Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería en Ciencias y Sistemas

IMPLEMENTACIÓN DEL MUEBLE URBANO PARA LA PRESENTACIÓN DE LA INFORMACIÓN (MUPI) INTERACTIVO, DENTRO DE LA UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

Luis Fernando Lara Lemus
Asesorado por el Ing. Edgar René Ornélis Hoíl

Guatemala, mayo de 2018

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

IMPLEMENTACIÓN DEL MUEBLE URBANO PARA LA PRESENTACIÓN DE LA INFORMACIÓN (MUPI) INTERACTIVO, DENTRO DE LA UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

TRABAJO DE GRADUACIÓN

PRESENTADO A LA JUNTA DIRECTIVA DE LA
FACULTAD DE INGENIERÍA
POR

LUIS FERNANDO LARA LEMUS
ASESORADO POR EL ING. Edgar René Ornéliz Hoíl

AL CONFERÍRSELE EL TÍTULO DE

Ingeniero en Ciencias y Sistemas

GUATEMALA, MAYO DE 2018

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

NÓMINA DE JUNTA DIRECTIVA

DECANO	Ing. Pedro Antonio Aguilar Polanco
VOCAL I	Ing. Angel Roberto Sic García
VOCAL II	Ing. Pablo Christian de León Rodríguez
VOCAL III	Ing. José Milton de León Bran
VOCAL IV	Br. Óscar Humberto Galicia Núñez
VOCAL V	Br. Carlos Enrique Gómez Donis
SECRETARIA	Inga. Lesbia Magalí Herrera López

TRIBUNAL QUE PRACTICÓ EL EXAMEN GENERAL PRIVADO

DECANO	Ing. Pedro Antonio Aguilar Polanco
EXAMINADOR	Ing. Marlon Francisco Orellana López
EXAMINADOR	Ing. César Rolando Batz Saquimux
EXAMINADOR	Ing. José Alfredo González Díaz
SECRETARIA	Inga. Lesbia Magalí Herrera López

HONORABLE TRIBUNAL EXAMINADOR

En cumplimiento con los preceptos que establece la ley de la Universidad de San Carlos de Guatemala, presento a su consideración mi trabajo de graduación titulado:

IMPLEMENTACIÓN DEL MUEBLE URBANO PARA LA PRESENTACIÓN DE LA INFORMACIÓN (MUPI) INTERACTIVO, DENTRO DE LA UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

Tema que me fuera asignado por la Dirección de la Escuela de Ingeniería en Ciencias y Sistemas, con fecha 02 de agosto 2016.

Luis Fernando Lara Lemus

Guatemala 27 de octubre de 2017

Ingeniero
Carlos Azurdia
Escuela de Ciencias y Sistemas
Facultad de Ingeniería
Universidad de San Carlos de Guatemala

Ingeniero Carlos Azurdia:

Me complace saludarle, haciendo referencia al trabajo de graduación titulado "**Implementación del Mueble Urbano para la Presentación de la Información (MUPI), Interactivo, dentro de la Universidad de San Carlos de Guatemala**", desarrollado por el estudiante universitario Luis Fernando Lara Lemus con número de carné **201114814** y DPI **2158 34514 0101**, que como asesor apruebo el contenido del mismo.

Para su conocimiento y efectos, sin otro particular, me suscribo.

Ing. Edgar René Ornelis Hoíl
Colegiado 4830
Asesor

Universidad San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería en Ciencias y Sistemas

Guatemala, 15 de Noviembre de 2017

Ingeniero
Marlon Antonio Pérez Türk
Director de la Escuela de Ingeniería
En Ciencias y Sistemas

Respetable Ingeniero Pérez:

Por este medio hago de su conocimiento que he revisado el trabajo de graduación del estudiante **LUIS FERNANDO LARA LEMUS** con carné **201114814 y CUI 2158 35514 0101,** titulado **"IMPLEMENTACIÓN DEL MUEBLE URBANO PARA LA PRESENTACIÓN DE LA INFORMACIÓN (MUPI) INTERACTIVO, DENTRO DE LA UNIVERSIDAD DE SAN CARLOS DE GUATEMALA"** y a mi criterio el mismo cumple con los objetivos propuestos para su desarrollo, según el protocolo.

Al agradecer su atención a la presente, aprovecho la oportunidad para suscribirme,

Atentamente,

Ing. Carlos Alfredo Azurdia
Coordinador de Privados
y Revisión de Trabajos de Graduación

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
INGENIERIA EN CIENCIAS Y SISTEMAS
COORDINACION TESIS

ESCUELA DE INGENIERÍA EN CIENCIAS Y SISTEMAS

UNIVERSIDAD DE SAN CARLOS
DE GUATEMALA

FACULTAD DE INGENIERÌA
ESCUELA DE INGENIERÍA EN
CIENCIAS Y SISTEMAS
TEL: 24767644

El Director de la Escuela de Ingeniería en Ciencias y Sistemas de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer el dictamen del asesor con el visto bueno del revisor y del Licenciado en Letras, del trabajo de graduación ***"IMPLEMENTACIÓN DEL MUEBLE URBANO PARA LA PRESENTACIÓN DE LA INFORMACIÓN (MUPI) INTERACTIVO, DENTRO DE LA UNIVERSIDAD DE SAN CARLOS DE GUATEMALA"****, realizado por el estudiante, LUIS FERNANDO LARA LEMUS aprueba el presente trabajo y solicita la autorización del mismo.*

"ID Y ENSEÑAD A TODOS"

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
DIRECCION DE INGENIERIA EN CIENCIAS Y SISTEMAS

Ing. Marlon Antonio Pérez Türk
Director
Escuela de Ingeniería en Ciencias y Sistemas

Guatemala, 23 de mayo de 2018

Universidad de San Carlos
de Guatemala

Facultad de Ingeniería
Decanato

Ref.DTG.D.182.2018

El Decano de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer la aprobación por parte del Director de la Escuela de Ingeniería en Ciencias y Sistemas, al trabajo de graduación titulado: **IMPLEMENTACIÓN DEL MUEBLE URBANO PARA LA PRESENTACIÓN DE LA INFORMACIÓN (MUPI) INTERACTIVO, DENTRO DE LA UNIVERSIDAD DE SAN CARLOS DE GUATEMALA,** presentado por el estudiante universitario: **Luis Fernando Lara Lemus,** y después de haber culminado las revisiones previas bajo la responsabilidad de las instancias correspondientes, se autoriza la impresión del mismo.

IMPRÍMASE:

Ing. Ángel Roberto Sic García
Decano en Funciones

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
DECANO
FACULTAD DE INGENIERIA

Guatemala, mayo de 2018

/cc

ACTO QUE DEDICO A:

Dios	Por permitirme haber llegado hasta este punto importante en mi vida, guiando y cuidando cada paso que doy.
Mis padres	Aura Lemus y Luis Gustavo Lara por haberme brindado todo su apoyo en los buenos y malos momentos, sin ustedes este logro no hubiera sido posible.
Mis hermanas	Georgi Lara y Wendy Lara por alentarme a siempre seguir adelante, creer en mí y brindarme consejos valiosos.

AGRADECIMIENTOS A:

Universidad de San Carlos de Guatemala	Por brindarme la oportunidad de pertenecer a esta gloriosa universidad y otorgarme este título académico.
Facultad de Ingeniería	Por abrirme las puertas de esta facultad y haberme brindado todo el equipo necesario para que desarrollara mis conocimientos.
Mis amigos de la Facultad	Por haberme ayudado incondicionalmente desde el inicio hasta el final de la carrera.

ÍNDICE GENERAL

ÍNDICE DE ILUSTRACIONES

FIGURAS

TABLAS

LISTA DE SÍMBOLOS

Símbolo	Significado
GB	Gigabytes
MB	Megabytes

GLOSARIO

MUPI	Mueble Urbano para la Presentación de la Información.
Hardware	Elementos físicos de una computadora, es decir, que se pueden tocar.
Software	Elementos lógicos de una computadora, son internos y no se pueden tocar solo usar.
Raspberry Pi	Computadora de pequeño tamaño y accesible a un bajo costo.
USB	Por sus siglas en inglés *Universal Serial Bus*, es un dispositivo de almacenamiento de datos.
Tarjeta SD	Por sus siglas en inglés *Secure Digital,* es un dispositivo de almacenamiento de datos para dispositivos portátiles.
HDMI	Por sus siglas en inglés, High-Definition Multimedia Interface. Es un estándar en la transmisión de audio y video digital.
SoC	Por sus siglas en inglés, *System on Chip*. Es un dispositivo que reúne múltiples componentes de una computadora en un mismo chip.
RAM	Por sus siglas en inglés, *Random Access Memory*. Es un tipo de memoria volátil, rápida y de mayor costo utilizada en una computadora.
DSI	Por sus siglas en inglés*, Display Serial Interface*. Es una interfaz de alta velocidad para la transmisión de video.

GPIO	Por sus siglas en inglés, *General Purpose Input/Output*. Es una serie de pines programables con los que cuenta la Raspberry Pi.
MySQL	Sistema de gestión de base de datos relacional analizado en este trabajo.
PostgreSQL	Sistema de gestión de base de datos relacional de código abierto analizado en este trabajo.
Oracle	Empresa estadounidense de software corporativo.
Workbench	Herramienta visual útil para gestionar la base de datos MySQL.
SSL	Por sus siglas en inglés, *Secure Sockets Layer*, es un protocolo que permite encriptar datos.
CPU	Por sus siglas en inglés, *Central Processing Unit*. Chip interno de una computadora que procesa las instrucciones.
PHP	Lenguaje de programación utilizado del lado del servidor.
JavaScript	Lenguaje de programación utilizado del lado del cliente y servidor.
IIS	Servidor web de Microsoft.
Apache	Servidor web de código abierto para Windows, Unix y Macintosh.
Nginx	Servidor web de código abierto.
Ethernet	Estándar que define las características de cableado en una red de área local.
UML	Lenguaje de modelado de sistemas.
Framework	Es un patrón o una estructura definida para el desarrollo o implementación de una aplicación.
MVC	Patrón de diseño Modelo, Vista y Controlador.

DOM	Interfaz para documentos HTML y XML. Permite la modificación de su contenido.
HTTP	Protocolo de comunicación que permite la transferencia de información por internet.
SOAP	Protocolo que define como dos objetos en diferentes procesos se pueden comunicar a través de datos XML.
Plugin	Complemento o aplicación que proporciona a otra aplicación alguna funcionalidad.
URI	Por sus siglas en inglés, *Uniform Resource Identifier*. Cadena de texto que identifica un recurso en la red.
Socket	Medio por el cual dos programas pueden intercambiar datos de forma ordenada.
POST	Método de peticiones HTTP
Carousel	Objeto para mostrar imágenes de forma cíclica.
Stakeholder	Parte interesada en el proyecto, puede ser una persona, organización o empresa.

RESUMEN

En publicidad existe un medio de comunicación muy utilizado que está sujeto a cambios y mejoras con la ayuda de la tecnología, a este medio para informar se le conoce como MUPI (Mobiliario Urbano para la presentación de la Información), puede ser implementado y llamado interactivo con ayuda de una Raspberry Pi, ya que puede obtener todas las funcionalidades de una computadora a un bajo costo y con un pequeño tamaño.

Una vez que se tiene el hardware requerido para crear un MUPI es necesario desarrollar un software que facilite a los estudiantes consultar la información y por otra parte permita a los usuarios administradores subir y gestionar la información a la plataforma, mostrar la creación e implementación paso a paso del software que permita lo mencionado anteriormente es el objetivo de este documento, en el que se observan cuatro capítulos que muestran las fases que cumplió el desarrollo de la aplicación.

En el primer capítulo se puede encontrar una serie de conceptos y antecedentes que hablan sobre el origen y evolución, tanto del MUPI como de la Raspberry Pi. Estos conceptos son necesarios para comprender de una mejor manera el alcance que puede tener el desarrollo de la aplicación y la implementación del sistema dentro de la universidad.

Luego de conocer más sobre el MUPI y Raspberry Pi se puede empezar a diseñar y proponer una solución a las necesidades encontradas, es por eso que en el segundo capítulo se documenta el análisis y diseño de la aplicación, mostrando los requerimientos planteados y las soluciones propuestas a cada requerimiento. En este capítulo se puede observar los requerimientos

funcionales y no funcionales, los diagramas necesarios para comprender el desarrollo de la aplicación, que van desde los prototipos de las pantallas hasta un diagrama de arquitectura.

Una vez habiendo analizado la solución propuesta, como en todo ciclo de desarrollo de software, es posible empezar a programar cada requerimiento planteado, es por eso que en el tercer capítulo se muestra el proyecto que da solución a los requerimientos, se puede observar la explicación técnica de cada archivo programado y la funcionalidad que cumple dentro de todo el proyecto.

Por último, en el cuarto capítulo se explican los casos de prueba planteados para verificar la funcionalidad de cada requerimiento, es decir, se prueba cada funcionalidad de la aplicación en busca de errores o bien de mejoras. Esto se realiza con el objetivo de constatar que la aplicación funciona y satisface las necesidades que fueron planteadas al inicio del documento.

OBJETIVOS

General

Proveer a la Universidad de San Carlos de Guatemala una plataforma de bajo costo y escalable para la integración y presentación de información a los estudiantes.

Específicos

1. Crear un sistema Web intuitivo que, con ayuda de una pantalla táctil o teclado y mouse, el usuario pueda observar y seleccionar la información de su interés.

2. Desarrollar un sistema Web para que los usuarios designados por el departamento correspondiente, puedan administrar la información que se publica y la que observan los estudiantes.

3. Proporcionar al sistema la capacidad de administrar la información en tiempo real, y con esto, presentar siempre información actualizada.

INTRODUCCIÓN

Desde la antigüedad el ser humano ha tenido la necesidad de expresar o comunicar sus ideas y pensamientos a los demás, es por eso que existen distintos medios de comunicación y dentro de la publicidad exterior se encuentra el concepto de MUPI (Mobiliario Urbano para la presentación de la Información), que consiste en un cartel iluminado con un soporte de metal expuesto al exterior, generalmente en las calles de las ciudades, el cual contiene publicidad.

Hoy en día la tecnología avanza a pasos agigantados, en un lapso corto de tiempo se tienen mejores productos que satisfacen de una forma más efectiva las necesidades de las personas. Un ejemplo de lo mencionado anteriormente se puede observar en la Raspberry Pi, es una computadora que tiene el tamaño de una tarjeta de crédito y todos los componentes indispensables de una computadora, cuenta con un bajo precio lo que la hace accesible a cualquier usuario.

La Universidad de San Carlos de Guatemala al tener un campus central muy grande y contar con distintos centros regionales en diferentes departamentos del país, es muy fácil perderse o bien no es fácil encontrar el destino que los estudiantes buscan, también existen ocasiones en que la información no es transmitida a todos los estudiantes. Es por eso que surge la necesidad de crear algo que facilite la orientación y presentación de la información a la comunidad estudiantil y personas que visitan la universidad.

Si se unen los dos conceptos mencionados al inicio; MUPI y Raspberry Pi, se puede obtener un medio de comunicación interactivo, que permite mostrar información a los usuarios y que estos interactúen en las calles con los dispositivos. Adicional es necesario crear una plataforma para facilitar a los usuarios el ingreso y publicación de información, así como una plataforma para observar la publicidad de una manera fácil e intuitiva. Lo mencionado anteriormente es el objetivo de este documento; mostrar paso a paso el desarrollo de una plataforma que pueda ser implementada en una Raspberry Pi y esta se convierta en un MUPI para la interacción con los usuarios, visualización de la publicidad y una plataforma para el ingreso y administración de la información a mostrar.

1. MARCO TEÓRICO

1.1. Publicidad Exterior y sus inicios

Kleppner (1993): “la publicidad exterior no sólo es la forma de publicidad más antigua, sino también la forma más antigua de comunicación de masas. Las pinturas en las paredes de las tumbas y las tablas de arcilla con inscripciones para el público datan de la época del Antiguo Imperio Egipcio, de hace 5 000 años”. También señala que con la llegada del automóvil y nuevas técnicas de impresión se origina la era moderna de la publicidad exterior, que posteriormente evolucionó en pizarras de carretera, y se convirtieron en la forma más común de publicidad exterior.

Como se puede observar, la publicidad exterior ha sido desde el inicio, con las culturas antiguas, un medio de comunicación para transmitir algún mensaje a un grupo de personas. Este medio ha ido evolucionando desde pinturas de paredes, en las civilizaciones antiguas, pasando por pizarras, luego, con el surgimiento y popularización de la imprenta, en carteles y ahora en nuestros días, con la accesibilidad y eficiencia de lámparas de bajo consumo energético, se puede observar los MUPIS.

1.1.1. ¿Qué es un MUPI? y sus características

El Mobiliario Urbano para la Información (MUPI) es un tipo de publicidad exterior, consiste en un soporte de metal que contiene un cartel iluminado y tiene por objetivo dar a conocer a un grupo de personas un producto o servicio. Barranza (2007), en su tesis “SIMILITUDES Y DIFERENCIAS DE LAS GIGANTOGRAFÍAS Y MUPIS DE LA ZONA 10, COMO UNA OPCIÓN DE PUBLICIDAD EXTERIOR” define MUPI como un medio exterior que se

encuentra ubicado en las paradas de autobuses, se han convertido en una tendencia publicitaria y estética. Los MUPIS son rótulos verticales, los cuales tienen una imagen, un mensaje corto y están iluminados por la noche.

Siguiendo con el trabajo realizado por Barranza, quien considera que un MUPI cuenta con las siguientes características:

- Dinámico: Porque cada 14 días la cartelera cambia, lo que permite que el mensaje no se vuelva parte del paisaje urbano.
- Impactante: Por su modernidad, estética, limpieza e iluminación. Es un foro muy atractivo para la creatividad.
- Rentable: Por su bajo costo en función al número de impactos recibidos por el *target group.*
- Recordable: Puesto que, al ser un medio de exposición voluntaria, el grupo objetivo verá los mensajes que llaman la atención incrementando su nivel de recordación.

1.1.2. MUPI en Guatemala

En nuestro país, la publicidad exterior, específicamente los MUPIS, es un campo muy bien aprovechado y que poco a poco ha ido en aumento, posicionándose como uno de los mejores medios para lanzar campañas publicitarias, debido a su amplio alcance y cobertura. Un claro alcance de los MUPIS se puede observar en los programas que implementa la Municipalidad de Guatemala por medio de la Unidad de Vía Pública, la cual se encarga de la regulación del Mobiliario Urbano Para la Información. Según el artículo de la Municipalidad de Guatemala y bajo el eslogan “Ciudad para Vivir” que se menciona en la tesis de Barranza (2007), los MUPIS, se empezaron a implementar desde el 01 de agosto del 2005, con los parabuses instalados en varios sectores de la ciudad.

