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

**Nota de Proyecto de Grado**

(Va escaneada y con la información completa. Se diligencia al momento de hacer la sustentación)

**Autorización de Uso a Favor de la UIS**

Va escaneada y con la información completa, se puede descargar de la página de la Biblioteca: <http://tangara.uis.edu.co/biblioweb/pags/pub/informacion/recurso_ejemplos_TG.jsp>

**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 de esta etapa.

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

**Tabla de Contenido**

Pág.

[Introducción 13](#_Toc8409868)

[1. Objetivos 14](#_Toc8409869)

[1.1 Objetivo general 14](#_Toc8409870)

[1.2 Objetivos específicos 14](#_Toc8409871)

[2. Estado del arte 15](#_Toc8409872)

[3. Marco de referencia 18](#_Toc8409873)

[3.1 Internet de las cosas (iot) 18](#_Toc8409874)

[3.2 Smart Campus 19](#_Toc8409875)

[3.3 Aplicaciones Web 19](#_Toc8409876)

[3.4 Single Page Applications 20](#_Toc8409877)

[3.5 Typescript 20](#_Toc8409878)

[3.6 Angular 21](#_Toc8409879)

[3.7 Aplicaciones web progresivas. 21](#_Toc8409880)

[3.8 Servicios REST. 22](#_Toc8409881)

[3.9 Websockets 22](#_Toc8409882)

[3.10 Broker de mensajería 22](#_Toc8409883)

[3.11 MQTT 23](#_Toc8409884)

[4. Marco metodológico 23](#_Toc8409885)

[4.1 Fase 1: Capacitación tecnológica 24](#_Toc8409886)

[4.2 Fase 2: Definición de la arquitectura 24](#_Toc8409887)

[4.3 Fase 3: Prototipado 25](#_Toc8409888)

[4.4 Fase 4: Validación y verificación 25](#_Toc8409889)

[5. Requisitos y especificaciones 26](#_Toc8409890)

[5.1 Requisitos funcionales 26](#_Toc8409891)

[5.2 Requisitos no funcionales 30](#_Toc8409892)

[5.3 Especificaciones alcanzadas 31](#_Toc8409893)

[6. Desarrollo del proyecto 32](#_Toc8409894)

[6.1 Contexto del proyecto 32](#_Toc8409895)

[6.2 Capacitación tecnológica 32](#_Toc8409896)

[6.2.1 Arquitecturas iot. 33](#_Toc8409897)

[6.2.3 Tecnologías y herramientas. 34](#_Toc8409898)

[6.2.4 Capacitación tecnológica. 37](#_Toc8409899)

[6.3 Definición de la arquitectura 38](#_Toc8409900)

[7. Validación 39](#_Toc8409901)

[7. Trabajo a futuro 40](#_Toc8409902)

[8. Conclusiones 41](#_Toc8409903)

[9. Recomendaciones (opcional) 42](#_Toc8409904)

[Referencias bibliográficas 43](#_Toc8409905)

**Lista de Tablas**

[Tabla 1 *Cumplimiento de Requerimientos* 31](#_Toc8410045)

**Lista de Figuras**

[*Figura 1.* Campos de aplicación del Internet de las Cosas 16](#_Toc8410402)

[*Figura 2.* Siete características del Internet de las cosas. 18](#_Toc8410403)

[*Figura 3.* Esquema metodología de trabajo 24](#_Toc8410404)

[*Figura 4.* Componentes Software Arquitecturas IoT 34](#_Toc8410405)

[*Figura 5.* Arquitectura Smart Campus. 39](#_Toc8410406)

**Lista de Apéndices**

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

Apéndice A. Ejemplo de apéndice.

**Lista de Siglas**

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

**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).

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 casos de uso y los dispositivos asociados a los mismos permitiéndoles de una manera sencilla crear aplicaciones que contribuyan a generar esta transformación digital.

\*Trabajo de grado

\*\*Facultad de Ingenierías Físicomecánicas. 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, APLICACIÓN, WEB

**DESCRIPTION:**

Lorem ipsum dolor sit amet, consectetur adipiscing elit. In id venenatis augue. Praesent ut sagittis justo, non efficitur orci. Suspendisse non justo et elit cursus semper sed a ante. Quisque pharetra, libero elementum consectetur consectetur, ligula lectus consequat urna, nec elementum nulla lacus eu nunc.

