Entrega Integración

OBJETIVOS

Esta entrega tiene por objetivo que el alumno sea capaz de:

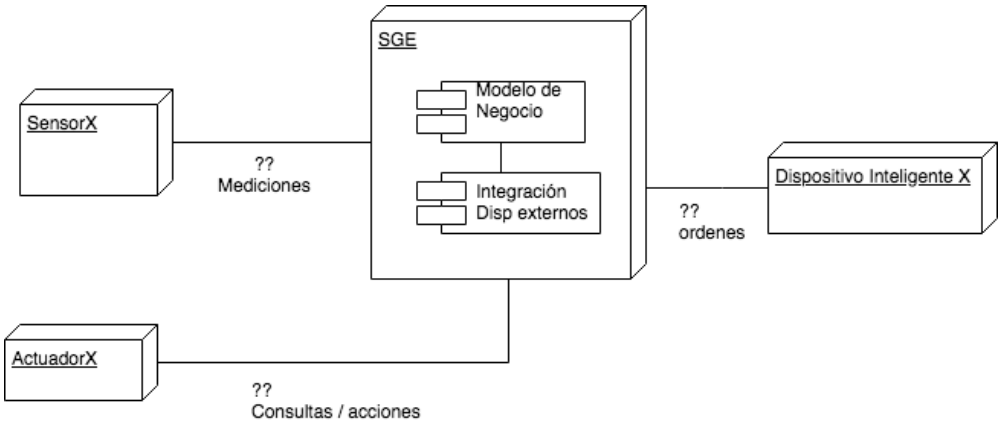
- Diseñar e implementar interfaces de integración, utilizando una mensajería y servicios web sobre HTTP.
- Comprender las diferencias básicas entre un esquema de comunicación sincrónico y asincrónico.

ENUNCIADO ESPECÍFICO

Integración

El sistema deberá comunicarse con los dispositivos físicos (externos) de los clientes residenciales, considerando la mensajería entre:

- ✓ El sistema y los sensores
- ✓ El sistema y los actuadores



Los siguientes ítems son restricciones impuestas al dominio:

- No está permitido integraciones a nivel datos, es decir, no se pueden hacer llamadas a otra base de datos mediante links o stored procedures.
- La integración entre el sistema y los sensores deberá ser implementada a través de un middleware; en particular, con una cola de mensajería (MQ).
- La integración entre el sistema y los dispositivos deberá darse a nivel aplicación. Se deberán exponer los servicios mediante una API.

Simulador de sensor

El *Sensor* deberá simular el comportamiento de un sensor físico instalado en el domicilio de algún cliente; por ejemplo: termómetro, barómetro, sensor de presencia, etc. Deberá generar mediciones para una o más variables del

ambiente e informarlas al sistema de gestión cada N minutos cumpliendo con las restricciones impuestas al dominio.

Simulador de dispositivo/actuador

El *Dispositivo/Actuador* deberá simular el comportamiento de un dispositivo inteligente físico instalado en el domicilio de algún cliente; por ejemplo: aire acondicionado, lámpara, televisión, heladera, lavarropa, etc. Deberá permitir la recepción de órdenes que actualicen el estado de funcionamiento del mismo desde el sistema de gestión.

Simulador de control

El *Control* deberá simular ser una botonera de, valga la redundancia, control que tiene el cliente para manejar o conocer el estado los dispositivos inteligentes. Deberá recibir, al menos, cinco tipos de órdenes/consultas distintas. La aplicación tendrá que contar con interfaz de usuario

ENTREGABLES

Concepción y Comunicación del Diseño

- **Diagrama de arquitectura general** actualizado: componentes, integraciones, protocolos, zonas de red, sentido de la información.
- **Otros diagramas complementarios**: si el equipo lo cree necesario, pueden realizarse diagramas complementarios para comunicar el diseño.
- **Documentación de la API** REST expuesta por el sistema de gestión.
- **Documento de configuración del ambiente:** debe especificar cómo y en qué orden se deben ejecutar las aplicaciones para mostrar un correcto funcionamiento del sistema en general; además de detallar los parámetros básicos necesarios.

Implementación

- Simulador de sensor: La aplicación no necesita contar con interfaz gráfica, es decir, basta con que la misma se ejecute desde línea de comandos. Cada instancia corrida de esta aplicación representará un nuevo sensor. Las mediciones simuladas deberán ser aleatorias.
- Simulador de dispositivo/actuador: Al igual que el simulador de sensor, no necesita contar con interfaz gráfica.
- Simulador de control: La aplicación necesita interfaz gráfica, y ésta puede ser simple del tipo CLI (command line interface).
- Todas las aplicaciones deben poder mostrar el estado de ejecución de las operaciones que realizan, para dejar en evidencia su correcto funcionamiento; por ejemplo: al recibir un pedido, partes claves de la resolución, respuesta al pedido, etc.

Recomendaciones

- Modelar el servidor web depende del lenguaje de programación utilizado, por lo cual se sugieren la utilización de los siguientes frameworks:

Java	• Framework: ***Spark*** • Documentación oficial: http://sparkjava.com/
C#	• Framework: ***ASP .Net*** • Documentación oficial: https://www.asp.net/
PHP	• Framework: ***Codeigniter*** • Documentación oficial: https://codeigniter.com/

- Implementar un cliente HTTP también depende del lenguaje de programación utilizado, motivo por el cuál recomendamos la librería Unirest (http://unirest.io/) la cual está disponible tanto para Java, como C# y PHP.
- Para usar una solución de mensajería machine-to-machine recomendamos usar el protocolo MQTT. Una implementación es moquette (https://github.com/andsel/moquette).
- Dejamos como ayuda el proyecto que se encuentra en el repositorio https://github.com/ezequieljsosa/dds-tp2018-entrega3-tips.