Si bien la Municipalidad de Guatemala se encarga de regular todo el Mobiliario Urbano, existe en nuestro país algunas empresas que se dedican a proporcionar este servicio a instituciones privadas, de las cuales se puede mencionar a Equipamientos Urbanos de Guatemala (EUGUA), quienes en su página oficial de internet mencionan que gestionan la comercialización de publicidad exterior, en vía pública, siendo parte del Grupo Equipamientos Urbanos, reconocido como el medio masivo de comunicación exterior más importante en Latinoamérica. También se puede mencionar a *Dreams Innovación Digital*, quienes en su página oficial de internet mencionan que son una empresa que se dedica a suplir las necesidades tecnológicas de sus clientes, ofreciendo alternativas de vanguardia, con énfasis en *marketing* digital y herramientas digitales innovadoras.

1.2. Conociendo la Raspberry Pi

La Raspberry Pi es una computadora que tiene el tamaño de una tarjeta de crédito, cuenta con todos los componentes indispensables de una computadora, como lo es; puertos USB, puertos ethernet, puerto para tarjeta SD, salida de audio y salida video HDMI, su innovación se basa, además de su tamaño (85x56 mm) y su bajo costo (35 $), en su potente procesador capaz de reproducir videos en alta definición. Este dispositivo fue desarrollado por la Fundación Raspberry Pi con el objetivo de estimular el aprendizaje de la informática en las escuelas, la fundación menciona en su página oficial de internet “La Fundación Raspberry Pi trabaja para poner el poder digital en las manos de todas las personas del mundo, proporcionamos computadoras de bajo costo y alto rendimiento a las personas, para aprender, resolver problemas y divertirse.”

1.2.1. Historia de la Raspberry Pi

MOCQ en su libro “Raspberry Pi 2, Utilice todo el potencial de su nano-ordenador” relata la cronología del surgimiento de esta mini computadora, mencionando que del 2006 al 2008, Eben Upton, fundador de la Fundación Raspberry Pi, realizó varios prototipos de lo que se convertiría en la Raspberry Pi. Después en el 2008 el proyecto se integra en un SoC (circuito integrado que agrupa todos los componentes de un ordenador), en el 2011 se construyen cincuenta prototipos de tarjetas alpha, que contaban con características idénticas a las de la Raspberry Pi actual. A finales de octubre de 2011 el sistema operativo RISC OS (*Reduced Instruction Set Computer Operating System*) funcionaba en la Raspberry Pi y en diciembre del mismo año se fabrican 100 circuitos impresos del modelo B. A finales de diciembre de 2011 la Raspberry Pi demuestra su capacidad arrancando con un sistema operativo Linux y reproduciendo video de alta definición.

Como se observa en la cronología de MOCQ, el avance de la raspberry pi del 2011 al 2015 ha sido muy rápido, presentando mejoras en el hardware y también presentando nuevos dispositivos compatibles, un ejemplo puede ser el lanzamiento de la pantalla oficial en septiembre de 2015, y no hay que olvidar el lanzamiento más reciente, febrero 2016 de la Raspberry Pi 3, que incluye nuevas mejoras, como Wifi y Bluetooth integrados a la placa y un procesador de 64 bits.

Continuando con la cronología de MOCQ, a finales de febrero se anuncia la comercialización oficial de la Raspberry Pi, y surgen los siguientes acontecimientos:

- 29 de febrero de 2012: el primer lote se vende en pocos minutos y se registran más de 100 000 pedidos anticipados durante el día. Al mismo tiempo la Fundación revela el modelo A que tendrá 256 MB y no los 128 MB que tenían previstos.
- 15 de octubre de 2012: el modelo B ahora es anunciado con 512 MB de memoria RAM.
- 14 de mayo de 2013: es anunciada la cámara oficial para la Raspberry Pi.
- 03 de junio de 2013: se anuncia NOOBS (*New Out of Box Software*), un cargador de sistema operativo muy simple de utilizar.
- Julio del 2014: la fundación anuncia el modelo B mejorado, llamado B+, con 4 puertos USB y con tarjeta micro SD.
- Noviembre 2014: se anuncia el modelo A+, cuenta con un solo puerto USB y no cuenta con puerto Ethernet, sin embargo, las dimensiones de la placa son reducidas.
- Febrero 2015: la raspberry Pi 2 se anuncia con las mismas dimensiones que el modelo B+, con procesador de 4 núcleos y 1 GB de memoria RAM.

Como se observa en la cronología de MOCQ, el avance de la raspberry pi del 2011 al 2015 ha sido muy rápido, presentando mejoras en el hardware y también presentando nuevos dispositivos compatibles, un ejemplo puede ser el lanzamiento de la pantalla oficial en septiembre de 2015, y no hay que olvidar el lanzamiento más reciente, febrero 2016 de la Raspberry Pi 3, que incluye nuevas mejoras, como Wifi y Bluetooth integrados a la placa y un procesador de 64 bits.

1.2.2. Primer acercamiento con la Raspberry Pi

La raspberry pi tiene un tamaño reducido, sin embargo, cuenta con todos los componentes de una computadora común y es posible conectar en ella, una pantalla, un mouse, un teclado, una memoria, una cámara, etc. Upton, y Halfacree en el libro "*Raspberry Pi User Guide*" dedican un capitulo para explicar los principales componentes y conectores de la raspberry, donde explican las funcionalidades de audio, video, mouse y teclado, almacenamiento y conexión a internet.

Los autores Upton y Halfacree mencionan que la raspberry pi cuenta con tres distintos tipos de salida de video: video compuesto, HDMI video y DSI video.

El video compuesto está disponible a través de un puerto amarillo y plata en la parte superior de la Pi es conocido como RCA y está diseñado para funcionar con dispositivos antiguos. Una mejor calidad de imagen es obtenida usando la conexión HDMI (*High Definition Multimedia Interface*). Al contrario de la conexión a través de video compuesto, el puerto HDMI proporciona una conexión digital de alta velocidad. La última salida de video es la DSI (*Display Serial Interface*), como mencionan los autores Upton y Halfacree, este es un conector de cinta pequeño protegido por una capa de plástico, utilizado generalmente en pantallas de tablets y teléfonos inteligentes.

La raspberry cuenta con dos opciones de audio, la más simple de utilizar es por medio del conector HDMI, ya que esta tecnología, trasmite la señal de audio y video a la misma vez. Por otra parte, si se utiliza algún monitor que no tenga altavoces, será necesario que se utilice el puerto negro 3.5 mm de audio que provee audio análogo y es ideal si se conectó un monitor por medio de la salida de video compuesto. Como se menciona en la guía oficial de la

Raspberry, escrita por Upton y Halfacree, hasta el momento solo se ha mencionado los puertos de salida, sin embargo, la Raspberry necesita interactuar con el usuario, es por eso que es necesario hablar sobre los puertos de entrada. Los autores mencionan que tanto para conectar un mouse como un teclado es necesario hacerlo por medio de los puertos USB (*Universal Serial Bus*). El número de puertos USB depende del modelo de la Pi, en las primeras versiones se tenían uno o dos puertos, sin embargo, en los modelos dos y tres de la Pi, se cuenta con cuatro puertos.

La Raspberry Pi necesita un sistema operativo para funcionar, en este caso, un sistema operativo basado en GNU/Linux del que se hablara en la sección 1.3 de este trabajo, por lo cual es necesario preparar una tarjeta SD (*Secure Digital*) con un sistema operativo descargado desde la página oficial www.raspberrypi.org/downloads. El objetivo de este trabajo no es mostrar los comandos necesarios para realizar la acción mencionada anteriormente, sin embargo, es importante mencionar que el libro oficial para la Raspberry Pi, "Raspberry Pi *User Guide*" de los autores Upton y Halfacree, en su página 23, menciona las instrucciones necesarias para ingresar una imagen del sistema en una tarjeta SD. Esta tarjeta contiene el sistema operativo para que la Raspberry Pi funcione, sin embargo, también funge como dispositivo de almacenamiento, es decir, funciona como disco duro para nuestra Pi.

La conexión a internet en la Raspberry Pi, es a través del puerto RJ45, con el que cuenta los modelos B, Raspberry Pi 2 y 3, sin embargo, si se cuenta con el modelo A, se necesita comprar un adaptador USB que proporcione la conexión al cable RJ45. Para conectar la Pi a internet, es necesario conectar un extremo del cable RJ45 al puerto *ethernet* de la Pi, y el otro extremo al router. Como menciona Upton y Halfacree en la guía oficial de la Raspberry Pi, existe la posibilidad de conectar la Pi a través de una red inalámbrica, sin embargo, el

hardware necesario para realizar esta acción, no se encuentra disponible de fábrica en la placa, razón por la cual es necesario el uso de un adaptador Wifi conectado a un puerto USB. Con la Raspberry Pi 3, existe una excepción, ya que este modelo trae de fábrica en la placa la antena receptora de señal Wifi y bluetooth, además de contar con el puerto ethernet para conectar un cable RJ45.

Una de las razones por las que la Raspberry Pi es interesante y divertida, es debido a su puerto GPIO (*General Purpose Input/Output*), el cual consiste en una serie de pines que permiten interactuar a la Pi con el usuario. Como se menciona en la documentación oficial en la página de la Raspberry Pi, los pines son una interfaz física entre la Pi y el mundo exterior. Los pines del puerto GPIO es posible programarlos, existen librerías disponibles para una gran variedad de lenguajes de programación, lo que permite realizar una gran cantidad de proyectos, y también es una de las razones por la que las Raspberry está enfocada en el aprendizaje o educación; un ejemplo del alcance de este puerto lo podemos observar en "*The Projects Book*", un libro oficial donde se mencionan varios proyectos que se pueden lograr hacer con la Raspberry Pi.

1.3. GNU/Linux

El termino GNU/Linux hace referencia a la combinación del sistema operativo GNU, desarrollado por el proyecto GNU el cual es liderado por Richard Stallman, y el núcleo Linux, desarrollado por Linus Torvalds. Este término suele confundirse con el sistema operativo completo, como se menciona en la página oficial del proyecto GNU (www.gnu.org/gnu): "Linux es el núcleo: el programa del sistema que se encarga de asignar los recursos de la máquina a los demás programas que el usuario ejecuta. El núcleo es una parte

esencial de un sistema operativo, pero inútil por sí mismo, solo puede funcionar en el marco de un sistema operativo completo".

Es importante indicar y entender ¿Qué es un sistema operativo?, como menciona el autor Bermúdez en su libro "*Debian GNU/Linux Para el Usuario Final*", un sistema operativo es un conjunto de programas base que dirigen y controlan tanto el hardware como el software; es una capa intermedia entre estos dos componentes.

Complementando la definición de sistema operativo de la cual se habló en el párrafo anterior, es necesario comprender el significado de un núcleo (*kernel*) y la relación que guarda con un sistema operativo. Hunger en el libro "*Debian GNU/Linux Bible*" menciona lo siguiente: "El componente central de un sistema operativo es llamado núcleo en Unix y sistemas operativos parecidos a Unix. El núcleo se comunica con el hardware básico de una computadora como, el microprocesador, memoria, y los controladores de dispositivos. Toda interacción entre el hardware y cualquier programa debe ser negociado a través del núcleo. El núcleo tiene el cuidado de trasladar las peticiones en la forma específica en la que habla cada dispositivo.".

Algunas de las características más importantes de Linux, que se mencionan por parte del autor Pons en el libro "*Linux:* Principios básicos de uso del sistema" son:

- Multitarea: está diseñado para ejecutar varios programas al mismo tiempo. Utiliza un asignador para ejecutar varias acciones con un mismo procesador y también sacar partido de arquitecturas multiprocesador.

- Multiusuario: el sistema permite el uso por parte de varias personas de los recursos que administra. Estas personas se distribuyen en grupos de usuarios.
- Sistemas de archivos: soporta un gran número de sistemas de archivos, además de los de tipo Unix, Windows, NTFS y Macintosh.

1.3.1. GNU/Linux en la Raspberry Pi

Al día de hoy existe una gran variedad de sistemas operativos que cumplen y satisfacen necesidades específicas de cada usuario para los cuales fueron creados, bajo este contexto y teniendo en cuenta lo mencionado en la sección 1.3, donde se indicó que GNU/Linux fue desarrollado con el objetivo de crear un sistema completamente libre, se puede mencionar que la Fundación Raspberry Pi creo su propio sistema operativo basado en GNU/Linux para proporcionar a los usuarios un sistema operativo optimizado para interactuar con la Pi. Como se indica en el libro "*Hacking Raspberry Pi*" del autor Warner, "La Fundación Raspberry Pi desarrollo una distribución oficial de Linux que es optimizada para la Raspberry Pi; esta distribución es llamada Raspbian".

Es importante mencionar que en la página oficial de Raspberry Pi están disponibles para descargar gratuitamente una serie de sistemas operativos desarrollados por terceros como lo es *UBUNTU MATE, WINDOWS 10 IT CORE, RISC OS, LIBREELEC*, entre otros y que por lo general algunos cuentan con alguna funcionalidad en específico.

1.4. Red

En la actualidad la comunicación entre personas y dispositivos se ha vuelto una necesidad y el surgimiento de nuevas tecnologías de comunicación está a la orden del día. Es importante definir el concepto de comunicación, y como mencionan los autores Gil, Pomares y Candelas en su libro “Redes y transmisión de datos”, la comunicación se puede entender como un intercambio de información entre entidades, en donde participa un sistema emisor, un sistema receptor y existe un medio de transmisión.

Un sistema de comunicaciones, permite la transmisión de datos en el proceso de comunicación, y todo sistema de comunicación debe cumplir con tres aspectos: el primero es proveer una interfaz para poder transmitir la información, esta es imprescindible para poder transformar los datos que se necesita enviar, en bits. El segundo aspecto que debe cumplir es el de la implementación de mecanismos de sincronismos, es decir, encargase de que las señales eléctricas se transporten y lleguen a su receptor adecuadamente. Por último, el tercer aspecto es el de contar con mecanismos de direccionamiento y encaminamiento, esto asegurara que el mensaje viaje por el camino adecuado, o bien, si existe algún inconveniente en algún medio, el mensaje pueda llegar al mismo destino, pero por otro camino.

Como mencionan los autores Gil, Pomares y Candelas los objetivos de una red de comunicaciones entre computadoras son:

- Compartir recursos
- Aumentar la tolerancia a fallos, distribuyendo la información almacenada o empleando distintos caminos para evitar la pérdida de datos.

- Reducir el costo monetario
- Potenciar la globalización, siendo capaces de comunicar cualquier punto del mundo.

2. ANÁLISIS Y DISEÑO

En este capítulo se explica y se analiza los requisitos del proyecto, mostrando de una forma gráfica y detallada las funcionalidades, riesgos, restricciones y en general el funcionamiento del sistema.

2.1. Evaluación y comparación para la toma de decisión

Es importante dar a conocer las razones por las que se escogió el software que se menciona en la sección 2.7.2 principalmente el utilizado para la base de datos y el lenguaje de programación en el servidor. Se realizó un análisis, evaluación y comparación del software tomando en cuenta diferentes aspectos.

Para la base de datos se tiene la opción de MySQL y PostgreSQL, siendo la primera la opción propuesta por la entidad encargada de administrar los MUPIS, y la segunda opción, propuesta por mi persona. A continuación, se presentan los aspectos evaluados y su ponderación:

2.1.1. Precio

El pago por licencia es un aspecto muy importante en el desarrollo de software por tal razón se tiene este aspecto con una ponderación de cinco puntos. MySQL pertenece a Oracle y según las necesidades y prestaciones que se necesiten en la base de datos así es la necesidad de adquirir una licencia, sin embargo, ofrece una versión gratis con funcionalidades restringidas. Por otra parte, PostgreSQL es totalmente gratuita por tal razón presenta una mejor evaluación en este aspecto.

2.1.2. Configuración

Este aspecto recibe una ponderación de tres puntos ya que la configuración depende del desarrollador y del administrador de la base datos. Ambos gestores de base datos son sencillos e intuitivos de configurar e instalar contando con archivos de configuración que cumplen con fines específicos, por esta razón fueron evaluadas con la misma nota.

2.1.3. Mantenimiento

Este aspecto cuenta con una ponderación de dos puntos, debido a que el mantenimiento corre por cuenta del administrador de la base de datos y es independiente del software del MUPI, sin embargo, ambos gestores de base de datos cuentan con sistemas de gestión y monitoreo muy buenos. En este caso MySQL fue evaluado con una nota mayor ya que para la administración y mantenimiento de la base de datos cuenta con la herramienta *Workbench* la cual es más intuitiva y fácil de utilizar.

2.1.4. Soporte técnico

Al igual que en el punto anterior este aspecto se encuentra ponderado con dos puntos ya que el soporte técnico a la base de datos es independiente al software del MUPI. La base de datos PostgreSQL recibe una mejor nota debido a que cuenta con una gran comunidad de usuarios y desarrolladores que apoyan y dan solución a cualquier problema o duda que se tenga, al contrario de MySQL que a pesar que también cuenta con una gran comunidad de usuarios y desarrolladores que apoyan, también dependiendo de la licencia que se tenga así será es el soporte que se reciba.

2.1.5. Seguridad

Este aspecto se encuentra ponderado con 4 puntos ya que es muy importante los métodos con los que cuente el gestor de base de datos para proteger la información que guardan. Ambas bases de datos cuentan con métodos de encriptación SSL e integran sus propios métodos de protección de datos. La diferencia en la nota obtenida por PostgreSQL es debido que logra proteger los datos con lo que llaman *Row Level Security* de forma sencilla, esto permite definir políticas de seguridad a los usuarios para gestionar las filas de una tabla.

2.1.6. Aprovechamiento de recursos

Este aspecto cubre el rendimiento del CPU y memoria RAM que utiliza la base de datos, por lo tanto, cuenta con una ponderación de 5 puntos. MySQL cuenta con una mejor nota en este aspecto ya que basado en las comparativas y comentarios realizados en la página https://www.g2crowd.com este gestor de base de datos aprovecha de manera más eficientes los recursos del servidor.

2.1.7. Velocidad en lectura y escritura

Este aspecto se encuentra ponderado con 5 puntos ya que para la implementación del sistema del MUPI es fundamental la velocidad de respuesta y escritura que pueda brindar la base de datos, bajo estos argumentos MySQL fue concebida desde sus inicios para otorgar una gran velocidad sobre consultas sencillas o bien que siguen un cierto patrón. Cabe mencionar que el aspecto velocidad es muy difícil de medir ya que depende mucho del escenario de prueba.

A continuación, se presenta el resumen de los aspectos evaluados con su respectivo peso y ponderación, tomando como referencia los siguientes sitios de internet; www.digitalocean.com, www.db-engines.com, www.g2crowd.com, consultados en agosto de 2017, los cuales realizan comparaciones de software.

Tabla I. **Comparación entre base de datos**

Aspecto	Peso (0-5)	MySQL		PostgreSQL	
		Nota	Total	Nota	Total
Precio	5	7	35	10	50
Configuración	3	9	27	9	27
Mantenimiento	2	10	20	7	14
Soporte técnico	2	9	18	10	20
Seguridad	4	9	36	10	40
Aprovechamiento de recursos	5	10	50	8	40
Velocidad en lectura y escritura	5	10	50	8	40
Total			236		231

Fuente: elaboración propia.

