



Consejo Zacatecano de Ciencia Tecnología e  
Innovación

Laboratorio de Software Libre

Universidad Anahuac

## Centro de Diseño y Validación de Intel GDC

Especificación de Diseño del Sistema  
del: Sistema Embebido Inteligente para Sanitarios Intel

Víctor Ramón Carrillo Quintero  
Héctor Aurelio Elías García  
Nunila Patricia Guzmán Reyes  
Rodrigo Rubio García

Fecha de actualización: 04/Diciembre/2018

<b>1. Introducción</b>	<b>3</b>
1.1. Propósito	3
1.2. Definiciones y acrónimos	3
<b>2. Consideraciones de diseño</b>	<b>4</b>
2.1. Suposiciones	4
2.2. Limitantes	4
2.3.1 Ambiente del Sistema web	5
2.3.2 Necesidades del sensor:	5
3.1. Antecedentes	6
3.2 Justificaciones	6
3.2.1 Justificación de decisiones del sistema web	6
3.2.2 Justificación de decisiones del sensor	6
3.2.3 Justificación de decisiones de programación en el sensor.	6
<b>4. Diseño de alto nivel</b>	<b>7</b>
4.1. Vista física de componentes	7
4.2. Diagrama específicos	8
4.2.0 Diagrama de componentes de servicios	8
4.2.1 Diagrama de clases Aplicación Web	9
4.2.2 Diagrama de clases cliente mqtt	10
4.2.3 Diagrama de Microcontrolador ATMEGA 328	11
4.2.4 Diagrama de sensor de movimiento HC-SR501	11
4.2.5 Diagrama de Modulo WiFi ESP-8201	12
4.2.6 Diagrama del circuito de sensado y comunicación:	12
4.2.7 Protoboard	13
4.2.8 Diagrama PCB del sensor:	13
4.3. Base de datos	14
4.3.1 Diseño relacional	14
4.3.2 Diseño de tablas	14
4.4 Especificación de campos de la base de datos	15
5.1. Acciones	16
5.1.1 Ver baños e información por zona:	16
5.1.2 Realizar un servicio al baño:	17
5.1.3 Introducción de registro de entradas a baño:	18
<b>6. Diseño de interfaz de usuario</b>	<b>18</b>
6.2 Vistas	18
6.2.1 Vista principal	18
6.2.2 Visualizar zonas	19
6.2.3 Visualizar baños	19
6.2.4 Confirmar servicio a baño	20
6.2.5 Reiniciar visitas	20
6.2.6 Modificar Capacidad	21

<b>7. Modo de operación del sistema de sensado.</b>	<b>22</b>
7.1 Partes del sistema	22

## 1. Introducción

En este documento se provee la Especificación de los Requerimientos de Diseño de alto nivel. A manera de resumen, se describe la arquitectura de la aplicación, diagramas, interfaces asociadas y esquemas de la base de datos.

### 1.1. Propósito

Describir detalladamente la arquitectura y el diseño del software para el Sistema Embebido Inteligente para Sanitarios Intel (SEISAI). Proporcionando las vistas necesarias para facilitar la comunicación y la comprensión del mismo. Cada elemento descrito en esta sección del documento ayudará a definir el cómo se desarrollará el sistema.

### 1.2. Definiciones y acrónimos

<b>SEISAI</b>	Sistema Embebido Inteligente para Sanitarios Intel.
<b>Front-end</b>	Es la parte del software que interactúa con los usuarios.
<b>Base de Datos</b>	Colección de información organizada de forma que, un programa de ordenador pueda seleccionar rápidamente los fragmentos de datos que necesite. Una base de datos es un sistema de archivos electrónico.
<b>Bootstrap</b>	Herramienta originalmente creada por Twitter, que permite crear interfaces web con CSS y JavaScript, cuya funcionalidad principal es la de adaptar la interfaz del sitio web al tamaño del dispositivo en que se visualice.
<b>Framework</b>	Entorno de trabajo o marco de trabajo es un conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular que sirve como referencia, para enfrentar y resolver nuevos problemas de índole similar.
<b>Angular</b>	Framework para aplicaciones web desarrollado en TypeScript, de código abierto, mantenido por Google, que se utiliza para crear y mantener aplicaciones web de una sola página.
<b>MySQL</b>	Sistema de gestión de base de datos relacional open source, basado en lenguaje de consulta estructurado SQL. Se puede ejecutar en casi cualquier plataforma, incluyendo Linux, UNIX y Windows.
<b>MQTT</b>	Protocolo de mensajería basado en ISO estándar publicación-suscripción. Funciona sobre el protocolo TCP / IP. Está diseñado para conexiones con ubicaciones remotas donde se requiere una "huella de código pequeño" o el ancho de banda de la red es limitado.

<b>MVC</b>	Modelo vista controlador (MVC) Modelo Vista Controlador (MVC) es un estilo de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos.
<b>Python</b>	Lenguaje de programación interpretado el cual hace énfasis en la sintaxis y código legible. Se trata de un lenguaje multiparadigma ya que soporta orientación a objetos, programación imperativa y funcional.
<b>TypeScript</b>	Es un lenguaje de programación libre y de código abierto desarrollado y mantenido por Microsoft. Es un superconjunto de JavaScript, que esencialmente añade tipado estático y objetos basados en clases.
<b>CodeIgniter</b>	Es un framework para el desarrollo de aplicaciones en php que utiliza el <b>MVC</b> . Permite mejorar la forma de trabajar y hacerlo a mayor velocidad tanto para el desarrollo de servicios REST o aplicaciones completas.

## 2. Consideraciones de diseño

### 2.1. Suposiciones

Este proyecto supone que el Laboratorio de Software libre, cozcylt y el Centro de Diseño y Validación de Intel GDC siendo el interesado se hará cargo de:

- Proporcionar el equipo necesario para host del Sistema Web.
- Rentar un broker de mqtt (como CloudMQTT) para la comunicación entre dispositivos.

### 2.2. Limitantes

- Al no tener el hardware necesario para alojamiento del sistema, será entregado su código y desarrollado en local desde computadoras personales.
- Para esta primera iteración, nos centraremos en la realización de un prototipo del sistema, la comunicación entre el dispositivo a una broker de prueba de CloudMQTT que tiene como capacidad 5 conexiones máximas y la realización de un front end básico con las funcionalidades de:
  - Ver baños por zona
  - Ver capacidades de baños
  - Ver entradas a baños
  - Realizar un servicio al baño
 Lo cual reiniciará la cantidad de entradas empezando el conteo en 0

### 2.3.1 Ambiente del Sistema web

El Sistema será en un entorno web, para ello se debe operar bajo los siguientes requisitos:

#### Software

- Python 3.6
- Angular 7.0
- MySQL 8.0.13
- Hbmqtt 0.9.4
- paho-mqtt 1.4.0
- PyMySQL 0.9.2
- Bootstrap 4.1
- Node 10.13.0
- CodeIgniter 3.1.4 (framework PHP) para servicios REST.

