VIDIC

(Visualization of Industrial Data In the Cloud)

Actores (objetos o elementos que pueden aparecer):

- Sistemas software:
  - Broker (servidor) MQTT
  - Broker WAMP
  - Base de datos históricos
  - Servidor datos tiempo real Websocket
  - Servidor datos históricos REST (JSON)
  - Aplicación web (dashboard)
- Reales:
  - Cliente final (ej: IQD)
  - Propiedad (“cliente del cliente final”; ej: Bodegas Reymos)
  - Instalación (ubicación física con uno o varios dispositivos)
  - Dispositivo (emisor de datos MQTT: autómata, gateway)

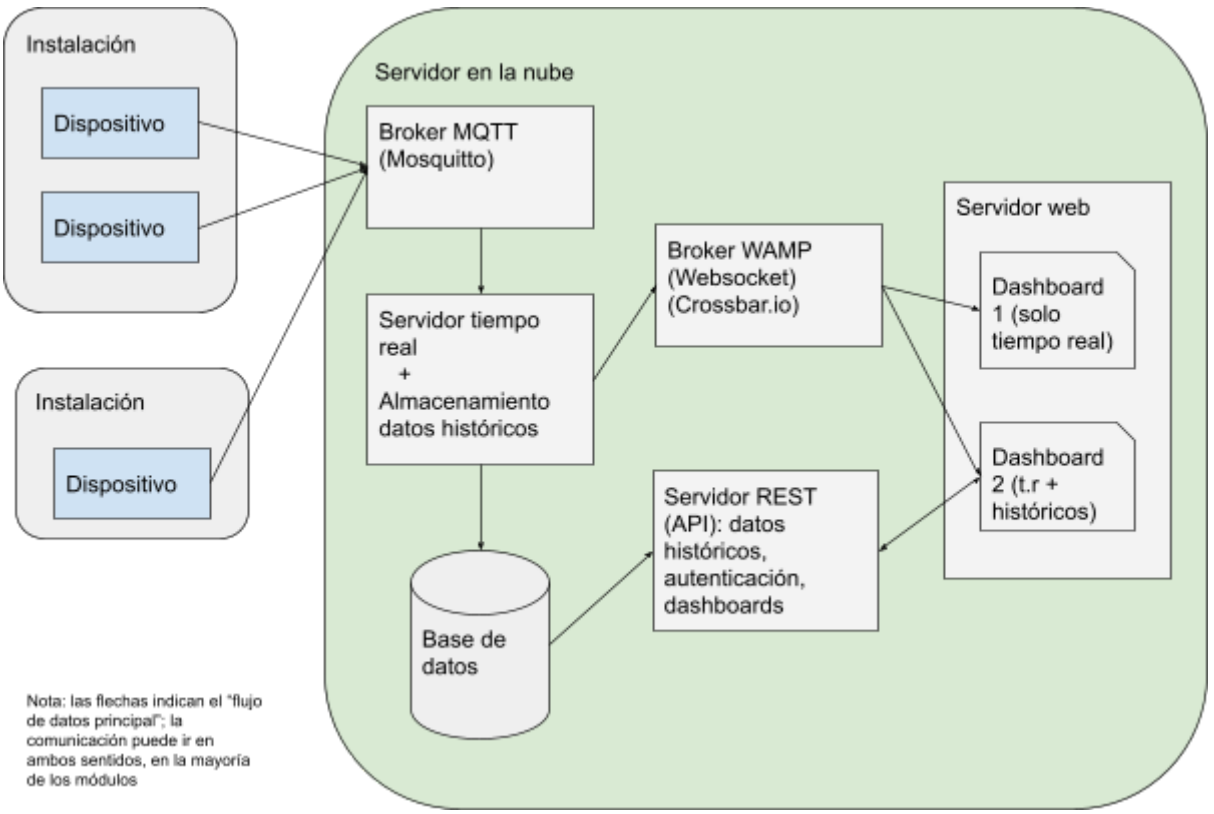
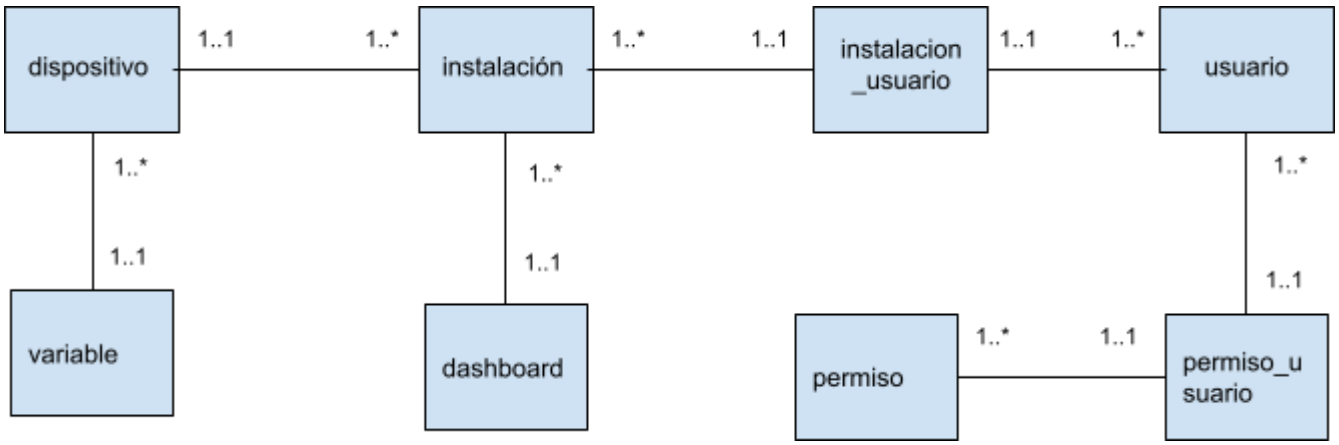


Diagrama UML de la base de datos relacional:



(Nota: los indicadores de cardinalidad de las relaciones se leen “saliendo” de la tabla. Por ejemplo, un dispositivo puede estar relacionado con una instalación; y una instalación puede estar relacionada con muchos dispositivos).

