ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA DE SISTEMAS

DISEÑO DE PROTOTIPO IOT BASADO EN LA PLACA NODEMCU ESP32 PARA CONTROLAR Y MEDIR EL DISTANCIAMIENTO SOCIAL EN LA FIS-EPN

PROYECTO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN SISTEMAS INFORMÁTICOS Y DE COMPUTACIÓN

CRISTIAN SEGUNDO FLORES ASIMBAYA

cristian.flores03@epn.edu.ec

Director: MSc. HERNAN DAVID ORDOÑEZ CALERO

hernan.ordonez@epn.edu.ec

Codirector: Ph.D. DIANA CECILIA YACCHIREMA VARGAS

diana.yacchirema@epn.edu.ec

Quito, Marzo 2023

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por Cristian Segundo Flores Asimbaya, bajo nuestra supervisión.

Firmado electrónicamente por:
HERNAN DAVID ORDONEZ CALERO

MSc. Hernán David Ordoñez Calero

DIRECTOR DE PROYECTO

Firmado digitalmente por DIANA CECILIA YACCHIREMA VARGAS

Ph.D. Diana Cecilia Yacchirema Vargas

CODIRECTOR DE PROYECTO

DECLARACIÓN

Yo, Cristian Segundo Flores Asimbaya declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedo mis derechos de propiedad intelectual correspondiente a este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normativa institucional vigente.

Firmado electrónicamente por:
CRISTIAN SEGUNDO
FLORES ASIMBAYA

Cristian Segundo Flores Asimbaya

DEDICATORIA

A Dios por guiarme a lo largo de toda mi vida y por colocar a las personas correctas en mi camino.

A mi madre por lo valiente y fuerte que ha sido críar a un hijo desde la distancia, por todos los consejos que me ha dado en mi adolescencia, por la gran fuerza que tiene para trabajar y por lo amorosa que es con migo en todo momento.

A mi padre por los buenos valores que me inculcó, por todo el apoyo incondicional a mis estudios y por siempre estar al pendiente de mí.

A mi novia por acomapañarme en toda mi vida universitaria, por darme fuerzas para seguir adelante en todo momento, por siempre ayudarme a ser la mejor versión de mi mismo y por ser la luz que guía mi sendero.

A mi amigo Juan Carlos Tituaña por haberme ayudado en mis estudios y por ser uno de mis mejores amigos. Descanza en paz querido amigo.

A mi compañero de estudios en la FIS Henry Aguilar que fue un amigo incondicional en toda mi vida universitaria. Descanza en paz querido amigo.

A mi segunda madre Gloria Collaguazo por enseñarme la humildad, la honestidad y darme todo su cariño. Descanza en paz mamá.

Cristian Flores

AGRADECIMIENTO

A mi hermano y a mi cuñada por brindarme su ayuda y cobijo en su hogar durante toda mi carrera académica. Gracias a su generosidad, pude centrarme en mis estudios y lograr mis metas. También quiero agradecer a mis hermanas por estar siempre al pendiente de mi progreso y apoyarme en todo momento y aspecto de mi vida.

A mi tercera madre Zoila Mingan por su ayuda y cuidado cada día a lo largo de toda mi vida academica.

A mis amigos de la universidad por todas las risas, las malas noches y las festejadas por culminar un semestre exitoso.

A mis primos Santos Collaguazo por ser mis hermanos en espiritu y llenarme el corazon por el vacio que dejo mi madre al partir al extragero para darme un futuro mejor, siempre les estaré eternamente agradecido a los cuatro por guiarme en los caminos de Dios y tratarme como parte de su familia.

A mi director de tesis MSc Hernán Ordoñez por creer en mi y darme su guía para culminar este gran hito de mi vida.

Cristian Flores

ÍNDICE DE CONTENIDO

Tabla de contenido

1 INTRODUCCIÓN 1

1.1. Planteamiento del problema 1

1.2. Objetivos 2

Objetivo General 2

Objetivos Específicos 2

1.3. Marco Teórico 3

Distanciamiento social 3

Placa NodeMCU ESP32 3

Arquitecturas IoT 5

BLE (Bluetooth Low Energy) 7

Cálculo de distancia mediante RSSI 8

Tecnologías de Aplicación 9

Scrum 13

Pruebas del sistema 16

2 METODOLOGÍA 19

2.1. Definición de roles Scrum 19

2.2. Levantamiento de Requerimientos 19

Historias de Usuario 19

Product Backlog Inicial 20

2.3. Sprint 0 21

Sprint Planning **21**

Desarrollo del Sprint **22**

Sprint Review **25**

Sprint Retrospective **27**

2.4. Sprint 1 ..... **28**

Sprint Planning **28**

Desarrollo del Sprint **28**

Sprint Review **32**

Sprint Retrospective **34**

2.5. Sprint 2 ..... **35**

Sprint Planning **35**

Desarrollo del Sprint **36**

Sprint Review **38**

Sprint Retrospective **40**

2.6. Sprint 3 ..... **41**

Sprint Planning **41**

Desarrollo del Sprint **42**

Sprint Review **45**

Sprint Retrospective **46**

2.7. Sprint 4 ..... **47**

Sprint Planning **47**

Desarrollo del Sprint **48**

Sprint Review **51**

Sprint Retrospective **53**

2.8. Sprint 5 ..... **54**

Sprint Planning **54**

Desarrollo del Sprint **55**

Sprint Review **58**

Sprint Retrospective **59**

2.9. Sprint 6 ..... **60**

Sprint Planning **60**

Desarrollo del Sprint **61**

Sprint Review **63**

Sprint Retrospective **65**

2.10. Sprint 7 ..... **66**

Sprint Planning **66**

Desarrollo del Sprint **67**

Sprint Review **69**

Sprint Retrospective **70**

3 PRUEBAS DEL SISTEMA **72**

3.1. Pruebas unitarias. **72**

3.2. Pruebas de Integración. **75**

Prueba de integración 1 ..... **75**

Prueba de integración 2 ..... **75**

Prueba de integración 3 ..... **76**

3.3. Pruebas de usabilidad. **77**

System Usability Scale (SUS) 77

4 CONCLUSIONES Y RECOMENDACIONES 85

4.1. Conclusiones 85

4.2. Recomendaciones 87

5 REFERENCIAS BIBLIOGRÁFICAS 88

6 ANEXOS 93

6.1. Anexo 1: Historias de usuario Sprint 0 93

6.2. Anexo 2: Historias de usuario Sprint 1 95

6.3. Anexo 3: Historias de usuario Sprint 2 97

6.4. Anexo 4: Historias de usuario Sprint 3 99

6.5. Anexo 5: Historias de usuario Sprint 4 101

6.6. Anexo 6: Historias de usuario Sprint 5 103

6.7. Anexo 7: Historias de usuario Sprint 6 105

6.8. Anexo 8: Historias de usuario Sprint 7 107

6.9. Anexo 9: Product backlog para el sprint 5 109

6.9. Anexo 10: Evidencias encuesta SUS 109

ÍNDICE DE FIGURAS

Figura 1. Placa NodeMCU ESP-32..... 3

Figura 2. Diagrama de puertos del NodeMCU ESP32 4

Figura 3. Arquitectura IoT Genérica de 5 capas[16] 5

Figura 4. Arquitectura IoT Genérica de 4 capas[17] 6

Figura 5. Arquitectura IoT Genérica de 3 capas[17] 6

Figura 6. Pila de protocolo BLE [20]..... 8

Figura 7. Scrum, principales artefactos y eventos. [40] 14

Figura 8. Arquitectura IoT de 4 capas usada en el presente prototipo IoT..... 23

Figura 9. Diagrama de flujo del algoritmo de control de contactos. 25

Figura 10. Gráfico Burndown Sprint 0..... 27

Figura 11. Diagrama de la conexión de los componentes del dispositivo IoT..... 31

Figura 12. Dispositivo IoT compactado..... 32

Figura 13. Gráfica Burndown Sprint 1 35

Figura 14. Diagrama de flujo del algoritmo de control de contactos 36

Figura 15. Función para el cálculo de distancia entre los dispositivos.. 37

Figura 16. Despliegue exitoso en la placa de desarrollo.. 37

Figura 17. Conexión entre la placa y el ordenador. 38

Figura 18. Conexión entre la placa y el ordenador. 38

Figura 19. Gráfico Burndown Sprint 2..... 41

Figura 20. Rutas del portal web. 43

Figura 22. Petición HTTP de tipo GET hacia el proyecto base back-end..... 44

Figura 23. Porcentaje de líneas de código cubiertas por pruebas unitarias. 44

Figura 24. Gráfico Burndown Sprint 3 47

Figura 25. Respuesta del endpoint auth/login. 49

Figura 26. Información del usuario almacenada en la base de datos. 49

Figura 27. Localstorage del portal web. 50

Figura 28. Formulario de registro para el portal web. 51

Figura 29. Gráfico Burndown Sprint 4. 54

Figura 30. Pantalla de vincular/desvincular flujo vincular. 57

Figura 31. Pantalla de vincular/desvincular flujo desvincular. 57

Figura 32. Gráfico Burndown Sprint 5. 60

Figura 33. Grafo de contactos realizados. 63

Figura 34. Tabla de históricos de contactos 63

Figura 35. Gráfico Burndown Sprint 6. 66

Figura 36. Botón de reportarse enfermo. 68

Figura 37. Alerta de posible contagio. 68

Figura 38. Gráfico Burndown Sprint 7. 71

Figura 39. Pruebas unitarias del servidor. 73

Figura 40. Pruebas unitarias del aplicativo web 74

Figura 41. Resultados encuesta SUS pregunta 1. 77

Figura 42. Resultados encuesta SUS pregunta 2. 78

Figura 43. Resultados encuesta SUS pregunta 3. 78

Figura 44. Resultados encuesta SUS pregunta 4. 79

Figura 45. Resultados encuesta SUS pregunta 5. 80

Figura 46. Resultados encuesta SUS pregunta 6. 80

Figura 47. Resultados encuesta SUS pregunta 7. 81

Figura 48. Resultados encuesta SUS pregunta 8. 82

Figura 49. Resultados encuesta SUS pregunta 9. 82

Figura 50. Resultados encuesta SUS pregunta 10. 83

Figura 51. Gráfico Preguntas vs Puntaje promedio obtenido. 84

Figura 52. Grafo de contactos realizados..... 111

Figura 53. Módulo de vinculación de dispositivo. 111

Figura 54. Módulo de login del sistema. 112

ÍNDICE DE TABLAS

Tabla 1. Términos de programación citados en la elaboración de la solución IoT 9

Tabla 2. Marcos de trabajo y librerías usadas en el desarrollo de la solución IoT. 10

Tabla 3. Entornos de desarrollo usados y gestor de base de datos. 12

Tabla 4. Interpretación de las puntuaciones de SUS mediante una escala de calificación. 17

Tabla 5. Roles realizados por el equipo Scrum para la elaboración del trabajo. 19

Tabla 6. Historias épicas para desarrollar. 19

Tabla 7. Product Backlog al iniciar la elaboración del proyecto plantedao. 20

Tabla 8. Historias de usuario seleccionadas del Product Backlog del sprint 0. 21

Tabla 9. Sprint Backlog del Sprint 0. 21

Tabla 10. Criterios de admisión del sprint 0. 26

Tabla 11. Product Backlog al finalizar sprint 0. 26

Tabla 12. Historias de usuario seleccionadas del Product Backlog. 28

Tabla 13. Sprint Backlog del Sprint 1. 28

Tabla 14. Componentes del dispositivo IoT. 30

Tabla 15. Costos del dispositivo IoT. 31

Tabla 16. Criterios de aceptación del Sprint 1. 33

Tabla 17. Product Backlog luego de finalizar el primer sprint. 33

Tabla 18. Historias de usuario seleccionadas del Product Backlog del sprint 2. 35

Tabla 19. Sprint Backlog del Sprint 2. 36

Tabla 20. Criterios de aceptación del sprint 2. 38

Tabla 21. Product backlog luego de dar por terminado el sprint 2. 39

Tabla 22. Historias de usuario seleccionadas del Product Backlog del sprint 3. 41

Tabla 23. Sprint Backlog del sprint 3. 42

Tabla 24. Criterios de aceptación del sprint 3. 45

Tabla 25. Product backlog al dar por terminado el sprint 3. 46

Tabla 26. Historias de usuario seleccionadas del Product Backlog del sprint 4. 47

Tabla 27. Sprint Backlog del sprint 4. 48

Tabla 28. Criterios de aceptación del sprint 4. 51

Tabla 29. Product backlog tras finalizar el sprint 4. 52

Tabla 30. Historias de usuario seleccionadas del Product Backlog del sprint 5. 54

Tabla 31. Sprint Backlog del sprint 5. 55

Tabla 32. Criterios de aceptación del sprint 5. 58

Tabla 33. Product backlog tras finalizar el sprint 5. 59

Tabla 34. Historias de usuario seleccionadas del Product Backlog del sprint 6. 61

Tabla 35. Sprint Backlog del sprint 6. 61

Tabla 36. Criterios de aceptación del sprint 6. 63

Tabla 37. Product backlog tras finalizar el sprint 6. 64

Tabla 38. Historias de usuario seleccionadas del Product Backlog del sprint 7. 66

Tabla 39. Sprint Backlog del sprint 7. 67

Tabla 40. Criterios de aceptación del sprint 7. 69

Tabla 41. Product backlog tras finalizar el sprint 7. 70

Tabla 42. Caso de prueba 1 – Registro e inicio de sesión. 75

Tabla 43. Caso de prueba 2 – Vinculación de dispositivo IoT. 75

Tabla 44. Caso de prueba 3 – Mostrar grafo de contactos recientes. 76

Tabla 45. Historia de usuario PDS1-03. 93

Tabla 46. Historia de usuario PDS2-01. 93

Tabla 47. Historia de usuario PDS1-01. 95

Tabla 48. Historia de usuario PDS1-02. 95

Tabla 49. Historia de usuario PDS2-02. 97

Tabla 50. Historia de usuario PDS2-03. 97

Tabla 51. Historia de usuario PDS3-01. 99

Tabla 52. Historia de usuario PDS4-01. 99

Tabla 53. Historia de usuario PDS3-04 99

Tabla 54. Historia de usuario PDS3-02 101

Tabla 55. Historia de usuario PDS4-02 101

Tabla 56. Historia de usuario PDS3-03. 103

Tabla 57. Historia de usuario PDS4-03. 103

Tabla 58. Historia de usuario PDS4-06. 104

Tabla 59. Historia de usuario PDS3-05. 105

Tabla 60. Historia de usuario PDS4-04. 105

Tabla 61. Historia de usuario PDS3-06. 107

Tabla 62. Historia de usuario PDS4-05. 107

Tabla 63. Product Backlog adaptado para el sprint 5. 109

RESUMEN

La pandemia del COVID-19 ha presentado un desafío global para la humanidad, y el aislamiento social se ha considerado en la principal medida preventiva para evitar la propagación del virus. En este trabajo, se propone el diseño de un prototipo de solución IoT para controlar y medir el distanciamiento social en la Facultad de Ingeniería de Sistemas de la Escuela Politécnica Nacional. El prototipo utiliza dispositivos IoT que miden la distancia entre ellos mediante la señal RSSI y alertan a los usuarios con una alarma sonora cuando se acercan a menos de 1.5 metros. Los datos obtenidos se envían a una base de datos, que se puede visualizar por medio de un portal web con la finalidad de que los usuarios puedan ver sus registros de contacto.

Se utilizó el desarrollador de proyectos trabajo Scrum para el desarrollo de la solución, lo que permitió mitigar los cambios inesperados en el diseño del prototipo IoT. La solución se probó con éxito con pruebas unitarias usando el marco de trabajo jest.js, así como con pruebas de integración top-down y down-top. Se realizó una encuesta de usabilidad del sistema, que obtuvo una puntuación de 85/100, lo que indica que la gran mayoría de los usuarios encuentra útil y fácil de usar la solución prototipo.

En futuros trabajos, se podrían realizar procesos de miniaturización para reducir el gasto de energía y los costos de construcción. También se recomienda la implementación de algoritmos de reducción de ruido para refinar el proceso de cálculo de distancias entre dispositivos y mejorar la precisión y fiabilidad de la solución.

ABSTRACT

The COVID-19 pandemic has presented a global challenge for humanity, and social distancing has become the primary preventive measure to avoid virus spread. In this work, we propose the design of an IoT prototype solution to control and measure social distancing at the Faculty of Systems Engineering at the Escuela Politécnica Nacional. The prototype utilizes IoT devices that measure the distance between them using RSSI signals and alert users with a sound alarm when they get closer than 1.5 meters. The collected data is sent to a database, which can be viewed through a web portal so that users can see their contact records.

The Scrum project developer was used to develop the solution, which allowed mitigating unexpected changes in the design of the IoT prototype. The solution was successfully tested with unit tests using the jest.js framework, as well as top-down and down-top integration tests. A system usability survey was carried out, which obtained a score of 85/100, indicating that the vast majority of users find the prototype solution useful and easy to use.

In future works, miniaturization processes could be carried out to reduce energy expenditure and construction costs. The implementation of noise reduction algorithms is also recommended to refine the process of calculating distances between devices and improve the accuracy and reliability of the solution.

1 INTRODUCCIÓN

1.1 Planteamiento del problema

La tecnología es una herramienta fundamental que el ser humano ha inventado, la cual no solo ha mejorado su calidad de vida en el día a día, sino que también le ha permitido hacer frente a las adversidades naturales y satisfacer sus necesidades básicas [1]. Desde el 2019, el virus conocido como COVID-19 (C-19) ha provocado la más grande crisis social y económica que ha sufrido la especie humana, desde las grandes pandemias, tales como la peste negra en 1346 D.C., o la gripe española en 1918 D.C [2]. En el año 2020, el C-19 llegó a 213 países del mundo [3], causando que las actividades humanas que requieran de proximidad entre personas tengan que ser realizadas con extrema cautela para evadir contagiarse con C-19. La propagación del virus ocurre principalmente de forma horizontal a través de gotículas que se liberan al toser, estornudar o hablar. Estas gotículas contienen partículas infecciosas que miden más de 5 micrómetros y pueden ser inhaladas por otras personas en las cercanías [4]. Estas partículas al tener un peso representativo, no logran dispersarse con facilidad por lo cual caen rápidamente al suelo. Cabe considerar que una persona puede contagiarse con C-19 si inhala las partículas líquidas procedentes de un individuo infectado por el virus. Por ello, existe alta importancia de mantenerse distanciado de otra persona con mínimo un metro de distancia [5]. Es necesario tener apoyo tecnológico para controlar y medir las distancias entre personas para lograr así reducir el nivel de propagación del C-19. El apoyo tecnológico ayudará al ser humano a cumplir medidas de bioseguridad en contra del C-19.

La mayoría de los países han optado medidas en contra del C-19, estas medidas tuvieron como acción principal la cuarentena de la población [6]. Esta medida de seguridad mostró una recesión económica a nivel global, debido a esto, muchos países apresuraron el regresar a las actividades laborales, teniendo en cuenta el distanciamiento social (DS) de las personas o como la Organización Mundial de la Salud (OMS) lo llama distanciamiento físico. El distanciamiento social significa estar físicamente alejado de otro individuo al menos por 2 metros. La OMS recomienda conservar una distancia mínima de un metro y medio, entre las personas. Esta es una medida precautelar que todas las personas deberían adoptar, incluyendo aquellas personas que se encuentran bien de salud, para evitar su exposición ante el virus [5]. Aquí nace una problemática, ¿Cómo se controla que toda la población este a no menos

de un metro y medio entre ellas?, ¿Se pueden medir estas distancias de alejamiento para recolectar datos y poder analizarlos posteriormente?

La presente investigación, propone generar un diseño e implementación de un prototipo de Internet de las cosas (IoT, Internet of Things por sus siglas en inglés) basado en la placa NodeMCU ESP32, la cual es una placa de baja inversión económica y de alta capacidad de reacción para el prototipado de soluciones IoT [7]. Este diseño e implementación tendrá como objetivos controlar y medir el distanciamiento social de los usuarios (profesores, personal administrativo, personal de apoyo, estudiantes y visitantes recurrentes) de la Facultad de Ingeniería de Sistemas (FIS) de la Escuela Politécnica Nacional (EPN). Este prototipo de dispositivo IoT tendrá la forma de un colgante para cada persona, el cual emitirá un sonido de alerta cuando detecte que la proximidad entre personas sea de menos de un metro y medio. Se enviará los resultados a una base de datos no relacional, donde se registrarán todas las mediciones de los dispositivos. La información obtenida podrá ser procesada para visualizar los datos mediante un portal web a nivel de usuario.

1.2 Objetivos

1.2.1 Objetivo General

Diseñar una solución IoT usando la placa de desarrollo NodeMCU ESP32 que permita controlar el distanciamiento social mediante una alerta sonora de proximidad, cuando los usuarios se acerquen a menos de 1,5 metros de distancia y enviar los datos a una base datos para mantener registros de estos a través de un portal web.

1.2.2 Objetivos Específicos

- Diseñar la arquitectura que permitirá la interacción de los dispositivos IoT usando señales inalámbricas WiFi y Bluetooth para medir el DS menor a 1,5 metros.
- Usar la placa NodeMCU ESP32 como componente principal de la arquitectura hardware del dispositivo IoT.
- Usar el Framework Scrum tanto para los diseños como para la implementación del prototipo que permita el cálculo de distancias entre los dispositivos partiendo de la Received Signal Strength Indicator (RSSI)
- Implementar un servidor de base de datos para la recolección de los datos enviados por los dispositivos IoT.

- Diseñar un portal web para la visualización de la información obtenida por la red de dispositivos IoT.

1.3 Marco Teórico

1.3.1 Distanciamiento social

El distanciamiento social es una solución simple y efectiva para prevenir el contagio de C-19 y cualquier otra enfermedad de contagio por proximidad. De hecho, muchos estudios han demostrado que la distancia óptima para evitar contagios es de al menos 1 metro [8], [9]. Las autoridades sanitarias y expertos concuerdan que es de vital importancia el correcto manejo del distanciamiento físico y el evitar aglomeraciones para minimizar el alcance y la velocidad con la que se propaga el C-19 [10]. Visto que, la mayoría de los expertos alrededor del mundo han llegado a la conclusión que el distanciamiento social puede ser una arma efectiva y eficiente para combatir esta enfermedad de contagio por proximidad es que la gran mayoría de naciones alrededor del mundo han optado por alentar fuertemente a su población en conservar un distanciamiento prudente entre personas y practicar este distanciamiento social [11]

1.3.2 Placa NodeMCU ESP32

La NodeMCU ESP32 es una placa de desarrollo la cual contiene integrada en ella un único chip System On Chip (SoC) llamado ESP32 desarrollado por Espressif Systems, el cual incorpora un módulo de Wi-Fi y Bluetooth de 2,4 GHz diseñado con tecnología de 40 nm de ultra bajo consumo de la empresa Taiwán Semiconductor Manufacturing Co. (TSMC) y que además está diseñado para obtener el mejor rendimiento de potencia, mostrando gran robustez, versatilidad y confiabilidad dentro de una serie de aplicaciones y escenarios a ser utilizado [12]. Gracias a estas características, esta placa está diseñada para ser usada en conjunto a aplicaciones móviles, dispositivos electrónicos portátiles e IoT. En la Figura 1, se muestra la placa.

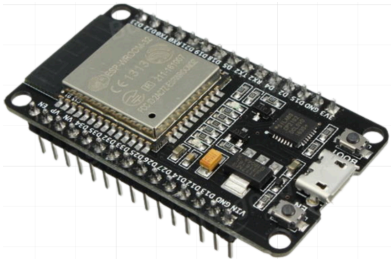


Figura 1. Placa NodeMCU ESP-32

Además de poseer dichas características que lo hacen candidato perfecto para diseñar soluciones IoT, la placa tiene un bajo costo de adquisición en el mercado. ESP32 utiliza un semiconductor complementario de óxido metálico (CMOS) que un instrumento que contiene baja capacidad de memoria totalmente integrada en placa base del equipo y que al mismo tiempo integra circuitos de calibración avanzados que permiten que la funcionalidad en conjunto elimine imperfecciones de acoplamiento con circuitos externos o que se ajuste a los cambios de las condiciones externas al chip [12] [13]. En otras palabras, ESP32 tiene la capacidad para ser usado de manera masiva en soluciones IoT ya que no requiere equipos Wi-Fi o bluetooth externos, costosos y especializados que se acoplen al circuito del dispositivo IoT diseñado.

Las características técnicas principales del NodeMCU ESP32 son [14]:

- Wifi: 802.11 b/g/n/e/i (802.11n @ 2.4 GHz hasta 150 Mbit/s).
- Bluetooth: Low Energy (BLE) y clásico.
- Procesador: Tensilica Xtensa Dual-Core 32-bit LX6.
- Frecuencia de reloj: 160 a 240 Mhz.
- Memoria: 448 KByte ROM, 520 KByte SRAM, 16KByte SRAM in Real Time Clock (RTC) QSPI Flash/SRAM, 4 Mbytes.
- Pines totales: 38.
- Voltaje de alimentación (USB): 5V DC.
- Voltaje entradas y salidas (I/O): 3.3V DC.

En la Figura 2, se expone la distribución correspondiente a los puertos que posee la placa [15]:

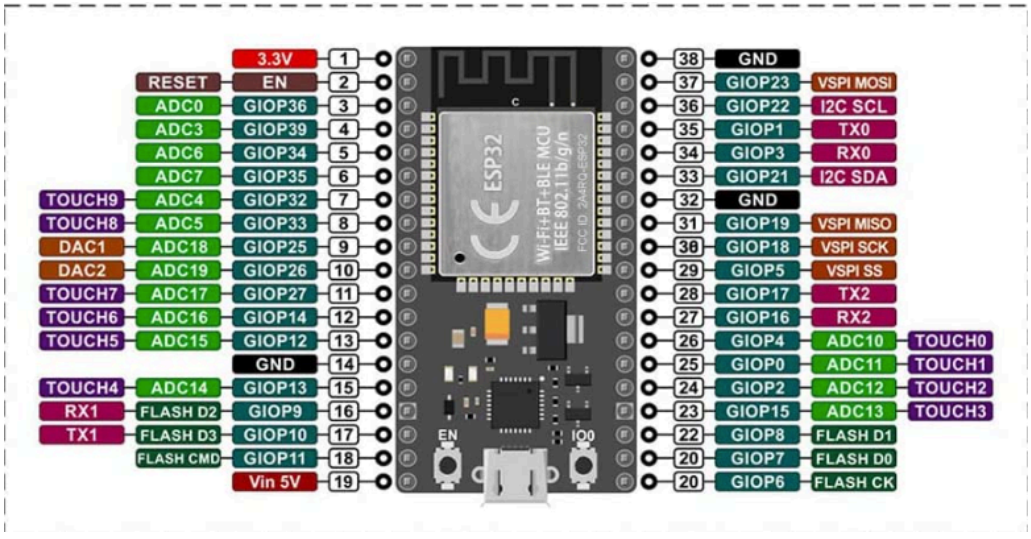


Figura 2. Diagrama de puertos del NodeMCU ESP32

Entre ventajas del dispositivo NodeMCU ESP32 podemos destacar las siguientes:

- Bajo coste de adquisición.
- Alto desempeño y bajo consumo de energía.
- Soporte dual de Wi-Fi y Bluetooth en una placa de tamaño reducido.
- Alta robustez, versatilidad y fiabilidad.

1.3.3 Arquitecturas IoT

En diversas soluciones de IoT, se han utilizado arquitecturas que se dividen en 5 capas distintas, como se aprecia en la Figura 3 [16]. Es importante recalcar que otros autores optan por describir una arquitectura de 4 capas y de 3 capas como muestran las Figuras 4 y 5 respectivamente.

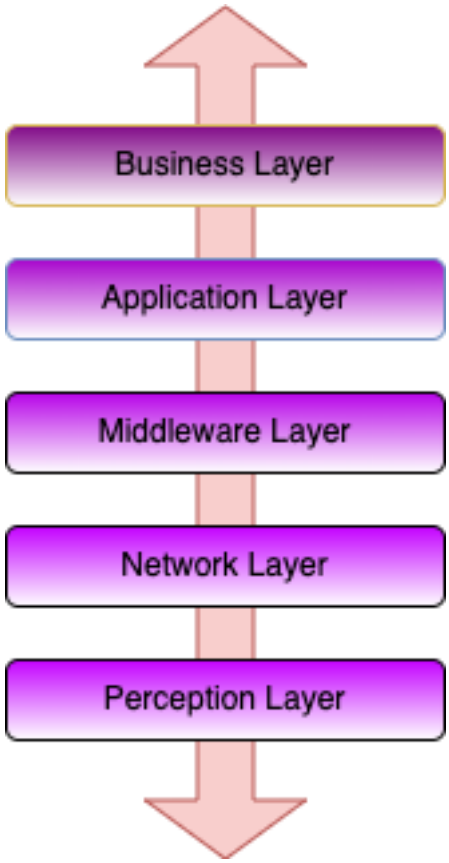


Figura 3. Arquitectura IoT Genérica de 5 capas[16]

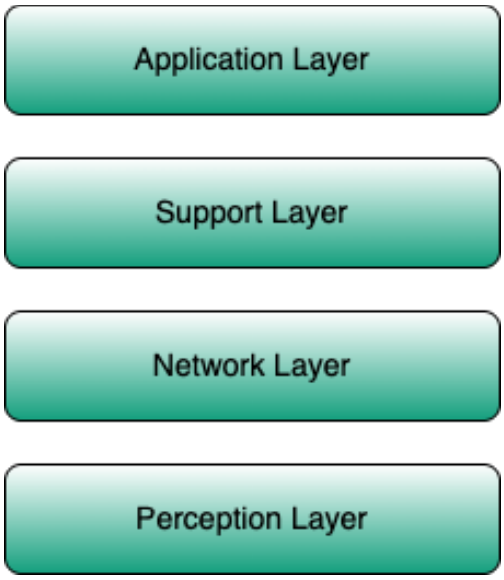


Figura 4. Arquitectura IoT Genérica de 4 capas[17]

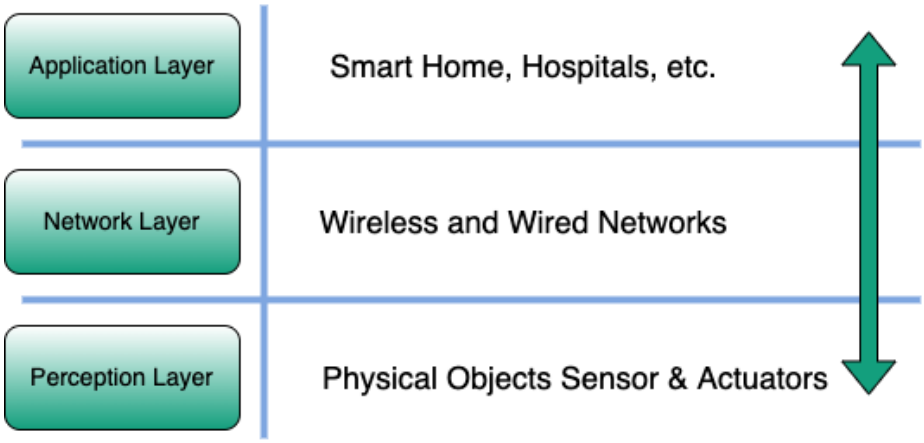


Figura 5. Arquitectura IoT Genérica de 3 capas[17]

1.3.3.1 Arquitectura de 4 Capas

La arquitectura de 4 capas surge a partir de la arquitectura de 3 capas, considerada la más sencilla de todas, pero que no ha sido suficiente para cumplir con los requerimientos de IoT debido a su constante evolución [17]. Es por eso que los investigadores han propuesto una nueva arquitectura que, manteniendo las tres capas ya existentes, agrega una capa adicional de soporte. A continuación, se presentan las cuatro capas de esta arquitectura:

- Capa de percepción: También conocida como la capa sensorial o capa de dispositivos y cuya función principal es la de recolectar información por medio de

dispositivos que actúan como sensores y en el caso de que existan dispositivos actuadores, es decir, dispositivos que al recibir una señal generan un movimiento físico, éstos realizarán una acción dependiendo de la información recolectada por los dispositivos con sensores [16] [17]. Algunos ejemplos de sensores son: sensores de humedad, de temperatura, de proximidad, etc. Y ejemplos de dispositivos actuadores son: motores, relés, solenoides, etc. Cabe recalcar que cada uno de los dispositivos contenidos dentro de esta deben tener un identificador único dentro de la red [18].

