



**iHome**

***TESINA QUE PRESENTA***

**Angel de Jesus Baños Tellez, Adrian Perez Jimenez, Al Farias  
Leyva, Dulce Yadira Salvador Antonio, Derek Sesni Carreño,  
Jesus Dominguez Ramirez, Obed Guzman Flores, Jose Arturo  
Garcia Gonzalez.**

***TÉCNICO SUPERIOR UNIVERSITARIO EN  
DESARROLLO DE SOFTWARE  
MULTIPLATAFORMA***

**MAYO, 2025**



# DEDICATORIA

Texto a desarrollar: el siguiente es un ejemplo de un reconocimiento, permitiendo al autor mencionar quienes hicieron posible su éxito. El texto no debe exceder de una hoja, debe ser breve, concreto.

## **Ejemplo:**

*Con cariño y gratitud, dedico este trabajo a todas las personas que han sido parte fundamental de mi crecimiento personal y académico, pero sobre todo a mi familia, por su amor incondicional, su apoyo constante y por creer en mí incluso en los momentos en que yo dudé. Cada palabra de aliento y cada gesto de comprensión han sido mi mayor fortaleza.*

*Con afecto,*

*[Tu nombre]*

# AGRADECIMIENTOS

Texto a desarrollar enfocado a agradecer el esfuerzo de las personas involucradas en el proyecto. El texto debe contemplar aspectos relaciones al agradecer el trabajo de los colaboradores, se puede incluir a los asesores académicos y empresariales, familiares o amigos.

Ejemplo:

*Al concluir esta tesina, deseo expresar mi más sincero agradecimiento a todas las personas que, de una u otra forma, contribuyeron a la realización de este trabajo.*

*A mi asesor/a, [Nombre del asesor/a], por su paciencia, guía y valiosas sugerencias, que fueron fundamentales para la construcción y desarrollo de esta investigación. Su compromiso y conocimientos han sido una fuente de inspiración.*

*A mis docentes, por brindarme las herramientas académicas y motivarme a seguir aprendiendo con pasión y dedicación.*

*A mi familia, por ser mi mayor motivación para seguir adelante.*

*A todas aquellas personas que, de manera directa o indirecta, contribuyeron a la culminación de este trabajo, les expreso mi más profundo reconocimiento a esa gran labor.*

*Con gratitud,*

*[Tu nombre]*

## **Resumen**

El resumen no debe exceder las 350 palabras y no puede contener ecuaciones, figuras, tablas o referencias. Debe indicar de manera concisa lo siguiente:

- ¿Qué se hizo?
- ¿Cómo se hizo?
- Los resultados principales y su importancia.

Palabras Clave – El autor debe proporcionar hasta 5 palabras clave (en orden alfabético) para ayudar a identificar los principales temas del artículo.

Se debe hacer referencia al tesoro de palabras clave de indexación IEEE antes de seleccionar las palabras clave para asegurarse de que las palabras seleccionadas son aceptables.

## **Abstract**

The sections are also dedicated to the preparation of recognitions and references. The summary should not exceed 350 words and can not contain equations, figures, tables or references. It should indicate concisely:

- What was done?
- How it was done?
- The main results and their importance.

Keywords - The author must provide up to 5 key words (in alphabetical order) to help identify the main topics of the article. The IEEE Indexing Thesaurus should be referenced before selecting the keywords to ensure that the selected words are acceptable.

## **Tabla de Contenidos**

Capítulo 1 Introducción e información general	1
Planteamiento del Problema	1
Objetivo General del Proyecto.	2
Objetivos Específicos.	2
Justificación del Proyecto	3
Limitaciones y Alcances.	3
Capítulo 2. Marco Teórico y Contextual	5
Marco Teórico	5
Marco Contextual	5
Capítulo 3 Metodología y Desarrollo	5
Capítulo 4 Resultados y Conclusiones	6
Resultados	7
Conclusiones	7
Recomendaciones	7
Lista de referencias	8
Apéndice	9
Vita	10

<b>Lista de tablas</b>	
<a href="#">Tabla 1</a> . Materiales IoT escalado real.....	29
<a href="#">Tabla 2</a> . Materiales IoT Maquetado.....	29



## Lista de figuras

<a href="#">Figura 1.</a>	instalación de dependencias.....	22
<a href="#">Figura 2.</a>	levantar backend en fase desarrollo.....	23
<a href="#">Figura 3.</a>	Configuración de Preferencias Arduino.....	26
<a href="#">Figura 4.</a>	Instalación de ESP8266.....	26
<a href="#">Figura 5.</a>	Selección de Puertos Wemos DI.....	27
<a href="#">Figura 6.</a>	Buyer Person.....	29
<a href="#">Figura 7.</a>	Modelo Canvas.....	30
<a href="#">Figura 8.</a>	Matriz de Responsabilidades.....	32
<a href="#">Figura 9.</a>	Cronograma de Actividades.....	33
<a href="#">Figura 10.</a>	Diagrama de Componentes.....	33
<a href="#">Figura 11.</a>	Diagrama de Arquitectura.....	35
<a href="#">Figura 12.</a>	Diagrama de Conexiones IoT.....	42
<a href="#">Figura 13.</a>	Configuración de Preferencias Arduino.....	42
<a href="#">Figura 14.</a>	Diagrama de Conexiones IoT.....	43
<a href="#">Figura 15.</a>	Diagrama de Conexiones IoT.....	43
<a href="#">Figura 16.</a>	Diagrama de Conexiones IoT.....	44
<a href="#">Figura 17.</a>	Lista de Componentes y Propósitos.....	30
<a href="#">Figura 18.</a>	Kitchen esquema mongoose.....	47
<a href="#">Figura 19.</a>	Conexión a MongoDB .....	48
<a href="#">Figura 20.</a>	WireFrames Móviles.....	72
<a href="#">Figura 21</a>	Mockups Móviles Web.....	72



## Capítulo 1

### Introducción e información general

La automatización del hogar mediante el internet de las cosas IoT ha cambiado la forma en que los usuarios interactúan con sus dispositivos domésticos. La integración de sensores, actuadores y sistemas de control conectados a internet permite una mayor eficiencia energética, seguridad y confort en los hogares.

El presente marco teórico explorará las bases de IoT aplicándola al hogar inteligente, los protocolos de comunicación más utilizados, las tecnologías clave y el impacto de la automatización en la vida cotidiana.

## **Planteamiento del Problema**

La automatización del hogar ha avanzado significativamente, pero sigue enfrentando barreras como la falta de personalización, la incompatibilidad entre dispositivos y los altos costos de implementación. Además, muchas soluciones dependen de servidores externos, generando preocupaciones sobre seguridad y privacidad.

La ausencia de estándares y la limitada accesibilidad dificultan que los usuarios adapten sus hogares de forma flexible y escalable. Por ello, es necesario desarrollar un sistema eficiente, seguro y asequible que integre IoT y computación en la nube, permitiendo una gestión centralizada y adaptable a diversas necesidades.

**Objetivo General del Proyecto.**

Desarrollar e implementar una solución integral de software para la automatización general de residencias, que permita a los usuarios controlar y monitorear de manera eficiente y segura diversas funciones del hogar, a través de una aplicación web móvil en un entorno multiplataforma.

## **Objetivos Específicos.**

1. Diseñar e implementar una interfaz de usuario amigable y accesible a través de una aplicación web, validando su usabilidad mediante pruebas con al menos 10 usuarios y garantizando un tiempo de respuesta corto.
2. Desarrollar un sistema de control centralizado para la operación automatizada de puertas, ventanas y luces, asegurando una reducción del consumo energético esto en comparación con otro sistema manual en un periodo establecido de prueba.
3. Implementar un sistema de gestión climática inteligente que se ajuste automáticamente la temperatura y humedad del ambiente dentro de la residencia esto basado en las necesidades del usuario y/o comunidades.
4. Instalar y configurar sensores de humedad, temperatura, presencia y distancia para monitorear en tiempo real las condiciones del entorno y la seguridad del hogar.
5. Desarrollar una arquitectura de software segura y escalable que soporte la integración y el correcto manejo eficiente de los diferentes dispositivos y sensores involucrados del hogar.
6. Realizar pruebas piloto para validar la funcionalidad, usabilidad y fiabilidad del sistema de automatización residencial, ajustando y mejorando el sistema basado en el feedback de los usuarios.

## **Justificación del Proyecto.**

La automatización del hogar representa un avance clave en la integración de tecnología en la vida cotidiana, ofreciendo **mayor comodidad, seguridad y eficiencia energética**. Sin embargo, su adopción sigue limitada por altos costos, falta de compatibilidad y preocupaciones sobre privacidad.

Este proyecto busca superar esas barreras mediante el desarrollo de una **solución innovadora, escalable y accesible**, basada en **IoT y computación en la nube**. Al optimizar el control y monitoreo del hogar, no solo mejora la experiencia del usuario, sino que también impulsa el uso eficiente de los recursos, fomentando un entorno inteligente, seguro y adaptable a diversas necesidades.

## **Limitaciones y Alcances.**

Algunas de las limitantes es la conectividad wi-fi, el sistema dependerá de una conexión a internet estable y la cobertura de red en el hogar. La inversión inicial es alta, aunque el ahorro energético es a largo plazo.

- El sistema dependerá de una conexión a internet estable.
- En áreas rurales o con mala conectividad esto puede ser un problema.
- Recuperación de inversión a largo plazo.
- El costo inicial puede ser alto.

## **Alcances**

El sistema propuesto busca desarrollar una solución para la automatización y monitoreo del hogar, permitiendo a los usuarios tener un mayor control de su entorno

- Automatización de dispositivos del hogar, el sistema controlará puertas, ventanas, luces, temperatura.
- Monitoreo en tiempo real e implementación de sensores de movimiento integrados en un foco el cual se encenderá al detectar presencia.
- Interfaz de usuario, se desarrollará una aplicación web/móvil intuitiva para interactuar con los dispositivos anteriormente mencionados.
- Escalabilidad, con la posibilidad de expandir el sistema con nuevos dispositivos esto puede variar según las necesidades del usuario.



## Capítulo 2. MARCO TEÓRICO Y MARCO CONTEXTUAL.

### 1. Marco Teórico

1.1. La domótica ha experimentado una evolución constante desde sus inicios en la automatización de las tareas domésticas básicas hasta sistemas avanzados que integran inteligencia artificial. En las décadas de 1970 y 1980, los sistemas de control remoto mediante cableado eran la norma, posteriormente con la llegada del protocolo x10 en 1975. En la actualidad, la proliferación del internet de las cosas (IoT) ha permitido el desarrollo de los hogares inteligentes, donde los dispositivos pueden comunicarse entre sí y con el usuario a través del internet [1].

Según Statista (2023), se prevé que el mercado global de hogares inteligentes alcance un valor aproximado de \$313.95 mil millones USD para el año 2027, impulsado por el crecimiento de la adopción de dispositivos IoT [2]. Esta tecnología permite optimizar el consumo energético, reducir costos y mejorando la seguridad de los hogares. Investigaciones recientes han demostrado que la automatización del hogar puede reducir el consumo energético hasta en un 30%, esto derivado de la optimización de la iluminación, climatización, etc [3].

### 1.2. Bases Teóricas

El internet de las cosas (IoT) es un paradigma tecnológico que permite la interconexión de dispositivos físicos a través de internet facilitando la recopilación y análisis de datos en tiempo real. En el contexto de los hogares inteligentes, los dispositivos IoT incluyen sensores,

actuadores, controladores y plataformas de gestión que optimizan el uso de los recursos y mejoran la experiencia de usuario [4].

Para garantizar la conectividad entre los dispositivos de un hogar inteligente, se utilizan diversos protocolos de comunicación, cada uno con características específicas según las necesidades de el sistema:

- **WebSockets:** Permite conexiones bidireccionales en tiempo real entre los dispositivos IoT y los servidores, lo que facilita la monitorización y control remoto en tiempo real [5].

## **2. Diferencial y aporte al proyecto**

A diferencia de otros sistemas comerciales, en este proyecto se busca ofrecer una solución modular y escalable que permita a los usuarios personalizar su sistema según sus necesidades específicas. Esto se logrará mediante el uso de WebSockets para la comunicación en tiempo real, garantizando la interoperabilidad entre dispositivos de diferentes fabricantes. Además, se pondrá un énfasis en la seguridad de la información mediante cifrado de datos y autenticación para evitar accesos no autorizados [6].

## **3. Metodologías y Tecnologías**

### **3.1. Domótica**

La domótica es el conjunto de tecnologías aplicadas al control y automatización inteligente de una vivienda, edificio u otro espacio, con el fin de mejorar la comodidad y eficiencia energética. A través de sistemas electrónicos y de software. [7].