### 2.3.2 Necesidades del sensor:

El sensor utilizado en este proyecto, HC-SR501 requiere para su óptimo funcionamiento:

- Voltaje de alimentación: de 5 a 12 VDC
- Configurar la salida de alarma en modo disparo repetitivo
- Salida de alarma de movimiento con ajuste de tiempo entre 3 segundos
- Tiempo de inicialización después de alimentar el módulo HC-SR05 debe transcurrir 1 minuto antes de que inicie su operación normal. Durante ese tiempo, es posible que el módulo active 2 ó 3 veces su salida.
- Tiempo de salida inactiva: cada vez que la salida pase de activa a inactiva, permanecerá en ese estado los siguientes 3 segundos. Cualquier evento que ocurra durante ese lapso es ignorado.
- Temperatura de operación: -15° a +70° C.

## 3. Arquitectura

Para este sistema se decidió:

- a. Como medio de comunicación entre el sensor de hardware y el servidor el protocolo MQTT y una instancia de prueba en CloudMQTT.com para las pruebas y prototipado.
- b. Una arquitectura de MVC y la utilización de Angular para la realización del front end.
- c. La utilización del framework PHP CodeIgniter para la realización de los servicios REST.

### 3.1. Antecedentes

Actualmente, las personas que se encargan de la realización del servicio para los baños en las oficinas de Intel, consiste en un recorrido que dura aproximadamente 1.5h por turno, a cada baño de las oficinas de intel, para conocer qué cosas hacen falta a cada baño.

### 3.2 Justificaciones

#### 3.2.1 Justificación de decisiones del sistema web

Se decidió usar la arquitectura MVC porque se conoce como la arquitectura ideal en el mundo de sistemas web, además que, proporciona una alta modificabilidad y al ser un proyecto que, para la primera entrega, se planea la realización de un prototipo, es probable que en la siguiente iteración se integren o cambien desarrolladores, y la utilización de MVC simplifica su integración. además:

- Al ir agregando más lógica de negocio y complejidad a la base de datos, se vuelve útil la separación en varias partes del modelo.
- La separación en capas permite reemplazar o modificar una capa sin afectar a módulos restantes ya que de esta forma el código del sistema es mucho más entendible.

Los servicios REST separados se utilizarán porque para futuras iteraciones se planea la realización de una aplicación móvil, la cual podría consumir estos mismos servicios.

#### 3.2.2 Justificación de decisiones del sensor

Se seleccionó el módulo PIR modelo HC-SR501, ya que además ser de bajo costo y pequeño, incorpora la tecnología más reciente en sensores de movimiento. El sensor utiliza 2 potenciómetros y un jumper que permiten modificar sus parámetros y adaptarlo a las necesidades de la aplicación: sensibilidad de detección, tiempo de activación, y respuesta ante detecciones repetitivas.

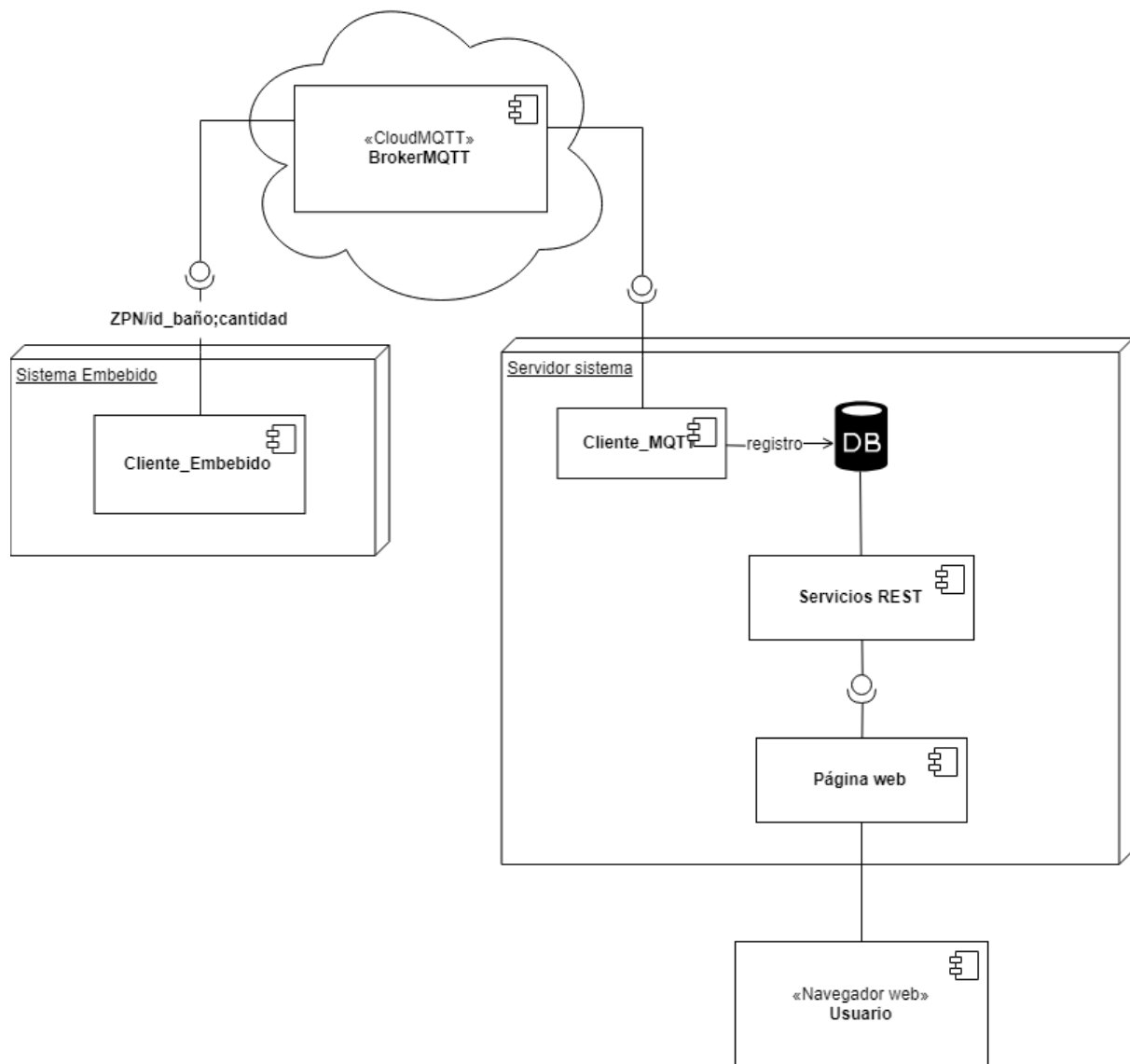
#### 3.2.3 Justificación de decisiones de programación en el sensor.