Mauris mollis aliquam tellus vitae imperdiet. Suspendisse blandit et felis at faucibus. Cras condimentum pretium convallis. Quisque eu neque elit. Nullam non urna tortor. Sed hendrerit diam at nunc rutrum, vitae lacinia ex bibendum.

\*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, 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, monitorizar el tráfico y la calidad del aire en ciudades, entre otros.

El Internet de las Cosas 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.

En este proyecto se presenta el diseño de una aplicación web para la administración de una arquitectura 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 arquitectura a los dispositivos y viceversa, permitiendo que los diferentes casos de uso de IoT sean implementados sobre esta arquitectura extensiva y 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.

## 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; 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 era promover el uso de protocolos de internet en objetos inteligentes, iniciando el camino para que esta corriente se convirtiera en una realidad.

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 estaba vacía. 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; *Smart Parking* de Telensa, donde mediante sensores se monitorea e informa a los usuarios los lugares ocupados y disponibles para parquear, contribuyendo a una mayor eficiencia en el parqueo de vehículos y descongestión de vías. Otros campos en los que esta tendencia es aplicada pretende dar cierta inteligencia a Edificios, Casas y Granjas, entre otras que permitiría resolver problemas cotidianos, reduciendo costos y/o generar 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 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 por la compatibilidad existente con diversos dispositivos utilizados (computadores, tablets, celulares), por la sencillez y gran crecimiento de las posibilidades del desarrollo web que permiten 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 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.



*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 para colectar y usar información para gestionar recursos de una manera eficiente e incluso contribuir en labores de investigación y desarrollo. Este concepto es básicamente 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.

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. 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 diversos 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, pensado para el desarrollo de aplicaciones robustas por lo cual es definido como un superset de Javascript.

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), que es orientado a objetos, por lo que soporta 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 Aplicaciones Web Progresivas.

Las Aplicaciones Web Progresivas (PWA) son híbridos entre aplicaciones web y aplicaciones nativas pues son en realidad páginas web que mediante tecnologías (Service Workers) se comportan como si fueran aplicaciones instaladas en los dispositivos de forma que pueden seguir ejecutándose en segundo plano, por fuera del navegador y pueden ser instaladas en el dispositivo.

Una principal característica de estas es el funcionamiento sin conexión a internet, donde el usuario puede ver el contenido cargado previamente y probar una versión más ligera en dispositivos móviles sin necesidad de descargar una aplicación nativa que es en general más pesada.

## 3.8 Servicios REST

Es un estilo arquitectural independiente del lenguaje de programación para desarrollar Servicios Web. Define un conjunto de principios arquitectónicos por los que se diseñan estos servicios enfocándose en los recursos del sistema incluyendo como se accede a la información y el estado de los mismos y cómo estos son transmitidos por HTTP a clientes que utilizan diversos lenguajes de programación. Hay cuatro principios básicos de diseño de servicios REST:

* Utiliza métodos HTTPs.
* Son *Stateless* (no mantienen el estado entre varias peticiones).
* Expone URIs con forma de directorios.
* Transfiere la información en formato JSON.

## 3.9 WebSockets

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

## 3.10 Broker de Mensajería

Se puede definir como un *middleware* (software que ayuda a interactuar o comunicarse 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.

Además, actúa como un mediador entre las comunicaciones de las aplicaciones, minimizando el grado de conocimiento entre ellas.

## 3.11 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 enviarán a determinados *publishers* que previamente así se lo hayan solicitado al *broker*.

Los *publishers* envían los mensajes a un canal llamado *topic*. Los *subscribers* (suscriptores) pueden leer esos mensajes.

# 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 Fase 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 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 Fase 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 tanto del proyecto Smart Campus como de la plataforma 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 Smart Campus.
* Definición de los requerimientos, funcionalidades y arquitectura de la aplicación Web.

## 4.3 Fase 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 Fase 4: Validación y verificación

Para esta fase se verificó si el procedimiento, la metodología usada y la implementación del prototipo cumple los requerimientos establecidos y, se revisó que 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 Arquitectura 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.
* Reporte de resultados.