Con respecto al lenguaje de programación se tiene también dos opciones, una de ellas muy conocida y utilizada durante muchos años y aún en la actualidad, me refiero a PHP. Por otra parte, se presenta la opción de utilizar JavaScript, una tecnología que, no teniendo mucho tiempo, es utilizada ampliamente para aplicaciones web innovadoras. A continuación, se presenta la comparación de las dos tecnologías:

2.1.8. Peticiones por segundo

Este aspecto fue ponderado con una nota de 5 puntos ya que es importante conocer la tecnología que permite procesar más respuesta en las peticiones realizadas al servidor. NodeJS al recibir las peticiones por un solo hilo y delegar el trabajo a hilos secundarios logra procesar a una mayor velocidad las peticiones siempre y cuando estas no conlleven un arduo procesamiento. Por otra parte, PHP es más eficiente al procesar peticiones que

demanden consumo de CPU pero menos rápido para responder. El sistema MUPI no requiere un arduo procesamiento de CPU ya que solo realiza consultas a una base de datos, razón por la cual la tecnología NodeJS recibe una mejor nota.

2.1.9. Consumo de CPU

Este aspecto cuenta con una nota de 5 puntos ya que es muy importante que el servidor no llegue a puntos críticos de consumo de recursos que puedan votar el servicio. Como se mencionó en la sección anterior PHP es más eficiente al momento de procesar peticiones que requieran un consumo de CPU alto, es por eso que recibe una mejor calificación, sin embargo, NodeJS recibe una calificación similar ya que el sistema que se trata de implementar no requiere de procesamiento de ningún algoritmo o funciones que demanden un consumo de CPU alto.

2.1.10. Soporte técnico

Este aspecto cuenta con una nota de 3 puntos debido a que no es un aspecto crítico para la elección de la tecnología, adicionalmente ambas tecnologías cuentan con una gran comunidad de desarrolladores que respalda y solventan cualquier duda. Es importante mencionar que, si bien PHP cuenta con más de 20 años en el mercado y ha sufrido grandes cambios desde sus inicios, NodeJS también ha crecido grandemente y es fuertemente utilizado en la industria desde su lanzamiento en 2009. Es por eso que ambas tecnologías fueron calificadas con la misma nota.

2.1.11. Flexibilidad

Por las mismas razones que el aspecto anterior, este, es ponderado con una nota de 3 puntos. PHP cuenta con una mayor flexibilidad ya que puede ser montado en varios servidores como IIS, Apache, Nginx, mientras que NodeJS instala sus propias librerías y servicios para utilizar JavaScript. Adicionalmente PHP cuenta aún con un mayor número de herramientas y plugins que solucionan necesidades de la comunidad. Debido a lo mencionado anteriormente PHP cuenta con una nota mayor a NodeJS.

A continuación, se presenta el resumen de los aspectos evaluados con su respectivo peso y ponderación, tomando como referencia los siguientes sitios de internet; www.benchmarksgame.alioth.debian.org, www.hostingadvice.com, www.belitsoft.com, consultados en agosto de 2017, estos realizan comparación de los softwares mencionados.

Tabla II. **Comparación entre lenguajes de programación**

Aspecto	**Peso (0-5)**	**PHP**		**NodeJS**	
		Nota	**Total**	**Nota**	**Total**
Peticiones por segundo	5	7	35	10	50
Consumo de CPU	5	10	50	8	40
Soporte técnico	3	10	30	10	30
Flexibilidad	3	10	30	9	27
Total			145		147

Fuente: elaboración propia.

2.2. Solución

En el tema anterior se observa la comparativa entre las posibles tecnologías a utilizar para desarrollar la solución, entre las que se encuentran MySQL y PostgreSQL como motores de bases de datos y PHP y NodeJS como lenguajes de programación del lado del servidor. Con respecto al motor de base de datos vemos a MySQL con una pequeña ventaja, sin embargo, el aspecto determinante para escoger este motor de base de datos para el desarrollo de la solución es la velocidad de lectura y escritura, en la que aventaja a su competidor PostgreSQL.

Por otra parte, se observa que NodeJS aventaja a PHP por una mínima diferencia, pero el aspecto determinante o bien de mayor importancia para el desarrollo de la solución MUPI, es la de peticiones por segundo en la cual aventaja NodeJS.

2.3. Usuarios identificados dentro del sistema

A continuación, se presentan los actores que interactúan en el sistema.

Tabla III. **Listado de actores**

Id	Nombre	Descripción
AC-01	Usuario administrador	Es la persona encargada de la administración de la información que se mostrará a través de los MUPIS, tiene habilidades para el manejo de software, adicionalmente conoce los límites y funcionalidades de todo el sistema.
AC-02	Usuario de MUPI	Es el usuario final del sistema, quien consultará información y tendrá interacción con el MUPI, no necesita experiencia en el manejo de software.

Fuente: elaboración propia.

2.4. Requerimientos funcionales

A continuación, se especifican los requerimientos funcionales identificados para el sistema, los cuales se obtuvieron en base a reuniones sostenidas con la entidad que se encargará de la administración de los MUPIS dentro la universidad.

Tabla IV. **Requerimientos funcionales**

Requerimientos Funcionales usuario AC-01		
Id	**Nombre**	**Descripción**
RF-01	Interfaz AC-01	El sistema contará con una interfaz de usuario propia y especializada para este tipo de usuario.
RF-02	Ingreso al sistema	El sistema desplegará una interfaz para inicio de sesión en la que se solicitará usuario y contraseña para ingresar.
RF-03	Registrar afiche	El sistema permitirá que este tipo de usuario registre un afiche publicitario en el servidor, llenando previamente el formulario correspondiente, en el cual se incluye seleccionar una imagen.
RF-04	Visualización de afiches	El sistema permitirá a este tipo de usuario visualizar los afiches publicitarios que ya se encuentren registrados.
RF-05	Desactivar y modificar afiche	El sistema permitirá a este tipo de usuario desactivar o modificar un afiche del sistema, para que no se muestre en el MUPI.
RF-06	Limpiar Servidor	El sistema permitirá borrar físicamente del servidor las imágenes de los afiches que estén cancelados, también permitirá que sean borrados de la base de datos.
Requerimientos Funcionales usuario AC-02		
RF-07	Interfaz AC-02	El sistema contará con una interfaz sencilla e intuitiva para este tipo de usuario.
RF-08	Selección de afiche en MUPI	El sistema permitirá a este tipo de usuario ver todos los afiches disponibles y seleccionar el que necesite ver.

Continuación Tabla IV.

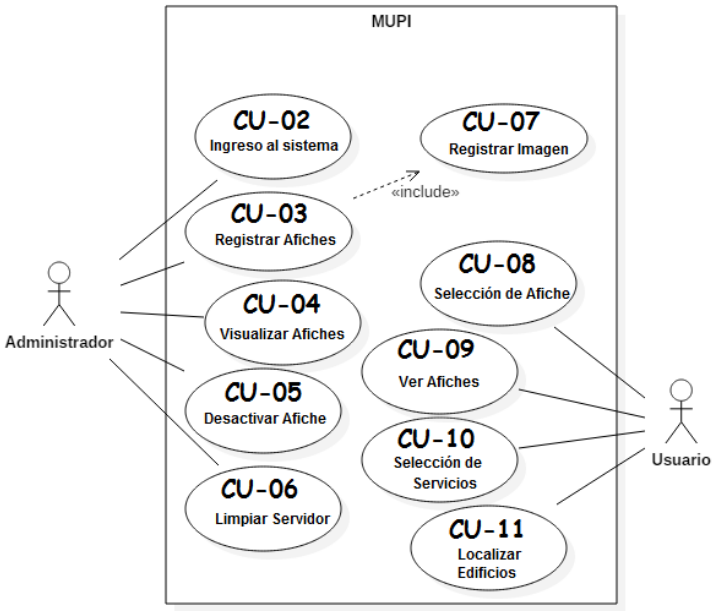
RF-09	Visualización de los afiches en MUPI	El sistema tendrá la opción, siendo este tipo de usuario, de mostrar los afiches en el MUPI de una manera cíclica, es decir, de forma rotativa donde una vez que llegue al fin, iniciará de nuevo automáticamente.
RF-10	Selección de servicios	El sistema permitirá a este tipo de usuario buscar y seleccionar algún servicio que se preste dentro de la universidad, e indicará la ubicación exacta del edificio donde se encuentra ese servicio.
RF-11	Ruta de edificios	El sistema, siendo este tipo de usuario, permitirá localizar cualquier edificio dentro de la universidad, a partir de donde el usuario se encuentre consultando el MUPI.

Fuente: elaboración propia.

2.5. Casos de uso

Con la finalidad de comprender mejor la interacción que tienen los usuarios con el sistema y visualizar las funcionalidades del mismo, se presentan los casos de uso, a continuación, el diagrama de casos de uso general:

Figura 1. **Diagrama de casos de uso**



Fuente: elaboración propia con software StarUML.

2.5.1. Caso de uso 1 – Ingreso al sistema

Tabla V. **CU-01 Ingreso al sistema**

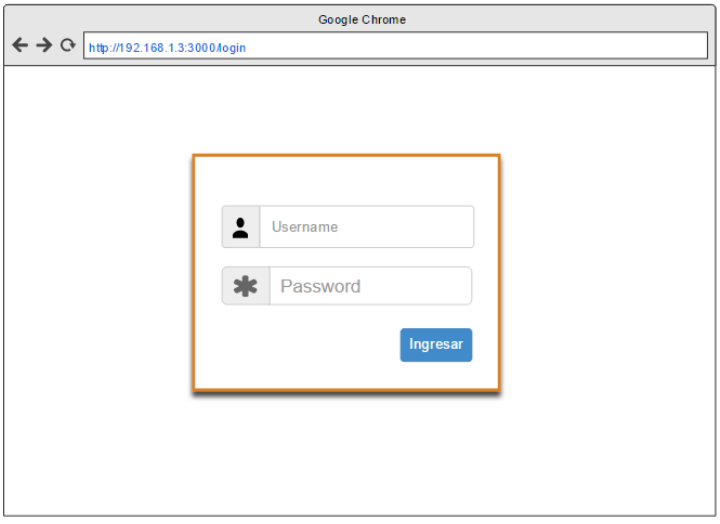
Caso de Uso	Ingreso al sistema	CU-01
Requerimientos relacionados	RF-02	
Precondiciones	El usuario AC-01 debe ser trabajador activo y tener un usuario y contraseña registrados en el sistema de usuarios de la universidad.	
Final exitoso	Ingreso al sistema	
Final fallido	Ingreso rechazado al sistema por usuario o contraseña incorrectas.	
Actores principales	AC-01	
Actores secundarios	Ninguno	
Evento de inicio	Usuario AC-01 solicita ingreso a la página de administración de los MUPIS.	

Continuación Tabla V.

	Eventos actor	Eventos sistema
Flujo principal	1. Ingresa a página web de inicio de sesión. 2. Ingresa usuario y contraseña y envía datos.	1. Muestra página web con formulario para ingresar al sistema. 2. Consulta sistema de usuarios provisto por la universidad y responde si el usuario y contraseña son correctos.
Información adicional	El formulario de ingreso al sistema contará con los siguientes campos: • Usuario: cadena de caracteres que identifica al usuario. • Contraseña: cadena de caracteres compuesta de letras y dígitos. El ingreso a esta parte del sistema dependerá de la disponibilidad del Servicio Web provisto por la universidad y externo a esta aplicación.	

A continuación, se presenta el prototipo de interfaz de usuario asociado a este caso de uso:

Figura 2. **RF-02 Ingreso al sistema**



Fuente: elaboración propia con software moqups 2.

2.5.2. Caso de uso 2 – Registrar afiche

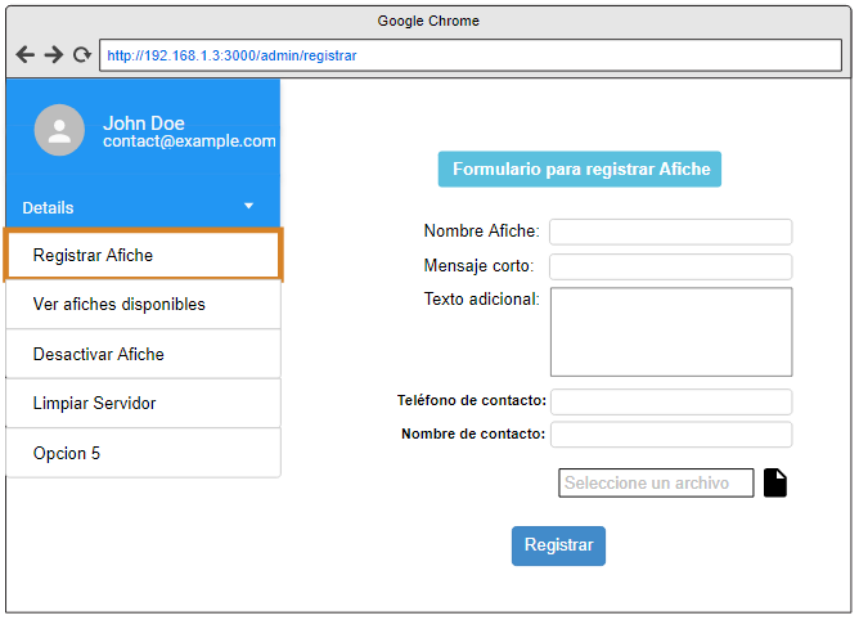
Tabla VI. **CU-02 Registrar afiche**

<table>
<tr><td>Caso de Uso</td><td colspan="2">Registrar afiche</td><td>CU-02</td></tr>
<tr><td>Requerimientos relacionados</td><td colspan="3">RF-03</td></tr>
<tr><td>Precondiciones</td><td colspan="3">Usuario AC-01 debe haber ingresado al sistema con usuario y contraseña correctas.</td></tr>
<tr><td>Final exitoso</td><td colspan="3">Afiche registrado con éxito en el sistema</td></tr>
<tr><td>Final fallido</td><td colspan="3">Afiche ingresado con datos erróneos o bien afiche rechazado por no llenar correctamente el formulario.</td></tr>
<tr><td>Actores principales</td><td colspan="3">AC-01</td></tr>
<tr><td>Actores secundarios</td><td colspan="3">Ninguno</td></tr>
<tr><td>Evento de inicio</td><td colspan="3">Usuario AC-01 selecciona la opción de registrar afiche en el menú principal de la página de administración.</td></tr>
<tr><td rowspan="2">Flujo principal</td><td>Eventos actor</td><td colspan="2">Eventos sistema</td></tr>
<tr><td>1. Selecciona opción de registrar afiche.
2. Ingresa datos al formulario y los envía.</td><td colspan="2">1. Muestra página web con formulario para el registro de afiches.
2. Registra datos de afiche en la base de datos y notifica al usuario del registro, ya sea exitoso o fallido.
3. Actualiza la información en todo el sistema.</td></tr>
<tr><td>Información adicional</td><td colspan="3">Campos para el formulario registro de afiche:
• Nombre afiche: combinación de letras para identificar el afiche.
• Mensaje corto: combinación de letras para poner un mensaje corto en el afiche.
• Texto adicional: combinación de letras para mostrar algún mensaje adicional en el afiche.
• Nombre contacto: combinación de letras que identifican a la persona responsable del afiche.
• Teléfono contacto: combinación de números que identifican el teléfono del responsable del afiche.
• Ruta afiche: combinación de letras que identifican la ruta física de la imagen del afiche.</td></tr>
</table>

Fuente: elaboración propia.

A continuación, se presenta el prototipo de interfaz de usuario asociado a este caso de uso:

Figura 3. **RF-03 y RF-07 Registro de afiche e imagen en el sistema**



Fuente: elaboración propia con software moqups 2.

2.5.3. Caso de uso 3 – Registrar imagen

Tabla VII. **CU-03 Registrar imagen**

Caso de Uso	Registrar imagen	CU-03
Requerimientos relacionados	RF-03	
Precondiciones	Usuario AC-01 debe haber ingresado al sistema con las credenciales correctas.	
Final exitoso	Imagen subida al servidor exitosamente	
Final fallido	Error al subir imagen Imagen ya existe en el servidor	
Actores principales	AC-01	
Actores secundarios	Ninguno	
Evento de inicio	Usuario AC-01 selecciona la opción para subir la imagen del afiche dentro del formulario para registrar afiche.	

Continuación Tabla VII.

Flujo principal	**Eventos actor**	**Eventos sistema**
	1. Selecciona botón para subir imagen. 2. Selecciona imagen a subir.	1. Muestra interfaz para que usuario busque y seleccione la imagen a subir. 2. Inicia proceso para subir imagen, notificándole al usuario si el proceso finaliza con éxito o no.
Información adicional	La extensión de los archivos de imagen es libre La imagen debe tener el ancho y largo correcto para visualizarse correctamente en el MUPI.	

Fuente: elaboración propia.

Para este caso de uso no se presenta un prototipo de interfaz de usuario, ya que es la misma presentada en CU-02.

2.5.4. Caso de uso 4 – Desactivar afiche

Tabla VIII. **CU-04 Desactivar afiche**

Caso de Uso	Desactivar afiche	CU-04
Requerimientos relacionados	RF-05	
Precondiciones	El afiche debe estar registrado y activo para poderse desactivar.	
Final exitoso	Afiche desactivado y MUPI actualizado	
Final fallido	Error al desactivar afiche o al actualizar MUPI	
Actores principales	AC-01	
Actores secundarios	Ninguno	
Evento de inicio	Usuario AC-01 selecciona la opción para desactivar afiche dentro del menú principal.	
Flujo principal	**Eventos actor**	**Eventos sistema**
	1. Selecciona opción para desactivar afiche. 2. Selecciona afiche dentro de la lista y presiona botón desactivar.	1. Muestra formulario para desactivar afiche. 2. Inicia proceso para desactivar y actualiza la base de datos. Notifica al usuario si el proceso finalizo con éxito o no.

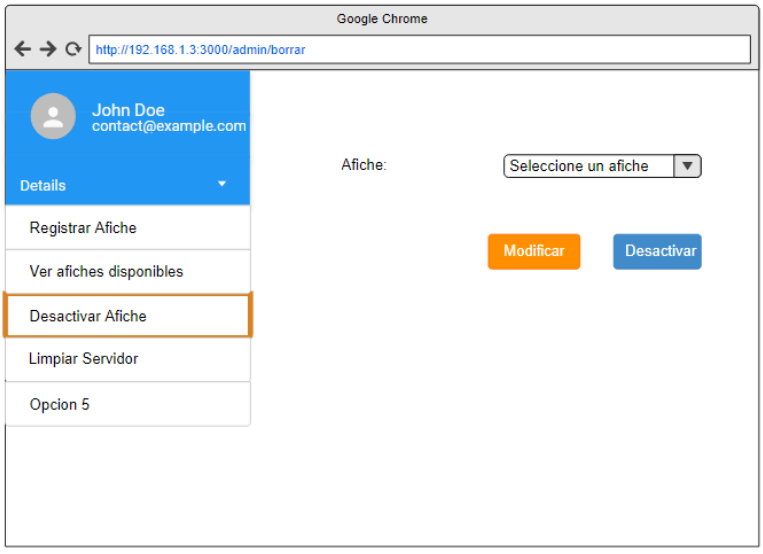
Continuación Tabla VIII.

