



Escuela Técnica Superior de Ingeniería Informática

INGENIERÍA DE COMPUTADORES

TRABAJO FIN DE GRADO

Chair Tracker

Autor/es:

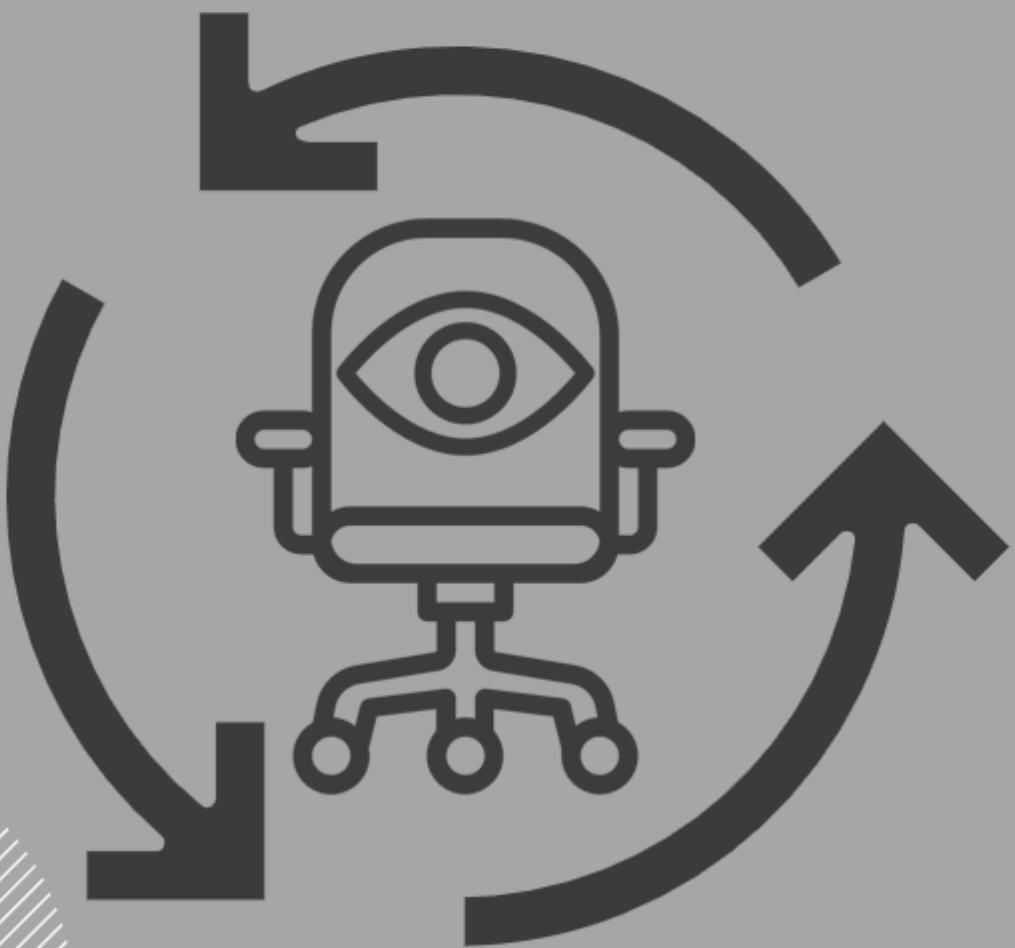
JOSÉ ARCINIEGA SALVATIERRA
JOSÉ ANTONIO GARCÍA MARTAGÓN

Tutor/es:

LUIS MIGUEL SORIA MORILLO

Primera convocatoria

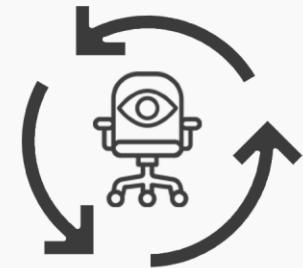
Curso 2021/2022



<<Chair Tracker>>

José Arciniega Salvatierra
José Antonio García Martagón

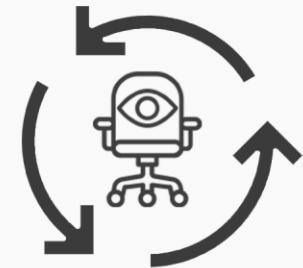
Trabajo Fin de Grado
Curso: 2021/22
Ingeniería de Computadores



Chair Tracker

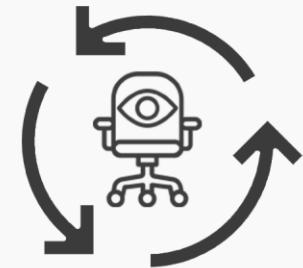
Índice

Índice	3
Índice de figuras	7
Índice de tablas	10
Resumen.....	11
Abstract.....	12
Bloque 1: Introducción.....	13
1.1 Contexto	14
1.1.1 Estado del arte.....	15
1.2 Análisis de mercado	18
Análisis funcional.....	18
Público objetivo.....	19
Análisis formal	20
Conclusiones.....	21
1.3 Glosario de términos	22
Bloque 2: Análisis.....	24
2.1 Objetivos	25
2.2 Mapa de historias de usuario	26
2.3 Catálogo de requisitos	28
2.3.1 Generales	28
2.3.2 De información	29
2.3.3 Funcionales	30
2.3.4 No funcionales.....	38
2.3.5 Reglas de negocio	40
2.4 Riesgos	41
Bloque 3: Planificación	43
3.1 Alcance del proyecto	44
3.2 Fases del proyecto.....	45
3.2.1 Fase 0: Inicio del proyecto	45
3.2.2 Fase 1: Planificación del proyecto	46
3.2.3 Fase 2: Diseño	47
3.2.4 Fase 3: Estudio de mercado	47



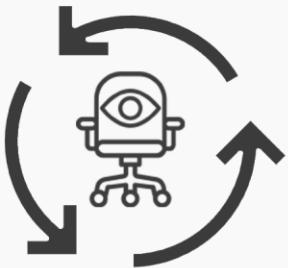
Chair Tracker

3.2.5 Fase 4: Desarrollo.....	48
3.2.6 Fase 5: Prueba y test.....	49
3.2.7 Fase 6: Cierre del proyecto	50
3.2.8 Documentación	50
3.3 Hitos del proyecto.....	50
3.3.1 Hito 1: Planificación del proyecto	50
3.3.2 Hito 2: Diseño de documentación y Hardware	50
3.3.3 Hito 3: Diseño Software	51
3.3.4 Hito 4: Estudio de mercado y revisión de la documentación	51
3.3.5 Hito 5: Desarrollo del Hardware	51
3.3.6 Hito 6: Desarrollo de la aplicación	52
3.3.7 Hito 7: Pruebas	52
3.3.8 Hito 8: Cierre del proyecto	52
3.4 Calendario de entregables	52
3.5 Diagrama de Gantt	53
3.6 Equipo planteado	62
3.7 Presupuesto del proyecto	63
Bloque 4: Análisis tecnológico.....	66
4.1 Herramientas utilizadas	67
4.1.1 Planificación	67
4.1.2 Diseño.....	67
4.1.3 Base de datos	68
4.1.4 API	68
4.1.5 Cliente.....	70
4.1.6 Boceto	70
4.1.7 App.....	71
4.1.8 PCB	72
4.1.9 Carcasa	73
4.1.10 Control de versiones	73
4.2 Lenguajes y plataformas.....	74
4.2.1 SQL	74
4.2.2 C	74



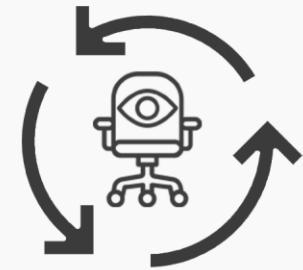
Chair Tracker

4.2.3 Java	75
4.2.4 TypeScript	75
4.2.5 React Native.....	76
Bloque 5: Diseño e Implementación	77
5.1 Software.....	78
5.1.1 Diseño arquitectura empleada	78
5.1.2 Base de datos	79
5.1.3 Verticles	80
5.1.4 API	80
Métodos GET	81
Métodos PUT	86
Métodos POST.....	87
Métodos DELETE.....	88
5.1.6 MQTT.....	89
Servidor	89
Cliente ESP8266	89
5.1.7 Cliente.....	90
Actuadores.....	90
Gestión alarmas.....	90
Gestión llamadas.....	91
5.1.8 Aplicación.....	92
Mockups de las pantallas de la app.....	92
Estructura de la aplicación	97
<i>Estructura de archivos</i>	97
<i>Estructura de navegación</i>	100
Aspectos destacables de implementación	101
Librerías utilizadas	107
5.1.9 Cambios introducidos.....	108
Base de datos	108
API.....	109
Métodos introducidos en la API	110
Diseño.....	111



5.2 Hardware	118
5.2.1 Fases del desarrollo del circuito	118
Prototipo en fase inicial de desarrollo	118
Prototipo en la segunda iteración de desarrollo	120
Prototipo en la tercera iteración de desarrollo	127
5.2.3 PCB	128
5.2.4 Carcasa	133
5.2.5 Sistema final	135
Bloque 6: Pruebas.....	140
6.1 Pruebas.....	141
6.1.1 Pruebas del hardware	141
6.1.2 Pruebas del software	143
Bloque 7: Cierre	144
7.1 Simulación de implementación en el mercado	145
7.2 Proyectos futuros	146
7.3 Conclusiones.....	147
Bibliografía	149
Anexos	155
1. Seguimiento temporal del proyecto	156
2. Manual de usuario.....	161
3. Actas de reuniones con el tutor	168

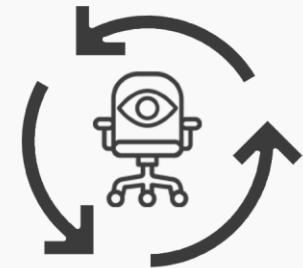
Chair Tracker



Chair Tracker

Índice de figuras

Figura 1. Mapa historias de usuario	27
Figura 2. Visión general fases planificación	54
Figura 3. Desglose de la planificación I	55
Figura 4. Desglose de la planificación II	56
Figura 5. Fase 0 de la planificación	57
Figura 6. Fase 1 y 2 de la planificación	58
Figura 7. Fase 3 de la planificación	59
Figura 8. Fase 4 de la planificación	60
Figura 9. Fase 5, 6 y Documentación de la planificación	61
Figura 10. Desglose del equipo de desarrollo del proyecto	63
Figura 11. Logo de Tom's planner	67
Figura 12. Logo de Photoshop	67
Figura 13. Logo de Illustrator	68
Figura 14. Logo de MySQL Workbench	68
Figura 15. Logo de Eclipse	69
Figura 16. Logo de Postman	69
Figura 17. Logo de MQTT Explorer	69
Figura 18. Logo de Visual Studio Code	70
Figura 19. Logo de Platform.io	70
Figura 20. Logo de Marvel App	71
Figura 21. Logo de Xcode	71
Figura 22. Logo de Android Studio	72
Figura 23. Logo de ESLint	72
Figura 24. Logo de KiCad	72
Figura 25. Logo de Autodesk fusión 360	73
Figura 26. Logo de Github	73
Figura 27. Logo de SQL	74
Figura 28. Logo de C	74
Figura 29. Logo de Java	75
Figura 30. Logo de TypeScript	75
Figura 31. Logo de React Native	76
Figura 32. Arquitectura completa de la aplicación	78
Figura 33. Esquema de la base de datos	80



Chair Tracker

Figura 34. Imágenes de los primeros mock-ups de la aplicación	96
Figura 35. Estructura de carpetas referentes a componentes ...	97
Figura 36. Estructura de carpetas referentes a contenedores ...	98
Figura 37. Estructura de archivos de una pantalla	98
Figura 38. Estructura de archivos del navegador	99
Figura 39. Estructura de carpetas referentes a las conexiones .	99
Figura 40. Flujo de navegación	100
Figura 41. Aspecto destacable del código de estilos	101
Figura 42. Aspecto destacable del código de hook de conexión a la API.....	102
Figura 43 .Aspecto destacable del código de almacenamiento en memoria	103
Figura 44. Aspecto destacable del código mqtt.....	104
Figura 45. Aspecto del código destacable del navegador	105
Figura 46. Aspecto del código destacable del uso de librerías	106
Figura 47. Listado de librerías utilizadas	107
Figura 48. Imágenes finales de la implementación de la app ..	117
Figura 49. Imágenes del prototipo inicial.....	120
Figura 50. Diagrama del transistor de encendido ESP-8266 ...	121
Figura 51. Diagrama de conexionado del dispositivo	122
Figura 52. Imágenes de la segunda iteración del dispositivo...	127
Figura 53. Imágenes de la primera PCB	128
Figura 54. Esquema del diseño de la primera PCB.....	129
Figura 55. Imágenes de la primera PCB	130
Figura 56. Imágenes de la primera PCB impresa.....	131
Figura 57. Imágenes de la PCB final.....	132
Figura 58. Imágenes del diseño de la carcasa 3D.....	134
Figura 59. Imagen comparativa entre las tres iteraciones de la carcasa.....	135
Figura 60. Ensamblado de la PCB	136
Figura 61. Imágenes del dispositivo final	139
Figura 62. Pruebas de voltaje y amperaje.....	142
Figura 63. Imágenes del dispositivo final	157
Figura 64. Desviaciones en la planificación I.....	158
Figura 65. Desviaciones en la planificación II.....	159
Figura 66. Desviaciones en la planificación III.....	160

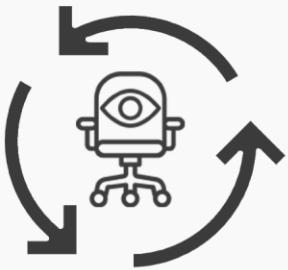
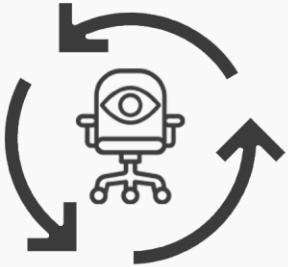


Figura 67. Configuración del AP del dispositivo	161
Figura 68. Configuración de credenciales del dispositivo	162
Figura 69. Welcome de la aplicación.....	163
Figura 70. Sección de alarmas de la aplicación	164
Figura 71. Sección de contactos de la aplicación.....	165
Figura 72. Sección de estadísticas de la aplicación	166
Figura 73. Sección de perfil de la aplicación	167
Figura 74. Sección de términos y condiciones de la aplicación	167

Chair Tracker

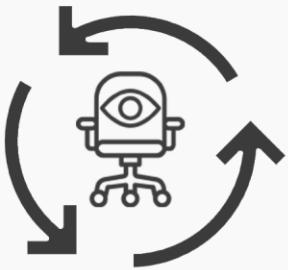




Índice de tablas

Tabla 1. Tabla comparativa entre aplicaciones y funcionalidad.	17
Tabla 2. Matriz de riesgos	42
Tabla 3. Tabla presupuesto hardware.....	64
Tabla 4. Tabla presupuesto personal.....	65

**Chair
Tracker**



Resumen

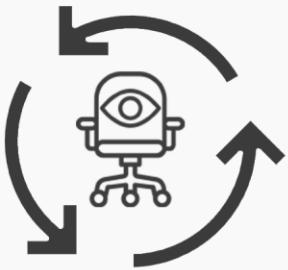
En este proyecto se presenta un sistema completo, combinando hardware y software, por el que se le ofrece al usuario un dispositivo con sensores y actuadores que combinados con su aplicación móvil (disponible en Android e iOS) permitirán diversas funcionalidades relativas a la higiene postural.

Gracias a la aplicación, el usuario podrá comprobar estadísticas del tiempo que pasa sentado además de poder establecer ciclos de trabajo, donde permanecerá sentado, y ciclos de descanso, donde deberá levantarse de la silla. Todo esto se integra con el dispositivo físico que mediante actuadores invitará al usuario a cumplir estos plazos establecidos en pos de que cuide de su salud.

No sólo se ofrecen las funcionalidades anteriormente mencionadas, sino que también se provee al usuario de herramientas para velar por su comodidad y concentración durante los ciclos de trabajo pues podrá recibir avisos de otros usuarios del sistema a modo de llamada, donde se le podrá notificar mediante una vibración del dispositivo físico.

En el desarrollo del proyecto se usan tecnologías como MySQL para la base de datos, Vert.x, ESP8266 y React Native para el desarrollo de la aplicación. Dando como resultado una aproximación de un producto final listo para ser lanzado al mercado.

Palabras claves: sistema empotrado, aplicación, higiene postural, método Pomodoro, React Native, Android, iOS.



Abstract

The aim of this project is to present a complete system, combining software and hardware, that offers the user a device with embedded sensors and actuators, together with a mobile app (available at Android and iOS). In this manner, we enable the user to use several features related to postural hygiene.

Thanks to the app, the user will have the option to check stats about their sitting time as well as establishing work cycles, where the user would still be sitting, and rest cycles, where the user would need to get up from the chair. Integrating all of these features on the app with the physical device, will make sure that the user meets these cycles in order to ensure his health.

Not only these features are offered but also, we provide the users with proper tools to maintain their comfort and concentration during working cycles since they would be able to keep in touch with other users without being interrupted due to the notification system that uses vibrating alarms in the physical device.

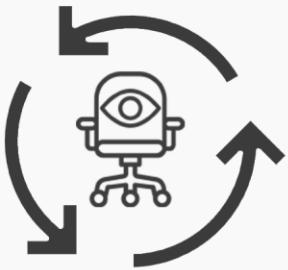
During this project, many technologies have been used in the development of this app such as MySQL, Vert.x, ESP8266 and React Native, just to name a few. Resulting in final product almost ready to be placed on the market.

Chair Tracker

Key Words: embedded system, app, postural hygiene, Pomodoro method, React Native, Android, iOS.

Bloque 1:

Introducción



1.1 Contexto

Cada vez son más las personas que pasan gran parte de su tiempo sentados frente al ordenador ya sea por trabajo u ocio.

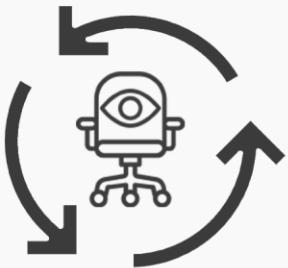
A esto debemos de sumar el factor de la pandemia que nos ha acontecido durante los últimos años y que ha tenido como consecuencia un incremento en los deberes que han decidido deslocalizarse optando por un modelo telemático. Ejemplo de ello han sido ciertos sectores del mercado laboral, educación, actividades extracurriculares, etc.

Además, son varios los estudios que avalan los efectos negativos que produce en nuestro organismo el sedentarismo y la mala higiene postural derivada de este mismo. Algunos de ellos son: Sedentarism and chronic disease risk in COVID 19 lockdown – a scoping review [\[8\]](#) o Sedentarism, a disease from xxi century [\[5\]](#)

Es por esto por lo que ahora más que nunca debemos de ser conscientes de las horas que pasamos sentados y realizar ciclos alternando periodos de tiempo de trabajo en la silla y descanso en una posición erguida. Precisamente este es el objetivo principal de nuestro dispositivo.

Desde nuestra perspectiva, podemos aportar una solución técnica y profesional que tratará de resolver, como primera instancia, la reducción de la cantidad de horas que los empleados pasan sentados en una silla.

Chair Tracker



Chair Tracker

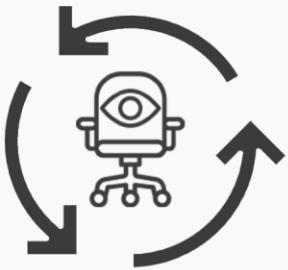
1.1.1 Estado del arte

Actualmente, existen varias propuestas que intentan abordar los mismos objetivos que se plantean en este proyecto.

Algunos de los mejores competidores que hemos encontrado son smart wearables, Chairless [\[7\]](#), Sitting Time Tracker [\[11\]](#) y Couch Potato - Sit Tracker [\[11\]](#).

La primera alternativa la encontramos en dispositivos inteligentes, cuya funcionalidad principal reside en la monitorización de la actividad física. Muchos de estos dispositivos permiten emitir un aviso cuando el usuario pasa mucho tiempo sentado. Sin embargo, se trata de una simple notificación que el usuario puede ignorar o incluso nunca llegar a ver. Además, no permiten generar ciclos de tiempo sentado y de tiempo de descanso, por lo que no tendría un uso realmente efectivo para la finalidad que se está buscando. Otro aspecto deficiente de este tipo de sistemas alude a la falta de personalización de estos tiempos. Así como la ausencia de datos del tiempo de sedentarismo, ya que la mayoría de estos solo tienen en cuenta el tiempo activo cuando el usuario se encuentra en movimiento.

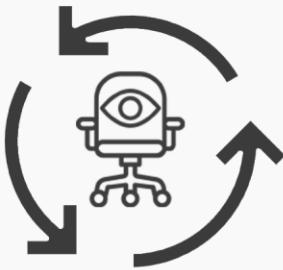
La siguiente propuesta es Chairless, una aplicación de la monitorización del tiempo que pasa un usuario sentado, disponible únicamente para dispositivos Android. Al tratarse únicamente de una aplicación móvil, basa su uso en el estado del giroscopio del teléfono, inhabilitando el uso del dispositivo si se quiere mantener el contador de tiempo sentado, ya que el más mínimo movimiento provoca la pausa de este. Muchos usuarios han reportado esto como un problema, debido a la inexactitud del sistema si la actividad que realiza el usuario sentado requiere del uso del dispositivo móvil.



Chair Tracker

Sitting Time Tracker es un prototipo de un dispositivo físico capaz de registrar el tiempo que se pasa frente al ordenador. Es una alternativa de código abierto, donde se explica tanto el montaje como el funcionamiento del código implementado en la placa. De cara a la creación de un producto, está mucho más alejado de la propuesta que se intenta alcanzar con nuestro proyecto, pues le faltan bastante de las funciones que queremos implementar. Esto es debido a que es un sistema bastante simple y carece de base de datos, aplicación para el usuario, dispositivo físico final, etc. Todas estas características son las que hacen que nuestro sistema sea amigable para cualquier tipo de usuario, en contraposición con este prototipo que requiere de conocimientos técnicos para poder ponerlo en funcionamiento.

Otra alternativa es la de Couch Potato - Sit Tracker, donde la propia aplicación es capaz de monitorizar el tiempo que se pasa sentado en el sofá y es capaz de generar estadísticas diarias con estos datos. Uno de los grandes inconvenientes que tiene es que solamente está disponible para iOS, en contraposición con la aplicación que explicamos anteriormente. Esto ha sido uno de los motivos que nos ha llevado a tomar la decisión de crear la aplicación agrupando a los dos grandes sistemas operativos existentes, de cara a satisfacer al mayor número de usuarios posibles, dejando a un lado el tipo de dispositivo que posean. Uno de los aspectos más preocupantes de esta aplicación en concreto, es el sistema de recompensas que ofrece. El usuario subirá de nivel a medida que haya alcanzado ciertos tiempos sentado, aspecto negativo porque puede llegar a incitar sedentarismo. Suponemos que este sistema de recompensas ha sido creado para generar la adherencia al uso de su aplicación, pero consideramos que puede



acarrear el impacto contrario del buscado por nuestra propuesta.

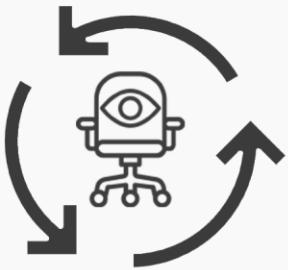
Existen multitud de aplicaciones que intentan abordar los mismos objetivos, pero se pueden reducir todas las funcionalidades que implementan con los ejemplos explicados.

A continuación, se muestra una tabla comparativa entre las alternativas anteriores y nuestra propuesta, para simplificar el estudio del arte realizado:

	Wearables	Chairless	Sitting Time Tracker	Couch Potato	Chair Tracker
iOS	✓	✗	✗	✓	✓
Android	✓	✓	✗	✗	✓
Dispositivo físico	✓	✗	✓	✗	✓
Base de datos	✓	✓	✗	✓	✓
Registro de tiempo levantado	✗	✗	✗	✗	✓
Registro de tiempo sentado	✓	✓	✓	✓	✓
Generación de estadísticas	✗	✓	✗	✓	✓
Capacidad de personalización	✗	✓	✗	✗	✓

Tabla 1. Tabla comparativa entre aplicaciones y funcionalidad

Chair Tracker



1.2 Análisis de mercado

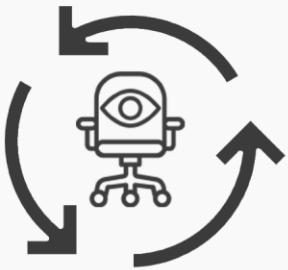
En este apartado se presenta un análisis del mercado donde se detallarán las funcionalidades del dispositivo presentado junto con el posible mercado al que se podrá enfrentar y los factores de forma que presenta su actual competencia.

Análisis funcional

Las funcionalidades que presenta nuestro sistema son las siguientes:

- **Registro de horas que pasa el usuario sentado:** una de las funciones principales de nuestro sistema es el registro de horas que pasa el usuario sentado que tendrán uso diversos campos de aplicación sin importar que este tiempo sea de ocio o de trabajo.
- **Muestreo de estadísticas:** el usuario podrá observar estadísticas sobre el punto anterior permitiéndole actuar en consecuencia según lo crea necesario. Todo esto de una forma sencilla e intuitiva en una aplicación móvil independientemente de la plataforma de preferencia.
- **Establecimiento de ciclos de trabajo y descanso:** gracias al uso de la aplicación el usuario definirá un tiempo en el cual se encontrará sentado y otro tiempo durante el cual deberá de descansar de la actividad que realice durante el tiempo sentado establecido anteriormente. Estos ciclos pueden ser establecidos para diferentes días de la semana y que se inicien en horas concretas.
- **Notificaciones al usuario:** para el cumplimiento de estos ciclos establecidos, tanto la aplicación como el dispositivo físico mantendrán comunicación para notificar al usuario de





forma sonora de que ha llegado el fin de su ciclo de trabajo o descanso y deberá descansar o reanudar su actividad, respectivamente.

- *Sistema de llamadas:* para evitar posibles distracciones, ya sea en un entorno laboral o de ocio, el usuario recibirá notificaciones gracias a otro de los actuadores del dispositivo físico, que vibrará avisándole de que deberá consultar el mensaje recibido de forma orgánica y completamente integrada, sin necesidad de acudir a aplicaciones externas que interrumpan el flujo en el que se encuentren.
- *Precisión:* el sistema ofrece una gran precisión respecto a la detección del usuario gracias a su sensorización lo cual facilita el uso y no implica al usuario en el proceso de inicio o pausa del contador del ciclo en el cual se encuentre.
- *Mejora de la higiene postural:* como consecuencia de todas las funcionalidades anteriores conseguimos fomentar la higiene postural mediante el descanso activo del usuario.

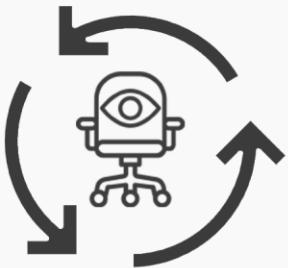
Público objetivo

Nuestro público objetivo es diverso debido a la versatilidad de nuestro sistema dentro de diferentes ámbitos de aplicación que comparten un problema común, la gestión del tiempo. Por ello se ha decidido realizar la siguiente división en bloques:

Chair Tracker

- *Empresas:* su objetivo es la obtención de un producto que sea capaz de llevar un registro de las horas de trabajo, en el entorno de oficina, de cada uno de sus empleados. Esto debe ser realizado de forma eficiente y contando únicamente las horas que el usuario se





encuentre frente al ordenador o herramienta de trabajo. Gracias a esto podrán llevar un conteo de las horas y poder realizar estudios asociados a la productividad.

- *Clientes finales:* su objetivo es distinto al público anterior pues existen diferentes ámbitos donde el usuario lo puede aplicar. Uno de sus usos principales es el conteo de horas para ganar perspectiva sobre las horas que pasa sentado y poder mejorar su higiene postural o ser más conscientes de su actividad.

Otros de los campos donde destaca es en el de TDAH donde las estadísticas pueden ser usadas para observar cuantas veces se levanta el usuario a lo largo de un tiempo establecido y cuál es el número de horas reales dedicadas a intentar realizar una actividad que requiera de encontrarse sentados.

De esta forma observamos que se trata de un producto versátil que tiene más de un campo de aplicación y por consecuencia, de comercialización haciendo posible la venta a clientes finales (B2C) o a otras empresas (B2B). Dicha cuestión se discutirá en más profundidad en el apartado del alcance del proyecto.

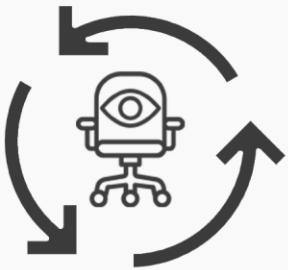
Chair Tracker

Análisis formal

Como forma de posibles soluciones a la problemática dada encontramos en el mercado los siguientes tipos:

- *Wearables:* se tratan de dispositivos vestibles que el usuario puede integrar en su vida diaria sin que supongan una molestia. Suelen presentar forma de reloj o pulsera, aunque





existen diversos tipos como sensores colocados en el pecho.

- **Aplicaciones:** el producto no depende de un dispositivo físico, sino que funciona instalando una aplicación en el smartphone del usuario. Esto implica limitaciones en el desarrollo pues no todos los móviles presentan el mismo tipo de sonorización o tienen la adecuada para llevar a cabo la función deseada de forma eficiente y hay que realizar aproximaciones. Existen diversos sistemas operativos, lo cual decide en gran parte su desarrollo.
- **Gadgets como objetos independientes:** este tipo de producto no dependen de ningún tipo de adaptador para su utilización. Algunos de ellos disponen de conexiones con aplicaciones móviles, redes WIFI o USB. Una de sus características es que necesitan de un método de recargar al incorporar una batería, pues, aunque el sistema sea inalámbrico, necesitará un suministrador eléctrico en ocasiones.

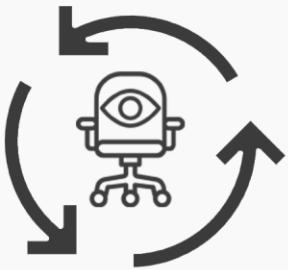
En este caso se ha optado por un factor de forma de gadget como objeto independiente pues es el que más se adecuado al cumplimiento total de las funcionalidades que se deseaban cumplimentar como objetivo. De esta forma, se consigue un dispositivo preciso, autónomo y con fácil usabilidad para el usuario a través de la aplicación móvil proporcionada.

Chair Tracker

Conclusiones

Una vez estudiados los tipos de forma, público objetivo, y funcionalidades podemos concluir con que gracias al factor de forma seleccionado podemos proveer al usuario con un sistema completo que cumple con todas las funcionalidades descritas y que como factor diferenciador respecto a sus competidores presenta una





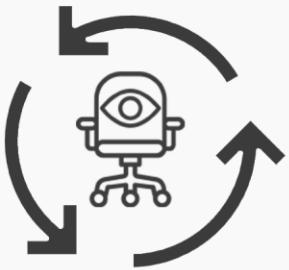
gran precisión en la detección del usuario, permitiendo de esta forma una integración casi transparente y sin necesidad de atención pues podrá establecer ciclos de trabajo y descanso en un horario y días de la semana determinados mientras se actualizan automáticamente sus estadísticas sin importar la plataforma que utilice en lo que respecta a su dispositivo móvil, gracias a su desarrollo multiplataforma en React Native. Esto se puede ver en mayor detalle en la tabla comparativa presentada en el estado del arte.

1.3 Glosario de términos

Los términos relativos al desarrollo del proyecto son los siguientes:

- **Usuario:** persona a la que se le asocia un sha1 de la MAC de una placa.
- **SHA-1:** función hash utilizada para encriptación de datos, en este caso, la MAC del ESP8266 que utilizamos como cliente.
- **Rol:** papel que puede tomar el usuario dentro de la empresa, distinguimos entre jefe y empleado.
- **Llamada:** aviso mediante vibración que podrá enviar un usuario a otro para notificar de cierto evento en concreto que puede especificar mediante una descripción.
- **Estado llamada:** la llamada puede presentar dos estados:
 - 'Pendiente' cuando el usuario aún no ha marcado como respondida la llamada desde la web.
 - 'Contestada' ha sido marcado el campo explicado anteriormente.
- **Alarma:** aviso sonoro que avisara al usuario de que debe levantarse o sentarse.



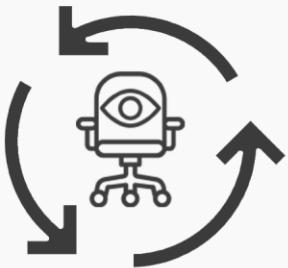


- *Ciclo trabajo/descanso:* número de veces que se ha completado un tiempo de trabajo o tiempo de descanso.
- *Remitente:* usuario que realiza la llamada.
- *Destinatario:* usuario que recibe la llamada.
- *Registro:* guardado de información de alarmas/llamadas para la posterior generación de estadísticas en la app.
- *Estadísticas:* reporte de las horas de trabajo (sentado) y descanso (levantado) clasificadas por alarma.

Chair Tracker



Bloque 2: Análisis



2.1 Objetivos

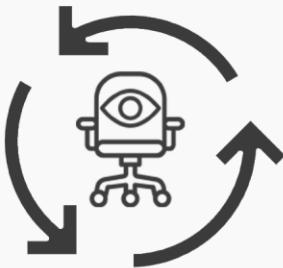
Entre los objetivos principales del proyecto se encuentra mejorar la higiene postural de nuestros usuarios ayudándonos de ciclos de trabajo, donde permanecerá sentado y ciclos de descanso en lo que se deberá de levantar.

Además, se quiere suministrar al usuario con una serie de estadísticas para que pueda realizar un control efectivo sobre sus horas de trabajo junto con una interfaz que le permita comunicarse con otros usuarios sin necesidad de causar distracción a lo largo de su ciclo de trabajo, esto se realizará a través de las denominadas llamadas. Todo esto se pretende que sea de fácil adopción para el usuario proveyéndolo con un ecosistema de fácil usabilidad a través de una aplicación móvil nativa para su smartphone.

De esta forma, el objetivo principal en cuanto a desarrollo se trata de la creación de un sistema hardware que sea capaz de controlar la lógica referente a la sonorización y actuadores que interactuaran con el usuario. Este hardware constará de un firmware integrado en la placa, además de diferentes capas de software como base de datos, API o la aplicación.

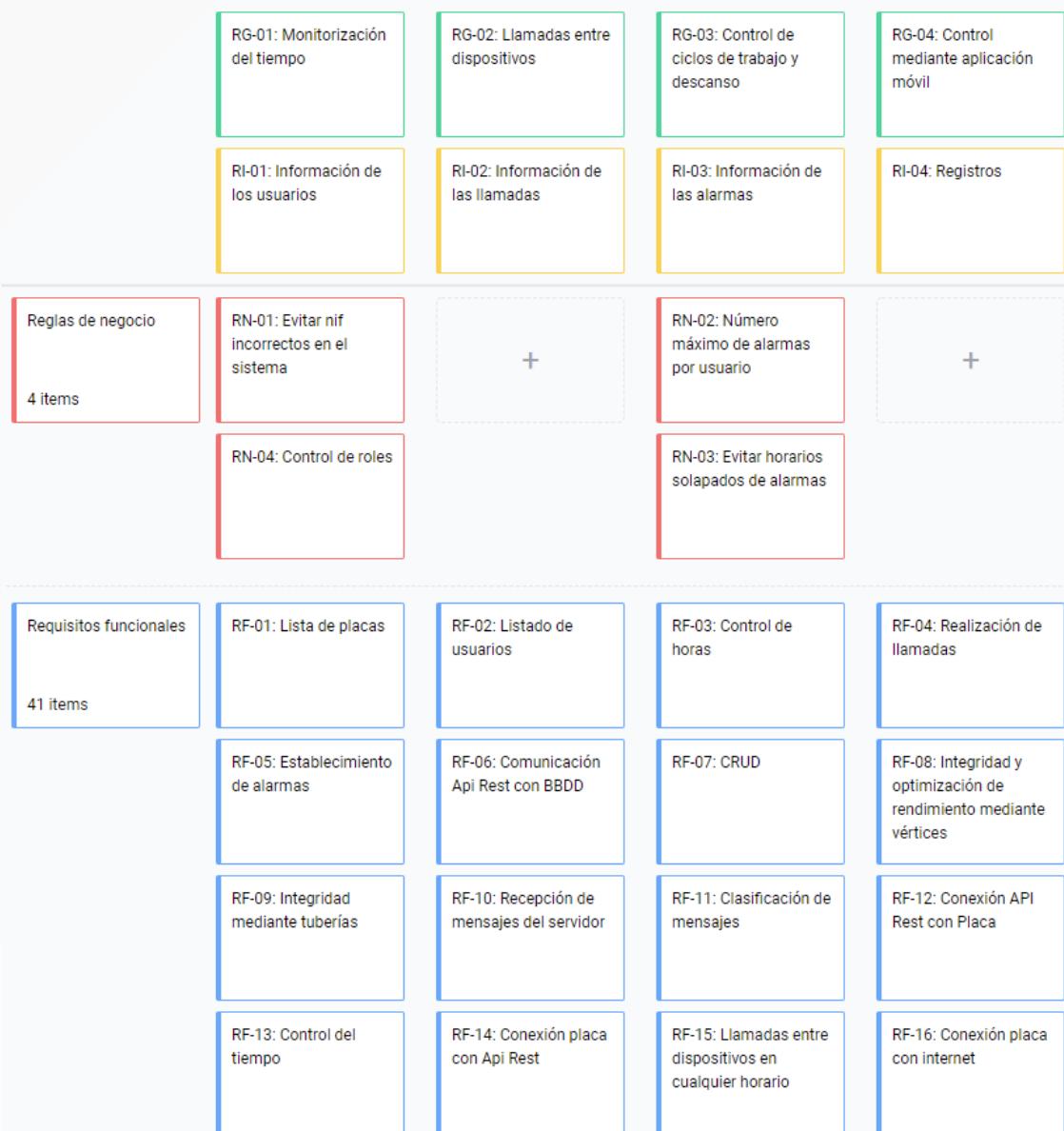
Como objetivo secundario tenemos, más a largo plazo, la ampliación de este proyecto a otros campos como por ejemplo en el sector de la educación, el transporte y la seguridad vial.