# 5. Requisitos y especificaciones

En este capítulo se detallan los requerimientos funcionales y no funcionales de la Aplicación Web de Administración de la Arquitectura IoT para Smart Campus. A demás se muestran las especificaciones logradas en la implementación final respecto a los requisitos.

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

## 5.1 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 y 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 exporte de 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 |

## 5.2 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 |

## 5.3 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 y entendible y ayudas al usuario donde se explican las funcionalidades. |
| 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

La visión de este proyecto es el diseño de un prototipo de una Aplicación Web para administrar una arquitectura 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 y 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 arquitecturas para IoT, 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 dispositivo intermedio entre estos y el Backend Cloud conocido como Gateway.
* Capa Gateway: Puente entre los dispositivos hardware y el Backend Cloud. Recibe la data de los sensores, realiza un procesamiento pequeño para disminuir la carga del servidor centralizado y luego le comunica la información al mismo 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 4.* Componentes Software Arquitecturas IoT. Tomada de Singh. (2019)

En la Figura 4 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 y transformarla y finalmente almacenarla respectivamente. Finalmente, la capa de Presentación o Aplicación donde se puede gestionar la Plataforma y visualizar la información de esta.

### 6.2.3 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 las razones para seleccionar estas.

* Lenguajes de Programación:

Una tendencia del desarrollo de aplicaciones web es la renderización del lado del cliente; lo cual 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, flexibilidad usando polimorfismo, crear mejores abstracciones de objetos/elementos cotidianos por medio de clases o interfaces, entre otros y evitar algunos posibles errores por ser también fuertemente tipado y compilado, no interpretado como otras de las alternativas; a demás 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 en la cual la aplicación solicita la información necesaria al servidor y este le responde y el canal de comunicación se cierra inmediatamente y donde la información no es actualizada si esta cambia. Esto se realizó mediante peticiones HTTP REST realizadas asíncronamente.

El segundo tipo de comunicación se realiza utilizando un protocolo Publish & Subscribe, esta para permitir 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 un mensaje 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 sobre WebSockets, pues MQTT es uno de los protocolos más 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.

* Frameworks:

Para seleccionar Frameworks 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 es el Framework que nativamente soporta 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; tiene una comunidad muy grande y está orientado en 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 que reduce la cantidad de tiempo necesaria para iniciar el mismo. A demás con este framework es muy sencillo convertir esta aplicación en una Aplicación Web Reactiva, y posee un gran número de librerías para realizar diversas funciones e integración con diversas librerías conocidas.

* 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 otros componentes software sean realizadas en segundo plano, para evitar que el Hilo principal de la aplicación sea bloqueado y la aplicación 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 permitir construir una aplicación web responsiva se seleccionó una pequeña librería llamada Angular Flex Layout, con la cual mediante Directivas se especifica el tamaño y demás propiedades 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 sin importar el tamaño de la pantalla.

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 SO Android. Usando a su vez la librería Angular Material, optimizada para el uso con el framework seleccionado, que provee componentes personalizables estilizados con este patrón que permiten acelerar el proceso de desarrollo y reducción de duplicación de código.

### 6.2.4 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.

## 6.2 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.

### 6.2.3 Definición de actores.

Los actores identificados son:

* Administrador
* Superusuario

Tabla 1.

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 2.