		3. Actualiza la información en todo el sistema.
Información adicional	El formulario para desactivar un afiche contará solo con el siguiente campo: • Nombre afiche: una lista desplegable que contiene todos los nombres de afiche disponibles.	

Fuente: elaboración propia.

A continuación, se presenta el prototipo de interfaz de usuario asociado a este caso de uso:

Figura 4. **RF-05 Desactivar y modificar afiche**



Fuente: elaboración propia con software moqups 2.

2.5.5. Caso de uso 5 – Listar afiches

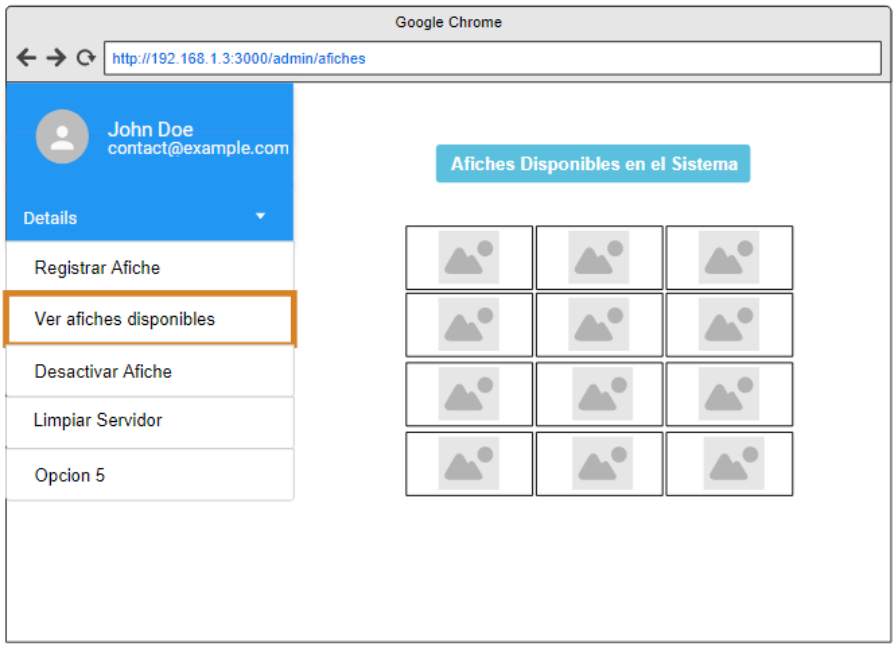
Tabla IX. **CU-05 Listar afiches**

Caso de Uso	Listar afiches	CU-05
Requerimientos relacionados	RF-04	
Precondiciones	Los afiches deben estar en estado disponible para poder verlos en esta lista.	
Final exitoso	Visualización de todos los afiches disponibles	
Final fallido	Visualización incorrecta o incompleta de la lista de los afiches disponibles.	
Actores principales	AC-01	
Actores secundarios	Ninguno	
Evento de inicio	Usuario AC-01 selecciona la opción de ver afiches en el menú principal.	
Flujo principal	**Eventos actor**	**Eventos sistema**
	1. Seleccionar opción para ver todos los afiches.	1. Muestra todos los afiches disponibles registrados en el sistema.
Información adicional	Ninguna	

Fuente: elaboración propia.

A continuación, se presenta el prototipo de interfaz de usuario asociado a este caso de uso:

Figura 5. **RF-04 Visualización de afiches**



Fuente: elaboración propia con software moqups 2.

2.5.6. Caso de uso 6 – Limpiar Servidor

Tabla X. **CU-06 Limpiar Servidor**

Caso de Uso	Limpiar Servidor	CU-06
Requerimientos relacionados	RF-06	
Precondiciones	Deben existir afiches cancelados previamente. La ruta de acceso física a la imagen en el servidor debe ser la que el sistema asigna por defecto.	
Final exitoso	Eliminación de los afiches previamente cancelados, tanto en el servidor como en base de datos.	
Final fallido	Error al buscar la ruta de acceso física de la imagen	
Actores principales	AC-01	
Actores secundarios	Ninguno	
Evento de inicio	Usuario AC-01 selecciona la opción de limpiar servidor en el menú principal.	

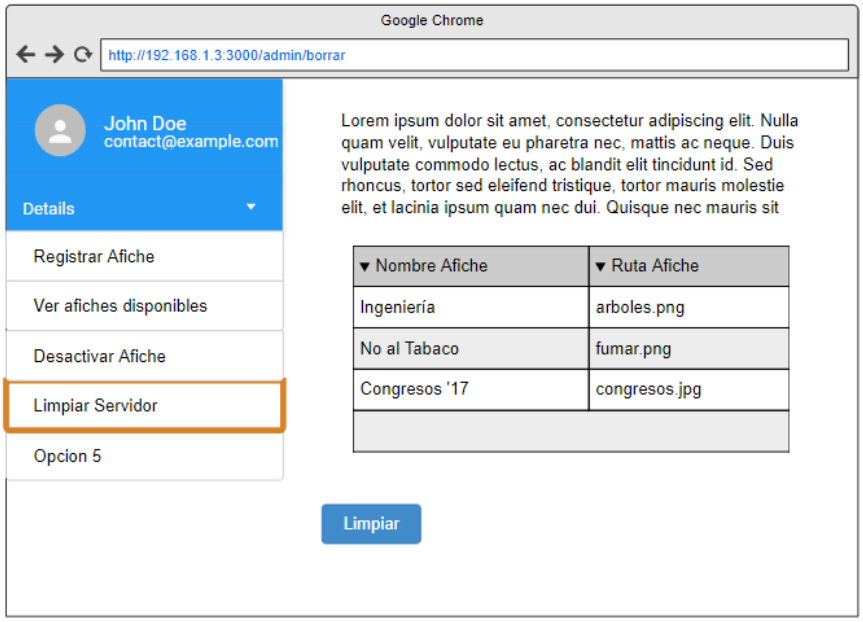
Continuación Tabla X.

	Eventos actor	Eventos sistema
Flujo principal	1. Selecciona opción de limpiar servidor en el menú. 2. Selecciona botón limpiar.	1. Verifica en base datos que afiches están en estado cancelado y los muestra al usuario. 2. Elimina del servidor y de base de datos los afiches y notifica al usuario si el resultado es exitoso o erróneo.
Información adicional	Esta opción elimina definitivamente la información de los servidores, por lo tanto, no es posible recuperarla.	

Fuente: elaboración propia.

A continuación, se presenta el prototipo de interfaz de usuario asociado a este caso de uso:

Figura 6. **RF-06 Limpiar Servidor**



Fuente: elaboración propia con software moqups 2.

2.5.7. Caso de uso 8 – Selección de afiche

Tabla XI. **CU-06 Selección de afiche**

Caso de Uso	Selección de afiche	CU-08
Requerimientos relacionados	RF-07	
Precondiciones	Los afiches deben estar registrados previamente en el sistema para poderlos observar.	
Final exitoso	Visualización del afiche seleccionado por el usuario	
Final fallido	Error al seleccionar o mostrar el afiche seleccionado	
Actores principales	AC-02	
Actores secundarios	Ninguno	
Evento de inicio	Usuario AC-02 selecciona la opción de ver afiches en el MUPI.	
Flujo principal	**Eventos actor**	**Eventos sistema**
	1. Selecciona opción de ver afiches en el MUPI. 2. Selecciona de la lista mostrada, un afiche.	1. Abre una ventana emergente, mostrando en una lista todos los afiches disponibles. 2. Muestra la imagen del afiche en la misma ventana emergente.
Información adicional	Ninguna	

Fuente: elaboración propia.

A continuación, se presenta el prototipo de interfaz de usuario asociado a este caso de uso:

Figura 7. **RF-07 Selección de afiche**

Fuente: elaboración propia con software moqups 2.

2.5.8. Caso de uso 9 – Ver afiches

Tabla XII. **CU-09 Ver afiches**

Caso de Uso	Ver afiches	CU-09
Requerimientos relacionados	RF-08	
Precondiciones	Los afiches deben estar en estado disponible para poder verlos en el MUPI.	
Final exitoso	Visualización de todos los afiches de una forma cíclica	
Final fallido	Visualización incorrecta o incompleta de los afiches	
Actores principales	AC-02	
Actores secundarios	Ninguno	
Evento de inicio	Ninguno, este será el comportamiento por defecto del MUPI.	
Flujo principal	**Eventos actor**	**Eventos sistema**
	Ninguno	1. Cada 5 segundos cambiará de afiche, hasta mostrar todos los de la lista, y al terminar, iniciará de nuevo el ciclo.
Información adicional	Ninguna	

Fuente: elaboración propia.

A continuación, se presenta el prototipo de interfaz de usuario asociado a este caso de uso:

Figura 8. **RF-08 Visualización de los afiches en el MUPI**

Fuente: elaboración propia con software moqups 2.

2.5.9. Caso de uso 10 – Selección de servicios

Tabla XIII. **CU-10 Selección de servicios**

Caso de Uso	Ver servicios	CU-10
Requerimientos relacionados	RF-10	
Precondiciones	Los servicios deben estar previamente registrados en la base de datos.	
Final exitoso	Visualización tanto de los servicios como el edificio donde se encuentran.	
Final fallido	Visualización incorrecta o incompleta de los servicios	
Actores principales	AC-02	
Actores secundarios	Ninguno	
Evento de inicio	Usuario AC-02 selecciona opción de ver servicios en el menú principal del MUPI.	

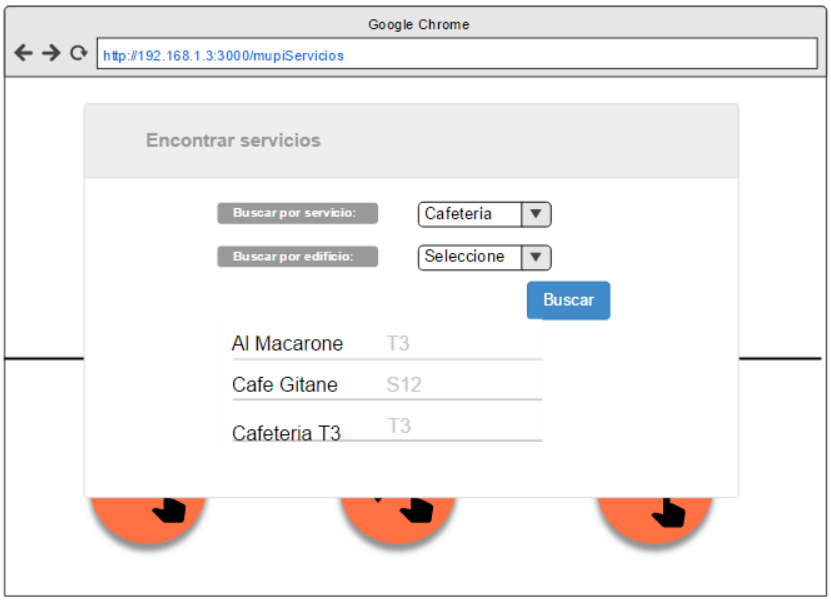
Continuación Tabla XIII.

	Eventos actor	Eventos sistema
Flujo principal	1. Selecciona opción de ver servicios en el menú principal del MUPI. 2. Selecciona un edificio o un servicio o también puede seleccionar una combinación de edifico y servicio para buscar alguna coincidencia.	1. En una ventana emergente muestra la lista de edificios y servicios disponibles. 2. Muestra en una tabla las coincidencias que encuentra con los parámetros de búsqueda ingresados por el usuario.
Información adicional	Ninguna	

Fuente: elaboración propia.

A continuación, se presenta el prototipo de interfaz de usuario asociado a este caso de uso:

Figura 9. **RF-10 Selección de servicios**



Fuente: elaboración propia con software moqups 2.

2.5.10. Caso de uso 11 – Localizar edificios

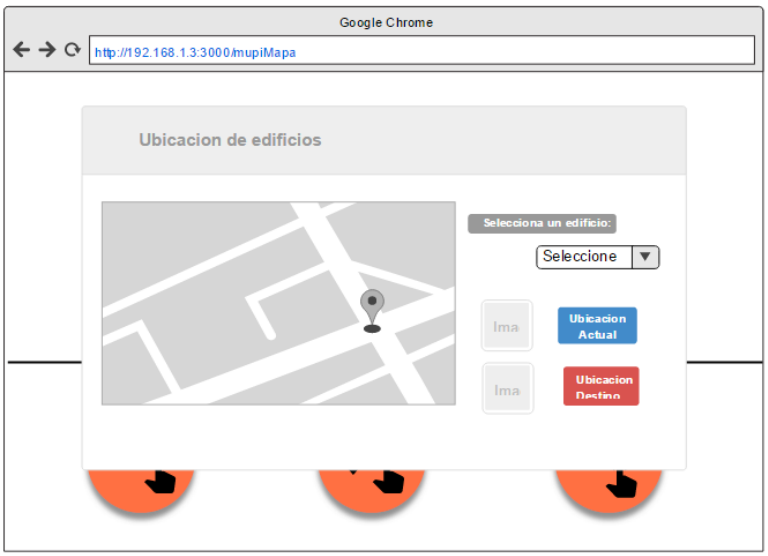
Tabla XIV. **CU-11 Localizar edificios**

Caso de Uso	Localizar edificios	CU-11
Requerimientos relacionados	RF-11	
Precondiciones	Los edificios de la universidad deben estar previamente registrados en la base de datos.	
Final exitoso	Visualización del edificio seleccionado en el mapa	
Final fallido	Visualización incorrecta o fallida del edificio en el mapa	
Actores principales	AC-02	
Actores secundarios	Ninguno	
Evento de inicio	Usuario AC-02 selecciona la opción de localizar edificio en el menú principal del MUPI.	
Flujo principal	**Eventos actor**	**Eventos sistema**
	1. Selecciona opción de localizar edificio en el menú principal. 2. Selecciona edificio a ser localizado.	1. Muestra en una ventana emergente, la lista de los edificios disponibles y un mapa donde se encuentra el usuario consultado actualmente el MUPI. 2. Muestra en el mapa la ubicación del edificio seleccionado por el usuario.
Información adicional	Ninguna	

Fuente: elaboración propia.

A continuación, se presenta el prototipo de interfaz de usuario asociado a este caso de uso:

Figura 10. **RF-11 Ruta de edificios**



Fuente: elaboración propia con software moqups 2.

2.6. Requerimientos no funcionales

A continuación, se especifican los requerimientos no funcionales identificados para el sistema, presentados y acordados siempre con la entidad que administrará los MUPIS en la universidad.

Tabla XV. **Requerimientos no funcionales**

Id	Aspecto	Descripción
RNF-01	Usabilidad	El sistema deberá ser intuitivo y fácil de utilizar, con un tiempo de aprendizaje menor a 5 minutos.
RNF-02		El sistema contará con mensajes de éxito y error en las operaciones que se realicen, tanto para la interfaz de usuario AC-01 como para usuario AC-02.
RNF-03		Las interfaces para los usuarios serán adaptativas, es decir, se adaptarán a cualquier pantalla, con lo que se garantiza el uso de cualquier pantalla en la implementación del sistema.

Continuación Tabla XV.

RNF-04	Disponibilidad	El tiempo de reinicio del sistema será menos a 10 minutos.
RNF-05		El sistema contará con la función de acceso remoto, con lo que se garantiza la ayuda en cualquier momento.
RNF-06		El MUPI podrá ser implementado en cualquier sistema operativo, ya que el sistema estará dentro de un servidor Web.
RNF-07	Eficiencia	El sistema será capaz de almacenar imágenes en el servidor, sin retrasos pausas en las demás funcionalidades.
RNF-08		El sistema tendrá la capacidad de aceptar una concurrencia de más de 50 MUPIs, asegurando el despliegue de imágenes y mapas en tiempo real.

Fuente: elaboración propia.

2.7. Arquitectura

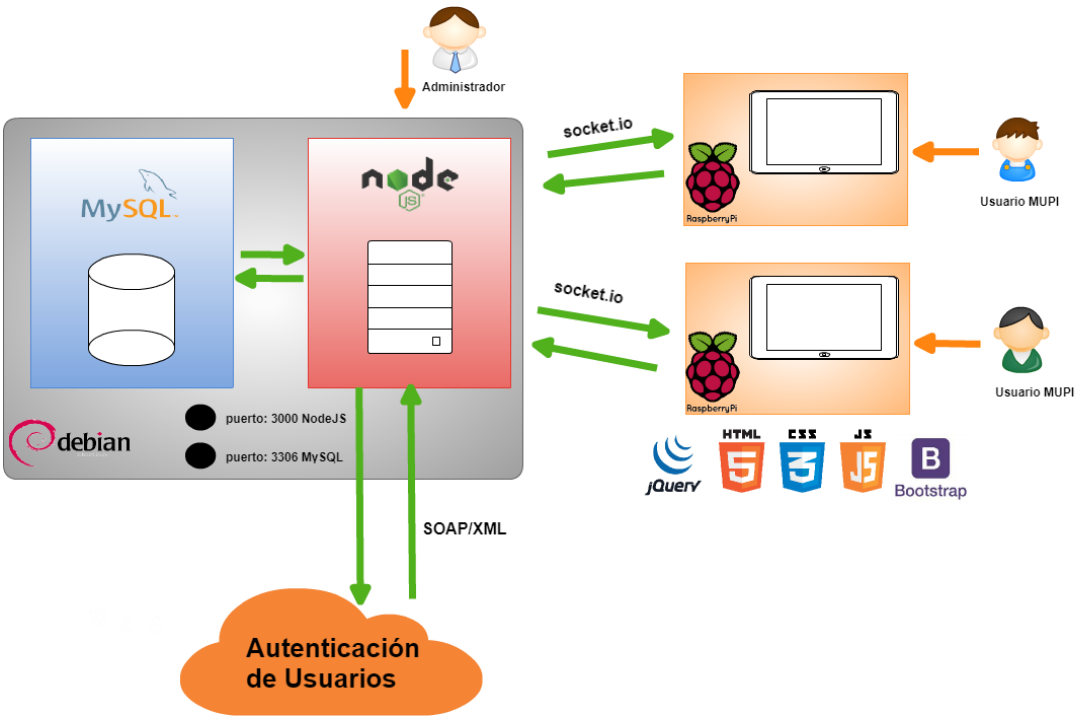
Para facilitar la comprensión y análisis del sistema a desarrollar, a continuación, se presenta de una forma gráfica, con ayuda del lenguaje UML (*Unfied Modeling Languaje*), y descriptiva la arquitectura del software, la cual está basada en el patrón de arquitectura MVC (Modelo, Vista, Controlador), el cual satisface de una forma clara y ordenada los requerimientos del sistema.

Para la construcción del sistema se utilizarán distintas tecnologías que se explicarán más adelante, sin embargo, es importante destacar el uso de NodeJS como núcleo del sistema y también el uso del Framework *Express* que permite la implementación rápida y flexible de la infraestructura web y también el uso de la herramienta *Express-Generator* la cual proporciona la implementación rápida e intuitiva del patrón de diseño MVC.

2.7.1. Definición de la arquitectura

A continuación, se presenta una visión general del funcionamiento a nivel de arquitectura del sistema:

Figura 11. **Diagrama de arquitectura del sistema**



Fuente: elaboración propia con software Cacoo.

