ARCHITECTURE REPORT

Service Oriented Architecture

Last updated on 07/11/21

João **BRILHANTE**
Quentin **LAROSE**
Ludovic **MARTI**
Enzo **BRIZIARELLI**
Charly **DUCROCQ**

UNIVERSITÉ CÔTE D'AZUR | POLYTECH NICE-SOPHIA

Table of contents

I. Introduction

1. Personas

- Nikola, the Chief Operating Officer responsible of the grid stability from an Energy Supplier.
- Charles, the CEO of Smartrix Grid.
- Pierre & Marie, technological-driven family house owners, Smartrix Grid customers.
- Elon, the Chief Technological Officer of a new-gen electric vehicle company.

2. User stories

Global user stories

1. **Household consumption**
   As Pierre & Marie,
   I want to see the electricity consumption of my household,
   In order to have a better understanding of what is consuming most of my energy and when.

2. **Production scalability**
   As Nikola,
   I need to guarantee that the production is equal to the consumption on the whole grid,
   In order for the grid to not go dark.

3. **Global consumption**
   As Charles,
   I want to have an overview of my clients' usage of energy,
   In order to assess their potential needs and adapt my commercial propositions.

4. **Electric vehicles**
   As Elon,
   I want my vehicles to be able to recharge overnight,
   So that my customers are able to drive to work each morning.

5. **Community management**
   As Charles,
   I want to have communities of households,
   So that the energy can be managed in a more local manner (local production, balance of global production of the community, ...).

6. **Consumptions peaks**
   As Nikola,
   I need to guarantee that there is never an energy consumption peak too large in a localized area,
   So that a specific part of the grid is never overloaded.

7. **Electric vehicles charge scheduling**
   As Charles,
   I want to be able to schedule the charge of the electric vehicle of my clients,
   So that the vehicles are not necessarily charging as soon as they are plugged-in, allowing for a smoother load on the grid.

8. **Solar panel handling**
   As Albert,
   I want to install solar panels on the roofs of residential households,
   So that they can provide a source of energy production during the day

9. **Household production purchasing**
   As Nikola,
   I want to be able to buy the excess of local production in residential households,
   So that the local production is not wasted in case the household is consuming less than what their local production is able to produce

10. **Household consumption restriction**
    As Charles,
    I want to have a control on non-essential electric consumption inside the residential areas,
    so that I can reduce the global consumption on the grid.

11. **Personal autarky reached**
    As Pierre & Marie,
    I want to be able to know if I have reached autarky (self-sufficiency regarding energy production/consumption),
    so that I can adjust my consumption in case I am not yet self-sufficient.

12. **Invoice overview**
    As Pierre & Marie,
    I want to be able to retrieve my monthly invoices of my energy consumption for the past year,
    so that I can get a big picture of how much I paid, and potentially how much I saved through Smartrix Grid.

13. **Invoice generation**
    As Pierre & Marie,
    I want to see how much I will have to pay for my energy consumption for the current month,
    so that I can adjust my budget as closely as possible.

14. **Community autarky**
    As Charles,
    I want to be able to see if a community of users has reached autarky,
    so that I can reason about the community as a self-contained part of the grid and adapt Smartrix business strategies.

15. Household battery

As Thomas,
I want to provide battery setups to Smartrix Grid customers with solar panels,
so that they can store the surplus of energy production, and use it later, helping them reach autarky.

16. Scalability

As Charles,
I want to have the capacity to handle a large number of households and communities (at least a few thousands),
so that my company can achieve the desired growth.

Chosen user stories

19. Limited Power Plant

As Nikola,
I want the system to adapt to a limited production capacity (thus not infinite)
in order to match the real-world production sites which are limited in size and in power.

20. Virtual Power Plant

As Pierre & Marie,
I want to be notified about the autarky state changes (reaching autarky, or not being in autarky anymore) of my household and my community,
so that I can be aware of my installation's status.

Non-implemented user stories

17. Virtual Power Plant

As Charles,
I want to aggregate the energy production and storage of my customers to build a Virtual Power Plant (VPP),
so that I have the capacity to trade the excess energy production on the global energy market.

18. Google is good example

As Charles,
I want to sell the device consumption/production data to partners
(example: x€ / 1000 datapoints) stored in Smartrix systems,
so that I can generate more revenue from the data collected, in accordance with privacy policies.

21. Virtual Power Plant

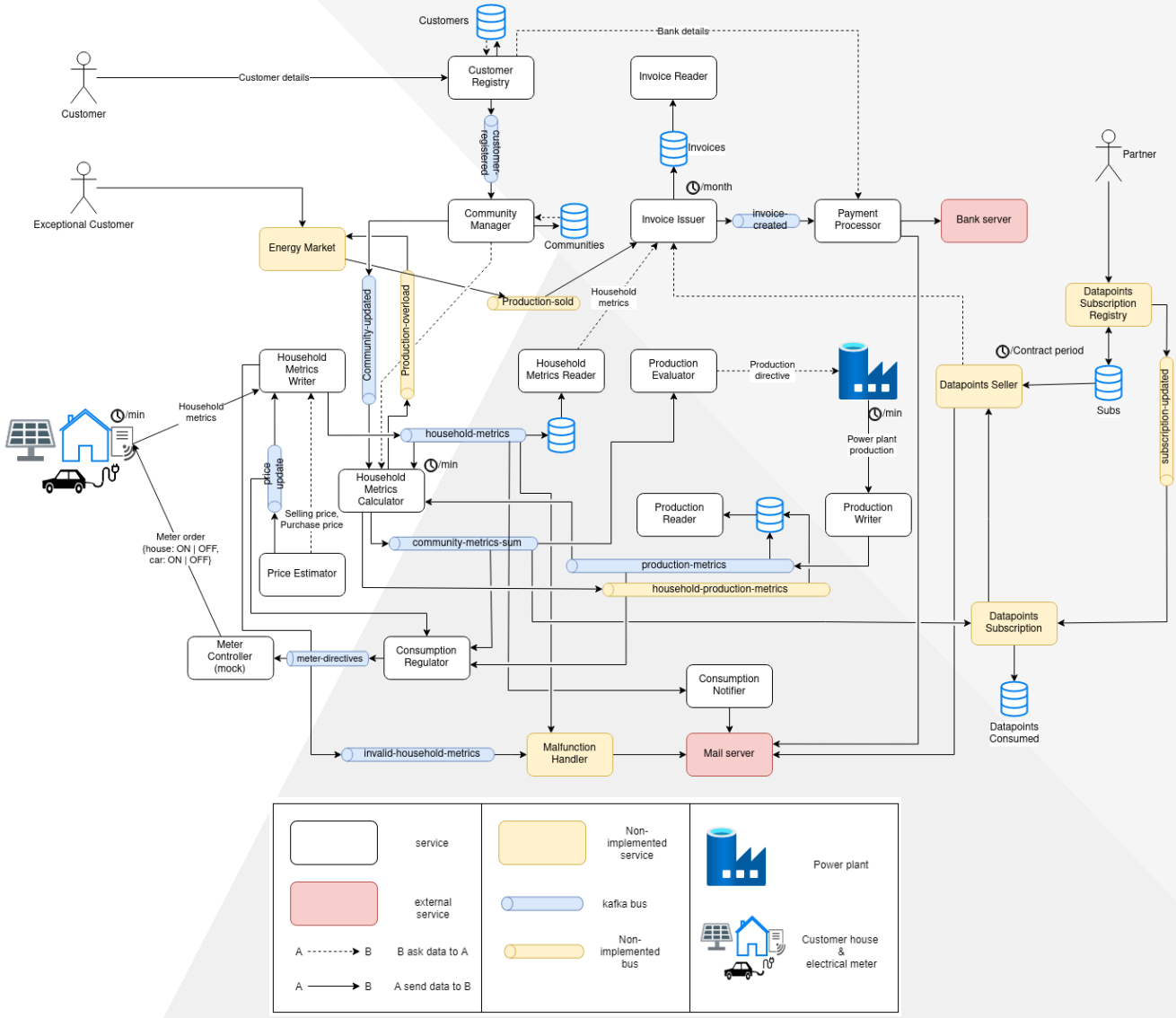
As Charles,
I want to monitor the data collection of the households and communities and be alerted of a malfunction,
so that I can send a technician to diagnose and resolve the issue.

