SAA - Sistema de Ahorro de Agua

# Introducción

Según datos de la OMS una ducha de 10 minutos consume 200 litros de agua cuando las recomendaciones indican que se deberían gastar 150% menos de agua por ducha. Hoy en día el agua es un recurso que hay que preservar por lo que hemos querido proponer un sistema inteligente capaz de realizar ahorros de agua. Efectivamente, la OMS recomienda utilizar menos agua durante los tiempo de ducha ya que esta agua es directamente evacuada al desagua, sin embargo este sistema permitiría reutilizar esta agua de ducha en diferentes puntos del hogar para realizar ahorros de agua. Este sistema analiza la calidad del agua que sale de la ducha, si el agua esta muy sucia no se reutiliza para no crear problemas de insalubridad, almacena esta agua en unos depósitos y la trata con regularidad para que no haya desarrollo de bacterias o algas y cuando se necesite en otros puntos de la vivienda en los cuales no se necesite agua potable se llevara hasta esos puntos.

El producto va destinado a cualquier persona que quiera realizar ahorros de agua y de esta manera proteger el medio ambiente. El proyecto esta pensado para instalarlo en una ducha o bañera aunque también se podría extender a otros puntos del hogar como por ejemplo al fregadero.

En un primer lugar, estudiaremos el funcionamiento detallado del sistema. Después veremos la base de datos diseñada para poder almacenar toda la información necesaria para un buen funcionamiento de la aplicación. Tras esto explicaremos los métodos de la API rest desplegados que necesitaremos. También será interesante ver el MQTT. Finalmente estudiaremos el funcionamiento del flujo cliente-servidor.

# Funcionamiento

Nos vamos a centrar en un primer momento en ahorrar la máxima cantidad de agua que se consume en el proceso de duchado de una persona. Para describir el funcionamiento de nuestro sistema tenemos que diferenciar entre las diferentes de una ducha promedio suponiendo que una persona se ducha con agua caliente normalmente:

Fase 1: El agua de la ducha está fría, y el usuario se encuentra fuera de la ducha esperando a que se caliente. Este agua cae al desagüe prácticamente limpia por lo que este agua es reutilizable.

Fase 2: El agua de la ducha ya se ha calentado y el usuario se encuentra dentro de la ducha. En esta fase el agua que cae por el desagüe no es reutilizable.

Fase 3: El agua de la ducha está caliente y el agua que cae por el desagüe es agua con jabón. En este caso podríamos reutilizar el agua pero tendría que pasar por algún tipo de filtro para eliminar el excesivo jabón.

Fase 4: El agua de la ducha está caliente pero ya el agua que cae por el desagüe o tiene jabón y es un agua muy parecida a la de la primera fase. Se puede reutilizar.

Una vez hemos diferenciado las diferentes fases de la ducha podemos ver como ahorrar el agua que se consume en cada una de estas fases. Para ello necesitaremos un sistema para diferenciar el agua de cada fase de la ducha para poder reutilizarla de la mejor manera. Dicho sistema será un sensor de turbidez del agua el cual nos indicará si ese agua se puede reutilizar o no, y en el caso de que se pueda reutilizar, en qué otras actividades se podría emplear. Para ello tenemos un depósito de agua servirá únicamente para rellenar el agua de la cisterna o para el agua de regado.. Para desviar el agua a un depósito o al desagüe en caso de que el agua no se pueda reutilizar (como el agua de la fase 3), utilizaremos un servomotor el cual se encargará de mover el agua de un sitio a otro.  Una vez el agua en el depósito el sistema la redistribuirá hacia los otros puntos de agua de la casa cuando se necesite.

# Organización

### Marzo

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Lunes | Martes | Miércoles | Jueves | Viernes | Sábado | Domingo |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 8 | 9 | 10 | 11 | 12 | 13  Poner en común las ideas | 14 |
| 15 | 16 | 17  Concretar una idea de proyecto | 18  Reunión de clase – Idea definitiva | 19 | 20  Desarrollo del modelo UML | 21  Definir los métodos necesarios |
| 22 | 23 | 24 | 25 | 26 | 27 | 28 |
| 29 | 30 | 31 |  |  |  |  |

### Abril

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Lunes | Martes | Miércoles | Jueves | Viernes | Sábado | Domingo |
|  |  |  | 1 | 2 | 3 | 4 |
| 5 | 6 | 7 | 8  Primera versión de la base de datos | 9  Primera versión de los métodos necesarios | 10 | 11 |
| 12 | 13 | 14 | 15 | 16 | 17 | 18 |
| 19 | 20 | 21 | 22 | 23 | 24 | 25 |
| 26 | 27 | 28 | 29 | 30 |  |  |

