universidad de león

Escuela de Ingenierías

Industrial, Informática y Aeroespacial

GRADO EN INGENIERÍA INFORMÁTICA

Documento de anexos

Aplicación web para la gestión de módulos hidropónicos inteligentes

Web application for the management of intelligent hydroponic modules

Autor: José Antonio López Pérez
Tutores: Lidia Sánchez González,
Jesús Fernández Fernández

(Julio, 2025)

Anexo 1 - Seguimiento de Proyecto

El seguimiento del proyecto se ha desarrollado mediante reuniones semanales y bisemanales donde se mostraban los avances de varios compañeros en sus proyectos. Se mostraban las funcionalidades implementadas durante el periodo comprendido entre la reunión anterior, se obtenía retroalimentación por parte del coordinador y se establecían los próximos objetivos. Además, se planteaban dudas y el coordinador orientaba a los desarrolladores para abordar los retos.

El periodo final del desarrollo las reuniones pasaron a ser individuales, enfocando principalmente al cierre de los desarrollos satisfaciendo los requisitos de cada proyecto y documentando correctamente el producto final.

Todo el registro de desarrollo queda reflejado en el correspondiente repositorio de código en Github donde se puede ver la evolución completa del proyecto. En el repositorio se agrupan todos los microservicios, el proyecto de despliegue y documentación relativa al proyecto.

El enlace del repositorio de código es el siguiente: https://github.com/jlopep09/GreenHouseIOT

