Diseño de una aplicación Web extensible para la administración de una plataforma IoT diseñada para Smart Campus.

Diego Federico Camacho Naranjo

Trabajo de Grado para optar por el título de Ingeniero de Sistemas

Director

Gabriel Rodrigo Pedraza Ferreira

PhD. Ciencias de la Computación

Universidad Industrial de Santander

Facultad de Ingenierías Físico Mecánicas

Escuela de Ingeniería de Sistemas e Informática

Bucaramanga

2019

Dedicatoria

“A Dios por darme la vida y las capacidades para llevar a cabo este proyecto y todas mis metas.

A mi familia por apoyarme en cada momento de mi vida, por su cariño y alentarme a ser mejor cada día y luchar por mis sueños.

Al profesor Gabriel por su guía durante el desarrollo de este proyecto.

A la UIS por todas sus enseñanzas durante estos años y todos los gratos momentos que viví durante esta etapa.

A mis amigos, por acompañarme desde el inicio de esta travesía, por todos los momentos que hemos compartido y los proyectos y metas que hemos alcanzado juntos.”

Tabla de Contenido

Lista de Tablas

Lista de Figuras

Lista de Apéndices

(Ver apéndices adjuntos en el CD y pueden visualizarlos en la Base de Datos de la Biblioteca UIS)

Lista de Siglas

AMQP:	Advanced Message Queuing Protocol
AWS:	Amazon Web Services
CoAP:	Constrained Application Protocol
HTTP:	Hypertext Transfer Protocol
IoT:	Internet of Things
JSON:	JavaScript Object Notation
MQTT:	Message Queuing Telemetry Transport
REST:	Representational State Transfer
SPA:	Single Page Application
URI:	Uniform Resource Identifier

Resumen

TITULO: DISEÑO DE UNA APLICACIÓN WEB EXTENSIBLE PARA LA ADMINISTRACIÓN DE UNA PLATAFORMA IOT DISEÑADA PARA SMART CAMPUS*

AUTOR: DIEGO FEDERICO CAMACHO NARANJO**

PALABRAS CLAVE: IOT, SMART CAMPUS, APLICACIÓN, WEB

DESCRIPCIÓN

El crecimiento tecnológico de nuestra época ha conectado el mundo y permitido el flujo de la información desde y hacia diversos lugares; mediante el Internet de las Cosas es posible que objetos comunes se conviertan en objetos inteligentes, envíen y reciban información; la cual, si es utilizada correctamente permitiría automatizar labores cotidianas.

Esta vertiente tecnológica puede ser aplicada a diferentes escalas, en este caso a un Campus Universitario donde se pretende, haciendo uso del IoT mejorar la calidad de vida de los individuos que hacen uso de las instalaciones universitarias a través de la transformación de estas en universidades inteligentes (Smart Campus), por ejemplo, permitiendo optimizar recursos físicos como inventario, la energía consumida o monitorear métricas importantes como la calidad del aire o la temperatura en diversas instalaciones del Campus.

En este proyecto de grado se presenta el diseño de una aplicación Web que permite gestionar una arquitectura Smart Campus mediante la cual los usuarios pueden registrar y gestionar sus aplicaciones y los dispositivos asociados a las mismas permitiéndoles de una manera sencilla crear y administrar aplicaciones que contribuyan a generar esta transformación digital.

Al final se desarrolló un caso de uso para probar las funcionalidades de la aplicación Web y la correcta integración de los componentes que componen la plataforma Smart Campus.

---

*Trabajo de grado

**Facultad de Ingenierías Fisicomecánicas. Escuela de Ingeniería de Sistemas e Informática

Director: Gabriel Rodrigo Pedraza Ferreira, PhD. Ciencias de la Computación

Abstract

TITLE: DESIGN OF AN EXTENSIBLE WEB APPLICATION TO MANAGE AN IOT PLATFORM FOR SMART CAMPUS

AUTOR: DIEGO FEDERICO CAMACHO NARANJO**

KEYWORDS: IOT, SMART CAMPUS, APPLICATION, WEB

DESCRIPTION:

The technological growth of our time has connected the world and allowed the flow of information to and from different places. Through the Internet of Things it is possible for common objects to become intelligent, send and receive information which, if used correctly, would automate daily tasks.

This technological aspect can be applied to different scales, in this case it's intended to a University, to make use of the IoT to improve the quality of life of the individuals who make use of the facilities through the transformation of these into intelligent universities (Smart Campus) for instance optimizing physical resources through inventories, measuring the energy used, air quality or temperature in specific campus facilities.

In this bachelor thesis it's presented the design of a Web application to manage a Smart Campus architecture allowing the users to register and manage their applications and the devices associated with them, allowing them also to create applications that contribute to generate this digital transformation.

At the end a use case was developed to test the functionalities of the Web application and the correct integration of the components that make up the Smart Campus platform.

---

*Bachelor Thesis

**Facultad de Ingenierías Físicomecánicas. Escuela de Ingeniería de Sistemas e Informática

Director: Gabriel Rodrigo Pedraza Ferreira, PhD. Computer Science

Introducción

El auge de la era digital ha llevado a que el número de dispositivos conectados a internet en el 2018 excediera los 17 mil millones (Lasse K, 2018), esto se debe a la tendencia de interconectar dispositivos y transformar objetos cotidianos en dispositivos inteligentes que reciben y envían información. Con el Internet de las Cosas (IoT) se pretende aprovechar la información producida por estos dispositivos para generar valor agregado a los usuarios y realizar labores cotidianas de forma automatizada permitiendo disminuir costos. Esta tendencia puede generar beneficios en todas las áreas del conocimiento; varios casos de uso populares son, monitorizar el estado de los pacientes, el tráfico y la calidad del aire en ciudades, entre otros.

El Internet de las Cosas se puede aplicar tanto para darle más inteligencia a ciudades, casas e incluso campus universitarios, lo cual es el objetivo de este proyecto; donde se pretende mediante la transformación digital permitir automatizar procesos realizados, gestionar de forma más eficiente recursos y en general mejorar la calidad de vida de todas las personas que pertenecen de una u otra forma a las universidades.

El desarrollo de aplicaciones IoT es un gran esfuerzo pues requiere una arquitectura Software robusta con la que se controlen, gestionen y administren dispositivos Hardware. Es por esto que en este proyecto se presenta el diseño de una aplicación web para la administración de una plataforma Smart Campus mediante la cual los usuarios pueden registrar y gestionar sus aplicaciones, dispositivos y procesos para permitir el flujo y almacenamiento de información desde la plataforma a los dispositivos y viceversa, permitiendo que los diferentes casos de uso IoT sean implementados sobre esta arquitectura extensiva brindándole a los usuarios una forma intuitiva de ver y gestionar los elementos pertenecientes a la misma.

1. Objetivos

1.1 Objetivo General

Diseñar una aplicación Web extensible que permita a los usuarios gestionar una plataforma IoT para Smart Campus.

1.2 Objetivos Específicos

- Diseñar una aplicación Web reactiva que se conecte, reciba y envíe información desde y hacia un backend acerca del estado de la plataforma IoT.
- Elaborar un prototipo funcional de una aplicación Web con una interfaz intuitiva, que permita a los usuarios comprender la información proveniente de sensores y dispositivos.
- Permitir a los usuarios monitorizar y gestionar los dispositivos asociados a la plataforma IoT Smart Campus.
- Proveer a los usuarios información sobre el estado de sus dispositivos y aplicaciones, para que puedan tomar decisiones con base en dicha información.

2. Estado del arte

El Internet de las cosas fue mencionado públicamente por primera vez por Kevin Ashton, profesor del MIT y desde entonces se utiliza este término para referirse a todo sistema de dispositivos interrelacionados con la habilidad de transferir información en una red sin requerir interacción humano a humano o humano a máquina (Santhi T., Rajendra J. & Vijayalakshmi, Y, 2006). Tras esta primera alusión muchos investigadores empezaron a trabajar en esta corriente; en el 2008 se creó la IPSO Alliance, una alianza para trabajar en el Internet de las cosas y cuyo objetivo es promover el uso de protocolos de internet en objetos inteligentes, iniciando el camino para que esta corriente se convirtiera en una realidad (Universidad de Alcalá, s.f.).

El primer proyecto conocido fue una máquina expendedora de refrescos en la Universidad Carneige Mellon, en donde mediante la web los usuarios podían revisar el estado de la máquina y saber si había refrescos disponibles para evitar dirigirse a la misma y gastar tiempo cuando esta se encontraba vacía (IoT Agenda, 2019). Aunque este suena como un ejemplo muy sencillo el potencial del Internet de las cosas es enorme y en el transcurso de los años se ha venido demostrando, por ejemplo con soluciones de empresas importantes como Google, Microsoft y Amazon quienes han desarrollado diversas plataformas las cuales permiten conectar a dispositivos la nube y a otros dispositivos de una manera sencilla y segura, usando diferentes protocolos para enviar, colectar, procesar información y tomar acciones inteligentes basadas en la misma permitiendo generar valor agregado a la información generada por los dispositivos.

Existen actualmente múltiples áreas de aplicación para el Internet de las cosas, una de las más novedosas consiste en proyectos sobre Ciudades Inteligentes en donde existen proyectos como ShotSpotter, que permite monitorizar y detectar el uso de armas de fuego en un área de cobertura 250 veces mayor que las aplicaciones de protección militares estándar permitiendo reducir el gasto

del tiempo del cuerpo de policías, ayudándoles a responder de manera más efectivas a emergencias y tiroteos reales y predecir patrones de crimen y áreas peligrosas (Shotspotter, s.f); Smart Parking de Telensa, donde mediante sensores se monitorea e informa a los usuarios los lugares ocupados y disponibles para parquear en una ciudad, contribuyendo a una mayor eficiencia en el parqueo de vehículos y descongestión de vías (Telensa, s.f). Esta tendencia es aplicada también para dar cierta inteligencia a edificios, casas, granjas, entre otras y permite resolver problemas cotidianos, reduciendo costos y/o generando predicciones de posibles eventos negativos para actuar antes de que estos se presenten.

Figura 1. Campos de aplicación del Internet de las Cosas.

Tomada de IoT World Today. (2017)

La mayoría de las soluciones integrales de Internet de las Cosas incluyen una plataforma de administración Web, que permite a los usuarios gestionar y monitorizar los dispositivos asociados a la misma y la información que se envía por medio de estos. La razón por la que principalmente

se seleccionan plataformas Web para cumplir este propósito es por la compatibilidad existente con diversos dispositivos utilizados (computadores, tablets, celulares), por la sencillez y gran crecimiento de las posibilidades del desarrollo Web lo que permite entre otras cosas mostrar información en tiempo real en diversos dispositivos.

3. Marco de Referencia

En este capítulo se describen las bases teóricas estudiadas para el desarrollo de este proyecto: Internet de las cosas, Smart Campus y herramientas de desarrollo Web.

3.1 Internet de las cosas (IoT)

El Internet de las cosas se refiere al vasto mundo de dispositivos interconectados con sensores embebidos, los cuales son capaces de proveer data y, en algunos casos permiten ser controlados a través de internet. Algunos ejemplos incluyen muchos dispositivos de automatización de hogares, pero hay muchos más, desde sensores de tráfico hasta medidores de la calidad del agua, etc.

Los sistemas de Internet de las cosas tienen aplicaciones en todas las industrias por su flexibilidad única y, la capacidad de ser adecuados a cualquier entorno. Estos sistemas mejoran la recopilación de datos, la automatización, las operaciones y mucho más a través de dispositivos inteligentes. Hay siete características cruciales para el internet de las cosas: Conectividad, Objetos, Información, Comunicación, Inteligencia, Acción y Ecosistema (I-Scoop, 2017).

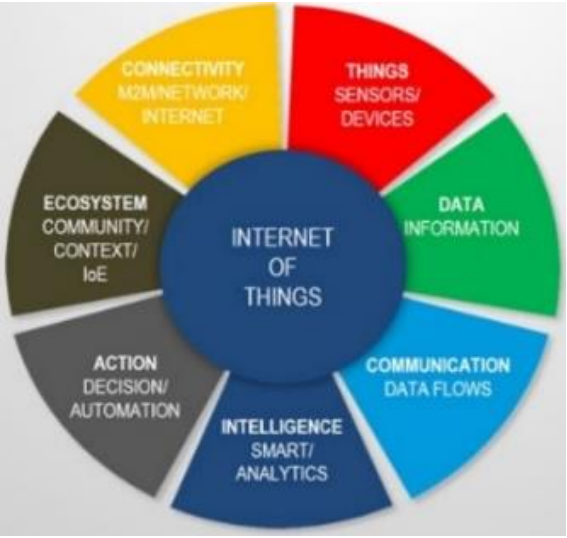

Figura 2. Siete características del Internet de las cosas. Tomada de I-Scoop. (2017).

3.2 Smart Campus

Smart Campus o Campus inteligente es un concepto que pretende que los campus o ciudades universitarias utilicen sensores, actuadores y diversos dispositivos inteligentes para colectar y usar información para gestionar recursos de una manera eficiente e incluso contribuir en labores de investigación y desarrollo. Este concepto es en otras palabras reducir el área de impacto de lo que ya conocemos como Ciudades Inteligentes.

Un ejemplo de un caso de uso para un Smart Campus es mediante sensores, controlar e informar acerca del estado de los parqueaderos, aulas y demás instalaciones físicas, actividades que usualmente son realizadas por personas y que, si se tecnifican podrían generar una reducción sustancial en gastos para las universidades.

3.3 Aplicaciones Web

Las aplicaciones Web son un tipo de software que se codifica en un lenguaje soportado por los navegadores web y cuya ejecución es llevada a cabo por el mismo (Wiboomedia. s.f.).

Las aplicaciones Web son muy populares debido a:

- La practicidad que ofrecen los navegadores web como clientes ligeros.

- La independencia del sistema operativo que se use en el ordenador o dispositivo móvil.
- La facilidad para actualizar y mantener aplicaciones web sin la necesidad de tener que distribuir el software o que se tenga que instalar el mismo por los usuarios potenciales.
- El libre acceso de los usuarios en cualquier momento, lugar o dispositivo, sólo con tener conexión a Internet y los datos de acceso.

El desarrollo web está creciendo a una tasa agresiva por la gran demanda de mejores interfaces y más amigables. Para desarrollar una aplicación web satisfactoriamente hay muchos factores que se deben tener en cuenta. Algunos de estos son:

- Diseño de Interfaz y experiencia de Usuario.
- Escalabilidad: buen uso del poder de computación, el ancho de banda y balanceo de carga entre servidores.
- Rendimiento: que sean rápidas y eficientes.
- Disponibilidad: El grado con el que los recursos del sistema están disponibles para su uso por el usuario final a lo largo de un tiempo dado.

3.4 Single Page Applications

Las Single Page Applications (SPA) o Aplicaciones de Página Única son aplicaciones Web que cargan todo el contenido en una sola página para mejorar y unificar la experiencia de usuario (Baquero, J, 2017). Como todo lo que se muestra y procesa pertenece a la misma página, al pasar de una sección a otra el navegador no es recargado y en lugar de esto, cuando la página es cargada por primera vez se cargan los recursos necesarios para la primera renderización y a posteriori se van descargando los de las demás secciones de la aplicación.

A pesar de que solo existe una página existen en realidad múltiples vistas, ya que al navegar entre ellas el usuario observa elementos e información diferente.

3.5 Typescript

Typescript es un lenguaje de programación creado por Microsoft para el desarrollo de aplicaciones con Javascript solucionando muchos problemas de este, fue creado para ser usado en el desarrollo de aplicaciones robustas, por lo cual es definido como un superset de Javascript (Código Facilito, 2016).

Unas de las principales características de Typescript es el tipado estático (las variables tienen un tipo de datos y los valores solo se pueden asignar a variables del tipo correspondiente), es orientado a objetos, por lo que permite crear y definir clases, interfaces, tipos de datos genéricos, argumentos y retornos tipados, entre otros. Esto ayuda a reducir los posibles errores de código y hacer que este sea a su vez más limpio. Este lenguaje es compilado a Javascript por lo que puede ser interpretado fácilmente por cualquier navegador.

3.6 Angular

Angular es un framework de código abierto para construir aplicaciones Web lanzado en 2016 que utiliza Typescript como lenguaje de Programación y es a su vez basado en componentes.

El desarrollo Web basado en componentes permite encapsular toda la lógica y estilos de una característica específica e incluso reutilizar esta lógica en diversas partes de la aplicación sin necesidad de replicar el código. También estas aplicaciones son modulares, donde un conjunto de código se dedica a cumplir un objetivo y exporta partes representativas del mismo, estos módulos pueden ser cargados de forma perezosa, por lo cual el tiempo de carga inicial de la aplicación es menor.

Otro gran beneficio de este framework es el enlace de datos (data binding) que se establece entre la lógica de negocio y la interfaz de usuario, de modo que cuando se realicen cambios en el

modelo, la vista se actualiza instantáneamente reaccionando rápidamente a cambios realizados mejorando la experiencia de usuario.

3.7 Servicios REST

REST (REpresentational State Transfer) es un estilo arquitectural independiente del lenguaje de programación para desarrollar Servicios Web (Rouse, M, 2017). Define un conjunto de principios arquitectónicos por los que se diseñan estos servicios enfocándose en los recursos del sistema incluyendo cómo se accede a la información y el estado de estos y cómo estos son transmitidos por HTTP (Hypertext Transfer Protocol) a clientes que utilizan diversos lenguajes de programación. Hay cuatro principios básicos de diseño de servicios REST (De Seta, L, 2008):

- Utiliza métodos HTTP.
- Son Stateless (no mantienen el estado entre varias peticiones).
- Expone URIs (Uniform Resource Identifier) con forma de directorios.
- Transfiere la información en formato JSON (JavaScript Object Notation).

3.8 WebSockets

Los WebSockets permiten la comunicación bidireccional entre aplicaciones web y procesos del lado del servidor generando un canal de comunicación bidireccional en el navegador que permite el envío y recepción de información entre el Cliente y el Servidor.

3.9 Broker de Mensajería

Se puede definir como un *middleware* (software que ayuda a la interacción o comunicación entre diferentes aplicaciones, facilitando las conexiones y sincronizaciones, abstrayendo la complejidad de las redes de conexiones, sistemas operativos y lenguajes de programación) orientado a mensajes, actuando como un agente de transferencia de mensajes, intercambiándolos entre diferentes aplicaciones, pudiendo ser estas aplicaciones: emisores o receptores. Se encarga de

traducir los mensajes de los productores a los consumidores. Estos mensajes son elementos que han sido formalmente definidos entre las diferentes aplicaciones que se comunican. También proporciona la validación, transformación y enrutamiento de los mensajes (Fernández, D, 2017).

3.10 MQTT

MQTT es un servicio de publicación/suscripción TCP/IP sencillo y sumamente ligero. Se basa en el principio cliente/servidor.

El servidor, llamado *broker*, recopila los datos que los *publishers* (los objetos comunicantes) le transmiten. Determinados datos recopilados por el *broker* se envían a determinados *publishers* que previamente hayan solicitado esta información al *broker*.

Los *publishers* envían los mensajes a un canal llamado *topic*. Los *subscribers* (suscriptores) pueden leer esos mensajes y van recibiendo estos a medida que van llegando.

4. Marco metodológico

La metodología de desarrollo que se utilizó a lo largo del proyecto está conformada por cinco fases. La primera fase corresponde a la capacitación necesaria para la realización de este proyecto, luego iterativamente se realizaron tres fases para el desarrollo de prototipos evolutivos junto con su respectiva validación y verificación.

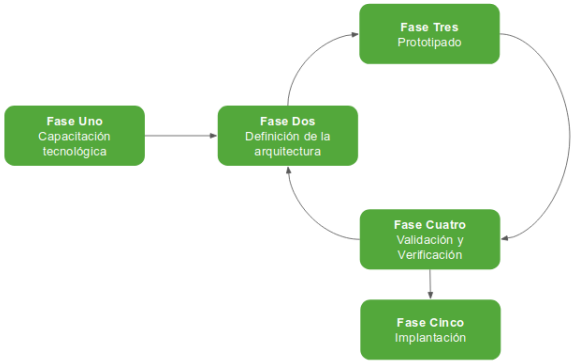

Figura 3. Esquema metodología de trabajo.

4.1 Capacitación tecnológica

En esta fase se investigó acerca de los fundamentos teóricos sobre el internet de las cosas, desarrollo de aplicaciones Web y demás conceptos relacionados. Se realizó una breve revisión de diversas plataformas IoT existentes en el mercado y tutoriales acerca de las mismas, así como se investigaron diversos protocolos, frameworks y lenguajes de programación necesarios para la realización del proyecto.

Las actividades realizadas en esta fase fueron las siguientes:

- Investigación de fundamentos teóricos y estado del arte.
- Indagación de plataformas IoT existentes en el mercado y algunas de sus características
- Selección de las tecnologías, frameworks y lenguajes de programación a utilizar.
- Capacitación en las tecnologías, frameworks y lenguajes de programación a utilizar.

4.2 Definición de la Arquitectura

En esta fase, en la primera iteración se definieron las funcionalidades, diseño y características de la plataforma de administración y la arquitectura para la solución IoT Smart Campus.