- Capa de red: También conocida como la capa de transporte. Ésta es la segunda capa de la arquitectura y es el corazón de todo el funcionamiento IoT, esta capa se encarga de transportar y transmitir la información obtenida por los sensores. El medio de transmisión de información puede ser cableado o inalámbrico. Esta capa también tiene la responsabilidad de conectar los dispositivos IoT con los dispositivos de red y estos a su vez con la internet [17].
- Capa de soporte: Esta capa tiene dos responsabilidades principales. La primera es que debe verificar que la información es enviada únicamente por usuarios auténticos del sistema [17]. Para verificar a los usuarios se pueden usar distintos métodos de autenticación como el uso de secretos tales como llaves de interfaz de programación (API key, por su definición en inglés), llaves de intérprete de órdenes seguro (SSH key, por su definición en inglés), claves y contraseñas. La segunda responsabilidad es manejar el correcto procesamiento y almacenamiento en bases de datos de la información obtenida por los usuarios del sistema [16]. La información para procesar puede llegar tanto de la capa de red, la inferior, como de la capa de aplicación, la superior.
- Capa de aplicación: Ésta capa es donde se definen todas las posibles aplicaciones que exploten los datos obtenidos por la tecnología IoT. Por lo general ejemplos de aplicaciones IoT pueden ser Casas Inteligentes, Ciudades Inteligentes, Salud Inteligente, Rastreo Animal, etc. La responsabilidad general de esta capa es proporcionar los servicios a las aplicaciones [17]. Es común que los servicios varíen para cada aplicación porque los servicios dependen de la información que es obtenida por los sensores y dependiendo del problema a resolver, estos pueden recolectar diferente información.

1.3.4 BLE (Bluetooth Low Energy)

El BLE, o también llamado Bluetooth inteligente, ha emergido como una de las mejores tecnologías inalámbricas de bajo poder [19]. A diferencia del Bluetooth tradicional, este

fue diseñado específicamente en busca de solucionar el al consumo de energía, facilitando el control y monitoreo de las aplicaciones. El BLE es una particularidad distintiva de la especificación de Bluetooth 4.0 [20].

Como en el Bluetooth clásico, protocolo de pila de BLE está conformado por dos partes principalmente; el Controlador y el Host. El mismo que se comprende de capas físicas y la capa de enlace de datos, estas dos capas son comúnmente implementadas en un System-on-Chip (SOC) con una antena integrada. Por otro lado, el Host trabaja en el procesador de las aplicaciones, este incluye todas las capas superiores: Control de enlace lógico (Logical Link Control) y el protocolo de adaptación (Adaptaction Protocol L2CAP), el protocolo de atributo (ATT), el perfil genérico de atributos (GATT) el protocolo de manejo de seguridad (SMP) y el Perfil de Acceso Genérico (GAP). La vinculación generada entre el Host y el Controlador se mantiene estandarizada como la Interfaz Controlador Host (HCI) [20], como se aprecia en la Figura 6.

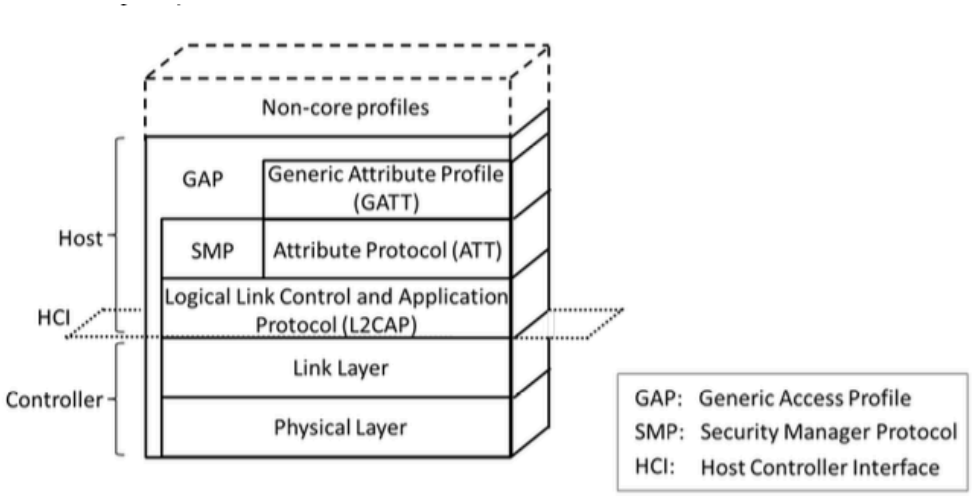


Figura 6. Pila de protocolo BLE [20]

Muchas de las características de este protocolo han sido heredadas del controlador de Bluetooth clásico, por ello se debe tener en cuenta que si el dispositivo únicamente tiene implementado BLE no se podrá comunicar con dispositivos que solo tengan Bluetooth clásico [20].

1.3.5 Cálculo de distancia mediante RSSI

Para realizar el cálculo de la distancia usando la señal RSSI se debe tomar en cuenta la siguiente fórmula que representa la relación entre el indicador de fuerza de señal recibida y la distancia que existe entre el transmisor y el receptor de señal [21] [22].

$$RSSI_{dBm} = -(10\,\eta\,log_{10}(d) + A)$$

Ecuación 1. Relación ente intensidad de señal recibida y distancia [21][22].

Donde η es la constante exponencial de la propagación de señal, d es el distanciamiento existente al emparejar el transmisor y el receptor y A es el valor de indicador de fuerza de reacción recibida a 1 metro de distancia del transmisor en un ambiente sin obstáculos [21].

De esta ecuación empírica se puede fácilmente despejar la distancia quedando la siguiente ecuación:

$$d = 10^{\left(-\frac{RSSI_{dBm}+A}{10\,\eta}\right)}$$

Ecuación 2. Fórmula para calcular la distancia entre emisor y receptor

En ambientes de espacio libre generalmente el valor de η tiene un valor teórico igual a 2, sin embargo, en la práctica puede variar dependiendo del ambiente en el que se vayan a colocar al transmisor y receptor. Gracias a investigaciones previas [23], se tiene un valor de $\eta = 2{,}6$ para ambientes de oficina con particiones construidas con materiales suaves. Usando esta última ecuación se puede obtener un valor estimado de distancia entre dos dispositivos inalámbricos.

1.3.6 Tecnologías de Aplicación

Las tecnologías utilizadas para el desarrollo de la solución prototipo, las cuales están a la vanguarda del desarrollo de software.

Lenguajes de programación utilizados

Tabla 1. Términos de programación citados en la elaboración de la solución IoT

Nombre	Uso	Descripción
JS Javascript (ES6)	Desarrollo de aplicación front-end y back-end.	Es un lenguaje utilizado en programación, considerado un soporte utilizado en programación orientada a objetos [24].

Nombre	Uso	Descripción
TypeScript (4.6.4)	Desarrollo de aplicación front-end y back-end.	Este es utilizado en un lenguaje de programación que agrega sintaxis a javascript para usarlo como un lenguaje fuertemente tipado [25]
C++ (1.22.0-97-gc752ad5-5.2.0)	Desarrollo de programa controlador del dispositivo IoT	Es un lenguaje de programación elaborado para extender las funcionalidades del lenguaje de programación C, siendo un lenguaje de programación estructurada y orientada a objetos, por ello se dice que es un lenguaje multiparadigma [26].
CSS (css3)	Desarrollo de estilos personalizados en la aplicación Front-end	Es un lenguaje basado en el diseño gráfico para personalizar la presentación de un documento, identificado como un escrito marcado HTML [27].
HTML (html5)	Desarrollo de documentos estructurados para la aplicación Front-end	Es un lenguaje de marcado para la formación de documentos estructurados que son visibles a través de un navegador web [28].

Marcos de trabajo y Librerías

Tabla 2. Marcos de trabajo y librerías usadas en el desarrollo de la solución IoT.

Nombre	Uso	Descripción
React JS (17.0.2)	Desarrollo de interfaces gráficas dentro de la aplicación Front-end	Es una base de datos para crear interfaces de origen gráfico de usuario para aplicaciones web [29].

Nombre	Uso	Descripción
Next JS (12.0.0)	Desarrollo de aplicación front-end.	Es un framework o marco de trabajo para React JS diseñado para crear aplicaciones web con alta escalabilidad, disponibilidad y rendimiento a nivel general [30].
Nest JS (8.0.0)	Desarrollo de aplicaciones back-end.	Es un framework para construir aplicaciones eficientes y escalables de lado del servidor basadas en Node js [31].
BLEDevice.h BLEUtils.h BLEServer.h (1.0.1)	Desarrollo de programación del dispositivo IoT	Librerías open source que contiene funciones para construir soluciones de BLE de manera sencilla sobre placas de desarrollo como arduino o esp-32 [32].
Jest (28.1.0)	Pruebas Unitarias	Es un framework para realizar pruebas unitarias de código en el lenguaje Javascript. Este marco de trabajo está enfocado en la simplicidad para realizar las pruebas [33].
D3.js (7.4.4)	Representación gráfica de datos.	D3.js (Data Driven Documents) es una base de datos de Javascript utilizada para la manipulación de documentos. Esta librería brinda la capacidad de usar las características de los navegadores modernos para generar componentes con un enfoque basado en datos para la manipulación del DOM [34].

Nombre	Uso	Descripción
 Redux (8.0.1)	Manejar estados dentro de la aplicación Front-end	React-Redux es una librería oficial de React para almacenar, actualizar y controlar el estado general de una aplicación React mediante el uso de un almacén de datos desde el cual pueden acceder todos los componentes de React [35].

Entornos de desarrollo, herramientas y base de datos

Tabla 3. Entornos de desarrollo usados y gestor de base de datos.

Nombre	Uso	Descripción
 Visual Studio Code (1.76.1)	IDE de desarrollo	VS Code sirve para editar códigos fuente ligero pero potente gracias a los plugins y es multiplataforma. Integrado con soporte para lenguajes como C, C++, Java, Python, PHP, Go, .Net, etc [36].
 Arduino IDE (1.8.20)	IDE de desarrollo para microcontroladores y placas de desarrollo.	Este contiene un editor de textos para codificar, un área de mensaje, una terminal integrada y una barra de herramientas para lograr una comunicación entre el hardware para cargar programas y comunicarse con él [37].
 MongoDB (4.4.5)	Base de datos.	Es una base de datos NoSQL direccionada a documentos y de código abierto [38].

Nombre	Uso	Descripción
 Postman (9.31.27)	Cliente HTTP para realizar pruebas con la aplicación Back-end	Postman es una plataforma para usar APIs y de esa manera conseguir probar el correcto funcionamiento de la aplicación Back-end [39].

1.3.7 Scrum

Este se caracteriza por ser básicamente un marco de trabajo rápido y ligero el mismo que logra proporcionar pasos destinados a controlar y manejar el desarrollo del producto software [40]. Partiendo una vinculación entre el modelo Incremental e Iterativo porque en cada iteración de desarrollo, el producto final debe tener un aumento incremental totalmente funcional y que debe satisfacer las necesidades del cliente [40].

Para dichas iteraciones scrum usa intervalos de tiempo denominados Sprints, los cuales son la pieza más pequeña de scrum [40], que pueden durar de 1 a 3 semanas. Las tareas que serán desarrolladas dentro de un sprint son tomadas del Sprint Backlog, el cual es la documentación de todos los requerimientos del proyecto que serán satisfechos en está iteración. El Sprint Backlog es alimentado por el Product Backlog el cual contiene toda la documentación de los requerimientos que son proporcionados por el Product Owner, uno de los roles que se juegan en scrum. Los requerimientos documentados se llaman Historias de Usuario.

Roles

Para trabajar con el marco scrum se necesita específicamente que cada integrante del equipo de desarrollo asuma un rol para completar el proyecto con éxito. Los roles son:

- **Product Owner:** Es la persona que está en constante contacto con el cliente con el fin de levantar toda la documentación necesaria en forma de Historias de Usuario. Él organiza, prioriza y determina cuales historias de usuario son desarrolladas en un sprint a fin de que al terminar el sprint se obtenga un producto con una funcionalidad incrementada [41].
- **Scrum Máster:** Es la persona que lidera y está al servicio del equipo de desarrollo con el fin de facilitar el flujo de trabajo. Está siempre al pendiente que

el marco de trabajo se cumpla al cien por ciento. También, debe estar vigilante ante cualquier problema que se presente a lo largo del sprint para mitigarlo lo antes posible y asegurarse que los demás integrantes sigan con su trabajo sin distracciones [41].

- **Development Team:** Es el equipo que se encarga de diseñar, desarrollar y realizar las pruebas de las soluciones a las historias de usuario con la finalidad de cubrir las necesidades del usuario. El grupo no debe sobrepasar las 9 personas y que por lo general está conformado por profesionales con un gran conjunto de competencias a fin de gestionar correctamente la auto-organización. Al final de cada sprint, los desarrolladores crearán incrementos de funcionalidad probados a partir de consumir los elementos del Product Backlog.

Eventos y Artefactos Scrum

El marco de trabajo no puede funcionar sin un conjunto de eventos y artefactos necesarios que describen el flujo de trabajo y que son producidos a lo largo del desarrollo respectivamente [42]. En la Figura 7 se expone el flujo de trabajo de Scrum.

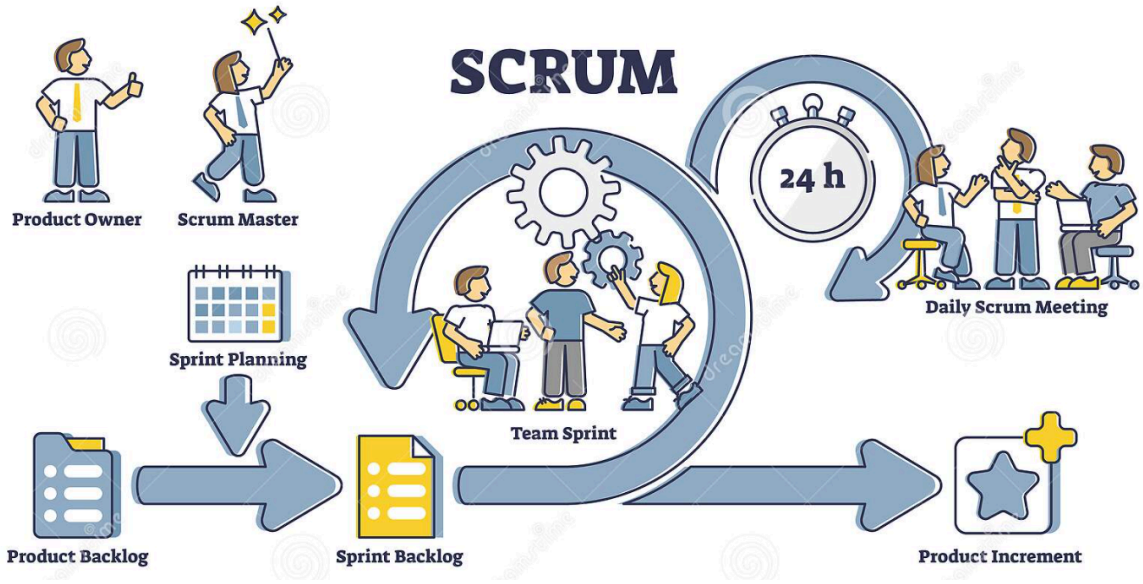


Figura 7. Scrum, principales artefactos y eventos. [40]

Los dispositivos que scrum utiliza se detallan a continuación:

- **Product Backlog:** Este enlista la documentación de los requisitos (que generalmente son historias de usuario), funcionalidades, mejoras, correcciones

y caracteristicas que serán desarrolladas en el producto final. Esta lista tiene la habilidad de cambiar y adaptarse dinámicamente dependiendo de los requerimientos del cliente, esta caracteristica de cambiar y adaptarse es lo que proporciona agilismo a todo el desarrollo.

- **Sprint Backlog:** Es una lista que contiene un subconjunto seleccionado del Product Backlog con el fin de ser desarrolladas, probadas e incrementadas al producto final a lo largo de un periodo de tiempo conocido como sprint. Esta lista es definida en el evento conocido como Sprint Planning.
- **Product Increment:** Es el resultado final de cada sprint, junto con la acumulación de los incrementos generados posteriormente, cada incremento debe estar en plena capacidad de ser usado y cumplir con los requerimientos del usuario.

Los eventos que scrum hace uso son los siguientes:

- **Sprint:** Este permite cubrir las necesidades del cliente es realizado en este evento. Un Sprint no es más que una iteración o ciclo con una duración que oxila entre una a cuatro semanas [40]. El resultado final de un sprint es un incremento funcional al producto final y que dicho incremento cumpla los requerimientos del cliente.
- **Sprint Planning:** El Sprint Planning o planificación del sprint es una reunión en la que participan todo el equipo scrum a fin de generar el Sprint Backlog que será desarrollado a lo largo del Sprint. En esta reunión se plantean los objetivos a ser alcanzados en el Sprint y se seleccionan los elementos del Product Backlog para alcanzar ese objetivo.
- **Daily Scrum:** E suna reunión que no debe sobrepasar los 15 minutos y en la cual cada integrante del equipo de desarrolladores debe responder a 3 interrogantes esenciales: ¿Qué hice ayer?, ¿Qué voy a hacer hoy? y ¿Existe algún impedimento para realizar mi tarea asignada? La respuesta de cada una de estas preguntas, contestadas por todos los miembros del equipo de desarrollo permite tener una visión clara sobre el avance del incremento a lo largo del Sprint. Por lo cual, permite mejorar la comunicación e identificar problemas o bloqueantes que retrasen el desarrollo del incremento [40].
- **Sprint Review:** Es una reunión que toma lugar al final de cada sprint y que tiene como objetivo inspeccionar el incremento obtenido en el sprint y adecuar el

Product Backlog en caso de requerirse [40]. En la reunión participa todo el equipo Scrum, clientes, stakeholders y miembros interesados de otros equipos.

- **Sprint Retrospective:** Es una reunión que toma lugar después del Sprint Review y precede del próximo Sprint Planning. Esta reunión mantiene como objetivo realizar una retroalimentación a todo el equipo de desarrollo para conocer los puntos que resultaron malos y los puntos que resultaron buenos para el desarrollo a lo largo del sprint que acaba de finalizar. Esta reunión genera planes de acción o mejoras que serán abordadas el siguiente sprint.

1.3.8 Pruebas del sistema

Estas son un conjunto de procesos diseñados para asegurar la calidad del sistema. Teniendo como objetivo comprobar que el sistema cumpla con sus objetivos previstos y que no realice acciones no deseadas [43]. Entre las pruebas más comunes se tienen a las pruebas unitarias, de integración y de usabilidad.

Pruebas unitarias

Generalmente son ejecutadas por los desarrolladores de dentro de un ambiente controlado para probar los componentes por separado. Es decir que se debe tomar las piezas más pequeñas del sistema software y probarlas de tal manera que se ignore el resto del sistema para así aislar el componente y probar su correcto funcionamiento por separado [44].

Pruebas de integración

Son pruebas que se utilizan cuando todos los componentes individuales son combinados en busca de conformar el sistema completo [45]. Por lo general son aplicadas después de ejecutar las pruebas unitarias que verifican la funcionalidad de los componentes del sistema antes de ser ensamblados juntos. Para generar los casos de pruebas se pueden escoger entre seis estrategias las cuales son en inglés: bottom-up, top-down, modified top-down, sandwich, modified sandwich y big-bang [45].

Pruebas de usabilidad

Son una técnica esencial para evaluar toda clase de productos software y sistemas. Autores definen a las pruebas de usabilidad como técnicas en las que los usuarios

interactúan de manera organizada con un producto o sistema en circunstancias controladas con el objetivo de realizar una tarea específica en un escenario realista y una vez finalizada la interacción se recogen datos acerca del comportamiento que expresan los usuarios en el proceso [46]. El System Usability Scale (SUS), también conocido como Escala de Usabilidad del Sistema en español, es una herramienta principalmente elegidas en las pruebas de usabilidad. A continuación, se describe en qué consiste esta herramienta.

El SUS es un artefacto metodológico que permite medir la usabilidad de un sistema, aplicativo u objeto [47]. Esta escala está conformada por 10 preguntas las cuales pueden ser puntuadas del uno al cinco donde, uno significa fuertemente en desacuerdo y cinco fuertemente de acuerdo [47]. Una vez obtenido el valor de cada una de las preguntas se aplica un método de cálculo para la obtener la puntación SUS. La Ecuación 3 muestra la fórmula para obtener el puntaje estándar SUS de una manera más consistente a partir de los puntajes obtenidos en las preguntas.

$$SUS = 2{,}5(20 + SUM(SUS01, SUS03, SUS05, SUS07, SUS9) - SUM(SUS02, SUS04, SUS06, SUS08, SUS10))$$

Ecuación 3. Cálculo del puntaje estándar SUS.

El puntaje SUS obtenido debe estar comprendido entre los valores de 0 y 100. Debido a que el puntaje SUS no es un porcentaje se debe aplicar un proceso de normalización de puntajes para obtener un ranking percentil [48]. La tabla 4 muestra el ranking que se puede obtener dependiendo del puntaje SUS.

Tabla 4. Interpretación de las puntuaciones de SUS mediante una escala de calificación.

Rango puntaje SUS	**Calificación**
84.1–100	A+
80.8–84.0	A
78.9–80.7	A-
77.2–78.8	B+
74.1–77.1	B
72.6–74.0	B-

Rango puntaje SUS	Calificación
71.1-72.5	C+
65.0–71.0	C
62.7–64.9	C-
51.7–62.6	D
0–51.6	F

2 METODOLOGÍA

Para elaborar el presente trabajo bajo la metodología Scrum lo primero que se realizó fue la definición de los roles Scrum que van a desempeñar los integrantes del proyecto.

2.1 Definición de roles Scrum

El mencionado trabajo se mantiene conformado por dos integrantes, por lo que es necesario que un participante asuma dos roles al mismo tiempo de los tres roles definidos. La Tabla 5 indica los roles asumidos por cada miembro.

Tabla 5. Roles realizados por el equipo Scrum para la elaboración del trabajo.

Equipo Scrum	
Rol	**Responsable**
Product Owner	Hernán Ordoñez
Scrum Master	Hernán Ordoñez
Development Team	Cristian Flores

2.2 Levantamiento de Requerimientos

2.2.1 Historias de Usuario

En este caso fue necesaria realizar 5 reuniones con el Ing. Hernán Ordoñez de manera virtual vía ZOOM, dentro de estas reuniones se consolidaron las historias épicas de usuario, como se muestra en la Tabla 6, que al completarse se tendrá listo el prototipo de solución IoT.

Tabla 6. Historias épicas para desarrollar.

CÓDIGO	**TÍTULO**	**PRIORIDAD**
PDS1	Diseño del dispositivo IoT usando la placa ESP-32, red y definición de la arquitectura del prototipo.	Alta
PDS2	Codificación del programa que controla el dispositivo IoT	Media
PDS3	Programación del Portal Web para visualizar la información recolectada.	Media

2.2.2 Product Backlog Inicial

Una vez consolidadas las historias épicas de usuario se procedió a crear el Product Backlog el cual contiene una lista detallada de las historias de usuario por cada historia épica. La Tabla 7 muestra el Product Backlog.

Tabla 7. Product Backlog al iniciar la elaboración del proyecto plantedao.

PRODUCT BACKLOG				
HISTORIA ÉPICA	**Historias de Usuario**			
	ID	**Título**	**Estimación (Fibonacci)**	**Prioridad**
PDS1	PDS1-01	Realizar el diagrama de dispositivo IoT.	8	Alta
	PDS1-02	Soldar los componentes físicos del dispositivo IoT	5	Alta
	PDS1-03	Definir de la arquitectura IoT a ser usada	5	Alta
PDS2	PDS2-01	Diseñar un algoritmo para controlar y enviar la información de los dispositivos IoT al servidor back-end.	5	Alta
	PDS2-02	Programar el Algoritmo y desplegarlo en los dispositivos IoT.	5	Media
	PDS2-03	Programar la emisión del nombre del dispositivo vía Bluetooth.	3	Media
PDS3	PDS3-01	Creación de proyecto base usando Nextjs	8	Media
	PDS3-02	Creación de pantalla de Inicio de sesión y registro de usuario.	5	Media
	PDS3-03	Creación de pantalla para vincular y desvincular dispositivo	5	Media
	PDS3-04	Creación de infogramas para evitar contagios de C-19	3	Baja
	PDS3-05	Creación de pantalla principal y tabla de históricos de contactos	5	Media

Sprint 0

2.2.3 Sprint Planning

Objetivo del Sprint

Definir la arquitectura y algoritmo de comunicación del sistema IoT.

Sprint Backlog

A continuación se exponen las historias a ser desarrolladas en el actual sprint (Tabla 8).

Tabla 8. Historias de usuario seleccionadas del Product Backlog del sprint 0.

ID	Descripción	Estimación	Prioridad
PDS1-03	Definir de la arquitectura IoT a ser usada	5	Alta
PDS2-02	Diseñar un algoritmo para controlar y enviar la información de los dispositivos IoT al servidor back-end.	5	Alta

Tabla 9. Sprint Backlog del Sprint 0.

Sprint Backlog	
Historia de Usuario	**Tareas**
PDS1-03	Escoger el tipo de arquitectura IoT más adecuada para el sistema.
	Graficar a detalle la capa encargada del procesamiento de la información.
	Mostrar los frameworks que serán utilizados para construir las capas de la arquitectura IoT.
PDS2-01	Realizar un diagrama de flujo para gestionar el contacto entre nodos IoT.
	Se deberá incluir dos llamadas HTTP hacia los endpoints encargado de recibir la información del contacto.
	El algoritmo debe tener en cuenta cuando inicia y termina un contacto.
	El flujo debe mostrar en qué momento emitir un sonido de alerta para los usuarios.

2.2.4 Desarrollo del Sprint

Historia PDS1-03: Definir de la arquitectura IoT a ser usada

Para completar esta historia se debió escoger entre las diferentes arquitecturas IoT existentes. Después de evaluar varias opciones, se optó por diseñar la solución IoT sobre una arquitectura de 4. La razón principal detrás de esta decisión fue la necesidad de procesar la información en un servidor especializado, debido a la complejidad y cantidad de datos que se maneja. Además, las condiciones actuales de la FIS no permiten que los dispositivos de red realicen procesamiento o almacenamiento de datos, limitando su capacidad a la mera transmisión de información. Debido a estas consideraciones la arquitectura de 4 capas cumple a la perfección los requerimientos del sistema. Desde arriba hacia abajo se tiene las siguientes implementaciones.

Para la capa de aplicación que sería la capa superior, se diagramó la implementación de un aplicativo web realizado en Next js, este aplicativo servirá de portal web para que los usuarios mediante sus credenciales puedan acceder a su información de contactos con otros usuarios.
Para la construcción del servidor principal que almacenará la información en la base de datos de NoSQL se utilizó el framework Nest js. Dentro de este servidor se construyó módulos de autenticación, de lógica de negocio, de procesamiento de datos y un módulo para la gestión de los contactos. Todos estos módulos interactúan directamente con la base de datos, las contraseñas de usuario se guardan encriptadas y la información de los contactos se almacenan generando históricos. Este servidor actuará en la capa de soporte de la arquitectura.

Para la capa de red se tiene a toda la infraestructura existente de red inalámbrica de la FIS, de la cual se hará uso para que los dispositivos IoT tengan salida al internet y puedan comunicarse con la capa de soporte sin ningún problema.

Por último, se tiene a la capa de percepción la cual la conformarán todos los dispositivos IoT diseñados. En la Figura 8, se puede observar la diagramación que contiene IoT.

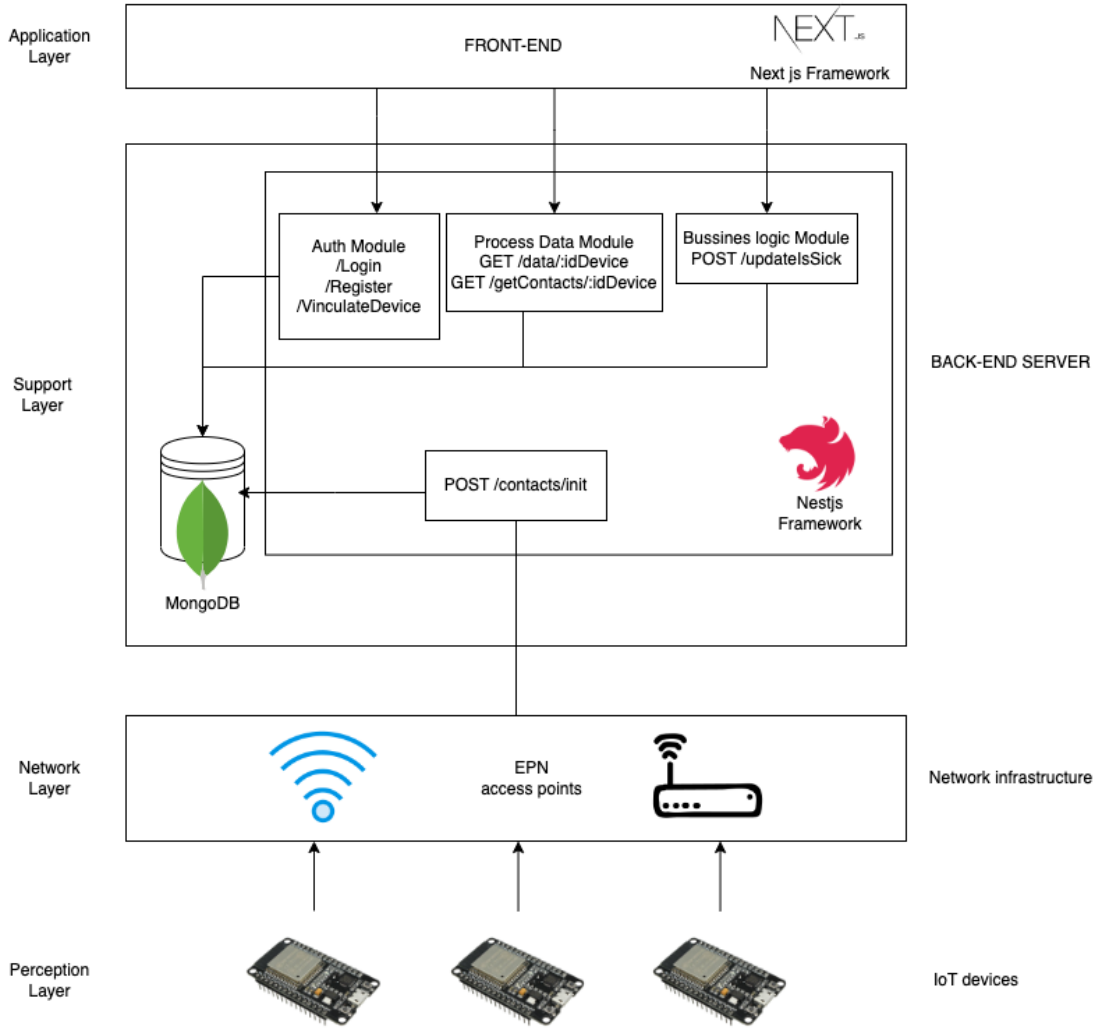


Figura 8. Arquitectura IoT de 4 capas usada en el presente prototipo IoT.

Historia PDS2-01: Diseñar un algoritmo para controlar y enviar la información de los dispositivos IoT al servidor back-end.

