SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO DE LEÓN

INSTITUTO TECNOLÓGICO DE LEÓN

División de Estudios de Posgrado e Investigación

"PROPUESTA DE UN SISTEMA MULTIAGENTE APLICADO A UN SMART CAMPUS"

TESIS

Que presenta:

ING. JOSÉ FABRICIO DE LA CRUZ PONCE

Para obtener el grado de:

MAESTRO EN CIENCIAS DE LA COMPUTACIÓN

Con la dirección de:

DR. CARLOS LINO RAMÍREZ

Con la co-dirección de:

DR. ARNOLDO DIAZ RAMÍREZ

Revisores:

DR. VÍCTOR MANUEL ZAMUDIO RODRÍGUEZ

DR. DAVID ASAEL GUTIÉRREZ HERNÁNDEZ

LEÓN, GUANAJUATO. JUNIO DE 2022

EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLOGICO NACIONAL DE MEXICO

Instituto Tecnológico de León
Departamento de...

León, Guanajuato, 01/agosto/2022

DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN
OFICIO No. DEPI-095-2021

C. JOSÉ FABRICIO DE LA CRUZ PONCE
PRESENTE

De acuerdo al fallo emitido por la Comisión Revisora, integrada por los: Dr. Carlos Lino Ramírez, Dr. Arnoldo Díaz Ramírez, Dr. Víctor Manuel Zamudio Rodríguez, Dr. David Asael Gutiérrez Hernández y considerando que cubre todos los requisitos establecidos en los Lineamientos Generales para la Operación del Posgrado del Tecnológico Nacional de México, se autoriza la impresión del trabajo de tesis titulado: "Propuesta de un sistema multiagente aplicado a un Smart campus". Lo que hacemos de su conocimiento para los efectos y fines correspondientes.

ATENTAMENTE

Excelencia en Educación Tecnológica®
Ciencia Tecnología y Libertad

LIC. PATRICIA DONATO JIMÉNEZ
SUBDIRECTORA ACADÉMICA

SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA
TECNOLOGICO NACIONAL DE MÉXICO
INSTITUTO TECNOLÓGICO DE LEÓN
DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

C.c.p. Expediente

JMCV/CLDG

Av. Tecnológico s/n Fracc. Industrial Julián de Obregón. C.P. 37290 León, Guanajuato.
Tel.: 477 7105200 e-mail: tecleon@leon.tecnm.mx tecnm.mx | leon.tecnm.mx

EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

León, Guanajuato., a 30 de junio de 2022

C. ING. LUIS ROBERTO GALLEGOS MUÑOZ
JEFE DE SERVICIOS ESCOLARES
P R E S E N T E

Por este medio hacemos de su conocimiento que la tesis titulada **"PROPUESTA DE UN SISTEMA MULTIAGENTE APLICADO A UN SMART CAMPUS"**, ha sido leída y aprobada por los miembros del Comité Tutorial para su evaluación por el jurado del acto de examen de grado al alumno (a) **C. José Fabricio de la Cruz Ponce,** con número de control **M14240570** como parte de los requisitos para obtener el grado de Maestro(a) en Ciencias de la Computación (MCCOM-2011-05).

Sin otro particular por el momento, quedamos de Usted.

A T E N T A M E N T E
COMITÉ TUTORIAL

Dr. Carlos Lino Ramírez
DIRECTOR

Dr. Arnoldo Diaz Ramírez
CODIRECTOR 6 REVISOR

Dr. Víctor Manuel Zamudio Rodríguez
REVISOR (a)

Dr. David Asael Gutiérrez Hernández
REVISOR (a)

INSTITUTO TECNOLÓGICO DE LEÓN

SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLOGICO NACIONAL DE MEXICO

DECLARACION DE AUTENTICIDAD Y DE NO PLAGIO

Yo, **José Fabricio de la Cruz Ponce** identificado con No. control: **M14240570**, alumno (a) del programa de la **Maestría en Ciencias de la Computación**, autor (a) de la Tesis titulada: "**PROPUESTA DE UN SISTEMA MULTIAGENTE APLICADO A UN SMART CAMPUS**" DECLARO QUE:

1.- El presente trabajo de investigación, tema de la tesis presentada para la obtención del título de **MAESTRO (A) EN CIENCIAS DE LA COMPUTACIÓN** es original, siendo resultado de mi trabajo personal, el cual no he copiado de otro trabajo de investigación, ni utilizado ideas, fórmulas, ni citas completas "stricto sensu", así como ilustraciones, fotografías u otros materiales audiovisuales, obtenidas de cualquier tesis, obra, artículo, memoria, etc. en su versión digital o impresa.

2.- Declaro que el trabajo de investigación que pongo a consideración para evaluación no ha sido presentado anteriormente para obtener algún grado académico o título, ni ha sido publicado en sitio alguno.

3.- Declaro que las pruebas o experimentos derivados de esta investigación fueron realizados bajo el consentimiento de los involucrados y con fines estrictamente académicos conforme a criterios éticos de confidencialidad.

Soy consciente de que el hecho de no respetar los derechos de autor y hacer plagio, es objeto de sanciones universitarias y/o legales por lo que asumo cualquier responsabilidad que pudiera derivarse de irregularidades den la tesis, así como de los derechos sobre la obra presentada.

Asimismo, me hago responsable ante el Tecnológico Nacional de México/ Instituto Tecnológico de León o terceros, de cualquier irregularidad o daño que pudiera ocasional por el incumplimiento de lo declarado.

De identificarse falsificación, plagio, fraude, o que el trabajo de investigación haya sido publicado anteriormente; asumo las consecuencias y sanciones que de mi acción se deriven, responsabilizándome por todas las cargas pecuniarias o legales que se deriven de ello sometiéndome a las normas establecidas en los Lineamientos y Disposiciones de la Operación de Estudios de Posgrado en el Tecnológico Nacional de México.

León, Guanajuato a 1 del mes de agosto de 2022

Nombre y firma del autor ()

José Fabricio de la Cruz Ponce

SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

ACUERDO PARA USO DE OBRA (TESIS DE GRADO)

A QUIEN CORRESPONDA

PRESENTE

Por medio del presente escrito, **José Fabricio de la Cruz Ponce** (en lo sucesivo el AUTOR) hace constar que es titular intelectual de la obra denominada: "**PROPUESTA DE UN SISTEMA MULTIAGENTE APLICADO A UN SMART CAMPUS**", (en lo sucesivo la OBRA) en virtud de lo cual autoriza al Tecnológico Nacional de México/Instituto Tecnológico de León (en lo sucesivo TECNM/IT León) para que efectúe resguardo físico y/o electrónico mediante copia digital o impresa para asegurar su disponibilidad, divulgación, comunicación pública, distribución, transmisión, reproducción, así como digitalización de la misma con fines académicos y sin fines de lucro como parte del Repositorio Institucional del TECNM/ ITLeón.

De igual manera, es deseo del AUTOR establecer que esta autorización es voluntaria y gratuita, y que de acuerdo a lo señalado en la Ley Federal del Derecho de Autor y la Ley de Propiedad Industrial el TECNM/IT León cuenta con mi autorización para la utilización de la información antes señalada, estableciendo que se utilizará única y exclusivamente para los fines antes señalados. El AUTOR autoriza al TECNM /IT León a utilizar la obra en los términos y condiciones aquí expresados, sin que ello implique se le conceda licencia o autorización alguna o algún tipo de derecho distinto al mencionada respecto a la "propiedad intelectual" de la misma OBRA; incluyendo todo tipo de derechos patrimoniales sobre obras y creaciones protegidas por derechos de autor y demás formas de propiedad intelectual reconocida o que lleguen a reconocer las leyes correspondientes. Al reutilizar, reproducir, transmitir y/o distribuir la OBRA se deberá reconocer y dar créditos de autoría de la obra intelectual en los términos especificados por el propio autor, y el no hacerlo implica el término de uso de esta licencia para los fines estipulados. Nada de esta licencia menoscaba o restringe los derechos patrimoniales y morales del AUTOR.

De la misma manera, se hace manifiesto que el contenido académico, literario, la edición y en general de cualquier parte de la OBRA son responsabilidad de AUTOR, por lo que se deslinda al (TECNM/ITLeón) por cualquier violación a los derechos de autor y/o propiedad intelectual, así como cualquier responsabilidad relacionada con la misma frente a terceros. Finalmente, el AUTOR manifiesta que estará depositando la versión final de su documento de Tesis, OBRA, y cuenta con los derechos morales y patrimoniales correspondientes para otorgar la presente autorización de uso.

En la ciudad de León, del estado de Guanajuato a los 1 días del mes de octubre de 2022.

Atentamente,

Nombre y firma autógrada de EL AUTOR

José Fabricio de la Cruz Ponce

DEDICATORIA

Dedico esta tesis a mis padres, familiares y amigos, que en todo momento me han acompañado a lo largo de mi vida y preparación profesional.

Así como también, a todo el personal docente y administrativo del Instituto Tecnológico de México en León, que me apoyaron durante mi estadía en la maestría durante los años 2020 a 2022, en especial entre otros al Dr. Carlos Lino Ramírez, Dr. Amoldo Díaz Ramírez, Dr. Víctor Manuel Zamudio Rodríguez, Dr. David Asael Gutiérrez Hernández y M.C.S. Claudia Leticia Díaz González, quienes me apoyaron en todo momento para la investigación y desarrollo que refiere esta tesis.

AGRADECIMIENTOS

Agradezco infinitamente por el apoyo incondicional que siempre me han brindado:

Mis padres:

M.F. C.P. Cruz Víctor de la Cruz Torres y Fabiola Ponce Pierrez

Mis abuelitos:

Eugenio Ponce Diaz y Martha Guillermina Pierrez Blanco

Víctor de la Cruz Solís y Eustolia Torres Ontiveros

Mis hermanos:

Lic. Víctor Eddie de la Cruz Ponce y C.P. Diana Marilú de la Cruz Ponce

Mi cuñada y sobrina:

Lic. Mara Alejandra Pablo Hernández y Gia Natalia de la Cruz Pablo

De igual forma agradezco a mi asesor de tesis al Dr. Carlos Lino Ramírez por brindarme su apoyo, observaciones y conocimientos importantes durante mi desarrollo académico.

Por último, a mis amigos y compañeros de instituto, por su amistad y apoyo moral.

¡¡¡MUCHAS GRACIAS!!!

Índice general

Índice de figuras

Resumen

En el trabajo de tesis, se realizó una integración de sensores, actuadores a una propuesta de una arquitectura basada en agentes inteligentes, los sensores se utilizaron para generar un histórico de los datos del ambiente que por tiempos de pandemia uno de los sensores importantes nos permite hacer la medición de CO_2 y así poder evitar contagios. Por otro lado, se utilizó un actuador para poder manipular una lampara y que el cada agente sea capas de manipular su ambiente por otro lado se hizo la integración de la detección de dispositivos móviles. El propósito de los agentes es llevar un sondeo o un reconocimiento del ambiente en el cual se instalen de esta manera poder proporcionar una estadía mas agradable a las personas que se encuentran en el ambiente, por otra parte, el objetivo de cada agente es poder tomar decisiones para mejorar dicha estadía.

Capítulo 1

Introducción

En el mundo actual la mayor parte de la población está muy acostumbrada al uso de dispositivos *Smart* ya sean móviles, tables, laptops, relojes inteligentes, pulseras inteligentes, etc. y en algunos de los casos desentiendes de esos mismos.

En el articulo ”Ubicación de dispositivos móviles en ambientes interiores por medio de análisis de radiación de redes WiFi y deformaciones de campo magnético”[Gómez R. y Pedraza, 2018], se menciona que la ”localización dentro de ambientes de interiores basada en dispositivos móviles modernos es un área actual de investigación, dado que constituye una fuente de información indispensable para el desarrollo de servicios según el contexto local del usuario. Es importante el estudio de las diferentes variables que permiten obtener información captable por teléfonos inteligentes sobre el entorno espacial del usuario, así como las fuentes de errores de esas informaciones y la forma de mitigarlos” [Gómez R. y Pedraza, 2018], pensando en esto se puede dar la creación de agentes inteligentes dentro de un ambiente que pueda detectar la ubicación y puedan surgir desarrollos de inteligencia artificial con el objetivo de ayudar a los habitantes del ambiente.

En los últimos años se han creado agentes inteligentes basado ambientes de interiores como, Alexa por parte de Amazon, Google Home por parte de Google que están enfocados en el hogar mismos que son capaces de detectar la voz del usuario y recibir ordenes para interactuar con el ambiente, por otro lado, existen diversas aplicaciones industriales, aplicaciones médicas, áreas de entretenimiento, aplicaciones comerciales.

Pensando en la escuela, en los docentes y alumnos surge la idea de crear un sistema multiagente que haga para el alumno ser más ameno su trayectoria como alumno. Los agentes tendrán como objetivo recopilar información del Instituto Tecnológico de León, la cual nos servirá para que una inteligencia artificial pueda analizarlos y tomar decisiones o simplemente dar recomendaciones a los administradores o docentes del plantel, ya sea de una problemática, del consumo de luz, la seguridad dentro y fuera del campus, o a los alumnos notificarles de sus materias si necesitan estudiar más y a su vez notificar a su asesor o maestro de que está fallando en la materia y poder ayudarlo.

El proyecto fue desarrollado en un sistema multiagente que será capaz de obtener los datos con diferentes sensores ya sean wifi, de humedad, bluetooth y ambientales. Los agentes estarán montados en un Raspberry Pi. Los datos obtenidos serán tratados con lógica difusa. La Lógica Difusa proporciona un mecanismo de inferencia que permite simular los procedimientos de razonamiento humano en sistemas basados en el conocimiento. La teoría de la lógica difusa proporciona un marco matemático que permite modelar la incertidumbre de los procesos cognitivos humanos de forma que pueda ser tratable por un computador

1.1. Antecedentes

Para esta investigación ya se cuenta con una base o una arquitectura que seguir, que es la del artículo ‘‘Propuesta de una arquitectura para un Smart Campus Universitario” [Cruz Parada y col., 2020] donde se describe una arquitectura con el uso de agentes inteligentes conectados a una aplicación móvil que a su vez esta conectada con un sistema interno de la institución con el objetivo o planteamiento de recolección de datos, análisis de información y toma de decisiones automatizadas [Cruz Parada y col., 2020].

Por otro lado, existen proyectos en los cuales se aplican los sistemas multiagenes, con la integración de sensores y actuadores, como por ejemplo en el artículo ”Designing a Multiagent System for Elderly Care” [Martínez y col., 2020], done se puede ver que tienen las bases de conocimiento para esta investigación.

1.2. Definición del problema

Los problemas que podemos encontrar en una institución educativa de nivel superior son muy parecidos a los problemas con los que cuenta una ciudad dado que somos una muestra bastante representativa de la misma, por lo que separamos los diferentes tipos de problemas en los siguientes grupos.

1.2.1. Académico

Muchos estudiantes tienen la iniciativa de empezar una carrera, y el instituto les brinda la atención necesaria para entrar a la carrera, pero una vez dentro, algunos alumnos empiezan a tener un mal desempeño, ya sea porque no comprenden las clases, no les agrada la carrera o simplemente perdieron el interés y no van a clases.

1.2.2. Bienestar físico, mental emocional y social

Parte del punto anterior de que los alumnos dejan de asistir a clases puede deberse a un problema de bienestar, emocional o social. La cuestión aquí, es el saber por qué de su deserción y dar apoyo a los alumnos que lo necesitan. Por otra parte, algunos alumnos se estresan en clase ya sea por estar mucho tiempo en clase o por no comprender un tema de la clase. La inseguridad dentro y fuera del instituto siempre está presente, puede ocurrir un accidente y nadie se puede dar cuenta de lo que está sucediendo.

1.2.3. Ecológico

Muchas de las veces pasando entre los pasillos nos encontramos que hay aulas en las que no hay nadie y se tiene la luz encendida o por ejemplo dadas ciertas horas de la noche se encienden las luces hasta el día siguiente, por otra parte, tenemos edificios que pueden estar mal ventilados, la humedad sea alta, haya algún tipo de gas, o el calor dentro del edificio no sea el más óptimo. Esta investigación se enfoco en el problema ecológico, ya que se plantea detectar la ubicación de los alumnos y docentes dentro de la institución, desarrollar un sistema con inteligencia artificial

para reducir el consumo eléctrico, monitorear datos ambientales y dejar los cimientos de una aplicación móvil. En un *Smart campus* universitario se pueden implementar múltiples sistemas con inteligencia artificial, como detectar el aislamiento de los alumnos dentro de la institución, implementar un sistema que detecte las emociones de los alumnos a través de preguntas o con algún otro método de la inteligencia artificial, se pueden implementar recomendaciones inteligentes dependiendo de las materias de los alumnos, entre muchos otros estos mismos podrán ser realizados en base a lo que se llegue con esta investigación.

¿Un sistema multiagente podrá hacer que mejore el desempeño de los estudiantes?

¿El sistema hará que se reduzca el consumo de energía eléctrica?

¿El sistema podrá detectar la ubicación de un dispositivo móvil entre todos los agentes atreves de bluetooth?

1.3. Justificación del proyecto

El proyecto atacara el problema ecológico mencionado en el punto 3 “Definición del problema” el cual, necesita ser atendido con urgencia y esperamos que podamos impactar en el entorno del estudiante, por parte de la institución se mejorara la calidad de las aulas o edificio respecto el ambiente en el interior teniendo una mejor calidad para los alumnos como docentes, por lo tanto el alumno tendrá una calidad de estudio y de vida mejor para que así pueda tener una mayor oportunidad de salir a delante en sus estudios profesionales o incluso llegar a un nivel de posgrado.

1.4. Objetivos

1.4.1. General

Desarrollar la propuesta de un sistema multiagente basado en lógica difusa aplicados en un *Smart campus* que nos permitan reducir el uso de energía eléctrica, registrar datos del ambiente como dispositivos móviles, composición del aire (CO y CO_2), temperatura, intensidad de luz, niveles de sonido y humedad.

1.4.2. Específicos

- Diseñar la propuesta de la integración de los agentes al *Smart campus*.
- Modelar base de datos que permitirá el buen funcionamiento del sistema multiagente.
- Desarrollar e implementar agente que permita la disminución de uso de energía eléctrica.
- Diseñar y crear la propuesta del agente que permita la obtención continua de datos del ambiente.

1.5. Alcances y Limitaciones

1.5.1. Alcances

- Reducir uso de la energía eléctrica.
- Detectar la ubicación de dispositivos móviles entre agentes.

1.5.2. Limitaciones

- Instalar agentes en el Instituto.
- No todos los alumnos o docentes harán uso del sistema.

1.6. Hipótesis

Es posible diseñar un sistema multiagente en el entorno de los alumnos y docentes, capaz de monitorear el ambiente continuo de la institución y por medio de lógica difusa reducir el consumo de energía eléctrica.

1.7. Método

El método a realizar para el proyecto, parte de los alumnos y del medio ambiente de la institución, se plantea una solución desde el punto de vista de la línea de investigación "Ambientes Inteligentes".

Se buscará un dispositivo de bajo consumo que pueda ser instalado en diferentes ambientes los cuales se puedan comunicar entre si y además puedan contar con diferentes sensores y actuadores para hacer el monitoreo de las personas en el instituto y el monitoreo del medio ambiente.

Posteriormente al dispositivo, se le instalaran los sensores de luz, de temperatura, de humedad, de sonido, etc. Se le instalar actuadores de ser necesarios. Al dispositivo se le instalará un lenguaje de programación el cual nos permita captar las señales bluetooth de los dispositivos móviles, también nos deberá permitir la lectura de los datos generados por los sensores y activar los actuadores de ser necesarios.

Se realizarán pruebas para la detección de las señales bluetooth para poder ubicar un dispositivo móvil en un ambiente u otro, se realizarán pruebas de la obtención de datos de los sensores.

Se implementará lógica difusa para la medición de los datos que se tienen y en caso de requerir una acción se utilizaran actuadores como ventiladores, mecanismos para cerrar puertas o abrir ventanas, reguladores de voltaje para la luz, etc.

Por último, se creará una app para visualizar los datos del ambiente.

Capítulo 2

Marco Teórico

En esta sección se explicaran temas, conceptos y tecnologías importantes para el desarrollo de la investigación propuesta en el capitulo 1

2.1. Ambiente inteligente

Los ambientes inteligentes son espacios o lugares, como por ejemplo aulas, pasillos, salones, salas de video o de conferencia, etc, que usan la tecnología de sistemas embebidos, así como otras tecnologías de la información y la comunicación, para crear ambientes interactivos que acerquen la computación al mundo físico y a los problemas cotidianos. Los ambientes inteligentes son sistemas en los que la computación es usada para introducir mejoras imperceptibles o superficiales en las actividades comunes, estos sistemas al final son casi transparentes y poco perceptibles para la mayoría de los usuarios [Steventon y Wright, 2010]. Una de las fuerzas motoras del interés emergente en los ambientes altamente interactivos, no solamente es lograr que las computadoras y los sistemas sean verdaderamente amigables con los usuarios, sino lograr que esencialmente sean invisibles o casi inexistentes para él usuario y así dando un beneficio o mejorando la calidad de su estadía en donde estén implementados estos mismos. En base a la definición de ambiente inteligente se contemplan los siguientes puntos integrando tecnología actual y hardware para la implementación o la creación de un ambiente inteligente.

2.1.1. Internet de las Cosas (IoT)

El concepto de IoT es sencillo, aunque su capacidad es ilimitada y su uso puede cambiar todo el paradigma de la tecnología heredada. Se basa en incrustar una interfaz de red en los objetos, permitiendo la comunicación entre ellos para proporcionar diversos servicios para los usuarios. En consecuencia, cada objeto tendrá su propio identificador, como una dirección de Protocolo de Internet (dirección IP) en la Internet actual que puede conectarse y comunicarse con otros objetos a través del entorno de red de IoT. A diferencia de la época anterior a la IoT, en la que los usuarios sólo podían obtener datos del el proveedor de servicios, los usuarios pueden acceder directamente a los sensores y dar órdenes a los actuadores [Lee y col., 2017].

2.1.2. Sistema difuso

Un sistema difuso está constituido por tres bloques principales: el de transformación de los valores numéricos en valores de Lógica difusa es decir un rango de números en el cual se puede considerar que un valor es aceptado como por ejemplo de tal a tal grado de temperatura hace frio; el motor de inferencia que emplea las reglas es decir si la temperatura es baja entonces hace frio; y el bloque de conversión de los valores de la Lógica difusa en valores numéricos es decir una vez evaluado el valor de entrada nos arroja un valor de salida aplicando lógica difusa. Un sistema que emplea lógica difusa actúa como una persona reacciona ante términos imprecisos como “hace calor” o “hace frio”.

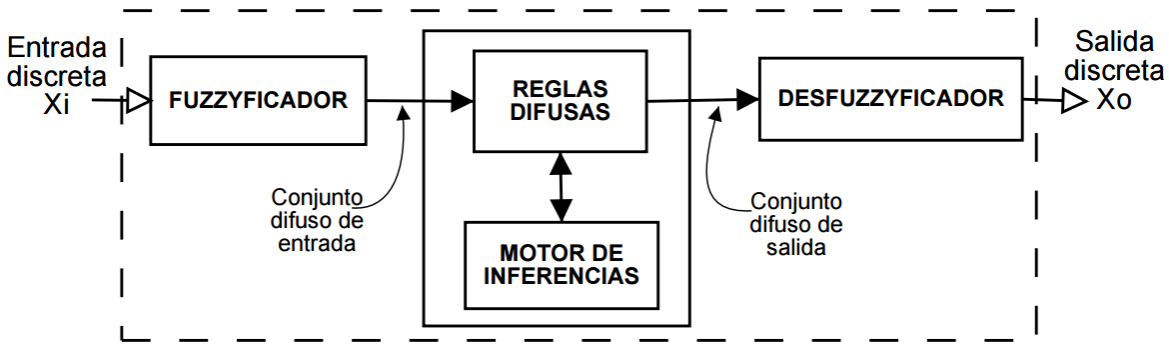


FIGURA 2.1: Sistema difuso (Caparrini y Work, 2019).