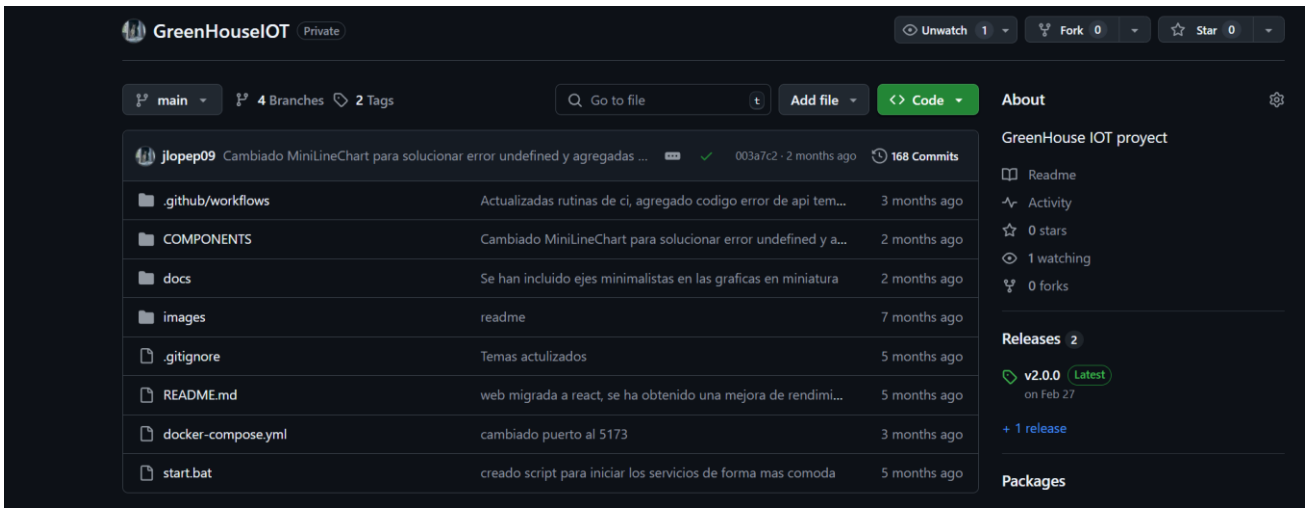


IMAGEN DEL REPOSITORIO DE CÓDIGO

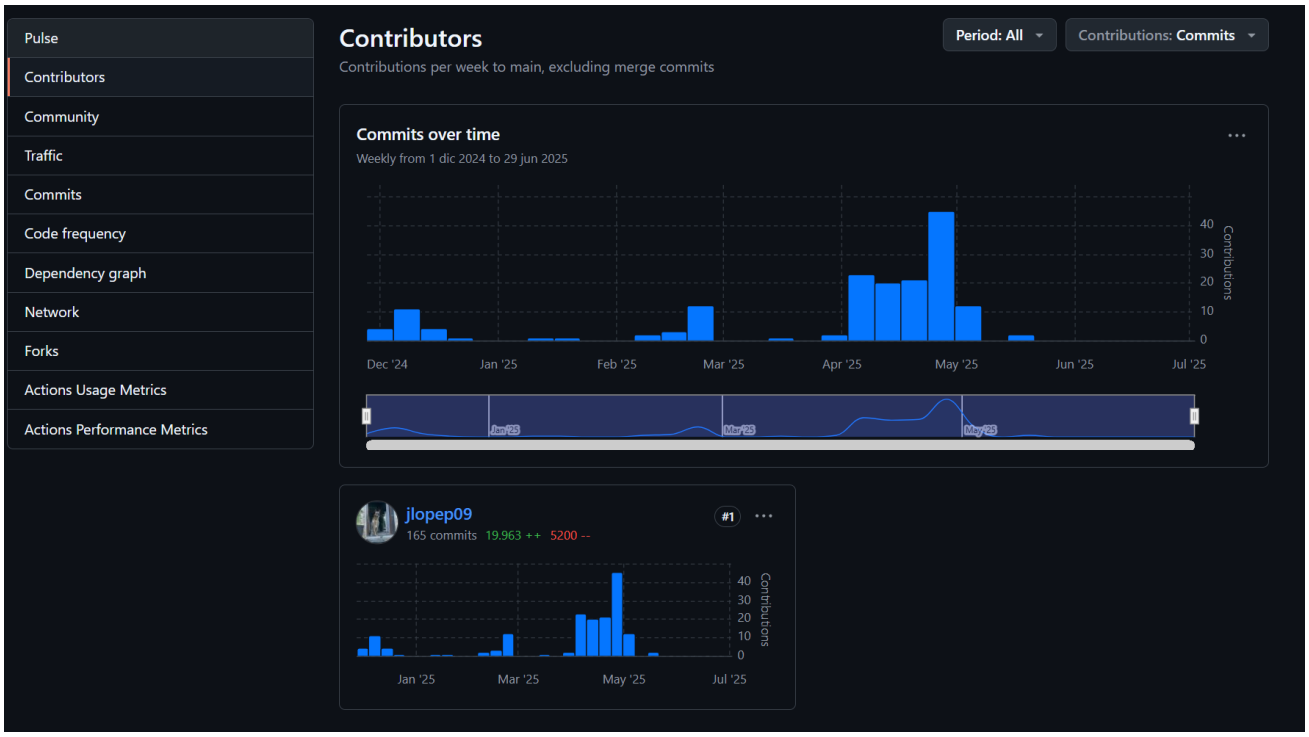


IMAGEN DE LOS CONTRIBUYENTES DEL PROYECTO Y GRÁFICA DE CONTRIBUCIONES POR FECHA.