Para cada una de las siguientes iteraciones, en esta etapa se revisó la arquitectura de la plataforma Smart Campus y la splicación de administración, y se realizaron los ajustes necesarios sobre estas para continuar con las iteraciones y llevar a cabo la implementación de las diversas características definidas previamente, así como la definición de otras que fueron necesarias para cumplir con los objetivos del proyecto.

Las actividades realizadas en esta fase fueron las siguientes:

- Especificación del alcance del proyecto y sus características.
- Definición de la arquitectura de la plataforma Smart Campus.
- Definición de los requerimientos, funcionalidades y arquitectura de la aplicación Web.

4.3 Prototipado

En el transcurso de esta fase se diseñó y desarrolló un prototipo funcional de la plataforma de administración de la solución IoT. Para cada iteración del ciclo se mejoró dicho prototipo, incluyendo más características y mejorando las existentes hasta cumplir todos los requerimientos establecidos previamente, también se consideró la integración de este prototipo con la arquitectura IoT.

Las actividades realizadas en esta fase fueron las siguientes:

- Diseño del prototipo de la plataforma de administración IoT.
- Integración de la plataforma de administración con el resto de la arquitectura IoT Smart Campus.

4.4 Validación

Para esta fase se verificó si el procedimiento, la metodología usada y si la implementación del prototipo cumple los requerimientos establecidos y, se revisó si esta brinda una solución a los objetivos planteados.

Las actividades realizadas en esta fase fueron las siguientes:

- Pruebas sobre los componentes de la plataforma de administración.
- Pruebas de integración de la plataforma de administración con la plataforma Smart Campus.
- Ajustes a los módulos de código en caso de ser necesario.
- Documentación de la implementación realizada.
- Revisión del cumplimiento de especificaciones y logro de objetivos.

4.5 Caso de Uso Implementación e Implantación

Luego de que la plataforma de administración de la arquitectura IoT pase las respectivas pruebas de verificación y validación, se procederá a desplegarla en servidores en producción donde se revisará su disponibilidad y rendimiento.

Las actividades realizadas en esta fase fueron las siguientes:

- Despliegue de la plataforma de administración en servidores de producción.
- Pruebas de Responsive.
- Aplicación de métricas para verificar el rendimiento.

5. Arquitectura de referencia, Requisitos y especificaciones

En este capítulo se muestra la arquitectura que se desea implementar desde la perspectiva de los usuarios, es decir, lo que estos deben desarrollar para poder crear aplicaciones IoT enfocadas a Smart Campus usando la plataforma; a su vez se detallan los requerimientos funcionales y no funcionales de la aplicación Web de administración y cuáles de estos fueron cumplidos en la implementación final del prototipo.

La seguridad de la plataforma Web y en general de la plataforma IoT se encuentra fuera del alcance de este proyecto.

5.1 Arquitectura de Referencia

A continuación, se muestra la arquitectura de referencia, creada con base en las arquitecturas estudiadas durante la etapa de capacitación tecnológica y representa la plataforma Smart Campus desde el punto de vista del usuario, es decir, es una guía para los usuarios que deseen implementar una aplicación utilizando la Plataforma desarrollada.

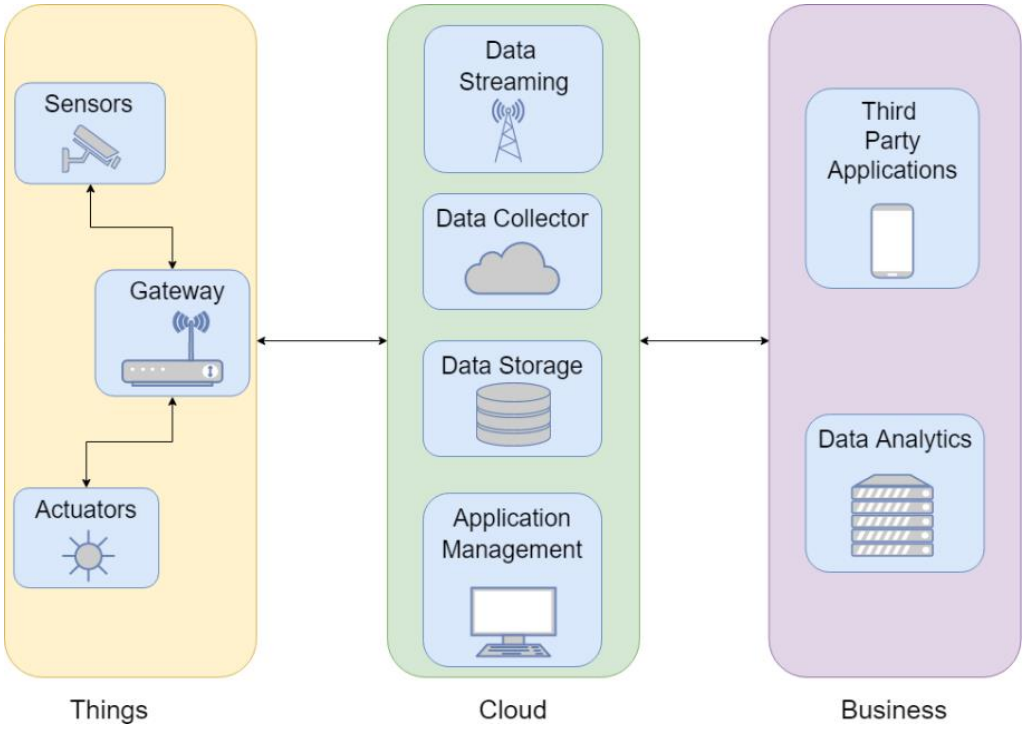

Figura 4. Arquitectura de referencia

Esta se compone por tres capas, la capa Things que representa la capa sensorial, y está compuesta por sensores, actuadores y gateways; la segunda es la capa Cloud donde se encuentran las aplicaciones de transmisión de datos, almacenamiento de información y gestión de aplicaciones; finalmente**,** la capa Business donde se encuentran las aplicaciones de terceros o externas y las aplicaciones de analítica de datos desarrolladas por los usuarios.

Para el desarrollo de una aplicación o caso de uso, los usuarios deben construir software para extraer información de sensores y recibirla en actuadores, esta información es enviada utilizando el software Gateway que se provee en la plataforma Smart Campus, donde se procesa y envía la información al software Backend o Cloud, el cual a su vez la expone para que otras aplicaciones de usuarios transformen o procesen la información recibida y, finalmente la muestren y den uso mediante aplicaciones externas o sea enviada a aplicaciones de analítica de datos.

5.2 Requisitos funcionales

Identificación del Requerimiento	RF01
Nombre	Autenticación de Usuarios
Descripción	La aplicación debe permitir a un usuario registrarse en la plataforma, iniciar sesión, recuperar su cuenta en caso de que olvide sus credenciales y modificar su perfil.
Prioridad	Alta

Identificación del Requerimiento	RF02
Nombre	Gestión de Aplicaciones
Descripción	La aplicación debe permitir a un usuario ya registrado consultar, editar y eliminar sus aplicaciones (casos de usos), crear nuevas y asignar Gateways a las mismas.
Prioridad	Alta

Identificación del Requerimiento	RF03
Nombre	Gestión de Gateways
Descripción	La aplicación debe permitir a un usuario ya registrado consultar la información y el estado, editar y eliminar sus Gateways y las propiedades de estos y registrar nuevos Gateways desde cero o copiando otros existentes.
Prioridad	Alta

Identificación del Requerimiento	RF04
Nombre	Gestión de Dispositivos
Descripción	La aplicación debe permitir a un usuario ya registrado consultar la información y el estado, editar y eliminar sus Dispositivos y las propiedades de estos y registrar nuevos Dispositivos desde cero o copiando otros existentes.
Prioridad	Alta

Identificación del Requerimiento	RF05
Nombre	Gestión de Procesos
Descripción	La aplicación debe permitir a un usuario ya registrado consultar, editar y eliminar sus Procesos y las propiedades de estos, registrar nuevos Procesos desde cero o copiando otros existentes, desplegarlos si son configurados para hacerlo o detenerlos en caso de que ya estén desplegados.
Prioridad	Alta
Identificación del Requerimiento	**RF06**
Nombre	Gestión de Usuarios

Descripción	La aplicación debe permitir a un usuario administrador consultar los usuarios de la plataforma, crear nuevos y eliminarlos.
Prioridad	Baja

Identificación del Requerimiento	RF07
Nombre	Visualización de información enviada
Descripción	La aplicación debe permitir a los usuarios ver la información enviada desde los Procesos y Dispositivos.
Prioridad	Alta

Identificación del Requerimiento	RF08
Nombre	Filtrado, ordenamiento y exportar información
Descripción	La aplicación debe permitir a los usuarios filtrar y ordenar la información de los objetos del sistema por sus atributos y exportar la misma a XLSX.
Prioridad	Media

Identificación del Requerimiento	RF09
Nombre	Visualización de estadísticas
Descripción	La aplicación debe permitir a los usuarios ver estadísticas del estado de la plataforma y la información que se envía por medio de esta.
Prioridad	Alta

Identificación del Requerimiento	RF10
Nombre	Notificaciones de cambios de estados
Descripción	La aplicación debe notificar a los usuarios los cambios de estados de sus Gateways y Procesos, gestionar las notificaciones que reciben y actualizar la información del estado de los Gateways/Procesos en la aplicación y sus respectivas estadísticas.
Prioridad	Alta

5.3 Requisitos no funcionales

Identificación del Requerimiento	RNF01
Nombre	Diseño Responsivo
Descripción	La aplicación debe tener un Diseño Responsivo de forma que la información se vea de forma agradable para diversos tamaños de pantalla

Prioridad	Alta

Identificación del Requerimiento	RNF02
Nombre	Diseño Intuitivo
Descripción	La aplicación debe tener un diseño intuitivo, de forma que esta muestra la información de una forma sencilla y comprensible para los usuarios y le permita navegar por las diferentes secciones fácilmente.
Prioridad	Alta

Identificación del Requerimiento	RNF03
Nombre	Alto rendimiento
Descripción	La aplicación debe ser rápida y no bloquearse con el flujo normal de navegación del usuario.
Prioridad	Alta

5.4 Especificaciones alcanzadas

En la siguiente tabla se detalla el estado final del cumplimiento de los requisitos funcionales y no funcionales del prototipo de la aplicación Web.

Tabla 1
Cumplimiento de Requerimientos

Requisito	Logrado	Estado final
RF01	*Sí*	*La aplicación permite al usuario registrarse, iniciar sesión, recuperar su cuenta y modificar su perfil.*
RF02	*Sí*	*La aplicación muestra al usuario sus Aplicaciones y le permite gestionarlas.*
RF03	*Sí*	*La aplicación muestra al usuario sus Gateways y le permite gestionarlos.*
RF04	*Sí*	*La aplicación muestra al usuario sus Dispositivos y le permite gestionarlos.*
RF05	*Sí*	*La aplicación muestra al usuario sus Procesos y le permite gestionarlos.*
RF06	*Sí*	*La aplicación permite a los administradores gestionar los Usuarios de la plataforma.*
RF07	*Sí*	*La aplicación permite a los usuarios ver los datos enviados por sus procesos.*
RF08	*Sí*	*La aplicación permite a los usuarios filtrar, ordenar y exportar la información de sus Aplicaciones, Gateways, Procesos y Dispositivos.*
RF09	*Sí*	*La aplicación muestra al usuario estadísticas del estado de sus Gateways y Procesos y de la información enviada por medio de estos.*
RF10	*Sí*	*La aplicación notifica a los usuarios cuando un Gateway o Proceso gestionado por la plataforma se activa o desactiva.*
RNF01	*Sí*	*La aplicación se ve y es usable para dispositivos de cualquier tamaño.*
RNF02	*Sí*	*La aplicación tiene un flujo sencillo, entendible y consistente.*

RNF03	*Sí*	*La aplicación responde rápidamente a las solicitudes del usuario y permite una navegación fluida por sus secciones.*

6. Desarrollo del Proyecto

6.1 Contexto del Proyecto

El objetivo central de este proyecto es el diseño de un prototipo de una aplicación Web para administrar una plataforma IoT orientada a Smart Campus por consiguiente cabe resaltar primero que este proyecto de grado se desarrolló paralelamente con tres trabajos de grado más que consisten en la solución Backend o Cloud, la solución de software para los Gateways y el despliegue de la arquitectura con el objetivo de facilitar la implementación de casos de uso relacionados con el Internet de las Cosas especialmente orientados a Smart Campus, como lo podría ser el control de la luminosidad de las instalaciones, el riego de zonas verdes controladas remotamente, control de temperatura en aulas de clase de manera centralizada y la recolección de diferentes datos como humedad, emisión de gases, cantidad de personas en un determinado lugar, geolocalización de inventario, entre otros.

6.2 Capacitación tecnológica

Durante esta etapa se realizó una breve investigación acerca de las Plataformas IoT existentes, posibles tecnologías a usar, selección de estas y posteriormente una capacitación en las tecnologías seleccionadas.

6.2.1 Arquitecturas IoT. Se realizó una revisión breve de las arquitecturas IoT más populares. En general se observó que estas están conformadas por varias capas:

- Capa sensorial: Conformada por los dispositivos que generan la información y están conectados a un intermedio entre estos y el Backend Cloud, estos dispositivos son comúnmente conocidos como Gateway.

- Capa Gateway: Puente entre los dispositivos hardware y el Backend Cloud. Recibe la información de los sensores, realiza un procesamiento pequeño para disminuir la carga del servidor centralizado y luego enviarla al servidor mediante protocolos cómo AMQP, MQTT, COAP o HTTP.
- Capa Cloud: Capa responsable del procesamiento (transformación o enriquecimiento), almacenamiento, control de seguridad y exponer la información.
- Capa de Aplicación: Permite gestionar los Gateways y Dispositivos de la Plataforma y consume la información existente en la Capa Cloud de acuerdo con la necesidad de los usuarios. Esta capa usualmente está conformada por una plataforma de administración brindada por la solución y demás aplicaciones creadas por los usuarios que consumen la información almacenada en la plataforma para construir casos de uso específicos.

Las arquitecturas estudiadas tienen diversas soluciones Software para cada una de las capas mencionadas previamente. En Azure, por ejemplo, es posible asignar a los dispositivos asociados a la plataforma propiedades, para permitir configurar los mismos o agregarles cierta meta dada útil para otras operaciones y la identificación del usuario.

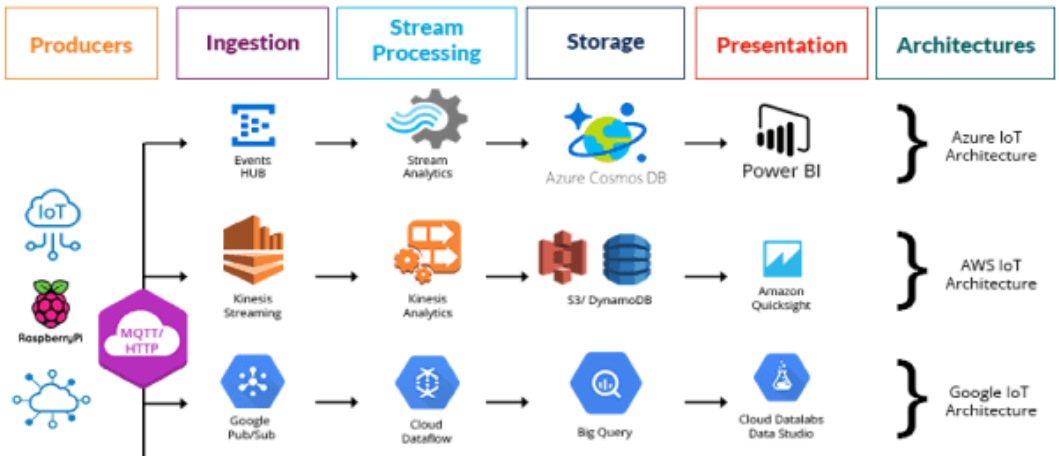

Figura 5. Componentes Software Arquitecturas IoT. Tomada de Singh. (2019)

En la Figura 5 se observan algunos componentes Software usados en las tres Plataformas IoT más conocidas, las cuales son: Azure, AWS y Google. Los Producers corresponden a la Capa Sensorial y a la Capa Gateway donde, usando Protocolos especiales para IoT se envía la información recibida a la Capa Cloud, la cual está compuesta por las subcapas de Ingestión, Stream Processing y Storage encargadas de recibir la información, procesar, analizarla, transformarla y almacenarla respectivamente. Finalmente, la capa de Presentación o Aplicación donde se puede gestionar la plataforma y visualizar la información de esta.

6.2.2 Tecnologías y herramientas. En esta subsección se hablará del lenguaje de programación, protocolos de comunicación, frameworks y librerías seleccionados para el desarrollo del proyecto, así como las alternativas que se consideraron para cada una de estas y sus características.

- Lenguajes de Programación:

Una tendencia del desarrollo de aplicaciones Web es la renderización del lado del cliente, esto permite reducir la carga de los servidores haciendo que la renderización ocurra utilizando los recursos de los clientes por medio del navegador Web. Los lenguajes basados en JavaScript son los ideales para cumplir este objetivo, por lo cual se decidió optar por alguno de ellos. Entre las opciones se encuentran Dart, Typescript, Elm, PureScript, CoffeScript y Vanilla Javascript. Se decidió optar por Typescript, pues es un lenguaje orientado a objetos, lo que permite reusar código mediante herencia, brinda flexibilidad usando polimorfismo, permite crear mejores abstracciones de objetos/elementos cotidianos por medio de clases o interfaces y evita algunos posibles errores por ser fuertemente tipado y compilado; además fue desarrollado por Microsoft por lo cual tiene muy buen soporte, documentación y una comunidad muy grande.

- Protocolos de Comunicación:

Para cumplir con los objetivos y requerimientos de la aplicación Web de administración es necesario dos tipos de comunicación con el servidor Cloud. La primera es comunicación HTTP básica con la cual la aplicación solicita la información necesaria al servidor, este le responde y el canal de comunicación se cierra inmediatamente, la información no es actualizada si esta cambia. Esta comunicación se realizó mediante peticiones HTTP REST realizadas asíncronamente, pues es un estándar para la Web y comparado con SOAP (Simple Object Access Protocol) es mucho más ligero y eficiente.

El segundo tipo de comunicación se realiza utilizando un protocolo Publish & Subscribe, y permite al cliente subscribirse a un tópico y continuar recibiendo la información por un canal de comunicación que permanece abierto recibiendo actualizaciones y mensajes siempre que el servidor envíe información a este tópico a el Broker de mensajería, que es el intermediario y se encarga de recibir y despachar la información a cada uno de los clientes. Para esto se utilizó MQTT (Message Queuing Telemetry Transport) sobre WebSockets, pues MQTT es uno de los protocolos Publish & Subscribe más populares y ligeros y, WebSockets es necesario para abrir la conexión bidireccional entre los productores/subscriptores en navegadores y, permitir compatibilidad entre diferentes versiones de estos.

- Framework:

Para seleccionar el Framework para desarrollar este prototipo se partió de las decisiones previas: construir la aplicación con renderización del lado del cliente y usando Typescript.

Hay una gran variedad de Frameworks que soportan estos requisitos, entre los que se destacan ReactJS, Angular y VueJS. Luego de un análisis se seleccionó Angular, pues soporta nativamente Typescript sin necesidad de configuración extra; porque es recomendable para aplicaciones Web

cuya base de código se espera sea incrementada y gestionada por varias personas de forma sencilla, como se podría dar más adelante para incluir nuevas funcionalidades al aplicativo sin demasiado esfuerzo; porque fue desarrollado por Google lo cual implica un muy buen soporte y cumplimiento de estándares de calidad y una comunidad muy grande; este Framework es a su vez orientado a componentes, lo que permite reutilizar y reducir el código.

También por defecto mediante el Angular CLI se puede generar un proyecto con una configuración inicial lista para producción lo cual reduce la cantidad de tiempo necesaria para iniciar el desarrollo de este y también posee un gran ecosistema con muchas librerías.

- Librerías:

Para aplicaciones del lado del cliente es muy importante manejar la asincronía, de modo que operaciones que pueden tomar cierto tiempo o que dependen de respuestas de otros componentes software sean realizadas en segundo plano, para evitar que el hilo principal de la aplicación sea bloqueado y esta se congele para el usuario final por un momento, o incluso indefinidamente en caso de un mal manejo de errores. Para esto se seleccionó la librería RxJS, una librería para programación reactiva usando Observables (objeto que emite información y eventos para que un Observer se subscriba, reciba esta información y reaccione ante ella) para permitir de forma sencilla construir operaciones asíncronas. Esta librería viene incluida en Angular y está optimizada para funcionar con este framework.

Para dotar de capacidades responsivas a la aplicación Web se seleccionó una pequeña librería llamada Angular Flex Layout, con la cual mediante directivas se especifica el tamaño y demás propiedades y comportamiento de elementos HTML de acuerdo con los diferentes tamaños de pantalla permitiendo que los usuarios puedan acceder a la información y que esta se vea agradable desde cualquier dispositivo.

Los estilos del prototipo de la aplicación se basaron en las guías y patrones de Materialize, creada por Google e implantada en gran variedad de aplicaciones, como lo son el Sistema Operativo Android. Usando a su vez la librería Angular Material, optimizada para el framework seleccionado, que provee componentes personalizables estilizados con este patrón que permiten acelerar el proceso de desarrollo y reducción de duplicación de código.