- Fuzzyficador: convierte las entradas del sistema, que son valores numéricos nítidos en

conjuntos borrosos aplicando una función de borrosificación.

- Base de conocimiento (Reglas Difusas): almacena las reglas SI-ENTONCES obtenidas de los expertos.
- Motor de inferencia: simula el razonamiento humano haciendo inferencia sobre las entradas recibidas y las reglas SI-ENTONCES almacenadas.
- Desfuzzyficador: convierte el conjunto borroso obtenido por el motor de inferencia en un valor numérico nítido que puede ser reutilizado.

La lógica difusa en un ambiente inteligente tratara de interpretar el ambiente como un ser humano utilizando datos del ambiente obtenidos con sensores y tomar decisiones.

2.1.3. Agente Inteligente

Un agente es una entidad física o virtual que posee ciertas características generales: es capaz de percibir el entorno con la ayuda de sensores y tener una representación parcial del mismo; es capaz de actuar sobre el entorno utilizando actuadores; puede comunicarse con los otros agentes (pueden ser humanos o no) que comparten su hábitat; tiene un conjunto de objetivos que gobiernan su comportamiento y posee recursos propios [Ponce y col., 2014].
Tipos de agentes:

- Agentes reactivos
- Agentes reactivos basados en modelo
- Agentes basados en objetivos
- Agentes basados en utilidad
- Agentes que aprenden
- Agentes de consultas

Se considera como agente inteligente a un sistema que puede realizar un proceso automatizado para alcanzar un objetivo, con la condición de que el sistema reconoce y actúa sobre un entorno, ya que puede percibir información, comunicar o recibir datos y realizar una acción correspondiente [Serna y col., 2019]. Cada agente es capaz de comprender su situación y se adapta a los entornos cambiantes a través la autoconfiguración. La percepción de la situación se logra mediante el aprendizaje y la modelización contextual de los datos de los eventos. Una vez que se aprende un conjunto de bases contextuales a partir de los datos de eventos de alta dimensión, se pueden representar diferentes escenarios mediante los coeficientes contextuales agrupados. Los agentes pueden entonces percibir la situación y localizar las regiones de interés (RoIs) a través de los escenarios identificados. Cada agente tiene una máquina de estados de comportamiento y una biblioteca de comportamiento; elige un cierto comportamiento según el individuo objetivos y el comportamiento de otros agentes [Sun y col., 2013].

2.1.4. Sistemas Multiagente

Los agentes operan en algún entorno que suele ser tanto computacional como físico, por ejemplo, un agente recomendador en la Web o un robot que se desplaza en un hospital, etc. El entorno puede ser abierto o cerrado y suele tener más de un agente. Si bien hay situaciones donde un agente puede operar solo, un creciente número de sistemas están siendo vistos en término de sistemas multiagente, compuestos por agentes autónomos que interactúan entre sí. Aplicaciones orientadas a los agentes se han utilizado en distintos dominios como sistemas de comercio electrónico, sistemas de gestión de procesos distribuidos, sistemas de control de tráfico, etc. [Ponce y col., 2014].

Siguiendo el paradigma de la Ingeniería de Software Orientada a Agentes –AOSE: Agent Oriented Software Engineering (Jennings, 2000) estos sistemas pueden desarrollarse como organizaciones de agentes denominados sistemas multiagente (MAS: Multi-Agent Systems) donde cada agente tendrá su funcionalidad e interactuarán unos con otros y con el entorno en el cuál habitan. Es necesario entonces, analizar propiedades del entorno para que los agentes puedan operar efectivamente e interactuar unos con otros convenientemente. Los entornos proveen una infraestructura computacional para que las interacciones se puedan realizar. Hay que considerar

los protocolos de comunicación y de interacción que utilizarán los agentes [Ponce y col., 2014].

Los protocolos de interacción permiten a los agentes tener conversaciones, que serán intercambios de mensajes estructurados. Para dar un ejemplo concreto de estos conceptos, un protocolo de comunicación puede especificar que los siguientes tipos de mensajes pueden ser intercambiados entre dos agentes [Ponce y col., 2014]:

- Proponer un curso de acción
- Aceptar una acción
- Rechazar una acción
- Retractar una acción
- Estar en desacuerdo con una acción
- Hacer una contrapropuesta de acción

Características de los Sistemas Multiagente [Ponce y col., 2014]:

- Los sistemas multiagente proveen una infraestructura que especifica los protocolos de comunicación y de interacción entre agentes.
- Los entornos multiagente suelen ser abiertos y suelen no tener un diseño centralizado.
- Contienen agentes autónomos y que pueden tener intereses propios o ser cooperativos. Pueden tener una meta común o metas independientes, si un grupo de agentes comparte una meta global, suelen denominarse colaborativos (por ej., un equipo de futbol) en cambio sí tienen distintas metas (generalmente opuestas) suelen ser competitivos (por ej. Agentes de dos equipos de futbol que se enfrentan).
- Pueden compartir conocimientos sobre el problema y las posibles soluciones, el conocimiento global que cada agente tiene puede incluir control global, consistencia global, metas globales, etc.

- Para que un conjunto de agentes pueda desarrollar una actividad conjunta en un entorno compartido debe existir algún tipo de coordinación, la cual puede ser muy compleja. En un grupo de agentes cooperativos puede utilizarse planificación de tareas y en un escenario competitivo, pueden negociarse propuestas.

Para la construcción de un sistema multiagente tenemos el framework PADE que nos sirve para desarrollar, ejecutar y administrar sistemas multiagente en entornos informáticos distribuidos. PADE está escrito 100% en Python y utiliza las bibliotecas del proyecto Twisted para implementar la comunicación entre los nodos de la red. Esto nos es realmente ultil dado que pyhton es uno de los mejores lenguajes para inteligencia artificial. Por otro lado, PADE es un software libre, licenciado bajo los términos de la licencia MIT, desarrollado por el Grupo de Redes Elétricas Inteligentes (GREI) del Departamento de Ingeniería Eléctrica de la Universidad Federal de Ceará. PADE fue desarrollado considerando los requisitos para un sistema de automatización. En base en eso ofrece en su biblioteca los siguientes recursos para el desarrollo de sistemas multiagente [GREI-UFC, 2018]:

- Orientación a objetos
- Entorno de ejecución
- Mensajes en el estándar FIPA-ACL
- Filtrado de mensajes
- Protocolos FIPA
- Comportamientos Cíclicos y Temporales
- Banco de Dados
- Envio de Objetos Serializados

2.1.5. Raspberry Pi

El Raspberry es un dispositivo extraordinario, es una computadora completamente funcional de un tamaño pequeño y de bajo costo. Ya sea que estés buscando un dispositivo que puedas

usar para navegar por el web o jugar a juegos, están interesados en aprender a escribir sus propios programas, o están buscando crear sus propios circuitos y dispositivos físicos. Estos han encontrado su camino en casas, aulas, oficinas, centros de datos, fábricas, e incluso barcos auto pilotados y globos espaciales [Halfacree, 2018].

El Raspberry Pi es conocido como una computadora de una sola placa, lo que significa que: es una computadora de escritorio, una laptop o un *smartphone*, pero construido sobre una única placa de circuito. El Raspberry Pi es pequeño, más o menos del mismo tamaño que una tarjeta de crédito, pero eso no significa que no sea poderoso: un Raspberry Pi puede hacer cualquier cosa que una computadora normal, aunque no necesariamente tan rápido [Halfacree, 2018].

La familia Raspberry Pi nació del deseo de fomentar una educación más práctica en informática en todo el mundo. Sus creadores, que se unieron para formar la Fundación Raspberry Pi sin ánimo de lucro, no tenían ni idea de que sería tan popular: los pocos miles construidos en 2012 para probar se agotaron inmediatamente, y millones han sido enviados a todo el mundo en los años posteriores. Estos han encontrado su camino en casas, aulas, oficinas, centros de datos, fábricas, e incluso barcos auto pilotados y globos espaciales [Halfacree, 2018].

Varios modelos de Raspberry Pi han sido lanzados desde el Modelo B original, cada uno de ellos con especificaciones mejoradas o características específicas para un caso de uso particular. La familia Raspberry Pi Zero, por ejemplo, es una versión diminuta de la Raspberry Pi de tamaño completo que deja caer algunas características en particular los múltiples puertos USB y el puerto de red alámbrica en favor de una disposición significativamente más pequeña y necesidades de energía más bajas [Halfacree, 2018].

2.1.6. Arduino

Arduino es una plataforma electrónica de código abierto basada en hardware y software de fácil uso. Las placas Arduino son capaces de leer entradas de la luz ambiental a través de un sensor, la pulsación de un botón o un mensaje de Twitter y convertirlas en una salida como para activar un motor, encender un LED, publicar algo en línea, etc. Para ello se utiliza el lenguaje de programación Arduino basado en Wiring y el software Arduino IDE (entorno de desarrollo integrado), basado en Processing [«Arduino», 2005]. Wiring es una plataforma

de prototipado electrónico de fuente abierta compuesta de un lenguaje de programación, un entorno de desarrollo integrado (IDE), y un microcontrolador [Barrag, 2003].

ESP32

ESP32 es la denominación de una familia de chips SoC de bajo costo y consumo de energía, con tecnología Wi-Fi y Bluetooth de modo dual integrada. El ESP32 emplea un microprocesador Tensilica Xtensa LX6 en sus variantes de simple y doble núcleo e incluye interruptores de antena, balun de radiofrecuencia, amplificador de potencia, amplificador receptor de bajo ruido, filtros, y módulos de administración de energía. El ESP32 fue creado y desarrollado por Espressif Systems y es fabricado por TSMC [«Overview: Espressif Systems», 2016].

2.1.7. Sensores

El término sensor se refiere a un elemento de medición que detecta la magnitud de un parámetro físico y lo cambia por una señal que puede procesar el sistema. Al elemento activo de un sensor se le conoce comúnmente como transductor. El diseño de sensores y transductores siempre involucra alguna ley o principio físico o químico que relaciona la cantidad de interés con algún evento medible [LATAM, 2021].

Los sistemas de monitorización y control requieren sensores para medir cantidades físicas tales como posición lineal, posición angular, desplazamiento, deformación, aceleración, presión, caudal, fuerza, velocidad lineal y velocidad angular, temperatura, intensidad lumínica, distancia y vibración [LATAM, 2021].

Los sensores son la interfaz perceptual entre un agente y su entorno. Los sensores pasivos, como las cámaras, son observadores verídicos del entorno: capturan las señales que son generadas por otras fuentes en el entorno. Los sensores activos, como un sónar, emiten energía dentro del entorno. Se basa en el hecho de que esta energía será reflejada y devuelta al sensor. Los sensores activos suelen proporcionar más información que los pasivos, pero a expensas de un mayor consumo y con el peligro de causar interferencias cuando se usan múltiples sensores al mismo tiempo. Sean activos o pasivos los sensores se pueden dividir en tres tipos, dependiendo de

si determina distancias a objetos, imágenes del entorno o propiedades del agente en sí mismo [Russell y col., 2011].

En ambientes inteligentes el rol de una red de sensores es proporcionar a los niveles superiores del sistema respuestas a las siguientes preguntas:

- Quién: Seguimiento e identificación de personas y animales domésticos, es decir, los actores del entorno.
- Dónde y Cuándo: Proporcionar un marco temporal para las asociaciones de ubicación y objetos para determinar el contexto.
- Qué: Reconocer las actividades, las interacciones, relaciones espacio-temporales, pero también mensajes, señales y signos lingüísticos y no lingüísticos.
- Por qué: Asociación de acciones con la semántica de la acción, los guiones y los planes, la identificación de tareas y patrones de comportamiento.
- Cómo: Rastreo del flujo de información flujo de información a través de múltiples modalidades, reconocimiento de expresiones movimientos, gestos.

[Pauwels y col., 2007].

2.1.8. Actuadores

Un actuador es un dispositivo inherentemente mecánico cuya función es proporcionar fuerza para mover otro dispositivo mecánico. La fuerza que provoca el actuador proviene de tres fuentes posibles: Presión neumática, presión hidráulica, y fuerza motriz eléctrica (motor eléctrico o solenoide). Dependiendo de el origen de la fuerza el actuador se denomina “neumático”, “hidráulico” o “eléctrico” [C, 2020].

2.1.9. Automatización

La automatización es el conjunto de elementos o procesos informáticos, mecánicos y electromecánicos que operan con mínima o nula intervención del ser humano. Normalmente se

utilizan para optimizar y mejorar el funcionamiento de un proceso que requiera la intervención del ser humano como puede ser en una fábrica o palta industrial donde el ensamblado de algún producto lo hace una máquina, como también puede ser la automatización de una ciudad inteligente [Logicbus, 2020].

2.1.10. Conectividad

Para la comunicación o conectividad del *smart campus* se tienen diversas opciones, estas mismas tienen el nombre de radiofrecuencia que se trata de una trasmisión de información inalámbrica que permite mayor flexibilidad y movilidad sin necesidad de modificaciones estructurales de la vivienda o la instalación. En este tipo de medio se encuentran las señales Bluetooth, ZigBee o Wifi que utilizan una banda de emisión libre a 2.4 GHz.

- Bluetooth: fue una tecnología de transmisión de radio que surgió para mejorar las carencias existentes en las redes infrarrojas que comenzaban a instalarse. Ahora comienza a fraguarse el concepto de PAN (Personal Area Network), basada en esta tecnología de comunicación, en la que varios dispositivos pueden conectarse formando una pequeña microrred. Soporta tasas de transferencia de haza 720 Kbps en distancias de hasta 10 metros.

- ZigBee: es una tecnología que permite transmisión de datos a baja velocidad entre 20 Kbps y 250 Kbps en distancias de entre 10 y 75 metros. Se trata del soporte más barato jamás creado, con una expansión de hasta 255 elementos conectados. Posee un consumo bajísimo debido a que cuando no se está usando el transmisor/receptor (transceiver) se desactiva.

- Wifi: es un tipo de red WLAN (Wireless Local Area Network), basada en el estándar y tecnología IEEE 802.11, que se utiliza en pequeñas áreas como alternativa a la LAN cableada. Existen varios estándares de esta que difieren en las tasas de transferencia, desde 11 Mbps a 100 Mbps; las bandas de radiofrecuencia de emisión 2,4 GHz (hasta tecnología 3G) o 5 GHz (tecnologías 4G y superiores); y los usos adecuados debido a la calidad de servicio.

[Carlos, 2020].

2.2. Socket

Socket designa un concepto abstracto por el cual dos procesos (posiblemente situados en dispositivos distintos) pueden intercambiar cualquier flujo de datos, generalmente de manera fiable y ordenada. El término socket también es usado como el nombre de una interfaz de programación de aplicaciones (API) para la familia de protocolos de Internet TCP/IP, provista usualmente por el sistema operativo. Estos mismos constituyen el mecanismo para la entrega de paquetes de datos provenientes de la tarjeta de red a los procesos o hilos apropiados. Un socket queda definido por un par de direcciones IP local y remota, un protocolo de transporte y un par de números de puerto local y remoto.

Para que dos procesos puedan comunicarse entre sí es necesario que se cumplan ciertos requisitos, como el que un proceso sea capaz de localizar al otro, que ambos procesos sean capaces de intercambiarse cualquier secuencia de octetos, es decir, datos relevantes a su finalidad, para ello son necesarios los dos recursos que originan el concepto de socket, un par de direcciones del protocolo de red (dirección IP, si se utiliza el protocolo TCP/IP), que identifican la computadora de origen y la remota, un par de números de puerto, que identifican a un programa dentro de cada computadora. Los sockets permiten implementar una arquitectura cliente-servidor. La comunicación debe ser iniciada por uno de los procesos que se denomina programa ”cliente”. El segundo proceso espera a que otro inicie la comunicación, por este motivo se denomina programa ”servidor”. Un socket es un proceso o hilo existente en la máquina cliente y en la máquina servidora, que sirve en última instancia para que el programa servidor y el cliente lean y escriban la información. Esta información será transmitida por las diferentes capas de red [Gilbert y col., 2012].

2.2.1. Socket.IO

Socket.IO es una biblioteca que permite la comunicación de baja latencia en tiempo real, bidireccional y basada en eventos entre un cliente y un servidor. Basado en WebSocket y proporciona garantías adicionales, como el respaldo al sondeo largo HTTP o la reconexión automática. Donde WebSocket es un protocolo de comunicación que proporciona un canal full-duplex y de baja

latencia entre el servidor y el navegador. Existen varias implementaciones de clientes en otros lenguajes de programación como JavaScript, Java, C++, Swift, Dart, Python, .Net, Rust y Kotlin. que dia a dia son mejorados y mantenidos por la comunidad.

Algunas de las características de Socket.IO son que, si un cliente intentará establecer una conexión WebSocket y esta no pueda establecerse recurrirá a un sondeo largo HTTP, esta es una característica sumamente importante a la decisión de elegir el framework de los socket dado que este todo el tiempo intentara establecer la conexión, otra característica importante es que la mayor parte de los navegadores soportan WebSockets (más del 97%), por otra parte en algunas condiciones particulares, la conexión WebSocket entre el servidor y el cliente puede interrumpirse sin que ambas partes sean conscientes del estado roto del enlace. Socket.IO incluye un mecanismo que comprueba periódicamente el estado de la conexión y cuando el cliente se desconecta, se reconecta automáticamente con un retardo exponencial, para no saturar al servidor [Socket.IO, 2014].

2.3. Dearrollo de software

La palabra ”software” se refiere al equipamiento lógico o soportelógico de una computadora, y comprende el conjunto de los componentes lógicos necesarios para llevar a cabo una funcion de una tarea específica, en contraposición a los componentes físicos del sistema(hardware) [Díaz, 2014]. El software, en su gran mayoría, está escrito en lenguajes de programación de alto nivel, ya que son más fáciles y eficientes para que los programadores los usen, porque son más cercanos al lenguaje natural respecto del lenguaje de máquina. Los lenguajes de alto nivel se traducen a lenguaje de máquina utilizando un compilador o un intérprete, o bien una combinación de ambos. El software también puede estar escrito en lenguaje ensamblador, que es de bajo nivel y tiene una alta correspondencia con las instrucciones de lenguaje máquina; se traduce al lenguaje de la máquina utilizando un ensamblador. Por lo tanto el desarrollo de software no es mas que una creación de código fuente que al ser ejecutado satisface la necesidad de una función para un objetivo en específico.

2.3.1. Front-End

El desarrollo web Front-end consiste en la conversión de datos en una interfaz gráfica para que el usuario pueda ver e interactuar con la información de forma digital usando paginas web o aplicaiones moviles [Codesido, 2009]. Principalmente una interfaz gracia está compuesta de HTML, CSS y JavaScript, con el paso de los años han surgido herramientas como los Frameworks para que el diseño de estas mismas sea más optimas y amigables con el usuario final.

2.3.2. Back-End

El desarrollo back-end consiste en la combinación de una base de datos y un software escrito en un lenguaje del lado del servidor". El back-end incluye servidores y bases de datos. Los servidores controlan el acceso de los usuarios a los archivos. Las bases de datos son colecciones de datos organizadas y estructuradas [Codecademy, 2021].

2.3.3. Framework

Un Framework (entorno de trabajo o marco de trabajo) es un conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular que sirve como referencia, para enfrentar y resolver nuevos problemas de índole similar. En el desarrollo de software, un entorno de trabajo es una estructura conceptual y tecnológica de asistencia definida, normalmente, con artefactos o módulos concretos de software, que puede servir de base para la organización y desarrollo de software. Típicamente, puede incluir soporte de programas, bibliotecas, y un lenguaje interpretado, entre otras herramientas, para así ayudar a desarrollar y unir los diferentes componentes de un proyecto. Representa una arquitectura de software que modela las relaciones generales de las entidades del dominio, y provee una estructura y una especial metodología de trabajo, la cual extiende o utiliza las aplicaciones del dominio [Riehle, 2000]. Gracias a los esto en la actualidad se pueden agilizar los procesos del desarrollo de un software y dar mejores resultados.

2.4. Bases de Datos

Una base de datos es una recopilación organizada de información o datos estructurados, que normalmente se almacena de forma electrónica en un sistema informático. Normalmente, una base de datos está controlada por un sistema de gestión de bases de datos (DBMS). En conjunto, los datos y el DBMS, junto con las aplicaciones asociadas a ellos, reciben el nombre de sistema de bases de datos, abreviado normalmente a simplemente base de datos. Los datos de los tipos más comunes de bases de datos en funcionamiento actualmente se suelen utilizar como estructuras de filas y columnas en una serie de tablas para aumentar la eficacia del procesamiento y la consulta de datos. Así, se puede acceder, gestionar, modificar, actualizar, controlar y organizar fácilmente los datos. La mayoría de las bases de datos utilizan un lenguaje de consulta estructurada (SQL) para escribir y consultar datos.[Oracle, 2020]

2.4.1. Relacional

Las bases de datos se hicieron predominantes en la década de 1980. Los elementos de una base de datos relacional se organizan como un conjunto de tablas con columnas y filas. La tecnología de bases de datos relacionales proporciona la forma más eficiente y flexible de acceder a información estructurada. Una base de datos relacional es un tipo de base de datos que almacena y proporciona acceso a puntos de datos relacionados entre sí. Las bases de datos relacionales se basan en el modelo relacional, una forma intuitiva y directa de representar datos en tablas. En una base de datos relacional, cada fila de la tabla es un registro con un ID único llamado clave. Las columnas de la tabla contienen atributos de los datos, y cada registro generalmente tiene un valor para cada atributo, lo que facilita el establecimiento de las relaciones entre los puntos de datos [Oracle, 2020].

2.4.2. No Relacional

Una base de datos NoSQL, o base de datos no relacional, permite almacenar y manipular datos no estructurados y semiestructurados. Las bases de datos NoSQL se hicieron populares a medida que las aplicaciones web se volvían más comunes y complejas [Oracle, 2020]. Es decir es

aquella que no usa el esquema tabular de filas y columnas que se encuentra en la mayoría de los sistemas de base de datos más tradicionales. En su lugar, las bases de datos no relacionales usan un modelo de almacenamiento que está optimizado para los requisitos específicos del tipo de datos que se almacena. Por ejemplo, los datos se pueden almacenar como pares clave/valor simple, como documentos JSON o como un grafo que consta de bordes y vértices [Microsoft, 2020].

Capítulo 3

Estado del Arte

En el Capítulo 2 se definieron algunos temas, conceptos y tecnologías, para el desarrollo del proyecto, en este capítulo se mostrarán algunos trabajos que tienen similitud con el proyecto a desarrollar. Ya sea que el tema es el mismo, pero con otra técnica o temas que se pueden usar para el desarrollo del proyecto.

3.1. Smart Cities, Smart Campus, Smart House

En el artículo "Propuesta de una arquitectura para un Smart Campus Universitario" se describe una arquitectura con el uso de agentes inteligentes conectados a una aplicación móvil que a su vez esta conectada con un sistema interno de la institución con el objetivo o planteamiento de recolección de datos, análisis de información y toma de decisiones automatizadas [Cruz Parada y col., 2020].

En el trabajo "Smart CEI Moncloa: An IoT-based Platform for People Flow and Environmental Monitoring on a Smart University Campus" se presenta una plataforma basada en IoT para la vigilancia del flujo de personas y del medio ambiente en un Campus Universitario Inteligente desplegada en el campus del CEI de Moncloa, haciendo especial hincapié en los principales retos tecnológicos que se han afrontado y en las soluciones que se han adoptado, así como en la funcionalidad, los servicios y el potencial que ofrece la plataforma [Alvarez Campana y col., 2017].