### Mayo

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Lunes | Martes | Miércoles | Jueves | Viernes | Sábado | Domingo |
|  |  |  |  |  | 1 | 2 |
| 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 10 | 11 | 12 | 13  Feed-back primer entregable | 14 | 15 | 16 |
| 17 | 18  Versión definitiva bdd | 19  Versión definitiva métodos | 20 | 21 | 22 | 23 |
| 24 | 25 | 26 | 27 | 28 | 29 | 30 |
| 31 |  |  |  |  |  |  |

### Junio

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Lunes | Martes | Miércoles | Jueves | Viernes | Sábado | Domingo |
|  | 1 | 2 | 3 | 4 | 5 | 6 |
| 7 | 8 | 9 | 10 | 11 | 12 | 13 |
| 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| 21 | 22 | 23 | 24 | 25 | 26 | 27 |
| 28 | 29 | 30 |  |  |  |  |

## Idea de partida

En primer lugar, la idea inicial del proyecto era desarrollar un dispensador de comida y de agua para mascotas. Sin embargo al ver que otro grupo había tenido la misma idea decidimos cambiar la base de nuestro proyecto. Por lo tanto, decidimos que íbamos a desarrollar un sistema de ahorro de agua para duchas ya que esto permitiría reducir el impacto medioambiental de cada vivienda y además ayudaría a sus usuarios realizar ahorros económicos.

## Desarrollo de la base de datos y de los métodos

En un primer tiempo realizamos una base de datos bastante sencilla ya que solo disponía de 2 clases y sin ningún tipo de relación y también desarrollamos los métodos asociados a ello. Sin embargo esta base de datos era demasiado sencilla y no permitía almacenar todos los datos necesarios y complicaba mucho el almacenamiento de información. De este modo, tras el primer entregable decidimos volver a diseñar la base de datos para que fuese mas completa, efectivamente la versión definitiva de la base de datos permite almacenar información sobre los usuarios, los dispositivos de los cuales disponen y recuperar toda la información acerca de los sensores y actuadores. Tras el desarrollo de esta base de datos tuvimos que volver a definir los métodos asociados.

