AmpVision

Adrián López González
Grado de Ingeniería Informática
Desarrollo web

Gregorio Robles Martínez
Santi Caballe Llobet

12/06/2020

FICHA DEL TRABAJO FINAL

Título del trabajo:	*AmpVision*
Nombre del autor:	*Adrián López González*
Nombre del consultor/a:	*Gregorio Robles Martínez*
Nombre del PRA:	*Santi Caballe Llobet*
Fecha de entrega (mm/aaaa):	06/2020
Titulación:	*Grado de Ingeniería Informática*
Área del Trabajo Final:	*Desarrollo Web*
Idioma del trabajo:	*Español*
Palabras clave	*IOT* *Analytics-Visualization* *React*

Resumen del Trabajo (máximo 250 palabras): *Con la finalidad, contexto de aplicación, metodología, resultados i conclusiones del trabajo.*

Durante los últimos años el incremento de dispositivos IOT[1] se ha visto incrementado y por ello el número de soluciones para poder interactuar con los datos obtenidos por este tipo de dispositivos. Estas soluciones comúnmente se categorizan por; ofrecer un producto (hardware) y una solución (software) ligada a este.

AmpVision busca ofrecer una solución alternativa ofreciendo un tipo de negocio basado en HaaS[2] y SaaS[3] y, además aportar una solución que facilite el desarrollo web ofreciendo componentes de alto rendimiento. Así pues, AmpVision ofrece 3 soluciones: AmpVision IOT, AmpVision y AmpVision UI.

Este trabajo se ha desarrollado siguiendo una metodología iterativa, donde se han separado las 3 partes del proyecto y tratado como únicas de manera que se puedan desarrollar de manera independiente para en el paso final poder unirlas.

Como resultado del trabo se ha obtenido un producto robusto y fiable el cual ofrece los 3 tipos de productos y cumple con creces con las expectativas puestas desde un inicio. Si bien es cierto que, para poder comercializar el producto es necesario un mayor trabajo, las bases de este son muy sólidas.

Como conclusión, el punto más difícil ha sido juntar los 2 mundos; hardware y software de manera que se pueda ofrecer un producto web que cumpla con las expectativas y que no trate de “reinventar la rueda”.

[1] **IOT** - Internet of things (Ver glosario para explicación).
[2] **HaaS** – Hardware as a Service (Ver glosario para explicación).
[3] **SaaS** – Software as a Service (Ver glosario para explicación).

Abstract (in English, 250 words or less):

During the last few years the increase of IOT devices has increased and therefore the number of solutions to be able to interact with the data obtained by this type of devices. These solutions are commonly categorized by; offering a product (hardware) and a solution (software) linked to it.

AmpVision seeks to offer an alternative solution by offering a type of business based on HaaS and SaaS and also to provide a solution that facilitates web development by offering high performance components. Therefore, AmpVision offers 3 solutions: AmpVision IOT, AmpVision and AmpVision UI.

This work has been developed following an iterative methodology, where the 3 parts of the project have been separated and treated as unique so that they can be developed independently and in the final step they can be joined together.

As a result of the work, a robust and reliable product has been obtained which offers all 3 types of products and more than meets the expectations set from the beginning. Although more work is needed to market the product, the foundations for this are very solid.

As a conclusion, the most difficult point has been to bring the 2 worlds together; hardware and software so that a web product can be offered that meets the expectations and that does not try to "reinvent the wheel".

Índice

Lista de figuras

1. Introducción

1.1 Contexto y justificación del Trabajo

Durante los últimos años, la demanda de dispositivos IOT se ha visto incrementado y por lo tanto el número de soluciones para poder interactuar con este tipo de dispositivos también. En el mercado existen multitud de dispositivos capaces de medir cualquier tipo de dato, pero, todos estos dispositivos se basan en x mediciones, es decir es necesario obtener x dispositivos para y mediciones.

De cara al usuario final el modelo de negocio de este tipo de servicios se basa en la compra del dispositivo y a continuación el acceso a una plataforma que en muchas ocasiones sigue un modelo freemium[4]. Plataformas como fitbit[5], ofrece este tipo de servicios la cual, ofrece dispositivos orientados a medir todos aquellos datos relacionados con el deporte y donde, es necesario comprar previamente el dispositivo para poder acceder a la plataforma la cual tiene secciones que deben ser obtenidas de forma independiente. Por otro lado, si el dispositivo queda dañado o queremos obtener una nueva versión deberemos comprar un nuevo dispositivo.

Así pues, nace la idea detrás de AmpVision la cual quiere romper este tipo de modelo ofreciendo una solución alternativa y un modelo de negocio basado en HaaS (Hardware as a Service) donde el cliente pueda tener la posibilidad de obtener cuando quiera cualquiera de los dispositivo ofrecidos y/o cambiarlo sin necesidad de volver a pagar y, además, tener acceso (compre o no algún dispositivo), a una plataforma SaaS (Software as a Service) donde tenga la posibilidad de conectar cualquier tipo de dispositivo y donde se puedan obtener las métricas de estos dispositivos y por lo tanto analizarlas y explotarlas sin necesidad de invertir en plataformas o sistemas de alto coste.

En definitiva, AmpVision busca ser una solución basada en el concepto de leasing[6] (solución que por el momento no se está explotando en el mundo IOT) y que, además, busca ofrecer dispositivos y un entorno que facilite la comunicación y la “explotación” de los datos de estos dispositivos IOT.

[4] **Freemium**: unión de las palabras “free” y “premium” para indicar que un servicio es gratuito, pero tiene partes de pago.
[5] https://www.fitbit.com/no/whyfitbit 09/06/2020, 12:50
[6] **Leasing**: arrendamiento.

1.2 Objetivos del Trabajo

Como se ha explicado en el punto anterior el objetivo es la creación de una plataforma HaaS y SaaS y que, además, sea capaz de comunicarse con cualquier dispositivo IOT.

Así pues, el proyecto se ha seccionado en 3 partes de forma que se puedan entregar tres productos los cuales están relacionados entre sí. Estos productos son: AmpVision IOT, AmpVision y AmpVision UI.

Con este proyecto se busca crear una plataforma donde se puedan crear dashboards[7] para analizar y monitorizar cualquier tipo de dato y dispositivo. Así mismo, busca crear un dispositivo al cual se le pueda conectar cualquier tipo de dispositivo IOT de manera que actúe como "middleware" de dispositivos IOT y sea este "middleware" el que se conecte con una plataforma de monitorización.

Por otro lado, también se busca crear una plataforma que provea de una librería la cual facilite el desarrollo de cualquier aplicación web y/o se pueda extender a aplicaciones de escritorio utilizando, por ejemplo, electron[8]. Esta plataforma, tiene que poder testear y analizar el rendimiento de manera sencilla, así como las regresiones de software si se decide cambiar ciertas partes del diseño.

Además, se han fijado los siguientes requisitos del proyecto:

Código	Descripción
F01	Diseñar librería de componentes en la plataforma Figma
F02	Convertir iconos de forma automática.
AMPU01	Programar los componentes diseñados en reactjs.
AMPU02	Los componentes tienen que poder adaptarse a requisitos externos (dimensiones, colores, etc)
AMPU03	Los componentes tienen que poder ser testeados de manera que se puedan evitar fallos de implementación en futuras revisiones de la librería.
AMPU04	Todos los componentes tienen que ser rápidos y no suponer un gran coste utilizarlos.
AMP01	Diseñar un api para conectar los dispositivos IOT con el backend.
AMP02	Crear un stack modular y que se pueda instalar en cualquier entorno de manera sencilla.
AMP03	Monitorizar y mostrar las mediciones de los dispositivos IOT.
AMP03	Automatizar la plataforma.
R01	El sistema debe ser robusto, pero con tiempos de respuesta rápidos.
R02	El sistema tiene que utilizar conexiones seguras bajo HTTPS.
R03	Ofrecer entornos de desarrollo y producción.

[7] **Dashboard:** representación gráfica de las principales métricas
[8] https://www.electronjs.org/ 09/06/2020, 17:38

1.3 Enfoque y método seguido

El proyecto se ha separado en 3 partes, las cuales son totalmente independientes de forma que, la planificación pueda ser modificada en función del tiempo y los problemas encontrados.
Como se ha especificado en los puntos anteriores se quiere desarrollar 3 productos; AmpVision IOT, AmpVision, AmpVision UI.

El proyecto se ha definido sobre dos premisas, la primera “divide y vencerás” donde se pretende seccionar el proyecto en pequeñas partes de forma que se facilite el desarrollo y la segunda, “no reinventar la rueda” es decir, si es posible utilizar software existente y adaptarlo siempre y cuando la solución cumpla los objetivos definidos, será mejor que desarrollar una solución entera desde 0.

Por otro lado, para desarrollar el proyecto se ha seguido una metodología de desarrollo iterativa, aunque con conceptos del desarrollo ágil como los tableros kanban[9].

[9] **Kanban:** tablero utilizado en el desarrollo de software ágil (Ver glosario para explicación).

1.4 Planificación del Trabajo

Nombre	Fecha Inicio	Fecha Fin	% completado
Iniciación y definición del proyecto			
Resumen del proyecto	20/2/2020	17/2/2020	100
Planificación del proyecto	20/2/2020	6/3/2020	100
Creación tableros Kanban	7/3/2020	8/3/2020	100
Preparación software/hardware base	9/3/2020	11/3/2020	100
Inicio del proyecto	11/3/2020	11/3/2020	100
AmpVision IOT devices			
Diseñar circuito basado en componentes prediseñados.	11/3/2020	25/3/2020	100
Conectar múltiples “remotes” a la “Base Station”	11/3/2020	11/3/2020	100
Imprimir carcasas 3D	11/3/2020	29/5/2020	100
Añadir componentes a una “Arduino board”	11/3/2020	15/3/2020	100
Conector “Base Station” a internet	11/3/2020	15/3/2020	100
Conectar multiples remotes a una mismo “Base Station”	11/3/2020	25/3/2020	100
AmpVision			
Crear API conexión IOT -> AmpVision	20/4/2020	20/4/2020	100
Añadir reverse proxy	20/4/2020	20/4/2020	100
Añadir servidor MQTT	20/4/2020	25/4/2020	100
Crear base de datos InfluxDB	20/4/2020	25/4/2020	100
Añadir agente que capture recoja los mensajes del servidor MQTT y los envíe a la base de datos	20/4/2020	25/4/2020	100
Obtener mediciones guardadas en la base de datos	10/5/2020	13/5/2020	100
Login y frontal web para presentar las mediciones	10/5/2020	13/5/2020	100
Testing & Security	20/4/2020	20/4/2020	100
AmpVision UI			
Diseñar componentes en Figma	11/3/2020	25/3/2020	100
Obtener iconos/colors de forma automática	26/3/2020	2/4/2020	100
*Programación componentes	6/4/2020	29/5/2020	30
Documentación componentes	6/4/2020	29/5/2020	100
Documentación librería y entorno	6/4/2020	29/5/2020	80
Live tests documentation	6/4/2020	29/5/2020	80
Live tests performance	6/4/2020	29/5/2020	100
Live tests accessibility	6/4/2020	29/5/2020	100
Security	6/4/2020	29/5/2020	100
Creación npm packages	29/5/2020	29/5/2020	100
Seguimiento del proyecto			
Entrega PEC 1	20/2/2020	6/3/2020	100
Informe de progreso (PEC2)	20/2/2020	10/4/2020	100
Entrega PEC2	9/3/2020	10/4/2020	100
Informe de progreso (PEC3)	20/2/2020	29/5/2020	100
Memoria	20/2/2020	12/6/2020	100
Entrega PEC4	1/6/2020	12/6/2020	100

* Se esperaba conseguir un total de unos 50 componentes, pero se ha terminado obteniendo unos 10 componentes, pero durante el desarrollo del proyecto se han añadido nuevos ítems que han completado y dado un mejor acabado al proyecto como por ejemplo el entorno de pruebas de los componentes.

1.5 Breve sumario de productos obtenidos

Plataforma orientada a la creación de dashboards para analizar y monitorizar cualquier tipo de dato y dispositivo. Desde datos de dispositivos IOT, recursos usados por un servidors, etc, pero además ofrece la posibilidad de poder llevar estos dashboards o los componentes para poder montar estos dashboards en otros entornos que no sean web, como por ejemplo aplicaciones móviles.

AmpVision ofrece 3 productos:

- **AmpVision IOT:** Hardware con la habilidad de monitorizar datos medioambientales tales como (luz, temperatura, humedad, bajo voltaje y presión barométrica).

- **AmpVision:** Entorno que ofrece un modelo basado en suscripción donde se puede configurar un entorno personalizado para poder monitorizar cualquier tipo de datos y conectarlo contra cualquier base de datos ya sea interna o externa. Entorno montado como microservicios lo que permite un escalado tanto horizontal como vertical.

- **AmpVision UI:** librería de componentes escritos puramente React para ser utilizados dentro del proyecto AMPVision pero suficientemente abierto como para ser utilizado en cualquier tipo de proyecto ya sea relacionado con AmpVision o no, además esta librería también ofrece un entorno donde poder documentar y testear todo componente ya sea de manera independiente o unida.

1.6 Breve descripción de los otros capítulos de la memoria

En los siguientes apartados se explicará el proyecto empezando por las tecnologías y metodologías utilizadas para desarrollar el proyecto.

Además, se ha separado cada apartado parte del proyecto de forma que se ha explicado cada apartado de forma independiente.

Así pues, tenemos:

- AmpVision IOT: explicación del desarrollo orientado al hardware que contiene el proyecto.
- AmpVision: explicación de la plataforma SaaS.
- AmpVision UI: explicación de la librería de componentes.

Además, también se ha incluido un apartado de seguridad puesto que, se ha intentado que el proyecto cumpla con los estándares de seguridad.

2. Componentes del proyecto

Arquitectura global del proyecto que muestra como cada parte del proyecto es independiente entre sí, pero, se pueden utilizar en conjunto.

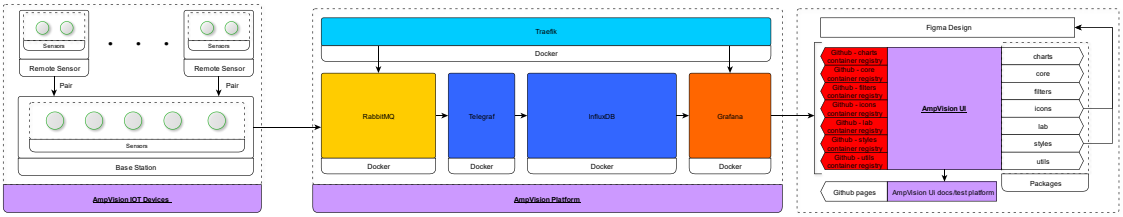

Ilustración 1 – Arquitectura global

2.1 Tecnologías

En la parte de los dispositivos IOT en principal lenguaje de programación que se ha utilizado ha sido **C/C++**.

El stack tecnológico AmpVision esta modularizado utilizando Docker[10], de manera que, cada pieza es totalmente independiente de las otras, aunque están interconectadas, pero no necesariamente tienen que estar en el mismo entorno, dicho de otro modo, la arquitectura diseñada permite tener cada pieza en diferentes sistemas. El stack tecnológico consta de:

- **Traefik**[11]: "*Edge Router"* que facilita la publicación de servicios. Recibe las solicitudes en nombre de su sistema y averigua qué componentes son los responsables de manejarlas.
- **RabbitMQ**[12]**:** software de negociación de mensajes de código abierto que funciona como un middleware de mensajería.
- **Telegraf**[13]**:** agente que recoge y reporta métricas.
- **InfluxDB**[14]**:** base de datos de series temporales diseñada para manejar altas cargas de escritura y consulta.
- **Grafana**[15]**:** software de código abierto de visualización y análisis. Permite consultar, visualizar, alertar y explorar métricas sin importar dónde estén almacenadas.

Además, dado que es un proyecto de desarrollo web se utilizarán los lenguajes de programación propios del desarrollo web:

- **Javascript**, **Typescript**
- **HTML/CSS**
- Etc

[10] **Docker:** contenedores, una unidad estandarizada de software (Ver glosario para explicación).
[11] https://docs.traefik.io/ 09/06/2020, 19:43
[12] https://www.rabbitmq.com/ 09/06/2020, 19:45
[13] https://docs.influxdata.com/telegraf/v1.14/ 09/06/2020, 19:47
[14] https://docs.influxdata.com/influxdb/v1.8/ 09/06/2020, 19:49
[15] https://grafana.com/docs/grafana/latest/ 09/06/2020, 19:50

Por otro lado, el entorno UI utiliza las siguientes tecnologías:

- **Jest**[16]**:** JavaScript testing framework mantenido por Facebook, Inc.
- **Enzyme**[17]**:** Utilidad para el framework React que facilita las pruebas con sus componentes.
- **Lerna**[18]**:** Una herramienta para gestionar proyectos de JavaScript con múltiples paquetes.

Además, dado que la liberia ui es un proyecto js el repositorio cuenta con un "set up" basado en:

- **Nodejs**[19]: es un entorno de servidores de código abierto escrito en javascript.
- **Npm/yarn**[20]**:** gestor de paquetes para el lenguaje de programación JavaScript.
- **Babel**[21]**:** cadena de herramientas que se utiliza principalmente para convertir el código ECMAScript 2015+ en una versión compatible con JavaScript.
- **Eslint**[22]**:** herramienta de análisis de código estático para identificar patrones problemáticos encontrados en el código JavaScript.
- **Prettier**[23]**:** herramienta que permite formatear el código de diferentes lenguajes de forma automática.
- **Commitlint**[24]**:** herramienta para ayudar a definir y siempre usar una estrategia de commits, la estrategia de commits utilizada en este proyecto es: https://github.com/conventional-changelog/commitlint

2.2 Metodologías de desarrollo de software aplicadas

- **TDD**: Test-driven development (TDD) es una práctica de ingeniería de software que involucra otras dos prácticas: Escribir las pruebas primero (Test First Development) y Refactorización (Refactoring).

- **Kanban**: sistema de gestión de proceso visual que le indica qué producir, cuándo producirlo, y cuánto producir.

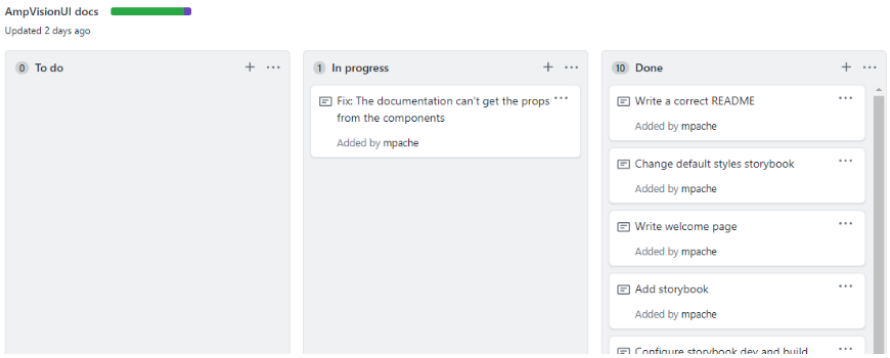

Ilustración 2 – Ejemplo de tablero Kanban

16 https://jestjs.io/docs/en/getting-started 09/06/2020, 19:53
17 https://github.com/enzymejs/enzyme 09/06/2020, 19:55
18 https://github.com/lerna/lerna 09/06/2020, 20:04
19 https://nodejs.org/en/docs/ 09/06/2020, 20:06
20 https://docs.npmjs.com/ https://classic.yarnpkg.com/en/docs/ 09/06/2020, 20:07
21 https://babeljs.io/docs/en/ 09/06/2020, 20:09
22 https://eslint.org/ 09/06/2020, 20:10
23 https://prettier.io/ 09/06/2020, 20:11
24 https://commitlint.js.org/#/ 09/06/2020, 20:13

2.3 Creación logotipo y marca

El proyecto mezcla los conceptos de “**amp**” y “**visión**”, por una parte “amp” o ampere, es la unidad base de corriente eléctrica en el Sistema Internacional de Unidades. Lleva el nombre de André-Marie Ampère[25], matemático y físico francés, considerado el padre de la electrodinámica.

Por otra parte, “visión” o visión es la capacidad de interpretar el entorno circundante utilizando la luz en el espectro visible reflejado por los objetos del entorno. Puesto que, la finalidad de la plataforma AmpVision es ofrecer un servicio donde visualizar las métricas obtenidas de los sensores, creemos que “vision” se acoplaba bastante bien.

Así nació AmpVision del cual puesto que son 2 conceptos unidos se decidió que el logo debía contener estos 2 conceptos en el logotipo.

Se desarrollo primeramente con los básicos colores negros y grises:

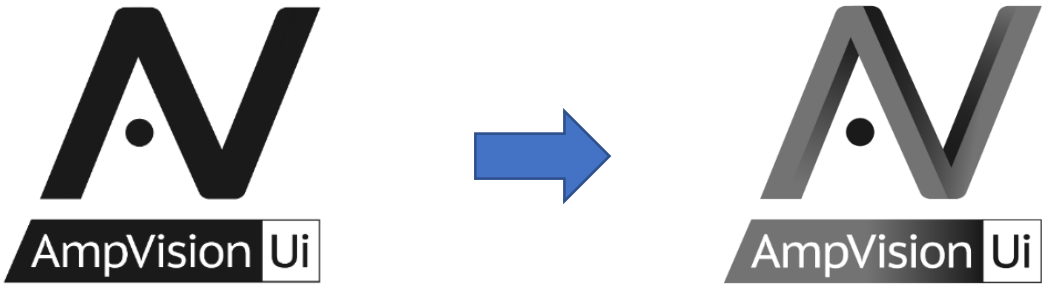

Ilustración 3 – Fase 1 y 2 del logotipo

Para terminar, obteniendo como el logotipo final:

Ilustración 4 – Logotipo final

[25] https://en.wikipedia.org/wiki/Andr%C3%A9-Marie_Amp%C3%A8re 09/06/2020, 20:18

2.4 AmpVision IOT

Repositorio Git (Privado)	https://github.com/mpache/AmpVision-iot

Hardware con la habilidad de monitorizar datos medioambientales tales como (luz, temperatura, humedad, bajo voltaje y presión barométrica).

Este hardware está separado en 2 partes:

- **Base Station:** interactúa con la plataforma ampVision o cualquier otra que se necesite utilizando un protocolo de mensajería como por ejemplo MQTT, AMQP, etc.
- **Remote:** Contiene los componentes y sensores que miden las diferentes métricas (humedad, temperatura, etc).