## ¿Cómo se implementa la domótica en la automatización de las cosas?

La implementación de la domótica en la automatización del hogar se basa en la integración de diversos dispositivos. Esto se logra mediante:

- **Sensores:** detectan los cambios en el entorno, temperatura, luz, movimiento o humedad.
- **Actuadores:** dispositivos que ejecutan acciones en respuesta a órdenes recibidas como el encender o abrir puertas.
- **Controladores:** procesan la información de los sensores y envían los comandos necesarios a los actuadores.
- **Protocolos de comunicación:** permiten la interconexión de dispositivos, ya sea mediante cableado o inalámbrica (WebSockets).
- **Interfaz de usuario:** aplicaciones web o móviles que nos permitirán a los usuarios monitorear y controlar nuestro hogar de manera remota y segura.

### Desafíos de la domótica

A pesar de sus grandes beneficios esta enfrenta algunos de los retos que enfrenta son:

- Poca compatibilidad.
- Seguridad y Privacidad esto si se manejan de manera incorrecta.
- Costo inicial debido a que los costos pueden ser elevados.

#### 4. Backend

¿Qué es el backend?

El backend es parte del desarrollo de software encargada de la lógica del negocio, la gestión de bases de datos, la autenticación de usuarios y la comunicación con el frontend. Se ejecuta en un servidor y proporciona servicios y datos necesarios para que el frontend funcione de manera correcta. En el contexto de la domótica, el backend es responsable de procesar toda la información recibida de todos los sensores, actuadores entre otros esto con el fin de permitir la interacción remota con el sistema [8]. En este proyecto se implementó un backend con Node.js con Express, debido a su eficiencia en el manejo de las peticiones asíncronas y su compatibilidad con WebSockets, lo que permite una comunicación en tiempo real.

#### 5. Framework

¿Qué es un framework?

Un framework es un conjunto de herramientas, bibliotecas y buenas prácticas que proporcionan una estructura predefinida para el desarrollo de software. Su propósito es facilitar la creación de aplicaciones al ofrecer código reutilizable, componentes modulares y una arquitectura establecida esto con el fin de reducir el tiempo y esfuerzo para su implementación.

Existen diferentes tipos de framework:

- **Frameworks frontend:** Angular, React, Vue.js.
- **Frameworks backend:** Express.js, Django, Spring Boot.
- **Frameworks móviles:** Flutter, React Native, Ionic.

Solo por mencionar algunos, en este caso se aplicó para el proyecto Angular.

## 6. Angular como framework

Angular es un framework de desarrollo web basado en TypeScript y desarrollado por Google. El principal objetivo es facilitar la creación de aplicaciones web de una sola página (SPA, *Single Page Application*) mediante una arquitectura basada en componentes [9].

Entre sus características destacan:

- Modularidad

- Data Binding

- Inyección de dependencias

- Ruteo dinámico

- Compatibilidad con herramientas modernas

En este proyecto, Angular se utilizó para el desarrollo del **frontend** del sistema domótica, aprovechando su estructura modular y la gran compatibilidad de herramientas como Tailwind CSS para el diseño y Highcharts para la visualización de datos.

## 7. Bases de Datos

Una base de datos es un sistema organizado para recopilar, almacenar y manejar grandes volúmenes de datos. Permite la fácil consulta, inserción, modificación y eliminación de la información de manera eficiente y segura. Las bases de datos pueden ser **relacionales o no relacionales** y estas son fundamentales en la gran mayoría de aplicaciones modernas.

Las bases de datos no relacionales son más flexibles, permitiendo almacenamiento de datos de manera no estructurada o en un formato más libre,

como documentos o pares clave-valor. Estas bases de datos son útiles para gestionar grandes cantidades de datos distribuidos y variados, y se emplean comúnmente en aplicaciones que requieren alta escalabilidad o no siguen un esquema fijo de datos [10].

### **Tecnologías empleadas**

El desarrollo del backend se basó principalmente en una arquitectura ligera, escalable y segura, utilizando las siguientes tecnologías y herramientas:

- Lenguaje de programación y entorno de producción

**Node.js:** Plataforma basada en JavaScript para ejecutar código en el servidor de manera asíncrona.

- Comunicación en tiempo real

**WebSockets,** esto con el fin de permitir la comunicación bidireccional en tiempo real entre los dispositivos IoT y el backend, optimizando la latencia y mejorando la UX [11].

- Seguridad

**bcryptjs:** Se usa para cifrar contraseñas antes de almacenarlas en la base de datos.

**jsonwebtoken (JWT):** Implementado para la autenticación de usuarios mediante tokens de acceso.

**dotenv:** Manejo de variables de entorno para mejorar la seguridad y evitar exponer credenciales en el código.

**CORS:** Configurado para permitir solicitudes solo desde orígenes autorizados, evitando ataques de tipo CSRF.

**Morgan:** Middleware para registrar las solicitudes HTTP esto para detectar posibles problemas de rendimiento y seguridad.

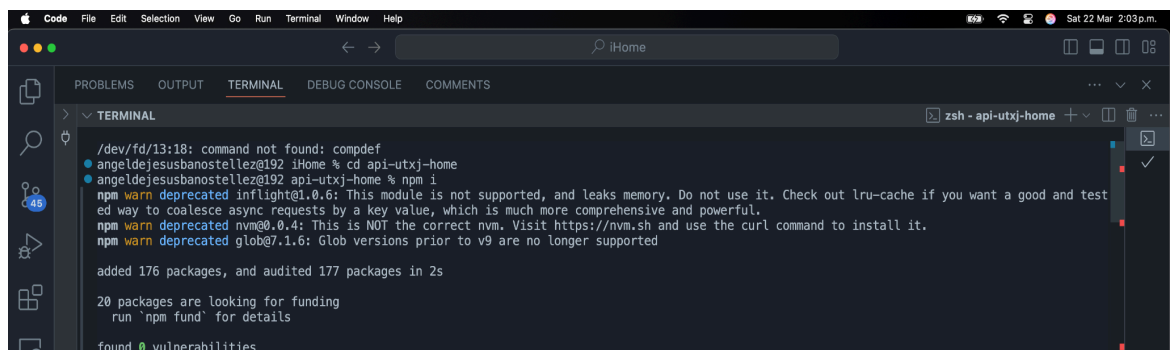
**Swagger-jsdoc & Swagger-UI-Express:** Utilizados para documentar el API y facilitar el consumo de los servicios por parte de los desarrolladores.

### Gestión de Dependencias

**npm:** Administrador de paquetes para tener una gestión dentro del proyecto.

**Nodemon:** Herramienta en desarrollo para reiniciar el servidor automáticamente cuando hay cambios en el código fuente.

Para la instalación de dependencias dentro del proyecto se debe ejecutar el comando **npm i** esto descargara las dependencias necesarias para el correcto funcionamiento.

A screenshot of a terminal window within a code editor. The terminal shows the execution of the 'npm i' command. It starts with a prompt from 'angeldejesusbanostellez@192 iHome'. The command 'cd api-utxj-home' is entered, followed by 'npm i'. The output shows several deprecation warnings: 'inflight@1.0.6: This module is not supported, and leaks memory. Do not use it. Check out lru-cache if you want a good and tested way to coalesce async requests by a key value, which is much more comprehensive and powerful.', 'npm@0.0.4: This is NOT the correct npm. Visit https://npm.sh and use the curl command to install it.', and 'glob@7.1.6: Glob versions prior to v9 are no longer supported'. The terminal then reports 'added 176 packages, and audited 177 packages in 2s', '20 packages are looking for funding', and 'run `npm fund` for details'. Finally, it states 'found 0 vulnerabilities'. The terminal window has tabs for 'PROBLEMS', 'OUTPUT', 'TERMINAL', 'DEBUG CONSOLE', and 'COMMENTS'. The 'TERMINAL' tab is active. The code editor's interface includes a menu bar with 'Code', 'File', 'Edit', 'Selection', 'View', 'Go', 'Run', 'Terminal', 'Window', and 'Help'. The status bar at the bottom shows 'Sat 22 Mar 2:03 p.m.'.

*Figura 1: instalación de dependencias*

Posteriormente al finalizar la instalación de las dependencias del proyecto debemos ejecutar el script previamente configurado en el package.json.

```
angeldejesusbanostellez@192 api-utxj-home % npm run start:dev

> api-utxj-home@1.0.0 start:dev
> nodemon src/main.js

[nodemon] 3.1.0
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node src/main.js`
CONNECTION_DB: undefined
CONNECTION_DB_LOCAL: mongodb://127.0.0.1:27017/iHome

#####
Express API REST Application is running and listening on:
  Local: http://localhost:3000
  Documentación Swagger en http://localhost:3000/api-docs
  Railway: https://api-utxj-home.up.railway.app/
#####

#####

Mongo Database Successfully Connected

#####
```

**Figura 2:** *npm run start:dev para iniciar el backend fase desarrollo*

## 8. IoT (Internet de las Cosas)

¿Qué es IoT y su relación con la domótica?

El Internet de las Cosas (IoT) se refiere a la red de dispositivos físicos que se conectan a internet y pueden compartir datos entre sí. Estos dispositivos permiten interactuar con ellos de manera remota, como en el caso de electrodomésticos, sistemas de seguridad, y más. En el contexto de la domótica, IoT desempeña un papel fundamental, ya que permite la automatización de las viviendas mediante sensores y dispositivos conectados, que pueden ser controlados desde cualquier lugar a través de internet (Bahga & Madiseti, 2014) [12].



## **Listado de sensores, actuadores y controladores**

**Sensores:** Dispositivos que recogen información del entorno, como temperatura, humedad, o movimiento. Estos datos se transmiten al sistema para su procesamiento.

**Actuadores:** Son dispositivos que ejecutan acciones físicas en respuesta a las señales provenientes de los sensores. Ejemplos incluyen luces inteligentes, cerraduras electrónicas, y termostatos.

**Controladores:** Son las plataformas o dispositivos encargados de gestionar los datos de los sensores y enviar órdenes a los actuadores.

Pueden ser aplicaciones basadas en la nube o sistemas locales.

Explicación de cada uno y su uso en el sistema.

**Sensores de temperatura:** Se utilizan para controlar el sistema de calefacción o aire acondicionado dentro de la casa, ajustándose a las condiciones ambientales.

**Sensores de movimiento:** Activan o desactivan dispositivos, como luces, dependiendo de la detección de movimiento en una zona específica.

**Actuadores de control de clima:** Dispositivos como termostatos inteligentes ajustan la temperatura según las preferencias del usuario y las condiciones del entorno.

Listado de materiales para el desarrollo del prototipo en escala real, incluyendo detalles específicos para su compra.

