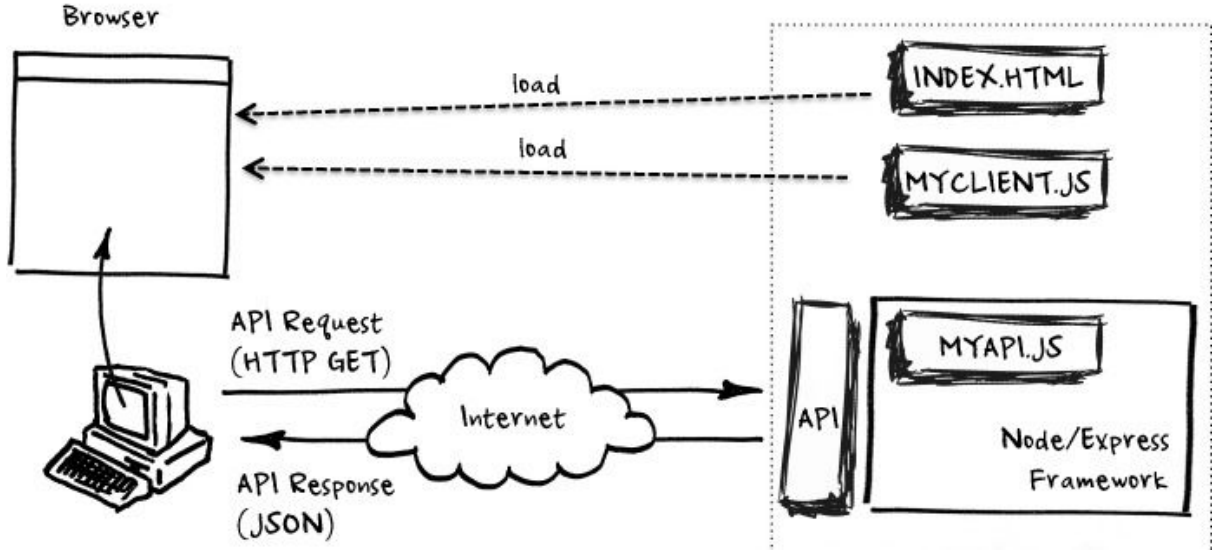
Consigna:

Proponer y documentar de qué manera puede darse la comunicación entre el sistema y los dispositivos (a alto nivel, sin detalles de implementación). Evaluar su impacto sobre el modelo de objetos.

Propuesta:
Se propone, para la comunicación entre el sistema y los dispositivos, una interfaz de comunicación utilizando el protocolo HTTP/HTTPs y JSON como formato de los datos a transferir.



La idea es habilitar una ruta en el sistema que funcione como punto de entrada de la comunicación con los dispositivos, funcionando así una API utilizando los métodos HTTP (GET, POST, PUT, DELETE) para la manipulación de los mensajes.

Por ejemplo:
Si habilitamos la ruta ficticia https://sistema-de-energia.com.ar/sensores los sensores pueden enviar, mediante una operación POST a esta ruta, información de una medición. Ésta se enviará en formato json en el cuerpo de la operación:

```
{
    "id_sensor": 100,
    "id_dispositivo": 101,
    "medicion": {
        "magnitud": "temperatura",
        "valor": 26
    }
}
```

Análogamente, se espera que los dispositivos dispongan una ruta en protocolo HTTP para que el sistema pueda enviarles mensajes como ser acciones a realizar. De esta forma el sistema puede lanzar ejecución de acciones en los dispositivos.