6.2.3 Capacitación tecnológica. En esta etapa se realizó una capacitación en el lenguaje de programación, framework y librerías seleccionadas. Mediante cursos online y leyendo documentación publicada por los creadores se aprendió como desarrollar un prototipo de una aplicación Web que cumpla los requerimientos funcionales y no funcionales establecidos siguiendo los mejores estándares, como por ejemplo investigando como organizar y estructurar los archivos, páginas y componentes de la aplicación para permitir optimizar el uso de recursos y hacer una base de código fácil de mantener e incrementar, aprovechando a su vez al máximo las características que brindan las herramientas seleccionadas.

6.3 Definición de casos de uso

Se construyeron diagramas de casos de uso con el fin de especificar el comportamiento del prototipo teniendo en cuenta la interacción con el usuario. En primera instancia para esta labor se identificaron los actores principales del sistema.

6.3.1 Definición de actores. Los actores identificados son:

- Administrador
- Superusuario

Tabla 2
Descripción de actor Administrador

Actor:	**Administrador**
Descripción:	Usuario con permisos para ver y gestionar su perfil, aplicaciones, gateways, dispositivos, datos y notificaciones.
Tipo:	Primario

Tabla 3
Descripción de actor Superusuario

Actor:	**Superusuario**
Descripción:	Usuario con permisos para ver y gestionar todos los usuarios, aplicaciones, gateways, dispositivos, datos y notificaciones de la plataforma.
Tipo:	Primario

6.3.2 Definición de casos de uso. Una vez determinados los actores del sistema, se procedió a la identificación y diseño de los casos de uso.

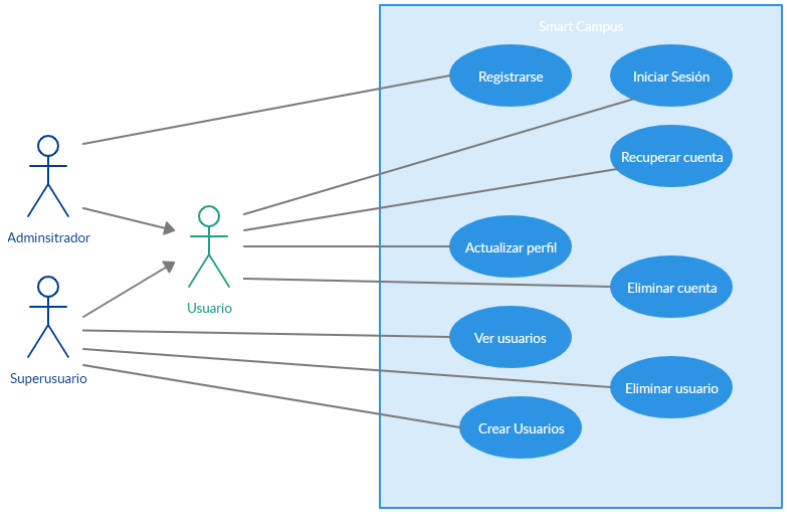

Figura 6. Diagrama Casos de Uso Módulo Usuario.

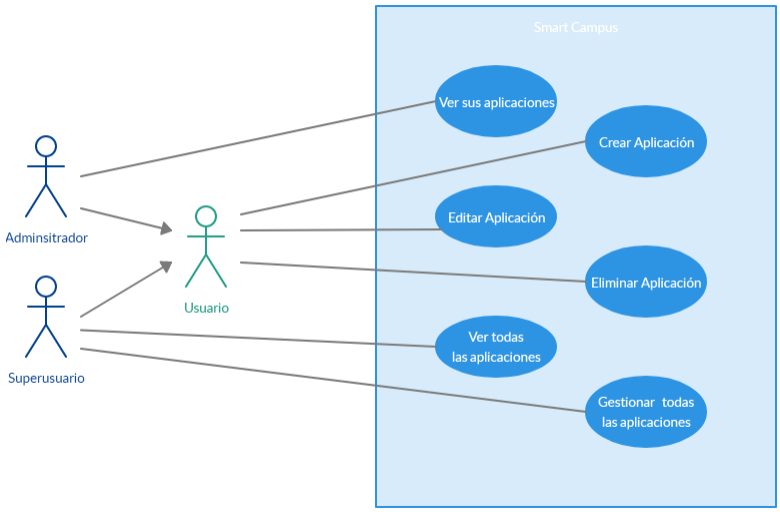

Figura 7. Diagrama Casos de Uso Módulo Aplicaciones.

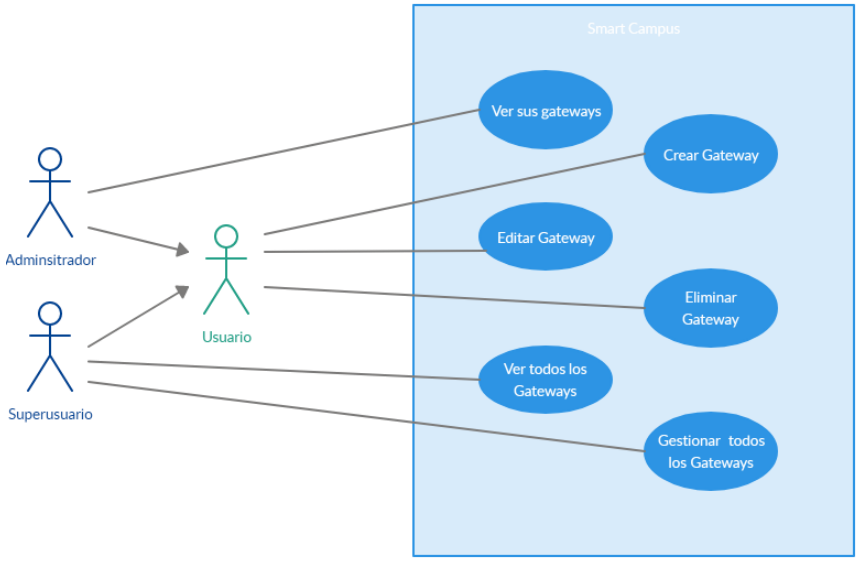

Figura 8. Diagrama Casos de Uso Módulo Gateways.

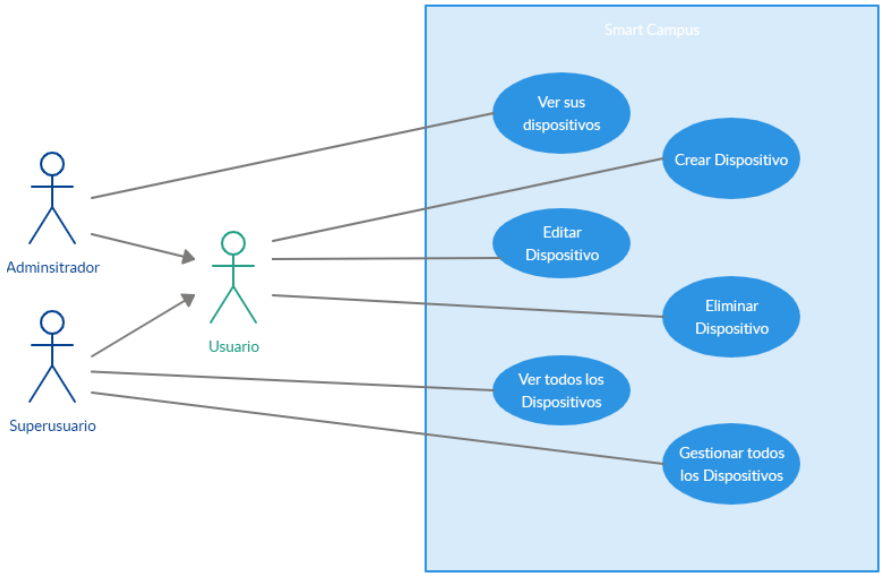

Figura 9. Diagrama Casos de Uso Módulo Dispositivos.

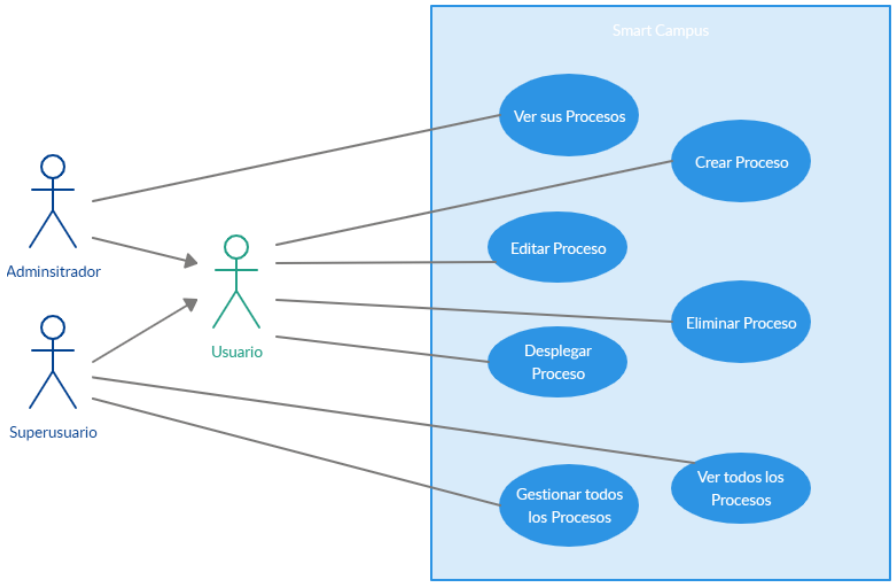

Figura 10. Diagrama Casos de Uso Módulo Procesos.

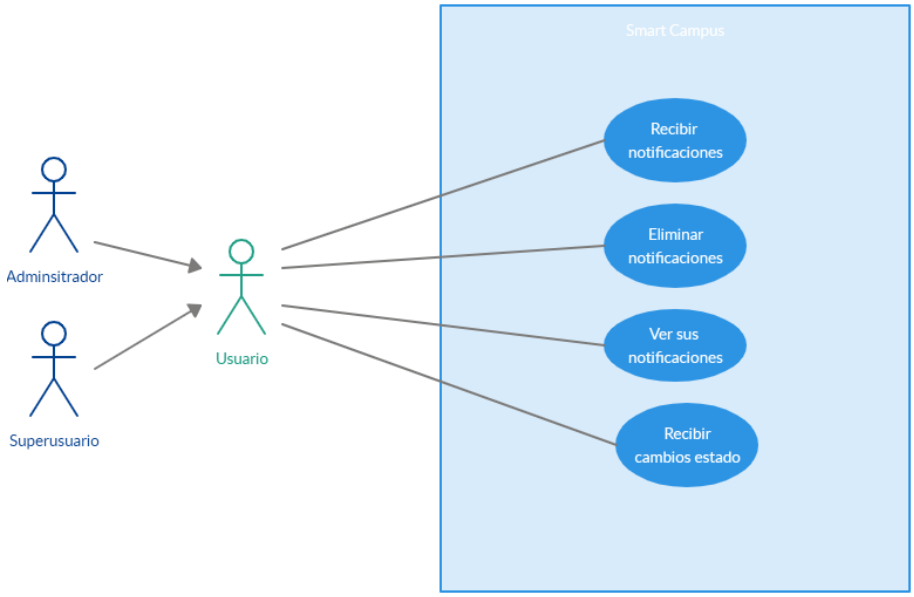

Figura 11. Diagrama Casos de Uso Módulo Notificaciones.

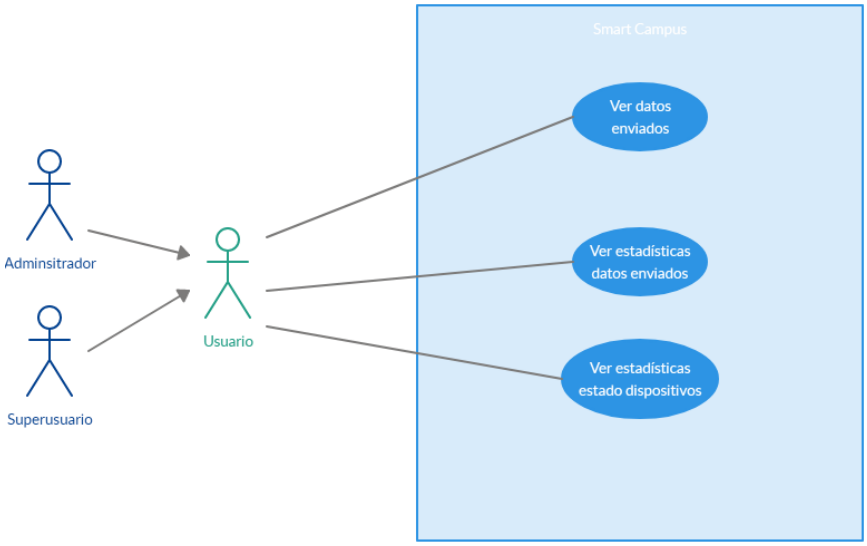

Figura 12. Diagrama casos de Uso Módulo de Datos y Estadísticas.

6.4 Definición de la arquitectura

Como se mencionó previamente, luego de la etapa de capacitación tecnológica se llevó a cabo un prototipado incremental, donde en tres ciclos se definió la arquitectura de la plataforma Smart Campus y de la aplicación Web de administración, para posteriormente realizar un prototipo de la misma, verificar el cumplimiento de los objetivos planteados para esta etapa y, repetir el ciclo revisando y haciendo mejoras en caso de ser necesario a la arquitectura de la solución, al prototipo y en caso de que estos estuvieran correctos se continuaría agregando funcionalidades hasta completar las especificaciones y los objetivos del proyecto.

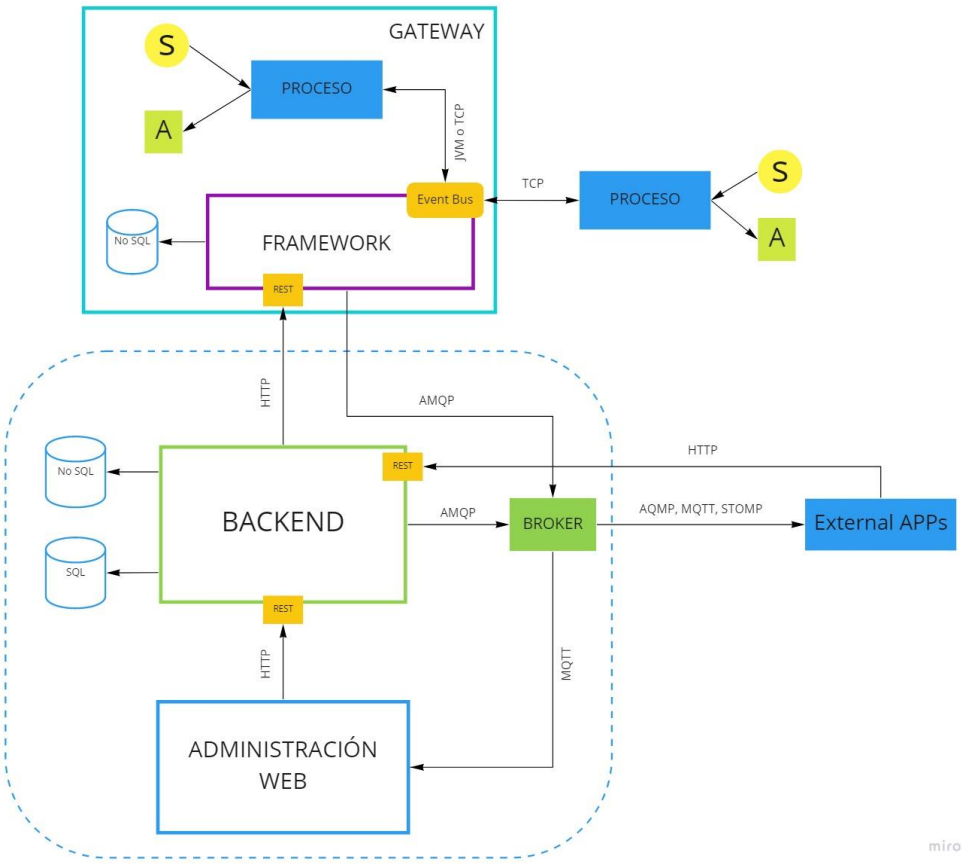

Figura 13. Arquitectura Smart Campus.

Vale la pena enfatizar que esta solución se desarrolló en conjunto con otros tres proyectos de grado, los cuales se encargaron de otros módulos Software para conformar la Infraestructura requerida para Smart Campus, pues esta es una tarea compleja principalmente teniendo en cuenta la necesidad de hacer de esta una plataforma escalable y que permita a los usuarios crear sus propias aplicaciones y casos de uso; estos proyectos son:

- **Gateway:** Donde se presenta el diseño de un framework de software extensible que permite a dispositivos tipo Gateway la comunicación y el almacenamiento de datos producidos y recibidos por sensores y/o actuadores, al mismo tiempo que provee la capacidad de conectarse con plataformas IoT enfocadas en Smart Campus (Gutiérrez, 2019).

- **Backend:** El cual se desarrolló con una arquitectura de microservicios de alta disponibilidad elaborada en Java usando el framework Spring Boot, que permite la integración de dispositivos y gateways a través de un broker y gracias a su escalabilidad puede manejar grandes volúmenes de datos que se generen a partir de estos y, a través de un API REST exponerlos para el uso que se les quiera dar, además, cuenta con una unidad de persistencia para almacenar la información de los elementos que estén presentes en la infraestructura (Arias y Estupiñan, 2019).
- **Despliegue:** Donde se presenta el diseño de una infraestructura software para el despliegue de una plataforma IoT en una infraestructura Cloud de alta disponibilidad en un entorno distribuido con el fin de proveer un entorno que pueda soportar cantidades masivas de solicitudes permitiendo una escalabilidad horizontal en la infraestructura hardware. (Rojas, 2019).

A continuación, se explicará a fondo la aplicación Web de administración que es el aporte realizado en este proyecto.

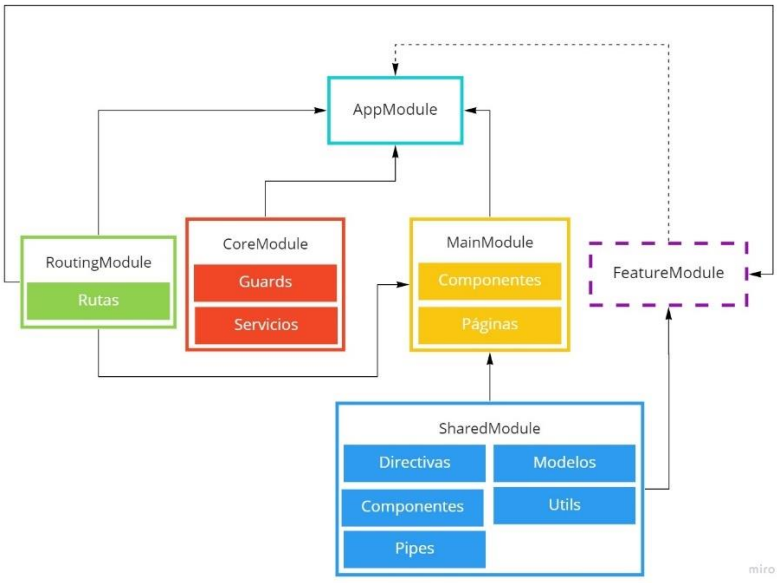

Figura 14. Arquitectura Aplicación Web.

Como se observa en la imagen, se planteó una arquitectura por módulos donde cada uno tiene funcionalidades específicas. Estos son:

- **AppModule:** Módulo central de la aplicación, en él se importan todos los demás módulos, contiene el componente central de la aplicación el cual es renderizado al inicio. Es el encargado de arrancar la aplicación.
- **RoutingModule:** Expone y configura las rutas para las páginas y módulos hijos de la aplicación y las condiciones que se deben cumplir para poder acceder y cargar los mismos.
- **CoreModule:** En este módulo se proveen los servicios (encargados de almacenar información utilizada en diferentes secciones de la aplicación y donde se realizan las peticiones HTTP y subscripciones), también las Guards, que son las encargadas de determinar de acuerdo con el estado de la aplicación si es posible acceder a una página o no para, basado en la configuración de rutas redireccionar al usuario a la sección correspondiente que si sea accesible.
- **SharedModule:** Expone modelos (clases e interfaces con lógica de domino), clases utilitarias implementadas, directivas (encargadas de dotar de atributos y funcionalidades a elementos HTML ya existentes u otros componentes), componentes y pipes (funciones encargadas de manipular la información para renderizarla de una forma específica).
- **MainModule:** Contiene los componentes implementados de forma no genérica para una funcionalidad específica y las páginas usadas actualmente por la aplicación para la gestión de la plataforma IoT. No es cargado de manera perezosa pues es el punto de entrada de la aplicación.
- **FeatureModule:** Módulos por funcionalidad que pueden ser creados a futuro para extender la aplicación Web, la idea es que los módulos adicionales se carguen de manera

perezosa, es decir solo se descargue el contenido de estos cuando el usuario ingresa en alguna sección incluida en ellos; esto con el objetivo de hacer que el tiempo de carga inicial de la aplicación no se vea incrementado por estas nuevas funcionalidades.