En el trabajo "The Campus as a Smart City: University of Málaga Environmental, Learning, and Research Approaches" plantean el ver un campus universitario como una ciudad inteligente en el cual recopilan datos ambientales y su procesamiento para lograr gestionar las zonas del instituto. Se apoyan en la aplicación masiva de las tecnologías de la información y la comunicación (TICs) y el paradigma de la Internet de las cosas (IoT), en el que un gran número de dispositivos distribuidos se conectan para transferir los datos recogidos [Fortes y col., 2019].

En el trabajo "Design and Experimental Validation of a LoRaWAN Fog Computing Based Architecture for IoT Enabled Smart Campus Applications" nos muestran una arquitectura en la que proponen LoRaWAN para la conectividad de un campus inteligente y tener todos los nodos IoT conectados sin que el costo sea muy elevado ya que el consumo de energía es bajo [Fraga Lamas y col., 2019].

El objetivo del estudio "A Multi-Agent-Based Intelligent Sensor and Actuator Network Design for Smart House and Home Automation" es desarrollar un Framework basado en sistemas multiagente con un conjunto de herramientas de diseño una casa inteligente y aplicaciones de automatización del hogar, de las cuales son: diseñar y controlar el comportamiento de los agentes individual mente basados en un modelo de creencias, deseos e intenciones; diseñar y controlar el comportamiento de los grupos de agentes basados en una política de regulación; y evaluar el rendimiento del sistema y optimizar los parámetros de diseño en base a un conjunto de métricas. Por último, se enfocan en cuatro puntos de interacción y colaboración entre los agentes: la detección, la acción, la decisión y la base de datos [Sun y col., 2013].

En el artículo '' A Survey on Internet of Things Enabled Smart Campus Applications" se nos hablan de las bases con las que tiene que contar un *Smart campus* con el fin de establecer una infraestructura para que la aplicación cree y ofrezca servicios de valor añadido mediante la detección cooperativa de entidades del entorno, es decir, seres humanos, espacios, máquinas, etc. Este mismo articulo menciona dos aspectos importantes en los que se basa este proyecto, mejorar la calidad de las instalaciones interiores (iluminación, confort térmico, niveles de ruido, ventilación natural, etc.) y mejorar o disminuir el uso de la energía [Abuarqoub y col., 2017].

En el artículo 'Smart Home Monitoring System Using ESP32 Microcontrollers'' nos comparten como utilizando un ESP32 pueden monitorear, las salas de estar a través de cámaras, las

temperaturas de la calefacción o el boiler, bloqueo de seguridad con contraseña entre otras cosas [Babiuch y Postulka, 2021]. Las cuales posterior mente uno puede introducir inteligencia artificial y el monitoreo pueda ser automático, generar alertas, etc.

3.2. Ambientes Inteligentes

En el artículo "Ambient Intelligence—The Next Step for Artificial Intelligence" [Ramos y col., 2008] se nos explica como debe ser un ambiente inteligente, que en la actualidad se crean ambientes con sensores y actuadores para interactuar con el ambiente, pero eso no lo convierte en un ambiente inteligente. Es necesario incluir alguna técnica de inteligencia artificial que le permita aprender y tomar propias decisiones. Mencionan algunas técnicas que puede incluir un ambiente inteligente como representación del conocimiento, lógica, ontologías, reconocimiento del habla, procesamiento del lenguaje natural, recuperación de la información, minería de textos, sistemas expertos, aprendizaje automático, planificación, inteligencia computacional, incompletitud e incertidumbre, visión por ordenador, optimización, sistemas de apoyo a la decisión, sistemas multiagente, tutoría inteligente, emoción, argumentación, etc

En el artículo "Designing a Multiagent System for Elderly Care" [Martínez y col., 2020] se desarrolla una estrategia para implementar un sistema multiagente que podrá ayudar a personas vulnerables o con capacidades diferentes, utilizando un Raspberry PI. Por otro lado, el proyecto está diseñado para la automatización dentro de una casa con el fin de lograr mejoras en la calidad de vida de las personas, que residirán en estas casas. Estas mejoras se realizan mediante la adición de servicios multiagente a la casa, para ello, estudian los 4 grandes grupos en los que se agrupan los servicios domóticos los cuales son: ahorro de energía, confort, seguridad y comunicaciones.

En el instituto ''Institut Teknologi Sumatera" tienen dos trabajos enfocados a la disminución y monitoreo de la energía eléctrica utilizando un ESP8266 en el cual montan un interruptor para apagar la luz en caso de que la habitación este vacía lo cual monitorean con un sensor en la entrada de la puerta para contar el numero de personas dentro, usan dos actuadores más, uno para controlar las cortinas y permitir el paso de luz, y otro para el encendido y apagado del aire

acondicionado [Yuliansyah y col., 2019a]. En el segundo trabajo implementan medidores que monitorean el consumo de energía las 24 horas del día, dichos monitores mandan la información al ESP8266 que posterior mente se almacena en la nube [Yuliansyah y col., 2019b].

En el artículo "Living Assistance Systems" [Nehmer y col., 2006] monitorean a los habitantes de una vivienda con el objetivo de mejorar la calidad de vida en la vivienda enfocándose en tres servicios tratamiento de emergencias, mejora de la autonomía, Confort. Cuentan que hay alternativas de seguimiento con GPS pero no son tan precisos, en este caso se utilizan cámaras y micrófonos. Para poder ver si una persona se levanta, puede caminar, si se queda sentado, si se está riendo con el objetivo de si se detecta alguna anormalidad del comportamiento poder actuar.

En el artículo ''A Fuzzy Logic Based System for Indoor Localisation using WiFi in Ambient Intelligent Environments" [Garcia-Valverde y col., 2013] a diferencia del articulo anterior que usan cámaras para determinar la ubicación de los habitantes en este caso para no invadir la privacidad de los avitantes se decantan por usar hardware distinto que son emisores y antenas de ultrasonido y radiofrecuencia. Para llevar a cabo la detección de habitantes utilizan lógica difusa empleando señales WiFi.

3.3. Dispositivos móviles

En el trabajo "Ubicación de dispositivos móviles en ambientes interiores por medio de análisis de radiación de redes WiFi y deformaciones de campo magnético" [Gómez R. y Pedraza, 2018] presenta un modelo de uso de la radiación de las redes WiFi y la deformación del campo magnético terrestre en estructuras cerradas en interiores como fuente de información del contexto de la ubicación de un dispositivo móvil, donde todas las labores de captura, procesamiento y análisis de datos sean hechas por un teléfono inteligente, sin la necesidad de asistencia externa y con la posibilidad futura de crear un sistema de navegación autónoma en interiores.

En el artículo ''Smart campus student management system based on 5G network and Internet of Things" [Liu, 2020] se nos plantea una manera de administrar un *Smart campus* en 4 capas, capa de aplicación de dominio, capa de procesamiento de datos, capa de gestión de datos, capa

ciberfísica. En el proyecto comparan redes telefónicas (2G, 3G, 4G y 5G) para los dispositivos Iot de las cuales la 5G tiene mejor desempeño.

En articulo '' A Study of MAC Address Randomization in Mobile Devices and When it Fails" [Martin y col., 2017] se muestran los resultados de un estudio a gran escala en el mundo sobre cual es la función de ocultar la dirección MAC (Media Access Control) para el rastreo de dispositivos, explica cuáles son la importancia de la privacidad de los datos y el mal uso que se le dio antes de ser regulado por los fabricantes de dichos dispositivos móviles. Por lo tanto, se llegan a generar alternativas para el rastreo de dispositivos móviles siempre y cuando el usuario de consentimiento.

En el artículo ''Performance Comparison of IoT Communication Protocols" [Moraes y col., 2019] se nos dice que la cantidad de sistemas basados en el Internet de las Cosas (IoT) ha crecido a un ritmo sin precedentes en los últimos años y dicha expansión tiende a continuar. Como consecuencia, se espera que se desplieguen miles de millones de dispositivos en diversos sectores (por ejemplo, sanidad, automoción) durante la próxima década. Debido a su heterogeneidad, la comunicación de los dispositivos IoT es una función destacada del sistema y, por tanto, se han propuesto distintos protocolos de comunicación para estos sistemas. Este artículo presenta una evaluación del rendimiento de los protocolos de comunicación IoT para la capa de aplicación: AMQP, CoAP y MQTT. El rendimiento, el tamaño de los mensajes y la pérdida de paquetes son las métricas adoptadas y los experimentos indican que el protocolo CoAP proporciona los mejores resultados.

3.4. Agentes inteligentes

En el trabajo "Use of Intelligent Agent Through Low-Cost Brain–Computer Interface to Analyze Attention and Meditation Levels by Gender" [Serna y col., 2019] se explica el diseño y desarrollo de un experimento para recolectar información sobre la atención y la meditación en diferentes personas, con el fin de analizar datos obtenidos a sujetos de prueba y comenzar a visualizar características de la población utilizando un agente inteligente.

En el artículo ''Using the ESP32 Microcontroller for Data Processing'' [Babiuch y col., 2019]

se muestran experiencias con el desarrollo de aplicaciones de los microcontroladores ESP32 y proporciona una revisión completa de las posibilidades de desarrollo de aplicaciones en esta plataforma en el área de medición y medición y procesamiento de datos. Los microcontroladores suelen conectarse con módulos IoT y otros sensores inteligentes y proporcionan datos al sistema superior. El ESP32 tiene tres bases fuertes de desarrollo las cuales son Arduino Core, Espressif IoT Development Framework y MicroPython. Donde podemos procesar datos y recopilarlos para ser enviados a un agente.

3.5. Conclusiones

Una vez que se realizaron las investigación de los trabajos relacionados llegamos a la conclusión de juntar las ideas de los diferentes trabajos y crear un sistema multiagente enfocado al *Smart campus* con sensores que sean independientes al agente en el cual el agente no de penda de los sensores en caso de falla el agente pueda seguir trabajando y poder monitorear el ambiente todo el tiempo, por otro lado para la detección de dispositivos móviles se optara por red bluetooth de bajo consumo para los agentes como para los dispositivos móviles, por ultimo se optara por un ESP32 para la recolección de datos del entorno y se los mandara a la Raspberry Pi para el procesamiento de los datos con lógica difusa donde esta regresara datos a la ESP32 para tomar acciones en el entorno.

Capítulo 4

Diseño e implementación

En este capítulo se hablará de cómo será el diseño y la implementación, dado que con anterioridad el instituto Tecnológico de México en León ya contaba con una infraestructura de desarrollo, donde se utilizaron algunos conceptos mencionados anterior mente, como los lenguajes de programación entre ellos PHP, Java, JavaScript, TypeScript; el tipo de base de datos en este caso se utilizó SQL que vimos que son tablas relacionadas, etc. Por lo tanto, se optó por partir de esa base para hacer que el sistema multiagente sea funcional con las bases de datos con las que ya se cuentan como para la funcionalidad de que un usuario pueda ver sus horarios de clase. Por otro lado, ya se contaba con un previo trabajo el cual era una propuesta de arquitectura para un *Smart campus*, donde se pretende hacer uso de agentes, para poder mandar mensajes entre agentes y usuarios. Tales como notificaciones o avisos importantes del Tecnológico. Utilizando esta arquitectura propuesta solo nos quedaría la integración de sensores para la manipulación de lámparas y el seguimiento de dispositivos móviles.

4.1. Análisis

Partiendo que ya tenemos la arquitectura como se ve en en la figura 4.1, dado que en esa arquitectura solo se ocupa el Raspberry como agente, solo nos queda hacer la integración de sensores como se aprecia en la figura 4.2.

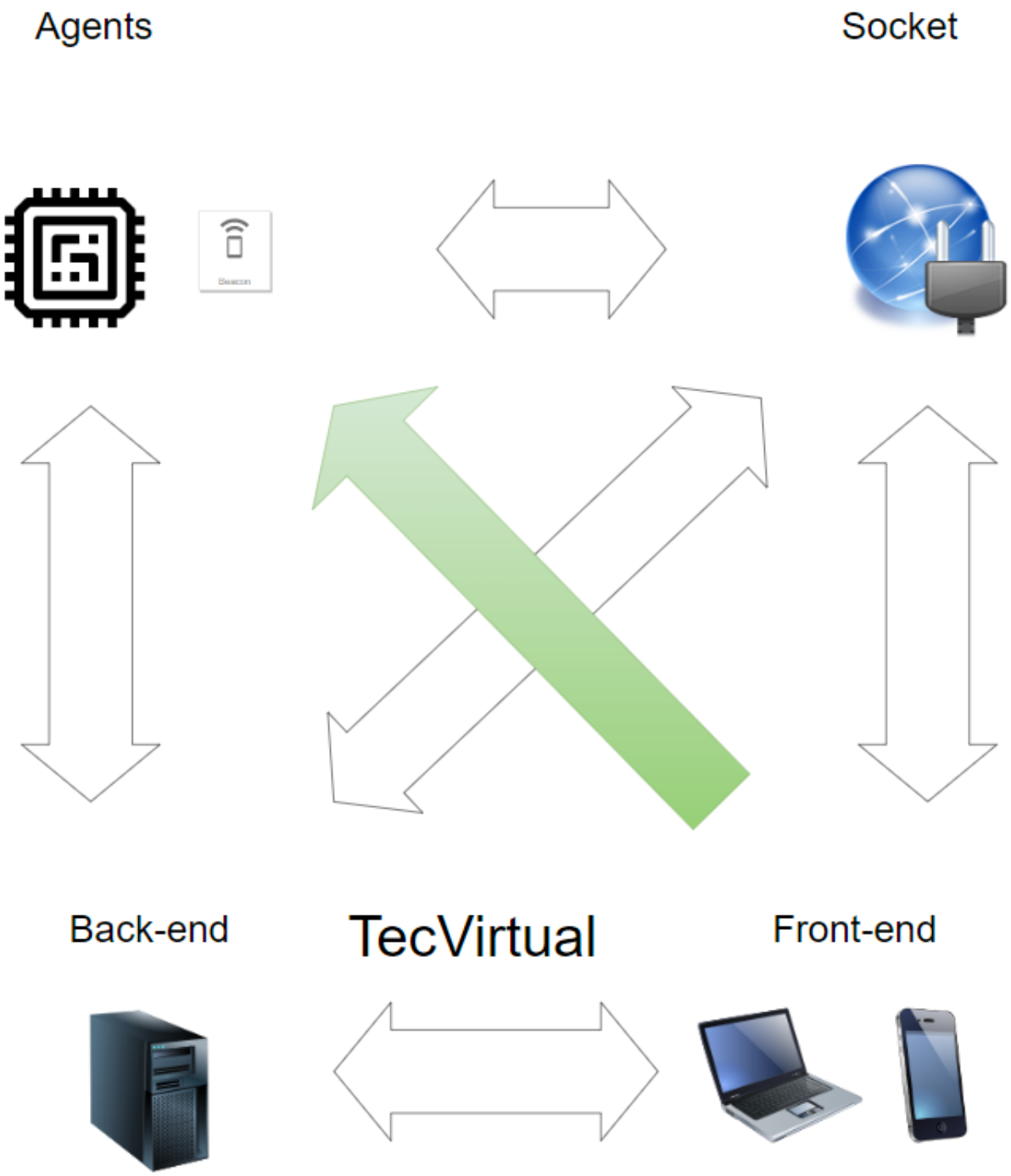


FIGURA 4.1: Propuesta de arquitectura para el campus inteligente (Cruz Parada y col., 2020).

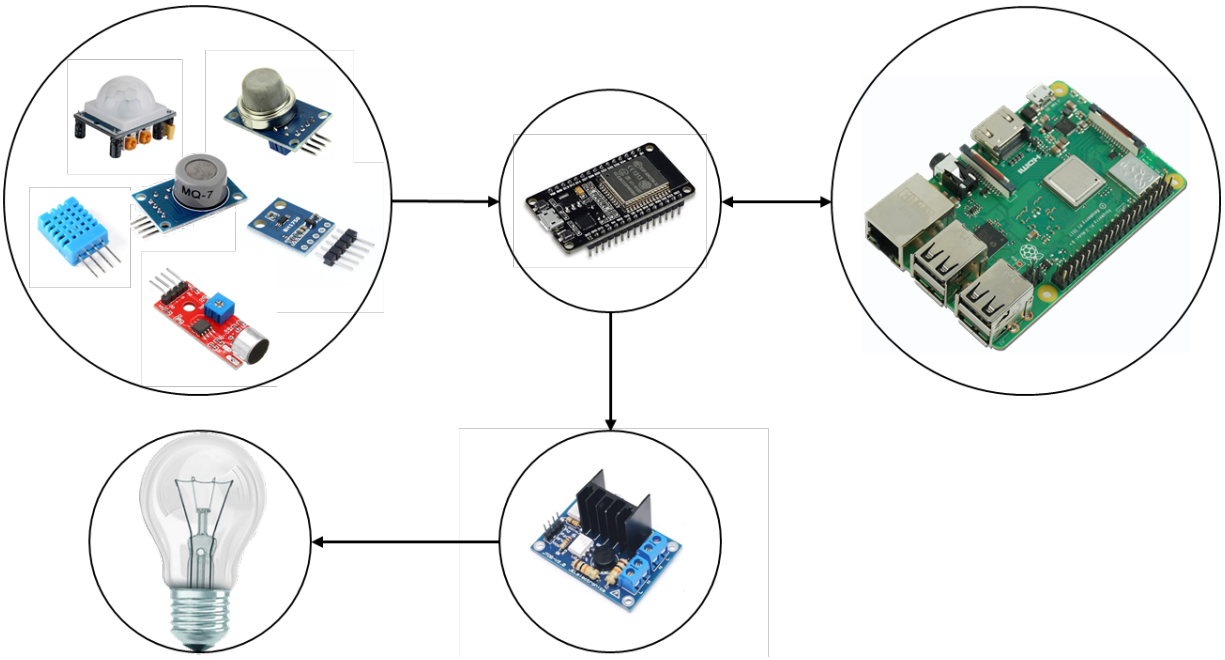


FIGURA 4.2: Estructura del agente(elaboración propia).

4.2. Tecnologías a utilizar

Dado que ya se tenían unas bases de tecnología en el instituto se mantendrán las mismas y se le incluirán nuevas como los sockets, agentes, sensores, actuadores. Para los sockets se utilizará Socket.IO en el lenguaje JavaScript para la parte del servidor, para la parte del agente será en Python y para los usuarios finales es decir los dispositivos móviles se utilizará TypeScript. Para los agentes estos serán una Raspberry al cual se le instalar el lenguaje de programación Python que utilizaremos con el Framework PADE. Para el uso de los sensores y actuadores se utilizará un ESP32 con base en Arduino que obtendrá los datos del ambiente con los sensores y los datos serán enviados a la Raspberry. La Raspberry por medio de lógica difusa calcula la intensidad que requiere y le manda los valores de la intensidad a la ESP32 que posteriormente la ESP32 manda al actuador que manipula la energía. Los tipos de sensores a utilizar son: un sensor presencia o de movimiento (PIR), un sensor de luz (BH1750), un sensor de ruido (KY-037), un sensor de temperatura y humedad (DHT11), un sensor de gas (Sensor MQ-7) para medir el monóxido de carbono (CO) y un sensor de calidad del aire (MQ-135) para medir dióxido de carbono (CO_2). Se utilizara un módulo dimmer que funciona con un triac el cual su función es detectar el cruce por cero de la corriente alterna de esta manera se puede manipular la intensidad de luz de una

lampara. Para almacenar los datos de los agentes se utilizará MongoDB que es una base de datos no relacional. Para almacenar los datos de los agentes se utilizará MongoDB que es una base de datos no relacional.

Para la lógica difusa utilizaremos las siguientes reglas:

- Si Luz es Alta entonces Corriente es Baja
- Si Luz es Baja entonces Corriente es Alta
- Si Luz es Media entonces Corriente es Media
- Si Hora es Mañana and Luz es Alta entonces Corriente es Media
- Si Hora es Tarde and Luz es Baja entonces Corriente es Baja
- Si Hora es Noche and Luz es Alta entonces Corriente es Media

De entradas tendremos luz de 0 a 30:

- Baja de 0,0,10
- Media de 5,15,25
- Alta de 20,30,30

De entradas tendremos horas de 0 a 24:

- Mañana de 0,0,10
- Tarde de 8,12,16
- Noche de 14,24,24

Salida de corriente de 20 a 80:

- Baja de 20,30,40
- Media de 40,50,60
- Alta de 60,70,80

Podemos ver las entradas y salida en forma de grafica en las figuras 4.3 , 4.4 y 4.5.

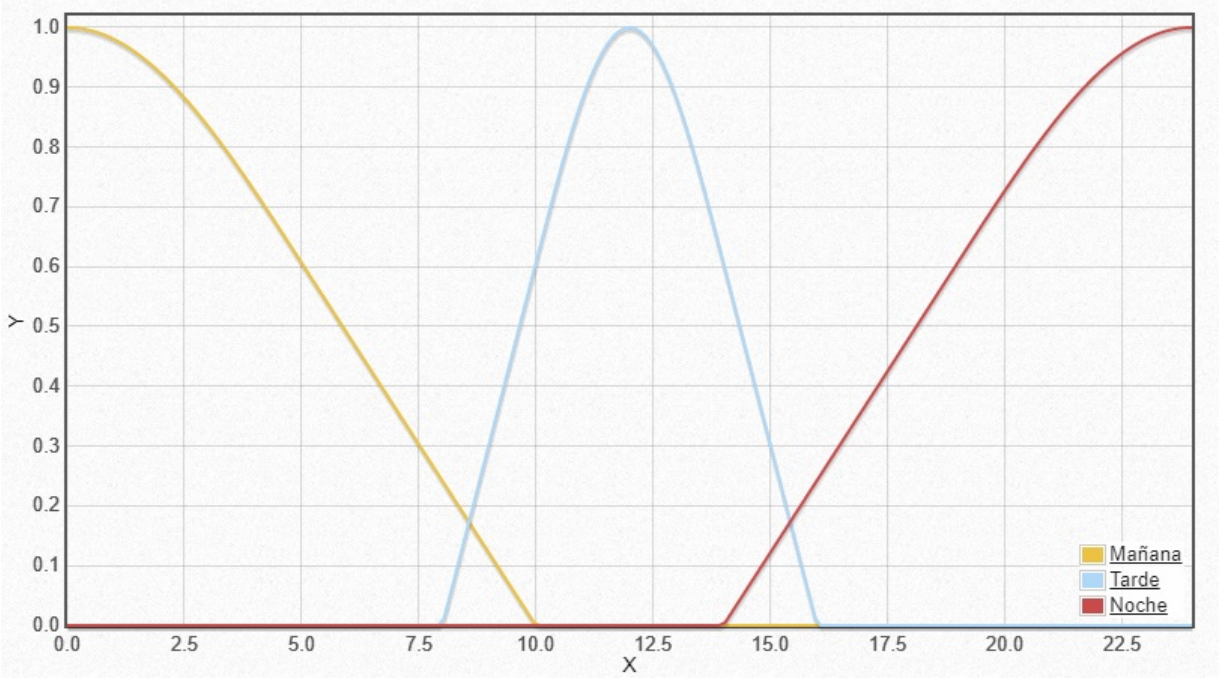


FIGURA 4.3: Entradas de hora (elaboración propia [JuzzyOnline, http://ritweb.cloudapp.net:8080/JuzzyOnline/]).