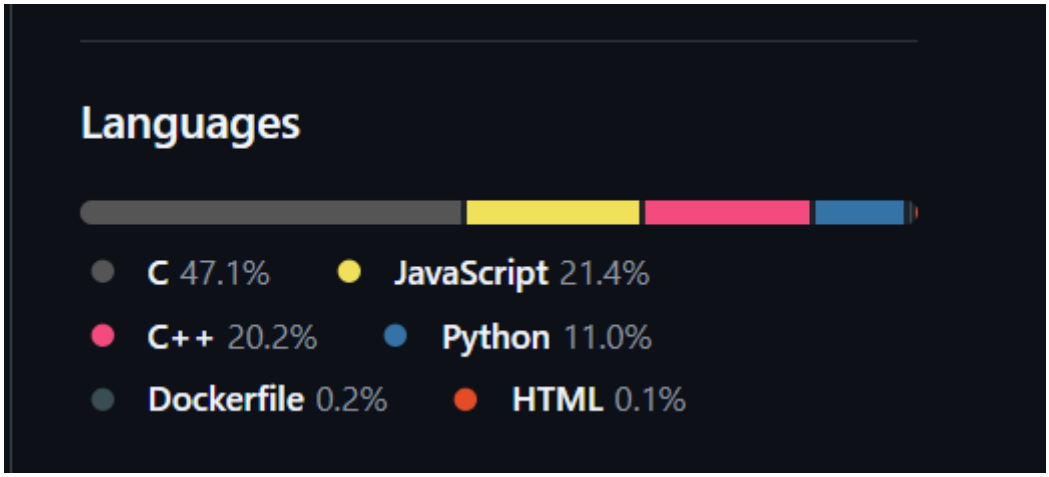


IMAGEN QUE RESUME LOS LENGUAJES UTILIZADOS EN EL PROYECTO.

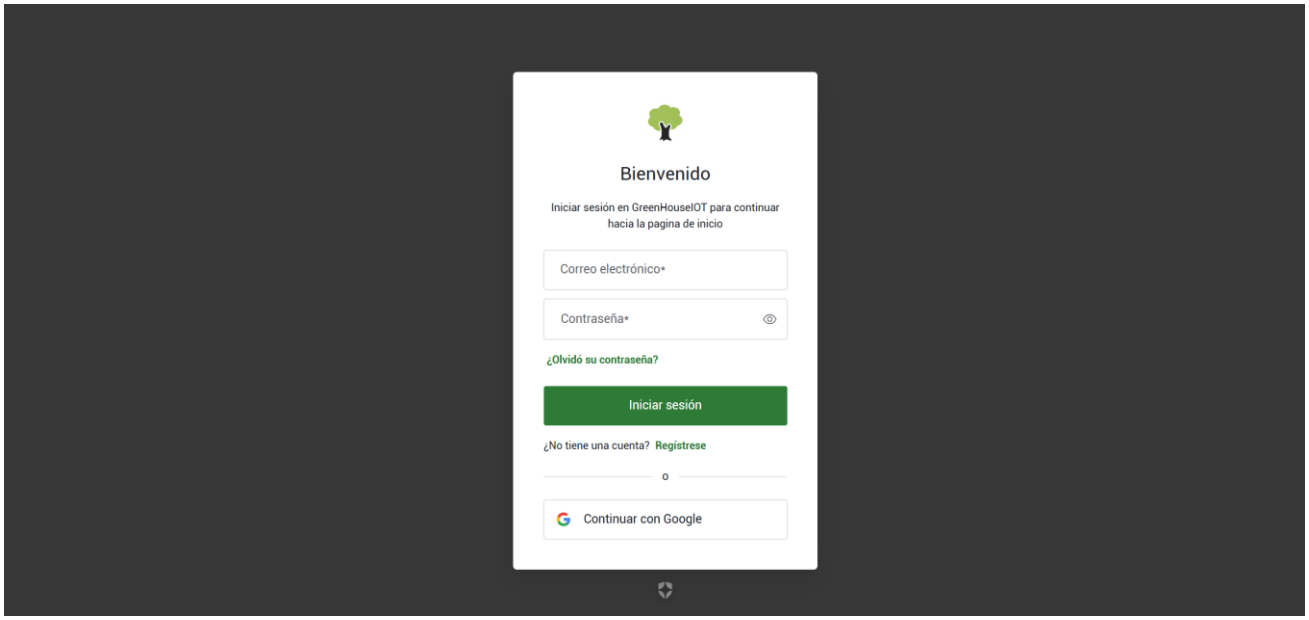
Anexo 2 - Manual de usuario

En este anexo se detalla los pasos a seguir para la instalación y uso del sistema por parte del usuario.