Los componentes de Angular están generalmente compuestos por un Template, donde se modela el HTML que se renderiza, la lógica o Controlador del componente y los estilos, pues este framework por defecto hace que el alcance de estos estilos sea aplicado únicamente dentro del componente, para evitar que afecten los de otros elementos del aplicativo. A los componentes, también se les inyecta servicios, estos son clases que se utilizan para almacenar información a nivel global, pues son singletons (solo hay una instancia de cada clase) y son instanciados por defecto por el framework, por lo cual estas instancias son pasadas a los componentes (inyectados) para utilizar esta información o sus métodos, entre estos llamados HTTP.

A continuación, se presenta la interacción entre el Componente (lógica), el Template y los Servicios.

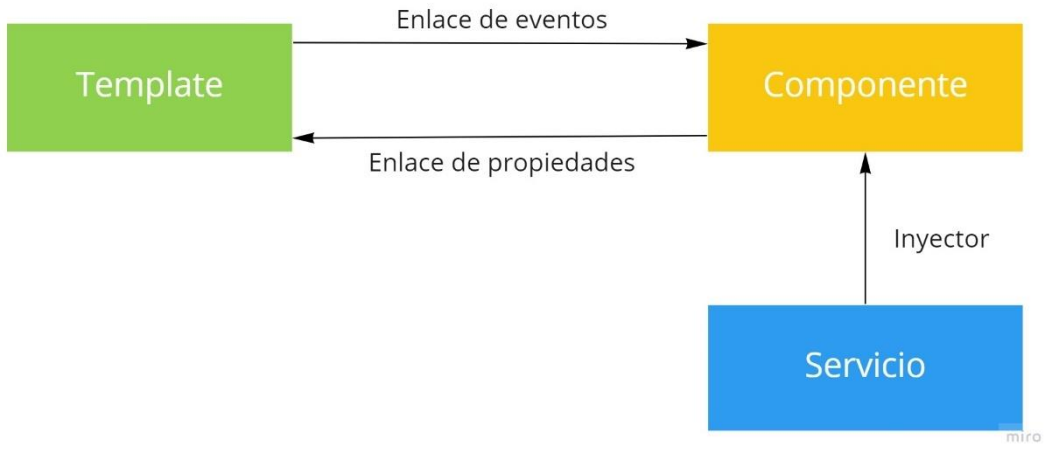

Figura 15. Interacción entre Templates, Componentes y Servicios.

La lógica de los componentes permite a la vista crear un enlace con las propiedades de los objetos contenidos dentro del mismo, de modo que si los objetos o sus propiedades cambian la vista es actualizada y el usuario puede ver automáticamente la información sin necesidad de que la página sea refrescada; también hay un enlace en la otra dirección, donde el template mediante eventos notifica que ocurrió un cambio o que el usuario interactuó con algún elemento y ejecuta una función. Las propiedades enlazadas pueden estar declaradas y almacenadas dentro del componente para uso local, o en un servicio para compartir esta información y enlaces entre distintos componentes.

Esta arquitectura se ve implementada usando la siguiente estructura de carpetas y módulos.

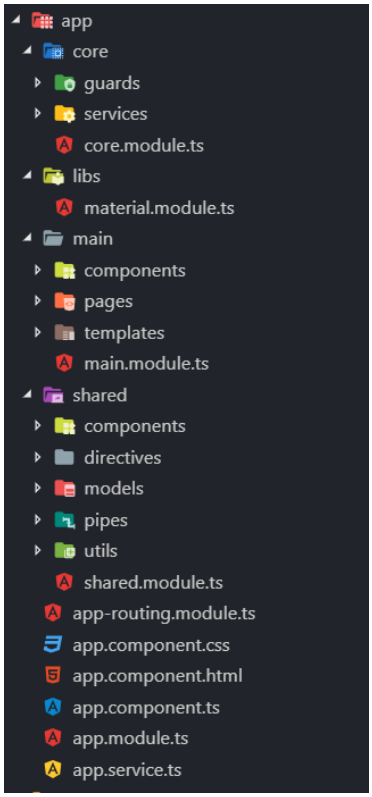

Figura 16. Estructura de carpetas y módulos.

Donde se observa el módulo *Core* definido dentro de la carpeta homónima en el archivo *core.module.ts*, allí son provistos todos los *guards* y *services* de la aplicación. Dentro del folder

lib se encuentra el módulo *Material* donde se importan y exportan para el uso de las diferentes vistas todos los componentes utilizados de la librería Angular Material, para estilizar la aplicación.

Posteriormente, dentro de la carpeta *main* se encuentra el módulo homónimo que contiene todos los componentes de uso específico (no genéricos), páginas y el template de la aplicación.

Dentro de *shared,* se encuentran todos los componentes genéricos, directivas, clases, interfaces, pipes y utilitarios de la aplicación.

Finalmente el *app.routing.module.ts* contiene la definición de las rutas de la aplicación y el *app.component (css, html & ts)* que son los archivos del componente inicial de la aplicación, el cual es renderizado cuando se carga la aplicación. El *app.module.ts* que es el módulo principal de la aplicación y donde todos los demás módulos deben ser importados, contiene el componente root y el *AppService (app.service.ts)* que es el Servicio principal de la aplicación, dónde se incluyen funciones y se almacena información global, como por ejemplo, el usuario autenticado.

6.5 Prototipado

Para el prototipo de la aplicación Web de administración de la plataforma Smart Campus, de acuerdo con los requerimientos (véase capítulo 5) y casos de uso (véase sección 6.2) especificados se dividieron las funcionalidades en diferentes vistas o páginas de la aplicación, las cuales se listan a continuación.

Tabla 4
Páginas del prototipo

Página	Descripción
Inicio de sesión	*Permite al usuario autenticarse en la aplicación.*
Registro de Usuario	*Permite al usuario registrarse (crear un usuario) en la aplicación.*
Recuperar Contraseña	*Permite al usuario ya registrado recuperar su contraseña si la olvida.*
Dashboard	*Permite al usuario ver las estadísticas mediante gráficas de sus dispositivos y procesos y consultar el histórico de datos enviados por estos.*
Aplicaciones	*Permite ver las aplicaciones del usuario, ordenarlas, filtrarlas, eliminarlas, exportar su información a XLSX y crear nuevas.*
Aplicación	*Permite ver la información detallada de una aplicación, editarla, asignar y desasignar Gateways a la misma.*

Gateways	*Permite ver los Gateways del usuario y el estado de estos, ordenarlos, filtrarlos, eliminarlos, exportar su información a XLSX, crear nuevos Gateways desde cero o copiando los atributos de alguno ya existente*.
Gateway	*Permite ver la información detallada de un Gateway, su estado, sus propiedades, Procesos y Dispositivos asociados, editarlo, editar y crear propiedades para el mismo y eliminar los Dispositivos y Procesos asociados a este.*
Procesos	*Permite ver los Procesos del usuario y sus estados, ordenarlos, filtrarlos, eliminarlos, exportar su información a XLSX, crear nuevos Procesos desde cero o copiando los atributos de alguno ya existente, desplegarlos si están configurados para ser desplegados por el sistema, o detenerlos si ya están corriendo.*
Proceso	*Permite ver la información detallada de un Proceso, su estado, sus propiedades, editarlo y editar y crear propiedades para el mismo.*
Dispositivos	*Permite ver los Dispositivos del usuario, ordenarlos, filtrarlos, eliminarlos, exportar su información a XLSX, crear nuevos Dispositivos desde cero o copiando los atributos de alguno ya existente.*
Dispositivo	*Permite ver la información detallada de un Dispositivo, sus propiedades, editarlo y editar y crear propiedades para el mismo.*
Usuarios	*Permite únicamente al Superusuario del Sistema, ver la información de todos los Usuarios de la plataforma, ordenarlos, filtrarlos, eliminarlos y exportar la información de estos a XLSX y crear nuevos usuarios.*
Notificaciones	*Permite ver las notificaciones enviadas para el usuario (acerca de actualizaciones del estado de sus Gateways y Procesos), leerlas a detalle, ordenarlas, filtrarlas, eliminarlas y, recibir como alertas las notificaciones entrantes*
Usuario	*Permite ver información del perfil, eliminar la cuenta y cerrar sesión.*

Como se mencionó, en la etapa de Prototipado incrementalmente se implementaron los requisitos especificados hasta llegar a cumplir con todos. Toda la aplicación Web se diseñó para que fuera responsiva, es decir sus elementos se acomodan para que pueda ser navegable correctamente tanto en dispositivos móviles como en computadores.

A continuación, se muestra el estado final de las vistas respectivas de la aplicación.

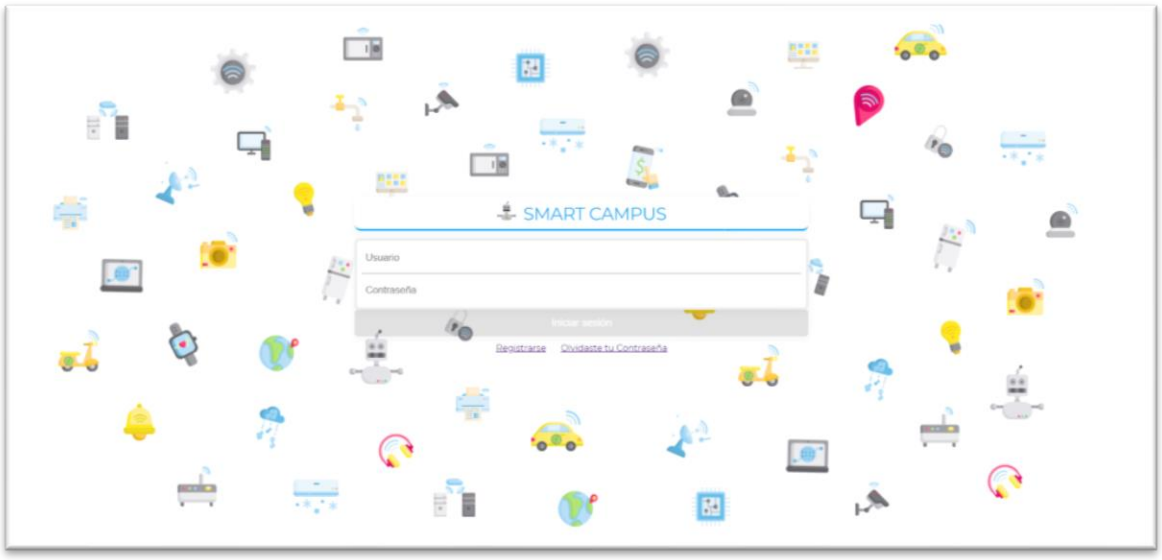

Figura 17. Vista de Inicio de sesión.

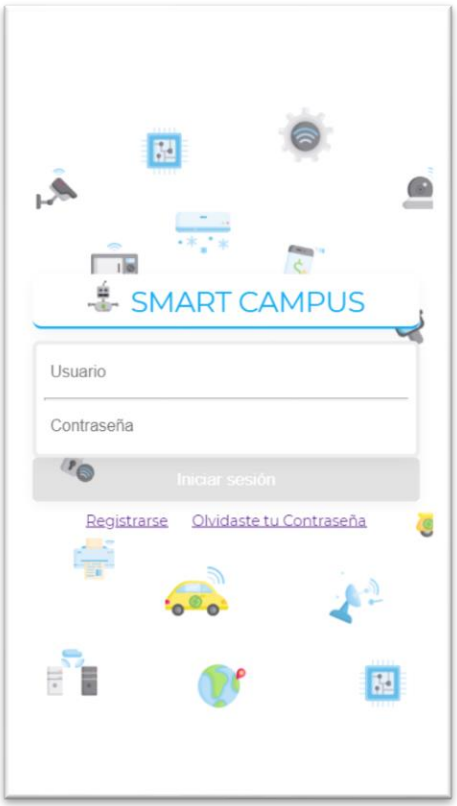

Figura 18. Vista de Inicio de sesión dispositivo móvil.

Como se puede observar en esta pantalla el usuario puede iniciar sesión, introduciendo su usuario y su contraseña si ya se encuentra registrado en el sistema, o, puede darse de alta en caso de que no lo esté. En la figura 19 se muestra a la izquierda de forma acercada el formulario de registro y a la derecha la información que debe ser introducida para poder recuperar la contraseña en caso de que esta sea olvidada, en este caso, solo es requerido el correo con el que el usuario fue registrado y, en un instante se envía un correo electrónico a esta cuenta con una nueva contraseña generada aleatoriamente.

Estos formularios validan que la información requerida sea insertada, cumpliendo con una longitud mínima, que las direcciones de correo introducidas sean válidas, el campo de confirmar contraseña sea igual que la contraseña introducida previamente. Esto para evitar que el usuario realice múltiples peticiones al servidor cuando se sabe que la operación que está realizando va a fallar pues la información introducida es incorrecta.

Figura 19. Vista detallada de los formularios de Registro y Cambio de contraseña.

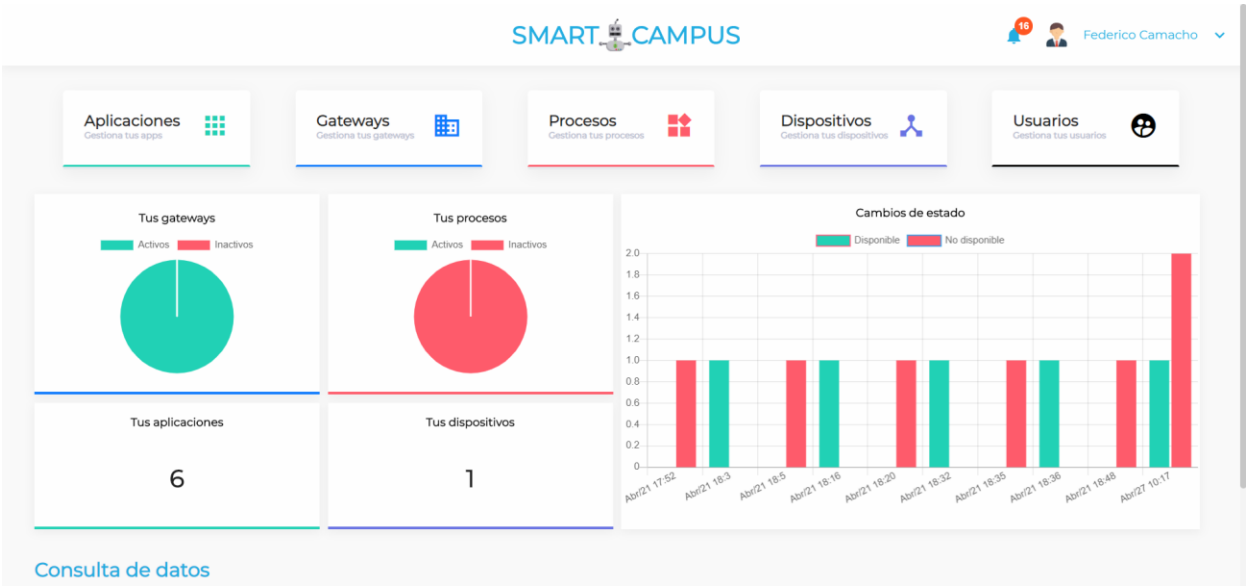

Figura 20. Dashboard de la Aplicación Web.

Luego de iniciar sesión o registrarse satisfactoriamente, el usuario es redirigido a la página de inicio o Dashboard de la plataforma, donde se observa en la barra de navegación en la parte superior el logotipo de la aplicación, el cual es un robot con el símbolo de la universidad representando el potencial de automatización y la gran transformación tecnológica que se puede presentar con el uso de esta plataforma. También se observa un icono de una campana el cual muestra la cantidad de notificaciones no leídas que tiene el usuario, las cuales le alertan de los cambios de estado de sus procesos y Gateways.

A la derecha se encuentra el nombre del usuario autenticado en la aplicación y permite, ver más información acerca del perfil del usuario, editar su información, eliminar su cuenta y cerrar sesión, lo cual se puede ver en la figura 21. Junto con las vistas de inicio de sesión, registro y recuperación de contraseña contribuye a que se cumpla a cabalidad el Requisito funcional *RF01 Autenticación de usuarios.*

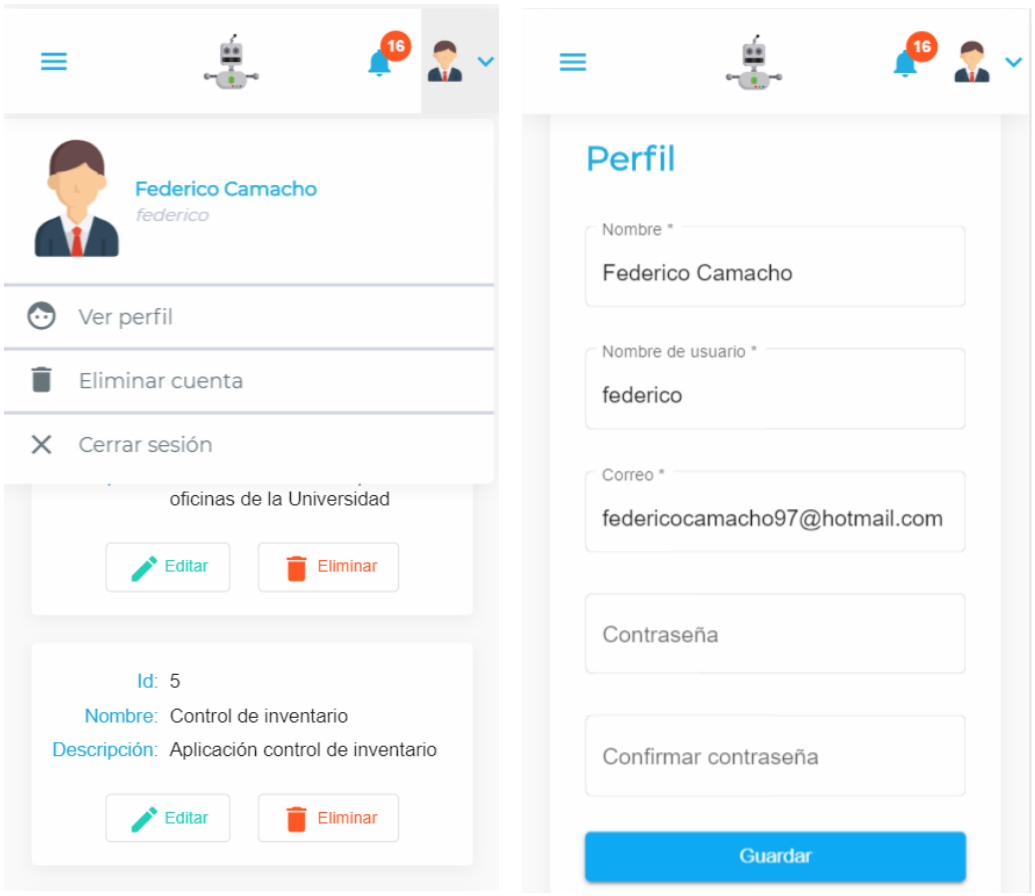

Figura 21. Tarjeta de gestión de usuario y edición de Perfil.

También en esta página de inicio (Dashboard) se pueden observar ciertas gráficas de administración, de estadísticas de datos y la posibilidad de consultar la información enviada por los dispositivos del usuario de acuerdo con diversos filtros, esto se explicará más adelante cuando se hable de los respectivos requisitos funcionales que cubren esta funcionalidad.

A su vez, se puede observar en las figuras 22 y 23, que es posible desde esta y las demás páginas de la aplicación Web navegar a otras secciones de administración del aplicativo, con el objetivo de reducir el número de clics necesarios para que el usuario pueda cambiar de sección y, hacer que este se oriente mejor en la plataforma.

Figura 22. Opciones de navegación para navegadores web.

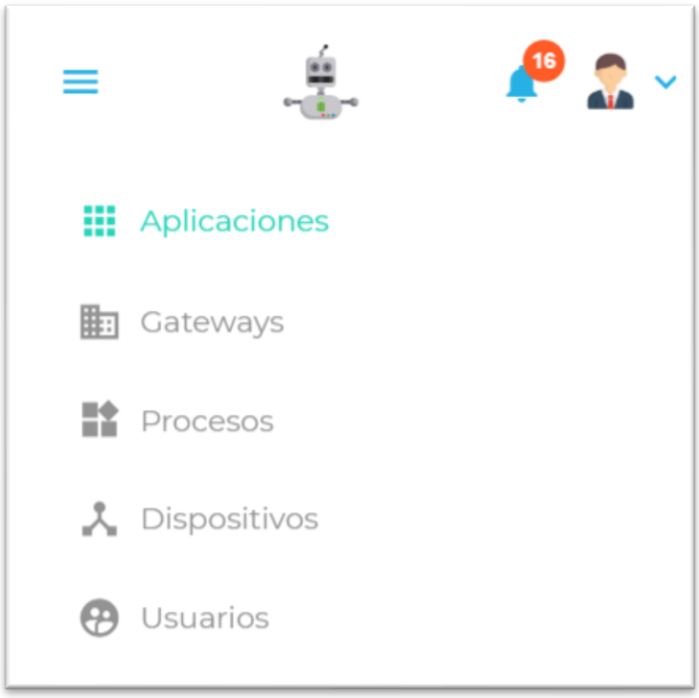

Figura 23. Opciones de navegación para dispositivos móviles.