Materiales IoT			
Categoría	Componente	Precio	Enlace
puerta de garaje	ABREPUERTAS MERIK 511M	\$6200 MXN	<a href="https://www.homedepot.com.mx/p/merik-abrepuestas-merik-511m-511m-208404">https://www.homedepot.com.mx/p/merik-abrepuestas-merik-511m-511m-208404</a>
ventanas	Sliding Window Motor Electric Manual	\$2991 MXN	<a href="https://www.aliexpress.com/ii/3256802438334018.html?gatewayAdapt=es p2glo4itemAdapt">https://www.aliexpress.com/ii/3256802438334018.html?gatewayAdapt=es p2glo4itemAdapt</a>
sensor de temperatura	HoneyWell Detector. de Temperatura	\$493 MXN	<a href="https://www.cyberpuerta.mx/Seguridad-y-Vigilancia/Sistemas-de-Alarma/Detectores-y-Sensores/Detectores-de-Temperatura/Honeywell-Detector-de-Temperatura-Fija-502-57-C-Alambrico-Blanco.html?srsltid=AfmB0ooPTph3y2GRdx8kF-CLT8YyMTGr42n42Nz9dD562i9Zf3ZyZWbglpl">https://www.cyberpuerta.mx/Seguridad-y-Vigilancia/Sistemas-de-Alarma/Detectores-y-Sensores/Detectores-de-Temperatura/Honeywell-Detector-de-Temperatura-Fija-502-57-C-Alambrico-Blanco.html?srsltid=AfmB0ooPTph3y2GRdx8kF-CLT8YyMTGr42n42Nz9dD562i9Zf3ZyZWbglpl</a>
sensor de humedad	HoneyWell Sensor de Humedad	\$4989 MXN	<a href="https://www.cyberpuerta.mx/Seguridad-y-Vigilancia/Sistemas-de-Alarma/Detectores-y-Sensores/Sensores-de-Humedad/Honeywell-Sensor-de-Humedad-y-Temperatura-H77-Alambrico-Gris.html">https://www.cyberpuerta.mx/Seguridad-y-Vigilancia/Sistemas-de-Alarma/Detectores-y-Sensores/Sensores-de-Humedad/Honeywell-Sensor-de-Humedad-y-Temperatura-H77-Alambrico-Gris.html</a>
sensor de movimiento	Honeywell Sensor de Movimiento PIR de Montaje en Pared	\$418 MXN	<a href="https://www.cyberpuerta.mx/Seguridad-y-Vigilancia/Sistemas-de-Alarma/Detectores-y-Sensores/Sensores-de-Movimiento/Honeywell-Sensor-de-Movimiento-PIR-de-Montaje-en-Pared-IS335T-Alambrico-Anti-Pet-12-Metros-Blanco.html">https://www.cyberpuerta.mx/Seguridad-y-Vigilancia/Sistemas-de-Alarma/Detectores-y-Sensores/Sensores-de-Movimiento/Honeywell-Sensor-de-Movimiento-PIR-de-Montaje-en-Pared-IS335T-Alambrico-Anti-Pet-12-Metros-Blanco.html</a>
sensor de agua	Winland Sensor para Nivel de Agua W-S-U	\$829 MXN	<a href="https://www.cyberpuerta.mx/Seguridad-y-Vigilancia/Sistemas-de-Ventilación: Alarma/Detectores-y-Sensores/Sensores-de-Humedad/Winland-Sensor-para-Nivel-de-Agua-W-S-U-Alambrico-Blanco.html">https://www.cyberpuerta.mx/Seguridad-y-Vigilancia/Sistemas-de-Ventilación: Alarma/Detectores-y-Sensores/Sensores-de-Humedad/Winland-Sensor-para-Nivel-de-Agua-W-S-U-Alambrico-Blanco.html</a>
bomba de agua	Armstrong Water Pump	\$4724 MXN	<a href="https://www.amazon.com.mx/Armstrong-bombas-223-305-Astro-circulaci%C3%B3n/dp/B00QW29XDU">https://www.amazon.com.mx/Armstrong-bombas-223-305-Astro-circulaci%C3%B3n/dp/B00QW29XDU</a>
sensor de luz	Sensor Movimiento y Luz Foco Casa Led	\$699 MXN	<a href="https://articulo.mercadolibre.com.mx/MLM-785906859-4-sensor-movimiento-y-luz-foco-casa-led-110v-a-240-fotocelda-...JM">https://articulo.mercadolibre.com.mx/MLM-785906859-4-sensor-movimiento-y-luz-foco-casa-led-110v-a-240-fotocelda-...JM</a>
aire acondicionado	Evaporador Carrier Aire acondicionado	\$1400 MXN	<a href="https://www.carrier.com.mx/residencial/84">https://www.carrier.com.mx/residencial/84</a>

**Tabla 1:** Tabla de Materiales IoT escalado real.

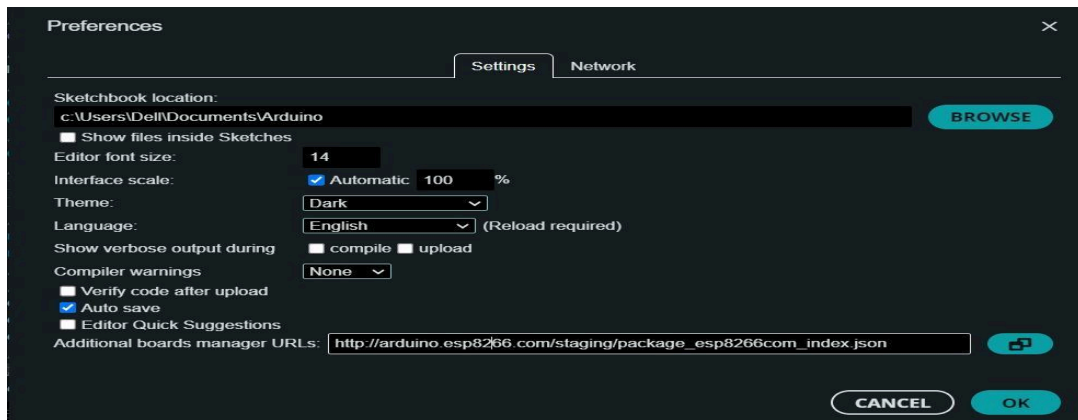
Listado de materiales para el desarrollo del prototipo en escalada maquettata así mismo incluyendo detalles para la compra de estos.

TablaloTMaqueta			
Categoría	Componente	Precio	Enlace
Sensores de Movimiento	Sensor ultrasónico HC-SR04	\$42 MXN	<a href="https://www.330ohms.com/products/sensor-ultrasonico-hc-sr04?srsltid=AfmB0or_2MVjuwKsDvF9YzWVu0GvDlxUCkfGqjK32ZFoCfPgDFw22T1e">https://www.330ohms.com/products/sensor-ultrasonico-hc-sr04?srsltid=AfmB0or_2MVjuwKsDvF9YzWVu0GvDlxUCkfGqjK32ZFoCfPgDFw22T1e</a>
Sensor de Presencia	Sensor de movimiento PIR	\$44 MXN	<a href="https://www.330ohms.com/products/sensor-de-movimiento-pir-1?_pos=1&amp;_psq=sensor+pir&amp;_ss=e&amp;_v=1.0">https://www.330ohms.com/products/sensor-de-movimiento-pir-1?_pos=1&amp;_psq=sensor+pir&amp;_ss=e&amp;_v=1.0</a>
Sensor de Humedad y Tempe	Sensor de temperatura y humedad DHT11	\$72 MXN	<a href="https://www.330ohms.com/products/modulo-sensor-de-humedad-y-temperatura-dht11-con-cables-jumper?_pos=1&amp;_sid=be6dbcfcc&amp;_ss=r">https://www.330ohms.com/products/modulo-sensor-de-humedad-y-temperatura-dht11-con-cables-jumper?_pos=1&amp;_sid=be6dbcfcc&amp;_ss=r</a>
Actuadores	Servomotor SG90 180°	\$86 MXN	<a href="https://www.330ohms.com/products/micro-servo-de-180-grados-sg90?_pos=2&amp;_sid=8979b59ee&amp;_ss=r">https://www.330ohms.com/products/micro-servo-de-180-grados-sg90?_pos=2&amp;_sid=8979b59ee&amp;_ss=r</a>
	Módulo relevador	\$84 MXN	<a href="https://www.330ohms.com/products/ventilador-5v-para-raspberry-pi-3?_pos=3&amp;_psq=ventilador&amp;_ss=e&amp;_v=1.0">https://www.330ohms.com/products/ventilador-5v-para-raspberry-pi-3?_pos=3&amp;_psq=ventilador&amp;_ss=e&amp;_v=1.0</a>
	Mini foco LED	\$54 MXN	<a href="https://www.ledsbesolar.com/producto/foco-led-tipo-mini-bulbo-4w-6500k/">https://www.ledsbesolar.com/producto/foco-led-tipo-mini-bulbo-4w-6500k/</a>
	Socket para foco	\$10.11 MXN	<a href="https://comercialelectrica.mx/producto/socket-de-porcelana-y-de-base-cuadrada-e26-marca-supplier/">https://comercialelectrica.mx/producto/socket-de-porcelana-y-de-base-cuadrada-e26-marca-supplier/</a>

**Tabla 2:** Tabla de Materiales IoT maquettato.

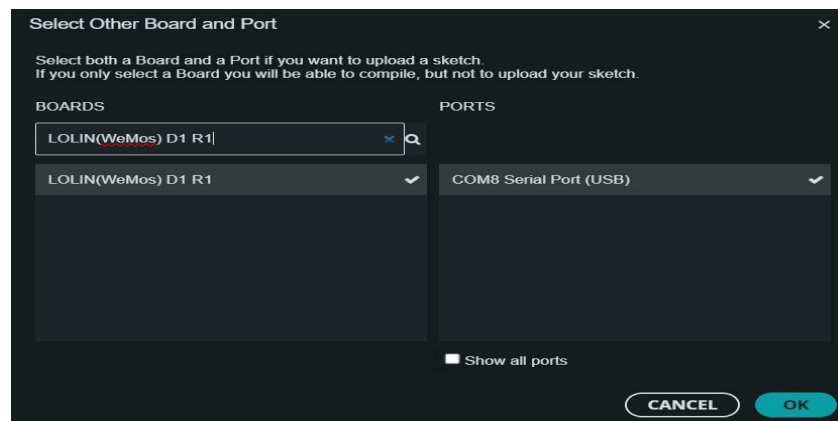
## 9. Configuración final de IoT

Configuración adicional dentro del software Arduino IDE para el correcto funcionamiento de los componentes adquiridos.



**Figura 3:** Configuración de preferencias del Arduino IDE (URLs).

Una vez que se colocó el link en preferencias, nos dirigimos al apartado de administrador de tarjetas, aquí buscamos y seleccionamos la placa esp8266 y la instalamos.



**Figura 4:** Instalación esp8266

Una vez descargado e instalado, en el apartado de dispositivos, seleccionamos el puerto en el cual está conectada la placa a nuestra computadora, y en "boards", buscamos y seleccionamos "LOLIN(WeMos)D1 R1".

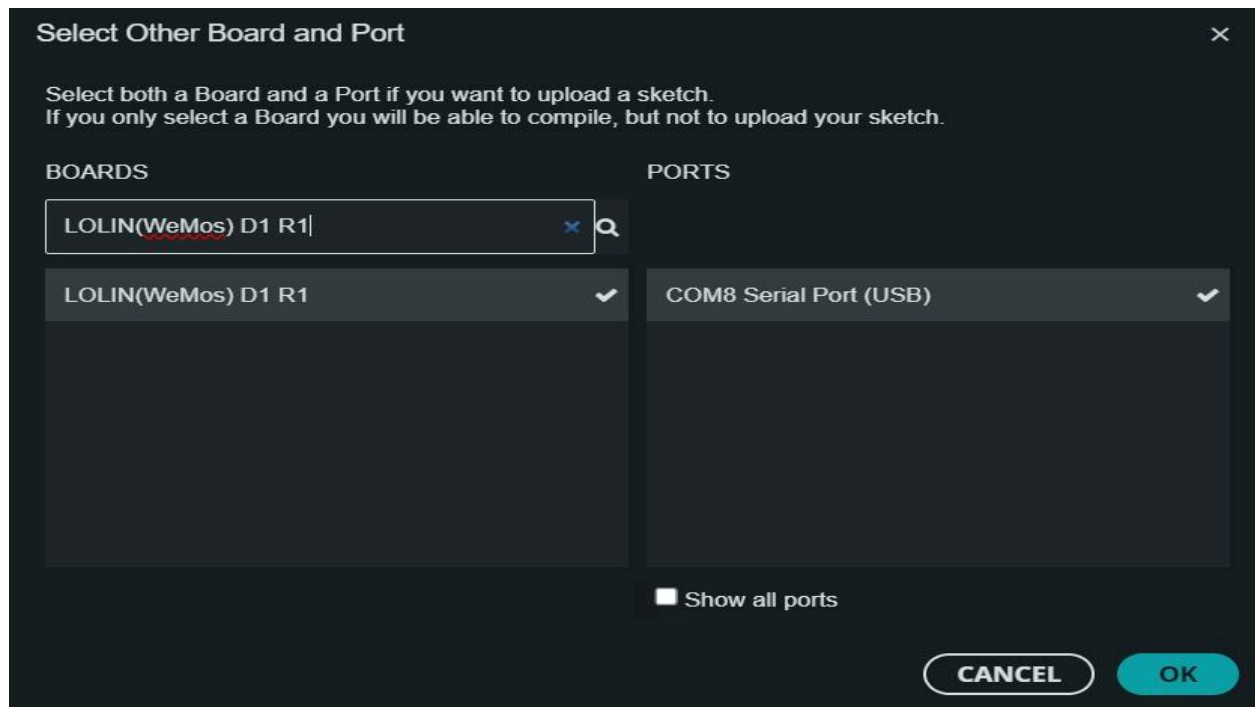


Figura 5: Seleccionaremos Puerto LOLIN(WeMos)D1 R1

## 10. Marco Contextual

### 10.1. Descripción del Problema:

Actualmente, la falta de personalización e integración entre dispositivos inteligentes representa un obstáculo para la adopción de la automatización del hogar. Muchos de los sistemas actuales que se encuentran en el mercado presentan costos elevados y estos requieren una infraestructura propietaria, lo cual radica en su limitación a la accesibilidad al público en general. Por lo tanto lo que se busca es desarrollar un sistema IoT asequible y flexible para la automatización de hogares, abordando la falta de compatibilidad entre los dispositivos y reduciendo los costos de implementación mediante el uso de hardware y software abierto.

### 10.2. Entorno de operación

El sistema está diseñado para su uso en ambientes domésticos, tanto en interiores como en exteriores. Su implementación radica en viviendas individuales, donde los usuarios requieran una solución de automatización accesible y personalizable.

### 10.3. Factores Sociales, Economicos y Tecnologicos

**Social:** El creciente interés por la domótica ha impulsado la demanda de soluciones más accesibles y flexibles para hogares inteligentes.