Ilustración 5 – Base Station y Remote (Caja negra)

Múltiples *Remotes* pueden ser conectados a una única *Base Station* de manera que el componente que interactúa con la plataforma AmpVision siempre será la estación base ya que esta necesita de conexión a internet, pero los sensores no deberían tener esta comunicación, se conectan a la estación base por medio de una comunicación por radio frecuencia.

2.4.1 Arquitectura

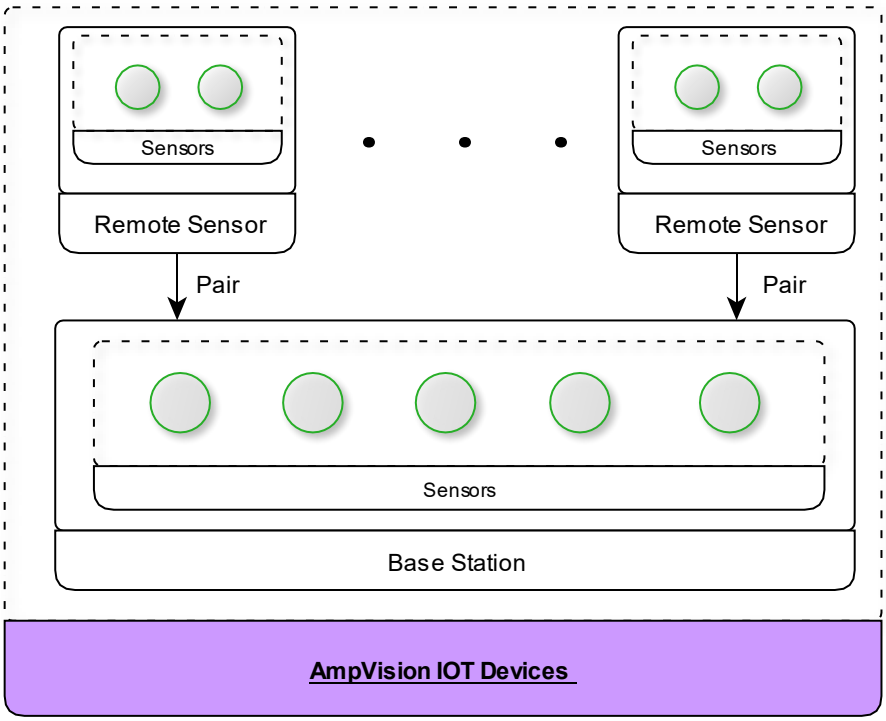

Ilustración 6 – Arquitectura IOT Devices

El componente encargado de conectarse con la plataforma AmpVision es la *Base Station* y aunque el encargado de obtener las mediciones del entorno es el *Remote*, la estación base puede contener algunos medidores para que el dispositivo no sea un simple middleware[26].

De esta manera tenemos los *remotes* los cuales son dispositivos que su finalidad es simplemente capturar mediciones y enviárselas a la estación base mediante wifi o radiofrecuencia.

Por otro lado, tenemos la estación base la cual se encarga de enviar todas las mediciones que reciba a la plataforma AmpVision mediante el protocolo MQTT el cual es un protocolo de red ligero de publicación y suscripción que transporta mensajes entre dispositivos.

Si por algún motivo el dispositivo no puede enviar las métricas el dispositivo mantendrá en cola las métricas y las enviará cuando sea posible.

[26] **Middleware:** programa informático que proporciona servicios a aplicaciones de software más allá de los disponibles en el sistema operativo.

2.4.2 Desarrollo

El primer paso del desarrollo ha sido escoger ciertos componentes o sensores los cuales se puedan acoplar a una placa Arduino[27], los sensores que se han añadido junto con sus especificaciones son los siguientes:

Sensor	**Especificaciones**
Sensor de humedad	- Response time 8s - Accuracy tolerance ± 3% - Hysteresis ≤ 1.5%
Sensor de presión	- Pressure range 300 - 1100 hPa - Relative accuracy ± 0.12 hPa - Absolute accuracy ± 1 hPa
Sensor de temperatura	- Operating range -40°C - 85°C - Full accuracy 0°C - 65°C
Sensor de calidad del aire	- Response time 1s
Sensor de luz (UV)	- Response time 1s
Sensor de luz (Visible)	- Response time 1s
Sensor de partículas	PM1.0, PM2.5 and PM10.0 concentration in both standard & environmental units
Sensor de Gas VOC	- Carbon monoxide CO 1 – 1000ppm - Ethanol C2H6OH 10 – 500ppm - Hydrogen H2 1 – 1000ppm - Ammonia NH3 1 – 500ppm - Methane CH4 > 1000ppm
Wifi Transmit	802.11g: +14 +- 2 dBm (at 54Mbps)
Wifi Receive	802.11g -85 dBm (at 54 Mbps, OFDM)

Dado que la idea de estos dispositivos es tener una mezcla de sensores ha sido necesario la creación de una carcasa 3D utilizando una impresora de forma que le proporcionen un mejor look&feel[28] y luego añadirle las placas.

[27] https://www.arduino.cc/en/main/docs 09/06/2020, 20:39

[28] **look&feel:** metáfora utilizada dentro del entorno de marketing para poder dar una imagen única a los productos.

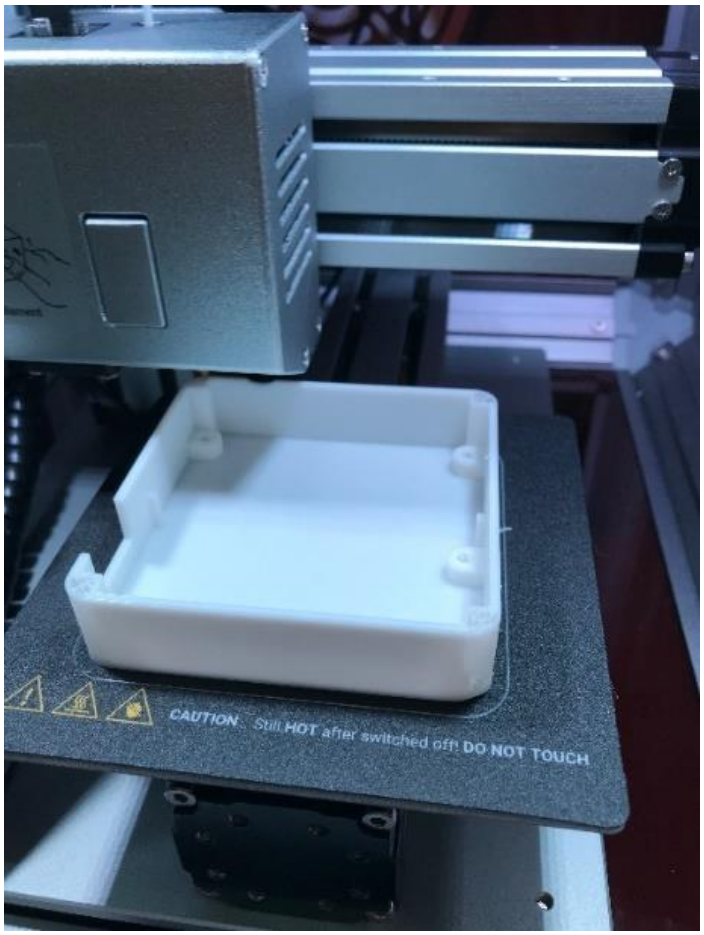

Ilustración 7 – AmpVision IOT impresión 3D

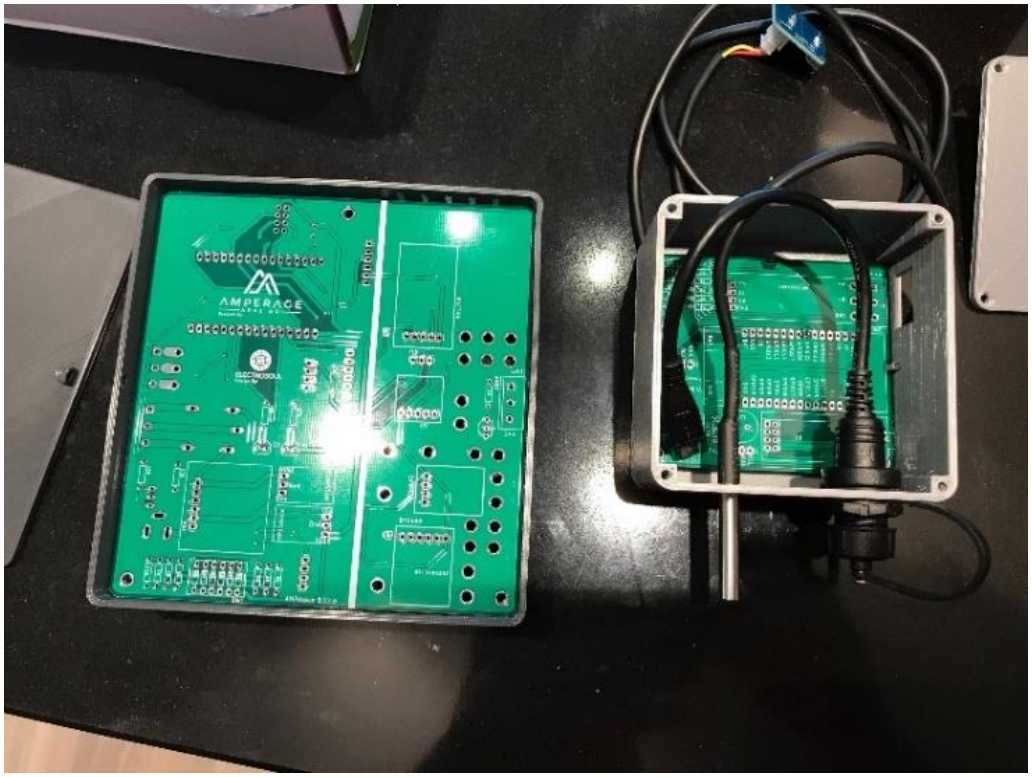

Ilustración 8 – Placas de los dispositivos AmpVision IOT

2.5 AmpVision

Repositorio Git (Privado)	https://github.com/mpache/AmpVision
Entornos de producción	
Traefik	https://traefik-ui.amperageiot.com/
RabbitMQ	https://rabbitmq.amperageiot.com/
Grafana	https://grafana.amperageiot.om/
Entorno de pruebas	
Grafana	http://84.234.166.132:3000/

Plataforma que recoge los datos de los dispositivos IOT y los guarda en una base de datos influxDB para ser utilizados y visualizados en un servicio de monitorización basado en Grafana.

Se ha decidido utilizar una base de datos InfluxDB puesto que, es una base de datos TSDB (Time series database) esta base de datos está optimizada para el almacenamiento rápido y de alta disponibilidad, explicado a fondo en el punto 2.5.3.

Por otro lado, se ha decidido utilizar una solución basada en Grafana para monitorizar los datos y explotarlos puesto que, esta plataforma a partir de la versión 7, ofrece la posibilidad de escribir “backend plugins” los cuales abren la puerta a expandir la solución añadiendo soluciones custom “Grafana Apps” que extiendan la funcionalidad. Puesto que, uno de los principios con los que se ha desarrollado el proyecto es el de, no reinventar la rueda, se ha considerado que, la utilización de esta plataforma open-source era la mejor opción.

Toda la arquitectura está desarrollada como microservicios utilizando Docker de manera que, todo el stack pueda ser trasladado de entorno o tener el stack en diferentes entornos compartiendo las mismas tecnologías de una manera sencilla. Además, desarrollar una plataforma de esta manera facilita el escalado tanto horizontal como vertical, así como su modularidad.

2.5.1 Arquitectura

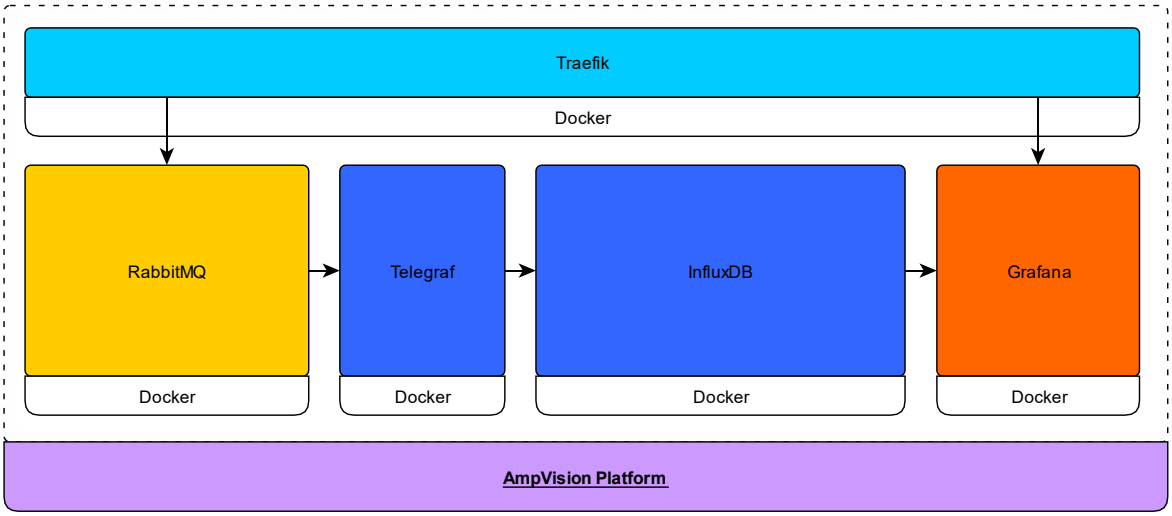

Ilustración 9 – Arquitectura AmpVision

Como podemos observar en el diagrama la plataforma está diseñada como microservicios de forma que cada pieza es independiente.

Además, una de las piezas más importantes es traefik, el cual es el enrutador que conecta una solicitud HTTP/TCP/UDP con un servicio x. Con esta pieza es posible añadir nuevos componentes de forma sencilla y que se puedan comunicar entre ellos o con el exterior.

Por otro lado, la pieza que se encarga de juntar los dispositivos IOT u obtener los mensajes de los dispositivos IOT es rabbitMQ el cual es un message-broker que utilizando el protocolo MQTT permite a cualquier dispositivo utilizar el "publish/subscribe messaging transport" y acto seguido es enviado al pool de agentes telegraf que lo publicará en la base de datos:

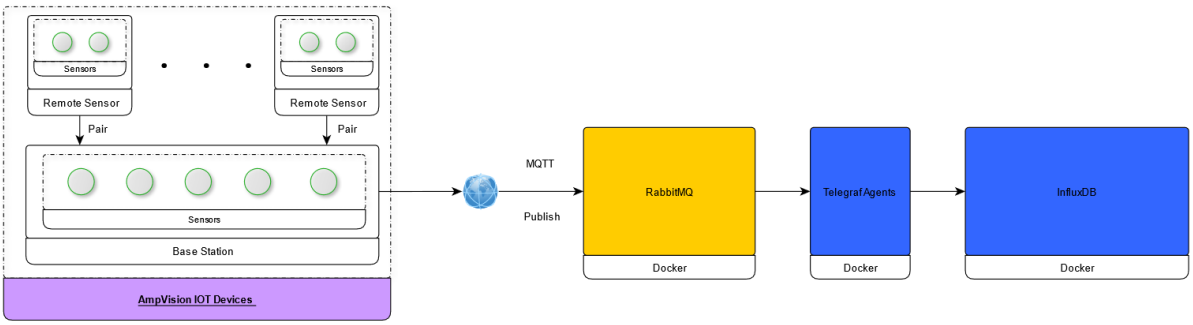

Ilustración 10 – Arquitectura AmpVision message broker

Una vez uno de los agentes telegraf recibe una métrica procede a enviarla a la base de datos correspondiente:

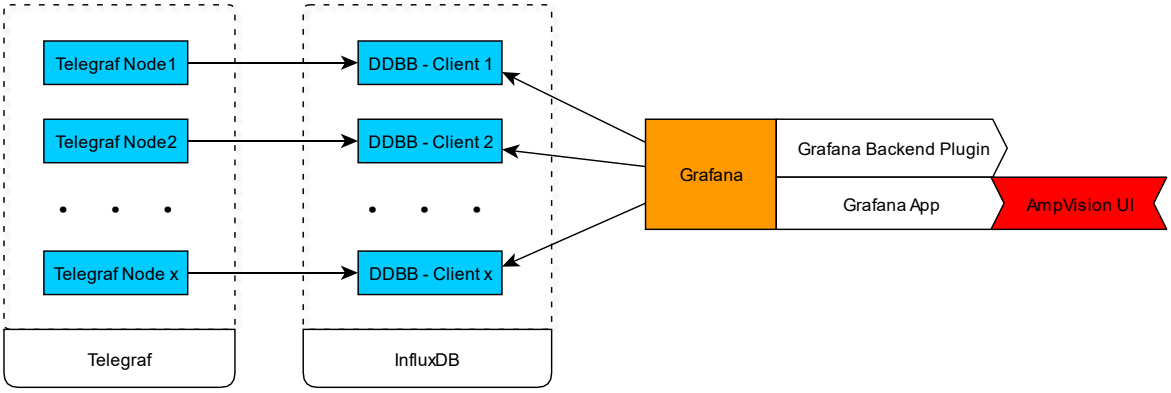

Ilustración 11 – Arquitectura AmpVision telegraf + influxDB connection

Como podemos observar en el diagrama un agente telegraf envía las lecturas a la base de datos InfluxDB correspondiente que luego pueden ser consultadas por Grafana, nuestro servicio de monitorización de los datos obtenidos por los dispositivos IOT. Grafana está conectada con x nodos de base de datos de tipo influxDB
Además, dentro de Grafana existe la posibilidad de desarrollar plugins tanto backend como Apps y donde estas Apps pueden utilizar la librería AmpVision UI, la cual ofrece funcionalidades y componentes de alto rendimiento y de alto renderizado de manera que es posible extender de una manera muy sencilla la funcionalidad ofrecida por grafana.

Por otro lado, esta arquitectura permite conectar dispositivos externos y que nada tenga que ver con AmpVision (simplemente quieren hacer uso de la plataforma SaaS) donde el único requisito es utilizar el protocolo MQTT, aunque, la plataforma es lo suficientemente abierta para aceptar nuevas piesas y/o protocolos.

Por ejemplo, en un futuro se podría aceptar también conexiones entrantes con el protocolo AMQP (Advanced message Queuing Protocol). Tal y como está creada la plataforma solo sería necesario configurar un nuevo agente telegraf el servicio que administre este protocolo y configurar una nueva entrada en nuestro enrutador, Traefik.

En la siguiente imagen podemos ver dicha arquitectura más en detalle y donde se muestra, las diferentes posibilidades para con software/hardware externo:

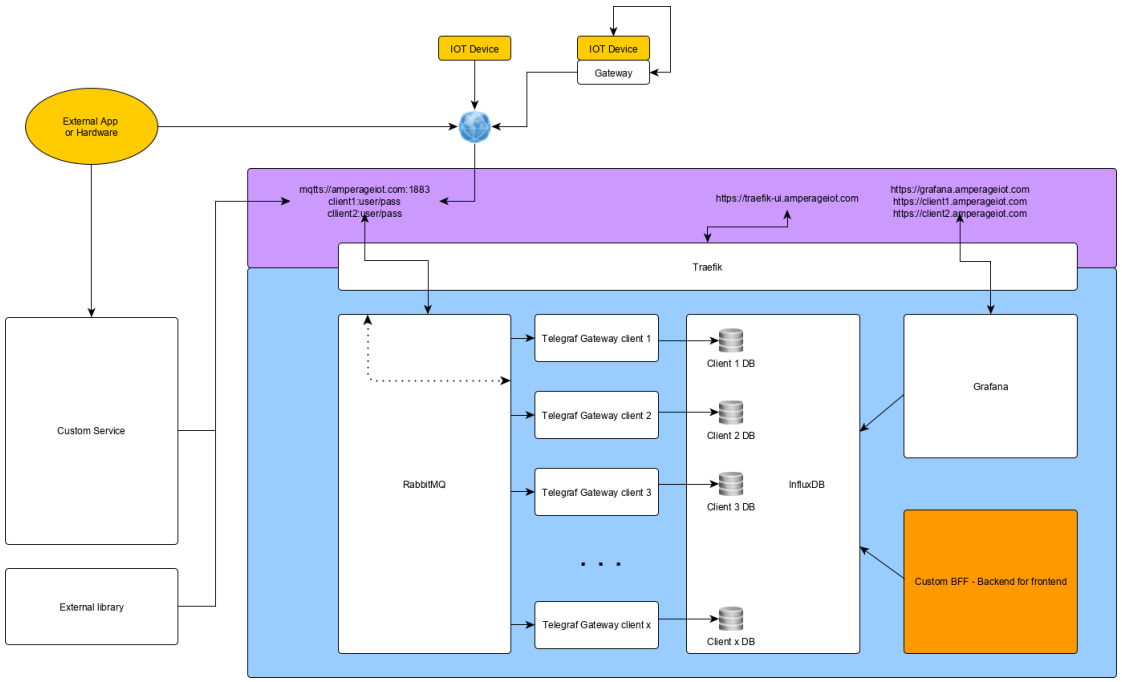

Ilustración 12 Arquitecuta detallada AmpVision

2.5.2 API

De manera que se pueda trabajar con los dispositivos y el servidor MQTT (RabbitMQ) sin necesidad de modificar los tópicos y por consecuencia el código interno de estos dispositivos se ha definido una API para trabajar tanto de cara al hardware como de cara al software.

