

Informe final Capstone:  
Desarrollo de ingeniería  
Proyecto  
“Energy Meter”



**Alumnos**

Francisco Galdames

Jorge Parra

Gabriel Soto

**Profesores evaluadores**

Julian Corbalán

Helton Smith

Francisco Valdivieso

**Profesor Asignatura**

Helton Smith

4 de diciembre 2024

# Índice

[**Índice 2**](#_heading=h.1fob9te)

[**Abstract (español) 4**](#_heading=h.3znysh7)

[**Abstract (English) 4**](#_heading=h.2et92p0)

[**Introducción 5**](#_heading=h.tyjcwt)

[**Propuesta de valor 6**](#_heading=h.1t3h5sf)

[**Metodología 7**](#_heading=h.4d34og8)

[**Desarrollo 10**](#_heading=h.2s8eyo1)

[Infraestructura del proyecto 10](#_heading=h.17dp8vu)

[¿Por qué optamos por este stack de tecnologías? 11](#_heading=h.3rdcrjn)

[Diagramación 11](#_heading=h.26in1rg)

[Vista Despliegue 11](#_heading=h.lnxbz9)

[Vista Escenarios 14](#_heading=h.2jxsxqh)

[Vista Física 15](#_heading=h.z337ya)

[Vista Lógica 16](#_heading=h.4i7ojhp)

[Vista Procesos 33](#_heading=h.1pxezwc)

[Capturas Aplicación 37](#_heading=h.49x2ik5)

[Ambiente y flujo de programación 46](#_heading=h.2p2csry)

[Features en detalle 47](#_heading=h.147n2zr)

[Home 48](#_heading=h.3o7alnk)

[Login 48](#_heading=h.23ckvvd)

[Register 49](#_heading=h.ihv636)

[Olvidé mi contraseña 50](#_heading=h.32hioqz)

[Dashboard 50](#_heading=h.1hmsyys)

[Dispositivos 50](#_heading=h.41mghml)

[Dispositivo 51](#_heading=h.2grqrue)

[Agregar dispositivos 51](#_heading=h.vx1227)

[Modificar dispositivo 51](#_heading=h.3fwokq0)

[Cuenta 52](#_heading=h.1v1yuxt)

[Modificar contraseña 52](#_heading=h.4f1mdlm)

[Notificaciones 53](#_heading=h.2u6wntf)

[Cerrar sesión 53](#_heading=h.19c6y18)

[Esquemas Oracle Cloud DB 53](#_heading=h.3tbugp1)

[Esquemas Firebase DB 55](#_heading=h.28h4qwu)

[Codificaciones de ejemplo 55](#_heading=h.nmf14n)

[Frameworks explicados 62](#_heading=h.37m2jsg)

[NextJS 62](#_heading=h.1mrcu09)

[React 62](#_heading=h.46r0co2)

[FastAPI 62](#_heading=h.2lwamvv)

[Herramientas utilizadas para el desarrollo 62](#_heading=h.111kx3o)

[Esp32 62](#_heading=h.3l18frh)

[Sistemas operativos 63](#_heading=h.206ipza)

[Ubuntu server 63](#_heading=h.4k668n3)

[Fedora linux 63](#_heading=h.2zbgiuw)

[Arch linux 63](#_heading=h.1egqt2p)

[Windows 10 y 11 64](#_heading=h.3ygebqi)

[Editores de códigos y entornos de desarrollo 64](#_heading=h.2dlolyb)

[Visual Studio Code 64](#_heading=h.sqyw64)

[Arduino IDE 64](#_heading=h.3cqmetx)

[Nvim 64](#_heading=h.1rvwp1q)

[Oracle Data Modeler 64](#_heading=h.4bvk7pj)

[Oracle SQL Developer 64](#_heading=h.2r0uhxc)

[Misceláneo 65](#_heading=h.1664s55)

[Google Chrome 65](#_heading=h.3q5sasy)

[Firefox 65](#_heading=h.25b2l0r)

[Postman 65](#_heading=h.kgcv8k)

[Runtimes 65](#_heading=h.34g0dwd)

[NodeJS 20 65](#_heading=h.1jlao46)

[Python3 3.10 65](#_heading=h.43ky6rz)

[GCC (Arduino) 66](#_heading=h.2iq8gzs)

[**Conclusiones Individuales 66**](#_heading=h.xvir7l)

[Reflexión grupal 67](#_heading=h.3hv69ve)

[**Competencias perfil de egreso 67**](#_heading=h.1x0gk37)

[**Bibliografías 68**](#_heading=h.4h042r0)

# Abstract (español)

El proyecto de Sistema de Gestión y Control de Energía Doméstica “Energy Meter” que realizamos buscando en solucionar la problemática del aumento de costo de la luz, en generar principalmente conciencia gracias a la página web que muestra el gasto eléctrico que es información enviada por una API que a su vez conectada a una base de datos alojada en un servidor que aloja los datos recolectados de dispositivos conectados a los aparatos eléctricos en el domicilio, el ¿Cómo fue construido el proyecto? Comenzamos primero en establecer la metodología de trabajo ágil Sprint como principal forma de organizar las tareas entre nosotros por consiguiente fue el decidir los roles de cada integrante del grupo, decidir las distintas herramientas que se ocuparan para el desarrollo decidiéndose para la página web el framework Next.js con el lenguaje de programación TypeScript siendo alojada con la plataforma Versel, la API utilizó la herramienta Postman para después usar el framework FastAPI con el lenguaje de programación Python conectando con la página web y a la base de datos Oracle Server creado con el lenguaje SQL y posteriormente de acuerdo a cómo transcurría el desarrollo se decidió para la simulación del dispositivo usar Godot.

# Abstract (English)

The project of control and management of domestic energy, “Energy Meter” who’s we make it searching to resolve the raising cost of the electric bill problem, generating awareness thanks to the web page that shows the electricity cost, information sented from an API who's at the same time connected to a database hosting in a server whose keep the collected data from devices connected to different electrical appliances in the residence. How was the project built? We started establishing the agile methodology and Sprint as the principal way to organize tasks between us in the project. After in the group we decide which role it will do, every integrant decides the different tools to work in the project development, in this case decide for the web page the framework Next.js whit the programming language Typescript being hosted on the platform Versel, the API was made using the tool called Postman after using the framework called FastAPI with the programming language Python connected to the web page and to the database Oracle Server created with the programming languageSQL, subsequently, how the development was going finally decides to use Godot for the device simulation.

# Introducción

Este informe busca mostrar el desarrollo de un proyecto que busca desarrollar una aplicación multiplataforma desde su infraestructura, código y producto final, la cual está diseñada para que los consumidores puedan monitorear su consumo eléctrico en tiempo real a través de métricas detalladas, desglosando por dispositivos, considerando el precio del consumo y mostrando comparativas históricas. Buscando fomentar el uso responsable de la electricidad.

La motivación de este proyecto se remonta en el contexto que vive la ciudadanía chilena, habiendo un notorio aumento de los precios de las cuentas eléctricas a mediado del 2024, el aumento exponencial del uso de dispositivos inteligentes que requieren estar encendidos la mayoría de su vida útil, y la falta de accesibilidad por parte de las empresas eléctricas para notificar el consumo de las casas en tiempo real. Bajo este escenario, resulta imperativo la necesidad de soluciones que promuevan la sostenibilidad, optimizar el consumo energético y reducir el costo de las personas.

Se buscará dar una visión general del proyecto, remarcando la relevancia que tendría en un contexto actual como el potencial impacto que tendría en la gestión de la electricidad de los usuarios.

# 

# Propuesta de valor

Buscamos ofrecer una solución integral para la administración de electricidad doméstica, solución que aborda la creciente necesidad de optimización de gastos a través de una aplicación y un ecosistema compuesto por la integración entre dispositivos, y un dashboard que presenta diversas métricas detalladas en tiempo real. Energy meter capacita a los usuarios para:

* **Tomar decisiones informadas** sobre su consumo eléctrico
* **Reducir costos** gracias a la identificación de ciertos patrones del propio cliente
* **Contribuir a la sostenibilidad,** incentivando a un consumo eléctrico responsable.

Esto se logra bajo la combinación de diversas tecnologías de punta, logrando que el sistema sea robusto, escalable y accesible.

# Metodología

Para desarrollar este proyecto, hemos optado por utilizar un **enfoque ágil** debido a que es lo que más manejamos y nos permite tener ciertas flexibilidades al momento de hacer cambios de último momento, eso sin mencionar que con las herramientas existentes hoy en día, podemos generar una documentación rápida usando plataformas de desarrollo ágil, por ejemplo trello, jira, github projects, etc. Para esta ocasión, nosotros estamos trabajando con github projects gracias a a la integración directa que ofrece con github; esto nos permite tener una trazabilidad y administración de tareas de una forma rápida, directa y más intuitiva a comparación de otras plataformas similares (esto viéndolo desde un punto de vista de un equipo de programadores).

Los roles que se han definido en este trabajo entre los integrantes del equipo son los siguientes:

**Francisco Galdames:**

* Apoyo en Base de Datos
* Desarrollador de Simulación
* Desarrollador API

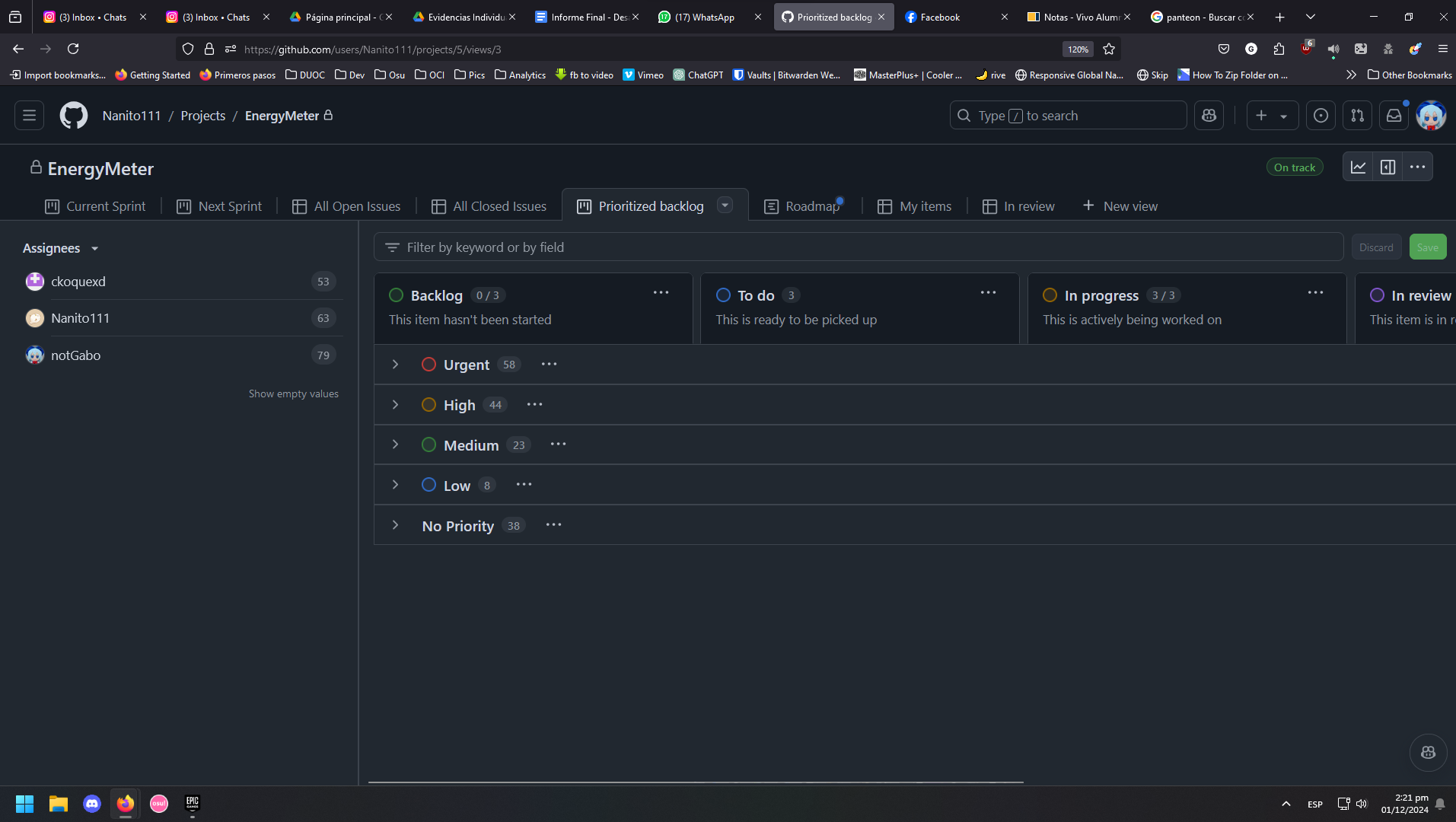
**Jorge Parra:**

* Administrador de Base de Datos
* Apoyo programación

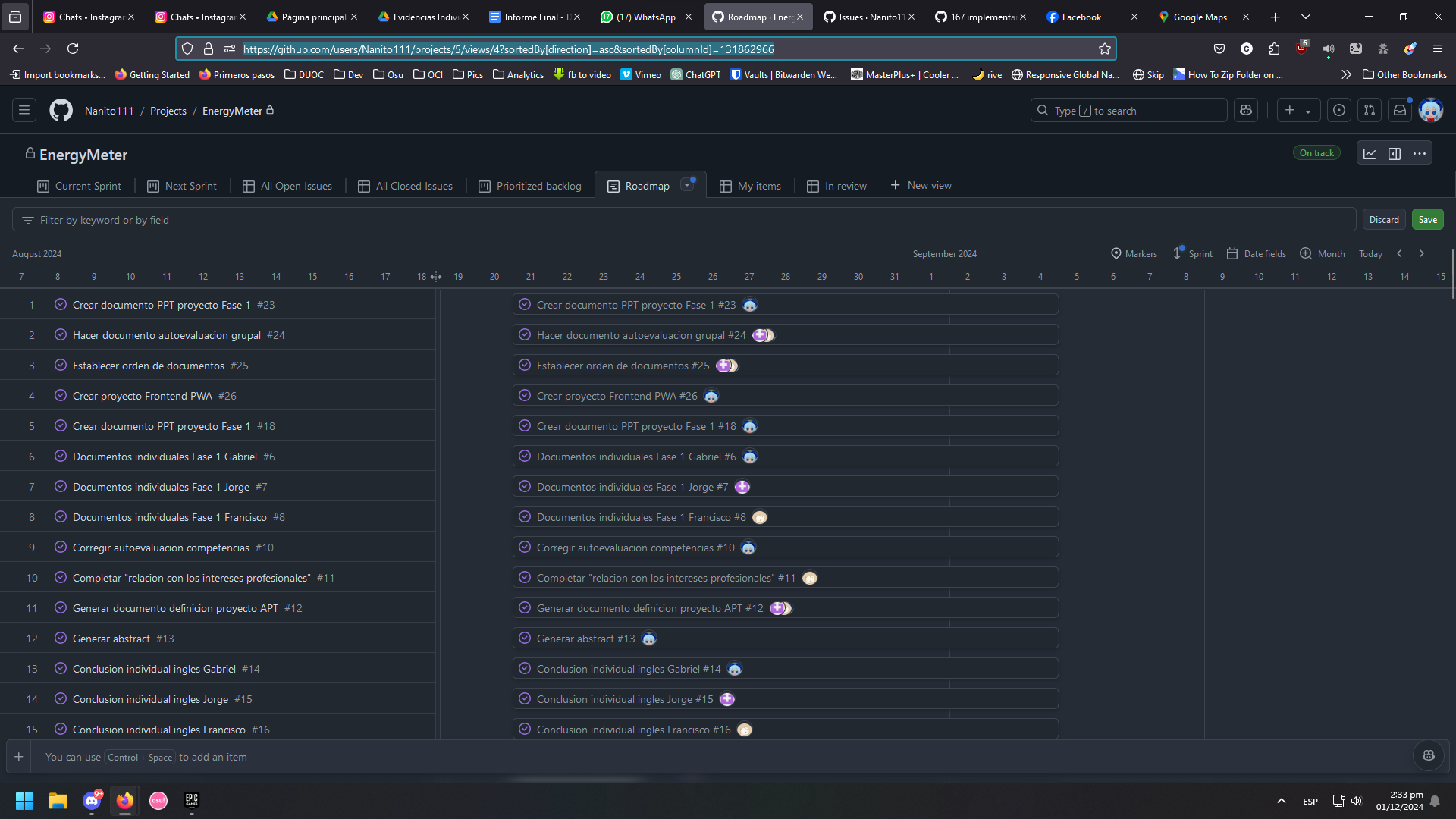
**Gabriel Soto:**

* Scrum Master
* Desarrollador página Web y App móvil

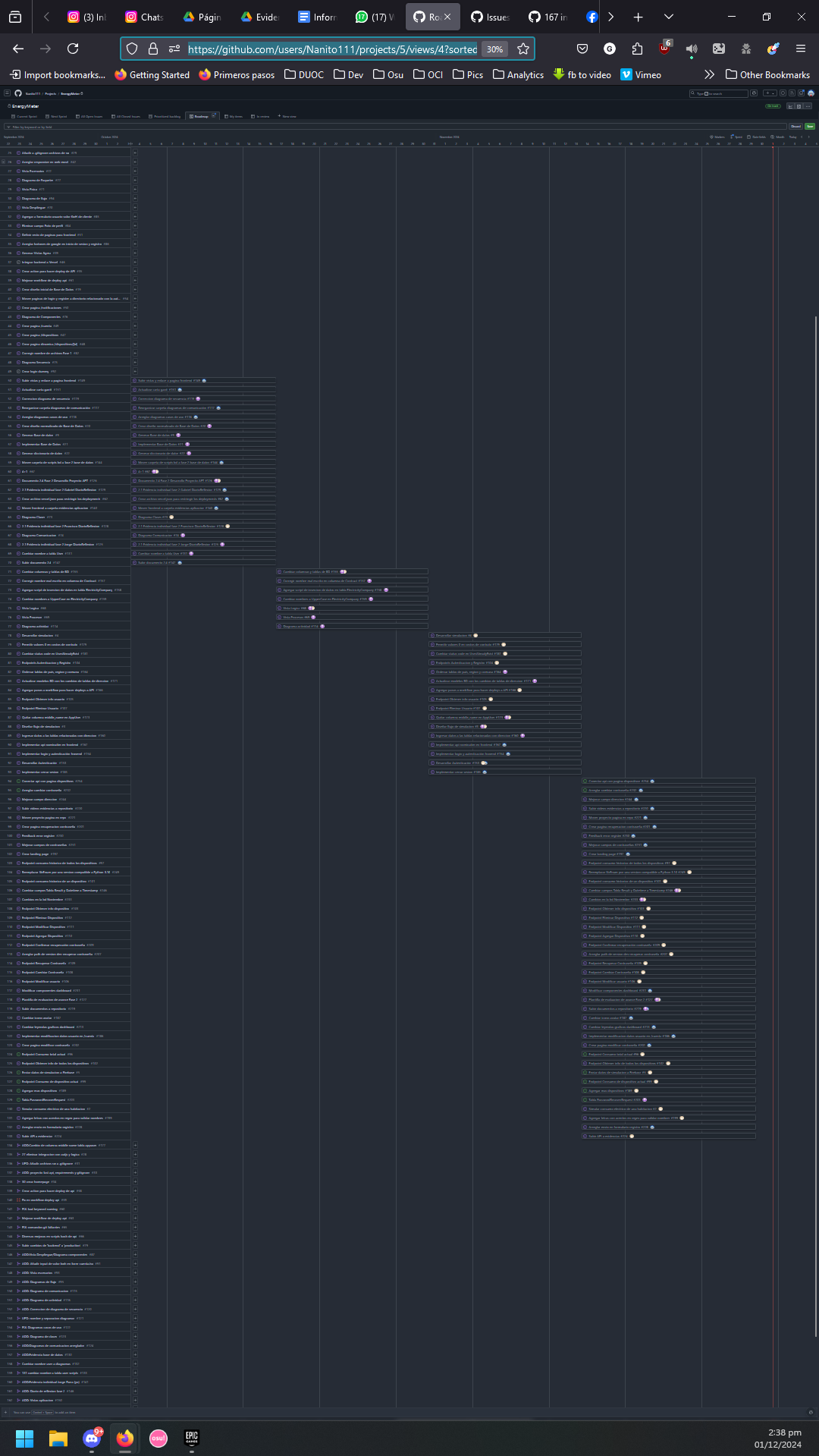
A continuación, se mostraran algunas evidencias referentes al desarrollo con github projects:



Vista del backlog de tareas en github projects, proyecto EnergyMeter



Preview vista roadmap organizada por sprints, equivalente a carta gantt en github projects. [Enlace directo [Requiere autenticación]](https://github.com/users/Nanito111/projects/5/views/4?sortedBy%5Bdirection%5D=asc&sortedBy%5BcolumnId%5D=131862966)

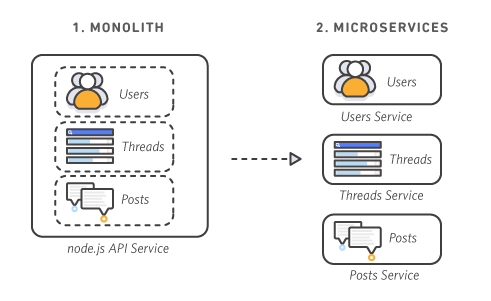


Vista roadmap en un zoom al 300%

# Desarrollo

## Infraestructura del proyecto

Este proyecto está subdividido en 4 hilos principales que son vitales para el funcionamiento adecuado del proyecto, cada rama está hosteada en un servicio dedicado para su funcionamiento, haciendo que este sea una arquitectura basada en microservicios



comparación de ejemplo de arquitectura monolítica y microservicios

Los hilos principales de los cuales depende nuestro proyecto son los siguientes:

1. Repositorio github ([Rama production](https://github.com/Nanito111/CAPSTONE_002D/)): Rama madre de las ramas posteriores. Esta rama sera la presentada para la comisión y contiene el contenido estable de todo lo programado en las ramas que nazcan a partir de esta además de la documentación necesaria del proyecto
   1. [Rama frontend](https://github.com/Nanito111/CAPSTONE_002D/tree/website): Esta rama contiene todo lo relacionado con la aplicación de nuestro proyecto, tanto para su versión web, su versión de escritorio y su versión móvil.
   2. [Rama backend](https://github.com/Nanito111/CAPSTONE_002D/tree/backend): Esta rama contiene todo lo relacionado con la API de nuestro proyecto.
   3. [Rama database](https://github.com/Nanito111/CAPSTONE_002D/tree/database): Esta rama contiene todos los scripts de la base de la base de datos, creaciones de tablas, dumpeos de esquemas, etc.
   4. [Rama simulation](https://github.com/Nanito111/CAPSTONE_002D/tree/simulation): Esta rama no es vital para el proyecto pero sí necesaria para mostrar el funcionamiento. Contiene una simulación corriendo en el motor gráfico Godot que se encarga de mostrar la interacción entre un cliente con la aplicación.
2. [Aplicación](https://capstone-002-d.vercel.app/): La aplicación está escrita en el framework NextJS, corriendo bajo el runtime nodejs. Utiliza tecnologías PWA para permitir la portabilidad entre web, escritorio y móvil de forma fácil y rápida en el desarrollo, y usa middlewares para controlar las peticiones entre la página, el hosting y la API para hacer un manejo de sesiones.
3. [API](http://129.151.120.46:3000/docs): Backend de nuestro proyecto, corre bajo el framework FastAPI corriendo bajo el runtime de python. Contiene la lógica general del proyecto, integraciones con la base de datos y los clientes web, lógica interna, controles de seguridad, etc.
4. Base de datos: Este proyecto consta de dos bases de datos:
   1. Oracle Cloud DB: Base de datos relacional. Para la persistencia general que contiene los datos de los usuarios, información histórica, y manejo de sesiones
   2. Firebase Realtime DB: Base de datos no relacional. Contiene los datos volátiles, información que será contenida durante el día y en tiempo real

### ¿Por qué optamos por este stack de tecnologías?

Nosotros como desarrolladores optamos por estas tecnologías buscando un equilibrio entre facilidad de desarrollo y eficacia+rapidez del servicio.