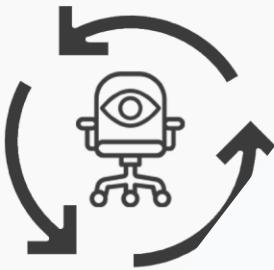
Chair Tracker



2.2 Mapa de historias de usuario

A continuación, se representa un resumen organizativo del catálogo de requisitos completo a modo de mapa de historias de usuario.





Requisitos no funcionales

8 items

RNF-01: Registro de auditoría

RNF-02: Cifrado MAC

RNF-03 Fácil usabilidad del sistema

RNF-04 Tiempos de respuesta rápidos

RNF-05 Seguridad de usuarios

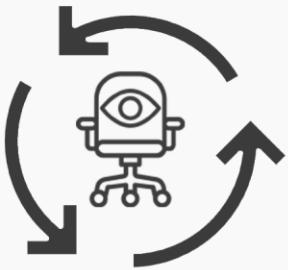
RNF-06 Control de roles

RNF-07 Respaldo de datos

RNF-08 Diseño personalizado

Chair Tracker

Figura 1. Mapa historias de usuario



2.3 Catálogo de requisitos

2.3.1 Generales

RG-01: Monitorización del tiempo.

Como jefe.

Quiero saber el tiempo de trabajo y descanso de mis empleados.

Para tener un registro actualizado sobre las horas tiempo productivo en relación con el descanso para así mejorar la higiene postural de mis empleados.

RG-02: Llamadas entre dispositivos.

Como usuario.

Quiero poder contactar con otros usuarios del sistema sin necesidad de abandonar el puesto de trabajo.

Para poder notificar de diferentes eventos como la finalización de una tarea o inicio de una reunión sin interrumpir al otro usuario.

RG-03: Control de ciclos de trabajo y descanso.

Como usuario.

Quiero poder establecer los tiempos de trabajo y descanso en un ciclo.

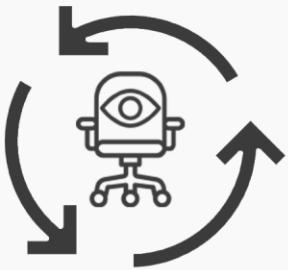
Para poder recibir un aviso sonoro al cumplir un ciclo para no exceder los tiempos establecidos sin necesidad de gestionarlo de forma manual.

RG-04: Control mediante aplicación móvil.

Como usuario.

Quiero poder gestionar las funciones principales del sistema a través de una aplicación móvil multiplataforma.

Para poder simplificar el modo de uso y recopilación de la información.



2.3.2 De información

RI-01: Información de los usuarios.

Como administrador.

Quiero conocer la clave HASH de la placa, el NIF, contraseña, la última fecha de acceso al sistema, el nombre, apellidos y su rol dentro de la aplicación. En caso de que se trate de un empleado deberá de aparecer el NIF de su jefe.

Para poder diferenciar a los distintos usuarios dentro de nuestra aplicación.

RI-02: Información de las llamadas.

Como administrador.

Quiero conocer el estado, desde el lugar que se realiza la llamada, así como el Hash Mac de la placa del remitente y destinatario.

Para poder llevar un control el registro de llamadas entre usuarios.

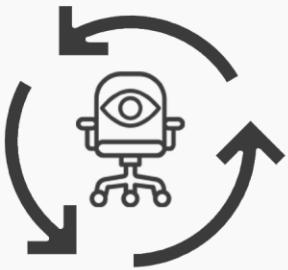
RI-03: Información de las Alarmas.

Como administrador.

Quiero conocer los días en los cuales la alarma se encuentra activa, el tiempo de inicio, tiempo de fin, tiempo planificado de trabajo y descanso además de los ciclos de trabajo y descanso completados asociados a esa alarma en concreto.

Para poder llevar un control el registro de alarmas de cada usuario.

Chair Tracker



RI-04: Registros.

Como administrador.

Quiero tener información sobre el tipo de registro que se va a almacenar (alarma o llamada), junto con el oid correspondiente en cada caso, la fecha, el tiempo total de trabajo, el tiempo total de descanso y en si se tratara de una llamada el Hash Mac del remitente y destinatario.

Para poder registrar las llamadas y poder llevar un control sobre el tiempo total de trabajo y descanso de los usuarios.

2.3.3 Funcionales

RF-01: Lista de placas.

Como jefe.

Quiero tener una lista de todas las placas del sistema.

Para poder llevar un control de las MAC de cada placa en cada usuario.

RF-02: Listado de usuarios.

Como jefe.

Quiero tener un registro de todos los usuarios del sistema.

Para poder generar un listado con todos los empleados de la empresa.

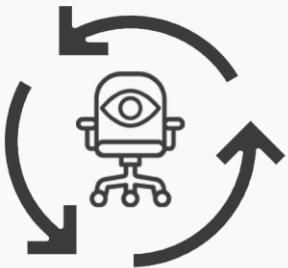
RF-03: Control de horas.

Como usuario.

Quiero tener un control sobre las horas de trabajo/descanso realizadas.

Para poder saber si cumplí mi horario de trabajo correctamente.





Chair Tracker

RF-04: Realización de llamadas.

Como usuario.

Quiero poder realizar llamadas a otra placa.

Para poder enviar avisos a mis compañeros de trabajo de forma sencilla.

RF-05: Establecimiento de alarmas.

Como usuario.

Quiero poder establecer alarmas en la placa fácilmente.

Para poder establecer mi horario de trabajo/descanso personalmente.

RF-06: Comunicación Api Rest con BBDD.

Como usuario.

Quiero poder conectar la Api Rest de forma sencilla.

Para poder garantizar el correcto funcionamiento del sistema de cara a la implementación final.

RF-07: CRUD.

Como usuario.

Quiero disponer en la Api Rest de un CRUD de las tablas de la BBDD.

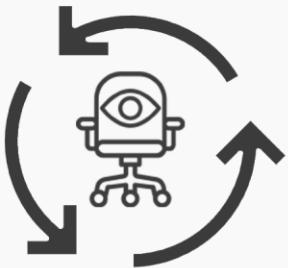
Para ser capaces de gestionar la información de forma más sencilla y permitir mejoras de funcionalidad en el sistema.

RF-08: Integridad y optimización de rendimiento mediante vértices.

Como usuario.

Quiero disponer de vértices en la Api Rest.

Para fomentar el trabajo en paralelo del sistema y mejorar la organización e integridad de las tareas.



RF-09: Integridad mediante tuberías.

Como usuario.

Quiero intercambiar mensajes entre los vértices.

Para comunicar información entre los distintos procesos del sistema.

RF-10: Recepción de mensajes del servidor.

Como usuario.

Quiero recibir y enviar mensajes Mqtt entre el dispositivo y la Api Rest.

Para poder tramitar la información crítica de alarmas y llamadas en el sistema.

RF-11: Clasificación de mensajes.

Como usuario.

Quiero poder enviar y recibir mensajes Mqtt según una clasificación.

Para poder usar los mensajes en el servidor dependiendo de la clasificación que atenderá a una funcionalidad determinada.

RF-12: Conexión API Rest con Placa.

Como usuario.

Quiero poder conectar de forma sencilla la Api Rest con el dispositivo.

Para poder garantizar el correcto funcionamiento del sistema de cara a la implementación final.

RF-13: Control del tiempo.

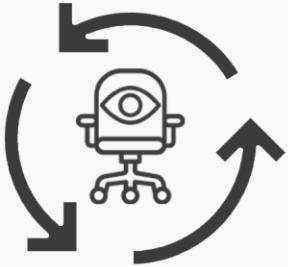
Como administrador del sistema.

Quiero delegar en servidores NTP lo relacionado con la fecha y hora.

Para tener un control exacto sobre los registros de los tiempos en la jornada laboral.

Chair Tracker





Chair Tracker

RF-14: Conexión placa con Api Rest.

Como usuario.

Quiero poder conectar de forma sencilla el dispositivo con la Api Rest.

Para poder garantizar el correcto funcionamiento del sistema de cara a la implementación final.

RF-15: Llamadas entre dispositivos en cualquier horario.

Como usuario.

Quiero poder enviar y recibir llamadas sea en horario de trabajo o de descanso.

Para evitar pérdidas de avisos importantes en la jornada laboral.

RF-16: Conexión placa con internet.

Como administrador del sistema.

Quiero tener conexión Wifi de forma sencilla en la placa.

Para poder comunicarnos con los distintos nodos de nuestro sistema de forma inalámbrica.

RF-17: Conexión Mqtt no bloqueante.

Como usuario.

Quiero poder realizar tareas en la placa sin perder funcionalidad.

Para ser capaz de recibir llamadas o mensajes de establecimiento de alarmas mientras se cuentan las horas de trabajo o descanso.

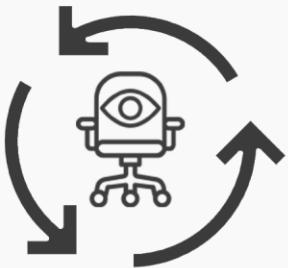
RF-18: Envío de mensajes Mqtt desde la placa.

Como usuario.

Quiero enviar mensajes Mqtt de forma automática.

Para poder establecer nuevas alarmas o llamadas entrantes fácilmente.





Chair Tracker

RF-19: Formateo horas.

Como administrador del sistema.

Quiero ser capaz de combinar los formatos de las horas de los servidores NTP con los usados por la BBDD y la Api Rest.

Para poder garantizar la integridad de las horas en el sistema.

RF-20: Obtención de la próxima alarma.

Como administrador del sistema.

Quiero que la placa actualice automáticamente la próxima alarma programada.

Para dar comodidad al usuario a la hora de realizar la jornada establecida.

RF-21: Aviso de alarma de trabajo/descanso.

Como usuario.

Quiero recibir un aviso sonoro y vibratorio por parte del dispositivo cuando empiece o acabe un periodo de trabajo o descanso.

Para poder trabajar sin prestar atención a cumplimentar los horarios propuestos.

RF-22: Aviso de llamada de otro usuario.

Como usuario.

Quiero recibir un aviso vibratorio cuando otro usuario me haga una llamada.

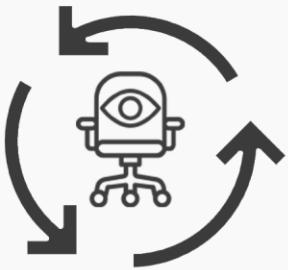
Para recibir el aviso sin dañar demasiado la concentración en el trabajo.

RF-23: Indicador de encendido/apagado.

Como usuario.

Quiero saber el estado del dispositivo mediante un LED.

Para no tener que comprobar manualmente ni perder tiempo en el inicio de la jornada laboral.



RF-24: Detección de actividad del trabajador.

Como administrador del sistema.

Quiero reconocer si un usuario se encuentra sentado o levantado de la silla.

Para gestionar realmente los ciclos de trabajo/descanso que realiza.

RF-25: Registro automático.

Como administrador del sistema.

Quiero que se actualicen automáticamente los ciclos de trabajo y descanso del usuario.

Para generar un registro de los tiempos de trabajo y descanso totales que se usarán para generar estadísticas.

RF-26: Prototipo del sistema.

Como administrador del sistema.

Quiero elaborar un prototipo para estipular tamaño, coste, montaje, funcionalidad...

Para poder dar una visión general al cliente, previa a la implementación final del sistema.

RF-27: Impresión de carcasa 3D.

Como administrador del sistema.

Quiero ofrecer al usuario una carcasa impresa en 3D.

Para mejorar el prototipo y acercarnos a la generación de un producto final.

RF-28: Login.

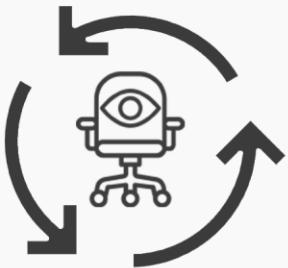
Como administrador del sistema.

Quiero tener un login en la aplicación.

Para poder identificar o registrar a un usuario.

Chair Tracker





Chair Tracker

RF-29: Página principal.

Como usuario.

Quiero tener una página principal en la aplicación.

Para permitir acceder de forma rápida a las funcionalidades de esta.

RF-30: Perfil usuario.

Como usuario.

Quiero poder ver mi información personal en la aplicación.

Para ser capaz de actualizar los datos personales de forma sencilla.

RF-31: Estadísticas de horas de trabajo/descanso.

Como usuario y jefe.

Quiero tener un registro en la aplicación de todas las horas de trabajo y descanso realizadas en un periodo de tiempo.

Para poder justificar el esfuerzo del trabajo realizado, así como la realización de horas extras.

RF-32: Creador de llamadas.

Como usuario.

Quiero poder llamar a otro usuario desde la aplicación.

Para mandar avisos a otro usuario de forma sencilla y rápida.

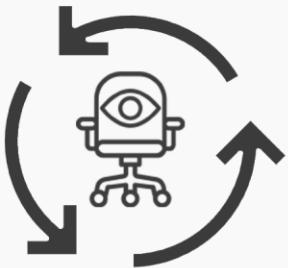
RF-33: Creador de alarmas.

Como usuario.

Quiero poder establecer mis propios horarios de trabajo a través de la aplicación.

Para crear flexibilidad y mejorar el rendimiento cumpliendo con el tiempo de descanso y trabajo totales.





Chair Tracker

RF-34: Conexión App con Api Rest.

Como administrador del sistema.

Quiero tener conexión de forma sencilla entre la aplicación y la Api Rest.

Para poder garantizar el correcto funcionamiento del sistema.

RF-35: Calendario de horarios de trabajo.

Como usuario.

Quiero disponer de un calendario con mi horario de trabajo en la aplicación.

Para poder cerciorarse de forma rápida y visual del correcto establecimiento de las alarmas.

RF-36: Acerca de.

Como administrador del sistema.

Quiero ofrecer información sobre la aplicación y el proyecto.

Para poder informar al usuario acerca de la empresa, además de ofrecer un contacto con la atención al cliente o desarrolladores.

RF-37: Agenda de contacto.

Como usuario.

Quiero disponer de una agenda de contactos en la aplicación.

Para facilitar las llamadas entre usuarios al no ser indispensable el uso de la MAC de las placas.

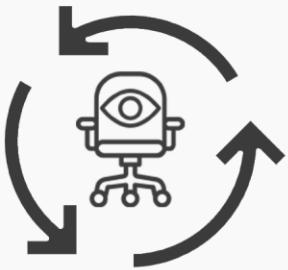
RF-38: Configuración Wifi.

Como usuario.

Quiero disponer de una página en la aplicación que facilite la configuración Wifi del sistema.

Para asegurar el correcto funcionamiento del sistema y mejorar su usabilidad.





2.3.4 No funcionales

RNF-01: Registro de auditoría.

Como jefe.

Quiero tener un historial de todas las alarmas y llamadas creadas en el sistema.

Para poder tener una buena integridad en caso de incidencia en el sistema.

RNF-02: Cifrado MAC.

Como administrador del sistema.

Quiero garantizar la integridad de la MAC de la placa.

Para evitar ataques al sistema mediante el robo de datos en la BBDD.

RNF-03 Fácil usabilidad del sistema.

Como usuario.

Quiero que el sistema sea amigable con el usuario, que la interfaz del sistema sea intuitiva y que muestre los datos correctamente.

Para el fácil uso del usuario y que sea capaz localizar rápidamente los elementos a usar, agilizando las labores dentro del sistema.

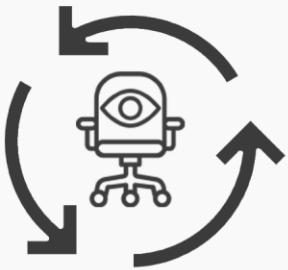
RNF-04 Tiempos de respuesta rápidos.

Como usuario y administrador.

Quiero que se proporcionen tiempos de respuesta rápidos menores a 3 segundos y que se evite que las consultas tarden demasiado en ser devueltas.

Para que la interacción entre usuario y sistema sea fluida, puesto que algunas acciones dependen del tiempo.





RNF-05 Seguridad de usuarios.

Como usuario y administrador.

Quiero que nuestras credenciales estén a buen recaudo en todo momento.

Para evitar suplantaciones de identidad y así impedir que sujetos ajenos al sistema accedan a él.

RNF-06 Control de roles

Como jefe.

Quiero limitar el acceso a la información de los usuarios usando roles según la LOPD.

Para evitar el acceso no autorizado a partes de la información almacenada en la aplicación.

RNF-07 Respaldo de datos.

Como administrador.

Quiero que el sistema ofrezca un respaldo de los datos de la base de datos.

Para evitar la pérdida de datos en el sistema.

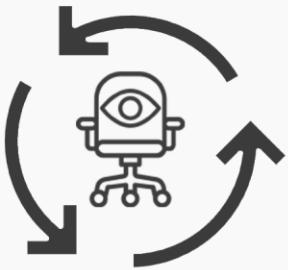
RNF-08 Diseño personalizado.

Como usuario.

Quiero que el sistema presente un diseño acorde con la empresa donde se muestren colores y líneas de diseño corporativas creando así una identidad.

Para mostrar una imagen cohesionada y funcional.

Chair Tracker



2.3.5 Reglas de negocio

RN-01 Evitar NIF incorrectos en el sistema.

Como administrador.

Quiero que los usuarios estén bien identificados mediante un NIF con el formato de ocho números y una letra.

Para solventar el problema de datos erróneos en el sistema.

RN-02 Número máximo de alarmas por usuario.

Como administrador.

Quiero que la cantidad máxima de alarmas establecidas para un usuario sea de cinco.

Para solventar un exceso de alarmas residuales sin validez y posibles problemas de memoria asociados.

RN-03 Evitar horario solapado de alarmas.

Como jefe.

Quiero que el horario de las alarmas esté bien establecido para que no se solapen los horarios de trabajo.

Para que no se solapen los horarios de trabajo.

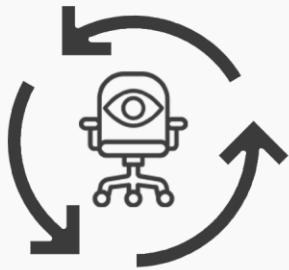
RN-04 Control de roles.

Como administrador.

Quiero que un usuario de tipo empleado tenga asociado el NIF de un jefe que se encuentre registrado en el sistema.

Para garantizar la jerarquía de la empresa dentro de nuestra aplicación.





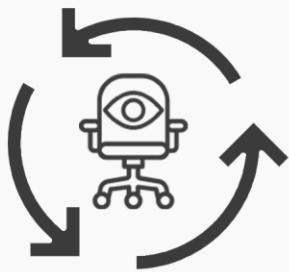
2.4 Riesgos

A continuación, se muestra la matriz de riesgos donde se describe el posible problema que se puede generar junto con la probabilidad de que este ocurra y el impacto que podría llegar a tener, por ello, se presenta un plan de contingencia.

Las probabilidades corresponden con los siguientes porcentajes:

- *Muy baja: 0-10%.*
- *Baja: 10-30%.*
- *Media: 40-60%.*
- *Alta con 60-80%.*

Descripción	Impacto	Probabilidad	Plan de contingencia
Saturación en el tráfico de datos.	Alto	Media	Se deberá de realizar un estudio de en qué parte del sistema se produce la limitación para intentar subsanarlo con los cambios pertinentes para su correcto funcionamiento.
Vulnerabilidad de los datos en la comunicación aplicación – servidor.	Medio	Baja	Se deberá de seleccionar otro Cipher Suite con mayor grado de seguridad.
Mayor dificultad de la esperada en el desarrollo del hardware.	Alto	Baja	Introducir temporalmente mayores esfuerzos temporales en la planificación correspondiente al desarrollo del dispositivo.



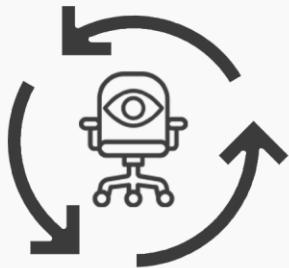
Mayor dificultad de la esperada en el desarrollo del software.	Alto	Baja	Introducir temporalmente mayores esfuerzos temporales en la planificación correspondiente al desarrollo de la aplicación.
Pruebas de funcionamiento fallidas.	Medio	Media	Se deberá replanificar la actividad para localizar el error y solventarlo.
Rotura del prototipo hardware.	Medio	Baja	Será necesaria una replanificación de la fase del proyecto para avanzar en él mientras se adquieren de nuevo los componentes para el montaje del prototipo.
Dificultad para encontrar stock de los componentes hardware necesarios.	Bajo	Baja	No es necesario ningún plan de contingencia.

Tabla 2. Matriz de riesgos

Chair Tracker

Bloque 3:

Planificación



3.1 Alcance del proyecto

Atendiendo a los requisitos anteriores podemos plantear un alcance, que establecerá el fin de la organización para este proyecto.

Este TFG se centra en el desarrollo de una aplicación móvil multiplataforma basado en React Native. Dicha aplicación constará de un desarrollo Full Stack, es decir, se abordarán desde Back-end y Front-end.

Además, se introducirán mejoras tanto código en lo que respecta a las partes desarrolladas durante la asignatura de Diseño de Aplicaciones Distribuidas, como en el dispositivo físico.

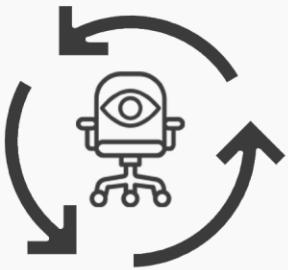
Este dispositivo físico será capaz de soportar las funcionalidades del sistema requeridas y, a su vez, será una iteración bastante cercana a un producto final, listo para su puesta en producción.

Ejecutaremos una fase de pruebas exhaustivas donde se comprobará cada bloque de código desarrollado y el sistema hardware fabricado, así como la puesta en marcha del sistema final.

Sin embargo, no se contempla la extensión del producto a otros sectores, como sugieren los planes a largo plazo de la visión del proyecto, así como ámbitos de la educación, transporte, etc.

Respecto al lanzamiento del sistema al mercado, se realizará un estudio de mercado dentro del sector laboral. Además, nos encargaremos del diseño y branding de la marca, así como la posible implantación en el mercado del producto.





3.2 Fases del proyecto

La planificación temporal se ha decidido repartir a través de fases. Estas fases están divididas mediante actividades, que a su vez engloban diversas tareas.

En total, se ha dividido el proyecto en 8 fases, 22 actividades y 32 tareas, las cuales se detallarán posteriormente.

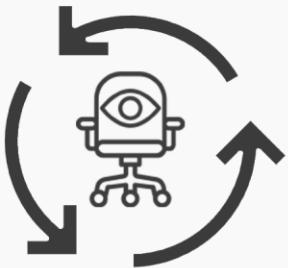
3.2.1 Fase 0: Inicio del proyecto

Esta fase inicial corresponde con los adelantos del proyecto realizados en la asignatura de Diseño de Aplicaciones Distribuidas, en la cual realizamos una primera iteración del hardware de nuestro sistema con su correspondiente Back-end.

Debemos prestar atención a que, aunque lo consideramos una fase del proyecto general, no será considerado como parte de la temporalidad del desarrollo del proyecto. Esto se puede observar en el diagrama de Gantt, pues no presentan ni fecha de inicio ni de fin.

Las actividades que consideraremos ya realizadas para este proyecto, son las siguientes:

- *A0.1 - Base de datos:* diseño, desarrollo e implementación de la base de datos en MySQL.
- *A0.2 - Servidor:* desarrollo e implementación de un servidor MQTT y HTTP mediante Vert.x.
- *A0.3 - API Rest:* desarrollo e implementación de una API Rest que conecta tanto con el cliente como con la base de datos.
- *A0.4 – Prototipo hardware:* diseño y desarrollo de un primer prototipo del circuito que se debe implementar. Así como una primera aproximación del



concepto físico que se deseará llegar a implementar para el sistema final.

- *A0.5 - Cliente:* desarrollo e implementación del firmware de la placa para poder llevar a cabo la funcionalidad básica de nuestro sistema.

3.2.2 Fase 1: Planificación del proyecto

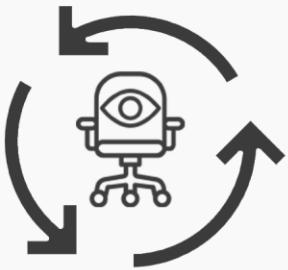
La primera fase correspondiente al proyecto trata de la planificación de este.

En esta, se realizarán análisis correspondientes a la funcionalidad del sistema, el alcance del proyecto, división temporal y de tareas, etc.

La división en actividades de la fase de planificación es la siguiente:

- *A1.1 - Análisis y viabilidad de las funciones principales del sistema.*
- *A1.2 - Estudio de las tecnologías disponibles para su desarrollo:* se realizará un análisis acerca de las tecnologías que se usarán en el proyecto.
- *A1.3 - Definición de objetivos del proyecto:* detalles acerca de los objetivos generales a partir de los problemas existentes y la solución planteada.
- *A1.4 - Análisis de requisitos:* división en requisitos de cada uno de los objetivos del proyecto. Se dividirán en generales, funcionales, no funcionales y de información. Además, tendrán que ser registrados mediante el formato de historias de usuario.
- *A1.5 - Mapa de historias de usuario:* agrupación y clasificación de los requisitos del proyecto en un mapa, según la correspondencia con objetivos generales y tipo de requisitos.
- *A1.6 - Análisis de riesgos:* análisis y clasificación de los inconvenientes que se pueden presentar a lo largo del desarrollo del sistema.





3.2.3 Fase 2: Diseño

La segunda fase corresponde al diseño general de los elementos que emplearemos a lo largo del desarrollo del proyecto. Se divide en las siguientes actividades:

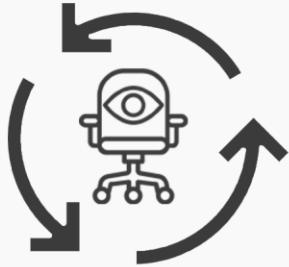
- *A2.1 – Diseño de imagen:* se definirá el diseño de una imagen corporativa y profesionalizada de todos y cada uno de los aspectos formales a utilizar, tratando de crear unas líneas de diseño a seguir.
- *A2.2 – Diseño de documentación:* diseño de plantilla de la documentación y estilos de texto utilizados, fuentes, etc.
- *A2.3 – Diseño del hardware:* se trata del diseño físico del dispositivo, incluyendo tanto visual como funcional con la selección de componentes necesarios para cumplir los objetivos y una carcasa que mantendrá las líneas de diseño anteriormente diseñadas.
 - *T2.3.1: Selección de componentes.*
 - *T2.3.2: Diseño de la PCB.*
 - *T2.3.3: Diseño 3D de la carcasa.*
- *A2.4 – Diseño de la aplicación:* realización del bocetado de la aplicación de cara a mantener las líneas de diseño y facilitar la posterior maquetación de la app.
 - *T2.4.1: Diseño de la navegación.*
 - *T2.4.2: Diseño de pantallas.*
 - *T2.4.3: Diseño de componentes.*

3.2.4 Fase 3: Estudio de mercado

Durante el desarrollo de esta fase se realizará un estudio de mercado tal y como se indica en las actividades correspondientes:

- *A3.1 – Estudio de los usuarios objetivo:* se realiza un estudio sobre los posibles usuarios que podrían llegar a estar interesados en este tipo de solución.
- *A3.2 – Estudio de la comercialización:* presentación de los posibles modelos de comercialización al que se





Chair Tracker

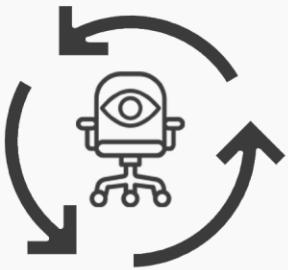
dirigirá el sistema final, teniendo en cuenta todos los tipos de mercados posibles.

- *A3.3 – Valor diferencial del producto:* comparativa entre los posibles competidores y nuestro sistema destacando cada una de las características y diferencias junto con la aportación realizada al mercado por parte de nuestro dispositivo.

3.2.5 Fase 4: Desarrollo

A lo largo de la fase 4 se realiza el desarrollo hardware y software del sistema, así como la integración de ambas partes para lograr un producto completo que provea al usuario con un dispositivo físico que cumpla con unos estándares de calidad y una aplicación acorde a los mismo para su gestión y uso.

- *A4.1 – Desarrollo del hardware:* se desarrollará la parte física del dispositivo con la PCB y carcasa diseñada para ser impresa en 3D.
 - *T4.1.1: Impresión del primer prototipo de PCB en fresadora.*
 - *T4.1.2: Ensamblaje de componentes en la PCB.*
 - *T4.1.3: Generación de gerbers y pedido en fabricante de PCBs.*
 - *T4.1.4: Ensamblaje PCB final.*
 - *T4.1.5: Impresión del diseño 3D de la carcasa.*
 - *T4.1.5: Ensamblaje del sistema hardware.*
- *A4.2 – Desarrollo de aplicación en React Native:* con el fin de ofrecer una aplicación multiplataforma que no dé de lado a ningún usuario debido a su sistema de preferencia se hará uso de la tecnología de React Native para el desarrollo de una aplicación completa e intuitiva.
 - *T4.2.1: Definición de la estructura de carpetas del proyecto.*



Chair Tracker

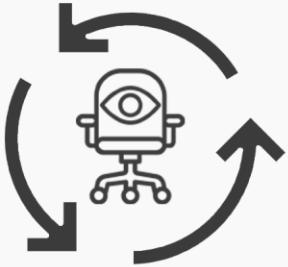
- *T4.2.2: Configuración del entorno de trabajo.*
- *T4.2.3: Desarrollo de las pantallas.*
- *T4.2.4: Desarrollo de la navegación.*
- *T4.2.5: Desarrollo de componentes.*
- *T4.2.6: Conexión con API Rest.*
- *T4.2.7: Definición del tipado.*
- *T4.2.8: Desarrollo e implementación de la lógica.*
- *T4.2.9: Mejoras de código.*

3.2.6 Fase 5: Prueba y test

Se llevarán a cabo diferentes tests para el aseguramiento de la calidad de ambas partes principales del desarrollo con el objetivo de alcanzar estabilidad general en el sistema y a lo largo de su uso.

- *A5.1 – Pruebas hardware*
 - *T5.1.1: Test de circuito en PCB de prueba.*
 - *T5.1.2: Funcionamiento del circuito integrado.*
 - *T5.1.3: Test de conexionado en PCB final.*
 - *T5.1.4: Funcionamiento del sistema.*
 - *T5.1.5: Medidas de voltaje e intensidad de los componentes en el sistema final.*
 - *T5.1.6: Test de temperatura.*
 - *T5.1.7: Test de caída.*
 - *T5.1.8: Test de duración de la batería.*
- *A5.2 – Pruebas software*
 - *T5.2.1: Test unitarios de la funcionalidad del sistema.*
 - *T5.2.2: Comprobaciones de errores de maquetación.*





3.2.7 Fase 6: Cierre del proyecto

Se valorará el cómputo general del proyecto, así como posibles mejoras de cara al futuro y una valoración final del desarrollo del proyecto.

3.2.8 Documentación

Durante las fases citadas anteriormente se realizará la documentación pertinente de forma concurrente según la finalización de los hitos, descritos en el siguiente punto.

3.3 Hitos del proyecto

A largo de este punto se detallan los hitos del proyecto y sus respectivas actividades asociadas, como forma de evaluación del progreso.

3.3.1 Hito 1: Planificación del proyecto

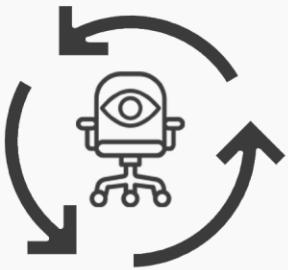
El primer hito se corresponde con la finalización de la primera fase del proyecto donde se evalúan las tareas completadas referentes a la funcionalidad del sistema junto con un análisis de las tecnologías disponibles en el mercado y porque se ha optado por la seleccionada.

Las tareas asociadas son las siguientes: *A1.1, A1.2, A1.3, A1.4, A1.5, A1.6*.

3.3.2 Hito 2: Diseño de documentación y Hardware

En este hito se evalúa tanto el diseño de la PCB con la selección correspondiente de sus componentes y el diseño de la carcasa en 3D como el diseño de las líneas de diseño generales del proyecto para generar una imagen identificativa y la correspondiente documentación.





Las actividades que forman parte de este hito son:
A2.1, A2.2, A2.3 (T2.3.1, T2.3.2, T2.3.3), A2.4 (T2.4.1, T2.4.2, T2.4.3, T2.4.4)

3.3.3 Hito 3: Diseño Software

En este hito se realiza una evaluación del diseño referente a la aplicación destacando sobre todo la realización de los mock-ups que muestran tanto su aspecto visual como el flujo de navegación entre diferentes pantallas. Corresponde con la finalización de la segunda fase de la planificación.

La actividad que forma parte de este hito es: **A2.4 (T2.4.1, T2.4.2, T2.4.3, T2.4.4)**

3.3.4 Hito 4: Estudio de mercado y revisión de la documentación

Se lleva a cabo la evaluación del estudio de mercado realizado que ayudará a dar idea del sector al que va dirigido el producto y posteriormente se realizará una revisión de la documentación aportada hasta el momento. Su finalización es coincidente con el fin de la fase 3.

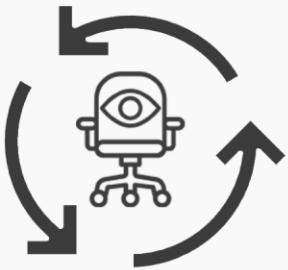
Las actividades que forman parte de este hito son:
A3.1, A3.2, A3.3.

3.3.5 Hito 5: Desarrollo del Hardware

El quinto hito de este proyecto se corresponde con el desarrollo hardware. A lo largo de este hito, se irán generando entregables correspondientes con las actividades más relevantes de la fase.

Las actividades que forman parte de este hito son:
A4.1 (T4.1.1, T4.1.2, T4.1.3, T4.1.4, T4.1.5).





3.3.6 Hito 6: Desarrollo de la aplicación

El siguiente hito tratará del fin del desarrollo de la aplicación. También durante este hito se crearán iteraciones del código como medio para recibir feedback durante el proceso de desarrollo, en lugar de evaluar todo el software una vez completo.

Las actividades que forman parte de este hito son:
A4.2 (T4.2.1, T4.2.2, T4.2.3, T4.2.4, T4.2.5, T4.2.6, T4.2.7, T4.2.8, T4.2.9).

3.3.7 Hito 7: Pruebas

En este hito se evaluarán las pruebas realizadas sobre nuestro proyecto, tanto en la parte hardware como en la parte software. Tendrá la finalidad de cerciorarnos de la calidad del sistema, con la que se dará por concluido el desarrollo.

Las actividades que forman parte de este hito son:
A5.1 (T5.1.1, T5.1.2, T5.1.3, T5.1.4, T5.1.5, T5.1.6, T5.1.7, T5.1.8), A5.2 (T5.2.1, T5.2.2)

3.3.8 Hito 8: Cierre del proyecto

Este es el último punto evaluable antes de realizar la entrega final, donde se revisará tanto el desarrollo del proyecto como la documentación generada.