Item Name	MQTT Topic	Device	Description
Altitude	mqtt_remote_altitude_topic	Remote	Approx. Altitude (m)
Bus Voltage	mqtt_remote_busVoltage_topic	Base	Bus Voltage (v)
Current	mqtt_current_topic	Base	Current (mA)
Description	mqtt_description_topic	Base	Device Description
Description	mqtt_remote_description_topic	Remote	Device Description
Device ID	mqtt_id_topic	Base	ID of the device
Firmware Version	mqtt_sketchversion	Base	Running Firmware
Firmware Version	mqtt_remote_sketchversion	Remote	Running Firmware
Gas Reference	mqtt_remote_gas_resistance_topic	Remote	VOC Gas Reference (KOhms)
Humidity	mqtt_remote_humidity_topic	Remote	Humidity (%)
IAQ	mqtt_remote_airquality_topic	Remote	IAQ
IAQ Score	mqtt_remote_IAQ_topic	Remote	IAQ Score
In	mqtt_subscribe_topic	Base	Incoming MQTT Sub
Light	mqtt_remote_light_topic	Remote	Light Level
Load Voltage	mqtt_loadVoltage_topic	Remote	Load Voltage (v)
Location	mqtt_location_topic	Base	Location of device
Location	mqtt_remote_location_topic	Remote	Location of device
NTP Day	mqtt_ntp_day_topic	Base	Weekday
NTP Time	mqtt_ntp_time_topic	Base	Time of Day
PM1.0	mqtt_data_pm1_standard_topi c	Base	Particulate PM 1.0
PM10	mqtt_data_pm10_standard_to pic	Base	Particulate PM 10
PM2.5	mqtt_data_pm25_standard_to pic	Base	Particulate PM 2.5
Power	mqtt_power_topic	Base	Power (mW)
Pressure	mqtt_remote_pressure_topic	Remote	Pressure (hPa)
Shunt Voltage	mqtt_ShuntVoltage_topic	Base	Shunt Voltage (mV)
Temperature Cº	mqtt_remote_c_temperature_topic	Remote	Temperature Cº
Temperature F	mqtt_remote_tempf_topic	Remote	Temperature F
Uptime	mqtt_uptime_topic	Base	Running Uptime
Uptime	mqtt_remote_uptime_topic	Remote	Running Uptime
UV Light	mqtt_remote_uvlight_topic	Remote	UV Light Level
VOC Gas	mqtt_MiCS5524_topic	Base	VOC Gas (ppm)

Además, dentro de AmpVision y los dispositivos IOT tenemos unos que pueden medir la calidad del aire, pero esta calidad de aire es medida en lo que se conoce como AQI (Air quality index)

AQI	Level	Health implications
0-50	Good	Air quality is satisfactory, and air pollution poses little or no risk.
51-100	Moderate	Air quality is acceptable. However, there may be a risk for some people, particularly those who are unusually sensitive to air pollution.
101-150	Unhealthy for Sensitive Groups	Members of sensitive groups may experience health effects. The general public is less likely to be affected.
151-200	Unhealthy	Some members of the general public may experience health effects; members of sensitive groups may experience more serious health effects.
201-300	Very Unhealthy	Health alert: The risk of health effects is increased for everyone.
301-500	Hazardous	Health warning of emergency conditions: everyone is more likely to be affected.

Por otro lado, las partículas se pueden agrupar en 3 grupos y se pueden mediar de 3 formas:

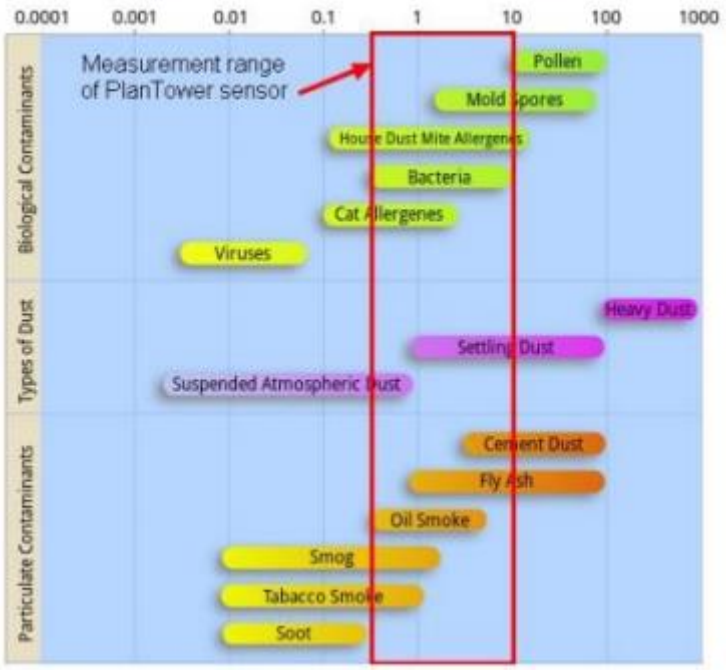

Ilustración 13 – Air quality levels

2.5.3 Database Schema

La base de datos utilizada es InfluxDB la cual es una “Time-Series Databases”. Las bases de datos de series temporales, como su nombre, son sistemas de bases de datos diseñados específicamente para manejar datos relacionados con el tiempo.

Generalmente una base de datos se basa en que estas tienen columnas y filas, cada una de ellas define una entrada en su tabla. A menudo, esas tablas están diseñadas específicamente para un propósito: una puede estar diseñada para almacenar usuarios, otra para fotos y finalmente para videos.

Las bases de datos de series temporales funcionan de manera diferente. Los datos todavía se almacenan en "colecciones", pero esas colecciones comparten un denominador común: se agregan con el tiempo, por lo tanto, no existe un esquema predeterminado.

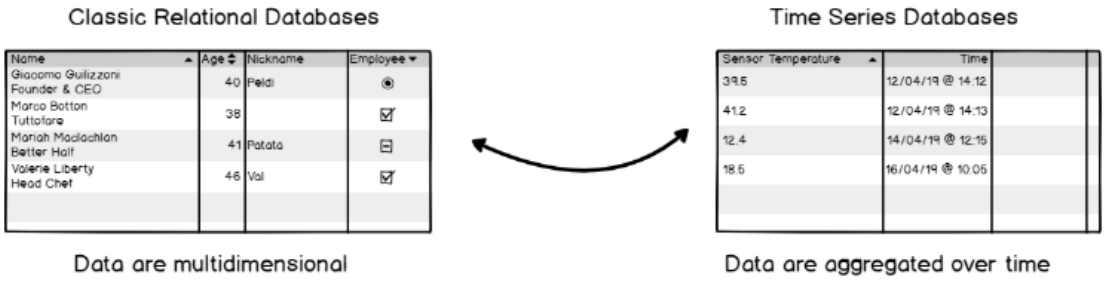

Ilustración 14 – Classic relational Databases vs Time series Databases

Los sistemas de bases de datos de series temporales se basan en el predicado de que necesitan ingerir datos de manera rápida y eficiente.

De hecho, las bases de datos relacionales tienen una tasa de ingestión rápida para la mayoría de ellas, de 20k a 100k filas por segundo. Sin embargo, la ingestión no es constante en el tiempo. Las bases de datos relacionales tienen un aspecto clave que las hace lentas cuando los datos tienden a crecer: los índices.

La base de datos de series temporales está optimizada para una tasa de ingestión rápida. Significa que tales sistemas de índice están optimizados para indexar datos que se agregan con el tiempo: como consecuencia, la tasa de ingestión no disminuye con el tiempo y se mantiene bastante estable, alrededor de 50k a 100k líneas por segundo en un solo nodo.

2.5.4 Plataforma

Para enseñar la plataforma, enseñaremos las diferentes piezas de las que se compone AmpVision.

Traefik

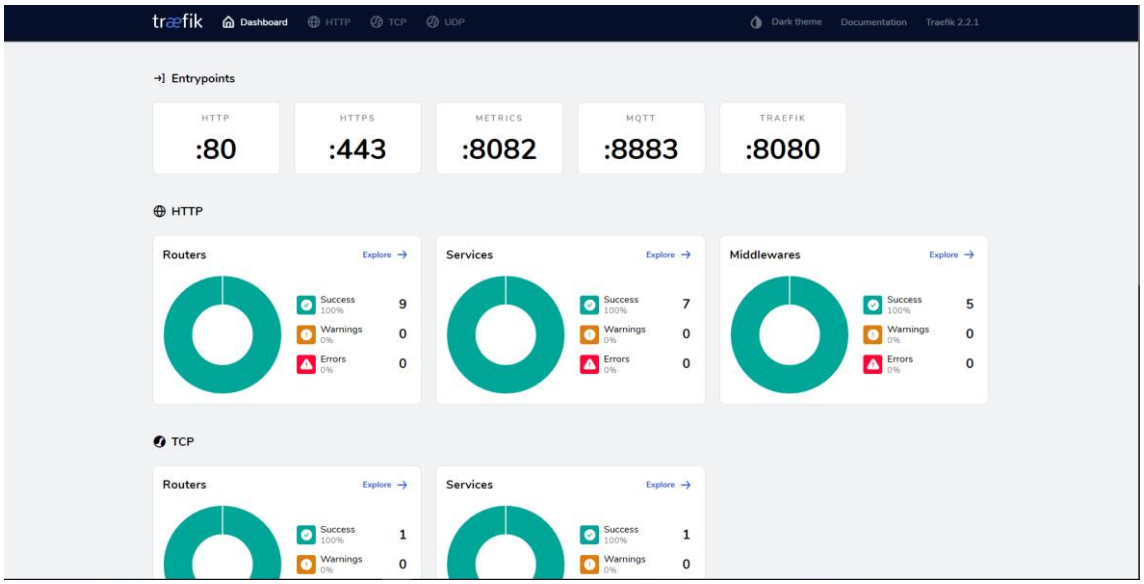

Ilustración 15 – Traefik global

Ahora bien, como se ha explicado, traefik sirve como Edge router por lo tanto tenemos que conectar las diferentes piezas a traefik para que este sea capaz de redirigir el tráfico.

Para ello es necesario crear los diferentes “routers” a los que queremos acceder junto con sus servicios asociados.

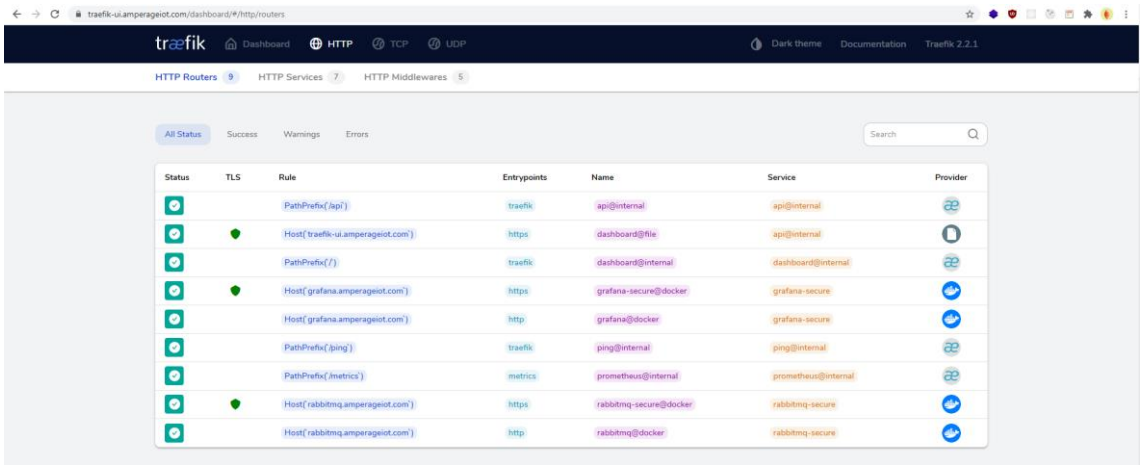

Ilustración 16 – Traefik HTTP Routers

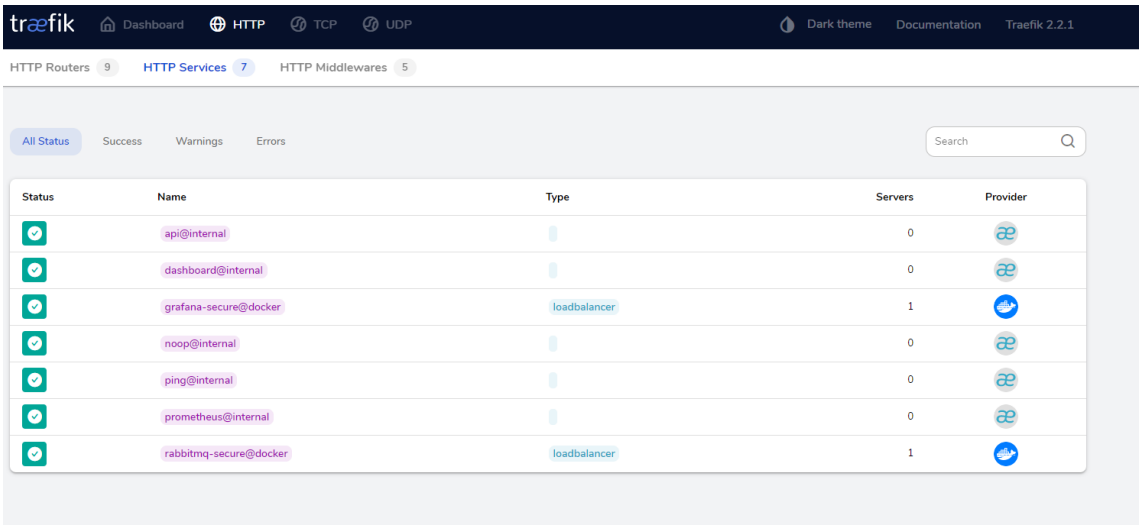

Ilustración 17 – Traefik HTTP Services

Algo interesante, es que también utilizamos traefik como balanceador de carga (como podemos ver en la imagen) aunque en este caso solo tenemos un servidor, pero si tuviéramos más las peticiones se balancearían automáticamente.

A continuación, se muestra la configuración del router y los servicios tanto para rabbitMQ como para Grafana:

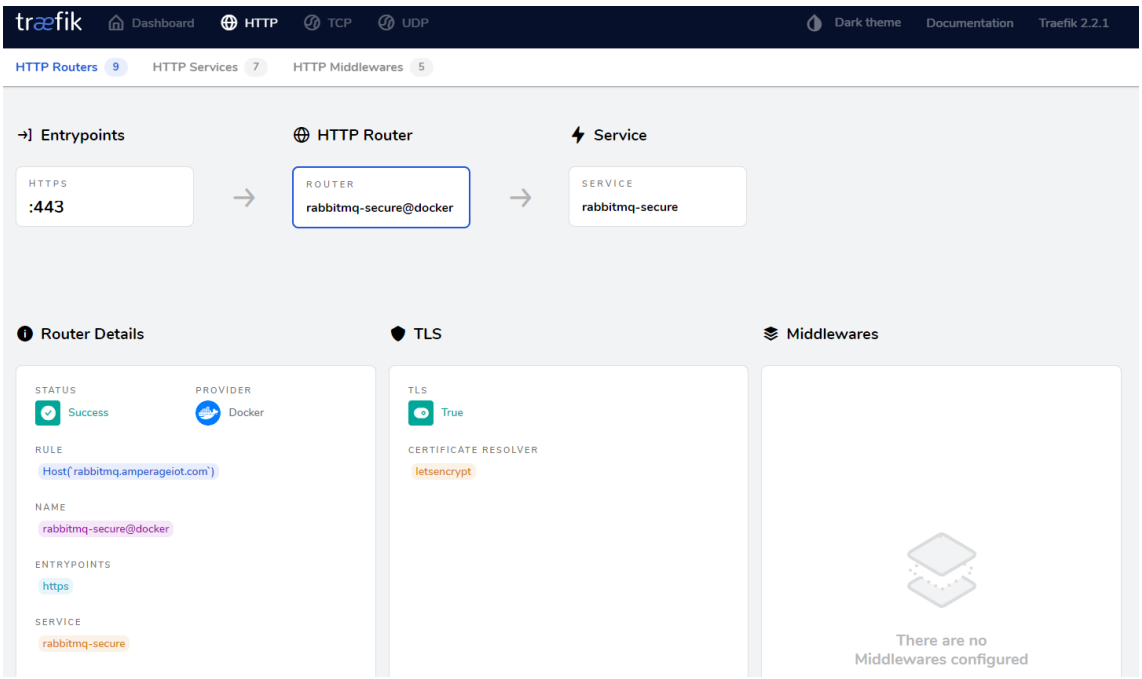

Ilustración 18 – Traefik HTTP Router rabbitMQ

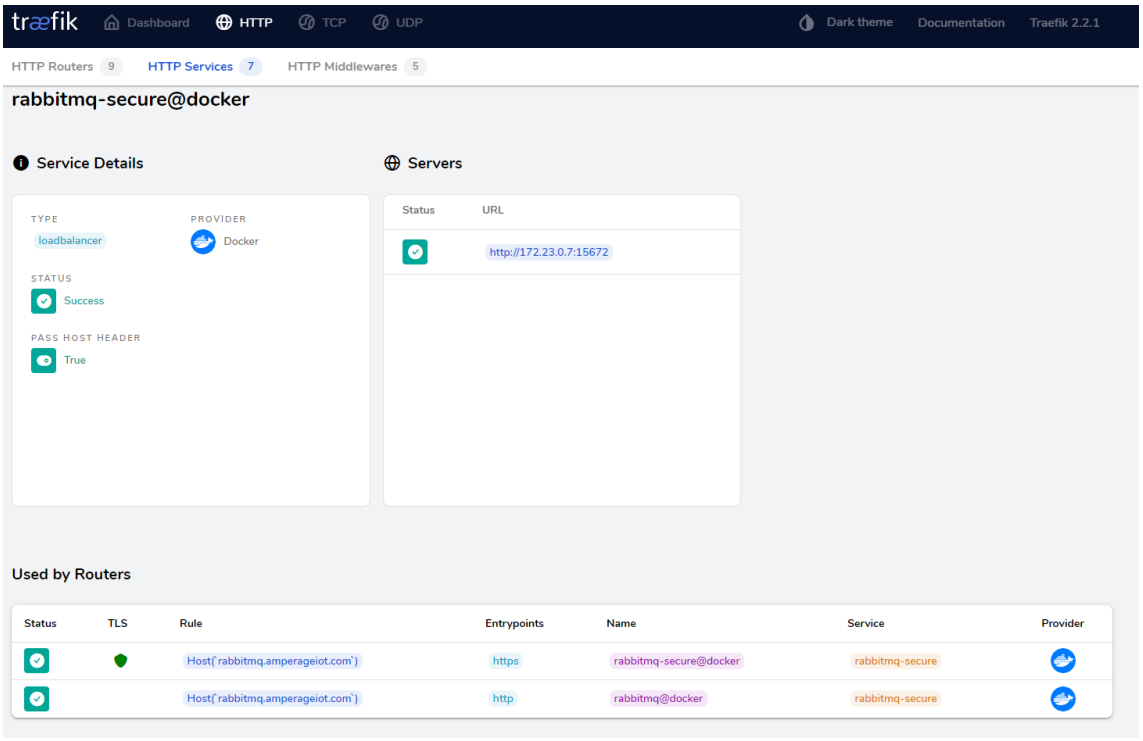

Ilustración 19 – Traefik HTTP service rabbitMQ

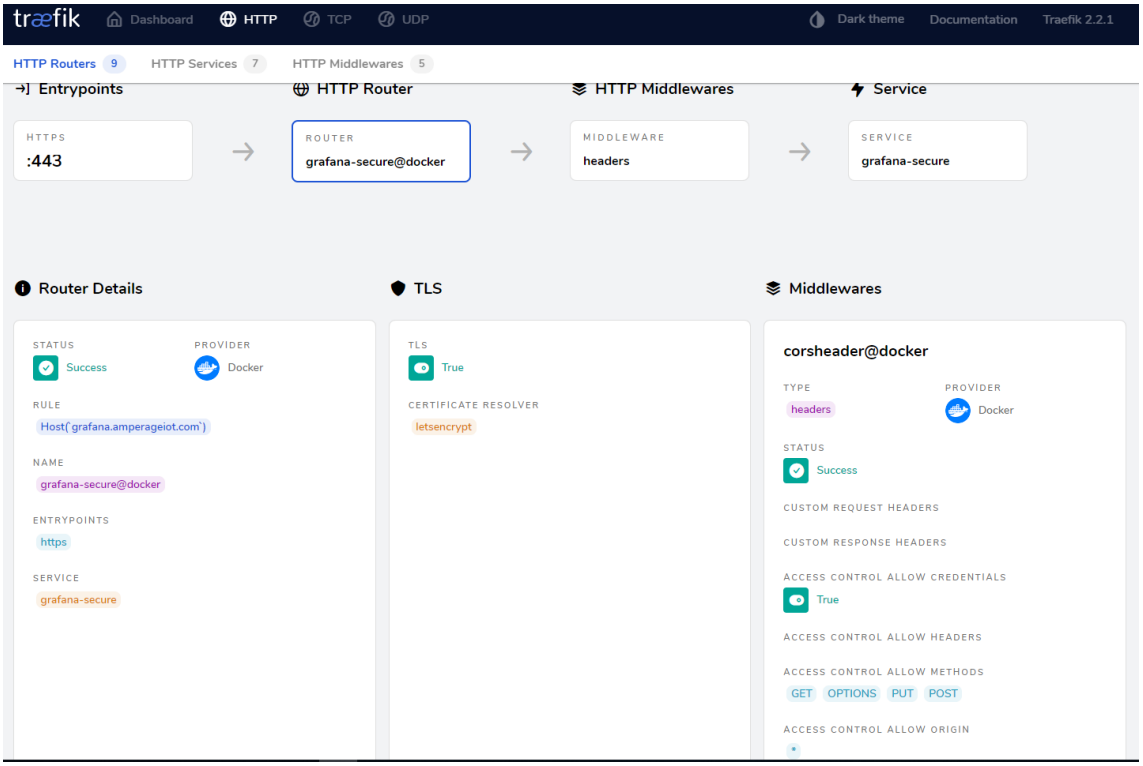

Ilustración 20 – Traefik HTTP Router Grafana

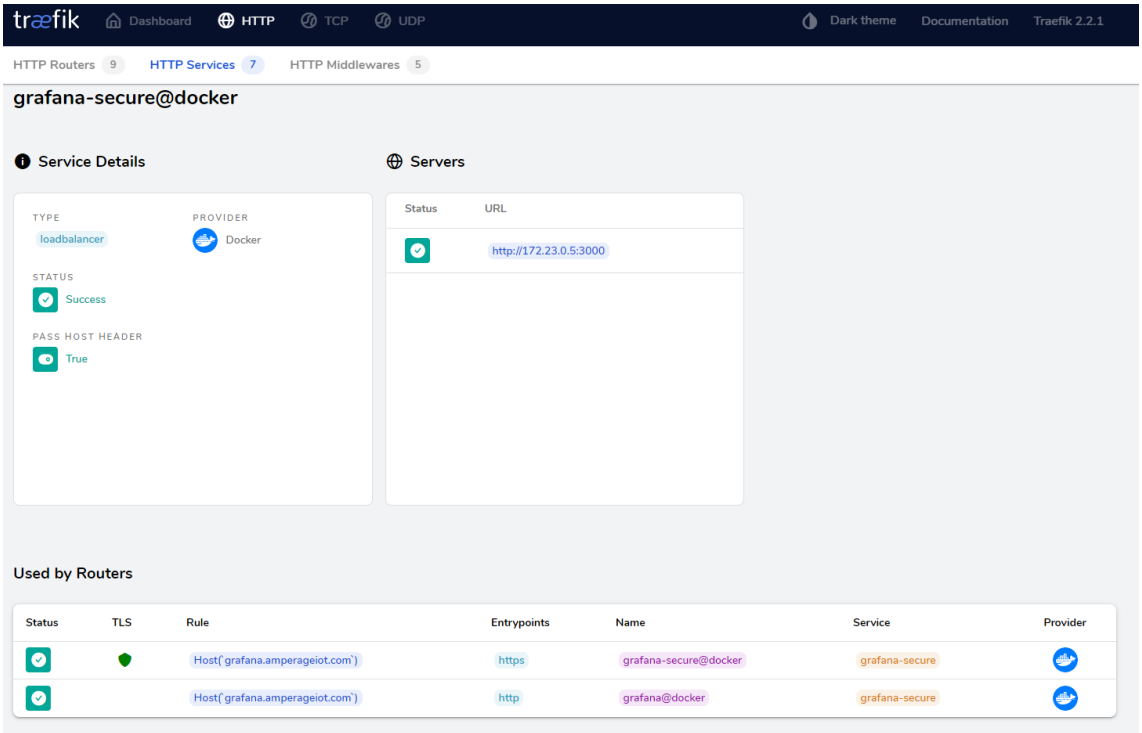

Ilustración 21 – Traefik HTTP Service Grafana

Ahora bien, la configuración anterior es para peticiones http, como se ha explicado anteriormente en este documento los dispositivos IOT envían los datos utilizando el protocolo MQTT el cual viaja a través de una conexión TCP