3. Hypothesis

- Each house is equipped with an electricity meter that sends a request every minute to the community terminal, will transmit the electricity consumption and production information since the last measurement. The community terminal will transmit this information to the system.
- The information sent by the meters includes the energy consumed and produced (solar panels), as well as the battery level and its evolution (limited to battery owners).
- The meter has cables that can be connected to day/night switches. These switches allow to manage the "off-peak" cycles: this corresponds to the time of the day when consumption is the lowest and when it is wiser to put cars and batteries on charge or to run electrical appliances that do not need to run 24-hour.
- Meters also offer an API to send directives (such as informing "off-peak" times, order some appliances to be put on standby, etc...) and request charge/discharge for batteries and electric cars.
- A household will always draw on its own production resources (solar panels) before buying energy from the system.
- The power plant will ask for an estimate of the energy to be produced and send its electrical production to the system every minute. The power plant will always communicate its actual production. Furthermore, the power-plant limit and manage itself.
- The grid use batteries to store the excess energy produced, to be used or sold to third party consumers on the global energy market.
- Partners can subscribe to the system to receive data of consumption and production by email, at regular intervals defined in the subscription. An invoice will be generated every month.
- We consider that an electricity meter is malfunctioning when it sends incorrect data or send nothing at all for a long period of time.

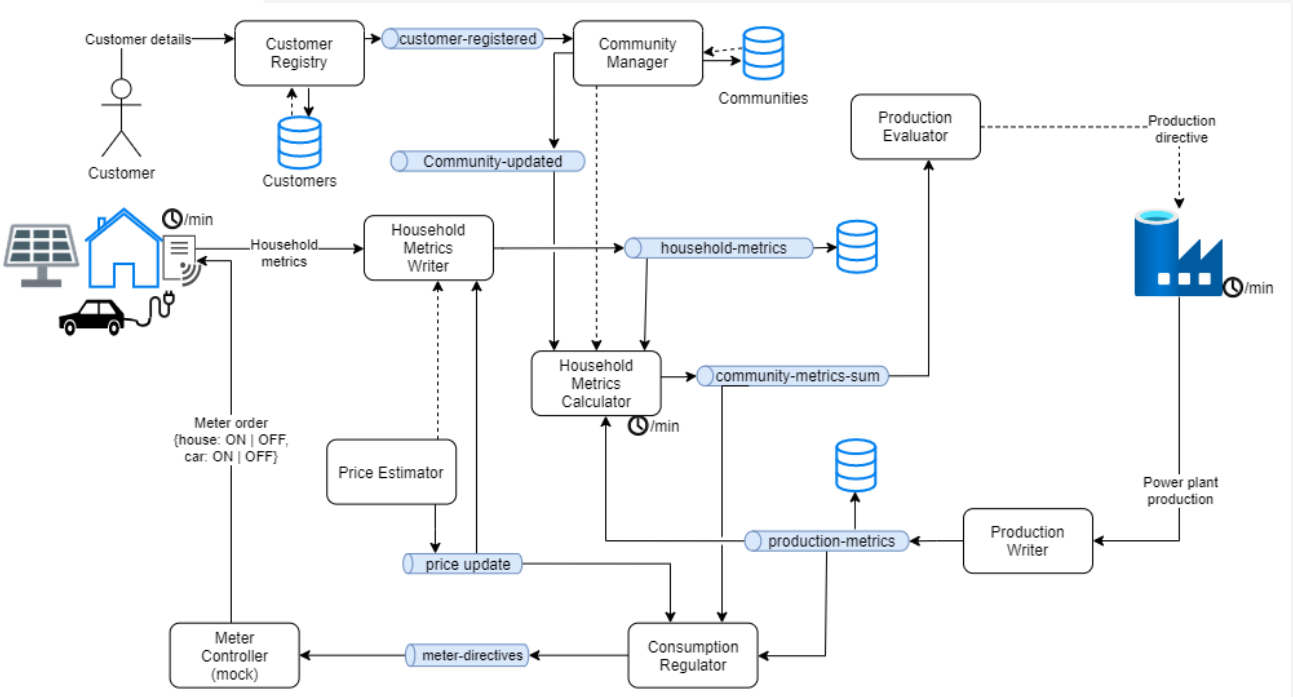
II. Architecture

1. Diagram



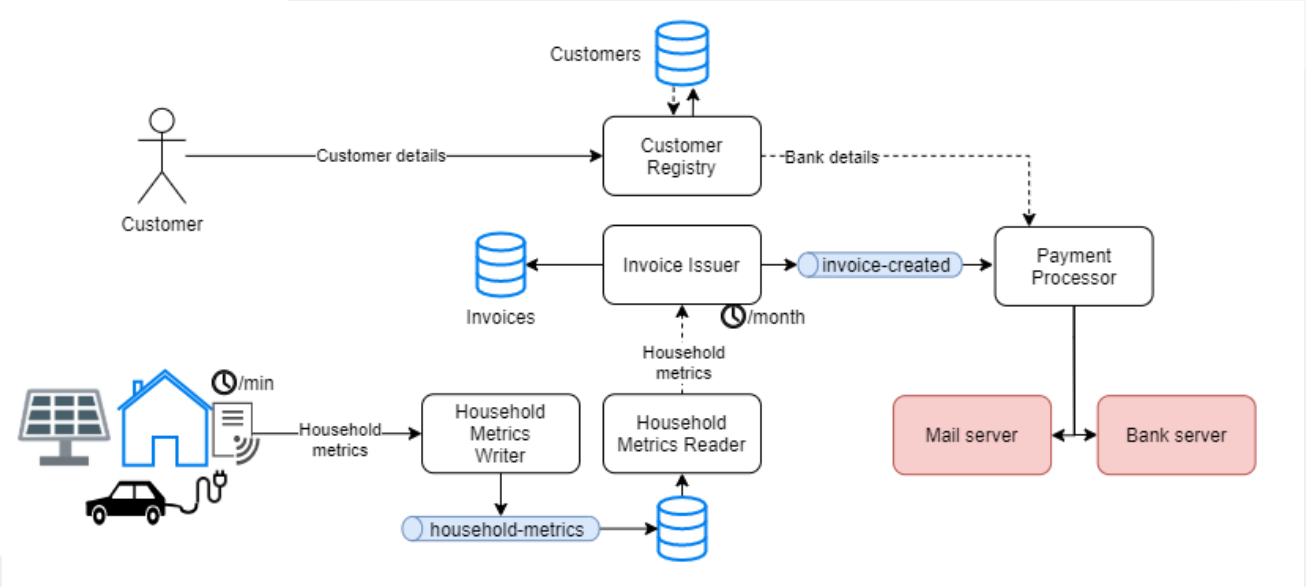
This first diagram can prove to be difficult to read, so in order to simplify its comprehension we made diagrams of each use case:

a. Regulation architecture



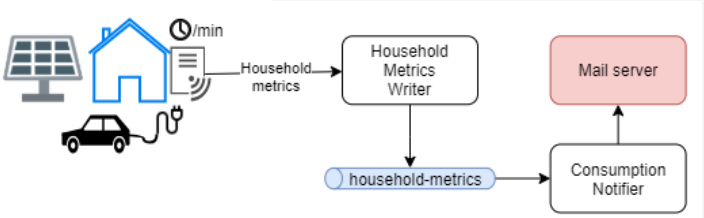
To regulate the consumption, the architecture shows how we collect data sent by the meters, propagate them in the other services through buses and save them in databases. Then we process them to perform a regulation, based on the consumption of each community in one minute and on the production of the same minute. Thus, the consumption and production of the next minute will be regulated based on the previous regulation.

b. Invoice creation architecture



An invoice must be generated each month for each customer. The diagram above shows how the customer's consumption and production data will be collected in order to generate an invoice that will be automatically paid and sent by email to the customer.

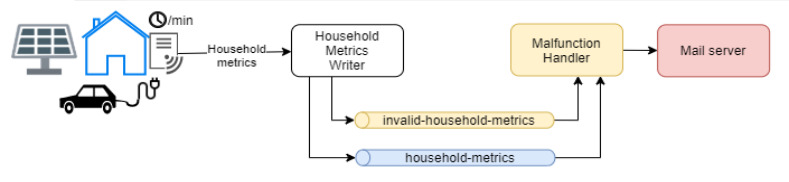
c. Autarky alert



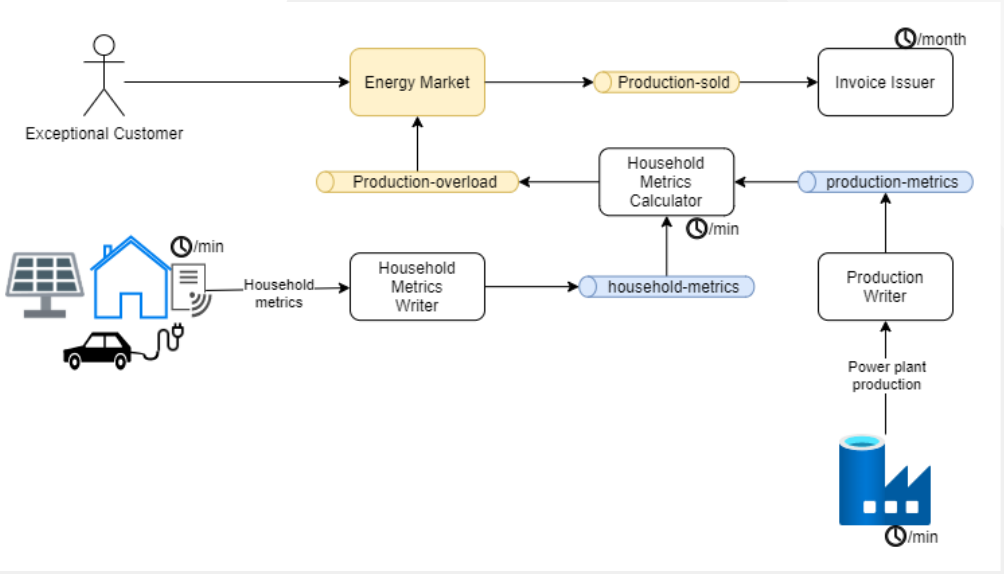
This part here shows how a client will be alerted if it leaves (or enters) a state of autarky.

d. Malfunction alert

To detect and alert a technician of grid malfunctions, a handler will react immediately to events in a specific topic in the bus (invalid).

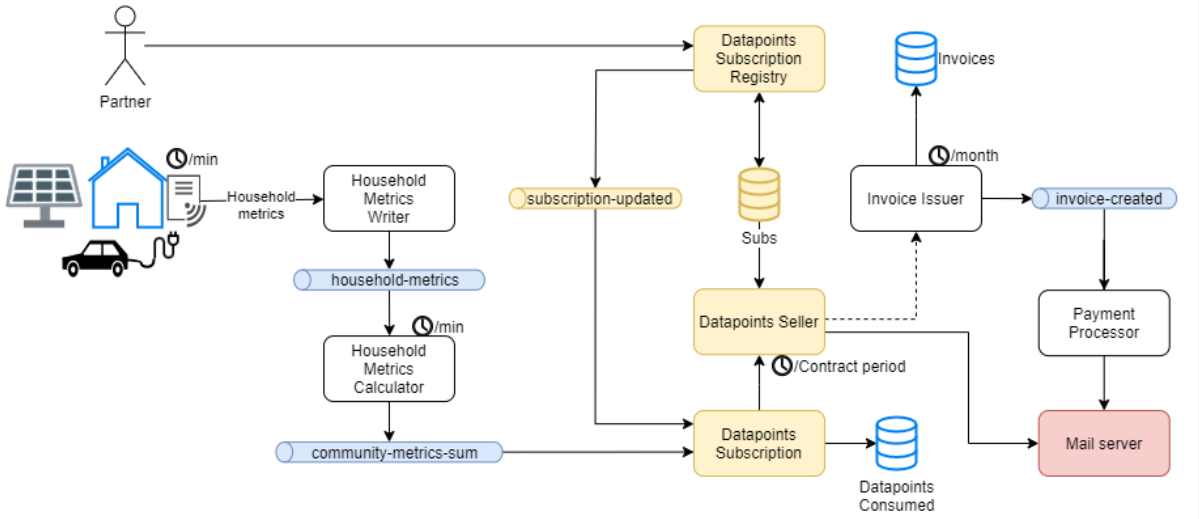


e. Global market & exceptional selling



One of the non-implemented user stories requires selling the surplus production to the global market. The above architecture will answer that concern, showing how the surplus is deducted and offered to the market.