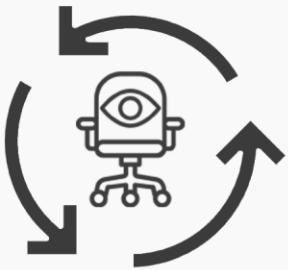
Las actividades que forman parte de este hito son: *A6*

Chair Tracker

3.4 Calendario de entregables

En cada entregable se mostrará una actualización del estado del proyecto, así como la documentación, actividades y tareas realizadas, elementos del desarrollo software o hardware, etc.





Las fechas propuestas para realizar estas revisiones son las siguientes:

- *1 de noviembre de 2021*
- *29 de enero de 2022*
- *27 de febrero de 2022*
- *15 de abril de 2022*
- *30 de mayo de 2022*
- *10 de junio de 2022*

3.5 Diagrama de Gantt

Para poder realizar la planificación temporal se han realizado un diagrama de Gantt donde se recogen todas las fases, actividades y tareas junto con el tiempo planificado para cada una de ellas.

A continuación, se muestra una imagen donde se aprecia la planificación en un plano general y que da idea de la duración total del proyecto y cada una de sus fases.

Tras esto, se muestra el diagrama de cada una de estas fases en mayor profundidad, donde podemos observar cada una de las tareas.

Chair Tracker





Figura 2. Visión general fases planificación

Chair Tracker

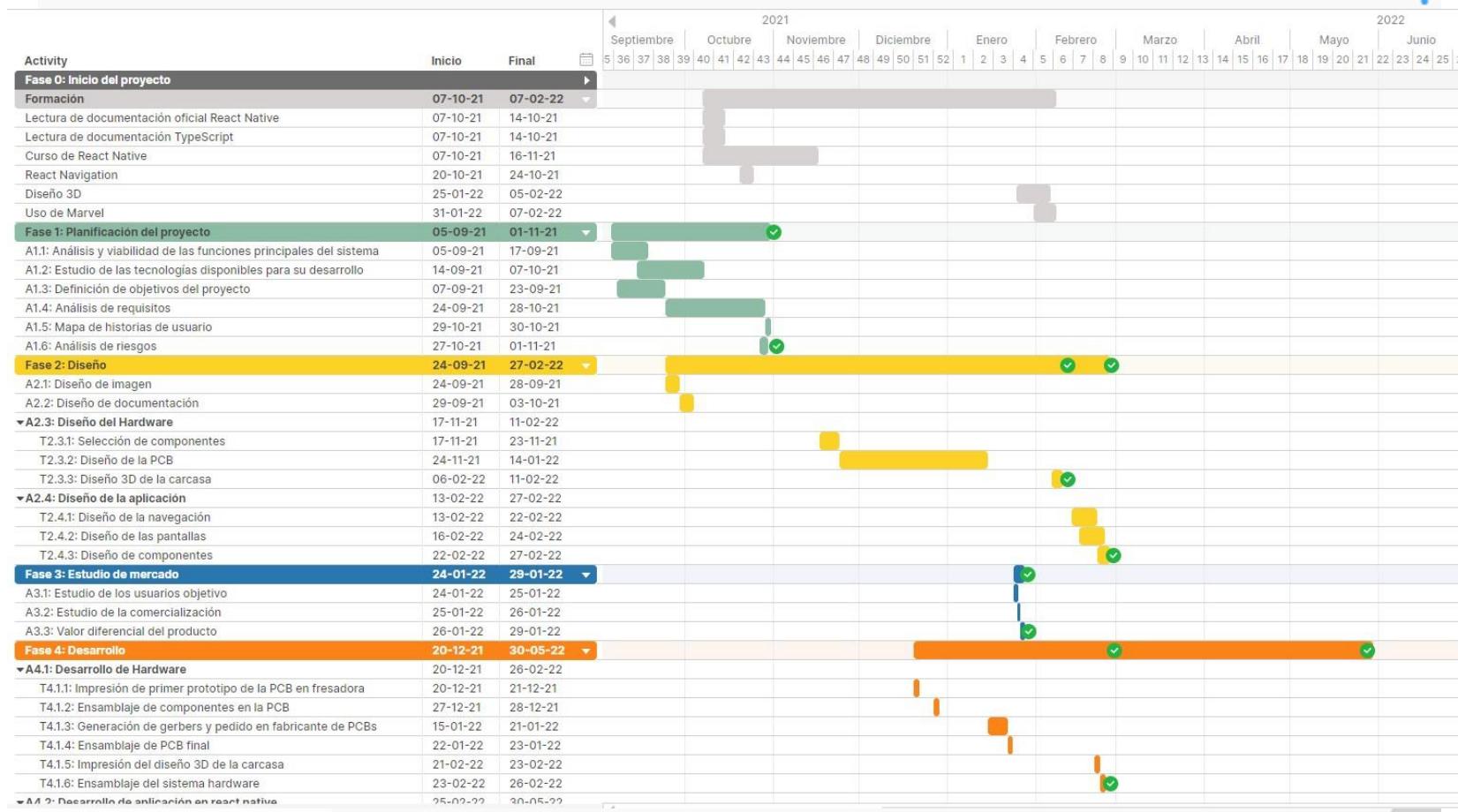


Figura 3. Desglose de la planificación I

Chair Tracker

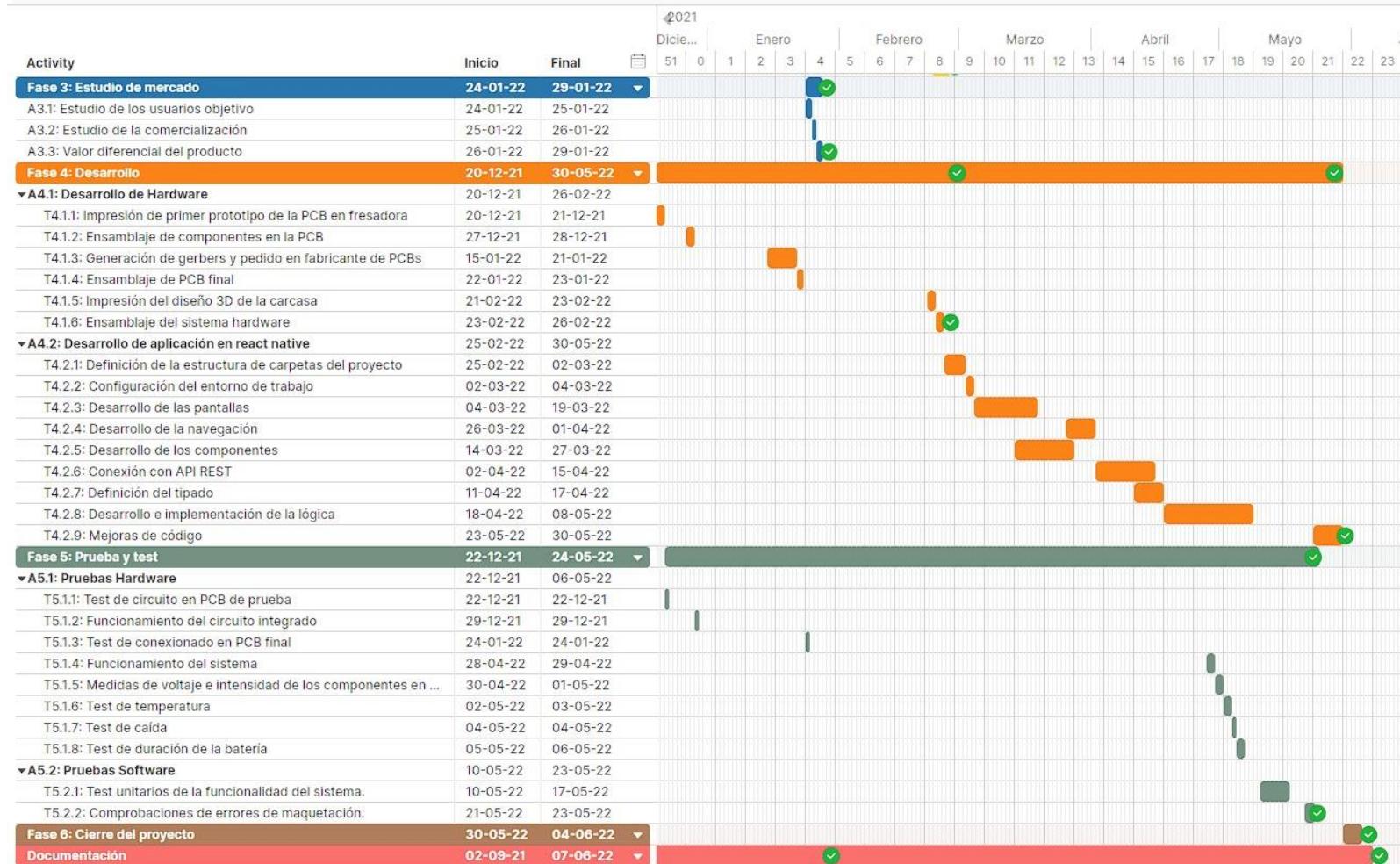


Figura 4. Desglose de la planificación II

Chair Tracker

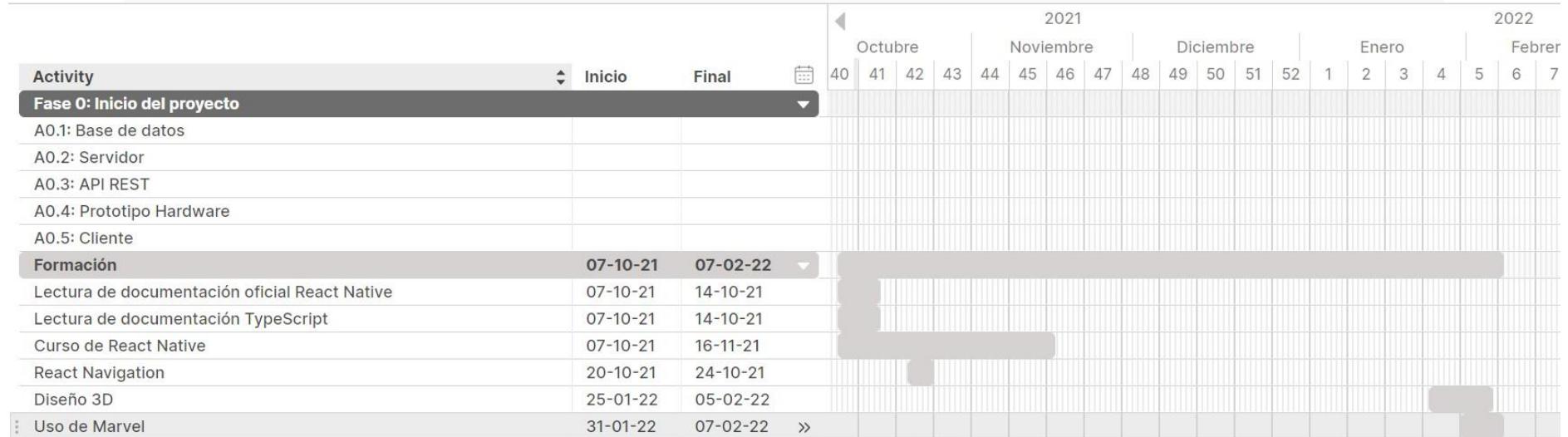


Figura 5. Fase 0 de la planificación

Chair Tracker

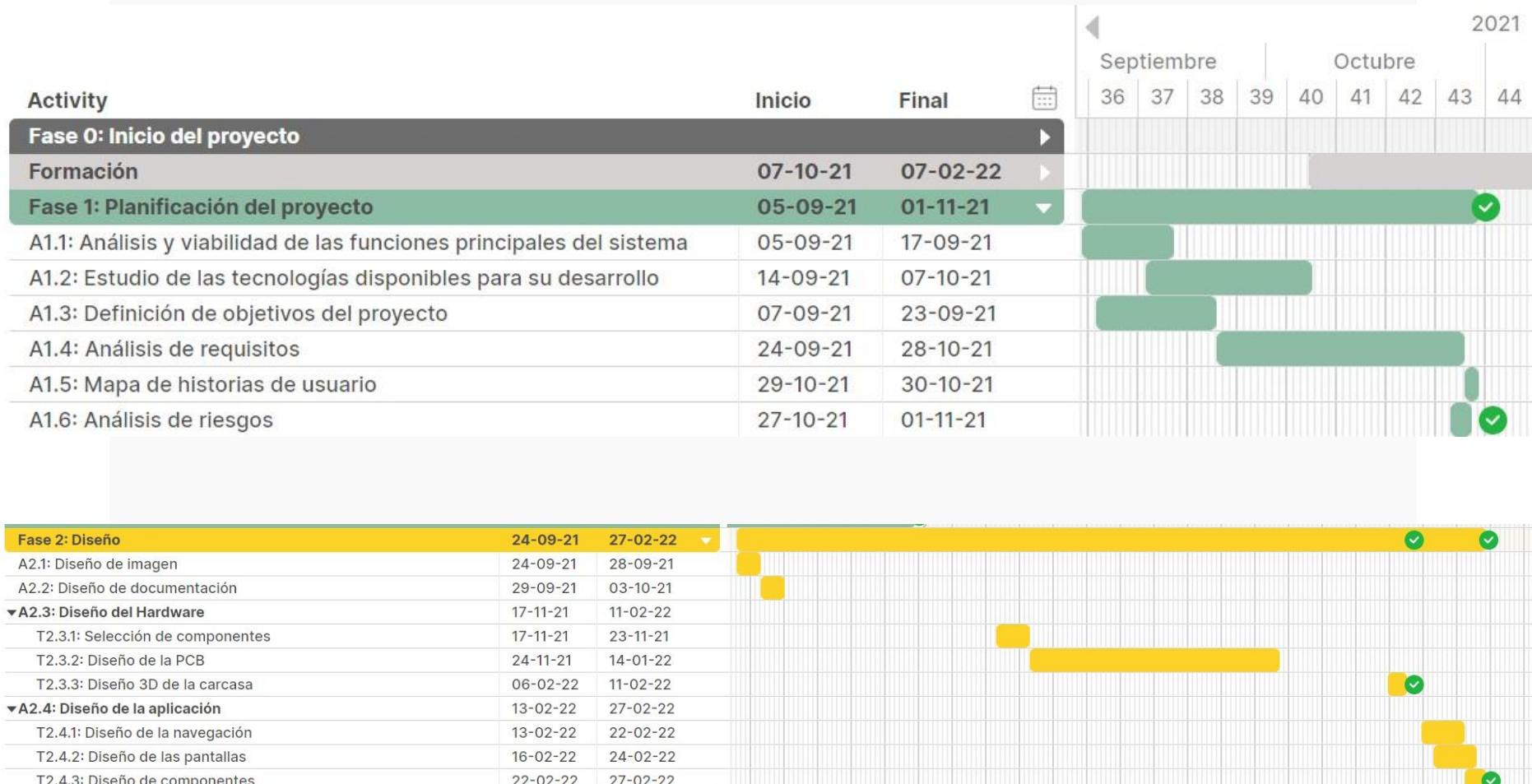


Figura 6. Fase 1 y 2 de la planificación

Chair Tracker

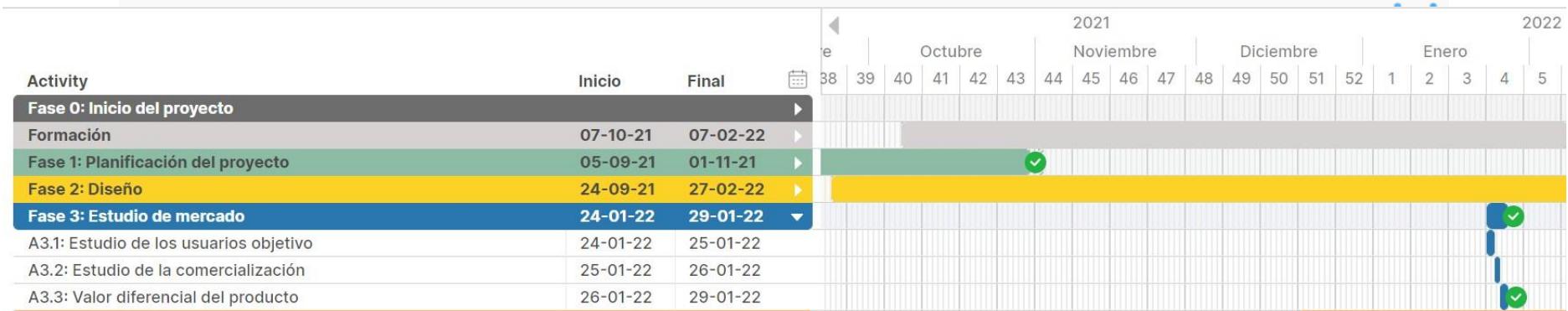


Figura 7. Fase 3 de la planificación

Chair Tracker

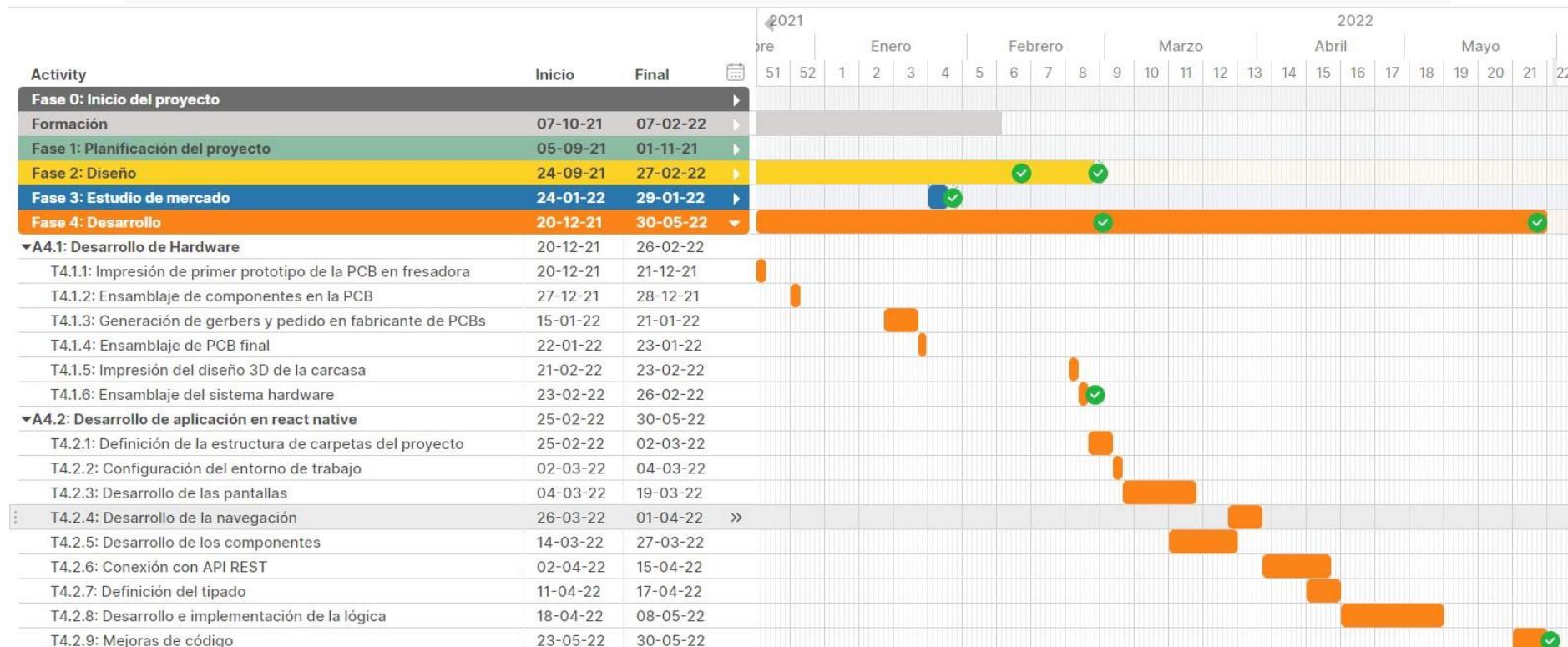


Figura 8. Fase 4 de la planificación

Chair Tracker

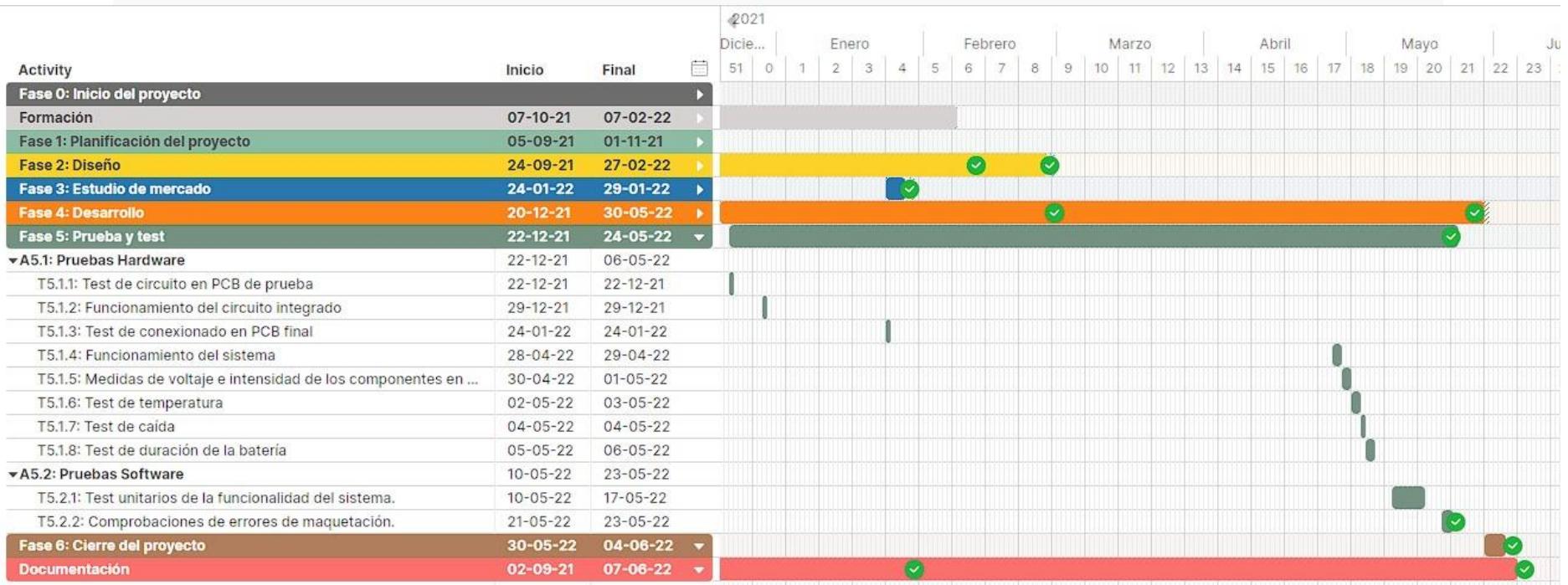
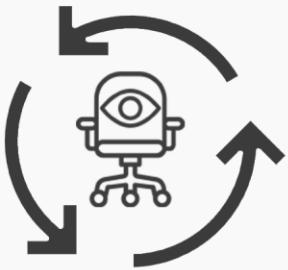


Figura 9. Fase 5, 6 y Documentación de la planificación

Chair Tracker



3.6 Equipo planteado

El equipo planteado queda definido mediante el siguiente diagrama. Constará de 5 departamentos principales con 7 trabajadores en total. Aunque esta división se trata de una estimación de los recursos humanos para la realización del proyecto, realmente ambos integrantes de este TFG nos hemos repartido las tareas equitativamente a lo largo de las fases, para poder así, adquirir conocimiento en cada una de las ramas que se tratan en la elaboración del proyecto, como lo son: diseño, desarrollo software, desarrollo hardware, marketing, etc.

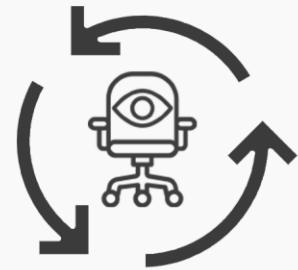
El departamento de gestión de proyecto será el encargado de planificar, gestionar y controlar las actividades planteadas a lo largo del desarrollo, así como generar la documentación pertinente en cada fase según el estado en el que se encuentre el proyecto.

El departamento de diseñadores tendrá como labor desarrollar las guías de líneas de diseño, que se utilizarán para crear una imagen de marca, la aplicación móvil y el modelado 3D de la carcasa del sistema.

El departamento de desarrolladores software tratará aquellas actividades relacionadas con el desarrollo de la aplicación móvil y de implementar las mejoras en el Back-end.

El departamento de desarrolladores hardware estará centrado en el desarrollo y fabricación de la PCB y montaje de componentes del sistema físico, pudiendo solucionar errores futuros del firmware de la placa.

Por último, los expertos en marketing se encargarán de realizar un estudio de mercado, donde se analizará el estado del arte, los productos competidores, análisis de



coste del proyecto, viabilidad, rentabilidad, ganancias, etc., tanto antes del desarrollo como en la posterior simulación de la implementación del producto en el mercado.

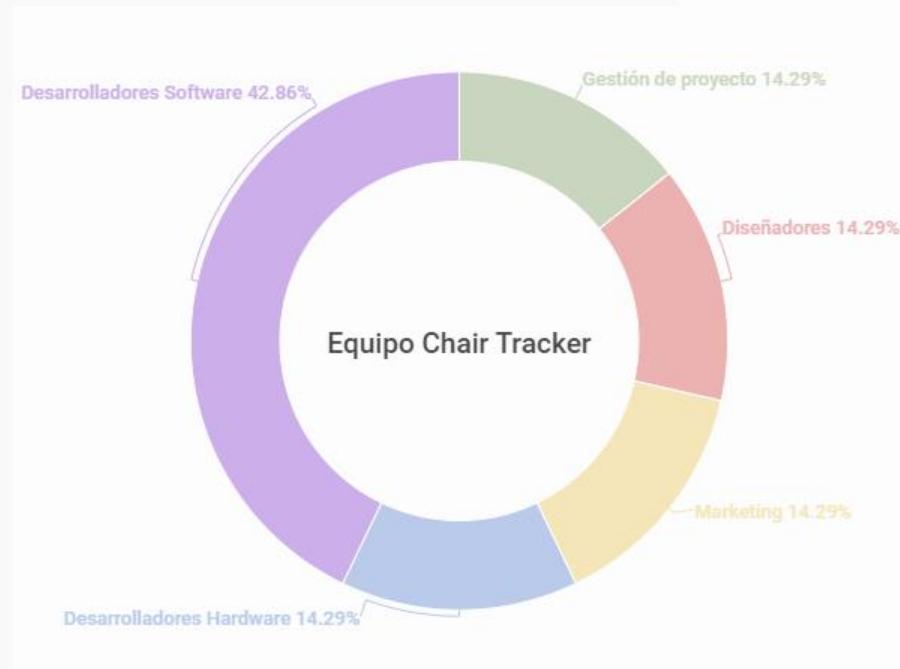


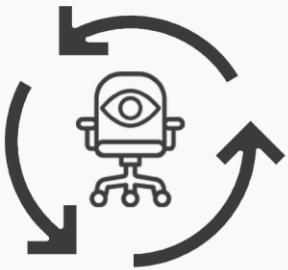
Figura 10. Desglose del equipo de desarrollo del proyecto

Chair Tracker

3.7 Presupuesto del proyecto

Se adjunta un desglose presupuestario de acuerdo con los gastos pertinentes que corresponde con la total ejecución de este proyecto.

Concepto	Cantidad	Importe total
Kit de desarrollo	x01	16.99€
ESP8266	x02	7.99€
Mini interruptor deslizante de 3 pines SS-12D00G3	x02	0.40€
Módulo de sensor ultrasónico HC-SR04 con soporte	x02	9.99€
Cables DuPont	x01	7.99€
18650 Battery Shield	x01	12.54€



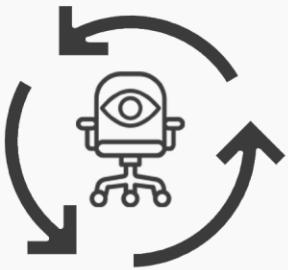
Batería litio Samsung ICR 18650 26J (x4)	x01	21.44€
Motor vibrador ADN1903071	x02	15.98€
Batería LIPO 2400mA	x01	4.73€
TP4056 USB-C	x02	0.68€
Kit de conectores DuPont y pines macho/hembra	x01	3.97€
Cepillo de desoldaduras	x01	1.33€
Alambre de soldadura	x01	5.85€
Soldador TS100	x02	65.98€
PCB	x10	9.37€
Carcasa	x02	8€
Total: 185.23€		

Tabla 3. Tabla presupuesto hardware

El presupuesto del software es inexistente debido a que ya se disponen de todas las herramientas que se estipulan para el desarrollo o, por el contrario, son Open Source.

A continuación, se estimarán el presupuesto necesario en recursos humanos, suponiendo un escenario en el cual una empresa real se hiciera cargo de la ejecución del proyecto (bajo las horas de trabajo estimadas para la ejecución de un TFG).

Chair Tracker

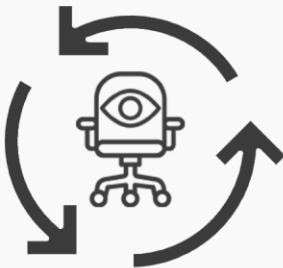


Gestión de proyecto			
Cargo	Nº horas	€/hora	Coste
Junior Project Manager	300	10.41	3123€
Diseñadores			
Cargo	Nº horas	€/hora	Coste
Junior Designer	30	6.50	195€
Desarrolladores Software			
Cargo	Nº horas	€/hora	Coste
Junior React Native Developer (1)	75	7.00	525€
Junior React Native Developer (2)	75	7.00	525€
Junior Back-end Developer (1)	50	6.60	330€
Desarrolladores Hardware			
Cargo	Nº horas	€/hora	Coste
Junior Hardware Developer	50	7.20	360€
Expertos en marketing			
Cargo	Nº horas	€/hora	Coste
Junior marketing Agent	20	6.60	330€
Presupuesto total			5388€

Tabla 4. Tabla presupuesto personal

Chair Tracker

Bloque 4: Análisis tecnológico



4.1 Herramientas utilizadas

Las herramientas utilizadas a lo largo de la ejecución del proyecto han sido clasificadas y explicadas en relación con su ámbito de uso, de la siguiente forma:

4.1.1 Planificación

El diagrama de Gantt ha sido generado a través de la herramienta Tom's Planner.

"Tom's Planner es un software de Diagrama de Gantt que se utiliza para la planificación de proyectos y que utiliza un interfaz de usuario muy intuitiva que permite la creación de calendarios de proyectos en línea, además de poder compartirlos y hacerlos colaborativos." [\[19\]](#)



Figura 11. Logo de Tom's planner

4.1.2 Diseño

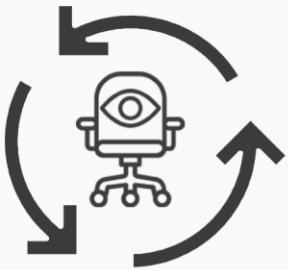
Para el diseño existente en la documentación, como el de todos los logos, portadas y marca de agua para las páginas hemos usado Adobe Photoshop CC y Adobe Illustrator CC.

"Adobe Photoshop es un editor de fotografías desarrollado por Adobe Systems Incorporated. Usado principalmente para el retoque de fotografías y gráficos, su nombre en español significa "taller de fotos"." [\[3\]](#)

Chair
Tracker



Figura 12. Logo de Photoshop



"Adobe Illustrator es un editor de gráficos vectoriales sirve para editar entre otras cosas. Es desarrollado y comercializado por Adobe Systems y constituye su primer programa oficial de su tipo en ser lanzado por esta compañía definiendo en cierta manera el lenguaje gráfico contemporáneo mediante el dibujo vectorial."[\[2\]](#)



Figura 13. Logo de Illustrator

4.1.3 Base de datos

En relación con el desarrollo de la base de datos hemos utilizado MySQL Workbench 8.0 CE.

"MySQL Workbench es una herramienta visual unificada para el diseño y desarrollo de arquitecturas de base de datos. MySQL Workbench ofrece modelado de datos, desarrollo SQL, además de herramientas de administración y configuraciones de servidor y mucho más."[\[23\]](#)

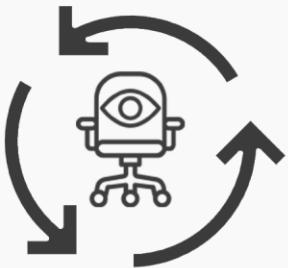


Chair
Tracker

Figura 14. Logo de MySQL Workbench

4.1.4 API

Atendiendo al desarrollo de la API Rest han sido necesarias las siguientes herramientas: Eclipse 2020-12, Postman y MQTT Explorer.



"Eclipse es una plataforma de software compuesto por un conjunto de herramientas de programación de código abierto multiplataforma para desarrollar lo que el proyecto llama "Aplicaciones de Cliente Enriquecido", opuesto a las aplicaciones "Cliente-liviano" basadas en navegadores." [\[12\]](#)



Figura 15. Logo de Eclipse

"Postman es una plataforma API dedicada al desarrollo y uso de APIs. Postman simplifica cada paso del ciclo de vida de la API y aerodinamiza la colaboración para poder crear mejores APIs." [\[25\]](#)



Figura 16. Logo de Postman

"MQTT Explorer es un cliente MQTT intuitivo que ofrece una visión general estructurada de tus temas MQTT y hace que trabajar con tus dispositivos/servicios en tu bróker sea simple." [\[24\]](#)

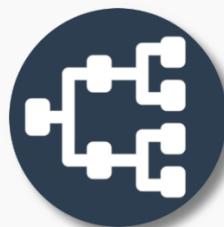
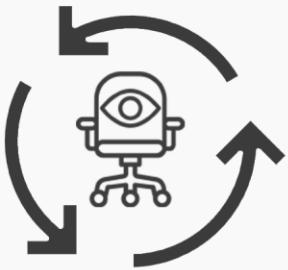


Figura 17. Logo de MQTT Explorer

Chair Tracker



4.1.5 Cliente

Para el firmware del ESP8266 se ha utilizado Visual Studio Code con la extensión de Platform.io.

"Visual Studio Code es un editor de código fuente ligero pero poderoso que funciona en cualquier sistema operativo. Incluye soporte para JavaScript, TypeScript y Node.js y tiene un rico ecosistema de extensiones para otros lenguajes." [38]

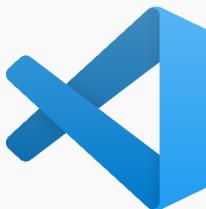


Figura 18. Logo de Visual Studio Code

"Platform.io es una herramienta multiplataforma, multi arquitectura y multi framework para ingenieros de sistemas empotrados y desarrolladores software que realizan aplicaciones para sistemas embebidos." [41]

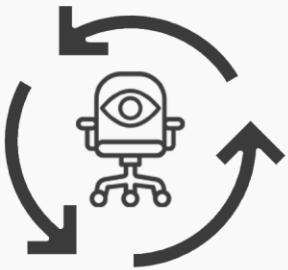


Figura 19. Logo de Platform.io

4.1.6 Boceto

Para la realización de los bocetos de la aplicación móvil se ha utilizado Marvel App.

"Es una herramienta web de diseño. Se creó esta plataforma de desarrollo online pensando en todo el mundo, dando como resultado una herramienta de diseño simple e intuitiva que permite a todos crear rápidamente bonitos recursos y bocetos. No hay curva de



aprendizaje y tampoco se requiere de ningún software adicional.”[\[21\]](#).



Figura 20. Logo de Marvel App

4.1.7 App

Se ha utilizado Xcode, Android Studio y Visual Studio Code para el desarrollo completo de la aplicación móvil multiplataforma. Dentro de Visual Studio Code ha sido necesario el uso de ESLint para poder generar un código final lo más profesionalizado posible.

“Xcode incluye todo lo necesario para el desarrollo dentro del ecosistema de Apple. Ofrece a los desarrolladores un flujo de trabajo unificado para el diseño de la interfaz de usuario, codificación, testeо y depuración.”[\[4\]](#)

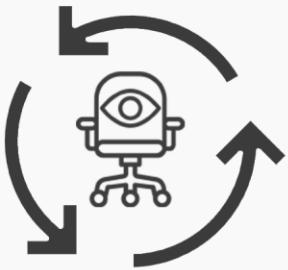


Figura 21. Logo de Xcode

Chair Tracker

“Android Studio es el entorno de desarrollo integrado (IDE) oficial para el desarrollo de apps para Android y está basado en IntelliJ IDEA. Además del potente editor de códigos y las herramientas para desarrolladores de IntelliJ, Android Studio ofrece incluso más funciones que





aumentan tu productividad cuando desarrollas apps para Android.”[\[18\]](#)



Figura 22. Logo de Android Studio

“ESLint es una herramienta de análisis de código estático para identificar patrones problemáticos encontrados en el código JavaScript o TypeScript.”[\[13\]](#)



Figura 23. Logo de ESLint

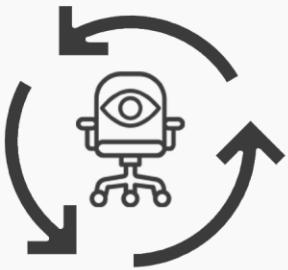
4.1.8 PCB

Usamos KiCad EDA para el diseño completo de la PCB.

