MEMORIA SOBRE PROYECTO “SMART BLINDS”

JUAN ANTONIO VALENZUELA GARCÍA

PABLO BARRERA DAZA

INDICE:

1. INTRODUCCION

Nuestro proyecto se llama “***Smart Blinds***”, o también traducido al español, persianas inteligentes.

Esto, es debido a que consta de un sistema, el cuál comentaremos próximamente, que hace que la persiana se cierre o se abra de forma autónoma.

La idea principal es la de crear un sistema de apertura y cierre automático de las persianas según la temperatura que haga en el entorno, para hacer del hogar un entorno más cómodo.

La propuesta del proyecto nos pareció una muy buena idea para implementar en el proyecto de la asignatura. Además de que esta

propuesta tenia unos fines u objetivos muy interesantes, en los que queremos hacer hincapié en el siguiente apartado.

2. OBJETIVOS

Algunos de los más esenciales que destacan entre los demás son, por ejemplo, la optimización del comfort térmico, que garantiza un entorno interior cómodo sin necesidad de intervención manual, o también por ejemplo, la eficiencia energética minimizando el uso de calefacción o aire acondicionado.

Otros objetivos importantes son la sostenibilidad, contribuyendo a la reducción de la huella de carbono, o la adaptabilidad y escalabilidad para implementarlos desde hogares, hasta oficinas y espacios comerciales, además de la seguridad y protección que supone evitar el exceso de calor tanto a las personas, como a los elementos sensibles a la luz solar directa de dentro del hogar.

Todos los objetivos mencionados anteriormente reflejan las motivaciones clave detrás del proyecto, centradas en la comodidad, eficiencia, innovación, sostenibilidad y seguridad, con el propósito de ofrecer una solución tecnológica avanzada que mejore la calidad de vida de los usuarios.

3. REALIZACION DEL PROYECTO

3.1 IMPLEMENTACION SERVLETS

También hay que resaltar muy brevemente el funcionamiento básico del sistema del proyecto, para después explicar muy detenidamente, paso por paso, la realización del mismo.

Principalmente, se usa un servomotor que ajusta el cierre y apertura de la persiana según indiquen los parámetros de temperatura de los sensores utilizados. Los cuáles, deberán estar situados uno relativamente cerca de la persiana y otro más alejado.

Cuando ambos sensores sobrepasen una temperatura indicada previamente, la persiana activará el mecanismo de cierre. En caso contrario, permanecerá abierta.

Una vez aclarado el funcionamiento principal, podemos dar paso la explicación de la realización del proyecto.

Para comenzar, hemos hecho el proyecto con los **Servlets**. El servlet de nuestro proyecto maneja diferentes operaciones para crear, leer y eliminar objetos Sensor utilizando las solicitudes HTTP apropiadas (GET, POST, DELETE) y utiliza JSON como formato de datos para la comunicación entre el cliente y el servidor.

También hay que mencionar, que para la ejecución y despliegue de los Servlets mencionados, se ha utilizado el servidor web conocido como Apache Tomcat, que se encarga de gestionar el ciclo de vida de los servlets, desde su inicialización (init) hasta el manejo de las solicitudes (doGet, doPost, doDelete) y su destrucción.

Un breve resumen de la funcionalidad de las distintas solicitudes seria:

GET: Un cliente envía una solicitud GET con el parámetro sensorId. El servlet responde con los detalles del Sensor en formato JSON.

POST: Un cliente envía una solicitud POST con un objeto Sensor en formato JSON en el cuerpo. El servlet añade o actualiza el sensor y responde con el objeto Sensor en JSON.

DELETE: Un cliente envía una solicitud DELETE con un objeto Sensor en formato JSON en el cuerpo. El servlet elimina el sensor correspondiente y responde con el objeto Sensor eliminado en JSON.

3.2 VERTX

El siguiente paso en la realización del proyecto es la implementación con **Vertx**.

Para comenzar hemos creado las entidades de los sensores y del actuador como clases de Java en Eclipse, cuyos atributos son idActuador, idPlaca, estado, grados,timeStamp e idGroup por parte del actuador.

Y idSensor, idPlaca, timeStamp, temperatura e idGroup por parte del sensor.

Una vez hecho esto, creamos la clase “RestServer”, que es una implementación de un servidor RESTful usando el framework Vertx, que gestiona las operaciones GET, POST,PUT, DELETE para las entidades: SensorEntity y ActuadorEntity.

Como resumen de esta fase, básicamente, creamos en esta clase métodos que hagan las operaciones mencionadas para cada una de las entidades y también, métodos que generen datos sintéticos para cada una de ellas, como el createSomeData.