En general la administración de la plataforma se dividió en 6 módulos. Gestion de Aplicaciones, Gateways, Procesos, Dispositivos, Usuarios (solo visible para el Superusuario) y Notificaciones, estas secciones son suficientes para cumplir los demás requerimientos establecidos y a demás dividir las acciones de esta forma permite que el usuario pueda navegar más fácilmente en la aplicación y que sea más sencillo administrar los elementos que pertenecen al Smart Campus.

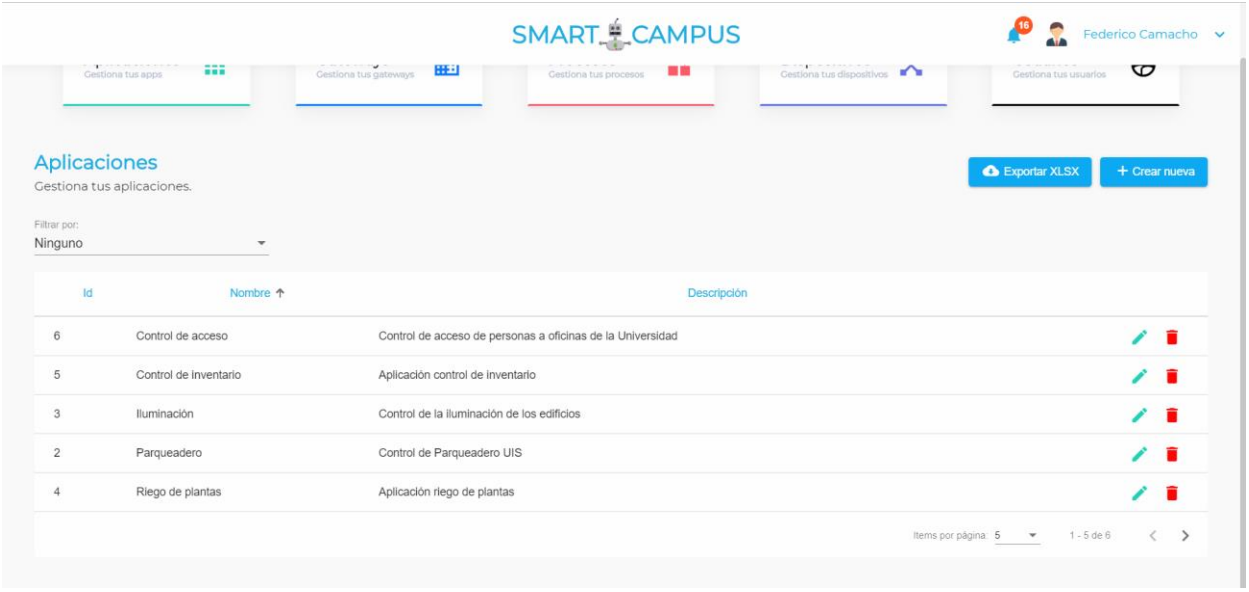

Figura 24. Vista de Aplicaciones.

Como se puede observar en la figura 24, para administrar las Aplicaciones se desarrolló esta vista, donde se permite al usuario ver sus aplicaciones y su descripción, crear una nueva aplicación, editarla, eliminarla, filtrar sus aplicaciones de acuerdo con Id, nombre o descripción, ordenarlas por alguno de sus atributos (en este ejemplo el nombre) y exportar la información de las aplicaciones en un archivo .XLSX. También la tabla muestra un número reducido de aplicaciones (5, 10 o 20 a elección del usuario) para no saturar al usuario con mucha información siendo presentada al mismo tiempo.

Al seleccionar el botón de edición de una aplicación, se abre una página en donde se muestra un formulario para modificar todos los atributos de esta y asignar, o desasignar Gateways. Esta vista también es utilizada, pero con toda la información en blanco cuando se está creando una nueva aplicación y puede ser observada en la figura 25. La tabla de Gateways asignados tiene las mismas funcionalidades que la tabla de aplicaciones (ordenamiento, filtrado, y paginación).

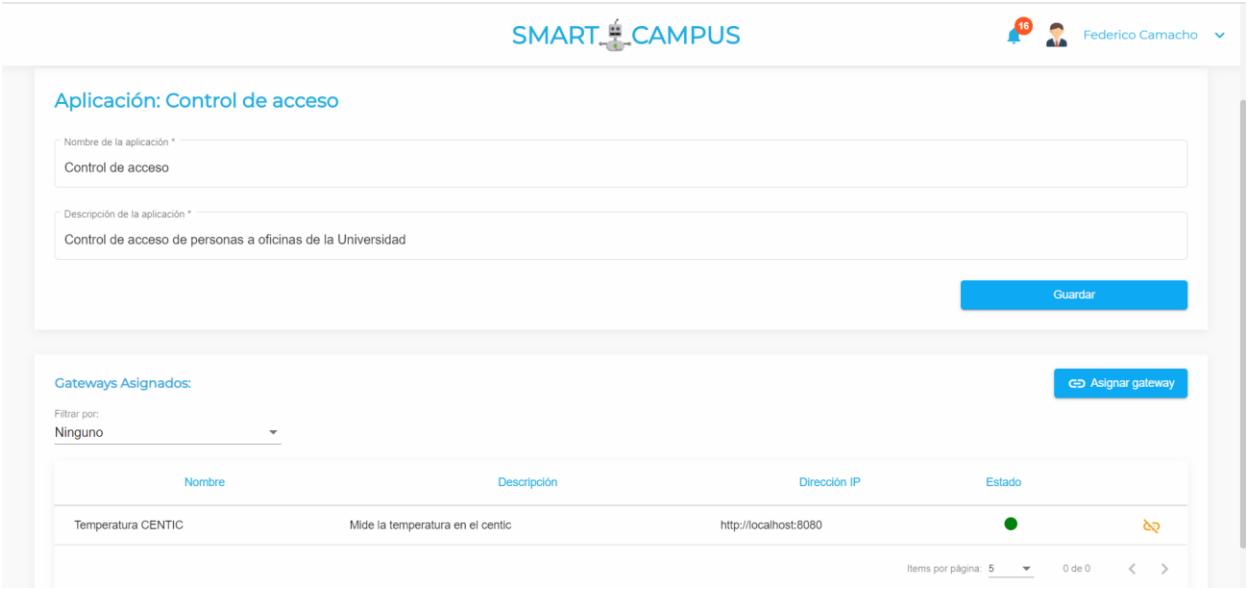

Figura 25. Vista de edición de Aplicación.

Con la vista de aplicaciones y la vista de edición/creación de aplicación y todas las características implementadas en estas se dio cumplimiento al requerimiento funcional RF02 *Gestión de Aplicaciones*.

Para mantener la consistencia entre vistas y hacer una experiencia de usuario más amigable, las demás secciones se desarrollaron de forma similar, mostrando la información con tablas y formularios estilizados similarmente, por ejemplo, en la figura 26 se muestra la vista de administración de Gateways, donde se permite al usuario ordenar por los datos que este necesite, filtrar por Id, nombre, descripción, dirección IP, estado, aplicación a la que pertenece el Gateway; editarlo, eliminarlo, crear nuevos desde cero o con base en Gateways ya existentes (copiando sus propiedades).

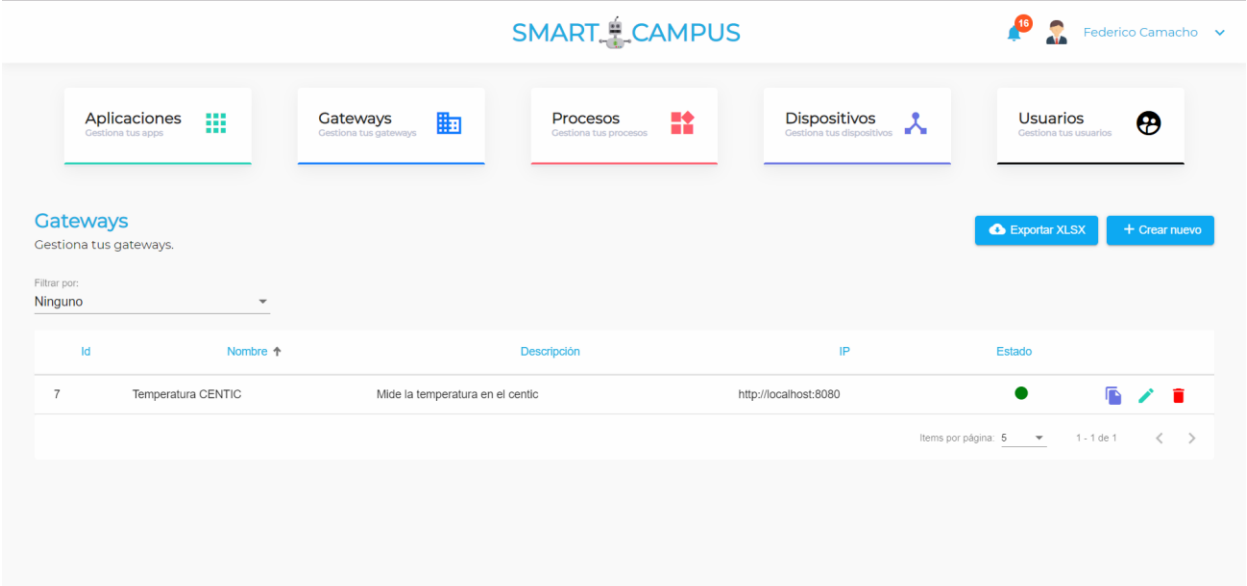

Figura 26. Vista de Gateways.

También al presionar el botón de edición se abre una nueva página como se muestra en la figura 27, donde se permite al usuario editar los atributos básicos de los Gateways, o sus propiedades, ver sus procesos y dispositivos y eliminarlos en caso de que lo deseen, como se puede observar en las figuras 27 y 28 y con estos se cumple el Requisito Funcional RF04 *Gestión de Gateways*.

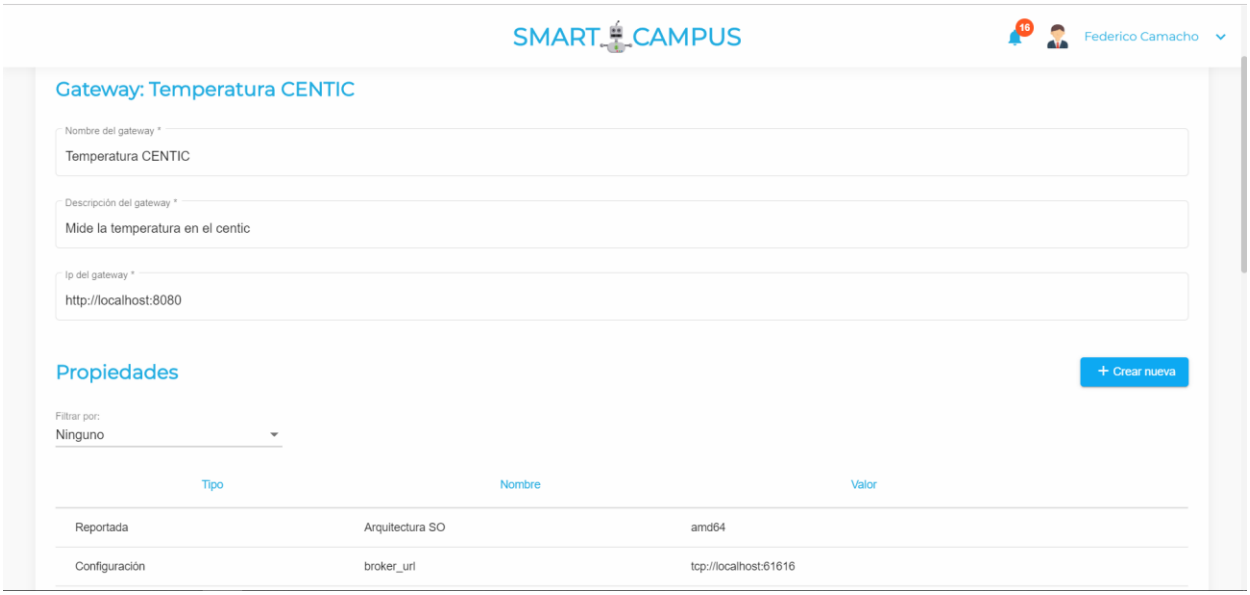

Figura 27. Vista de edición de Gateway.

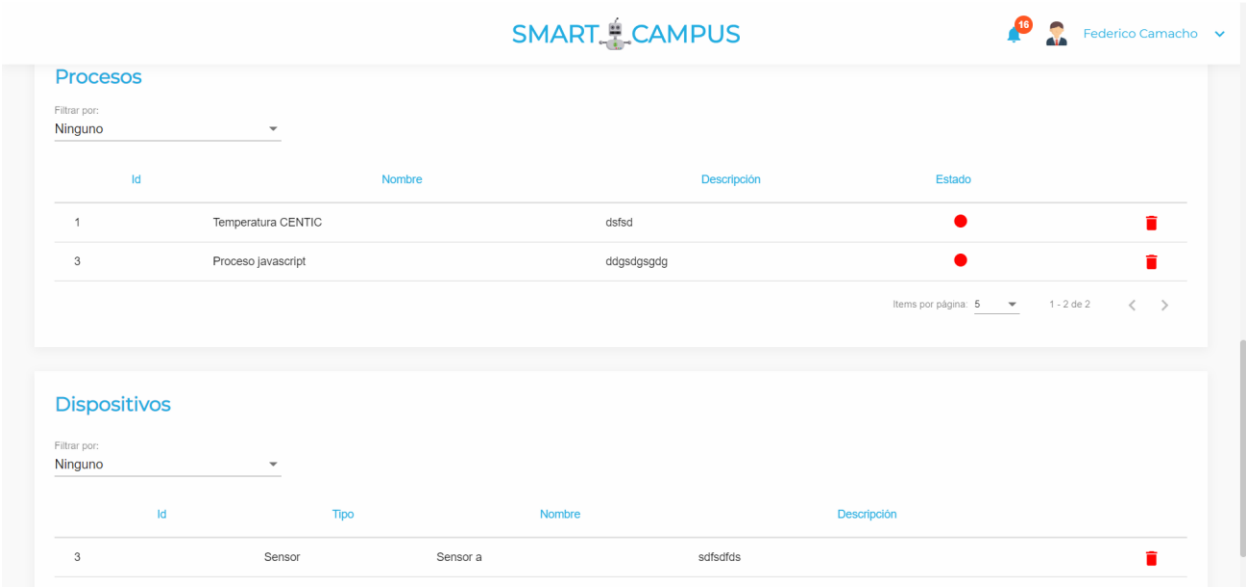

Figura 28. Vista de edición de Gateway Parte II.

Igualmente se implementó la administración de procesos, con el agregado de que, para los procesos configurados para ser desplegados es posible en el listado de procesos desplegarlos, detenerlos si ya están corriendo o reiniciarlos, como se puede observar en la figura 29. La vista para editar procesos y sus propiedades se observa en la figura 30.

Los procesos en la arquitectura representan las unidades de código que manejan la lógica de los casos de uso de los usuarios.

Por ejemplo, un Proceso es el código en el que se recopila información sobre los sensores de temperatura de una sala o una sección de esta, para enviar la información al Gateway al que está asociado y/o el sensor que envía la información para ser distribuida por medio de toda la plataforma y que esta sea visible por el usuario en la aplicación Web cuando la consulta mediante peticiones HTTP al servidor y muchas más cosas; pues al ser desarrollado por el usuario este tiene opciones casi ilimitadas y la plataforma además le provee una manera sencilla de enviar la información a otros Gateways y Procesos para ejecutar acciones de acuerdo con las medidas recibidas, e incluso con la programación de procesos orientados hacia actuadores puede recibir

esta información y realizar acciones de acuerdo a esta; por ejemplo si la temperatura es muy alta se considera que hay un incendio en una sala y por ende se deben encender los aspersores del lugar, lo que evitaría daños catastróficos mientras se espera de la respuesta humana ante esta situación.

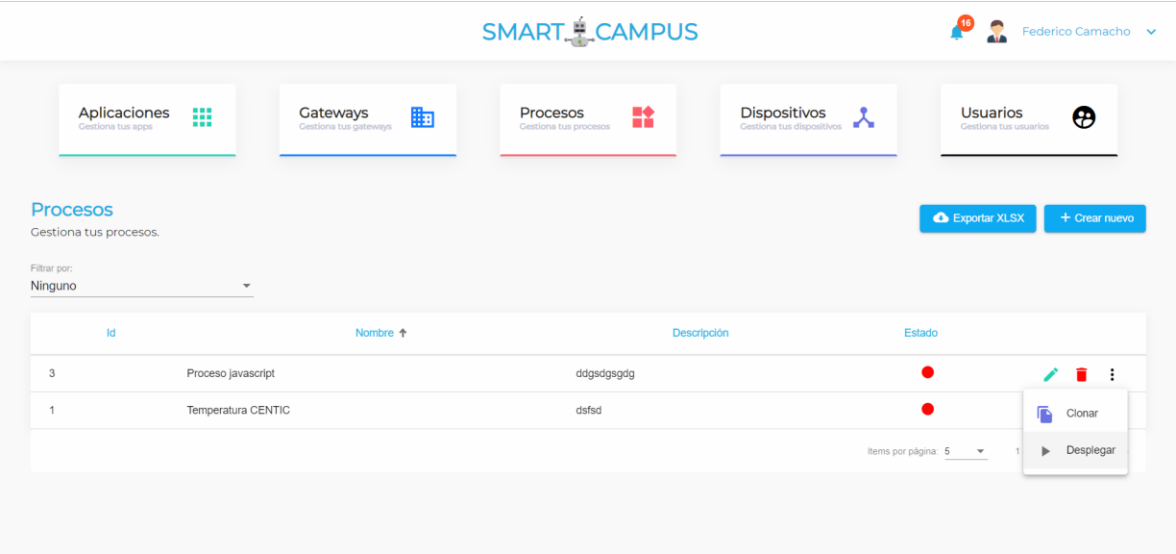

Figura 29. Vista de Procesos.

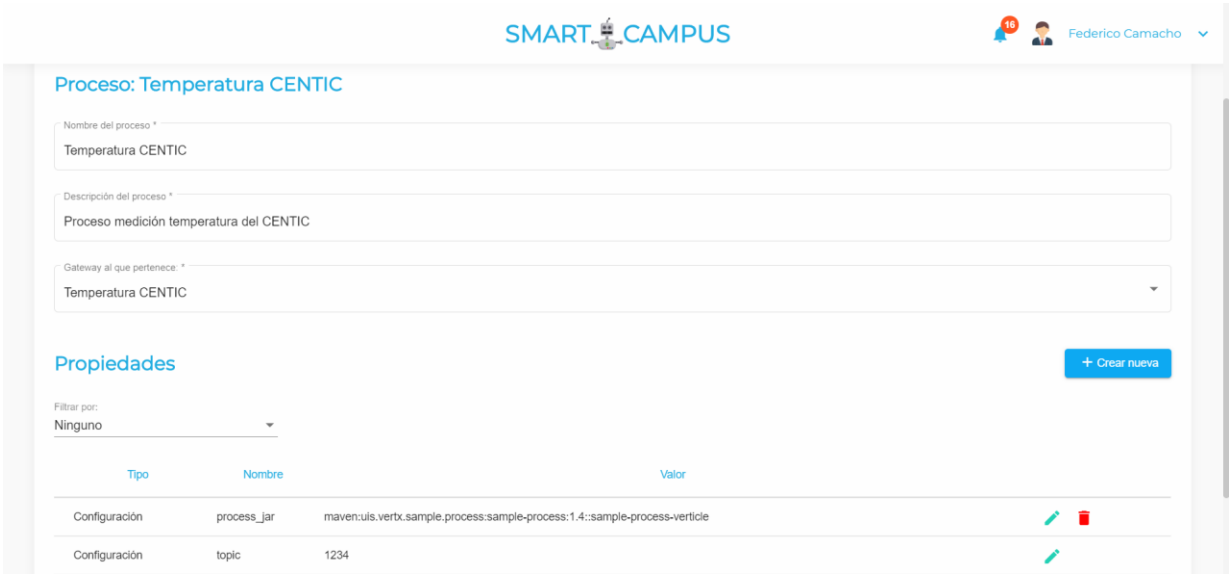

Figura 30. Vista de edición de Proceso.

Para poder desplegar un Proceso dentro de la arquitectura, este debe ser un proceso desarrollado en Java y, en el Gateway en el que este reside se debe instalar Maven (herramienta para la gestión y construcción de proyectos Java que permite manejar dependencias de una manera sencilla) y, el JAR del Proceso desarrollado debe ser instalado en el repositorio local de Maven con el comando:

```
mvn install:install-file -Dfile="temperature-process-0.0.1.jar" -
DgroupId="com.uis.iot" -DartifactId=" temperature-process" -Dversion="0.0.1"
-Dpackaging="jar"
```

Donde *temperature-process-0.0.1.jar* es el nombre del archivo JAR generado luego de construir y empaquetar el proceso desarrollado, *com.uis.iot* es el nombre de la organización que se le desee dar para identificarlo, *temperature-process* es el nombre del proyecto, *0.0.1* es la versión de este.