“Es una herramienta multiplataforma y de código abierto que permite la automatización de los diseños electrónicos, incluyendo un editor de esquemático, un editor de PCB (Layout) y un visor 3D.”[\[31\]](#).



Figura 24. Logo de KiCad



4.1.9 Carcasa

Para la elaboración del diseño 3D de la carcasa de nuestro sistema utilizaremos Fusion 360.

"Fusion 360 es una plataforma de software de modelado 3D, CAD, CAM y PCB basada en la nube destinada al diseño y la fabricación de productos." [\[10\]](#)



Figura 25. Logo de Autodesk fusión 360

4.1.10 Control de versiones

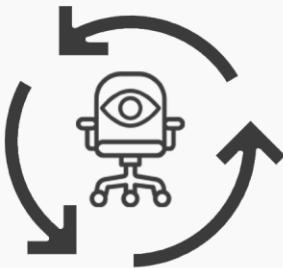
La herramienta necesaria para llevar a cabo un desarrollo dentro de un entorno seguro, además de fomentar la agilidad dentro del flujo de trabajo ha sido Github.

"GitHub es un fork para alojar proyectos utilizando el sistema de control de versiones Git. Se utiliza principalmente para la creación de código fuente de programas de ordenador." [\[16\]](#)



**Chair
Tracker**

Figura 26. Logo de Github



4.2 Lenguajes y plataformas

Para el desarrollo tecnológico de este proyecto, se han utilizado 4 lenguajes de programación, que son los siguientes:

4.2.1 SQL

El lenguaje utilizado para la elaboración de nuestra base de datos relacional ha sido SQL.

"SQL es un lenguaje de dominio específico, diseñado para administrar, y recuperar información de sistemas de gestión de bases de datos relacionales." [9].



Figura 27. Logo de SQL

4.2.2 C

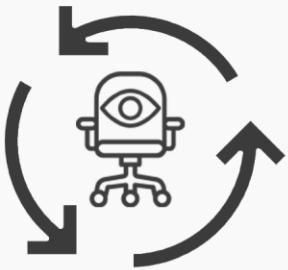
El firmware que lleva incorporado el ESP8266 ha sido desarrollado bajo el lenguaje de programación C.

"C es un lenguaje de programación de propósito general originalmente desarrollado por Dennis Ritchie entre 1969 y 1972 en los Laboratorios Bell, como evolución del anterior lenguaje B, a su vez basado en BCPL. Al igual que B, es un lenguaje orientado a la implementación de sistemas operativos, concretamente Unix." [6]

Chair
Tracker



Figura 28. Logo de C



4.2.3 Java

El desarrollo completo de la API Rest de nuestro sistema está realizado en Java.

"Java es un lenguaje de programación y una plataforma informática comercializada por primera vez en 1995 por Sun Microsystems. Hay muchas aplicaciones y sitios web que no funcionarán a menos que tenga Java instalado y cada día se crean más. Java es rápido, seguro y fiable. Desde portátiles hasta centros de datos, desde consolas para juegos hasta súper computadoras, desde teléfonos móviles hasta Internet, Java está en todas partes." [\[40\]](#)



Figura 29. Logo de Java

4.2.4 TypeScript

La aplicación móvil multiplataforma combina dos lenguajes de programación a lo largo de su desarrollo, uno de ellos es TypeScript.

"TypeScript es un lenguaje de programación libre y de código abierto desarrollado y mantenido por Microsoft. Es un superconjunto de JavaScript, que esencialmente añade tipos estáticos y objetos basados en clases." [\[36\]](#)

**Chair
Tracker**

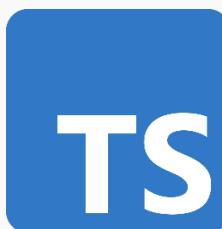
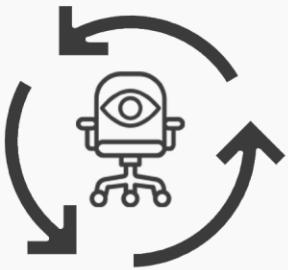


Figura 30. Logo de TypeScript



4.2.5 React Native

El otro lenguaje de programación en el que se basa nuestra aplicación corresponde con React Native.

"React Native es un marco de software de interfaz de usuario de código abierto creado por Meta Platforms, Inc. Se utiliza para desarrollar aplicaciones para Android, Android TV, iOS, macOS, tvOS, Web, Windows y UWP al permitir a los desarrolladores usar el marco React junto con capacidades de la plataforma nativa." [\[29\]](#)

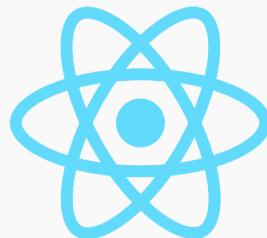
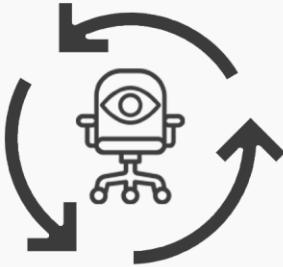


Figura 31. Logo de React Native

**Chair
Tracker**

Bloque 5: Diseño e Implementación



5.1 Software

En este apartado, explicaremos tanto la fase de diseño como el resultado del desarrollo de toda la parte software de nuestro sistema. En el directorio del proyecto se ofrecerá todo el código implementado para poder tener un nivel mayor de detalle sobre la codificación de las decisiones tomadas a lo largo de las diferentes fases.

5.1.1 Diseño arquitectura empleada

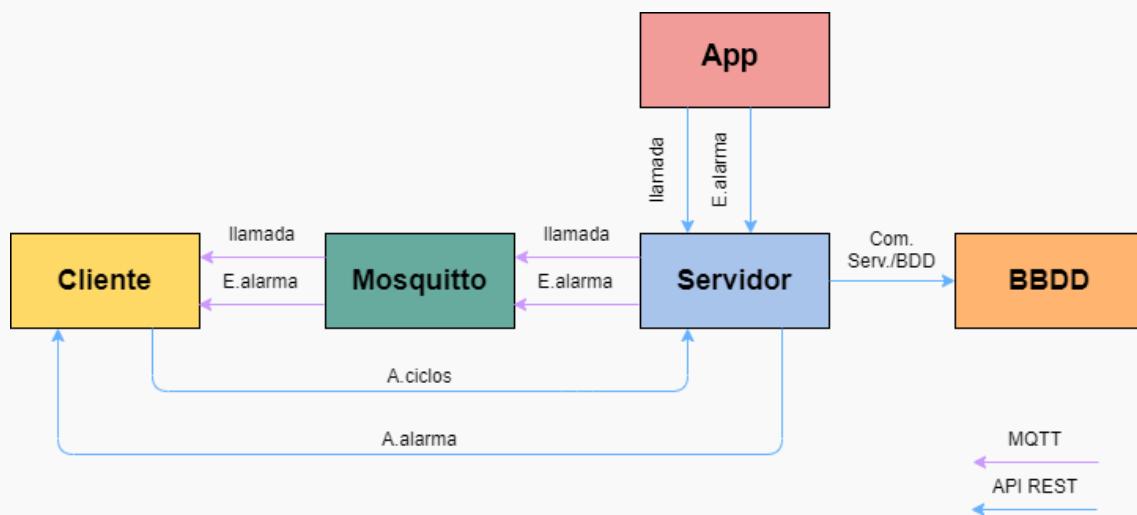
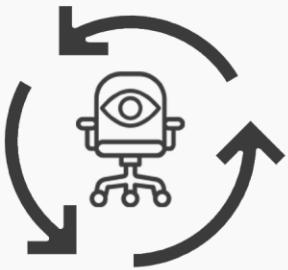


Figura 32. Arquitectura completa de la aplicación

El flujo de datos tiene comienzo desde la interfaz de la app que desarrollaremos, desde esta el usuario podrá establecer alarmas o realizar llamadas a otro usuario.

En el caso de que se establezca una alarma desde la web, se hará uso de la API REST alojada en el servidor que introducirá los datos en la BBDD, tras esto, el cliente realizará un get, el cual es el método obtenerAlarmasUsuario() de la API REST para obtener la lista actualizada de alarmas y poder calcular la próxima alarma con el método obtenerProximaAlarma(), una vez activa el cliente se encargará de realizar los cálculos



temporales para más tarde, al finalizar la alarma, actualizar los datos en la BDD a través de la API REST del server con el método editarAlarma().

Cuando finalice la alarma se volverá a obtener la próxima o en caso de no estar activa ninguna de ellas, recibiremos un mensaje MQTT para actualizar la lista del cliente a la que se encuentra en la BBDD.

Cuando hacemos una llamada desde la app mediante la API REST, nuestro servidor publicará un mensaje en Mosquitto que recibirá nuestro cliente para activar el motor vibrador.

5.1.2 Base de datos

La base de datos para nuestro sistema es bastante simple, debido a que la única información que deberemos de almacenar será la de usuarios, placas que se le asocian y sus respectivas alarmas y llamadas, que además dispondrán de registros para llevar a cabo un historial y seguimiento de estas.

A continuación, se detallará el diagrama UML que implementa las tablas, propiedades y relaciones necesarias para este sistema.

También se han tenido en cuenta las reglas de negocio expuestas anteriormente mediante triggers, procedures y checkeos de reglas.

Chair Tracker



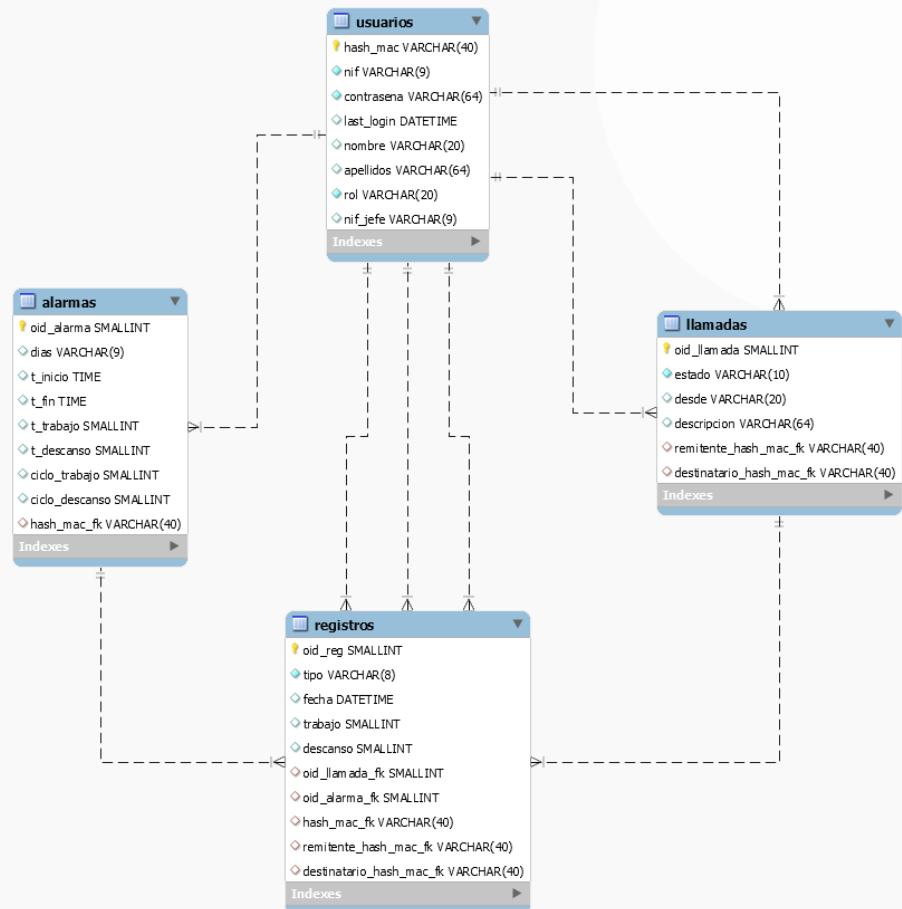
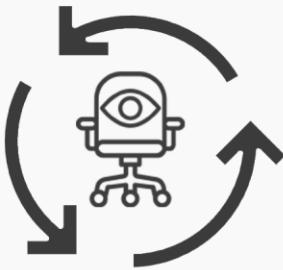


Figura 33. Esquema de la base de datos

5.1.3 Verticles

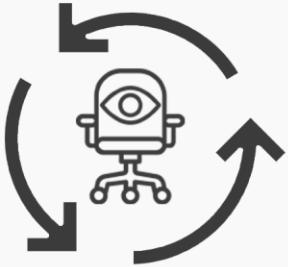
Hemos tomado la decisión de desplegar tres verticles. Uno dedicado al servidor HTTP, otro a la conexión a la BBDD y un último dedicado a ser cliente MQTT de nuestro broker Mosquitto.

Estos tres verticles se comunicarán a través de paso de mensajes mediante el bus de eventos.

Chair Tracker

5.1.4 API

La API se ejecuta en el servidor y cuenta con los métodos necesarios para la total comunicación con la BBDD con métodos GET, PUT, POST, Delete:



Métodos GET

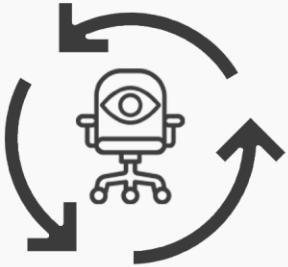
Tenemos métodos de obtención de los datos de cada tabla en su totalidad, además, hemos implementado otros métodos para obtener datos relacionados con el usuario que nos serán de utilidad a la hora de realizar peticiones desde el cliente.

Los métodos son los siguientes:

- obtenerUsuarios
 - Routing:
`/api/usuarios/`
 - Entrada: json vacío
 - Salida:

```
"1":{  
    "hash_mac": "valor",  
    "nif": " valor",  
    "contrasena": " valor ",  
    "last_login": valor,  
    "nombre": " valor ",  
    "apellidos": " valor ",  
    "rol": " valor ",  
    "nif_jefe": valor  
},
```
- obtenerAlarma
 - Routing:
`/api/alarmas/`
 - Entrada: json vacío
 - Salida:

```
1:{  
    "oid_alarma": 1,  
    "dias": " valor ",  
    "t_inicio": " valor ",  
    "t_fin": "valor",  
    "t_trabajo": valor,  
    "t_descanso": valor,  
    "ciclo_trabajo": valor,  
    "ciclo_descanso": valor,  
    "hash_mac_fk": "valor"  
},
```



Chair Tracker

- obtenerAlarmasUsuario
 - Routing:
`/api/alarmas/hash_mac`
 - Entrada:

```
{  
  "hash_mac_fk": " mac del usuario a  
  observar "  
}
```
 - Salida:

```
"1":{  
  "oid_alarma": 1,  
  "dias": " valor ",  
  "t_inicio": " valor ",  
  "t_fin": "valor",  
  "t_trabajo": valor,  
  "t_descanso": valor,  
  "ciclo_trabajo": valor,  
  "ciclo_descanso": valor,  
  "hash_mac_fk": "valor"  
},
```

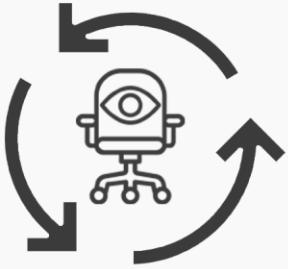
- obtenerLlamadas
 - Routing: json vacío
`/api/llamadas/`
 - Entrada: json vacío
 - Salida:

```
"1":{  
  "oid_llamada": 1,  
  "estado": "valor",  
  "desde": " valor ",  
  "descripcion": " valor",  
  "remitente_hash_mac_fk": " valor ",  
  "destinatario_hash_mac_fk": " valor "  
},
```

- obtenerLlamadasEnviadasUsuario
 - Routing:
`/api/llamadas/enviadas/hash_mac`
 - Entrada:

```
{  
  "remitente_hash_mac_fk": " mac del  
  usuario a observar "  
}
```
 - Salida:

```
"1":{
```



```
"oid_llamada": 1,  
"estado": "valor",  
"desde": " valor ",  
"descripcion": " valor",  
"remitente_hash_mac_fk": " mac del  
usuario a observar ",  
"destinatario_hash_mac_fk": " valor "  
}
```

- obtenerLlamadasRecibidasUsuario

- Routing:

```
/api/llamadas/recibidas/hash_mac
```

- Entrada:

```
{  
  "destinatario_hash_mac_fk": " mac del  
  usuario a observar "  
}
```

- Salida:

```
"1":{  
  "oid_llamada": 1,  
  "estado": "valor",  
  "desde": " valor ",  
  "descripcion": " valor",  
  "remitente_hash_mac_fk": " valor "  
  "destinatario_hash_mac_fk": " mac del  
  usuario a observar "  
}
```

- obtenerRegistros

- Routing: json vacío

```
/api/registros
```

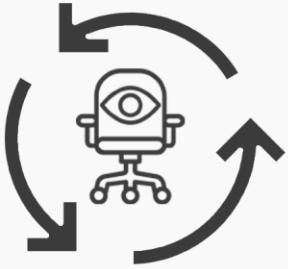
- Entrada: json vacío

- Salida:

```
"1":{ "oid_reg": 1,  
      "tipo": "valor",  
      "fecha": "2021-05-23T21:36:52",  
      "trabajo": 2910,  
      "descanso": 980,  
      "oid_llamada_fk": null,  
      "oid_alarma_fk": 1,  
      "hash_mac_fk": "mac123",  
      "remitente_hash_mac_fk": null,  
      "destinatario_hash_mac_fk": null  
}
```

Chair Tracker





Chair Tracker

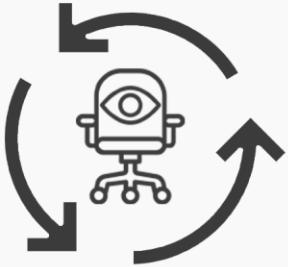
- obtenerRegistrosLlamadas
 - Routing: json vacío
</api/registros/llamadas>
 - Entrada: json vacío
 - Salida:

```
"4": {  
    "oid_reg": 4,  
    "tipo": "valor",  
    "fecha": "valor",  
    "trabajo": null,  
    "descanso": null,  
    "oid_llamada_fk": 1,  
    "oid_alarma_fk": null,  
    "hash_mac_fk": null,  
    "remitente_hash_mac_fk": "valor",  
    "destinatario_hash_mac_fk": "valor"  
}
```
- obtenerRegistrosAlarmas
 - Routing: json vacío
</api/registros/alarmas>
 - Entrada: json vacío
 - Salida:

```
"1": {  
    "oid_reg": 1,  
    "tipo": "A",  
    "fecha": "2021-05-23T21:36:52",  
    "trabajo": 2910,  
    "descanso": 980,  
    "oid_llamada_fk": null,  
    "oid_alarma_fk": 1,  
    "hash_mac_fk": "mac123",  
    "remitente_hash_mac_fk": null,  
    "destinatario_hash_mac_fk": null  
}
```
- obtenerRegistrosAlarmasUsuario
 - Routing:
/api/registros/alarmas/hash_mac
 - Entrada:

```
"hash_mac_fk": " mac del usuario a  
observar "
```
 - Salida:

```
"1": {  
    "oid_reg": 1,  
    "tipo": "A",  
    "fecha": "2021-05-23T21:36:52",  
    "trabajo": 2910,  
    "descanso": 980,  
    "oid_llamada_fk": null,  
    "oid_alarma_fk": 1,  
    "hash_mac_fk": "mac123",  
    "remitente_hash_mac_fk": null,  
    "destinatario_hash_mac_fk": null  
}
```



```
"fecha": "2021-05-23T21:36:52",
"trabajo": 2910,
"descanso": 980,
"oid_llamada_fk": null,
"oid_alarma_fk": 1,
"hash_mac_fk": "mac123",
"remitente_hash_mac_fk": null,
"destinatario_hash_mac_fk": null
}
```

- obtenerRegistrosLlamadasEnviadas

- Routing:

```
/api/registros/llamadas/enviadas
/hash_mac
```

- Entrada: {

```
    "remitente_hash_mac_fk": " mac del
    usuario a observar "
```

```
}
```

- Salida: "4": {

```
    "oid_reg": 4,
    "tipo": "L",
    "fecha": "valor",
    "trabajo": null,
    "descanso": null,
    "oid_llamada_fk": 1,
    "oid_alarma_fk": null,
    "hash_mac_fk": null,
    "remitente_hash_mac_fk": " mac     del
    usuario a observar ",
    "destinatario_hash_mac_fk": "valor
}
```

- obtenerRegistrosLlamadasRecibidas

- Routing:

```
/api/registros/llamadas/recibida
s/hash_mac
```

- Entrada: {

```
    "destinatario_hash_mac_fk": " mac
    del usuario a observar "
```

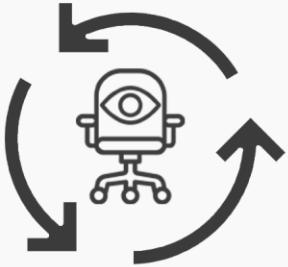
```
}
```

- Salida: "4": {

```
    "oid_reg": 4,
    "tipo": "L",
    "fecha": "valor",
    "trabajo": null,
    "descanso": null,
    "oid_llamada_fk": 1,
```



Chair Tracker



```
"oid_alarma_fk": null,  
"hash_mac_fk": null,  
"remitente_hash_mac_fk": "valor",  
"destinatario_hash_mac_fk": " mac del  
usuario a observar"  
}
```

Métodos PUT

También poseemos métodos para la edición de todas las tablas exceptuando la de registro que será actualizada automáticamente mediante triggers.

Los métodos son los siguientes:

- editarUsuario

- Routing:

</api/alarmas/editarUsuario>

- Entrada: {

```
"hash_mac": "mac del usuario a  
editar",  
"nif": "valor",  
"contrasena": "valor",  
"last_login": valor,  
"nombre": "valor",  
"apellidos": "valor",  
"rol": "valor",  
"nif_jefe": valor  
}
```

- Salida: mensaje de confirmación

- editarAlarma

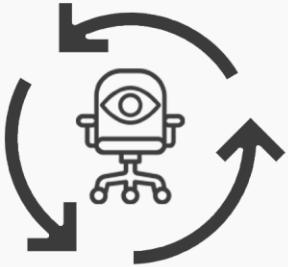
- Routing:

</api/alarmas/editarAlarma>

- Entrada: {

```
"oid_alarma": oid alarma a editar,  
"dias": " valor ",  
"t_inicio": " valor ",  
"t_fin": "valor",  
"t_trabajo": valor,  
"t_descanso": valor,  
"ciclo_trabajo": valor,  
"ciclo_descanso": valor,  
"hash_mac_fk": "valor"  
}
```





- Salida: mensaje de confirmación

- editarLlamada

- Routing:

</api/alarmas/editarLlamada>

- Entrada:

```
{  
    "oid_llamada": oid llamada a editar,  
    "estado": "valor",  
    "desde": " valor ",  
    "descripcion": " valor ",  
    "remitente_hash_mac_fk": " valor ",  
    "destinatario_hash_mac_fk": " valor "  
}
```

- Salida: mensaje de confirmación

Métodos POST

Los métodos para añadir elementos a las tablas son los siguientes:

- anadirUsuario

- Routing:

</api/usuarios/anadirUsuario>

- Entrada: {

```
"hash_mac": "valor",  
"nif": " valor ",  
"contrasena": " valor ",  
"last_login": valor,  
"nombre": " valor ",  
"apellidos": " valor ",  
"rol": " valor ",  
"nif_jefe": valor  
}
```

- Salida: mensaje de confirmación

- anadirAlarma

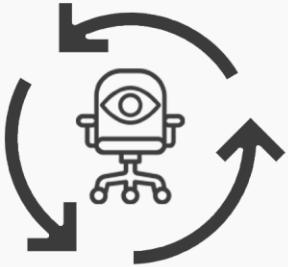
- Routing:

</api/usuarios/anadirAlarma>

- Entrada: {

```
"dias": " valor ",  
"t_inicio": " valor ",  
"t_fin": "valor",  
"t_trabajo": valor,
```





```
"t_descanso": valor,  
"ciclo_trabajo": valor,  
"ciclo_descanso": valor,  
"hash_mac_fk": "valor"  
}
```

- Salida: mensaje de confirmación

- anadirLlamada

- Routing:

</api/usuarios/anadirLlamada>

Entrada: {

```
"oid_llamada": 1,  
"estado": "valor",  
"desde": " valor ",  
"descripcion": " valor ",  
"remitente_hash_mac_fk": " valor ",  
"destinatario_hash_mac_fk": " valor "  
}
```

- Salida: mensaje de confirmación

Métodos DELETE

Si en algún momento necesitamos eliminar algún elemento de la BBDD, usaremos estos métodos (no será disponible eliminar llamadas ni registros para evitar malentendidos):

- borrarUsuario

- Routing:

</api/usuarios/borrarUsuario>

- Entrada:

```
{  
  "hash_mac": "mac del usuario a borrar"  
}
```

- Salida: mensaje de confirmación

- borrarAlarma

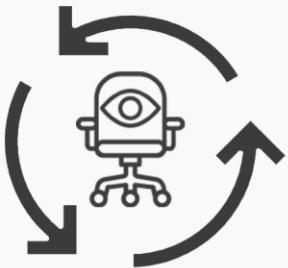
- Routing:

</api/usuarios/borrarUsuario>

- Entrada:

```
{  
  "oid_alarma": 1  
}
```

- Salida: mensaje de confirmación



5.1.6 MQTT

Para la implementación del estándar MQTT en nuestro proyecto hemos hecho uso del broker de Mosquitto, el cual corre en el puerto 8083 y necesitaría de identificación.

Servidor

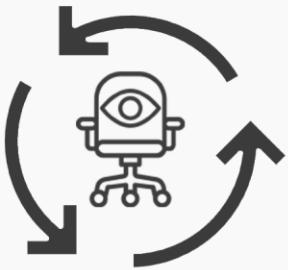
Desde el servidor utilizamos la librería de VERT.X para actuar como clientes al broker de Mosquitto y poder publicar mensajes. Estos son:

- Realización de una publicación de mensaje bajo la ruta `/destinatario_hash_mac/llamadas/` cuando añadimos una llamada desde la API REST usando el método `anadirLlamada`.
- Realización de una publicación de mensaje bajo la ruta `/hash_mac_fk/refresh/` cuando añadimos, editamos o añadimos una alarma desde la API REST usando los métodos `anadirAlarma`, `editarAlarma` y `borrarAlarma` respectivamente. Esto se hace con el objetivo de notificar al cliente siempre que haya un cambio en sus alarmas.

Cliente ESP8266

Desde el ESP8266 hemos implementado la librería PubSubClient para actuar como clientes respecto al broker al igual que hacemos en el servidor, en este caso estaremos suscritos a las rutas en las cuales publica el servidor, correspondiendo `destinatario_hash_mac` y `hash_mac_fk` al sha1 que realizamos a nuestra MAC al conectarnos a la red WIFI. De esta forma únicamente nos llegarán mensajes asociados a nuestro usuario, evitando así el recibir mensajes destinados a otros clientes. Según sean estos realizaremos distintas acciones:





Chair Tracker

- Al recibir una publicación de mensaje bajo la ruta `/destinatario_hash_mac/llamadas/` haremos uso de los actuadores, haciendo vibrar el dispositivo.
- Al recibir de una publicación de mensaje bajo la ruta `/hash_mac_fk/refresh/` realizaremos una búsqueda de la próxima alarma más cercana con los datos actualizados desde el servidor.

5.1.7 Cliente

Actuadores

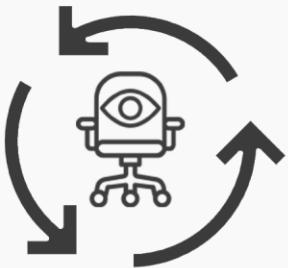
Nuestro principal actuador se trata de un sensor de distancia por ultrasonidos, concretamente el HC-SR04, que gestionamos mediante la librería Ping y que nos permite saber si el usuario se encuentra sentado o no en la silla para actuar en consecuencia en el contador que se explicara a continuación con mayor detalle.

Además, para complementar a este nuestro sistema presenta una alarma (buzzer activo) que enviará avisos sonoros para que el usuario descance o vuelva al puesto de trabajo.

Para el sistema de notificación de llamadas usamos un motor de vibración que permitirá al usuario percatarse de esta llamada sin confundirlo con el aviso sonoro que generan las alarmas.

Gestión alarmas

Para la gestión de alarmas nuestro sistema realizará una búsqueda de la próxima alarma más cercana en el tiempo y que corresponda al día actual, tomando como referencia para la hora y fecha los obtenidos mediante un NTP Client.



Esta operación se llevará a cabo no sólo en el setup, sino también cuando finalice nuestra alarma actual y en caso de no existir ninguna activa, realizaremos esta búsqueda cuando recibamos un mensaje MQTT en la ruta comentada anteriormente.

La búsqueda es realizada con la invocación del método obtenerProximaAlarma() que se encarga de ejecutar una petición por medio de la API REST de las alarmas asociadas a nuestro hash_mac para así realizar los cálculos pertinentes con la lista actualizada tal y como la encontramos en la base de datos.

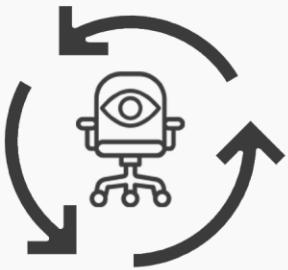
Una vez nos encontramos en la franja horaria de una de las alarmas, se iniciará un contador para los ciclos de trabajo y otro para los ciclos de descanso. Dependiendo de si el usuario se encuentra sentado en la silla o levantado, se irán actualizando estos contadores y cuando estos coincidan con el tiempo de trabajo o descanso, haremos uso de la alarma para notificar al usuario.

Tras completar su franja horaria, realizaremos un update a través de la API REST para actualizar los tiempos totales de trabajo y descansos realizados durante esta franja horaria.

Gestión Llamadas

Para la implementación de llamadas en el cliente debemos de gestionar la recepción de llamadas puesto que el envío se realizará en un futuro mediante la interfaz web.

Para ello, cuando se añada una alarma en la BBDD por medio de la API REST, recibiremos un mensaje MQTT que será el aviso para el cliente de que deberá de activar



el motor de vibración para notificar al usuario de una llamada entrante.

5.1.8 Aplicación

Este es una de las actividades que más tiempo han demandado a lo largo de las fases de diseño y de desarrollo, puesto que además de cubrir las necesidades y funciones requeridas por el sistema, se ha debido tener en cuenta la facilidad de uso por parte de cualquier tipo de usuario.

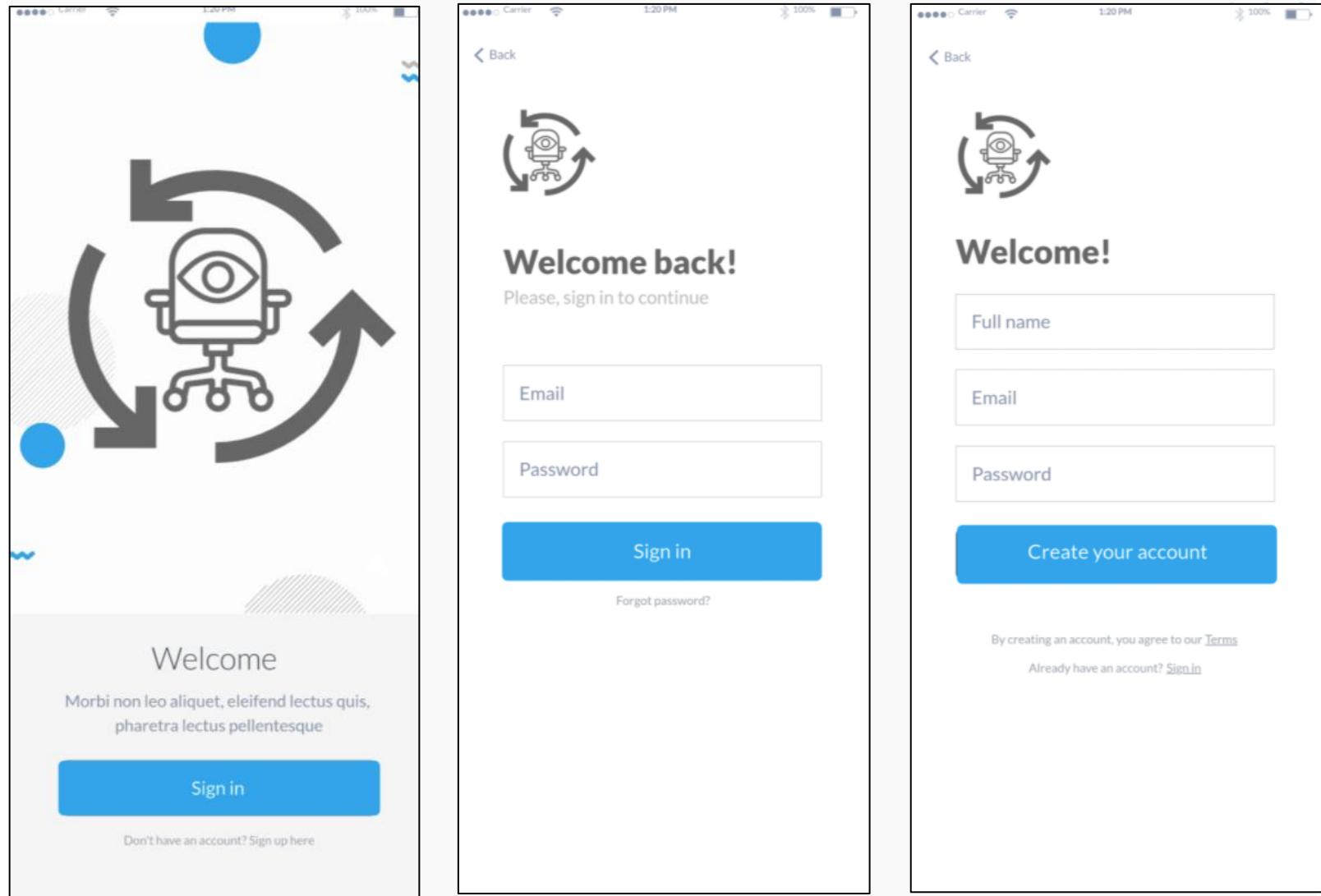
Mockups de las pantallas de la app

Tras realizar un estudio sobre como poder englobar todos los requisitos del proyecto, de cara a implementar una aplicación completa y de manejo sencillo, se proponen los siguientes diseños para la posterior implementación de pantallas, componentes, navegación, lógica, etc.