Descripción de actor Superusuario

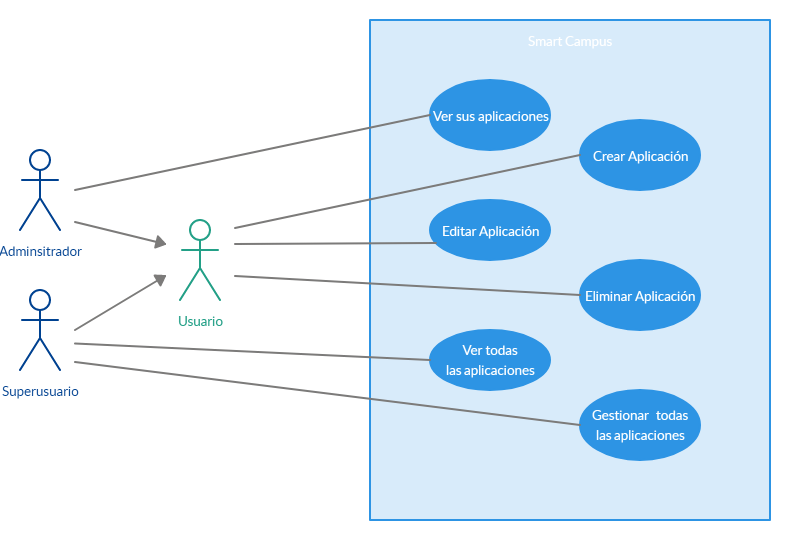
|  |  |
| --- | --- |
| **Actor:** | Superusuario |
| **Descripción:** | Usuario con permisos para ver y gestionar todos los usuarios, aplicaciones, gateways, dispositivos, datos y notificaciones de la aplicación. |
| **Tipo:** | Primario |

### 6.2.4 Definición de casos de uso.

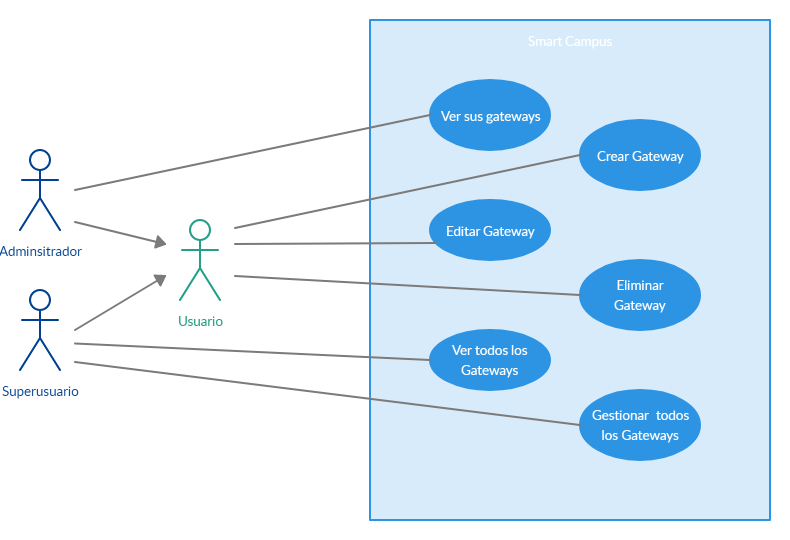
Una vez determinados los actores del sistema, se procedió a la identificación y diseño de los casos de uso.



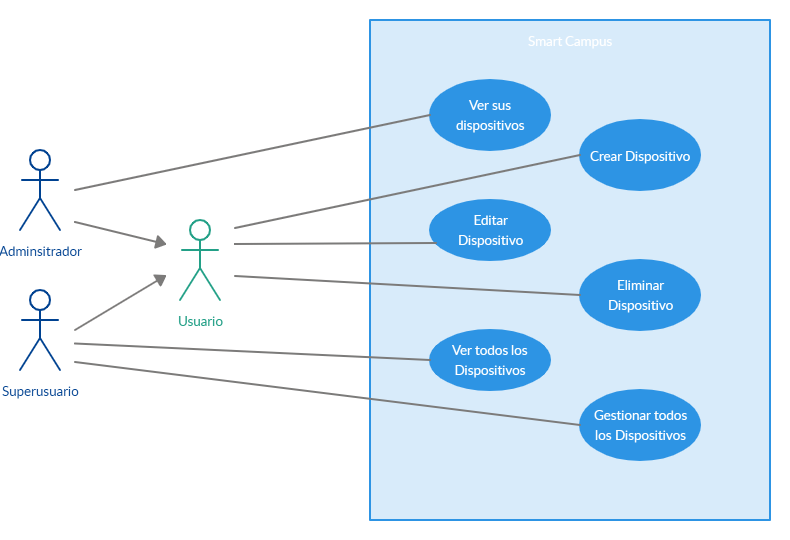
*Figura 5.* Diagrama Casos de Uso Módulo Usuario.