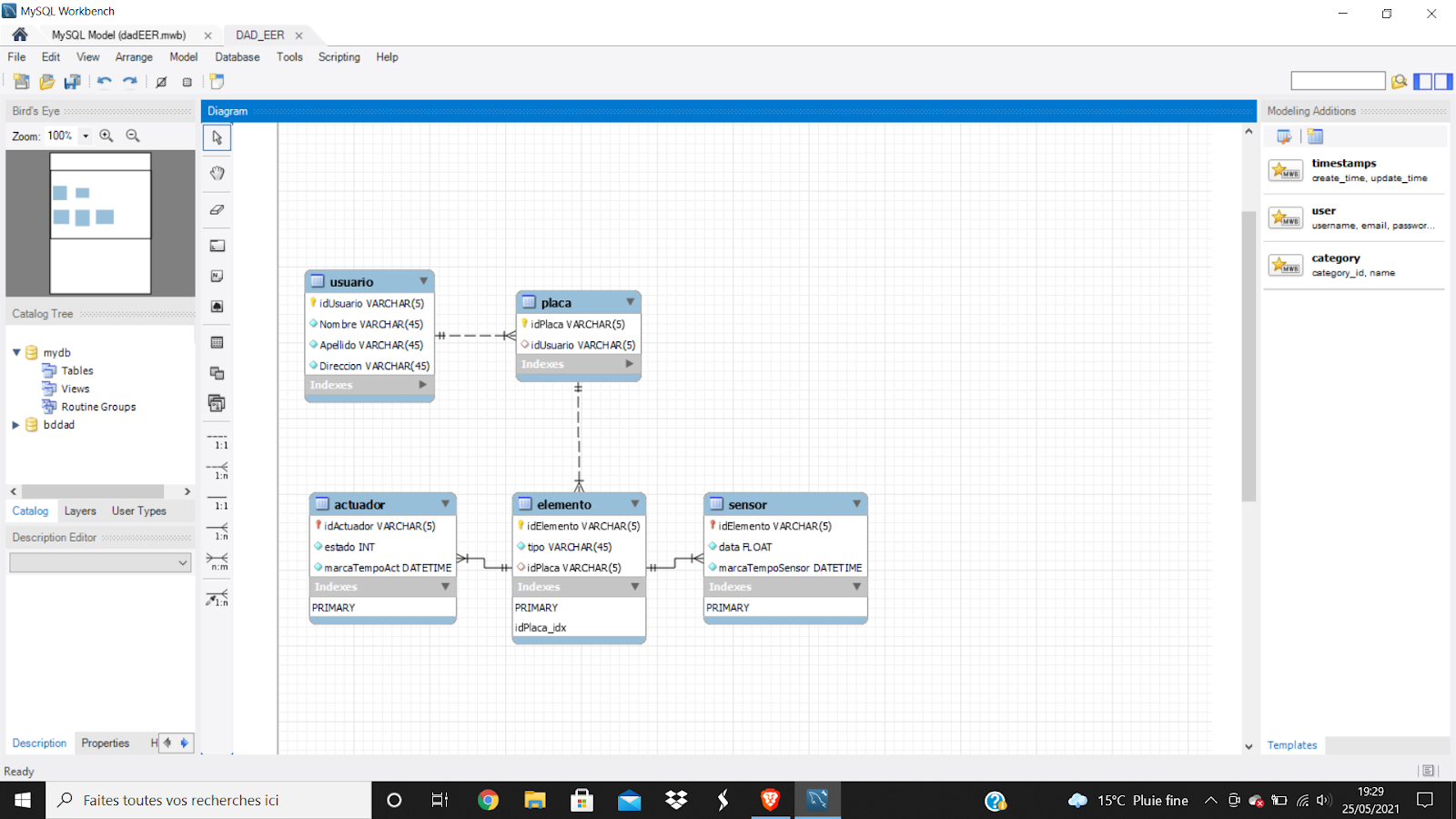
# Base de datos

La base de datos diseñada para este proyecto se compone de 5 clases, usuario, placa, elemento, actuador y sensor. Las dos primeras permiten almacenar todos los datos que puedan ser útiles para la gestión de usuario ya que la clase usuario almacena las coordenadas de los usuarios y la clase placa los datos sobre un microcontrolador. De esta manera al acceder a un usuario se puede acceder a los microcontroladores de los cuales dispone ya que un usuario puede haber instalado este sistemas varias veces.

Por otra parte, la clase elemento, está conectada a la clase placa, ya que un sistema dispone de varios elementos (sensores y actuadores). Los elementos disponen de un campo “tipo” que permite indicar si es un sensor o un actuador y saber su función. Esta clase tiene dos clases hijas que permiten almacenar los datos medidos por un sensor o el estado de un actuador, así como la hora à la cual se ha medido este dato.

La base de datos permite almacenar los datos de los usuario así como los dispositivos de los cuales dispone. Este diseño permite acceder a varias medidas tanto de los sensores como de los actuadores pero poder borrar fácilmente esta lista de datos, gracias al campo de marcas temporales. Efectivamente, se pueden borrar estos datos cuando la marca temporal sea muy antigua o cada vez que el sistema se reinicia, ya que los datos medidos pierden su utilidad al cabo de un tiempo. Sin embargo, se pueden borrar fácilmente estos datos sin perder la lista de los elementos conectados a los actuadores.

Por último, todas las relaciones entre clases son de tipo “ON CASCADE” lo que permite que todas las modificaciones en una clase se repercuten en las otras sin tener que acceder a cada clase para modificar cada datos modificados o eliminados.



# API Rest

## Métodos GET

**Llamada a actuadores y sensores :**

* **Sensor por tipo**

El microcontrolador sabe los elementos conectados a el y hace una llamada a la base de datos pasando el tipo de sensor que quiere y recupera todas las medidas realizadas por todos los sensores de dicho tipo conectados a este sistema.

Llamada API :

api/sensor/tipo=?

Parámetros :

tipo : parámetro obligatorio. Se le debe pasara este parámetro para saber de que a tipo de sensor se le pide los datos medidos.

Este método es útil para ir informando al usuario de lo que se ha podido medir durante sus duchas.

* **Actuador por tipo**

El microcontrolador sabe los elementos conectados a el y hace una llamada a la base de datos pasando el tipo de actuador que quiere y recupera todas las actuaciones realizadas por todos los actuadores de dicho tipo conectados a este sistema.

Llamada API :

api/actuador/tipo=?

Parámetros :

tipo : parámetro obligatorio. Se le debe pasara este parámetro para saber de que a tipo de actuador se le pide las actuaciones.

Este método es útil para ir informando al usuario de lo que se ha ido produciendo durante sus duchas.

* **Ultimo dato por tipo de actuador**

El microcontrolador sabe los elementos conectados a el y hace una llamada a la base de datos pasando el tipo de sensor que quiere y recupera solamente la medida mas reciente realizada por dicho tipo de sensor.

Llamada API :

api/lastSensor/tipo=?

Parámetros :

tipo : parámetro obligatorio. Se le debe pasara este parámetro para saber de que a tipo de sensor se le pide los datos medidos.

Este método es útil para el sistema en si ya que para actuar solo necesita analizar el dato medido más reciente.