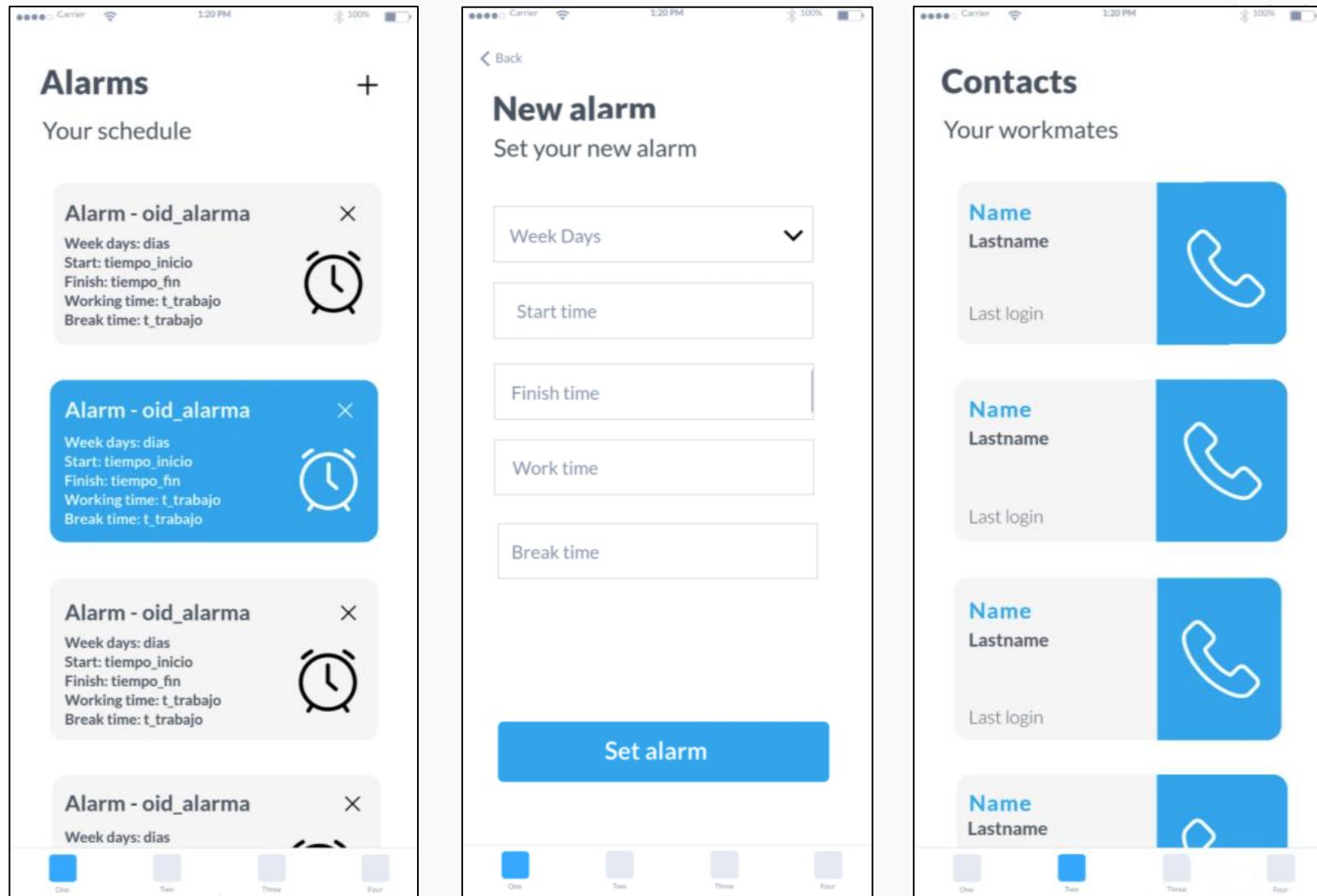
Primeramente, se ofrecerá una visión general de los mockups. Después, aparecen individualmente para enfatizar los detalles.

Chair Tracker

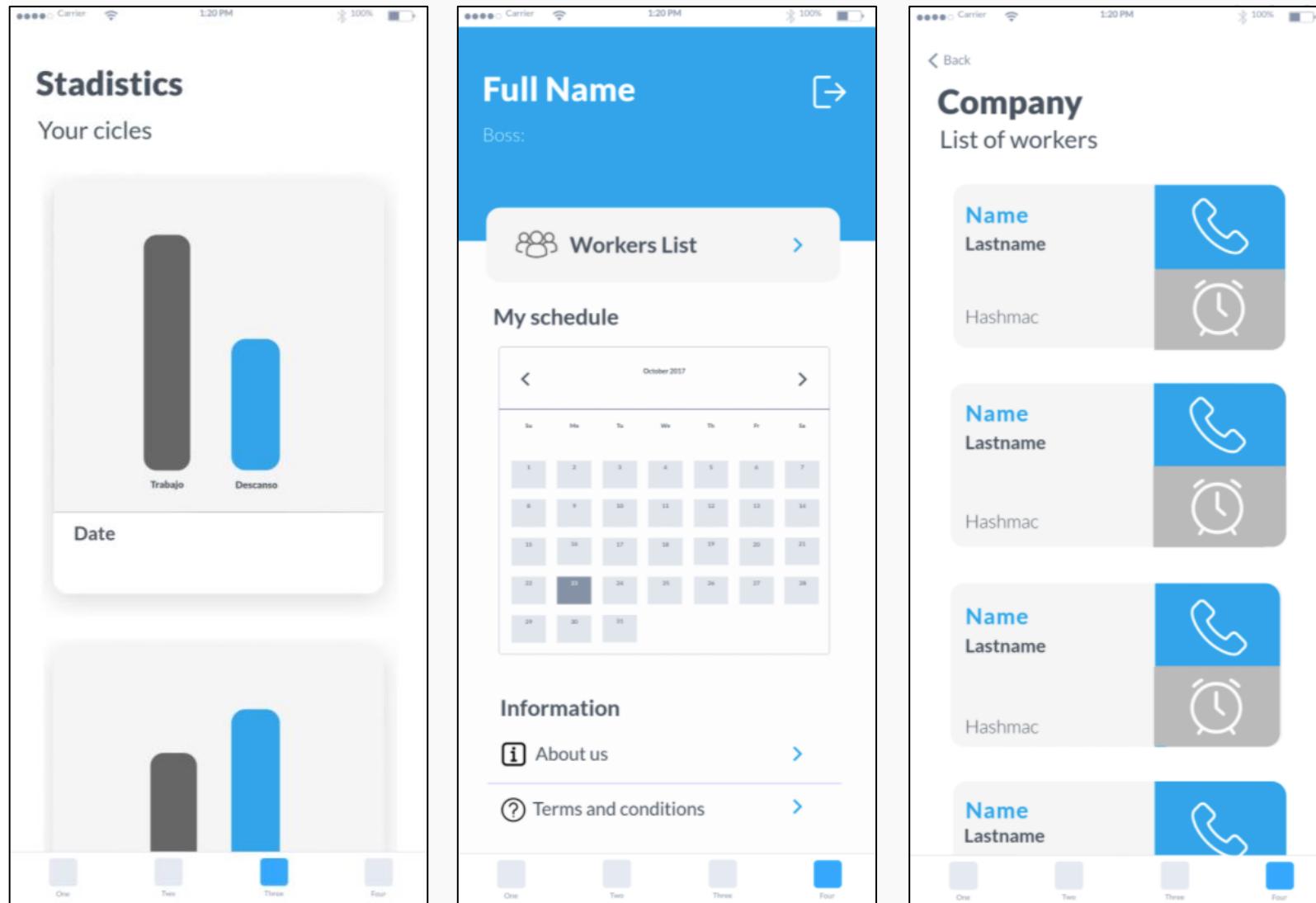




Chair Tracker



Chair Tracker

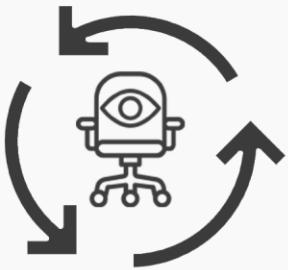


Chair Tracker

Chair Tracker



Figura 34. Imágenes de los primeros mock-ups de la aplicación



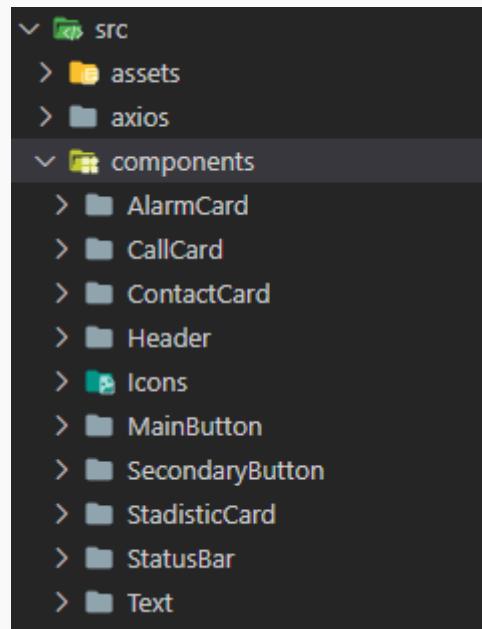
Estructura de la aplicación

En este apartado se expondrá la estructura que hemos seguido a lo largo del proyecto de la aplicación. Explicando brevemente el contenido de cada pantalla, además de la estructura de carpetas y archivos que hemos decidido implementar.

Estructura de archivos

En primer lugar, la estructura de archivos del proyecto se encuentra organizada en diversas subcarpetas dependiendo de su funcionalidad, son las siguientes:

- **Componentes:** se trata de elementos que pueden ser reutilizados a lo largo de la aplicación, dotándonos así de modularidad y aprovechamiento de código, es el caso del header, iconos, tarjetas o botones.

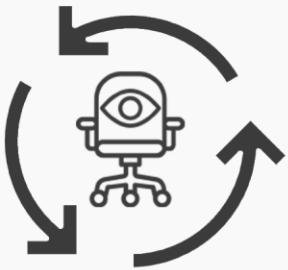


Chair Tracker

Figura 35. Estructura de carpetas referentes a componentes

- **Contenedores:** contiene todas y cada una de las pantallas que podemos encontrar a lo largo de la aplicación, ya se traten de





pantallas de autentificación, principales o modales

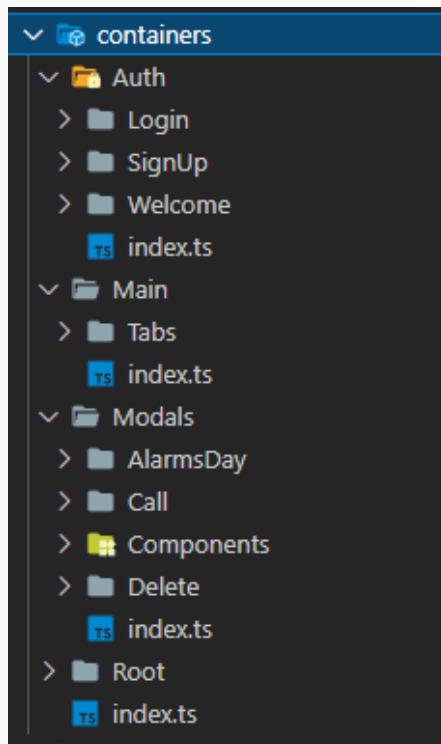


Figura 36. Estructura de carpetas referentes a contenedores

Dentro de estas carpetas encontramos a su vez 4 tipos de archivos principales: connect.ts, destinado a la lógica y recogida de datos que serán utilizados en index.tsx encargado de renderizar la propia pantalla y sus elementos, los cuales, reciben sus estilos en el archivo styles.ts destinado a ello. Además de un archivo types.ts donde se definen los tipos que serán utilizados en esta pantalla.

Chair Tracker

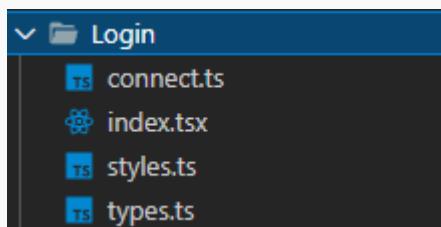
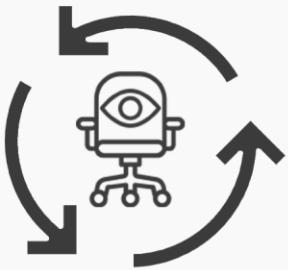


Figura 37. Estructura de archivos de una pantalla



- **Navegador:** encontramos los archivos destinados a la navegación, definiendo el navigator y los tipos que serán usados junto con las pantallas. Se utiliza la misma jerarquía de archivos de los contendores.

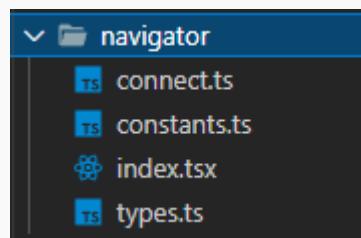


Figura 38. Estructura de archivos del navegador

- **Conexiones:** son varias las carpetas destinadas a diversos tipos de conexiones, axios, dirigida a la conexión con la API Rest mediante el uso de hooks. También se hace referencia a la carpeta de models utilizada para la normalización y tipado de los objetos una vez recibidos.

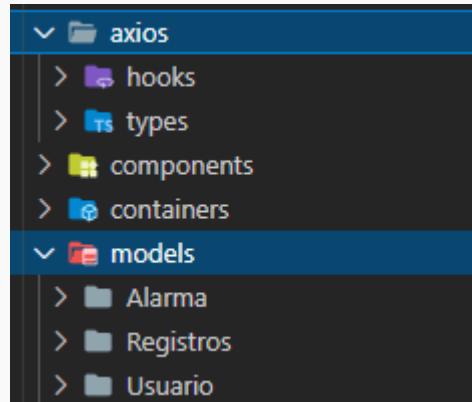
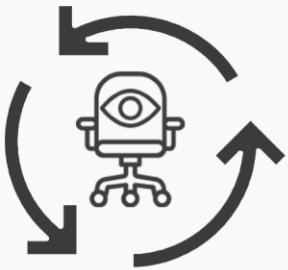


Figura 39. Estructura de carpetas referentes a las conexiones

Chair Tracker

- **Conexiones:** en esta categoría destacan la carpeta types donde definimos las variables de entorno usadas a lo largo de toda la aplicación y que deberán almacenarse en un archivo .env para mantener la



confidencialidad de estas y la carpeta theme donde encontramos diferentes apartados en lo referente al tema de la aplicación como colores o dimensiones del dispositivo.

Estructura de navegación

En este punto se presenta la estructura de navegación mediante un grafo.

Las flechas indican la dirección de la navegación, el color amarillo indica que nos encontramos en la parte referente a autentificación, verde pantalla principal, azul una secundaria y morado hace referencia a los modales.

Ch... Tracker

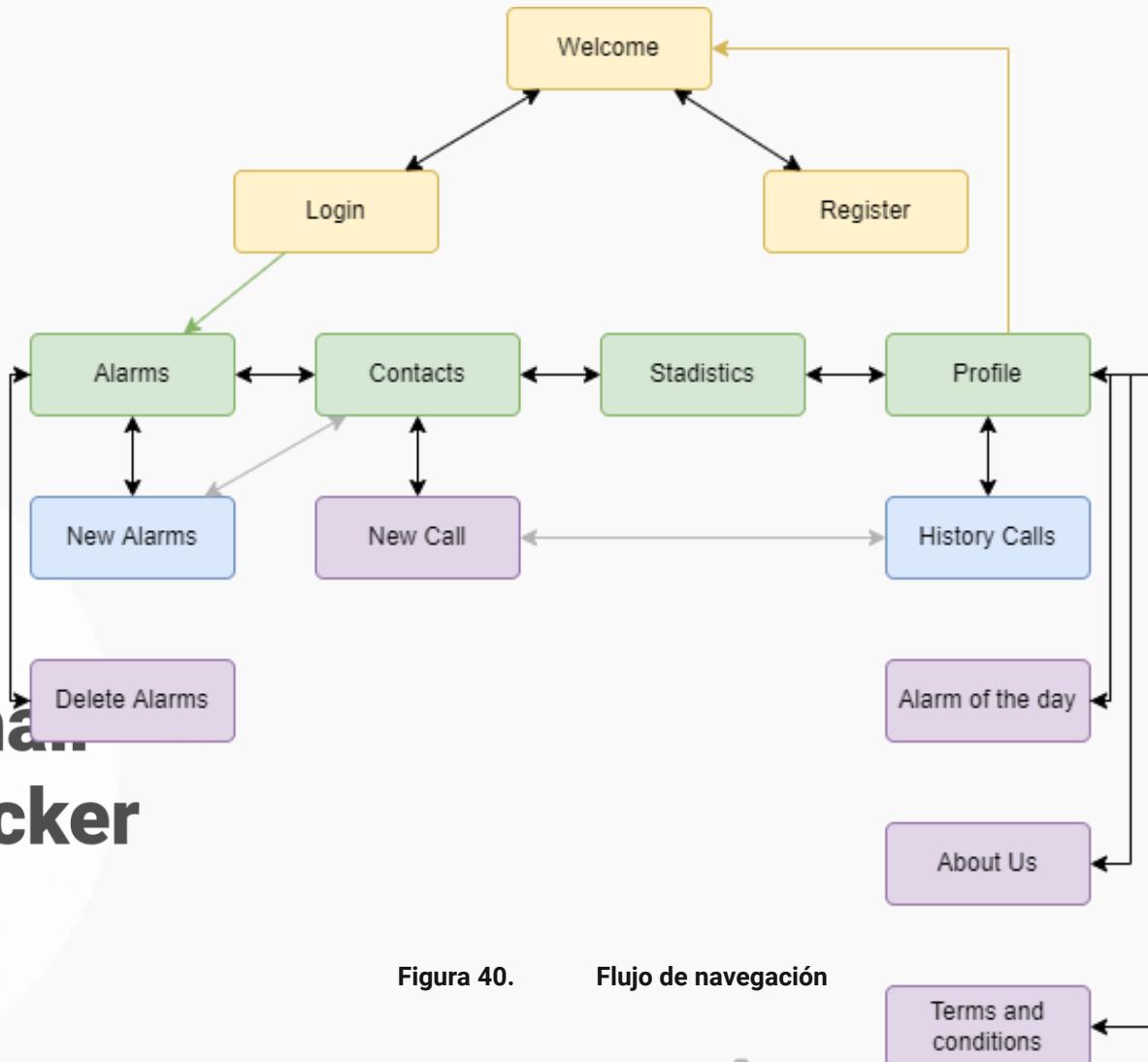
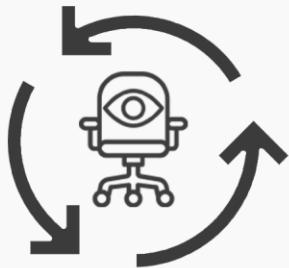


Figura 40. Fluo de navegación



Chair Tracker

Aspectos destacables de implementación

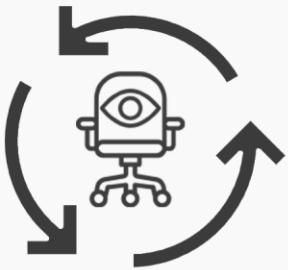
A continuación, comentaremos los aspectos más importantes que nos ha permitido otorgar a nuestra aplicación de una implementación lo más profesional posible.

En primer lugar, hemos realizado una maquetación de los componentes y de las pantallas totalmente adaptables a cualquier tipo de dispositivo. Por lo tanto, la aplicación tendrá una apariencia muy similar sin distinguir entre Android o iOS, incluso con distintos tamaños de pantalla. Para conseguir esto, hemos optado por usar tamaños dependientes del tamaño de pantalla, estilos condicionales entre Android e iOS, diseños válidos tanto para pantallas pequeñas como grandes, etc.

```
export const Container = styled.View<ContainerProps>`  
  flex: 1;  
  padding-top: ${({ safeTop }) => `${safeTop}px`};  
  background-color: ${({ theme }) => theme.colors.deepOcean};  
`;  
  
export const ImageContainer = styled.View`  
  flex: 2;  
  background-color: ${({ theme }) => theme.colors.deepOcean};  
`;  
  
export const Background = styled(FastImage).attrs({  
  resizeMode: 'cover',  
})`  
  border-radius: 12px;  
  bottom: 0;  
  left: 0;  
  position: absolute;  
  right: 0;  
  top: 0;  
  height: 100%;  
`;  
  
export const SubContainer = styled.View<ContainerProps>`  
  flex: 1;  
  background-color: ${({ theme }) => theme.colors.beforeBlue}  
  align-items: center;  
  justify-content: center;  
  padding-bottom: ${({ safeBottom }) => `${safeBottom}px`};  
`;  
  
export const Title = styled(TextBase)`  
  font-size: 30px;  
  color: ${({ theme }) => theme.colors.cultured};  
  font-weight: 400;  
`;
```

Figura 41. Aspecto destacable del código de estilos





También hemos sido conscientes de desarrollar una app lo más optimizada posible. Se ha tratado siempre de evitar repeticiones de código, ya sea mediante el uso de componentes (en maquetación) como a través de la creación de hooks o métodos personalizados dependiendo de la funcionalidad requerida (en lógica).

Es precisamente el uso de hooks lo que nos ha permitido mejorar bastante el funcionamiento de nuestra aplicación. Se ha necesitado el uso de casi todos: useState, useMemo, useCallback, useEffect... sumado a los creados por nosotros mismos.

Uno de los hooks claves para poder conectarnos a la API Rest desarrollada es el de useAxios, donde implementamos distintos métodos capaces de enviar o solicitar información para cada parte funcional de la app.

```
import { AXIOS_ENDPOINT } from '@env';
import { useAxios } from 'use-axios-client';
import type { Alarma } from 'axios/types/alarma';
import { normalizeAlarma } from 'models/Alarma';

export const useAlarmasByHashMac = (hashMac?: string | null) => {
  const { data, loading, error, refetch } = useAxios<JSON>({
    baseURL: AXIOS_ENDPOINT,
    method: 'get',
    url: '/alarmas/hash_mac',
    headers: {
      'content-type': 'application/json',
    },
    params: { hash_mac_fk: hashMac },
  });

  const auxData = data || '';
  const normalizedData: Alarma[] = [];

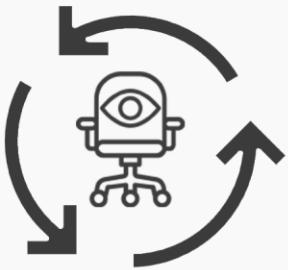
  Object.values(auxData).forEach((value) => {
    normalizedData.push(normalizeAlarma(value));
  });

  return { normalizedData, loading, error, refetch };
};
```

Chair Tracker

Figura 42. Aspecto destacable del código de hook de conexión a la API





El modelado de todos estos datos ha sido una parte fundamental para poder tratarlos correctamente usando TypeScript, actuando, así como enlace entre los JSON recibidos por la API y los objetos que se usan a lo largo de todos los componentes y pantallas.

Los datos se han manejado de distinta forma dependiendo del uso que tienen en la aplicación, por ejemplo, los datos de identificación del usuario se guardan en la caché del dispositivo tras el login del mismo.

```
const useConnect = () => {
  const { goBack, canGoBack } = useNavigation();

  const [nif, setNif] = useState('');
  const [password, setPassword] = useState('');
  const { normalizedData } = useUsuarioExist(nif, password);

  useEffect(() => {
    AsyncStorage.setItem('logged', normalizedData.existe.toString());
    AsyncStorage.setItem('hash_mac', normalizedData.hash_mac);
    AsyncStorage.setItem('nombre', normalizedData.nombre);
    AsyncStorage.setItem('nif_jefe', normalizedData.nif_jefe);
    if (normalizedData.existe === 1) {
      RNRestart.Restart();
    }
  });

  const handleGoBack = useCallback(() => {
    if (canGoBack()) {
      goBack();
    }
  }, [canGoBack, goBack]);

  return { handleGoBack, nif, setNif, setPassword };
};

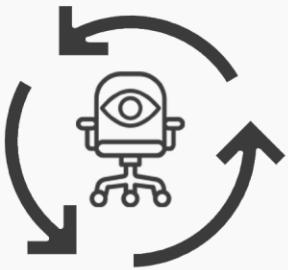
export default useConnect;
```

Chair Tracker

Figura 43. Aspecto destacable del código de almacenamiento en memoria

Otros datos como los de alarmas, llamadas o estadísticas se cargan al entrar en la aplicación. Aunque estos son más especiales debido a que cambian





constantemente (creación de alarmas, cumplimiento de ciclos de trabajo/descanso, realización de llamadas...) por lo que hemos visto necesario el incluir un refresco automático mediante suscripciones MQTT de los datos por cada cambio que sufran estos datos en el back-end. Además, para que el usuario se cerciore de que sus datos están siempre actualizados, hemos incorporado también un refresco manual en las pantallas.

```
const mqttClient = new Mqtt.Client('tcp://' + MQTT_ENDPOINT);

mqttClient.connect(
{
  clientId: '1001',
  username: 'root',
  password: 'root',
},
(err: any) => {
  console.log(err);
},
);

mqttClient.on(Mqtt.Event.Connect, () => {
  mqttClient.subscribe([hashMac + '/alarmas/refresh'], [0]);
  mqttClient.subscribe(['null/alarmas/refresh'], [0]);
});

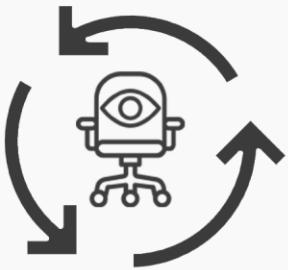
mqttClient.on(Mqtt.Event.Message, (topic: any, message: any) => {
  console.log('Mqtt Message:', topic, message.toString());
  refetch();
});
```

Figura 44. Aspecto destacable del código mqtt

Chair Tracker

Esta sensación de seguridad por parte del usuario la ofrecemos también en la integridad de los datos creados o eliminados en la aplicación. Todos los formularios para la creación de elementos como usuarios, llamadas o alarmas son validados antes de permitir que el usuario realice la petición. Al igual que en el borrado de alarmas, donde aparece un modal de confirmación con el objetivo de evitar el borrado con pulsaciones erróneas.





Estos modales otorgan fluidez en la experiencia de usuario, porque en formularios pequeños o menos importantes, no navegamos a una pantalla distinta para realizarlos, sino que superponemos este tipo de modales sobre las pantallas donde se llaman. Al igual que a la hora de mostrar información menos relevante como los “Términos y condiciones” o el “Acerca de”, emergen otro tipo de modales.

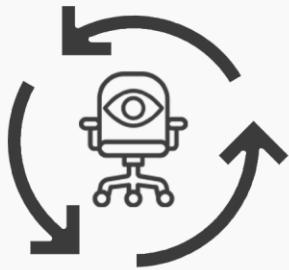
```
return (
  <Navigator screenOptions={generalStackScreenOptions}>
    {logged === '1' ? (
      <Group screenOptions={generalStackScreenOptions}>
        <Screen name="Tabs" component={Tabs} />
        <Screen name="NewAlarms" component={NewAlarms} />
        <Screen name="CallHistory" component={CallHistory} />
      </Group>
    ) : (
      <Group screenOptions={generalStackScreenOptions}>
        <Screen name="Welcome" component={Welcome} />
        <Screen name="SignUp" component={SignUp} />
        <Screen name="Login" component={Login} />
      </Group>
    )}
    <Group screenOptions={modalStackScreenOptions}>
      <Screen name="AboutUs" component={AboutUs} />
      <Screen name="TermsAndConditions" component={TermsAndConditions} />
    </Group>
    <Group screenOptions={transparentModal}>
      <Screen name="DeleteModal" component={DeleteModal} />
      <Screen name="CallModal" component={CallModal} />
      <Screen name="AlarmsDayModal" component={AlarmsDayModal} />
    </Group>
  </Navigator>
);
```

Figura 45. Aspecto del código destacable del navegador

Por último, en nuestra opinión, hemos intentado integrar a conciencia todas las librerías que hemos necesitado para conseguir tanto un aspecto como una funcionalidad más profesional. Un ejemplo de ello ha sido la integración de formularios, pickers, calendario, elementos nativos de cada sistema, etc.

```
<CalendarItem
  current={new Date().toDateString()}
  monthFormat={'MMM yyyy'}
  minDate={'2022-01-01'}
  maxDate={'2025-12-31'}
  onDayPress={handleToAlarmsDayModal}
  onPressArrowLeft={(subtractMonth) => subtractMonth()}
  onPressArrowRight={(addMonth) => addMonth()}>
```

Chair Tracker



Chair Tracker

```
export const CalendarItem = styled(Calendar).attrs({
  hideExtraDays: true,
  disableMonthChange: true,
  firstDay: 1,
  disableAllTouchEventsForDisabledDays: true,
  enableSwipeMonths: true,
})`  
  align-self: center;
  width: ${({ theme }) => theme.device.width * 0.9}px;
  padding-bottom: 5px;
  background-color: ${({ theme }) => theme.colors.marineBlue};
  margin-top: 10px;
  margin-bottom: 20px;
`;  
  


```
onSubmit={handleLogIn}
validationSchema={yup.object().shape({
 nif: yup
 .string()
 .matches(/[0-9]{8}[A-Z]{1}/, 'NIF must have 00000000A format')
 .required('NIF is required'),
 contrasena: yup.string().required('Password is required'),
})}

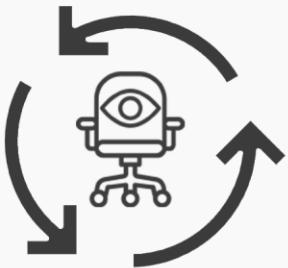
{({{
 handleSubmit,
 handleChange,
 handleBlur,
 values,
 errors,
 touched,
}) => [
 <>
 <InputText
 placeholder="NIF"
 onChangeText={handleChange('nif')}
 onBlur={handleBlur('nif')}
 value={values.nif}
 keyboardType="default"
 />

 {errors.nif && touched.nif && <TextError>{errors.nif}</TextError>}

 <InputText
 placeholder="Password"
 onChangeText={handleChange('contrasena')}
 onBlur={handleBlur('contrasena')}
 value={values.contrasena}
 keyboardType="default"
 secureTextEntry
 />
 </>
]}
```


```

Figura 46. Aspecto del código destacable del uso de librerías



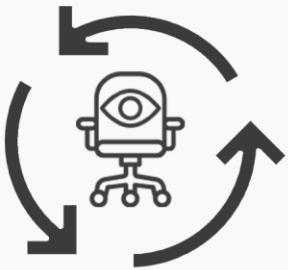
Chair Tracker

Librerías utilizadas

Hemos usado una gran cantidad de librerías a lo largo de la aplicación por lo que las enumeraremos brevemente, mostrando la versión usada y el objetivo de su uso:

```
"dependencies": [
  "@hookform/resolvers": "2.8.5",
  "@react-native-async-storage/async-storage": "1.17.3",
  "@react-native-community/datetimepicker": "6.1.0",
  "@react-native-community/hooks": "2.8.1",
  "@react-native-picker/picker": "2.4.0",
  "@react-navigation/bottom-tabs": "6.0.9",
  "@react-navigation/core": "6.1.0",
  "@react-navigation/material-top-tabs": "6.0.6",
  "@react-navigation/native": "6.0.6",
  "@react-navigation/native-stack": "6.2.5",
  "axios": "0.26.1",
  "date-fns": "2.28.0",
  "formik": "2.2.9",
  "intl-pluralrules": "1.3.1",
  "lodash": "4.17.21",
  "lottie-ios": "3.2.3",
  "lottie-react-native": "5.0.1",
  "react": "17.0.2",
  "react-hook-form": "7.22.5",
  "react-native": "0.66.4",
  "react-native-bootsplash": "3.2.4",
  "react-native-calendars": "1.1282.0",
  "react-native-chart-kit": "6.12.0",
  "react-native-date-picker": "4.2.0",
  "react-native-dotenv": "3.3.1",
  "react-native-fast-image": "8.5.11",
  "react-native-gesture-handler": "2.1.0",
  "react-native-native-mqtt": "github:FrozenPyrozen/rn-native-mqtt"
  "react-native-pager-view": "5.4.15",      You, 4 weeks ago • fea
  "react-native-reanimated": "2.3.1",
  "react-native-restart": "0.0.24",
  "react-native-safe-area-context": "3.3.2",
  "react-native-screens": "3.10.1",
  "react-native-svg": "12.3.0",
  "react-native-switch-pro": "1.0.5",
  "react-native-webview": "11.15.0",
  "styled-components": "5.3.3",
  "use-axios-client": "2.0.0",
  "yup": "0.32.11"
],
```

Figura 47. Listado de librerías utilizadas



Las librerías más relevantes para el desarrollo de la aplicación han sido: `async-storage` [26] (para el almacenamiento de información en caché), `picker` [28], `datetimepicker` [27], `formik` [15] y `yup` [20] (para la realización de formularios y su validación), `styled-components` [30] (para la creación de componentes estilizados sin la necesidad de usar StyleSheet). Y otros auto descriptivos, pero igual de importantes como `calendar` [29], `chart-kit` [17], `mqtt` [14] o `use-axios-client` [37].

5.1.9 Cambios introducidos

Para el desarrollo de una aplicación móvil con mejor funcionalidad y facilidad de uso, fueron necesarias las siguientes modificaciones sobre base de datos, API y diseño:

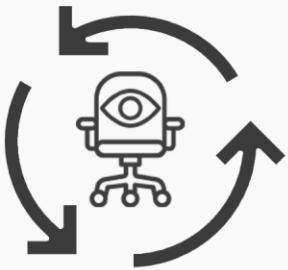
Base de datos

- **Tablas:**

Los primeros cambios introducidos a las tablas son referentes a la longitud del `hash_mac` del usuario que pasa de 40 caracteres a 10, con el objetivo de facilitar el registro de este en el sistema.

En los registros fueron necesarias más modificaciones, con el fin de mostrar una información más completa y detallada sobre los datos almacenados de las llamadas. Se incluyeron los campos de: `remitente_nombre`, `destinatario_nombre`, `desde`, `descripción`. Así como la eliminación de las referencias a las claves secundarias, para que los datos se queden almacenados permanentemente pese a futuras eliminaciones de alarmas.





- **Triggers:**

Uno de los cambios referentes a los *triggers* de la BBDD hace referencia a la clasificación de los registros de las alarmas por los meses del año. Esto tiene la finalidad de generar nuevos registros cada mes para poder así tener estadísticas más precisas sobre las alarmas.

Tras las pruebas software nos percatamos de un error en el trigger de solapamiento de alarmas, por lo que se tuvo que adaptar en distintos aspectos. El primero de ellos, que permitiese el solapamiento de horas si ninguno de los días establecidos en la nueva alarma solapaba a los horarios de las ya creadas. Por último, que prohibiera la creación de alarmas con el mismo horario.

Los triggers de creación automática de los registros de alarmas y llamadas también fueron modificados para que soportasen los nuevos datos implementados.

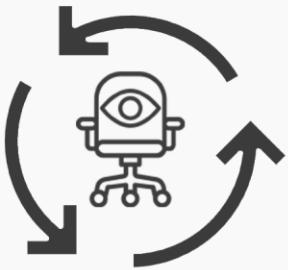
API

Uno de los cambios más necesarios para poder utilizar los métodos creados en la API fue la adaptación de estos para que funcionasen a través tanto de peticiones con cuerpo, como con peticiones por parámetros. Fue debido a la librería de *Axios* para React Native, donde es obligatorio utilizar parámetros para la utilización de *métodos GET*.

Chair Tracker

También se requirió de la implementación de un nuevo método para realizar el Log In del usuario. Este método es el encargado de establecer si el usuario introducido existe o no en la BBDD.





Para mostrar una agenda de contactos coherente con el entorno de cada usuario, se agregó un método que filtra la lista de contactos por el *hash_mac* del usuario.

Con el objetivo de aportar una utilidad extra al calendario implementado, se creó un nuevo método para devolver las alarmas de cada usuario filtradas por un día de la semana.

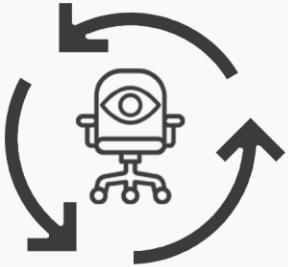
Con la finalidad de ampliar y clasificar la información en el futuro, se creó un método que filtrase los registros de un usuario por año, para así mostrar en la aplicación los registros de ese año.

Por último, se realizaron los cambios oportunos para la adaptación a las modificaciones realizadas en la base de datos cambios en los *topics* de los mensajes MQTT.

Métodos introducidos en la API

Los nuevos métodos son los siguientes:

- existeUsuario
 - Routing:
</api/usuarios/existeUsuario>
Entrada: {
 "nif": "00000000A",
 "contraseña": "valor",
}
○ Salida: {
 "existe": 0 || 1,
}
- obtenerContactos
 - Routing:
</api/usuarios/contactos>
Entrada: {
 "hash_mac": "valor",
}
○ Salida: mensaje de confirmación



Chair Tracker

- obtenerAlarmasUsuarioDia
 - Routing:
/api/alarmas/hash_macDia
 - Entrada: {
 "hash_mac_fk": "valor",
 "dias": "valor",
}
 - Salida: mensaje de confirmación

- obtenerRegistrosAlarmasUsuarioAnyo
 - Routing:
/api/registros/alarmas/hash_macAnyo
 - Entrada: {
 "hash_mac_fk": "valor",
 "anyo": "valor",
}
 - Salida: mensaje de confirmación

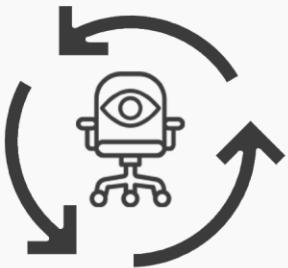
Diseño

El cambio más notorio respecto a los Mockups planteados fue la elección de una paleta de colores con más cohesión, además de ayudar a reducir la fatiga visual al optar por colores más oscuros.

Los tamaños de los componentes fueron modificados para adaptarse a la pantalla de cada usuario, para mantener los mismos márgenes y proporciones a lo largo de toda la aplicación, sin importar el tamaño del dispositivo.

Las tarjetas de las alarmas fueron modificadas para optimizar su tamaño y poder mostrarse en la pantalla en filas de 2 tarjetas. Las tarjetas de los contactos sufrieron un leve cambio para permitir el establecimiento de alarmas por parte de los jefes a sus empleados. Todos los componentes, además, fueron cambiados para reducir información innecesaria o redundante en ellos.