f. Datapoint selling architecture

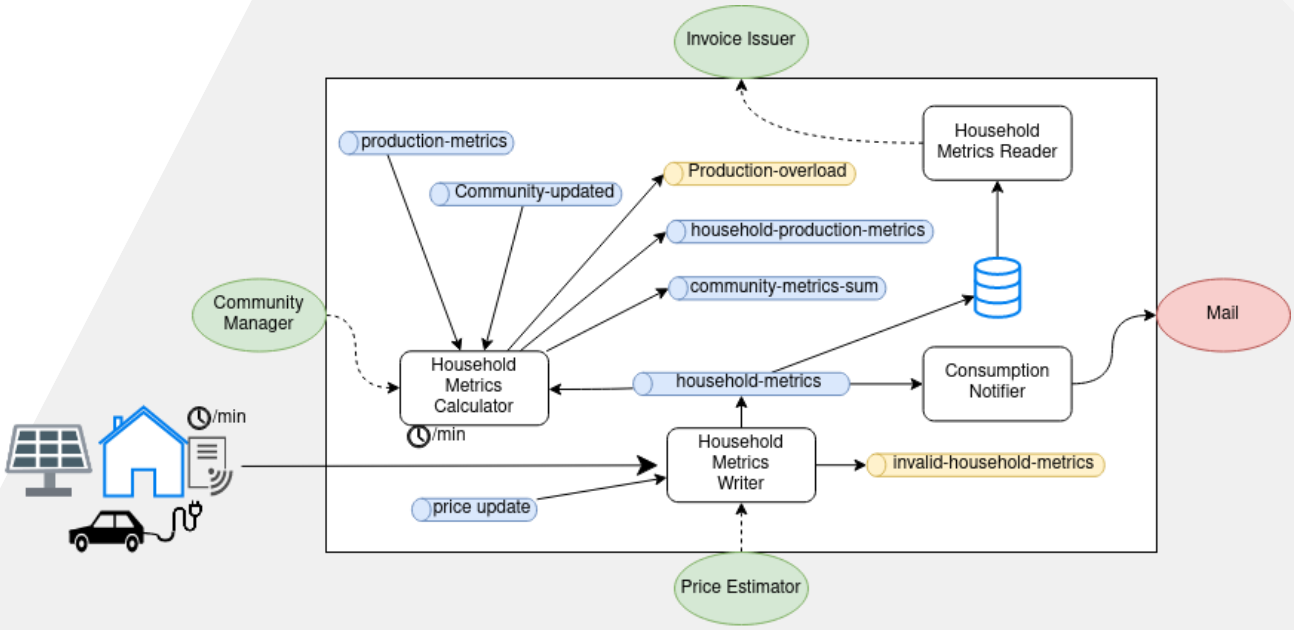


Another one of the non-implemented user stories is about selling datapoints. This involves registering subscriptions of third-party companies, collecting, and sending the data, as well as recording the transactions to invoices (keep financial history).

2. Choices

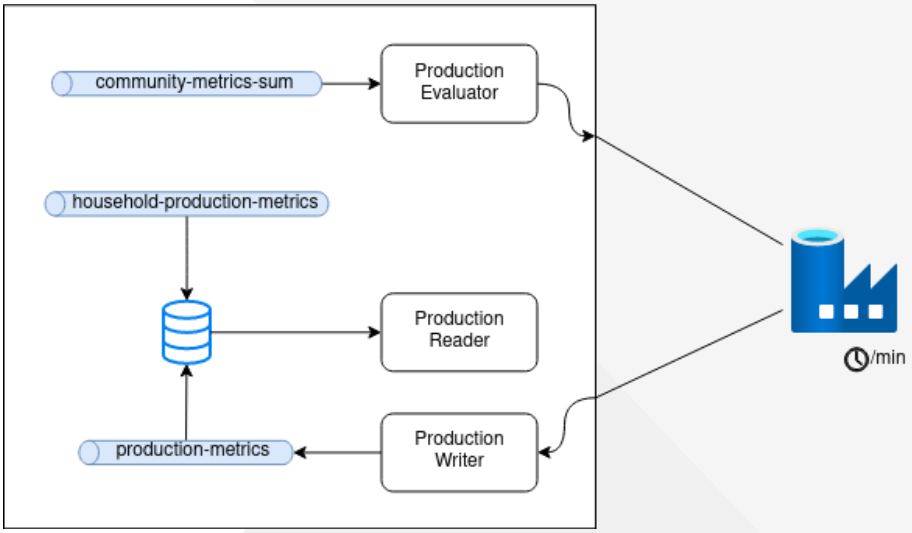
g. Metrics writer, reader, calculator, and consumption notifier

Our first concern was being able to manage large number of calls from the meters, to not overload our services or give them too many responsibilities. To achieve this, we decided to separate the metrics reading and writing responsibilities into two distinct services: *household metrics reader* and *household metrics writer*.



h. Production writer, reader, and evaluator

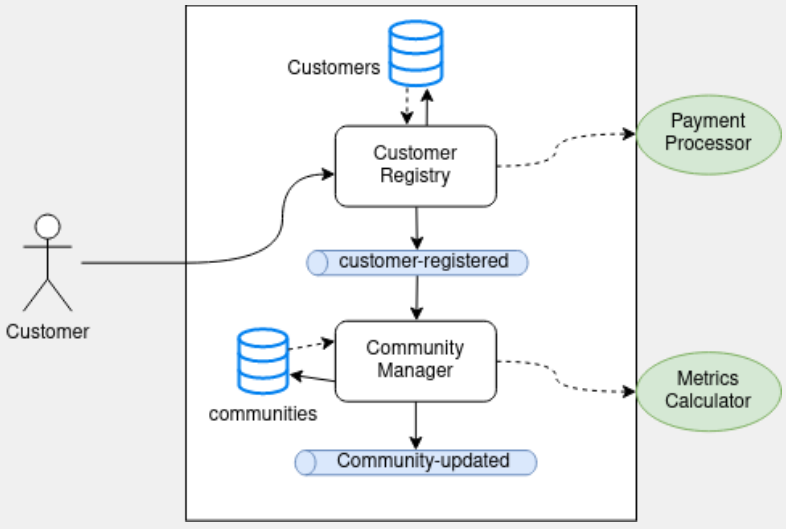
In the same way, we needed similar separation of services for the production (read / write). We can also predict many calls to manage. Thus, the *production writer* and the *production reader* were defined. To stabilize production (according to consumption) we also needed a service capable of estimating the production needed for the next hour: *production evaluator*.



i. Customer registry and community manager

Our system should be able to produce invoices and process payments. Therefore, we made a *customer registry* service to allow customers to register and provide their banking information and mail address.

To divide the network into several unique communities, we implemented the *community manager* service responsible for dividing the electric meters into communities, based on their locality. (A given meter can only be present in 1 community at a given time).



j. Consumption regulator and meter controller

Considering the energy load of connected cars, our strategy evolved over the weeks. At first, we thought that this feature depended only on the *production evaluator* service that should estimate the consumption of these cars during the night.

