveille la grille, remarque que la batterie de la voiture est à 80% et que la grille est surchargée, et du coup il bloque le super chargeur et il planifie son déblocage en fin de soirée. (US7)
- Thomas, propose à P&M son produit, et il les a convaincus de stocker une partie de leur énergie sur des batteries. Maintenant P&M stockent de l'énergie sur la batterie (US15).
- P&M consultent leur compte, et il remarque que pour aujourd'hui, ils ont plus de consommation que de production, du coup ils utilisent l'énergie stocké sur la batterie pour parvenir à l'autarcie (US 11).
- Ce mois, c'est l'anniversaire de Marie, la fille de P&M, et ils veulent voir combien ils vont devoir payer pour leur énergie consommée du mois en cours, afin qu'ils puissent ajuster leur budget au plus près possible et préparer une fête pour l'anniversaire de Marie (US13)

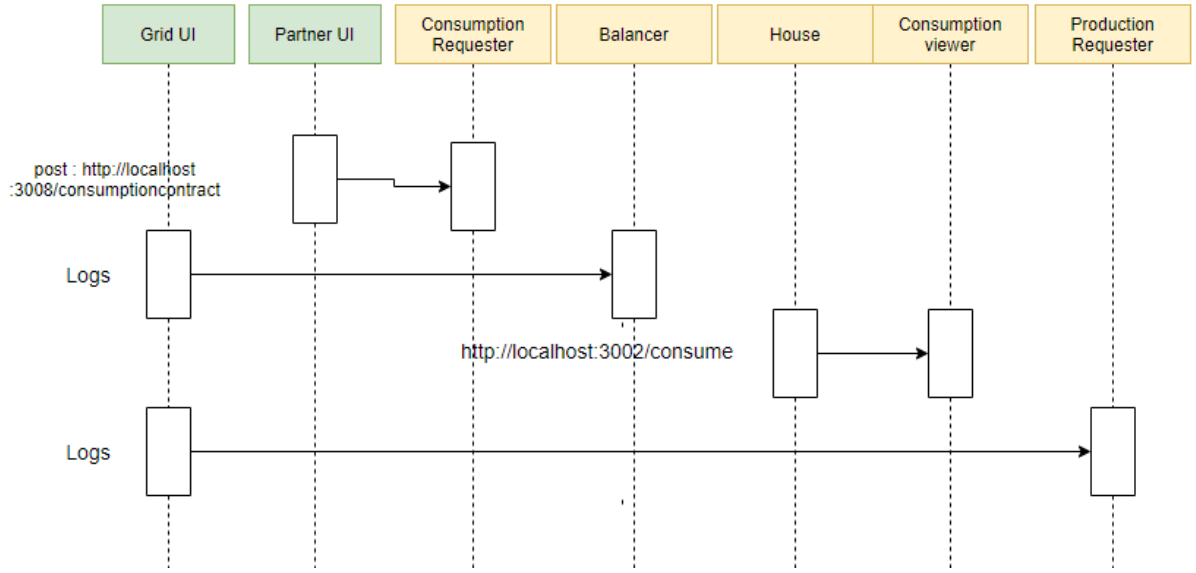
Diagramme de séquence du scénario 1 :



Scénario 2)

- Elon est en district X, il prévient d'un pic de consommation d'électricité pour 22h.(US4)
- A 21h, Nikolas remarque sur la balance que la production du district X a parvenu 80% de la limite de production (US5 et US2),
- Il décide de bloquer les consommations non essentielles (US10).
- Après 1h, le district Y parvient à l'autarcie et du coup Nikolas débloque les consommations non essentielles. (US14)

Diagramme de séquence du Scénario 2 :



Fonctionnalités non couvertes

Comme annoncé dans les points précédents, nous aurions pu développer davantage les fonctionnalités liées à Charles et au monitoring des données de quartiers. Pour ce faire nous aurions pu créer un service Communities qui réaliserait des agrégats quotidiens ou heure par heure sur la consommation, la production d'électricité des quartiers.
Cela aurait permis de traiter plus facilement et plus rapidement les données sans avoir à les recalculer à chaque appel et à passer par plusieurs services.
Nous aurions pu développer les services de statistiques afin de proposer des fonctionnalités de prédiction. Par exemple, au lieu de simplement observer un déséquilibre entre la production et la consommation nous aurions pu le prévoir dans certains quartiers et prévenir les fournisseurs avant. En effet, en rassemblant et en comparant les consommations des mêmes périodes aux précédentes années, on pourrait les estimer à l'avance.