Se utilizaron comandos AT, decisión que nos puede ayudar a implementar a futuro otro módulo de comunicación con gran facilidad, ya sea GSM, WIFI o bien Sigfox. Con esto logramos que este dispositivo tenga escalabilidad.

Por otro lado la comunicación entre el microcontrolador y el módulo, en este caso wifi, se logra a través de USART a 115200 baudios y se configuró el microcontrolador para que trabaje a doble velocidad, con esto logramos una óptima recepción de datos por parte del módulo, por otro lado la recepción de datos sería errónea si tuviera una velocidad normal.

## 4. Diseño de alto nivel

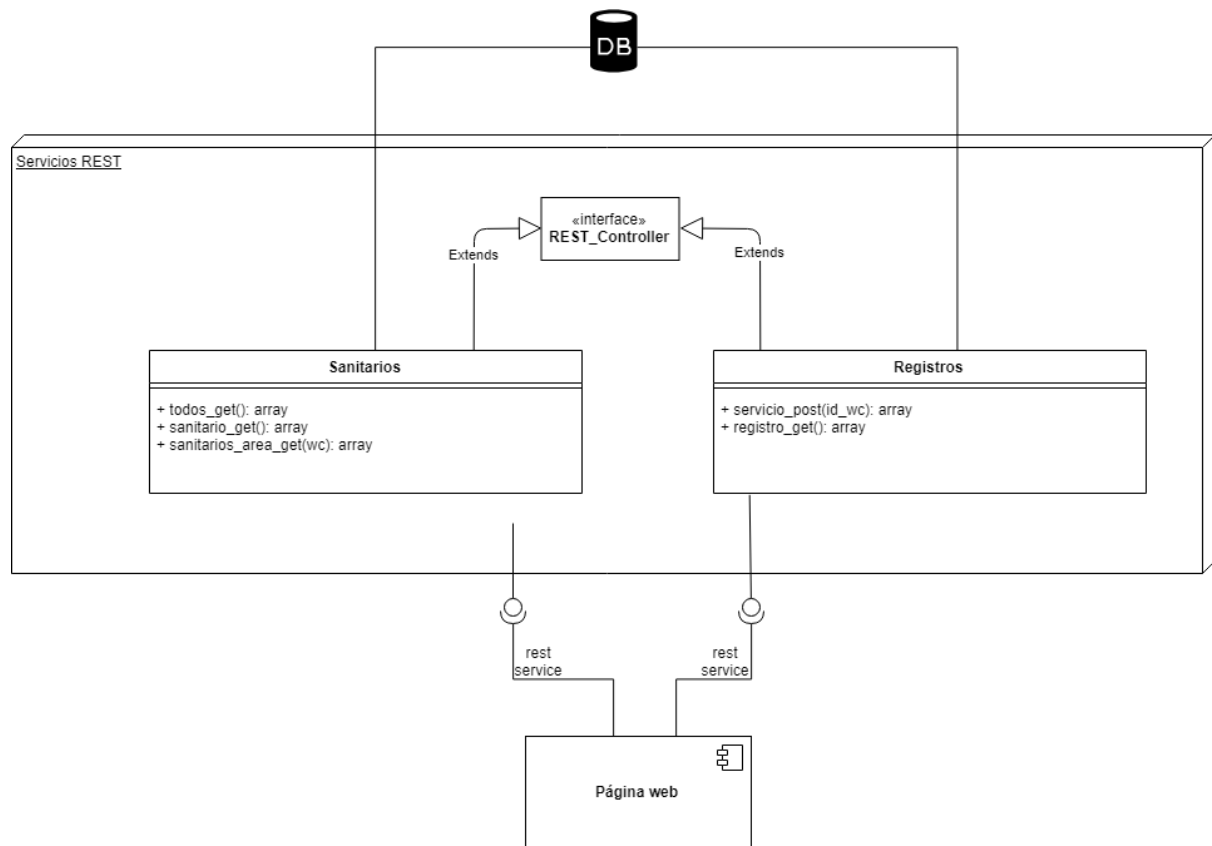
### 4.1. Vista física de componentes



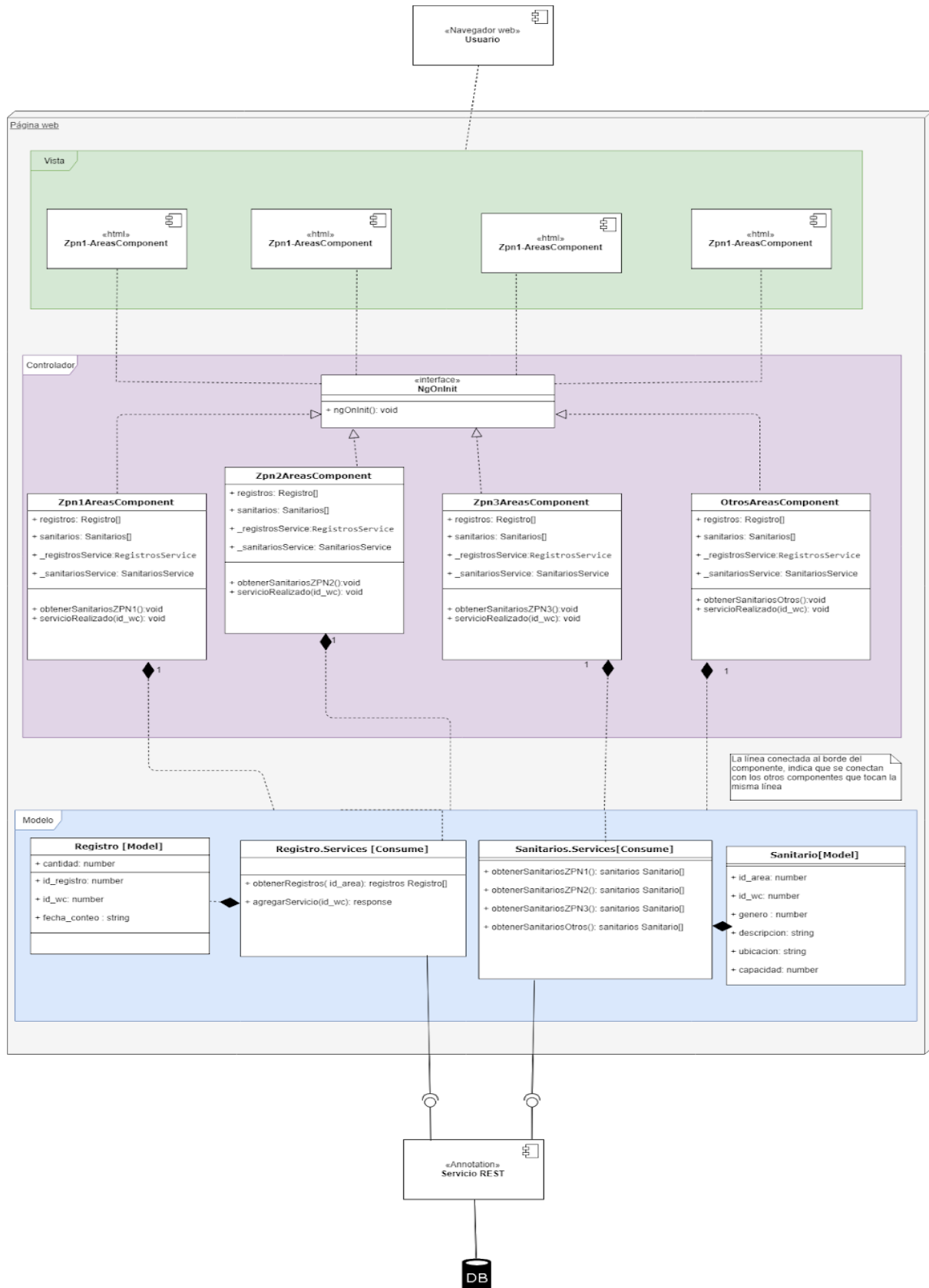


## 4.2. Diagrama específicos

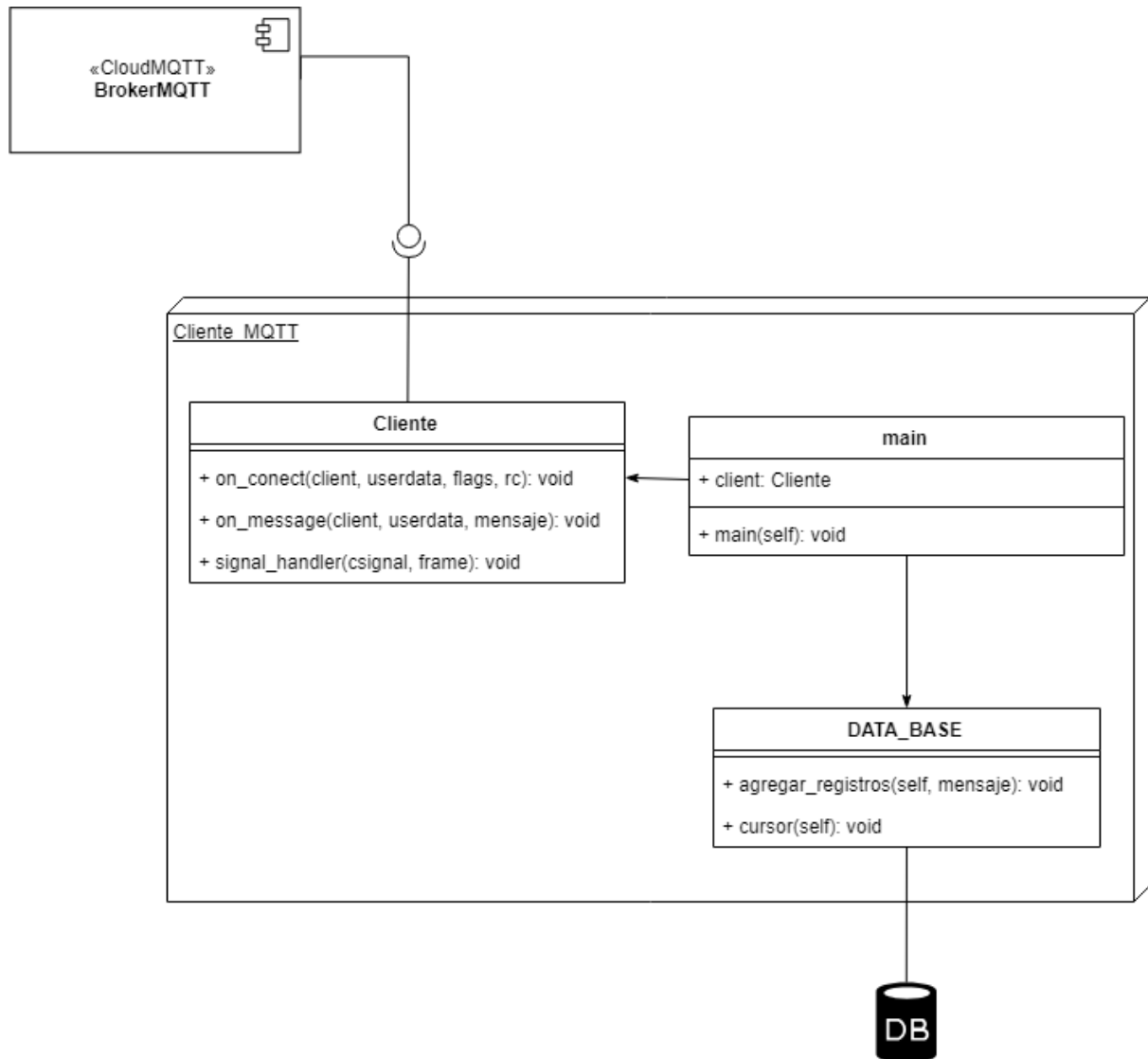
### 4.2.0 Diagrama de componentes de servicios



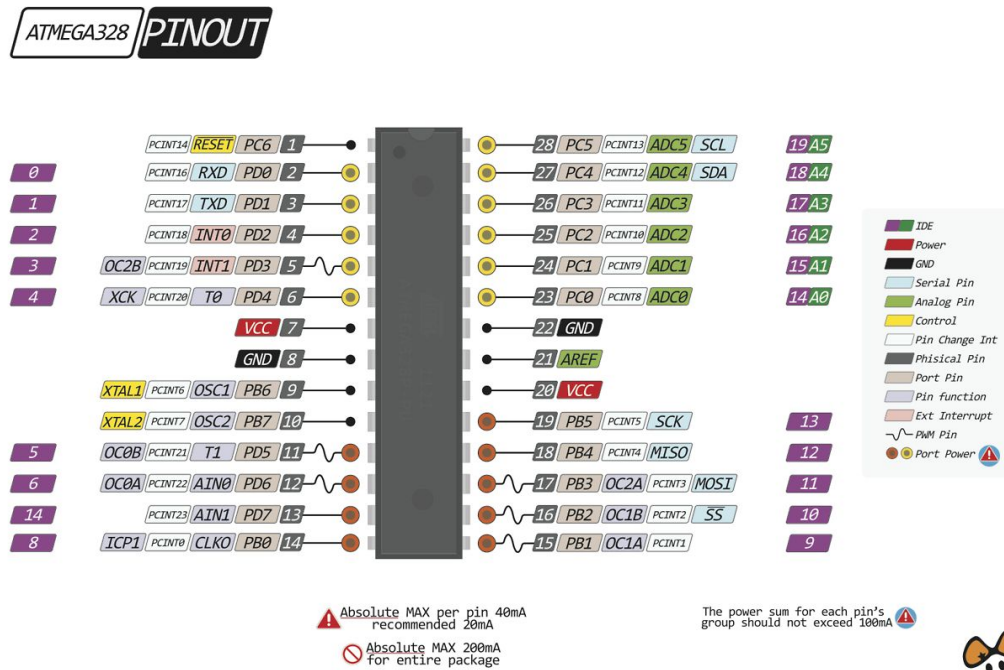
## 4.2.1 Diagrama de clases Aplicación Web