Envío de datos:

Definir protocolo “payload” MQTT.

- Cómo se identifica el dispositivo: El dispositivo se identificará en la conexión MQTT con un usuario y contraseña únicos (al menos el usuario). Este usuario deberá ser exactamente el mismo que el usado en su certificado de cliente para conexiones TLS. El usuario estará en formato: “id_cliente”. De esta forma obtenemos el identificador del dispositivo. Por ejemplo: usuario= “ReymosIQD” contraseña= *******
- Formato de los datos: Los datos se enviarán a través de MQTT, del dispositivo al broker, utilizando el módulo MessagePack, y se desempaquetarán en el broker usando la librería “msgpack”. Posteriormente, los datos se transferirán a formato JSON y se trabajará con ese formato en el resto de módulos de la aplicación.
- El contenido enviado en cada mensaje MQTT consiste en:
    - id_dispositivo
    - map {id_variable:valor}

Recepción de datos

- Se usa un broker MQTT independiente, Mosquitto. Recibirá los mensajes MQTT de los dispositivos, y los otros módulos del servidor (envío datos en tiempo real, almacenamiento de datos históricos) se suscribirán para procesar los mensajes recibidos.

Almacenamiento de datos

- Puede integrarse con el módulo de recepción de datos.

- Una tabla por dispositivo, con tantas columnas como variables tenga el dispositivo (más una de timestamp)

Servidor Websocket tiempo real

- Usaremos el protocolo WAMP (protocolo de publicar/suscribir) sobre conexiones Websocket. Habrá un broker “Crossbar.io” al que se harán las peticiones; los clientes pueden usar la librería “Autobahn”, que tiene versiones Python y Javascript.
- Los “topic” en WAMP son jerárquicos; lo aprovecharemos para reunir las variables de un dispositivo como “subtopics” del “topic” de ese dispositivo. Opcional: los dispositivos pueden ser a su vez “subtopic” de las instalaciones.
- El “payload” de los mensajes irá en formato JSON; el contenido puede ser el mismo que para los mensajes MQTT, solo que en JSON. El mensaje con un conjunto de valores de un dispositivo puede ser, por ejemplo: {X:11, Y:0.22, Z:false} (X,Y,Z=id de las variables).
- Las conexiones Websocket deberían ser seguras (TLS). Aunque mientras estén en la misma máquina quizás no sea algo "crítico".

Servidor REST históricos

- Implementará una API para que la aplicación web pueda consultar los valores de variables en rangos de tiempo.
- También implementará funciones para comprobar usuario+contraseña, y para devolver los permisos del usuario.
- Al igual que el servidor Websocket, debería usar conexiones seguras.

Aplicación web

- Debería ser un servidor seguro (HTTPS), por lo que necesita un certificado digital
- La identificación del usuario se hará por nombre de usuario y contraseña (guardados en la BD)
- La aplicación web mostrará al usuario una lista de sus dashboard, si tiene permiso para ver más de uno; si solo puede ver uno, entrará directamente en su dashboard. (Si no tiene permiso para ver ninguno, se mostrará un error.)
- Por tanto, la aplicación estará compuesta por varias páginas: formulario login, lista de dashboard del usuario, y los dashboard que se hayan creado. La página inicial será la de login, y las demás no se podrán abrir si el usuario no se identifica.
- La aplicación debe mantener al usuario identificado mientras dure la sesión, aunque cambie de página (si tiene más de un dashboard, debe poder salir del que está viendo y elegir otro). Podemos dar la opción de “Cerrar sesión”, para que el usuario pueda “desidentificarse”; esta opción debería estar en los dashboard, y en la lista de dashboard.

Ideas a tener en cuenta:

- Realizar los casos de prueba de los usuarios de la aplicación: identificarse, elegir un dashboard entre los que puede ver, poder salir del dashboard y elegir otro, cerrar sesión, también poder seleccionar el rango de horas para los datos históricos de la gráfica.
- Definimos una tabla “permiso” con los tipos de permiso que se pueden asignar; por ejemplo, (id=1, nombre=’iniciar sesión’). Tendrá una columna "tabla_relacionada" para indicar si el tipo de permiso se concede para otra tabla; por ejemplo (id=2, nombre=’ver dashboard’, tabla_relacionada=’dashboard’). Para asignar los permisos a los usuarios, lo haremos con una tabla intermedia “usuario_permiso” que relacione un usuario con un permiso; y con una columna “id_relacionado” si el permiso está asociado a otra tabla. Por ejemplo (cliente_id=1, permiso_id=2, id_relacionado=7) para que el cliente con id=1 pueda ver el dashboard con id=7. Así dejamos abierta la posibilidad de definir otro tipo de permisos relacionados con una tabla distinta, por ejemplo, (id=3, nombre="accionar dispositivo", tabla_relacionada=”dispositivo”).