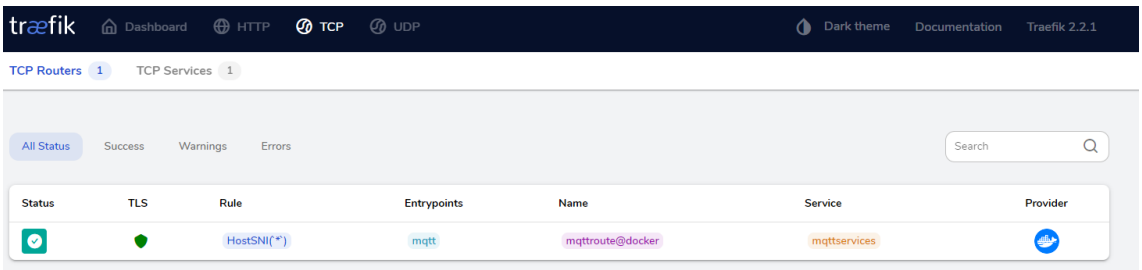

Ilustración 22 – Traefik TCP routers

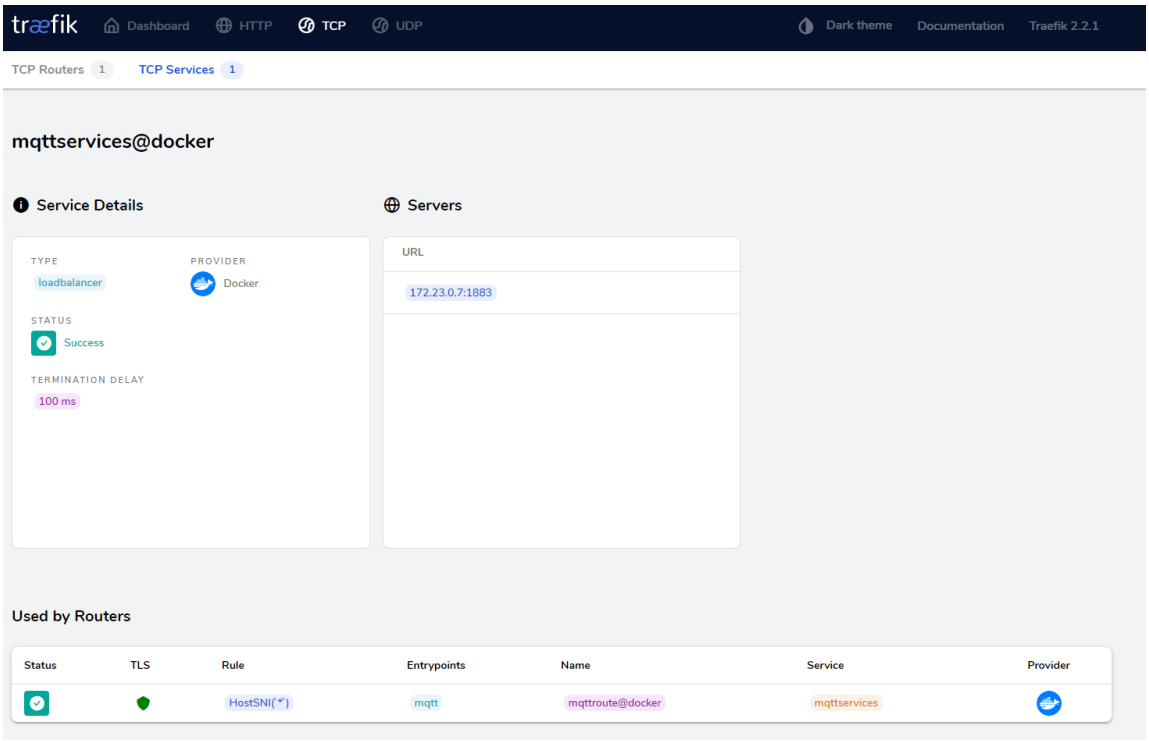

Ilustración 23 – Traefik MQTT Service

Podemos observar que todos los servicios están utilizando contenedores dockers para modular nuestra plataforma de forma que podamos mover estos servicios, es decir toda la plataforma está pensada para que use microservicios[29].

[29] **Microservices:** consiste en construir una aplicación como un conjunto de pequeños servicios.

RabbitMQ

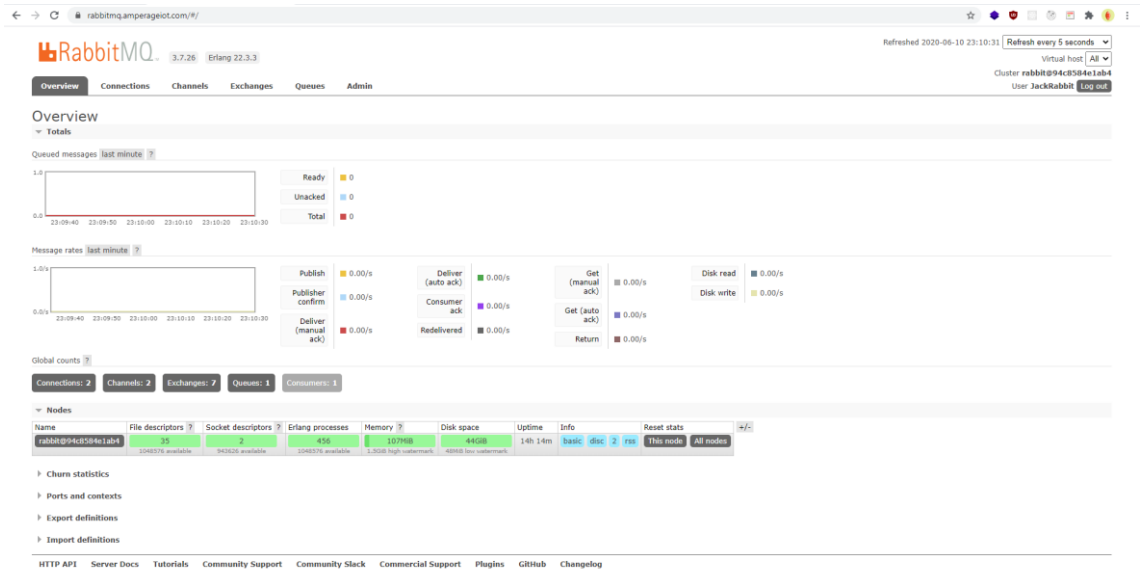

Ilustración 24 – Global RabbitMQ

Como podemos observar en el diagrama de la arquitectura, las colas de rabbitMQ están conectadas a los agentes telegraf los cuales son los encargados de enviar las métricas que reciben a la base de datos:

Ilustración 25 – RabbitMQ queues

MQTT

En este punto es donde unimos los 2 productos, AmpVision IOT con AmpVision.

Para enseñar esto podemos enseñar los mensajes que se están enviado por el protocolo MQTT.

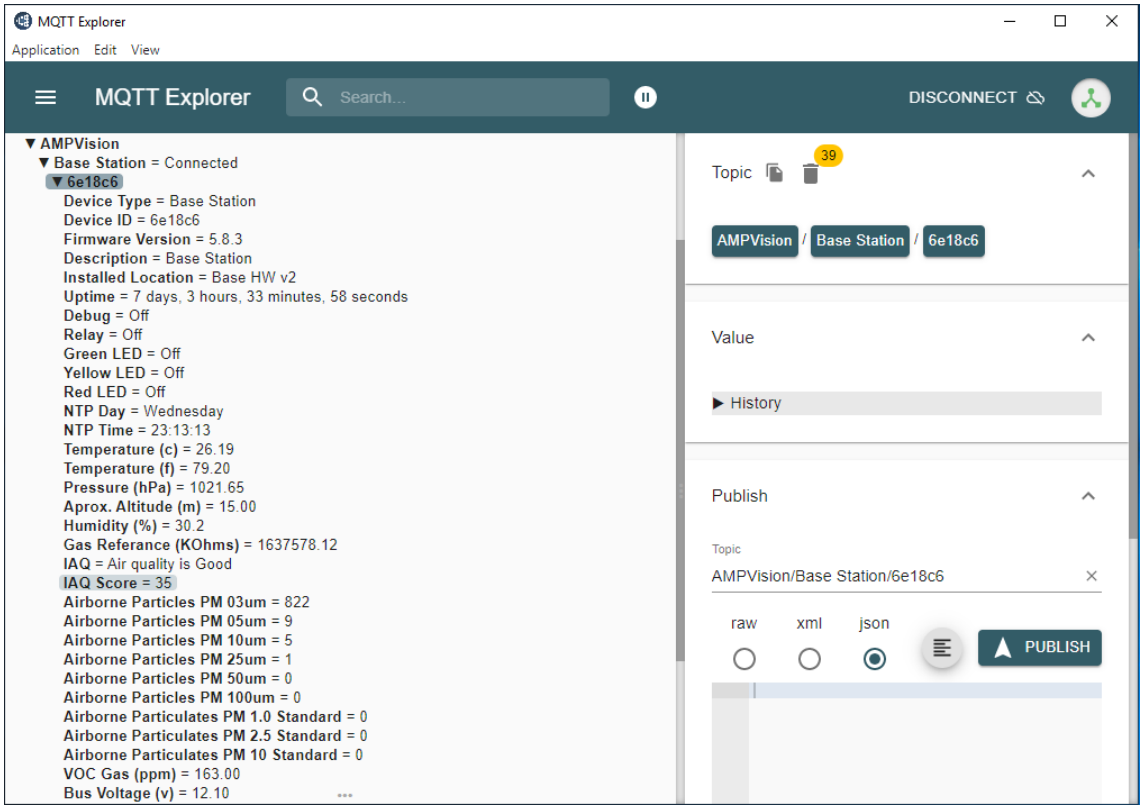

Ilustración 26 – AmpVision IOT + AmpVision

Podemos ver los mensajes que esta enviada la “Base Station” la cual tiene un “Remote” conectado el cual tiene por id ***6e18c6***.

Grafana

Para visualizar y analizar las métricas que viajan utilizando la plataforma utilizamos Grafana, la cual es capaz de visualizar cualquier tipo de dato.

Ilustración 27 - Grafana Base Station

En la imagen anterior, podemos ver todas las métricas enviadas por el dispositivo ***6e18c6.***

Además, como hemos explicado anteriormente en este documento, queremos una plataforma SaaS que sea capaz de visualizar datos independientemente de sí utilizan nuestros dispositivos IOT, de hecho, también utilizamos Grafana para visualizar el estado de nuestra propia plataforma:

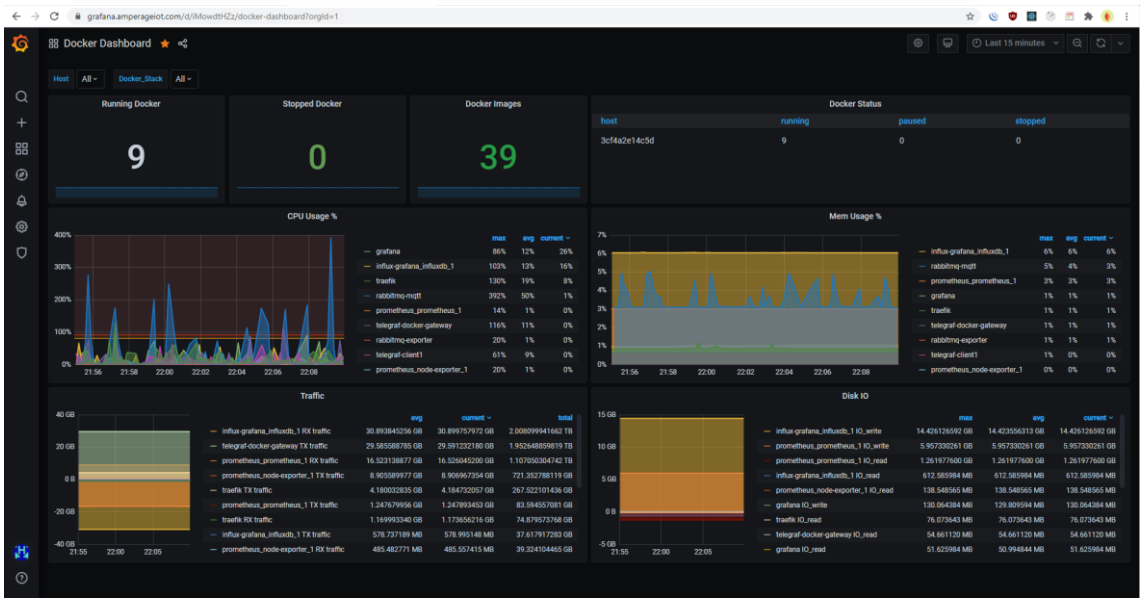

Ilustración 28 – Grafana production environment

2.6 AmpVision UI

Repositorio Git	https://github.com/mpache/AmpVision-UI
Entorno de documentación	https://mpache.github.io/AmpVision-UI/

Entorno y librería de componentes escritos en React que ofrece componentes orientados a la generación de dashboards o simplemente para facilitar el desarrollo de cualquier web de una forma más fácil y eficiente puesto que todos los componentes están testeados en el propio entorno que se ofrece.

La librería está diseñada y mantenida utilizando la plataforma Figma (https://www.figma.com/) una herramienta de diseño de interfaz colaborativa. Esta plataforma permite crear y compartir diseños o prototipos los cuales pueden ser cualquier cosa (aplicación móvil, un componente, una página web, etc) y, además, es posible obtener el código utilizado mediante los “Figma tokens”, por lo que es posible automatizar la generación de, por ejemplo, iconos.
Para más información ver Anexos.

Puesto que se trata de una Liberia de componentes es necesario tener un entorno donde mostrar la documentación de estos componentes, pero AmpVision UI busca ir un paso más allá ofreciendo un entorno donde se pueda documentar, compartir y testear todos y cada uno de los componentes. Este entorno no solo sirve como un portal para mantener una buena documentación, sino también para tener una manera fácil de probar manualmente cada componente y permitir a los usuarios no técnicos la posibilidad de contribuir. Además, este entorno permite testear la accesibilidad y el performance de cada componente.

Así pues, con el entorno de AmpVision UI es posible:

- Plain documentation.
- Live tests documentation.
- Live tests performance.
- Live tests accessibility.

De esta manera, todo componente que se desarrolle para la plataforma ofrece las garantías suficientes para que este sea accesible y rápido, dos puntos muy importantes de cara a utilizar una librería u otra.

2.6.1 Arquitectura

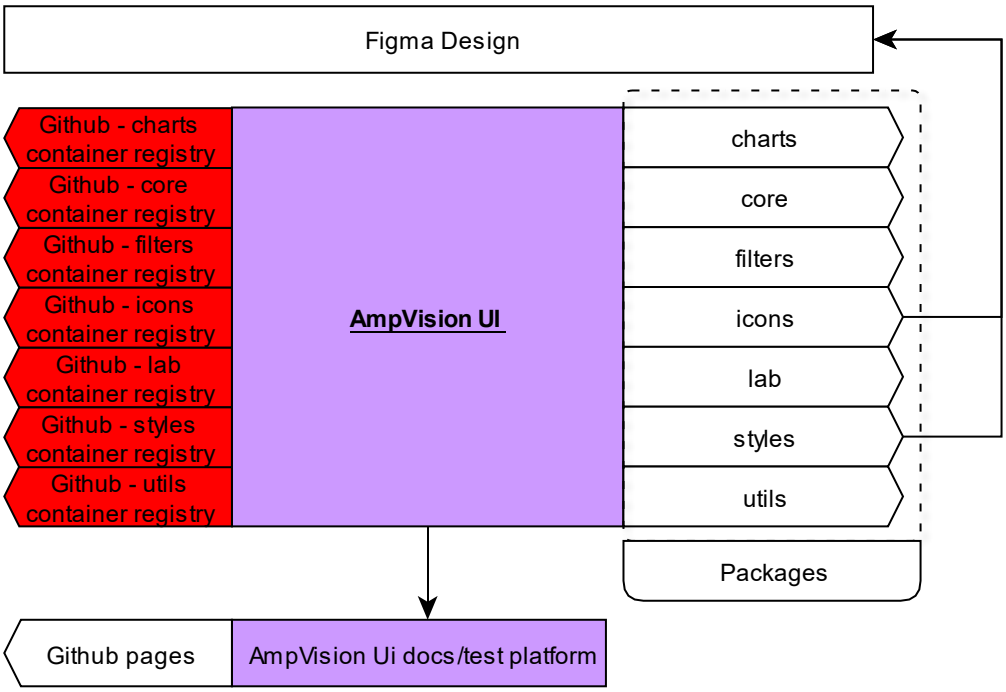

Ilustración 29 – Arquitectura AmpVision UI

Monorepo[30] que ofrece diferentes paquetes con diferentes funcionalidades explicadas a continuación:

- **Charts:** React charts componentes.
- **Core:** componentes principales.
- **Filters:** Filtro optimizado que permite crear filtros AND u OR. Este filtro está desarrollado utilizando un algoritmo en forma de árbol y, optimizado para permitir el máximo rendimiento.
- **Icons:** conjunto de iconos en formato svg optimizados y autogenerados.
- **Lab:** este paquete se utiliza a modo de incubadora para todos aquellos componentes que aún no están listos para pasar al core.
- **Styles:** este paquete aloja el estilo de los componentes y la funcionalidad del sistema de temas para permitir una customización de los componentes de forma externa.
- **Utils:** este paquete aloja utilidades funcionales.

[30] **Monorepo:** estrategia de desarrollo de software en la que el código de muchos proyectos se almacena en el mismo repositorio.

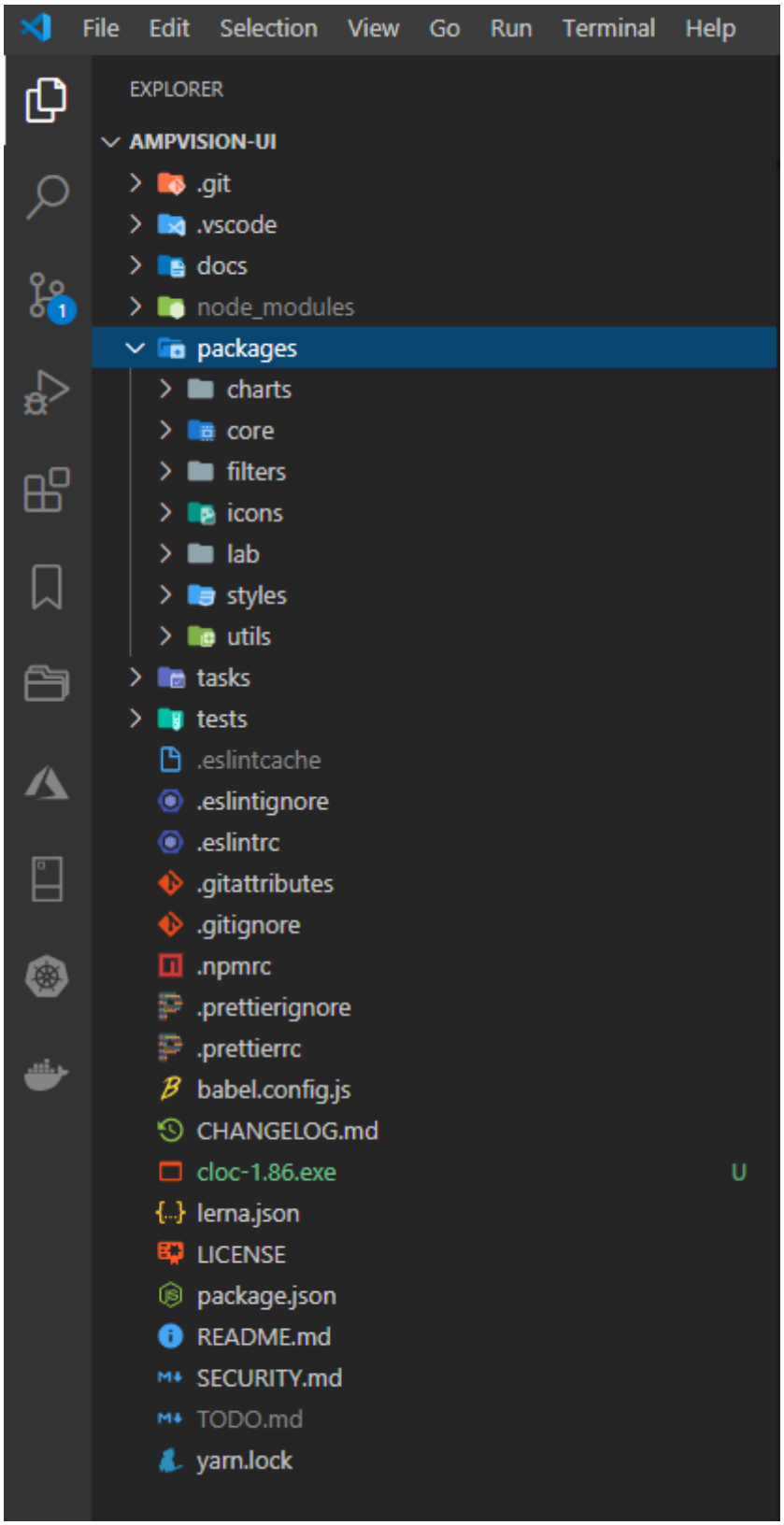

Ilustración 30 AmpVision UI packages

La librería de componentes ha sido desarrollada utilizando el concepto de diseño atómico[31] donde:

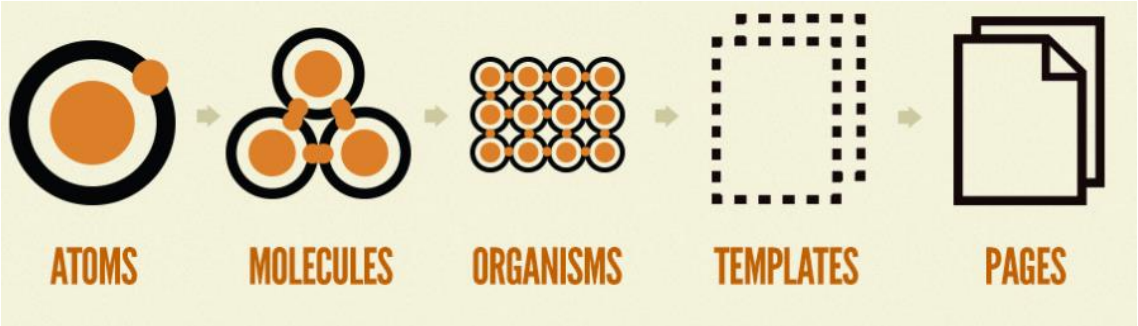

Ilustración 31 – Atomic design

- **Átomos**: los componentes básicos, todos aquellos componentes que sean únicos, por ejemplo, un Button.
- **Molecules**: grupos de átomos unidos entre sí, es decir, las moléculas son componentes resultantes de utilizar 1 o más componentes átomos, por ejemplo, un Toggle Button.
- **Organisms**: Los organismos son componentes que utilizan moléculas y/o átomos. Por ejemplo, un datepicker. Estos componentes son los de más alto nivel y los que ofrecen una menor customización.
- **Templates**: las templates constan de grupos de organismos que forman una página.
- **Pages**: son casos específicos de plantillas.