Diagrama en el que se puede observar las 3 capas del patrón de arquitectura MVC, las cuales se identifican de la siguiente manera:

- Modelo (azul): capa para manejar los datos del sistema
- Vista (anaranjado): capa que presenta los datos e interactúa con el usuario.
- Controlador (rojo): capa intermediaria entre la vista y los datos, responsable del manejo y control de la aplicación.

2.7.2. Plataforma de software base

Para la capa de vista del modelo MVC serán implementadas las siguientes tecnologías:

Tabla XVI. **Software utilizado en el cliente**

Aspecto	Tecnología
Sistema operativo	El cliente será implementado en una Raspberry Pi, por lo tanto, se estará utilizando un sistema operativo basado en GNU/Linux.
Navegador Web	Firefox o Google Chrome
Lenguaje de programación	Javascript
Tecnologías WEB	CCS3 HTML5 JQuery

Fuente: elaboración propia.

Para la capa de modelo serán implementadas las siguientes tecnologías:

Tabla XVII. **Software utilizado en la base de datos**

Aspecto	Tecnología
Sistema operativo	Debian GNU/Linux
Base de datos	MySQL

Fuente: elaboración propia.

Continuando con el modelo MVC, para la capa de control serán estas tecnologías las que se implementarán:

Tabla XVIII. **Software utilizado en el servidor**

Aspecto	Tecnología
Sistema operativo	Debian GNU/Linux
Servidor Web	NodeJS
Lenguaje de programación	Javascript
Frameworks	Express

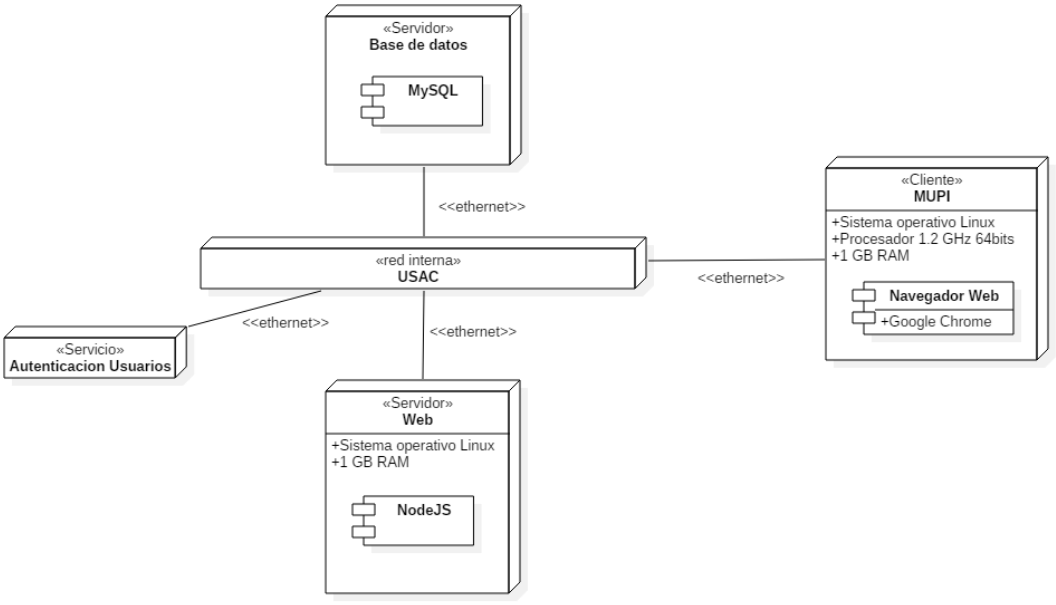
Fuente: elaboración propia.

Es importante mencionar que en la sección 2.1 se evalúa las tecnologías mencionadas con respecto a sus competidores, dando a conocer las razones del porque son las seleccionadas para satisfacer las necesidades del sistema.

2.7.3. Diagrama de despliegue

Con el objetivo de comprender mejor la parte física del sistema se presenta el diagrama de despliegue, el cual muestra las conexiones físicas que existirán entre cada nodo. Un nodo es un elemento de hardware que tiene memoria y capacidad de procesamiento, en otras palabras, para nuestro sistema, es una computadora.

Figura 12. **Diagrama de despliegue**

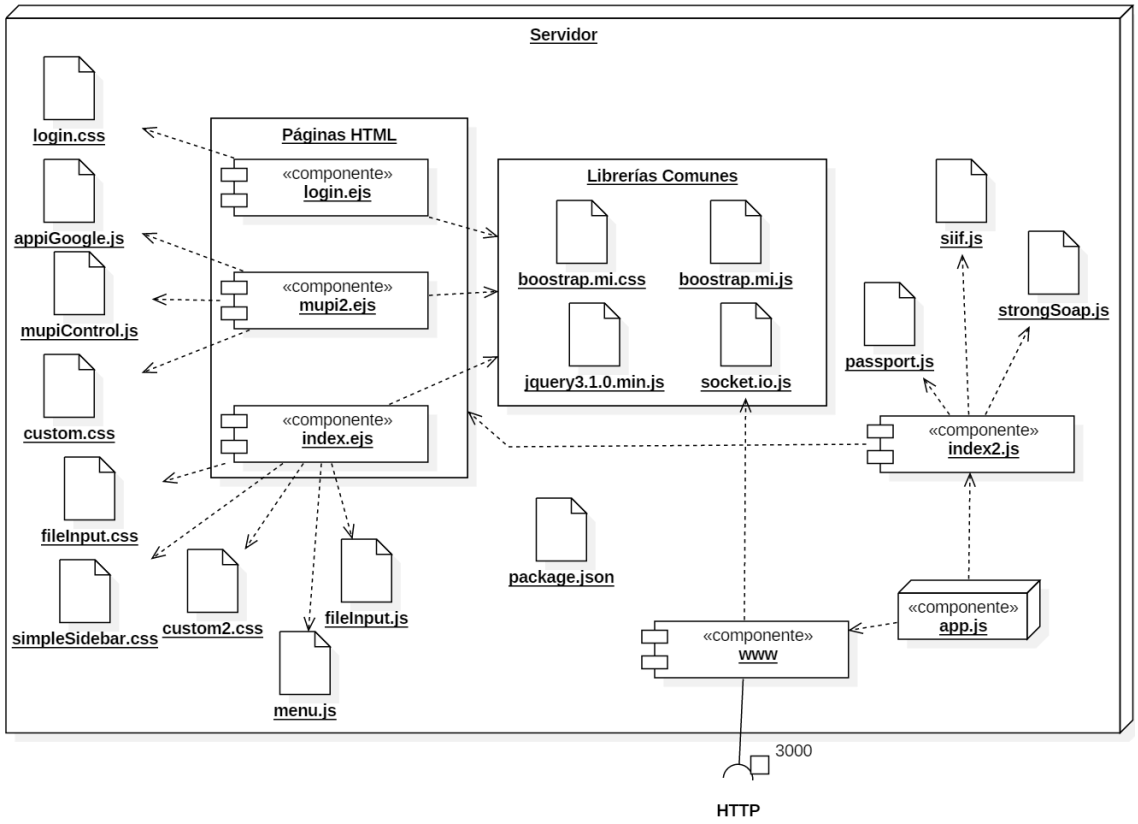


Fuente: elaboración propia con software StarUML.

2.7.4. Diagrama de componentes

Los componentes de un sistema pueden definirse como los elementos que proporcionan alguna funcionalidad al software, pueden ser archivos, ejecutables, librerías e incluso módulos. El objetivo de este diagrama es proporcionar una visualización de los componentes que conforman el sistema y la dependencia que existe entre ellos.

Figura 13. **Diagrama de componentes**

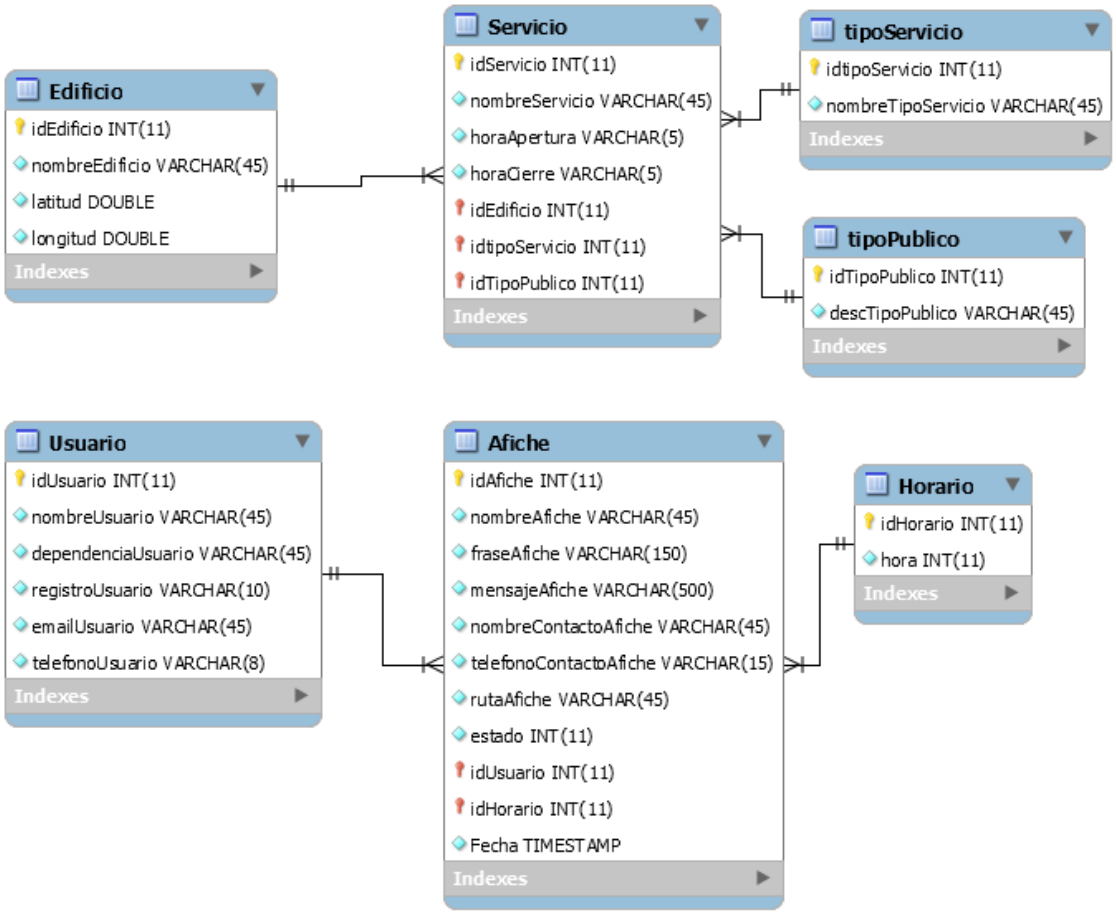


Fuente: elaboración propia con software StarUML.

2.7.5. Diagrama de base de datos

Entender la relación que guardan las estructuras en una base de datos permite entender de una mejor manera el funcionamiento general del sistema, además da una mejor perspectiva de los requisitos o limitaciones con las que cuenta en general el sistema.

Figura 14. **Diagrama entidad relación**



Fuente: elaboración propia con software MySQL Workbench.

3. DOCUMENTACION TÉCNICA

En este capítulo se describe el funcionamiento de cada uno de los recursos que componen el sistema, su importancia, contenido, ubicación y relación con otros recursos. Conocer a detalle los aspectos mencionados anteriormente conducen a un mejor entendimiento del sistema a nivel técnico y en consecuencia permite dimensionar y evaluar cambios que requiera el sistema en un futuro, corregir errores de desarrollo, descartar un posible error de usuario o simplemente darle mantenimiento a la aplicación.

3.1. Dependencias

En el desarrollo de cualquier sistema de software siempre es necesaria la implementación de librerías escritas y desarrolladas por otras personas. Estas librerías externas permiten interactuar con otras tecnologías y proveen de mayores funcionalidades al sistema en desarrollo.

A continuación, se presenta una tabla con las dependencias instaladas de forma global, es decir, librerías que pueden ser invocadas desde cualquier ubicación del sistema operativo.

Tabla XIX. **Dependencias globales**

Nombre	Versión	Utilización	Descripción
express-generator	4.15.0	Servidor	Generador de plantillas
	Permite crear rápidamente las carpetas necesarias para crear un proyecto con el paradigma MVC (Modelo, Vista, Controlador). Referencia: http://expressjs.com/es/starter/generator.html		
node-gyp	3.6.2	Servidor	Compilador para NodeJS
	Necesario para compilar la librería strong-soap que se describe en la tabla de dependencias locales. Referencia: https://github.com/nodejs/node-gyp		
npm	2.15.11	Servidor	Gestor de paquetes
	Esta librería se instala automáticamente en la instalación de NodeJS y es útil ya que permite administrar cualquier librería ya sea global o local. Referencia: https://www.npmjs.com/		
pm2	2.6.1	Servidor	Administrador de procesos
	Instalada para la administración del proceso que permite ejecutar el servidor indefinidamente, además permite rastrear errores y verificar en general el consumo de recursos de la aplicación. Referencia: http://pm2.keymetrics.io/		

Fuente: elaboración propia.

En la siguiente tabla se presentan las librerías que son instaladas y necesarias por cada aplicación que se cree, es decir, estas librerías son instaladas internamente en cada aplicación que se cree en el lenguaje NodeJS.

Tabla XX. **Dependencias locales**

Nombre	Versión	Utilización	Descripción
bluebird	3.5.0	Servidor	Creador de métodos síncronos
	Permite crear funciones o métodos síncronos, los cuales fueron utilizados en la consulta al Web Service y obtención de datos desde MySQL. Referencia: http://bluebirdjs.com/docs/getting-started.html		
connect-flash	0.1.0	Servidor	Mensajes Flash
	Útil para enviar mensajes por medio de la petición HTTP. Referencia: https://www.npmjs.com/package/connect-flash		
cookie-parser	1.4.3	Servidor	Administrador de procesos
	Instalado por defecto al utilizar express-generator sin embargo no se hace uso de él. Referencia: https://www.npmjs.com/package/cookie-parser		
ejs	2.5.2	Cliente	Administrador de plantillas
	Permite hacer uso de variables enviadas en la respuesta del servidor en el DOM del HTML Referencia: https://www.npmjs.com/package/ejs		
express	4.13.4	Servidor	Infraestructura Web
	Facilita toda la infraestructura web y es instalado por defecto al utilizar express-generator. Referencia:http://expressjs.com/es/		
express-session	1.15.5	Servidor	Administrador de sesiones
	Permite crear sesiones de usuario y mantener la información del lado del servidor. Referencia: https://www.npmjs.com/package/express-session		
morgan	1.7.0	Servidor	Registrador
	Muestra información sobre las peticiones que recibe el servidor, es instalado por defecto al utilizar express-generator. Referencia: https://github.com/expressjs/morgan		
Mysql	2.12.0	Servidor	Conexión a MySQL
	Facilita la interfaz de conexión hacia la base datos.		
passport	0.4.0	Servidor	Inicio de sesión
	Provee la interfaz necesaria para crear rápidamente un sistema de inicio de sesión, esta librería hace uso de express-session y passport-local. Referencia: http://passportjs.org/		
passport-local	1.0.0	Servidor	Inicio de sesión
	Necesaria para poder crear un sistema de inicio de sesión donde la consulta la realice a una base de datos y no a Facebook o Gmail. Referencia: http://passportjs.org/		
serve-favicon	2.3.0	Cliente	Ícono de página
	Permite configurar el ícono que se mostrará en el navegador web cuando se ingrese al sitio del sistema.		

Continuación Tabla XX.

	Referencia: https://www.npmjs.com/package/serve-favicon		
socket.io	1.7.1	1.7.1	Web Sockets
	Útil para realizar aplicaciones en tiempo real, ya que la transmisión de datos se realiza por medio de un socket. Referencia: https://socket.io/		
strong-soap	1.2.6	Servidor	Conexión con WebService
	Permite la conexión hacia un Web Service con la tecnología SOAP, en este caso, útil para validar el usuario y contraseña en el inicio de sesión. Referencia: https://github.com/strongloop/strong-soap		

Fuente: elaboración propia.

3.2. Directorio del sistema

Este desarrollo está basado en una serie de archivos JavaScript, CSS, imágenes y archivos JSON, conocer la ubicación de los archivos dentro del sistema es fundamental para poder corregir cualquier tipo de error o bien agregar en un futuro algún modulo que aporte más funcionalidades a la aplicación.

A continuación, se describe la ubicación de cada archivo y de una forma muy general su contenido, es importante mencionar que existen una serie de archivos que se crean al momento de instalar las librerías, en los cuales no se profundizará.

- mupi_V5
  - bin
    - **www**: este archivo es creado por defecto por la librería express-generator, no cuenta con una extensión, sin embargo, al abrirlo se observa el código necesario para

crear el servidor. En la sección 3.3 se detalla el uso de este archivo.

- node_modules: en esta carpeta se encuentran todas librerías instaladas localmente.
- public
  - css
    - ✓ **custom.css:** hoja de estilo para la página principal
    - ✓ **custom2.css:** hoja de estilo para la página del administrador.
    - ✓ **login.css:** hoja de estilo para la página de inicio de sesión.
    - ✓ **fileinput.min:** hoja de estilo para el plugin externo que se utiliza para subir imágenes al servidor.
    - ✓ **boostrap.min.css:** pertenece al framework boostrap
  - fonts: carpeta para almacenar tipos de letras utilizables en las hojas de estilo.
  - images: en esta carpeta se encuentran las imágenes utilizadas en las hojas de estilo o en las páginas HTML que pertenecen al sistema y no cambian en el transcurso de la ejecución.
  - img: esta carpeta almacena las imágenes utilizadas por el plugin para subir imágenes al servidor.
  - iSubidass: esta carpeta contiene las imágenes que se suben y registran en el sitio para el administrador, como parte del afiche.
  - js
    - ✓ **boostrap.mi.js:** pertenece al framework boostrap

- ✓ **fileinput.min.js:** archivo javascript que contiene la lógica del plugin utilizado para subir imágenes al servidor.
- ✓ **jquery-3.1.0.min.js:** pertenece al framework Jquery
- ✓ **menu.js:** contiene la lógica y acciones de la página del sitio del administrador.
- ✓ **mupiControl.js:** contiene la lógica y acciones de la página principal del Mupi.

- service: en esta carpeta se encuentran las imágenes de los servicios que se muestran en la página principal del Mupi.

- routes
  - **index2.js:** este archivo es el módulo con el cual se manejan todas las rutas o URI's del sistema.
  - **siif.js:** este archivo contiene el módulo para conectarse al Web Service de la Universidad y consultarlo.
- views
  - **error.ejs:** página que se muestra al momento de encontrarse un error en el sistema.
  - **index.ejs:** página que se muestra al ingresar al sitio del administrador siempre que el usuario este registrado y haya iniciado sesión.
  - **login.ejs:** página que se muestra al ingresar al sitio del administrador.
  - **mupi2.ejs:** página principal del sistema en donde se muestran los afiches.
- **app.js:** archivo que contiene toda la configuración del servidor
- **package.json:** archivo que contiene un resumen de todas las dependencias utilizadas en el sistema.