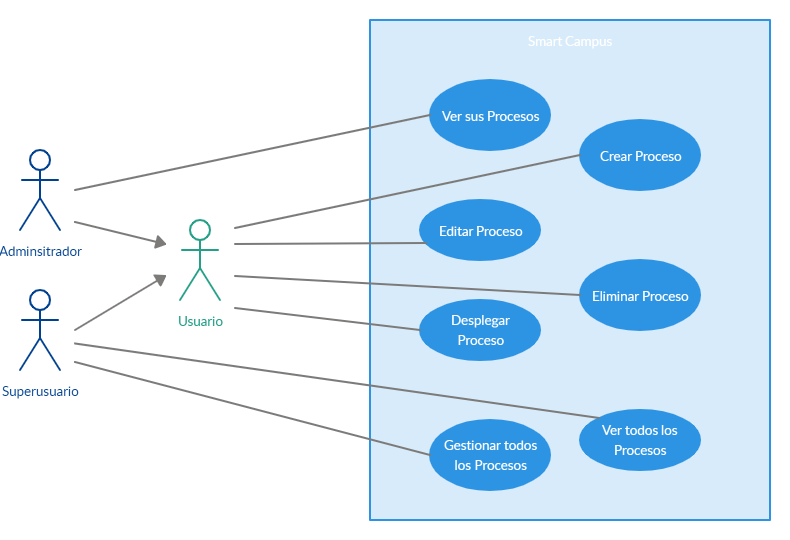
*Figura 6.* Diagrama Casos de Uso Módulo Aplicaciones.



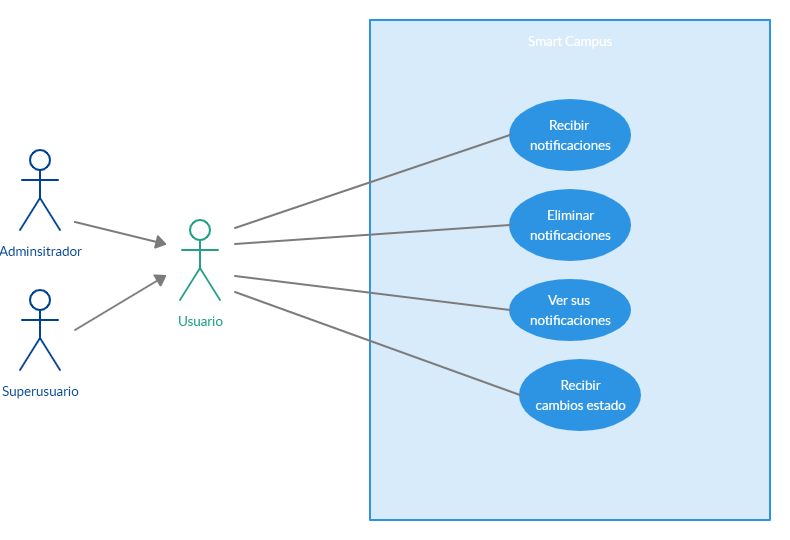
*Figura 7.* Diagrama Casos de Uso Módulo Gateways