* **Ultimo dato por tipo de sensor**

El microcontrolador sabe los elementos conectados a el y hace una llamada a la base de datos pasando el tipo de actuador que quiere y recupera solamente el último dato enviado por dicho tipo de actuador.

Llamada API :

api/lastActuador/tipo=?

Parámetros :

tipo : parámetro obligatorio. Se le debe pasara este parámetro para saber de que a tipo de actuador se le pide las actuaciones.

Este método es útil para el sistema en si ya que para actuar solo necesita analizar el último estado de un actuador.

## Métodos POST

**Añadir línea a la base de datos :**

* **Usuario**

Es necesario el método para añadir un usuario, si alguien se quiere registrar.

Llamada API :

api/usuario

Parámetros :

No se necesitan parámetros en la ruta ya que no se accede a ninguna línea sino que se añade una. Sin embargo para definir el método se necesita pasar como parámetros los valores que se van a añadir a la tabla.

* **Placa**

Es necesario el método para añadir una placa, si un usuario quiere comprar un dispositivo.

Llamada API :

api/placa

Parámetros :

No se necesitan parámetros en la ruta ya que no se accede a ninguna línea sino que se añade una. Sin embargo para definir el método se necesita pasar como parámetros los valores que se van a añadir a la tabla.

* **Elemento**

Es necesario el método para añadir un elemento, cuando hay que añadir un elemento al sistema.

Llamada API :

api/elemento

Parámetros :

No se necesitan parámetros en la ruta ya que no se accede a ninguna línea sino que se añade una. Sin embargo para definir el método se necesita pasar como parámetros los valores que se van a añadir a la tabla.

* **Sensor**

Es necesario el método para añadir una medida cuando el sistema lo necesite.

Llamada API :

api/sensor

Parámetros :

No se necesitan parámetros en la ruta ya que no se accede a ninguna línea sino que se añade una. Sin embargo para definir el método se necesita pasar como parámetros los valores que se van a añadir a la tabla.

* **actuador**

Es necesario el método para añadir la información sobre el estado de un actuador cuando el sistema lo necesite.

Llamada API :

api/elemento

Parámetros :

No se necesitan parámetros en la ruta ya que no se accede a ninguna línea sino que se añade una. Sin embargo para definir el método se necesita pasar como parámetros los valores que se van a añadir a la tabla.

## Métodos DELETE

**Eliminar línea a la base de datos :**

Hemos decidido desarrollar un método delete para las clases de usuarios, de sistemas y de elementos.

* **Usuario**

Es necesario el método para añadir un usuario, si alguien se quiere registrar.

Llamada API :

api/usuario/idUsuario=?

Parámetros :

idUsuario : Parametro obligatorio. Se necesita pasarle el id de la línea que se quiere eliminar ya que es la clave primaria de cada clase.

* **Placa**

Es necesario el método para añadir una placa, si un usuario quiere comprar un dispositivo.

Llamada API :

api/placa/idPlaca=?

Parámetros :

idPlaca : Parametro obligatorio. Se necesita pasarle el id de la línea que se quiere eliminar ya que es la clave primaria de cada clase.

* **Elemento**

Es necesario el método para añadir un elemento, cuando hay que añadir un elemento al sistema.

Llamada API :

api/elemento/idElemento=?

Parámetros :

idElemento : Parametro obligatorio. Se necesita pasarle el id de la línea que se quiere eliminar ya que es la clave primaria de cada clase.

# ESP32

El código para el sigue una estructura en la que la mayoría de los casos solo debemos adaptarlos a nuestro proyecto :

* **Setup** : Iniciamos la conexión WIFI e inicializamos el cliente MQTT.
* **Loop** : si el cliente no está conectado nos reconectamos. En un primer momento, el usuario elige si quiere agua fría o agua caliente para ducharse, en el caso de que quiera agua caliente la variable global “ducha” estará à 1 y en el caso contrario valdrá -1. Tras la elección, se establece la conexión, y se repite indefinidamente el ciclo del sistema de control del sistema de direccionamiento del agua. À partir de ese momento, se verifican varias condiciones para saber si el agua es recuperable o no. Efectivamente, si el usuario ha elegido la opción de ducharse con agua caliente, si la temperatura del agua está por debajo de la temperatura de ducha se enviará la señal HIGH por el pin 4 para así activar la válvula que permitirá enviar el agua hacia los depósitos de agua reutilizable, en el caso contrario la salida del pin 4 será LOW por lo que se enviará esta agua hacia otra tubería. Por último, se verificará que los datos enviados por el sensor de turbidez, ya que si esta à 0 se enviará la señal HIGH por el pin 6 para que el agua sea enviada hacia los depósitos de agua reutilizable, en el caso en que el sensor envíe 1, es que el agua estara sucia y no se podrá reutilizar y se tirara esta agua.
* Funciones MQTT