Para realizar el diseño de un algoritmo capaz de controlar y enviar la información de los contactos a las capas superiores de la arquitectura se debió tener en cuenta los contactos pueden ser con más de un nodo al mismo tiempo, supongamos que un grupo de 5 usuarios deciden reunirse por cualquier motivo, el dispositivo debe ser capaz de registrar a cada uno de los 5 usuarios, para ello, el algoritmo debe hacer uso de un bucle principal en el cual el primer paso será obtener los Ids de los dispositivos IoT cercanos, estos identificadores serán únicos para cada uno de los dispositivos y tendrán la forma SDM-nnn donde nnn será el número de dispositivo, por ejemplo SDM-001. Luego para cada uno de estos Ids se debe calcular la distancia en base a su RSSI, si la distancia es menor a 1,5 metros se ejecutará un condicional preguntando si el Id del dispositivo ha

sido guardado previamente en memoria, en el caso de que no esté registrado en memoria, se emitirá el sonido de alerta para posteriormente ejecutar una llamada HTTP al endpoint */contacts/init* con un parámetro de cuerpo isInit en true. Una vez ejecutada la llamada se procede a almacenar el Id en memoria y se repite el bucle.

Para el caso en el que la distancia sea mayor a 1,5 metros, el siguiente paso es evaluar el mismo condicional preguntando si el Id está en memoria, para el caso en que el Id si este en memoria, se ejecuta una llamada HTTP al endpoint /contacts/init con un parámetro de cuerpo isInit en false. Posteriormente se procede a eliminar el Id de la memoria del dispositivo y se repite el bucle.

De esta manera solo quedan dos caminos que el algoritmo puede tomar. Para el caso en que el Id del dispositivo se encuentre a menos de 1,5 metros y el Id ya se encuentre en memoria, simplemente se emitirá el sonido de alarma y se repite el bucle. Y para el caso en que la distancia sea mayor a 1,5 y el Id del dispositivo no se encuentre en memoria simplemente se repite el bucle.

Tomando estos pasos en cuenta se logró obtener un diagrama de flujo que representa al algoritmo. En la Figura 9 se presenta el diagrama de flujo

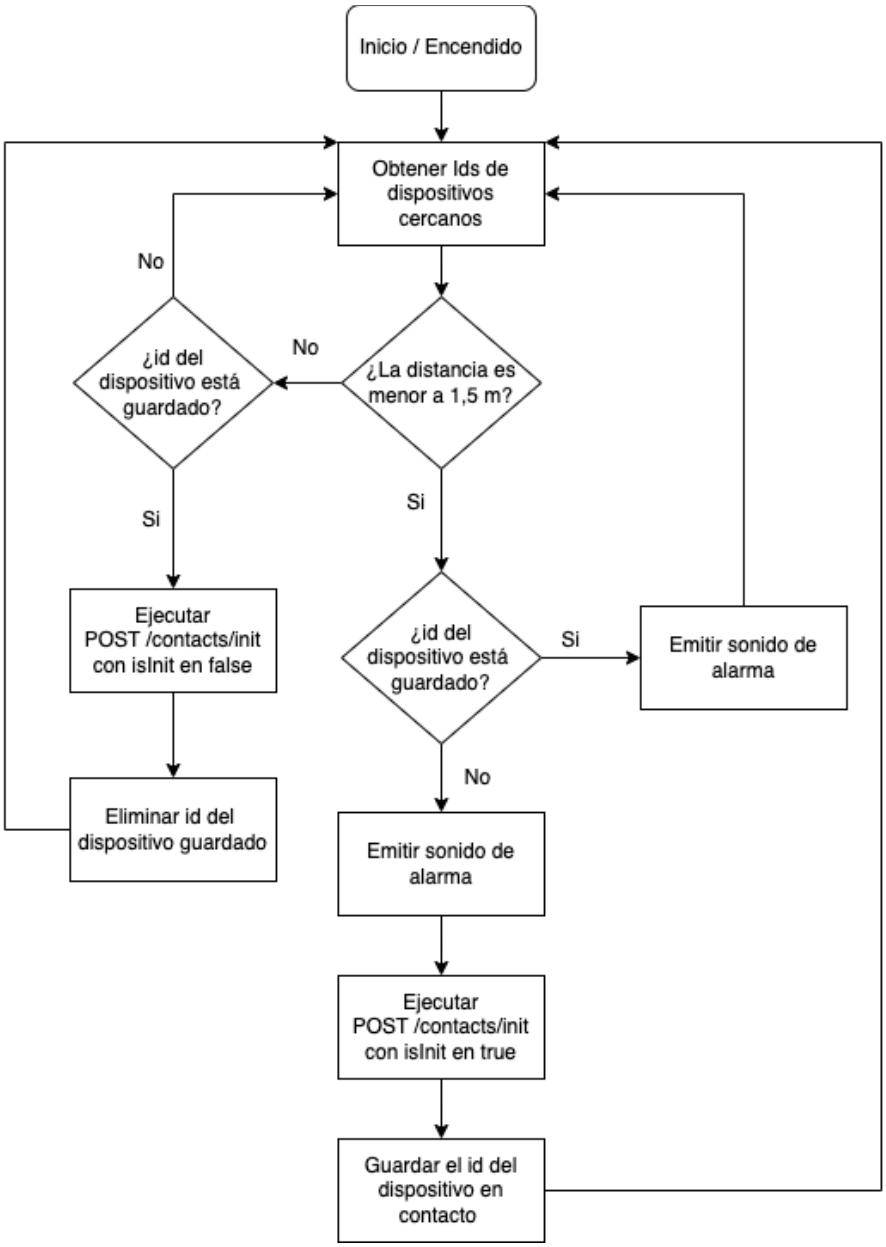


Figura 9. Diagrama de flujo del algoritmo de control de contactos.

2.2.5 Sprint Review

Revisión de Criterios de Aceptación.

Una vez finalizado el sprint, se logró completar con éxito los criterios de aceptación, la Tabla 10, muestra los criterios cumplidos.

Tabla 10. Criterios de admisión del sprint 0.

Historia de Usuario	Criterio	Estado
PDS1-03	Se podrá visualizar claramente el framework usado para construir las capas del sistema.	Cumplido
	Podrá visualizar un diagrama que muestra los endpoints encargados de la entrada y salida de los datos.	Cumplido
PDS2-01	Se podrá visualizar un diagrama de flujo del algoritmo de gestión de contactos	Cumplido

Adaptación del Product Backlog

Al finalizar sprint 0, el product backlog queda intacto, únicamente se cambia de estado las historias finalizadas. En la Tabla 11, se expone el product backlog actualizado.

Tabla 11. Product Backlog al finalizar sprint 0.

PRODUCT BACKLOG				
HISTORIA ÉPICA	**Historias de Usuario**			
	ID	**Título**	**Estimación (Fibonacci)**	**Estado**
PDS1	PDS1-01	Realizar el diagrama de dispositivo IoT.	8	Por Implementar
	PDS1-02	Soldar los componentes físicos del dispositivo IoT	5	Por Implementar
	PDS1-03	Definir de la arquitectura IoT a ser usada	5	Terminado
PDS2	PDS2-01	Diseñar un algoritmo para controlar y enviar la información de los dispositivos IoT al servidor back-end.	5	Terminado
	PDS2-02	Programar el Algoritmo y desplegarlo en los dispositivos IoT.	5	Por Implementar
	PDS2-03	Programar la emisión del nombre del dispositivo vía Bluetooth.	3	Por Implementar
PDS3	PDS3-01	Creación de proyecto base usando Nextjs	8	Por Implementar

	PDS3-02	Creación de pantalla de Inicio de Sesión y registro de usuario.	5	Por Implementar
	PDS3-03	Creación de pantalla para vincular y desvincular dispositivo	5	Por Implementar
	PDS3-04	Creación de infogramas para evitar contagios de C-19	3	Baja Por Implementar
	PDS3-05	Creación de pantalla principal y tabla de históricos de contactos	5	Por Implementar

2.2.6 Sprint Retrospective

La velocidad de desarrollo del sprint fue aceptable debido a que a partir del segundo al cuarto día y del séptimo al décimo día se completaron las tareas en un excelente tiempo, como se puede apreciar el gráfico Burndown generado para este sprint (Figura 10).

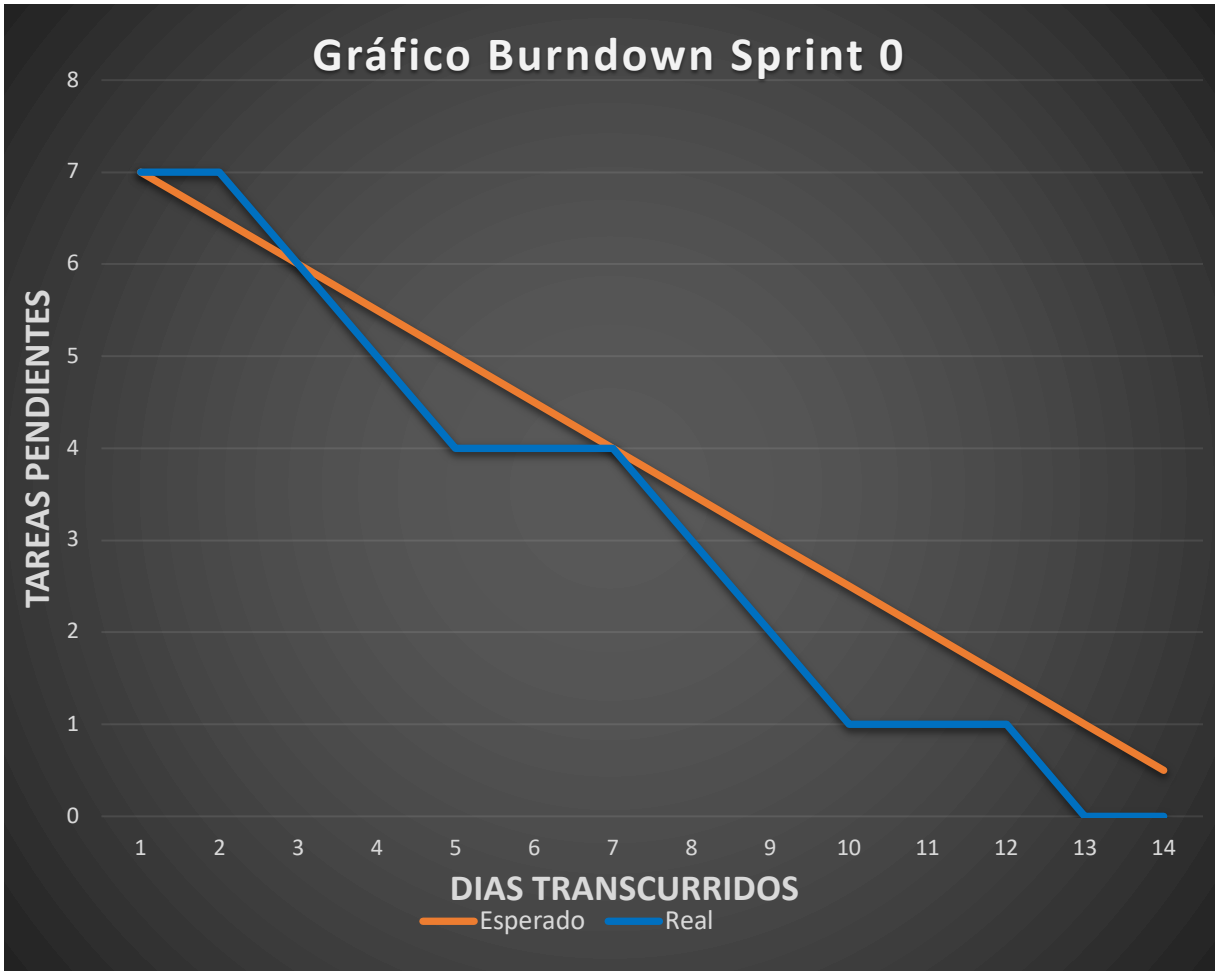


Figura 10. Gráfico Burndown Sprint 0

2.3 Sprint 1

2.3.1 Sprint Planning

Objetivo del Sprint

Construir el dispositivo IoT usando como base la placa NodeMCU ESP-32.

Sprint Backlog

A continuación, se encuentran detalladas las historias de usuario procedentes del Product Backlog que fueron desarrolladas en el Sprint (Tabla 12).

Tabla 12. Historias de usuario seleccionadas del Product Backlog.

ID	Descripción	Estimación	Prioridad
PDS1-01	Realizar el diagrama de dispositivo IoT.	8	Alta
PDS1-02	Soldar los componentes físicos del dispositivo IoT	5	Alta

Tabla 13. Sprint Backlog del Sprint 1.

Sprint Backlog	
Historia de Usuario	**Tareas**
PDS1-01	Investigar una forma de alimentar al dispositivo usando una batería de litio recargable.
	Realizar un listado de los componentes electrónicos que tiene el dispositivo IoT
	Realizar un análisis de coste por componente y coste total del dispositivo IoT
	Realizar un diagrama de conexión entre los componentes electrónicos
PDS1-02	Unir los componentes en un protoboard y verificar que la conexión entre los componentes sea correcta y de acuerdo con las especificaciones de los componentes.
	Soltar los componentes IoT y compactarlos al tamaño de un carné o gafete

2.3.2 Desarrollo del Sprint

A continuación, se detalla la elaboración de cada historia de usuario dentro del Sprint Backlog.

Historia PDS1-01: Realizar el diagrama de dispositivo IoT.

En primer lugar, se consideró la forma de alimentar el dispositivo IoT, teniendo en cuenta su portabilidad como una característica clave. Por esta razón, el diseño se inició por la etapa de alimentación. Es importante recordar que la placa NodeMCU ESP-32, tal como se mencionó en el marco teórico, opera con un voltaje de alimentación de 5V [14]. De esta manera, se garantiza que el dispositivo pueda funcionar adecuadamente, al mismo tiempo que se cumple con los requisitos de portabilidad y eficiencia energética necesarios para su aplicación. También hay que tener en cuenta que el dispositivo debe tener la capacidad de mantenerse operativo durante toda una jornada laboral de 8 horas. Por ello se optó por conseguir la Batería de iones de litio recargable de marca UltraFire de 4200 mAh, que proporciona un voltaje de 4.2V. El siguiente paso fue conseguir un módulo de carga para cargar la batería, el módulo usado fue el TP4056 con entrada USB tipo C, el cual es un cargador linear que proporciona voltaje y corriente constantes para cargar una batería de iones de litio [49], cuyas características se acoplan perfecto a la batería seleccionada.

Una vez con la alimentación solucionada, se presentó un nuevo reto. Debido a que las especificaciones de la placa NodeMCU ESP-32 señalan que debe tener una alimentación de 5V y que nuestra batería únicamente proporciona 4.2V fue necesario buscar un módulo que eleve el voltaje a los 5V requeridos para no comprometer maliciosamente la placa de desarrollo. Se optó por usar el módulo MT3608 el cual es un Step Up que permite elevar el voltaje hasta 28V y que es usado para aplicaciones de bajo consumo de energía [50].

Por último, se tiene un Buzzer el cual servirá para alertar sonoramente de la proximidad entre los dos dispositivos IoT. Basándose en los componentes previamente adquiridos, se elaboró la Tabla 14.

Tabla 14. Componentes del dispositivo IoT.

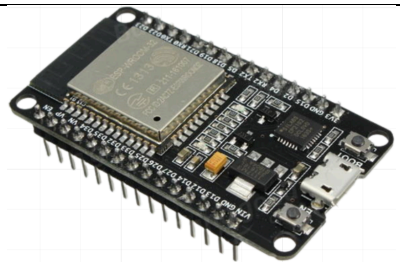
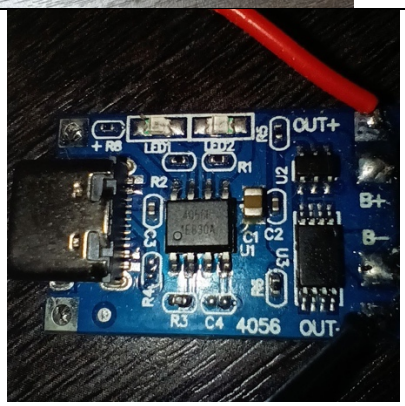
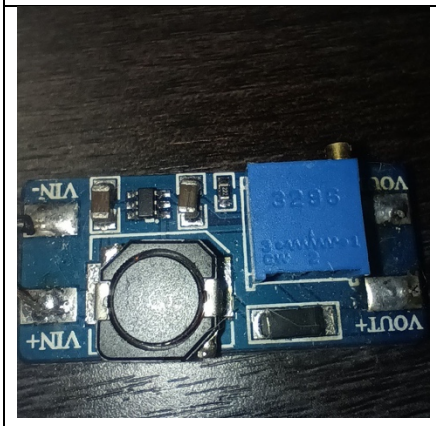
Imagen	Nombre	Utilidad
	NodeMCU ESP32	Manejar toda la lógica de contactos.
	Bateria BRC 18650 y su ranura de carga	Alimentar todo el circuito del dispositivo.
	Módulo de carga TP4056	Cargar la batería de iones de litio.
	Módulo Step Up MT3608	Elevar el voltaje recibido de la batería de 4.2V a 5V

Imagen	Nombre	Utilidad
	Buzzer	Emitir una alerta sonora al no cumplir la distancia estipulada por el distanciamiento social.

Una vez analizados los componentes y justificado su utilidad, se realizó la Tabla 15 detallando los costos de los componentes.

Tabla 15. Costos del dispositivo IoT.

Nombre del componente	Cantidad	Precio Unitario + IVA	Total
NodeMCU ESP-32	2	$8.00	$16.00
BRC 18650	1	$6.00	$6.00
TP4056	1	$2.00	$2.00
MT3608	1	$3.00	$3.00
		Total:	$27.00

La Figura 11, expone el diagrama que conecta los componentes del dispositivo IoT.

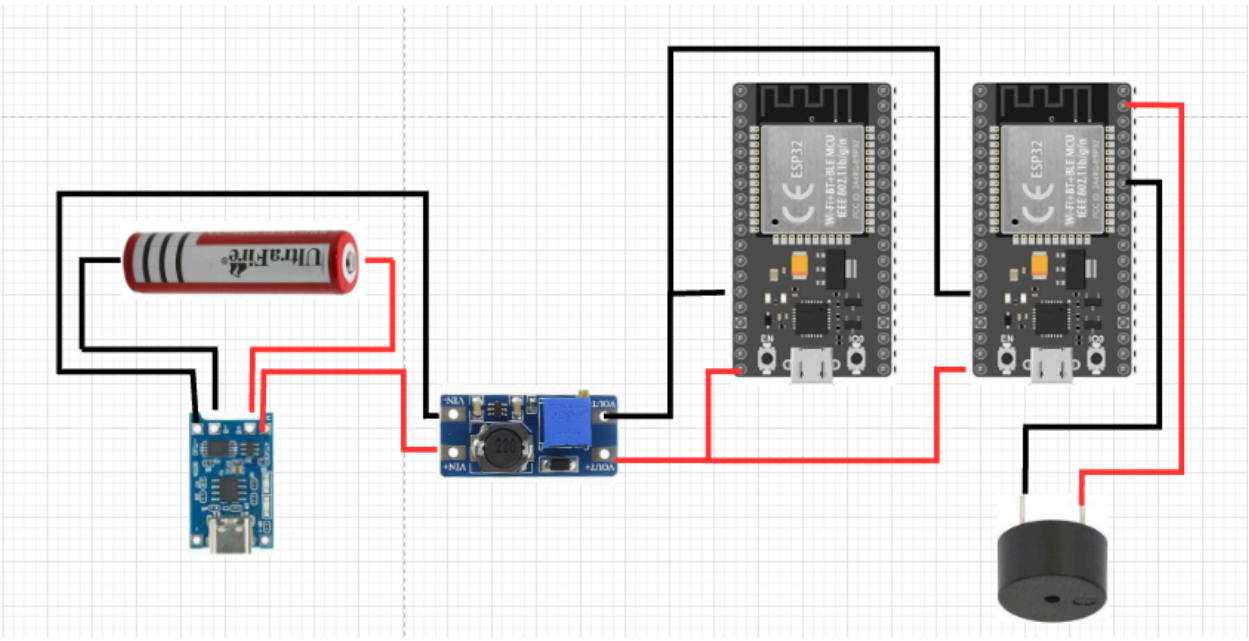


Figura 11. Diagrama de la conexión de los componentes del dispositivo IoT

Historia PDS1-02: Soldar los componentes físicos del dispositivo IoT

Debido a que el dispositivo IoT propuesto para este proyecto debe encontrarse en una posición estratégica para lograr la medición de distancia entre personas, se optó por colocarlo lo más cerca que se pueda del rostro de los usuarios. El gafete es un instrumento de suma utilidad para los corporativos ya que, además de proporcionar una capa extra de seguridad al momento de autenticar al usuario, es de uso obligatorio al entrar a edificios, oficinas u espacios laborales y generalmente los usuarios mantienen su gafete colgando del cuello. Gracias a estas características se pensó en diseñar el dispositivo del tamaño de un gafete para ser acoplado de una manera más orgánica para el usuario.

Para realizar la soldadura de los componentes se necesitó de cinta aislante, estaño, cautín y cable. Una vez soldado los componentes, el dispositivo IoT queda compacto como muestra la Figura 12

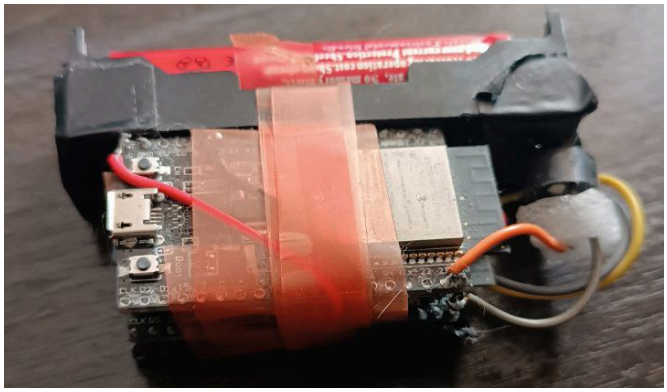


Figura 12. Dispositivo IoT compactado.

2.3.3 Sprint Review

Este cumplió a la perfección el objetivo planteado, se obtuvo un dispositivo IoT usando la placa de desarrollo NodeMCU ESP-32. No se encontró ninguna dificultad ni impedimento para llegar al objetivo.

Revisión de Criterios de Aceptación.

Una vez finalizado el sprint se puede apreciar los criterios de aceptación cumplidos según las historias que fueron desarrolladas en este tiempo, ver Tabla 16.

Tabla 16. Criterios de aceptación del Sprint 1.

Historia de Usuario	Criterios	Estado
PDS1-01	Podrá revisar una tabla que detalla cada uno de los componentes que han sido usados en la construcción del dispositivo IoT	Cumplido
	Podrá visualizar un diagrama de conexión de componentes del dispositivo.	Cumplido
	Podrá visualizar una tabla de costes de cada componente.	Cumplido
PDS1-02	Podrá visualizar que el dispositivo tiene sus componentes correctamente soldados.	Cumplido
	Podrá visualizar que el tamaño del dispositivo es el de un carné o gafete, en altura y anchura.	Cumplido

Adaptación del Product Backlog

Luego de la finalización del sprint 1, no se encontró necesidad de modificar el Product Backlog por lo que simplemente se retiran las historias finalizadas quedando como dicho avance como se puede ver a continuación (Tabla 17).

Tabla 17. Product Backlog luego de finalizar el primer sprint.

PRODUCT BACKLOG				
HISTORIA ÉPICA	**Historias de Usuario**			
	ID	**Título**	**Estimación (Fibonacci)**	**Estado**
PDS1	PDS1-01	Realizar el diagrama de dispositivo IoT.	8	Terminado
	PDS1-02	Soldar los componentes físicos del dispositivo IoT	5	Terminado
	PDS1-03	Definir de la arquitectura IoT a ser usada	5	Terminado
PDS2	PDS2-01	Diseñar un algoritmo para controlar y enviar la información	5	Terminado

		de los dispositivos IoT al servidor back-end.		
	PDS2-02	Programar el Algoritmo y desplegarlo en los dispositivos IoT.	5	Por Implementar
	PDS2-03	Programar la emisión del nombre del dispositivo vía Bluetooth.	3	Por Implementar
PDS3	PDS3-01	Creación de proyecto base usando Nextjs	8	Por Implementar
	PDS3-02	Creación de pantalla de Inicio de Sesión y registro de usuario.	5	Por Implementar
	PDS3-03	Creación de pantalla para vincular y desvincular dispositivo	5	Por Implementar
	PDS3-04	Creación de infogramas para evitar contagios de C-19	3	Baja Por Implementar
	PDS3-05	Creación de pantalla principal y tabla de históricos de contactos	5	No implementado

2.3.4 Sprint Retrospective

Dentro de la Figura 13, se puede apreciar la gráfica de Burndown la misma que expone el progreso del equipo mientras dura el sprint. Pese a que algunas tareas tomaron más tiempo de lo esperado, en la última semana de trabajo se logró completar las tareas asignadas.

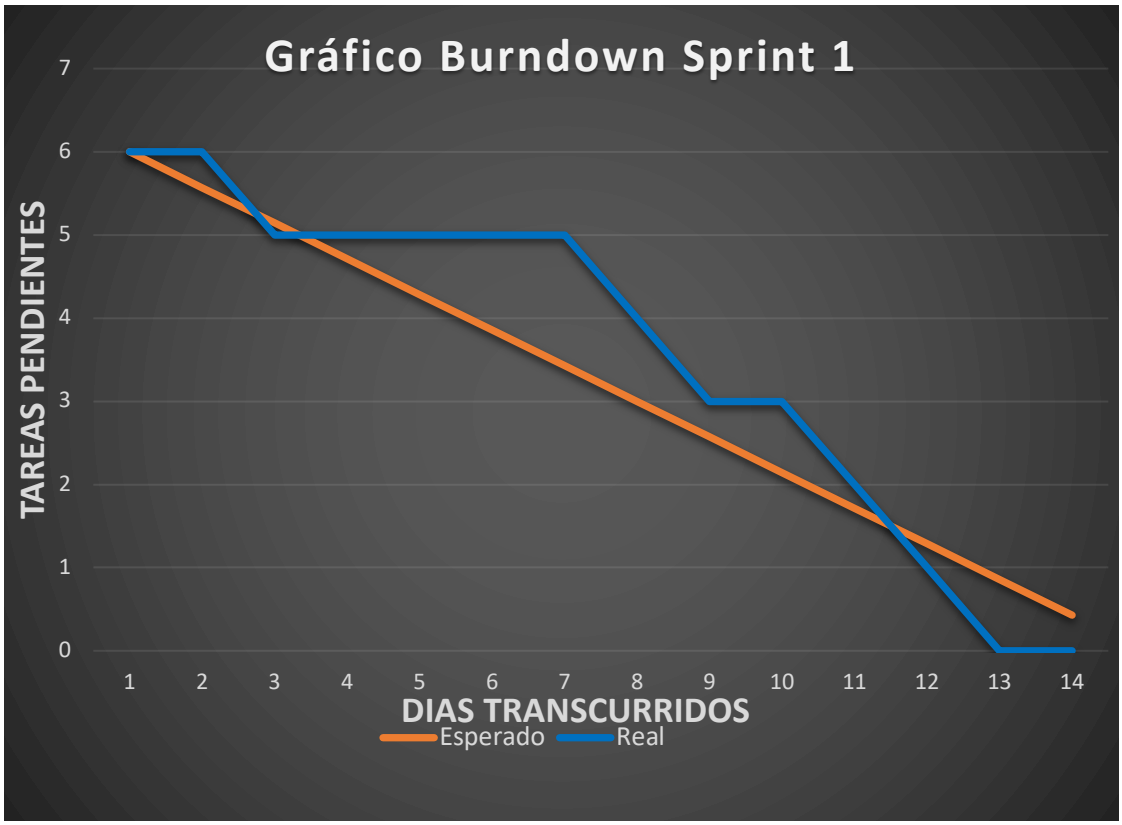


Figura 13. Gráfica Burndown Sprint 1

2.4 Sprint 2

2.4.1 Sprint Planning

Objetivo del Sprint

Implementar el algoritmo de gestión de contactos en la placa NodeMCU-ESP32

Sprint Backlog

En la Tabla 18, se presenta un listado de las historias de usuario adquiridas del Product Backlog que se utilizaran en el Sprint.

Tabla 18. Historias de usuario seleccionadas del Product Backlog del sprint 2.

ID	Descripción	Estimación	Prioridad
PDS2-02	Programar el Algoritmo y desplegarlo en los dispositivos IoT.	5	Alta
PDS2-03	Programar la emisión del nombre del dispositivo vía Bluetooth.	3	Alta

Tabla 19. Sprint Backlog del Sprint 2.

Sprint Backlog	
Historia de Usuario	**Tareas**
PDS2-02	Instalar librerías necesarias para controlar el BLE de los dispositivos IoT
	Instalar librerías para controlar el Wifi del dispositivo.
	Instalar librerías necesarias para realizar llamadas HTTP desde el dispositivo IoT.
	Crear un espacio de memoria para almacenar 200 números únicos de los dispositivos IoT.
	Programar el algoritmo desarrollado en la historia PDS2-02 en Arduino IDE
	Desplegar el programa en los dispositivos IoT.
PDS2-03	Usar las librerías BLEDevice.h, BLEUtils.h y BLEServer.h para realizar la emisión de señal BLE usando el nombre SDM-### donde los numerales representan el número de dispositivo.
	Generar una tabla de dos identificadores únicos uuid v4 para cada uno de los 200 dispositivos IoT.
	Emitir en bucle el nombre del dispositivo IoT.

2.4.2 Desarrollo del Sprint

Historia PDS2-02: Programar el Algoritmo y desplegarlo en los dispositivos IoT.

El primer paso para completar la historia de usuario es brindar la capacidad de conectarse a internet al dispositivo ESP-23, para esto se usó la librería Wifi.h, una vez configurado la conexión a internet, se debió definir todas las constantes globales a ser usadas en todo el programa, estas constantes son las presentadas en la Figura 14.

```
1  const char* ssid = "wifi net name";
2  const char* password = "password";
3  const char* thisDeviceId = "SDM-001";
4  const char* serverName = "http://192.168.101.2:3022/contacts/init";
5  const char* jwtToken = "Bearer JWT";
```

Figura 14. Diagrama de flujo del algoritmo de control de contactos

Las constantes ssid y password son usadas para la conexión a internet, la constante thisDeviceId usada para almacenar el nombre del dispositivo, serverName guarda la ruta para las peticiones HTTP, jwtToken es el json web token usado para la autenticación con el servidor.

Para realizar llamadas HTTP se debió instalar la librería “HTTPClient.h” y para enviar la información en el cuerpo de la petición en formato json se usó la librería “ArduinoJson.h”. Para la creación del espacio de memoria para almacenar los ids de los dispositivos IoT simplemente fue necesario crear un array de números enteros con un espacio de 200.

En cuanto a la programación, se continuó con el diagrama de flujo diseñado en el sprint anterior, implementando todos los flujos planificados. Uno de los aspectos clave que se debe resaltar es el cálculo de la distancia, para lo cual se empleó una función que presenta detallados en la Figura 15.

```
1  double calculateDistance(int rssi) {
2    double distance =  pow(10, - ((rssi - A) / (10 * n)));
3
4    return distance;
5  }
```

Figura 15. Función para el cálculo de distancia entre los dispositivos.

Para el despliegue del código fuente en los dispositivos se utilizó el programa arduino IDE, el primer paso fue compilar el programa y una vez que el programa se compile con éxito se lo puede subir directo a la placa de desarrollo que debe estar previamente conectada al computador por medio de cable USB. En la Figura 16 y 17, se puede observar el despliegue del código desde arduino IDE y la conexión de la placa de desarrollo y el ordenador respectivamente.