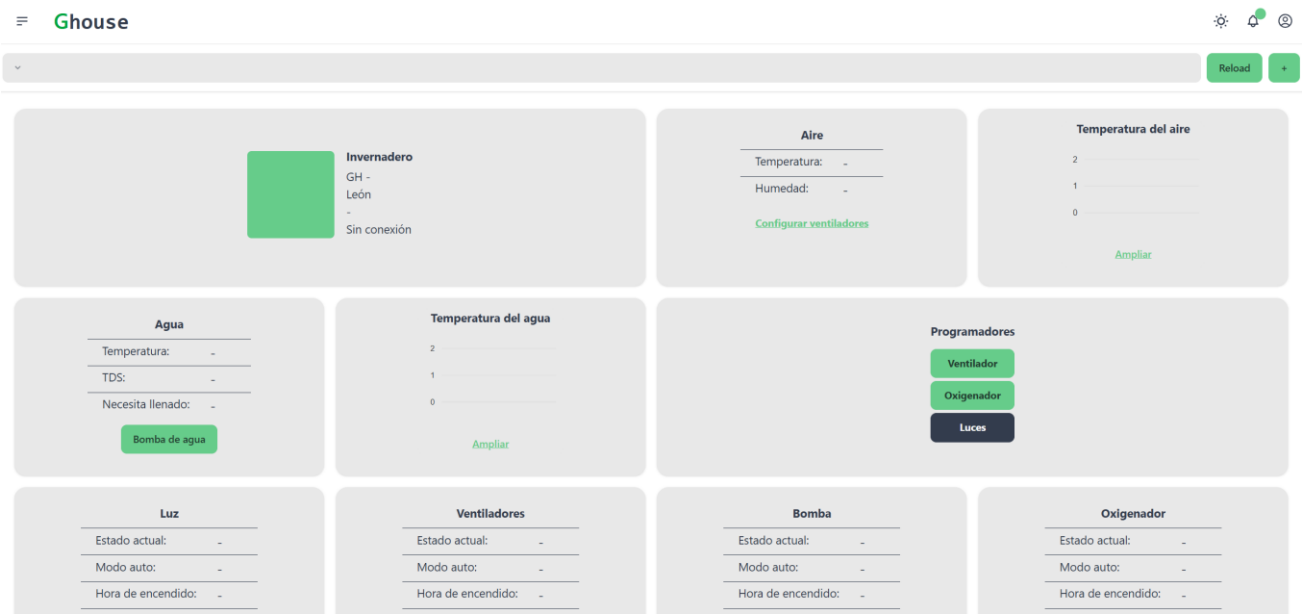
El primer paso es la adquisición/montaje de un módulo hidropónico. Es requisito necesario la configuración de algunas variables por parte del usuario como las credenciales wifi, la dirección ip local del módulo y diversas configuraciones extra. Una vez encendido todas las configuraciones pueden ser modificadas desde una interfaz web haciendo uso de la ip local del módulo en el puerto 80.

Tras la configuración, se debe encender el servicio de envío de datos en un equipo como un ordenador o una raspberry. Este servicio usa algunas variables de entorno sencillas que pueden modificarse como el servidor donde enviarán los datos. El proceso de configuraciones anteriores puede demorarse pocos minutos si el usuario está habituado al proceso. Con todo ello iniciado, el resto de las acciones se realizarán desde la interfaz web del proyecto, hosteado en el servidor VPS. En este ejemplo se encuentra en la dirección https://vps.joselp.com/