*Figura 8.* Diagrama Casos de Uso Módulo Dispositivos



*Figura 9.* Diagrama Casos de Uso Módulo Procesos



*Figura 10*. Diagrama Casos de Uso Módulo Notificaciones



*Figura 11.* Diagrama casos de Uso Módulo de Datos y Estadísticas

## 6.3 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 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.

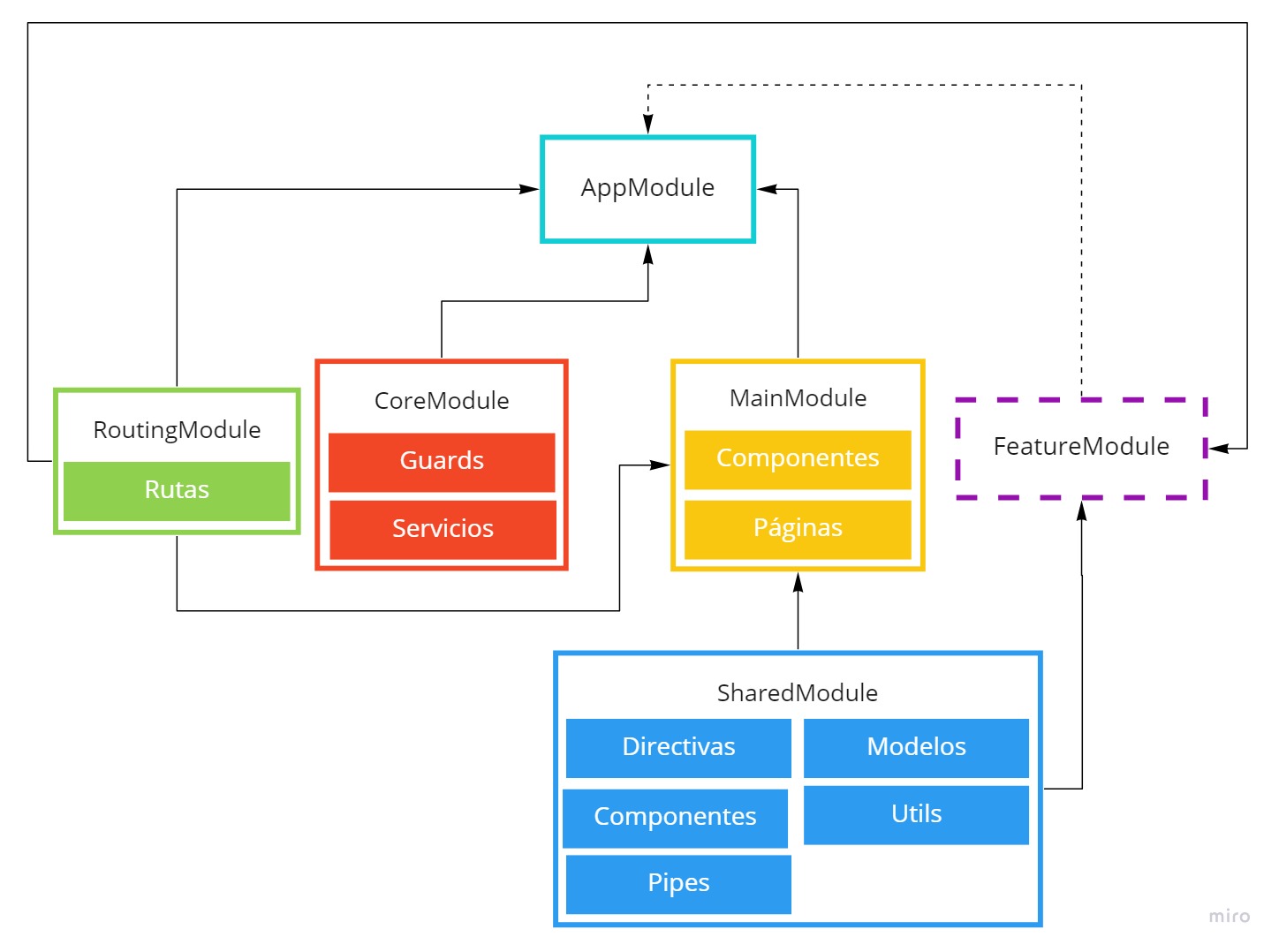


*Figura 12.* Arquitectura Smart Campus.

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 y finalmente, una arquitectura de Despliegue de la plataforma IoT en una infraestructura Cloud de alta disponibilidad en un entorno distribuido (Arias y Estupiñan, 2019).
* **Despliegue:** En este trabajo de investigación 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.