**Económico:** La reducción de costos en hardware IoT ha permitido lograr la proliferación de proyectos

**Tecnológico:** Avances en los microcontroladores y protocolos de comunicación han mejorado la eficiencia y seguridad de los sistemas.

### Capítulo 3 Metodología y Desarrollo

Antes de comenzar con el desarrollo, es importante tener en cuenta lo siguiente

¿Cuál es el problema que resolverá el producto IoT?

**Romper la falta de personalización, incompatibilidad entre los dispositivos y enfrentar los costos altos de implementación.**

¿Qué tipo de datos se van a medir?

**Los datos serán transmitidos por los componentes IoT, tipo numéricos, caracteres.**

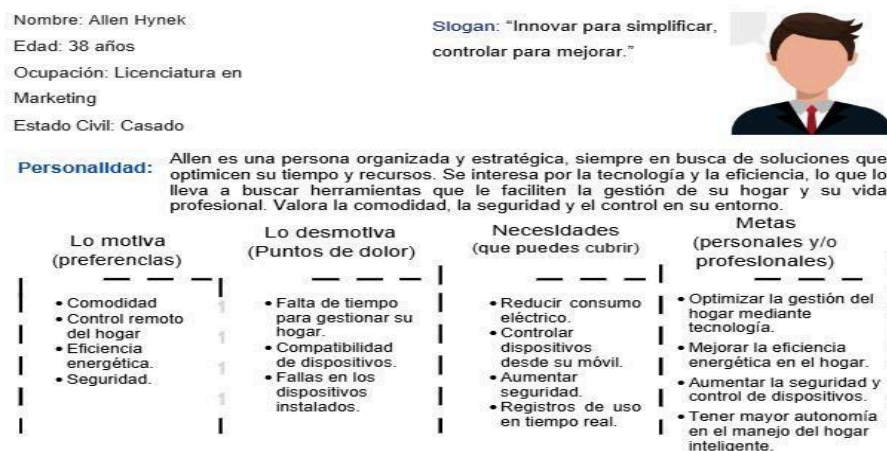
¿Cuál es el entorno de operación?

**Se enfoca para uso doméstico en interiores y exteriores.**

¿Se requiere conectividad constante o puede trabajar offline?

**Se requiere conectividad constante para poder trabajar.**

Un **Buyer Persona** es una representación semi-ficticia del cliente ideal basada en datos reales y suposiciones fundamentadas sobre su comportamiento, necesidades y motivaciones.



**Figura 6: Buyer Person**



Figura 7: Modelo Canvas

## 1. Documentación Técnica

El sistema de iHome sigue una arquitectura de microservicios, donde cada servicio es responsable de una función específica (por ejemplo, control de iluminación, gestión de energía, notificaciones). Los servicios se comunican entre sí a través de APIs RESTful, asegurando una arquitectura desacoplada que facilita su mantenimiento y expansión.

**Componentes clave:**

**Frontend:** El frontend está desarrollado en Angular y sigue el patrón de diseño MVC (Modelo-Vista-Controlador). Este frontend es responsable de la interfaz de usuario donde los usuarios pueden interactuar con los dispositivos y servicios.

**Backend:** El backend está basado en Node.js y Express, que gestionan las peticiones y la lógica de negocio. Este servidor se comunica con la base de datos y otros servicios para obtener o almacenar datos.

**Base de Datos:** La base de datos principal es MongoDB, que almacena los datos de usuario, configuraciones de dispositivos y registros de eventos. Se ha optado por MongoDB debido a su escalabilidad y flexibilidad para manejar grandes volúmenes de datos no estructurados.

**Servicios IoT:** Para la integración de dispositivos IoT, se utiliza MQTT como protocolo de comunicación, permitiendo que los dispositivos se conecten al sistema para ser controlados y monitoreados en tiempo real.

## Instalación y Configuración

- Node.js versión mínima 14.x.
- MongoDB versión mínima 4.x.
- Angular CLI.
- WebSockets.

### 2. Pasos para Clonar el repositorio:

`git clone <repositorio-url>.`



Navega a la carpeta del backend y ejecuta:

```
npm install
```

Esto instalará las dependencias necesarias del backend.

Navega a la carpeta del frontend y ejecuta:

```
npm install
```

Esto instalará las dependencias necesarias del frontend.

Inicia el servidor backend:

```
npm run start:dev
```

Inicia el frontend:

```
ng serve
```

Con todos los pasos anteriores podrás acceder <http://localhost:4200> de manera local.

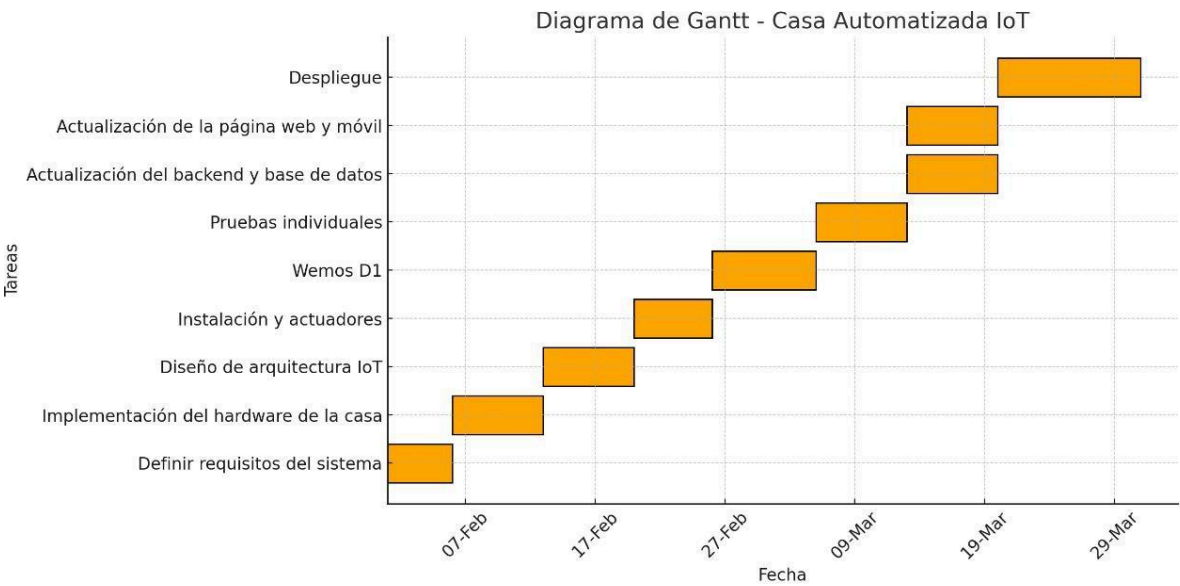
- **Desarrollo a futuro:**

Se planea la integración de más protocolos de comunicación esto mejorará considerablemente la compatibilidad con dispositivos de distintas gamas.

Figura 8: Matriz de Responsabilidades

RACI						
Tarea / Actividad	Product Owner	Scrum Master	Equipo Desarrollo	UI / UX desingner	IoT Engineer	QA Tester
Definir requisitos del SO	A	I	C	C	C	R
Blacklog	A	C	C	I	I	R
Desarrollo Movil / Web	A	I	R	R	C	I
Configuracion de Protocolos MQTT	R	C	C	I	C	R
Dsieño de UI	C	I	R	R	C	I
Realizar pruebas de integración	A	I	I	I	I	R
Gestionar Despliegue en el servidor local	C	I	R	C	C	C
Documentación interna	I	I	R	C	C	R

Figura 9: Cronograma de Actividades



3. Arquitectura del Sistema

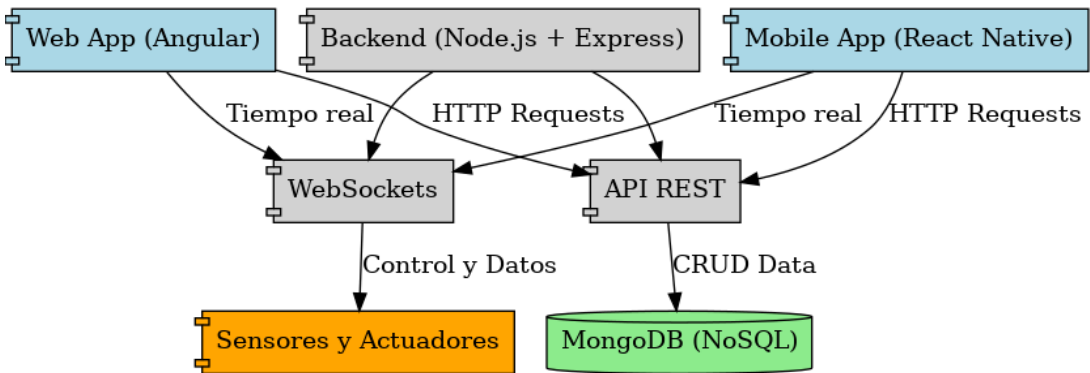
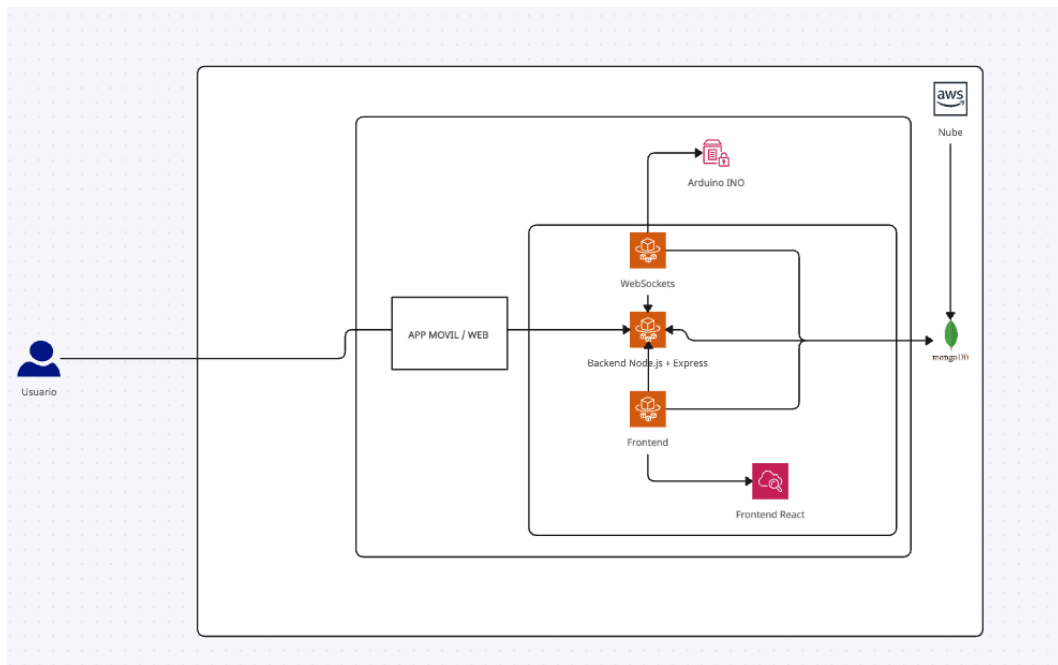


Figura 10: Diagrama de componentes del sistema

Figura 11: Diagrama de arquitectura



#### 4. Requerimientos Funcionales y No Funcionales

##### **Control de Dispositivos del Hogar:**

- El sistema debe permitir al usuario controlar dispositivos como luces, puertas, ventanas, ventilación, temperatura y sensores de presencia.
- Los dispositivos podrán ser encendidos, apagados o ajustados según los parámetros definidos por el usuario.

##### **Monitoreo en Tiempo Real:**

- El sistema debe contar con sensores de movimiento integrados con dispositivos inteligentes (por ejemplo, foco inteligente que se enciende al detectar movimiento).
- Monitoreo en tiempo real de la temperatura y humedad dentro del hogar mediante sensores DHT21.

**Automatización Programada:**

- Los usuarios podrán programar los dispositivos para que se enciendan o apaguen automáticamente a determinadas horas del día o según la detección de ciertos eventos (por ejemplo, presencia o temperatura).

**Interfaz de Usuario:**

- El sistema debe proporcionar una aplicación web y móvil para que los usuarios puedan interactuar con los dispositivos de manera intuitiva y eficiente.
- La aplicación debe permitir la visualización y control de dispositivos, y el estado de los sensores en tiempo real.

**Escalabilidad:**

- El sistema debe ser escalable para permitir la integración de nuevos dispositivos en el futuro sin afectar el rendimiento.

**Accesibilidad:**

- El sistema debe ser accesible desde diferentes plataformas (web y móvil) y ser completamente funcional en ambas.

**No Funcionales****Conectividad:**

- El sistema debe requerir conectividad constante para el correcto funcionamiento (Wi-Fi estable en el hogar).
- En caso de pérdida de conectividad, el sistema debe ser capaz de mantener las funciones básicas hasta que la conexión sea restaurada.