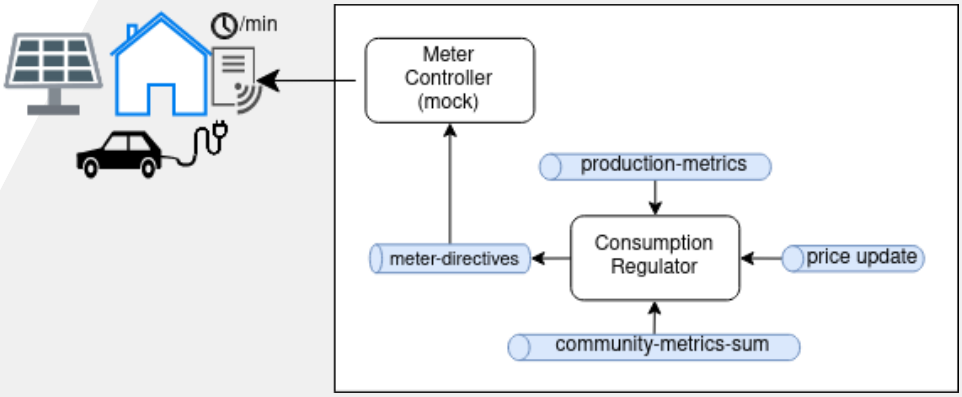
Then, we considered the charging stations of these cars as meters on their own, controlled by an external service that provides an interface for us to use in order to authorize (or not) the consumption. These changes followed the user story 7.

Finally, we learned more about modern's electricity meters, especially about day/night switches and the "off-peak" principle. Theses switches offers 3 modes: ON/OFF/AUTO, that would allow our service to take the correct directive corresponding to the needs of the network. Therefore, we decided to assume that a car charging station will be linked to the meter by one of these switches, thus we could control the consumption of the charging station and potentially other devices that can be added in the future.

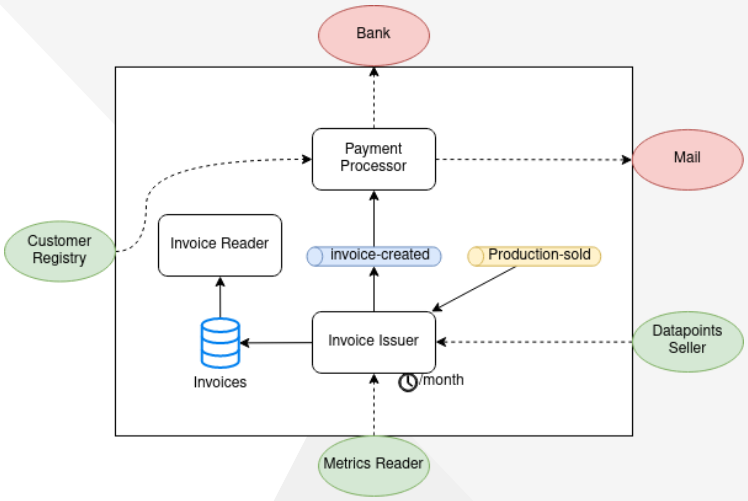
Because of this, we created the *consumption regulator* service responsible of managing the states of all switches for all meters, per community. We also created a *meter-controller* service that will sends directives to the meters. Calculation and sending responsibilities are separated if we want to parallelize the work by community (while the directives of community 2 are being calculated, ready directives of community 1 will be sent).

The *meter controller* service is obviously mocked because simulation of working meters (with RPC communications) didn't add that much value.

The addition of a battery per customer led us to consider price variations. Indeed, below a certain amount, we will ask the battery to charge. Otherwise, we will ask them to discharge.

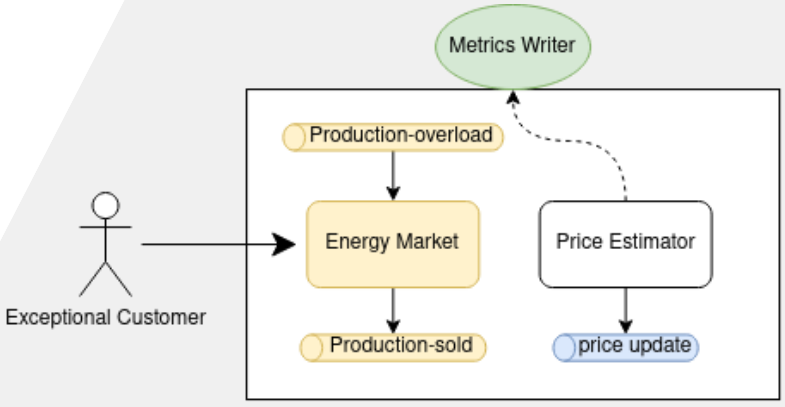


k. Invoice issuer, reader, and payment processor



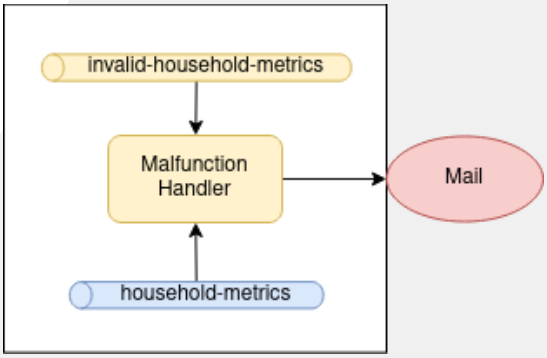
To generate invoices, we created an *invoice issuer* service that generates an invoice per customer every month. The *invoice reader* service will allow customers to access these invoices. Once an invoice is generated, the payment will be automatically made by the *payment processor* service with the bank information given by the *customer registry* service.

l. Energy market and price estimator



To allow spontaneous purchases in the global market, we set up an *energy market* service responsible of selling surplus energy.

m. Malfunction handler



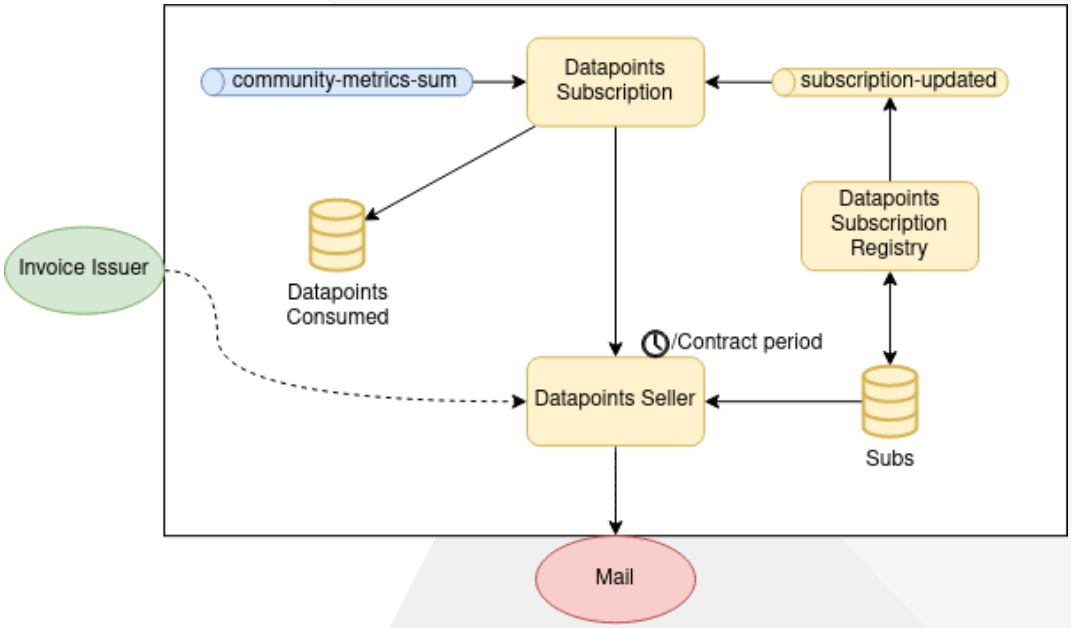
We have defined two types of malfunctions: absence of metrics and inconsistency in the metrics sent.