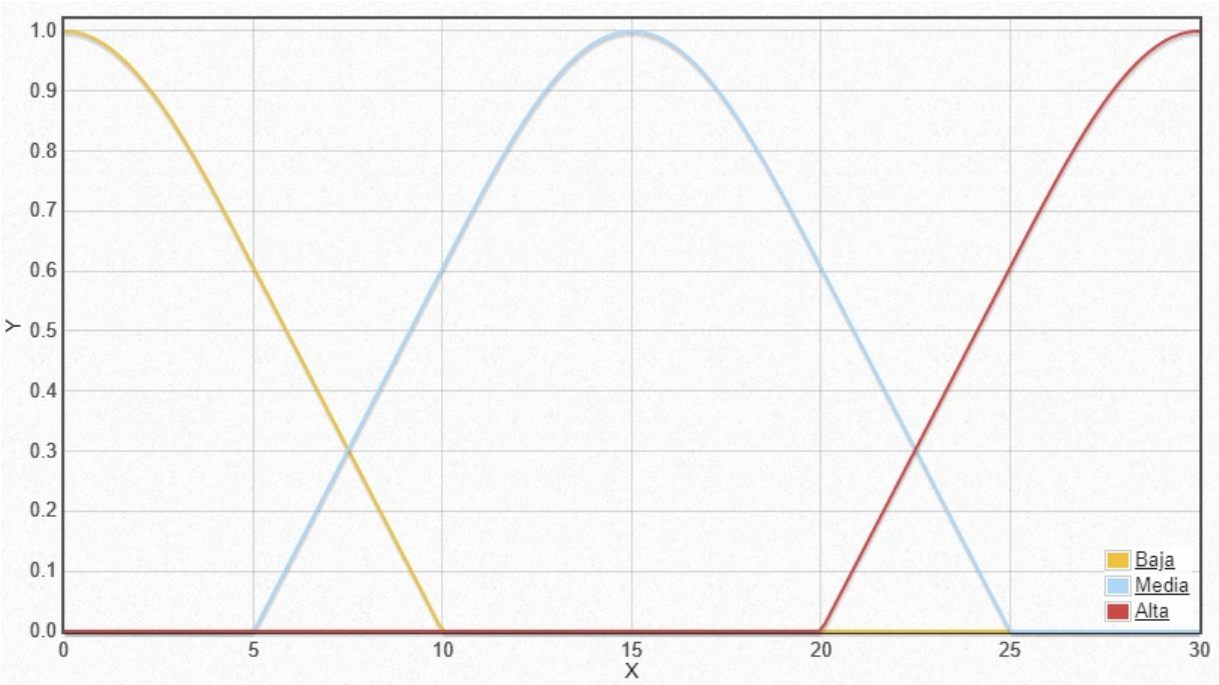


FIGURA 4.4: Entradas de luz (elaboración propia [JuzzyOnline, http://ritweb.cloudapp.net:8080/JuzzyOnline/]).

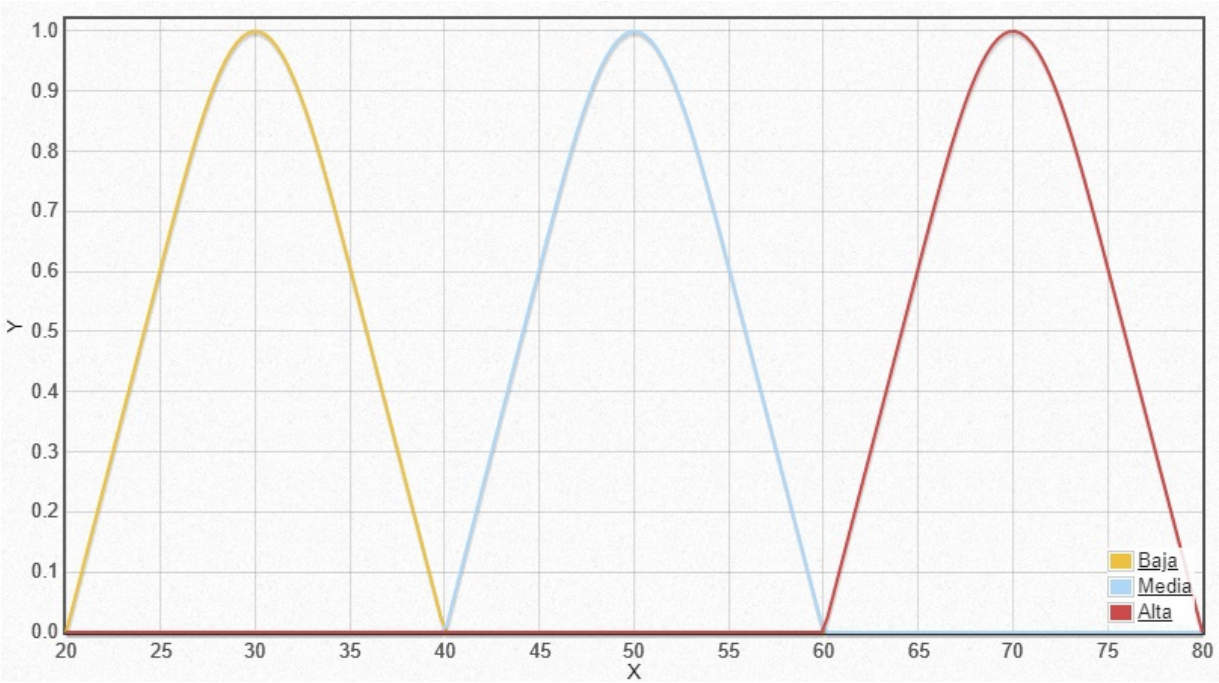


FIGURA 4.5: Salida de corriente (elaboración propia [JuzzyOnline, http://ritweb.cloudapp.net:8080/JuzzyOnline/]).

4.3. Prueba de los sensores y actuadores

Una vez montado el agente verificamos que el ESP32 al mandarle la palabra ''SENSORS'' nos retorne los valores de los sensores que en este caso los primeros dos son CO y CO_2, los otros cuatro temperaturas y los últimos dos los índices de calor, dado que estaba montado en una protoboard tendía a tener fallos como se ve en la figura 4.6. En tal caso de que el sensor PIR detectara a alguien le mandaría un mensaje con la intensidad calculada que arroja la lógica difusa entre 20 a 80 que le mandara al dimmer para encender la lampara, y en el caso contrario manda la palabra ''OFF'' para desactivar el dimmer. En las figuras 4.7 , 4.8 y 4.9 podemos ver las primeras pruebas de montaje.

FIGURA 4.6: Monitor de Arduino con los valores de los sensores (elaboración propia).

4.4. Prueba de comunicación entre las tecnologías selecionadas

Se inicia el servicio de socket del lado del servidor, y posterior mente encendemos el agente para ver si hay conectividad entre el cliente socket y el servidor. A al iniciar el agente el socket lo detecta y lo guarda en un arreglo de agentes como podemos ver en las figuras 4.11 y 4.12. Y por el lado del usuario es igual, al iniciar sesión en la ampliación se conecta al socket como se ve en las figuras 4.10 y 4.12

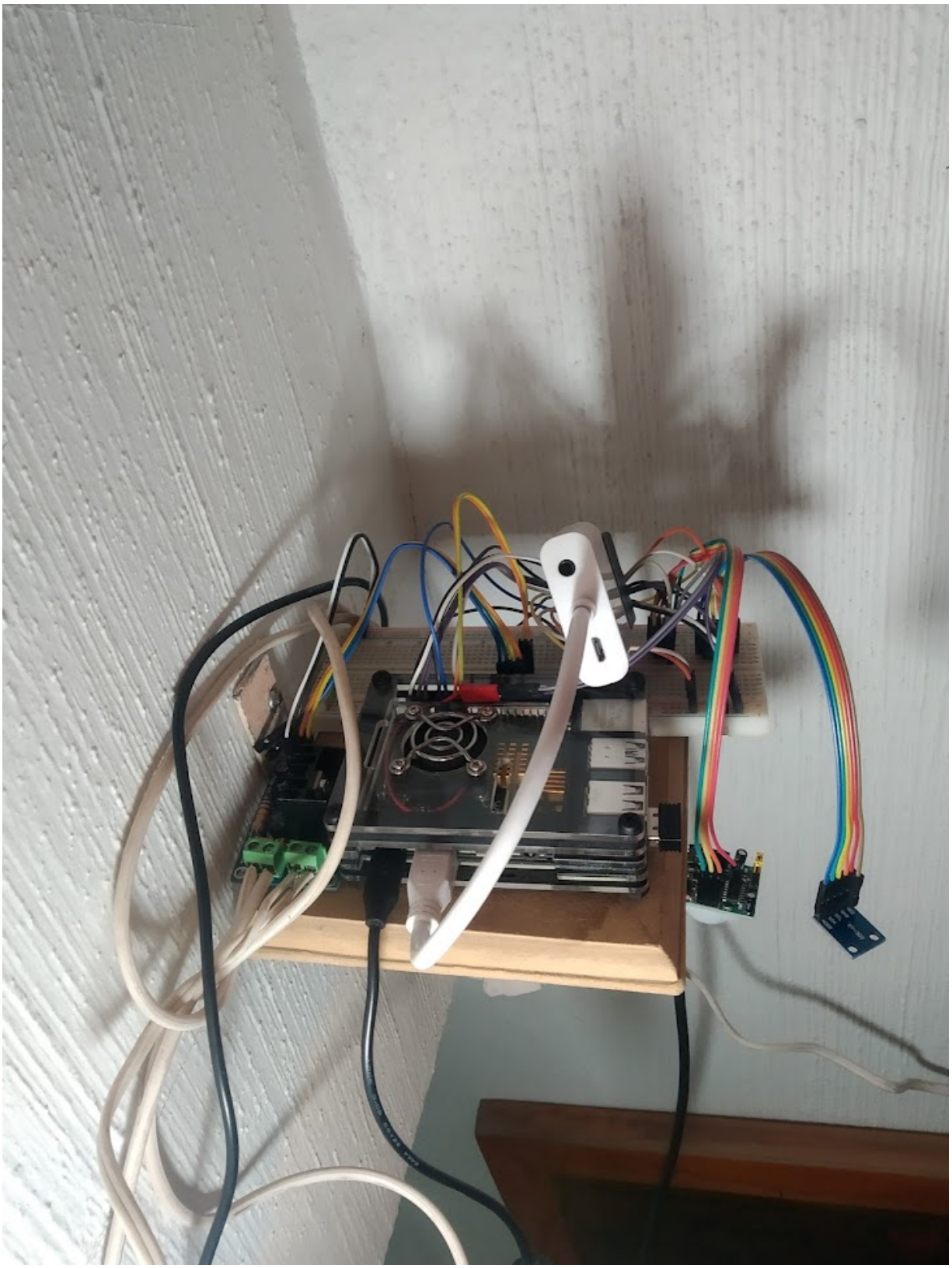


FIGURA 4.7: Prototipo 1 (elaboración propia).

FIGURA 4.8: Prototipo 1 encendido (elaboración propia).

FIGURA 4.9: Prototipo 1 apagado (elaboración propia).

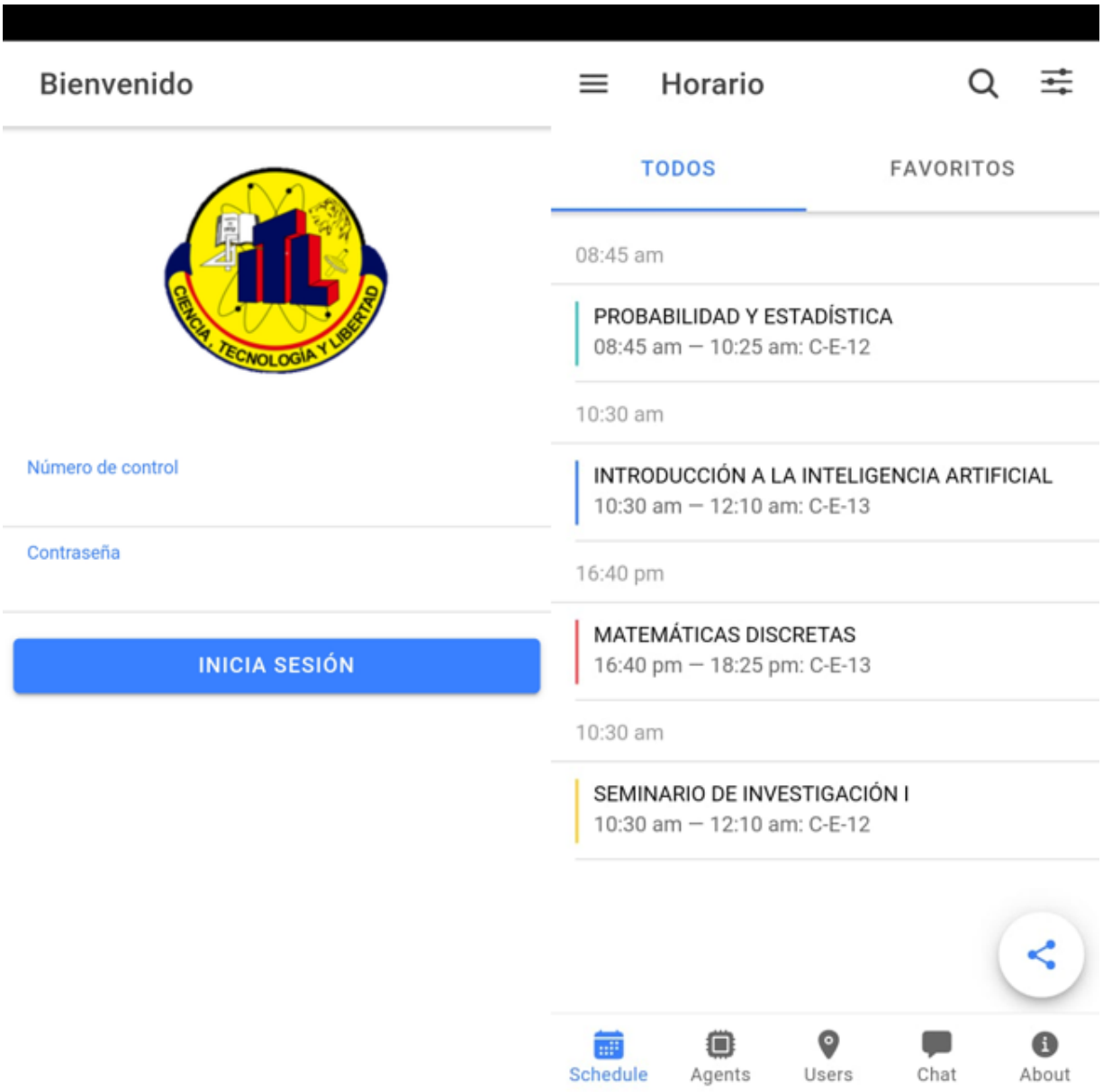


FIGURA 4.10: Pantalla de inicio y horario escolar (elaboración propia).

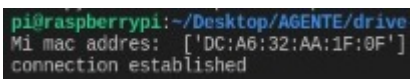


FIGURA 4.11: Agente conectado con el servidor de socket (elaboración propia).

```
fabricio@LAPTOPFABRICIO:~/Escritorio/Smart Campus/SmartCampusSocket$ node socket.js
Servidor de sockets activo 5000
Agents
[
  User {
    socketId: 'IuRdRZOT5Qu-OYUkAAAB',
    id: 242606372691727,
    type: 0
  }
]
```

FIGURA 4.12: Encendido del socket y un agente conectado (elaboración propia).

4.5. Detección de dispositivos móviles

Para la detección de dispositivos móviles es necesario construir nuestra aplicación utilizando un Framework de front-end en este caso se utilizó ''ionic" que tiene una gran variedad de librerías, una de ellas que nos ayudara a la detección de los dispositivos utilizando el bluetooth del dispositivo haciendo un consumo extremadamente bajo. Dentro de la aplicación se ejecutan ciertos métodos para que el usuario cada cinco minutos inicie un escaneo de los bluetooth encendidos con el método ''startScanning", generando una lista con el método ''onDeviceDiscovered" que posterior mente compara con los agentes conectados con el método ''checkAgentList" si llega a encontrar uno manda un mensaje al socket diciéndole en que agente se encuentra como se ve en la figura 4.13, esto es una solución practica en la que el usuario puede permitir o no el dar su ubicación entre agentes. Y por el lado del socket al utilizar el método ''checkAgentList" se manda a llamar la lista de todos los agentes que están conectados como se ve en la figura 4.14.

```
startScanning() {
    this.bluetoothSerial.discoverUnpaired().then((success) => {
        success.forEach((value, key) => {
            this.onDeviceDiscovered(value);
        });
    }, (err) => { console.log(err); });
}

onDeviceDiscovered(device) {
    this.ngZone.run(() => {
        this.devices.push(device);
        this.checkAgentList(device)
    });
}

checkAgentList(device) {
    this.userData.getAgentOnlineList().then((list)=>{
        list.forEach((agentId,index)=>{
            if (agentId == device.id) {
                this.socket.here(agentId);
                return;
            }
        })
    })
}
```

FIGURA 4.13: Código fuente de la aplicación para la detección de agentes (elaboración propia).

```
startScanning() {
    this.bluetoothSerial.discoverUnpaired().then((success) => {
        success.forEach((value, key) => {
            this.onDeviceDiscovered(value);
        });
    }, (err) => { console.log(err); });
}

onDeviceDiscovered(device) {
    this.ngZone.run(() => {
        this.devices.push(device);
        this.checkAgentList(device)
    });
}

checkAgentList(device) {
    this.userData.getAgentOnlineList().then((list)=>{
        list.forEach((agentId,index)=>{
            if (agentId == device.id) {
                this.socket.here(agentId);
                return;
            }
        })
    })
}
```

FIGURA 4.14: Código fuente del socket para la obtener la lista de agentes (elaboración propia).

4.6. Almacenar los datos

Ya que tenemos los datos de los sensores y los dispositivos móviles es conveniente tener dichos datos almacenados para poder generar históricos, para eso empezaremos por el back-end que es al que le mandaremos los datos y que posteriormente serán almacenados en la base de datos no relacional, como podemos ver en la figura 4.15 el socket recibe un mensaje desde la aplicación que detecto un agente para avisar que se encuentra en él para posteriormente registrarlo en la base de datos. Y para guardar los datos de los sensores el agente manda los datos de los sensores directamente a la base de datos como se ve en la figura 4.16.

```
async function updateUserPosition(agentId, userId) {

    var client = mongoDb.MongoClient;

    await client.connect(uri, { useUnifiedTopology: true }).then((client) => {
        const database = client.db(databaseName);
        const collection = database.collection("position_user");

        collection.insertOne({
            'agent': agentId,
            'user': userId,
            'updatedAt': new Date()
        }, () => {
            client.close();
        })
    }).catch((e) => {
        console.log(e);
    });
}
```

FIGURA 4.15: Código fuente del socket para guardar la ubicación de un usuario (elaboración propia).

```python
def update_sensors(id, sensors):
    agentUpdate = False
    col = db['agent']
    query = { "_id" : id }
    values = col.find_one(query)['dates']
    if values[-1]['date']< datetime.now()-timedelta(seconds=5*60):
        values.append({
            "date": datetime.now(),
            "co2": sensors[0],
            "co": sensors[1],
            "noise": sensors[2],
            "humidity": sensors[3],
            "temperature_c": sensors[4],
            "temperature_f": sensors[5],
            "heat_index_c": sensors[6],
            "heat_index_f": sensors[7]
        })
        agentUpdate=True
    val = { "$set":  { "dates": values } }
    if col.find_one(query):
        col.update_one(query , val)
    return agentUpdate
```

FIGURA 4.16: Código fuente del agente para guardar los datos de los sensores (elaboración propia).

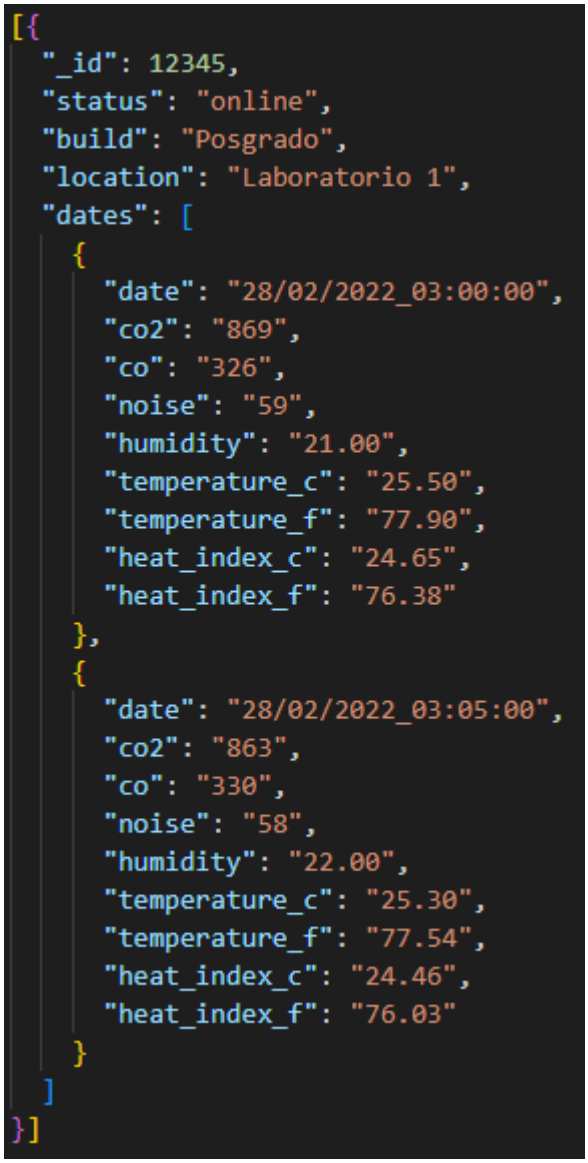
4.7. Base de datos

Una vez que sabemos que tipos de datos vamos a almacenar y dado que son datos que constante mente estarán en lectura y escritura, se opto por utilizar una base de datos no relacional ya que una de sus características principales que son utilizadas para aplicaciones en tiempo real y de rápida ejecución. Otra característica importante la forma en que ese almacenan los datos, siendo de manera semiestructurada, los que nos permite crear registros únicos sin la necesidad de crear una fila completa de datos, por ejemplo, si tenemos un agente montado con sensores de humedad, ruido y calidad del aire. Y se quisiera almacenar solo la temperatura ambiental de un agente extra, en una base de datos relacional crearíamos una fila con todos los datos de los sensores dejando nulos los que no se utilizan, por el ladro contrario con una base de datos no relacional solo crearíamos un registro con el sensor que se utiliza como se ve en la figura 4.19.

El diseño de la base de datos esta pensado de manera que se puedan ir integrando agentes o sensores sin alterar su funcionamiento. Para el registro y actualización de los agentes se utilizara el diseño de las figuras 4.17 y 4.18 donde tenemos los datos de los agentes como el identificador, estatus, edificio, aula y las fechas, dentro de las fechas tenemos lo que es la fecha y hora del momento de captura, dióxido de carbono, monóxido de carbono, ruido, humedad, temperatura en grados Celsius, temperatura en grados Fahrenheit, índice de calor en grados Celsius y índice de calor den grados Fahrenheit. Para el registro de las ubicaciones de los usuarios se utilizara el diseño de la figura 4.20 donde tenemos el identificador del registro, el agente donde se encontró el usuario, el usuario que se encontró, la fecha y hora que se encontró.

