**Estructura de Datos**

1. **Colecciones**
   1. Servers

* serverId (string): Identificador único para cada servidor. [Indexado]
* name (string): Nombre descriptivo del servidor.
* endpoint (string): URL o dirección IP del servidor. (¿Se podría usar para gestionar las bajas y altas de servidores?)
  1. Devices
* serial (string): Número de serie del dispositivo; será el identificador único. [Indexado]
* apikey (string): Clave API asociada a cada dispositivo. [Indexado]
* lastCommunication (date): Fecha y hora del último mensaje recibido.
* serverId (string): Identificador del servidor al que está conectado el dispositivo. Relación con la colección Servers.
  1. Messages
* serial (string): Número de serie del dispositivo que envió el mensaje. Relación con la colección Devices. [Indexado]
* timestamp (date): Fecha y hora en que se recibió el mensaje. [Indexado]
* messageType (string): Tipo de mensaje. [Confirmar si hay que indexarlo]
* content (json object): Contenido del mensaje.
  1. User
* name (string): cadena de texto asociada.
  1. Password
* text (string): cadena de texto asociada.

1. **Implementación de Consultas y Relaciones**

De cara a las consultas he encontrado que para nuestro caso lo mejor sería combinar consultas con filtro con la inclusión de recursos relacionados. Las consultas con filtros se hacen directamente en las URL de json-server mediante campos clave. Los recursos relacionados usan el parámetro \_embed para incluir relaciones en las búsquedas, ej: obtener un dispositivo y todos sus mensajes:

*/devices?serial=abc123&\_embed=messages.*

Indexar campos es para agilizar las búsquedas de datos concretos al usar filtros, ya que son campos que serán accedidos con frecuencia.

Por último, en cuanto al tiempo que mantendremos la información guardada había pensado entre una y dos semanas para no sobrecargar json-server.