In order to manage these cases, we have set up the *malfunction handler* service which will send an email to the responsible technician.

To do this, it will listen and analyze the received *household metrics* to detect unexpected gaps. In

addition, it will look at potential format errors detected and shared by the *household metrics writer service* via the *invalid household metrics* bus.

n. Datapoint subscription, subscription registry and seller



As far as datapoints are concerned, we have divided subscriptions in which companies will subscribe in order to buy our datapoints. These subscriptions will be registered by the *datapoint subscription registry* service.

The datapoints will be collected by the *datapoint subscription* service which will transfer them to the datapoint seller service responsible for storing them in the database.

At regular intervals (defined by the subscription), the datapoint seller service will send an email to the subscriber with the data he has subscribed to.

And every month, the same service will be called by the invoice issuer service to know the datapoints to be invoiced.

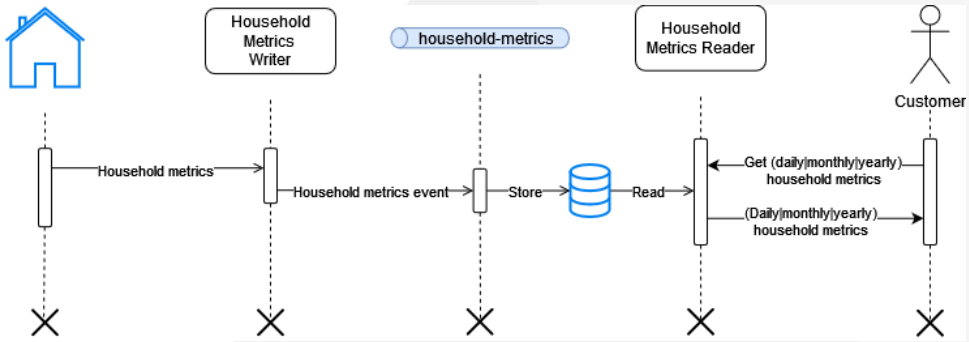
III. Scenarios

1. Household consumption

Given a customer named Roger,
And a few registered consumptions of Roger,
When Roger asks for a daily report of his consumption
Then he got a graph of his consumption of the day by hour

Explanation:

The point of this scenario is to prove the reading and writing capability of a consumption.



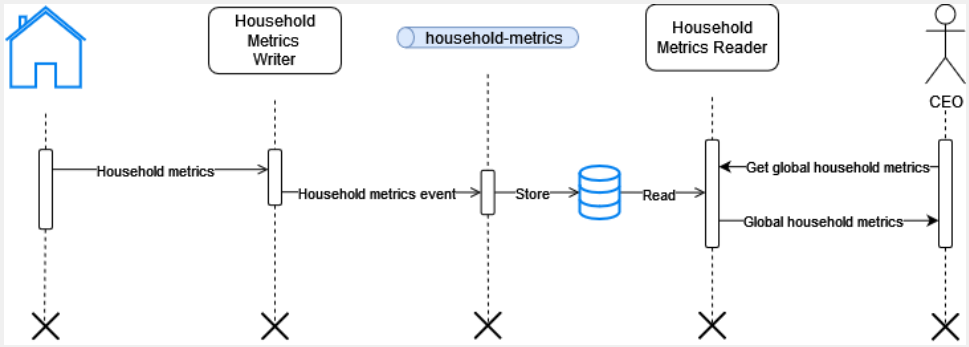
2. Global consumption

Scenario 1

Given a few registered consumptions during the month
When a global view of the consumption of the month is asked
Then a graph of the consumption of the month by days is returned

Explanation:

Like the first one, this scenario proves the reading and writing capability of all consumptions but in a larger scaling.



3. Production scalability

Scenario 1

Given a few registered consumptions during the month
And a production registered the last hour
When a production manager asks for the next production to make
Then he got an estimation of the needed production based on the last production and the recent consumption

Explanation:

The point of this scenario is to prove that the production advice is estimated based on previous measures. This scenario is hardly testable. We need to find a better way to test it.

(See sequence diagram below)

4. Consumption regulation

Scenario 1

Given a community

When the community increase his consumption in a way that production cannot follow
Then the community will be limited to avoid underloading

Explanation:

The point of this scenario is to prove that an abnormally high consumption will be detected to avoid local consumption peak.

(See sequence diagram below)