Por otro lado, los componentes icons y styles se comunican con la plataforma Figma para generar sus componentes de forma automática, esto es posible utilizando su API, https://www.figma.com/developers/api

Además, el componente “filters” proporciona un algoritmo en árbol para poder filtrar resultados en ms.

Suponiendo que tenemos un filtro con la siguiente estructura:

- F1
  - F1-child1
  - F1-child2
- F2
  - F2-child1
- F3
- F4
- F5

[31] https://bradfrost.com/blog/post/atomic-web-design/ 11/06/2020, 17:06

Para poder acelerar el filtrado de los datos se genera la siguiente estructura en árbol:

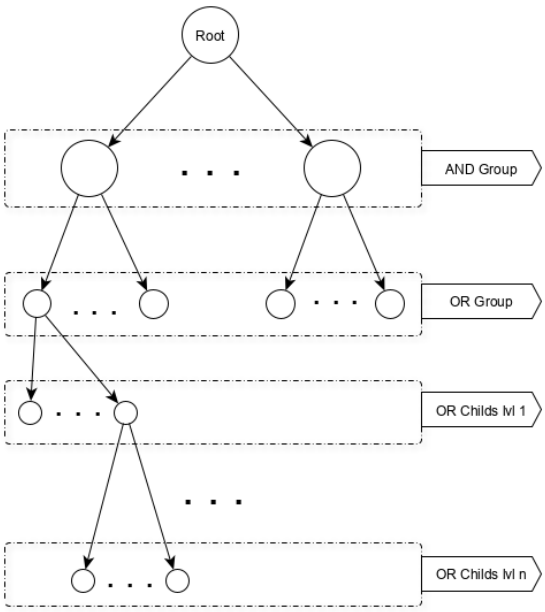

Ilustración 32 AmpVision UI filters tree structure

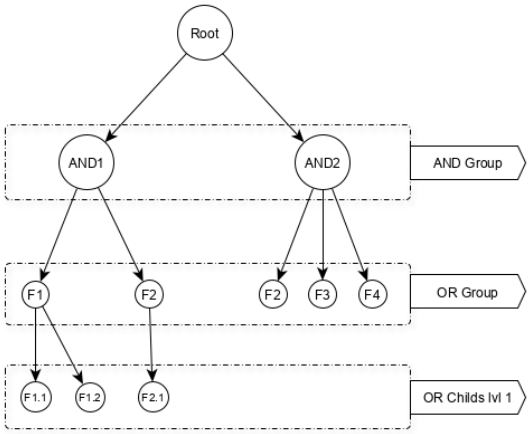

Ilustración 33 – AmpVision UI tree example

La cual indica que los filtros dentro del bloque “and” se filtran entre ellos como operaciones OR, pero entre filtros de diferente bloque “and” se filtrarán utilizando operaciones AND. Es decir, una operación AND significa que para mostrar un dato es necesario que se cumplan todas las condiciones y una OR solo es necesario que se cumpla 1.

El algoritmo esta desarrollado utilizando un árbol donde los filtros seleccionados se sitúan siempre a la izquierda, de esta manera, se consigue evitar iteraciones haciendo que la complejidad algorítmica pase de exponencial a logarítmica pudiendo ofrecer un filtrador optimizado.

La complejidad algorítmica de las operaciones es la siguiente:

Operación	**Notación O**
Búsqueda	O(log n)
Insertar	O(log n)
Filtrar nodos seleccionados	O(log n)
Obtener futuros resultados	O(n)

Complejidad total O(log n + n)

2.6.2 Entorno de desarrollo y documentación

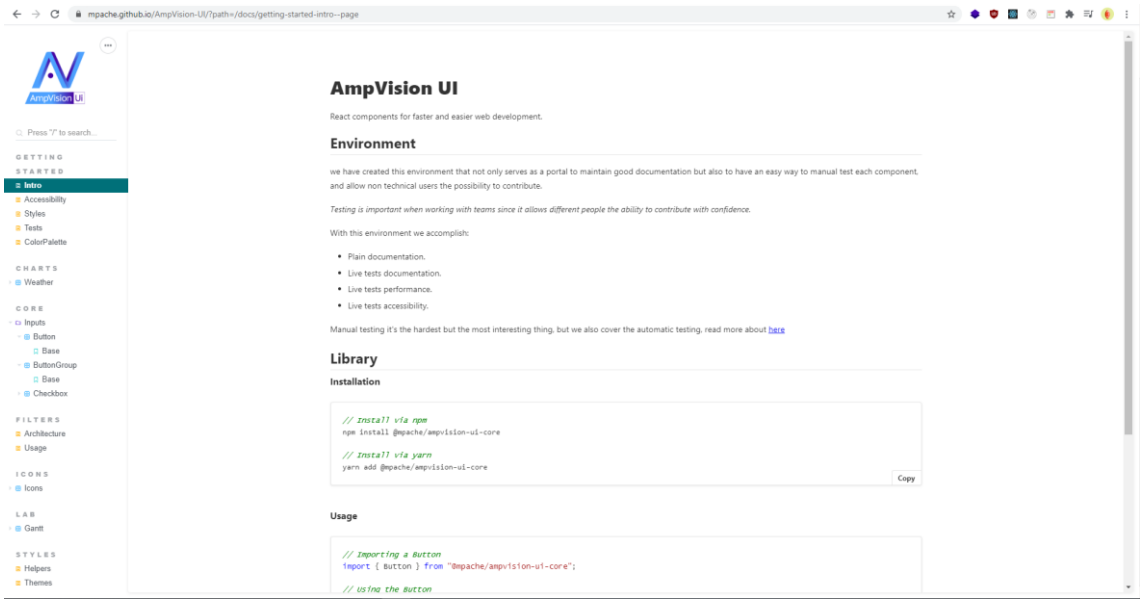

Ilustración 34 AmpVision UI environment1

Colores generados automáticamente obtenidos del diseño realizado en la plataforma Figma:

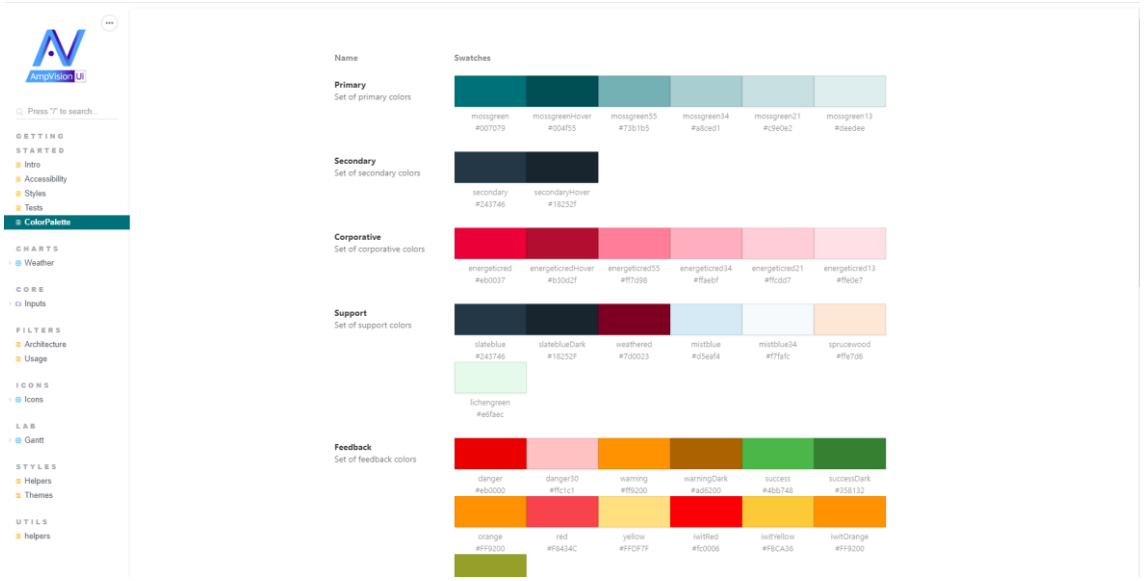

Ilustración 35 AmpVision UI color pallete

Iconos generados automáticamente obtenidos del diseño realizado en la plataforma Figma:

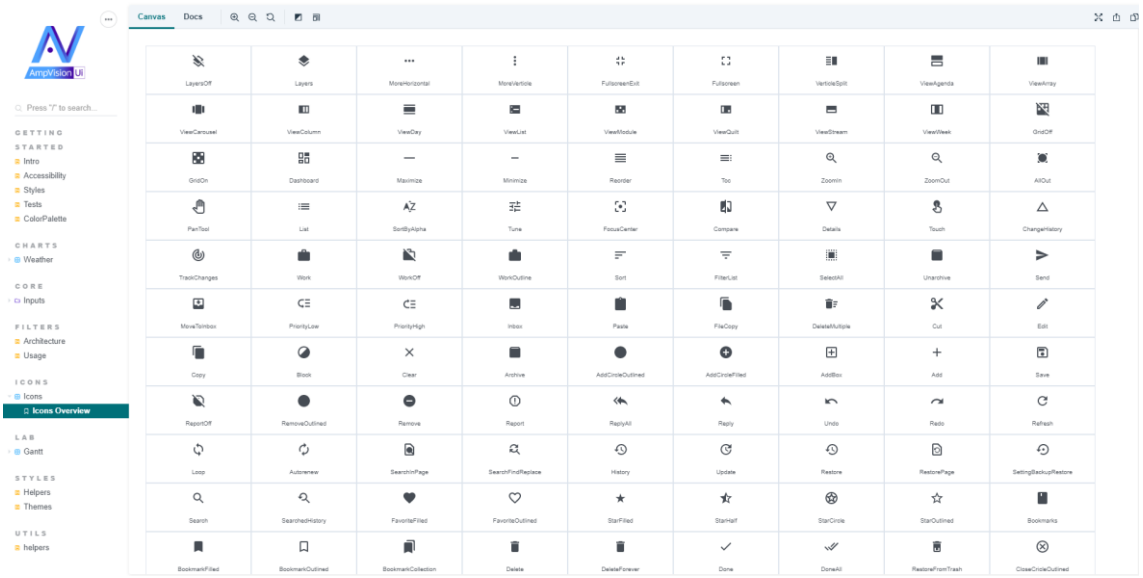

Ilustración 36 AmpVision UI icons

Ejemplo de documentación de componentes:

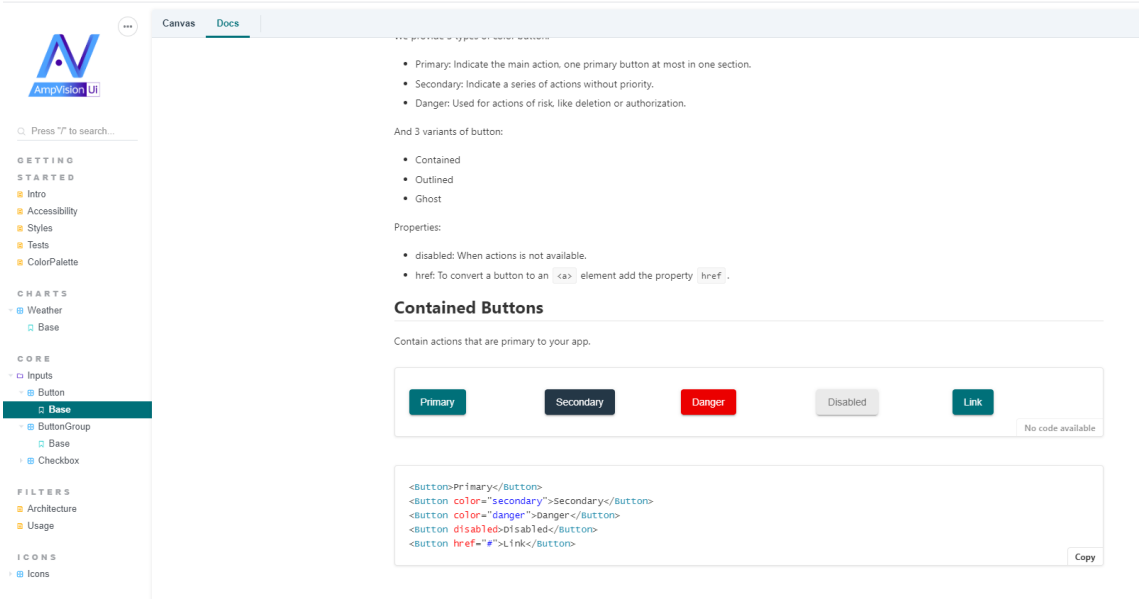

Ilustración 37 – AmpVision UI components documentation

Ejemplos de uso en vivo, donde el usuario puede probar los componentes y sus diferentes opciones

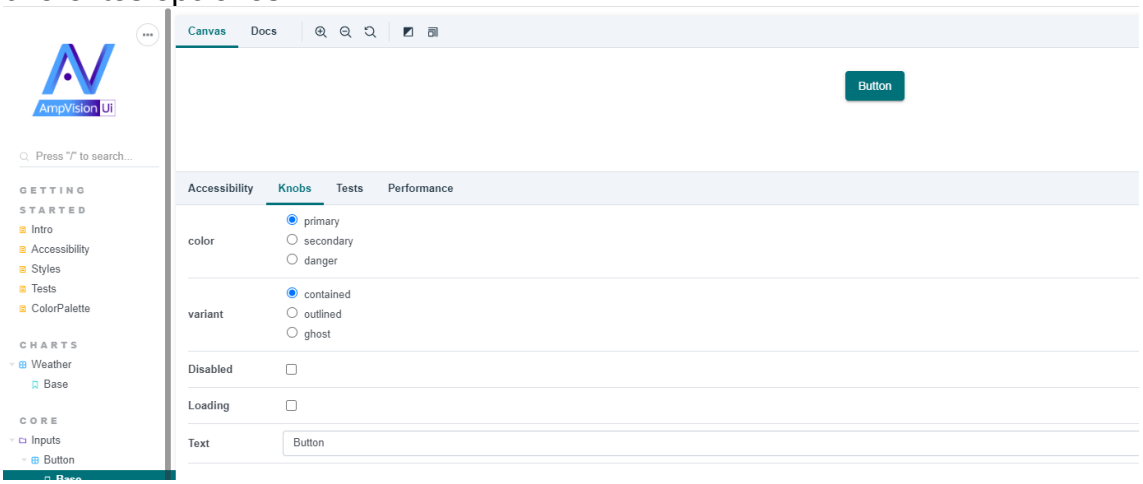

Ilustración 38 – AmpVision UI live tests documentation

Además, el entorno permite tanto analizar la accesibilidad de los componentes en vivo como el performance de estos.

Cabe destacar que, las pruebas de accesibilidad son pruebas automáticas que comprueban que los componentes siguen los estándares impuestos por el W3[32]

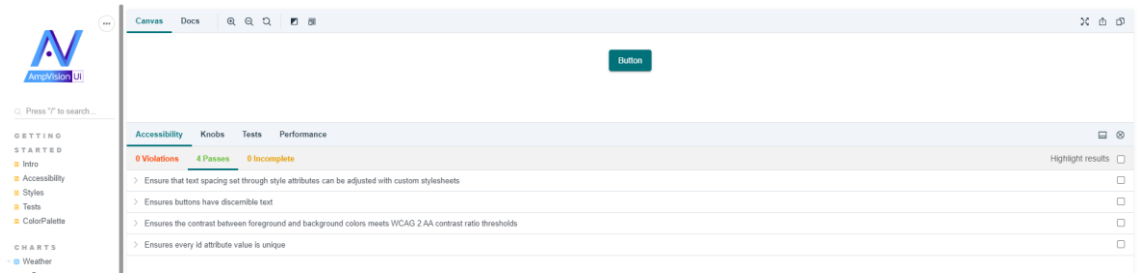

Ilustración 39 – AmpVision UI live accessibility tests

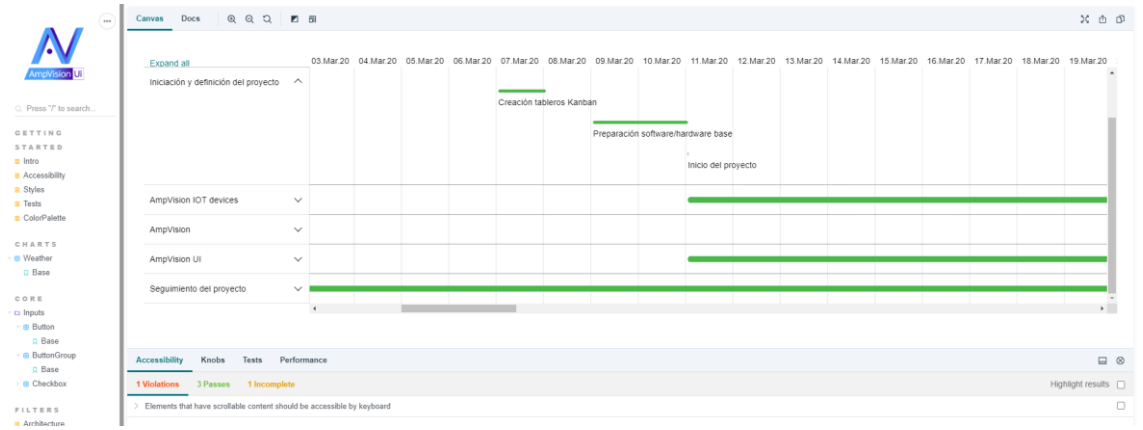

Ilustración 40 AmpVision UI live accessibility tests 2

[32] https://www.w3.org/WAI/standards-guidelines/ 11/06/2020 21:04

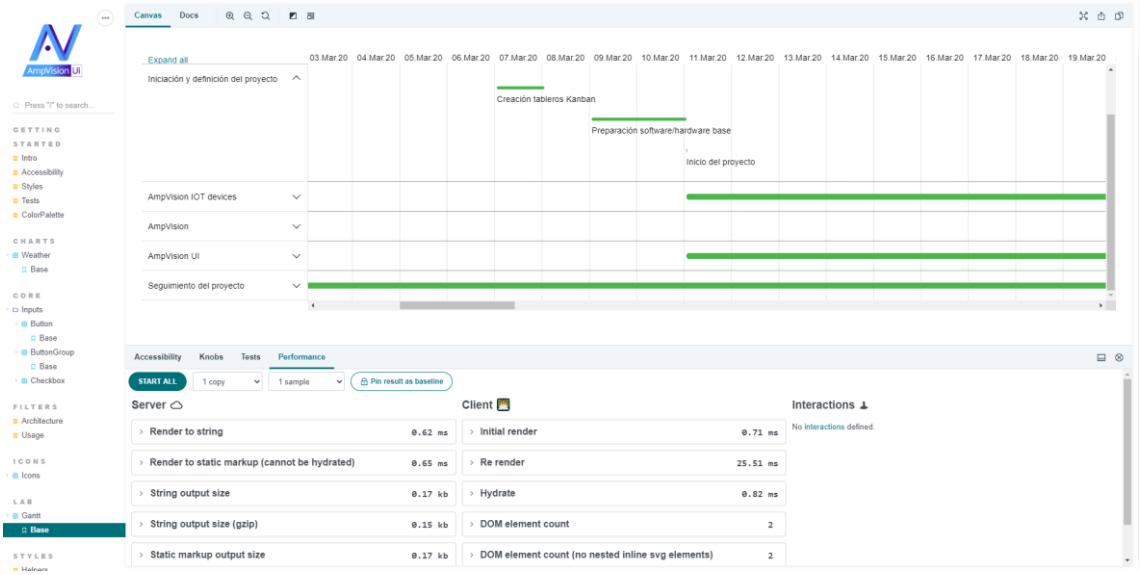

Ilustración 41 AmpVision UI live tests performance

Este tipo de pruebas es muy importante ya que, de una manera visual, demuestra como de costoso es utilizar el componente en cuestión y, además donde está el coste puesto que, un componente puede ser muy rápido de renderizar, pero muy costoso renderizarlo de nuevo o lo contrario.