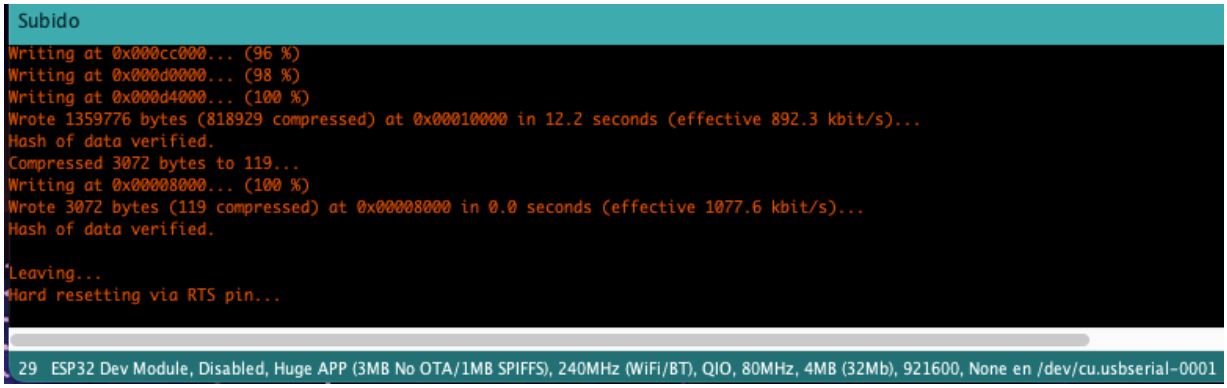


Figura 16. Despliegue exitoso en la placa de desarrollo.

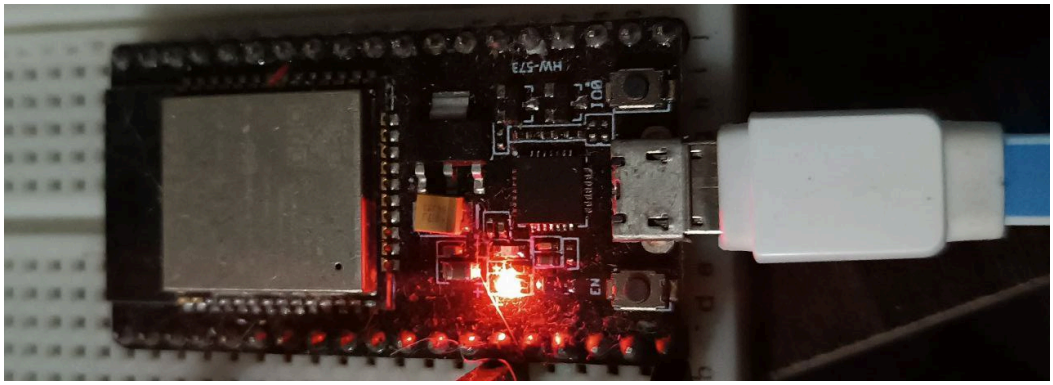


Figura 17. Conexión entre la placa y el ordenador.

Historia PDS2-03: Programar la emisión del nombre del dispositivo vía Bluetooth.

Para la primera tarea de la historia de usuario realizada fue necesario instalar las librerías usando el programa Arduino IDE. La Figura 18 muestra las librerías incluidas en el código

```
1  #include <BLEDevice.h>
2  #include <BLEUtils.h>
3  #include <BLEServer.h>
```

Figura 18. Conexión entre la placa y el ordenador.

Debido a que, para iniciar la difusión de la señal BLE con el nombre del dispositivo usando la librería BLEServer es necesario obtener 2 identificadores únicos uuid versión 4, se utilizó una herramienta llamada “uuid generator” para crear los identificadores únicos. Finalmente se crea un bucle infinito para emitir la señal cada segundo.

2.4.3 Sprint Review

Revisión de Criterios de Aceptación.

Una vez finalizado el sprint, se logró completar con éxito los criterios de aceptación, la Tabla 20 muestra los criterios cumplidos.

Tabla 20. Criterios de aceptación del sprint 2.

Historia de Usuario	Criterios	Estado
PDS2-02	El dispositivo emitirá un sonido de alerta cuando un usuario se acerque a menos de 1,5 m de otro usuario	Cumplido

	que este usando otro dispositivo	
	El dispositivo enviará una petición HTTP de tipo POST al endpoint /contacts/init	Cumplido
	El dispositivo debe dejar de emitir la alerta sonora y deberá enviar una petición tipo POST al endpoint /contacts/init	Cumplido
PDS2-03	Se podrá visualizar que el nombre del dispositivo está siendo emitido para ser leído por todos los dispositivos cercanos.	Cumplido

Adaptación del Product Backlog

Durante el desarrollo del sprint se determinó que se requería la implementación adicional de una épica para el desarrollo del servidor back-end. Por ello, se añadieron 6 nuevas historias que conforman la nueva épica PDS4. La Tabla 21 muestra el nuevo Product backlog.

Tabla 21. Product backlog luego de dar por terminado el sprint 2.

PRODUCT BACKLOG				
HISTORIA ÉPICA	**Historias de Usuario**			
	ID	**Título**	**Estimación (Fibonacci)**	**Estado**
PDS1	PDS1-01	Realizar el diagrama de dispositivo IoT.	8	Terminado
	PDS1-02	Soldar los componentes físicos del dispositivo IoT	5	Terminado
	PDS1-03	Definir de la arquitectura IoT a ser usada	5	Terminado
PDS2	PDS2-01	Diseñar un algoritmo para controlar y enviar la información de los dispositivos IoT al servidor back-end.	5	Terminado
	PDS2-02	Programar el Algoritmo y desplegarlo en los dispositivos IoT.	5	Terminado
	PDS2-03	Programar la emisión del nombre del dispositivo vía Bluetooth.	3	Terminado

PRODUCT BACKLOG				
HISTORIA ÉPICA	**Historias de Usuario**			
	ID	**Título**	**Estimación (Fibonacci)**	**Estado**
PDS3	PDS3-01	Creación de proyecto base usando Nextjs	5	Por implementar
	PDS3-02	Creación de pantalla de Inicio de Sesión y registro de usuario.	8	Por implementar
	PDS3-03	Creación de pantalla para vincular y desvincular dispositivo	5	Por implementar
	PDS3-04	Creación de infogramas para evitar contagios de C-19	3	Por implementar
	PDS3-05	Creación de pantalla principal y tabla de históricos de contactos	8	Por implementar
	PDS3-06	Botón reportarse enfermo/sano	3	Por implementar
PDS4	PDS4-01	Creación del proyecto base usando Nest js	5	Por implementar
	PDS4-02	Crear endpoints para manejar el registro e inicio de sesión de usuario.	5	Por implementar
	PDS4-03	Crear endpoints para guardar los contactos registrados por el dispositivo IoT.	3	Por implementar
	PDS4-04	Crear endpoints para obtener la información del grafo de contactos y la tabla de históricos.	5	Por implementar
	PDS4-05	Crear endpoint para registrar el estado de enfermo y alertar a los demás nodos.	8	Por implementar
	PDS4-06	Crear endpoint para vincular o desvincular un dispositivo a una cuenta de usuario.	3	Por implementar

Sprint Retrospective

En el gráfico Burndown del sprint 2 muestra una velocidad de desarrollo aceptable sin embargo se tuvo mucha presión durante los días 11, 12 y 13 mostrando mayor velocidad en esos días. La Figura 19 muestra el gráfico obtenido para el presente sprint.

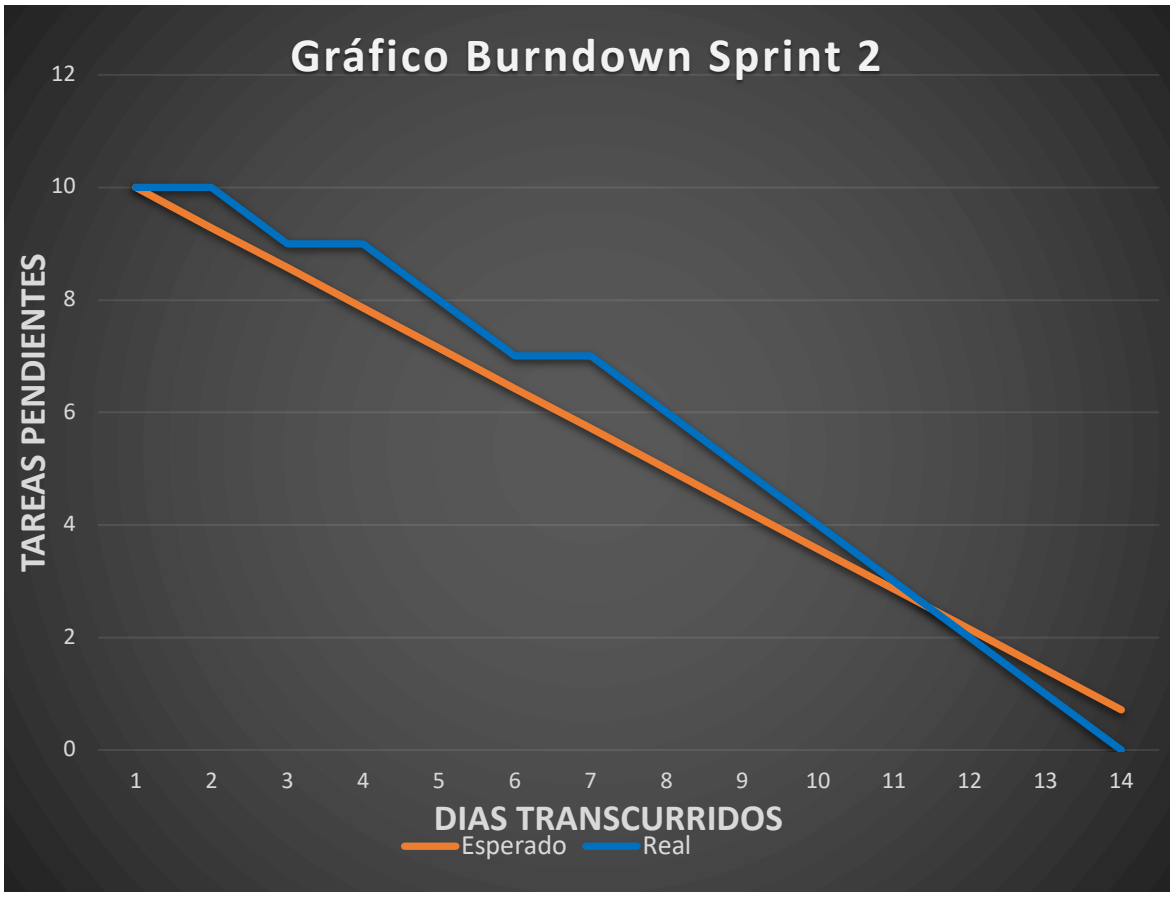


Figura 19. Gráfico Burndown Sprint 2

2.5 Sprint 3

2.5.1 Sprint Planning

Objetivo del Sprint

Crear los proyectos base tanto de la aplicación back-end como de la aplicación front-end.

Sprint Backlog

Dentro de la Tabla 22, se presentan las historias a ser formadas en el actual sprint.

Tabla 22. Historias de usuario seleccionadas del Product Backlog del sprint 3.

ID	Descripción	Estimación	Prioridad
PDS3-01	Creación de proyecto base usando Nextjs	5	Alta
PDS4-01	Creación del proyecto base usando Nest js	5	Alta

ID	Descripción	Estimación	Prioridad
PDS3-04	Creación de infogramas para evitar contagios de C-19	3	Alta

Tabla 23. Sprint Backlog del sprint 3.

Sprint Backlog	
Historia de Usuario	**Tareas**
PDS3-01	Crear un proyecto nuevo usando el comando “create next-app” y configurarlo con Typescript.
	Añadir las librerías: Axios, Lodash, MUI, React-redux, socket.io, jest, d3 y husky.
	Configurar el React-redux store para gestionar el estado de la aplicación web globalmente.
	Configurar las siguientes páginas: home, history, linkDevice, y recomendations.
	Levantar ambiente de pruebas usando la librería jest.
PDS4-01	Crear un proyecto nuevo usando el comando “nest new proyect-name” y configurarlo con Typescript
	Añadir las librerías: jest, luxon, lodash, mongose, bycript, passport, passport-jwt, passport-local, husky, y socket.io
	Levantar ambiente de pruebas usando jest.
PDS3-04	Realizar 3 infogramas sobre la prevención, síntomas y recomendaciones para evitar contagios de C-19 y exponerlos en la ruta “/recomendations”

2.5.2 Desarrollo del Sprint

Historia PDS3-01: Creación de proyecto base usando Nextjs

Para la creación del proyecto base usando Nextjs simplemente se siguió al pie de la letra la documentación oficial provista por Vercel, la empresa creadora del framework. Una vez completada la creación del proyecto y levantada de manera local se puede apreciar la pantalla de bienvenida a Nextjs. Gracias a la facilidad del framework para crear las diferentes rutas que tendrá el aplicativo web simplemente es necesario crear ficheros con el nombre de las rutas dentro del fichero pages que se crea automáticamente con el mando anterior. La Figura 20 muestra el fichero pages con sus respectivas pantallas.

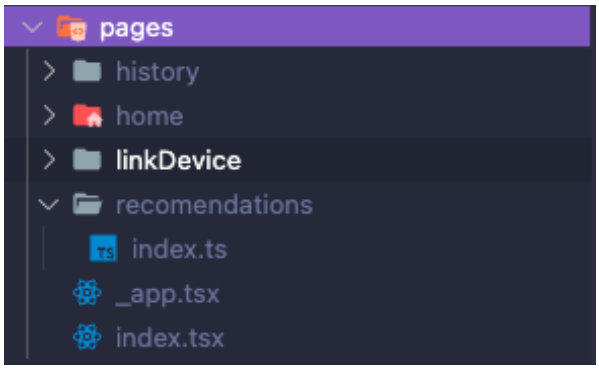


Figura 20. Rutas del portal web.

Para completar la tarea de añadir las librerías necesarias para el desarrollo se utilizó el comando “npm i nombre-librería”, comando el cual añade de manera automática las librerías y dependencias necesarias para el posterior desarrollo. Estas dependencias instaladas se pueden apreciar en el archivo de configuración llamado package.json.

Debido a que la aplicación maneja sesiones de usuario y una gran cantidad de estados se optó por hacer uso de la librería React-redux para facilitar el mano global de los estados de la aplicación y mejorar el rendimiento de la aplicación.

Para levantar el ambiente de pruebas fue necesario configurar los archivos jest.config.js y jest.setup.tsx y para correr las pruebas unitarias se creó el comando “npm run val” para validar el código escrito. En la Figura 21, se expone el valor porcentual del código cubierto por las pruebas unitarias.

File	% Stmts	% Branch	% Funcs	% Lines
All files	100	100	100	100

Figura 21. Porcentaje de lineas de código cubiertas por pruebas unitarias del proyecto front-end.

Historia PDS4-01: Creación del proyecto base usando Nest js

Para la implementación del proyecto base en Nestjs, de igual forma que se lo hizo con el proyecto front-end se buscó en la documentación oficial del Framework para encontrar los comandos e indicaciones necesarias para crear el proyecto. Una vez completada la creación del framework se pudo realizar una petición HTTP de tipo GET hacia la ruta http://localhost:3022 y se obtuvo la respuesta por defecto del framework mostrando así que la creación del proyecto base se ha completado con éxito. En la Figura 22, se puede

apreciar la petición realizada desde el programa Postman y la respuesta por parte del aplicativo web.

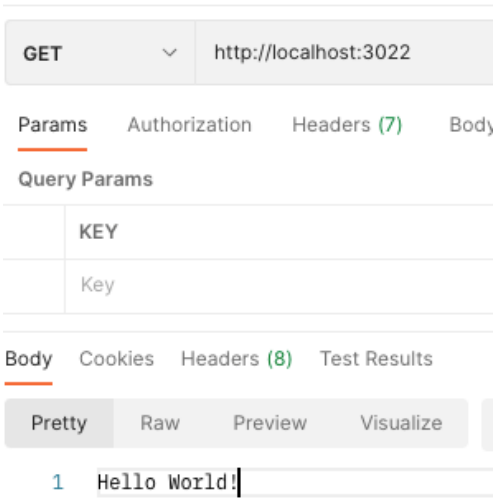
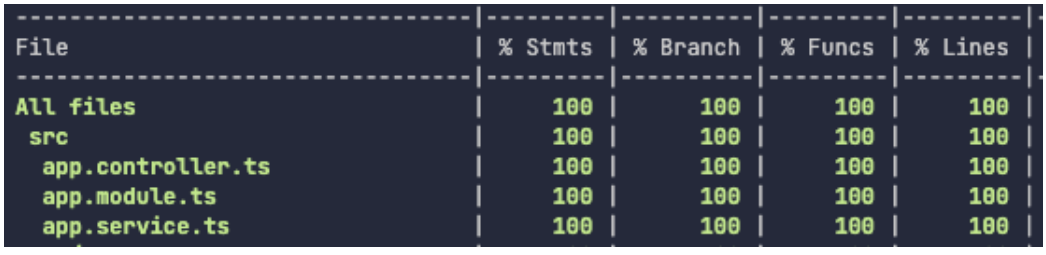


Figura 22. Petición HTTP de tipo GET hacia el proyecto base back-end.

Para añadir las librerías y dependencias solicitadas para el posterior desarrollo dentro del proyecto base se utilizó el mismo comando que para la historia PDS3-01, el comando “npm i nombre-librería”.

Y finalmente para levantar el ambiente de pruebas se debió configurar el archivo jest.config.js y crear de igual forma el comando “npm run val” para correr las pruebas unitarias del proyecto y validar el porcentaje de código cubierto por las mismas. La Figura 23 muestra el porcentaje cubierto para el proyecto base back-end.



File	% Stmts	% Branch	% Funcs	% Lines
All files	100	100	100	100
src	100	100	100	100
app.controller.ts	100	100	100	100
app.module.ts	100	100	100	100
app.service.ts	100	100	100	100

Figura 23. Porcentaje de líneas de código cubiertas por pruebas unitarias.

Historia PDS3-04: Creación de infogramas para evitar contagios de C-19

Para completar esta historia de usuario fue necesario obtener información sobre las mejores prácticas para evitar contagios y crear vectores SVG para realizar ilustraciones que puedan ayudar al entendimiento de las recomendaciones. Finalmente se importaron estas ilustraciones en formato SVG al proyecto y exponerlas en la ruta "/recomendations".

2.5.3 Sprint Review

Revisión de Criterios de Aceptación.

Una vez finalizado el sprint, se logró completar con éxito los criterios de aceptación, la Tabla 24 muestra los criterios cumplidos.

Tabla 24. Criterios de aceptación del sprint 3.

Historia de Usuario	Criterios	Estado
PDS3-01	Se podrá visualizar la pantalla inicial del Proyecto en Nextjs.	Cumplido
PDS4-01	Se podrá visualizar una petición HTTP de tipo GET exitosa.	Cumplido
PDS3-04	Se podrá visualizar infogramas sobre medidas para evitar el C-19	Cumplido

Adaptación del Product Backlog

Durante el desarrollo del sprint 3, no se determinó ninguna entrada nueva de historias, únicamente se actualizó el estado las historias de usuario completadas. La Tabla 25 expone el Product backlog actualizado al dar por terminado el sprint 3.

Tabla 25. Product backlog al dar por terminado el sprint 3.

PRODUCT BACKLOG				
HISTORIA ÉPICA	**Historias de Usuario**			
	ID	**Título**	**Estimación (Fibonacci)**	**Estado**
PDS3	PDS3-01	Creación de proyecto base usando Nextjs	5	Terminado
	PDS3-02	Creación de pantalla de Inicio de Sesión y registro de usuario.	8	Por implementar
	PDS3-03	Creación de pantalla para vincular y desvincular dispositivo	5	Por implementar
	PDS3-04	Creación de infogramas para evitar contagios de C-19	3	Terminado
	PDS3-05	Creación de pantalla principal y tabla de históricos de contactos	8	Por implementar
	PDS3-06	Botón reportarse enfermo/sano	3	Por implementar
PDS4	PDS4-01	Creación del proyecto base usando Nest js	5	Terminado
	PDS4-02	Crear endpoints para manejar el registro e inicio de sesión de usuario.	5	Por implementar
	PDS4-03	Crear endpoints para guardar los contactos registrados por el dispositivo IoT.	3	Por implementar
	PDS4-04	Crear endpoints para obtener la información del grafo de contactos y la tabla de históricos.	5	Por implementar
	PDS4-05	Crear endpoint para registrar el estado de enfermo y alertar a los demás nodos.	8	Por implementar
	PDS4-06	Crear endpoint para vincular o desvincular un dispositivo a una cuenta de usuario.	3	Por implementar

2.5.4 Sprint Retrospective

En el gráfico Burndown del sprint 3 que los primeros días de desarrollo se tuvo un cierto retraso debido a que se tuvo que indagar en las documentaciones oficiales de los frameworks para determinar la mejor forma de implementarlos en los proyectos bases.

Por resto la segunda mitad del sprint tuvo una velocidad aceptable. La Figura 24 muestra el gráfico obtenido para el presente sprint.

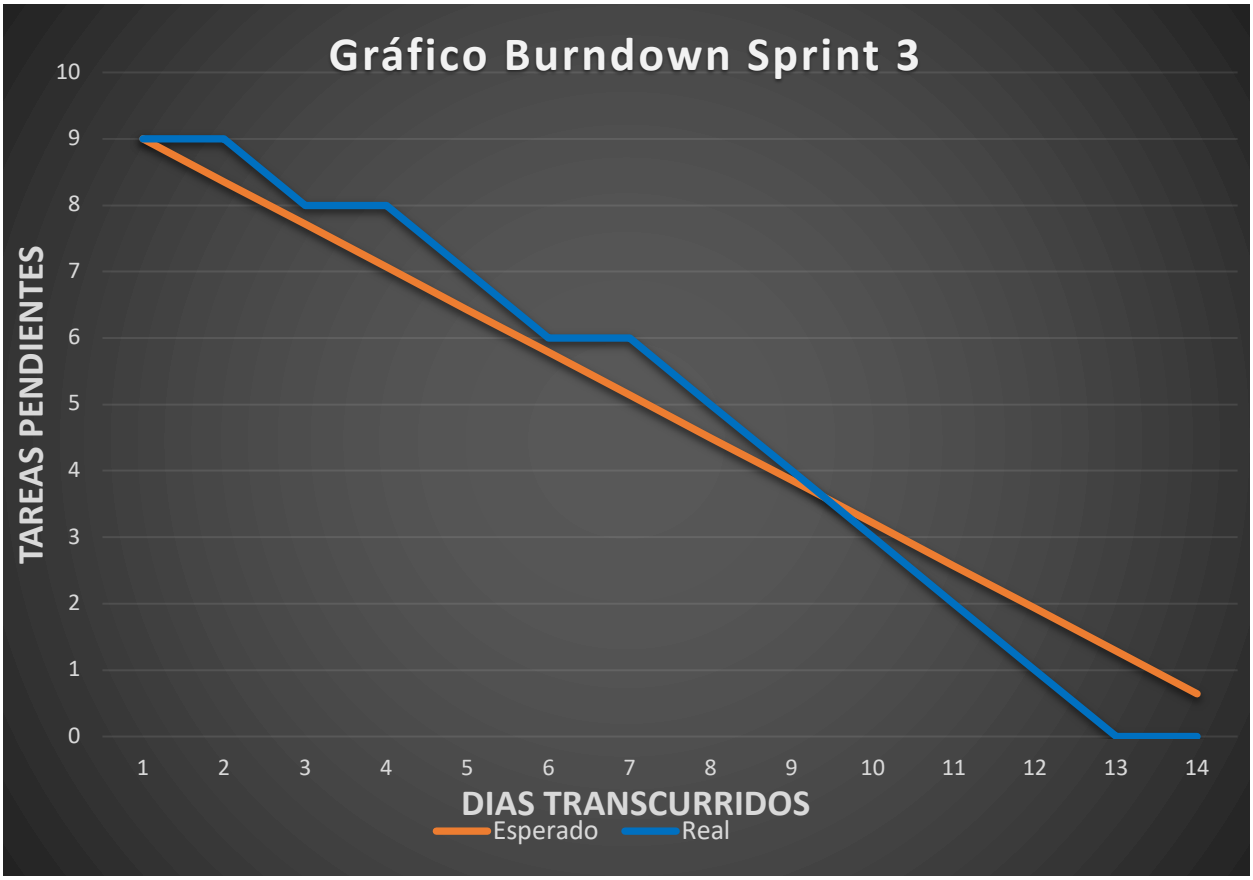


Figura 24. Gráfico Burndown Sprint 3

2.6 Sprint 4

2.6.1 Sprint Planning

Objetivo del Sprint

Implementar el flujo completo de registro e inicio de sesión de usuario.

Sprint Backlog

En la Tabla 26, se exponen las historias a ser desarrolladas en el actual sprint.

Tabla 26. Historias de usuario seleccionadas del Product Backlog del sprint 4.

ID	Descripción	Estimación	Prioridad
PDS3-02	Creación de pantalla de Inicio de Sesión y registro de usuario.	8	Alta
PDS4-02	Crear endpoints para manejar el	5	Alta

	registro e inicio de sesión de usuario.		

Tabla 27. Sprint Backlog del sprint 4.

Sprint Backlog	
Historia de Usuario	**Tareas**
PDS3-02	Crear un formulario de inicio de sesión con los campos: “ingrese email” e “ingrese contraseña y un botón con la leyenda “Ingresar al sistema”.
	Almacenar el JWT en el localstorage para mantener la sesión del usuario.
	Crear formulario de registro de usuario con los siguientes campos: “Nombres y Apellidos”, “Correo Electrónico”, “Contraseña” y “Confirmar Contraseña” y un botón con la leyenda “Registrarse”.
	Implementar un botón para intercambiar de formularios en caso de que sea necesario.
PDS4-02	Exponer endpoint “auth/login” para manejar una petición HTTP de tipo Post de inicio de sesión y responder el JWT al cliente para mantener la sesión activa por 1 día
	Exponer endpoint “users/Register” para manejar una petición HTTP de tipo Post para almacenar la información del nuevo usuario en el sistema.
	Almacenar las contraseñas de los usuarios encriptadas en la base de datos de datos.

2.6.2 Desarrollo del Sprint

Historia PDS4-02: Crear endpoints para manejar el registro e inicio de sesión de usuario.

Para el registro de nuevos usuarios al sistema se expuso un endpoint en la ruta “users/Register” el cual mediante se procesa la información provista en el cuerpo de la petición. La información es almacenada en la base de datos y la contraseña provista por el usuario es almacenada de forma segura. Una vez completado el proceso de registro, se podrá iniciar sesión en el sistema.

Una vez que el usuario se ha registrado en el sistema se debió exponer el endpoint “auth/login” el cual responde un JWT que será almacenado en el cliente para controlar de esa manera la sesión de usuario. La Figura 25 muestra el JWT que el endpoint responde.

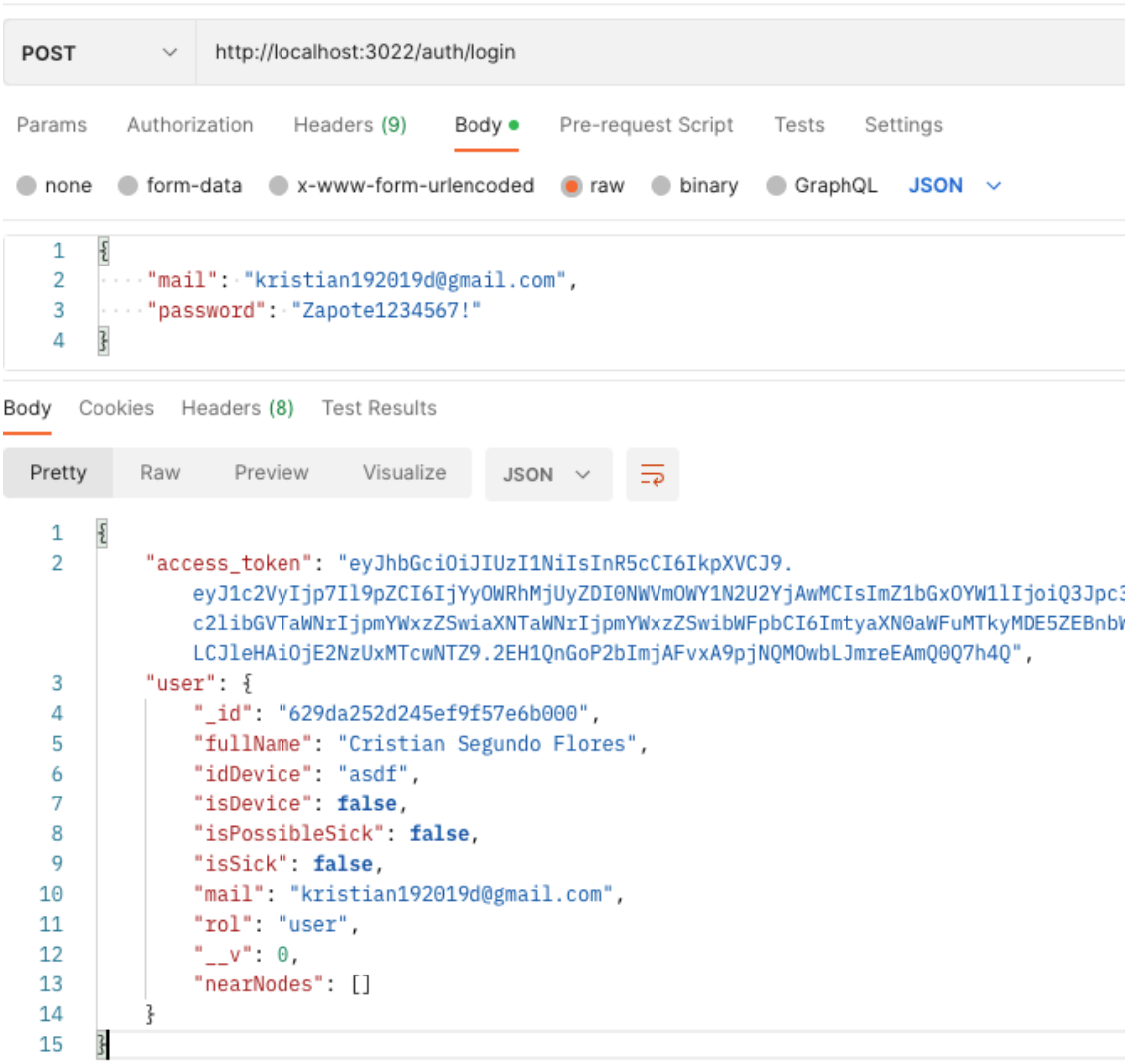


Figura 25. Respuesta del endpoint auth/login.

Dentro de la Figura 26 se puede apreciar la información de un usuario y como las contraseñas se almacenan de manera segura dentro de la base de datos.

Figura 26. Información del usuario almacenada en la base de datos.

Historia PDS3-02: Creación de pantalla de Inicio de Sesión y registro de usuario.

Para completar el formulario de inicio de sesión se hizo uso de la librería MUI para los inputs requeridos para ingresar el correo electrónico y la contraseña. Se implementó el botón “Ingresar al sistema” el cual envía una petición HTTP de tipo POST al endpoint “auth/login” para obtener el JWT y almacenarlo. En la Figura 27, se puede apreciar el localstorage una vez iniciado sesión en el portal web y el JWT almacenado.

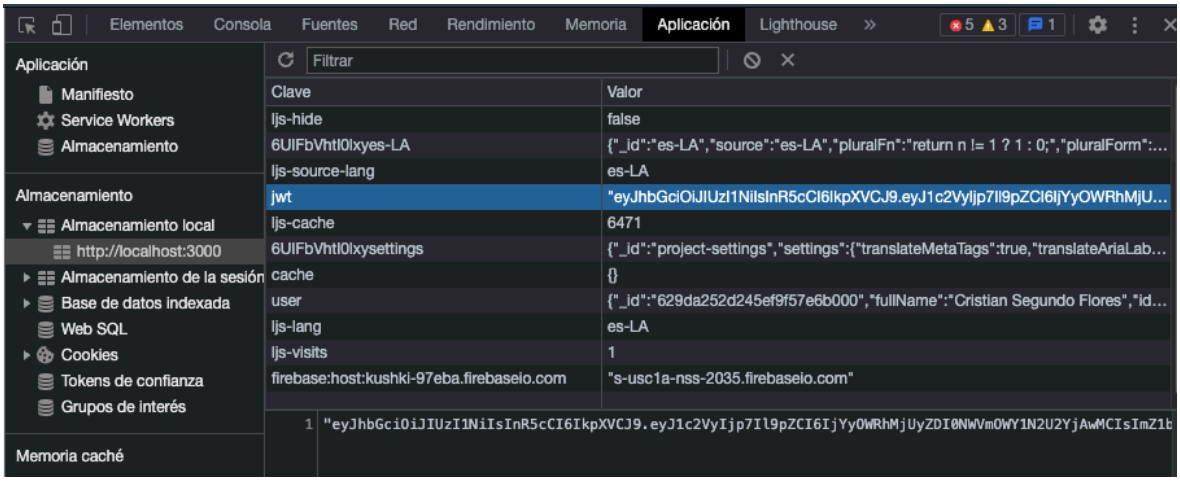


Figura 27. Localstorage del portal web.