3.3. Archivos indispensables

Después de conocer las librerías de las que depende el sistema y analizar las carpetas y archivos de los cuales se compone, es importante conocer más profundamente el contenido de cada archivo.

El objetivo de esta sección es poder dar un resumen y explicación de las funciones o procedimientos que se encuentran en los siguientes archivos:

3.3.1. www

Archivo que contiene la creación del servidor, es decir, este archivo es el que se ejecuta inicialmente y da origen al sistema, se encuentra en la ruta /bin/www y cuenta con las siguientes funciones:

- io.on('connection', function): esta función recibe otra función como parámetro la cual se ejecuta al iniciar el servidor, contiene la lógica del Web Socket él cual permite enviar y recibir mensajes del cliente en tiempo real.
- limpiarServidor(afiches): esta función tiene como parámetro un arreglo de objetos 'Afiche', es utilizada para eliminar definitivamente las imágenes del servidor y es invocada desde el Web Socket.
- normalizePort(val): esta función recibe como parámetro el puerto y tiene como objetivo preparar el servidor para poder escuchar en el puerto que se le indique por medio del parámetro.
- onError(error): este método le indica al servidor que acción realizar en caso de un evento de error.
- onListening(): este método le indica al servidor que acción realizar en caso de un evento entrante.

Es importante indicar que en este archivo se invoca el archivo app.js, ubicado en la raíz del directorio, el cual contiene otras configuraciones del servidor y es indispensable invocar este archivo.

3.3.2. index2.js

Este archivo cuenta con todas las acciones a realizar de parte del servidor cuando una URI es invocada de parte del cliente, también cuenta con las funciones para realizar la conexión y todas las acciones sobre la base de datos y por último cuenta con toda la lógica de inicio de sesión.

A continuación, se detallan las funciones mencionadas anteriormente:

- mysql.createConnection(): esta función recibe un objeto JSON en donde se especifican todos los datos para realizar la conexión hacia MySQL. Retorna un objeto con el cual se realizará la conexión.
- connection.connect(function): este método realiza la conexión a la base de datos y la mantiene activa.
- router.get('/login',function): utilizada para enviarle al cliente la página de inicio sesión, siempre y cuando ingrese a la ruta mupies.usac.edu.gt/login del sitio.
- router.get('/',function): utilizada para enviarle al cliente la página principal del mupi, siempre y cuando ingrese a la ruta mupies.usac.edu.gt del sitio.
- router.get('/index',confirmarAutenticacion,function): utilizada para enviarle al cliente la página principal del administrador. Esta ruta está protegida y puede ser accedida solo si el usuario esta autenticado en el sistema.

- router.get(‘/logout’,confirmarAutenticacion,function): utilizada para finalizar sesión y redirigir al cliente a la página de inicio de sesión. Esta ruta está protegida y puede ser accedida solo si el usuario esta autenticado en el sistema.
- router.post(‘/admin’,passport.authenticate,function): utilizada para redirigir al cliente a la página principal del administrador siempre y cuando realice un inicio de sesión exitoso o bien, redirigir al cliente a la página de inicio de sesión si ingresa mal su usuario y contraseña. Esta función hace uso de la librería passport.js
- router.post(‘/cancelarA’, confirmarAutenticacion,function): esta función cambia el estado del afiche, escribiendo en la base de datos un 0 en el campo ‘estado’ del afiche que el cliente seleccione. Esta ruta está protegida y puede ser accedida solo si el usuario esta autenticado en el sistema.
- router.post(‘/modificarA’,function): esta función es la encargada de realizar una modificación de un afiche en la base de datos. Esta ruta está protegida y puede ser accedida solo si el usuario esta autenticado en el sistema.
- router.post(‘/buscarCancelados’,function): esta función tiene como objetivo encontrar en la base de datos los afiches que se encuentren en estado 0, es decir, que se encuentren cancelados y retornar toda su información. Esta ruta está protegida y puede ser accedida solo si el usuario esta autenticado en el sistema.
- router.post(‘/afichesDisponibles’,function): esta función es la encargada de buscar en la base de datos solo los afiches disponibles y retornar toda su información.
- router.post(‘/afichesBD’,function): esta función es utilizada para obtener la información de todos los afiches disponibles en la base de datos sin importar si están cancelados o activos.

- router.post(‘/afichesBD2’,function): función utilizada para obtener toda la información de un afiche en específico, es utilizada desde el sitio del mupi.
- router.post(‘/nombreAfiche’,function): esta función es utilizada para validar el nombre del afiche en la base de datos, ya que si no retorna nada la consulta, significa que no existe en base de datos el nombre del afiche ingresado. Esta ruta está protegida y puede ser accedida solo si el usuario esta autenticado en el sistema.
- router.post(‘/obEdificios’,function): esta función es utilizada para llenar las listas desplegables que tienen los números de edificios en el sitio del mupi, obtiene los resultados directamente de la base de datos.
- router.post(‘/obEdificio’,function): esta función es utilizada para obtener las coordenadas de ubicación de un edificio solicitado directamente de la base de datos. Es utilizada desde el sitio del mupi.
- router.post(‘/obTipoServicio’,function): esta función es utilizada para llenar las listas desplegables del sitio del mupi y lo que obtiene de la base de datos son los tipos de servicio que existen.
- router.post(‘/busquedaxedificio’,function): función utilizada para realizar la búsqueda de servicios filtrado por edificio en la base de datos y es llamada desde el sitio del mupi.
- router.post(‘/busquedaxservicio’,function): función utilizada para realizar la búsqueda por servicio en la base de datos y es llamada desde el sitio del mupi.
- router.post(‘/busquedaxediservi’,function): función utilizada para buscar un servicio filtrado por número de edificio en la base de datos y es llamada desde el sitio del mupi.
- router.post(‘/busquedaServicioIndividual’,function): esta función es llamada desde el sitio del mupi y ejecuta una consulta en la base de datos, para obtener todos los datos de un servicio en específico.

- router.post(‘/carga3’, confirmarAutenticacion,function): esta función es utilizada para subir las imágenes físicamente al servidor y es llamada desde el sitio del administrador. Esta ruta está protegida y puede ser accedida solo si el usuario esta autenticado en el sistema.
- router.post(‘limpiar’, confirmarAutenticacion,function): esta función es llamada desde el sitio del administrador y es utilizada para eliminar las imágenes físicamente del servidor ejecutando órdenes propias del sistema operativo. Esta ruta está protegida y puede ser accedida solo si el usuario esta autenticado en el sistema.
- router.post(‘carga2’,confirmarAutenticacion,function): esta función es utilizada para la inserción de los afiches en la base de datos, en el sitio del administrador. Esta ruta está protegida y puede ser accedida solo si el usuario esta autenticado en el sistema.
- passport.use(new LocalStrategy): la implementación de esta función es requerida por la librería ‘passport’ y define las reglas a utilizar para validar el inicio de sesión de un usuario. En esta función se hace la llamada al Web Service de la universidad para validar código y contraseña del usuario.
- passport.serializeUser(function): esta función es propia de la librería ‘passport’ y es utilizada para guardar los datos del usuario en el servidor y ser utilizados por la misma librería.
- passport.deserializeUser(function): esta función es propia de la librería ‘passport’ y es utilizada para obtener los datos de un usuario cada vez que la librería lo requiera.
- confirmarAutenticacion(req,res,next): esta función verifica que exista un usuario actual con sesión iniciada, si no existe entonces redirige al usuario a la página de inicio de sesión. Esta función es utilizada para proteger todas las rutas que requieran de un inicio de sesión.

- verifcarDueno(idAficheO,usuarioA): esta función es utilizada para restringir la desactivación y modificación de un afiche por parte de alguien que no sea el dueño, recibe como parámetros el número del afiche a modificar/cancelar y el número que identifica al usuario en la sesión actual. Otra característica importante de esta función es que utiliza Promesas las cuales permiten volver el código asíncrono en síncrono. Si la función encuentra que el usuario es el dueño del afiche retorna un 1, entonces permite realizar la acción requerida, si no devuelve un 0 indicando que no es el dueño.

3.3.3. siif.js

Este archivo ubicado en /routes cuenta con la interfaz de comunicación con el Web Service de la universidad, utilizado para validar el código y contraseña del usuario que ingresa al sitio administrador, cuenta con una única función la cual es requerida por la librería strong-soap.

El servicio que valida código y contraseña de la universidad utiliza la tecnología SOAP y es necesario enviarle un JSON con la información a validar, puede referirse a este archivo para ver el JSON que se envía. Adicionalmente, el servicio responde con un 1 si la autenticación fue exitosa o con un 0 si fue incorrecta.

3.3.4. app.js

En este archivo se encuentra la configuración especifica del servidor, es utilizado por el archivo /bin/www y en él se pueden realizar las siguientes definiciones:

- Indicar la carpeta en donde se encuentran las páginas HTML a cargar
- Indicar el motor de vistas a utilizar, que en este caso se utiliza EJS
- Indicar la ubicación del icono que se muestra en el navegador
- Instanciar todos los middlewares que se utilizan
- Indicar o definir las variables globales que se utilicen
- Indicar las URI’s del sitio
- Indicar que acción tomar al encontrar un error en las vistas

Puede referirse al archivo app.js para observar lo mencionado anteriormente.

3.3.5. index.ejs

Página con código HTML que contiene toda la parte que se muestra al usuario cuando ingresa al sitio del administrador.

3.3.6. login.ejs

Página con código HTML que contiene la parte que se muestra al ingresar a la página de inicio de sesión.

3.3.7. mupi2.ejs

Página con código HTML que contiene toda la parte que el usuario visualiza cuando ingresa al sitio del mupi.

3.3.8. menu.js

Este archivo cuenta con toda la lógica y acciones de los elementos HTML que se encuentran en el archivo /views/index.ejs, utiliza JQuery y llamadas por el método POST para obtener datos del servidor.

3.3.9. mupiControl.js

Este archivo cuenta con toda la lógica y acciones de los elementos HTML que se encuentran en el archivo /views/mupi2.ejs, utiliza JQuery y llamadas por el método POST para obtener datos del servidor. Adicionalmente, este archivo cuenta con la implementación de la librería de Google para visualizar mapas, para obtener acceso a este servicio de Google es necesario registrarse en su página y obtener un token que se especifica en la llamada a la librería, en este archivo.

3.4. Procesos del sistema

El sistema cuenta con cuatro procesos indispensables para su funcionalidad y administración. Conocer el flujo interno de cada uno de estos procesos permite comprender y predecir el comportamiento de las diferentes funcionalidades de la aplicación. Por tal razón se presentan los diagramas de flujo de dichos procesos, con el objetivo de mostrar la reacción del sistema derivado de las acciones de los usuarios y así tener una mejor y más amplia perspectiva del sistema.

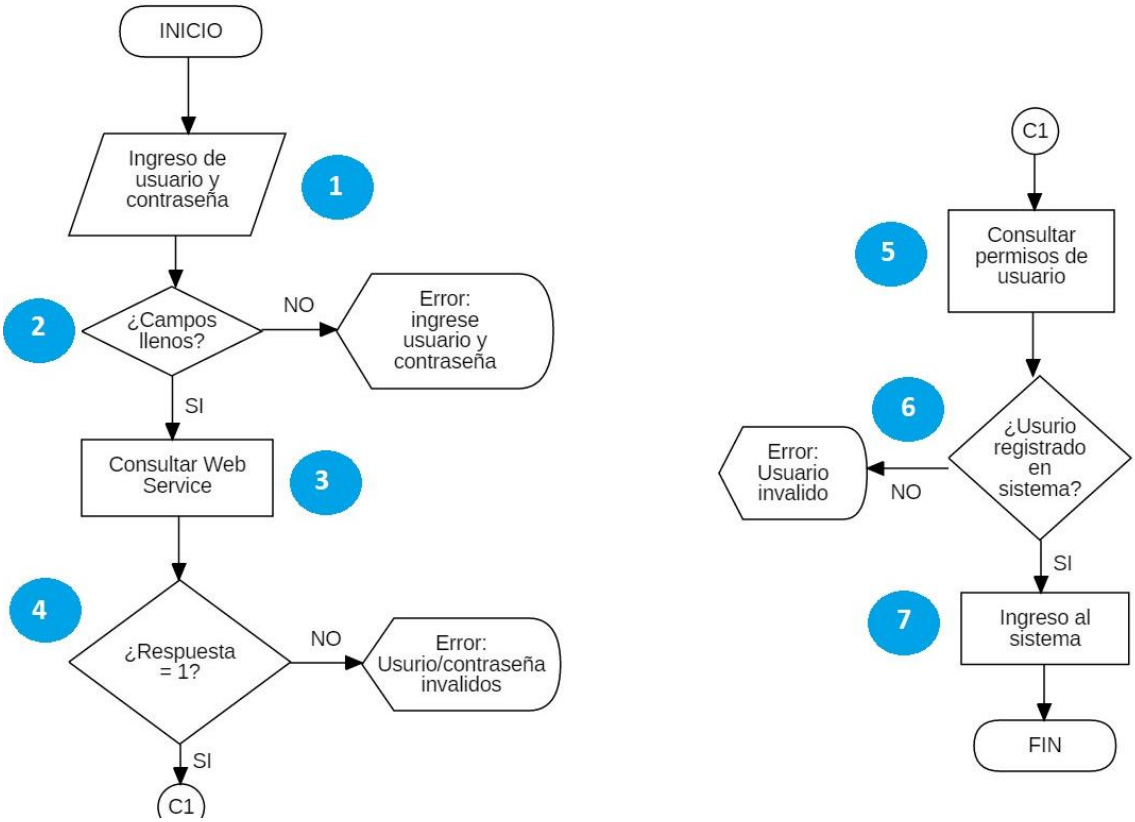
3.4.1. Inicio de sesión

El primer proceso al que el usuario tiene acceso es el de inicio de sesión, el cual se describe a continuación:

1. El usuario administrador ingresa sus credenciales en la página web, la cual se encuentra físicamente en la ruta /mupi_v4/views/login.ejs.
2. La primera validación se realiza por medio Boostrap y CSS, consiste en validar que los campos no viajen vacíos al servidor.
3. Posteriormente se envía al servidor la petición POST a la ruta /admin, que se encuentra en el archivo /mupi_v4/routes/index2.js, la cual se encarga de conectarse al Web Service de la universidad y verificar que el usuario exista y este registrado en los servidores de la universidad.
4. El Web Service de la universidad responde con 1 si la autenticación es correcta y con un 0 en caso contrario, esto se valida en el mismo archivo del punto anterior.
5. En el mismo archivo y ruta que se mencionó en el inciso 3 se valida en base de datos local que el usuario tenga permisos para ingresar al sitio, por medio de una consulta a la base de datos.
6. Si la respuesta de la base de datos es correcta se registra la sesión en el sistema.
7. Se hace una petición POST a la ruta /login la cual se encuentra en el archivo /mupi_v4/routes/index2.js, la cual carga la página login.ejs ubicada en el directorio /mupi_v4/views.

A continuación, se presenta el diagrama que resume el proceso descrito anteriormente:

Figura 15. **Diagrama de flujo, inicio de sesión**



Fuente: elaboración propia con software StarUML.

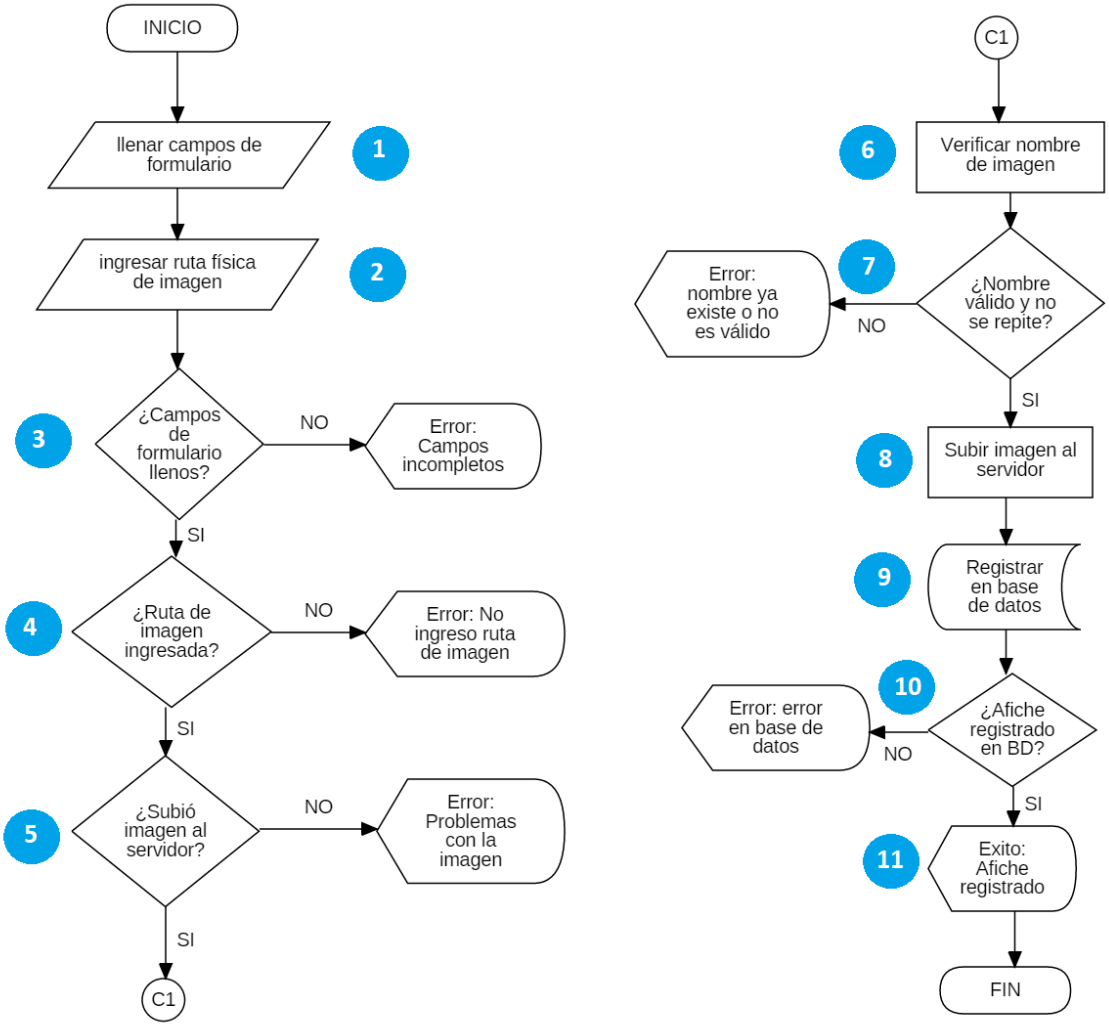
3.4.2. Registrar afiche

Otro proceso al cual tienen acceso los usuarios administradores es el de registrar un afiche, el cual se describe a continuación:

- Se ingresa datos a todos los campos del formulario de la página index.ejs la cual se encuentra en el directorio /mupi_v4/views.
- Dentro de la página mencionada anteriormente se debe utilizar el plugin para seleccionar una imagen. El plugin registra la ruta física donde se encuentra la imagen para posteriormente cargarla.
- La primera validación se realiza en el archivo menu.js el cual se encuentra en el directorio /mupi_v4/public/js y se encarga de verificar que todos los campos del formulario estén llenos.
- La segunda validación se realiza por medio de una bandera que se activa al momento de haber seleccionado una imagen para subir, si la bandera está activa significa que el usuario selecciono una imagen, si no, retornará un error, esto se valida en el mismo archivo del inciso anterior.
- Se realiza una petición POST desde el archivo menu.js a la ruta /carga3 ubicada en el archivo index2.js la cual sube la imagen al servidor.
- Se realiza una validación si el nombre del nuevo afiche ya existe en los afiches registrados, esto se realiza por medio de una petición POST a la ruta /nombreAfiche.
- La petición anterior retorna un 0 si el nombre es válido y no se repite
- Se realiza una petición POST a la ruta /carga2 ubicada en el archivo index2.js.
- La petición anterior se encarga de registrar todos los datos del afiche en la base de datos.
- Se valida que el registro en base de datos haya sido correctamente insertado, esto en el archivo menu.js.
- Por último, se envía un mensaje por socket a todos los demás MUPIS para que se actualicen, esto se hace siempre en el archivo menu.js.