Por ejemplo:
Se tiene un dispositivo que provee la ruta https://aire-acondicionado-1.compania.com.ar y al cual se le puede pedir su estatus mediante una operación GET y obtenerlo como un mensaje json:

```
{
    "id": 100,
    "estatus": "prendido"
}
```

Y permite enviar una acción a realizar mediante una operación POST con los detalles en el cuerpo del mensaje como json:

```
{
    "operacion": "apagar"
}
```

Para usar efectivamente este mecanismo propuesto es necesario definir formas de securizar la comunicación como así también autenticar y autorizar la comunicación entre las partes: se puede implementar HTTPS para la comunicación y utilizar identificaciones previamente definidas para validar el acceso al sistema (conocidas como “api keys”) o implementar un sistema más complejo de login y tokens (como ser OAuth)

Impacto sobre el modelo de objetos:

Esta propuesta nos llevaría a crecer el modelo de objetos con nuevas clases cuya responsabilidad sea realizar las operaciones HTTP como así también la codificación y decodificación de los distintos formatos de los datos de los mensajes (JSON por ejemplo). A su vez, mientras que la comunicación entrante que el sistema disponga tendría un esquema de datos único y documentado por nosotros, es esperable que distintos fabricantes utilicen mecanismos distintos y/o implementen formatos o esquemas diferentes para sus mensajes con sus dispositivos y sensores. Por ejemplo:

- Enviar un id de la operación a realizar en vez de un string en un json
- Utilizar operación GET o PUT en vez de POST para ejecutar una acción
- Requerir parámetros extra en la url a utilizar (“query string”)
- Entender únicamente protocolo SOAP o formato XML

Para poder dar soporte a cada particularidad de cada fabricante se hace necesario implementar clases que adapten el sistema para poder enviar y recibir mensajes de y hacia los distintos dispositivos de fabricantes utilizando el patrón Adapter.

Algunas características y ventajas de las API Rest usado como base para esta propuesta (tomado de: https://bbvaopen4u.com/es/actualidad/api-rest-que-es-y-cuales-son-sus-ventajas-en-el-desarrollo-de-proyectos):

- **Protocolo cliente/servidor sin estado:** cada petición HTTP contiene toda la información necesaria para ejecutarla, lo que permite que ni cliente ni servidor necesiten recordar ningún estado previo para satisfacerla. Aunque esto es así, algunas aplicaciones HTTP incorporan memoria caché. Se configura lo que se conoce como protocolo cliente-caché-servidor sin estado: existe la posibilidad de definir algunas respuestas a peticiones HTTP concretas como cacheables, con el objetivo de que el cliente pueda ejecutar en un futuro la misma respuesta para peticiones idénticas. De todas formas, que exista la posibilidad no significa que sea lo más recomendable.

- Las operaciones más importantes relacionadas con los datos en cualquier sistema REST y la especificación HTTP son cuatro: **POST** (crear), **GET** (leer y consultar), **PUT** (editar) y **DELETE** (eliminar).

- **Los objetos en REST siempre se manipulan a partir de la URI:** Es la URI y ningún otro elemento el identificador único de cada recurso de ese sistema REST. La URI nos facilita acceder a la información para su modificación o borrado, o, por ejemplo, para compartir su ubicación exacta con terceros.

- **Separación entre el cliente y el servidor:** el protocolo REST separa totalmente la interfaz de usuario del servidor y el almacenamiento de datos. Eso tiene algunas ventajas cuando se hacen desarrollos. Por ejemplo, mejora la portabilidad de la
- interfaz a otro tipo de plataformas, aumenta la escalabilidad de los proyectos y permite que los distintos componentes de los desarrollos se puedan evolucionar de forma independiente.

- **Visibilidad, fiabilidad y escalabilidad:** La separación entre cliente y servidor tiene una ventaja evidente y es que cualquier equipo de desarrollo puede escalar el producto sin excesivos problemas. Se puede migrar a otros servidores o realizar todo tipo de cambios en la base de datos, siempre y cuando los datos de cada una de las peticiones se envíen de forma correcta. Esta separación facilita tener en servidores distintos el front y el back y eso convierte a las aplicaciones en productos más flexibles a la hora de trabajar.

- **La API REST siempre es independiente del tipo de plataformas o lenguajes:** la API REST siempre se adapta al tipo de sintaxis o plataformas con las que se estén trabajando, lo que ofrece una gran libertad a la hora de cambiar o probar nuevos entornos dentro del desarrollo. Con una API REST se pueden tener servidores PHP, Java, Python o Node.js. Lo único que es indispensable es que las respuestas a las peticiones se hagan siempre en el lenguaje de intercambio de información usado, normalmente XML o JSON.