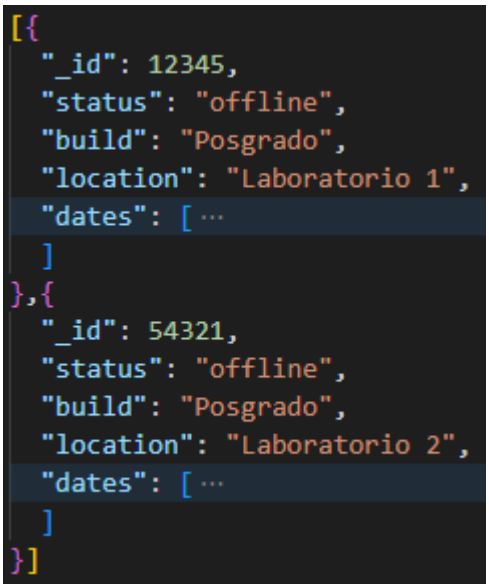
4.8. Construcción de la arquitectura

Dado que ya se tenía un avance de la misma, solo se mostraron algunas figuras de la nueva implementación a la arquitectura como la detección de dispositivos y la manipulación de la luz. De igual manera se compartirá el código en la sección de apéndices.



```
[{
  "_id": 12345,
  "status": "online",
  "build": "Posgrado",
  "location": "Laboratorio 1",
  "dates": [
    {
      "date": "28/02/2022_03:00:00",
      "co2": "869",
      "co": "326",
      "noise": "59",
      "humidity": "21.00",
      "temperature_c": "25.50",
      "temperature_f": "77.90",
      "heat_index_c": "24.65",
      "heat_index_f": "76.38"
    },
    {
      "date": "28/02/2022_03:05:00",
      "co2": "863",
      "co": "330",
      "noise": "58",
      "humidity": "22.00",
      "temperature_c": "25.30",
      "temperature_f": "77.54",
      "heat_index_c": "24.46",
      "heat_index_f": "76.03"
    }
  ]
}]
```


FIGURA 4.17: Diseño de base de datos: colección agente (elaboración propia).



```
[{
  "_id": 12345,
  "status": "offline",
  "build": "Posgrado",
  "location": "Laboratorio 1",
  "dates": [ ···
  ]
},{
  "_id": 54321,
  "status": "offline",
  "build": "Posgrado",
  "location": "Laboratorio 2",
  "dates": [ ···
  ]
}]
```


FIGURA 4.18: Ejemplo de la colección agente con dos agentes (elaboración propia).

```
[{
  "_id": 54321,
  "status": "offline",
  "build": "Posgrado",
  "location": "Laboratorio 2",
  "dates": [
    {
      "date": "28/02/2022_03:00:00",
      "co2": "869",
      "co": "326",
      "humidity": "21.00"
    },
    {
      "date": "28/02/2022_03:05:00",
      "humidity": "22.00"
    }
  ]
}]
```


FIGURA 4.19: Ejemplo de la colección agente cambiando los sensores (elaboración propia).

```
[{
  "_id": "1",
  "agent": 12345,
  "user": "M14240570",
  "updatedAt": "28/02/2022_10:36:45"
},{
  "_id": "2",
  "agent": 12345,
  "user": "M14240570",
  "updatedAt": "28/02/2022_10:37:33"
},{
  "_id": "3",
  "agent": 54321,
  "user": "M14240570",
  "updatedAt": "28/02/2022_10:42:31"
},{
  "_id": "4",
  "agent": 54321,
  "user": "M14240570",
  "updatedAt": "28/02/2022_10:47:29"
}]
```


FIGURA 4.20: Diseño de base de datos: colección ubicación de los usuarios (elaboración propia).

Capítulo 5

Pruebas y Resultados

Durante el desarrollo del proyecto, identificamos tres puntos clave la detección de dispositivos móviles, la integración de censores y la manipulación de luz, de las cuales se hicieron pruebas durante el proceso per una vez finalizado podemos ver mas datos, tener un panorama diferente de para que nos puede servir un sistema enfocado en el bienestar de las personas que día a día recorren las instalaciones del Instituto.

5.1. Agente Inteligente

Nuestro agente al final se mantuvo como el de la figura 4.2 quedando de la siguiente manera 5.1, el porque se utilizo un ESP32 para los sensores, dado que algunos sensores los valores que arrojan son valores análogos la Raspberry pi los interpreta como 1 o 0 a menos que se implemente microchips en los sensores y el ESP32 nos permite hacer la lectura conectando directamente los sensores, si se puede observar el agente al final quedo montado en una Raspberry Pi 3 dado que no exige mucho recurso y se sigue manteniendo el bajo consumo de energía para que sea viable.

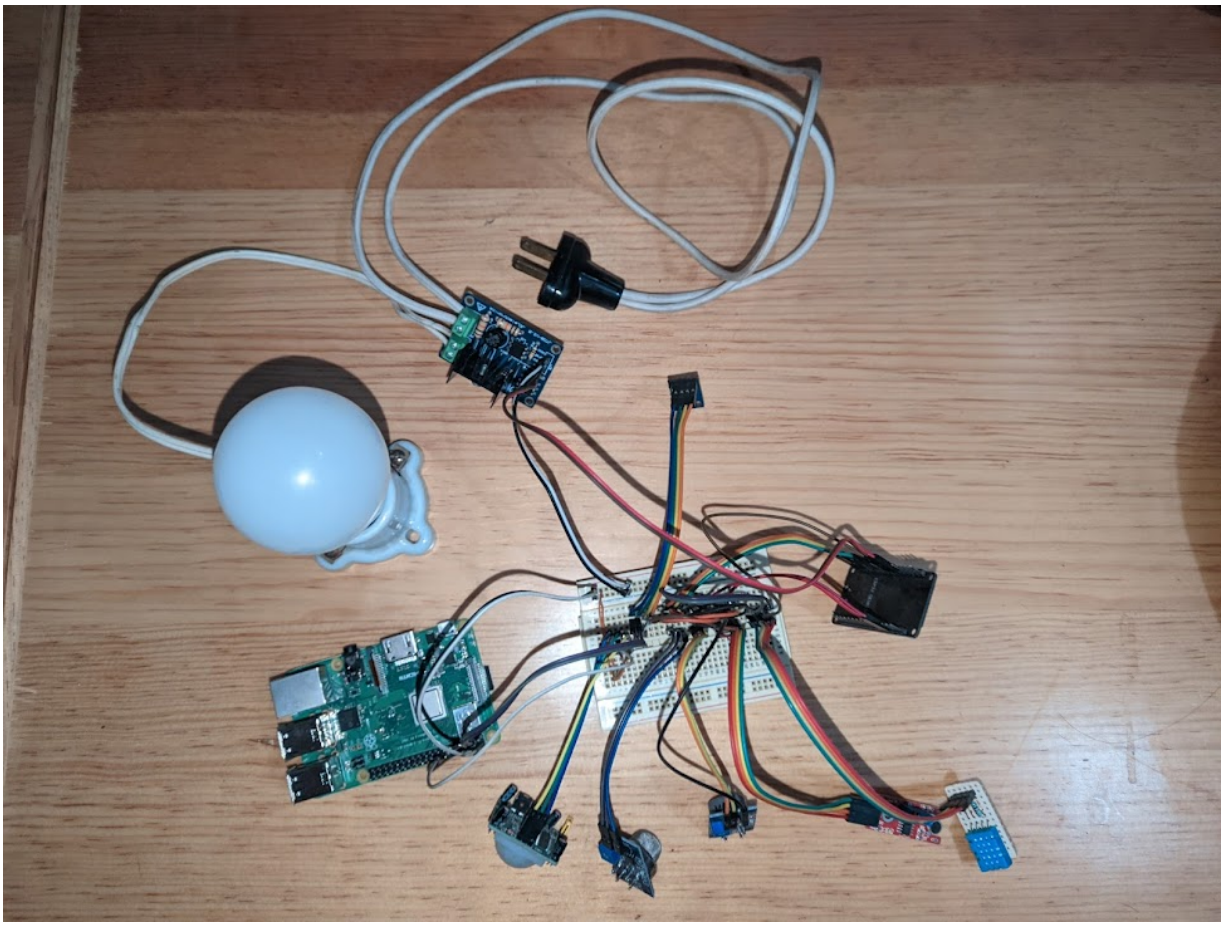


FIGURA 5.1: Estructura del agente montado (elaboración propia).

5.1.1. Sensores

Como podemos ver en las figuras 5.3, 5.4, 5.5, 5.6 y 5.7 tenemos las gráficas de CO, CO_2, la temperatura la contaminación del ruido y el porcentaje de humedad de aproximada mente tres horas, si se deja que el agente siga recolectando datos, en posibles trabajos futuros se tenga una base de conocimiento. y en la figura 5.2 podemos ver como se va almacenando la información en la base de datos.

FIGURA 5.2: Información de los sensores en la base de datos (elaboración propia).

5.1.2. Detección de dispositivos

La detección de dispositivos móviles se logro adecuadamente donde en la figura 5.8 como se almacenan las ubicaciones, vemos que sale el agente y la hora en la que estuvo el usuario en ese agente. En la figura 5.11 la cual es provisional y de pruebas dado que la ubicación de los usuarios no podrá estar a la vista para cualquier usuario.

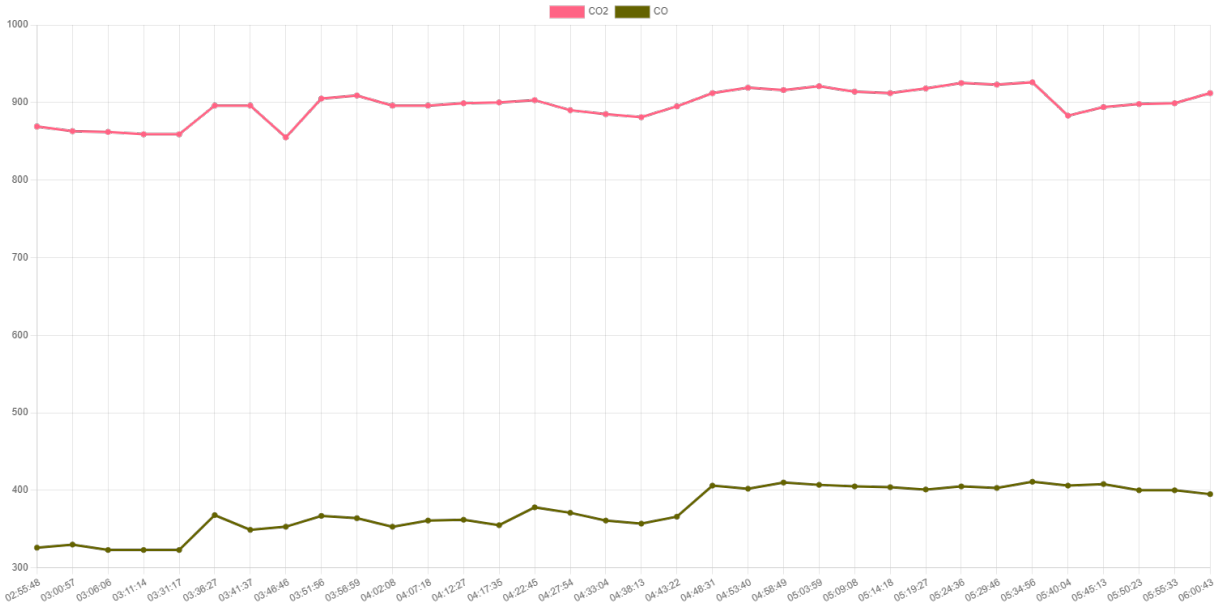


FIGURA 5.3: *CO* y CO_2 (elaboración propia).

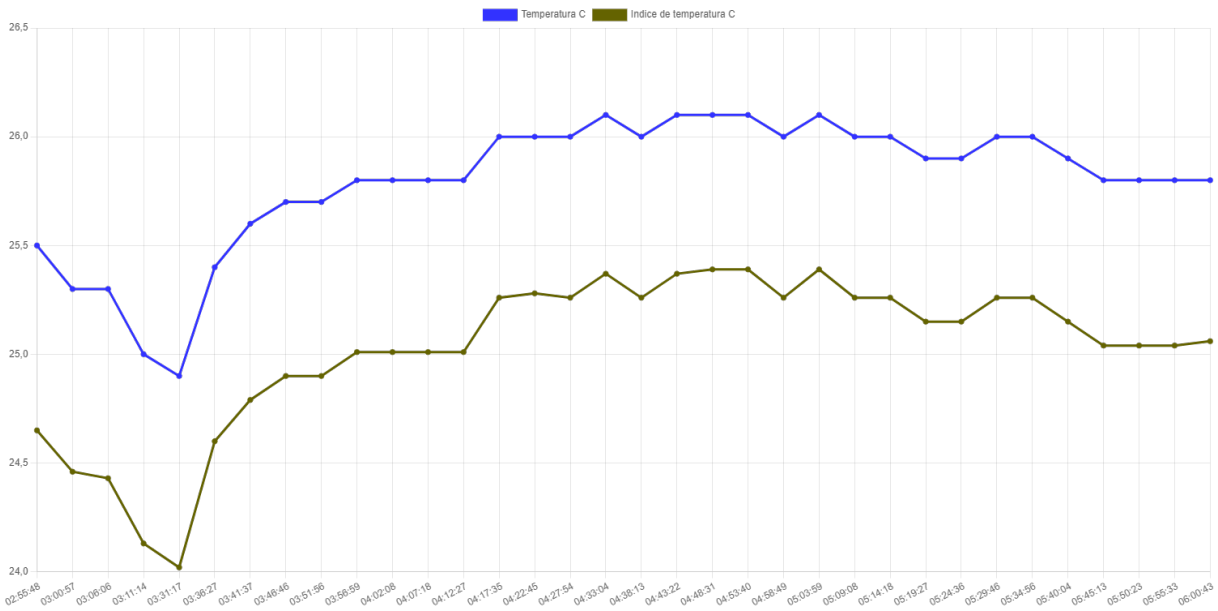


FIGURA 5.4: Temperatura en grados celsius (elaboración propia).

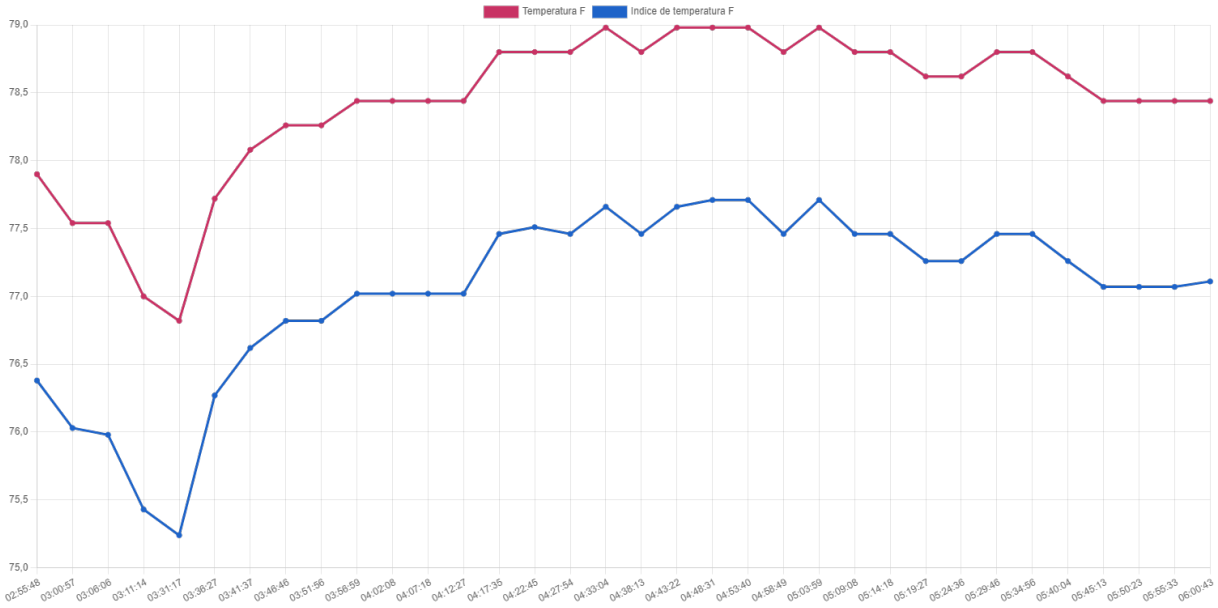


FIGURA 5.5: Temperatura en grados fahrenheit (elaboración propia).

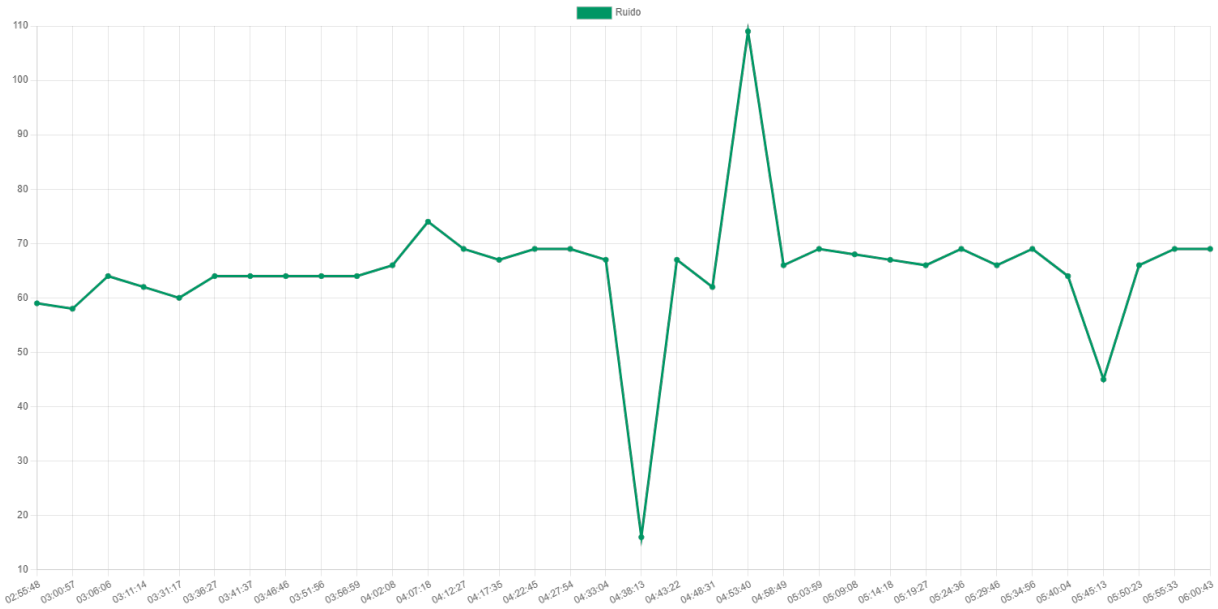


FIGURA 5.6: Ruido (elaboración propia).

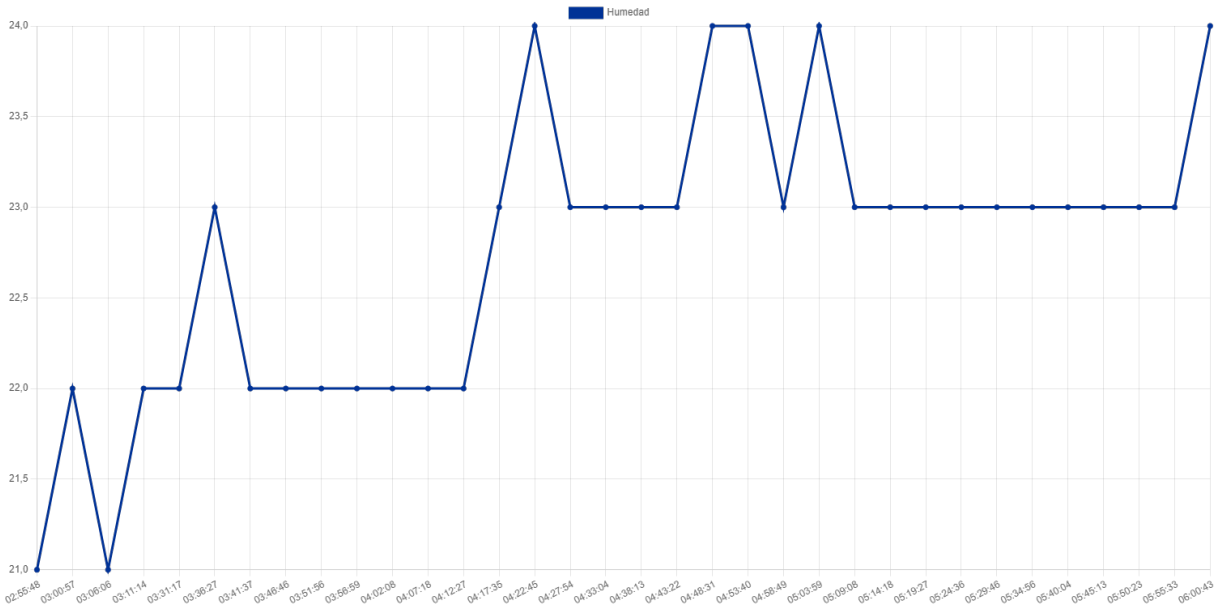


FIGURA 5.7: Humedad (elaboración propia).


```
_id: ObjectId("6218eede514bbd37c01cbe20")
agent: 242606372691727
user: "M14240580"
updatedAt: 2022-02-25T14:59:42.032+00:00

_id: ObjectId("621c9c8a2f84b1e21da05a33")
agent: 354653214445
user: "M14240580"
updatedAt: 2022-02-28T09:57:30.531+00:00

_id: ObjectId("621ca5bdad563eed1f632969")
agent: 242606372691727
user: "M14240570"
updatedAt: 2022-02-28T10:36:45.398+00:00

_id: ObjectId("621ca5edad563eed1f63296a")
agent: 4666478613
user: "M14240570"
updatedAt: 2022-02-28T10:37:33.168+00:00
```


FIGURA 5.8: Ubicación de usuarios guardados en la base de datos (elaboración propia).

5.2. Sistema multiagente

Para que nuestro sistema sea realmente un sistema multiagente se deben implementar más de los mismos agentes de la sección anterior, este caso se puede mantener la misma arquitectura de los agentes como en la figura 5.9, si se quisiera modificar el agente solo se tendrá que modificar el código en dicho agente dado que se trabajó con un Framework para sistemas multiagente (PADE), es posible manejar un código diferente en cada agente para la lectura de los sensores, por otro lado el uso de una base de datos no relacional (MongoDB) nos permite almacenar los datos de los sensores aunque estos sean diferentes.

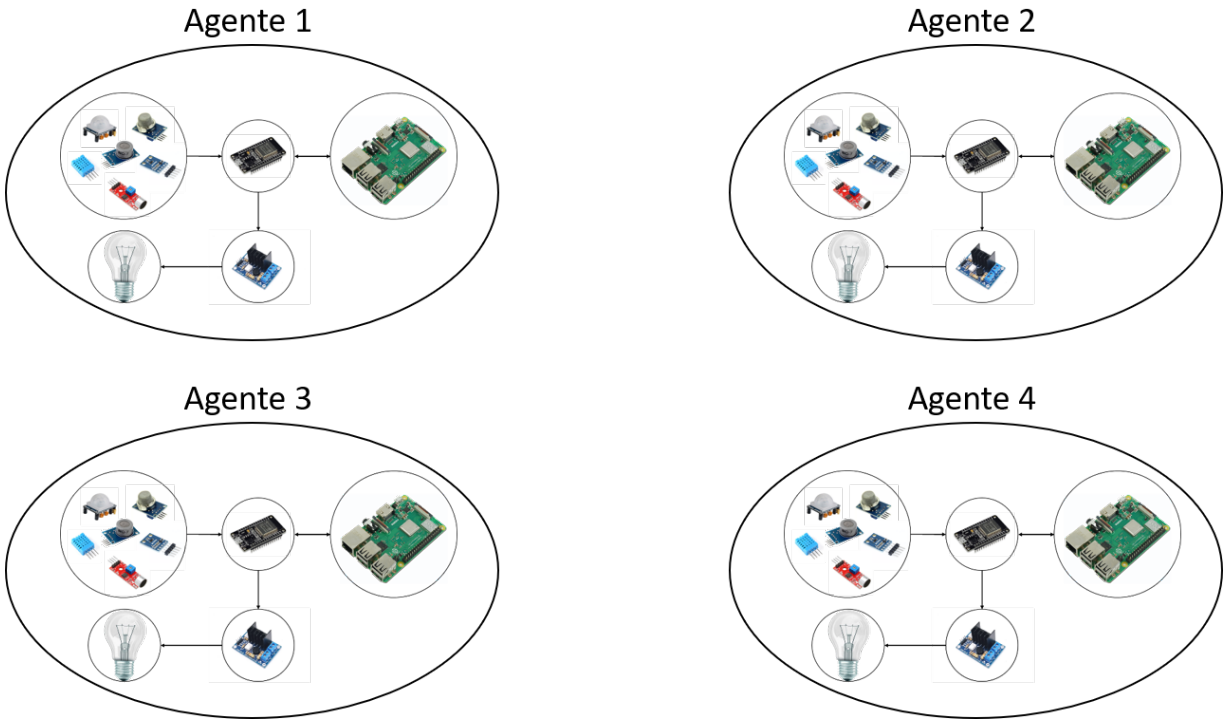


FIGURA 5.9: Sistema multiagente (elaboración propia).