Los formularios se han creado más completos que en los mockups, permitiendo la inclusión de validaciones y mensajes de errores. También se adaptaron con el objetivo de estar más integrados con el diseño de la aplicación.

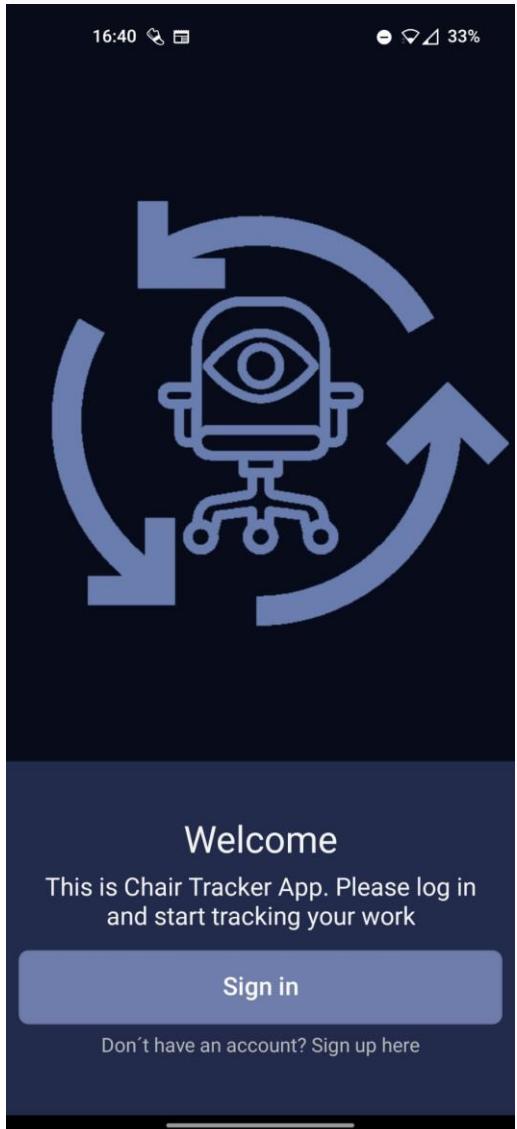
Por último, un cambio tanto de diseño como de funcionalidad se centró en el componente del perfil, que, en vez de mostrar una lista redundante de contactos, mostraría el registro de llamadas, salientes y entrantes, que aluden al usuario.

Además, con el objetivo de confirmar las acciones del usuario, así como crear interfaces más intuitivas fue necesario el establecimiento de modales en distintas ocasiones.

El resultado ha sido el siguiente:

Chair Tracker





16:41 33%

< Back

Welcome!

HashMac

NIF

Password

Name

Surnames

Boss

Register

By creating an account, you agree to our Terms

16:41 33%

< Back

Welcome back!

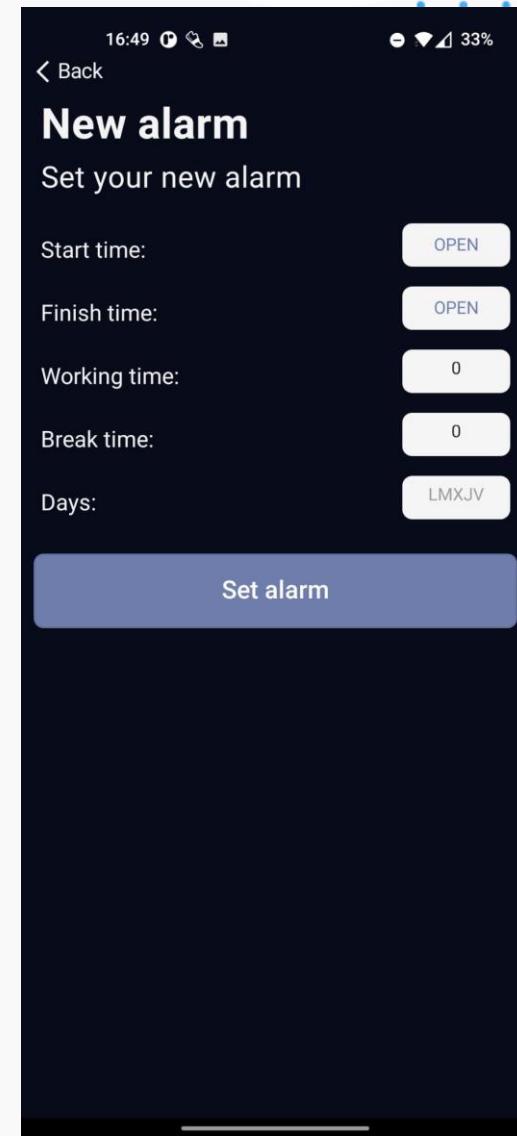
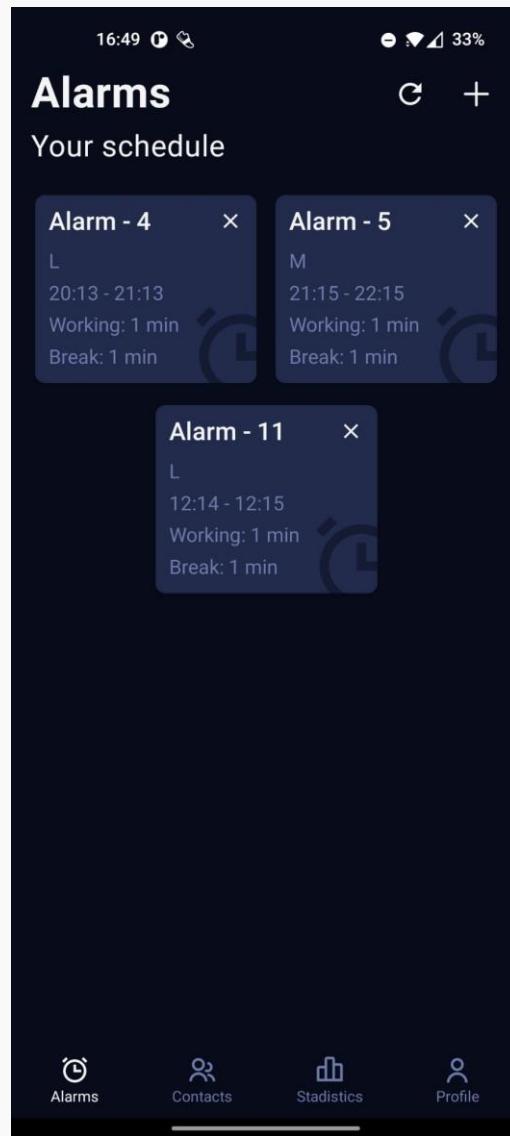
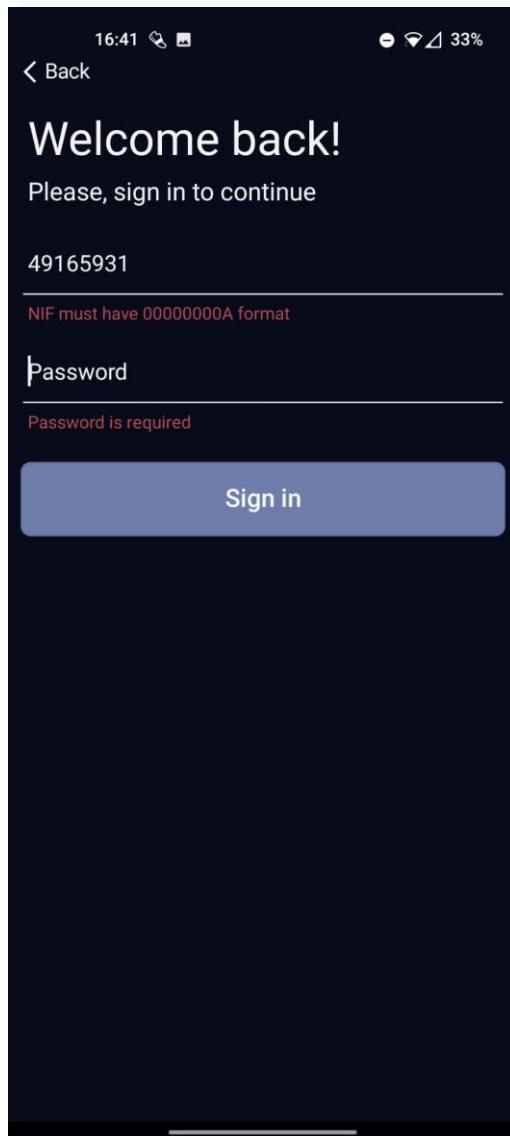
Please, sign in to continue

NIF

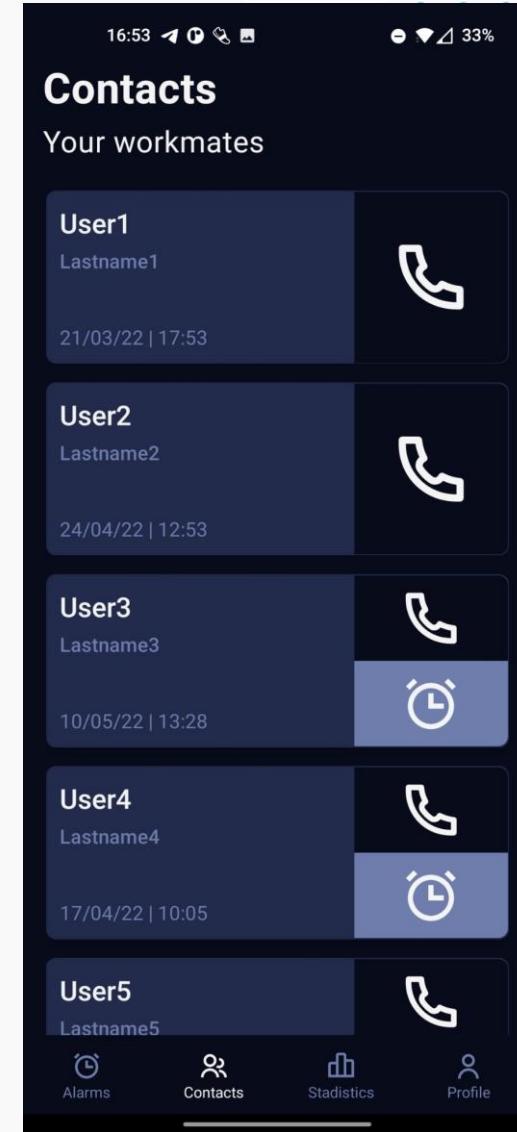
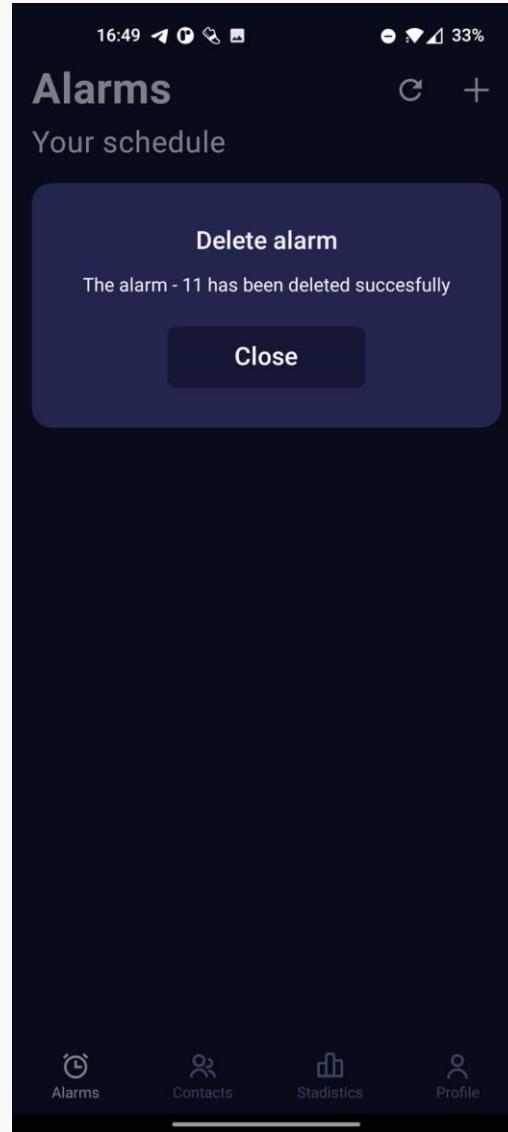
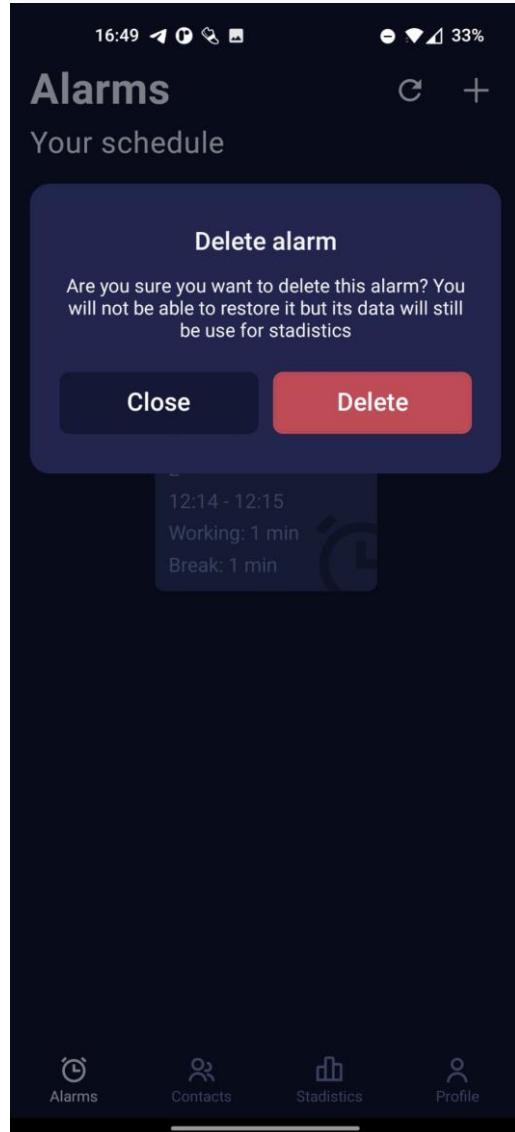
Password

Sign in

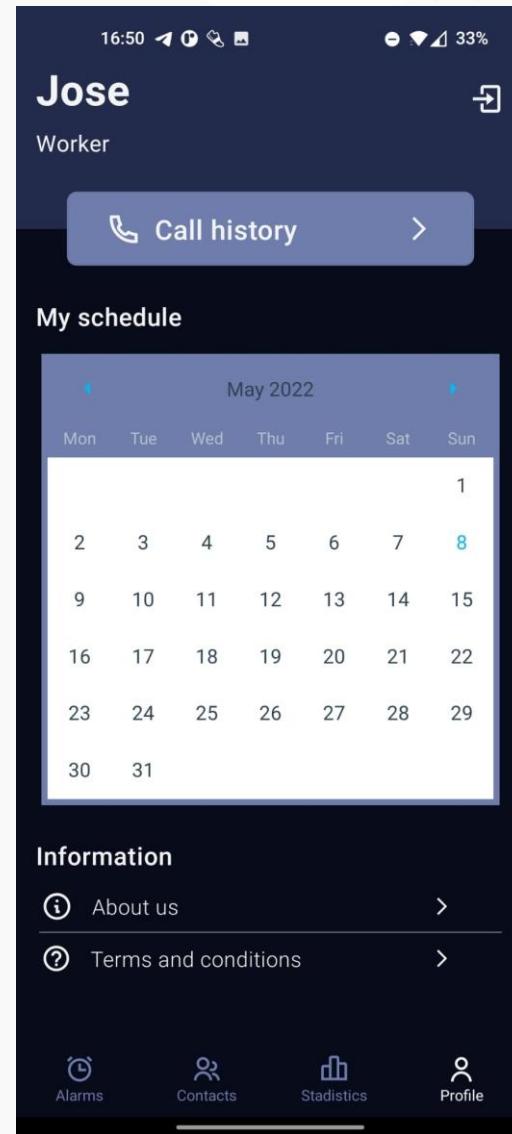
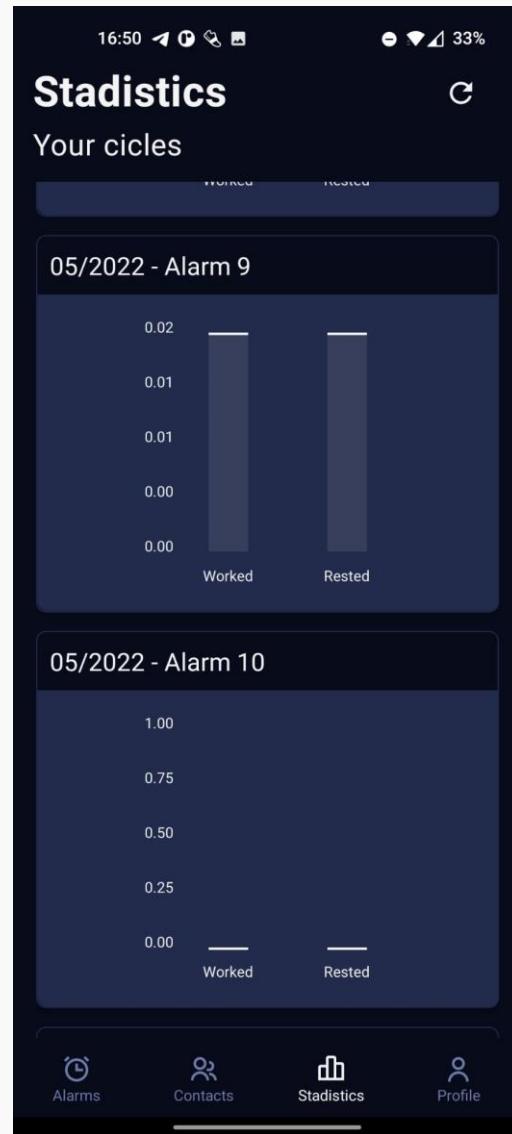
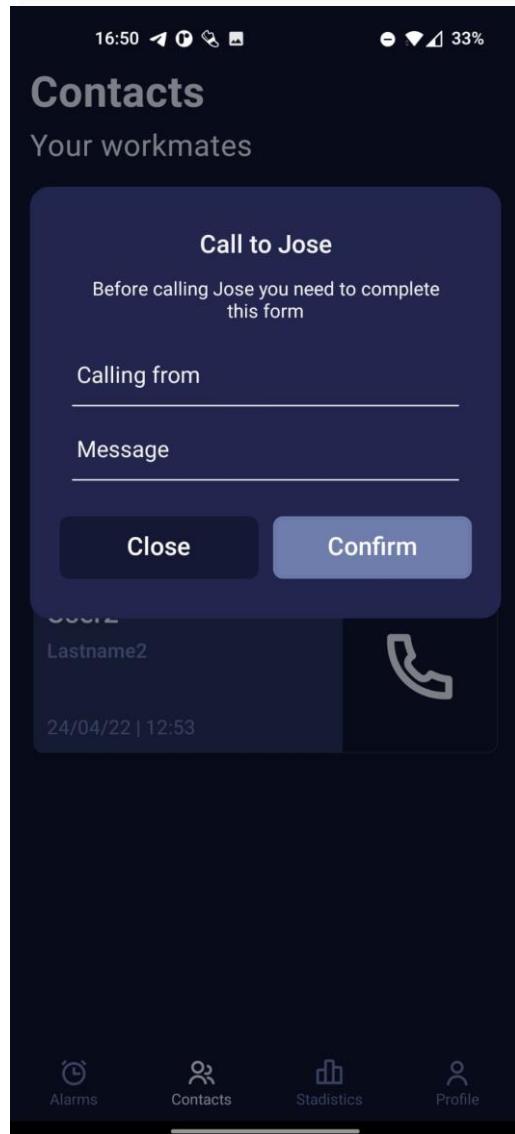
Chair Tracker



Chair Tracker



Chair Tracker



Chair Tracker

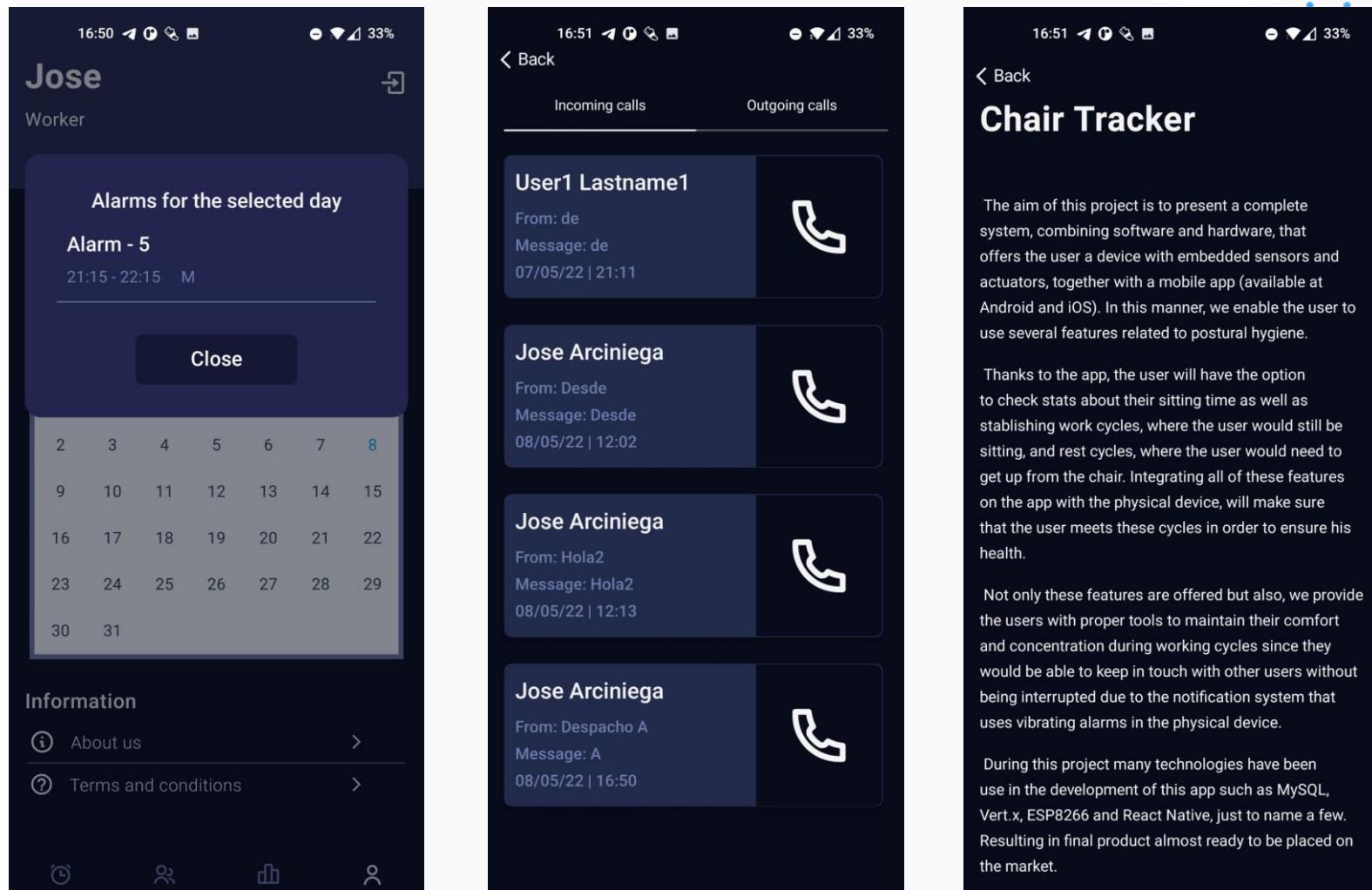
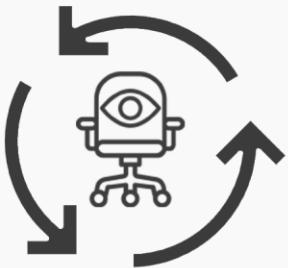


Figura 48. Imágenes finales de la implementación de la app

Chair Tracker



5.2 Hardware

En este otro apartado, explicaremos tanto la fase de diseño como el resultado del desarrollo de toda la parte hardware de nuestro sistema. En el directorio del proyecto se ofrecerán los archivos fuente implementados para poder tener un nivel mayor de detalle sobre la resolución de las decisiones tomadas a lo largo de las diferentes fases.

5.2.1 Fases del desarrollo del circuito

Mostraremos todas las etapas e iteraciones que ha sufrido nuestro prototipo hardware del sistema, así como el resultado propuesto como dispositivo bastante cercano a un producto final.

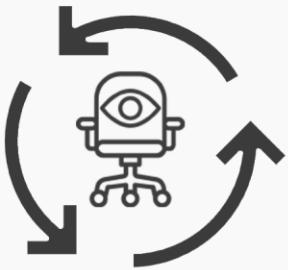
Prototipo en fase inicial de desarrollo

Para el prototipado inicial se ha hecho uso de una fuente 12V 8.3A, por lo que el voltaje ha tenido que ser regulado con un convertidor DCDC que permite regular el voltaje para obtener los 5V que serán usados para la alimentación del sensor de proximidad y el motor de vibración debido a las limitaciones del voltaje del ESP.

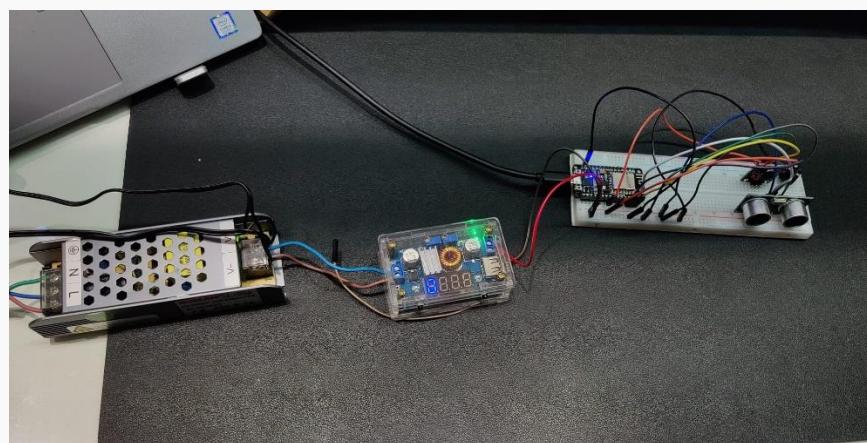
Para evitar el ruido en las señales de lectura que estábamos obteniendo en el HC-SR04 alimentado con una fuente externa, hemos tenido que normalizar las señales de tierra, conectando así el ESP a la tierra de la fuente para tomar esa tierra común como referencia.

Las señales TRIGGER y ECHO están conectadas al ESP en las líneas D1 y D2. La alarma y motor vibrador están conectados a las D5 y D0, respectivamente.

El ESP es aún alimentado mediante el USB del PC para así facilitar el desarrollo y tener acceso a la consola Serial del dispositivo.



En iteraciones futuras el sistema será autónomo y estará alimentado con baterías 18650 que tendrán la posibilidad de ser recargadas.



Chair Tracker



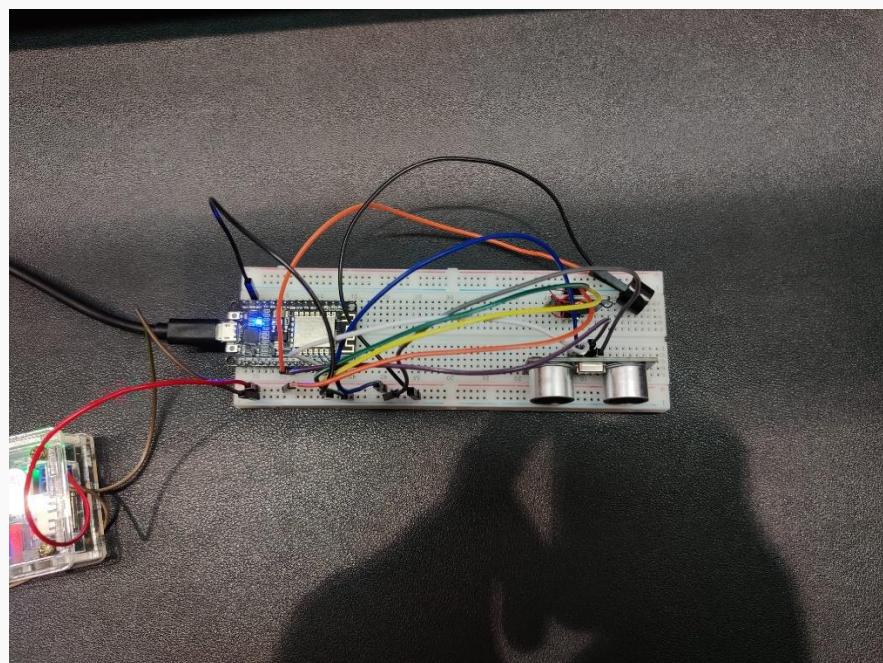
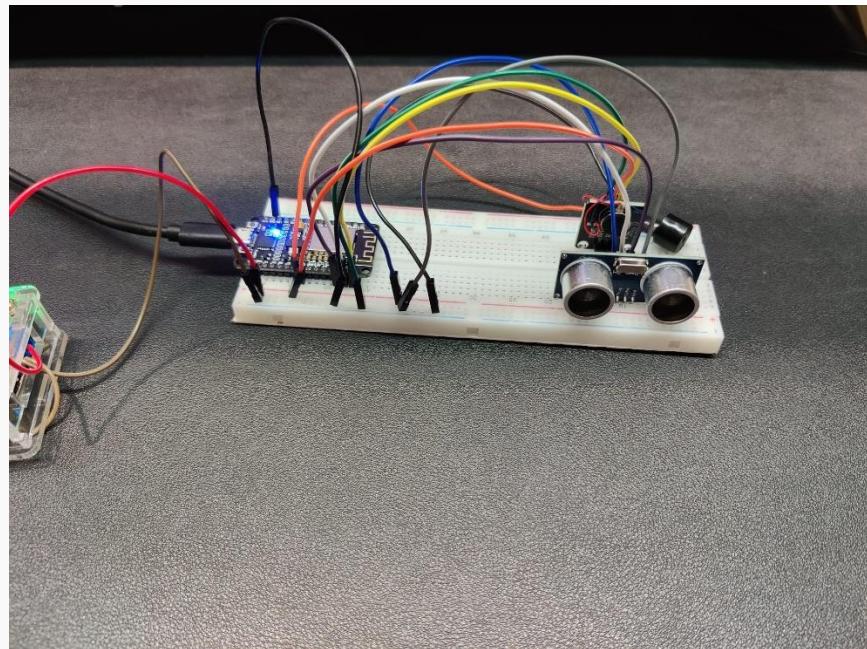
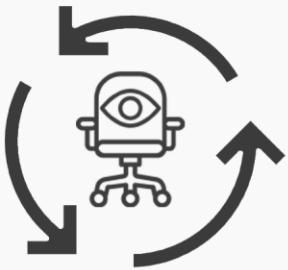
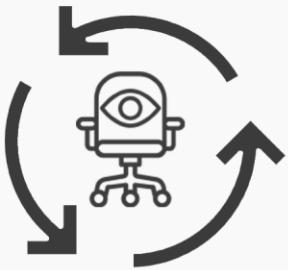


Figura 49. Imágenes del prototipo inicial

Chair Tracker

Prototipo en la segunda iteración de desarrollo

Para este prototipado se ha hecho uso de un battery shield que hace uso de dos baterías 18650 y que nos suministra una línea de 5V, otra de 3.3V y



alimentación mediante USB. Además, ofrece la posibilidad de recargar las baterías mediante un USB.

Como comentamos en la fase inicial necesitamos estos 5V para alimentar los sensores HC-SR04 y el motor vibrador. El plan inicial era alimentar el ESP8266 mediante el USB, pero hemos considerado que debido a que se trata de un producto en vías de desarrollo, lo más adecuado era realizar un prototipo orientado precisamente a esta parte del desarrollo por lo que era necesario añadir un USB que nos permitiera conectar el dispositivo a la consola Serial.

De esta forma, alimentamos el microcontrolador mediante la entrada Vin y hemos instalado un adaptador USB OTG MicroUSB-USB A hembra, a través del cual conectaremos un cable USB A macho-macho dirigido al PC de desarrollo. Es de importancia recalcar que es posible alimentar el ESP por la entrada Vin mientras mandamos datos por el USB gracias al diodo Schottky que se encuentra instalado entre la línea Vin y la USB. Tal y como se muestra en el esquemático:

POWER

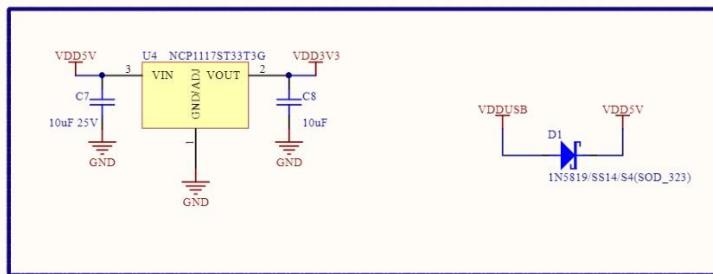
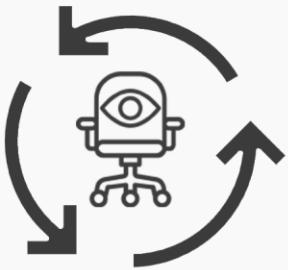


Figura 50. Diagrama del transistor de encendido ESP-8266

Esto nos obliga a que si queremos conectar el USB debemos de encender antes el dispositivo haciendo uso de las baterías o podríamos tener problemas de voltaje.

Chair Tracker





Pero esto no supone problema al tratarse de una unidad de desarrollo y no un producto final de cara al usuario el cual no tendría la posibilidad de conectarse al serial y alimentaríamos el microcontrolador mediante el USB, lo cual le brinda más protecciones al usuario.