Se implementó de igual manera el formulario de registro de usuario con los inputs necesarios para recolectar la información de nombres completos, correo electrónico y contraseña, para enviar los datos proporcionados por el usuario al servidor para ser procesados se hace uso de un botón “Registrarse” para consumir el endpoint “users/Register”. En la Figura 28, se puede apreciar el formulario de registro de la aplicación.

Figura 28. Formulario de registro para el portal web.

2.6.3 Sprint Review

Revisión de Criterios de Aceptación.

Una vez finalizado el sprint, se logró completar con éxito los criterios de aceptación, la Tabla 28 muestra los criterios cumplidos.

Tabla 28. Criterios de aceptación del sprint 4.

Historia de Usuario	Criterio de aceptación	Estado
PDS3-02	Cuando el usuario intente registrarse en el sistema, entonces se desplegará el formulario de registro.	Cumplido
	Cuando el usuario complete el formulario de registro entonces el usuario podrá ingresar al sistema con su usuario y contraseña proporcionados	Cumplido
	Cuando el usuario coloca sus credenciales y da clic en iniciar sesión, entonces el usuario puede ingresar al portal web y ver su información.	Cumplido
PDS4-02	Cuando el usuario envíe su información al servidor,	Cumplido

Historia de Usuario	Criterio de aceptación	Estado
	entonces se podrá visualizar la información registrada exitosamente en la base de datos.	
	Cuando el usuario envíe la información de sus credenciales entonces se podrá visualizar un JWT para gestionar el inicio de sesión en el sistema	Cumplido

Adaptación del Product Backlog

Durante el desarrollo del sprint 4, no se consideró la entrada de nuevas historias de usuario, únicamente se actualizó el estado de las historias completadas. La Tabla 29 muestra el product backlog

Tabla 29. Product backlog tras finalizar el sprint 4.

PRODUCT BACKLOG				
HISTORIA ÉPICA	**Historias de Usuario**			
	ID	**Título**	**Estimación (Fibonacci)**	**Estado**
PDS3	PDS3-01	Creación de proyecto base usando Nextjs	5	Terminado
	PDS3-02	Creación de pantalla de Inicio de Sesión y registro de usuario.	8	Terminado
	PDS3-03	Creación de pantalla para vincular y desvincular dispositivo	5	Por implementar
	PDS3-04	Creación de infogramas para evitar contagios de C-19	3	Terminado
	PDS3-05	Creación de pantalla principal y tabla de históricos de contactos	8	Por implementar
	PDS3-06	Botón reportarse enfermo/sano	3	Por implementar
PDS4	PDS4-01	Creación del proyecto base usando Nest js	5	Terminado
	PDS4-02	Crear endpoints para manejar el registro e inicio de sesión de usuario.	5	Terminado

PRODUCT BACKLOG				
HISTORIA ÉPICA	**Historias de Usuario**			
	ID	**Título**	**Estimación (Fibonacci)**	**Estado**
	PDS4-03	Crear endpoints para guardar los contactos registrados por el dispositivo IoT.	3	Por implementar
	PDS4-04	Crear endpoints para obtener la información del grafo de contactos y la tabla de históricos.	5	Por implementar
	PDS4-05	Crear endpoint para registrar el estado de enfermo y alertar a los demás nodos.	8	Por implementar
	PDS4-06	Crear endpoint para vincular o desvincular un dispositivo a una cuenta de usuario.	3	Por implementar

2.6.4 Sprint Retrospective

En el gráfico Burndown del sprint 4, durante la primera mitad del sprint la velocidad de desarrollo estuvo a la par de lo esperado mientras que la segunda mitad se presentó un aumento en la velocidad de desarrollo, sin embargo, los días 9, 10 y 11 el desarrollo se vio estancado, pero no fue impedimento para terminar el sprint a tiempo, ver Figura 29.

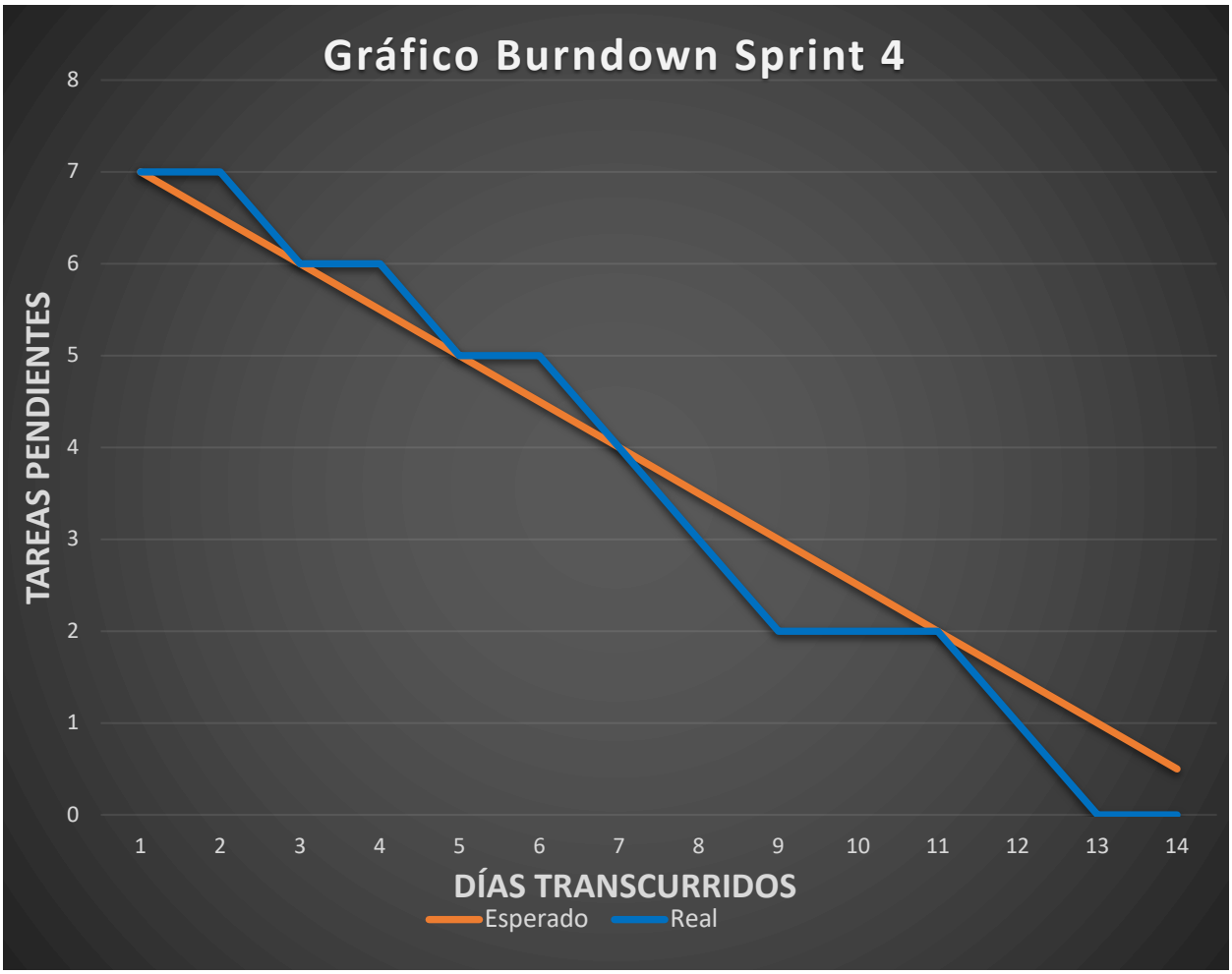


Figura 29. Gráfico Burndown Sprint 4.

2.7 Sprint 5

2.7.1 Sprint Planning

Objetivo del Sprint

Implementar lógica para vincular y desvincular usuarios con los dispositivos IoT.

Sprint Backlog

En la tabla 30, se encuentran las historias a ser realizadas en el actual sprint.

Tabla 30. Historias de usuario seleccionadas del Product Backlog del sprint 5.

ID	Descripción	Estimación	Prioridad
PDS3-03	Creación de pantalla para vincular y desvincular dispositivo	5	Alta
PDS4-03	Crear endpoints para guardar los	3	Alta

ID	Descripción	Estimación	Prioridad
	contactos registrados por el dispositivo IoT.		
PDS4-06	Crear endpoint para vincular o desvincular un dispositivo a una cuenta de usuario.	3	Alta

Tabla 31. Sprint Backlog del sprint 5.

Sprint Backlog	
Historia de Usuario	**Tareas**
PDS3-03	Implementar una cabecera de navegación para el portal web
	Agregar la nueva pantalla dentro de la ruta /linkDevice.
	Implementar un botón para desvincular el dispositivo en caso de que el usuario ya tenga un dispositivo vinculado a su cuenta.
	Implementar un botón de vincular y un textfield para ingresar el código del dispositivo a vincular en caso de la cuenta no tenga ningún dispositivo vinculado.
PDS4-03	Exponer endpoint HTTP de tipo POST /contacts/init y configurar el acceso al mismo mediante jwt con el rol de dispositivo.
	Para el endpoint /contacts/init se debe permitir parámetros de cuerpo con la siguiente información: idDevice de tipo string, idConctactDevice de tipo string, rssi de tipo number e isInit de tipo boolean.
	En caso de que la petición sea exitosa entonces el endpoint deberá responder un estado 201 caso contrario un 304.
PDS4-06	Exponer endpoint HTTP de tipo PATCH /users/vinculateDevice y configurar el acceso al mismo mediante jwt con el rol de usuario.
	En caso de que la petición sea exitosa entonces el endpoint deberá responder un estado 200, caso contrario un 304

2.7.2 Desarrollo del Sprint

Historia PDS4-03: Crear endpoints para guardar los contactos registrados por el dispositivo IoT.

Durante la creación del endpoint fue necesario configurar un decorador personalizado para controlar el acceso por roles. El archivo roles.decorator.ts es el encargado de gestionar que roles serán los que puedan acceder al recurso. De esa manera,

únicamente las peticiones que contengan el jwt proporcionado por el mismo sistema y además sea del rol correcto podrán obtener una respuesta exitosa del endpoint.

Para controlar el inicio y finalización de los contactos se implementó un parámetro de cuerpo llamado isInit de tipo booleano, dependiendo de este parámetro si es positivo entonces la información del contacto se almacenará en la base de datos, en caso de ser negativo se buscará el último registro con fecha de finalización del contacto en 0 para actualizar esa fecha por la actual. Con ello se consigue que cada registro de contacto tenga la información de la hora de inicio y la hora de fin del contacto.

Historia PDS4-06: Crear endpoint para vincular o desvincular un dispositivo a una cuenta de usuario.

La implementación del endpoint encargado de vincular o desvincular el dispositivo se lo ha realizado dentro del módulo de usuarios ya que, dentro de la tabla que almacena la información de los usuarios existe un campo llamado idDevice el cual almacena el id del dispositivo IoT. Para evitar inconsistencia en la vinculación de los dispositivos con las cuentas de usuario, es decir que por ejemplo existan dos registros en la tabla de usuarios que posean el mismo idDevice, se realiza una búsqueda para determinar si el id del dispositivo que se pretende vincular ya está vinculado a alguien más. Si se determina que no existe ninguna inconsistencia, entonces se procede a modificar la información del usuario actualizando el id del dispositivo y de esta manera realizar la vinculación.

Historia PDS3-03: Creación de pantalla para vincular y desvincular dispositivo.

Para la creación de la pantalla de vincular y desvincular el dispositivo simplemente fue necesario implementar un pequeño formulario para obtener el id del dispositivo. Cabe recalcar que el id del dispositivo descrito previamente en la historia de usuario PDS2-01 debe ser previamente compartida con el usuario del dispositivo al momento de su entrega. Dicho formulario contiene una entrada de texto y alado un botón para enviar la información por medio de HTTP al servidor para que la información sea almacenada correctamente en la base de datos, esto para el proceso de vinculación. La Figura 30 muestra la pantalla para vincular un dispositivo al usuario con sesión activa.

Figura 30. Pantalla de vincular/desvincular flujo vincular.

Para el proceso de desvinculación del dispositivo, simplemente se implementó un botón para desvincular y al darle clic se envía una petición HTTP al servidor para desvincular a ese dispositivo de la cuenta de usuario. La Figura 31 muestra la pantalla de vincular/desvincular con un dispositivo listo para ser desvinculado.

Figura 31. Pantalla de vincular/desvincular flujo desvincular.

2.7.3 Sprint Review

Revisión de Criterios de Aceptación.

Una vez finalizado el sprint, se logró completar con éxito los criterios de aceptación, la Tabla 32 muestra los criterios cumplidos.

Tabla 32. Criterios de aceptación del sprint 5.

Historia de Usuario	Criterio de aceptación	Estado
PDS3-03	Cuando se encuentre en la pantalla de vincular/desvincular dispositivo, entonces se podrá vincular el dispositivo mediante el nombre del dispositivo.	Cumplido
	Cuando se encuentre en la pantalla de vincular/desvincular dispositivo, entonces se podrá desvincular el dispositivo mediante un botón.	Cumplido
PDS4-03	Cuando el usuario se encuentre a una distancia inferior a 1,5 metros de otro usuario, entonces se podrá visualizar el contacto registrado exitosamente en la base de datos.	Cumplido
PDS4-06	Cuando este esté en la pantalla de vincular dispositivo, entonces se podrá vincular su dispositivo usando el código único para cada dispositivo.	Cumplido
	Cuando este esté en la pantalla de vincular dispositivo, entonces se podrá vincular su dispositivo de esa cuenta de usuario.	Cumplido

Adaptación del Product Backlog

Durante el desarrollo del sprint 4, no se consideró la entrada de nuevas historias de usuario, únicamente se actualizó el estado de las historias completadas. La Tabla 33 muestra el product backlog

Tabla 33. Product backlog tras finalizar el sprint 5.

PRODUCT BACKLOG				
HISTORIA ÉPICA	**Historias de Usuario**			
	ID	**Título**	**Estimación (Fibonacci)**	**Estado**
PDS3	PDS3-01	Creación de proyecto base usando Nextjs	5	Terminado
	PDS3-02	Creación de pantalla de Inicio de Sesión y registro de usuario.	8	Terminado
	PDS3-03	Creación de pantalla para vincular y desvincular dispositivo	5	Terminado
	PDS3-04	Creación de infogramas para evitar contagios de C-19	3	Terminado
	PDS3-05	Creación de pantalla principal y tabla de históricos de contactos	8	Por implementar
	PDS3-06	Botón reportarse enfermo/sano	3	Por implementar
PDS4	PDS4-01	Creación del proyecto base usando Nest js	5	Terminado
	PDS4-02	Crear endpoints para manejar el registro e inicio de sesión de usuario.	5	Terminado
	PDS4-03	Crear endpoints para guardar los contactos registrados por el dispositivo IoT.	3	Terminado
	PDS4-04	Crear endpoints para obtener la información del grafo de contactos y la tabla de históricos.	5	Por implementar
	PDS4-05	Crear endpoint para registrar el estado de enfermo y alertar a los demás nodos.	8	Por implementar
	PDS4-06	Crear endpoint para vincular o desvincular un dispositivo a una cuenta de usuario.	3	Terminado

2.7.4 Sprint Retrospective

El diagrama Burndown muestra que el desarrollo logró completarse de manera exitosa, antes de la fecha de finalización y completando todas las tareas planificadas durante el

sprint. El equipo de desarrollo fue capaz de mantener una velocidad constante durante todo el desarrollo lo que sugiere que la planificación y estimación de las tareas fue precisa. Es importante destacar que en los primeros días el número de tareas pendientes fue mayor al esperado lo que puede indicar que la estimación de esas historias de usuario fue demasiado optimista. La Figura 32 muestra el gráfico Burndown del sprint 5.

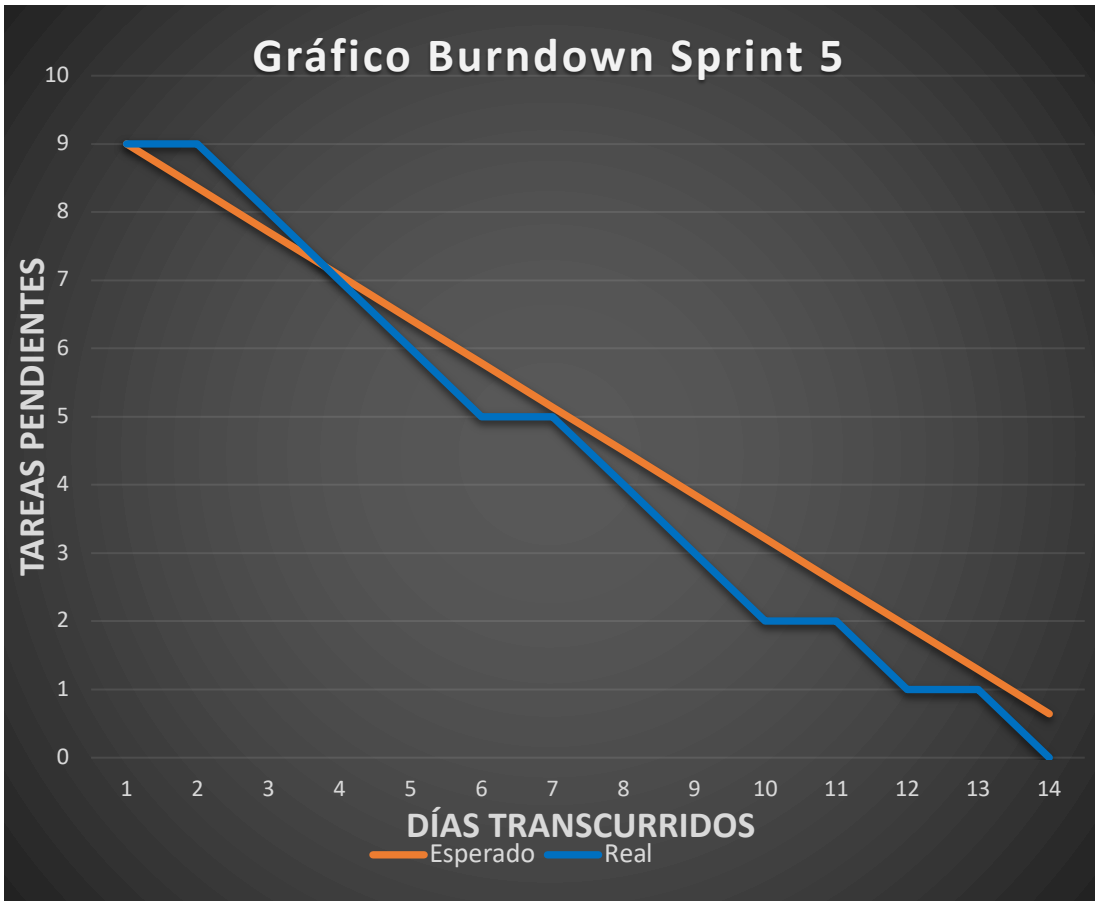


Figura 32. Gráfico Burndown Sprint 5.

2.8 Sprint 6

2.8.1 Sprint Planning

Objetivo del Sprint

Implementar grafo de contactos cercanos realizados y tabla de historial de contactos.

Sprint Backlog

En la Tabla 34, se encuentran las historias a ser realizadas en el actual sprint.

Tabla 34. Historias de usuario seleccionadas del Product Backlog del sprint 6.

ID	Descripción	Estimación	Prioridad
PDS3-05	Creación de pantalla principal y tabla de históricos de contactos	8	Alta
PDS4-04	Crear endpoints para obtener la información del grafo de contactos y la tabla de históricos.	5	Alta

Tabla 35. Sprint Backlog del sprint 6.

Sprint Backlog	
Historia de Usuario	**Tareas**
PDS3-05	Crear la ruta /home para alojar la pantalla principal del sistema.
	Crear la ruta /history para alojar la tabla de contactos históricos
	Implementar un gráfico de nodos utilizando la información histórica junto con la librería d3.js y alojara en la página principal.
	Implementar una tabla de históricos con las columnas Contactos, Duración, Id del dispositivo y Fecha de contacto.
	Para obtener la información necesaria para graficar el nodo consumir el endpoint /contacts/data/:idDevice.
	Para obtener la información necesaria para la tabla de históricos consumir el endpoint /contacts/getContacts/:idDevice
PDS4-04	Exponer endpoint HTTP de tipo GET en la ruta /contacts/data/:idDevice para obtener la información necesaria para realizar el gráfico de nodos.
	Exponer endpoint HTTP de tipo GET en la ruta /contacts/getContacts/:idDevice para obtener la información necesaria para la tabla de históricos.
	El endpoint /contacts/data/:idDevice deberá ser accesible únicamente para usuarios con el rol “user”.
	El endpoint /contacts/getContacts/:idDevice deberá ser accesible únicamente para usuarios con el rol “user”.

2.8.2 Desarrollo del Sprint

Historia PDS4-04: Crear endpoints para obtener la información del grafo de contactos y la tabla de históricos.

Para exponer el endpoint /contacts/data/:idDevice fue necesario implementar una función asíncrona para realizar múltiples consultas a la base de datos y construir un objeto de respuesta que tiene las propiedades nodes y links. La propiedad nodes es una matriz de objetos que representan los dispositivos y contactos encontrados, mientras que la propiedad links es una matriz de objetos que representan las relaciones entre los dispositivos y contactos encontrados.

De igual forma para exponer el endpoint /contacts/getContacts/:idDevice se implementó una función asíncrona que realiza consultas a la base de dato para obtener la información sobre los contactos asociados a un dispositivo en particular. Como entrada recibe el id del dispositivo para obtener el historial de contacto. Con la respuesta de la base de datos se construye un objeto con las propiedades duration, idDevice y date, información que será retornada al cliente que realizó la petición.

Una vez expuestos los endpoints, se les agregó la capa de autorización por roles para admitir únicamente a las peticiones que contengan el tipo de rol de usuario.

Historia PDS3-05: Creación de pantalla principal y tabla de históricos de contactos.

La creación de las rutas /home y /history no fue relativamente complicada gracias a la facilidad que provee Nextjs en la creación de nuevas rutas con sus respectivos contenidos. Para la pantalla principal fue necesario usar la librería d3.js con la cual se puede representar información de forma gráfica y ha sido usado para renderizar un grafo como muestra la Figura 33.

Figura 33. Grafo de contactos realizados.

Para la tabla de históricos de igual manera se consume uno de los endpoints creados en la historia pasada y con la información que este responde se realiza la tabla como se muestra en la Figura 34.

Contactos	Duración	ID del dispositivo	Fecha de contacto
MD MARIA DOLORES	· 3 MINUTOS	2	2022/06/22
MD MARIA DOLORES	· 15 SEGUNDOS	2	2022/06/22

Figura 34. Tabla de históricos de contactos.

2.8.3 Sprint Review

Una vez finalizado el sprint, se logró completar con éxito los criterios de aceptación, la Tabla 36 muestra los criterios cumplidos.

Tabla 36. Criterios de aceptación del sprint 6.

Historia de Usuario	Criterio de aceptación	Estado
PDS3-05	Cuando el usuario se encuentre en la pantalla principal del portal, entonces se podrá visualizar un gráfico de	Cumplido

	nodos mostrando los contactos registrados.	
	Cuando el usuario se encuentre en la pantalla de históricos, entonces podrá visualizar una tabla con los contactos de hasta hace 2 semanas.	Cumplido
PDS4-04	Cuando el usuario se dirija a la ruta inicial del proyecto, entonces se podrá visualizar una petición HTTP de tipo GET exitosa.	Cumplido

Adaptación del Product Backlog

Durante el desarrollo del sprint 6, no se consideró la entrada de nuevas historias de usuario debido a que las funcionalidades del proyecto van de acuerdo con los requerimientos del cliente, ver Tabla 37.

Tabla 37. Product backlog tras finalizar el sprint 6.

PRODUCT BACKLOG				
HISTORIA ÉPICA	**Historias de Usuario**			
	ID	**Título**	**Estimación (Fibonacci)**	**Estado**
PDS3	PDS3-01	Creación de proyecto base usando Nextjs	5	Terminado
	PDS3-02	Creación de pantalla de Inicio de Sesión y registro de usuario.	8	Terminado
	PDS3-03	Creación de pantalla para vincular y desvincular dispositivo	5	Terminado
	PDS3-04	Creación de infogramas para evitar contagios de C-19	3	Terminado
	PDS3-05	Creación de pantalla principal y tabla de históricos de contactos	8	Terminado
	PDS3-06	Botón reportarse enfermo/sano	3	Por implementar
PDS4	PDS4-01	Creación del proyecto base usando Nest js	5	Terminado
	PDS4-02	Crear endpoints para manejar el registro e inicio de sesión de usuario.	5	Terminado

	PDS4-03	Crear endpoints para guardar los contactos registrados por el dispositivo IoT.	3	Terminado
	PDS4-04	Crear endpoints para obtener la información del grafo de contactos y la tabla de históricos.	5	Terminado
	PDS4-05	Crear endpoint para registrar el estado de enfermo y alertar a los demás nodos.	8	Por implementar
	PDS4-06	Crear endpoint para vincular o desvincular un dispositivo a una cuenta de usuario.	3	Terminado

2.8.4 Sprint Retrospective

El gráfico Burndown para este sprint muestra que las tareas lograron completarse con éxito antes de la fecha de finalización del sprint. Para los días 10 y 11 se presentó un estancamiento en el desarrollo del sprint debido a la curva de aprendizaje que la librería d3.js ya que esta fue necesaria para realizar el grafo de nodos. Sin embargo, este retraso fue solventado en los días posteriores. En la Figura 35, se muestra el gráfico Burndown del presente sprint.

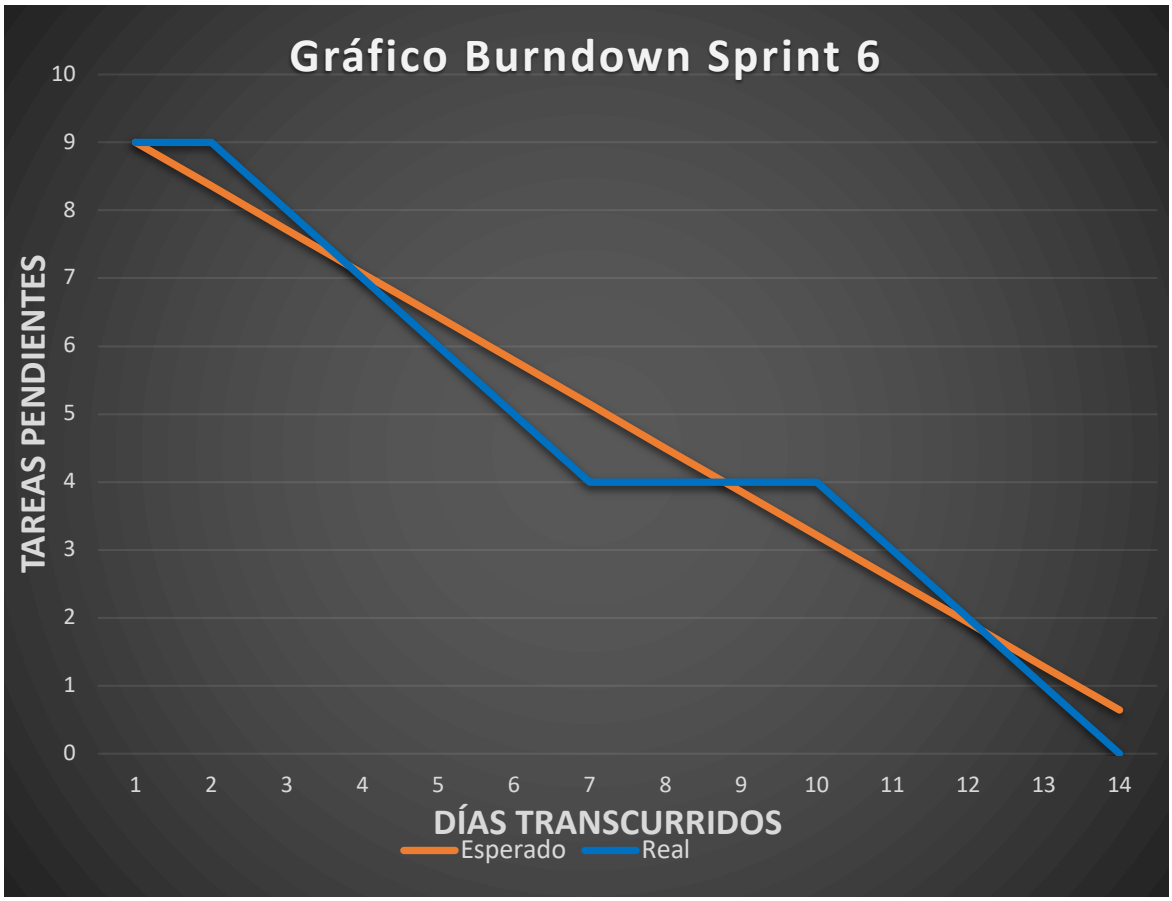


Figura 35. Gráfico Burndown Sprint 6.

2.9 Sprint 7

2.9.1 Sprint Planning

Objetivo del Sprint

Implementar funcionalidad de registro de contagio y alerta de posible contagio.

Sprint Backlog

En la Tabla 38, se encuentran las historias a ser realizadas en el actual sprint.

Tabla 38. Historias de usuario seleccionadas del Product Backlog del sprint 7.

ID	Descripción	Estimación	Prioridad
PDS3-06	Botón reportarse enfermo/sano	3	Alta
PDS4-05	Crear endpoint para registrar el estado de enfermo y alertar a los demás nodos.	8	Alta

Tabla 39. Sprint Backlog del sprint 7.

Sprint Backlog	
Historia de Usuario	**Tareas**
PDS3-06	Implementar el botón “REPORTARSE ENFERMO” el cual será usado para abrir el modal de confirmación de reportarse enfermo.
	El modal de confirmación de reportarse enfermo tendrá el mensaje ¿Ha resultado positivo en una prueba de COVID-19?, en caso de dar clic en sí, enviar una petición de tipo PATCH al endpoint /users/updateIsSick para actualizar el estado de enfermedad a positivo.
	Implementar el botón “REPORTARSE SANO” el cual será usado para abrir el modal de confirmación de reportarse sano.
	El modal de confirmación de reportarse sano tendrá el mensaje ¿Ha resultado negativo en una prueba de COVID-19?, en caso de dar clic en sí, enviar una petición de tipo PATCH al endpoint /users/updateIsSick para actualizar el estado de enfermedad a negativo.
	Cuando el usuario se encuentre en la pantalla principal se deberá realizar una conexión al websocket.
	Si el websocket recibe el mensaje de posible enfermedad, entonces se deberá recargar la información del usuario y mostrar una alerta de posible contagio en la pantalla principal.
PDS4-05	Exponer endpoint HTTP de tipo PATCH en la ruta /users/updateIsSick para actualizar el estado de enfermedad de un usuario.
	Crear un websocket para manejar el estado de alerta de posible enfermedad
	Para el caso en que algún usuario actualiza su estado de enfermedad a positivo entonces los usuarios que han tenido contacto con el usuario recibirán un mensaje de posible contagio por medio del websocket.

2.9.2 Desarrollo del Sprint

Historia PDS4-05: Crear endpoint para registrar el estado de enfermo y alertar a los demás nodos.

Para exponer el endpoint /users/updateIsSick fue necesario implementar una función que actualice el estado de enfermedad del usuario. Esta función recibe como parámetro el correo electrónico del usuario y el estado de la enfermedad, si es positiva o negativa. Si la actualización del estado es positiva entonces todos los usuarios que han tenido

contacto serán actualizados en el campo isPossibleSick para ser alertados de un posible contagio. Finalmente se enviará un mensaje por medio del websocket para generar una alerta visual en la pantalla home.