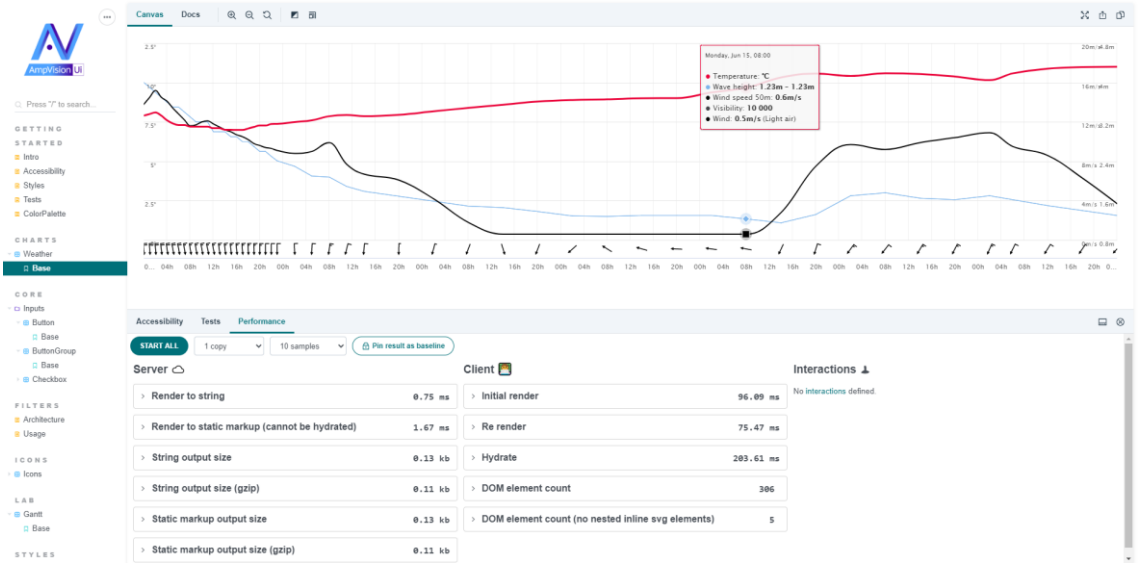

Ilustración 42 – AmpVIsion UI live test performance 2

2.6.3 Testing

Ahora bien, también se han desarrollado diferentes pruebas de forma que todos los componentes queden testeados.

1. Tests estructurales

Aquí nos centraremos en la estructura de la interfaz de usuario y cómo se presenta.

2. Tests de interacción

La interfaz de usuario se trata de la interacción con el usuario. Hacemos esto con un montón de elementos de la interfaz de usuario, como botones, enlaces y elementos de entrada. Con las pruebas de interacción, debemos probar si funcionan correctamente.

Estos tipos de pruebas se han conseguido utilizando la librería Jest y enzyme la cual de una manera sencilla ofrece la posibilidad de realizar estas pruebas.

All files

89.87% Statements 1003/1116 **52.79%** Branches 104/197 **89.65%** Functions 684/763 **89.78%** Lines 993/1106

Press *n* or *j* to go to the next uncovered block, *b*, *p* or *k* for the previous block.

File	Statements		Branches		Functions		Lines	
packages/filters/src	100%	48/48	100%	14/14	100%	18/18	100%	48/48
packages/filters/src/entity	0%	0/0	0%	0/0	0%	0/0	0%	0/0
packages/filters/src/utils	100%	141/141	97.75%	87/89	100%	33/33	100%	133/133
packages/icons/src	100%	615/615	100%	0/0	100%	615/615	100%	615/615
packages/styles/src	0%	0/0	0%	0/0	0%	0/0	0%	0/0
packages/styles/src/theme	0%	0/0	0%	0/0	0%	0/0	0%	0/0
packages/styles/src/utils	100%	45/45	100%	0/0	100%	15/15	100%	45/45
packages/styles/src/variables	100%	65/65	100%	0/0	100%	1/1	100%	65/65
packages/utils/src	0%	0/0	0%	0/0	0%	0/0	0%	0/0
packages/utils/src/helpers	44.67%	88/197	3.19%	3/94	2.53%	2/79	44.1%	86/195
packages/utils/src/hooks	0%	0/4	100%	0/0	0%	0/2	0%	0/4
tests/config	100%	1/1	100%	0/0	100%	0/0	100%	1/1

Ilustración 43 – AmpVision UI tests estructurales y de interacción

3. CSS/Style Testing

La interfaz de usuario tiene que ver con los estilos y, con las pruebas de estilo, estamos evaluando el aspecto de nuestros componentes de la interfaz de usuario entre cambios de código. Este es un tema bastante complejo y lo hacemos comparando imágenes y guardando instantáneas de la documentación en cada versión. Con este tipo de pruebas se intenta evitar al máximo posible las regresiones en el código producidas por cambios en los estilos que al ojo humano pueden pasar desapercibidos, pero una maquina los puede detectar sin problema alguno.

Para demostrar esto, forzaremos un cambio en el componente de tiempo:

Ilustración 44 AmpVision UI automated visual testing

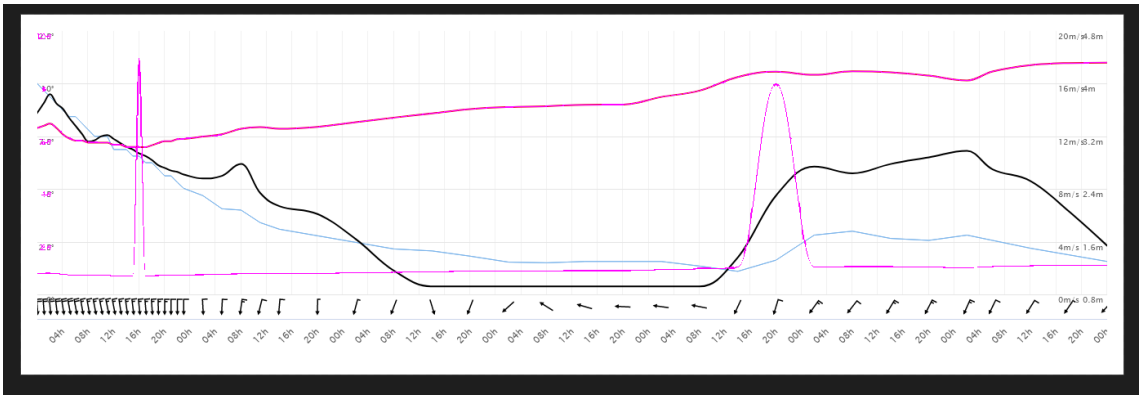

Ilustración 45 AmpVision UI automated visual testing result

Como podemos observar, las imágenes son comparadas con una instantánea anterior y se marca la diferencia, ahora bien, no necesariamente tiene que ser un error puede ser un cambio que hayamos realizado a conciencia por eso se dice que este tipo de pruebas son de resolución humana. Si el cambio es correcto simplemente tendremos que aceptar el nuevo cambio, pero, al contrario, si no lo es deberemos arreglar el fallo.

2.6.4 Github container registry

Todos los paquetes del producto AmpVision UI están subidos a GitHub de manera que se pueden instalar desde cualquier aplicación javascript.

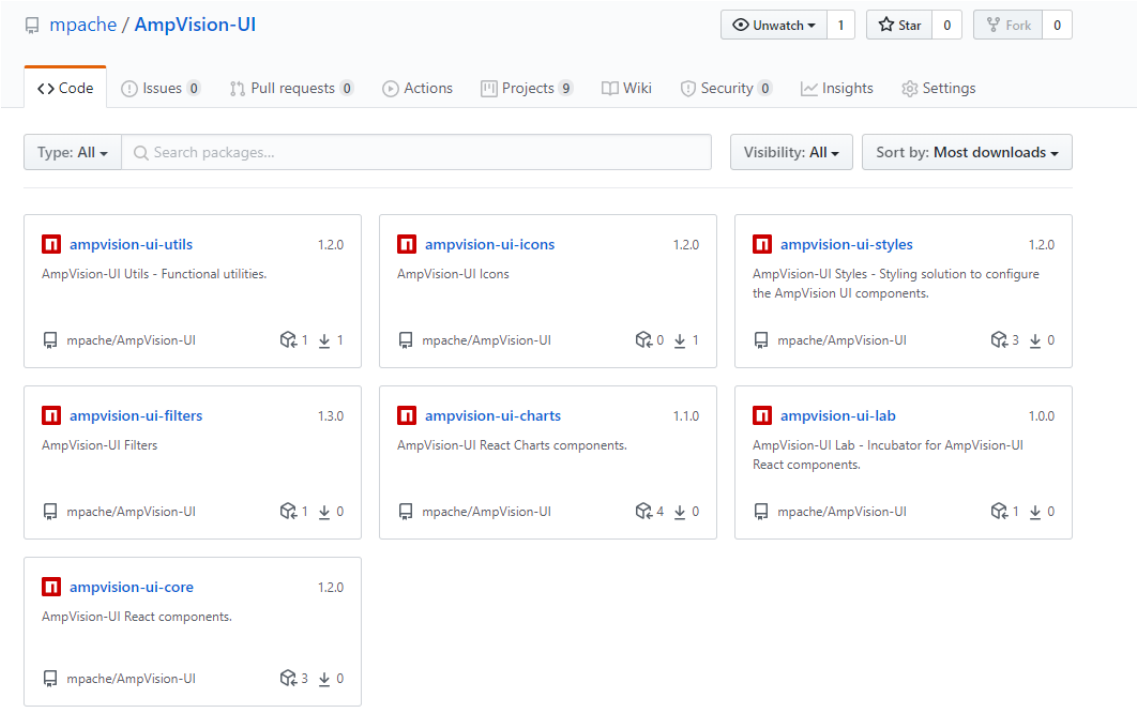

Ilustración 46 AmpVision UI Github Container Registry

2.6.5 Automatización

Todo el repositorio esta pensado para que lance todos los procesos de forma automática, es decir para poder publicar una nueva versión se han de superar todas y cada una de las pruebas y estas pruebas son lanzadas automáticamente.

Las pruebas que el repositorio tiene que pasar para que se acepta una versión son:

- El código tiene que cumplir con los estándares
- Se tienen que superar todas las pruebas unitarias.
- Se tienen que superar todas las pruebas de integración.
- Se tienen que superar todas las pruebas de accesibilidad.
- Se tienen que superar todas las pruebas de automatización visuales.

A simple vista esto parece más trabajo, pero se consigue dotar al producto final de una calidad y confianza extremas.

2.7 Seguridad

Dado que toda la plataforma está basada en contenedores de manera que esta se pueda modularizar uno de los principales problemas es controlar la seguridad de estos puesto que, es posible y así se ha demostrado escalar privilegios en contenedores.

Por otra parte, puesto que todo el código está mantenido en repositorios GIT (github) y, por lo tanto, es importante tener una manera de auditar todo este código en busca de vulnerabilidades.

La plataforma escogida para automatizar y auditar todos los repositorios de los que consta AmpVision es Snyk[33].

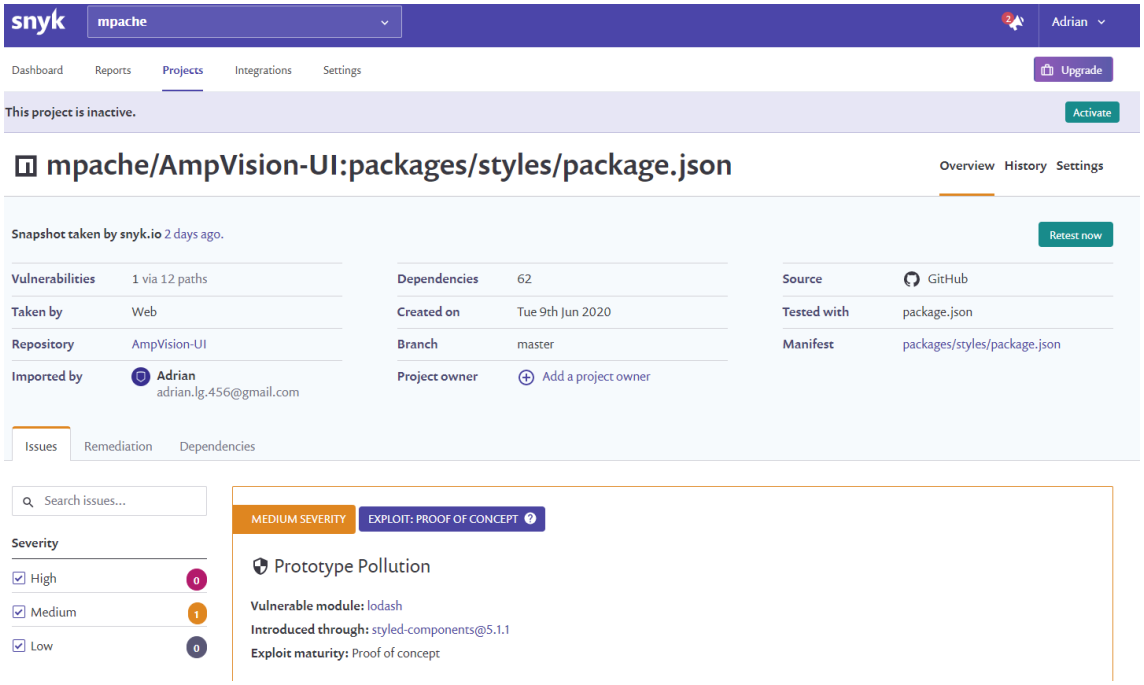

Ilustración 47 – Snyk example

Por último, los entornos están auditados utilizando OpenVas[34], el cual es un escáner de vulnerabilidades con todas las funciones. Sus capacidades incluyen pruebas no autenticadas, pruebas autenticadas, varios protocolos industriales y de Internet de alto y bajo nivel, ajuste de rendimiento para escaneos a gran escala y un potente lenguaje de programación interno para implementar cualquier tipo de prueba de vulnerabilidad.

[33] https://snyk.io/ 11/06/2020 20:09
[34] https://www.openvas.org/ 11/06/2020 20:20

2.8 Ejemplos de uso de la plataforma

Como ejemplo de un proyecto el cual une la plataforma AmpVision con una plataforma totalmente independiente tenemos un software de cámaras de seguridad las cuales pueden contar personas, matriculas y más funcionalidad añadida a la cual se le puede añadir los paneles de medición obtenidos de la plataforma AmpVision.

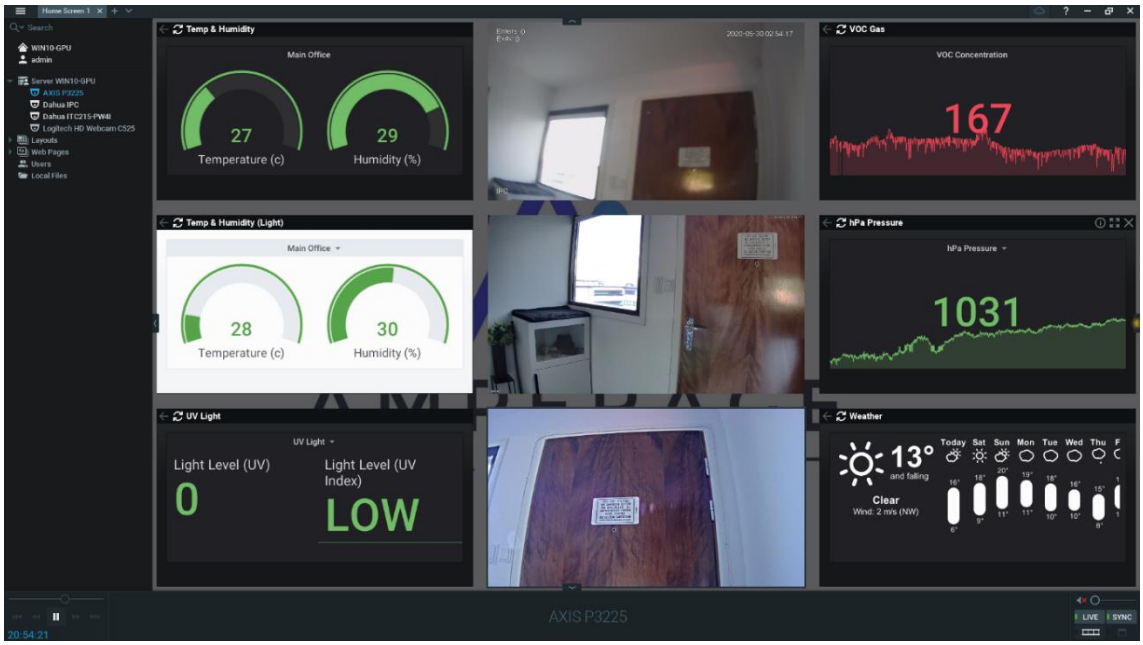

Ilustración 48 Ejemplo de uso AmpVision

Además, también tenemos un proyecto web que está utilizando los componentes de AmpVision UI, en concreto los componentes*: Weather Chart*, *Gantt Diagram, Button* y *Filters.*

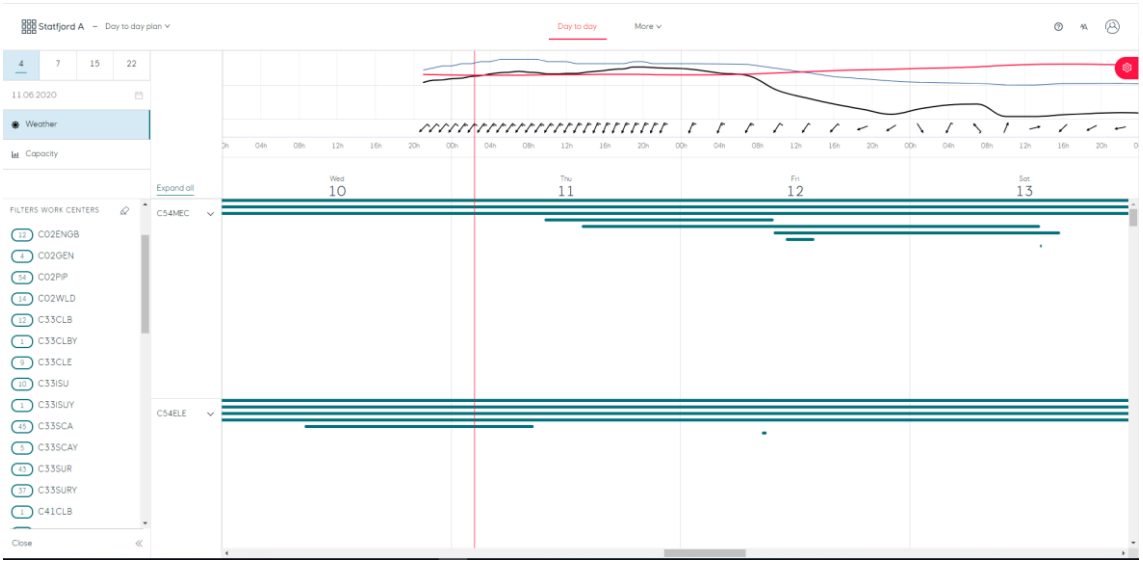

Ilustración 49 Ejemplo de uso AmpVision UI

3. Conclusiones

Durante la realización de este proyecto este ha sido modificado 2 veces puesto que, con la experiencia que se iba obteniendo nuevas puertas y posibilidades se iban abriendo de forma que el proyecto quedaba más robusto y completo.

Posiblemente la mejor lección que se ha aprendido es, compartir experiencias con otros profesionales del sector, estos pueden ayudarte a enfocar el proyecto de la forma adecuada. Durante la realización de este proyecto, fue posible la participación en la grafanaconf, conferencia online sobre Grafana, la cual dio la oportunidad de entablar conversación con otros profesionales que utilizan esta plataforma y ayudaron a dar una nueva visión al proyecto.

Si bien es cierto que, comparando el objetivo inicial con el objetivo final podríamos decir que la parte de AmpVision UI no se ha completado al 100% aunque, esto es debido a que se han añadido nuevas piezas, como el entorno de pruebas de esta librería y, toda la plataforma AmpVision. Estas piezas han complementado el proyecto y le dan un resultado final mucho mejor que el planteado inicialmente.

A nivel de planificación, puesto que se ha seguido una metodología de trabajo iterativa cogiendo muchos aspectos del desarrollo ágil (backlog y tablero Kanban), se ha conseguido superar con creces las expectativas incluso habiendo modificado o mejor dicho expandido el proyecto final que se quería obtener.

Ahora bien, la pieza AmpVision IOT aunque es un buen comienzo, o una buena idea, aún necesita de mucho trabajo, si estos dispositivos tuvieran una brecha de seguridad o bien se quisiera actualizar su firmware actualmente es necesario instalar manualmente dicho firmware, cosa que limita en gran medida su efectividad. Por otro lado, esto también afecta a que entorno se conectan ahora mismo los dispositivos conectan con el entorno de pruebas y para poder modificar esto es necesario actualizar el firmware manualmente. Si bien, esto es un problema se ha encontrado una posible solución para el futuro la cual no se ha podido explorar, esta pasa por utilizar la plataforma o ecosistema platformio[35], en principio utilizando esta plataforma podríamos conseguir que el firmware se actualizara mediante una conexión de internet.

[35] https://platformio.org/ 11/06/2020, 21:50

4. Glosario

- **IOT:** Internet Of Things es un sistema de dispositivos informáticos interrelacionados, máquinas mecánicas y digitales dotadas de identificadores únicos (UID) y la capacidad de transferir datos a través de una red sin necesidad de interacción entre personas o entre personas y computadoras.

- **HaaS:** Hardware as a Service es un modelo de adquisición similar al arrendamiento o a la concesión de licencias en el que el hardware perteneciente a un proveedor de servicios gestionados (MSP) se instala en las instalaciones del cliente y un acuerdo de nivel de servicio (SLA) define las responsabilidades de ambas partes.

- **SaaS:** Software as a Service es un modelo de distribución de software en el que un proveedor tercero aloja aplicaciones y las pone a disposición de los clientes a través de Internet.

- **Kanban:** es un método de gestión y mejora del trabajo en todos los sistemas humanos. Este enfoque tiene por objeto gestionar el trabajo equilibrando las demandas con la capacidad disponible y mejorando el manejo de los cuellos de botella a nivel de sistema.

- **Docker:** conjunto de productos de plataforma como servicio (PaaS) que utiliza la virtualización a nivel de sistema operativo para entregar software en paquetes llamados contenedores. Los contenedores están aislados entre sí y agrupan su propio software, bibliotecas y archivos de configuración; pueden comunicarse entre sí a través de canales bien definidos.

- **Edge Router:** un enrutador especializado situado en el límite de una red que permite a una red interna conectarse a redes externas.

- **HTTP:** (Hypertext Transfer Protocol) es el protocolo de comunicación que permite las transferencias de información en la World Wide Web

- **TCP:** (Transmission Control Protocol) capa intermedia entre el protocolo de red (IP) y la aplicación.

- **MQTT:** protocolo de red ligero de publicación-suscripción que transporta los mensajes entre los dispositivos.