Quedando el conexionado tal y como se muestra en la siguiente imagen:

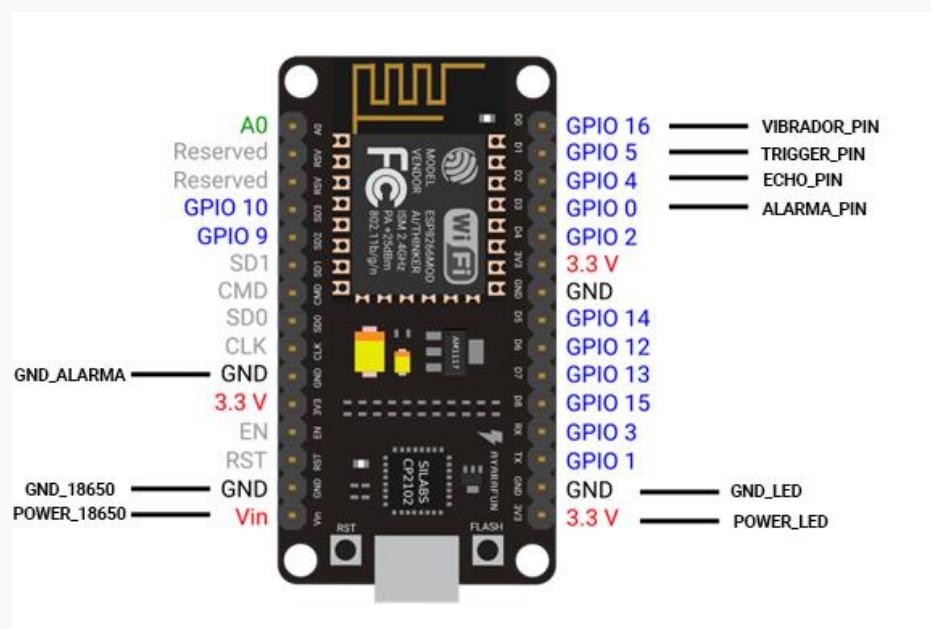
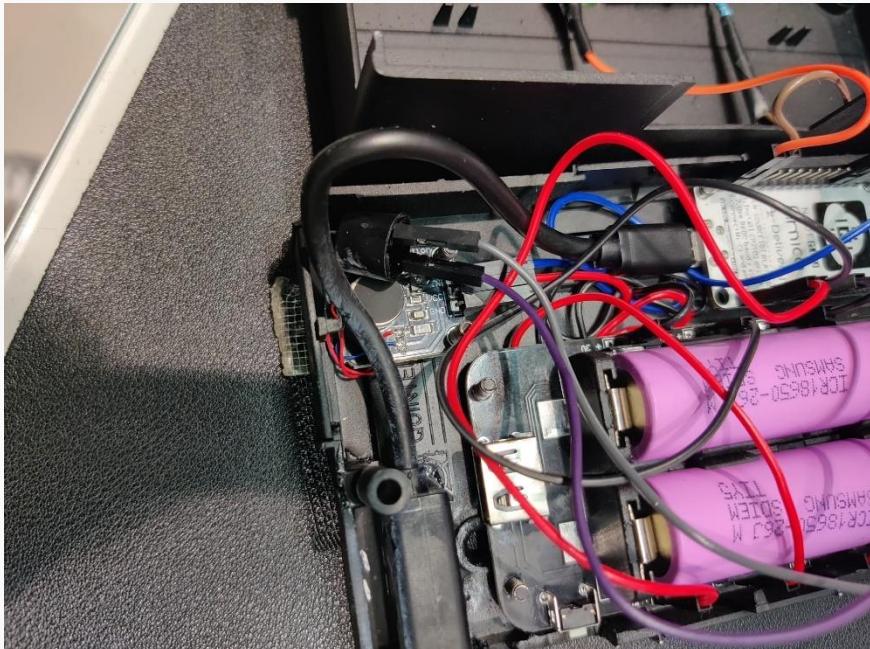
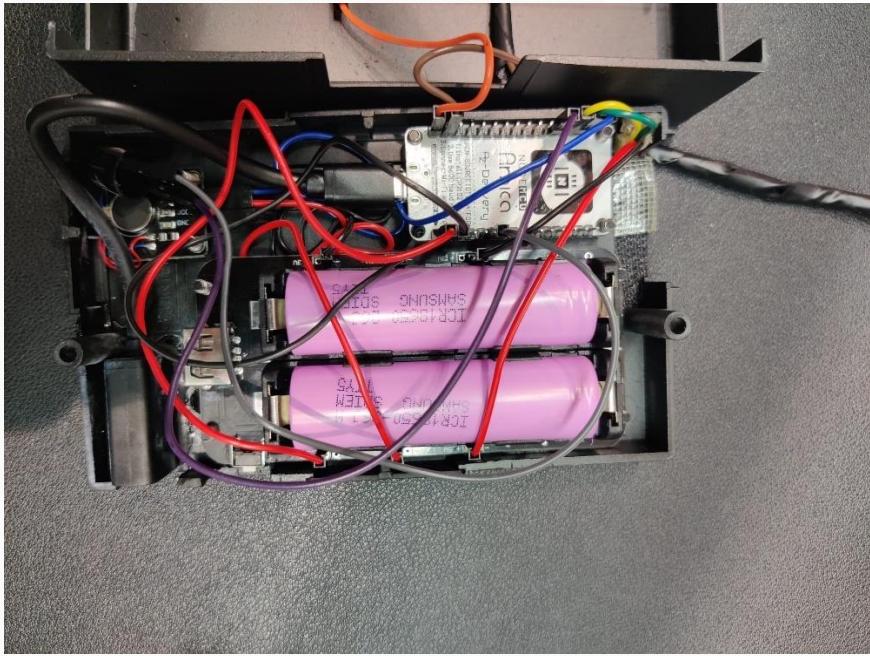
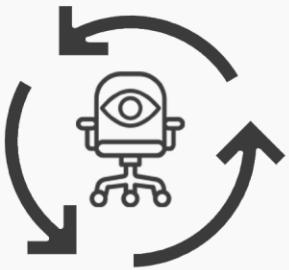


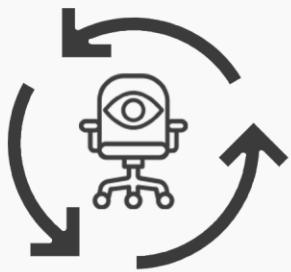
Figura 51. Diagrama de conexionado del dispositivo

Hemos usado una caja contenedora la cual alberga los componentes sujetos mediante tornillos e irá instalada en la parte inferior de la silla y el sensor es instalado en el exterior con un soporte, en este caso hemos optado por instalarlo en el brazo de la silla. El montaje completo se puede observar en las siguientes imágenes:

Chair Tracker

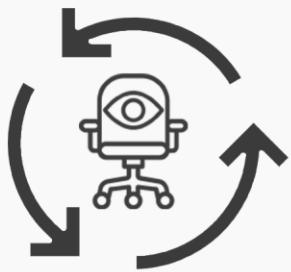


Chair Tracker

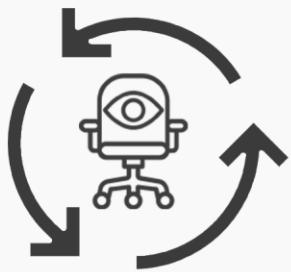


Chair Tracker





Chair Tracker



Chair Tracker



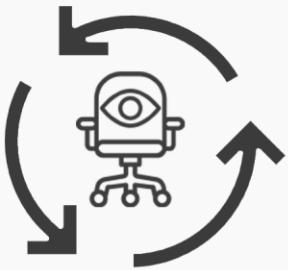


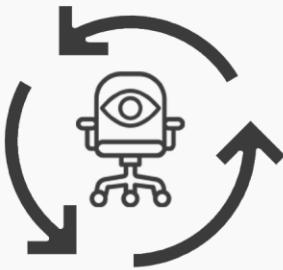
Figura 52. Imágenes de la segunda iteración del dispositivo

Chair Tracker

Prototipo en la tercera iteración de desarrollo

Para esta iteración comprendíamos que era necesario realizar una reducción de nuestro sistema ya que ocupaba un gran espacio, para solucionar este problema introdujimos dos cambios de componentes referentes a la alimentación, usando ahora una batería LIPO de menor tamaño y un TP4056 como regulador de tensión y mecanismo de carga de esta.

Este cambio de componentes junto con la implementación de las técnicas de diseño 3D y diseño de



PCBs que se detallarán a continuación permitieron reducir el circuito del prototipo a lo siguiente:

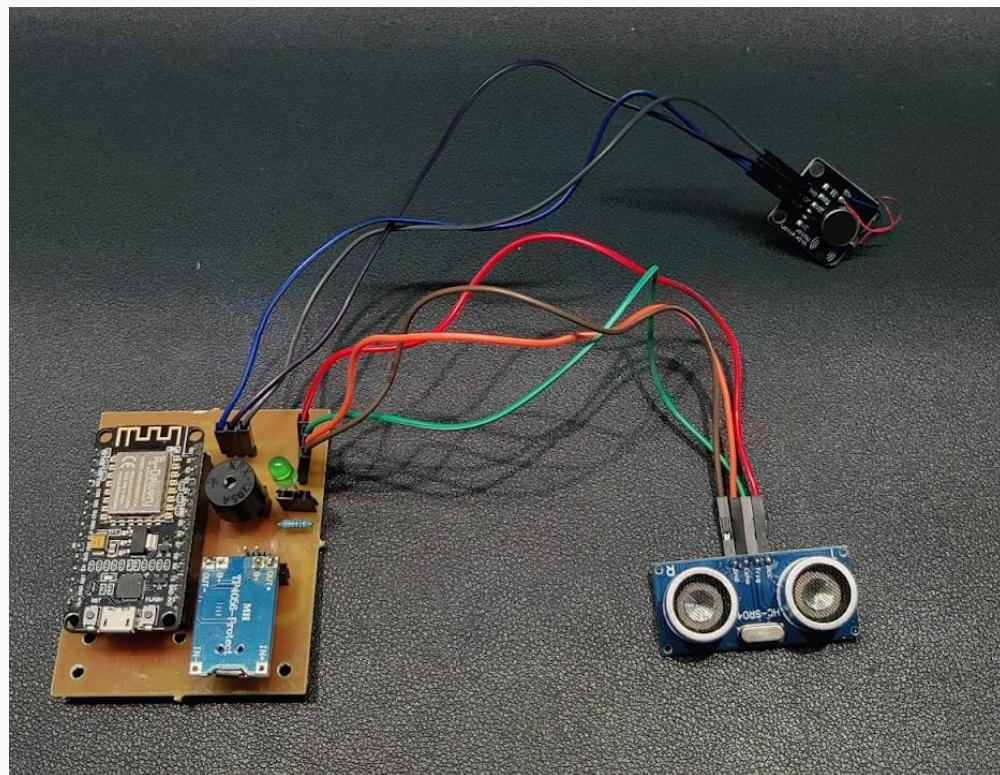


Figura 53. Imágenes de la primera PCB

5.2.3 PCB

Debemos de tener en cuenta que el diseño se deberá realizar utilizando una única cara y que el tamaño de cable que utilizaremos corresponde con el recomendado para señales digitales, 0,4064 mm pues todas las señales que tenemos son procedentes del microcontrolador y las referentes a alimentación presentan un voltaje de 5V.

Chair Tracker

Nos hemos apoyado en una de las mayores comunidades web existentes para la obtención de símbolos y huellas de nuestros componentes, concretamente, Snapeda. [33] [34] [35] Gracias a esto hemos podido diseñar el siguiente esquemático basado en el diagrama mostrado en la figura 54:

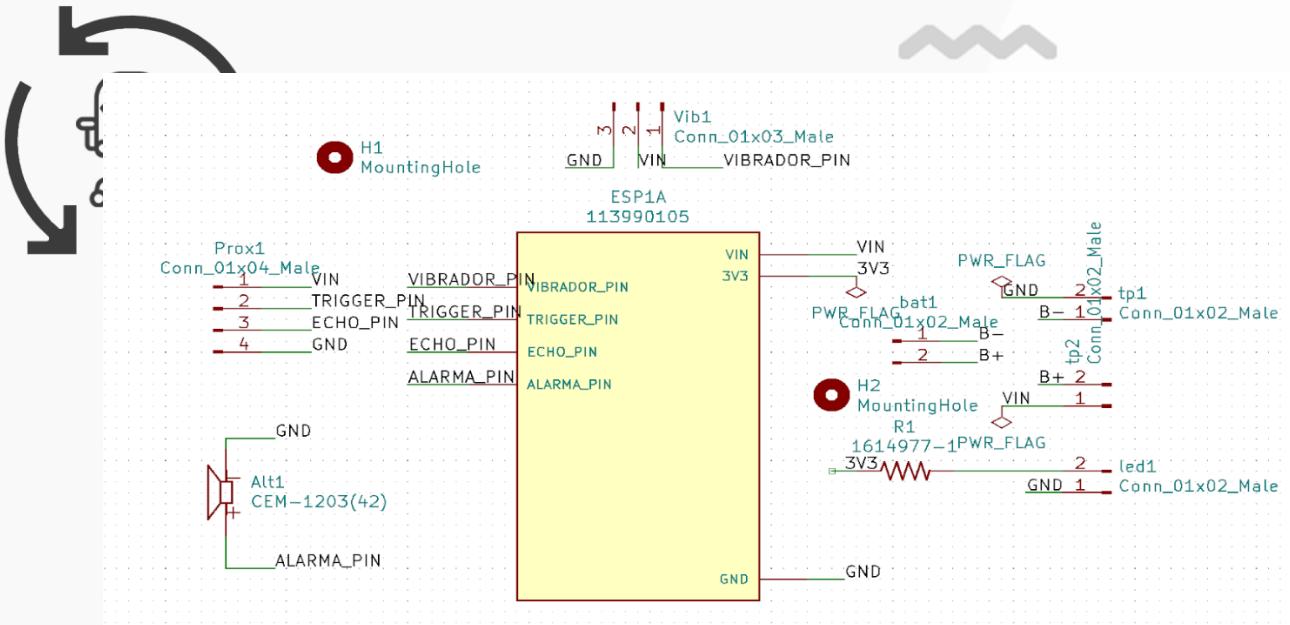


Figura 54. Esquema del diseño de la primera PCB

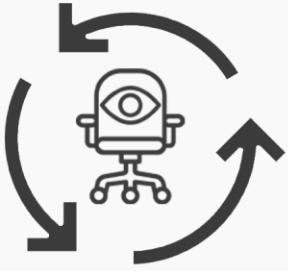
El diseño tuvo que ser a una cara debido a que las primeras versiones de la placa se construyeron en una máquina de troquelado por lo que se debió tener en cuenta diversas consideraciones distribuyendo los componentes de forma que ocupen lo menos posible. También es muy importante que las conexiones entre los distintos componentes se crucen lo mínimo posible.

Una vez distribuidos, realizamos la traza de las conexiones con el grosor indicado, procurando no formar ángulos de 90 grados, no dejar zonas aisladas, dejando un margen de separación entre pistas..., entre otras recomendaciones, dejando sin conectar todas las conexiones que vayan a tierra.

Después, para conectar las tierras, establecimos una conexión conjunta de toda la capa, que no toca las pistas trazadas.

Sin embargo, cuando la construimos el primer prototipo nos dimos cuenta de que podríamos haber integrado de una mejor forma el componente TP4056, por lo que decidimos realizar otra PCB completa.

Chair Tracker



Realizamos todos los pasos anteriores, pero con una redistribución de los componentes un poco más compleja, dando como resultado, el diseño físico final que tendrá la PCB de nuestro sistema:

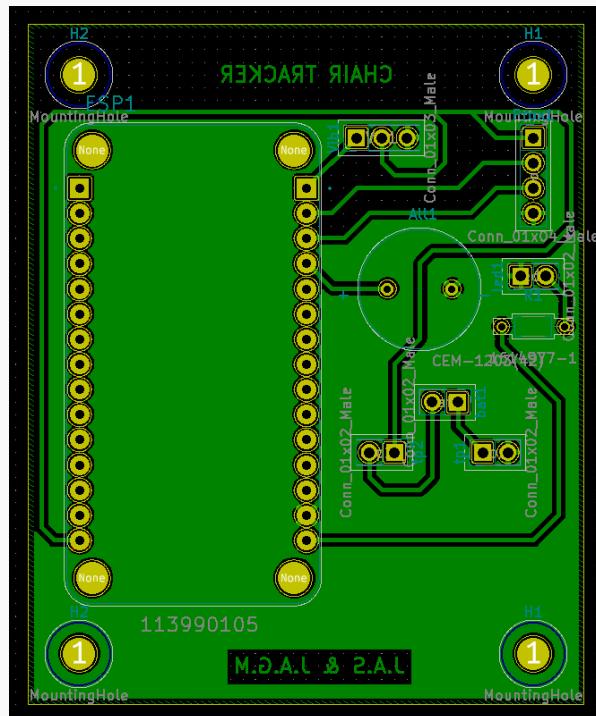
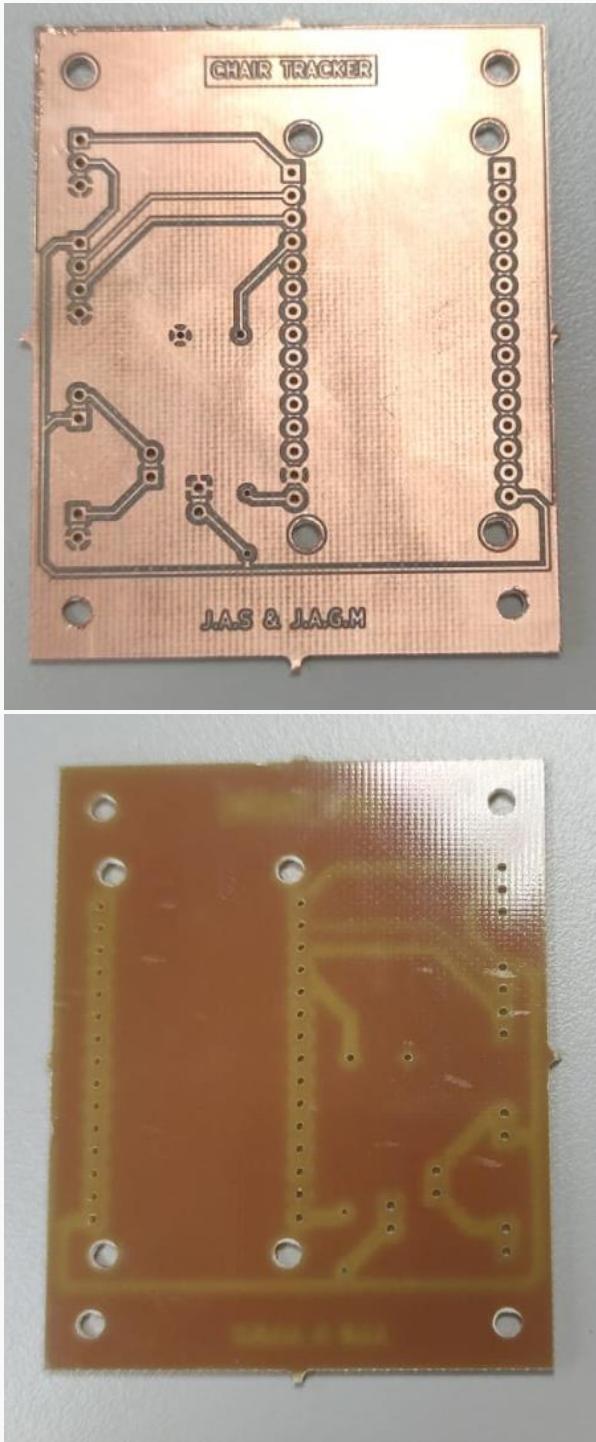
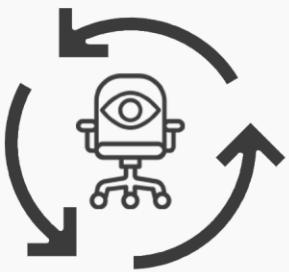


Figura 55. Imágenes de la primera PCB

Los resultados de las primeras impresiones fueron las mostradas:

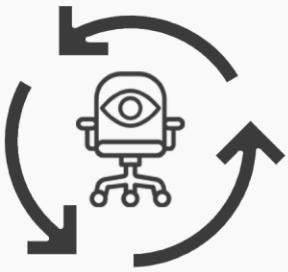
Chair Tracker



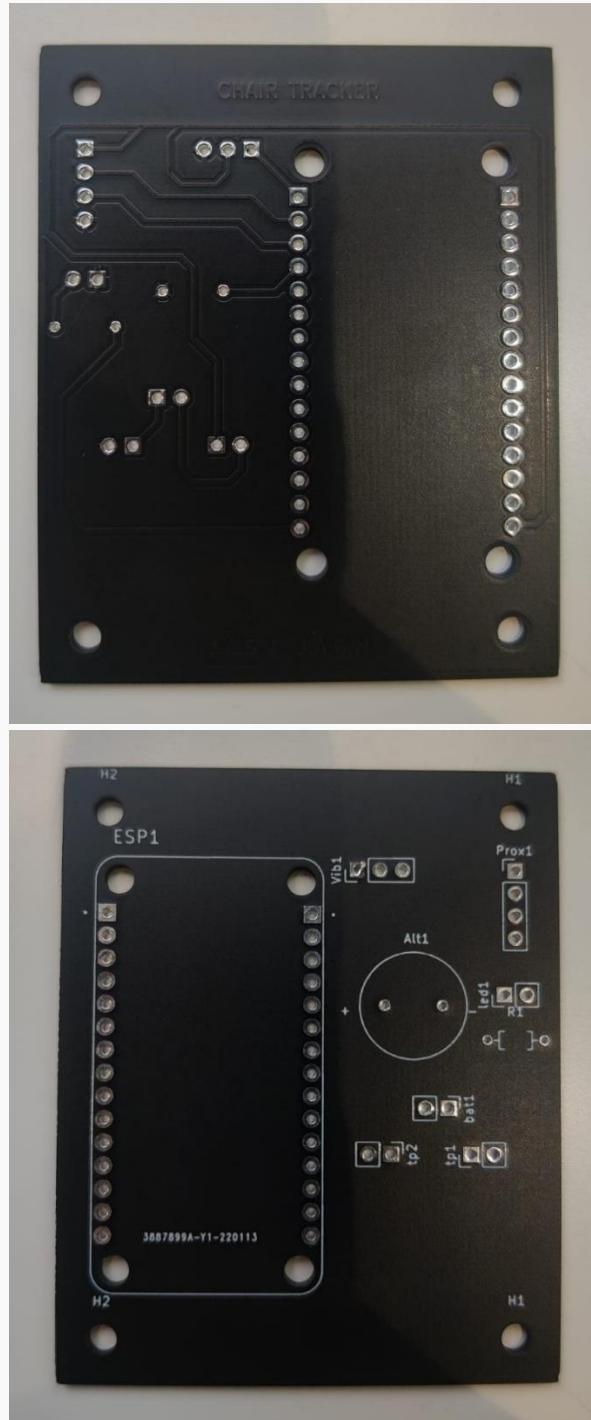
Chair Tracker

Figura 56. Imágenes de la primera PCB impresa

Que tras comprobar su diseño y funcionamiento en la tercera iteración del prototipo, se decidió optar por una solución más profesional con una PCB a doble capa, solucionando así pequeños detalles como la oxidación del cobre y la durabilidad del producto, así como contar con

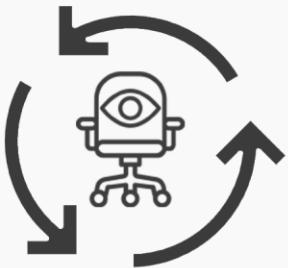


una máscara de soldadura que facilita todo el proceso de montaje de componentes. El resultado fue el siguiente:



Chair Tracker

Figura 57. Imágenes de la PCB final



5.2.4 Carcasa

El diseño de la carcasa tuvo como base el proyecto Project-Box-Template [32] que ofrecía una solución al problema de no tener que usar tornillería para lo que iba a ser la base y tapa de la caja.

A este se le realizaron diversas modificaciones como la adición de pilares, también llamados stand-offs, para separar la batería del montaje de la PCB.

Además, se le realizaron extrusiones para acomodar el acceso al puerto USB-C de carga y facilitar el montaje del interruptor y el cable dedicado a la sensorización externa junto con personalización de la marca.



**Chair
Tracker**

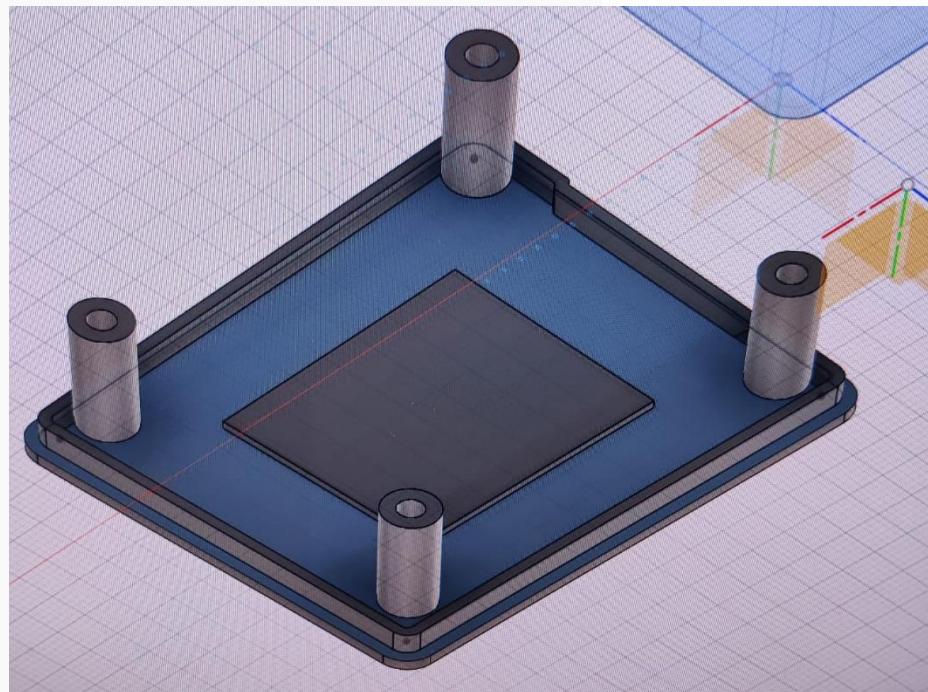
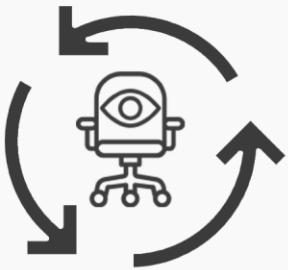


Figura 58. Imágenes del diseño de la carcasa 3D

Tras el proceso de impresión, se realizó un montaje preliminar en el cual se descubrieron algunos fallos de diseño como la necesidad de aumentar la altura de los pilares que soportaban la placa, y, por tanto, de la altura total de la caja, o la necesidad de cambio de ubicación de alguno de los agujeros destinados a los puertos.

Tras realizar dos iteraciones de prototipado se logró la impresión de una versión final con la capacidad necesaria para albergar todos los componentes en el menor espacio posible.

La siguiente imagen muestra las tres versiones de forma comparativa:

Chair Tracker

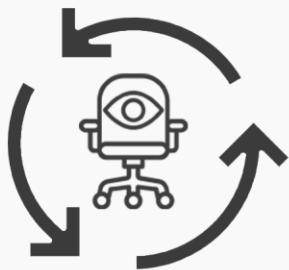
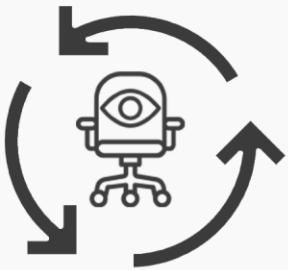


Figura 59. Imagen comparativa entre las tres iteraciones de la carcasa

5.2.5 Sistema final

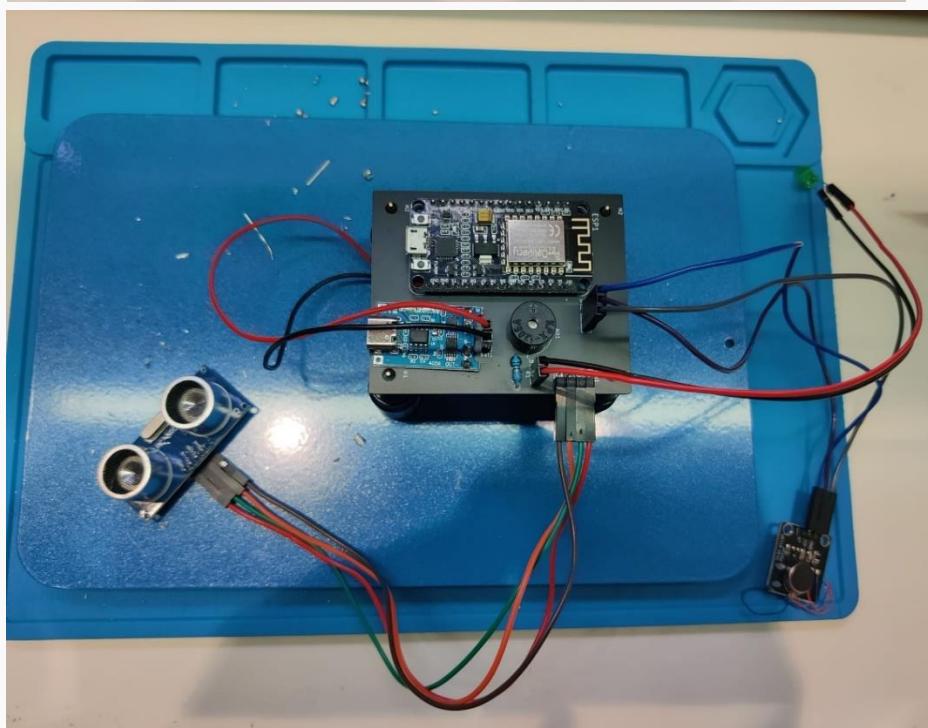
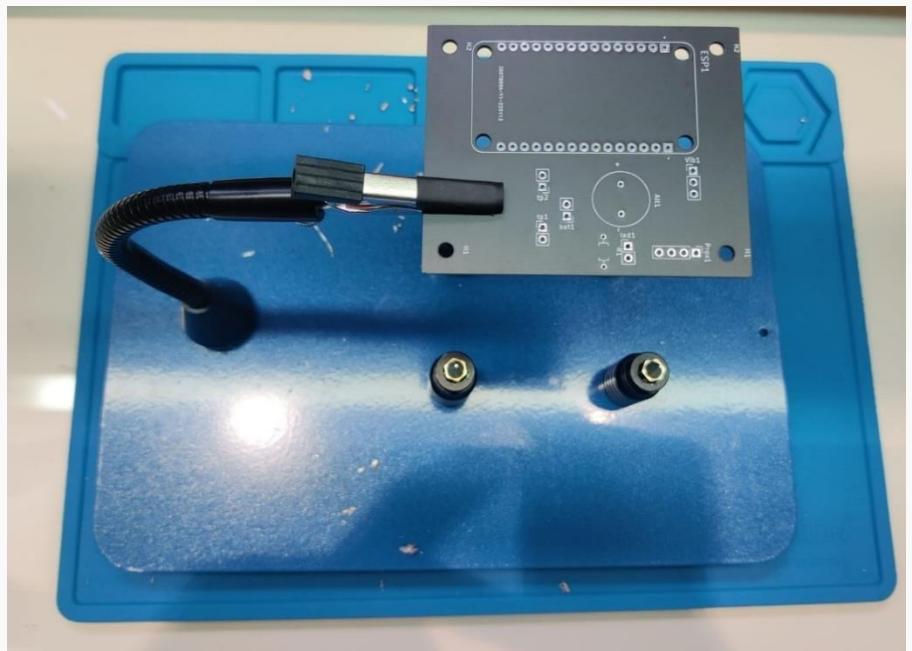
Una vez seleccionados y fabricados los componentes hardware necesarios para la realización del sistema físico final, solamente falta realizar el ensamblaje de la PCB (1) y el montaje del propio circuito integrado en la carcasa del sistema (2).

- 1) Para ensamblar los componentes electrónicos a la PCB, hemos utilizado soldaduras. El proceso consta de la unión de los metales de la PCB (máscara de soldadura) con los componentes a través de un material de unión, en este caso una aleación de estaño a 400°C.



Para facilitar el proceso, se ha hecho uso de una estación de trabajo que permite el ajuste de soportes magnéticos para poder sujetar la PCB y los componentes.

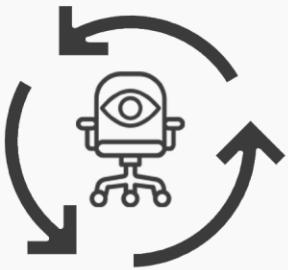
El resultado de este primer paso es el siguiente:



Chair Tracker

Figura 60. **Ensamblado de la PCB**



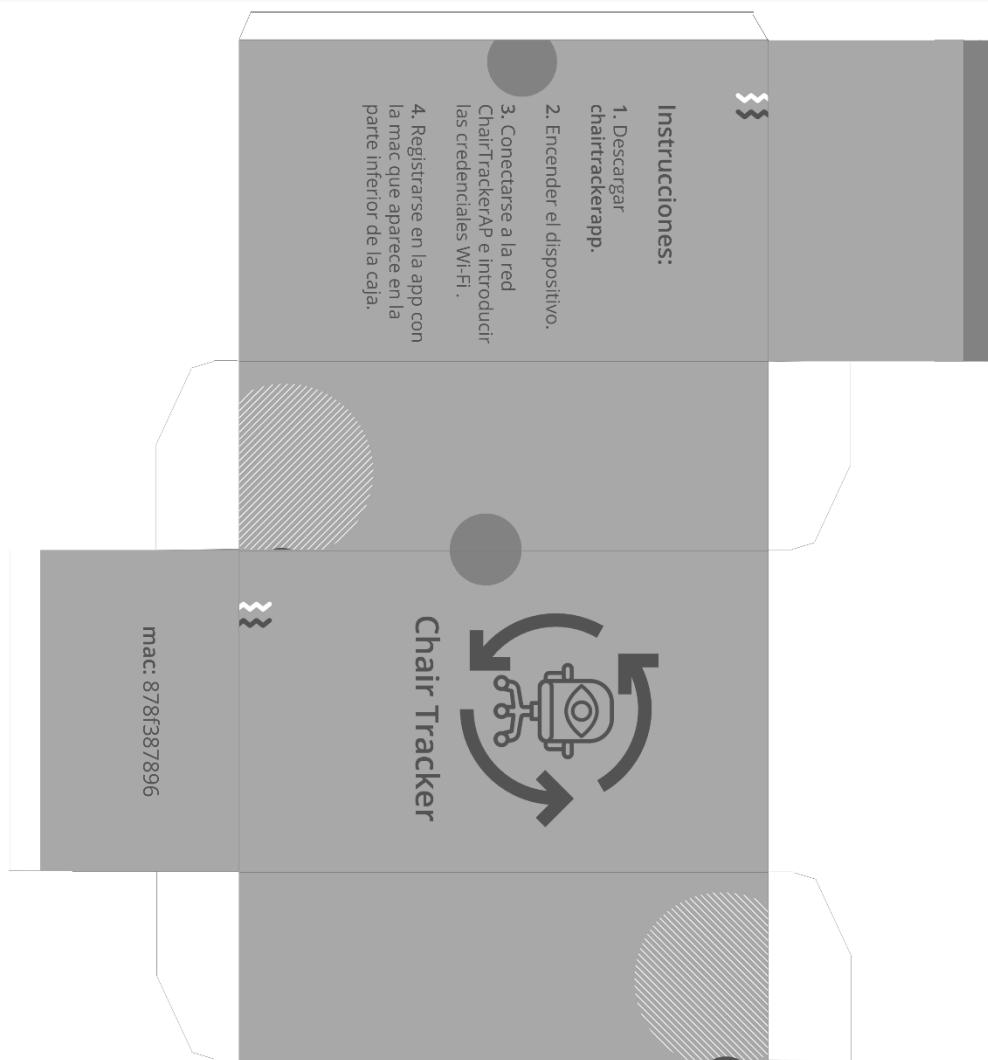


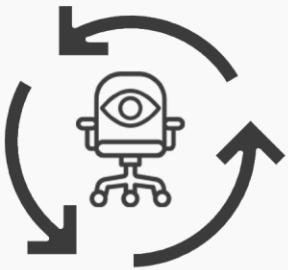
Chair Tracker

- 2) Para el montaje de los componentes tenemos 4 componentes principales en el interior de la carcasa: batería, PCB, vibrador e interruptor debemos de tener en cuenta la necesidad de incluir cablería externa a la PCB.

La batería se encuentra situada en la parte inferior, justo debajo de la PCB que se encuentra atornillada en los pilares, dejando así el espacio suficiente tanto en la parte inferior como superior, donde se encuentra el vibrador.

De esta forma queda finalizado el desarrollo hardware del dispositivo final que será entregado al usuario junto con un packaging ideado para facilitar el registro en el sistema pues proporciona la mac necesaria para el registro, Tal y como se observa a continuación.





Chair Tracker



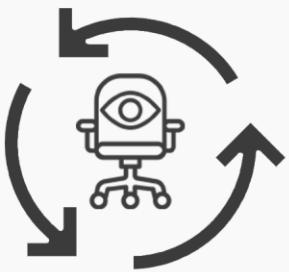
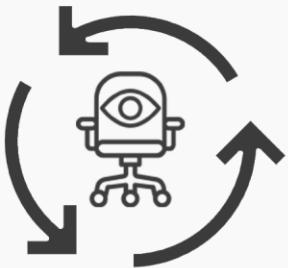


Figura 61. Imágenes del dispositivo final

Chair Tracker

Bloque 6:

Pruebas



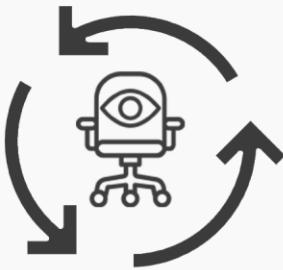
6.1 Pruebas

Para garantizar la calidad de la implementación de nuestro sistema, se han realizado una serie de pruebas tanto hardware como software, dando así un plus de seguridad al usuario de que tanto el funcionamiento como el dispositivo son correctos.

6.1.1 Pruebas del hardware

Las pruebas relacionadas con los componentes hardware del proyecto que se han realizado son:

- *Test de circuito en PCB de prueba:* para comprobar que la PCB está bien diseñada, primero con un polímetro comprobamos que todas las pistas se conectan correctamente y que no hay ningún cortocircuito.
- *Funcionamiento del circuito integrado:* al completar con éxito la prueba anterior, verificaremos que el sistema actúa como se espera y que todos los componentes se encuentran conectados y en funcionamiento.
- *Test de conexionado en PCB final:* se realizan las dos pruebas anteriores sobre la PCB que va en el producto final, de forma que garantizamos que todo el hardware está en buen estado y listo para usarse.
- *Funcionamiento del sistema:* una vez el circuito del sistema final está montado, procedemos a forzar todas las acciones posibles que nuestro sistema tendría que realizar: medida de la distancia, avisos sonoros, vibración...
- *Medidas de voltaje e intensidad de los componentes en el sistema final:* el sistema necesita una alimentación global de 5V,



Chair Tracker

provistos por la batería ya incluida (aunque también se puede conectar vía USB) y la intensidad que circula es de 3.7A.