También hemos definidos rutas en esta clase que manejan las solicitudes HTTP que permiten hacer las operaciones correspondientes.

Por último, creamos las clases ActuadorEntityListWrapper y SensorEntityListWrapper que almacenan la información en una lista.

Ahora hay que hacer la transformación de los repositorios en memoria, a una BBDD que se verá en el siguiente apartado.

3.3. CONEXIÓN DE BBDD

Lo siguiente es hacer la conexión de una **BBDD** mediante **HeidiSQL**, donde creamos tablas para cada una de las entidades con sus correspondientes atributos.

Una vez ya creada las tablas en SQL, podemos ir haciendo pruebas y ver como mediante los métodos adecuados, las tablas van añadiendo o eliminando objetos.

Esto permite gestionar la información necesaria en la API Rest.

3.4 CODIGO FIRMWARE

También es necesario modificar el código **firmware** en Python, según los parámetros que usemos en nuestro proyecto, los cuales ya han sido descritos anteriormente.

Esto permite realizar peticiones a los endpoints y obtener los datos procedentes de los sensores de temperatura y su incersión mediante POST en la BBDD.

El código firmware se encarga de leer los datos de los sensores y envía comando al actuador para controlarlo, ya que el actuador responde a los datos cerrando o abriendo la persiana con el servomotor.

Para ello, primero se conecta a una red WiFi asignando los parámetros correspondientes. Después inicializa los sensores y el servomotor conectados a pines específicos, y se asigna la configuración inicial del cliente MQTT para conectarse a un bróker en la dirección IP y puertos especificados, como Mosquitto que es un intermediario entre el servidor y el dispositivo.

También se define su callback para manejar los mensajes recibidos, controlando el servomotor según ON/OFF.

Después disponemos de una función loop() que realiza la lectura de los sensores DHT11 y envía los datos de las temperaturas leídas al servidor HTTP mediante solicitudes POST.

En resumen, el código permite monitorear la temperatura del entorno en tiempo real utilizando sensores DHT11 y controlar un servomotor basado en estos valores. La comunicación se realiza a través de MQTT para la interacción en tiempo real y HTTP REST para registrar los datos en un servidor.

3.5 MQTT

Para finalizar la realización del proyecto, finalizamos con la implementación de MQTT.

Para ello, primero tenemos la clase “MainVerticle”, están las implementaciones utilizadas en el “RestServer” las cuales también lanzamos junto la configuración de MQTT.

Pero además de lo mencionado, se han creado configuraciones de rutas HTTP, y manejo de solicitudes HTTP y de excepciones para tratar posibles errores.

También hemos incluido métodos que además de hacer las operaciones básicas, puedan también obtener el ultimo sensor por su ID o grupo, por ejemplo.

Hay que mencionar además a MQTT Explorer, que es una herramienta que hemos usado para visualizar y monitorear los mensajes MQTT enviados y recibidos en el sistema para detectar posibles problemas.

Con esto acabaríamos de describir de manera esquemática la realización del proyecto siguiendo distintos pasos, comentando brevemente qué se hace en cada uno de ellos, sin entrar en una explicación demasiado extensa.

4. CONCLUSIONES

Finalmente, para concluir con la memoria de nuestro proyecto, comentaremos algunos aspectos que hemos aprendido sobre la realización del proyecto.

En cuanto a la realización , lo que más nos ha costado es la parte de MQTT, ya que al ser novatos en este ámbito nos ha resultado un poco complicado al principio. Aun así, hemos aprendido mucho a base de pruebas y errores constantes hasta dar con la tecla.

Las partes restantes de la realización nos han resultados un poco mas sencillas que la anteriormente mencionada. En resumen, ha sido un proyecto desafiante y motivador que finalmente hemos podido llevar a cabo a pesar de algunas complicaciones.

Como forma de cerrar este apartado de conclusiones, queremos mencionar algunas propuestas que se podrían implementar para mejorar nuestro proyecto, que tiene mucho margen de mejora de cara al futuro.

Algunas de ellas son la implementación de una programación horaria que según las horas del día, abre o cierre la persiana para hacer un entorno mas confortable. Otras también muy útiles serian la implementación de un control remoto mediante una aplicación móvil

o mando inalámbrico, o la de establecer más parámetros que hagan del hogar un lugar más cómodo como sensores de humedad o de calidad de aire.

Esto sumado a la eficiencia energética, su facilidad de uso, su versatilidad y adaptabilidad hace que sea una muy buena opción que sustituya en un futuro a las persianas tradicionales, mejorando a su vez la calidad de vida de los usuarios.