5. Bibliografía

[1] **IOT** - Internet of things
[2] **HaaS** – Hardware as a Service
[3] **SaaS** – Software as a Service
[4] **Freemium**: unión de las palabras “free” y “premium” para indicar que un servicio es gratuito, pero tiene partes de pago.
[5] https://www.fitbit.com/no/whyfitbit 09/06/2020, 12:50
[6] **Leasing**: arrendamiento
[7] **Dashboard:** representación gráfica de las principales métricas
[8] https://www.electronjs.org/ 09/06/2020, 17:38
[9] **Kanban:** tablero utilizado en el desarrollo de software ágil.
[10] **Docker:** contenedores, una unidad estandarizada de software (Ver glosario para explicación).
[11] https://docs.traefik.io/ 09/06/2020, 19:43
[12] https://www.rabbitmq.com/ 09/06/2020, 19:45
[13] https://docs.influxdata.com/telegraf/v1.14/ 09/06/2020, 19:47
[14] https://docs.influxdata.com/influxdb/v1.8/ 09/06/2020, 19:49
[15] https://grafana.com/docs/grafana/latest/ 09/06/2020, 19:50
[16] https://jestjs.io/docs/en/getting-started 09/06/2020, 19:53
[17] https://github.com/enzymejs/enzyme 09/06/2020, 19:55
[18] https://github.com/lerna/lerna 09/06/2020, 20:04
[19] https://nodejs.org/en/docs/ 09/06/2020, 20:06
[20] https://docs.npmjs.com/ https://classic.yarnpkg.com/en/docs/ 09/06/2020, 20:07
[21] https://babeljs.io/docs/en/ 09/06/2020, 20:09
[22] https://eslint.org/ 09/06/2020, 20:10
[23] https://prettier.io/ 09/06/2020, 20:11
[24] https://commitlint.js.org/#/ 09/06/2020, 20:13
[25] https://en.wikipedia.org/wiki/Andr%C3%A9-Marie_Amp%C3%A8re 09/06/2020, 20:18
[26] **middleware:** programa informático que proporciona servicios a aplicaciones de software más allá de los disponibles en el sistema operativo.
[27] https://www.arduino.cc/en/main/docs 09/06/2020, 20:39
[28] **look&feel:** metáfora utilizada dentro del entorno de marketing para poder dar una imagen única a los productos.
[29] **Microservices:** consiste en construir una aplicación como un conjunto de pequeños servicios.
[30] **Monorepo:** estrategia de desarrollo de software en la que el código de muchos proyectos se almacena en el mismo repositorio.
[31] https://bradfrost.com/blog/post/atomic-web-design/ 11/06/2020, 17:06
[32] https://www.w3.org/WAI/standards-guidelines/ 11/06/2020 21:04
[33] https://snyk.io/ 11/06/2020 20:09
[34] https://www.openvas.org/ 11/06/2020 20:20
[35] https://platformio.org/ 11/06/2020, 21:50

6. Anexos

Figma Design

A continuación, se muestran los diseños realizados en la plataforma figma.

Colors

Ilustración 50 – Colors

Typography

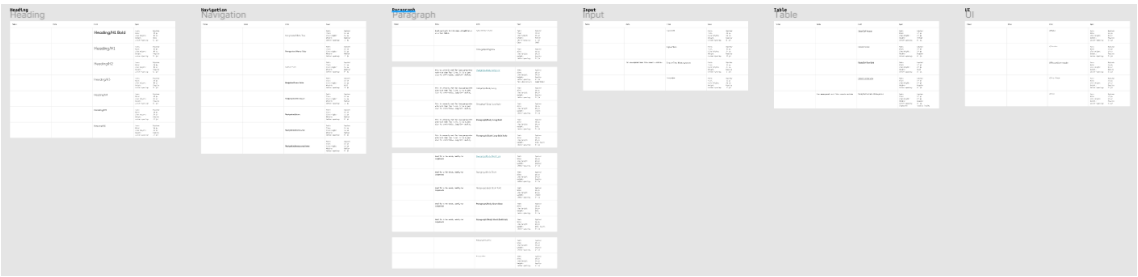

Ilustración 51 - Typographys

Icons

Accessibility

Accessibility

Token	Role	Link	Component
		accessibility/pregnant-woman	
		accessibility/accessible	
		accessibility/wheelchair	
		accessibility/accessible-forward	
		accessibility/language	
		accessibility/google-translate	
		accessibility/translate	
		accessibility/hearing	

Ilustración 52 – Accessibility Icons

Arrows

Arrows

Token	Role	Link	Component
		arrows/chevron-down	
		arrows/chevron-left	
		arrows/chevron-right	
		arrows/chevron-up	
		arrows/arrow-back	
		arrows/arrow-down	
		arrows/arrow-forward	
		arrows/arrow-up	
		arrows/arrow-back-ios	
		arrows/arrow-forward-ios	
		arrows/arrow-drop-down	
		arrows/arrow-drop-up	
		arrows/arrow-drop-right	
		arrows/arrow-drop-left	
		arrows/unfold-less	
		arrows/unfold-more	
		arrows/first-page	
		arrows/last-page	
		arrows/subdirectory-arrow-left	
		arrows/subdirectory-arrow-right	
		arrows/swap-horizontal	
		arrows/swap-horizontal-circle	
		arrows/swap-vertical	
		arrows/swap-vertical-circle	