En la creación del websocket se necesitó hacer uso de la librería socket.io. Primero se creó un submódulo especializado para manejar el inicio conexión y la desconexión. Una vez implementado el submódulo se hace uso de este dentro del submódulo de usuario para ser notificados sobre una posible alerta de contagio.

Historia PDS3-06: Botón reportarse enfermo/sano

Para completar esta historia de usuario se implementó un botón dentro de la pantalla principal como muestra la Figura 36. Al dar clic en el botón se desplegará un modal para confirmar la actualización del estado de contagio. De igual forma si el usuario se reportó enfermo, el botón cambiará a Reportarse Sano y de la misma manera al dar clic se mostrará un modal para confirmar la actualización.

Figura 36. Botón de reportarse enfermo.

En la Figura 37, se puede apreciar el mensaje de alerta que es recibida por medio de una conexión websocket. La conexión del websocket inicia cuando el usuario entra en la pantalla principal del portal. La conexión finaliza cuando el usuario finaliza la sesión.

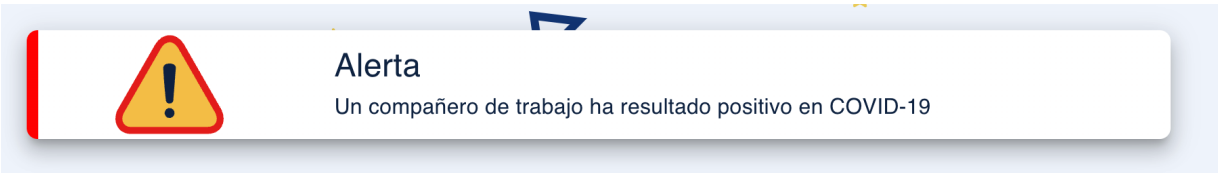


Figura 37. Alerta de posible contagio.

2.9.3 Sprint Review

Una vez finalizado el sprint, se logró completar con éxito los criterios de aceptación, la Tabla 40 muestra los criterios cumplidos.

Tabla 40. Criterios de aceptación del sprint 7.

Historia de Usuario	Criterio de aceptación	Estado
PDS3-06	Cuando el usuario se encuentre en la pantalla principal del portal, entonces se podrá visualizar un botón para poder reportase enfermo y al darle clic alertará a los usuarios que tuvieron un contacto cercano.	Cumplido
	Cuando el usuario se encuentre en la pantalla principal del portal, entonces se podrá visualizar un botón para reportarse sano y al darle clic este cambiará el estado de enfermo a sano del usuario.	Cumplido
PDS4-05	Cuando se haya enviado la petición HTTP de tipo PATCH, entonces se podrá visualizar en la base de datos que su estado de enfermedad ha cambiado a true.	Cumplido
	Cuando se haya enviado la petición HTTP de tipo PATCH, entonces se podrá visualizar en la base de datos que el estado de su enfermedad ha cambiado a false.	Cumplido

Adaptación del Product Backlog

Tras finalizar el sprint 7, no se identificó ningún cambio a ser realizado, por lo que, todas las historias del Product Backlog quedan en estado de finalizado, concluyendo así el desarrollo, ver Tabla 41.

Tabla 41. Product backlog tras finalizar el sprint 7.

PRODUCT BACKLOG				
HISTORIA ÉPICA	**Historias de Usuario**			
	ID	**Título**	**Estimación (Fibonacci)**	**Estado**
PDS3	PDS3-01	Creación de proyecto base usando Nextjs	5	Terminado
	PDS3-02	Creación de pantalla de Inicio de Sesión y registro de usuario.	8	Terminado
	PDS3-03	Creación de pantalla para vincular y desvincular dispositivo	5	Terminado
	PDS3-04	Creación de infogramas para evitar contagios de C-19	3	Terminado
	PDS3-05	Creación de pantalla principal y tabla de históricos de contactos	8	Terminado
	PDS3-06	Botón reportarse enfermo/sano	3	Terminado
PDS4	PDS4-01	Creación del proyecto base usando Nest js	5	Terminado
	PDS4-02	Crear endpoints para manejar el registro e inicio de sesión de usuario.	5	Terminado
	PDS4-03	Crear endpoints para guardar los contactos registrados por el dispositivo IoT.	3	Terminado
	PDS4-04	Crear endpoints para obtener la información del grafo de contactos y la tabla de históricos.	5	Terminado
	PDS4-05	Crear endpoint para registrar el estado de enfermo y alertar a los demás nodos.	8	Terminado
	PDS4-06	Crear endpoint para vincular o desvincular un dispositivo a una cuenta de usuario.	3	Terminado

2.9.4 Sprint Retrospective

Finalmente, para el último análisis del desarrollo se tiene la Figura 38 que muestra el gráfico Burndown generado en este sprint. En aspectos generales el sprint resultó un

éxito. Las tareas se completaron antes de la fecha de finalización del sprint lo que demuestra que la estimación de las historias fue correcta.

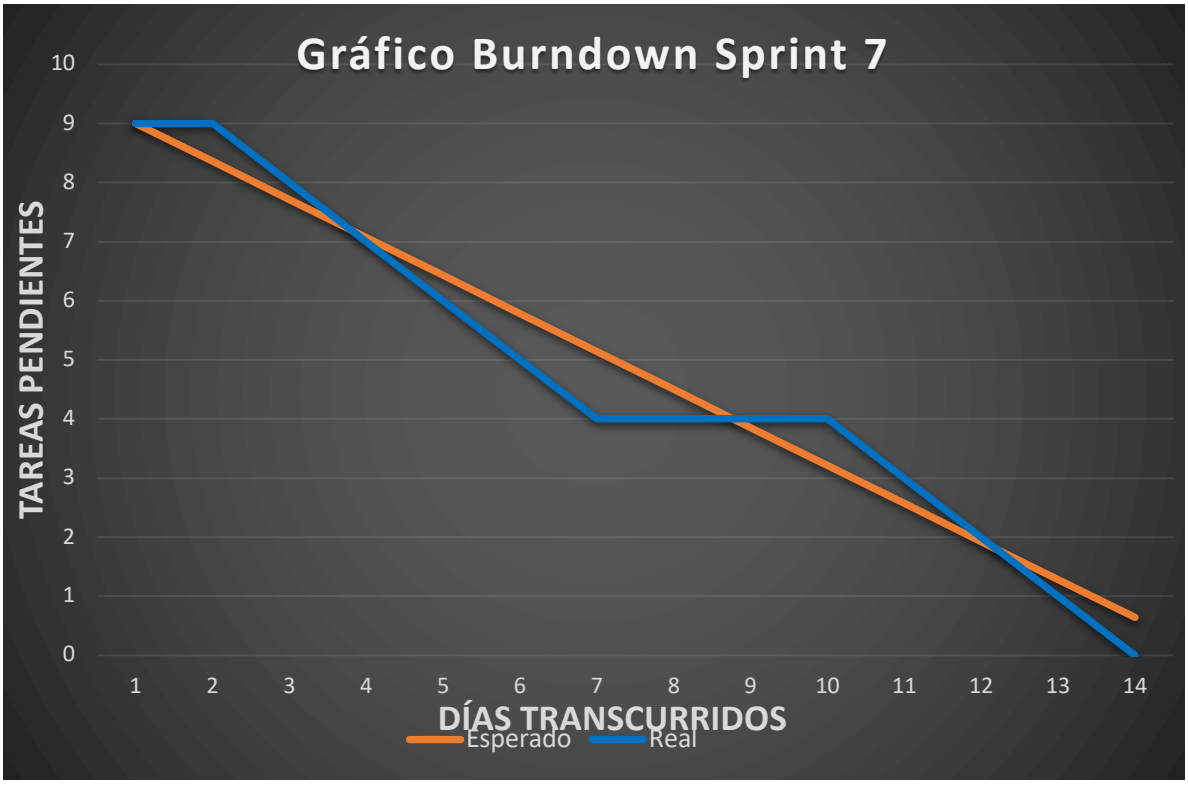


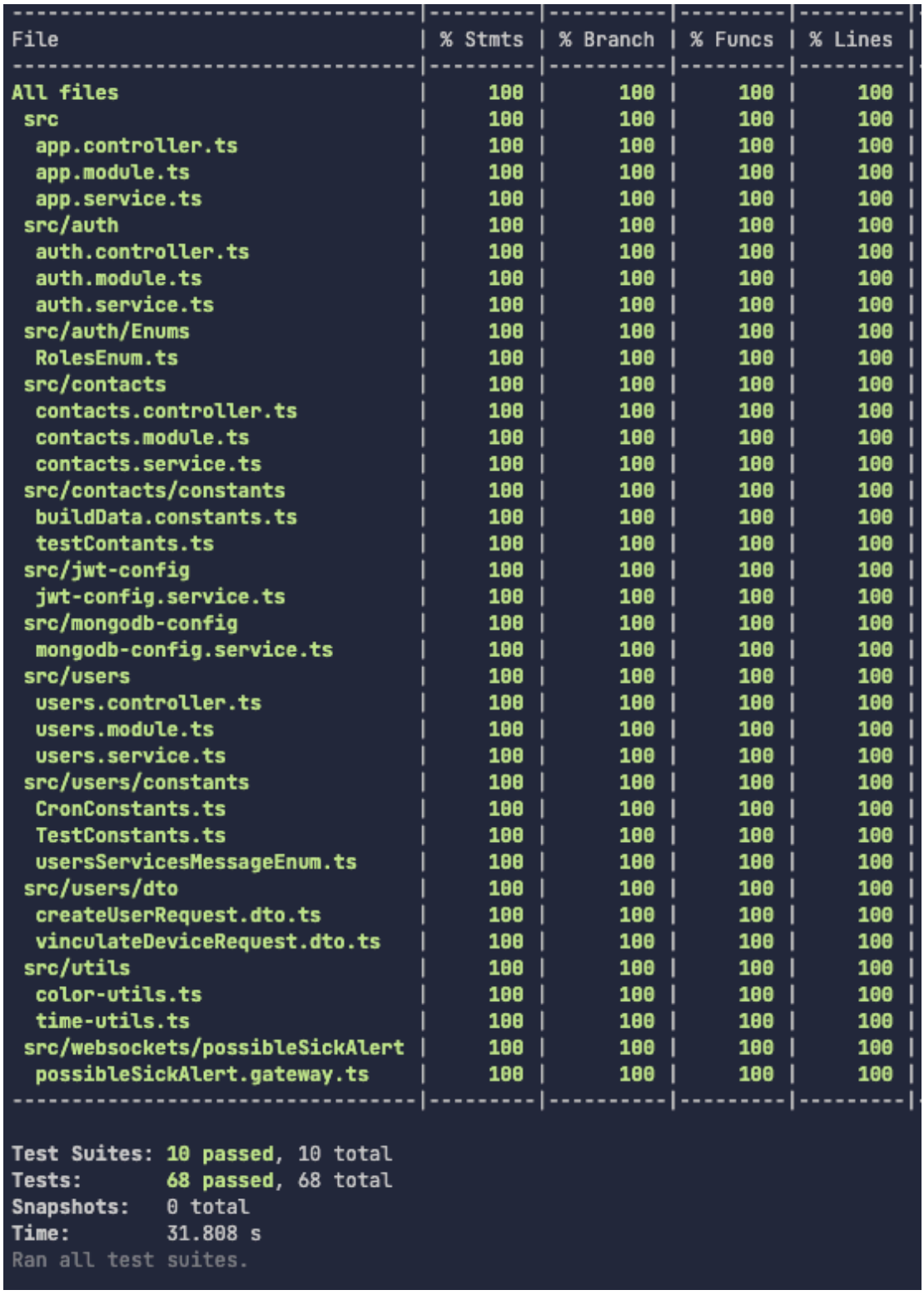
Figura 38. Gráfico Burndown Sprint 7.

3 PRUEBAS DEL SISTEMA

3.1 Pruebas unitarias.

Para garantizar la calidad y la fiabilidad del sistema completo fue necesario implementar pruebas unitarias. Debido a que el prototipo de la solución IoT consta de múltiples dispositivos y componentes interconectados, es altamente probable que se presenten errores en el sistema. Las pruebas unitarias permiten probar individualmente cada uno de estos componentes y verificar que su funcionamiento sea el esperado.

La Figura 39 muestra el porcentaje de líneas de código cubiertas por pruebas unitarias para el servidor. Mostrando que las 68 pruebas unitarias han pasado con éxito y cubriendo así el 100% de líneas de código.

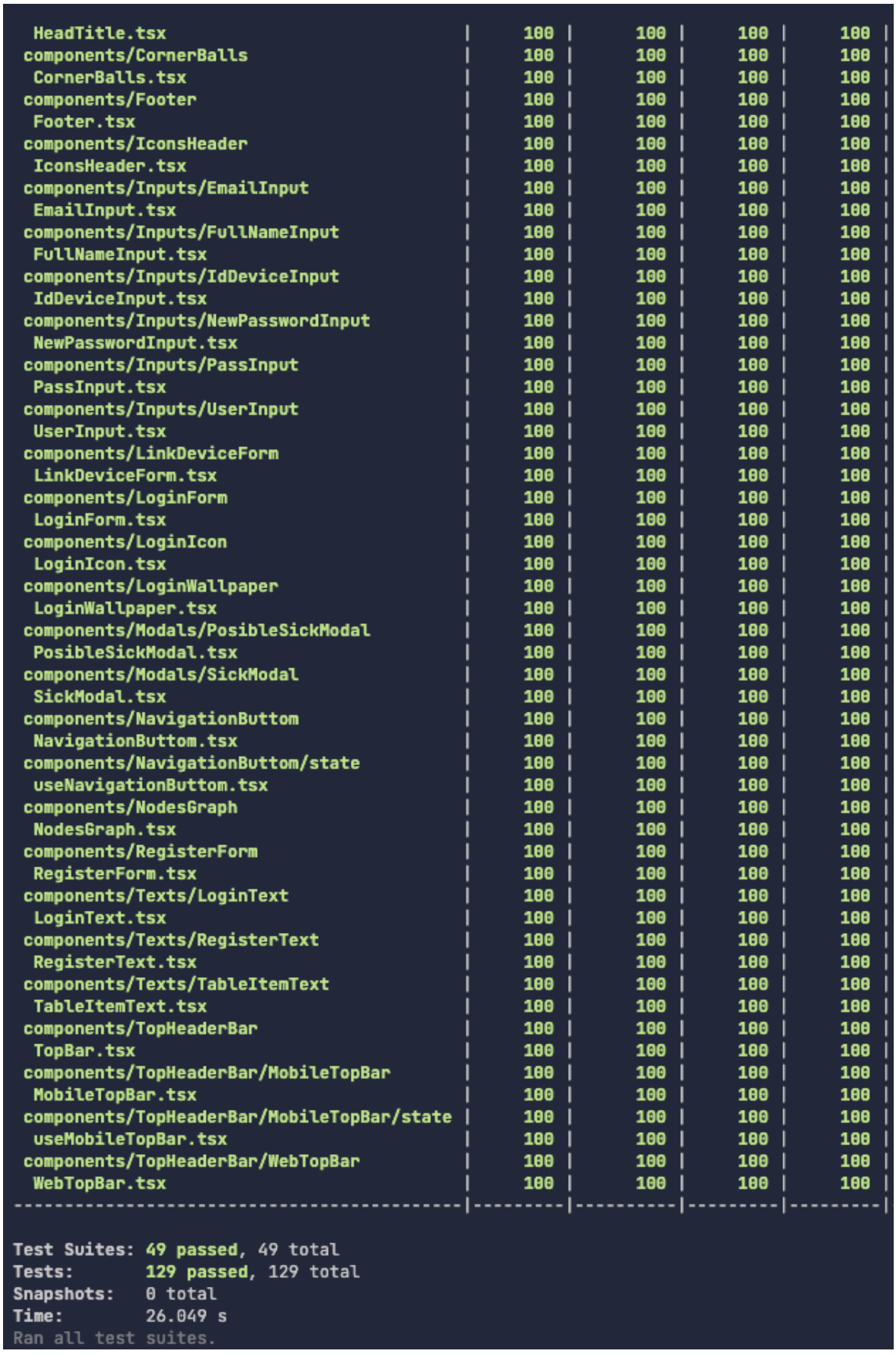


File	% Stmts	% Branch	% Funcs	% Lines
All files	100	100	100	100
src	100	100	100	100
app.controller.ts	100	100	100	100
app.module.ts	100	100	100	100
app.service.ts	100	100	100	100
src/auth	100	100	100	100
auth.controller.ts	100	100	100	100
auth.module.ts	100	100	100	100
auth.service.ts	100	100	100	100
src/auth/Enums	100	100	100	100
RolesEnum.ts	100	100	100	100
src/contacts	100	100	100	100
contacts.controller.ts	100	100	100	100
contacts.module.ts	100	100	100	100
contacts.service.ts	100	100	100	100
src/contacts/constants	100	100	100	100
buildData.constants.ts	100	100	100	100
testContants.ts	100	100	100	100
src/jwt-config	100	100	100	100
jwt-config.service.ts	100	100	100	100
src/mongodb-config	100	100	100	100
mongodb-config.service.ts	100	100	100	100
src/users	100	100	100	100
users.controller.ts	100	100	100	100
users.module.ts	100	100	100	100
users.service.ts	100	100	100	100
src/users/constants	100	100	100	100
CronConstants.ts	100	100	100	100
TestConstants.ts	100	100	100	100
usersServicesMessageEnum.ts	100	100	100	100
src/users/dto	100	100	100	100
createUserRequest.dto.ts	100	100	100	100
vinculateDeviceRequest.dto.ts	100	100	100	100
src/utils	100	100	100	100
color-utils.ts	100	100	100	100
time-utils.ts	100	100	100	100
src/websockets/possibleSickAlert	100	100	100	100
possibleSickAlert.gateway.ts	100	100	100	100

```
Test Suites: 10 passed, 10 total
Tests:       68 passed, 68 total
Snapshots:   0 total
Time:        31.808 s
Ran all test suites.
```

Figura 39. Pruebas unitarias del servidor.

La Figura 40, por otro lado, muestra el porcentaje de líneas de código cubiertas por pruebas unitarias para el aplicativo web. Mostrando que las 129 pruebas unitarias han pasado con éxito y cubriendo así el 100% de líneas de código.



```
 HeadTitle.tsx                                |     100 |      100 |     100 |     100 |
components/CornerBalls                        |     100 |      100 |     100 |     100 |
 CornerBalls.tsx                              |     100 |      100 |     100 |     100 |
components/Footer                             |     100 |      100 |     100 |     100 |
 Footer.tsx                                   |     100 |      100 |     100 |     100 |
components/IconsHeader                        |     100 |      100 |     100 |     100 |
 IconsHeader.tsx                              |     100 |      100 |     100 |     100 |
components/Inputs/EmailInput                  |     100 |      100 |     100 |     100 |
 EmailInput.tsx                               |     100 |      100 |     100 |     100 |
components/Inputs/FullNameInput               |     100 |      100 |     100 |     100 |
 FullNameInput.tsx                            |     100 |      100 |     100 |     100 |
components/Inputs/IdDeviceInput               |     100 |      100 |     100 |     100 |
 IdDeviceInput.tsx                            |     100 |      100 |     100 |     100 |
components/Inputs/NewPasswordInput            |     100 |      100 |     100 |     100 |
 NewPasswordInput.tsx                         |     100 |      100 |     100 |     100 |
components/Inputs/PassInput                   |     100 |      100 |     100 |     100 |
 PassInput.tsx                                |     100 |      100 |     100 |     100 |
components/Inputs/UserInput                   |     100 |      100 |     100 |     100 |
 UserInput.tsx                                |     100 |      100 |     100 |     100 |
components/LinkDeviceForm                     |     100 |      100 |     100 |     100 |
 LinkDeviceForm.tsx                           |     100 |      100 |     100 |     100 |
components/LoginForm                          |     100 |      100 |     100 |     100 |
 LoginForm.tsx                                |     100 |      100 |     100 |     100 |
components/LoginIcon                          |     100 |      100 |     100 |     100 |
 LoginIcon.tsx                                |     100 |      100 |     100 |     100 |
components/LoginWallpaper                     |     100 |      100 |     100 |     100 |
 LoginWallpaper.tsx                           |     100 |      100 |     100 |     100 |
components/Modals/PosibleSickModal            |     100 |      100 |     100 |     100 |
 PosibleSickModal.tsx                         |     100 |      100 |     100 |     100 |
components/Modals/SickModal                   |     100 |      100 |     100 |     100 |
 SickModal.tsx                                |     100 |      100 |     100 |     100 |
components/NavigationButtom                   |     100 |      100 |     100 |     100 |
 NavigationButtom.tsx                         |     100 |      100 |     100 |     100 |
components/NavigationButtom/state             |     100 |      100 |     100 |     100 |
 useNavigationButtom.tsx                      |     100 |      100 |     100 |     100 |
components/NodesGraph                         |     100 |      100 |     100 |     100 |
 NodesGraph.tsx                               |     100 |      100 |     100 |     100 |
components/RegisterForm                       |     100 |      100 |     100 |     100 |
 RegisterForm.tsx                             |     100 |      100 |     100 |     100 |
components/Texts/LoginText                    |     100 |      100 |     100 |     100 |
 LoginText.tsx                                |     100 |      100 |     100 |     100 |
components/Texts/RegisterText                 |     100 |      100 |     100 |     100 |
 RegisterText.tsx                             |     100 |      100 |     100 |     100 |
components/Texts/TableItemText                |     100 |      100 |     100 |     100 |
 TableItemText.tsx                            |     100 |      100 |     100 |     100 |
components/TopHeaderBar                       |     100 |      100 |     100 |     100 |
 TopBar.tsx                                   |     100 |      100 |     100 |     100 |
components/TopHeaderBar/MobileTopBar          |     100 |      100 |     100 |     100 |
 MobileTopBar.tsx                             |     100 |      100 |     100 |     100 |
components/TopHeaderBar/MobileTopBar/state    |     100 |      100 |     100 |     100 |
 useMobileTopBar.tsx                          |     100 |      100 |     100 |     100 |
components/TopHeaderBar/WebTopBar             |     100 |      100 |     100 |     100 |
 WebTopBar.tsx                                |     100 |      100 |     100 |     100 |
-----------------------------------------------|---------|----------|---------|---------|

Test Suites: 49 passed, 49 total
Tests:       129 passed, 129 total
Snapshots:   0 total
Time:        26.049 s
Ran all test suites.
```

Figura 40. Pruebas unitarias del aplicativo web.

3.2 Pruebas de Integración.

En el presente trabajo se utilizaron una prueba bottom-up y dos pruebas top-down.

3.2.1 Prueba de integración 1

Tabla 42. Caso de prueba 1 – Registro e inicio de sesión.

Caso de prueba 1 – Registro e inicio de sesión.	
Objetivo:	Permitir al usuario normal registrarse e iniciar sesión con sus credenciales.
Tipo de prueba	Top-down
Precondiciones:	N//A
Tipo de usuario:	Usuario normal
Datos de prueba:	Usuario: Juan Zapata Correo electrónico: juan123@epn.edu.ec Contraseña: Juan123#
Procedimiento	1. Ingresar al URL localhost:3000. 2. Buscar y dar clic en el botón “Registrarse”. 3. Llenar los datos del formulario de registro. 4. Dar clic en el botón “registrarse”. 5. Buscar y dar clic en el botón “Iniciar Sesión”. 6. Ingresar datos de correo electrónico y contraseña. 7. Dar clic en “Ingresar al sistema”.
Resultado esperado:	El usuario podrá acceder a la pantalla principal del sistema.
Resultados obtenidos:	Prueba exitosa: Si: ✅ No:
Casos de excepción:	En los formularios de Registro e Inicio de Sesión se valida que los campos no se encuentren vacíos y presenta un mensaje de error con el mensaje “Formulario con errores” y en cada campo que tenga error se muestra el correspondiente mensaje de error.
Solución:	N/A

3.2.2 Prueba de integración 2

Tabla 43. Caso de prueba 2 – Vinculación de dispositivo IoT.

Caso de prueba 2 – Vinculación de dispositivo IoT.	
Objetivo:	Permitir al usuario vincular su dispositivo.
Tipo de prueba	Top-down
Precondiciones:	El usuario debe tener una sesión activa en el sistema.
Tipo de usuario:	Usuario normal
Datos de prueba:	Código de dispositivo: SDM-002
Procedimiento	1. Ingresar al URL localhost:3000/linkDevice. 2. Buscar e ingresar el código del dispositivo en el campo de texto “Ingrese Código”.

	3. Dar clic en el botón “Vincular”.
Resultado esperado:	El usuario podrá ver el mensaje “Id de dispositivo vinculado!”.
Resultados obtenidos:	Prueba exitosa: Si: ✅ No:
Casos de excepción:	El campo de texto “Ingrese Código” se valida que no tenga un valor vacío.
Solución:	N/A

3.2.3 Prueba de integración 3

Tabla 44. Caso de prueba 3 – Mostrar grafo de contactos recientes.

Caso de prueba 3 – Mostrar grafo de contactos recientes.	
Objetivo:	El sistema deberá mostrar un grafo de contactos recientes en la pantalla principal del sistema.
Tipo de prueba	Down-top
Precondiciones:	1. El usuario deberá tener una sesión activa en el sistema. 2. El usuario deberá tener contactos realizados con otros usuarios del sistema.
Tipo de usuario:	Usuario normal

Datos de prueba:	Contacto realizado:
	{
	"_id": {
	"$oid": "640aa09693e191a32bfdeb0c"
	},
	"idDevice": "SDM-005",
	"idContactDevice": "SDM-004",
	"rssi": -79,
	"distance": 1.06,
	"timestampInit": 1678418070440,
	"timestampEnd": 1678418083187,
	"__v": 0
	}

Procedimiento	1. El usuario con el dispositivo SDM-002 se acercará a menos de 1.50 metros al usuario con el dispositivo SDM-001. 2. Los usuarios se alejarán terminando el contacto. 3. El usuario SDM-002 Ingresar al URL localhost:3000/home.
Resultado esperado:	El usuario podrá ver su contacto realizado en el grafo de contactos.
Resultados obtenidos:	Prueba exitosa: Si: ✅ No:
Casos de excepción:	N/A
Solución:	N/A

3.3 Pruebas de usabilidad.

3.3.1 System Usability Scale (SUS)

La encuesta se realizó a 27 estudiantes dentro de las instalaciones de la Escuela Politécnica Nacional. Los estudiantes ejecutaron personalmente las 3 pruebas de integración descritas previamente y al finalizar el proceso se aplicó la encuesta, obteniendo así los siguientes resultados:

Pregunta 1 - ¿Usted usaría frecuentemente el sistema?

Para la primera pregunta se obtuvo que el 45% de los encuestados están fuertemente de acuerdo en que el sistema será usado con frecuencia, a estos se les suma el 33% que afirman sentirse de acuerdo. El 15% muestra un sentimiento de neutralidad y finalmente se tiene a un 7% que cree que no usaría el sistema frecuentemente, ver la Figura 41.

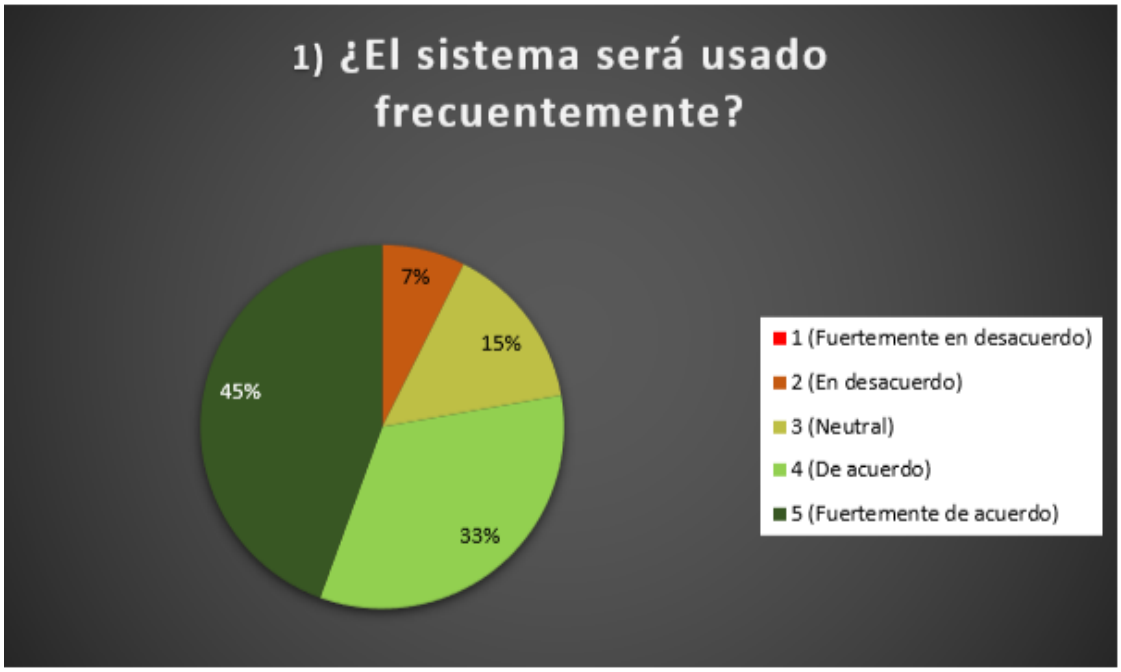


Figura 41. Resultados encuesta SUS pregunta 1.

Pregunta 2 – ¿El sistema es complejo de usar?

Para la segunda pregunta se tiene que el 37% de los encuestados se sienten fuertemente en desacuerdo en que el sistema es complejo de usar, a estos se suman el 26% que se encuentran de igual forma en desacuerdo. El 22% de los encuestados

mantienen una posición neutral y el 15% se siente de acuerdo en que el sistema es complejo de usar, ver Figura 42.

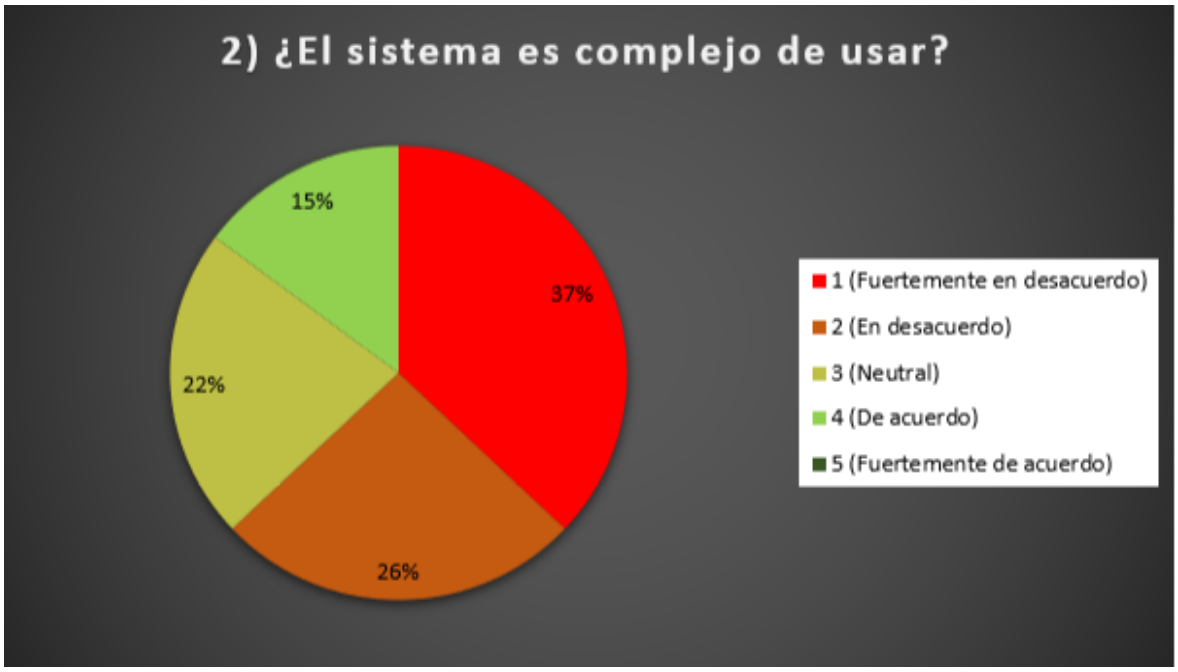


Figura 42. Resultados encuesta SUS pregunta 2.