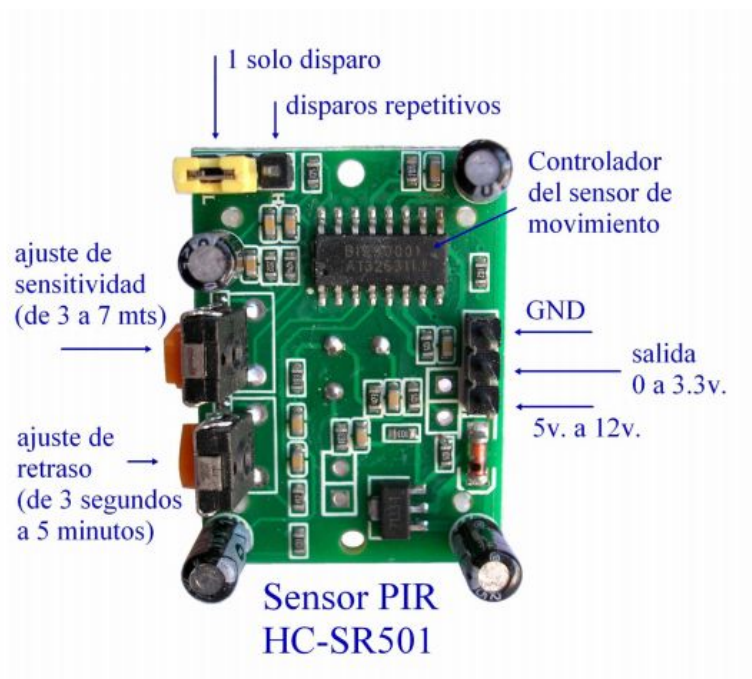
#### 4.2.2 Diagrama de clases cliente mqtt



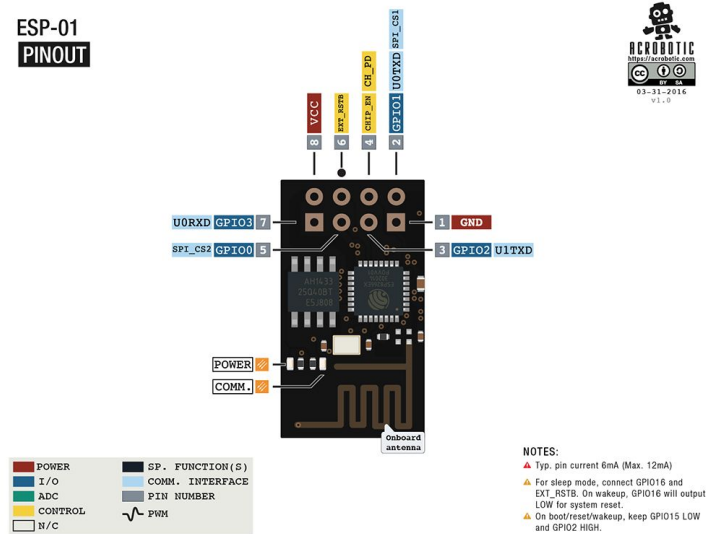
#### 4.2.3 Diagrama de Microcontrolador ATMEGA 328



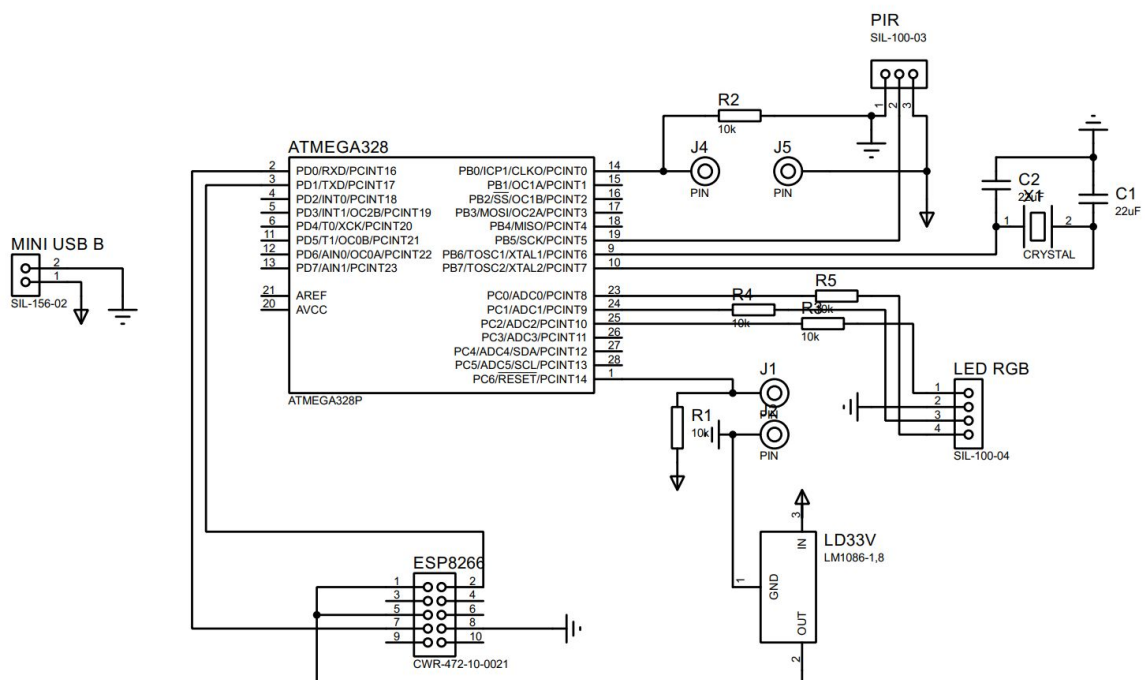
#### 4.2.4 Diagrama de sensor de movimiento HC-SR501