5.3. Aplicación institucional

La aplicación institucional se terminó con éxito, aun se puede mejorar, sabiendo que por el momento solo se cuentan con algunas funciones básicas como los horarios, el registro de agentes con sus sensores, ubicación de usuarios un inicio de sesión como podemos ver en las figuras 5.10, 4.10 y 5.11

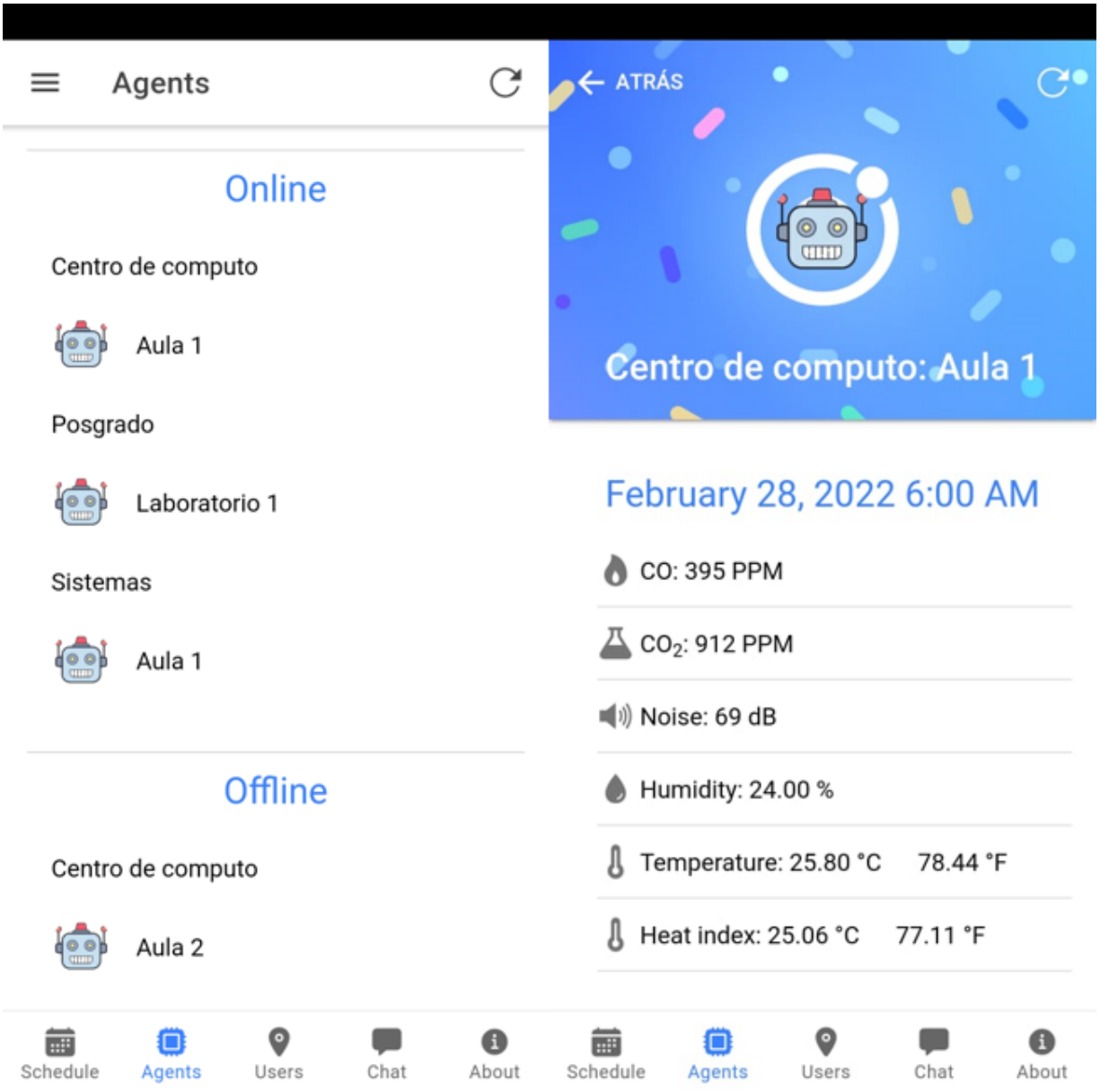


FIGURA 5.10: Datos de los agentes en la aplicación (elaboración propia).

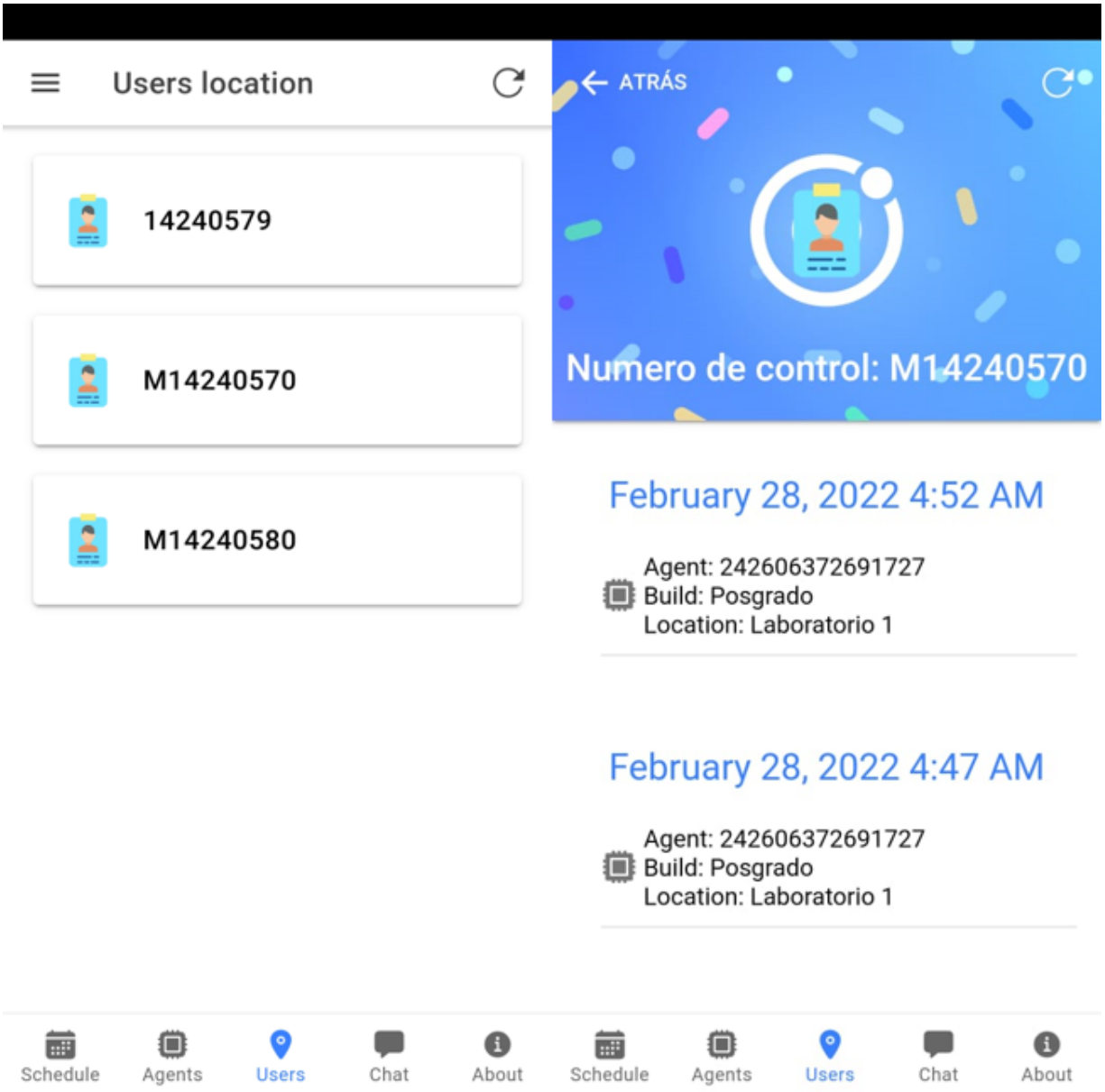


FIGURA 5.11: Ubicación de usuarios en la aplicación (elaboración propia).

5.4. Arquitectura y Smart Campus

Una vez hecho todo el desarrollo se puede decir que ya tenemos el sistema multiagente para el *Smart campus*, le proceso a seguir es replicar estos mismos agentes y empezarlos a colocar por todo el instituto. El uso de la arquitectura se conservo se aplico la propuesta al pie de la letra incluyendo mejoras para que el campus pueda considerarse un ambiente inteligente, aun que le proceso no es de a la noche a la mañana.

5.5. Conclusiones

En términos generales se obtuvieron buenos resultados, se logro construir un sistema multiagente capas de funcionar si un agente se cae, o si los sensores fallan, se puede observar en tiempo real los datos ambientales de un agente se puede ver si un agente o aula esta muy saturada de personas o si el ambiente no es lo más idóneo, por el ruido por la temperatura o la contaminación del aire.

Capítulo 6

Conclusiones y Trabajo Futuro

La tecnología no se detiene, investigando algunos temas uno se da cuenta de que pasan meses y sale algo nuevo referente a un tema la inteligencia artificial, en un principio de la investigación no había tantos artículos referentes a los sistemas multiagente y ahora vemos que están empezando a surgir más interés en este tema. Y de cualquier otro el área de la inteligencia artificial tiene muchas oportunidades, descubrimientos o invenciones que puede hacer que cambie la vida tecnológica como la conocemos como por ejemplo el internet de las cosas, hoy en la actualidad un gran porcentaje de la población cuenta con un asistente inteligente.

Algunas ideas para este proyecto se quedaron en el camino, como el pase de lista automático, imaginando que todos los estudiantes y docentes usen la aplicación. Al entrar al salón de clases se les tomaría automáticamente la asistencia, ya sea que sea por la presencia del dispositivo móvil o incluir una cámara al agente y con técnicas de *Deep Learning* identificar a los alumnos. E inclusive de ser posible mejorar la técnica con la que se determina la intensidad de luz utilizando técnicas de *Machine Learning*.

Otro punto interesante es como en el artículo que pueden mover las cortinas o activar el aire acondicionado para adaptarse a una temperatura templada sí que tenga que existir la intervención humana y de eso poder tener un registro de como ah estado el ambiente durante un periodo establecido.

Otro punto importante que tiene que ver con la salud mental y física de un alumno o docente, imaginar que una persona se aísla, los agentes pueden detectar un comportamiento extraño

o un patrón diferente al habitual y si de ser necesario notificar a un docente para dará apoyo, recomendarle actividades para que se active, etc.

Un trabajo a futuro también podría ser que el sistema haga recomendaciones inteligentes según sus materias, sus grupos de estudio, actividades extraescolares.

Incluir el riego inteligente al sistema, suponiendo que hay un sistema automático que se dedica al riego, que el agente detectara que el sistema de riego está en fallo, pueda checar la humedad de la tierra y mandar notificaciones a las personas cerca de ahí y se pueda hacer algo al respecto como el riego manual o reparar el sistema de riego.

Bibliografía

Gómez R., C. A. & Pedraza, L. F. (2018). Ubicación de dispositivos Móviles en Ambientes Interiores por medio de Análisis de radiación de Redes wifi y deformaciones de Campo Magnético. *Ingeniare. Revista chilena de ingeniería*, *26*(2), 203-212. https://doi.org/10.4067/s0718-33052018000200203

Cruz Parada, J. M., Zamudio Rodriguez, V. M., Lino Ramirez, C. & Gutierrez Hernandez, D. A. (2020). Propuesta de una arquitectura para un smart campus Universitario. *Revista de Innovación Sistemática*, 21-27. https://doi.org/10.35429/jsi.2020.14.4.21.27

Martínez, M. I. R., Ramírez, C. L., Rodríguez, V. M. Z., Hernandez, D. A. G. & Soberanes, H. P. (2020). Designing a Multiagent System for Elderly Care. *Ambient Intelligence and Smart Environments*, *28*, 9-18. https://doi.org/10.3233/AISE200018

Steventon, A. & Wright, S. (2010). *Intelligent spaces: The application of pervasive ICT*. Springer science+Business media.

Lee, S. K., Bae, M. & Kim, H. (2017). Future of IoT networks: A survey. *Applied Sciences*, *7*, 1072. https://doi.org/10.3390/app7101072

Caparrini, F. S. & Work, W. W. (2019). *Introducción a la lógica Difusa*. http://www.cs.us.es/~fsancho/?e=97

Ponce, J., Soto, T. A. & Sayur, Q. F. (2014). *Inteligencia Artificial*. Iniciativa Latinoamericana de Libros de Texto Abiertos.

Serna, B., Baltazar, R., Cruz-Parada, P., Meza, J., Manríquez, J. & Zamudio, V. (2019). Use of intelligent agent through low-cost brain–computer interface to analyze attention and meditation levels by gender. *Agents and Multi-agent Systems: Technologies and Applications 2019*, 163-174. https://doi.org/10.1007/978-981-13-8679-4_14

Sun, Q., Yu, W., Kochurov, N., Hao, Q. & Hu, F. (2013). A multi-agent-based intelligent sensor and Actuator Network Design for smart house and home automation. *Journal of Sensor and Actuator Networks*, *2*(3), 557-588. https://doi.org/10.3390/jsan2030557

GREI-UFC. (2018). *PADE Python Agent DEvelopment framework*. https://pade.readthedocs.io/

Halfacree, G. (2018). *The official Raspberry Pi Beginner's Guide: How to use your new computer*. Raspberry Pi Press.

Arduino. (2005). https://www.arduino.cc/

Barrag, H. (2003). http://wiring.org.co/

Overview: Espressif Systems. (2016). https://www.espressif.com/

LATAM, M. (2021). *Sensores*. https://www.mecatronicalatam.com/es/tutoriales/sensores/

Russell, S. J., Norvig, P., Juan, M. C. R. & Aguilar, J. L. (2011). *Inteligencia artificial: Un Enfoque Moderno*. Pearson Educación.

Pauwels, E. J., Salah, A. A. & Tavenard, R. (2007). Sensor networks for Ambient Intelligence. *2007 IEEE 9th Workshop on Multimedia Signal Processing*. https://doi.org/10.1109/mmsp.2007.4412806

C, E. V. (2020). *Actuadores*. http://www.aie.cl/files/file/comites/ca/abc/actuadores.pdf

Logicbus. (2020). *¿Qué es la automatización?* http://www.logicbus.com.mx/automatizacion.php

Carlos, E. R. B. (2020). *Sistemas integrados y Hogar Digital*. Ediciones Paraninfo, S.A.

Gilbert, D., Fiske, S. & Lindzey, G. (2012). *Redes de computadoras*. PEARSON EDUCACIÓN.

Socket.IO. (2014). *Socket.IO*. https://socket.io/

Díaz, A. G. (2014). *Ingeniería de software avanzada*. Centro Cultural Ítaca, S. C.

Codesido, I. (2009). *What is front-end development?* https://www.theguardian.com/help/insideguardian/2009/sep/28/blogpost

Codecademy, T. (2021). *What is back end?* https://www.codecademy.com/resources/blog/what-is-back-end/

Riehle, D. (2000). *Framework Design*. https://riehle.org/computer-science/research/dissertation/diss-a4.pdf

Oracle. (2020). *¿Qué es una base de datos?* https://www.oracle.com/mx/database/what-is-database/

Microsoft. (2020). *Datos no relacionales Y NoSQL - Azure Architecture Center*. https://docs.microsoft.com/es-es/azure/architecture/data-guide/big-data/non-relational-data

Alvarez Campana, M., López, G., Vázquez, E., Villagrá, V. & Berrocal, J. (2017). Smart Cei Moncloa: An IOT-based platform for people flow and environmental monitoring on a Smart University campus. *Sensors*, *17*(12), 2856. https://doi.org/10.3390/s17122856

Fortes, S., Santoyo Ramón, J., Palacios, D., Baena, E., Mora-García, R., Medina, M., Mora, P. & Barco, R. (2019). The campus as a smart city: University of Málaga Environmental, learning, and research approaches. *Sensors*, *19*(6), 1349. https://doi.org/10.3390/s19061349

Fraga Lamas, P., Celaya Echarri, M., Lopez Iturri, P., Castedo, L., Azpilicueta, L., Aguirre, E., Suárez Albela, M., Falcone, F. & Fernández Caramés, T. M. (2019). Design and experimental validation of A Lorawan fog computing based architecture for IOT enabled Smart Campus Applications. *Sensors*, *19*(15), 3287. https://doi.org/10.3390/s19153287

Abuarqoub, A., Abusaimeh, H., Hammoudeh, M., Uliyan, D., Abu-Hashem, M. A., Murad, S., Al-Jarrah, M. & Al-Fayez, F. (2017). A survey on internet of things enabled Smart campus applications. *Proceedings of the International Conference on Future Networks and Distributed Systems*. https://doi.org/10.1145/3102304.3109810

Babiuch, M. & Postulka, J. (2021). Smart Home Monitoring System using ESP32 microcontrollers. *Internet of Things*. https://doi.org/10.5772/intechopen.94589

Ramos, C., Augusto, J. C. & Shapiro, D. (2008). Ambient Intelligence—the Next Step for Artificial Intelligence. *IEEE Intelligent Systems*, *23*(2), 15-18. https://doi.org/10.1109/mis.2008.19

Yuliansyah, H., Corio, D., Yunmar, R. A. & Kahar Aziz, M. R. (2019a). Smart-room technology implementation based on internet of things toward Smart Campus in Institut Teknologi sumatera. *IOP Conference Series: Earth and Environmental Science*, *258*, 012053. https://doi.org/10.1088/1755-1315/258/1/012053

Yuliansyah, H., Corio, D., Yunmar, R. A. & Kahar Aziz, M. R. (2019b). Energy Monitoring System based on internet of things toward Smart Campus in Institut Teknologi sumatera. *IOP Conference Series: Earth and Environmental Science*, *258*, 012008. https://doi.org/10.1088/1755-1315/258/1/012008

Nehmer, J., Becker, M., Karshmer, A. & Lamm, R. (2006). Living Assistance Systems. *Proceedings of the 28th international conference on Software engineering*. https://doi.org/10.1145/1134285.1134293

Garcia-Valverde, T., Garcia-Sola, A., Hagras, H., Dooley, J. A., Callaghan, V. & Botia, J. A. (2013). A fuzzy logic-based system for indoor localization using WIFI in ambient intelligent environments. *IEEE Transactions on Fuzzy Systems*, *21*(4), 702-718. https://doi.org/10.1109/tfuzz.2012.2227975

Liu, H. (2020). WITHDRAWN: Smart campus student management system based on 5G network and internet of things. *Microprocessors and Microsystems*, 103428. https://doi.org/10.1016/j.micpro.2020.103428

Martin, J., Mayberry, T., Donahue, C., Foppe, L., Brown, L., Riggins, C., Rye, E. C. & Brown, D. (2017). A study of MAC address randomization in mobile devices and when it fails. *Proceedings on Privacy Enhancing Technologies*, *2017*(4), 365-383. https://doi.org/10.1515/popets-2017-0054

Moraes, T., Nogueira, B., Lira, V. & Tavares, E. (2019). Performance comparison of IOT communication protocols. *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*. https://doi.org/10.1109/smc.2019.8914552

Babiuch, M., Foltynek, P. & Smutny, P. (2019). Using the ESP32 microcontroller for Data Processing. *2019 20th International Carpathian Control Conference (ICCC)*. https://doi.org/10.1109/carpathiancc.2019.8765944

Christian, W. & Mathieu, P. (2020). http://ritweb.cloudapp.net:8080/JuzzyOnline/

Pravalika, V. & Prasad, C. R. (2019). Internet of Things Based Home Monitoring and Device Control Using Esp32. *International Journal of Recent Technology and Engineering*, *8*(4), 58-62.

Anexo A

Ponencias

En esta sección se adjuntan los reconocimientos por haber expuesto en el congreso Internacional de Ingenierías 2020 llevado a cabo por el TECNM Campus Misantla y en el congreso Internacional Multidisciplinario de ingenieras 2022 llevado a cabo por la Universidad Politécnica Juventino Rosas, en este último se publicó un artículo titulado “Sistema multiagente integrado a un Smart campus para la detección de dispositivos móviles y administración de lámparas”.

TECNM Campus Misantla

CONGRESO INTERNACIONAL DE INGENIERÍAS

Otorga la presente

CONSTANCIA

A: José Frabicio de la Cruz Ponce.

Por su participación en la modalidad **PONENCIA TÉCNICA** con el trabajo titulado:

Construcción de la arquitectura de un Smart Campus universitario.

Realizado en el marco del Congreso Internacional de Ingenierías 2020, los días 26, 27 y 28 de noviembre del 2020, en modalidad virtual

M.E. JOSÉ ROBERTO ARENAS MARTÍNEZ
Director General
Instituto Tecnológico Superior de Misantla

Misantla, Ver., a 28 de noviembre de 2020

Proyecto apoyado por el CONACYT

UPJR

UNIVERSIDAD POLITÉCNICA
JUVENTINO ROSAS

Otorga el presente

RECONOCIMIENTO A:

José Fabricio de la Cruz Ponce

Por haber presentado su ponencia titulada:

<<Sistema multiagente integrado a un Smart campus para la detección de dispositivos móviles y administración de lámparas>>

En el marco del Congreso Internacional Multidisciplinario de Ingenierías 2022, realizado del 02 al 03 de junio de 2022 en la Ciudad de Guanajuato, Gto. México

"Formación para una vida mejor"

M.I. JOSÉ GABRIEL AGUILERA GONZÁLEZ

SECRETARIO ACADÉMICO

Anexo B

Código fuente del Smart Campus

En esta sección se adjuntan todo el código fuente del agente, del socket y algunas partes de código más importantes de la aplicación móvil dado que tiene una gran cantidad de código. Estas líneas de código son las más importantes si se quisiera replicar el sistema del Smart Campus, dado que contiene las bases para de la funcionalidad.