**Seguridad:**

- La comunicación entre el dispositivo IoT y la plataforma debe estar asegurada con TLS/SSL o protocolos de comunicación seguros (por ejemplo, MQTT seguro).

- El sistema debe permitir la autenticación y autorización de los usuarios para asegurar que solo los usuarios autorizados puedan controlar los dispositivos.

#### **Usabilidad:**

- La interfaz debe ser intuitiva y fácil de usar para usuarios con conocimientos limitados en tecnología.
- Debe permitir la configuración de dispositivos y la creación de automatizaciones sin complicaciones.

#### **Rendimiento:**

- El sistema debe ser capaz de manejar múltiples dispositivos de forma simultánea sin degradar su rendimiento, garantizando tiempos de respuesta rápidos para todas las interacciones del usuario.

#### **Mantenimiento:**

- El sistema debe ser fácil de mantener, con un plan de actualizaciones Over-the-Air (OTA) y monitoreo de los dispositivos desplegados.

**Escalabilidad:**

- El sistema debe poder manejar el crecimiento tanto en número de dispositivos como en la cantidad de usuarios que interactúan con el sistema.

**Reglas de Negocio****Control de Acceso:**

- Solo los usuarios autenticados pueden modificar las configuraciones de los dispositivos. Un usuario no autenticado debe tener acceso solo a la visualización de los datos (sin poder modificar configuraciones).

**Activación por Movimiento:**

- Los dispositivos de control de luz deben activarse automáticamente cuando un sensor de movimiento detecte presencia, a menos que el sistema esté en modo de ahorro de energía o haya sido configurado para no activar dispositivos automáticamente.

**Programación de Dispositivos:**



- Un dispositivo solo puede ser programado para encenderse o apagarse a horas específicas si la conectividad de red es estable en ese momento. Si no hay conexión, la programación no podrá ejecutarse correctamente.

#### **Expansión de Dispositivos:**

- El sistema debe permitir agregar nuevos dispositivos sin interrumpir el funcionamiento de los dispositivos ya instalados.

#### **Notificaciones al Usuario:**

- El sistema debe notificar al usuario cuando se detecte alguna anomalía en los dispositivos (por ejemplo, mal funcionamiento o desconexión de un dispositivo).

#### **Compatibilidad de Dispositivos:**

- El sistema debe ser compatible con una variedad de dispositivos inteligentes, tanto en el ámbito de dispositivos de uso común como luces, como de dispositivos más avanzados como termostatos inteligentes.