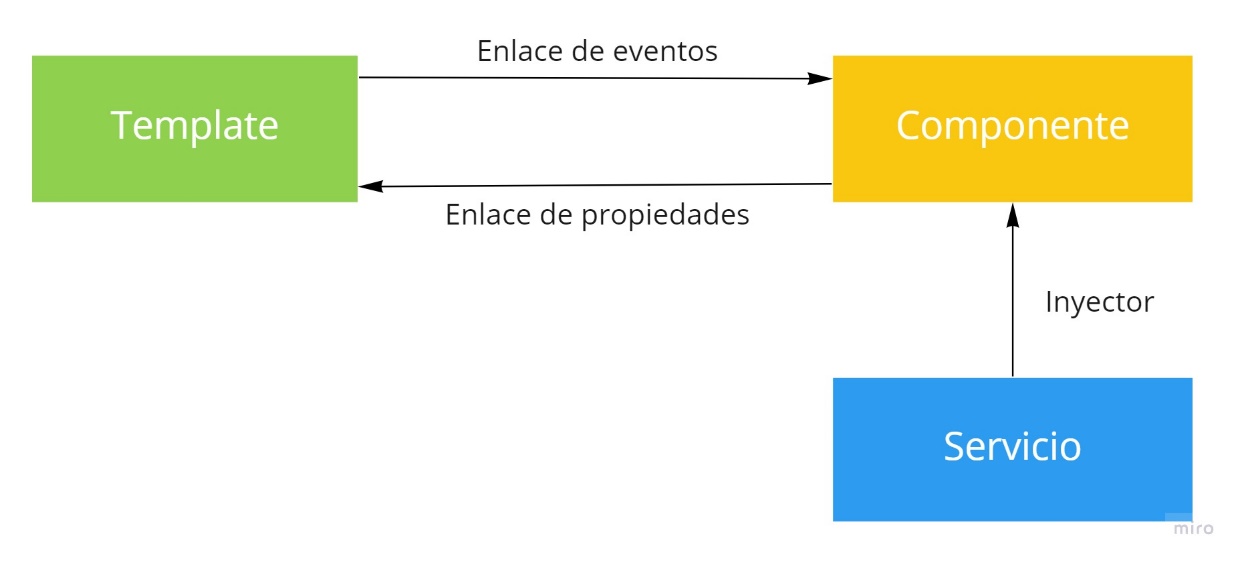
*Figura 13.* Arquitectura Aplicación Web

Como se observa en la imagen, se planteó una arquitectura por módulos donde cada uno tiene funcionalidades específicas y son importados de acuerdo con las necesidades. Estos son:

* **AppModule:** Módulo central de la aplicación, en el se importan todos los demás módulos y contiene el componente central de la aplicación, 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 para poder acceder y cargar a 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 y 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 descarga 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 configurados por un Template, donde se modela el HTML que se renderiza para ese componente, la lógica o Controlador del componente, los estilos, pues este framework por defecto hace que el alcance de estos estilos sea aplicado únicamente dentro del componente, para evitar que afecte n 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 diversos métodos, entre estos llamados HTTP.

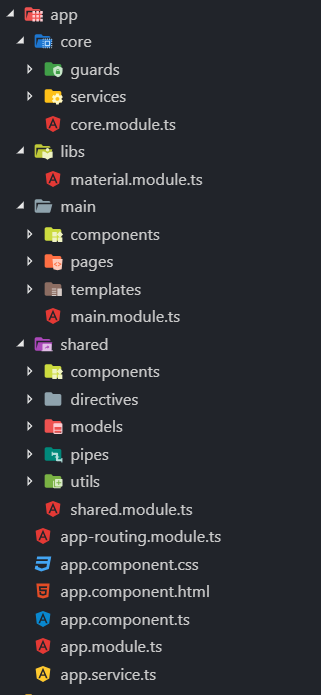
A continuación, se presenta la interacción entre el Componente (lógica), el Template y los Servicios.



*Figura 14.* 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 refrescar toda la página, también hay un enlace en la otra dirección, donde el template mediante eventos notifica que hubo un cambio o que el usuario interactuó con algún elemento, lo que causa que se ejecute una función o se realice una petición en el código. 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 folders y módulos.



*Figura 15.* Estructura de folders 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 o *root* 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.

# 7. Validación

# 8. Trabajo a futuro

# 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, que permite hacer una aplicación web fácilmente extensible 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.

# 10. Recomendaciones (Opcional)

(Va en capitulo separado de las conclusiones y en este apartado se expresa las perspectivas del autor a fin de complementar con nuevas ideas a la investigación original)

# Referencias Bibliográficas

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/>.

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_>.

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/>.

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>.

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/>.

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/>.

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>.

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/>.

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

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/>.

Ramírez, I. (2019). ¿Qué es una Aplicación Web Progresiva o PWA? Xataka.com. Recuperado el 29 de abril de 2019, <https://www.xataka.com/basics/que-es-una-aplicacion-web-progresiva-o-pwa>.

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/>.

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

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/>.

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/>.

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

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

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

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