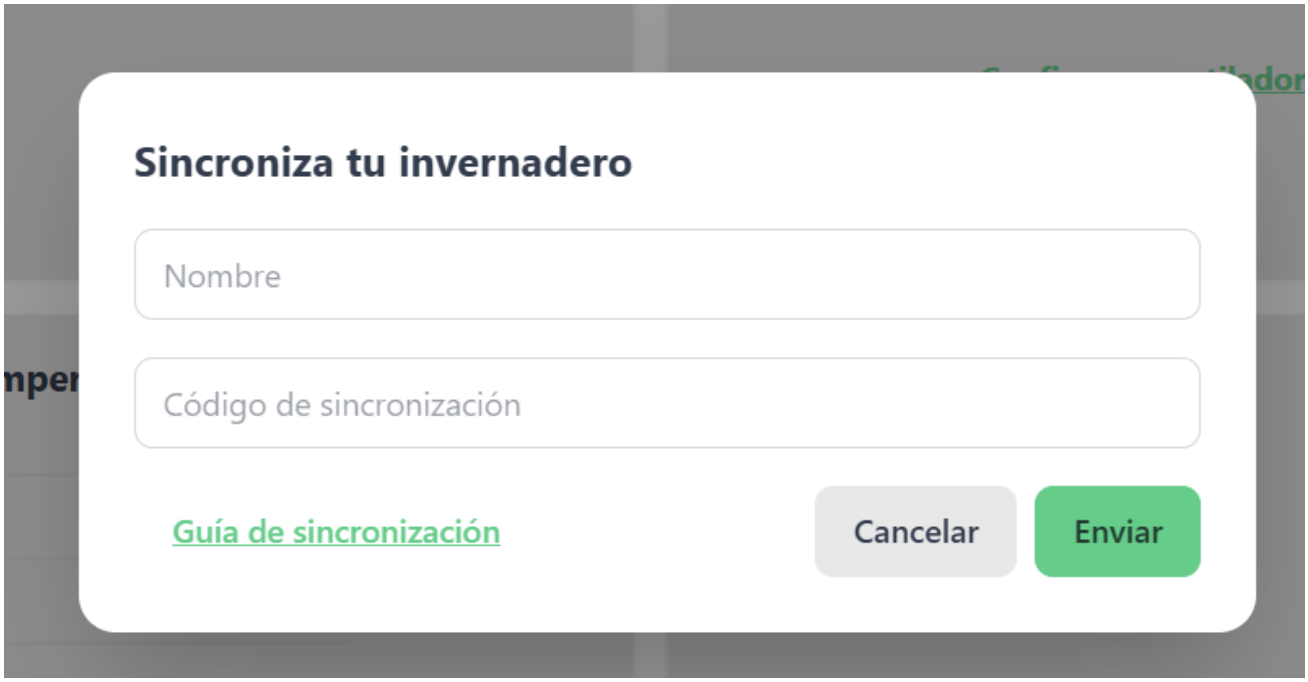
Cuando el usuario accede por primera vez debe registrarse en la aplicación, para ello puede elegir entre usar su cuenta de Google o crear una cuenta específica del servicio.



Se encontrará en la página de inicio un menú con información en blanco, ya que no hay ningún módulo sincronizado.