A continuación, se prestan el diagrama que resume el proceso anterior:

Figura 16. **Diagrama de flujo, registrar afiche**



Fuente: elaboración propia con software StarUML.

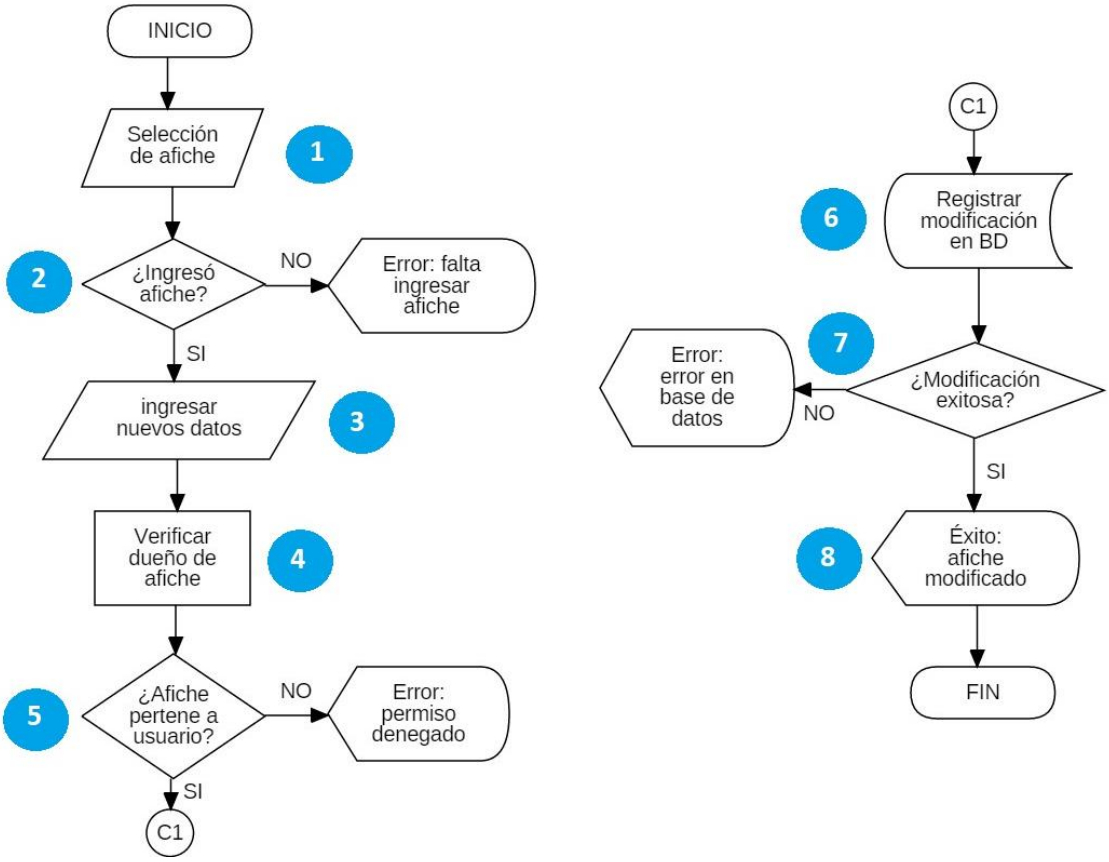
3.4.3. Modificar afiche

Continuando con las funcionalidades disponibles para los usuarios administradores, a continuación, se describe el proceso para modificar un afiche:

- En la página index.ejs ubicada en el directorio /mupi_v4/views se selecciona de un combobox el afiche a modificar.
- Por medio de la variable index ubicada en el archivo menu.js se valida que exista un afiche seleccionado previamente.
- Se realiza una petición POST a la ruta /afichesBD2 para obtener los datos del afiche seleccionado y mostrarlos al usuario en un formulario para que los pueda modificar.
- Se realiza una petición POST a la ruta /modificarA en la cual el primer paso es verificar si el usuario actual tiene permitido modificar el afiche.
- La función verificarDueno que se encuentra en el archivo index2.js devuelve un 1 si el usuario actual tiene permitido modificar el afiche.
- Al comprobar el dueño del afiche, se realiza la inserción de los nuevos datos del afiche en la base de datos. Esto se realiza en el archivo index2.js
- Si la inserción se realiza con éxito devuelve un 1.
- Se notifica al usuario que la modificación tuvo éxito. Esto se realiza en el archivo menu.js

A continuación, se prestan el diagrama que resume el proceso anterior:

Figura 17. **Diagrama de flujo, modificación de afiche**



Fuente: elaboración propia con software StarUML.

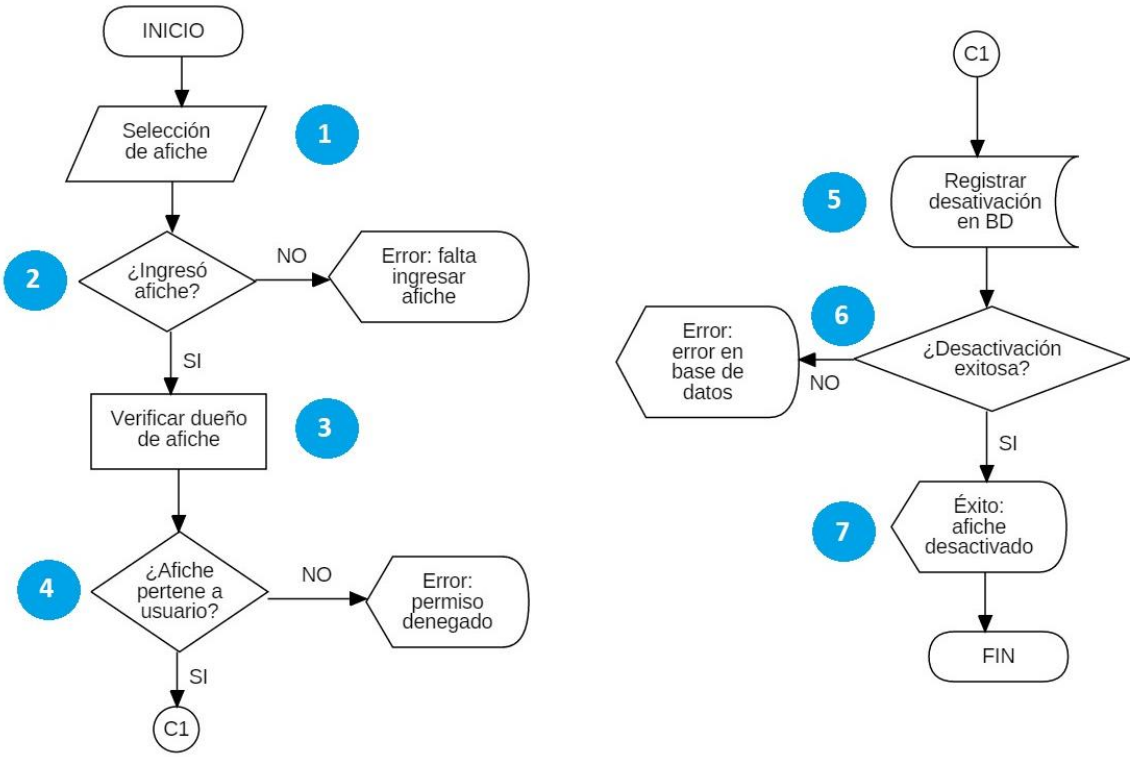
3.4.4. Desactivación de afiche

Por último, se cuenta con el proceso de desactivación de un afiche, el cual se describe a continuación:

- En la página index.ejs ubicada en el directorio /mupi_v4/views se selecciona de un combobox el afiche a desactivar.
- Por medio de la variable index ubicada en el archivo menu.js se valida que exista un afiche seleccionado previamente.
- Se realiza una petición POST a la ruta /cancelarA en la cual el primer paso es verificar si el usuario actual tiene permitido desactivar el afiche.
- La función verificarDueno que se encuentra en el archivo index2.js devuelve un 1 si el usuario actual tiene permitido desactivar el afiche.
- Al comprobar el dueño del afiche, se realiza la desactivación cambiando el campo estado en la base de datos. Esto se realiza en el archivo index2.js.
- Si la desactivación se realiza con éxito devuelve un 1
- Se notifica al usuario que la desactivación tuvo éxito. Esto se realiza en el archivo menu.js.

A continuación, se prestan el diagrama que resume el proceso anterior:

Figura 18. **Diagrama de flujo, desactivación de afiche**



Fuente: elaboración propia con software StarUML.

3.5. Requisitos recomendados e instalación

En esta sección se detallan los aspectos técnicos que se deben de tomar en cuenta para realizar la instalación del sistema, estos aspectos son los recomendados y con los cuales se garantiza un correcto funcionamiento, sin embargo, es posible incorporar el sistema a un ambiente que contenga aspectos técnicos parecidos.

También es importante conocer como ejecutar correctamente la aplicación para su correcto funcionamiento, es por eso que se presentan una serie de comandos útiles con los que se puede gestionar y administrar la aplicación.

Los requisitos mínimos que se recomiendan para la instalación del servicio que contiene todo el sistema son los siguientes:

Tabla XXI. **Requisitos recomendados del servidor**

Aspecto	**Recomendado**
Frecuencia de CPU	1.5 GHz
RAM	2GB
Sistema Operativo	GNU/Linux
Versión NodeJS	4.8.4
MySQL	5.5.57
Internet	Requerido

Fuente: elaboración propia.

La implementación del cliente se llevará a cabo en una Raspberry Pi 3 la cual tiene los siguientes aspectos técnicos con la que se asegura un correcto funcionamiento del sistema:

Tabla XXII. **Requisitos recomendados del cliente**

Aspecto	**Recomendado**
Frecuencia de CPU	1.2Ghz
RAM	1GB
Sistema Operativo	Raspbian strech versión septiembre 2017
Navegador Web	Chromium (integrado en el sistema operativo)
Internet	Requerido

Fuente: elaboración propia.

La instalación del sistema es muy sencilla, sin embargo, es necesario confirmar que todas las librerías y requisitos descritos en las secciones anteriores de este capítulo se cumplan. A continuación, se presentan los pasos necesarios para levantar el servicio NodeJS:

1. Descomprimir el archivo ‘mupi_v4.rar’ el cual contiene una carpeta con el mismo nombre.
2. Una vez obtenida la carpeta llamada ‘mupi_v4’ copiarla en una ruta del servidor de su preferencia, puede ser cualquier ruta, se recomienda que sea corta y que no la olvide.
3. Dentro la carpeta mencionada anteriormente, ubicar la carpeta ‘bin’ ya que dentro de ella se encuentra el archivo ‘www’ el cual hay que ejecutar para iniciar el servicio.
4. Estando en la ruta /mupi_v4/bin/ ejecutar el comando `pm2 start www` El cual inicia el servicio que mantendrá siempre en ejecución el servidor NodeJS, incluso cuando se reinicie el servidor físicamente.
5. Una vez iniciado el servicio podemos confirmar su estado consultándolo con el siguiente comando `pm2 show www`. Este comando nos mostrará información general del servicio.

Por otra parte, si en algún momento se requiriera detener el servicio para realizar algún cambio o simplemente mantenimiento del servidor, se puede ejecutar el siguiente comando:

```
pm2 stop www
```

Existen más comandos disponibles en la librería pm2 por lo cual se puede consultar su documentación en internet para obtener un conocimiento más avanzado de esta herramienta.

3.6. Solución de problemas

En esta sección encontrará la solución a errores que pueden presentarse durante la gestión del sistema:

3.6.1. La página principal del MUPI no muestra el carousel de imágenes

Este problema se presenta cuando el sistema no logra conectarse a la base datos, la cual se encuentra en el mismo servidor que el sistema principal. Es necesario confirmar que los datos de conexión a la base de datos sean los correctos. A continuación, se presentan los pasos para verificar el problema y solucionarlo.

1. Confirmar error en la conexión a la base de datos con el comando `pm2 logs www`. Con este comando se mostrará el registro de mensajes que deja la aplicación. Si observa el mensaje 'Error connecting database' significa que el sistema no pudo conectarse.
2. Observe los datos de la conexión en el archivo /mupi_v4/routes/index2.js en la línea 14 a la 19. Confirme con el administrador de la base de datos que los parámetros de conexión no han cambiado, en caso contrario, digite los nuevos datos para que el sistema logre conectarse.

Si el error persiste y el carousel no se muestra en la página del MUPI verifique que el navegador WEB este actualizado o bien utilice el navegador Web recomendado en la sección 3.5 que hace referencia a los requisitos recomendados e instalación.

3.6.2. Ningún usuario administrador logra ingresar a la página de administración después de ingresar sus credenciales

Este problema puede presentarse debido a un error en la conexión entre el Web Service de la universidad y el sistema principal. Es importante recordar el proceso de ingreso de sesión de los usuarios, por lo que puede verificar la sección 3.4.1 que hace referencia al proceso de inicio de sesión. A continuación, se presentan los pasos para verificar y solucionar el problema:

1. Confirmar error en la conexión al Web Service con el comando `pm2 logs www`. Con este comando se mostrará el registro de mensajes que deja la aplicación. Si observa el mensaje 'Error en SOAP = Error' significa que el sistema no pudo conectarse.
2. Verifique con el encargado del Web Service de la universidad que los datos de conexión hacia el servicio no han cambiado o bien que el servicio se encuentre en línea.
3. Si el servicio ha sufrido algún cambio puede dirigirse al archivo /mupi_v4/routes/siif.js en la línea 8 y 20 y realizar la actualización de datos de conexión.

Si después de verificar los pasos anteriores el error persiste, debe asegurarse que el servidor donde se encuentra alojada la aplicación del MUPI se encuentre en la misma red que el servidor que aloja el Web Service de la universidad.

3.6.3. La opción de localizar edificios en la página principal del MUPI no muestra el mapa de Google.

Este error puede presentarse en la raspberry pi, debido a que pierde la conexión a internet. El sistema muestra un mensaje informando la falta de conexión a internet.

Es importante recordar que el sistema hace uso del API de Google para mostrar el mapa, y este a su vez necesita conexión a internet, por lo que es necesario ingresar remotamente a la Raspberry pi y verificar la conexión a internet.

Cuando la Raspberry pi logre de nuevo tener conexión a internet, reinicie o refresque la página Web para confirmar de nuevo el correcto funcionamiento de la misma.

4. PRUEBAS

Definir y realizar casos de pruebas es parte fundamental en el proceso de aseguramiento de la calidad del software ya que permiten identificar y corregir errores en una etapa temprana en el ciclo de desarrollo de una aplicación.

También permite verificar que los requerimientos que inicialmente se trazaron hayan sido alcanzados y que la parte interesada o *Stakeholder* esté satisfecha con el software que se entrega como producto final.

En este capítulo se describen los casos de prueba identificados para el sistema y se presentan los resultados obtenidos en cada prueba. El objetivo de este capítulo es mostrar que el sistema funciona correctamente, validar que los requerimientos planificados estén cubiertos, así como identificar posibles errores o limitaciones que no se hayan tenido en cuenta.

4.1. Pruebas de software funcionales

Las pruebas funcionales se realizan a partir de los requerimientos funcionales definidos en el segundo capítulo, a partir de ellos se establecen casos de prueba y se contrastan los resultados obtenidos con los resultados esperados.

En este caso, para la realización de este tipo de pruebas se utiliza la técnica de caja negra, la cual se enfoca en las entradas y salidas esperadas por el software sin importar internamente como este compuesta la aplicación. Para este tipo de pruebas también es válido utilizar la técnica de caja blanca, sin embargo, la funcionalidad de este sistema está basado en consultar una base de datos y mostrar los resultados, es decir, no cuenta con un arduo proceso de

análisis de datos o bien no cuenta con algoritmos internos que se ejecuten para proveer una salida al sistema. Es por eso que las pruebas de caja negra se ajustan más a lo requerido en este capítulo.

A continuación, se describen los casos de prueba identificados y ejecutados para este sistema:

Tabla XXIII. **Caso de Prueba – Ingreso al sistema**

ID: CP01		**Fecha**: 18-09-2017
Título: Ingreso al sistema		**Área**: Administrador
Descripción: El inicio de sesión permite el ingreso al sistema por lo tanto se debe ingresar un código y contraseña y presionar el botón 'Entrar' para que el servidor valide las credenciales y así determinar si es permitido el ingreso o no. Este proceso pasa por 2 validaciones, la primera es validar la credenciales en el Web Service interno de la universidad, y la segunda es validar que el código del usuario ingresado este registrado en la base de datos de la aplicación.		
Datos de entrada	**Datos de salida**	**Datos esperados**
Código y contraseña válidos en el Web Service y código registrado en la aplicación.	Entrar al sistema	Entrar al sistema
Código y contraseña válidos en el Web Service y código no registrado en la aplicación.	Mensaje de error	Mensaje de error
Código y contraseña no válidos para el Web Service y código no registrado en aplicación.	Mensaje de error	Mensaje de error

Fuente: elaboración propia.

Tabla XXIV. **Caso de Prueba – Registrar Afiche**

ID: CP02		**Fecha**: 18-09-2017
Título: Registrar afiche		**Área**: Administrador
Descripción: Para registrar un afiche es necesario llenar un formulario con cinco campos y seleccionar/subir una imagen al servidor.		
Datos de entrada	**Datos de salida**	**Datos esperados**
Se llenan los cinco campos con información correcta, se selecciona y sube una imagen al servidor.	Mensaje de éxito	Mensaje de éxito
Se deja un campo vacío, se selecciona una imagen y se sube al servidor.	Mensaje de error indicando que falta un campo	Mensaje de error indicando que falta un campo
Se llenan los cinco campos y no se selecciona ninguna imagen para subir al servidor.	Mensaje de error indicando que no ha seleccionado ninguna imagen	Mensaje de error indicando que no ha seleccionado ninguna imagen
Se llenan los cinco campos pero el campo título contiene un nombre de afiche repetido.	Mensaje de error indicando que el nombre de afiche ya existe	Mensaje de error indicando que el nombre de afiche ya existe

Fuente: elaboración propia.