Ilustración 53 – Arrows Icons

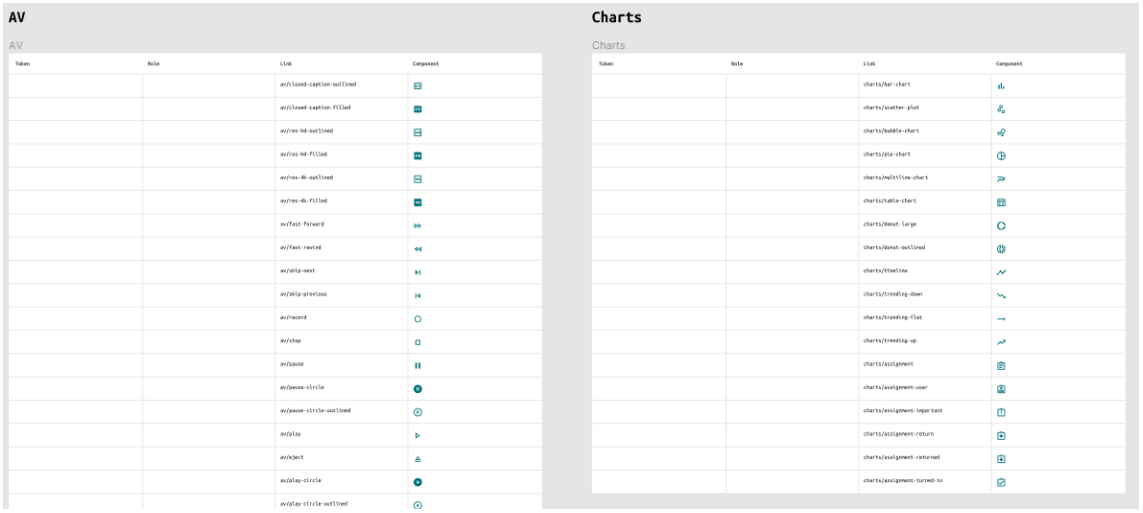

Ilustración 54 – AV and Charts Icons

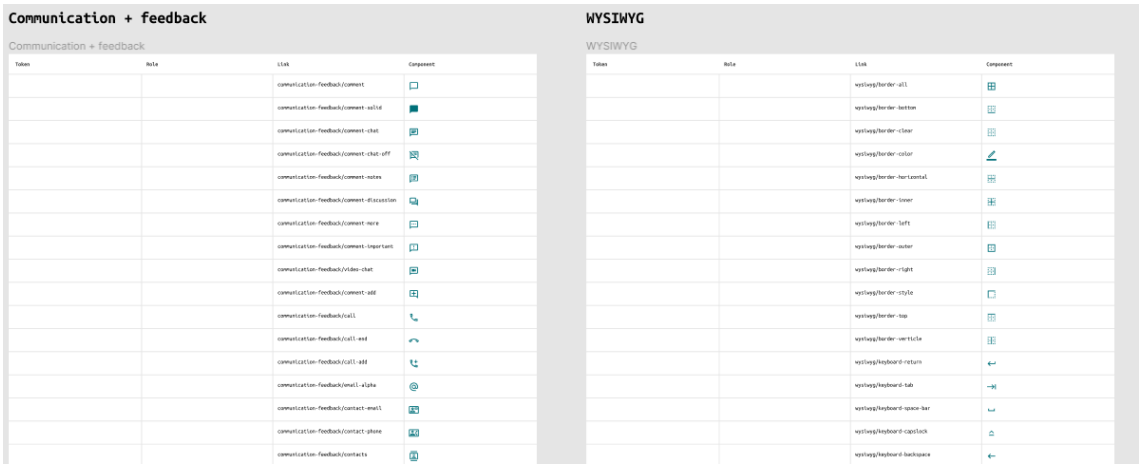

Ilustración 55 – Feedback and wysiwyg icons

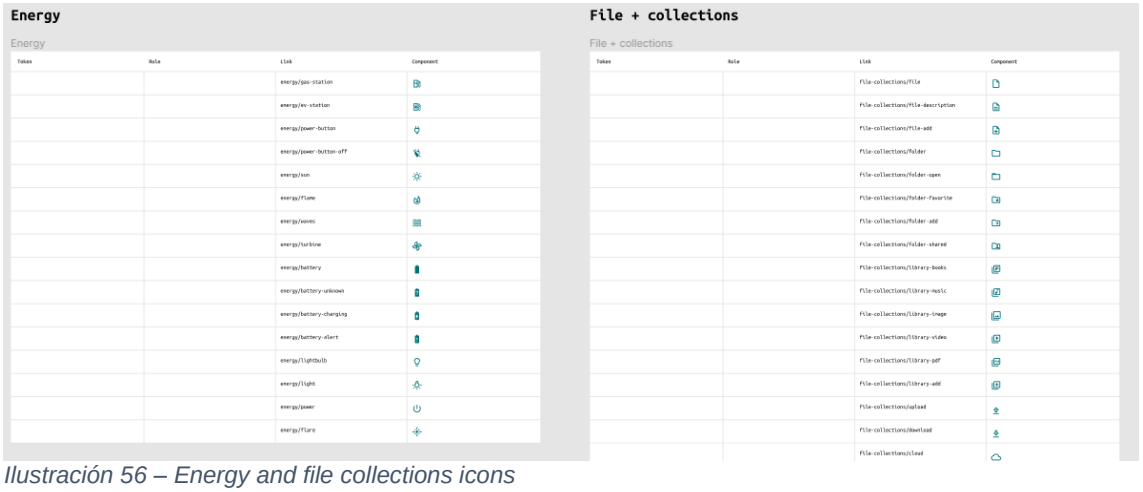

Ilustración 56 – Energy and file collections icons

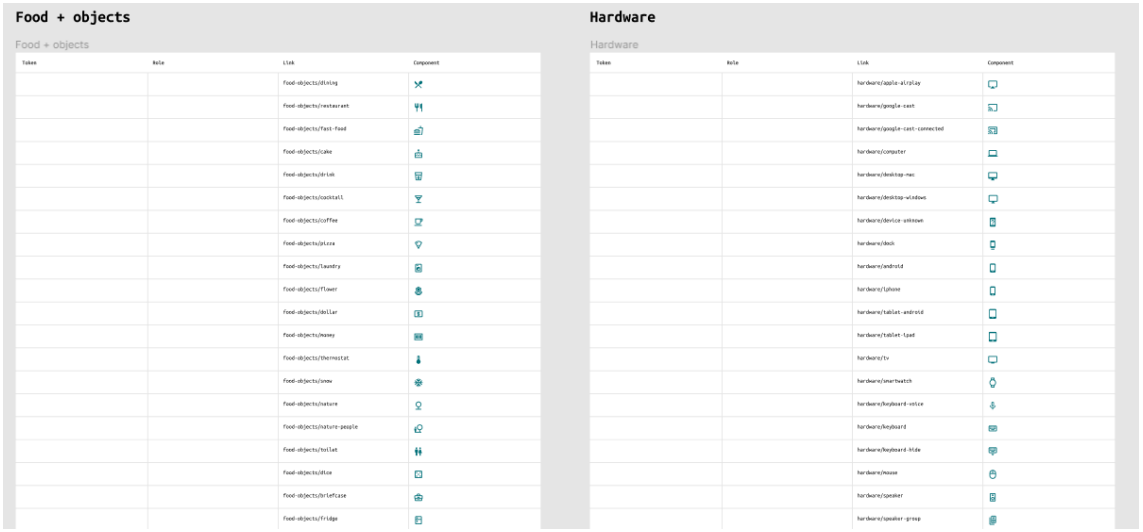

Ilustración 57 – Food and hardware icons

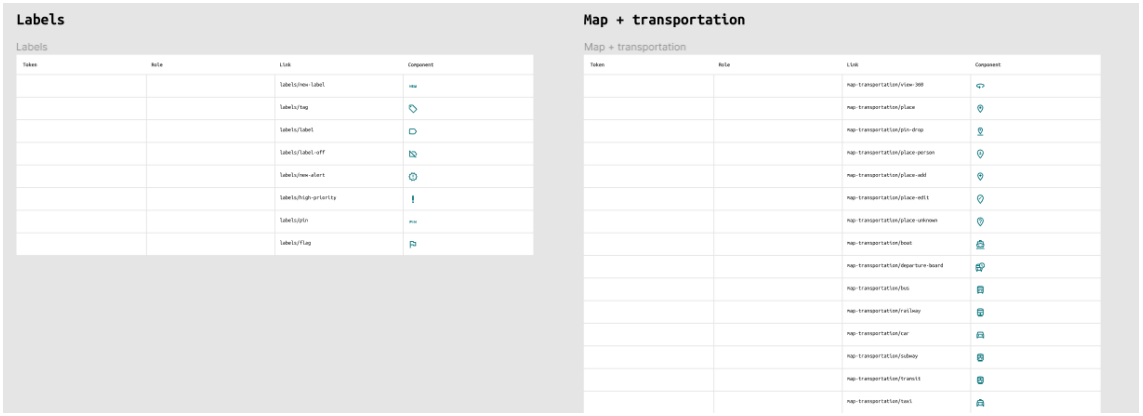

Ilustración 58 – Labels and transport icons

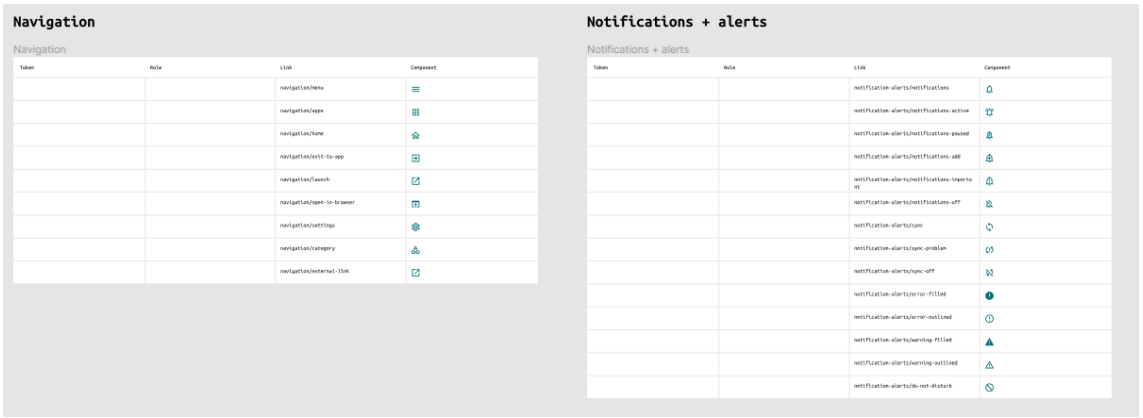

Ilustración 59 – Navigation and notification alerts

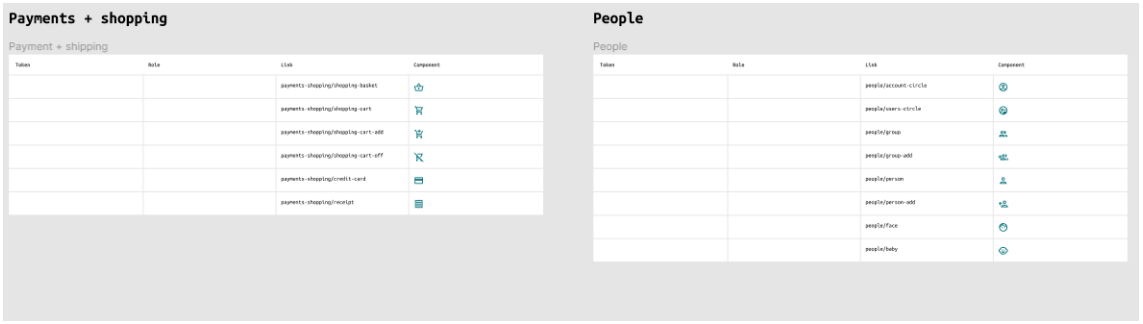

Ilustración 60 – Payments and people

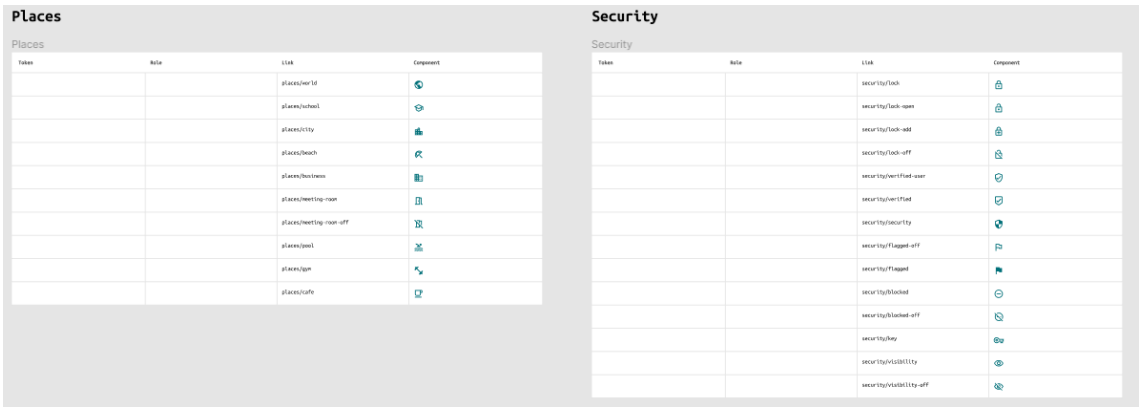

Ilustración 61 – Places and Security

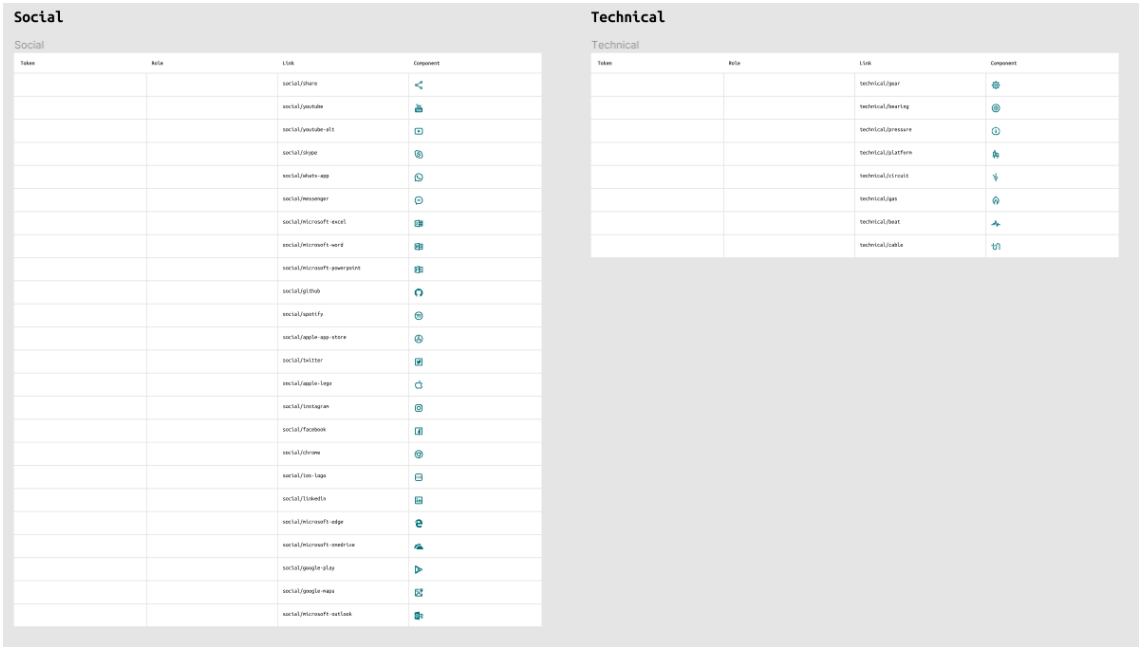

Ilustración 62 – Social and Technical

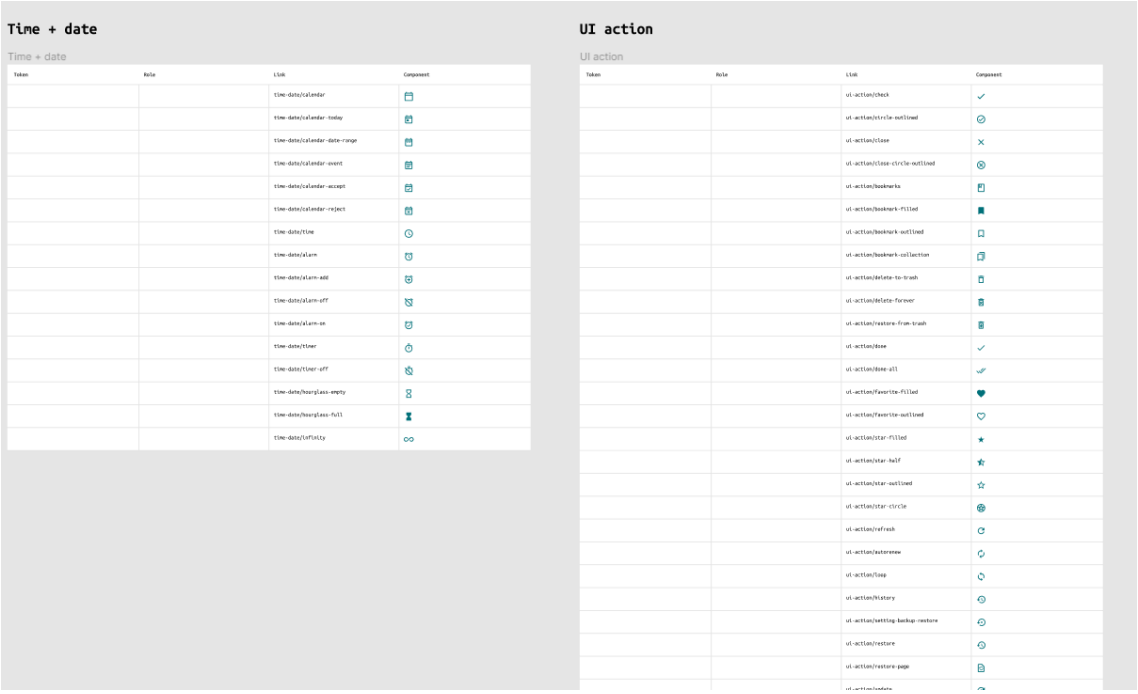

Ilustración 63 – Time and UI action

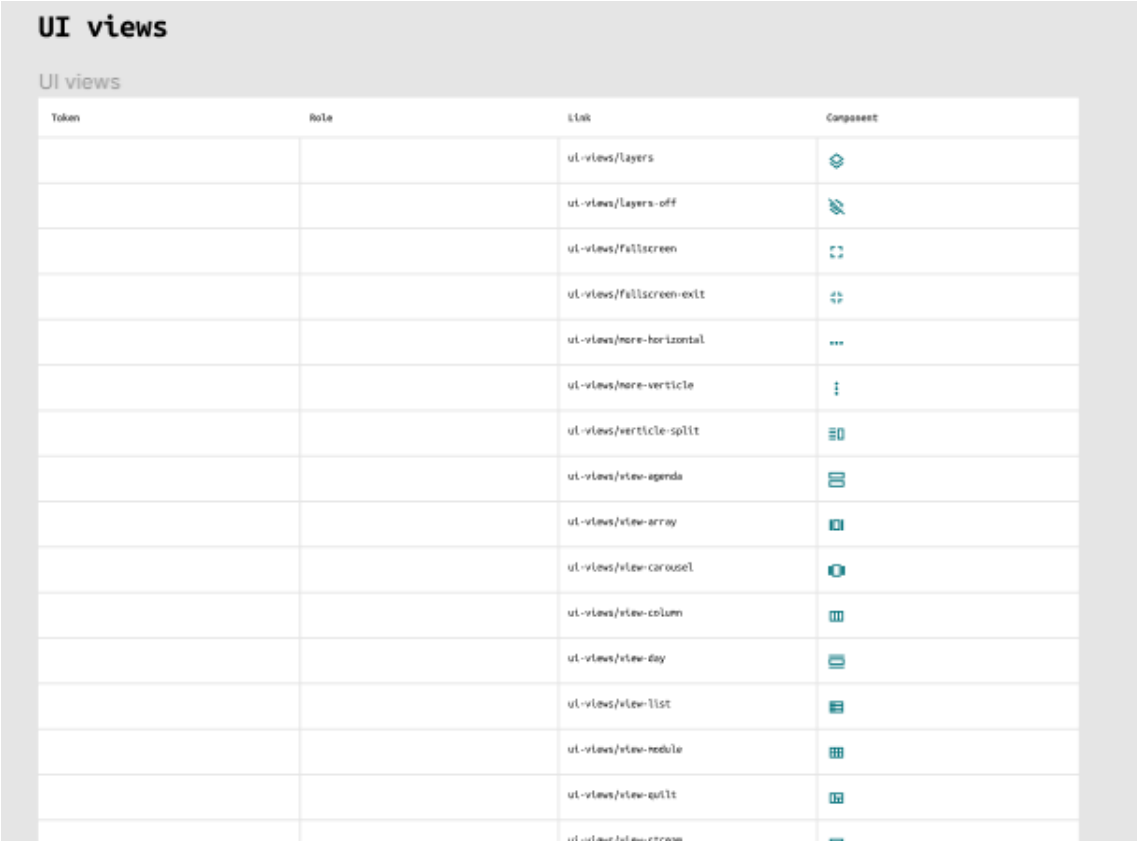

Ilustración 64 – UI icons

Accordion

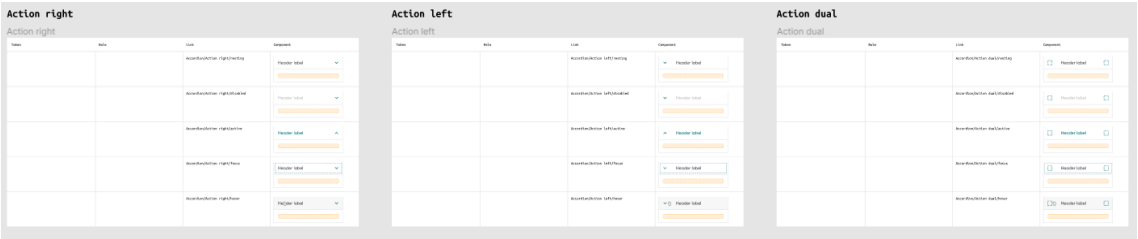

Ilustración 65 - Accordion

Primary Button

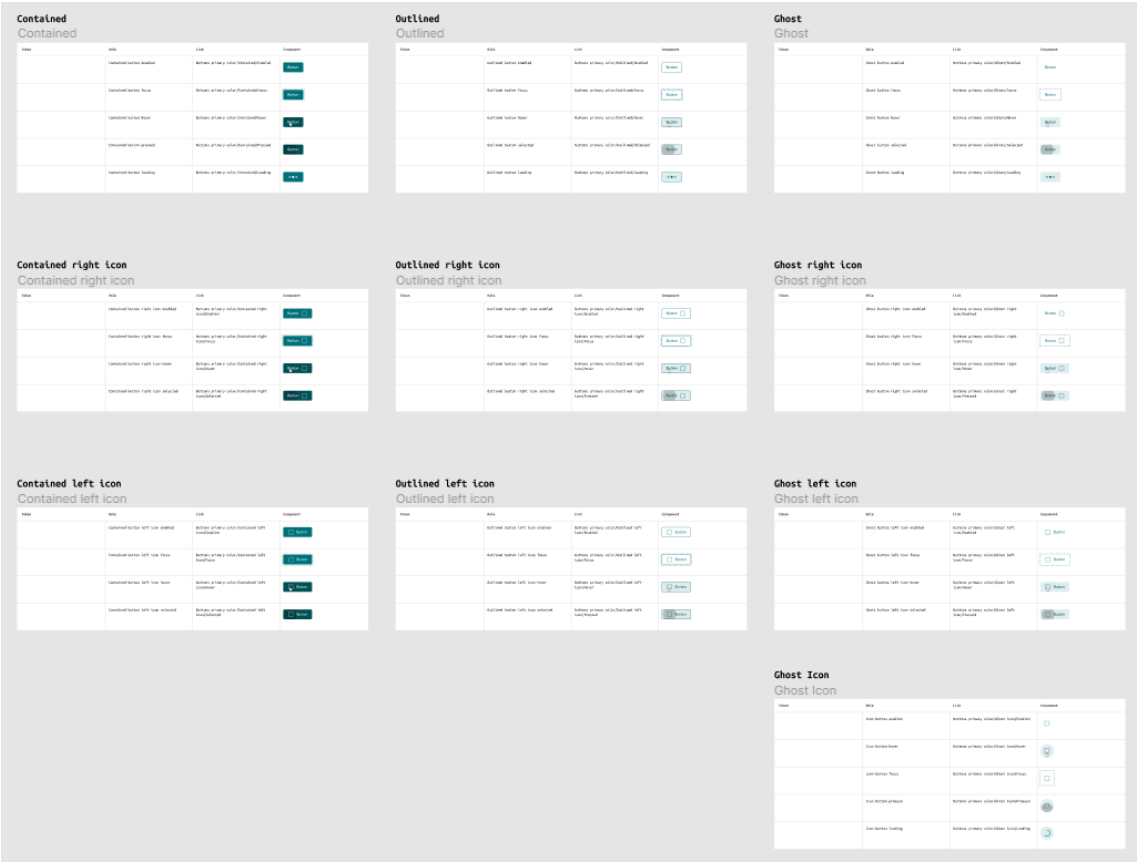

Ilustración 66 – Primary button

Secondary Button

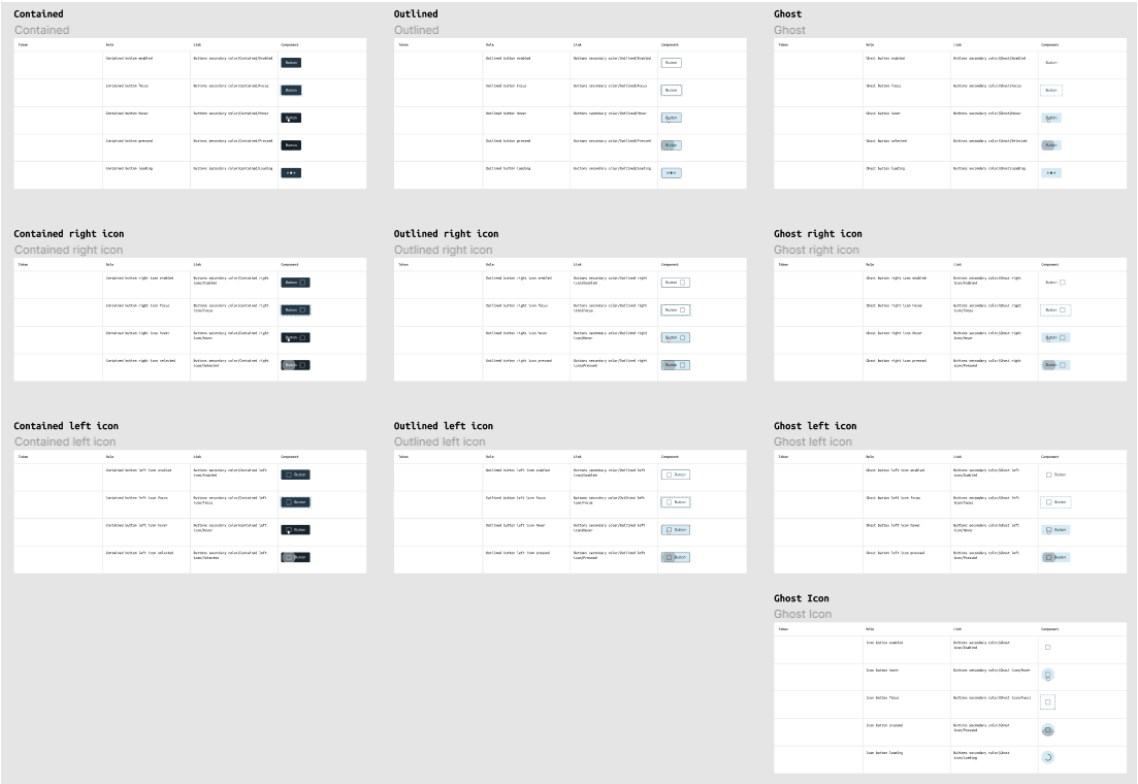

Ilustración 67 – Secondary button

Danger Button

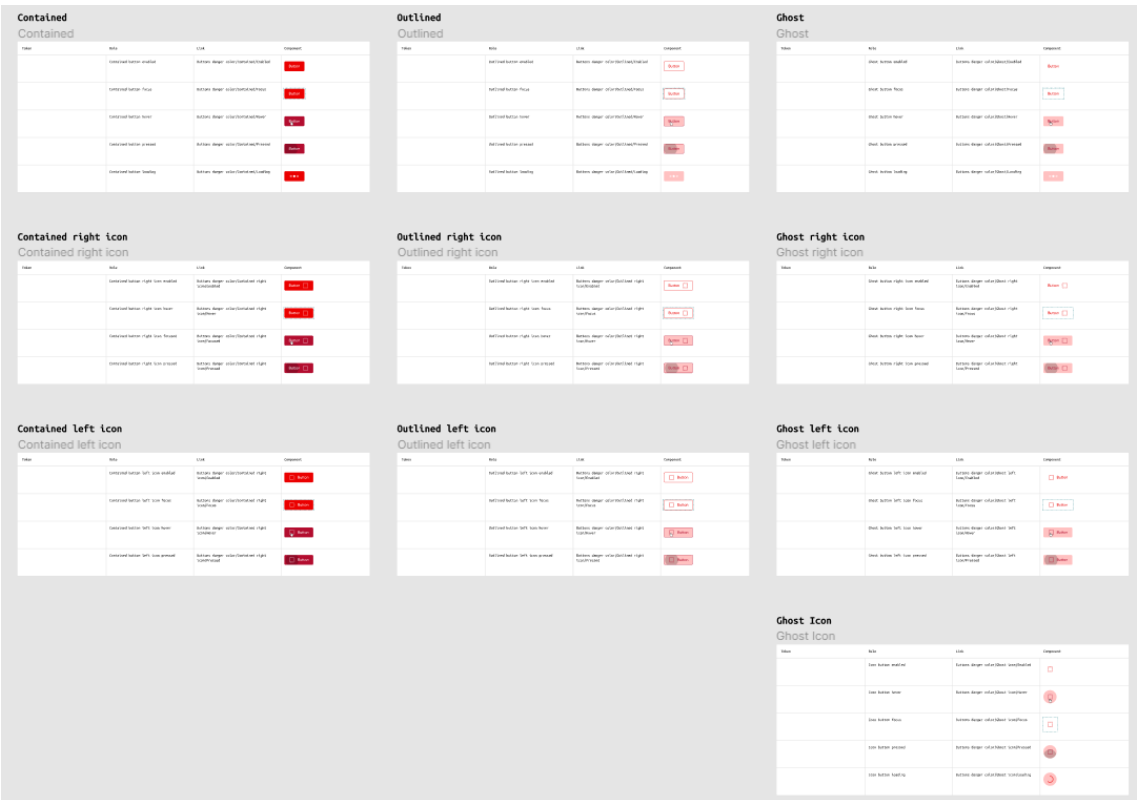

Ilustración 68 – Danger button

Disabled Button

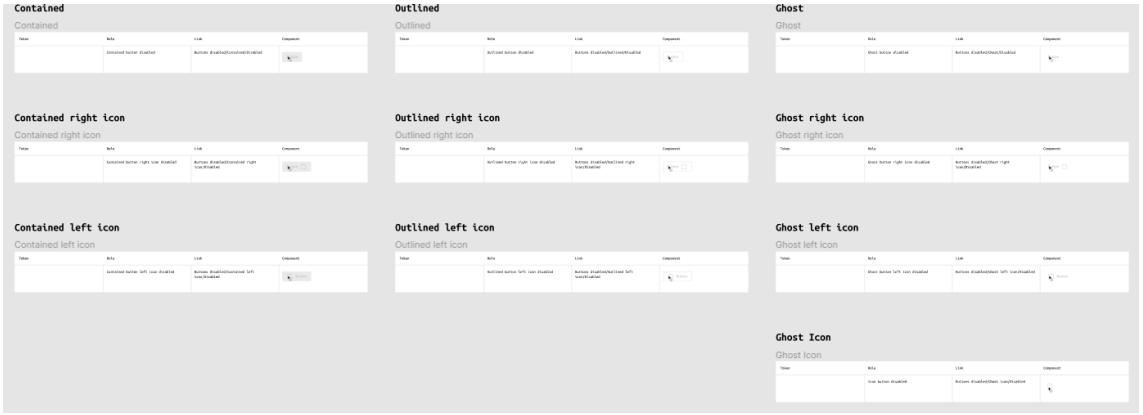

Ilustración 69 – Disabled button

Buttons Toggle (Group Buttons)

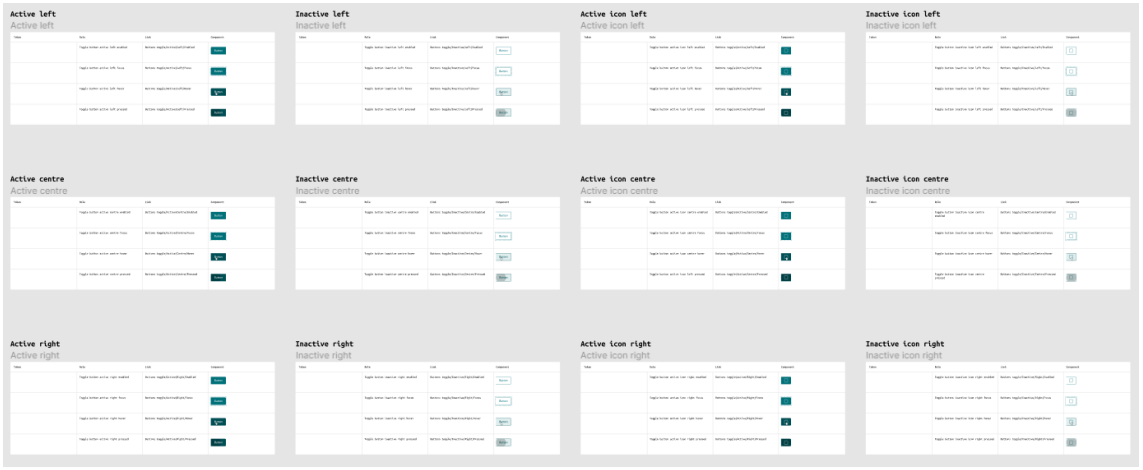

Ilustración 70 – Group buttons

Cards

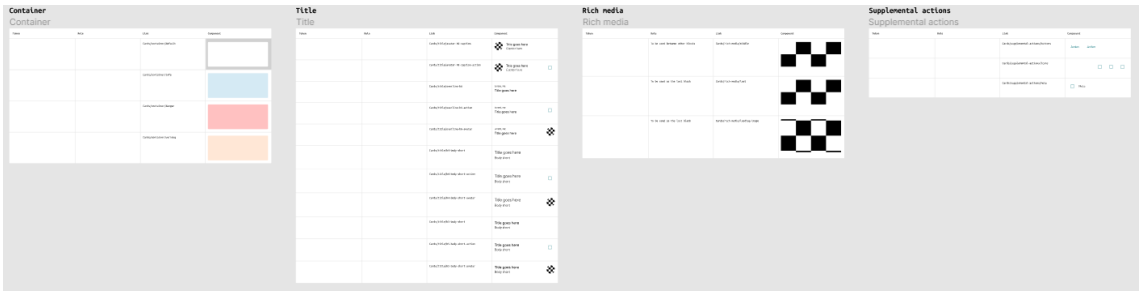

Ilustración 71 - Cards

Date picker

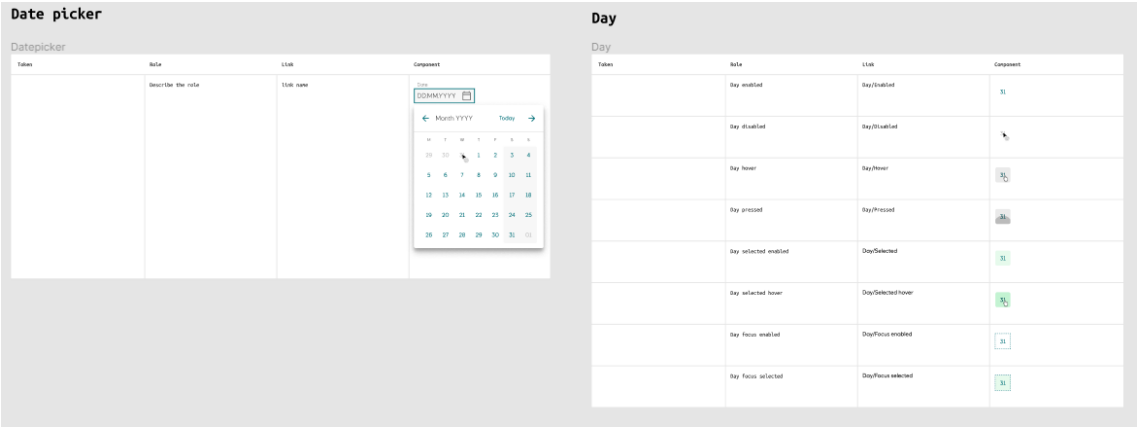

Ilustración 72 – Date picker

Tabs

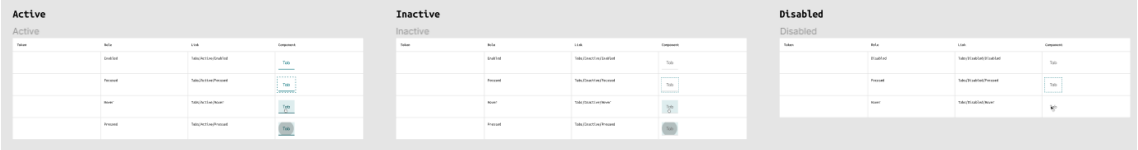

Ilustración 73 – Tabs

Selection controls

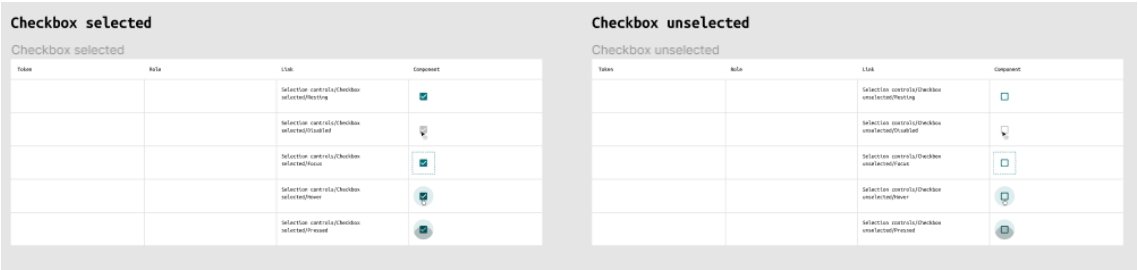

Ilustración 74 – Checkbox

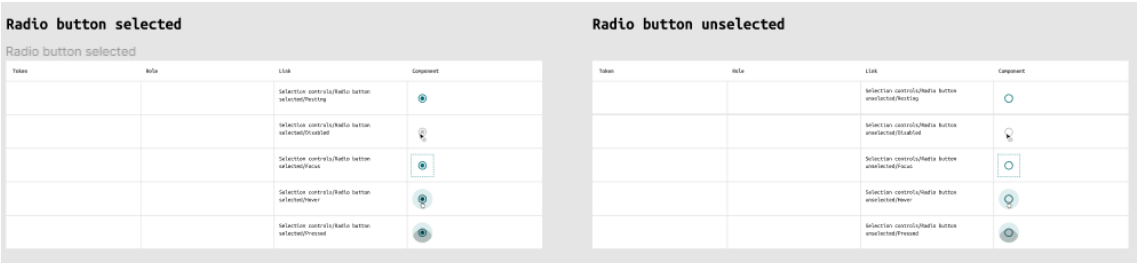

Ilustración 75 - Radios