B.1. Código fuente del agente

agent.py

```
from pade.misc.utility import display_message, start_loop, call_later
from pade.core.agent import Agent
from pade.acl.messages import ACLMessage
from pade.acl.aid import AID
from pade.behaviours.protocols import TimedBehaviour
from sys import argv
from socket_connector import sio, updateSensores
import bluetooth
from bluetooth.ble import BeaconService
from database import update_status, update_sensors
import os
```

```
from dotenv import load_dotenv
import SensorPir as sensores
import RPi.GPIO as GPIO

GPIO.setup(13, GPIO.IN)  # Cruce 0
GPIO.setup(15, GPIO.OUT)  # Salidda luz
on_off = False
intensidad = 20

load_dotenv()

### Agente ###
class ComportTemporal(TimedBehaviour):
    def __init__(self, agent, time):
        super(ComportTemporal, self).__init__(agent, time)

    def on_time(self):
        global on_off
        global intensidad
        super(ComportTemporal, self).on_time()
        #print('Aqui ando')
        nearby_devices = bluetooth.discover_devices()
        print (nearby_devices)
        sensors, luz, intensidad, on_off = sensores.hay_alguien()
        if update_sensors(my_id, sensors):
            updateSensores()
        print(sensors, luz, intensidad, on_off)

    def on_start(self):
```

```
        super(ComportTemporal, self).on_start()

        # Update database
        update_status(my_id)

        # Socket
        sio.connect(os.getenv('SOCKET_URI'))

class Agente(Agent):
    def __init__(self, aid):
        super(Agent, self).__init__(aid=aid, debug=False)

        comp_temp = ComportTemporal(self, 5.0)

        self.behaviours.append(comp_temp)

        sio.wait()

if __name__ == '__main__':
    # Get ID
    mac = bluetooth.read_local_bdaddr()
    print("Mi mac addres: ", mac)
    my_id = int(mac[-1].replace(':', ''), 16)

    # Turn On Agent
    agents_list = list()
    agente = Agente(AID(name=str(my_id)))
    agents_list.append(agente)
```

```
    start_loop(agents_list)
```

socket_connector.py

```
import socketio
import bluetooth


sio = socketio.Client()


@sio.event
def connect():
    print('connection established')
    # Get ID
    mac = bluetooth.read_local_bdaddr()
    my_id = int(mac[-1].replace(':', ''), 16)
    sio.emit('joinAgent',my_id)


@sio.event
def my_message(data):
    #print('message received with ', data)
    sio.emit('my response', {'response': 'my response'})


@sio.event
def updateSensores():
    sio.emit('getAgents')


@sio.event
def disconnect():
    print('disconnected from server')
```

sensores.py

```
import serial
import time

MAX_BUFF_LEN = 255

typeSensor = ['CO2','CO','Sound','Humidity','Temperature␣C',
    'Temperature␣F','Heat␣index␣C', 'Heat␣index␣F']

# Serial port(windows-->COM), baud rate, timeout msgse
port = serial.Serial("/dev/ttyUSB0", 115200, timeout=1)

# read one char (default)
def read_ser(num_char = 1):
        string = port.read(num_char)
        return string.decode()

# Write whole strings
def write_ser(cmd):
        cmd = cmd + '\n'
        port.write(cmd.encode())

def main():
    write_ser("SENSORS")
    string = read_ser(MAX_BUFF_LEN)
    if(len(string)):
        sensors = string.split(',')
    return sensors

def onOff(status):
```

```
    write_ser(status)
```

logicaDifusa.py

```
import numpy as np
import skfuzzy as fuzz


# Generar variables del universo
#   Rango de horas es de 0 a 24, rango de luz solar es de 0 a 10
#   Corriente de salida en voltaje es de 20 a 80
x_time = np.arange(0, 25, 1)
x_light = np.arange(0, 31, 1)
x_current = np.arange(20, 81, 1)


def defuzzified(aggregated):
    # Calcular el resultado defuzzificado
    current = fuzz.defuzz(x_current, aggregated, 'centroid')
    return current


def fuzzificaton(val1,val2):
    # print(val1)
    # print(val2)
    # Generar funciones de membres a difusas
    time_lo = fuzz.trimf(x_time, [0, 0, 10])
    time_md = fuzz.trimf(x_time, [8, 12, 16])
    time_hi = fuzz.trimf(x_time, [14, 24, 24])


    light_lo = fuzz.trimf(x_light, [0, 0, 10])
    light_md = fuzz.trimf(x_light, [0, 10, 30])
    light_hi = fuzz.trimf(x_light, [20, 30, 30])
```

```
current_lo = fuzz.trimf(x_current, [20, 30, 40])
current_md = fuzz.trimf(x_current, [40, 50, 60])
current_hi = fuzz.trimf(x_current, [60, 70, 80])

# Necesitamos la activacion de nuestras funciones de membresia
    borrosa en estos valores.
# Los valores exactos val1 y val2 no existen en nuestros
    universos...
# Esto es para lo que existe la membres a fuzz.interp_membership

time_level_lo = fuzz.interp_membership(x_time, time_lo, val1)
time_level_md = fuzz.interp_membership(x_time, time_md, val1)
time_level_hi = fuzz.interp_membership(x_time, time_hi, val1)

light_level_lo = fuzz.interp_membership(x_light, light_lo, val2)
light_level_md = fuzz.interp_membership(x_light, light_md, val2)
light_level_hi = fuzz.interp_membership(x_light, light_hi, val2)

#Reglas
current_activation_hi = np.fmin(light_level_lo, current_hi)

active_rule2 = np.fmin(light_level_hi, time_level_lo)
active_rule3 = np.fmin(light_level_hi, time_level_hi)
active_rule4 = np.fmax(light_level_md, np.fmax(active_rule2,
    active_rule3))

current_activation_md = np.fmin(active_rule4, current_md)

active_rule5 = np.fmin(light_level_lo, time_level_md)
```

```
    active_rule6 = np.fmax(light_level_hi, active_rule5)
    current_activation_lo = np.fmin(active_rule6, current_lo)

    aggregated = np.fmax(current_activation_hi,
        np.fmax(current_activation_md,current_activation_lo))
    return defuzzified(aggregated)
```

database.py

```
import os
from turtle import update
from dotenv import load_dotenv
import pymongo as mongo
from datetime import datetime, timedelta
load_dotenv()

# Database
cliente = mongo.MongoClient(os.getenv('DATABASE_URI'))
db = cliente[os.getenv('DATABASE_NAME')]
agentUpdate = False

def update_status(id):
    col = db['agent']
    query = { "_id" : id }
    val =   { "$set": { "status" : "online" } }
    if col.find_one(query):
        col.update_one(query , val)
    else:
        col.insert_one({ "_id" : id , "status" : "online",
            "build":"Posgrado", "location":"Laboratorio 1",
            "dates":[]})
```

```
def update_sensors(id, sensors):
    agentUpdate = False
    col = db['agent']
    query = { "_id" : id }
    values = col.find_one(query)['dates']
    if values==[]:
        values=[{
            "date": datetime.now(),
            "co2": sensors[0],
            "co": sensors[1],
            "noise": sensors[2],
            "humidity": sensors[3],
            "temperature_c": sensors[4],
            "temperature_f": sensors[5],
            "heat_index_c": sensors[6],
            "heat_index_f": sensors[7]
        }]
        agentUpdate=True
    elif values[-1]['date']< datetime.now()-timedelta(seconds=5*60):
        values.append({
            "date": datetime.now(),
            "co2": sensors[0],
            "co": sensors[1],
            "noise": sensors[2],
            "humidity": sensors[3],
            "temperature_c": sensors[4],
            "temperature_f": sensors[5],
            "heat_index_c": sensors[6],
```

```
            "heat_index_f": sensors[7]
        })
        agentUpdate=True
    val = { "$set":  { "dates": values } }
    if col.find_one(query):
        col.update_one(query , val)
    return agentUpdate
```

sensorPir.py

```
import RPi.GPIO as GPIO
import LogicaDifusa as ld
import SensorLuz as sl
import Sensores as sen
from datetime import datetime
from decimal import Decimal

GPIO.setwarnings(False)
GPIO.setmode(GPIO.BOARD)
GPIO.setup(16, GPIO.IN)  # PIR motion sensor data



def hay_alguien():
    luz = 0
    intensidad = 20
    if GPIO.input(16) == 1:
        #print ("Hay alguien")
        luz = sl.main()
        if Decimal(luz) > 30:
            luz = 30
        intensidad = ld.fuzzificaton(datetime.now().hour +
```

```
            datetime.now().minute/60, luz)
        on_off=True
        sen.onOff(str(intensidad))
    else:
        #print("No hay nadie")
        on_off=False
        sen.onOff("OFF")
    return sen.main(), luz, intensidad, on_off
```

sensorLuz.py

```
import smbus
import time

# Define some constants from the datasheet
DEVICE      = 0x23 # Default device I2C address

POWER_DOWN = 0x00 # No active state
POWER_ON    = 0x01 # Power on
RESET       = 0x07 # Reset data register value

# Start measurement at 4lx resolution. Time typically 16ms.
CONTINUOUS_LOW_RES_MODE = 0x13
CONTINUOUS_HIGH_RES_MODE_1 = 0x10
CONTINUOUS_HIGH_RES_MODE_2 = 0x11
ONE_TIME_HIGH_RES_MODE_1 = 0x20
ONE_TIME_HIGH_RES_MODE_2 = 0x21
ONE_TIME_LOW_RES_MODE = 0x23

#bus = smbus.SMBus(0) # Rev 1 Pi uses 0
bus = smbus.SMBus(1)  # Rev 2 Pi uses 1
```

```
def convertToNumber(data):
    result=(data[1] + (256 * data[0])) / 1.2
    return (result)

def readLight(addr=DEVICE):
    # Read data from I2C interface
    data = bus.read_i2c_block_data(addr,ONE_TIME_HIGH_RES_MODE_1)
    return convertToNumber(data)

def main():
    lightLevel=readLight()
    #time.sleep(0.5)
    return format(lightLevel,'.2f')

if __name__=="__main__":
    main()
```

sensores.ino

```
#include <RBDdimmer.h>//https://github.com/RobotDynOfficial/RBDDimmer
#include "DHT.h"

#define SENSOR1 4
#define SENSOR2 34
#define SOUND 35
#define DELAY 500
#define DHTPIN 5
#define DHTTYPE DHT11    // DHT 11
DHT dht(DHTPIN, DHTTYPE);
```

```
const int zeroCrossPin  = 18;
const int acdPin  = 19;
// variables to store the value coming from the sensor
int CO2SensorValue = 0;
int COSensorValue = 0;
int SoundSensorValue = 0;
float h = 0;
float t = 0;
float f = 0;
float hif = 0;
float hic = 0;
uint8_t idx = 0;
char c;
char str[255];

dimmerLamp acd(acdPin, zeroCrossPin);

void setup() {
  //pinMode(LED, OUTPUT);
  Serial.begin(115200);
  //Serial.println("Sensor start");
  //Serial.println(F("DHTxx test!"));
  dht.begin();
  acd.begin(NORMAL_MODE, ON);
}

void loop() {
  if (Serial.available() > 0) {
    c = Serial.read();
```

```
if (c != '\n') {
  str[idx++] = c;
}
else {
  str[idx++] = '\0';
  idx = 0;
  if (String(str).equals("SENSORS")) {
    // read the value from the sensor:
    CO2SensorValue = analogRead(SENSOR1);
    COSensorValue = analogRead(SENSOR2);
    SoundSensorValue = analogRead(SOUND);

    h = dht.readHumidity();
    t = dht.readTemperature();
    f = dht.readTemperature(true);

    // Check if any reads failed and exit early (to try again).
    if (isnan(h) || isnan(t) || isnan(f)) {
      Serial.println(F("Failed to read from DHT sensor!"));
      return;
    }

    // Compute heat index in Fahrenheit (the default)
    hif = dht.computeHeatIndex(f, h);
    // Compute heat index in Celsius (isFahreheit = false)
    hic = dht.computeHeatIndex(t, h, false);

    Serial.print(String(CO2SensorValue) + "," +
        String(COSensorValue) + "," +
```

```
            String(SoundSensorValue) + "," +
            String(h) + "," + String(t) + "," +
            String(f) + "," + String(hic) + "," +
            String(hif));
      } else if (String(str).equals("OFF")) {
        acd.setState(OFF);
      } else {
        acd.setPower(String(str).toFloat());
      }
    }
  }
}
```

B.2. Código fuente del servidor socket

socket.js

```
var mongoDb = require('mongodb');
const dotenv = require('dotenv');
dotenv.config();
const uri = process.env.URI;
const { get } = require("https");
const { type } = require("os");
var databaseName = process.env.DATABASE_NAME;
var users = [];
var agents = [];

const { createServer } = require("http");
const { Server } = require("socket.io");

const httpServer = createServer();
const io = new Server(httpServer, { /* options */ });

// A class that describes the attributes
// that each socket user has
class User {
    constructor(socketId, id, type) {
        this.socketId = socketId;
        this.id = id;
        this.type = type;
    }
}

// Functions to manage the list of users and agents
```

```
function addUser(socketId, id, type) {
    let user = new User(socketId, id, type);
    users.push(user);
    return user;
}

function addAgent(socketId, id, type) {
    let agent = new User(socketId, id, type);
    agents.push(agent);
    return agent;
}

function getUserBySocketId(id) {
    return users.filter((user) => user.socketId === id)[0];
}

function getAgentBySocketId(id) {
    return agents.filter((user) => user.socketId === id)[0];
}

function getUserById(id) {
    return users.filter((user) => user.id === id)[0];
}

function getAgentById(id) {
    return agents.filter((user) => user.id === id)[0];
}

function removeUser(id) {
```

```
    let user = getUserById(id);

    if (user) {
        users = users.filter((user) => user.id !== id);
    }
    return user;
}

function removeAgent(id) {
    let user = getAgentById(id);

    if (user) {
        agents = agents.filter((user) => user.id !== id);
    }
    return user;
}

io.sockets.on('connection', function (socket) {

    // When human user join to the socket service
    socket.on('join', (user) => {
        removeUser(user.id);
        socket.join(user.type);
        addUser(socket.id, user.id, user.type);
        getAgentsOnlineDB(socket.id);
        getAgentsDB(socket.id)
        getUserLocationDB(socket.id)
        console.log("Users")
        console.log(users)
```

```
});

// Update the position of the user's match with an agent
socket.on('here', (macAddressAgent) => {
    try {
        agentId = getIdFromMac(macAddressAgent);
        //console.log(agentId)
        user = getUserBySocketId(socket.id);
        updateUserPosition(agentId, user.id);
        getUserLocationDB(1)
    } catch (e) { }
});

// When agent user join to the socket service
socket.on('joinAgent', (agentId) => {
    removeAgent(agentId);
    addAgent(socket.id, agentId, 0);
    setAgentStatus(agentId, true);
    getAgentsOnlineDB(1);
    getAgentsDB(1);
    console.log("Agents")
    console.log(agents)
});

// Send a broadcast message by the type
socket.on('message', function (msg) {
    let user = getUserBySocketId(socket.id);
    if (user) {
        io.sockets.to(user.type).emit('message', msg);
```

```
        }
    });

    socket.on('getAgents', ()  => {
        getAgentsDB(1);
    });

    // Triggered when a user disconect
    socket.on('disconnect', () => {
        let user = getUserBySocketId(socket.id);

        if (user) {
            removeUser(user.id);
        } else {
            let agent = getAgentBySocketId(socket.id);

            if (agent) {
                removeAgent(agent.id);
                setAgentStatus(agent.id, false);
                getAgentsOnlineDB(1);
                getAgentsDB(1);
            }
        }
        console.log(users)
    })
});

//--------------------HELPERS-----------------------
// Change de Mac Address of the device for an identifier
```

```
function getIdFromMac(mac) {
    return parseInt(mac.replaceAll(':', ''), 16);
}

// Change de identifier of the agent for the mac address
function getMacFromID(id) {
    return id.toString(16).match(/.{1,2}/g).join(':').toUpperCase();
}

// Gets the identifications of the agent list
function getMacAddressList(agentsOnline) {
    let list = [];

    agentsOnline.forEach(agent => {
        id = agent['_id'];
        list.push(getMacFromID(id));
    });

    return list;
}

//-----------------DATABASE-------------------
// Gets the list of the agents online in database
async function getAgentsOnlineDB(id) {

    var client = mongoDb.MongoClient;

    await client.connect(uri, { useUnifiedTopology: true }).then(
        (client) => {
```

```
        const database = client.db(databaseName);
        const collection = database.collection("agent");

        // Query for a agents that has the status 'online'
        const query = { status: "online" };

        const options = {};

        collection.find(query, options).toArray(function (
            err, result) {
            if (err) throw err;
            //console.log(result);
            io.sockets.to(id).emit('setAgentsOnline',
                getMacAddressList(result));
            client.close();
        });
    }).catch((e) => {
        console.log(e);
    });
}

// Gets the list of the agents in database
async function getAgentsDB(id) {

    var client = mongoDb.MongoClient;

    await client.connect(uri, { useUnifiedTopology: true }).then(
        (client) => {
        const database = client.db(databaseName);
```

```
        const collection = database.collection("agent");

        // Query for a agents that has the status 'online'
        const query = {};

        const options = {};

        collection.find(query, options).toArray(function (
            err, result) {
            if (err) throw err;
            io.sockets.to(id).emit('setAgents', result);
            client.close();
        });
    }).catch((e) => {
        console.log(e);
    });
}

// Gets the list of the agents in database
async function getUserLocationDB(id) {

    var client = mongoDb.MongoClient;

    await client.connect(uri, { useUnifiedTopology: true }).then(
        (client) => {
        const database = client.db(databaseName);
        const collection = database.collection(
            "user_location_agent_data");
```

```
        // Query for a agents that has the status 'online'
        const query = {};

        const options = {};

        collection.find(query, options).toArray(function (
            err, result) {
            if (err) throw err;
            io.sockets.to(id).emit('setLocationUser', result);
            client.close();
        });
    }).catch((e) => {
        console.log(e);
    });
}

// Update agent status in database when connecting or disconnecting
async function setAgentStatus(agentId, bool) {
    var client = mongoDb.MongoClient;
    let status = (bool) ? 'online' : 'offline'

    await client.connect(uri, { useUnifiedTopology: true }).then(
        (client) => {
        const database = client.db(databaseName);
        const collection = database.collection("agent");

        // Query for a agents that has the status 'online'
        const query = { _id: agentId };
```

```
        const options = { $set: { status: status } };

        collection.updateOne(query, options, function (err, result) {
            if (err) throw err;
            client.close();
        });
    }).catch((e) => {
        console.log(e);
    });
}

// Update agent status in database when connecting or disconnecting
async function updateUserPosition(agentId, userId) {

    var client = mongoDb.MongoClient;

    await client.connect(uri, { useUnifiedTopology: true }).then(
        (client) => {
        const database = client.db(databaseName);
        const collection = database.collection("position_user");

        collection.insertOne({
            'agent': agentId,
            'user': userId,
            'updatedAt': new Date()
        }, () => {
            client.close();
        })
    }).catch((e) => {
```

```
        console.log(e);
    });
}

// Begin the socket service
httpServer.listen(process.env.PORT, function () {
    console.log('Servidor de sockets activo', process.env.PORT);
});
```

B.3. Código fuente de la aplicación móvil

config.xml

```
<?xml version='1.0' encoding='utf-8'?>
<widget id="com.ionicframework.conferenceapp" version="0.0.1"
    xmlns="http://www.w3.org/ns/widgets"
    xmlns:cdv="http://cordova.apache.org/ns/1.0">
    <name>Smart ITL</name>
    <description>Smart Campus University by
        Intituto Tecnologico de Leon.</description>
    <author email="chema.cruz.p@gmail.com, fabrixcp@gmail.com"
        href="https://www.itleon.edu.mx/">Jose Maria Cruz Parada y
        Jose Fabricio de la Cruz Ponce</author>
    <content src="index.html" />
    <access origin="*" />
    <allow-intent href="http://*/*" />
    <allow-intent href="https://*/*" />
    <allow-intent href="tel:*" />
    <allow-intent href="sms:*" />
    <allow-intent href="mailto:*" />
    <allow-intent href="geo:*" />
    <preference name="ScrollEnabled" value="false" />
    <preference name="android-minSdkVersion" value="22" />
    <preference name="BackupWebStorage" value="none" />
    <preference name="SplashMaintainAspectRatio" value="true" />
    <preference name="FadeSplashScreenDuration" value="300" />
    <preference name="SplashShowOnlyFirstTime" value="false" />
    <preference name="SplashScreen" value="screen" />
    <preference name="SplashScreenDelay" value="3000" />
    <platform name="android">
```

```
<allow-intent href="market:*" />
<icon density="ldpi"
src="resources/android/icon/drawable-ldpi-icon.png" />
<icon density="mdpi"
src="resources/android/icon/drawable-mdpi-icon.png" />
<icon density="hdpi"
src="resources/android/icon/drawable-hdpi-icon.png" />
<icon density="xhdpi"
src="resources/android/icon/drawable-xhdpi-icon.png" />
<icon density="xxhdpi"
src="resources/android/icon/drawable-xxhdpi-icon.png" />
<icon density="xxxhdpi"
src="resources/android/icon/drawable-xxxhdpi-icon.png" />
<splash density="land-ldpi" src=
"resources/android/splash/drawable-land-ldpi-screen.png" />
<splash density="land-mdpi" src=
"resources/android/splash/drawable-land-mdpi-screen.png" />
<splash density="land-hdpi" src=
"resources/android/splash/drawable-land-hdpi-screen.png" />
<splash density="land-xhdpi" src=
"resources/android/splash/drawable-land-xhdpi-screen.png" />
<splash density="land-xxhdpi" src=
"resources/android/splash/drawable-land-xxhdpi-screen.png" />
<splash density="land-xxxhdpi" src=
"resources/android/splash/drawable-land-xxxhdpi-screen.png" />
<splash density="port-ldpi" src=
"resources/android/splash/drawable-port-ldpi-screen.png" />
<splash density="port-mdpi" src=
"resources/android/splash/drawable-port-mdpi-screen.png" />
```

```
        <splash density="port-hdpi" src=
        "resources/android/splash/drawable-port-hdpi-screen.png" />
        <splash density="port-xhdpi" src=
        "resources/android/splash/drawable-port-xhdpi-screen.png" />
        <splash density="port-xxhdpi" src=
        "resources/android/splash/drawable-port-xxhdpi-screen.png" />
        <splash density="port-xxxhdpi" src=
        "resources/android/splash/drawable-port-xxxhdpi-screen.png" />
    </platform>
    <platform name="ios">
        <allow-intent href="itms:*" />
        <allow-intent href="itms-apps:*" />
        <icon height="57"
        src="resources/ios/icon/icon.png" width="57" />
        <icon height="114"
        src="resources/ios/icon/icon@2x.png" width="114" />
        <icon height="40"
        src="resources/ios/icon/icon-40.png" width="40" />
        <icon height="80"
        src="resources/ios/icon/icon-40@2x.png" width="80" />
        <icon height="120"
        src="resources/ios/icon/icon-40@3x.png" width="120" />
        <icon height="50"
        src="resources/ios/icon/icon-50.png" width="50" />
        <icon height="100"
        src="resources/ios/icon/icon-50@2x.png" width="100" />
        <icon height="60"
        src="resources/ios/icon/icon-60.png" width="60" />
        <icon height="120"
```

```
        src="resources/ios/icon/icon-60@2x.png" width="120" />
        <icon height="180"
        src="resources/ios/icon/icon-60@3x.png" width="180" />
        <icon height="72"
        src="resources/ios/icon/icon-72.png" width="72" />
        <icon height="144"
        src="resources/ios/icon/icon-72@2x.png" width="144" />
        <icon height="76"
        src="resources/ios/icon/icon-76.png" width="76" />
        <icon height="152"
        src="resources/ios/icon/icon-76@2x.png" width="152" />
        <icon height="167"
        src="resources/ios/icon/icon-83.5@2x.png" width="167" />
        <icon height="29"
        src="resources/ios/icon/icon-small.png" width="29" />
        <icon height="58"
        src="resources/ios/icon/icon-small@2x.png" width="58" />
        <icon height="87"
        src="resources/ios/icon/icon-small@3x.png" width="87" />
        <icon height="1024"
        src="resources/ios/icon/icon-1024.png" width="1024" />
        <splash height="1136"
        src="resources/ios/splash/Default-568h@2x~iphone.png"
        width="640" />
        <splash height="1334"
        src="resources/ios/splash/Default-667h.png" width="750" />
        <splash height="2208"
        src="resources/ios/splash/Default-736h.png" width="1242" />
        <splash height="1242"
```

```
        src="resources/ios/splash/Default-Landscape-736h.png"
        width="2208" />
        <splash height="1536"
        src="resources/ios/splash/Default-Landscape@2x~ipad.png"
        width="2048" />
        <splash height="2048"
        src="resources/ios/splash/Default-Landscape@~ipadpro.png"
        width="2732" />
        <splash height="768"
        src="resources/ios/splash/Default-Landscape~ipad.png"
        width="1024" />
        <splash height="2048"
        src="resources/ios/splash/Default-Portrait@2x~ipad.png"
        width="1536" />
        <splash height="2732"
        src="resources/ios/splash/Default-Portrait@~ipadpro.png"
        width="2048" />
        <splash height="1024"
        src="resources/ios/splash/Default-Portrait~ipad.png"
        width="768" />
        <splash height="960" src=
        "resources/ios/splash/Default@2x~iphone.png" width="640" />
        <splash height="480"
        src="resources/ios/splash/Default~iphone.png" width="320" />
        <splash height="2732"
        src="resources/ios/splash/Default@2x~universal~anyany.png"
        width="2732" />
    </platform>
    <plugin name="cordova-plugin-device" spec="^2.0.2" />
```

```
    <plugin name="cordova-plugin-inappbrowser" spec="^3.0.0" />
    <plugin name="cordova-plugin-splashscreen" spec="^5.0.2" />
    <plugin name="cordova-plugin-whitelist" spec="^1.3.2" />
    <plugin name="cordova-plugin-ionic-webview" spec="2.0.0-beta.1" />
    <plugin name="cordova-plugin-statusbar" spec="^2.4.2" />
    <plugin name="cordova-plugin-ionic-keyboard" spec="^2.1.2" />
    <engine name="ios" spec="^4.5.5" />
    <engine name="android" spec="7.0.0" />
</widget>
```