Luego de esto al momento de registrar el Proceso en la plataforma de administración es necesario en las propiedades del Proceso agregar una propiedad con nombre *process_jar* y de tipo configuración con un valor del estilo:

maven:organization-name:process-jar-name:jar-version::verticle-config

Donde el *verticle-config* es un archivo JSON de configuración en el que se indica que clases debe instanciar como Verticles, u objetos del framework Vert.x, el cual es utilizado en el framework del Gateway para correr los procesos en la misma máquina virtual y de esta manera reducir el consumo de memoria y realizar una comunicación directa mediante un bus de eventos entre el Software del Gateway y el de los procesos.

Esta es una breve descripción de la manera de desarrollar procesos desplegables dentro del software del Gateway sin embargo no se entra en detalle pues esto fue implementado en el proyecto de Gateway *"Diseño de un framework software extensible para dispositivos tipo gateway integrados en plataformas IoT para Smart Campus"* (Gutiérrez V, 2019). Y en este caso la plataforma de administración permite la configuración de Procesos de esta forma, desplegarlos o detenerlos con solo un clic desde la plataforma de administración, brindando a los usuarios un

control sobre los elementos que conforman la plataforma sin necesidad de ir directamente a la ubicación geográfica de los Gateways para poder correr o detener estos programas.

A su vez es obligatorio que los procesos tengan una propiedad de configuración llamada *topic* este es el tópico del Broker de mensajeria en el que se envía la información relevante para ese proceso y, el usuario puede en el código de este escuchar a los mensajes que se le envíen y reaccionar ante los mismos.

Con esta implementación en el prototipo se cumple con el Requerimiento Funcional RF05 *Gestión de Procesos.*

A nivel de la plataforma, los dispositivos son los sensores, encargados de producir información o medidas, como por ejemplo un sensor de temperatura o calidad del aire y los actuadores, aquellos elementos encargados de realizar acciones físicas ante una entrada respectiva. Estos dispositivos deben ser registrados en la plataforma por medio de la aplicación de administración, para permitir al usuario darles atributos que puedan ser utilizados dentro de los Procesos. Para esta labor, se desarrolló la vista de gestión de dispositivos, que se observa en la figura 31.

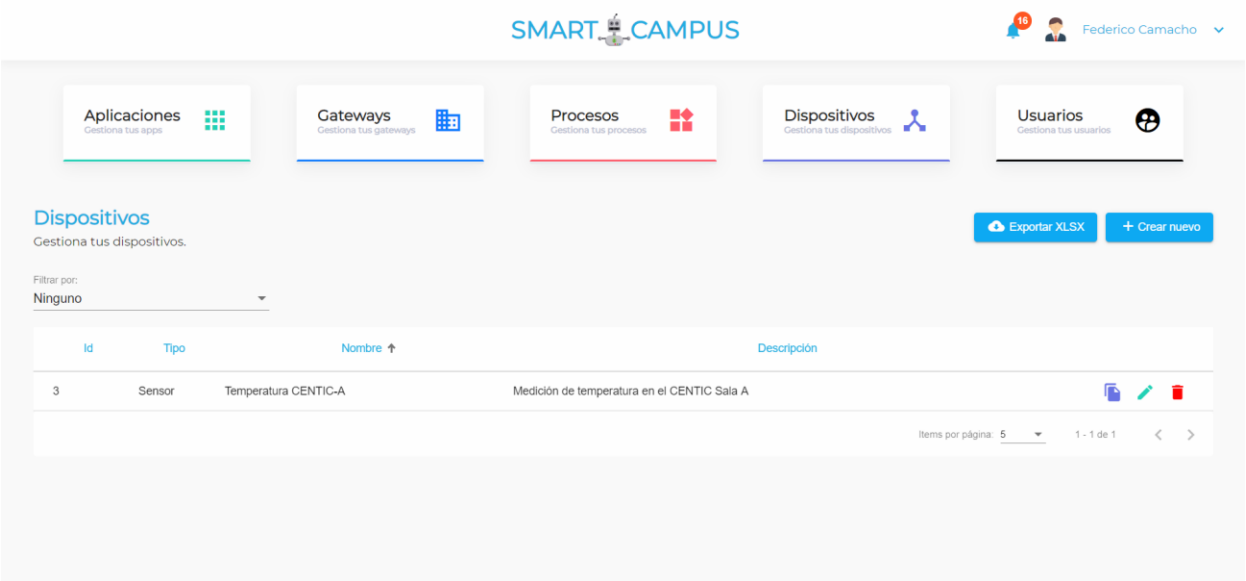

Figura 31. Vista de Dispositivos.

Donde también es posible crear nuevos procesos desde cero u a partir de otro ya existente, editarlos, eliminarlos, exportar estos en un XLSX, listarlos todos, ordenados por alguno de sus atributos o filtrarlos por Id, tipo (sensor o actuador) o descripción. Al momento de editar un dispositivo se dirige a la página de edición de dispositivo donde puede modificar sus propiedades, también esta misma página es vista al momento de registrar un nuevo dispositivo, solo que, con el formulario vacío y listo para ser diligenciado por el usuario, esta vista de edición se observa en la figura 32 y con esta implementación se cumple el Requerimiento Funcional RF04 *Gestión de Dispositivos.*

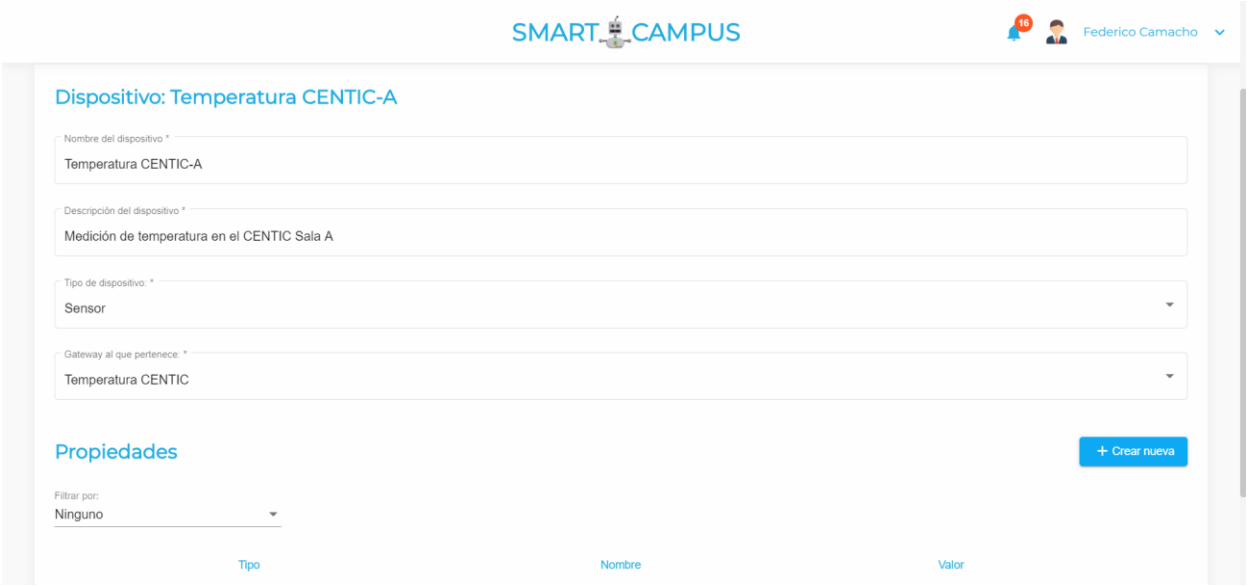

Figura 32. Vista de Edición de Dispositivo.

Como se mencionó anteriormente, el Superusuario a su vez puede ver la información de todos los usuarios de la plataforma (excepto obviamente sus contraseñas por cuestiones de seguridad), eliminar cuentas si lo desea o crear nuevas. Esto se observa en la figura 33 y da por cumplido el Requerimiento Funcional RF06 *Gestión de Usuarios*.

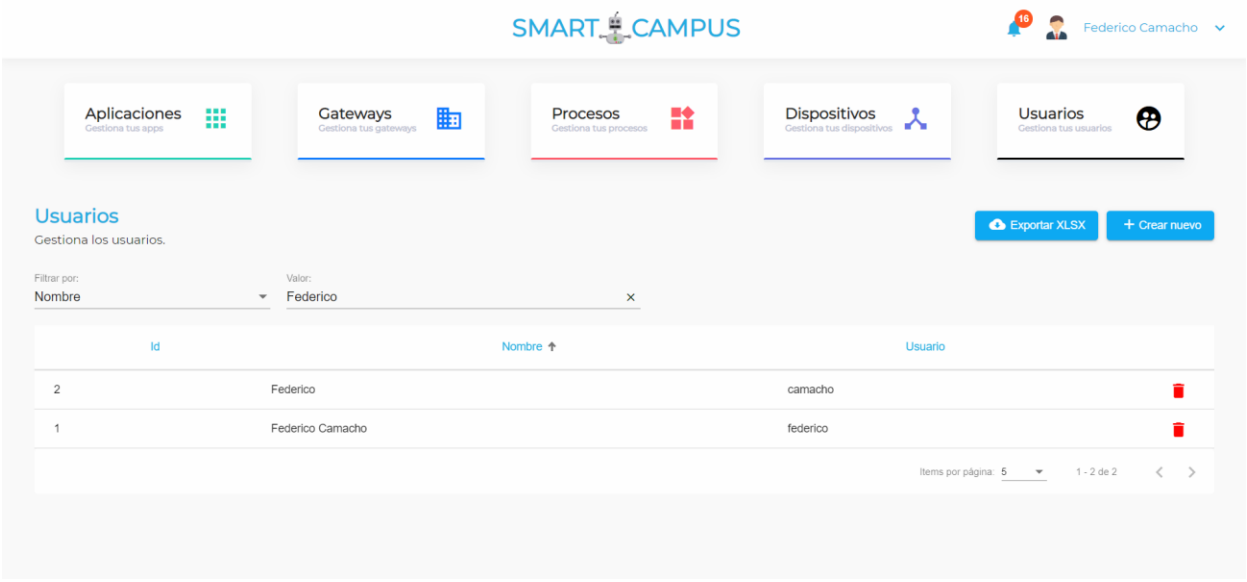

Figura 33. Vista de Gestión de Usuarios.

También se desarrolló una sección para permitir a los usuarios consultar la información enviada por sus dispositivos en la plataforma, esta permite consultar por aplicación, gateway, proceso o tópico y en un rango de fechas o en todo el historial, ordenar esta información y exportarla a XLSX, esto para cumplir con el Requerimiento Funcional RF07 *Consulta de Datos*, como se puede observar en la figura 34.

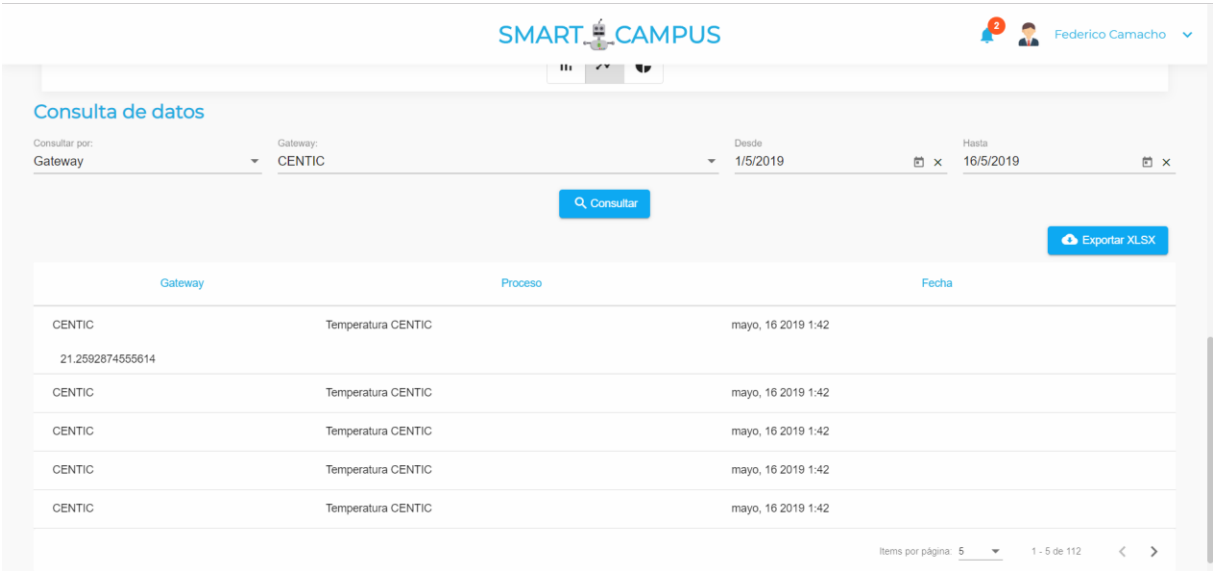

Figura 34. Consulta de datos.

Volviendo a la página de inicio o Dashboard, en esta también se encuentran estadísticas de administración, como se puede observar en la figura 35, donde se muestran los gateways y procesos del usuario con la cantidad de estos que están activos e inactivos y una gráfica de barras con el historial de cambios de actividad (inactivo/activo) de los gateways y procesos. Por ejemplo, en esta figura se muestra que el usuario tiene todos sus procesos inactivos (posando el cursor sobre la gráfica se ve a detalle la cantidad de elementos) y sus gateways activos y, a las 20:43 se reactivó un dispositivo (en este caso un gateway), esto complementado con las notificaciones, explicadas posteriormente permiten al usuario monitorear el estado de gateways y procesos.

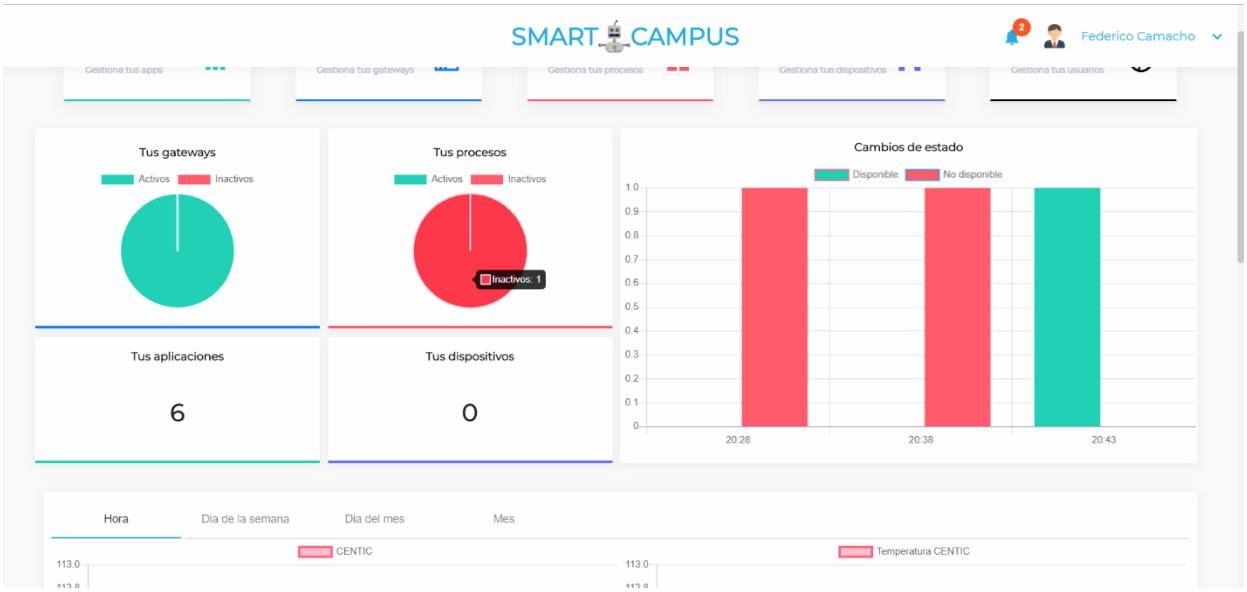

Figura 35. Estadísticas de Administración.

También se presentan estadísticas mediante gráficas de los datos enviados, aquí el usuario puede escoger si desea ver estadísticas por hora del día, por día de la semana, día del mes o mes, y además graficarlas con 3 tipos diferentes, gráfico de barras, lineal o torta.

En la figura 36 se observan las gráficas de barras por día del mes, donde se observan los mensajes que se enviaron en estos días (13, 14, 16 y 20), posando el ratón sobre la barra del día 16 para Gateways se puede observar mediante ese tooltip que se enviaron 112 mensajes, para el

Gateway CENTIC. En las figuras 37 y 38 se muestran a su vez las gráficas de datos enviados por día de la semana (línea) y por mes (torta), esto para cumplir con el requerimiento funcional RF09 *Visualización de estadísticas.*

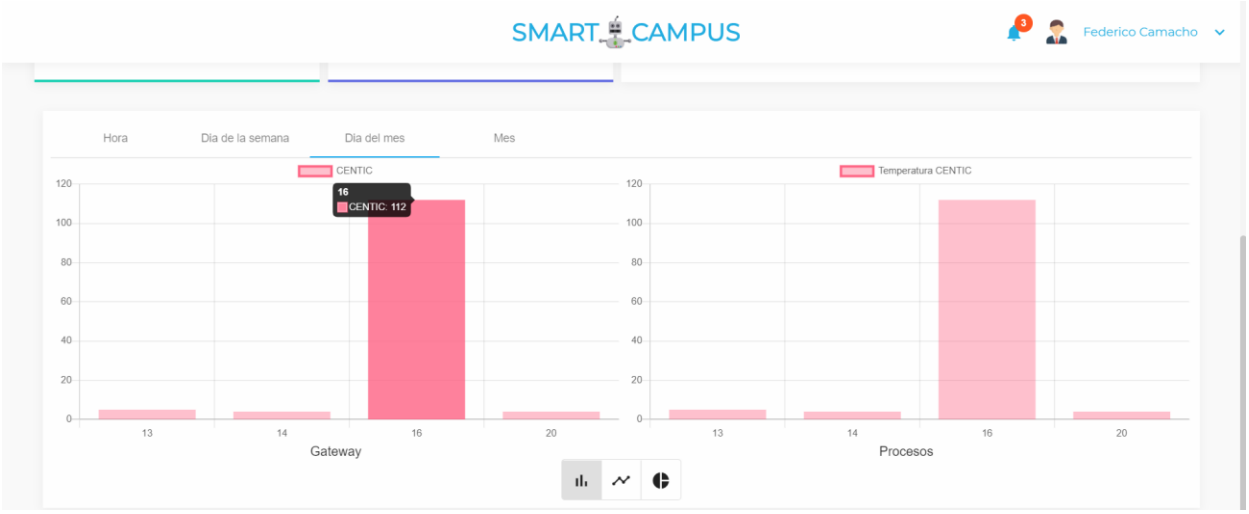

Figura 36. Gráfica de barra de datos enviados por día del mes.

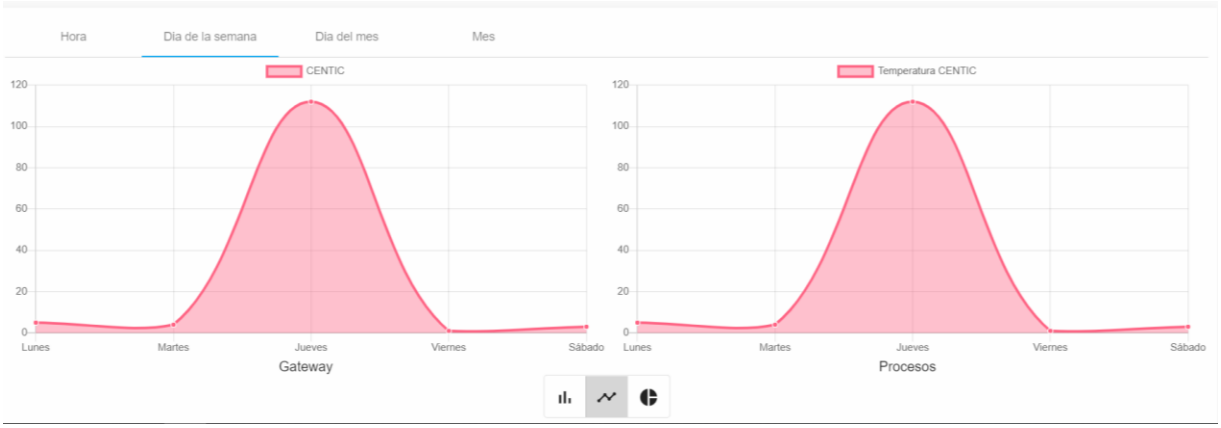

Figura 37. Gráfica de datos enviados por día de la semana.