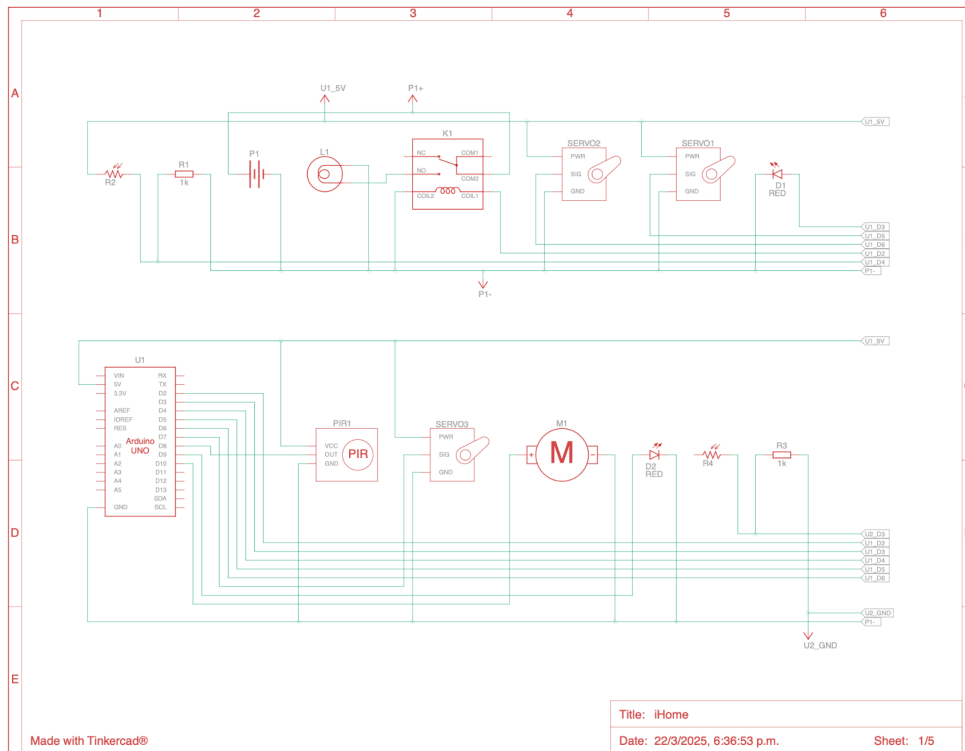


Figura 12: Diagrama de Conexiones IoT

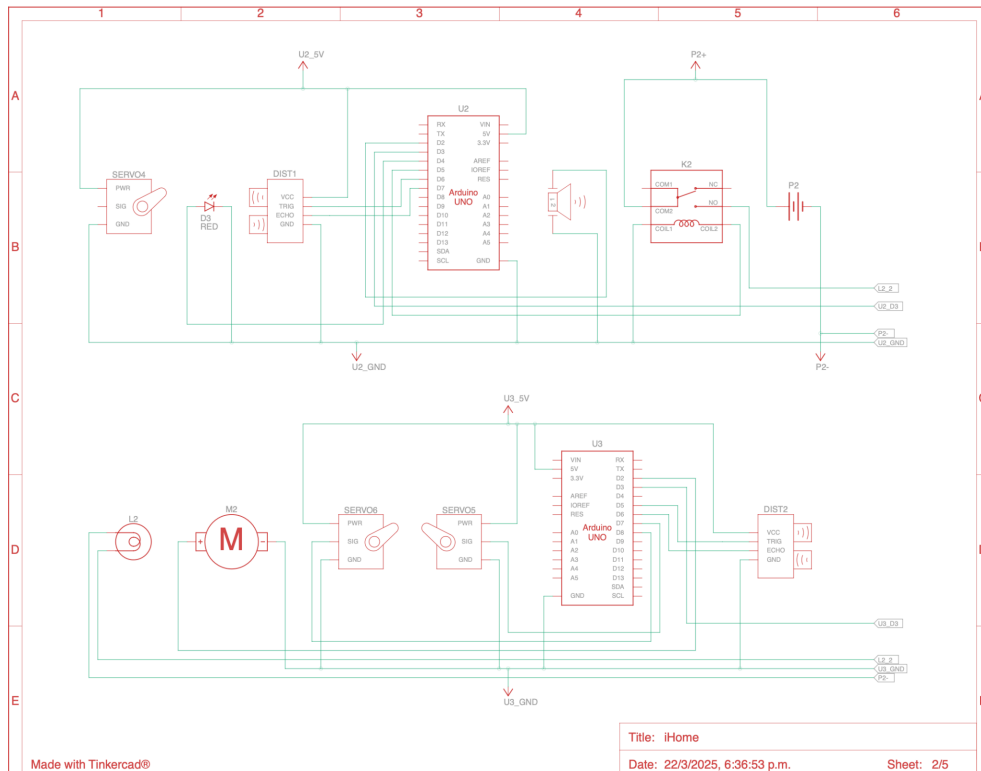


Figura 13:Diagrama de Conexiones IoT

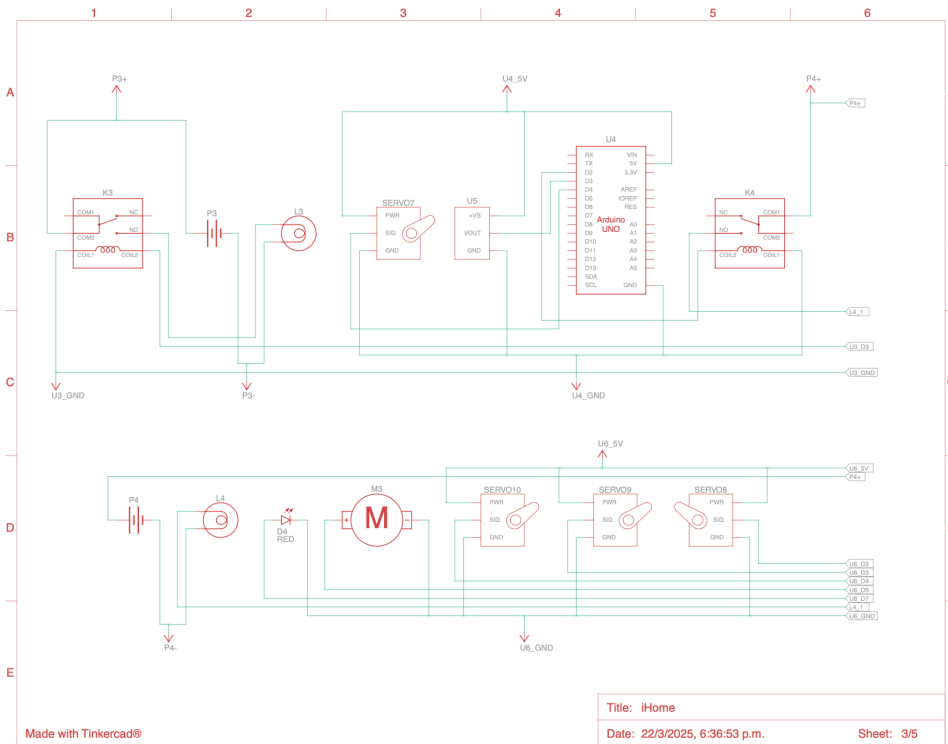


Figura 14:Diagrama de  
Conexiones IoT

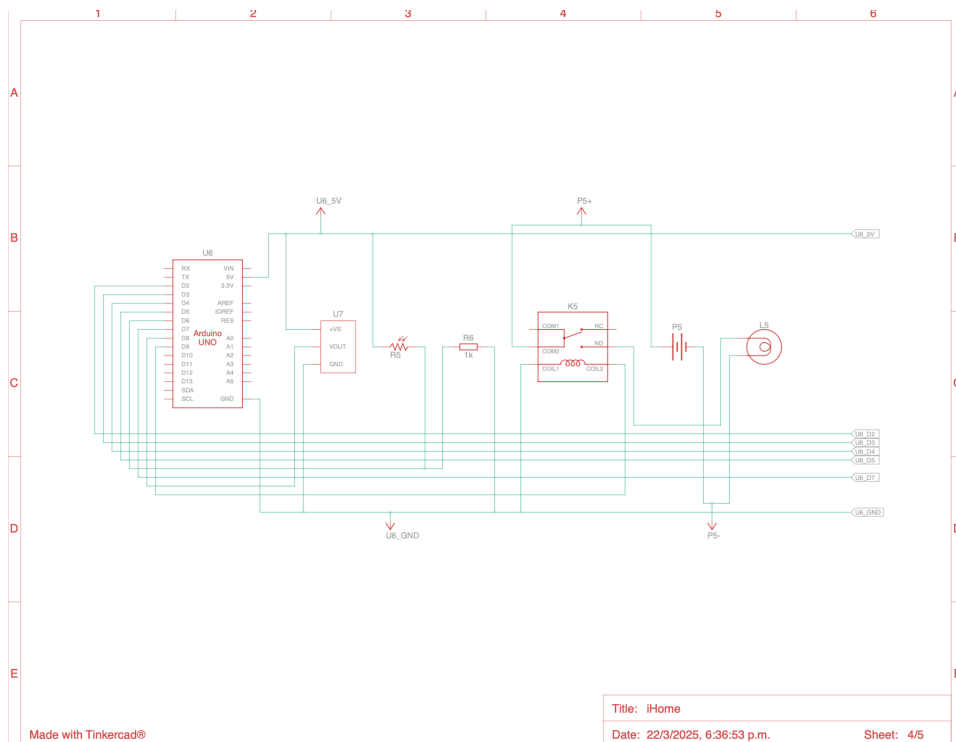


Figura 15:Diagrama de  
Conexiones IoT

Figura 16:Diagrama de Conexiones IoT

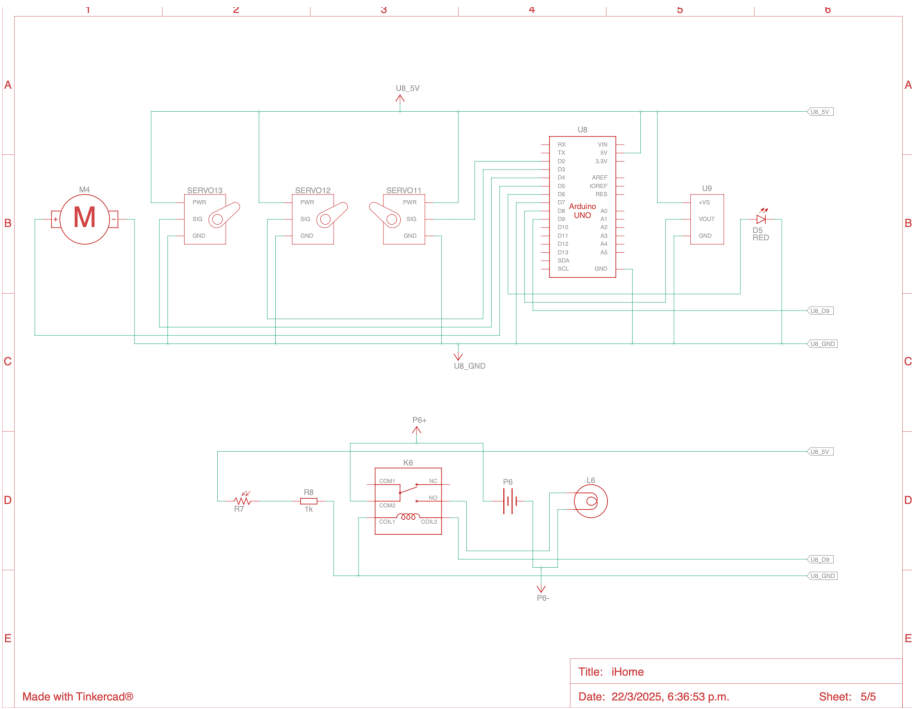


Figura 17: Lista de Componentes y Propósitos

Componente	Modelo	Propósito
Sensor de humedad y temperatura	DHT11	Medir los niveles de humedad en el ambiente, así como la temperatura en un área determinada.
Sensor de luz	LDR	Detectar la presencia de luz, y medir la intensidad de ésta.
Sensor de proximidad	HC-SR04	Emitir sonidos ultrasónicos para detectar obstáculos cercanos.
Sensor de presencia	HC-SR501	Detectar movimientos que se realicen dentro de un rango específico.
Relé	SRD-05VDC-SL-C	Permitir o denegar el paso de energía a un componente para su funcionamiento esperado.
Micro servo	SG-90	Realizar movimientos dentro de un rango especificado, en este caso se mide en grados.
Bomba de agua	MOT-300	Propulsar agua a través de un conducto para hacerlo llegar a un recipiente.
Placa controladora	Wemos D1(esp8266)	Activar, desactivar, y dar funcionalidad a los dispositivos electrónicos conectados a esta, haciendo uso de WiFi.

## 1. Diseño de arquitectura

Estructura del backend.

Se planea desplegar el app en alguna de las plataformas de hosting en la nube tales como, vercel o heroku, lo que permitirá una fácil escalabilidad y disponibilidad del proyecto.

La estructura del backend sigue una organización de encarpetao:

```
src/  
| — config/  
|   └─ database.config.js  
| — controllers/  
|   └─ authController.js  
|   └─ bathrooms.controller.js  
|   └─ bedrooms.controller.js  
|   └─ garage.controller.js  
|   └─ kitchen.controller.js  
|   └─ livingroom.controller.js  
|   └─ rooms.controller.js  
|   └─ index.js  
| — dao/  
|   └─ bathrooms.dao.js  
|   └─ bedrooms.dao.js  
|   └─ garage.dao.js  
|   └─ livingroom.dao.js  
|   └─ rooms.dao.js
```

- | └─ index.js
- | ─ errors/
- | └─ NotFoundException.error.js
- | ─ middlewares/
- | └─ GlobalErrorHandler.middleware.js
- | ─ models/
- | └─ bathrooms.schema.js
- | └─ bedrooms.schema.js
- | └─ garage.schema.js
- | └─ kitchen.schema.js
- | └─ livingroom.schema.js
- | └─ rooms.schema.js
- | ─ routes/
- | └─ auth.routes.js
- | └─ bathrooms.routes.js
- | └─ bedrooms.routes.js
- | └─ garage.routes.js
- | └─ kitchen.routes.js
- | └─ livingroom.routes.js
- | └─ rooms.routes.js
- | └─ index.js
- | ─ utils/
- | └─ common/

```
| |— rooms.structure.js
| |— functions/
| |— getCollectionByLocation.js
|— app.js
|— main.js
|— swagger.js
```

## 2. Modelo de Bases de Datos (NoSQL)

La base de datos contiene las siguientes colecciones:

- bathrooms (Baños).
- bedrooms (Dormitorios).
- garage (Garaje).
- kitchen (Cocina).
- livingroom (Sala de estar).
- rooms (Habitaciones en general).

Cada colección tiene su propio esquema definido en la carpeta/models

### Ejemplo de un esquema Mongoose

Figura 18: Kitchen

```
import { Schema, model } from "mongoose";

const kitchenSchema = new Schema({
  arduinoIp: String,
  type: String,
  name: String,
  brand: String,
  model: String,
  specifications: [{}],
  location: String,
  status: String,
  registeredDate: {
    type: Date,
    default: Date.now
  },
  owner: String,
  readings: [{
  }],
  actions: [{
  }]
},{
  versionKey: false
});
```

### 3. Conexión con MongoDB en Express

En el archivo de configuración /config/database.js, la conexión con MongoDB se realiza de la siguiente manera:

```
import mongoose from 'mongoose';
import dotenv from 'dotenv';

dotenv.config({ path: 'src/.env' });

const connectionUrl = process.env.CONNECTION_DB ||
process.env.CONNECTION_DB_LOCAL;
console.log('CONNECTION_DB:', process.env.CONNECTION_DB);
console.log('CONNECTION_DB_LOCAL:', process.env.CONNECTION_DB_LOCAL);

try {
  mongoose.connect(connectionUrl)
    .then(() => {
      console.log(`
#####

Mongo Database Successfully Connected

#####
`);
    })
    .catch(err => {
```

Figura 19: Conexión a MongoDB.

### 4. Configuración de WebSockets en el Backend

El servidor de WebSockets está implementado en Node.js con la biblioteca 'socket.io' utilizando la siguiente lógica:

```
io.on('connection',(socket)=>{

  console.log("Usuario conectado")

  socket.on('disconnect',()=>{

    console.log("Usuario desconectado")

  })

})
```



- Implementación en socket los controladores de Create y Delete (Ejemplo):

```
bathroomsController.createBathroomData = async (req, res, next) => {  
  
  try {  
  
    /* get newData for bathroom of request body  
  
    const newData = req.body;  
  
    if (!newData) {  
  
      throw new NotFoundException("To create bathroom information, is required data  
  
      by body. ")  
  
    }  
  
    const saveData = await bathroomDao.createBathroomData(newData);  
  
    io.emit('bathrooms DataChanged') // Emitir evento para notificar que los datos  
  
    han cambiado  
  
    res.status(200).json({  
  
      success: true,  
  
      message: "New data for bathrooms is successfully created.",  
  
      data: saveData  
  
    })  
  
    } catch (error) {  
  
      console.error(`Error in Bathrooms Controller: createBathroomData:  
  
      ${error.message}`);  
  
      // next is a http request method that let to continue to next middleware in the list  
  
      (this case the next  
  
      middleware is error handler)
```

```
// error handler is a global middleware that is used to response errors to client, this
manage errors

next(error) //Continue to global error handler (error middleware)

throw error;

}

}

bathroomsController.deleteBathroomRecordById = async (req, res) => {

  try {

    const { id } = req.params;

    if (!id) {

      return res.status(400).json({ message: 'ID no proporcionado' });

    }

    const result = await bathroomDao.deleteRecordById(id);

    if (result.nModified === 0) {

      return res.status(404).json({ message: 'Registro no encontrado' });

    }

    io.emit('bathroomsDataChanged') // Emitir evento para notificar que los datos han
    cambiado

    res.status(200).json({ message: 'Registro eliminado correctamente' });

  } catch (error) {

    console.error(`Error en deleteRecord: ${error.message}`);

    res.status(500).json({ message: `Error al eliminar el registro: ${error.message}`

    });

  }

}
```

```
}
```

```
};
```

```
io.emit('bathroomsDataChanged') // Emitir evento para notificar que los datos han  
cambiado.
```

## 5. Desarrollo de Software

### 5.1. Firmware (Código IoT)

El Firmware del dispositivo IoT está desarrollado en C++ y ejecutando en un microcontrolador como ESP8266 o ESP32. Se encarga de la comunicación con el backend a través de WiFi y WebSockets, permitiendo el monitoreo y control remoto de dispositivos en tiempo real:

#### **Lenguaje Utilizado:**

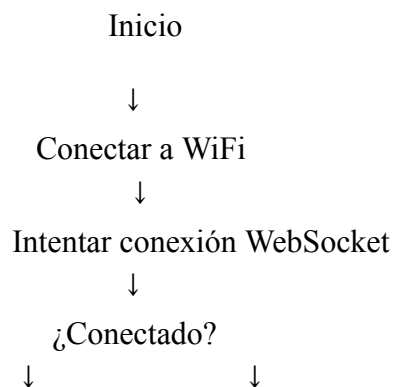
C++

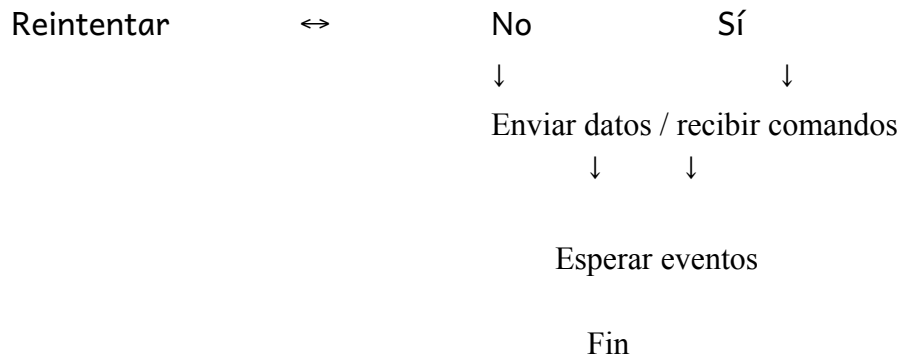
#### **Flujo del Programa:**

El firmware sigue el siguiente flujo de ejecución:

- Conectar a la red WiFi utilizando las credenciales de seguridad correctas y configuradas en el código.
- Establecer una conexión WebSocket con el servidor backend.
- Enviar datos de sensores y/o recibir comandos de control.
- Manejar eventos de conexión/desconexión y reintentar en caso de que se presente algún error.

Ejemplo de Flujo:





6. Código de ejemplo que se requiere para un funcionamiento pleno de una recamara (C++).

```
//Libreria para hacer peticiones http
```

```
#include <ESP8266HTTPClient.h>
```

```
//Libreria para conectarse a WIFI
```

```
#include <ESP8266WiFi.h>
```

```
//Libreria a la par del WIFI
```

```
#include <WiFiClient.h>
```

```
//Libreria para los servos
```

```
#include <Servo.h>
```

```
//Libreria para el WEB SERVICE
```

```
#include <ESP8266WebServer.h>
```

```
//Libreria para el dht11
```

```
#include <DHT.h>
```

```
// Definimos el pin digital donde se conecta el sensor
```

```
#define DHTPIN D5
```

```
// Definimos el tipo de sensor
```

```
#define DHTTYPE DHT11
```

```
//Definimos el pin del LDR

#define pinLDR D15

//Definimos el pin del led interior

#define pinLedIndoor D12

//Definimos el pin del led exterior

#define pinLedOutdoor D11

//Definimos el pin del ventilador

#define relayPin D10

// Inicializamos el sensor DHT11

DHT dht(DHTPIN, DHTTYPE);

//Wifi

const char *ssid = "HONOR X7";const char *password = "12345678";

//Url base (Api/Server local/etc.)

const String serverName = "http://192.168.176.164:3003/api/v1";

//Variables para sensor humedad

float h=0.0;

float t=0.0;

//Variables para el LDR

int valorLDR = 0;

int angulo;

// Definimos los pins para los servos

// Objetos tipo servo

Servo servoWindowL;
```

```
Servo servoWindowR;

Servo servoDoor;

//Servidor del wifi

WiFiServer server(80);

// Variables para el estatus de las puertas

bool windowsOn=false;

bool windowLeftOn = false;

bool windowRightOn = false;

bool doorOpen = false;

bool ledIndoorOn = false;

bool ledOutdoorOn = false;

bool ventiladorOn=false;

bool controlLedOutOn=false;

bool controlFanOn=false;

//Variable para el "ASYNC"

unsigned long lastMillis;

String dataHumiditySensor;String dataLdrSensor;

String dataServo;

String dataLed;

String dataFan;

String response = "{\"Status\":\"successfully\"}";

IPAddress ip;

String ipString;
```

```
void setup() {  
  
  // Iniciamos el puerto serial  
  
  Serial.begin(9600);  
  
  pinMode(pinLedIndoor, OUTPUT);  
  
  pinMode(pinLedOutdoor, OUTPUT);  
  
  pinMode(relayPin, OUTPUT);  
  
  // Comenzamos el sensor DHT  
  
  dht.begin();  
  
  //Conexion a Wifi  
  
  WiFi.begin(ssid, password);  
  
  Serial.println("Conectando...");  
  
  while (WiFi.status() != WL_CONNECTED) {  
  
    delay(500);  
  
    Serial.print(".");  
  
  }  
  
  Serial.println("Conectado a la red IP:");  
  
  Serial.println(WiFi.localIP());  
  
  Serial.println("Conexion exitosa...");  
  
  //Iniciamos el servidor Web  
  
  server.begin();Serial.println("Servidor WEB Iniciado");  
  
  // No se :v(Rulo sabe)  
  
  servoWindowL.attach(D2);  
  
  servoWindowR.attach(D0);  
}
```



```
servoDoor.attach(D4);

servoWindowL.write(0);

servoWindowR.write(180);

servoDoor.write(0);

//Se almacena el milisegundo(0)

lastMillis = millis();

//Ip del arduino

ip = WiFi.localIP();

ipString = ip.toString();

insertActionsDefault();

}

void loop() {

//Hace el rol de "asyn" los milisegundos actuales se los resta a los pasados cuando pasan

10

segundos se ejecuta la funcion

if ((millis() - lastMillis) >= 10000) {

humiditySensorReading();

dataHumiditySensor = new TempHumData("Recámara 1", "CAORGRIS");

postRequest("/bedrooms/",dataHumiditySensor);

LdrSensorReading();

dataLdrSensor = new LdrData("Recámara 1","CAORGRIS");

postRequest("/bedrooms/",dataLdrSensor);

lastMillis = millis();
```

```

}

//Manda a llamar la logica del servidor webif(!controlLedOutOn){

LdrSensorReadingLed();

}

if(!controlFanOn){

humiditySensorReadingFan();

}

LdrSensorReading();

webServer();

}

void postRequest(String endPoint, String body) {

WiFiClient client;

HTTPClient http;

if (WiFi.status() == WL_CONNECTED) {

String serverPath = serverName + endPoint;

Serial.print("Connecting to server: ");

Serial.println(serverPath);

if (http.begin(client, serverPath)) {

http.addHeader("Content-Type", "application/json");

int httpCode = http.POST(body);

if (httpCode > 0) {

Serial.print("HTTP status: ");

Serial.println(httpCode);

```

```
String payload = http.getString();

Serial.println("Response: ");

Serial.println(payload);

} else {

Serial.print("HTTP error code: ");

Serial.println(httpCode);}

http.end();

} else {

Serial.println("Failed to connect to server");

}

} else {

Serial.println("WiFi not connected");

}

}

void webServer(){

WiFiClient client = server.available();

Serial.print("IP del arduino: ");

Serial.println(WiFi.localIP());

if(client.available()){

String req = client.readStringUntil('\r');

Serial.println(req);

if (req.indexOf("OPTIONS") != -1) {

handleOptions(client);
```

```
client.stop();

return;

}

if(req.indexOf("windowsOpen") != -1 && !windowsOn){

windowsOn = true;

for(angulo = 180; angulo >=0; angulo=angulo-2)

{

servoWindowL.write(180-angulo);

servoWindowR.write(angulo);

}dataServo = newServoData("Ventana Izquierda","Recámara 1","CAORGRIS","true");

postRequest("/bedrooms/",dataServo);

dataServo = newServoData("Ventana Derecha","Recámara 1","CAORGRIS","true");

postRequest("/bedrooms/",dataServo);

}

if(req.indexOf("windowsClose") != -1 && windowsOn){

windowsOn = false;

for(angulo = 0; angulo <=180; angulo=angulo+2)

{

servoWindowL.write(180-angulo);

servoWindowR.write(angulo);

}

dataServo = newServoData("Ventana Izquierda","Recámara 1","CAORGRIS","false");

postRequest("/bedrooms/",dataServo);
```

```
dataServo = newServoData("Ventana Derecha","Recámara 1","CAORGRIS","false");  
postRequest("/bedrooms/",dataServo);  
  
}  
  
if(req.indexOf("doorOpen") != -1 && !doorOpen){  
for(angulo = 0; angulo <=180; angulo=angulo+2)  
  
{  
servoDoor.write(angulo);  
  
}  
  
doorOpen = true;  
  
dataServo = newServoData("Puerta","Recámara 1","CAORGRIS","true");  
postRequest("/bedrooms/",dataServo);  
  
}  
  
if(req.indexOf("doorClose") != -1 && doorOpen){  
for(angulo = 180; angulo >=0; angulo=angulo-2)  
  
{servoDoor.write(angulo);  
  
}  
  
doorOpen = false;  
  
dataServo = newServoData("Puerta","Recámara 1","CAORGRIS","false");  
postRequest("/bedrooms/",dataServo);  
  
}  
  
if(req.indexOf("ledIndoorOn") != -1 && !ledIndoorOn){  
  
ledIndoorOn = true;  
  
digitalWrite(pinLedIndoor, HIGH);
```

```
dataLed = newLedData("Led Interior","Recámara 1","CAORGRIS","true");  
postRequest("/bedrooms/",dataLed);  
  
}  
  
if(req.indexOf("ledIndoorOff") != -1 && ledIndoorOn){  
    ledIndoorOn = false;  
  
    digitalWrite(pinLedIndoor, LOW);  
  
    dataLed = newLedData("Led Interior","Recámara 1","CAORGRIS","false");  
    postRequest("/bedrooms/",dataLed);  
  
}  
  
if(req.indexOf("ledOutdoorOn") != -1 && !ledOutdoorOn){  
    ledOutdoorOn = true;  
  
    digitalWrite(pinLedOutdoor, HIGH);  
  
    dataLed = newLedData("Led Exterior","Recámara 1","CAORGRIS","true");  
    postRequest("/bedrooms/",dataLed);  
  
}  
  
if(req.indexOf("ledOutdoorOff") != -1 && ledOutdoorOn){  
    ledOutdoorOn = false;  
  
    digitalWrite(pinLedOutdoor, LOW);  
  
    dataLed = newLedData("Led Exterior","Recámara 1","CAORGRIS","false");  
    postRequest("/bedrooms/",dataLed);}  
  
if(req.indexOf("fanOn") != -1 && !ventiladorOn){  
    ventiladorOn = true;  
  
    digitalWrite(relayPin, HIGH);
```

```
dataFan = newFanData("Recámara 1","CAORGRIS","true");  
postRequest("/bedrooms/",dataFan);  
  
}  
  
if(req.indexOf("fanOff") != -1 && ventiladorOn){  
    ventiladorOn = false;  
  
    digitalWrite(relayPin, LOW);  
  
    dataFan = newFanData("Recámara 1","CAORGRIS","false");  
    postRequest("/bedrooms/",dataFan);  
  
}  
  
if(req.indexOf("controlLedOutOn") != -1){  
    controlLedOutOn=true;  
  
}  
  
if(req.indexOf("controlLedOutOff") != -1){  
    controlLedOutOn=false;  
  
}  
  
if(req.indexOf("controlFanOn") != -1){  
    controlFanOn=true;  
  
}  
  
if(req.indexOf("controlFanOff") != -1){  
    controlFanOn=false;  
  
}  
  
client.println("HTTP/1.1 200 OK");  
  
client.println("Content-Type: application/json");
```

```
client.println("Access-Control-Allow-Origin: *");client.println("Content-Length: " +
String(response.length()));

client.println();

client.println(response);

Serial.print("Cliente desconectado: ");

Serial.println(client.remoteIP());

client.flush();

client.stop();

}

}

void humiditySensorReading(){

h = dht.readHumidity();

t = dht.readTemperature();

if (isnan(h) || isnan(t)) {

Serial.println("Error obteniendo los datos del sensor DHT11");

return;

}

Serial.print("Humedad: ");

Serial.print(h);

Serial.print(" %\t");

Serial.print("Temperatura: ");

Serial.print(t);

}
```



```

void humiditySensorReadingFan(){
    h = dht.readHumidity();
    t = dht.readTemperature();
    if (isnan(h) || isnan(t)) {
        return;}
    if(t>=23 && !ventiladorOn){
        ventiladorOn = true;
        digitalWrite(relayPin,HIGH);
        dataFan = newFanData("Recámara 1","CAORGRIS","true");
        postRequest("/bedrooms/",dataFan);
    }else if(t<23 && ventiladorOn){
        ventiladorOn = false;
        digitalWrite(relayPin,LOW);
        dataFan = newFanData("Recámara 1","CAORGRIS","false");
        postRequest("/bedrooms/",dataFan);
    }
}

void LdrSensorReading(){
    valorLDR = analogRead(pinLDR);
    Serial.print("Lectura LDR: ");
    Serial.println(valorLDR);
}

void LdrSensorReadingLed(){

```

```
valorLDR = analogRead(pinLDR);

if(valorLDR<1000 && !ledOutdoorOn){

ledOutdoorOn=true;

digitalWrite(pinLedOutdoor,HIGH);

dataLed = newLedData("Led Exterior","Recámara 1","CAORGRIS","true");

postRequest("/bedrooms/",dataLed);

}else if(valorLDR>1000 && ledOutdoorOn){

ledOutdoorOn=false;

digitalWrite(pinLedOutdoor,LOW);

dataLed = newLedData("Led Exterior","Recámara

1","CAORGRIS","false");postRequest("/bedrooms/",dataLed);

}

}

void insertActionsDefault(){

dataServo = newServoData("Ventana Izquierda","Recámara 1","CAORGRIS","false");

postRequest("/bedrooms/",dataServo);

dataServo = newServoData("Ventana Derecha","Recámara 1","CAORGRIS","false");

postRequest("/bedrooms/",dataServo);

dataServo = newServoData("Puerta","Recámara 1","CAORGRIS","false");

postRequest("/bedrooms/",dataServo);

dataLed = newLedData("Led Interior","Recámara 1","CAORGRIS","false");

postRequest("/bedrooms/",dataLed);

dataLed = newLedData("Led Exterior","Recámara 1","CAORGRIS","false");
```

```

postRequest("/bedrooms/",dataLed);

dataFan = newFanData("Recámara 1","CAORGRIS","false");

postRequest("/bedrooms/",dataFan);

}

void handleOptions(WiFiClient client) {

client.println("HTTP/1.1 200 OK");

client.println("Access-Control-Allow-Origin: *");

client.println("Access-Control-Allow-Methods: POST, GET, OPTIONS");

client.println("Access-Control-Allow-Headers: Content-Type");

client.println("Access-Control-Max-Age: 10000");

client.println("Content-Length: 0");

client.println();

}

String newServoData (String paramName, String paramLocation, String paramOwner,
String
paramValue){

return "{\"arduinoIp\": \"" + String(ipString) + "\",\"type\": \"Actuador\", \"name\": \"" +
paramName + "\", \"brand\": \"TowerPro\", \"model\": \"SG90\", \"specifications\":
[{\"name\": \"Velocidad de operación\", \"value\": 0.12, \"unit\":
\"segundos/60°\"}, {\"name\":
\"Torque\", \"value\": 2.5, \"unit\": \"kg-cm\"}, {\"name\": \"Voltage de
operación\", \"value\":

```

```

5, \"unit\": \"V\"}, {\"name\": \"Corriente de operación\", \"value\": 10, \"unit\":
\"mA\", \"type\":
\"VCD\"}, {\"name\": \"Consumo eléctrico\", \"value\": 0.5, \"unit\": \"W\"}], \"location\":
\"\" +
paramLocation + "\", \"status\": \"Disponible\", \"owner\": \"\" + paramOwner +
\", \"readings\": [
], \"actions\": [{\"name\": \"Activación mecánica\", \"value\": \" + String(paramValue) +
\"}]}\";
}

String newLedData(String paramName, String paramLocation, String paramOwner,
String
paramValue){
return \"{\\\"arduinoIp\\\": \"\" + String(ipString) + \"\", \\\"type\\\": \\\"Actuador\\\", \\\"name\\\": \"\" +
paramName + \"\", \\\"brand\\\": \\\"Genérico\\\", \\\"model\\\": \\\"Diodo LED\\\", \\\"specifications\\\":
[{\\\"name\\\": \\\"Voltage de operación\\\", \\\"value\\\": 3.3, \\\"unit\\\": \\\"V\\\"}], \\\"location\\\": \"\" +
paramLocation + \"\", \\\"status\\\": \\\"Disponible\\\", \\\"owner\\\": \"\" + paramOwner +
\", \\\"readings\\\": [
], \\\"actions\\\": [{\\\"name\\\": \\\"Activación eléctrica\\\", \\\"value\\\": \" + String(paramValue) +
\"}]}\";
}

String newLdrData(String paramLocation, String paramOwner){
return \"{\\\"arduinoIp\\\": \"\" + String(ipString) + \"\", \\\"type\\\": \\\"Sensor\\\", \\\"name\\\":