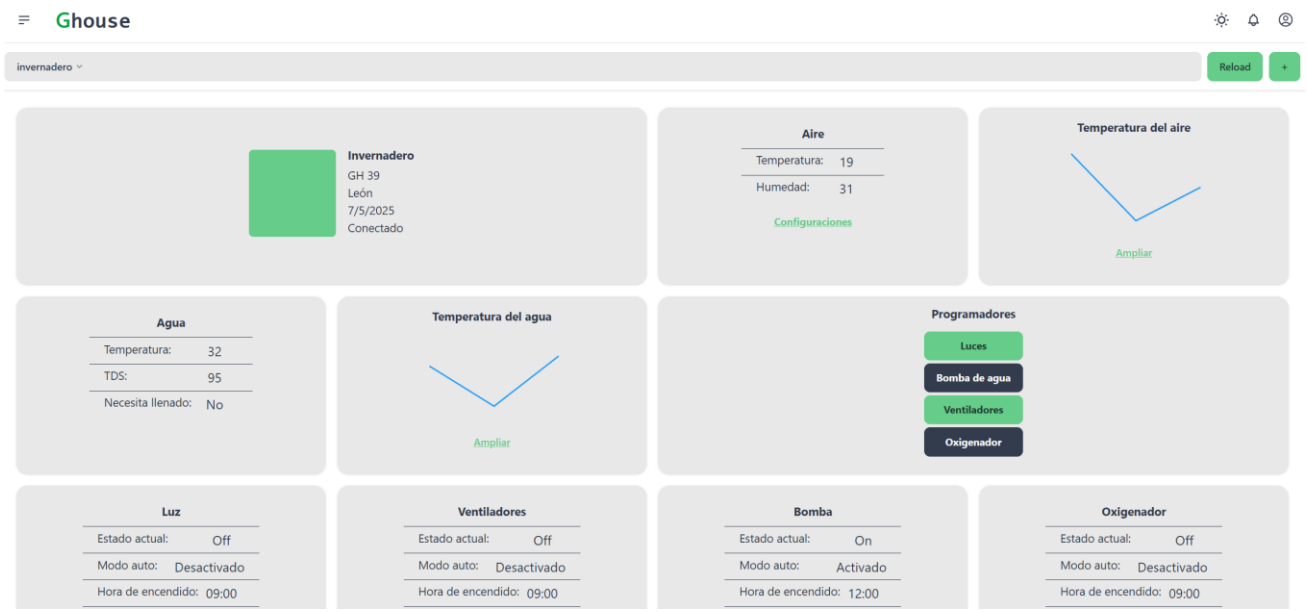


En la sección superior izquierda se encuentra el botón de sincronización representado por el símbolo +. Cuando se acciona el botón aparece un formulario para sincronizar el módulo, el usuario debe introducir las credenciales específicas de su módulo.

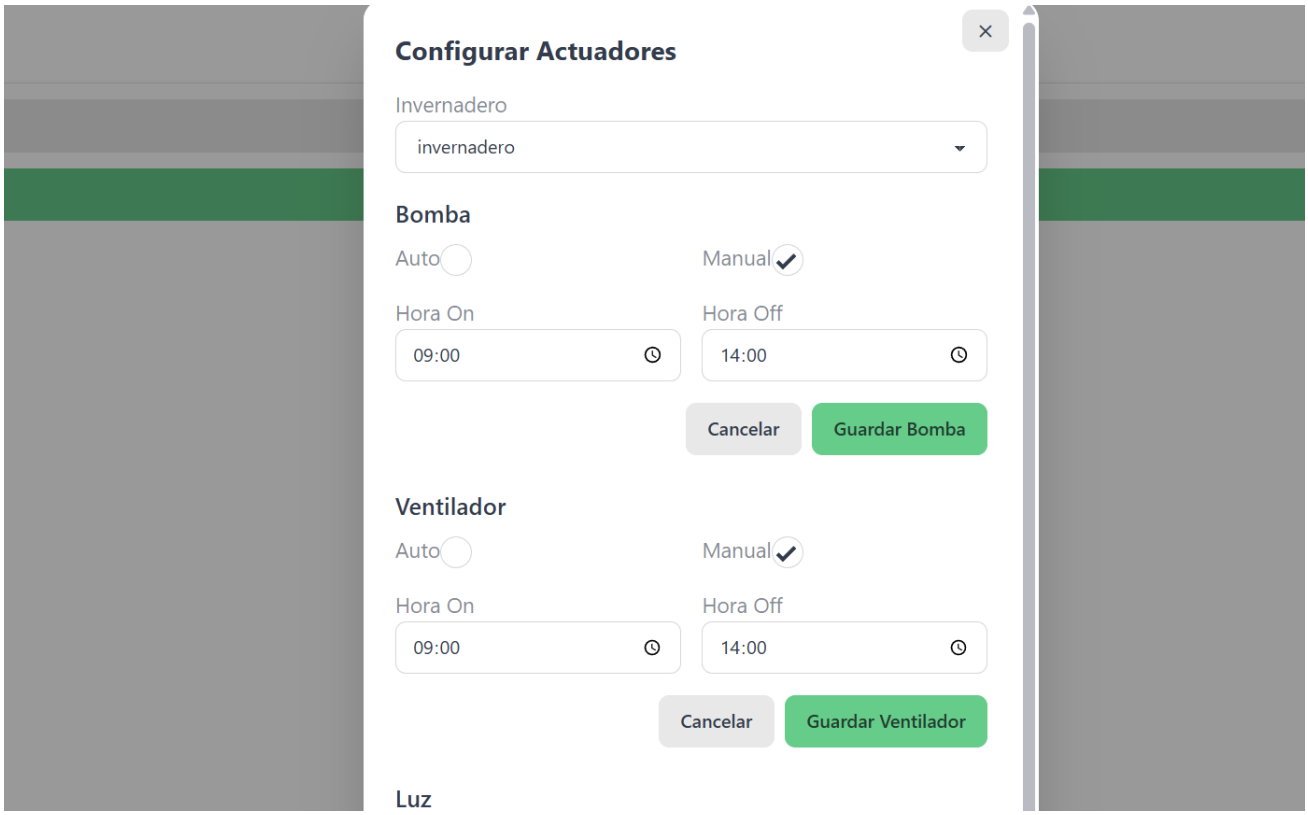


Una vez sincronizado, aparecerá el invernadero en el listado superior.