Esto justificando el uso de ciertas tecnologías en la parte cruda del proyecto, como Fast API y python para la rapidez y facilidad de desarrollo, NextJs para la facilidad del desarrollo, typescript para la eficacia+rapidez. Esto sin mencionar que nosotros como desarrolladores ya teníamos cierta experiencia previa en el uso de este stack.

Las bases de datos a utilizar las elegimos principalmente por:

* Oracle Cloud DB: Escogimos esta base de datos debido a que ya tenemos experiencia previa manejando estas bases de datos, además de que tenemos una capa gratuita que la estamos aprovechando para este proyecto y el lugar físico donde se encuentra la base de datos es en chile, por lo que existe poca latencia
* Firebase Realtime DB: Escogimos esta base de datos como una forma de poder equilibrar la carga de la base de datos Oracle, esto debido a que la base de datos anterior debido a su capa gratuita, tiene algunas limitaciones de rendimiento. Por lo que utilizaremos esta base de datos para todas las operaciones que sean en tiempo real, ya que es una base de datos no relacional, proporcionando un gran rapidez para datos simples de poca escala.

La mayoría de las tecnologías que estamos usando para este proyecto están respaldadas por empresas más grandes que ofrecen una integración directa de hosteo para el uso personal gratuito. Como Vercel (NextJS), Oracle y Google (Firebase)

## Diagramación

Estos diagramas muestran las 5 vistas del modelo 4+1 buscando describir la arquitectura de nuestro sistema de software.

### Vista Despliegue

Ilustra el sistema desde el punto de vista del programador. Enfoca la administración de los artefactos del proyecto. Utiliza componentes UML para describir los componentes del sistema.

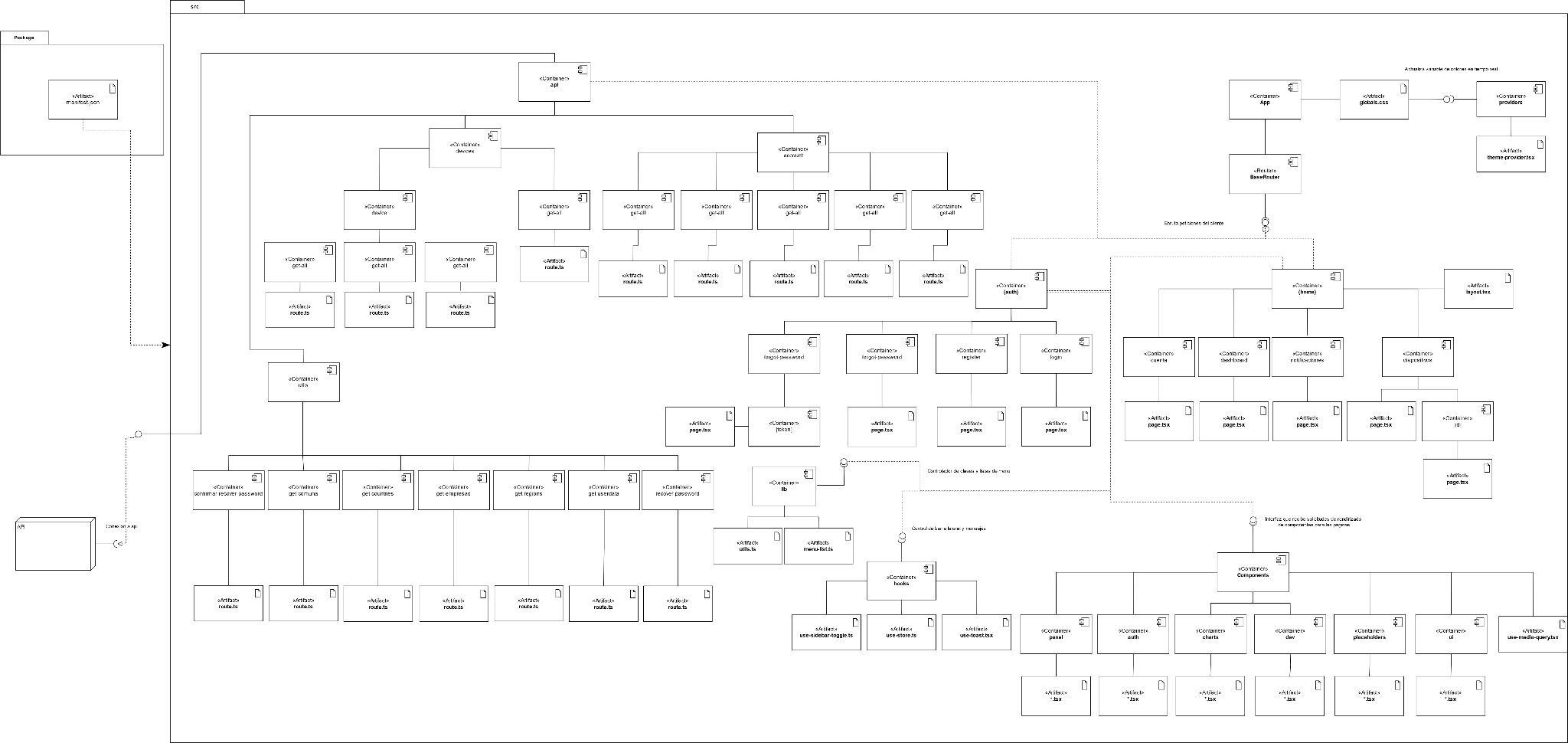
****

Diagrama de componentes

### 

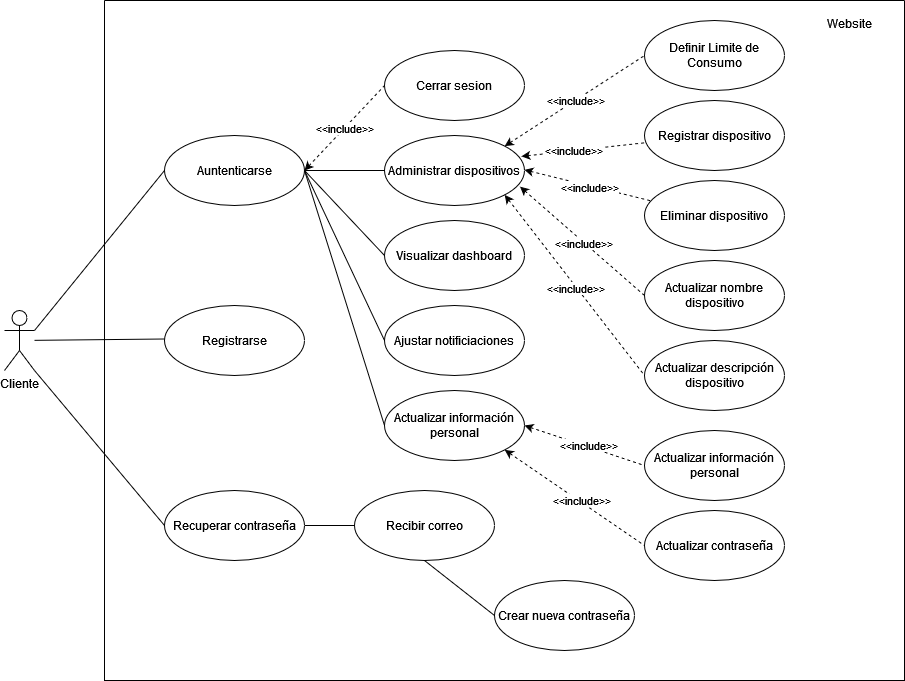
### 

Diagrama de paquetes

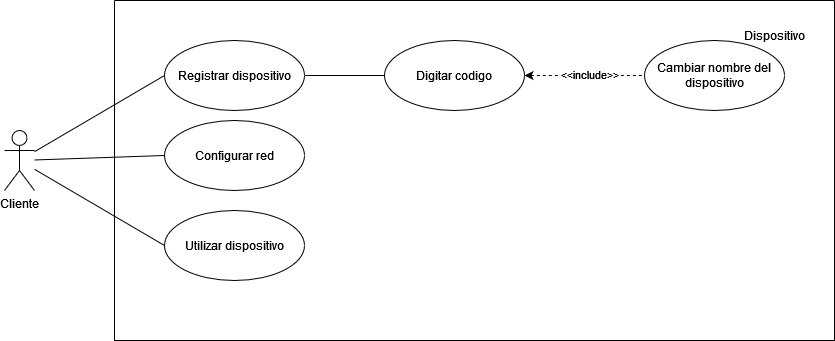
### 

### Vista Escenarios

Ilustra el conjunto de casos de usos y describe la secuencia de interacciones entre los objetos y procesos desde el punto de vista de un actor



Casos de uso plataforma web



Casos de uso dispositivo

### Vista Física

Describe el sistema desde un punto de vista de un ingeniero, muestra la topología de los componentes en un nivel más físico, así como las conexiones que tengan.

