1. Chequeo de Fases Independientes:

- **Fase 1:** Se ha implementado un script con Ajax para acceder al archivo JSON local. Esto permite la carga de datos de manera asíncrona.
- **Fase 2:** Se ha creado un formulario de acceso al administrador.
- **Fase 3:** Se ha diseñado el documento con escudos, título y nombre para imprimir en PDF con datos locales.
- **Fase 4:** Se ha instalado y configurado un servidor Node para alojar datos del módulo con la gestión adecuada de CORS (Cross-Origin Resource Sharing).
- **Fase 5:** La aplicación se ha conectado con el servidor Node y se ha verificado su funcionamiento.

2. Ventajas y Desventajas de la Comunicación Asíncrona:

Ventajas:

- **Eficiencia:** Permite realizar múltiples tareas simultáneamente, evitando bloqueos en la interfaz de usuario.
- **Mejora la Experiencia del Usuario:** Al no bloquear la interfaz, los usuarios pueden interactuar sin interrupciones.
- **Menos Consumo de Recursos:** Al no esperar bloqueos, se evitan solicitudes innecesarias y se reduce el uso de recursos.
- **Actualizaciones Dinámicas:** Facilita la actualización de contenido en tiempo real sin recargar la página.

Desventajas:

- **Complejidad:** La gestión de asincronía puede complicar el código y aumentar la posibilidad de errores.
- **Manejo de Errores:** Identificar y manejar errores puede ser más complejo en comparación con operaciones síncronas.
- **Seguimiento de Estado:** Mantener el seguimiento del estado puede ser desafiante debido a la naturaleza no bloqueante.

3. Mecanismo de Comunicación Asíncrona:

La comunicación asíncrona es un enfoque donde las solicitudes y respuestas no esperan la finalización de una tarea antes de continuar con la siguiente. En el contexto web, se implementa comúnmente a través de tecnologías como Ajax y Promesas. Permite la ejecución de operaciones en segundo plano sin interrumpir el flujo principal de la aplicación.

4. Propiedades y Métodos de XMLHttpRequest:

- **Propiedades:**
    - **onreadystatechange:** Define una función que se llamará cuando cambie el estado de la solicitud.

    - **readyState:** Representa el estado de la solicitud (0: no inicializado, 1: conexión establecida, 2: solicitud recibida, 3: procesando, 4: completado).
    - **responseText:** Contiene la respuesta de la solicitud como texto.
    - **status:** Representa el estado HTTP de la solicitud (200: OK, 404: No encontrado, etc.).
- **Métodos:**
    - **open(method, url, async):** Inicia una solicitud (GET o POST) con el método especificado, la URL y la asincronía.
    - **send(data):** Envía la solicitud al servidor. Puede incluir datos en el caso de POST.
    - **setRequestHeader(header, value):** Agrega un encabezado HTTP a la solicitud.

5. Librerías Actuales para Actualización Dinámica:

- **Socket.io:** Para aplicaciones en tiempo real que permiten la comunicación bidireccional entre el cliente y el servidor.
- **Fetch API:** Proporciona una interfaz más moderna y flexible para realizar solicitudes HTTP de manera asíncrona.
- **Axios:** Una librería basada en Promesas para realizar solicitudes HTTP tanto desde el navegador como desde Node.js.