El contenido disponible del invernadero será mostrado al usuario de forma dinámica.



Las interfaces son intuitivas para que el usuario sepa en todo momento que está viendo/configurando. Además, mediante el menú de navegación puede acceder a secciones como el de configuración para programar sus dispositivos. En la siguiente imagen se muestran algunas configuraciones.



Anexo 3 – Proyecto Ansible

La idea de este proyecto es configurar de forma automatizada un servidor VPS para alojar servicios basados en imágenes Docker. El servidor detectará cuando subes una nueva imagen de tus servicios a DockerHub y hará un nuevo despliegue. El objetivo es configurarlo todo de forma automática en poco tiempo para centrarte en desarrollar el proyecto sin tener que pensar en la infraestructura en exceso. Esta alternativa te permite desplegar tus proyectos de forma más económica respecto a servicios de hosting dedicados al basar su despliegue en un servidor VPS.

Ansible es una herramienta de automatización de TI de código abierto que permite gestionar múltiples servidores y sistemas de forma centralizada. Esta herramienta dispone de una gran cantidad de funcionalidades, pero en este proyecto solo utilizaremos las relacionadas con el despliegue de microservicios Docker y configuraciones básicas.

Ansible se basa en la ejecución de código YAML organizado por tareas dentro de archivos llamados playbooks. Además, permite reunir todos los host destino en un archivo inventory.ini

En nuestro caso, solo hay un host destino.

```
[vps]
vps.joselp.com ansible_user=root ansible_port=22
```

En el playbook principal del proyecto se reúnen las referencias a todos los playbook que agrupan características más concretas, en la siguiente imagen se muestra el orden de ejecución.

```
- import_playbook: playbooks/user.yml
- import_playbook: playbooks/ssh.yml
- import_playbook: playbooks/dependencies.yml
- import_playbook: playbooks/firewall.yml
- import_playbook: playbooks/docker.yml
- import_playbook: playbooks/traefik.yml
- import_playbook: playbooks/greenhouse-db.yml
- import_playbook: playbooks/greenhouse.yml
- import_playbook: playbooks/watchtower.yml
```

Los 4 primeros playbooks se centran en configuraciones más generales, ansible nos automatizará el proceso de creación de un usuario con permisos específicos, configuración de inicio de sesión solo con una clave SSH específica, instalación de dependencias y configuración de un firewall que permita solo los puertos necesarios de nuestro despliegue. Posteriormente se ejecuta el playbook docker.yml, encargado de su instalación desde el repositorio oficial. Traefik es un servidor proxy inverso que nos permite personalizar y aumentar la seguridad de las comunicaciones. Los siguientes playbooks son los relevantes en nuestra aplicación. Algunas tareas importantes son las siguientes.

```
    - name: Iniciar sesión en DockerHub
      shell: echo "{{ docker_pass }}" | docker login -u "{{ docker_user }}"
--password-stdin
```

En esta tarea se inicia sesión en nuestra cuenta de Dockerhub para poder obtener las imágenes publicadas.

```
- name: Lanzar servicio db (MariaDB)
      docker_container:
        name: mariadb
        image: mariadb:latest
        state: started
        restart_policy: always
        networks:
          - name: mariadb_network
        env:
          MARIADB_ROOT_PASSWORD: "{{ mariadb_root_password }}"
        published_ports:
          - "3306:3306"
```

Se lanza el servicio de MariaDB.

```
    - name: Copiar script SQL al VPS
      copy:
                src:   /mnt/c/Users/Jose/Desktop/repositoreios/despliegue-
gh/resources/init.sql  # Ruta en tu máquina local
        dest: /tmp/init.sql
      become: yes
      become_method: sudo

    - name: Copiar script SQL al contenedor
      community.docker.docker_container_copy_into:
        container: mariadb
        path: /tmp/init.sql  # Ruta en el VPS
        container_path: /tmp/init.sql

    - name: Ejecutar script SQL dentro del contenedor
```

```
      shell: docker exec mariadb mariadb -uroot -p{{ mariadb_root_password
}} -e "source /tmp/init.sql"
      args:
        executable: /bin/bash
```