### 

Diagrama de despliegue

### 

### Vista Lógica

Enfocada en describir la estructura y las funcionalidades del sistema

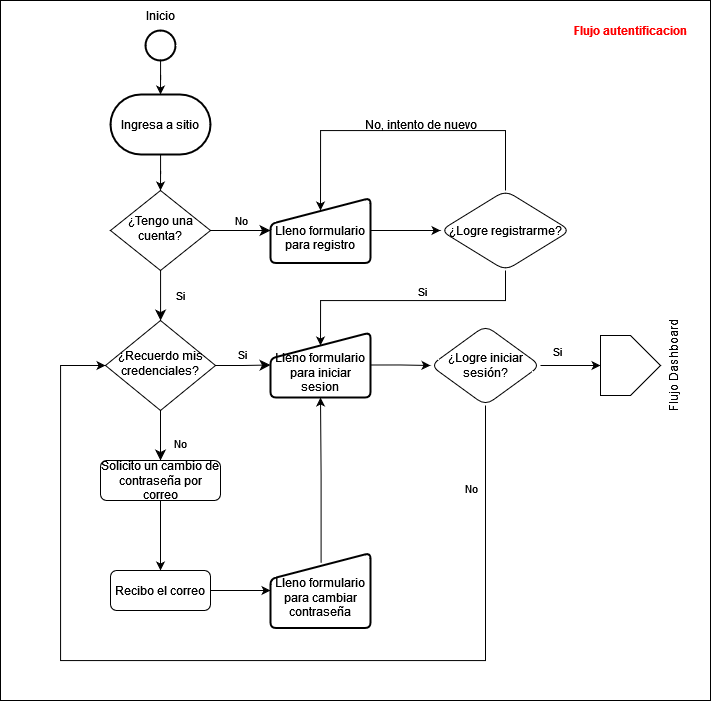


Diagrama de flujo - autenticación

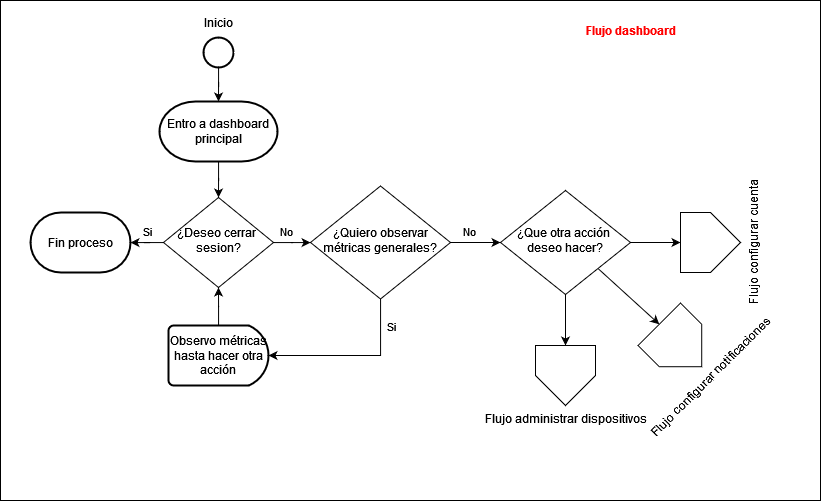


Diagrama de flujo - dashboard

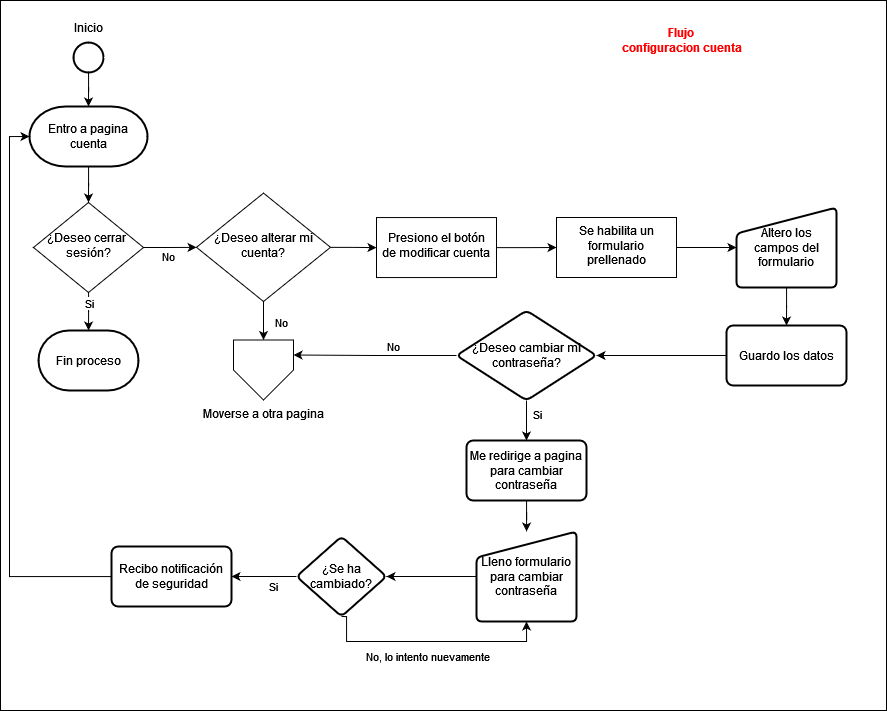


Diagrama de flujo - cuenta

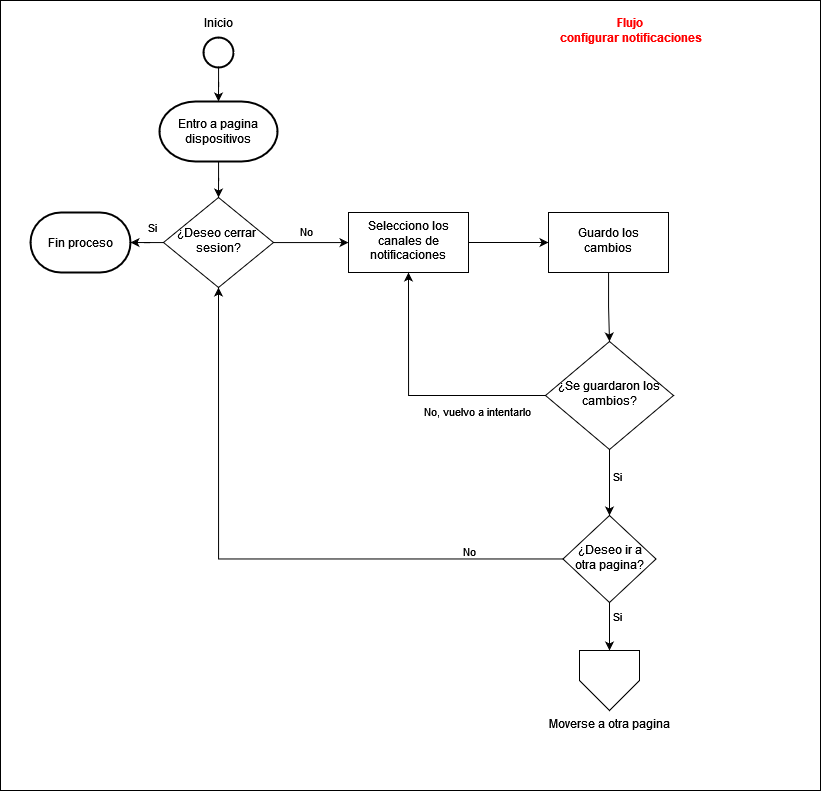


Diagrama de flujo - configurar notificaciones

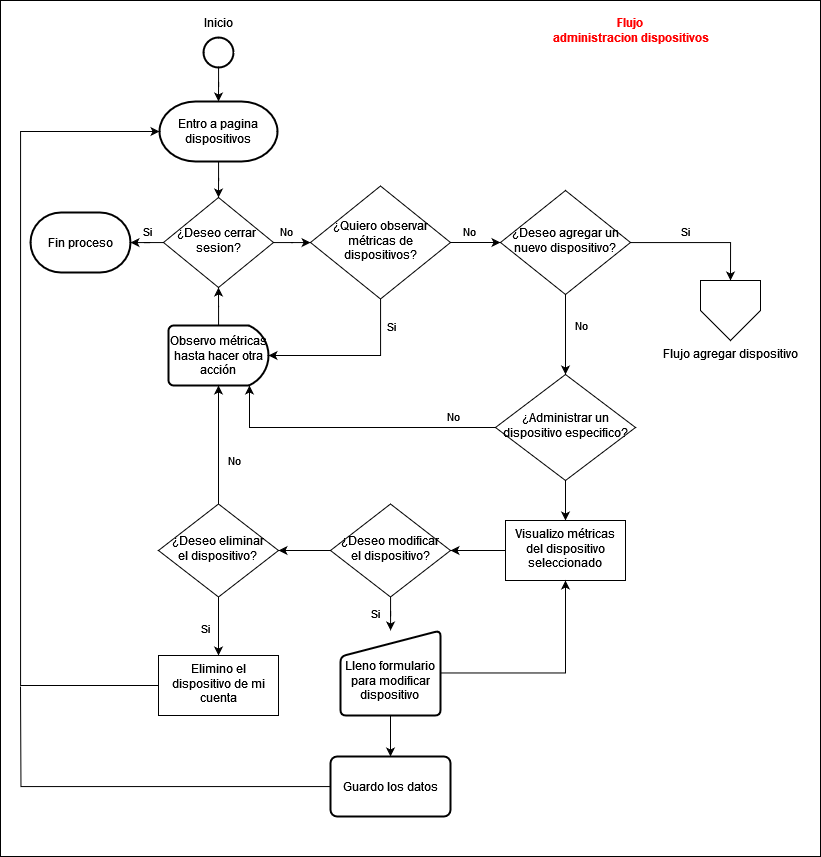


Diagrama de flujo - Administrar dispositivos

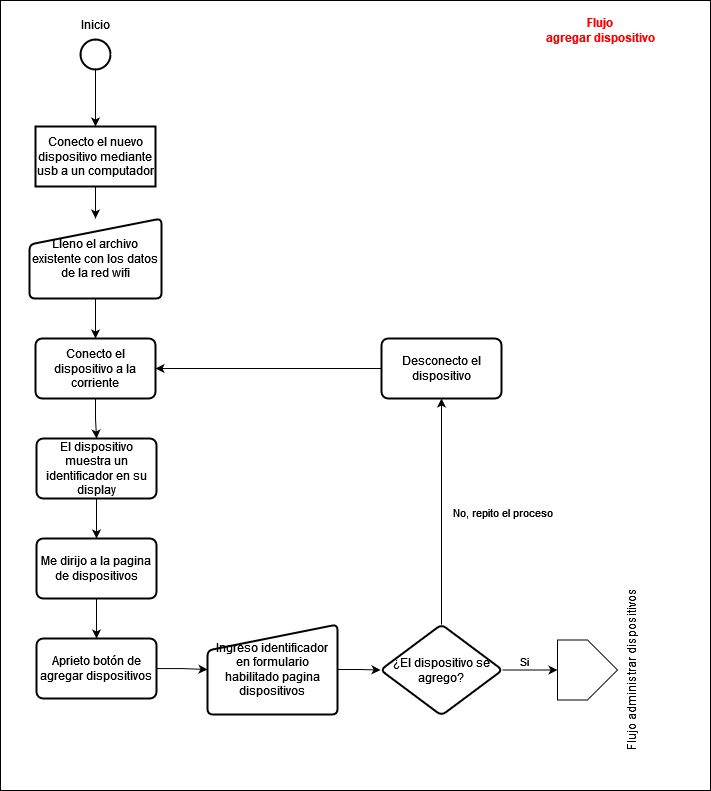


Diagrama de flujo - Agregar dispositivo

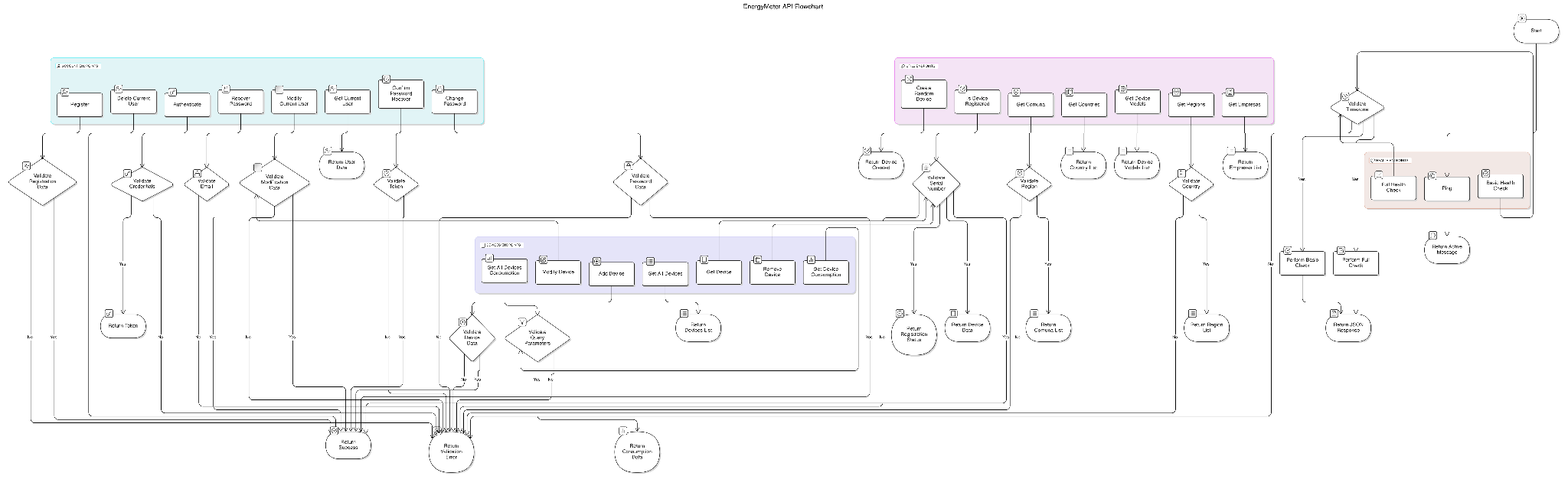


Diagrama de flujo - Flujo de api

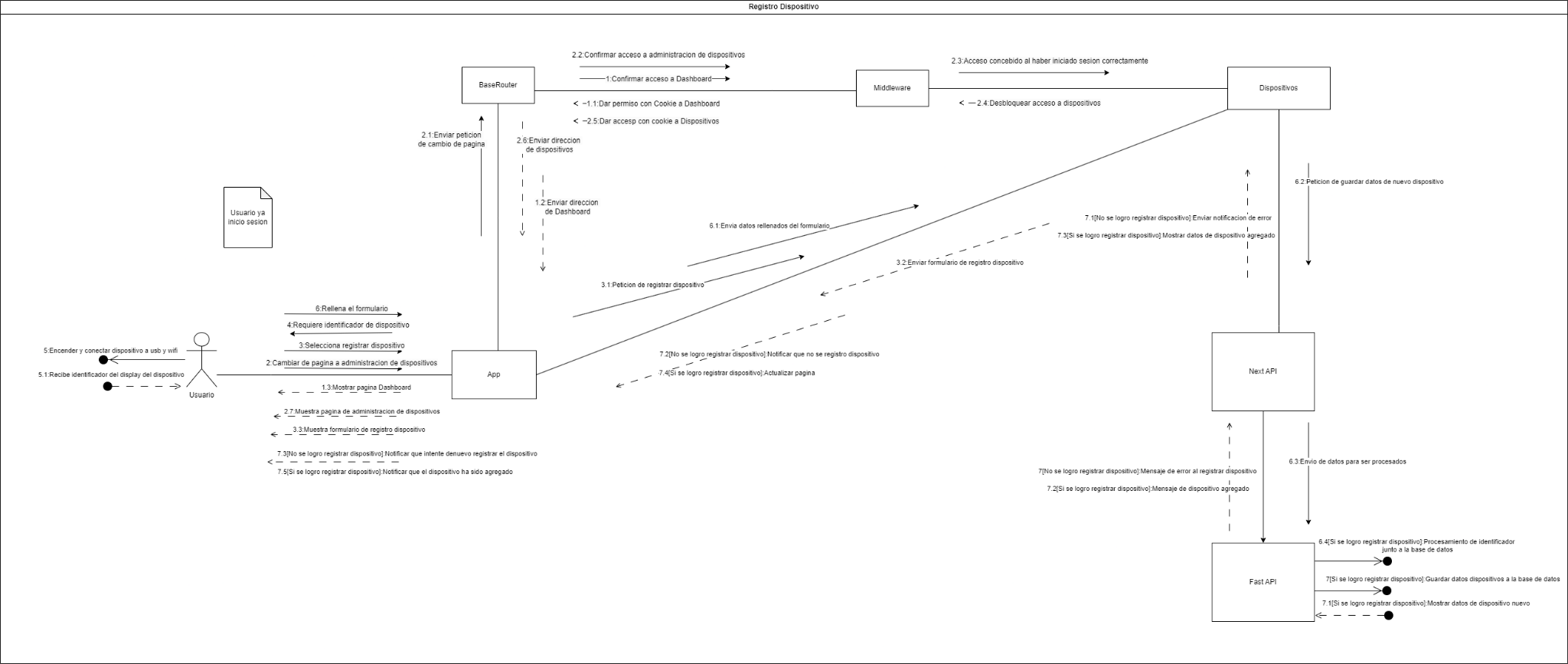


Diagrama de comunicación dispositivo

### 

Diagrama de comunicación inicio de sesión

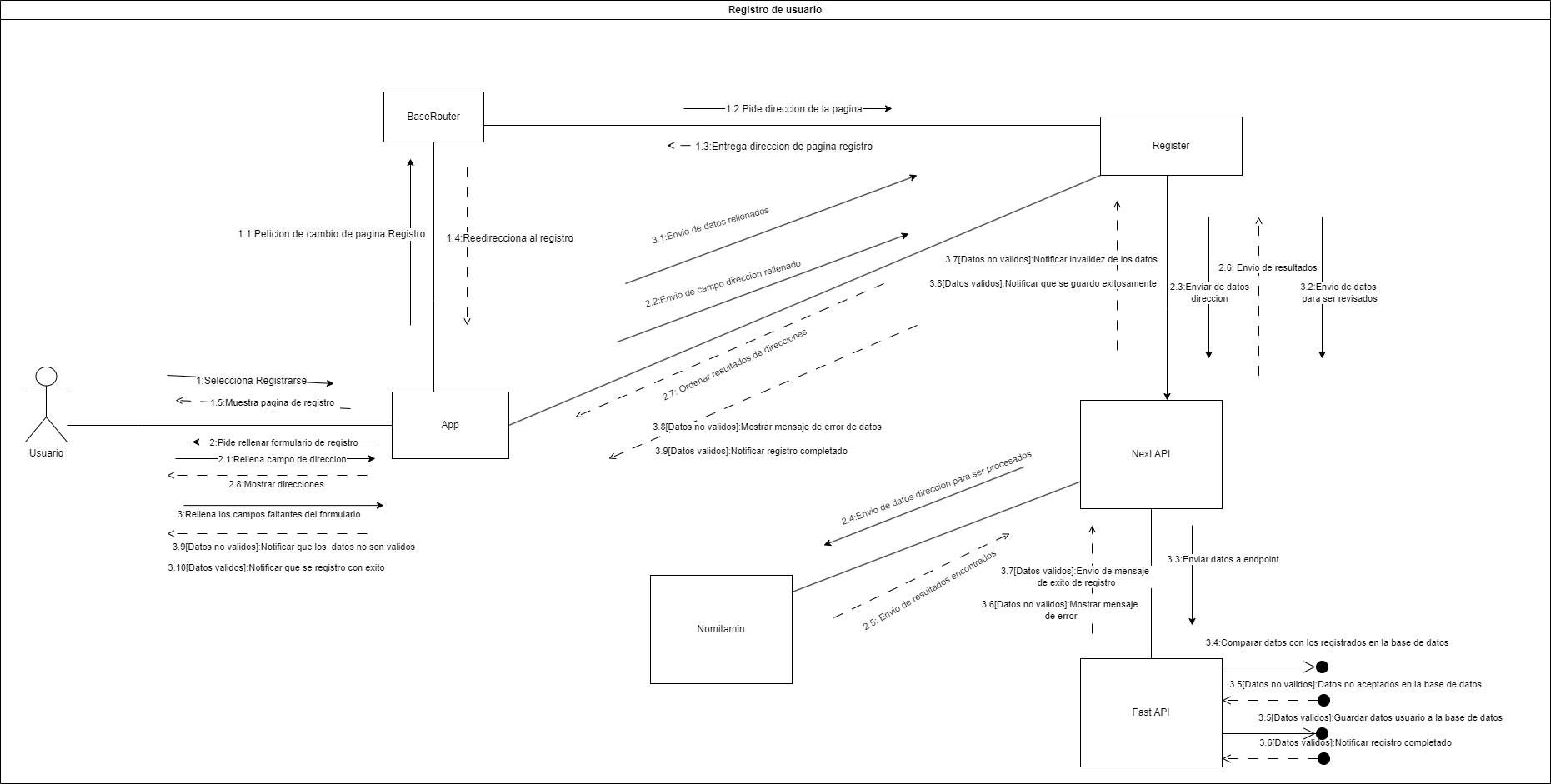


Diagrama de comunicación

### 

Diagrama comunicación dashboard

### 

Diagrama de comunicación Administración de dispositivos

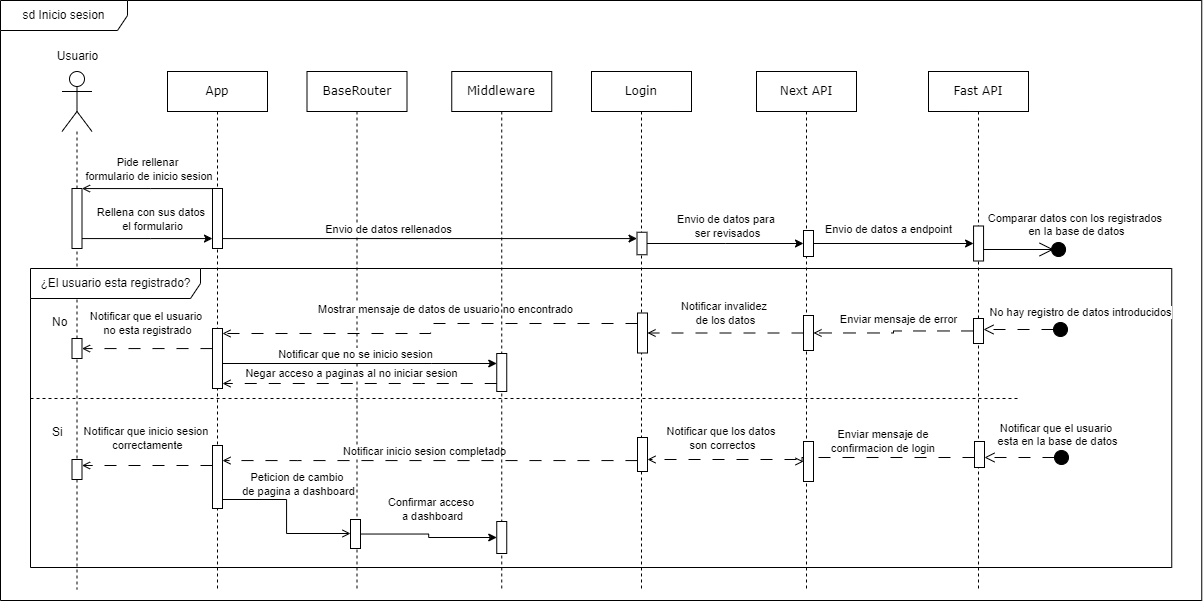


Diagrama de secuencia inicio de sesión

### 

Diagrama de secuencia de Registro de usuario

### 

Diagrama de secuencia de Dashboard

### 

Diagrama de secuencia de Administración de dispositivos

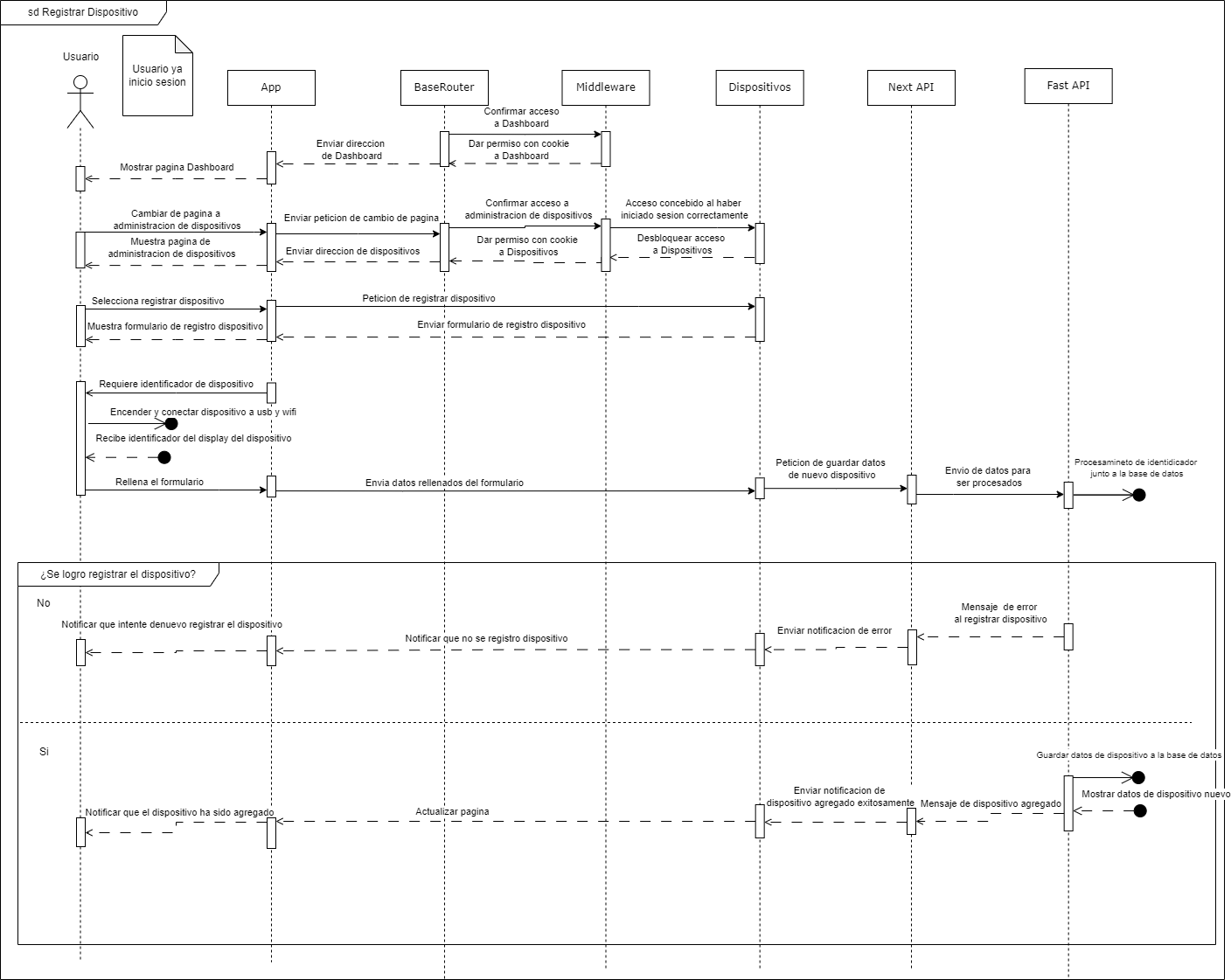


Diagrama de secuencia de Registro de dispositivo

### Vista Procesos

Describe los procesos del sistema y cómo estos se comunican, enfocándose en el comportamiento de estos.