Tabla XXV. **Caso de Prueba – Visualizar Afiche**

ID: CP03		**Fecha**: 19-09-2017
Título: Visualizar afiches		**Área**: Administrador
Descripción: En esta opción es posible ver la imagen y el titulo de los afiches disponibles en el servidor, a su vez, es posible observar en tiempo real cuando un usuario agrega o desactiva un afiche ya que esta opción se refresca.		
Datos de entrada	**Datos de salida**	**Datos esperados**
Ninguno. Solamente se selecciona la opción de ver los afiches.	Afiches disponibles en el servidor	Afiches disponibles en el servidor
Afiche agregado con éxito al servidor.	Afiches disponibles en el servidor	Afiches disponibles en el servidor
Afiche desactivado con éxito en el servidor.	Afiches disponibles en el servidor	Afiches disponibles en el servidor

Fuente: elaboración propia.

Tabla XXVI. **Caso de Prueba – Modificar/Desactivar Afiche**

ID: CP04		**Fecha**: 19-09-2017
Título: Modificar/Desactivar afiche		**Área**: Administrador
Descripción: Para modificar un afiche o bien desactivarlo es necesario seleccionarlo dentro de la lista de los afiches, sin embargo, es requerido ser el dueño registrado en el sistema del afiche ya que si no lo es, no puede modificar ni desactivar el afiche seleccionado.		
Datos de entrada	**Datos de salida**	**Datos esperados**
Modificar un campo del afiche seleccionado, siendo el dueño.	Mensaje de éxito	Mensaje de éxito
Modificar un campo del afiche seleccionado, no siendo el dueño.	Mensaje de error indicando que no es el dueño del afiche	Mensaje de error indicando que no es el dueño del afiche
Desactivar un afiche siendo el dueño.	Mensaje de éxito	Mensaje de éxito
Desactivar un afiche no siendo el dueño.	Mensaje de error indicando que no es el dueño del afiche	Mensaje de error indicando que no es el dueño del afiche

Fuente: elaboración propia.

Tabla XXVII. **Caso de Prueba – Limpiar Servidor**

ID: CP05		**Fecha**: 19-09-2017
Título: Limpiar servidor		**Área**: Administrador
Descripción: Cuando un usuario desactiva un afiche este es marcado en base de datos con una bandera indicando que esta desactivado, sin embargo para borrarlo físicamente de la base de datos y del servidor es necesario presionar el botón de limpiar		
Datos de entrada	**Datos de salida**	**Datos esperados**
Ninguno. Solamente se selecciona la opción.	Lista de afiches desactivados previamente	Lista de afiches desactivados previamente
Presionar el botón de limpiar	Lista vacía y botón de limpiar desactivado	Lista vacía y botón de limpiar desactivado

Fuente: elaboración propia.

Tabla XXVIII.**Caso de Prueba – Selección de Afiche**

ID: CP06		**Fecha**: 21-09-2017
Título: Selección de afiche		**Área**: MUPI
Descripción: La interfaz del usuario del MUPI cuenta con esta opción para visualizar todos los afiches disponibles, es posible seleccionar el afiche que se necesita ver desde una lista desplegable.		
Datos de entrada	**Datos de salida**	**Datos esperados**
Ninguno. Solamente se selecciona la opción.	Lista de afiches disponibles para consultar	Lista de afiches disponibles para consultar
Selección de un afiche	Título del afiche con su imagen	Título del afiche con su imagen

Fuente: elaboración propia.

Tabla XXIX. **Caso de Prueba – Visualización de Afiches**

ID: CP07		**Fecha**: 21-09-2017
Título: Visualización de afiches		**Área**: MUPI
Descripción: al ingresar a la página principal del MUPI es posible observar las imágenes de los afiches, las cuales se muestran de una forma cíclica, cambiando de imagen cada 5 segundos. Adicionalmente se puede observar en tiempo real cualquier cambio de parte del usuario administrador en la visualización de los afiches.		
Datos de entrada	**Datos de salida**	**Datos esperados**
Ninguno. Solamente se selecciona la opción.	Imágenes de los afiches rotando cada 5 segundos	Imágenes de los afiches rotando cada 5 segundos
Registro de un afiche por parte del usuario administrador.	Imágenes de los afiches, incluyendo la nueva imagen, rotando cada 5 segundos	Imágenes de los afiches, incluyendo la nueva imagen, rotando cada 5 segundos
Desactivar un afiche por parte del usuario administrador.	Imágenes de los afiches, excluyendo la imagen desactivada, rotando cada 5 segundos	Imágenes de los afiches, excluyendo la imagen desactivada, rotando cada 5 segundos

Fuente: elaboración propia.

4.2. Pruebas de software no funcionales

Este tipo de pruebas se basan en los requerimientos no funcionales identificados en el diseño de la aplicación, en este caso, especificados en el capítulo dos. Dentro de los requerimientos no funcionales más importantes se encuentra el de la cantidad de usuarios que puedan ingresar al mismo tiempo al sistema, por lo que se realizaron pruebas de carga con el objetivo de medir los tiempos de respuesta del sistema.

A continuación, se presenta un resumen de los resultados obtenidos en las pruebas de carga, realizadas con ayuda del software libre escrito en java, Apache JMeter, el cual cuenta con muchas funciones para realizar pruebas de carga y rendimiento sobre un sistema. Es importante mencionar que se tomaron las rutas más importantes y concurrentes dentro del sistema.

Tabla XXX. **Resultados de pruebas de carga**

Ruta	Usuarios por segundo	Tiempo promedio de respuesta
/afichesDisponibles	400	2.63
/afichesBD	450	3.33
/afichesBD2	600	0.8
/obEdificios	600	1.8
/obEdificio	800	1.3
/obTipoServicio	400	0.5
/busquedaxedificio	550	0.7
/busquedaxservicio	650	1.11
/busquedaxediservi	600	0.9
/busquedaServicioIndividual	650	1.7

Fuente: elaboración propia.

Como se puede observar en la tabla anterior, el sistema permite interactuar a una gran cantidad de usuarios con tiempos de respuesta relativamente bajos. Existen dos rutas las cuales cuentan con tiempo de respuesta bastante alto,

esto es debido a que la consulta a la base de datos obtiene más información que las demás rutas.

4.3. Implementación del prototipo de MUPI

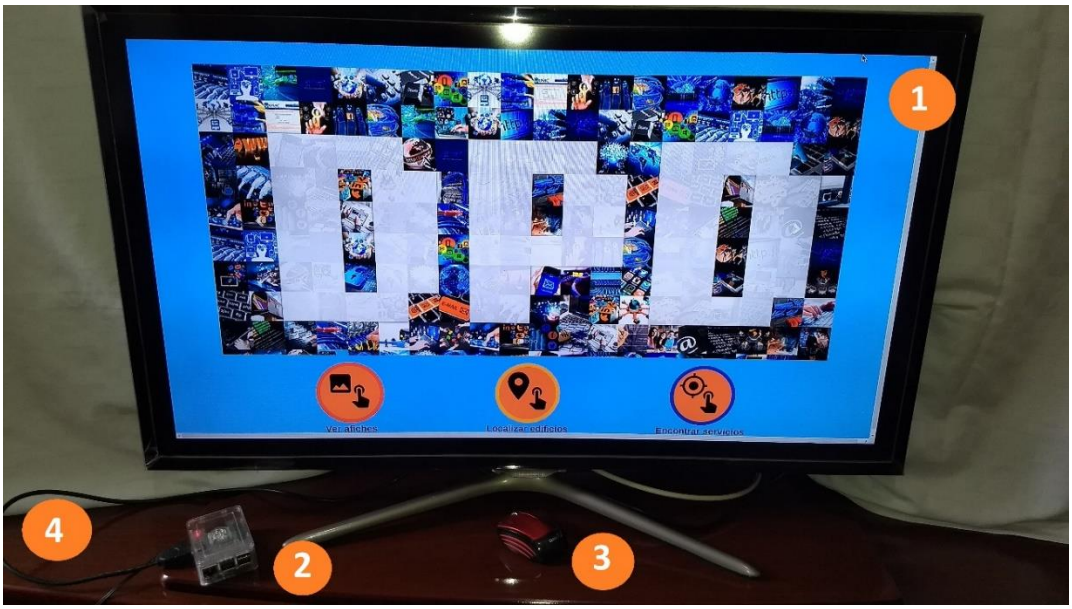
Este prototipo tiene como objetivo mostrar los elementos que son necesarios para la implementación de un MUPI. Es necesario contar con los siguientes elementos:

1. Pantalla de cualquier tamaño. Para este ejemplo se utilizó una pantalla de 40 pulgadas.
2. Raspberry Pi. Para este ejemplo se utilizó la versión 3
3. Mouse. Para este ejemplo se utilizó un mouse inalámbrico
4. Cable HDMI. Utilizado para conectar la Raspberry Pi a la pantalla

Es importante mencionar que existe otra posibilidad para realizar la implementación de un MUPI, la cual consiste en utilizar una pantalla táctil y con esto sería innecesario el mouse.

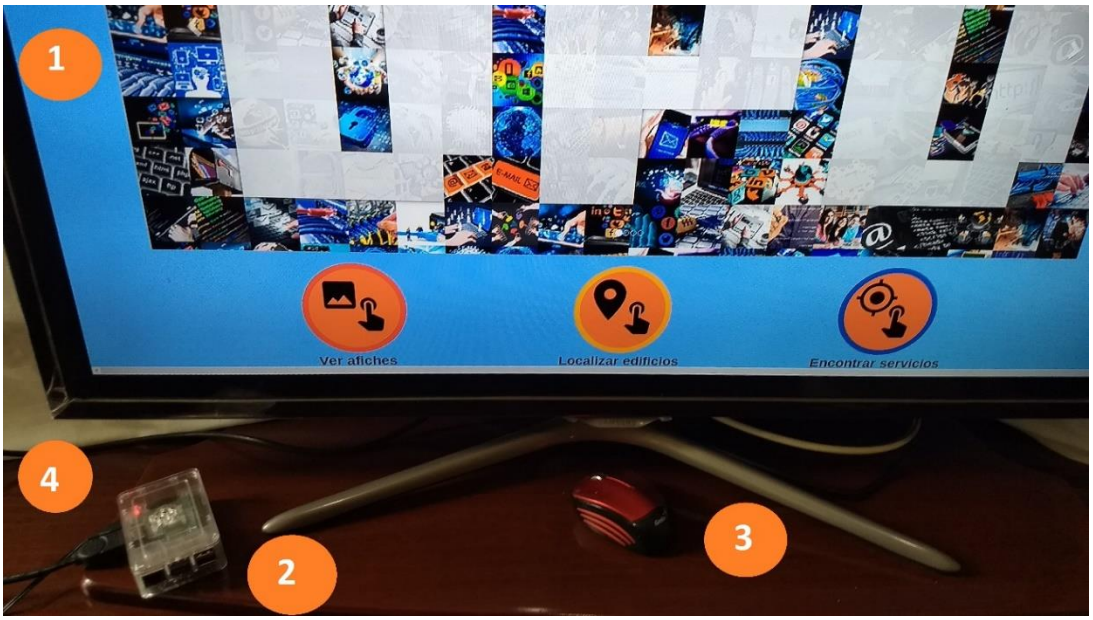
A continuación, se presentan 3 imágenes que muestran el prototipo implementado con cada uno de los elementos mencionados anteriormente. En la tercera imagen se muestra una de las opciones del MUPI accedida desde el mouse inalámbrico.

Figura 19. **Prototipo MUPI**



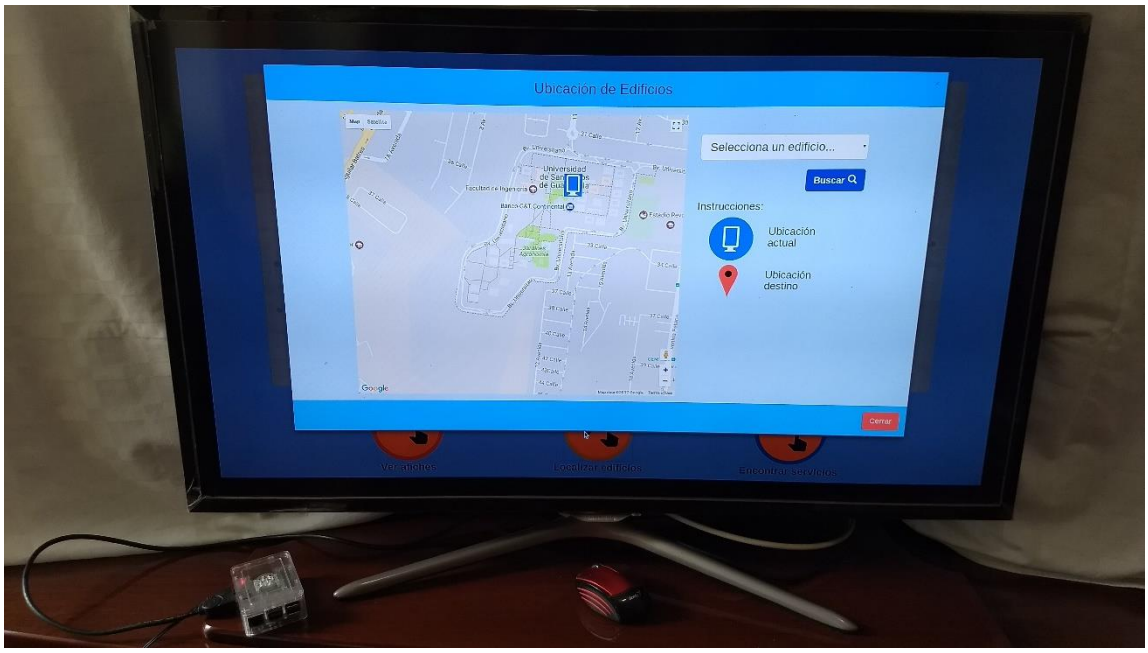
Fuente: elaboración propia.

Figura 20. **Prototipo MUPI de cerca**



Fuente: elaboración propia.

Figura 21. **Opción de mapas en el MUPI**



Fuente: elaboración propia.

4.3.1. Costos de implementación

A continuación, se presentan dos cotizaciones de materiales necesarios para la implementación del MUPI.

Tabla XXXI. **Costos de implementación**

	Con Pantalla Táctil (Q.)	**Sin pantalla Táctil (Q.)**
Pantalla	(22 pulgadas) 2,200	(32 pulgadas)1 900
Raspberry Pi	400	400
Mouse	No necesario	70
Cable HDMI	80	80
Total	Q.2 680	Q. 2 450

Fuente: elaboración propia.

El costo de la Raspberry pi incluye el estuche para proteger el dispositivo, el cargador y la tarjeta SD.

CONCLUSIONES

1. Los usuarios finales que hacen uso de la aplicación al tener poco o mediano conocimiento en tecnología prefieren hacer uso de métodos convencionales para obtener información dentro la universidad.

2. Los usuarios administradores tienen las mismas credenciales en todas las plataformas de la universidad que utilizan, de la misma manera la aplicación desarrollada debe hacer uso de estas credenciales.

3. Las imágenes que utilizan los afiches pueden ser cargadas al servidor desde cualquier punto donde se tenga acceso a la red de la universidad.

RECOMENDACIONES

1. Desarrollar una interfaz web intuitiva y fluida basada en la filosofía SPA (Single-Page Application), para los usuarios que consultan la información en los MUPIS.

2. Implementar una conexión al servicio web que administra las credenciales dentro de la universidad para que los usuarios administradores utilicen el mismo usuario y contraseña en la aplicación desarrollada.

3. Crear una conexión por medio de web sockets que permita la conexión en tiempo real de todos los usuarios y así visualizar las imágenes en el momento que sean cargadas al servidor.

BIBLIOGRAFÍA

1. BERMÚDEZ SILVA, Irving Alexander. *Debian GNU/Linux Para el Usuario Final*. Venezuela: Grupo ICCO, C.A., 2008. 463 p.

2. GIL, pablo. POMARES, Jorge. CANDELAS, Francisco. *Redes y Transmisión de datos*. España: Universidad de Alicante, 2010. 192 p.

3. GRIERA, Jordi Íñigo. ORDINAS BARCELÓ, José María. ALABERN, Llorenç. OLIVÉ, Enric. FUENTES, Jaume. TOURRUELLA, Guiomar. *Estructura de redes de computadores*. España: UOC, 2009. 336 p.

4. HUNGER, Steve. *Debian GNU/Linux Bible*. Wiley, 2001. 696 p.

5. KLEPPNER, Otto. RUSSELL, Thomas. LANE, W. Ronal. KING, Karen. *Publicidad*. México: Pearson Educación, 2005. 784 p.

6. MOCQ, François. Raspberry Pi 2 *Utilice todo el potencial de su nano-ordenador*. España: Ediciones ENI, 2016. 651 p.

7. PONS, Nicolas. Linux *Principios básicos de uso del sistema*. España: Ediciones ENI, 2016. 338 p.

8. SANTOS BARRAZA, Brenda, *Similitudes y diferencias de las gigantografías y mupis de la zona 10, como una opción de*

publicidad exterior, Trabajo de graduación de Licenciatura en Ciencias de la Comunicación, Universidad de San Carlos de Guatemala, Facultad de Ciencias de la Comunicación, 2007. 86 p.

9. UPTON, Eben. HALFACREE, Gareth. *Raspberry Pi User Guide*. Wiley, 2016. 312 p.

10. WARNER, Timothy. *Hacking Raspberry Pi*. Estados Unidos: Que, 2014. 369 p.

ANEXOS

Anexo 1. **Certificado de entrega y aceptación de la aplicación**

USAC
TRICENTENARIA
Universidad de San Carlos de Guatemala

Guatemala 18 de Octubre de 2017

CERTIFICADO ACEPTACIÓN

PROYECTO: Sistema para la Administración de MUPIS
ORGANIZACIÓN: Universidad de San Carlos de Guatemala
CLIENTE: Departamento de Procesamiento de Datos

Por medio de la presente se da cierre formal al proyecto, por las razones especificadas a continuación:

- El sistema cumple con los requisitos funcionales y no funcionales establecidos al inicio del proyecto.
- Actualmente el sistema está implementado y funciona correctamente en el equipo brindado por la organización.
- Se realizaron las pruebas respectivas sobre el sistema, encontrando que todas las funcionalidades se cumplen satisfactoriamente.
- Se entregó un documento que contiene toda la información técnica del sistema.
- Se entregó las credenciales que dan acceso al servidor físico y a la base de datos.

DP JEFATURA

Ingeniera Mayra Ortela Corado García
Jefe Departamento de Procesamiento de Datos

DP Departamento de Procesamiento de Datos
Dirección General Financiera
Sótano Edif. de Rectoría, Ciudad Universitaria, z.12, Guatemala.
PBX: 2418-8000 / Extensión: 9651 / Directo: 2418-9651
www.usac.edu.gt

Fuente: elaboración por parte del Departamento de Procesamiento de Datos.