Para no tener que crear manualmente la estructura de la base de datos, se han creado unas tareas encargadas de copiar y ejecutar un script SQL que crea todas las tablas y elementos. Una vez tenemos la base de datos, pasaremos a la creación del backend.

```
    - name: Lanzar contenedor de data_processing
      docker_container:
        name: data_processing
        image: jlopep09/greenhouse-backend:latest
        state: started
        restart_policy: always
        networks:
          - name: red_local_compartida
          - name: mariadb_network
          - name: traefik_proxy
        env:
            DB_HOST: "{{ DB_HOST }}"
            DB_USER: "{{ DB_USER }}"
            DB_PASSWORD: "{{ DB_PASSWORD }}"
            DB_NAME: "{{ DB_NAME }}"
            DB_PORT: "{{ DB_PORT }}"
            SECRET_TOKEN: "{{ SECRET_TOKEN }}"
        expose:
          - "8002"
        labels:
          traefik.enable: "true"
          traefik.docker.network: traefik_proxy

          # Enrutado solo desde /api del dominio
            traefik.http.routers.backend.rule: "Host(`vps.joselp.com`) &&
PathPrefix(`/api`)"
          traefik.http.routers.backend.entrypoints: "websecure"
          traefik.http.routers.backend.tls: "true"
          traefik.http.routers.backend.tls.certresolver: "letsencrypt"

          traefik.http.services.backend.loadbalancer.server.port: "8002"
                                          traefik.http.middlewares.backend-
stripprefix.stripprefix.prefixes: "/api"

          traefik.http.routers.backend.middlewares: "backend-stripprefix"
```

Al solo disponer de un servidor vps ha sido necesaria la incorporación del servidor proxy inverso, en la imagen superior observamos que se ha configurado el backend en un router desde el dominio vps.joselp.com/api

Finalmente se crean los contenedores del servicio web y de Kafka. Al querer desplegar de forma pública la aplicación, fue necesaria la incorporación de un sistema de usuarios. Para ello se usó Auth0 con el fin de no almacenar contraseñas en la base de datos. Auth0 está configurado para solo permitir conexiones por el protocolo https por lo que ha sido necesario configurar el proxy para que solo permita conexiones por este protocolo y nos brinde una certificación SSL/TLS.

```
    - name: Lanzar contenedor de web
      docker_container:
        name: web
        image: jlopep09/greenhouse-web:latest
        state: started
        restart_policy: always
        networks:
          - name: red_local_compartida
          - name: traefik_proxy
        env:
          VITE_APP_NAME: "GreenhouseIOT"
          VITE_DDBB_API_IP: "{{ VITE_DDBB_API_IP }}"
          VITE_SECRET_TOKEN: "{{ SECRET_TOKEN }}"

        expose:
          - "5173"
        labels:
          traefik.enable: "true"
          traefik.docker.network: traefik_proxy
          traefik.http.routers.web.rule: "Host(`vps.joselp.com`)"
          traefik.http.routers.web.entrypoints: "web"
          traefik.http.routers.web.middlewares: "https-redirect"



            traefik.http.middlewares.https-redirect.redirectscheme.scheme:
"https"

          traefik.http.routers.websecure.rule: "Host(`vps.joselp.com`)"
          traefik.http.routers.websecure.entrypoints: "websecure"
          traefik.http.routers.websecure.tls: "true"
          traefik.http.routers.websecure.tls.certresolver: "letsencrypt"
          traefik.http.services.web.loadbalancer.server.port: "5173"
```

Traefik enruta cualquier petición por el puerto 80 al puerto 443 encriptado por https. Estas peticiones obtienen los datos del servicio Vite ejecutado en el puerto 5173.
Con todo configurado, solo falta recibir datos desde el módulo de invernadero haciendo uso de publicaciones en el topic de Kafka.