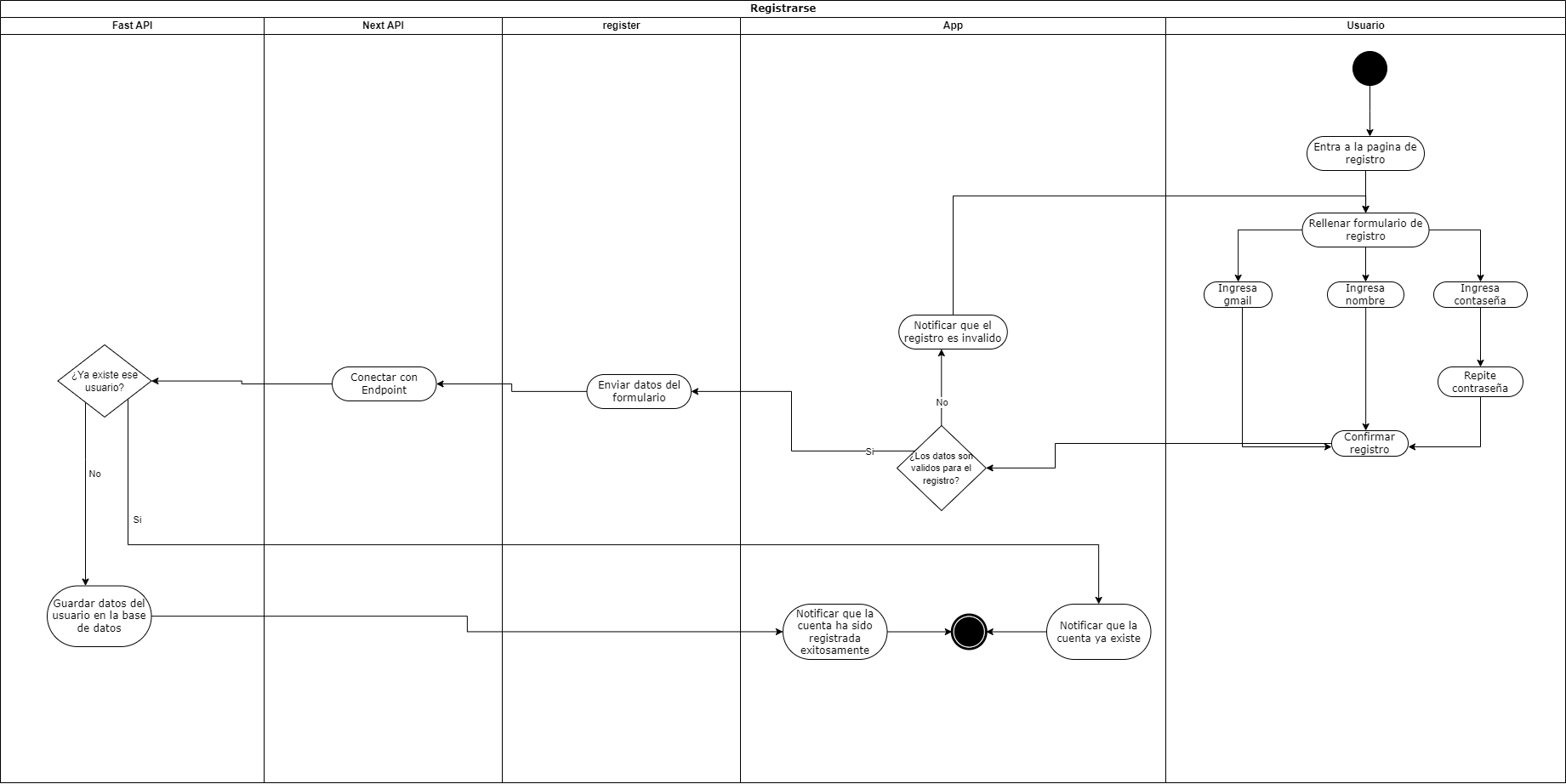


Diagrama de actividad de Registro de usuario

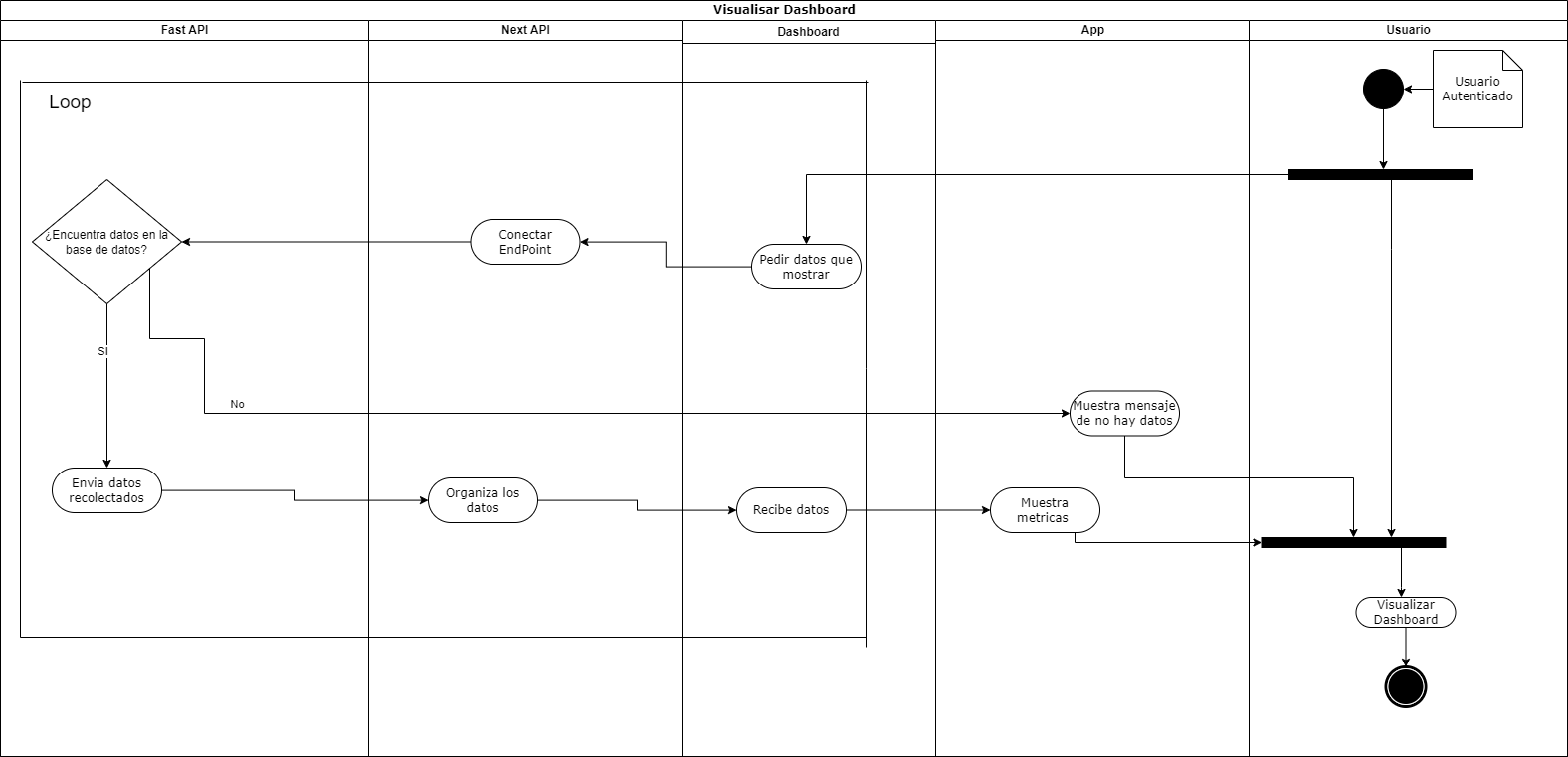
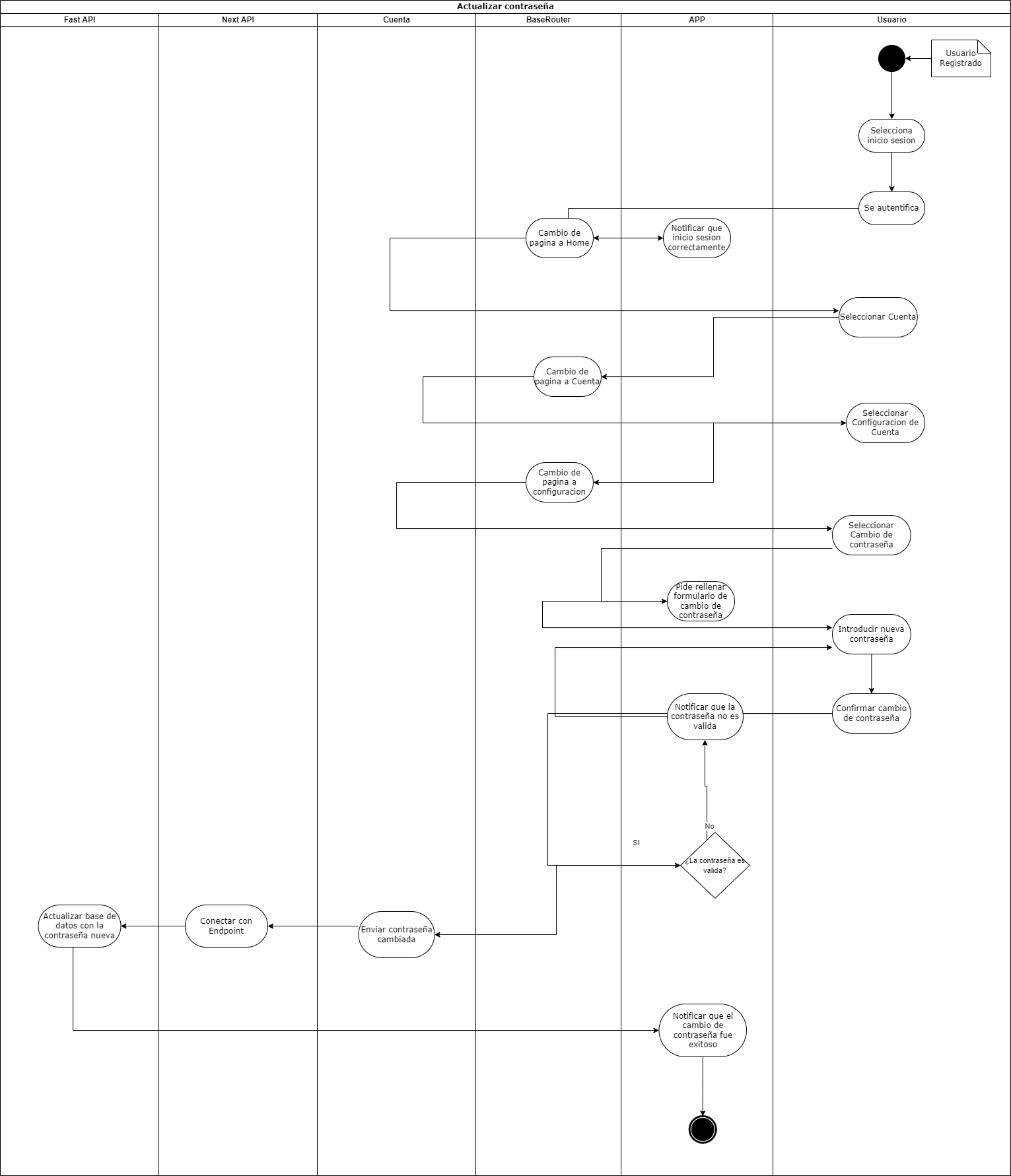


Diagrama de actividad de Dashboard

Diagrama de actividad de Actualización de contraseña

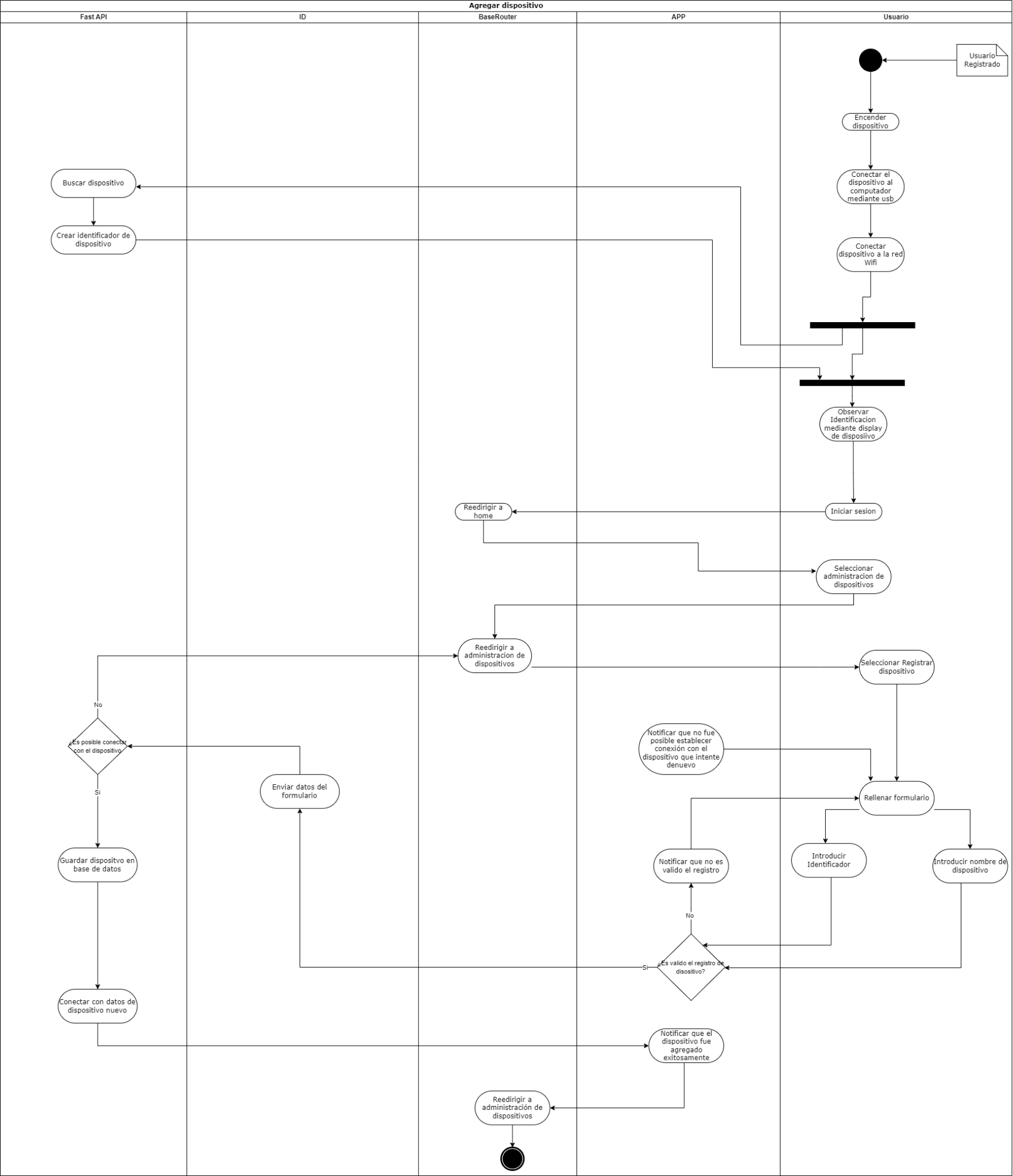
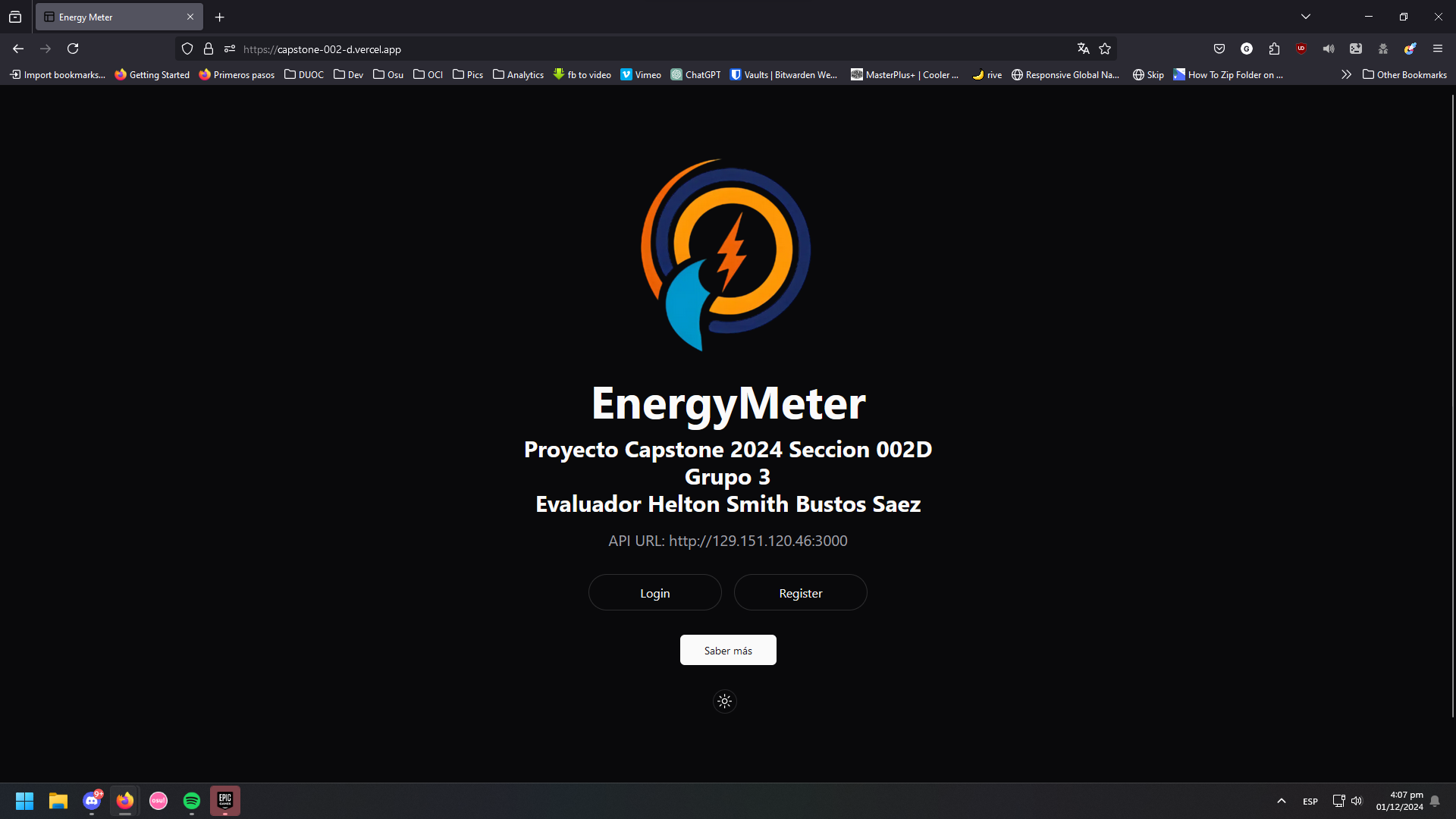
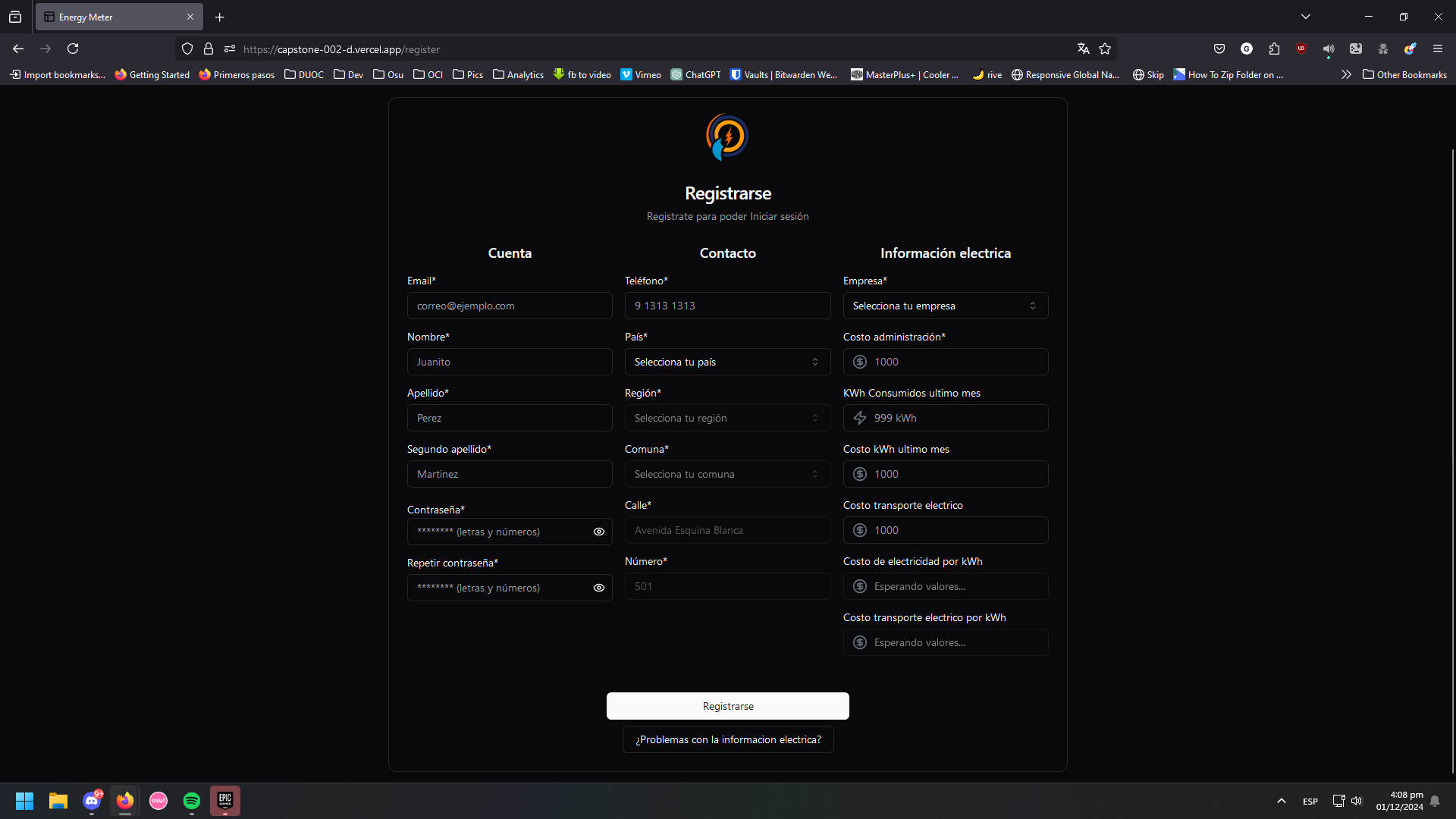