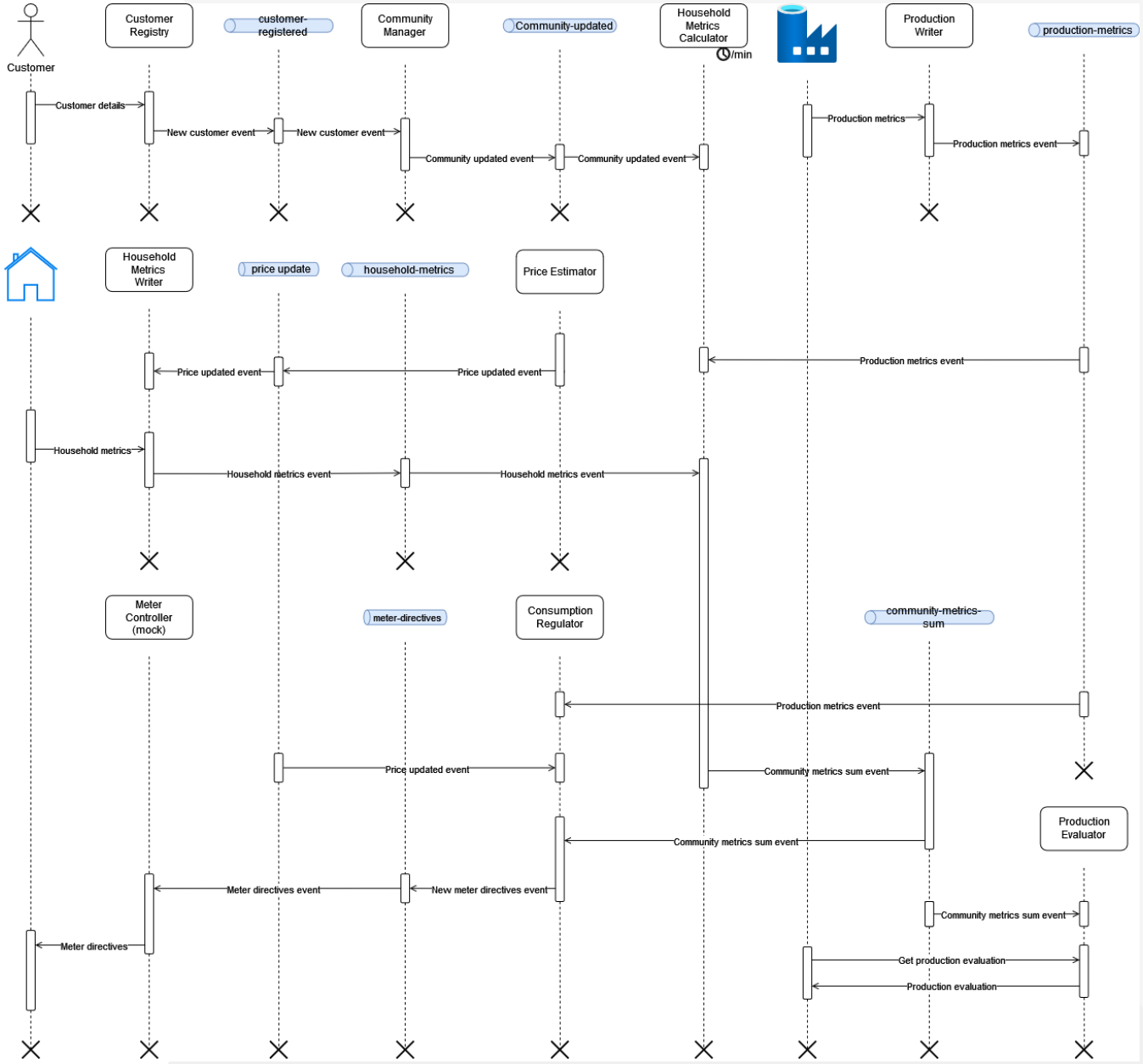
Scenario 2

Given a limited community

When the production is too high for the consumption
Then the community will be no longer limited to consume more and avoid overloading.

Explanation:

The point of this scenario is to prove that a small consumption will be detected to avoid production peak.



5. Electric vehicles charge scheduling

Scenario 1

Given a car A plug in the community X scheduled for the time slot 23h - 00h
And a car B plug in the community Y scheduled for the time slot 00h - 01h

When the car A is charging
Then the car B is not charging

When the car B is charging
Then the car A is not charging

When the night end
Then all cars are charged

Explanation:

The point of this scenario is to prove that all cars will be charged overnight but not at the same time.

(Same sequence diagram as above)

6. Battery regulation

Scenario 1

Given a customer who have a battery in his house

When the price of the electricity on the grid goes below the standard
Then the battery of the customer will be charging

When the price of the electricity on the grid goes above the standard
Then the battery of the customer will be discharging

Explanation:

The point of this scenario is to prove that battery will be charging only when it's worth for the customer, discharging otherwise.

(Same sequence diagram as above)

7. Invoice creation and access

Scenario 1

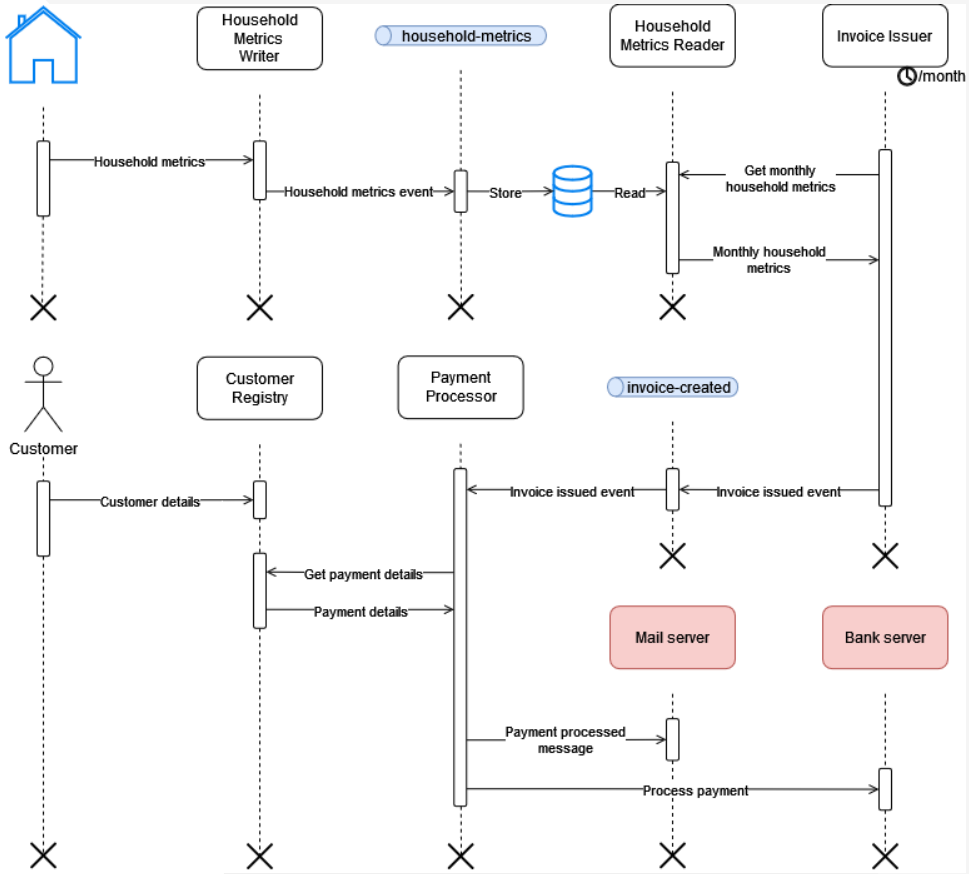
Given a customer

When the month is over,
Then an invoice for the customer's month consumption will be generated.

When the customer required his invoices
Then he get them.