Pregunta 3 - ¿El sistema es fácil de usar?

Para la tercera pregunta se tiene que el 63% de los encuestados está fuertemente de acuerdo con que el sistema es fácil de usar, a estos se suman el 30% que se siente de acuerdo. El restante 7% mantiene una posición neutral, ver Figura 43.

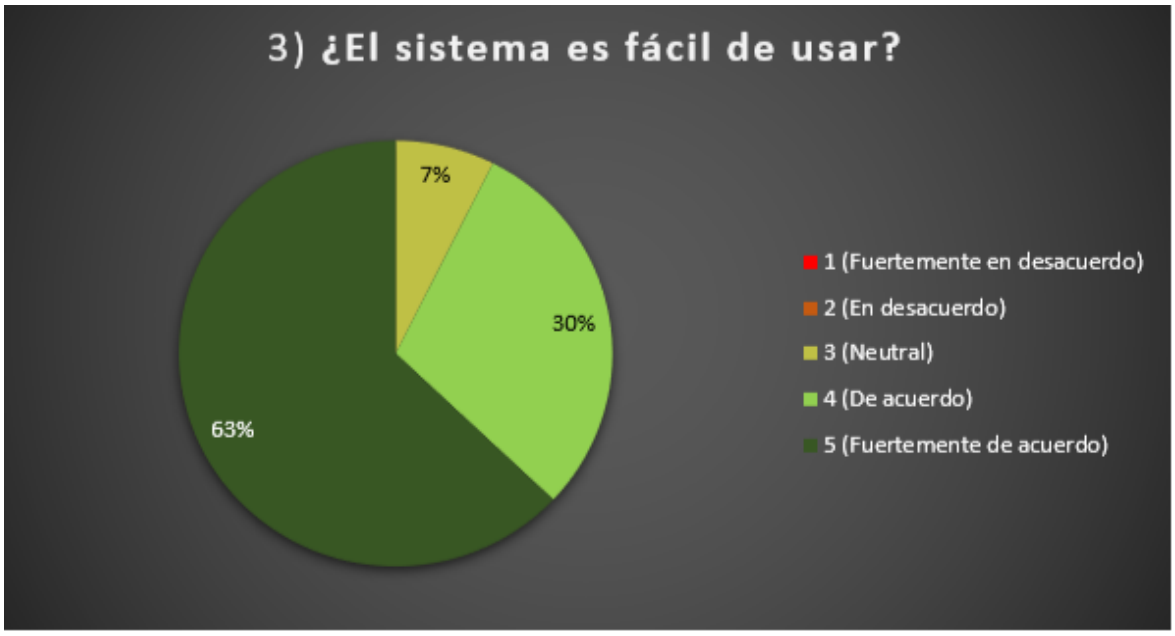


Figura 43. Resultados encuesta SUS pregunta 3.

Pregunta 4 - ¿Para hacer uso del sistema necesitó ayuda del personal técnico?

Para la cuarta pregunta se tiene que el 22% de los encuestados están fuertemente de acuerdo en que no se necesitó la ayuda del personal técnico para poder usar el sistema, a estos se suman el 37% que están de acuerdo en que no requieren la ayuda del personal técnico. El 26% mantiene una posición neutral y el 15% restante está de acuerdo en que se necesita la ayuda de un técnico para poder utilizar el sistema, ver Figura 44.

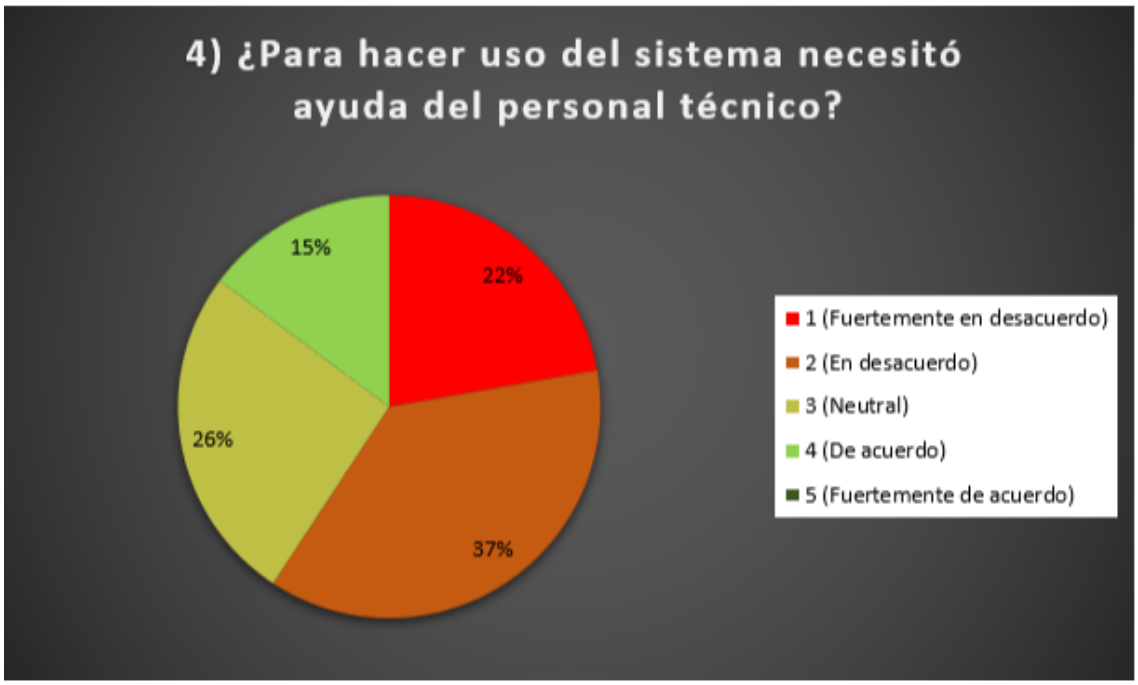


Figura 44. Resultados encuesta SUS pregunta 4.

Pregunta 5 - ¿Las funcionalidades del sistema están bien integradas?

Para la quinta pregunta se tiene que el 48% de los encuestados están fuertemente de acuerdo en que las funcionalidades están bien integradas, a estos se suman el 30% que están de acuerdo. El 22% restante de los encuestados mantienen una posición neutral, ver Figura 45.

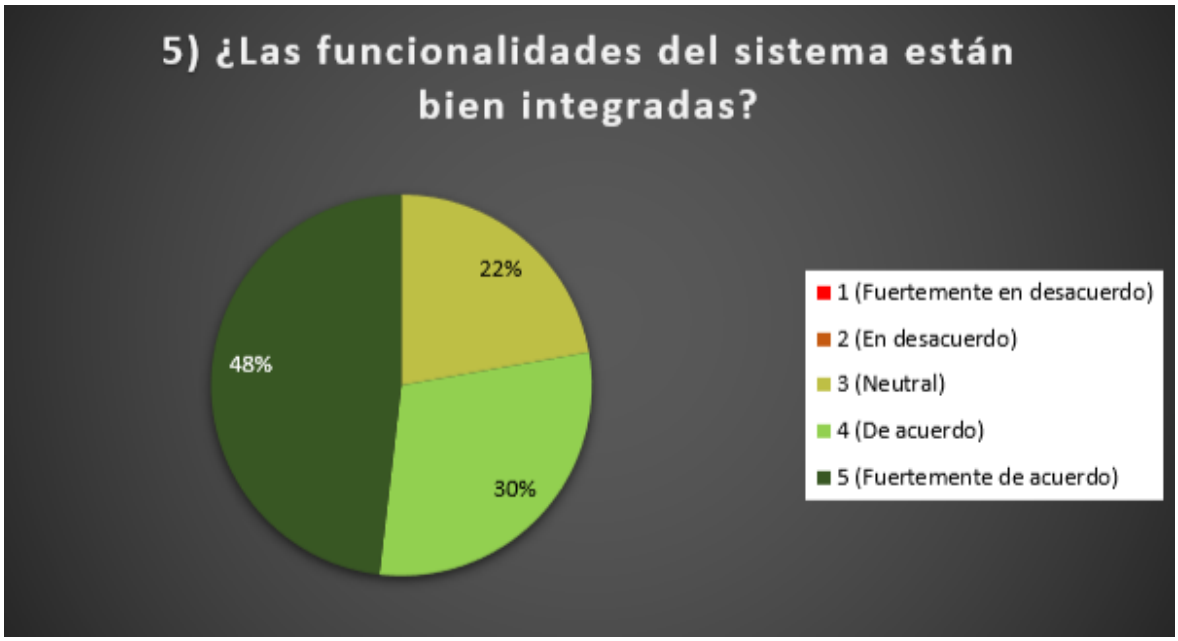


Figura 45. Resultados encuesta SUS pregunta 5.

Pregunta 6 - ¿El sistema tiene muchas inconsistencias?

Para la sexta pregunta se tiene que el 41% cree fuertemente que no hay muchas inconsistencias en el sistema, a estos se suman otro 41% de encuestados que están de acuerdo en que no hay muchas inconsistencias en el sistema. El 11% mantiene una postura neutral y el 7% restante está de acuerdo en que el sistema si tiene muchas inconsistencias.

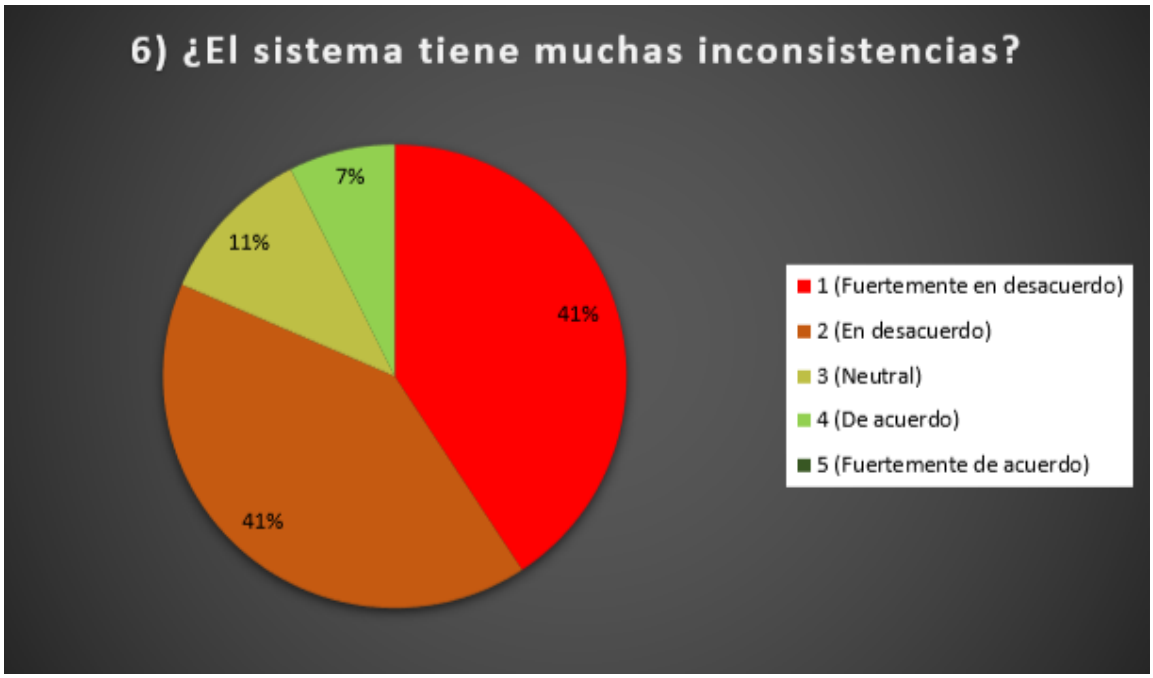


Figura 46. Resultados encuesta SUS pregunta 6.

Pregunta 7 - ¿Otros usuarios podrían aprender a usar el sistema de manera rápida?

Para la séptima pregunta se tiene que el 56% de los encuestado están fuertemente de acuerdo en que una gran cantidad de personas pueden aprender a usar el sistema de manera rápida, a estos se suman el 33% que se encuentran de acuerdo. El 7% mantiene una posición neutral y el 4% restante están desacuerdo, ver Figura 47.

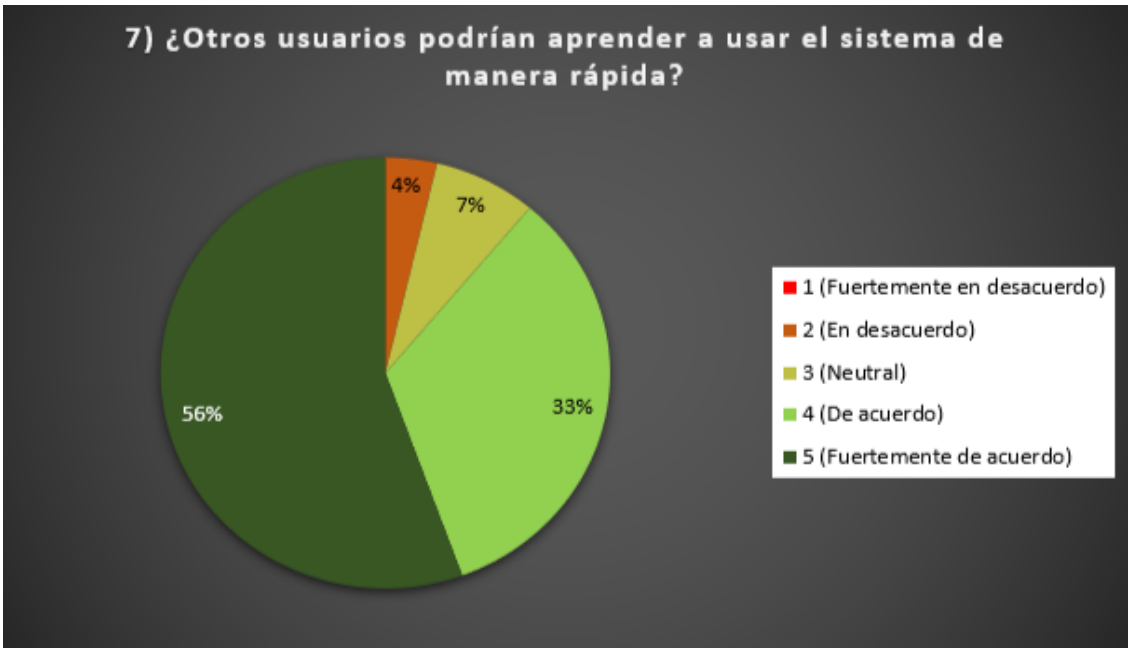


Figura 47. Resultados encuesta SUS pregunta 7.

Pregunta 8 - ¿El sistema es difícil de usar?

Para la octava pregunta se tiene que el 48% de las personas se encuentran fuertemente en desacuerdo con respecto a que el sistema es muy difícil de usar, a estos se suman el 33% que se sienten en desacuerdo. El 11% mantiene una posición neutral y el 8% restante cree fuertemente que el sistema si es muy extraño de usar, ver Figura 48.

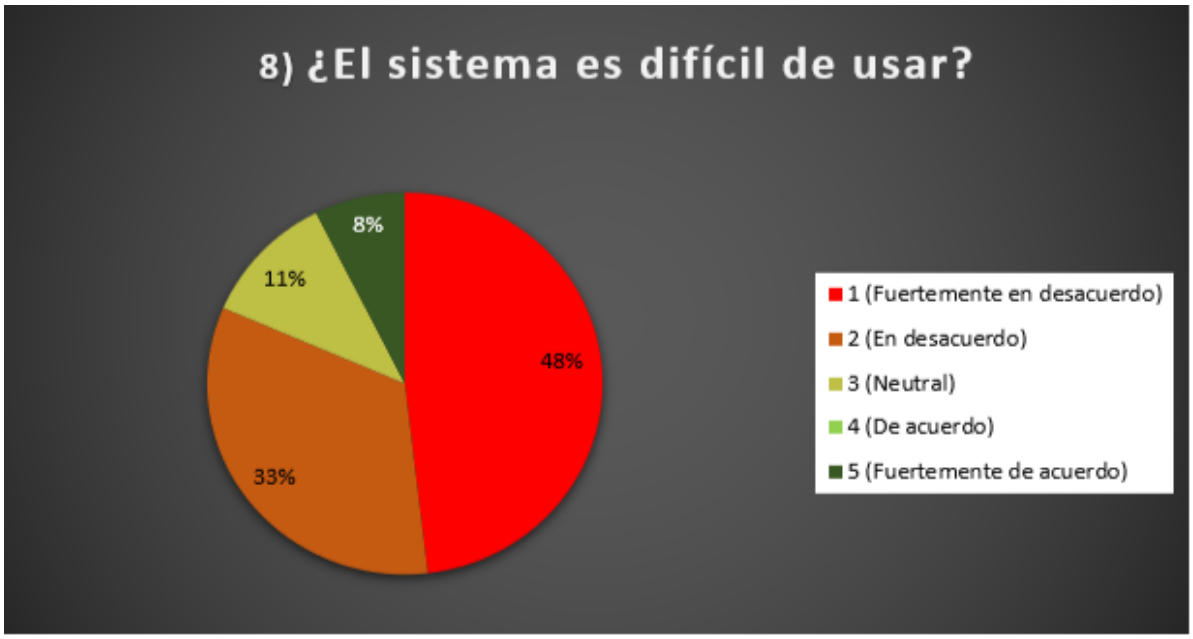


Figura 48. Resultados encuesta SUS pregunta 8.

Pregunta 9 - ¿La experiencia de uso del sistema fue satisfactoria?

Para la novena pregunta se tiene que el 41% de los encuestados están fuertemente de acuerdo en que la experiencia al usar el sistema es satisfactoria, a estos se suman el 33% que se siente de acuerdo. El 18% mantiene una posición neutral. El 4% se siente en desacuerdo y el otro 4% restante están fuertemente en desacuerdo, ver Figura 49.

Figura 49. Resultados encuesta SUS pregunta 9.

Pregunta 10 - ¿El sistema demanda de muchos conocimientos previos para hacer uso del mismo?

Para la pregunta final se tiene que el 45% de los encuestados están fuertemente de acuerdo en que no se necesita de muchos conocimientos previos para hacer uso del sistema, a estos se suman el 19% que está de acuerdo. El 22% se mantiene en una postura neutral. El 7% siente de acuerdo con respecto a que necesitan conocimientos previos antes de usar el sistema, a estos se suman el 7% restante que está fuertemente de acuerdo en que necesita obtener conocimientos previos para usar el sistema, ver Figura 50.

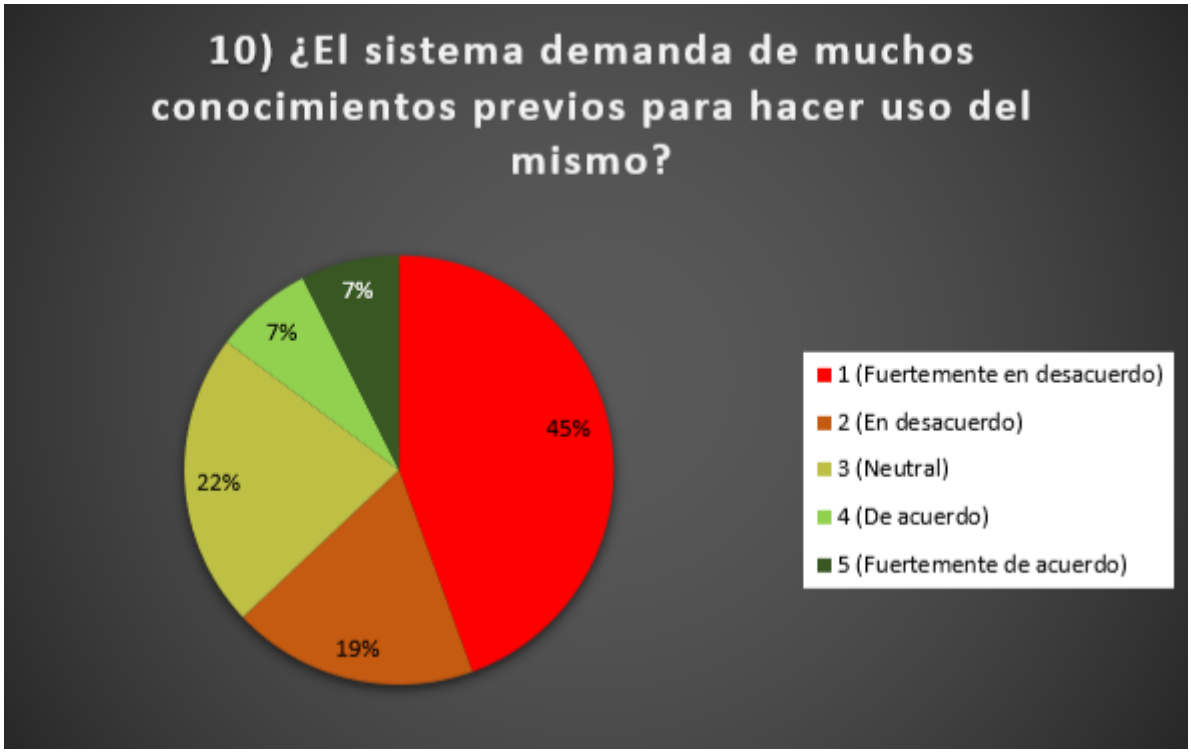


Figura 50. Resultados encuesta SUS pregunta 10.

Una vez terminadas las encuestas en la Figura 51, se puede apreciar un gráfico relacionando las preguntas de la encuesta con sus puntajes promedios obtenidos para cada una.

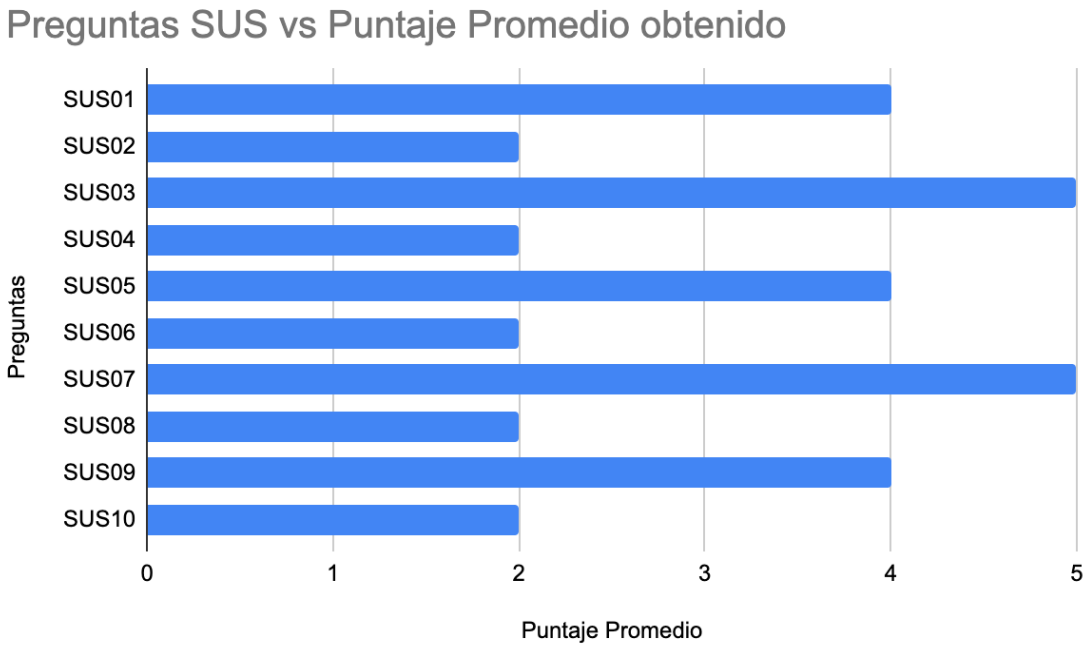


Figura 51. Gráfico Preguntas vs Puntaje promedio obtenido.

Una vez obtenidos los puntajes promedios se procede a realizar el cálculo del puntaje SUS para ello se aplica la Ecuación 3 descrita previamente y se obtiene un puntaje de 85 / 100.

De acuerdo con los resultados obtenidos se puede decir que:

- El puntaje SUS obtenido corresponde a una calificación A+
- Basado en la calificación se puede concluir que la usabilidad del sistema es Excelente

Las preguntas que se han obtenido en promedio el puntaje más alto corresponden a la pregunta SUS03 y SUS07 lo que da a entender que el sistema es de fácil uso y que las personas encuestadas pueden imaginar la mayoría de las personas pueden aprender a usar el sistema de manera sencilla. Por el contrario, se tienen a las preguntas SUS02, SUS04, SUS06, SUS08 y SUS10 que han obtenido en promedio 2 puntos lo que nos afirma que el sistema no es percibido como innecesariamente complejo, las personas no creen que necesiten la ayuda técnica para utilizar el sistema, el sistema no es percibido como inconsistente, el sistema no es extraño de usar y que las personas no necesitan aprender muchas cosas antes de usar el sistema. En términos generales el sistema ha pasado con éxito las pruebas de usabilidad.

4 CONCLUSIONES Y RECOMENDACIONES

4.1 Conclusiones

- La solución propuesta en este trabajo hace una contribución importante en contra de la transmisión de enfermedades con carácter contagioso y a favor de la salud pública, especialmente en ambientes laborales. La capacidad de los dispositivos IoT de medir y alertar sobre la proximidad entre usuarios y almacenar esta información dentro de una base de datos con características claves de este prototipo.
- La colecta de datos y la elaboración de gráficos de proximidad proporciona información valiosa para los administradores y trabajadores, permitiéndoles prevenir el riesgo de contagio y mejorar la seguridad en el lugar de trabajo. La alerta sonora de proximidad también es una herramienta efectiva para promover el distanciamiento social en los entornos de trabajo y reducir el contacto físico innecesario.
- Este tipo de soluciones tecnológicas son cada vez más necesarias ante un contexto de prevención de enfermedades y pueden hacer una contribución importante para asegurar la protección sanitaria de las personas y la continuidad de las actividades laborales.
- La arquitectura diseñada para esta solución de IoT se basó en una estructura de 4 capas debido a que la funcionalidad de cada capa permite cumplir con los requerimientos especificados en el prototipo, lo que facilitó el diseño, la implementación y facilitará el futuro mantenimiento de la solución propuesta. Dentro de la capa de percepción, se logró la conexión de los dispositivos IoT a través de bluetooth para medir el DS menor a 1,5 metros. El prototipado de los dispositivos IoT se realizó con la placa NodeMCU ESP32 que se utilizó como componente principal de la arquitectura hardware del dispositivo IoT. La capa de red permitió la transmisión entre los dispositivos y la base de datos mediante una conexión wifi. La capa de soporte permitió almacenar los datos en una base de datos MongoDB y el procesamiento de la información recibida por los dispositivos IoT utilizando un servidor Node js. Por último, la capa de aplicación permitió la observación y análisis de los datos recopilados en un portal web, el cual fue desarrollado usando la librería ReactJS y para los gráficos la librería D3-js.

- El componente MT3608 ha demostrado su capacidad para mantener un voltaje de 5V constante para la alimentación del dispositivo IoT, independientemente del voltaje de entrada de la batería. Esta característica permite mantener una potencia de emisión de señal BLE constante, lo que a su vez mejora la precisión a la hora del cálculo de distancia entre dispositivos.
- El dispositivo IoT se prototipó utilizando la placa NodeMCU ESP32 como componente principal de su arquitectura, aprovechando sus características de conexión wifi y BLE. El dispositivo cumple satisfactoriamente con su función de recolectar y transmitir datos precisos y confiables, gracias al diseño cuidadoso del hardware y a la implementación adecuada de su programación.
- Durante el diseño del prototipo, se identificaron inconsistencias al utilizar una única placa NodeMCU ESP32 para emitir y recibir señales BLE. Para solucionar esto, se utilizó una configuración de dos placas, donde una actúa como emisor de señal y la otra como receptor y procesador para así satisfacer los requerimientos de la solución prototipo.
- La elección de MongoDB como base de datos sin afinidad para esta solución IoT resultó ser una decisión acertada. MongoDB se adaptó sin ninguna complicación a la solución diseñada, proporcionando una gran flexibilidad y escalabilidad en la gestión de los datos. La estructura de documentos de MongoDB permitió una fácil integración de los datos agrupados por los dispositivos IoT, lo que simplificó significativamente el proceso de almacenamiento y gestión de estos. Además, MongoDB permitió una rápida consulta y análisis de los datos, lo que fue esencial para la visualización de estos a través del portal web diseñado.
- La implementación del framework Scrum en este proyecto de prototipado de IoT ha demostrado ser una elección acertada debido a su gran versatilidad y facilidad de uso. Scrum permitió un enfoque ágil en el diseño, desarrollo e implementación del prototipo, lo que permitió adecuarse rápidamente a los cambios y desafíos generados durante el proyecto. La flexibilidad de Scrum al momento realizar la estimación de historias de usuario permitió otorgar holguras de tiempo de desarrollo para realizar tareas en las cuales se implementaron tecnologías con poco o nulo conocimiento del equipo de desarrollo. Estas holguras de tiempo fueron aprovechadas por el equipo de desarrollo para aprender sobre la marcha y entregar el incremento en las fechas acordadas. Como resultado, el prototipo IoT que calcula las distancias entre dispositivos a través del uso del RSSI se diseñó e implementó de tal forma que cumple con todos los requerimientos planteados.

- La encuesta SUS realizada para evaluar la usabilidad del prototipo de solución IoT diseñado arrojó una calificación A+, lo que indica que el sistema ha sido aceptado favorablemente por los usuarios y que tiene un alto potencial para su implementación en las instalaciones de la EPN. Esta calificación también demuestra que el prototipo cumple con los requisitos de usabilidad y satisfacción del usuario, lo que reduce el riesgo de realizar mejoras adicionales antes de su implementación en un ambiente de producción.

4.2 Recomendaciones

- Si bien el uso de dos placas NodeMCU-ESP32 dentro de un mismo dispositivo cumplió los requerimientos del prototipo, con la finalidad de minimizar costos en la fabricación del dispositivo IoT, se puede optimizar su diseño utilizando una sola placa de desarrollo. Para lograr esto, se recomienda implementar el uso de hilos para crear dos tareas que se ejecuten simultáneamente, una para emitir y otra para recibir y procesar la señal. Esta mejora permitirá reducir los costos de producción y simplificar el diseño del dispositivo IoT.
- Aunque el uso del RSSI para la medición de distancias entre dispositivos ha demostrado ser eficaz, se puede mejorar aún más mediante la implementación de un algoritmo de disminución de ruido en la detección del RSSI. El uso de un algoritmo de disminución de ruido puede mejorar enormemente la precisión en el cálculo de distancias entre dispositivos.
- Al analizar los resultados de la encuesta SUS, se ha identificado que los usuarios han expresado la necesidad de contar con apoyo técnico para utilizar el sistema IoT. En este sentido, se sugiere implementar una estrategia de capacitación que incluya la creación de manuales de usuario y videos tutoriales que expliquen de manera clara y ordenada las funcionalidades de la solución. De esta forma, se podrá optimizar la experiencia de los usuarios e incrementar el puntaje SUS obtenido.
- Para implementar eficazmente en un ambiente de producción este prototipo, es importante considerar la implementación de un hosting y la utilización de un dominio web propio para manejar las peticiones HTTP mediante nombres de dominio, lo que facilitará la administración de los dispositivos IoT. Además, se recomienda implementar el protocolo HTTPS para mejorar la seguridad del sistema y proteger la información transmitida.

5 REFERENCIAS BIBLIOGRÁFICAS

[1] G. Basalla, La evolución de la tecnología, Barcelona, España: Drakontos, 2011, pp. 18.