Diagrama de actividad de Añadir dispositivo

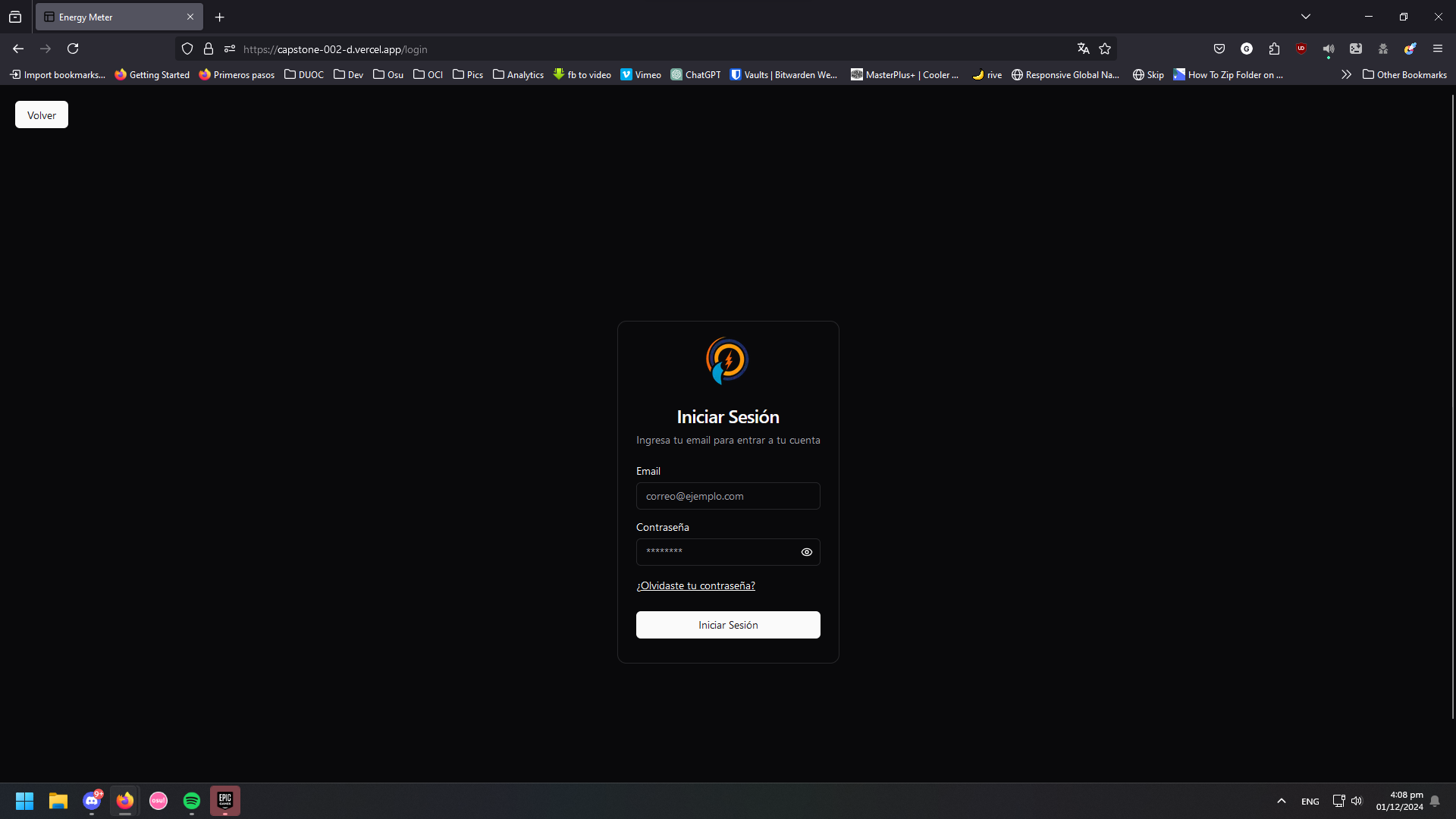
### Capturas Aplicación

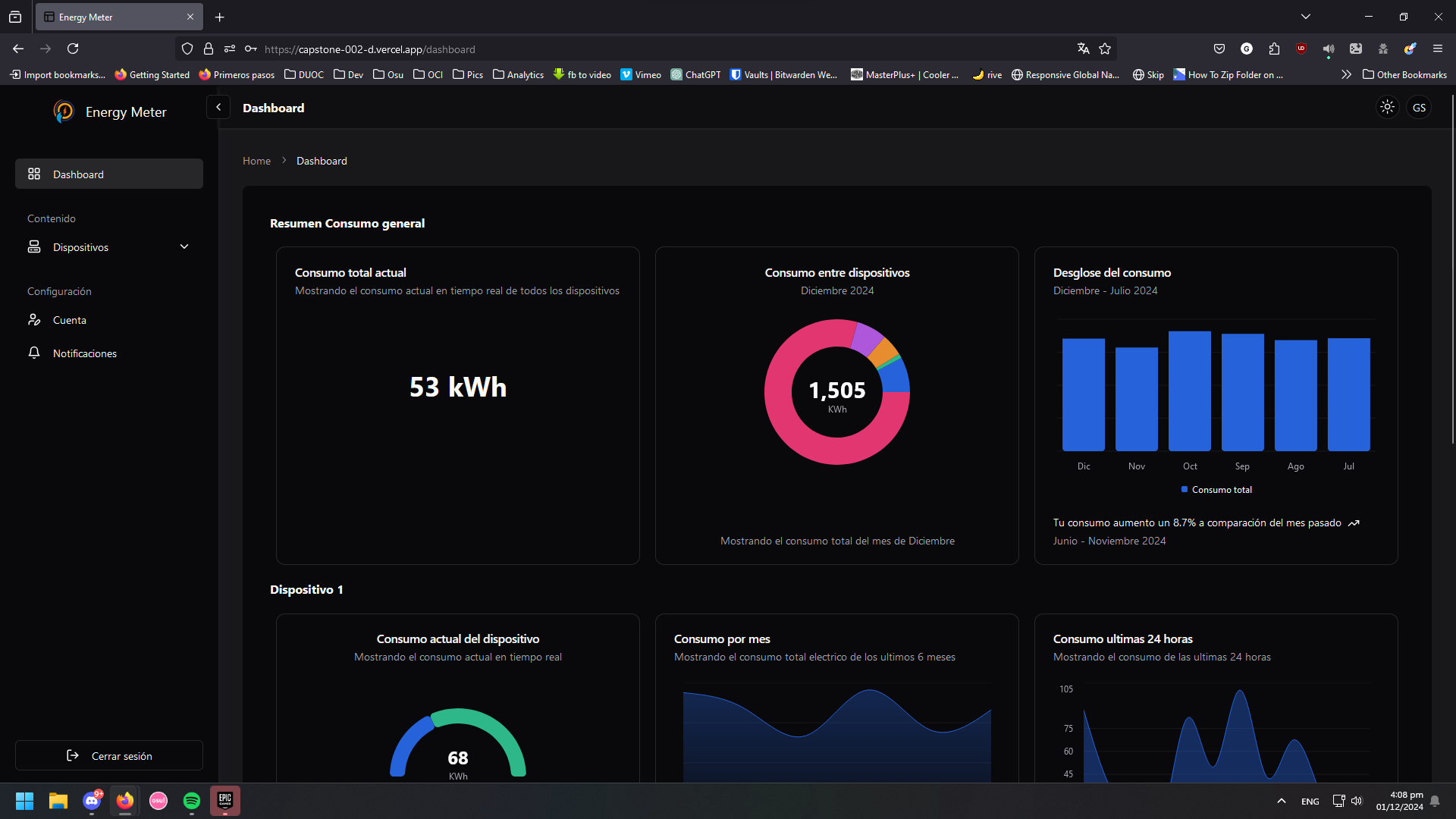


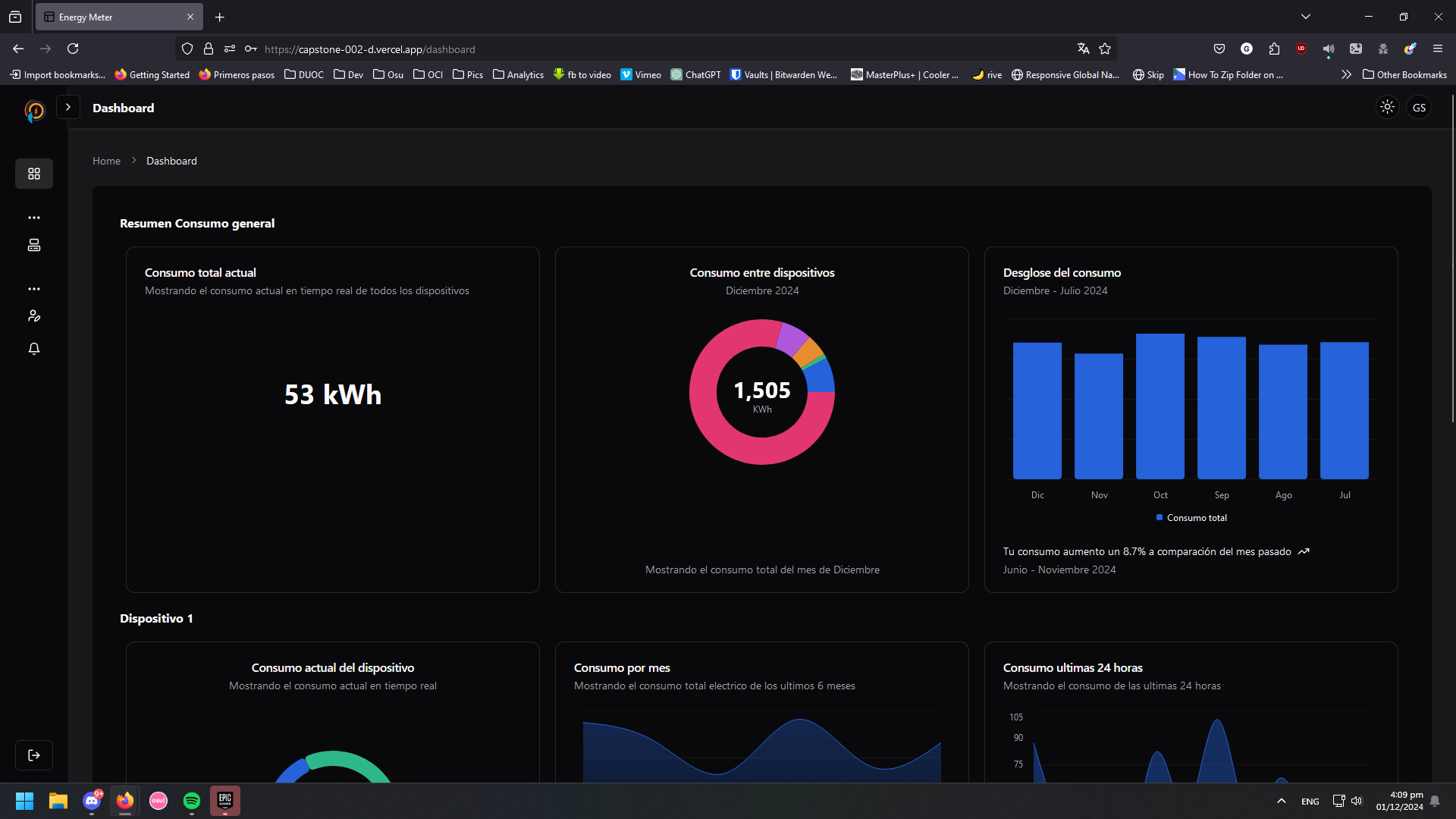
Home page

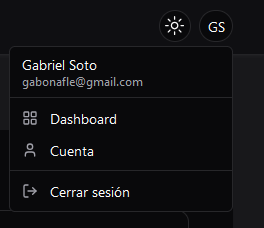


/register

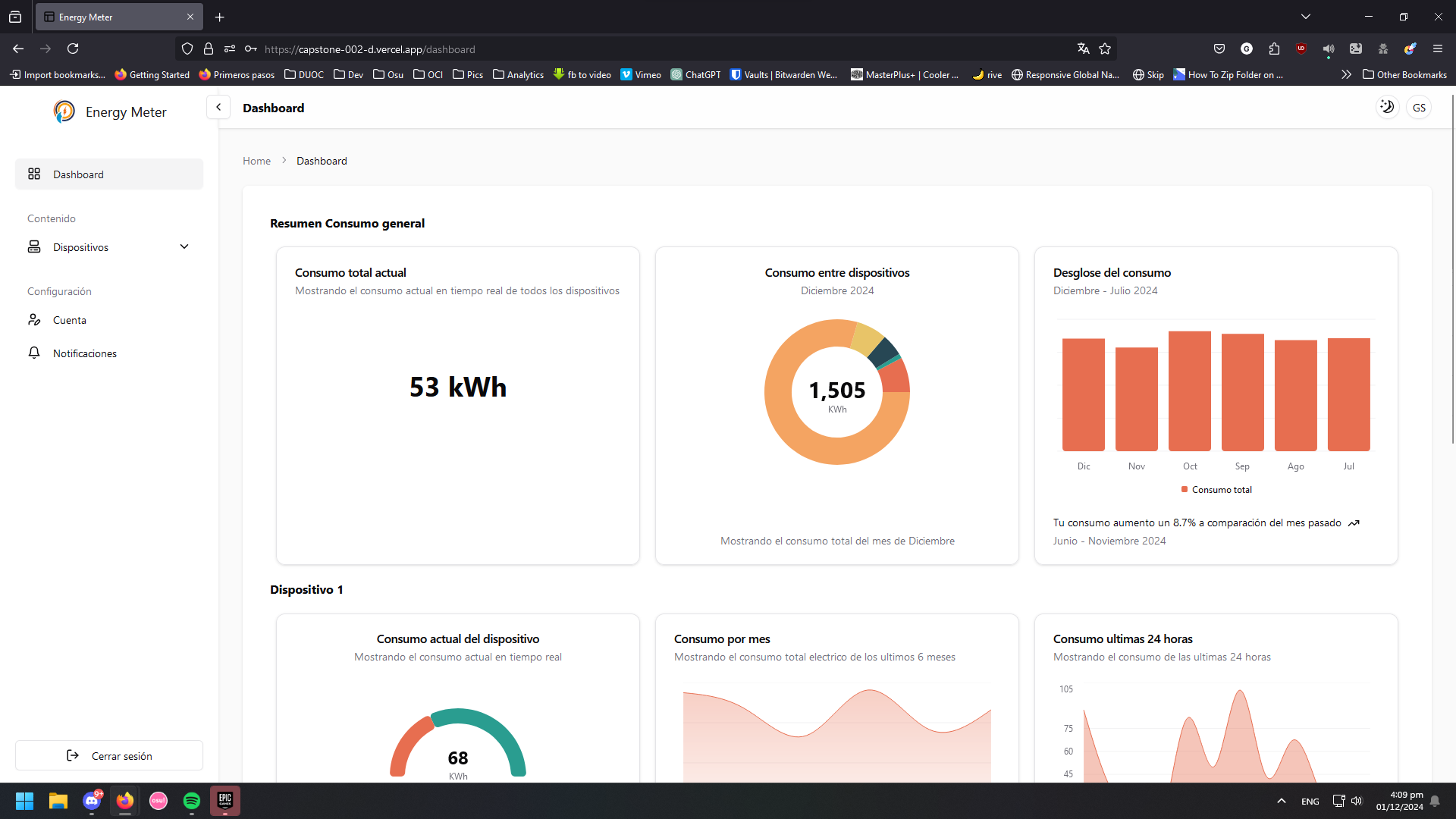
/login

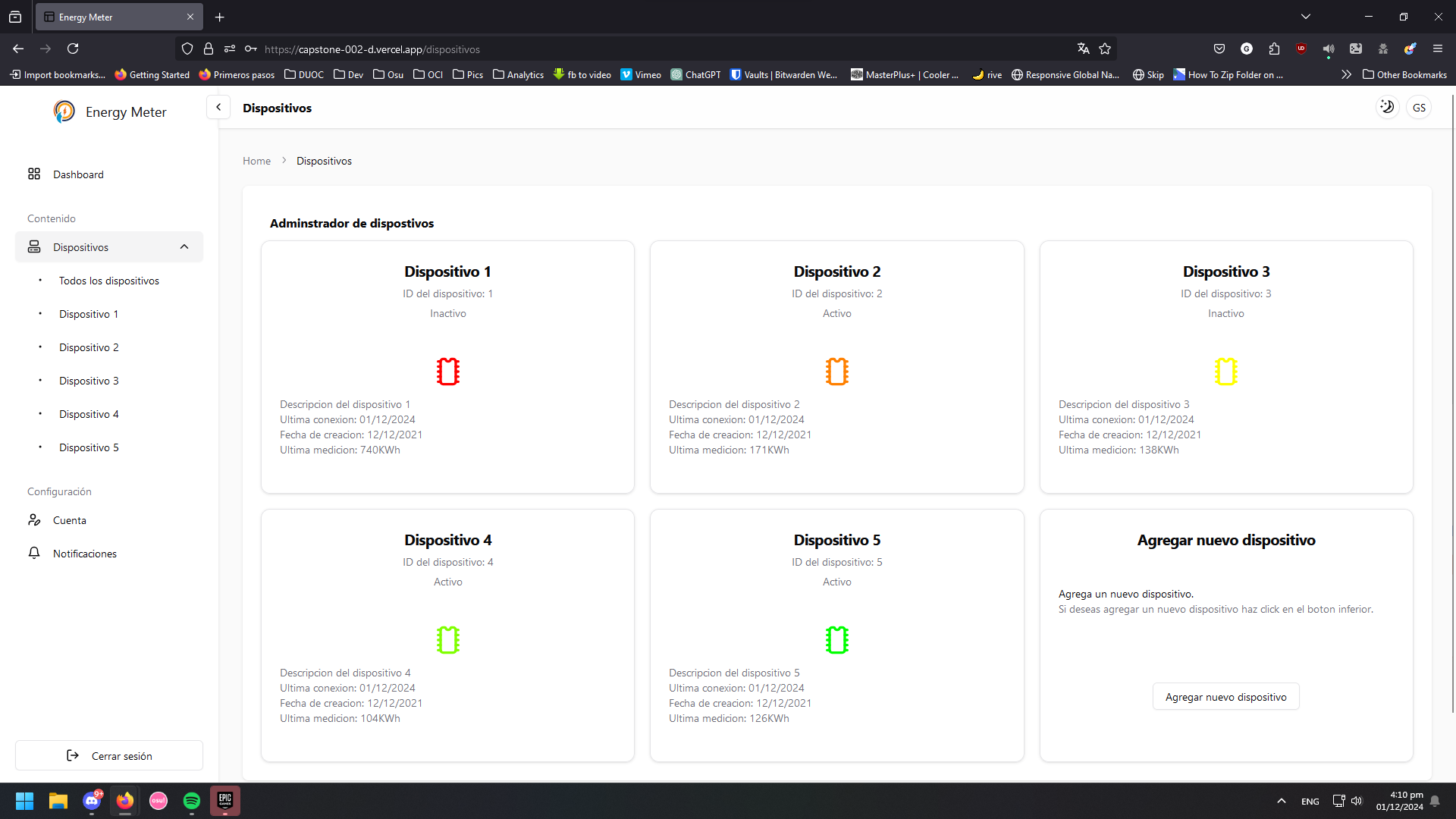
/dashboard

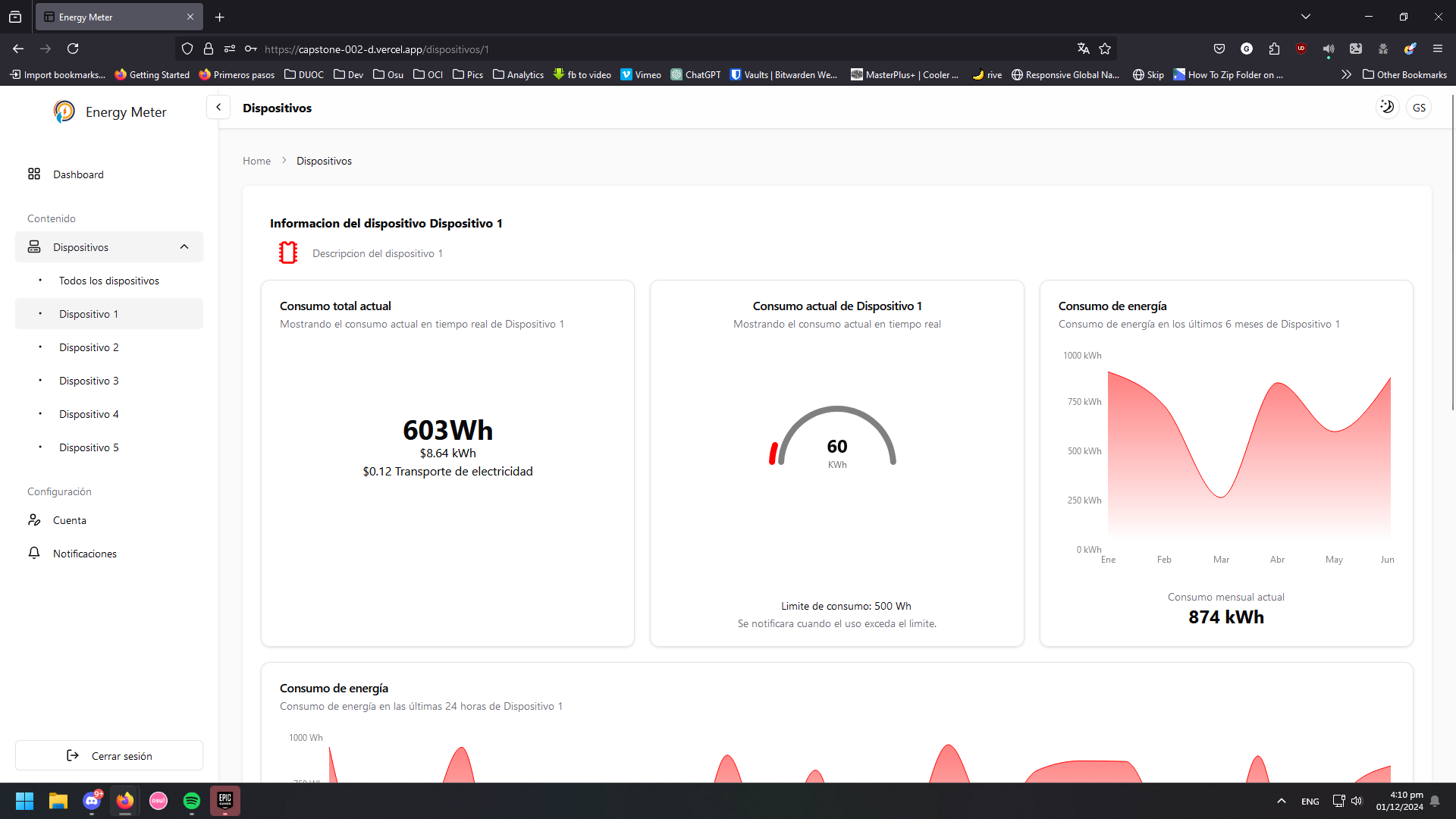
/dashboard sidebar colapsada

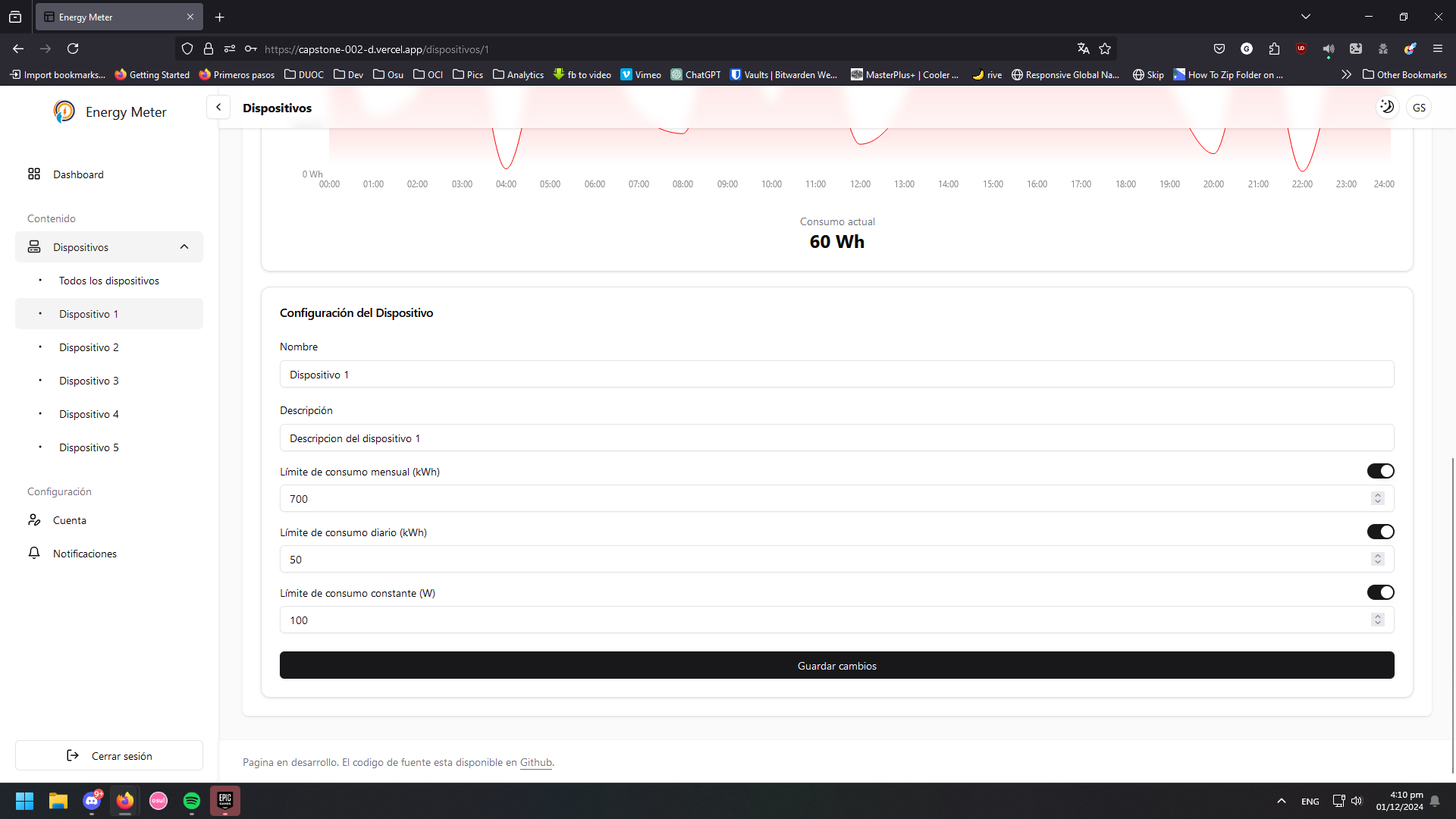


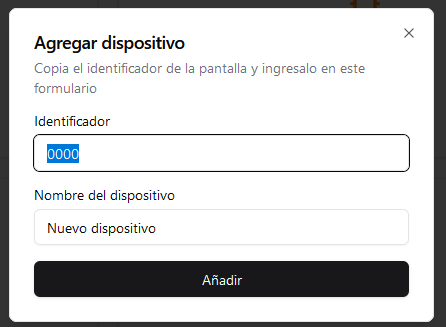
Menú usuario

Tema claro

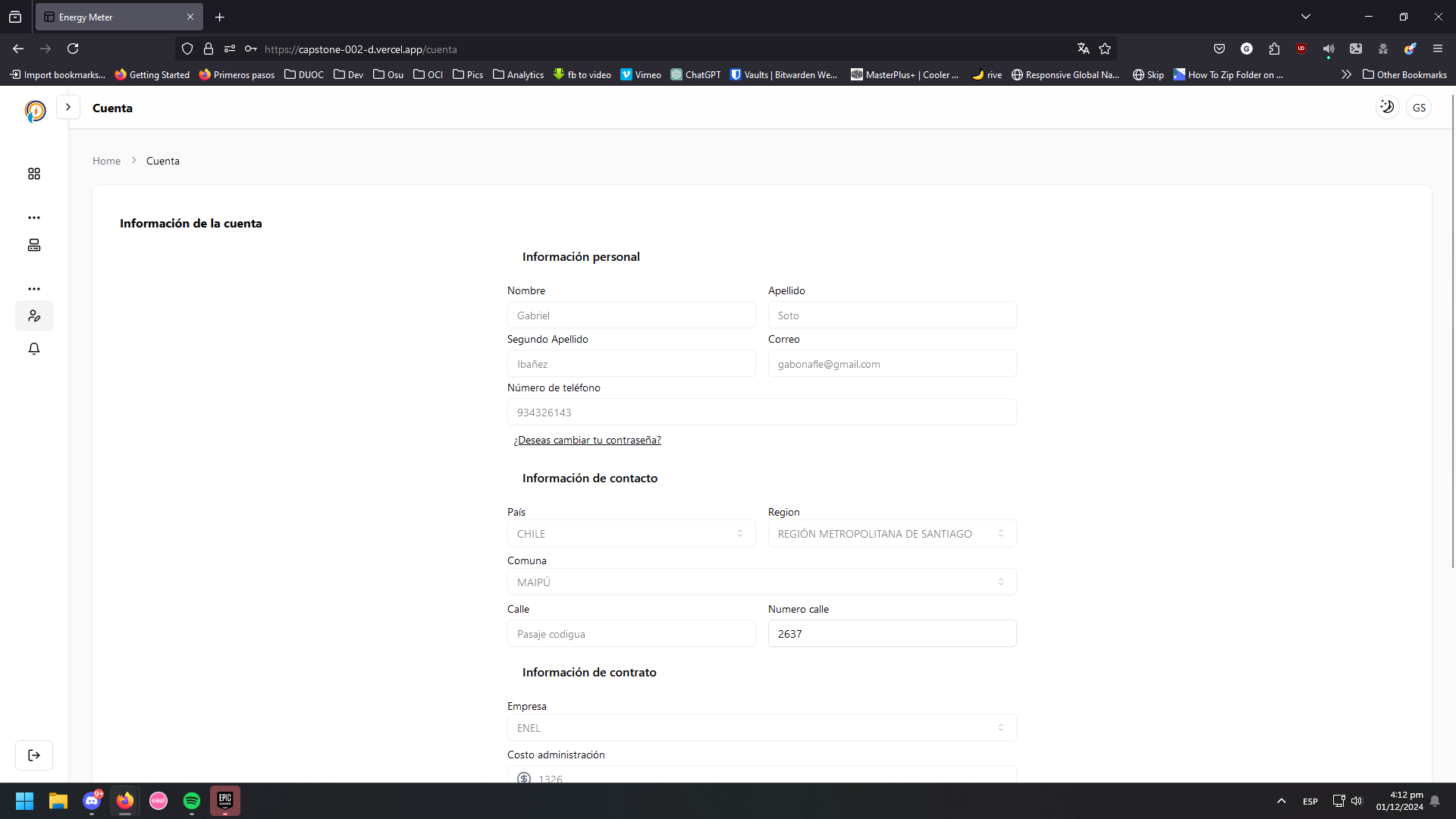
/dispositivos

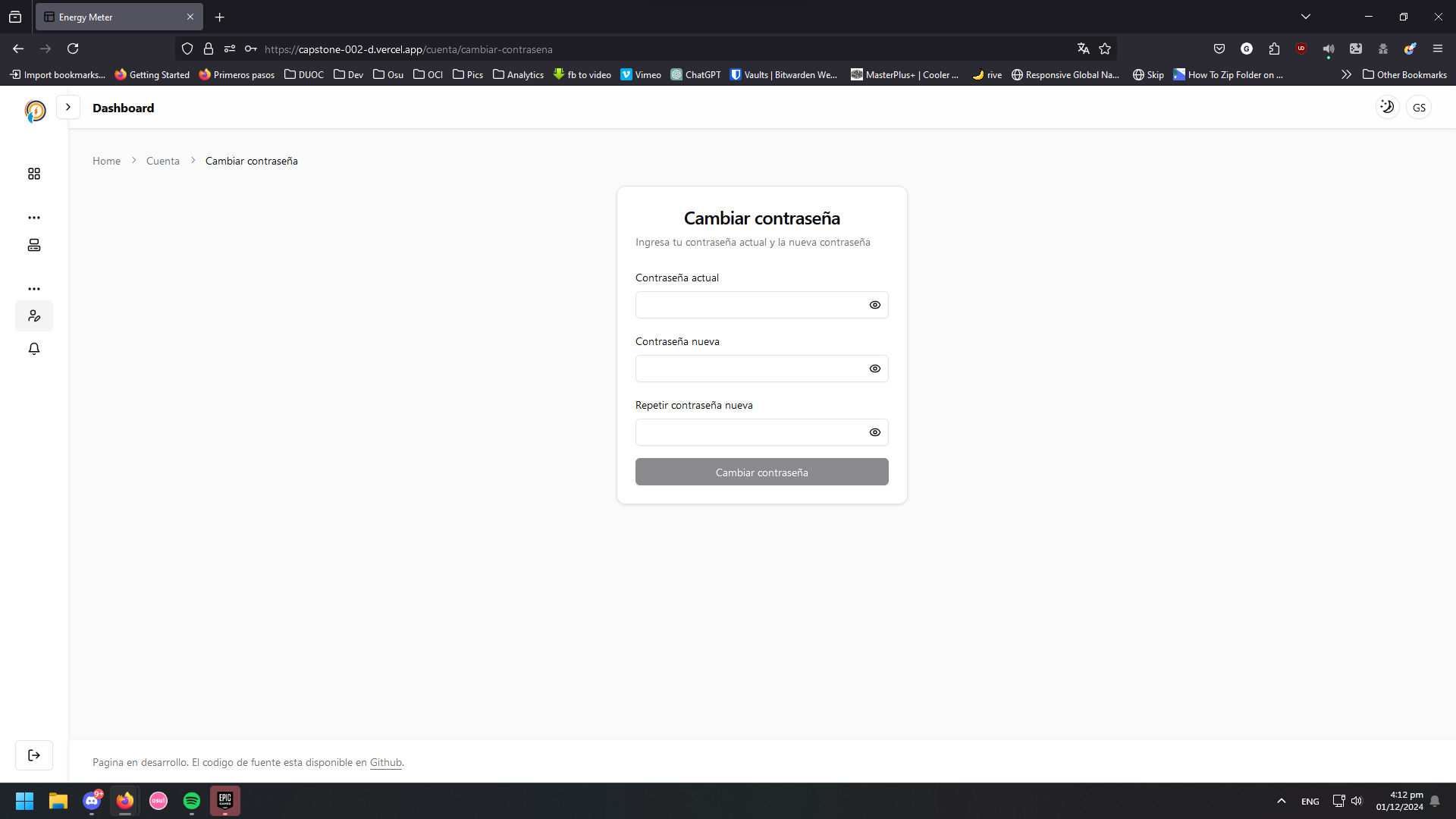
/dispositivo/[id]

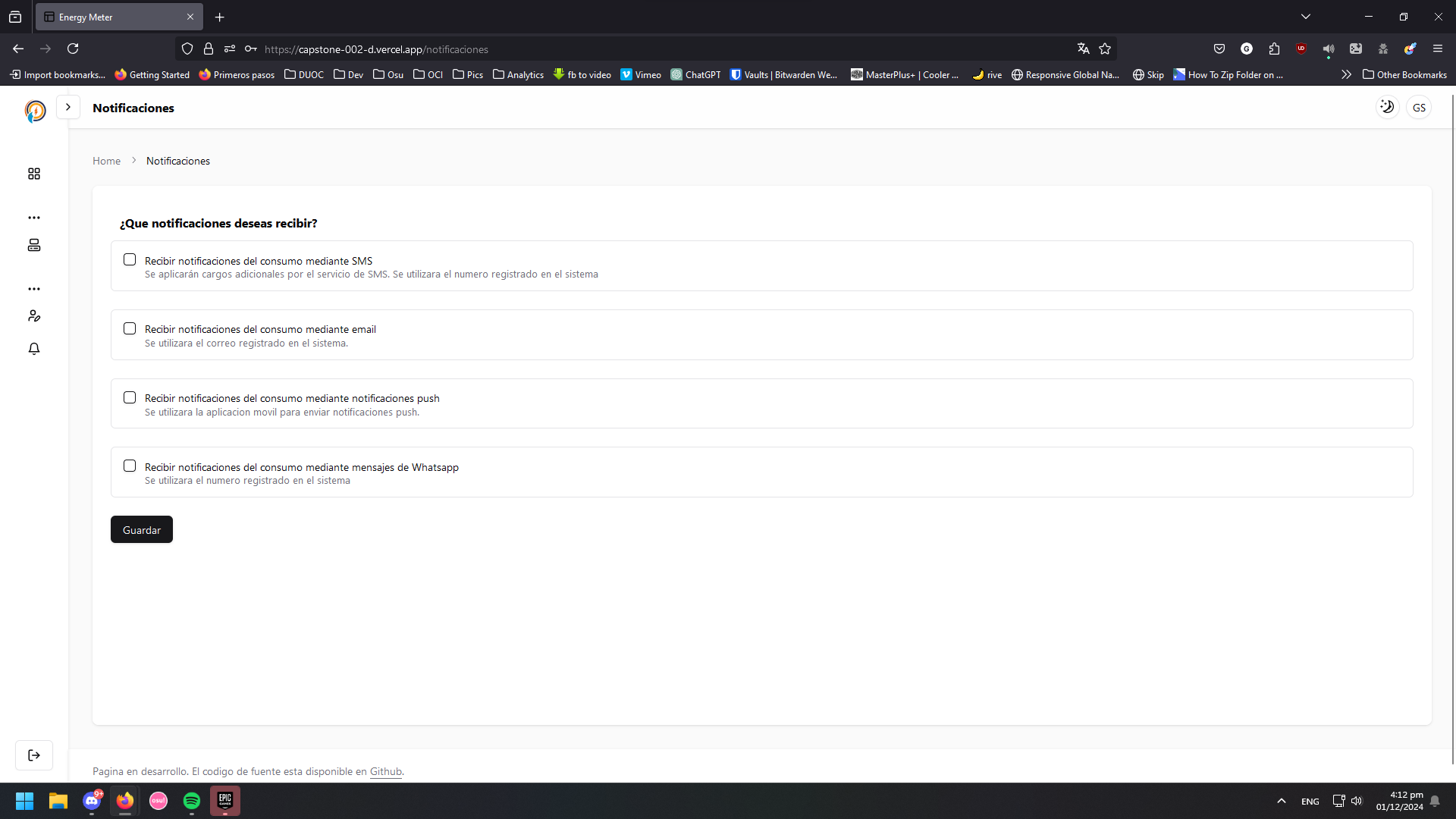
/dispositivo/[id] Edicion de dispositivos

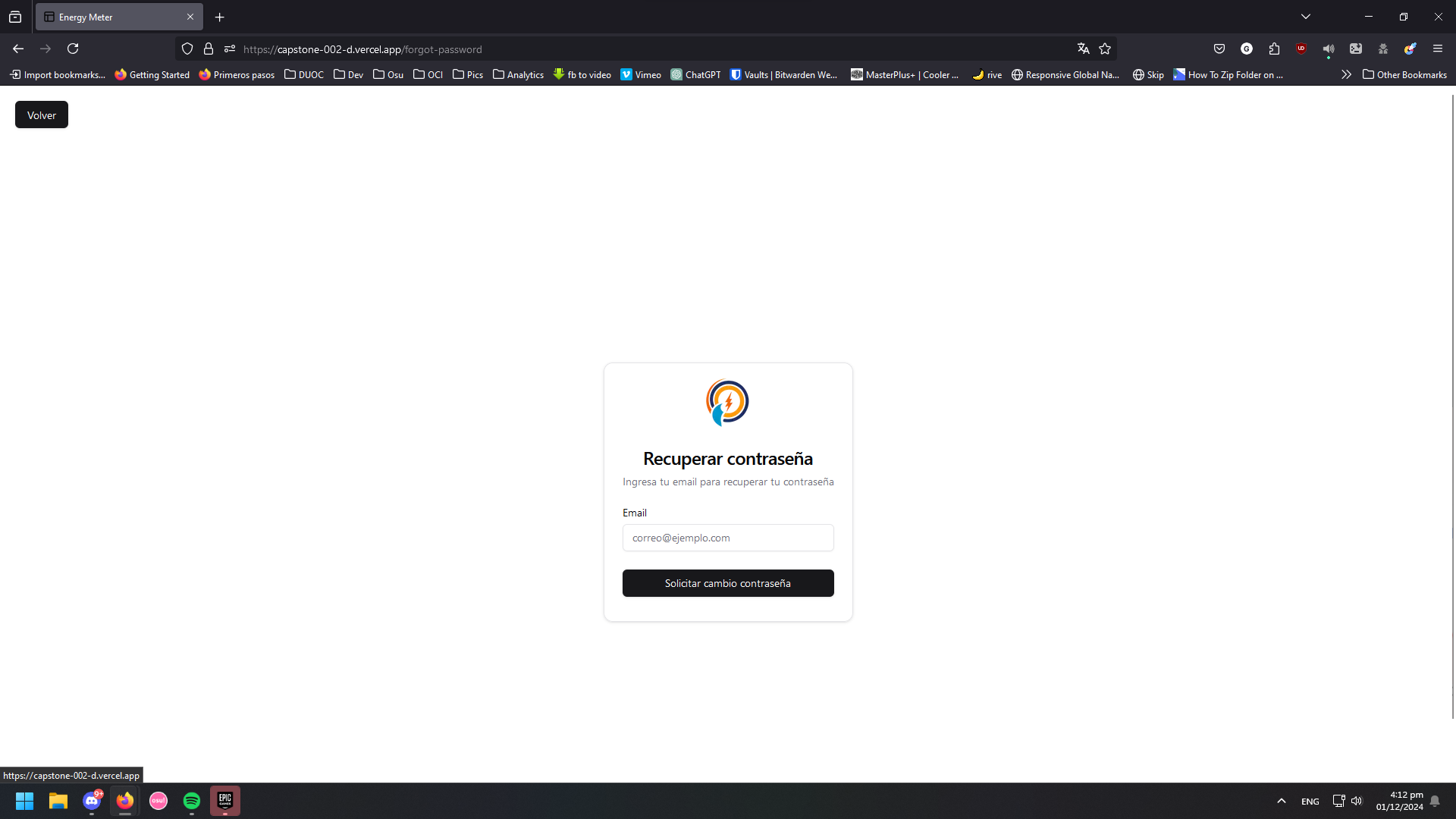


/dispositivos Añadir dispositivo

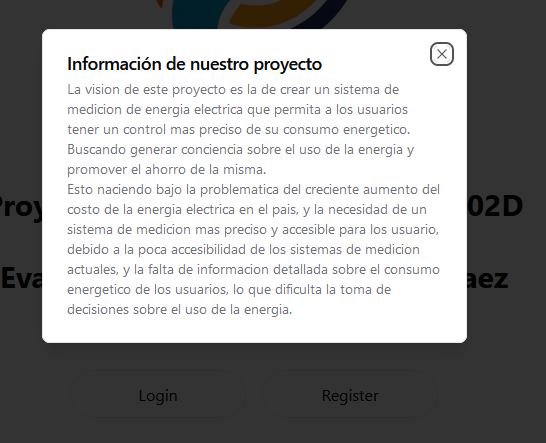
/cuenta

/cambiar-contrasena

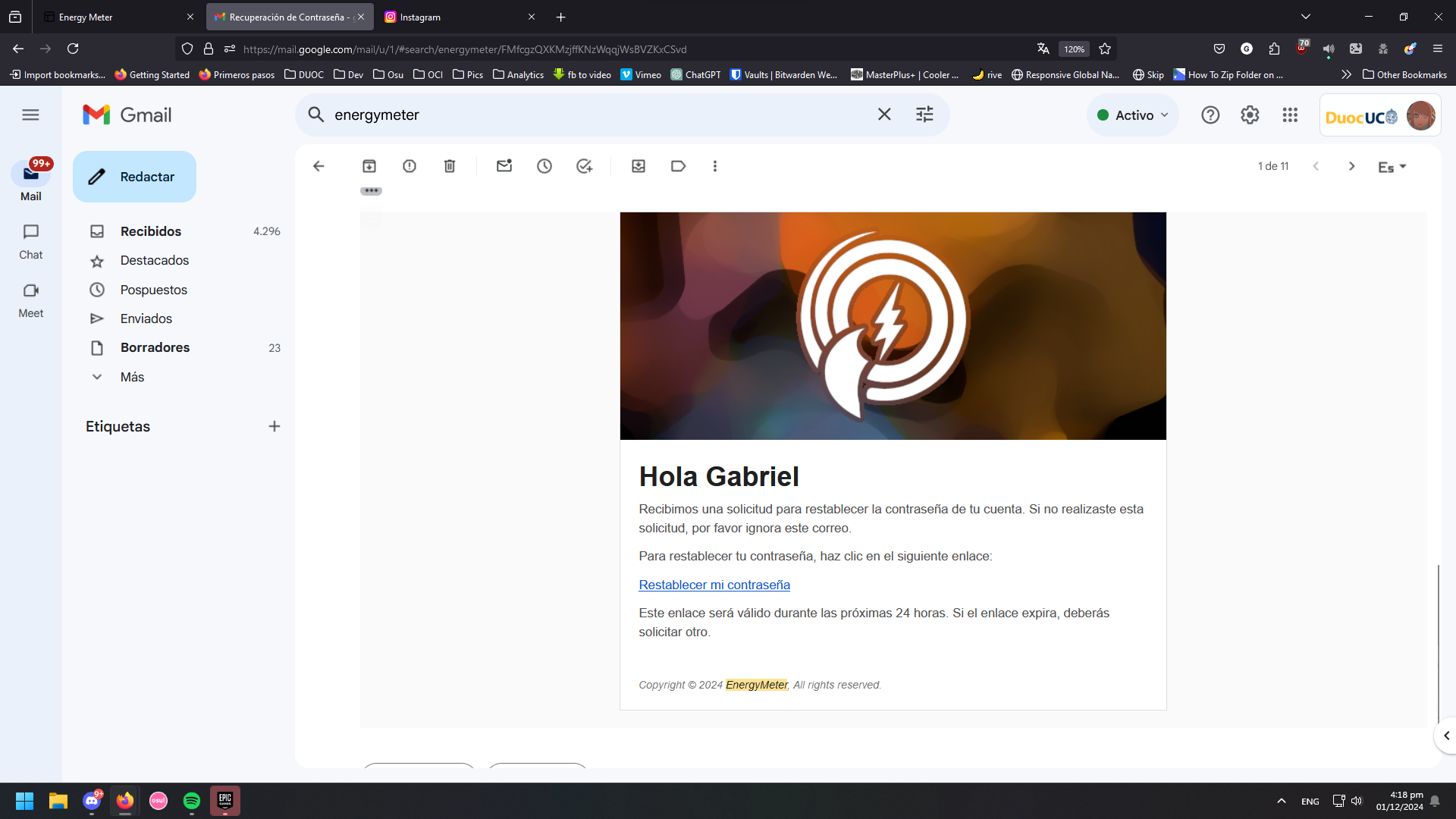
/notificaciones



/forgot-password



home page - Información del proyecto



Recuperar contraseña



Dashboard app movil

## Ambiente y flujo de programación

Para el desarrollo de este sistema, se están utilizando diversos ambientes de desarrollo, estando regidos por las ramas existentes en github:

* **Rama website (Frontend)**
  + Ambiente local: Empleada en la máquina del desarrollador, puede estar ejecutando cualquier sistema operativo, pero obligatoriamente debe tener instalados los runtimes de nodejs y los gestores de npm para poder instalar las librerías y módulos necesarios. Aquí se permite tener más libertad de desarrollo, tanto para debugeo, pruebas, etc, ya que no refiere a ser una versión estable del sistema.
  + Ambiente preview: Empleada en un servidor vercel. Las ramas preview antes de hacer un deploy, buscan un archivo de configuración para saber cuales son las ramas que están involucradas con lo referente al frontend (para no hacer un deploy de una feature del backend y termine cayendo en un error). Esta rama busca demostrar una vista previa antes de hacer oficial una feature desarrollada, sirve para mostrar entre desarrolladores lo de una forma más real lo que sería un producto final, facilitando el proceso de code reviewing y la aprobación de pull request en github.