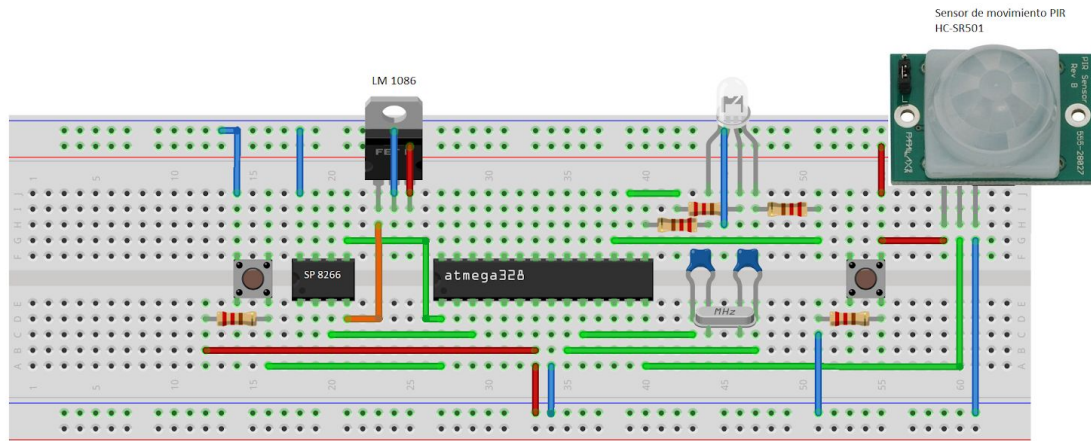
#### 4.2.5 Diagrama de Modulo WiFi ESP-8201



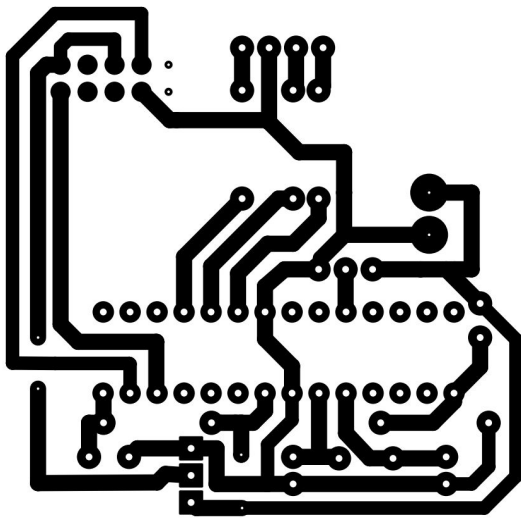
#### 4.2.6 Diagrama del circuito de sensado y comunicación:



#### 4.2.7 Protoboard

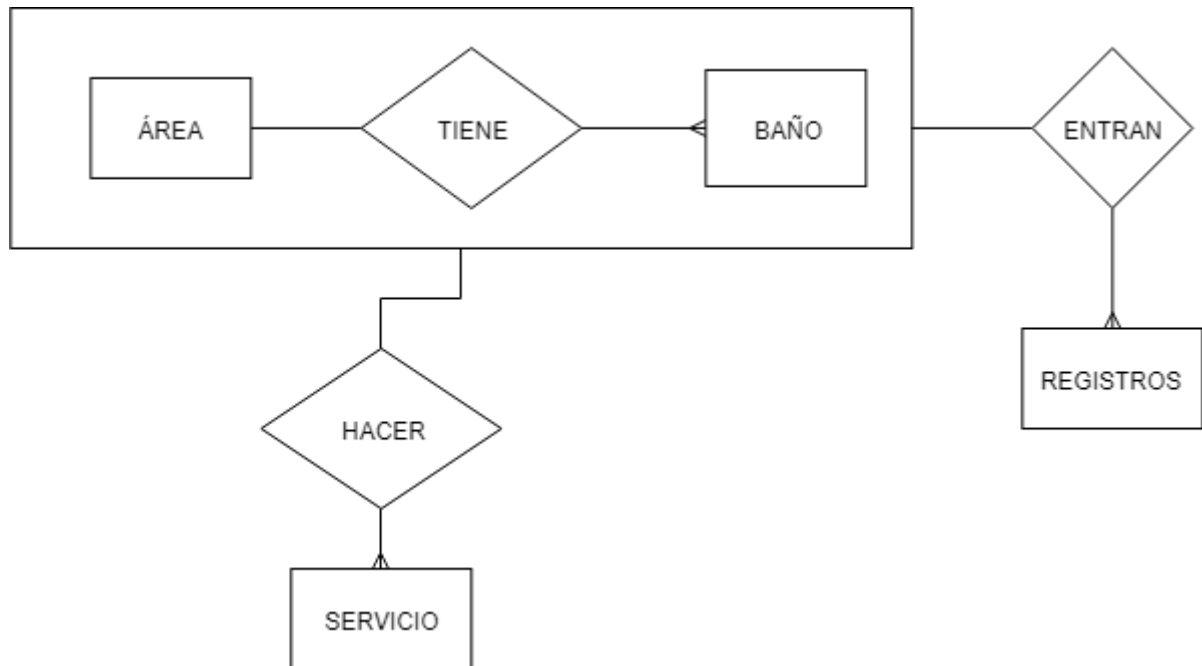


#### 4.2.8 Diagrama PCB del sensor:

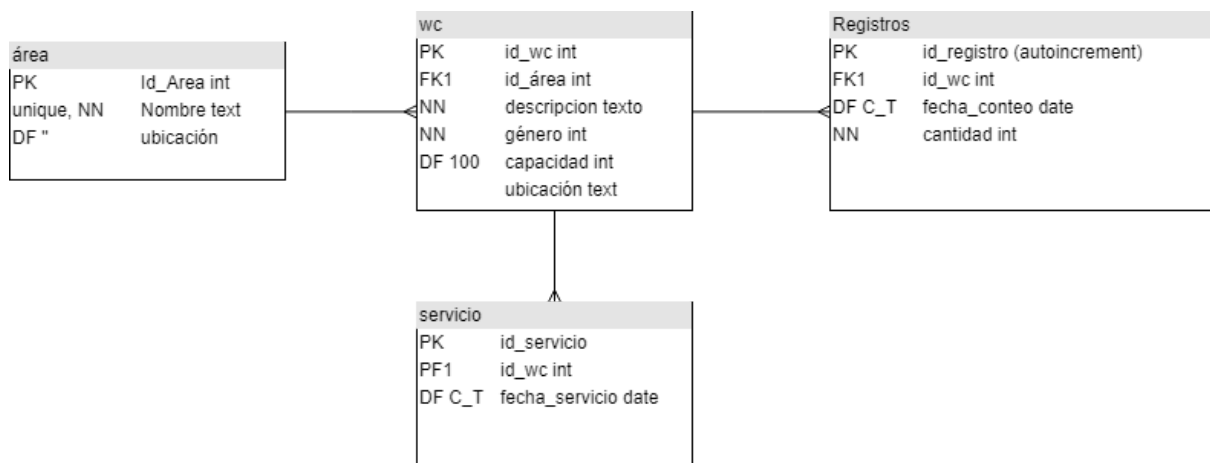


## 4.3. Base de datos

### 4.3.1 Diseño relacional



### 4.3.2 Diseño de tablas



#### 4.4 Especificación de campos de la base de datos

Tabla	Campo	Tipo	Restricciones	Descripción
área	id_area	INT	Primary key autoincrement	Identificador de una área.
	Nombre	VARCHAR(10)	Unique Not null	Nombre dado a una área.
	ubicación	VARCHAR(50)	Default “	Descripción del lugar donde se encuentra.
wc	id_wc	int	primary key, autoincrement	identificador del baño.
	id_area	int	Foreign key	Área a la cual pertenece el baño.
	descripción	varchar(100)	Not null	Descripción de un baño.
	género	int	Not null	Género de un baño (1 para hombres 0 para mujeres)
	capacidad	int	Default 100	La cantidad de personas en la cual es necesario hacer un servicio.
Registros	id_registro	int	Primary key, autoincrement.	identificador del registro
	id_wc	int	Not null, Foreign Key	id del baño del que se registran las entradas.
	fecha_conteo	Date	Default current timestamp	Fecha en que se registra la entrada de personas.
	cantidad	int	Not null	Cantidad de personas que se registra entraron.
Servicio	id_servicio	int	Primary key	Identificador del servicio realizado.
	id_wc	int	Foreign key	Baño al que se

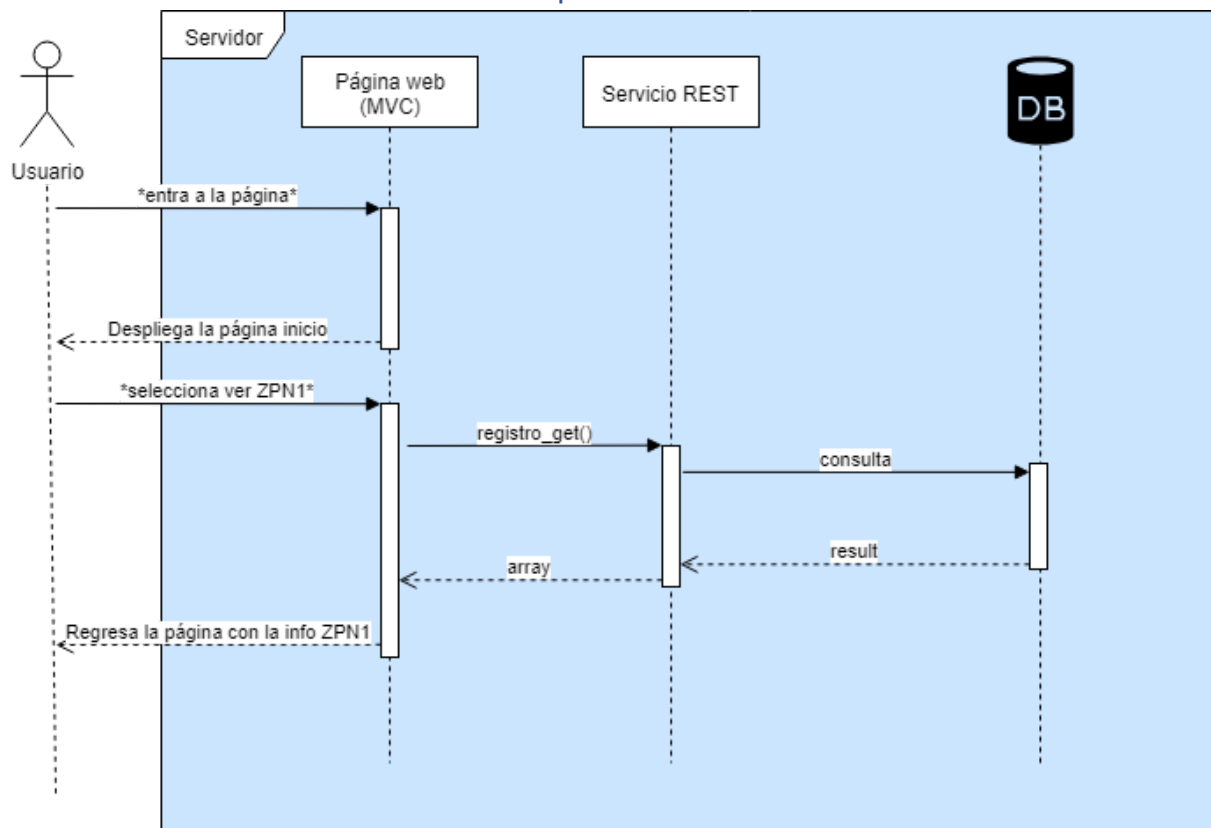


				le hace servicio.
	fecha_servicio	Date	Default current timestamp	Fecha en que se hizo el servicio.