src/app/app-routing.module.ts

```
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { CheckTutorial } from './providers/check-tutorial.service';

const routes: Routes = [
  {
    path: '',
    redirectTo: '/tutorial',
    pathMatch: 'full'
  },
  {
    path: 'account',
    loadChildren: () => import('./pages/account/account.module').then(
        m => m.AccountModule)
  },
  {
    path: 'support',
    loadChildren: () => import('./pages/support/support.module').then(
        m => m.SupportModule)
```

```
  },
  {
    path: 'login',
    loadChildren: () => import('./pages/login/login.module').then(
        m => m.LoginModule)
  },
  {
    path: 'signup',
    loadChildren: () => import('./pages/signup/signup.module').then(
        m => m.SignUpModule)
  },
  {
    path: 'app',
    loadChildren: () => import('./pages/tabs-page/tabs-page.module')
        .then(m => m.TabsModule)
  },
  {
    path: 'tutorial',
    loadChildren: () => import('./pages/tutorial/tutorial.module')
        .then(m => m.TutorialModule),
    canLoad: [CheckTutorial]
  }
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule {}
```

src/app/app.module.ts

```
import { HttpClientModule } from '@angular/common/http';
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { InAppBrowser } from '@ionic-native/in-app-browser/ngx';
import { SplashScreen } from '@ionic-native/splash-screen/ngx';
import { StatusBar } from '@ionic-native/status-bar/ngx';
import { IonicModule } from '@ionic/angular';
import { IonicStorageModule } from '@ionic/storage';

import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { ServiceWorkerModule } from '@angular/service-worker';
import { environment } from '../environments/environment';
import { FormsModule } from '@angular/forms';
import { LocalNotifications } from
    '@ionic-native/local-notifications/ngx';
import { Device } from '@ionic-native/device/ngx';
import { BLE } from '@ionic-native/ble/ngx';
import { BluetoothSerial } from '@ionic-native/bluetooth-serial/ngx';



@NgModule({
  imports: [
    BrowserModule,
    AppRoutingModule,
    HttpClientModule,
    FormsModule,
    IonicModule.forRoot({
      backButtonText: 'Atr s'
```

```
    }),
    IonicStorageModule.forRoot(),
    ServiceWorkerModule.register('ngsw-worker.js', {
      enabled: environment.production
    })
  ],
  declarations: [AppComponent],
  providers: [
    InAppBrowser,
    SplashScreen,
    StatusBar,
    LocalNotifications,
    Device,
    BLE,
    BluetoothSerial
  ],
  bootstrap: [AppComponent]
})
export class AppModule {}
```

src/app/app.component.ts

```
import { Component, OnInit, ViewEncapsulation } from '@angular/core';
import { Router } from '@angular/router';
import { SwUpdate } from '@angular/service-worker';

import { MenuController, Platform, ToastController } from
    '@ionic/angular';

import { SplashScreen } from '@ionic-native/splash-screen/ngx';
import { StatusBar } from '@ionic-native/status-bar/ngx';
```

```
import { Storage } from '@ionic/storage';

import { UserData } from './providers/user-data';

import { Socket } from './providers/socket';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.scss'],
  encapsulation: ViewEncapsulation.None
})
export class AppComponent implements OnInit {
  appPages = [
    {
      title: 'schedule',
      url: '/app/tabs/schedule',
      icon: 'calendar'
    },
    {
      title: 'Agents',
      url: '/app/tabs/agents',
      icon: 'hardware-chip'
    },
    {
      title: 'User location',
      url: '/app/tabs/user_location',
      icon: 'location'
```

```
    },
    {
      title: 'Chat',
      url: '/app/tabs/chat',
      icon: 'chatbox'
    },
    {
      title: 'About',
      url: '/app/tabs/about',
      icon: 'information-circle'
    }
  ];
  loggedIn = false;
  dark = false;

  constructor(
    private menu: MenuController,
    private platform: Platform,
    private router: Router,
    private splashScreen: SplashScreen,
    private statusBar: StatusBar,
    private storage: Storage,
    private userData: UserData,
    private swUpdate: SwUpdate,
    private toastCtrl: ToastController,
    public socket: Socket
  ) {
    this.initializeApp();
  }
```

```
async ngOnInit() {
  this.checkLoginStatus();
  this.listenForLoginEvents();

  this.swUpdate.available.subscribe(async res => {
    const toast = await this.toastCtrl.create({
      message: 'Update available!',
      position: 'bottom',
      buttons: [
        {
          role: 'cancel',
          text: 'Reload'
        }
      ]
    });

    await toast.present();

    toast
      .onDidDismiss()
      .then(() => this.swUpdate.activateUpdate())
      .then(() => window.location.reload());
  });
}

initializeApp() {
  this.platform.ready().then(() => {
    this.statusBar.styleDefault();
```

```
      this.splashScreen.hide();
    });
  }

  checkLoginStatus() {
    return this.userData.isLoggedIn().then(loggedIn => {
      return this.updateLoggedInStatus(loggedIn);
    });
  }

  updateLoggedInStatus(loggedIn: boolean) {
    setTimeout(() => {
      this.loggedIn = loggedIn;
    }, 300);
    this.storage.get('ion_did_tutorial').then((val) => {
      if (val) {
        if (loggedIn == false) {
          this.router.navigateByUrl('/login');
        }
      }else{
        this.openTutorial();
      }
    });
  }

  listenForLoginEvents() {
    window.addEventListener('user:login', () => {
      this.updateLoggedInStatus(true);
    });
```

```
    window.addEventListener('user:signup', () => {
      this.updateLoggedInStatus(true);
    });

    window.addEventListener('user:logout', () => {
      this.updateLoggedInStatus(false);
    });
  }

  logout() {
    this.userData.logout().then(() => {
      return this.router.navigateByUrl('/login');
    });
  }

  openTutorial() {
    this.menu.enable(false);
    this.storage.set('ion_did_tutorial', false);
    this.router.navigateByUrl('/tutorial');
  }
}
```

src/app/app.component.html

```
<ion-app [class.dark-theme]="dark">
  <ion-split-pane contentId="main-content">

    <ion-menu contentId="main-content">
      <ion-content>
        <ion-list lines="none">
```

```
          <ion-list-header>
            Smart ITL
          </ion-list-header>
          <ion-menu-toggle autoHide="false" *ngFor="let p of appPages;
␣␣␣␣␣␣␣␣␣␣␣␣let i = index">
            <ion-item [routerLink]="p.url" routerLinkActive="selected"
                routerDirection="root" detail="false">
              <ion-icon slot="start" [name]="p.icon + '-outline'">
                </ion-icon>
              <ion-label>
                {{p.title}}
              </ion-label>
            </ion-item>

          </ion-menu-toggle>
        </ion-list>

        <ion-list *ngIf="loggedIn" lines="none">
          <ion-list-header>
            Account
          </ion-list-header>

          <ion-menu-toggle autoHide="false">
            <ion-item routerLink="/account" routerLinkActive="active"
                routerDirection="root" detail="false">
              <ion-icon slot="start" name="person"></ion-icon>
              <ion-label>
                Account
              </ion-label>
```

```
    </ion-item>
  </ion-menu-toggle>

  <ion-menu-toggle autoHide="false">
    <ion-item routerLink="/support" routerLinkActive="active"
        routerDirection="root" detail="false">
      <ion-icon slot="start" name="help"></ion-icon>
      <ion-label>
        Support
      </ion-label>
    </ion-item>
  </ion-menu-toggle>

  <ion-menu-toggle autoHide="false">
    <ion-item button (click)="logout()" detail="false">
      <ion-icon slot="start" name="log-out"></ion-icon>
      <ion-label>
        Logout
      </ion-label>
    </ion-item>
  </ion-menu-toggle>

  <ion-item>
    <ion-icon slot="start" name="moon-outline"></ion-icon>
    <ion-label>
      Dark Mode
    </ion-label>
    <ion-toggle [(ngModel)]="dark"></ion-toggle>
  </ion-item>
```

```
    </ion-list>

    <ion-list *ngIf="!loggedIn" lines="none">
      <ion-list-header>
        Account
      </ion-list-header>

      <ion-menu-toggle autoHide="false">
        <ion-item routerLink="/login" routerLinkActive="active"
            routerDirection="root" detail="false">
          <ion-icon slot="start" name="log-in"></ion-icon>
          <ion-label>
            Login
          </ion-label>
        </ion-item>
      </ion-menu-toggle>

      <ion-menu-toggle autoHide="false">
        <ion-item routerLink="/support" routerLinkActive="active"
            routerDirection="root" detail="false">
          <ion-icon slot="start" name="help"></ion-icon>
          <ion-label>
            Support
          </ion-label>
        </ion-item>
      </ion-menu-toggle>

      <ion-menu-toggle autoHide="false">
```

```
        <ion-item routerLink="/signup" routerLinkActive="active"
            routerDirection="root" detail="false">
          <ion-icon slot="start" name="person-add"></ion-icon>
          <ion-label>
            Signup
          </ion-label>
        </ion-item>
      </ion-menu-toggle>

      <ion-item>
        <ion-icon slot="start" name="moon-outline"></ion-icon>
        <ion-label>
          Dark Mode
        </ion-label>
        <ion-toggle [(ngModel)]="dark"></ion-toggle>
      </ion-item>
    </ion-list>

    <ion-list lines="none">
      <ion-list-header>
        Tutorial
      </ion-list-header>
      <ion-menu-toggle autoHide="false">
        <ion-item button (click)="openTutorial()" detail="false">
          <ion-icon slot="start" name="hammer"></ion-icon>
          <ion-label>Show Tutorial</ion-label>
        </ion-item>
      </ion-menu-toggle>
    </ion-list>
```

```
      </ion-content>
    </ion-menu>

    <ion-router-outlet id="main-content"></ion-router-outlet>

  </ion-split-pane>

</ion-app>
```

src/app/app.component.scss

```
ion-menu ion-content {
  --padding-top: 20px;
  --padding-bottom: 20px;

  --background: var(--ion-item-background, var(--ion-background-color,
    #fff));
}

/* Remove background transitions for switching themes */
ion-menu ion-item {
  --transition: none;
}

ion-item.selected {
  --color: var(--ion-color-primary);
}

/*
 * Material Design Menu
*/
```

```
ion-menu.md ion-list {
  padding: 20px 0;
}

ion-menu.md ion-list-header {
  padding-left: 18px;
  padding-right: 18px;

  text-transform: uppercase;
  letter-spacing: .1em;
  font-weight: 450;
}

ion-menu.md ion-item {
  --padding-start: 18px;

  margin-right: 10px;

  border-radius: 0 50px 50px 0;

  font-weight: 500;
}

ion-menu.md ion-item.selected {
  --background: rgba(var(--ion-color-primary-rgb), 0.14);
}

ion-menu.md ion-item.selected ion-icon {
  color: var(--ion-color-primary);
```

```
}


ion-menu.md ion-list-header,
ion-menu.md ion-item ion-icon {
  color: var(--ion-color-step-650, #5f6368);
}

ion-menu.md ion-list:not(:last-of-type) {
  border-bottom: 1px solid var(--ion-color-step-150, #d7d8da);
}



/*
 * iOS Menu
*/
ion-menu.ios ion-list-header {
  padding-left: 16px;
  padding-right: 16px;

  margin-bottom: 8px;
}

ion-menu.ios ion-list {
  padding: 20px 0 0;
}

ion-menu.ios ion-item {
  --padding-start: 16px;
  --min-height: 50px;
```

```
}


ion-menu.ios ion-item ion-icon {
  font-size: 24px;
  color: #73849a;
}

ion-menu.ios ion-item.selected ion-icon {
  color: var(--ion-color-primary);
}
```

src/app/providers/socket.ts

```
import { Injectable } from '@angular/core';
import { io } from 'socket.io-client';
import { environment } from '../../environments/environment';
import { AlertController, ToastController } from '@ionic/angular';
import { UserData } from './user-data';
import { Scan } from './scan';

@Injectable({
    providedIn: 'root'
})
export class Socket {
    socket: any;
    scanning = false;
    username = null;

    constructor(
        private toastCtrl: ToastController,
        private scan: Scan,
```

```
        private userData: UserData,
        public alertController: AlertController
    ) { }

    start() {
        console.log(environment.urlServerSocket)
        this.socket = io(environment.urlServerSocket, {
            transports: ['websocket']
        })
        this.startScaning()
        console.log('empieza conexi n');
        this.socket.on('connect', () => {
            this.userData.getUsername().then(username => {
                this.socket.emit('join', { id: username, type: 1 }, (
                    error) => {
                    if (error) { } else { }
                });
            });
        });

        this.socket.on("disconnect", () => {
            console.log('disconnected from server');
        });

        this.socket.on('connect_error', (error) => {
            console.log('connect_error due to', error.message);
            this.scanning = false;
        });
```

```
        this.socket.on('setAgentsOnline', (list) => {
            if (list[0]) {
                console.log(list)
                console.log("Lista␣llena")
                this.userData.setAgentsOnlineList(list).then(() => {
                    this.scanning = true;
                })
            } else {
                console.log("Lista␣vacia")
                this.scanning = false;
            }
        });

        this.socket.on('setAgents', (list) => {
            console.log(list)
            this.userData.setAgentsList(list).then(() => {
            })
        });

        this.socket.on('setLocationUser', (list) => {
            console.log(list)
            this.userData.setUsersLocationList(list).then(() => {
            })
        });
    }


    startScaning() {
        setInterval(() => {
```

```
            //this.userData.getAgentOnlineList()
            if (this.scanning) {
                //this.socket.emit('getAgentsOnline')
                //this.socket.emit('getAgents')
                this.scan.start();
            }
        }, .5 * 60000);
    }

    here(agentId: String) {
        this.socket.emit('here', agentId)
    }
}
```

src/app/providers/scan.ts

```
import { Injectable, NgZone } from '@angular/core';
import { BLE } from '@ionic-native/ble/ngx';
import { AlertController } from '@ionic/angular';
import { UserData } from './user-data';
import { BluetoothSerial } from '@ionic-native/bluetooth-serial/ngx';

@Injectable({
    providedIn: 'root'
})
export class Scan {
    devices: any[] = [];
    socket = null

    constructor(
        private ble: BLE,
```

```
        private ngZone: NgZone,
        private userData: UserData,
        public alertController: AlertController,
        public bluetoothSerial: BluetoothSerial
    ) { }

    start() {
        this.devices = [];
        this.startScanning();

    }

    startScanning() {
        this.bluetoothSerial.discoverUnpaired().then((success) => {
            success.forEach((value, key) => {
                this.onDeviceDiscovered(value);
            });
        }, (err) => { console.log(err); });
    }

    onDeviceDiscovered(device) {
        this.ngZone.run(() => {
            this.devices.push(device);
            this.checkAgentList(device)
        });
    }

    checkAgentList(device) {
        this.userData.getAgentOnlineList().then((list)=>{
```

```
            list.forEach((agentId,index)=>{
                if (agentId == device.id) {
                    this.socket.here(agentId);
                    return;
                }
            })
        })
    }
}
```

src/app/pages/agent-list/agent-list.ts

```
import { Component, ViewChild } from '@angular/core';
import { UserData } from '../../providers/user-data';

@Component({
  selector: 'page-agent-list',
  templateUrl: 'agent-list.html',
  styleUrls: ['./agent-list.scss'],
})
export class AgentListPage {
  agents: any[] = [];
  constructor(private userData: UserData) { }

  ionViewDidEnter() {
    this.cargarAgentes();
  }

  doRefresh(event) {
    console.log('Begin async operation');
    this.cargarAgentes();
```

```
    setTimeout(() => {
      console.log('Async operation has ended');
      event.target.complete();
    }, 2000);
  }

  cargarAgentes() {
    this.userData.getAgentsList().then((list: any[]) => {
      const agents = list;
      console.log(agents)
      //Ordenar por edificio y aula
      agents.sort(function (x, y) {
        var n = x.build.localeCompare(y.build);
        if (n !== 0) {
          return n;
        }
        return x.location.localeCompare(y.location);
      });
      var build = agents.map(function (agent) {
        return agent.build; });
      var sorted = build.sort();
      var builds = sorted.filter(function (value, index) {
        return value !== sorted[index + 1];
      });
      var agentsOnline = []
      var agentsOffline = []
      for (var i = 0; i < builds.length; i++) {
        var agentsTemp = []
        var agentsOnlineTemp = []
```

```
        var agentsOfflineTemp = []
        agentsTemp = agents.filter(function (data) {
            return data.build == builds[i] })
        agentsOnlineTemp = agentsTemp.filter(function (data) {
            return data.status == 'online' })
        agentsOfflineTemp = agentsTemp.filter(function (data) {
            return data.status == 'offline' })
        if (agentsOnlineTemp.length > 0) {
          agentsOnline.push({ 'build': builds[i],
            'agents': agentsOnlineTemp })
        }
        if (agentsOfflineTemp.length > 0) {
          agentsOffline.push({ 'build': builds[i],
            'agents': agentsOfflineTemp })
        }
      }
      this.agents = [];
      this.agents.push({'status': 'Online','builds': agentsOnline })
      this.agents.push({'status': 'Offline','builds': agentsOffline })
      console.log(this.agents)
    })
  }
}
```

src/app/pages/agent-detail/agent-detail.ts

```
import { Component } from '@angular/core';
import { ActivatedRoute } from '@angular/router';
import { ConferenceData } from '../../providers/conference-data';
import { ActionSheetController } from '@ionic/angular';
import { InAppBrowser } from '@ionic-native/in-app-browser/ngx';
```

```
import { UserData } from '../../providers/user-data';
import * as moment from 'moment';

@Component({
  selector: 'page-agent-detail',
  templateUrl: 'agent-detail.html',
  styleUrls: ['./agent-detail.scss'],
})
export class AgentDetailPage {
  agent: any[] = [];
  dates: any[] = [];

  constructor(
    private dataProvider: ConferenceData,
    private route: ActivatedRoute,
    public actionSheetCtrl: ActionSheetController,
    public confData: ConferenceData,
    public inAppBrowser: InAppBrowser,
    private userData: UserData
  ) { }

  ionViewWillEnter() {
    this.cargarDatos();
  }

  cargarDatos() {
    this.userData.getAgentsList().then((data: any[]) => {
      const agentId = this.route.snapshot.paramMap.get('agentId');
      //this.agent = data.filter(function (data) {
```

```
        return data._id == speakerId })[0];
      const agent = data.find(function (data) {
        return data._id == agentId });
      agent['dates'].sort((x, y) => new Date(y.date).getTime() -
        new Date(x.date).getTime());
      console.log(agent);
      for (var i in agent['dates']) {
        agent['dates'][i]['date'] = moment.utc(
            agent['dates'][i]['date']).format('LLL');
      }
      console.log(agent);
      this.agent = agent;
      console.log(this.agent);
    })
  }
}
```

src/app/pages/user-location-list/user-location-list.ts

```
import { Component } from '@angular/core';
import { UserData } from '../../providers/user-data';

@Component({
  selector: 'page-user-location-list',
  templateUrl: 'user-location-list.html',
  styleUrls: ['./user-location-list.scss'],
})
export class UserLocationListPage {
  usersLocation: any[] = [];

  constructor(private userData: UserData) { }
```

```
  ionViewDidEnter() {
this.cargarUsuarios();
  }
  cargarUsuarios(){
    this.userData.getUsersLocationList().then((list: any[]) => {
      const usersLocation = list;
      usersLocation.sort((x, y) => x.user.localeCompare(y.user));
      var user = usersLocation.map(function (data) {
        return data.user; });
      var sorted = user.sort();
      var users = sorted.filter(function (value, index) {
        return value !== sorted[index + 1];
      });
      var userLocation = []
      for (var i = 0; i < users.length; i++) {
        var userLocationTemp = []
        userLocationTemp = usersLocation.filter(function (data) {
            return data.user == users[i] })
        userLocation.push({ 'user': users[i],
            'locations':userLocationTemp })
      }
      console.log(userLocation)
      this.usersLocation = userLocation
    })
  }
}
```

src/app/pages/user-location-detail/user-location-detail.ts

```
import { Component } from '@angular/core';
```

```
import { ActivatedRoute } from '@angular/router';
import { ConferenceData } from '../../providers/conference-data';
import { ActionSheetController } from '@ionic/angular';
import { InAppBrowser } from '@ionic-native/in-app-browser/ngx';
import { UserData } from '../../providers/user-data';
import * as moment from 'moment';

@Component({
  selector: 'page-user-location-detail',
  templateUrl: 'user-location-detail.html',
  styleUrls: ['./user-location-detail.scss'],
})
export class UserLocationDetailPage {
  user: any[] = [];

  constructor(
    private dataProvider: ConferenceData,
    private route: ActivatedRoute,
    public actionSheetCtrl: ActionSheetController,
    public confData: ConferenceData,
    public inAppBrowser: InAppBrowser,
    public userData: UserData
  ) {}

  ionViewWillEnter() {
    this.cargarDatos();
  }

  cargarDatos() {
```

```
    this.userData.getUsersLocationList().then((data: any[]) => {
      const userLocationId = this.route.snapshot.paramMap.get(
        'userLocationId');
      const user = data.filter(function (data) {
        return data.user == userLocationId });
      user.sort((x, y) => new Date(y.updatedAt).getTime() -
        new Date(x.updatedAt).getTime());
      for (var i in user) {
        user[i]['updatedAt'] = moment(
            user[i]['updatedAt']).format('LLL');
      }
      console.log(user);
      this.user = [{user:userLocationId, datos:user}];
      this.user = this.user[0]
      console.log(this.user)
    })
  }
}
```