[2] G. Huguet. (2020, marzo 25). Grandes pandemias de la historia. [Online]. Available: https://historia.nationalgeographic.com.es/a/grandes-pandemias-historia_15178/4

[3] Thebaselab. (2020) COVID-19 Coronavirus Dashboard by thebaselab. [Online]. Available: https://coronavirus.thebaselab.com [Último acceso: 16 March 2023].

[4] OMS. (2020). Modes of transmission of virus causing COVID-19: implications for IPC precaution recommendations. [Online]. Available: https://www.who.int/news-room/commentaries/detail/modes-of-transmission-of-virus-causing-covid-19-implications-for-ipc-precaution-recommendations. [Último acceso: 16 March 2023].

[5] OMS. (2020). Preguntas y respuestas sobre la enfermedad por coronavirus (COVID-19). [Online]. Available: https://www.who.int/es/emergencies/diseases/novel-coronavirus-2019/advice-for-public/q-a-coronaviruses?gclid=CjwKCAjwmf_4BRABEiwAGhDfSfJOVNLTiZA82kNr-CANsVqMhk3yeiRTfiiRr9APnrfS6nclXxc37xoCwk0QAvD_BwE [Último acceso: 16 March 2023].

[6] DW. (2020, marzo 15). Coronavirus, minuto a minuto: Más países decretan cuarentena en América. [Online]. Available: https://www.dw.com/es/coronavirus-minuto-a-minuto-más-países-decretan-cuarentena-en-américa-16032020/a-52788178 [Último acceso: 16 March 2023].

[7] M. Kashyap, V. Sharma, N. Gupta. “Taking MQTT and NodeMcu to IOT: Comunication in Internet of Things”. Procedia Computer Science. vol 132. pp 1611-1618.

[8] A. Haque, W. Karim, SMH Kabir, and A.K. Tarofder, "Understanding Social Distancing Intention among University Students during Covid-19 Outbreak: An Application of Protection Motivation Theory," in IEEE Xplore Digital Library, vol. 83, pp. 16360-16377, May-June 2020.

[9] A. Ksentini and B. Brik, "An Edge-Based Social Distancing Detection Service to Mitigate COVID-19 Propagation," in IEEE Internet of Things Magazine, vol. 3, no. 3, pp. 35-39, Sep. 2020, doi: 10.1109/IOTM.0001.2000138.

[10] F. Ahmed, N. Zviedrite, and A. Uzicanin, "Effectiveness of Workplace Social Distancing Measures in Reducing Influenza Transmission: A Systematic Review," BMC Public Health, vol. 18, p. 518, Apr. 2018, doi: 10.1186/s12889-018-5446-1.

[11] C. Paun et al., "How Europe Is Responding to the Coronavirus Pandemic," Politico, Mar. 13, 2020. [Online]. Available: https://www.politico.eu/article/how-europe-is-responding-to-the-coronavirus-pandemic/. [Último acceso:16 March 2023].

[12] ESPRESSIF. (2019). ESP32 Series Datasheet. [Online]. Available: https://pdf1.alldatasheet.com/datasheet-pdf/view/1148023/ESPRESSIF/ESP32.html [Último acceso: 16 March 2023].

[13] Intel. (2020). Cómo borrar el CMOS para restablecer la configuración del BIOS en sistemas con procesadores Intel® para desktop en caja. [Online]. Available: https://www.intel.la/content/www/xl/es/support/articles/000025368/processors.html [Último acceso: 16 March 2023].

[14] Waveshare. (2023). NodeMCU-32S, ESP32 WiFi+Bluetooth Development Board. [Online]. Available: https://www.waveshare.com/nodemcu-32s.htm. [Último acceso: 16 March 2023].

[15] Amazon. (2023). ESP32 Pinout. [Online]. Available: https://m.media-amazon.com/images/I/61afDc5iOYL._AC_SL1001_.jpg. [Último acceso: 16 March 2023].

[16] S. Kraijak and P. Tuwanut, "A survey on IoT architectures, protocols, applications, security, privacy, real-world implementation and future trends," in Proceedings of the 11th International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM 2015), 2015, pp. 1-6, doi: 10.1049/cp.2015.0714.

[17] M. Burhan, R. Rehman, B. Khan, and B.-S. Kim, "IoT Elements, Layered Architectures and Security Issues: A Comprehensive Survey," Sensors, vol. 18, no. 9, p. 2796, Aug. 2018, doi: 10.3390/s18092796.

[18] M. Al-Bahri, K. Ruslan, and B. Aleksey, "Integrating Internet of Things with the Digital Object Architecture," in Internet of Things, Smart Spaces, and Next Generation Networks and Systems, O. Galinina, S. Andreev, S. Balandin, and Y. Koucheryavy, Eds. Springer, Cham, 2019, vol. 11660, pp. 563-574, doi: 10.1007/978-3-030-30859-9_47.

[19] S. M. Darroudi and C. Gomez, "Bluetooth Low Energy Mesh Networks: A Survey," Sensors, vol. 17, no. 7, pp. 1467-1488, Jul. 2017, doi: 10.3390/s17071467.

[20] C. Gomez, J. Oller and J. Paradells, "Overview and Evaluation of Bluetooth Low Energy: An Emerging Low-Power Wireless Technology," in Sensors, vol. 12, pp. 11734-11753, 2012, doi: 10.3390/s120911734.

[21] Lee, B.-G.; Do, K.-H.; Chung, W.-Y. WSN Based 3D Mobile Indoor Multiple User Tracking. In Proceedings of the 2009 IEEE Sensors, Christchurch, New Zealand, 25–28 October 2009; pp. 1598–1603. doi: 10.1109/ICSENS.2009.5398494.

[22] E. Lau, B. G. Lee, S.-C. Lee, and W.-Y. Chung, "Enhanced Rssi-Based High Accuracy Real-Time User Location Tracking System For Indoor And Outdoor Environments," International Journal on Smart Sensing and Intelligent systems, vol. 1, Jan. 2008, doi: 10.21307/ijssis-2017-306.

[23] R. Mehra and A. Singh, "2013 3rd IEEE International Advance Computing Conference (IACC) - Real time RSSI error reduction in distance estimation using RLS algorithm," in IEEE, Ghaziabad, 2013, pp. 661-665, doi: 10.1109/IAdCC.2013.6514305.

[24] MDN. (2023). JavaScript. [Online]. Available: https://developer.mozilla.org/es/docs/Web/JavaScript. [Último acceso: 16 March 2023].

[25] TypeScript. (2023). TypeScript is JavaScript with syntax for types. [Online]. Available: https://www.typescriptlang.org [Último acceso: 16 March 2023].

[26] B. Stroustrup, El lenguaje de programación C++, 3rd ed. Madrid: Addison-Wesley Iberoamericana España, 1998, pp. XII, 940.

[27] W3C. (2023). CSS Snapshot 2023. [Online]. Available: https://www.w3.org/TR/CSS/ [Último acceso: 16 March 2023].

[28] Whatwg. (2023). HTML Living Standard. [Online]. Available: https://html.spec.whatwg.org/multipage/ [Último acceso: 16 March 2023].

[29] React. (2023). React .- Getting Started. [Online]. Available: https://legacy.reactjs.org/docs/getting-started.html [Último acceso: 16 March 2023].

[30] Vercel. (2023). Next.js on Vercel. [Online]. Available: https://vercel.com/docs/frameworks/nextjs [Último acceso: 16 March 2023].

[31] NestJS. (2023). Introduction. [Online]. Available: https://docs.nestjs.com [Último acceso: 16 March 2023].

[32] Github. (2018). Bluetooth BLE. [Online]. Available: https://github.com/nkolban/esp32-snippets/blob/master/Documentation/BLE%20C%2B%2B%20Guide.pdf [Último acceso: 16 March 2023].

[33] Jest. (2023). Jest is a delightful JavaScript Testing Framework with a focus on simplicity. [Online]. Available: https://jestjs.io [Último acceso: 16 March 2023]

[34] D3. (2023). Data Driven Documents. [Online]. Available: https://d3js.org [Último acceso: 16 March 2023]

[35] ReactRedux. (2023). Getting Started with React Redux. [Online]. Available: https://react-redux.js.org/introduction/getting-started [Último acceso: 16 March 2023].

[36] Visual Studio Code. (2023). Getting Started. [Online]. Available: https://code.visualstudio.com/docs [Último acceso: 16 March 2023]

[37] Arduino. (2023). Overview of the Arduino IDE 1. [Online]. Available: https://docs.arduino.cc/software/ide-v1/tutorials/Environment [Último acceso: 16 March 2023]

[38] MongoDB. (2023). What is MongoDB. [Online]. Available: https://www.mongodb.com/docs/manual/?_ga=2.50547434.2107720433.1669861358-614172622.1669861358 [Último acceso: 16 March 2023]

[39] Postman. (2023). What is Postman. [Online]. Available: https://www.postman.com [Último acceso: 16 March 2023].

[40] A. Srivastava, S. Bhardwaj, and S. Saraswat, "SCRUM model for agile methodology," in 2017 International Conference on Computing, Communication and Automation (ICCCA), 2017, pp. 864-869, doi: 10.1109/CCAA.2017.8229928.

[41] Ken Schwaber y Jeff Sutherland, "La Guía de Scrum: La Guía Definitiva de Scrum: Las Reglas del Juego," Noviembre de 2017, desarrollado y soportado por los Creadores de Scrum, Español / Spanish South American.

[42] K. Schwaber and M. Beedle, "Agile Software Development with Scrum, 1st edition," Upper Saddle River, New Jersey, Prentice Hall, 2002, ISBN: 0130676349.

[43] S. K. Singh y A. Singh, "Software Testing," Vandana Publications, Lucknow, India, 2019, ISBN: 978-81-941110-6-1.

[44] P. Runeson, "A survey of unit testing practices," IEEE Software, vol. 23, no. 4, pp. 22-29, 2006, doi: 10.1109/MS.2006.91.

[45] H.K.N. Leung y L. White, "[IEEE Comput. Soc. Press Conference on Software Maintenance 1990 - San Diego, CA, USA (26-29 Nov. 1990)] Proceedings. Conference on Software Maintenance 1990 - A study of integration testing and software regression at the integration level," pp. 290-301, 1990, doi: 10.1109/icsm.1990.131377.

[46] A.M. Wichansky, "Usability testing in 2000 and beyond," Ergonomics, vol. 43, no. 7, pp. 998-1006, 2000, doi: 10.1080/001401300409170.

[47] Lewis, James R. (2018). The System Usability Scale: Past, Present, and Future. International Journal of Human-Computer Interaction. pp 1-14. doi:10.1080/10447318.2018.1455307

[48] B. Klug, "An Overview of the System Usability Scale in Library Website and System Usability Testing," Gibson D. Lewis Health Science Library, University of North Texas Health Science Center, vol. 1, no. 6, pp. 1-4, 2017, doi: 10.3998/weave.12535642.0001.602.

[49] NanJing Top Power ASIC Corp. (2019). 1A Standalone Linear Li-Ion Battery Charger with Thermal Regulation in SOP-8. [Online]. Available:

https://dlnmh9ip6v2uc.cloudfront.net/datasheets/Prototyping/TP4056.pdf [Último acceso: 16 March 2023].

[50] AEROSEMI. (2020). MT3608 - High Efficiency 1.2MHz 2ª Step Up Converter. [Online]. Available: https://pdf1.alldatasheet.com/datasheet-pdf/view/1131968/ETC1/MT3608.html [Último acceso: 16 March 2023].

6 ANEXOS

6.1 **Anexo 1:** Historias de usuario Sprint 0

Tabla 45. Historia de usuario PDS1-03.

Historia de usuario		
ID: PDS1-03		**Puntos estimados: 5**
Definir de la arquitectura IoT a ser usada		
Sprint N°: 1	**Prioridad:** Alta	**Estado:** Por Hacer
Descripción: Como usuario administrador, requiero tener un diagrama de la arquitectura de comunicación del sistema para poder consultar de forma breve los módulos del sistema.		
Criterios de Aceptación		
Dado	**Cuando**	**Entonces**
Que el usuario administrador desea visualizar la arquitectura de comunicación del sistema.	Desee conocer que framework ha sido utilizado para construir las diferentes capas del sistema.	Se podrá visualizar claramente el framework usado para construir las capas del sistema.
Que el usuario administrador desea conocer el flujo de datos dentro del sistema.	Desee consultar los endpoints encargados del tratamiento de los datos.	Podrá visualizar un diagrama que muestra los endpoints encargados de la entrada y salida de los datos.

Tabla 46. Historia de usuario PDS2-01.

Historia de usuario		
ID: PDS-01		**Puntos estimados: 5**
Diseñar un algoritmo para controlar y enviar la información de los dispositivos IoT al servidor back-end.		
Sprint N°: 1	**Prioridad:** Alta	**Estado:** Por Hacer

Descripción: Como usuario administrador del sistema, requiero tener un diagrama del algoritmo a ser implementado en cada dispositivo IoT para adjuntarlo como documentación del sistema.		
Criterios de Aceptación		
Dado	**Cuando**	**Entonces**
Que el usuario administrador desea visualizar el algoritmo de gestión de contactos de los dispositivos IoT.	Desee visualizar el funcionamiento del dispositivo IoT	Se podrá visualizar un diagrama de flujo del algoritmo de gestión de contactos

6.2 **Anexo 2:** Historias de usuario Sprint 1

Tabla 47. Historia de usuario PDS1-01.

Historia de usuario		
ID: PDS1-01		**Puntos estimados:** 8
Título: Realizar el diagrama de dispositivo IoT.		
Sprint N°: 0	**Prioridad:** Alta	**Estado:** Por Hacer
Descripción: Como administrador de la red, deseo tener un diagrama del dispositivo IoT que será usado para medir la distancia de las personas.		
Criterios de Aceptación		
Dado	**Cuando**	**Entonces**
Que el administrador desea conocer como está constituido el dispositivo IoT	Desee consultar los componentes con los cuales está constituido	Podrá revisar una tabla que detalla cada uno de los componentes que han sido usados en la construcción del dispositivo IoT
Que el administrador desea conocer cómo se interconectan los componentes del dispositivo	Desee consultar las conexiones del dispositivo	Podrá visualizar un diagrama de conexión de componentes del dispositivo.
Que el administrador desea conocer el coste estimado del dispositivo IoT	Desee consultar el precio por componente y el precio final del dispositivo IoT	Podrá visualizar una tabla de costes de cada componente.

Tabla 48. Historia de usuario PDS1-02.

Historia de usuario		
ID: PDS1-02		**Puntos estimados:** 5
Título: Soldar los componentes físicos del dispositivo IoT		
Sprint N°: 0	**Prioridad:** Alta	**Estado:** Por Hacer
Descripción: Como cliente de la red, deseo tener el dispositivo soldado y		

	compactado al tamaño de un carné o gafete para poder ser colgado al cuello.	
Criterios de Aceptación		
Dado	**Cuando**	**Entonces**
Que el cliente desea tener el dispositivo IoT ensamblado.	El cliente tenga el dispositivo en sus manos.	Podrá visualizar que el dispositivo tiene sus componentes correctamente soldados.
Que el cliente desea que el dispositivo sea del tamaño de un carné o gafete	El cliente tenga el dispositivo en sus manos.	Podrá visualizar que el tamaño del dispositivo es el de un carné o gafete, en altura y anchura.

6.3 **Anexo 3:** Historias de usuario Sprint 2

Tabla 49. Historia de usuario PDS2-02.

Historia de usuario		
ID: PDS2-02		**Puntos estimados: 5**
Programar el algoritmo y desplegarlo en los dispositivos IoT.		
Sprint N°: 2	**Prioridad:** Alta	**Estado:** Por Hacer
Descripción: Como usuario del sistema IoT, requiero que el dispositivo IoT pueda ser programado con el algoritmo para controlar el distanciamiento social, para poder garantizar el cumplimiento del distanciamiento social.		
Criterios de Aceptación		
Dado	**Cuando**	**Entonces**
Que el usuario se encuentra usando su dispositivo IoT	Este se acerque a menos de 1,5 m de otro usuario que este usando otro dispositivo	El dispositivo emitirá un sonido de alerta.
Que el usuario se encuentra usando su dispositivo IoT	Este se acerque a menos de 1,5 m de otro usuario que este usando otro dispositivo	El dispositivo enviará una petición HTTP de tipo POST al endpoint /contacts/init
Que el usuario se encuentra usando su dispositivo IoT	Este se aleje a más de 1,5 m de otro usuario.	El dispositivo debe dejar de emitir la alerta sonora y deberá enviar una petición tipo POST al endpoint /contacts/init

Tabla 50. Historia de usuario PDS2-03.

Historia de usuario		
ID: PDS2-03		**Puntos estimados: 3**
Programar la emisión del nombre del dispositivo vía Bluetooth.		
Sprint N°: 2	**Prioridad:** Alta	**Estado:** Por Hacer
Descripción: Como usuario del sistema IoT, quiero que el dispositivo pueda emitir el nombre del dispositivo a través del BLE, para poder identificar de manera automática y rápida al dispositivo entre varios dispositivos		

cercanos.		
Criterios de Aceptación		
Dado	**Cuando**	**Entonces**
Que el usuario se encuentra usando su dispositivo IoT	Este lo tenga encendido	Se podrá visualizar que el nombre del dispositivo está siendo emitido para ser leído por todos los dispositivos cercanos.

6.4 **Anexo 4:** Historias de usuario Sprint 3

Tabla 51. Historia de usuario PDS3-01.

Historia de usuario		
ID: PDS3-01		**Puntos estimados: 5**
Creación de proyecto base usando Nextjs		
Sprint N°: 3	**Prioridad:** Alta	**Estado:** Por Hacer
Descripción: Como usuario del sistema IoT, quiero que el portal web sea programado usando Nextjs, para poder tener un portal web creado con tecnologías recientes, escalable y mantenible.		
Criterios de Aceptación		
Dado	**Cuando**	**Entonces**
Que el usuario ingresa al portal web	Este en la página de inicio.	Se podrá visualizar la pantalla inicial del Proyecto en Nextjs.

Tabla 52. Historia de usuario PDS4-01.

Historia de usuario		
ID: PDS4-01		**Puntos estimados: 5**
Creación del proyecto base usando Nest js		
Sprint N°: 3	**Prioridad:** Alta	**Estado:** Por Hacer
Descripción: Como desarrollador, quiero crear un proyecto base utilizando Nest js para tener una estructura organizada, facilitar el proceso de desarrollo y mantener una buena calidad del código.		
Criterios de Aceptación		
Dado	**Cuando**	**Entonces**
Que el usuario desea visualizar el servidor encendido	Se dirija a la ruta inicial del proyecto	Se podrá visualizar una petición HTTP de tipo GET exitosa.

Tabla 53. Historia de usuario PDS3-04

Historia de usuario		
ID: PDS3-04		**Puntos estimados: 3**
Creación de infogramas para evitar contagios de C-19		
Sprint N°: 3	**Prioridad:** Alta	**Estado:** Por Hacer

Descripción: Como usuario del sistema IoT, quiero tener una sección de información sobre medidas y buenas prácticas para evitar el contagio de C-19.		
Criterios de Aceptación		
Dado	**Cuando**	**Entonces**
Que el usuario desee conocer las medidas de seguridad para evitar contagios de C-19	Tenga una sesión activa en el portal web y navegue al apartado de	Se podrá visualizar infogramas sobre medidas para evitar el C-19

6.5 **Anexo 5:** Historias de usuario Sprint 4

Tabla 54. Historia de usuario PDS3-02

Historia de usuario		
ID: PDS3-02		**Puntos estimados:** 8
Creación de pantalla de Inicio de Sesión y registro de usuario.		
Sprint N°: 4	**Prioridad:** Alta	**Estado:** Por Hacer
Descripción: Como usuario del sistema IoT, quiero tener una pantalla de registro e inicio de sesión de usuario, para permitir a los usuarios del portal web acceder a los servicios disponibles mediante un proceso seguro y fácil..		
Criterios de Aceptación		
Dado	**Cuando**	**Entonces**
Que el usuario ingresa a la pantalla de registro e inicio de sesión	El usuario intente registrarse en el sistema	Se desplegará el formulario de registro.
Que el usuario ingresa a la pantalla de registro e inicio de sesión	El usuario complete el formulario de registro	El usuario podrá ingresar al sistema con su usuario y contraseña proporcionados.
Que el usuario ingresa a la pantalla de registro e inicio de sesión	El usuario coloca sus credenciales y da click en iniciar sesión	El usuario puede ingresar al portal web y ver su información.

Tabla 55. Historia de usuario PDS4-02

Historia de usuario		
ID: PDS4-02		**Puntos estimados: 5**
Crear endpoints para manejar el registro e inicio de sesión de usuario.		
Sprint N°: 4	**Prioridad:** Alta	**Estado:** Por Hacer
Descripción: Como desarrollador, quiero crear endpoints para manejar el registro e inicio de sesión de usuario en la aplicación. Para permitir a los usuarios crear una cuenta y acceder a ella de forma segura.		
Criterios de Aceptación		
Dado	**Cuando**	**Entonces**

Que el usuario desea crear un usuario nuevo	Este envíe su información al servidor	Se podrá visualizar la información registrada exitosamente en la base de datos.
Que el usuario desea iniciar sesión dentro del sistema	Este envíe la información de sus credenciales	Se podrá visualizar que el servidor proporciona un Json Web Token para gestionar el inicio de sesión en el sistema.

6.6 **Anexo 6:** Historias de usuario Sprint 5

Tabla 56. Historia de usuario PDS3-03.

Historia de usuario		
ID: PDS3-03		**Puntos estimados: 5**
Creación de pantalla para vincular y desvincular dispositivo		
Sprint N°: 5	**Prioridad:** Alta	**Estado:** Por Hacer
Descripción: Como usuario del sistema IoT, quiero poder vincular y desvincular el dispositivo en mi cuenta, para tener la opción de cambiar de dispositivo cuando lo requiera.		
Criterios de Aceptación		
Dado	**Cuando**	**Entonces**
Que el usuario desea vincular su dispositivo IoT	Se encuentre en la pantalla de vincular/desvincular dispositivo	Se podrá vincular el dispositivo mediante el nombre del dispositivo.
Que el usuario desea vincular su dispositivo IoT	Se encuentre en la pantalla de vincular/desvincular dispositivo	Se podrá desvincular el dispositivo mediante un botón.

Tabla 57. Historia de usuario PDS4-03.

Historia de usuario		
ID: PDS4-03		**Puntos estimados: 3**
Crear endpoints para guardar los contactos registrados por el dispositivo IoT.		
Sprint N°: 5	**Prioridad:** Alta	**Estado:** Por Hacer
Descripción: Como usuario, quiero poder guardar los contactos registrados en mi dispositivo IoT en el sistema. Para poder tener acceso a ellos desde el portal web.		
Criterios de Aceptación		
Dado	**Cuando**	**Entonces**

Que el usuario se encuentra usando su dispositivo IoT	Se encuentre a una distancia inferior a 1,5 metros de otro usuario	Se podrá visualizar el contacto registrado exitosamente en la base de datos.

Tabla 58. Historia de usuario PDS4-06.

Historia de usuario		
ID: PDS4-06		**Puntos estimados: 3**
Crear endpoint para vincular o desvincular un dispositivo a una cuenta de usuario.		
Sprint N°: 5	**Prioridad:** Alta	**Estado:** Por Hacer
Descripción: Como usuario, quiero poder vincular o desvincular un dispositivo IoT a mi cuenta de usuario en el sistema. Para poder utilizar el dispositivo con mi cuenta y en caso de ser necesario cambiarme a otro dispositivo.		
Criterios de Aceptación		
Dado	**Cuando**	**Entonces**
Que el usuario desea vincular su dispositivo IoT	Este esté en la pantalla de vincular dispositivo	Se podrá vincular su dispositivo usando el código único para cada dispositivo.
Que el usuario desea desvincular su dispositivo	Este esté en la pantalla de vincular dispositivo	Se podrá vincular su dispositivo de esa cuenta de usuario.

6.7 **Anexo 7:** Historias de usuario Sprint 6

Tabla 59. Historia de usuario PDS3-05.

Historia de usuario		
ID: PDS3-05		**Puntos estimados:** 8
Creación de pantalla principal y tabla de históricos de contactos		
Sprint N°: 6	**Prioridad:** Alta	**Estado:** Por Hacer
Descripción: Como usuario del sistema IoT, quiero visualizar mis contactos en la pantalla principal y ver mi historial de contactos, para tener conocimiento sobre los contactos que he tenido las ultimas 2 semanas.		
Criterios de Aceptación		
Dado	**Cuando**	**Entonces**
Que el usuario desea conocer sus contactos.	Se encuentre en la pantalla principal del portal.	Se podrá visualizar un gráfico de nodos mostrando los contactos registrados.
Que el usuario desea conocer sus contactos.	Se encuentre en la pantalla de históricos.	Se podrá visualizar una tabla con los contactos de hasta hace 2 semanas.

Tabla 60. Historia de usuario PDS4-04.

Historia de usuario		
ID: PDS4-04		**Puntos estimados: 5**
Crear endpoints para obtener la información del grafo de contactos y la tabla de históricos.		
Sprint N°: 6	**Prioridad:** Alta	**Estado:** Por Hacer
Descripción: Como usuario, quiero poder obtener la información de mis contactos en forma de grafo y la tabla de históricos de las personas con las que he tenido contacto. Para poder tener una visión general de mis relaciones y la actividad de contacto de manera fácil y accesible.		
Criterios de Aceptación		
Dado	**Cuando**	**Entonces**

Que el usuario desea visualizar el servidor encendido	Se dirija a la ruta inicial del proyecto	Se podrá visualizar una petición HTTP de tipo GET exitosa.

6.8 **Anexo 8:** Historias de usuario Sprint 7

Tabla 61. Historia de usuario PDS3-06.

Historia de usuario		
ID: PDS3-06		**Puntos estimados: 3**
Botón reportarse enfermo/sano		
Sprint N°: 7	**Prioridad:** Alta	**Estado:** Por Hacer
Descripción: Como usuario del sistema IoT, quiero poder reportarme enfermo dentro de la aplicación cuando obtenga un positivo en alguna prueba de C-19, para. poder alertar a las personas que tuvieron contacto cercano en las últimas 2 semanas.		
Criterios de Aceptación		
Dado	**Cuando**	**Entonces**
Que el usuario desea reportarse enfermo	Se encuentre en la pantalla principal del portal.	Se podrá visualizar un botón para poder reportase enfermo y al darle clic alertará a los usuarios que tuvieron un contacto cercano.
Que el usuario desea reportarse sano.	Se encuentre en la pantalla principal del portal	Se podrá visualizar un botón para reportarse sano y al darle clic este cambiará el estado de enfermo a sano del usuario.

Tabla 62. Historia de usuario PDS4-05.

Historia de usuario		
ID: PDS4-05		**Puntos estimados: 8**
Crear endpoint para registrar el estado de enfermo y alertar a los demás nodos.		
Sprint N°: 7	**Prioridad:** Alta	**Estado:** Por Hacer
Descripción: Como usuario, quiero poder registrar mi estado de enfermedad en el sistema, para alertar a los demás nodos (usuarios) con los que he tenido contacto recientemente y poder contribuir en la prevención de la		

	propagación de enfermedades y proteger la salud de las personas a mi alrededor.	
Criterios de Aceptación		
Dado	**Cuando**	**Entonces**
Que el usuario desea alertar que ha sido diagnosticado positivo al C-19	Haya eviado la petición HTTP de tipo PATCH.	Se podrá visualizar en la base de datos que su estado de enfermedad ha cambiado a true.
Que el usuario desea alertar que ya se ha recuperado	Haya enviado la petición HTTP de tipo PATCH.	Se podrá visualizar en la base de datos que el estado de su enfermedad ha cambiado a false.

6.9 **Anexo 9:** Product backlog para el sprint 5

Tabla 63. Product Backlog adaptado para el sprint 5.

PRODUCT BACKLOG				
HISTORIA ÉPICA	**Historias de Usuario**			
	ID	**Título**	**Estimación (Fibonacci)**	**Prioridad**
PDS1	PDS1-01	Realizar el diagrama de dispositivo IoT.	8	Alta
	PDS1-02	Soldar los componentes físicos del dispositivo IoT	5	Alta
	PDS1-03	Definir de la arquitectura IoT a ser usada	5	Alta
PDS2	PDS2-01	Diseñar un algoritmo para controlar y enviar la información de los dispositivos IoT al servidor back-end.	5	Media
	PDS2-02	Programar el Algoritmo y desplegarlo en los dispositivos IoT.	5	Media
	PDS2-03	Programar la emisión del nombre del dispositivo vía Bluetooth.	3	Media
PDS3	PDS3-01	Creación de proyecto base usando Nextjs	5	Media
	PDS3-02	Creación de pantalla de Inicio de Sesión y registro de usuario.	8	Media
	PDS3-03	Creación de pantalla para vincular y desvincular dispositivo	5	Media
	PDS3-04	Creación de infogramas para evitar contagios de C-19	3	Baja
	PDS3-05	Creación de pantalla principal y tabla de históricos de contactos	8	Media
	PDS3-06	Botón reportarse enfermo/sano	3	Alta
PDS4	PDS4-01	Creación del proyecto base usando Nest js	5	Alta
	PDS4-02	Crear endpoints para manejar el registro e inicio de sesión de usuario.	5	Alta

	PDS4-03	Crear endpoints para guardar los contactos registrados por el dispositivo IoT.	3	Alta
	PDS4-04	Crear endpoints para obtener la información del grafo de contactos y la tabla de históricos.	5	Alta
	PDS4-05	Crear endpoint para registrar el estado de enfermo y alertar a los demás nodos.	8	Alta
	PDS4-06	Crear endpoint para vincular o desvincular un dispositivo a una cuenta de usuario.	3	Alta

6.10 **Anexo 10:** Evidencias encuesta SUS

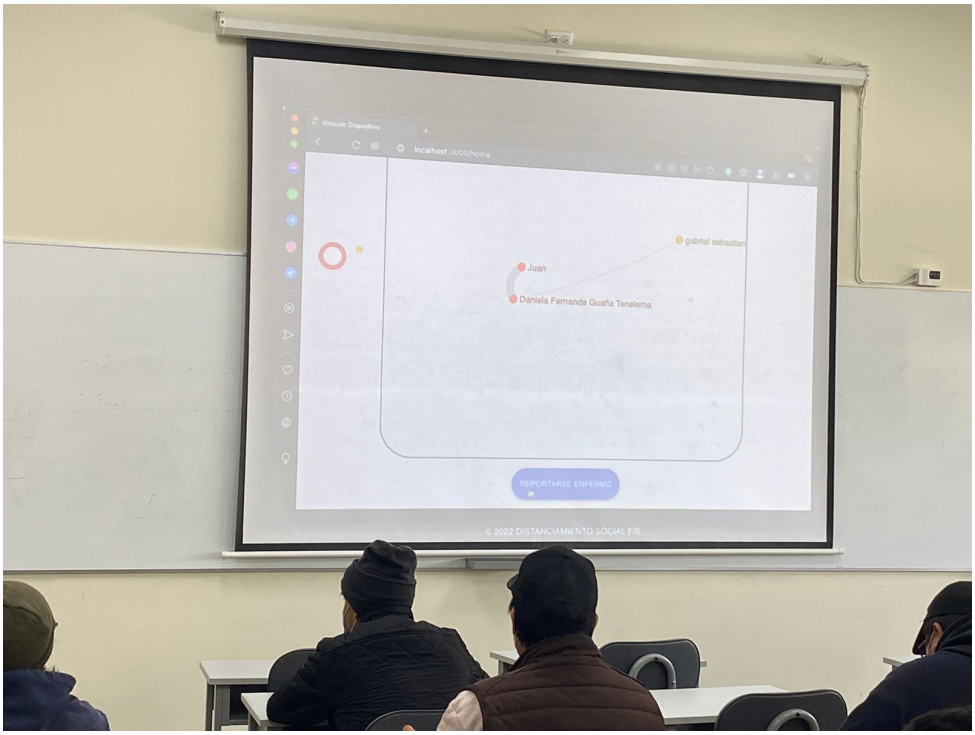


Figura 52. Grafo de contactos realizados.

Figura 53. Módulo de vinculación de dispositivo.

Figura 54. Módulo de login del sistema.