```

```

    \"Fotorresistencia\", \"brand\": \"Genérico\", \"model\": \"LDR\", \"specifications\":
    [{\"name\":
    \"Rango espectral\", \"minValue\": 400, \"maxValue\": 700, \"unit\": \"nm\"}, {\"name\":
    \"Rango
    de respuesta\", \"minValue\": 0, \"maxValue\": 1024}, {\"name\": \"Voltage de
    operación\", \"value\": 5, \"unit\": \"V\"}, {\"name\": \"Corriente de operación\", \"value\":
    0.45, \"unit\": \"mA\", \"type\": \"VCD\"}, {\"name\": \"Consumo de operación\", \"value\":
    1.56, \"unit\": \"W\"}], \"location\": \"\" + paramLocation + \"\", \"status\":
    \"Disponible\", \"owner\":
    \"\" + paramOwner + \"\", \"readings\": [{\"name\": \"Detección de
    Iluminación\", \"value\": \"\" +
    String(valorLDR) + \"\"}], \"actions\": [ ]}];
    }

    String newTempHumData(String paramLocation, String paramOwner){
    return \"{\\\"arduinoIp\\\": \"\" + String(ipString) + \"\", \"type\": \"Sensor\", \"name\":
    \"Temperatura
    y Humedad\", \"brand\": \"Genérico\", \"model\": \"DHT11\", \"specifications\": [{\"name\":
    \"Rango de medición de temperatura\", \"minValue\": 0, \"maxValue\": 50, \"unit\":
    \"°C\"}, {\"name\": \"Rango de medición de humedad\", \"minValue\": 20, \"maxValue\":
    90, \"unit\": \"%\"}, {\"name\": \"Voltage de operación\", \"value\": 5.5, \"unit\":
    \"V\"}, {\"name\":
    \"Corriente de operación\", \"value\": 2.5, \"unit\": \"mA\", \"type\": \"VCD\"}, {\"name\":

```

```

    \"Consumo eléctrico\", \"value\": 0.00125, \"unit\": \"W\"}], \"location\": \"\" +
    paramLocation +
    "\", \"status\": \"Disponible\", \"owner\": \"\" + paramOwner + "\", \"readings\": [{ \"name\":
    \"Detección de Temperatura\", \"value\": \" + String(t) + \", \"measurementUnit\":
    \"°C\"}, { \"name\":
    \"Detección de Humedad\", \"value\": \" + String(h) + \", \"measurementUnit\": \"%\"} }];
    }String newFanData(String paramLocation, String paramOwner, String paramValue){
    return \"{ \"arduinoIp\": \"\" + String(ipString) + \"\"
    , \"type\": \"Actuador\", \"name\":
    \"Ventilador\", \"brand\": \"SY\", \"model\": \"DC BRUSHLESS FAN\", \"specifications\":
    [{ \"name\":
    \"Tamaño\", \"value\": 30, \"unit\": \"mm\"}, { \"name\": \"Peso\", \"value\": 9, \"unit\":
    \"g\"}, { \"name\": \"Sonido\", \"value\": 25, \"unit\": \"dB\"}, { \"name\": \"Velocidad de
    operación\", \"value\": 8000, \"unit\": \"rpm\"}, { \"name\": \"Desplazamiento de
    aire\", \"value\":
    3.3, \"unit\": \"cfm\"}, { \"name\": \"Voltage de operación\", \"value\": 5, \"unit\":
    \"V\"}, { \"name\":
    \"Corriente de operación\", \"value\": 0.13, \"unit\": \"A\", \"type\": \"VCD\"}, { \"name\":
    \"Consumo eléctrico\", \"value\": 1.56, \"unit\": \"W\"}], \"location\": \"\" + paramLocation
    +
    "\", \"status\": \"No disponible\", \"owner\": \"\" + paramOwner + "\", \"readings\": [
    ], \"actions\":
    [{ \"name\": \"Activación mecánica\", \"value\": \" + String(paramValue) + \"\"} ]}";

```

El código presentado muestra cómo es un código principal para el funcionamiento (automatización).

## DashBoard Web y Móvil



Figura 20: Wireframes UI/UX

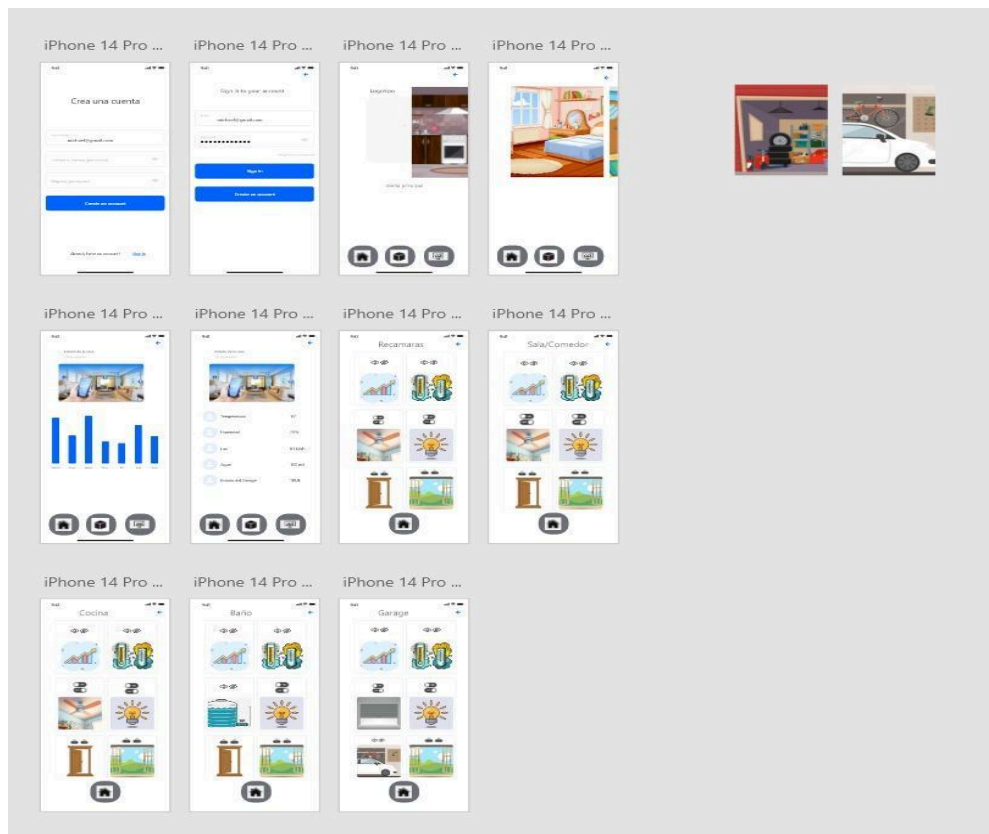


Figura 21: Mockups Móviles UI/UX





## **Documento de despliegue y escalabilidad**

En este documento de despliegue se encargará de detallar los pasos necesarios para desplegar y escalar el sistema de automatización de una casa inteligente que está basada en sensores y actuadores IoT. Lo cual este sistema incluye una aplicación móvil como web para su producción.

### **3.1 Instalación**

#### **3.1.1 Requisitos Previos**

Los siguientes son los requisitos necesarios previos que se recomiendan tener antes de iniciar la instalación:

##### **Hardware**

- Computadora o laptop con la capacidad suficiente para poder alojar la aplicación web.
- Celular para poder instalar o ejecutar la aplicación móvil.
- Acceso a internet ya sea por medio de un router con conectividad a esta.
- Dispositivos IoT adquiridos (sensores de temperatura, humedad, etc.).

##### **Software**

- Sistema operativo: principalmente se recomienda Windows.
- Base de Datos: MongoDB.
- Backend: Node.js con Express.js.
- Control de versiones: Github.
- Frontend web: Desarrollada en Angular.
- Aplicación móvil: Desarrollada en React Native.

- Plataforma IoT: Arduino IDE.

### 3.1.2 Instrucciones de Instalación

#### Instalación de la Aplicación Móvil

1. Clonar el repositorio de Git desde la consola y entrar a la carpeta:

```
git clone https://github.com/usuario/proyecto-iHome.git  
cd proyecto-iot/App Móvil
```

2. Instalar las dependencias básicas desde la consola:

```
Npm install
```

3. Compilar y ejecutar en un dispositivo físico (Android):

```
Npx run Android
```

### 3.1.3 Escalabilidad

La escalabilidad es un factor necesario para la eficiencia del sistema de iHome, se debe considerar los siguientes aspectos:

#### 1. Escalabilidad Horizontal

- configurar réplicas locales o en la nube para simular un entorno distribuido.
- Implementar un sistema de caché para reducir la carga de la base de datos.

#### 2. Escalabilidad Vertical

- Aumentar los recursos del servidor según la demanda.
- Optimizar consultas a la base de datos para mejorar el rendimiento.
- Monitorear el uso de CPU, RAM y tráfico con herramientas como Prometheus y Grafana.

## 3.2 Actualizaciones y Mantenimiento

### 3.2.1 Cómo realizar actualizaciones OTA en el firmware IoT

Para actualizar el firmware de los dispositivos IoT de manera remota:

1. Configurar un servidor de actualizaciones OTA usando AWS IoT, Firebase Remote Config o un servidor HTTP propio.
2. Subir el nuevo firmware a un repositorio accesible para los dispositivos.
3. Configurar los dispositivos IoT para comprobar periódicamente la versión más reciente.
4. Descargar e instalar el firmware de forma segura y validar la actualización antes de aplicarla.

### 3.2.2 Monitoreo del sistema con Grafana y Prometheus

1. Instalar Prometheus en el servidor:
2. `sudo apt install -y prometheus`
3. Configurar Prometheus para recolectar métricas del backend y la base de datos.
4. Instalar Grafana para visualizar los datos:
5. `sudo apt install -y grafana`
6. Configurar dashboards en Grafana para monitorear el rendimiento del sistema, consumo de recursos y actividad de los sensores IoT.

7. Establecer alertas en Grafana para notificar anomalías en el sistema.

Este proceso garantiza un despliegue eficiente y un mantenimiento adecuado para el sistema de automatización de la casa inteligente a través del uso de sensores y actuadores.

## **Capítulo 4 Resultados y Conclusiones**

### Resultados

En este apartado deberá redactarse los resultados y entregables obtenidos, haciendo la comparativa de un antes y un después en las actividades cotidianas de la empresa, es importante denotar aquellas fortalezas, debilidades, oportunidades y amenazas. Incluir resultados a través de pruebas, implementación de métricas e integrar que normas validaron las métricas.

### Conclusiones

Reflexión comparativa entre la problemática y los resultados, el alumno deberá evaluar si los objetivos fueron o no alcanzados.

### Recomendaciones

Indicaciones que en base a la experiencia el alumno deja sobre el contexto del problema, los recursos o los entregables generados con la finalidad de promover la mejora continua de la empresa.

## Lista de referencias

- [1] Gutiérrez, A., & López, R. (2021). *Historia y evolución de la domótica*. Editorial Tecno.
- [2] Statista. (2023). *Smart Home Market Report 2023*. Disponible en: <https://www.statista.com>.
- [3] García, J., et al. (2022). *Eficiencia energética en hogares inteligentes*. Revista de Tecnología y Energía, 35(2), 45-58.
- [4] Ashton, K. (2019). *Internet of Things: A brief history*. Tech Journal, 12(3), 18-25.
- [5] Fette, I., & Melnikov, A. (2011). *The WebSocket Protocol*. IETF RFC 6455.
- [6] Stallings, W. (2018). *Cryptography and Network Security: Principles and Practice*. Pearson.
- [7] <https://www.ferrovial.com/es/recursos/domotica/>
- [8] <https://www.gluo.mx/blog/backend-que-es-y-para-que-sirve>
- [9] Angular. (2025). *Angular - Framework para el desarrollo web*. Recuperado de <https://angular.io/docs>
- [10] MongoDB. (2025). *MongoDB: The database for modern applications*. Recuperado de <https://www.mongodb.com>
- [11] <https://www.redeszone.net/tutoriales/internet/que-es-websocket/>
- [12] Bahga, A., & Madisetti, V. (2014). *Internet of Things: A Hands-On Approach*. 1st ed. Vijay Madisetti.



## Apéndice

Las tablas y figuras pueden ir en el apéndice como se mencionó anteriormente. También es posible usar el apéndice para incluir datos en bruto, instrumentos de investigación y material adicional.

## Vita

Acá se incluye una breve biografía del autor de la tesis.