  + Ambiente producción: Empleada en un servidor vercel. Rama estable y producción de la página. Todas las features terminadas y estables terminan su ciclo de vida de desarrollo cumplido acá. El flujo es muy parecido a la rama preview, una vez que la Pull request haya sido aprobada, la feature será mergeada hacia la rama website, haciendo que los cambios, features, hotfix o eliminaciones de features se vean reflejadas en la página principal de nuestro proyecto.
* **Rama backend (API)**
  + Ambiente local: Empleada en la máquina del desarrollador, idealmente debe estar ejecutándose en un sistema operativo linux o en un ambiente WSL, también debe tener instalado el runtime python3 en su versión 3.10 para la instalación de los paquetes mediante pip y la ejecución del subproyecto. Al igual que la rama anterior, en este ambiente se permite una libertad de desarrollo pero menor, esto debido a que no hay una rama preview para poder previsualizar el cómo serían los añadidos del proyecto a una versión más estable.
  + Ambiente producción: Empleada en una vps oracle cloud. Rama estable y producción de la API. Todas las features terminadas y estables terminan su ciclo de vida de desarrollo cumplido acá. Debe estar ejecutada bajo un ambiente linux, idealmente ubuntu server para asegurar la estabilidad/seguridad (actualizaciones) y una estandarización del sistema. Como requisitos obligatorios la instancia debe tener instalada el runtime de python3 en su versión 3.10 para la instalación de paquetes mediante pip y la ejecución en estable del subproyecto.
* **Rama base de datos** 
  + Ambiente producción: Empleada en su respectiva instancia cloud de base de datos. Esta es una rama muy importante y delicada, por lo que toda pull request con scripts debe estar aprobada por lo menos por una persona que tenga bastos conocimientos del sistema. Obligatoriamente se emplearán dos personas (idealmente el administrador con el desarrollador backend) para desarrollar y aprobar los trabajos hechos en esta rama. Estos scripts, tanto de dumpeo, como de creación de esquemas, vistas, etc. serán insertados manualmente previamente sea haya modelado su esquema de datos en la respectiva base de datos.