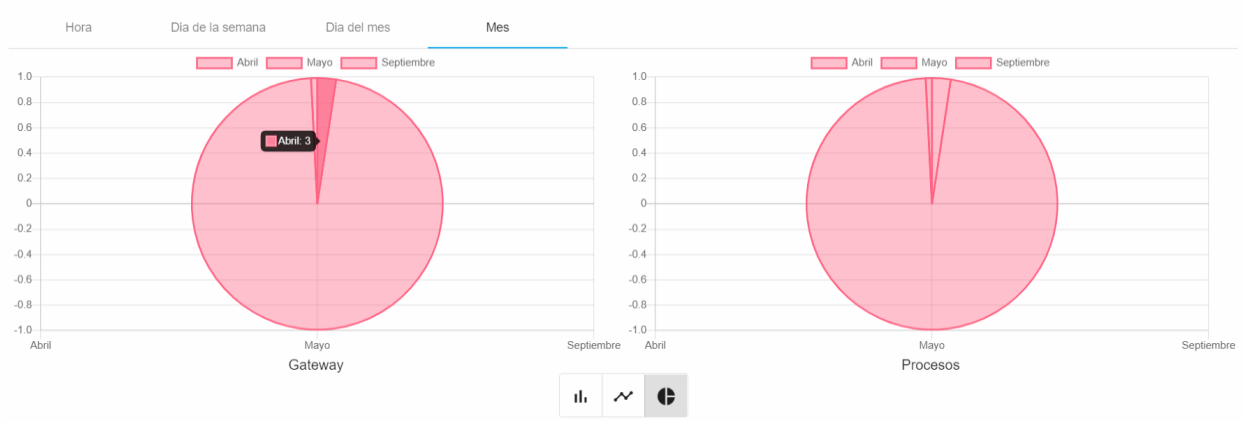

Figura 38. Gráfica de torta de datos enviados por mes.

Siempre que se verifica que hubo un cambio de estado de gateways o procesos del usuario, este recibe una notificación la cual se muestra como una alerta, como se muestra en la esquina inferior derecha de la figura 39, donde el usuario recibió una notificación de que su Gateway *Raspberry* está disponible nuevamente.

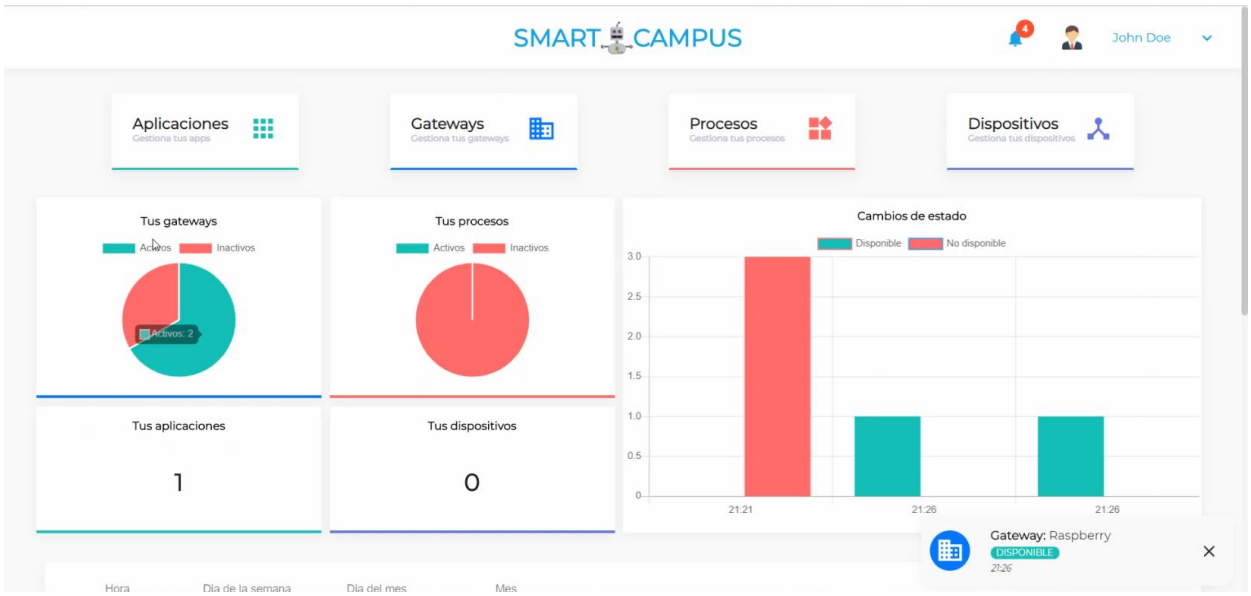

Figura 39. Notificación recibida.

El historial de notificaciones es visible para el usuario, y este puede desde una pestaña ver las notificaciones pendientes por leer y abrirlas como se ve en la figura 40 o abrir una o verlas todas

en detalle. Estas notificaciones también se pueden ordenar, filtrar, y eliminar, para hacer más personalizada la experiencia del usuario.

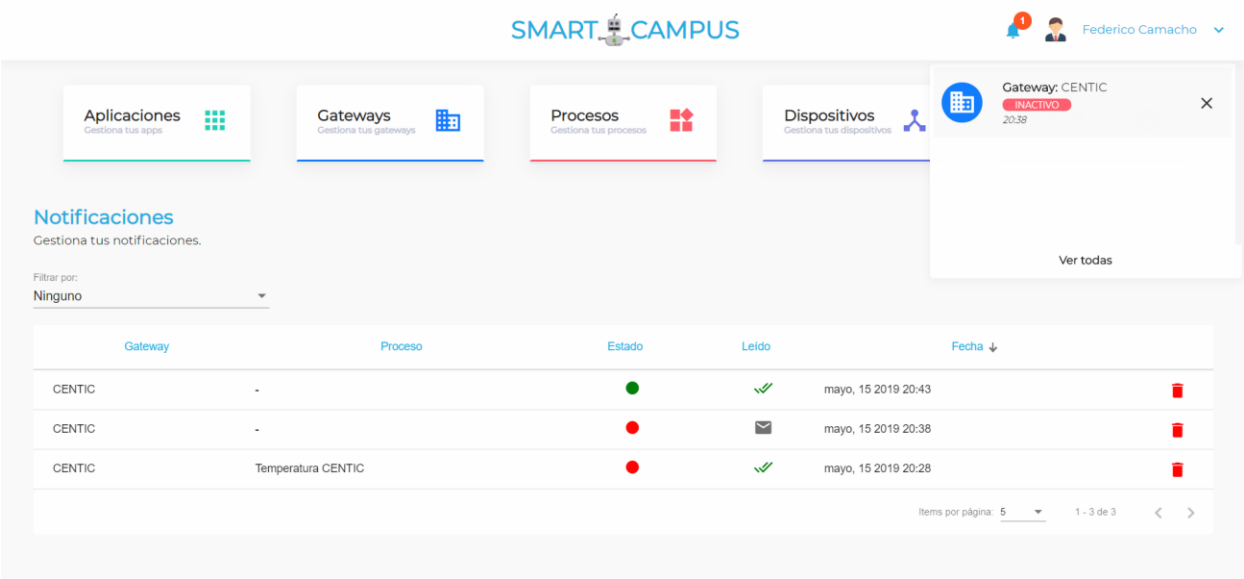

Figura 40. Vista de Notificaciones.

6.6 Validación

En esta fase se describen los escenarios planteados para validar el funcionamiento de la aplicación Web con base en los requerimientos funcionales y no funcionales descritos en el capítulo 5.

6.6.1 Verificación Compilación. En primera instancia se comprueba la compilación y funcionamiento correcto de la aplicación Web, para permitir más a delante que esta sea extendida por más usuarios, y cumplir con las características extensivas requeridas para este prototipo. Estas pruebas se realizaron en el mismo equipo en el que se desarrolló la plataforma y las especificaciones son las siguientes.

Tabla 5
Especificaciones equipo de computo

Característica	Descripción
Procesador	Intel Core i7-8550U 2.0GHz
Arquitectura del procesador	64 bits, procesador x64
Memoria RAM	8GB DDR4
Almacenamiento	SSD 240GB

Tabla 6
Versiones del Software utilizado

Software	Versión
Sistema operativo	Windows 10 Home
ActiveMQ Broker	5.15.8
NPM	6.4.1
Angular CLI	7.3.7
Node	8.11.3
Typescript	3.2.4
Google Chrome	74.0.3729

El resultado de la compilación se observa en la figura 41, esto se realiza utilizando los parámetros de producción (--prod) esto compila la aplicación reduciendo el tamaño de los archivos haciéndolos más eficientes, eliminando código "muerto" o sin dependencias y utilizando solo las dependencias externas necesarias para producción. La bandera AoT (--aot) Ahead of Time que permite que la aplicación no sea compilada en tiempo de ejecución reduciendo el tiempo de carga pues no es compilada primero en el navegador, sino que esta sea pre-compilada.

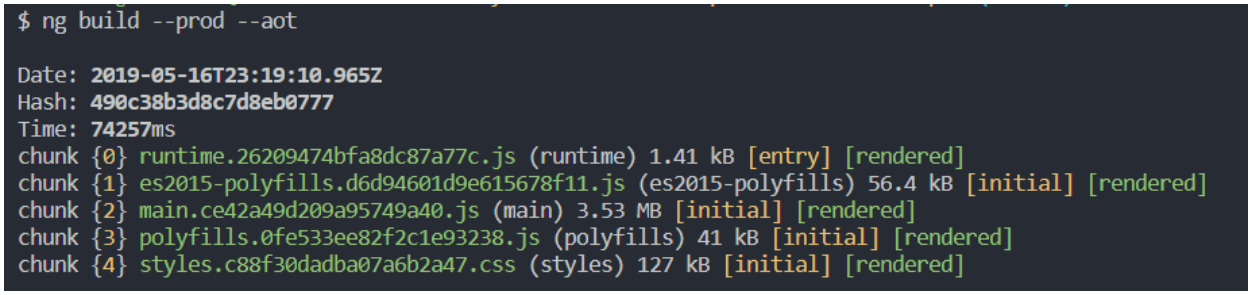

Figura 41. Resultado de compilación.

6.6.2 Verificación escenarios de prueba. Para la validación se planteó un plan de pruebas con el fin de verificar los casos de uso y requerimientos planteados. Dichos planes consistieron en diferentes escenarios en donde hay una respuesta esperada y esta se compara con la respuesta obtenida en la ejecución.

Tabla 7
Escenarios de Prueba

Escenario de Prueba	Valores Entrada	Respuesta	Resultado

Crear usuario	Tabla 8	Tabla 8	Satisfactorio
Iniciar sesión	Tabla 9	Tabla 9	Satisfactorio
Recuperación contraseña	Tabla 10	Tabla 10	Satisfactorio
Crear Aplicación	Nombre: Temperatura Descripción: Temperatura	Creada	Satisfactorio
Crear Gateway	Tabla 11	Tabla 11	Satisfactorio
Editar Gateway	Tabla 12	Tabla 12	Satisfactorio
Crear Proceso	Nombre: Temperatura CENTIC Descripción: Medir temperatura Gateway: CENTIC Tópico: temperatura Process_JAR: maven:com.uis.iot:temperature-process:0.0.1::temperature-process-verticle	Creado	Satisfactorio
Desplegar Proceso	Proceso: Temperatura CENTIC	Desplegado, enviando información	Satisfactorio
Consultar datos enviados	Proceso: Temperatura CENTIC	Disponible	Satisfactorio
Consultar datos enviados	Aplicación: Temperatura	Disponible	Satisfactorio
Consultar datos enviados	Gateway: CENTIC	Disponible	Satisfactorio
Detener Proceso	Proceso: Temperatura CENTIC	Detenido, detenido envío información	Satisfactorio
Recibir Notificación	Luego de detener Proceso Temperatura CENTIC	Notificación inactividad recibida	Satisfactorio

Tabla 8

Casos de Prueba Registro de Usuario

Campo	Caso 1	Caso 2	Caso 3	Caso 4	Caso 5
Usuario	John.doe	John.doe	-	John.doe	John.doe
Nombre	John Doe	John Doe	John Doe	John Doe	John Doe Jr
Email	John.com	John@mail.com	John@mail.com	John@mail.com	Johnjr@mail.com
Contraseña	1234	1234	1234	1234	1234
Repetir	1234	12345	1234	1234	1234
Respuesta	Email invalido	Contraseñas no coinciden	Usuario requerido	Registro exitoso	El usuario ya existe
Resultado	Exitoso	Exitoso	Exitoso	Exitoso	Exitoso

Tabla 9

Casos de Prueba Inicio de Sesión

Campo	Caso 1	Caso 2	Caso 3	Caso 4
Usuario	John.doe	John.jr	-	John.doe
Contraseña	abcd	1234	1234	1234
Respuesta	Contraseña incorrecta	Usuario no existe	Usuario requerido	Inicio de sesión exitoso

Resultado	Exitoso	Exitoso	Exitoso	Exitoso

Tabla 10
Casos de Prueba Recuperar contraseña

Campo	Caso 1	Caso 2	Caso 3
Correo	John.com	John.rj@mail.com	John@mail.com
Respuesta	Correo invalido	Correo no registrado	Contraseña enviada
Resultado	Exitoso	Exitoso	Exitoso

Tabla 11
Casos de Prueba Crear Gateway

Campo	Caso 1	Caso 2	Caso 3	Caso 3
Nombre	CENTIC	-	CENTIC	BIBLIOTECA
Descripción	Gat. CENTIC	Gat. CENTIC	Gat. CENTIC	Gat. BIBLIOTECA
IP	prueba	http://localhost:8080	http://localhost:8080	http://localhost:8080
Respuesta	No se pudo conectar al Gateway	Nombre requerido	Gateway Creado, Propiedades informativas generadas	Gateway ya registrado
Resultado	Exitoso	Exitoso	Exitoso	Exitoso

Tabla 12
Casos de Prueba Editar Gateway

Campo	Caso 1	Caso 2	Caso 3
Nombre	CENTIC	CENTIC	CENTIC
Descripción	Gat. CENTIC	Gat. CENTIC	Gat. CENTIC
IP	prueba	http://localhost:8080	http://localhost:8080
Propiedades	-	db_cleanup_time (CONFIG):4320	db_cleanup_time (CONFIG): 4320 ubicación (INFORMATIVA): CENTIC
Respuesta	No se pudo conectar al Gateway, no guardado	Actualización correcta, La base de datos del Gateway se limpia cada 4320 min (3 días)	Gateway actualizado correctamente
Resultado	Exitoso	Exitoso	Exitoso

6.6.3 Verificación de responsive. En la verificación también se comprobó el correcto funcionamiento, visualización de la información e interacción en dispositivos móviles, el dispositivo utilizado para pruebas es el siguiente.

Tabla 13
Especificaciones dispositivo móvil de pruebas

Característica	Descripción
Nombre	Xiaomi Mi A1
Procesador	Qualcomm Snapdragon 625
Chipset	Octa-core 2.0 GHz Cortex-A53
GPU	Adreno 506
Memoria RAM	4GB
Almacenamiento	64 GB

Algunas de las vistas ya mostradas en la sección 6.4 se revisaron para esta validación en el dispositivo móvil, como se observa en las figuras 42 a 46. Se encontró que la información se muestra correctamente para tamaños de pantalla reducidos y que la interacción del usuario es cómoda y sencilla, y este no pierde ninguna funcionalidad por navegar por la aplicación web desde un dispositivo móvil, comprobando que se cumple con el requisito no funcional RNF01 *Diseño Responsivo.*

A su vez, al permitir al usuario ordenar, filtrar, paginar y exportar la información de los elementos que conforman la plataforma Smart Campus, manteniendo la consistencia entre todas las vistas para que sea más sencillo ubicarse y saber manejar las funcionalidades pues la interacción necesaria es consistente se logró cumplir con el RNF02 *Diseño intuitivo.*

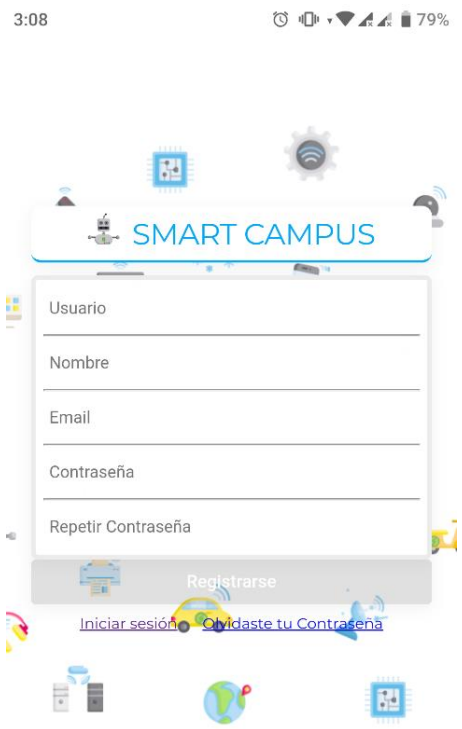

Figura 42. Responsive Registro de Usuario.

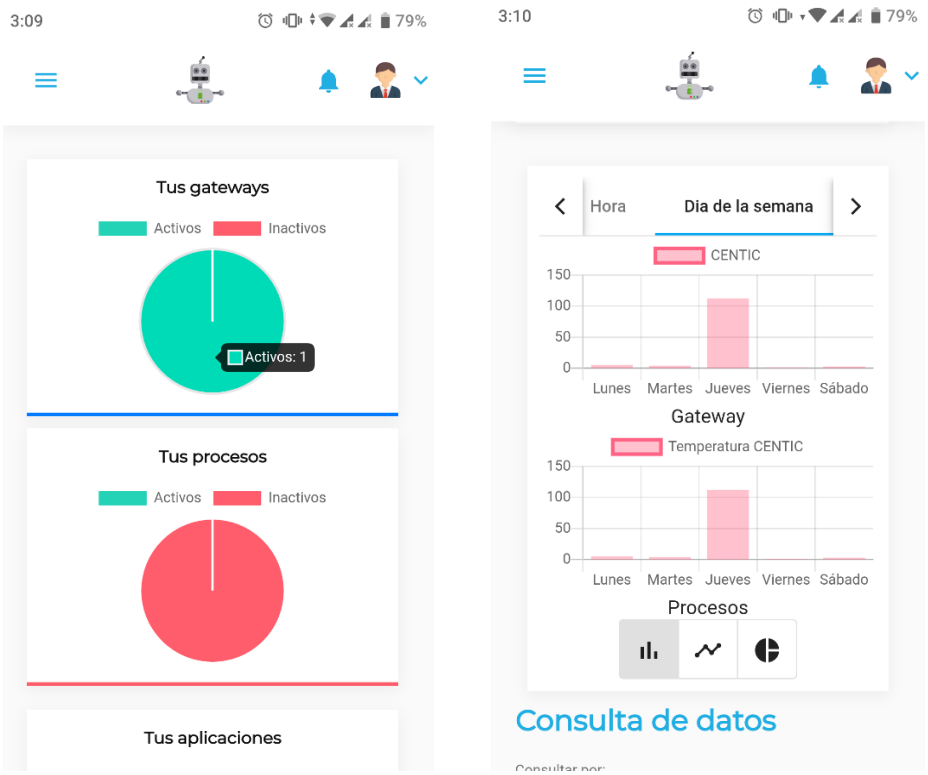

Figura 43. Responsive Estadísticas (Página de Inicio).

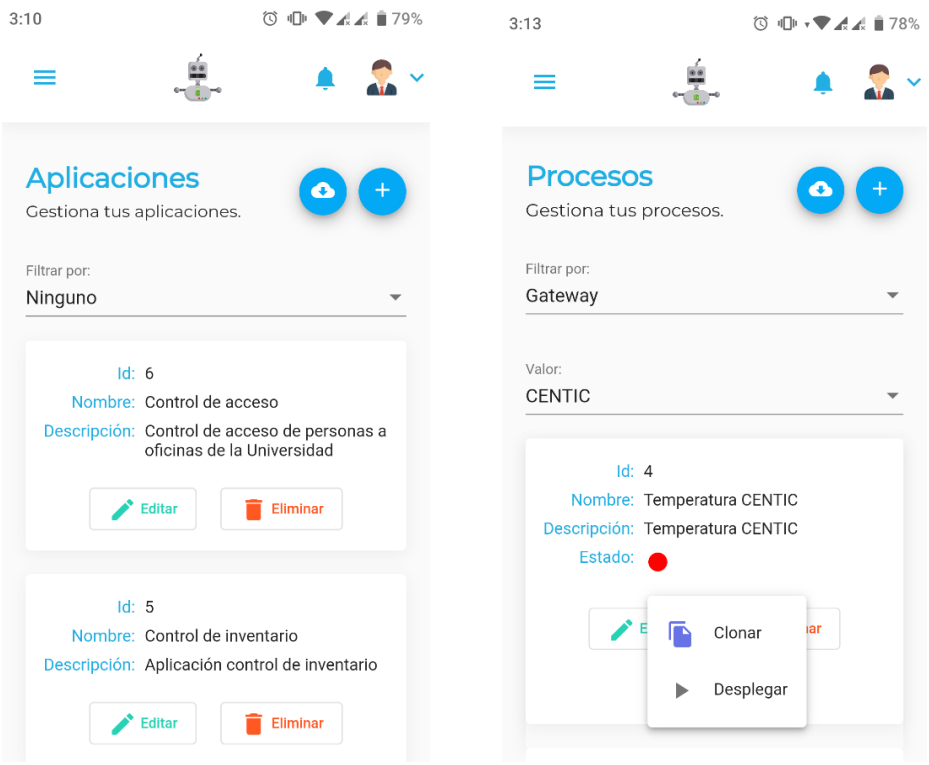

Figura 44. Responsive Aplicaciones y Procesos.