Explanation:

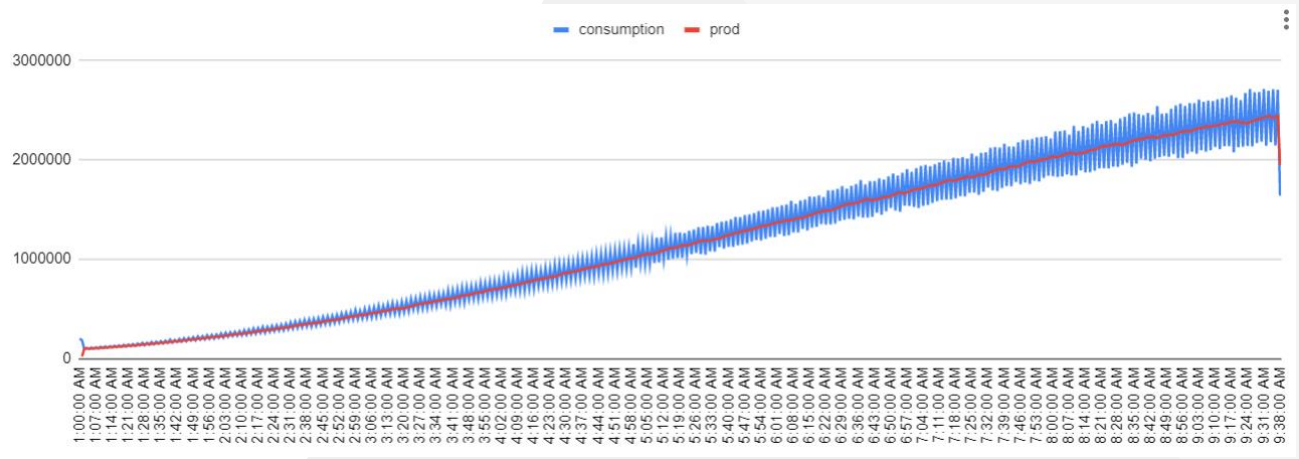
The point of this scenario is to prove that invoices can be generated and accessed.


Household Metrics Writer
household-metrics
Household Metrics Reader
Invoice Issuer
/month
Household metrics
Household metrics event
Store
Read
Get monthly household metrics
Monthly household metrics
Customer
Customer Registry
Payment Processor
invoice-created
Customer details
Invoice issued event
Invoice issued event
Get payment details
Payment details
Mail server
Bank server
Payment processed message
Process payment

IV. Simulator

To have a better visualization of our system's regulation, we developed a simulator that generate a *csv* file by taking the following parameters:

- nbCommunity: the number of communities
- nbHouseByCommunity: the number of customers by community
- averageDayConsumptionByCommunity: average consumption (in Wh) per community during the day
- averageNightConsumptionByCommunity: average consumption (in Wh) per community during the night
- startingProd: initial production
- startingHour: hour when simulation starts
- endingHour: hour when simulation ends



Results of a simulation.

The picture shows a simulation of consumption and production for 5 communities of 10 households at the beginning of the day. You can see the global electric consumption and production increasing at the morning.
The up and down spikes correspond to a delta variation of 10% per household. The production is following the consumption by asking an evaluation to the system at each revolution.

V. Understanding of the subject

1. Coverage of User Stories by our architecture

o. Overview

- **Consumption-writer** → User Story 1 and 3
- **Consumption-reader** → User Story 1 and 3
- **Production-writer** → User Story 2
- **Production-reader** → User Story 2
- **Production-evaluator** → User Story 2
- **Community-manager** → User Story 5, 6 and 7
- **Consumption-regulator** → User Story 4, 6 and 7

p. Explanations

First, through an overview of User Stories, we can see that responsibilities are distributed fairly.

Next, let's take a look at User Story 2, that requires the production be equal to the consumption. We needed to be able to "write a production", visualize it, then make sure that consumption and production were equal (by giving an instruction to produce a certain amount of energy to our power-plants).

Through this approach, we split this User Story into 3 different services, each with its own responsibility that can be used independently: *production-writer* to write the production(s), *production reader* to read the production(s), *production evaluator* to ensure that the production is equal to the consumption.

In a same way, we must make the consumption reading available for the customer, CEO and Chief Operating Officer. To achieve this, we set up 2 services, one allowing to read (*consumption reader*) and another to write (*consumption writer*). We must be able to read the total consumption and the consumption of each user, over a given period.

Finally, we implemented the *consumption regulator,* and the *community manager* services to regulate consumption. The *regulator* allows us to limit consumption in order to avoid a "blackout" in case of overconsumption. It's also linked with the *community manager* (who is in charge of dividing customers into communities) to avoid a "blackout" for a community.

As for the User Stories related to electric cars and Elon's persona, no service is directly responsible for them. They are covered by our choices of architecture, with the car charging station included in a customer's household.

VI. Event data models

household-metrics: HouseholdMetricsEvent

HouseholdMetricsEvent
customerId: number
consumption: number
boughtConsumption: number
production: number
soldProduction: number
purchasePrice: number
sellingPrice: number
date: Date [OPTIONAL]

communities-metrics-calculated: CommunityMetricsEvent[]

CommunityMetricsEvent
totalConsumption: number
totalBoughtConsumption: number
totalProduction: number
totalSoldProduction: number

Community
communityId: number
customersIds: number[ ]

household-production-metrics : ProductionMetrics
production-metrics : ProductionMetrics

ProductionMetrics
production: number
type: ProductionType
date: Date

ProductionType <<ENUM>>
HOUSEHOLD
POWER_PLANT

community-updated: Community[]

Community
communityId: number
customerIds: number[ ]

invoice-created: Invoice

Invoice
communityId: number
amount: number
firstDate: Date
lastDate: Date

meter-directives: MeterIndication

MetersIndication
communityId: number
customersIds: number[ ]
house: DirectiveStatus
cars: DirectiveStatus

DirectiveStatus <<ENUM>>
ON
OFF
AUTO

price-updated: Price

Price
price: number
date: Date [OPTIONAL]

VII. Limits, constraints, and potential improvement

1. Parallel contribution and dependence

The main problem resides in the relation of the parallel interdependent processing cycles. Indeed, a regulation of the consumption requires the whole consumption and production of the minute. This implies that each meter, each power plant, and each community meter must be perfectly synchronized. In reality, this is impossible to achieve.

This leads us to create mechanisms to "synchronize" the data.

For example, the calculation of community metrics will be done all the n-metrics received (with ‘n’ the number of customers), while keeping in RAM the last metrics of each customer. Rather than performing the processing every minute, this synchronization mechanism is easier to test and its result will be closer to reality. In return, RAM will be exploited (these services are stateful).

The same principle applies to the regulation which is started at each *community-metrics-calculated* event. So, if the production is not received before the metrics are calculated, the regulation will be based on the previous production.

2. Kafka Connect

Now, our databases are populated by our services. Our original idea (visible on our diagrams) was to use Kafka Connect (mechanism of Kafka) so buses write directly in the databases. We couldn't implement this due to lack of time, although this would have relieved the service's responsibilities of database writing. It would also have been possible to define rules to limit the update of the database (every hour for example) to avoid transaction lock.

3. Household metrics calculator: single point of failure

Currently, the *household metrics calculator* service has too many responsibilities. It produces a data set used by too many scenarios; therefore, it is not parallelizable. Thus, if this service fails, many scenarios are at risk. It would be possible to remove some of the load from this service by splitting its responsibilities into three separate services: one can be responsible of updating community metrics, another can manage client's productions and the last one can compute production excesses. That would grant us fault isolation, but it could imply consistency and complexity issues.

4. Simulation: improve it

Our simulator was designed late in the development, although it allowed us to visualize operations and correct bugs. It's an execution over time that goes through a large part of the architecture.

The current version only proposes to define average consumption per household for day and night but does not manage batteries or cars, nor does it simulate the behavior of a solar panel or set a production limit.

Adding these features would allow us to correct and adjust the system's behavior, as well as being able to simulate many more different use cases.