### Features en detalle

La arquitectura, tal como está explicada en los diagramas consta de una constante conversación entre diversos microservicios. Resumido seria lo siguiente:

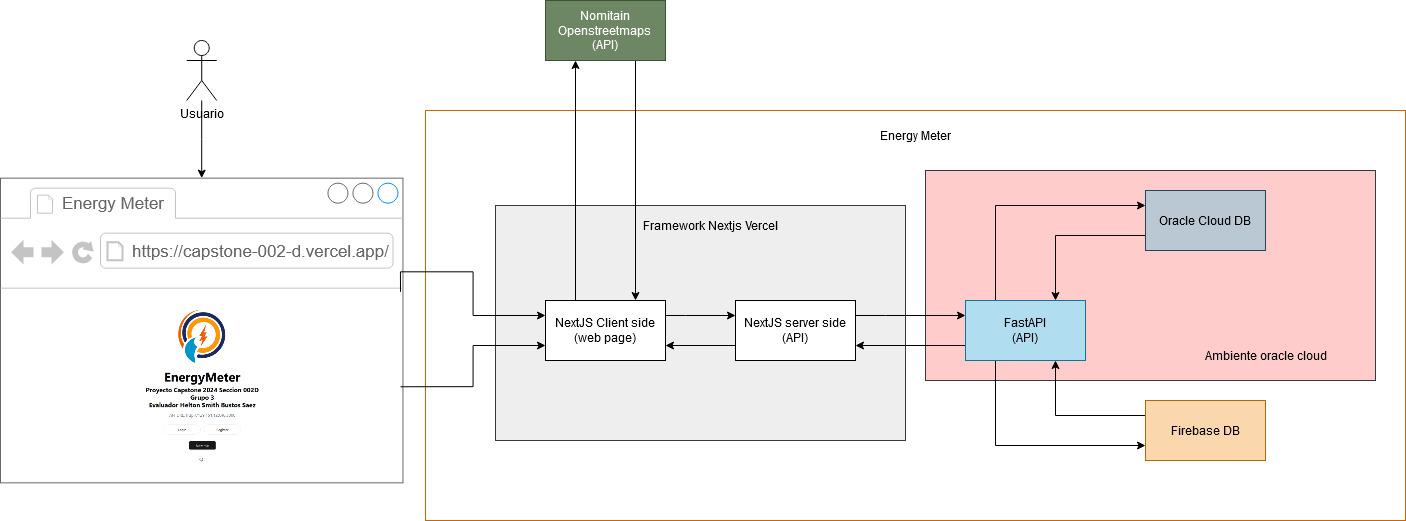


Diagrama general simple del proyecto

¿Por qué optamos por un sistema doble API? Por seguridad y eficiencia:

Seguridad: La dirección algunos endpoints sensibles quedan expuestos en la versión del código minificada en el código del cliente, estos pueden verse en los devtools del navegador. Eso generaría que algún usuario con los conocimientos suficientes pueda pasarse de listo y lograr llegar a información que no debe llegar por un procedimiento normal. Por lo que optamos por usar los endpoints con enfoque de server side que ofrece el framework para que actúe como puente entre la página y la API real como una capa adicional de seguridad. De esta forma podemos ocultar lo que serían los endpoints oficiales que contienen la lógica sensible de nuestro proyecto, como la autenticación o el retorno de información personal en FastAPI, mientras que NextJS solo se encarga de devolverlos.

Eficiencia: FastAPI a diferencia de NextJS es un framework que está hecho con un enfoque al desarrollo de APIs, por lo que su rendimiento es mayor bajo estas tareas. Si bien es cierto que las APIs de NextJS tienen una mejor integración con las páginas hechas en el mismo framework, tiende a tener una diferencia sustancial con el tiempo de respuesta que ofrece, por lo que optamos utilizar una arquitectura un poco más compleja para asegurar tiempos de cargas más óptimos. Sin embargo, hay factores que no podemos controlar, como las especificaciones del ambiente del usuario, los recursos de la máquina de este, o su conexión a internet.

Con nuestras bases de datos pasa algo parecido. Oracle cloud tiene muchas facilidades para el desarrollo de sistemas ya que posee ciertas funciones que aligeran bastante el desarrollo. Sin embargo en la capa gratuita que utilizamos, se queda muy corta para la cantidad de peticiones que se harán. Por lo que por motivos económicos, nos vemos forzados a utilizar una segunda base de datos para manejar la persistencia de los datos en tiempo real, escogimos Firebase por su accesibilidad.

A continuación se explicaran las features de la página en un nivel de detalle más técnico, intentando explicar como funcionaria en pseudocodigo

#### Home

Página básica, solamente sirve para dar una bienvenida al usuario cuando entre al dominio. Contiene botones para redirigir al login, redirigir al registro, cambiar de tema, un botón con información del proyecto y por motivos de facilidad técnica un link con la documentación de la API generado por la misma API.

#### Login

Página para poder autenticar al usuario, consta de dos entradas para el email y la contraseña y dos botones: uno para iniciar sesión y otro que redirige al usuario a una página para que recupere su contraseña. El flujo sería el siguiente:

1. Primero que todo el usuario debe ingresar sus credenciales, el sistema por la parte del cliente se dará cuenta si es que el usuario no ha llenado alguno de los campos, por lo que se verá obligado a rellenar esas credenciales. Existe la posibilidad de que el usuario tenga conocimientos de programación, por lo que si el usuario llegase a forzar el envío del formulario con un campo menos, no funcionaria debido a que tiene una capa de validación en la API interna del framework web y la API conectada a la base de datos como medidas de seguridad. En caso de que el usuario no recuerde su contraseña, tendrá la posibilidad de accionar el enlace a “olvide mi contraseña”, explicada en su respectivo flujo.
2. Una vez el usuario llene el formulario completamente y accione el botón, los datos escritos se pararan a un objeto en formato JSON a un endpoint interno del framework que está en el mismo dominio de la página (/api/account/authenticate/), estos datos serán recibidos y los enviará “en crudo” crudo a la API en FastAPI y el endpoint quedará esperando una respuesta mientras FastAPI procese los datos.
3. FastAPI procederá a procesar los datos obtenidos, de modo que hará un hash de la contraseña, y junto con el email consultará los datos en la base de datos en la tabla de usuarios mediante el uso de Modelos. Si retorna un valor, la API generará un Token que estará vinculado al usuario y este token será retornado a la NextJs con el código http 202. En caso de que no retorne un valor o la petición por algún motivo le falte un parámetro obligatorio como pueden ser el email o la contraseña, o incluso tenga parámetros demás la API retorna un respectivo código HTTP, sea 500 o 400.
4. El endpoint recibirá el código de estado HTTP con su contenido adicional (mensajes por header o body). Si el código HTTP es 202, la API de NextJs generará varias cookies, siendo la más importante la que contiene el token recibido. Con este token se podrá acceder a los endpoints protegidos por el middleware. En caso de que el retorno del código http sea distinto de 202, el frontend se encargará de desplegar un mensaje de error.

#### Register

1. Página dividida en 3 columnas (Datos personales, dirección e información de electricidad) con varios campos para obtener información personal. Todos los campos son validados a medida que estos se van llenando de forma automática.
   1. La columna de datos personales consta de campos básicos de identificación. Tales como Nombre, Apellidos, Correo, contraseña, etc.
   2. La columna de dirección, consta de campos SELECT, estos campos son listas de direcciones desglosadas para estandarizar la información de forma numérica y sea más fácil de procesar por la API. Al ser esto estandarizado, hicimos que cada campo se vaya desbloqueando a medida que el campo anterior haya sido llenado. En resumen:  
      Primero se debe llenar el campo del país, luego el campo de la región, luego el campo de la comuna, luego el campo de la calle y finalmente el campo del número. Siendo estos últimos dos que no son de tipo lista. Adicionalmente, el campo de calle no pasa por una petición controlada por nosotros. Pasa por una API totalmente externa a nosotros llamada Nomitamin, esto con tal de asegurar de que el usuario esté ingresando direcciones válidas y reales.
   3. La columna de información electrica contiene una lista de empresas de electricidad que se consumen de la misma manera que las de dirección, y campos de números para ingresar los costos de la última cuenta del mes y generar un precálculo que será enviado a la API. Estos últimos campos son opcionales.
2. El procedimiento es casi idéntico al endpoint de login. Los datos se transforman a objetos, se envían al endpoint /api/account/register para enviarlos a la API y esperar una respuesta.
3. FastAPI Recibe estos datos, los desglosa y les da su debido procesamiento, guardando la información una vez esté hasheada, validada y asegurada según el tipo de dato. Si todo esto sale bien, retorna un código http de estado 202, de no ser así, se retorna un código de error 400 al igual que con el login.
4. La API de NextJS recibe el código http y según su valor mostrará un mensaje de cumplimiento o de error en el frontend.

#### Olvidé mi contraseña

1. Página que contiene un solo campo de tipo email y un botón. Este formulario se encarga de tomar el campo ingresado para hacer una solicitud mediante el puente de NextJs API una vez el email se haya transformado en un objeto, y del mismo modo que antes, el endpoint esperará la respuesta de la API. (Esto será bastante recurrente, por lo que desde ahora esta explicación se omitirá debido a que en todas las peticiones se ejecuta el mismo procedimiento)
2. FastAPI procesa el email recibido utilizando la cláusula where, en caso de que se haya encontrado un email, mediante gmail enviaremos un correo hacia la dirección del usuario con un enlace generado que contiene la siguiente estructura:  
   [https://capstone-002-d.vercel.app/recover/{token](https://capstone-002-d.vercel.app/recover/%7Btoken)}  
   El token es generado por la API al momento de que se encuentra una cuenta con un correo igual al proporcionado.
3. La página generada tendrá un formulario de 2 campos de contraseñas, en la cual hará una nueva petición a FastAPI de tipo PUT para poder actualizar el campo de contraseña según el correo correo. ¿Cómo obtendremos la información del correo? en base al mismo token generado, ya que este token funciona como llave de sesión y queda vinculado a la cuenta de forma temporal.
   1. En caso de que un usuario tome este enlace post cambio de contraseña, la página redireccionará hacia el login.
   2. En caso de que la petición hacia la API para hacer el cambio de contraseña de un error, retorna un código http de estado 500 o 400 según el tipo de error y se desplegará un error para que se vuelva a intentar.

#### Dashboard

Funciona como página de bienvenida al usuario después de que este inicia sesión, esta página mostrará diversas métricas generales de su consumo actual, histórico y desglosado por los dispositivos que tenga a su disposición. Estos datos serán obtenidos usando como referencia el token vinculado a la cuenta, se hará una petición a la API para obtener toda la información referente a los dispositivos, tanto generales históricas y específicas.

FastAPI por su parte, clasifica las peticiones que se requiera obtener en tiempo real como históricas para ir a buscarlas a sus respectivas bases de datos. Los controles de envío funcionan al igual que el resto de peticiones, si se cumple la petición el sistema funciona como debería, mostrando la información correcta y esperada en cuanto a métricas. De no ser así, mostrará el correspondiente mensaje de error.

#### Dispositivos

Esta página se encarga de administrar de forma general los dispositivos del usuario

1. Primero que todo, hace una solicitud a la API consultando los dispositivos que tiene a su disposición con su respectiva información.
2. Si la solicitud es correcta, la API le retorna una lista con subobjetos con todos los datos de los dispositivos.
3. Con estos datos, la página muestra de forma organizada todos los dispositivos con los datos obtenidos.
4. Si el usuario lo desea, haciendo click en el título del dispositivo, será redireccionado a la página del dispositivo para mostrar su información en detalle.

#### Dispositivo

Página que muestra la información de un dispositivo específico previamente seleccionado. Muestra la información histórica y en tiempo real, adicionalmente muestra el probable costo monetario del dispositivo en tiempo real.

Para conseguir esta información, se hace el siguiente procedimiento:

1. Usando el token de usuario como llave y el identificador serial presente en la URL se harán peticiones a la API para poder obtener la información referente al dispositivo tanto histórica como en tiempo real.
2. El usuario no podrá acceder a los dispositivos de los demás usuarios debido a que se necesitan tanto un token de inicio de sesión, como el identificador serial, este identificador queda vinculado a un usuario como forma de autenticación entre dispositivos. Por lo que si el usuario con un token AAAA quiere ver el dispositivo 2222 perteneciente al usuario con token BBBB, este no podrá hacerlo.

Del mismo modo, si algunas de estas peticiones llegasen a caer en algún error, se mostrará el debido mensaje al usuario para que lo vuelva a intentar.

#### Agregar dispositivos

Existe una tarjeta en la página Dispositivos que permite al usuario añadir un dispositivo nuevo.

1. Al accionar esta tarjeta, se abre un formulario encima de la página con dos campos, un campo para ingresar un número serial, y un campo para asignarle un nombre.
2. El usuario debe ingresar los datos proporcionando el código serial mostrado en el display del dispositivo que aparece al momento de encenderlo y conectarlo al wifi por primera vez.
3. Opcionalmente puede ingresar un nombre
4. Se hará una petición a la API para poder registrar este dispositivo y vincularlo a la cuenta del usuario, siguiendo el mismo flujo.

#### Modificar dispositivo

Existe un formulario en cada página del dispositivo en la parte inferior de la página que permite modificar la información relevante del dispositivo. Tales como cambiar el nombre, dar asignación de límites de consumo diario, mensual, y constantes. Estos se harán mediante una petición POST a la API y al igual que todos los formularios, estos valores están validados apenas el usuario escriba los respectivos valores.

#### Cuenta

Página que contiene un formulario con toda la información relevante del usuario, previamente rellenado con la información que el usuario proporcionó en su registro, adicionalmente contiene un enlace que redirige a la página de modificar contraseña.

La información pre obtenida se obtiene mediante una petición a la API:

1. Obteniendo la cookie de sesión, se hace una petición get a la API que busca obtener los datos del usuario,
2. Si la petición se cumple con éxito retorna un objeto con toda la información del usuario exceptuando la contraseña,
3. De no ser así. la página mostrará el mensaje notificando según el tipo de error que presente

#### Modificar contraseña

Página enfocada al cambio de contraseña del usuario, consta de 3 campos que son validados apenas se ingresan son validados. Los campos son: contraseña actual, nueva contraseña, repetir nueva contraseña.

Una vez estos campos son validados, se hacen tres peticiones a la misma API:

1. La primera petición se hace al mismo endpoint para obtener los datos del cliente, tal como se hace en la página de /cuenta. Esto se hace con el fin de poder obtener el email.
   1. En caso de que esto falle, se pedirá al usuario que intente nuevamente mediante un mensaje en la parte inferior del formulario.
2. Una vez ya obtenido el email, se utilizará la contraseña proporcionada en el campo de contraseña actual para hacer una petición al endpoint de autenticación.
   1. Si el endpoint retorna un valor 202 quiere decir que la contraseña actual ingresada es correcta, haciendo que el flujo continúe.
   2. En caso de que retorne 401 quiere decir la contraseña actual ingresada es incorrecta o el email no se obtuvo de la forma correcta, mostrando el respectivo mensaje de error
   3. Si retorna 500 quiere decir que hay un error interno en el servidor,mostrando el respectivo mensaje de error
3. Ya validada la contraseña actual, se hará una nueva petición a la API a un endpoint de modificación de contraseña, este endpoint recibe como parámetro el token del usuario y el campo de nueva contraseña. Esto se transforma en un objeto y se envía mediante PUT a la API. De ahí el procedimiento es lo mismo que siempre:
   1. En caso de que todo pase de manera correcta, la API retorna un código 202 mostrando un mensaje de que la contraseña de cambio de manera exitosa
   2. Si da un error 400 o 500 se muestra el respectivo mensaje de error.

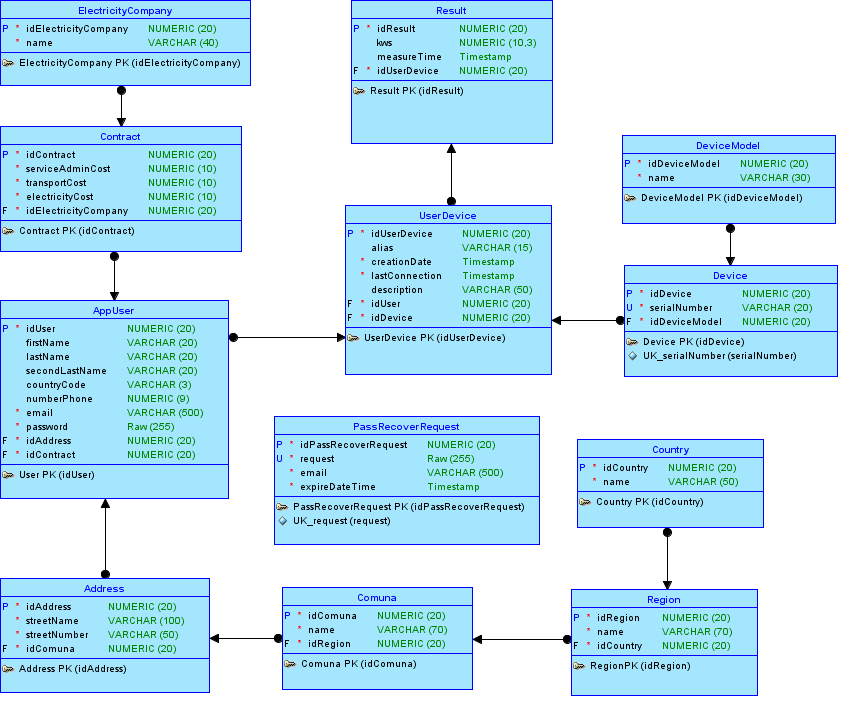
#### Notificaciones

Página para configurar las notificaciones del usuario. Esta página consta de un formulario simple con una lista de checkbox. El usuario debe elegir cómo le gustaría que le llegaran las notificaciones entre sms, whatsapp, email, o notificaciones push. Una vez el usuario confirme sus configuraciones, se enviará una petición a la API mediante el método PUT con un objeto que contiene la lista de los cambios. La API Guardará esta información en la base de datos

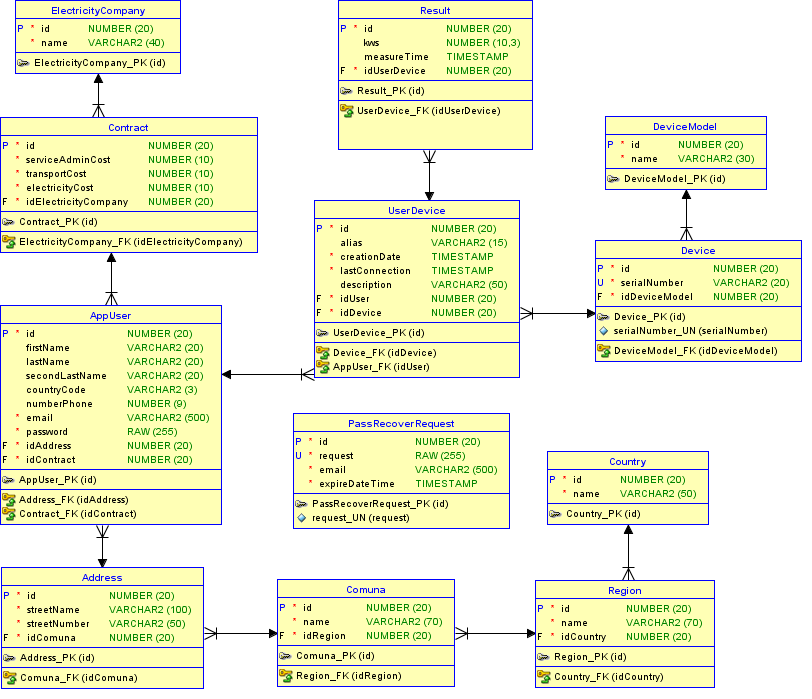
#### Cerrar sesión

Botones que están presente a lo largo de casi todo el proyecto por parte del front end, siendo los únicos lados donde no están presentes es en las páginas que no están protegidas. Esta feature consta solamente de enviar una petición a la API de NextJS en el endpoint /api/account/killsession, en el cual esto se encargará de buscar la cookie con el token de sesión y eliminarlo del cliente. En caso de que esta tarea no se pueda cumplir redirigirá hacia la página de login.