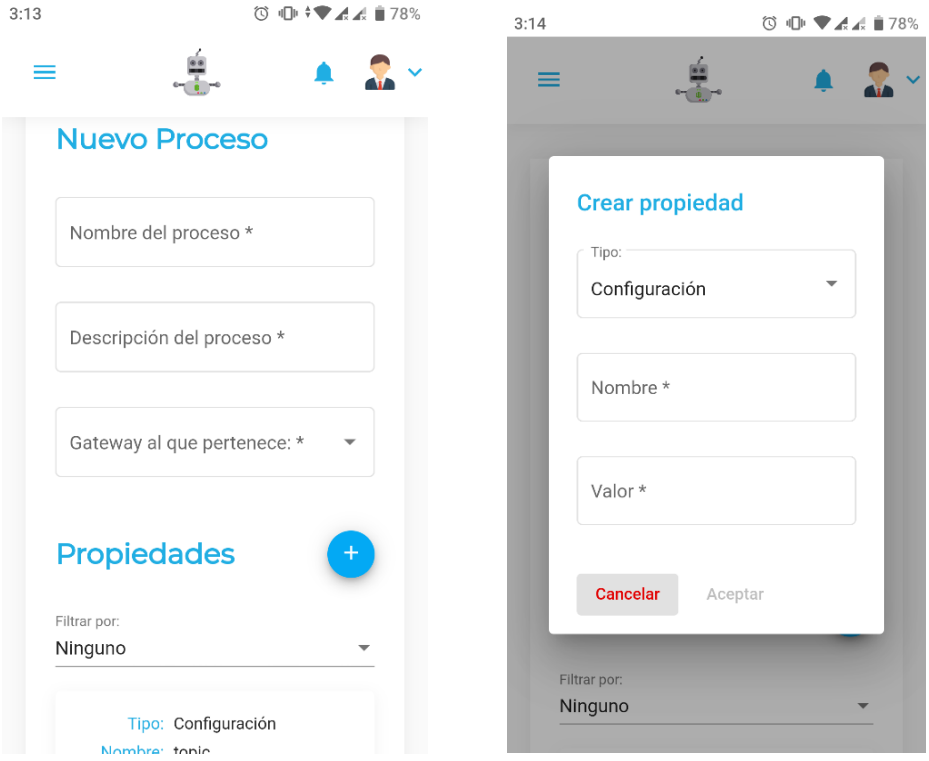

Figura 45. Responsive creación de Proceso y sus propiedades.

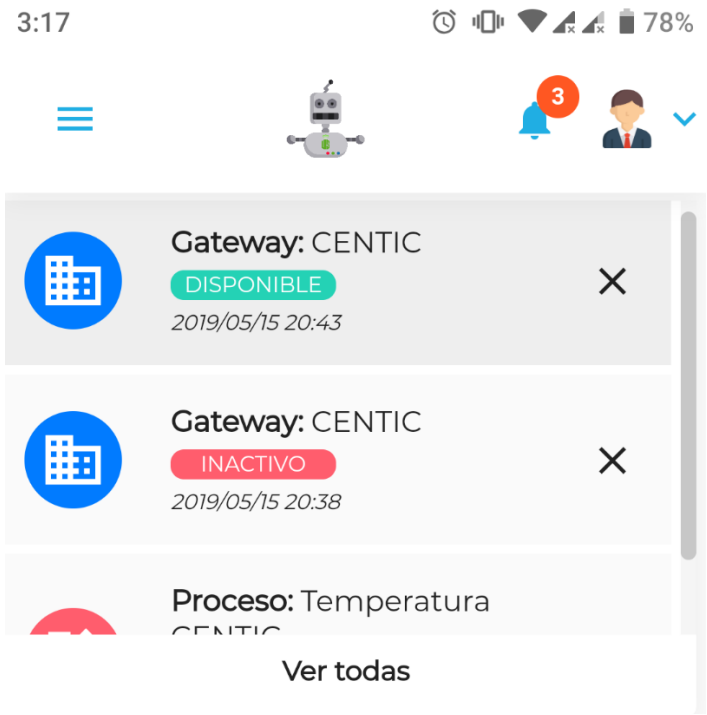

Figura 46. Responsive vista nuevas notificaciones.

6.6.4 Verificación mediante métricas. Para validar que el prototipo de la aplicación de administración de la plataforma Smart Campus se desarrollara de forma correcta y que esta sea eficiente se corrió sobre la misma una serie de pruebas, utilizando una herramienta llamada *Perf. Tool.* Los resultados obtenidos mediante esta herramienta son los siguientes

- **Número de peticiones HTTP:** Mide la cantidad de peticiones HTTP necesarias para cargar la información, esto incluye petición del contenido a renderizar (HTML, CSS y JavaScript) y peticiones de información (llamados al servidor), entre menor sea esta medida mejor, pues implica una carga inicial más eficiente.

  Valor límite: 200 peticiones.

  Resultado prueba: 10.

  Calificación: Excelente.

- **Not found:** Número de errores 404 recibidos.

Valor límite: 0.

Resultado prueba: 0.

Calificación: Excelente.

- **Múltiples peticiones:** Número de recursos estáticos solicitados más de una vez.

  Valor límite: 1.

  Resultado prueba: 0.

  Calificación: Excelente.

- **Redirecciones:** Número de redirecciones realizadas. Este resultado es excelente porque demuestra que la renderización del lado del cliente y el uso de una Single Page Application hace que no haya redirecciones y que, por ende, sea más eficiente y se brinde una mejor experiencia de usuario.

  Valor límite: 2.

  Resultado prueba: 0.

  Calificación: Excelente.

- **Tamaño de la respuesta más grande:** El tamaño de la respuesta más grande.

  Valor promedio: 5 MB.

  Resultado prueba: 3.5 MB.

  Calificación: Buena.

Adicionalmente se utilizó la extensión del navegador Google Chrome creada por Google llamada Augury para probar el rendimiento de la aplicación web, donde se obtuvieron los siguientes resultados:

- **Primera renderización de contenido:** Tiempo que le toma a la primer imagen o texto en ser renderizado. Resultado: 0.4s.
- **Primera renderización entendible del contenido:** Tiempo que toma en ser renderizado el contenido primario de la aplicación. Resultado: 1.4s.
- **Velocidad de indexación:** Qué tan rápido son renderizados los elementos visibles en la página. Resultado: 1.3s.
- **Tiempo a interactivo:** La cantidad de tiempo que le toma a la aplicación en ser totalmente interactiva. Resultado: 1.1s.
- **Tiempo primera inactividad de la CPU:** Marca el tiempo en el que la CPU está lo suficientemente libre para manejar correctamente las entradas/interacciones del usuario. Resultado: 1.4s.
- **Latencia estimada de interacción:** Estimado de que tanto tiempo le toma a la aplicación responder a la interacción del usuario durante los 5s en los que esta se encuentra más ocupada. Resultado: 20ms.

Con estos resultados y como se pueden comprobar en la figura 46 se comprueba que el rendimiento de la aplicación web es muy bueno y esta responde rápido al usuario obteniendo una calificación de 100 de 100 puntos posibles, demostrando que el requerimiento no funcional RNF03 fue cumplido correctamente.

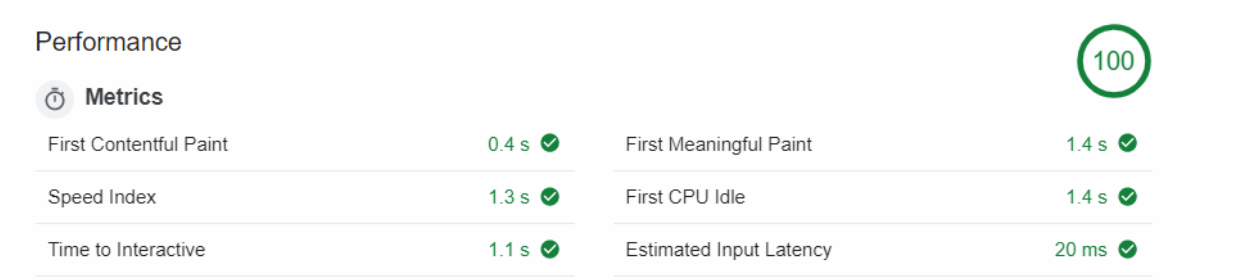

Figura 47. Resultados de métricas de rendimiento.

7. Caso de Uso: Implementación e Implantación

En esta sección se realizó el despliegue de la plataforma IoT enfocada a Smart Campus, implementado un prototipo para demostrar que todos los componentes se integran correctamente. Para lograr esto, fue necesaria la participación de los autores de los proyectos de grado mencionados en los capítulos anteriores relacionados con la Plataforma IoT.

En el escenario planteado un supervisor desea activar una red de bombillos ubicados en dos laboratorios de ingeniería eléctrica cuando el voltaje de un generador eléctrico pase un umbral de seguridad para alertar al personal presente en dichos espacios.

Además de esto, el usuario desea monitorear la temperatura generada en otro laboratorio en el cual se almacenan algunas sustancias químicas que deben permanecer constantemente sobre una temperatura mínima. Para esto, instala un sensor de temperatura en el laboratorio, y un led en su oficina, el cual deberá encenderse en caso de que la temperatura en el laboratorio descienda bajo un umbral previamente determinado.

Para simular el escenario anterior se hizo uso de dos dispositivos tipo gateway, una minicomputadora con pines análogos, tres actuadores (LEDs), un sensor de temperatura y un potenciómetro; a su vez, se crearon 4 procesos y 1 aplicación externa en la que se muestran los datos capturados en el caso de uso.

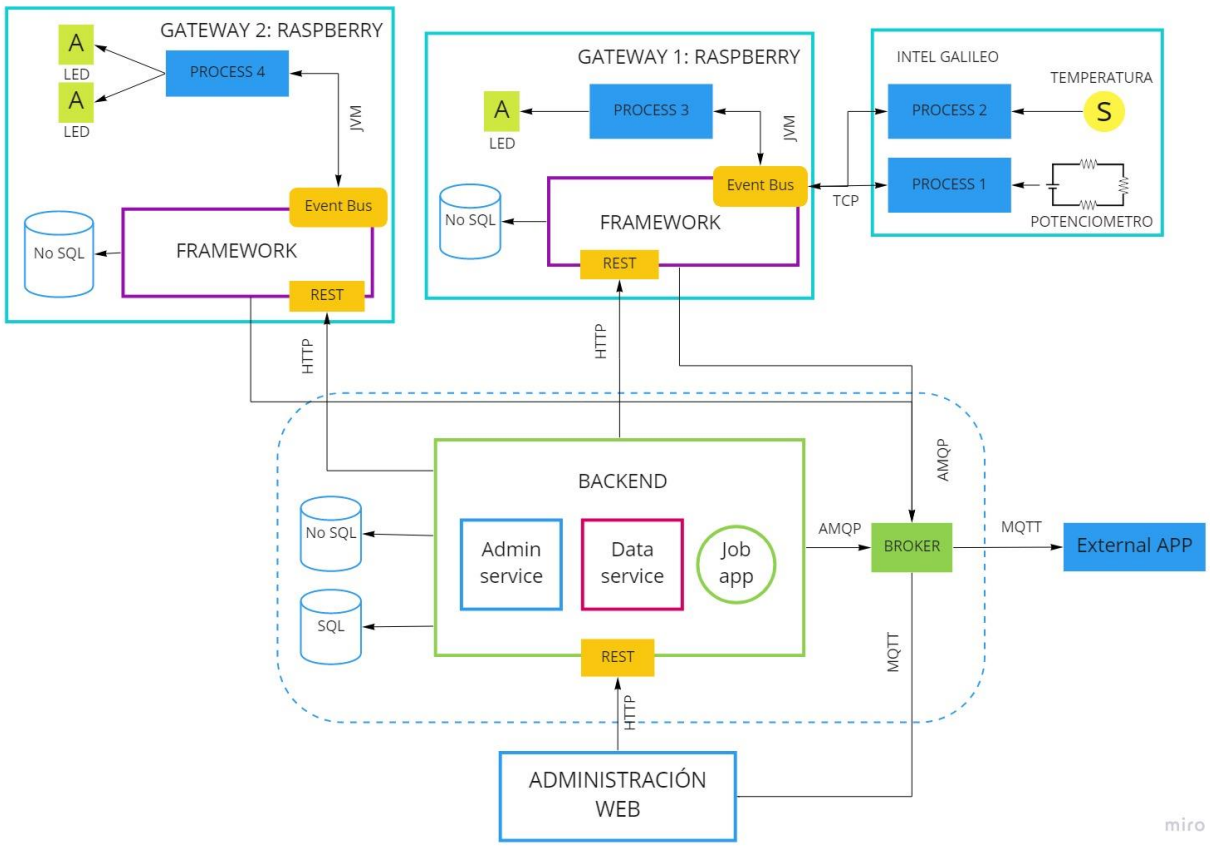

Figura 48. Arquitectura planteada para el caso de uso.

A nivel lógico se utilizaron cuatro procesos definidos a continuación:

- Proceso 1: Este proceso se encargó de enviar los datos generados en el circuito conectado a la placa Intel Galileo (microcomputador). Fue desarrollado en lenguaje Python y se conectó al Gateway 1 a través de TCP.
- Proceso 2: Este proceso se encargó de dar instrucciones al LED ubicado en el Gateway 1 para prenderse o apagarse dependiendo de los mensajes que llegaban a través del bus de eventos.
- Proceso 3: Este proceso se encargó de dar instrucciones a los LED ubicados en el Gateway 2 para prenderse o apagarse dependiendo de los mensajes que llegaban a través del bus de eventos.
- Proceso 4: Este proceso se encargó de manejar los datos generados por el sensor de temperatura ubicado en el Gateway 2 enviándolos a través del bus de eventos.

Para hacer esto fue necesario registrar y configurar una aplicación (Control laboratorio), los dos Gateways y los cuatro procesos en la aplicación Web de administración y, mediante la misma se mandaron a desplegar los procesos 2, 3 y 4 (el proceso 1 no es desplegable pues no es Java). La aplicación externa para la prueba fue creada dentro del proyecto de la aplicación de administración y se conecta al broker de mensajería para graficar la información recibida. En el apéndice C se presenta un video donde se explica a detalle este caso de uso y se observa en ejecución.

8. Trabajo a futuro

- Implementar un módulo de Reglas (Rule Engine) en la plataforma Smart Campus, que permita al usuario configurar gráficamente condiciones que se deben ejecutar en los dispositivos Gateways para procesar información o realizar acciones.
- Implementar grupos de usuario y permitir asignar aplicaciones a estos grupos, para que sean gestionados por más de una persona de acuerdo con los deseos del creador del caso de uso.

9. Conclusiones

Una aplicación Web es una buena elección para administrar una arquitectura Smart Campus pues, puede ser accedida desde diversos dispositivos como computadores, tablets y celulares siendo esto más cómodo para los usuarios.

Se desarrolló una arquitectura extensible basada en componentes que usa como plataforma tecnológica Angular, pues más adelante es posible agregar funcionalidades de forma sencilla reutilizando código ya existente, por ejemplo, incluyendo casos de uso por defecto.

La elección de una tecnología que utiliza renderización del lado del cliente hace que la aplicación Web sea más eficiente y permite reducir la carga en el lado del servidor.

El diseño de este prototipo y de la arquitectura Smart Campus demuestra que es factible generar una transformación tecnológica en las universidades donde, apoyándose en esta plataforma se pueden automatizar muchos procesos realizados.

Se logró conectar la aplicación de administración con el resto de la arquitectura, consumiendo la información asociada al usuario desde el servidor y a su vez recibiendo notificaciones y alertas en tiempo real permitiéndole a estos gestionar y monitorizar sus casos de uso y dispositivos asociados.

Se logró mostrar la información (tanto la utilizada para gestionar los elementos de la plataforma, como la proveniente de dispositivos) de forma intuitiva permitiendo a los usuarios filtrar, ordenar y exportar la misma.

Referencias Bibliográficas

Arias, K. y Estupiñan, F. (2019). Diseño del componente software backend orientado a una plataforma IoT diseñada para Smart Campus. Bucaramanga, Colombia.

Baquero, J. (2017). Single-Page Application, todo un website desde única página. Arsys.es. Recuperado el 29 de abril de 2019, https://www.arsys.es/blog/programacion/diseno-web/spa-unica-pagina/.

Bhalani, R. (2019). TypeScript vs JavaScript. Educba.com. Recuperado el 5 de mayo de 2019, https://www.educba.com/typescript-vs-javascript/.

Bonnet, S. (2015). MQTT: Un protocolo específico para el internet de las cosas. Digitaldimensions.solutions.es. Recuperado el 29 de abril de 2019, http://www.digitaldimension.solutions/es/blog-es/opinion-de-expertos/2015/02/mqtt-un-protocolo-especifico-para-el-internet-de-las-cosas/.

Código Facilito. (2016). Qué es Typescript. Codigofacilito.com. Recuperado el 29 de abril de 2019, https://codigofacilito.com/articulos/typescript.

De Seta, L. (2008). Introducción a los servicios web RESTful. Dosideas.com. Recuperado el 29 de abril de 2019, https://dosideas.com/noticias/java/314-introduccion-a-los-servicios-web-restful.

Fernández, D. (2017). Bróker de mensajería. Blog.bi-geek.com. Recuperado el 29 de abril de 2019, https://blog.bi-geek.com/broker-mensajeria/.

Gutiérrez, C. (2019). Diseño de un framework software extensible para dispositivos tipo gateway integrados en plataformas IoT para Smart Campus. Bucaramanga, Colombia.

I-Scoop. (2017). What is the Internet of Things? Internet of Things definitions. I-Scoop.eu. Recuperado el 29 de abril de 2019, https://www.i-scoop.eu/internet-of-things/.

IoT Agenda. (2019). Internet of Things (IoT). Internetofthingsagenda.techtarget.com. Recuperado el 27 de abril de 2019, https://internetofthingsagenda.techtarget.com/definition/Internet-of-Things-IoT.

IoT World Today. (2017). The definitive list of smart cities projects taking the world by storm. Iotworldtoday.com. Recuperado el 27 de abril de 2019, https://www.iotworldtoday.com/2017/09/28/definitive-list-smart-cities-projects-taking-world-storm/.

Jurado L., Velásquez W., Vinueza N. (2014). Estado del Arte de las Arquitecturas de Internet de las Cosas (IoT). Recuperado el 27 de abril de 2019, https://www.academia.edu/7197061/Estado_del_Arte_de_las_Arquitecturas_de_Internet_de_las_Cosas_IoT_.

Kolce, J. (2018). 10 Languages That Compile to JavaScript. Sitepoint.com. Recuperado el 5 de mayo de 2019, https://www.sitepoint.com/10-languages-compile-javascript/.

Lasse K. (2018) Stare of the IoT 2018: Number of IoT devices now at 17B. Iot-analytics.com. Recuperado el 27 de abril de 2019, https://iot-analytics.com/state-of-the-iot-update-q1-q2-2018-number-of-iot-devices-now-7b/.

Navdeep, S. (2019). IoT Analytics Platform for Real-Time Data Ingestion, Streaming Analytics. Xenonstack.com. Recuperado el 5 de mayo de 2019, https://www.xenonstack.com/blog/iot-analytics-platform-solutions/.

Nick, P. (2017). The Future is Angular: An Introduction to Angular 2. Upwork.com. Recuperado el 29 de abril de 2019, https://www.upwork.com/hiring/development/angular-2-framework/.

Quek T. (2017). The advantages and disadvantages of Internet Of Things (IoT). Linkedin.com. Recuperado el 27 de abril de 2019, https://www.linkedin.com/pulse/advantages-disadvantages-internet-things-iot-tommy-quek/.

Rojas, J. (2019). Definición de una infraestructura cloud de alta disponibilidad en un entorno distribuido para el despliegue de una plataforma IoT. Bucaramanga, Colombia.

Rouse M. (2017). Internet de las cosas (IoT). Searchdatacenter.techtarget.com. Recuperado el 29 de abril de 2019, https://searchdatacenter.techtarget.com/es/definicion/Internet-de-las-cosas-IoT.

Rouse, M. (2017). REST (REpresentational State Transfer). Searchmicroservices.techtarget.com. Recuperado el 29 de abril de 2019, https://searchmicroservices.techtarget.com/definition/REST-representational-state-transfer.

Santhi T., Rajendra J. & Vijayalakshmi, Y. (2006). A review on the state of art of Internet of Things. Ijarcee.com. Recuperado el 27 de abril de 2019, https://www.ijarcce.com/upload/2016/july-16/IJARCCE%2038.pdf

Software Testing Help. (2019). 10 Best IoT Platforms To Watch Out In 2019. Softwaretestinghelp.com. Recuperado el 27 de abril de 2019, https://www.softwaretestinghelp.com/best-iot-platforms/.

Universidad de Alcalá. s.f. Origen e historia del Internet of Things. Master-Internet-ofThings.com. Recuperado el 27 de abril de 2019, https://www.master-internet-of-things.com/historia-iot/.

Wiboomedia. s.f. ¿Qué son las Aplicaciones Web? Ventajas y Tipos de Desarrollo Web. Wiboomedia.com. Recuperado el 29 de abril de 2019, https://wiboomedia.com/que-son-las-aplicaciones-web-ventajas-y-tipos-de-desarrollo-web/.