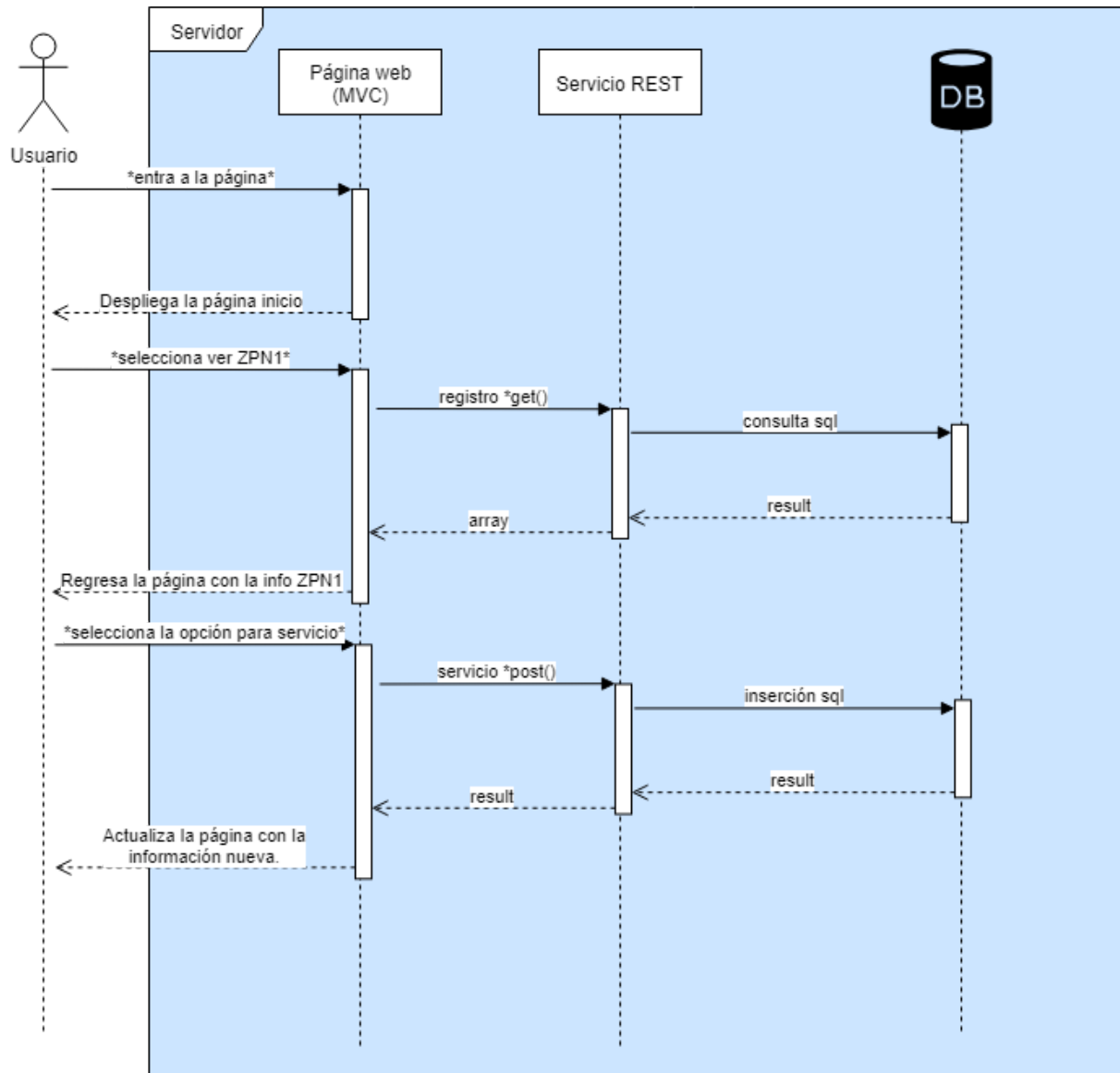
## 5. Diseño de bajo nivel

### 5.1. Acciones

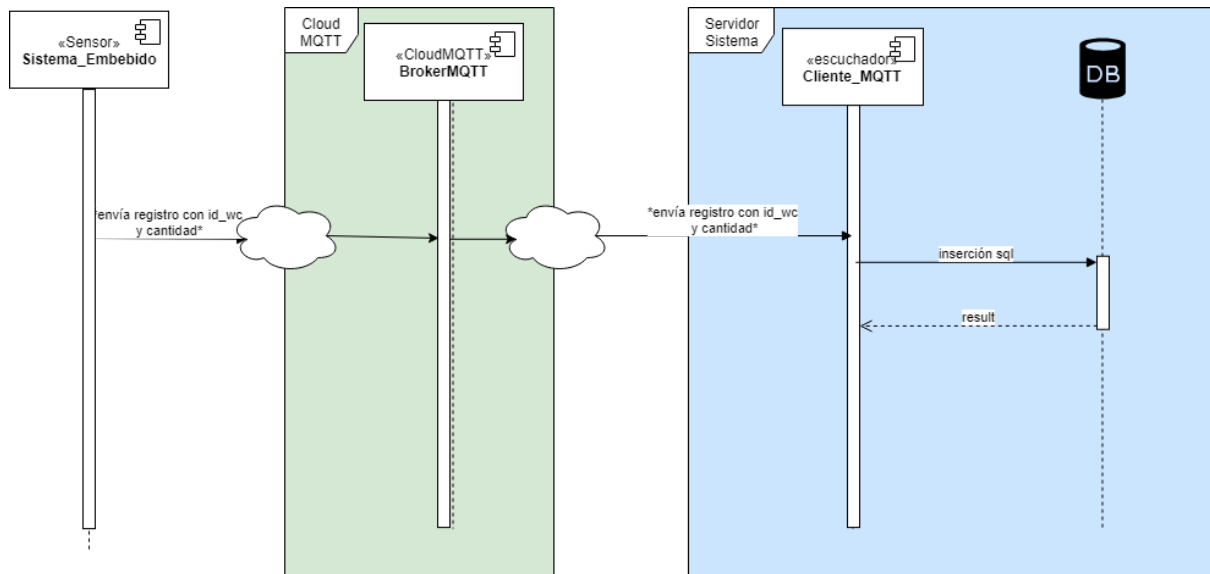
#### 5.1.1 Ver baños e información por zona:



### 5.1.2 Realizar un servicio al baño:



### 5.1.3 Introducción de registro de entradas a baño:



## 6. Diseño de interfaz de usuario

### 6.2 Vistas

#### 6.2.1 Vista principal



### 6.2.2 Visualizar zonas

**Intel Janitory Service Efficiency**

**CAPACIDADES**

ZPN1 <sup>1</sup>	ZPN2	ZPN3 LABS <sup>1</sup>
ZPN1.0 - GYM	ZPN2.1	ZPN3
ZPN1.0	ZPN2.2	
ZPN1.1	ZPN2.3	
ZPN1.1 - auditorio	ZPN2.4	
ZPN1.2		
ZPN1.3		

### 6.2.3 Visualizar baños

**Janitory Service Efficiency**

**ZPN1.0 - GYM**


Icon	VISITAS:	CAPACIDAD:	Alert
Woman	314	265	*Servicio necesario * ¿Se le dio servicio?
Man	159	358	¿Se le dio servicio?

## 6.2.4 Confirmar servicio a baño

**Janitory Service Efficiency**

ZPN1<sup>1</sup> ZPN2 ZPN3 OTROS<sup>1</sup> CAPACIDADES

**ZPN1.0 - GYM**

 VISITAS: **314** CAPACIDAD: **265**

\*Servicio necesario \*  
**¿Le dio servicio?**

**¿Ya se le dio servicio a este baño?**

**Aún no** **Si, Servicio realizado**

## 6.2.5 Reiniciar visitas

**Janitory Service Efficiency**

ZPN1 ZPN2 ZPN3 OTROS<sup>1</sup> CAPACIDADES

**ZPN1.0 - GYM**

 VISITAS: **0** CAPACIDAD: **265**

**¿Se le dio servicio?**

 VISITAS: **159** CAPACIDAD: **358**

**¿Se le dio servicio?**

**ZPN1.0 -GYM** **ZPN1.0** **ZPN1.1** **ZPN1.1 -AUDITORIO** **ZPN1.2** **ZPN1.3** **EDITAR CAPACIDADES**

## 6.2.6 Modificar Capacidad

**Janitory Service Efficiency**

**ZPN1****ZPN2****ZPN3****OTROS****CAPACIDADES**

1

# CAPACIDADES DE ZPN1

ÁREA	HOMBRES	MUJERES
	Notificar cada:	
ZPN1.0 GYM	358	265
ZPN1.1	200	300
ZPN1.1 Auditorio	200	300
ZPN1.2	200	300
ZPN1.3	200	300

EDITAR

## 7. Modo de operación del sistema de sensado.

### 7.1 Partes del sistema

### 7.2