### Esquemas Oracle Cloud DB

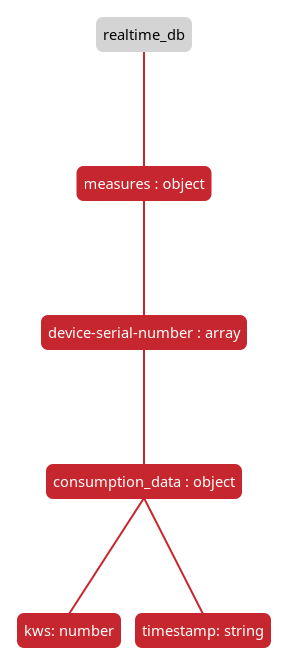


Modelo de datos lógico base de datos Oracle



Modelo de datos relacional base de datos Oracle

### Esquemas Firebase DB



Esquema firebase realtime db

## Codificaciones de ejemplo

Como se ha mencionado anteriormente, este proyecto ha sido desarrollado con NextJS y FastAPI, por lo que a continuación se mostraran algunos ejemplos del código, explicando en parte lo que hacen algunos fragmentos en sus respectivos lenguajes de programación.

**Register**

****

Declaración de variables que alteran su estado en react, considerables como getters y setters en otros lenguajes de programación que tenga un paradigma orientado a objetos.



Transformación de datos de formulario a un objeto para ser posteriormente enviado a la API

****

Manejo de contraseñas en el FrontEnd, en los cuales se utilizan expresiones regulares para la validación del formato de la contraseña, buscando asegurar que la contraseña ingresada sea lo suficientemente robusta.

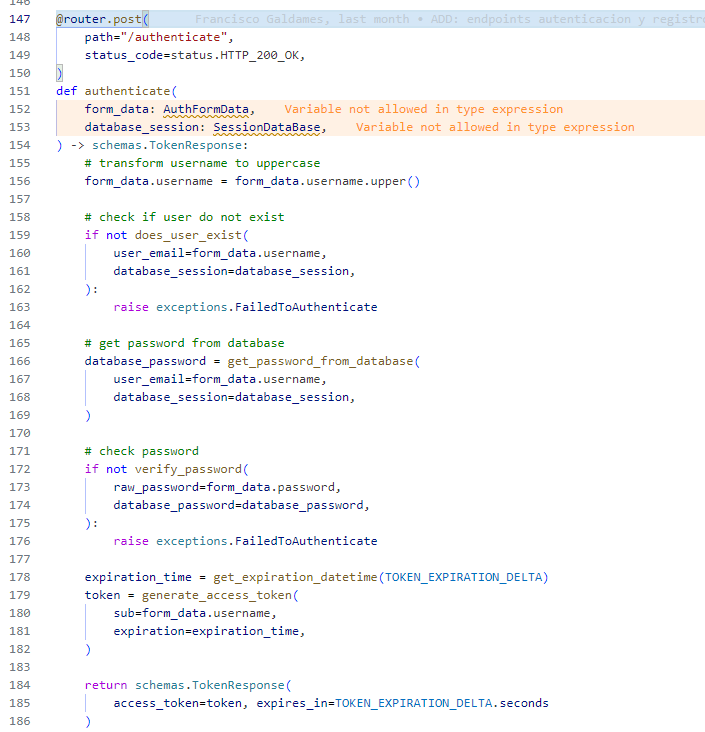
**Login**

****

Petición de autenticación a API. En este fragmento se aprecia todo el procedimiento de la petición desde el punto de vista del frontend con su debido control de errores, seteo de mensajes y uso de códigos HTTP.



Código de la API de NextJS Actuando como puente entre el frontend y la API en FastAPI. Aquí se aprecia cómo se reciben los datos desde el frontend para posteriormente ser enviados y esperar la respuesta de FastAPI. También se aprecia cómo se guardan las cookies en el cliente.



Código en python para el framework de FastAPI, mostrando el endpoint de autenticación y el retorno de datos.



Código middleware, encargado de proteger las páginas cuando el usuario no tiene el token de sesión.

Un middleware es un componente de código a bajo nivel que se ejecuta en todas las peticiones del sistema, antes de que se ejecute. Por lo que internamente, el orden de comunicación entre peticiones al servidor sería la siguiente:



Flujo de peticiones en sistema desde Cliente hasta base de datos

Primero se declaran las rutas estáticas, tales como dashboard, cuenta, dispositivos, notificaciones. Luego las rutas variables con el uso de una expresión regular.

Dentro de la función del middleware, se busca la cookie “authenticated”, que es la que tiene el token de autenticación, y haciendo uso de las librerías de backend de NextJs se redefine una redirección según la ruta que se tenga que acceder.

Ya definidas las constantes de autenticación y rutas protegidas, el código evalúa mediante la sentencia if si existe la cookie y si la ruta a acceder está protegida, para determinar si es necesario redireccionar a la página login, de no ser así, el middleware permite la redirección hacia la página solicitada.

## Frameworks explicados

### NextJS

Framework de desarrollo web, enfocado tanto para frontend como para backend basándose en las bibliotecas de react. Permite escribir APIs y tiene beneficios como la compresión de imágenes, integración directa con Vercel, optimizaciones de código al momento de buildear el proyecto.

#### React

Biblioteca de desarrollo frontend basada en componentes. Permite escribir código de servidor como de cliente que sea reutilizable. Usa un sistema de DOM Reactivo para poder alterar los estados de los componentes de forma virtual en tiempo real sin la necesidad de una recarga completa de la página, y de Javascript XML (JSX), una variación de javascript que permite escribir código XML para posteriormente ser procesado y que su salida esté en el DOM Reactivo.

### FastAPI

Framework dedicado para el desarrollo de APIs en Python. Permite una rápida escalabilidad de los sistemas debido a su fácil sintaxis.

## Herramientas utilizadas para el desarrollo

### Esp32

Esp32 es un SoC (System on chip) que cuenta con las especificaciones justas y adecuadas para nuestro proyecto. Permite crear circuitos y programarlos bajo el lenguaje de programación C, o python según su firmware. Esta placa viene con un procesador de 32bits dual core a 240 mhz, una memoria RAM de 512 KiB y SRAM de 320KiB de ROM, esto permite almacenar pequeños scripts ya compilados sin la necesidad de una memoria externa. Además, el dispositivo viene con ciertos agregados tales como: conectividad wifi, bluetooth, diversos modos de energía para asegurar una eficiencia y un buen rendimiento según el caso de uso. Se puede programar tanto en Vs Code y en Arduino Ide. En este proyecto planeamos utilizar este SoC para crear un dispositivo que ayude a medir la electricidad y mande esta información a la API.

### Sistemas operativos

Un sistema operativo es un conjunto de programas en ejecución de una máquina que gestiona los recursos físicos, y provee servicios a los diversos procesos según su relevancia y prioridad. Esto puede variar dependiendo del tipo de usuario y uso que tenga el sistema operativo. A continuación se mostraran los sistemas operativos utilizados en el desarrollo de este proyecto, siendo la mayoría de estas distribuciones linux, Pero ¿Qué es linux?

Linux, o mas bien GNU/Linux es una terminología aplicada a una distribución de programas que siguen el principio POSIX, usando de base el kernel Linux (basado el antiguo unix, un sistema operativo escrito en C que se usaba en universidades y ambientes científicos para programación) Como diversos programas que se encargan de lo que sería la gestión de ventanas, servicios, etc.

Los siguientes sistemas operativos a mostrar, están basados en distribuciones Linux y sistemas propietarios de Microsoft..



Arquitectura por capas de un sistema operativo

#### Ubuntu server

Distribución de linux basada en Debian, enfocada para su uso en servidores, busca el equilibrio entre la seguridad mediante actualizaciones con cierta frecuencia mensual y la estabilidad intentando seguir la filosofía de Debian. En nuestro contexto, se utiliza para los ambientes de producción de nuestro poyecto.

#### Fedora linux

Distribución de linux basada en la antigua distribución RedHat, siendo una versión comunitaria de esta. Está enfocada para el uso más personal pero también tiene un enfoque de desarrollo, por lo que es una distribución ideal para ingenieros, programadores y científicos. En nuestro caso, Francisco Galdames utiliza esta distribución de linux para desarrollar.

#### Arch linux

Distribución de linux basada en la distro CRUX que sigue los principios KISS “Keep it simple, stupid!”. Famosa por su extrema ligereza y rapidez, así como de su potencial complejidad al ser una distribución extremadamente limpia y personalizable, casi sin programas que permitan una experiencia Out of the box. Si bien esta distribución es muy rápida a comparación de las anteriores dos, si se actualiza con mucha frecuencia tiende a tener problemas de estabilidad debido a la mayoría de canales de actualización que ofrece esta distribución son muy “bleeding edge”, la tecnología desarrollada no ha tenido mucho tiempo para ser probada adecuadamente.

En nuestro caso, Gabriel Soto utiliza esta distribución para desarrollar y escribir documentos.

#### Windows 10 y 11

Sistema operativo hecho por Microsoft basado en el antiguo sistema operativo DOS, actualmente es lo más estandarizado para un usuario común, permitiendo tener un ambiente de programación fácil de usar pero no tan configurable como sería una distribución Linux. Ideal para gente que prefiere una experiencia Out of the box. Actualmente este sistema operativo está siendo utilizado por Gabriel Soto y Jorge Parra para el desarrollo de este proyecto.

### Editores de códigos y entornos de desarrollo

#### Visual Studio Code

Editor de código abierto hecho por Microsoft. Permite la personalización mediante extensiones, aumentando su usabilidad para distintos tipos de proyectos. Volviéndolo un estándar en la actualidad para programar

#### Arduino IDE

Editor de código abierto hecho por la fundación de arduino. Es un derivado de visual studio code pero con un enfoque de desarrollo exclusivo para dispositivos IoT de arduino, raspberrys y/o esp, etc

#### Nvim

Editor de código abierto hecho por la comunidad basado en el antiguo vi y vim. Al igual que Vs Code, permite aumentar su usabilidad mediante extensiones para distintos tipos de proyectos, con la diferencia de que este editor tiene un enfoque de consola y no posee las mismas integraciones que podría tener VS Code a cambio de un mucho mejor rendimiento, Haciendo que sea una alternativa viable para muchos desarrolladores.

#### Oracle Data Modeler

Software para modelar bases de datos hecho por Oracle. Permite generar esquemas de bases de datos para los motores de Oracle una vez el modelo esté terminado y validado.

#### Oracle SQL Developer

Software para desarrollar y administrar bases de datos hecho por Oracle para motores de Oracle. Es un IDE de SQL que permite consultar, crear tablas, vistas, procedimientos, etc,

### Misceláneo

#### Google Chrome

Navegador web estándar de la actualidad, siendo un fork del proyecto de código abierto Chromium (patrocinado principalmente por Google). Es el navegador por excelencia del usuario final, teniendo la mayoría de las funcionalidades definidas por la web estables y funcionando de forma correcta. Además de tener integración directa con los servicios de Google, es ideal para la mayoría de usuarios que poseen Teléfonos Android e Iphones. Utilizado para probar la página web

#### Firefox

Navegador web alternativo a Google Chrome, A diferencia de la mayoría de navegadores de la actualidad, no es un fork de Chromium ni tampoco utiliza el motor javascript V8 ni el motor de renderizado Blink que utilizan los navegadores Chromium, si no que usa sus propios motores, sin perder casi ninguna de las funcionalidades esenciales que ofrece la competencia. Firefox es un navegador ideal para desarrolladores o personas que buscan una experiencia más privada en su día a día. Además de que hace un mejor uso de recursos de la máquina. Utilizado para probar la página web

#### Postman

Postman es un proyecto que nace a partir del framework Electron, un framework de JavaScript desarrollado para poder llevar las tecnologías web a las tecnologías nativas de escritorio utilizando de base Chromium. El principal objetivo de Postman trata de poder probar los endpoints de las APIs desarrolladas de una forma mucho más sencilla, permitiendo el uso de variables, intercambiar entre los métodos HTTP y poder visualizar las salidas de cada petición. Actualmente, se usa principalmente para poder desarrollar APIs en nuestro proyecto.

### Runtimes

#### NodeJS 20

NodeJS es un runtime desarrollado en C++, si principal objetivo es llevar el lenguaje javascript que es comúnmente utilizado exclusivamente en el desarrollo de páginas web por parte del cliente en un entorno local, permitiendo poder escribir aplicaciones de escritorio, de servidor y comúnmente de backend. En nuestro proyecto es utilizado para poder ejecutar el framework NextJs.

#### Python3 3.10

Python es un lenguaje de programación y runtime escrito en C++. Tiene una sintaxis básica que permite crear un código rápido de programar y una sintaxis legible, muy parecida al lenguaje

natural inglés. En nuestro proyecto es utilizado para poder programar la API y ejecutar el framework FastAPI

#### GCC (Arduino)

GCC arduino es un runtime de programación que soporta los lenguajes de programación C y C++ con un enfoque de programación para dispositivos IoT de Arduino, Raspberry Pi y/o ESP, etc

# Conclusiones Individuales

**Francisco Galdames**

El proyecto considera resolver problemas que son comunes para la sociedad pero que generan un gran impacto en las vidas de las personas. Acerca de esto concluyó que el proyecto tiene mucho potencial, sólo necesita más desarrollo. También nuestro proyecto depende económicamente del público consumidor, sin estos no puede sostenerse la mantención del mismo. Ahora, refiriéndome a mi carrera, he podido desarrollar aún más mis habilidades y formación como ingeniero informático.

**Jorge Parra:**

Considero que el proyecto nos ayudó a poner en práctica lo ha aprendido en la carrera debido a usar el desarrollo web, base de datos junto a organizar el proyecto con metodología ágil que permitieron realizar una solución que considero escalable y viable para el mercado largo plazo, debido a que como mencionamos varias veces está muy arraigado a tomar en cuenta una problemática que puede afectar en los gastos comunes de las personas que este caso la el gasto de electricidad reflejado en la cuenta de la luz por lo que el considerar esto siento que el vender esto como un proyecto puede considerarse una buena inversión al permitir ser conciente de acuerdo a los gastos antes mencionados.

**Gabriel Soto**

Considero que nuestro proyecto toca un tema crítico que las empresas de electricidad, a pesar del avance tecnológico no han sabido resolver: La falta de herramientas accesibles para poder monitorear la electricidad EN TIEMPO REAL que puedan ofrecer métricas detalladas y que tengan funcionalidades intuitivas casi no existen, y creo que un medidor con una luz parpadeante no es suficiente para saber cómo va nuestro consumo eléctrico. Nuestro producto podría permitir a los usuarios auto informarse antes de tomar decisiones, reducir costos, adoptar consumos eléctricos más responsables, o incluso descubrir fugas de energía en sus domicilios.

Además de tratarse de un mercado prácticamente nuevo (al no existir solución directa de las empresas), el proyecto presenta un alto potencial de rentabilidad. Por lo que creo que es posible aprovechar la economía de escala para reducir los costos de fabricación de los dispositivos, haciendo que estos sean accesibles para el público final.

Considerando todas las métricas que podríamos tomar, este mercado podría expandirse a puntos de aplicaciones de ciencias de datos, análisis predictivos, etc. Posicionando nuestro proyecto como una solución innovadora, escalable y sostenible.

# Reflexión grupal

Creemos que nuestro proyecto tiene mucho potencial, ya que tal como hemos mencionado con anterioridad, toca una problemática nueva y creciente con los anuncios de incremento del precio de la electricidad para las Regiones Metropolitana y de Valparaíso, sin embargo, creemos que nuestro proyecto es muy dependiente del apoyo del público, ya que para poder sacar una ganancia y que pueda ser un proyecto autosuficiente necesitamos de un apoyo de los clientes finales que compren los dispositivos que tenemos planificados manufacturar. Puede ser una inversión riesgosa, si consideramos que uno de nuestros integrantes motiva mucho la idea de la economía de escala para generar productos a bajo costo. Pero respondería a la problemática propuesta en su conclusión, que sería la de tener un producto que satisface la necesidad de monitorear el gasto energético doméstico a bajo costo, modular y universal, ya que no se limitaría a componentes de un ecosistema, como puede ser la serie de productos de electrodomésticos inteligentes de marcas como Xiaomi, Samsung, Apple, Lg, etc.

Es una tarea larga, pero no imposible. Lamentablemente, subestimamos mucho el tiempo de desarrollo debido a que en un principio avanzamos a ritmos agigantados en este proyecto, esto generó que en un punto del desarrollo nos relajamos mucho y terminamos con un proyecto que si bien es presentable ante la comisión, no cumple nuestras expectativas para ser considerado como un proyecto completo, solamente un prototipo que le falta mucho por pulir. Puede que esto sea muy relativo según el tipo de estudiante, sin embargo nosotros siempre hemos intentado entregar la mejor calidad de nuestros trabajos grupales en las evaluaciones finales, por lo que creemos que pudimos haber rendido más para este proyecto.

Del mismo modo, también creemos que si bien poseemos muchos conociemientos, nos falta mucho por aprender tanto en nuestras habilidades duras como blandas.

# Competencias perfil de egreso

Las competencias presentes y ejercidas en este proyecto son las siguientes:

* Ofrecer propuestas de solución informática analizando de forma integral los procesos de acuerdo a los requerimientos de la organización
* Gestionar proyectos informáticos ofreciendo alternativas para la toma de decisiones de acuerdo a los requerimientos de la organización
* Realizar pruebas de certificación tanto de los productos como de los procesos utilizando buenas prácticas definidas por la industria
* Comunicarse de forma oral y escrita usando el idioma inglés en situaciones socio-laborales a un nivel básico, según la tabla de competencias TOEIC y CEFR
* Construir el modelo arquitectónico de una solución sistemática que soporte los procesos de negocio de acuerdo a los requerimientos de la organización y estándares de la industria.
* Implementar soluciones sistémicas para automatizar y optimizar procesos de negocio de acuerdo a las necesidades de la organización.
* Resolver las vulnerabilidades sistémicas para asegurar que el software construido cumple con las normas de seguridad exigidas por la industria.
* Construir modelos de datos para soportar los requerimientos de la organización de acuerdo a un diseño definido y escalable en el tiempo
* Desarrollar una solución de software utilizando técnicas que permitan sistematizar el proceso de desarrollo, mantenimiento y asegurar el logro de los objetivos.
* Programar consultas o rutinas para manipular la información de una base de datos de acuerdo a los requerimientos de la organización.
* Construir programas y rutinas de variada complejidad para dar solución a requerimientos de la organización, acordes a tecnologías de mercado y utilizando buenas prácticas de codificación

# Bibliografías

* Duoc UC (Mayo 7, 2024). Formato Informe [<https://bibliotecas.duoc.cl/documentos-academicos-y-presentaciones/formato-informes>]
* Sebastián Dote (Julio 12, 2024). Alza en las cuentas de la luz en Chile: cuáles son los artefactos que consumen más electricidad. [<https://elpais.com/chile/2024-07-12/alza-en-las-cuentas-de-la-luz-en-chile-cuales-son-los-artefactos-que-consumen-mas-electricidad.html>]
* CNN Chile (Noviembre 13, 2024). Cuentas de luz volverán a subir en enero de 2025: Alza será de 11% en la RM y 15% en Valparaíso. [<https://www.cnnchile.com/programas-completos/cuentas-luz-subiran-enero-2025-alza-11-region-metropolitana-15-valparaiso_20241113/>]
* Iván Ramírez (Julio 3, 2018).¿Qué es una Aplicación Web Progresiva o PWA? [<https://www.xataka.com/basics/que-es-una-aplicacion-web-progresiva-o-pwa>]
* José M. Piquer (Abril 9, 1999). Arquitectura del Sistema Operativo [<https://users.dcc.uchile.cl/~jpiquer/Docencia/SO/aps/node16.html>]
* Yúbal Fernández (Septiembre 27, 2018).¿Qué es exactamente GNU/Linux? [<https://www.xataka.com/basics/cual-es-la-diferencia-entre-linux-y-gnu-linux>]
* Pavel Lazarev (Octubre 13, 2023) Collection of Diagrams to Use in Your React App for Effective Data Visualization. [<https://dev.to/plazarev/collection-of-diagrams-to-use-in-your-react-app-for-effective-data-visualization-4o7n>]
* Gabriel L. Manor (Enero 18, 2024). Best Practices for Authentication and Authorization in API.[<https://www.permit.io/blog/best-practices-for-api-authentication-and-authorization>]
* Biblioteca UNAM. ¿Cómo hacer citas y referencias en formato APA? [<https://bibliotecas.unam.mx/index.php/desarrollo-de-habilidades-informativas/como-hacer-citas-y-referencias-en-formato-apa>]
* NextJS. Server and Client Composition Patterns [<https://nextjs.org/docs/app/building-your-application/rendering/composition-patterns>]
* FastApi.OAuth2 with Password (and hashing), Bearer with JWT tokens [<https://fastapi.tiangolo.com/tutorial/security/oauth2-jwt/?h=jwt>]
* SQL Alchemy (Octubre 15, 2024). ORM Quick Start. [<https://docs.sqlalchemy.org/en/20/intro.html>]
* Espressif (Octubre, 2024). ESP32 Datasheet [https://www.espressif.com/sites/default/files/documentation/esp32\_datasheet\_en.pdf]