- **Test de temperatura:** para garantizar que el sistema es capaz de funcionar en la mayoría de las condiciones de trabajo, se ha probado su uso al lado de un radiador (con una temperatura de 45°C aproximadamente) y en un congelador (a unos -8°C aproximadamente).
- **Test de caída:** se han realizado caídas intencionadas del sistema desde diferentes alturas: 0.5m, 1m y 1.5m, para probar la resistencia, sobre todo de la carcasa, a posibles golpes que pueda sufrir el dispositivo.
- **Test de duración de la batería:** el sistema es capaz de aguantar con alarmas activas 12 horas.

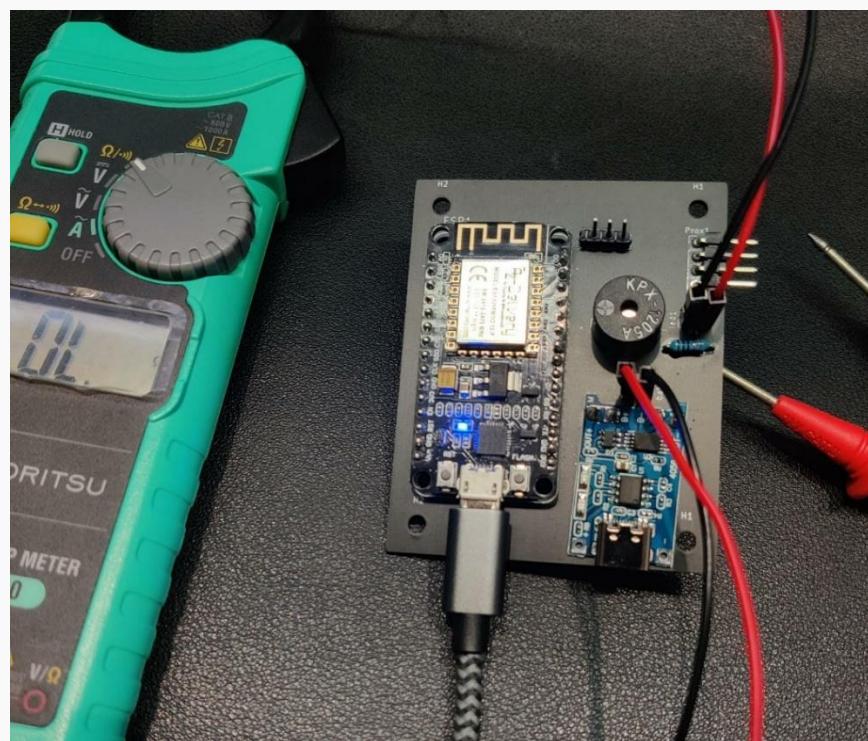
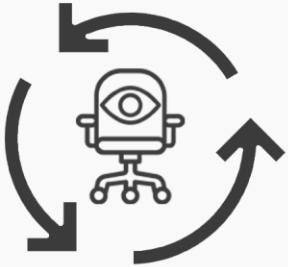


Figura 62. Pruebas de voltaje y amperaje



Todas las pruebas anteriores se han ejecutado con éxito y garantizan tanto la calidad del sistema como el funcionamiento deseado. Además, nos aportan datos muy relevantes de cara a la salida al mercado del producto, así como la resistencia, duración de batería, consumo...

6.1.2 Pruebas del software

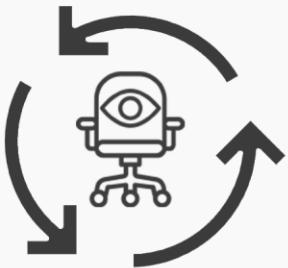
Mientras tanto, las pruebas que han englobado todo el software desarrollado han sido:

- *Test unitarios de la funcionalidad del sistema:* estas pruebas consisten en realizar cada posible acción que pueda realizar el usuario. Por ejemplo, añadir, editar o borrar información, navegación entre distintas pantallas, verificación de datos...
- *Comprobaciones de errores de maquetación:* se han hecho pruebas sobre cada uno de los componentes que conforman las pantallas de la aplicación. Verificando así el correcto funcionamiento global de la aplicación. Además, la app final ha sido compilada, tanto en Android como en iOS, para todos los tamaños de pantalla posibles.

Chair Tracker



Bloque 7: Cierre



7.1 Simulación de implementación en el mercado

Llegados a este punto del proyecto, faltaría únicamente subir las aplicaciones tanto a App Store como a Google Play, que tiene un coste adicional no contemplado en el alcance de nuestro proyecto.

Una vez subidas las aplicaciones, y puesto en marcha la campaña publicitaria, elaborada por un agente de marketing, faltaría establecer los precios y el modelo de negocio que adoptará el proyecto en el futuro con el fin de obtener beneficios.

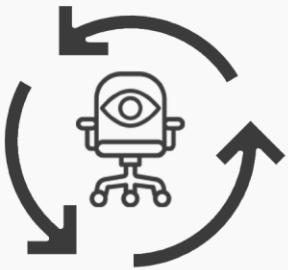
La idea principal es vender tanto el hardware como el software a un precio inicial, al que se le irá sumando el desembolso de una cuota mensual de cada cliente relacionada con el mantenimiento y soporte de nuestro sistema al completo.

Los costes del lanzamiento de la app en Google Play son de 25 dólares iniciales para dar de alta la cuenta de desarrollador, más un 30% de los ingresos con las suscripciones de la aplicación (que se reducirá al 15% al cabo de los primeros 12 meses).

El lanzamiento de la app en App Store es de 99 dólares anuales, tras pasar la aprobación de la aplicación llevada a cabo por la propia plataforma.

El coste material de la producción de un dispositivo hardware es de 42€ aproximadamente (reducibles con la elaboración de numerosos dispositivos).





7.2 Proyectos futuros

Este Trabajo de Fin de Grado se trata de una primera iteración de un proyecto completo. Pero se encuentra abierto a mejoras en varios aspectos:

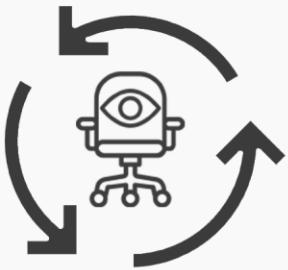
A nivel de hardware se podrían mejorar los componentes y los materiales para obtener un sistema final más potente y con mejores acabados.

En cuanto a la parte software, podríamos incluir nuevas funcionalidades tal y como notificaciones en el móvil del usuario, tener mayor capacidad de alarmas creadas simultáneamente, incluir chats...

Pero para todo esto, también se necesitaría un presupuesto más elevado que el empleado en esta primera versión del proyecto.

Desde otra perspectiva, como comentamos en los objetivos finales del proyecto, se podría expandir este sistema a números sectores y con unos ámbitos de aplicación diferentes a los planteados. Por ejemplo, cabría la posibilidad de implementar el proyecto para la regulación de horas en trabajos de conducción de vehículos (como camioneros, taxistas...), para el establecimiento de objetivos de concentración para niños con TDAH, etc.

Estas últimas ampliaciones solamente necesitarían de refactorizaciones en el back-end (base de datos y API) y posiblemente una elección de una paleta de colores diferente para cada sector en el front-end (aplicación).



7.3 Conclusiones

El inicio de este Trabajo de Fin de Grado se remonta al inicio de la asignatura Desarrollo de Aplicaciones Distribuidas correspondiente al 3º año del grado de Ingeniera de Computadores. Donde tuvo origen la idea, en parte por la situación que nos acontecía en aquellos momentos, el confinamiento.

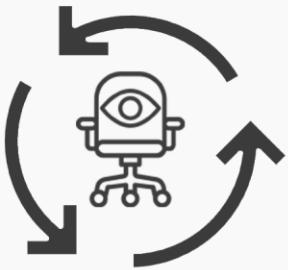
En estos momentos fuimos conscientes de la necesidad de un sistema que nos permitiera contabilizar las horas que pasábamos sentados y la importancia de mantener unos hábitos de trabajo responsables respecto a la higiene postural.

Al inicio de este 4º curso surgió la idea de continuar desarrollando este proyecto como base donde poner a prueba los conocimientos adquiridos a lo largo de toda la carrera, junto con nuestras inquietudes personales en cuanto a nuevas tecnologías se refiere.

De esta forma comenzamos a realizar un estudio técnico en profundidad sobre los requisitos y un análisis de como poder afrontar estos mediante diversas tecnologías que conformarían un sistema completo y escalable. Además de presentar un ecosistema grato y accesible para cualquier tipo de usuario.

De esta forma se concluyó que React Native, hasta entonces desconocida para nosotros, se trataría de uno de los pilares de este nuestro TFG, pues nos permitiría cumplir todos estos requisitos y nos abría un campo nuevo de investigación en la propia tecnología, herramientas e implementación.

De esta forma podemos afirmar que se ha logrado llevar a cabo todos nuestros objetivos de forma satisfactoria a lo largo del desarrollo permitiéndonos



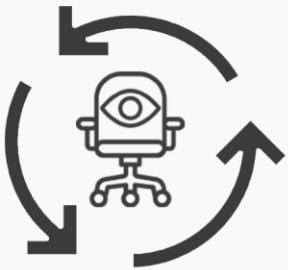
combinar conocimientos adquiridos no solo software en tanto a sistemas empotrados, bases de datos, protocolos, APIs, etc. Sino también al desarrollo de hardware con la selección de componentes, diseño de PCB o diseño 3D.

Dejando hueco para distintas competencias como planificación de proyectos, o adquisición de nuevas capacidades que serán útiles de cara a la inserción en el mundo laboral como, por ejemplo, las normas conventional commits para gestión de versiones, que ha sido tan útil para el desarrollo del proyecto y donde se deja constancia del mismo en el siguiente repositorio: https://github.com/josarcsl/chair_tracker_repository.

Chair Tracker



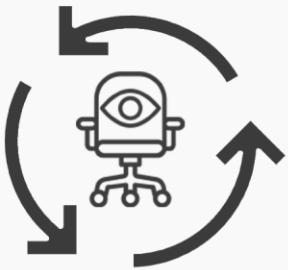
Bibliografía



Chair Tracker

- [1] (s. f.-a). *GitHub - axios/axios: Promise based HTTP client for the browser and node.js.* Axios GitHub. Recuperado 13 de marzo de 2022, de <https://github.com/axios/axios>
- [2] Adobe Illustrator. (2021, 29 septiembre). En *Wikipedia, la enciclopedia libre.* https://es.wikipedia.org/wiki/Adobe_Illustrator
- [3] Adobe Photoshop. (2022, 20 enero). En *Wikipedia, la enciclopedia libre.* https://es.wikipedia.org/wiki/Adobe_Photoshop
- [4] App Store. (2012, 16 febrero). *Xcode.* App Store. Recuperado 21 de enero de 2022, de <https://apps.apple.com/es/app/xcode/id497799835?mt=12>
- [5] Arocha Rodulfo, J. I. (2019). Sedentarism, a disease from xxi century. *Clínica e Investigación en Arteriosclerosis (English Edition)*, 31(5), 233–240. <https://doi.org/10.1016/j.artere.2019.04.001>
- [6] C (lenguaje de programación). (2022, 14 enero). En *Wikipedia, la enciclopedia libre.* [https://es.wikipedia.org/wiki/C_\(lenguaje_de_programaci%C3%B3n\)](https://es.wikipedia.org/wiki/C_(lenguaje_de_programaci%C3%B3n))
- [7] *Chairless - Smart sitting time tracker - Apps on Google Play.* (s. f.). Google Play. Recuperado 30 de enero de 2022, de https://play.google.com/store/apps/details?id=com.runtimeerrorstudio.chairless&hl=en_US&gl=US
- [8] Chandrasekaran, B., & Ganesan, T. B. (2020). Sedentarism and chronic disease risk in COVID 19 lockdown – a scoping review. *Scottish Medical Journal*, 66(1), 3–10. <https://doi.org/10.1177/0036933020946336>
- [9] colaboradores de Wikipedia. (s. f.). *SQL.* Wikipedia, la enciclopedia libre. <https://es.wikipedia.org/wiki/SQL>
- [10] *Comprar suscripción de Fusion 360 / Obtener Precios del software de diseño CAD/CAM de Fusion /Tienda en linea de Autodesk España.* (2022, 13 enero). Fusion 360. Recuperado





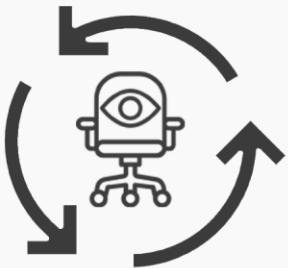
21 de enero de 2022, de

<https://www.autodesk.es/products/fusion-360/overview>

- [11] Devlioti, V. (2018, 24 agosto). *Enough With The Fit Trackers, Here Is the Sit Tracker*. Wersm. Recuperado 30 de enero de 2022, de <https://wersm.com/enough-with-the-fit-trackers-here-is-the-sit-tracker/>
- [12] Eclipse (software). (2021, 19 junio). En *Wikipedia, la enciclopedia libre*. [https://es.wikipedia.org/wiki/Eclipse_\(software\)](https://es.wikipedia.org/wiki/Eclipse_(software))
- [13] ESLint. (2021, 23 diciembre). En *Wikipedia*. <https://en.wikipedia.org/wiki/ESLint>
- [14] F. (s. f.-b). *GitHub - FrozenPyrozen/rn-native-mqtt: An MQTT client for React Native that actually works and exposes a simple Javascript interface*. GitHub. <https://github.com/FrozenPyrozen/rn-native-mqtt>
- [15] Formik. (s. f.). Formik. <https://formik.org/>
- [16] GitHub. (2022, 13 enero). En *Wikipedia, la enciclopedia libre*. <https://es.wikipedia.org/wiki/GitHub>
- [17] (s. f.-c). *GitHub - indiespirit/react-native-chart-kit: React Native Chart Kit: Line Chart, Bezier Line Chart, Progress Ring, Bar chart, Pie chart, Contribution graph (heatmap)*. GitHub. <https://github.com/indiespirit/react-native-chart-kit>
- [18] *Introducción a Android Studio / Desarrolladores de Android /*. (s. f.). Android Developers. Recuperado 21 de enero de 2022, de <https://developer.android.com/studio/intro?hl=es-419>
- [19] *Introduction - Toms planner - 1.* (s. f.). Toms Planner. Recuperado 21 de enero de 2022, de <https://plan.tomsplanner.com/documentation/manual/en/1/en/topic/introduction>
- [20] J. (s. f.-d). *GitHub - jquense/yup: Dead simple Object schema validation*. GitHub. <https://github.com/jquense/yup>

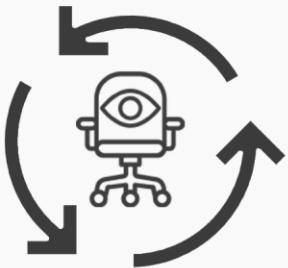


Chair Tracker



Chair Tracker

- [21] Marvel. (s. f.). *Marvel - The design platform for digital products. Get started for free.* <https://marvelapp.com/>
- [22] Mobile UI Kit (Community). (s. f.). Figma. Recuperado 3 de mayo de 2022, de <https://www.figma.com/community/file/94212424128911731> 8
- [23] MySQL :: MySQL Workbench. (s. f.). MySQL Defición. Recuperado 21 de enero de 2022, de <https://www.mysql.com/products/workbench/>
- [24] Nordquist, T. N. (s. f.). *MQTT explorer.* MQTT explorer. Recuperado 21 de enero de 2022, de <http://mqtt-explorer.com/>
- [25] Postman. (s. f.). Definición Postman. Recuperado 21 de enero de 2022, de <https://www.postman.com/>
- [26] R. (s. f.-e). *GitHub - react-native-async-storage/async-storage: An asynchronous, persistent, key-value storage system for React Native.* GitHub. <https://github.com/react-native-async-storage/async-storage>
- [27] R. (s. f.-f). *GitHub - react-native-datetimepicker/datetimepicker: React Native date & time picker component for iOS, Android and Windows.* GitHub. <https://github.com/react-native-datetimepicker/datetimepicker>
- [28] R. (s. f.-g). *GitHub - react-native-picker/picker: Picker is a cross-platform UI component for selecting an item from a list of options.* GitHub. <https://github.com/react-native-picker/picker>
- [29] React Native. (2022, 13 enero). En Wikipedia. https://en.wikipedia.org/wiki/React_Native
- [30] S. (s. f.-h). *GitHub - styled-components/styled-components: Visual primitives for the component age. Use the best bits of ES6 and CSS to style your apps without stress* GitHub. <https://github.com/styled-components/styled-components>
- [31] S. (s. f.-i). *KiCad EDA Loader Download.* Component Search Engine. <https://componentsearchengine.com/library/kicad?gclid=Cj0K>



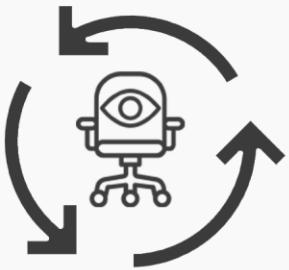
CQjwyYKUBhDJARIsAMj9lkEQVBQ_4wU2-

El_Hf7iR0z_T2s9vYcKaVVIPbVzLQbMKPfJPtZgDAsaAjd
nEALw_wcB

- [32] S. (2019, 12 marzo). *GitHub - SensorsIot/Project-Box-Templates*. GitHub. Recuperado 22 de enero de 2022, de <https://github.com/SensorsIot/Project-Box-Templates>
- [33] symbol 1614977-1 1614977-1. (s. f.). Resistor Symbol&FootPrint. Recuperado 18 de enero de 2022, de <https://www.snapeda.com/parts/1614977-1/TE%20Connectivity/view-part/?ref=search&t=1614977-1>
- [34] symbol 113990105. (s. f.). NodeMCU V2 Symbol&FootPrint. Recuperado 18 de enero de 2022, de <https://www.snapeda.com/parts/NodeMCU%20v2/Seeed%20Technology%20Co.,%20Ltd/view-part/>
- [35] symbol CEM-1203(42) CEM-1203(42). (s. f.). CEM-1203 Symbol&FootPrint. Recuperado 18 de enero de 2022, de [https://www.snapeda.com/parts/CEM-1203\(42\)/CUI%20Devices/view-part/](https://www.snapeda.com/parts/CEM-1203(42)/CUI%20Devices/view-part/)
- [36] TypeScript. (2022, 17 enero). En *Wikipedia, la encyclopédia libre*. <https://es.wikipedia.org/wiki/TypeScript>
- [37] useAxios. (s. f.). Use-Axios-Client. <https://use-axios-client.io/>
- [38] Visual Studio Code - Code Editing. Redefined. (2021, 3 noviembre). Visual Studio Code. Recuperado 21 de enero de 2022, de <https://code.visualstudio.com/>
- [39] W. (s. f.-j). GitHub - wix/react-native-calendars: React Native Calendar Components [31] [31]. GitHub. <https://github.com/wix/react-native-calendars>
- [40] What is Java. (s. f.). Java. Recuperado 21 de enero de 2022, de https://www.java.com/es/download/help/whatis_java.html#:~:text=Java%20es%20un%20lenguaje%20de,en%201995%20por%20Sun%20 Microsystems.&text=Java%20es%20r%C3%A1A

Chair Tracker



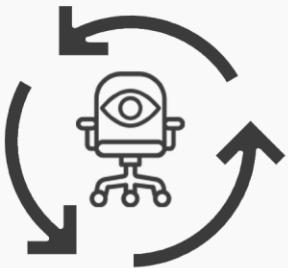


1pido%2C%20seguro%20y,Java%20est%C3%A1%20en%20todas%20partes

- [41] *What is PlatformIO? — PlatformIO latest documentation.*
(s. f.). PlatformIO. Recuperado 21 de enero de 2022, de
<https://docs.platformio.org/en/latest/what-is-platformio.html>

Chair Tracker

Anexos



1. Seguimiento temporal del proyecto

Se incluye un resumen acerca de la planificación temporal realizada al comienzo de este proyecto, frente a los tiempos reales de ejecución de cada fase de este.

Por lo general, nos hemos ajustado correctamente a la planificación realizada, por lo que se podría decir que ha sido una planificación exitosa. En cuanto que se han cumplido los tiempos finales de entrega.

Hemos sufrido algunas desviaciones, tanto positivas como negativas a lo largo del desarrollo del proyecto. Las negativas se han debido principalmente a la formación en una nueva tecnología como lo es React Native. Donde el desconocimiento nos llevó a pensar que sería todo más básico.

Sin embargo, el empleo del tiempo en formación ha reducido los tiempos de desarrollo, debido al profundo aprendizaje de los conceptos y a la rápida resolución de las dificultades de programación encontradas. Permitiéndonos así, incluir al final del proyecto una serie de mejoras que, en nuestra opinión, han profesionalizado el resultado final.

Chair Tracker



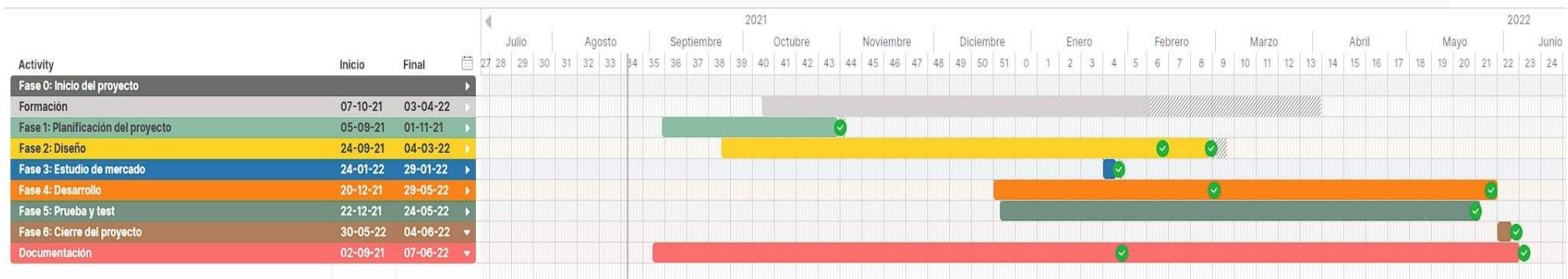


Figura 63. Imágenes del dispositivo final

Chair Tracker

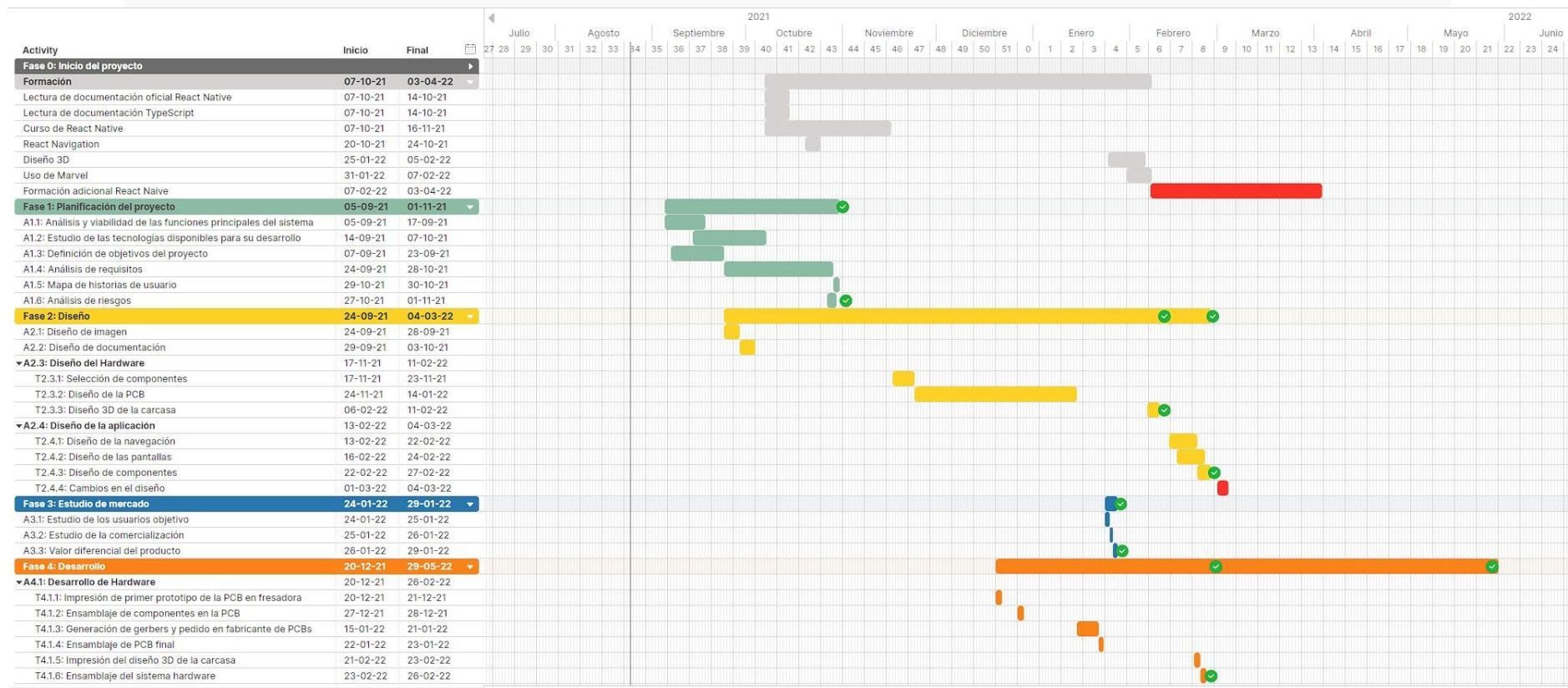


Figura 64. Desviaciones en la planificación I

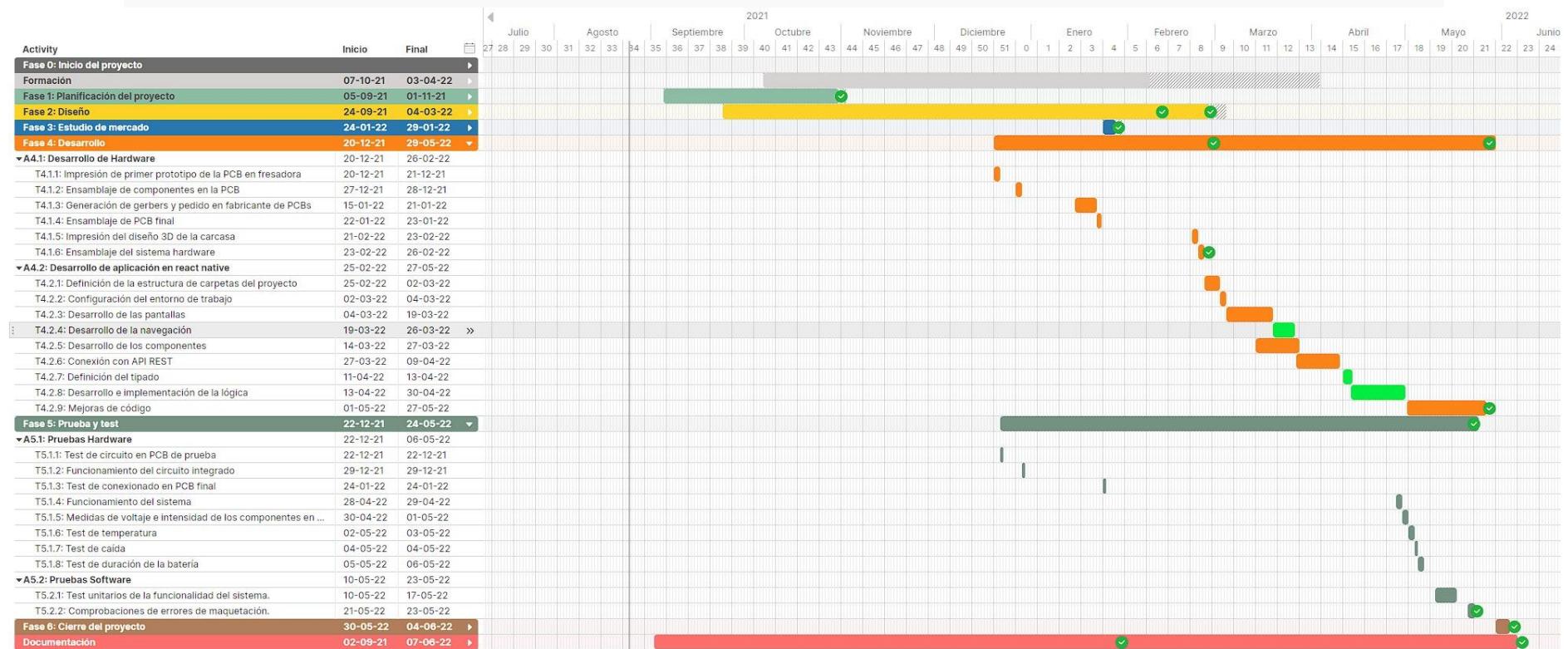


Figura 65. Desviaciones en la planificación II

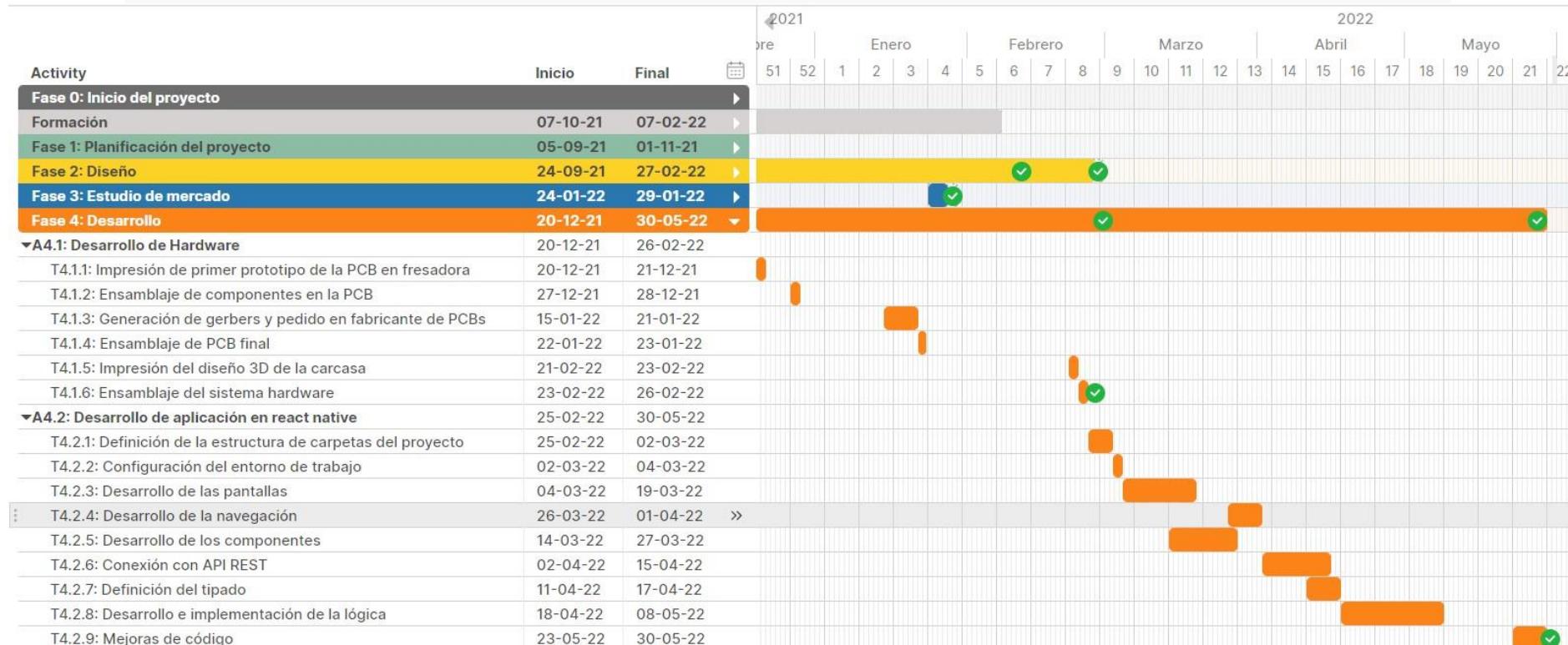
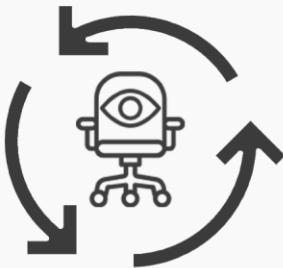


Figura 66. Desviaciones en la planificación III



2. Manual de usuario

En este apartado se expondrá a continuación un manual de usuario que tiene como fin la resolución de cualquier tipo de dudas que le puedan surgir al usuario acerca del uso de la aplicación.

En primer lugar, debemos conectar el dispositivo físico a la red Wi-Fi en la que vaya a ser utilizado. Para ello, deberemos de encender el dispositivo e irnos a los ajustes de conexión de nuestro smartphone donde aparecerá un punto de acceso con el nombre de ChairTrackerAP, como se muestra en la imagen 67. Nos conectaremos a esta red e introduciremos las credenciales de nuestro Wi-Fi.

Este paso solamente será necesario la primera vez que iniciemos el sistema.

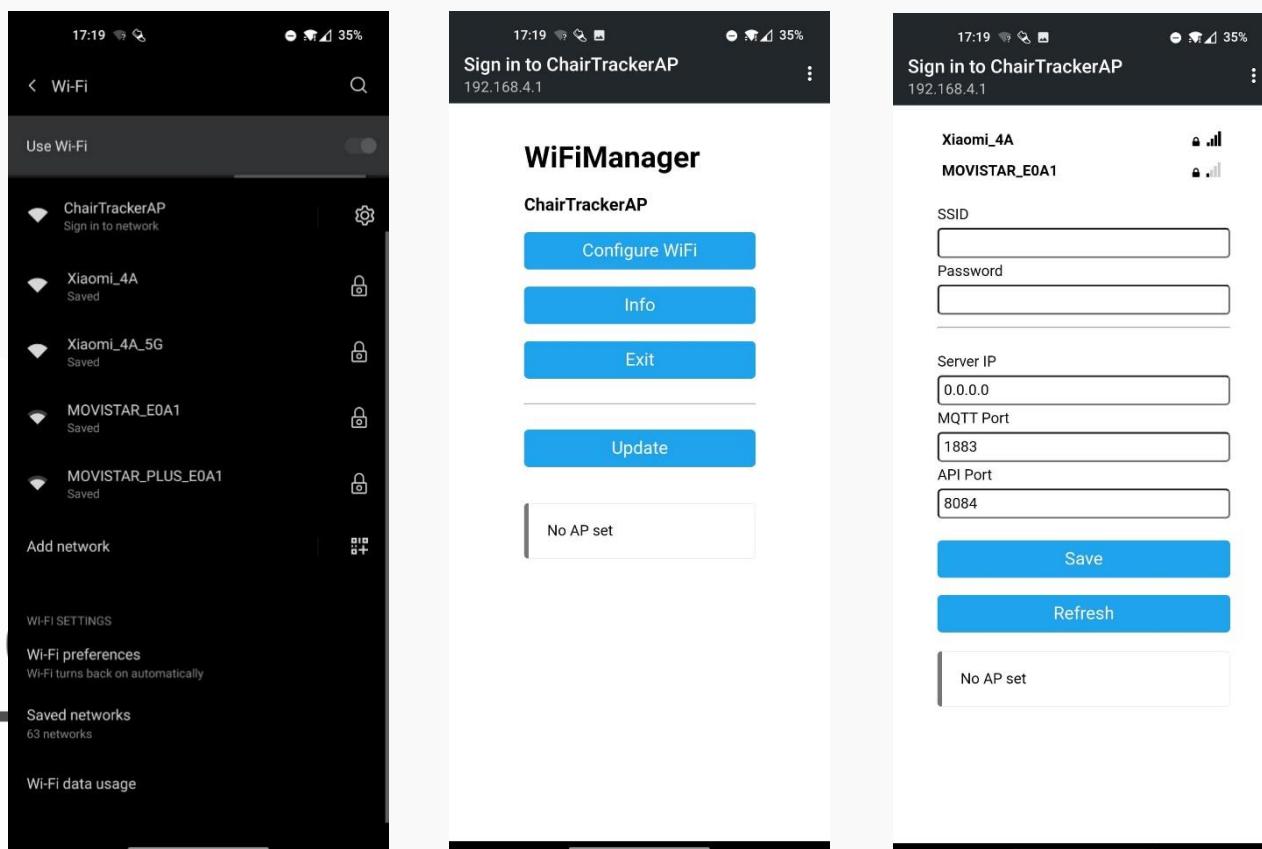


Figura 67. Configuración del AP del dispositivo

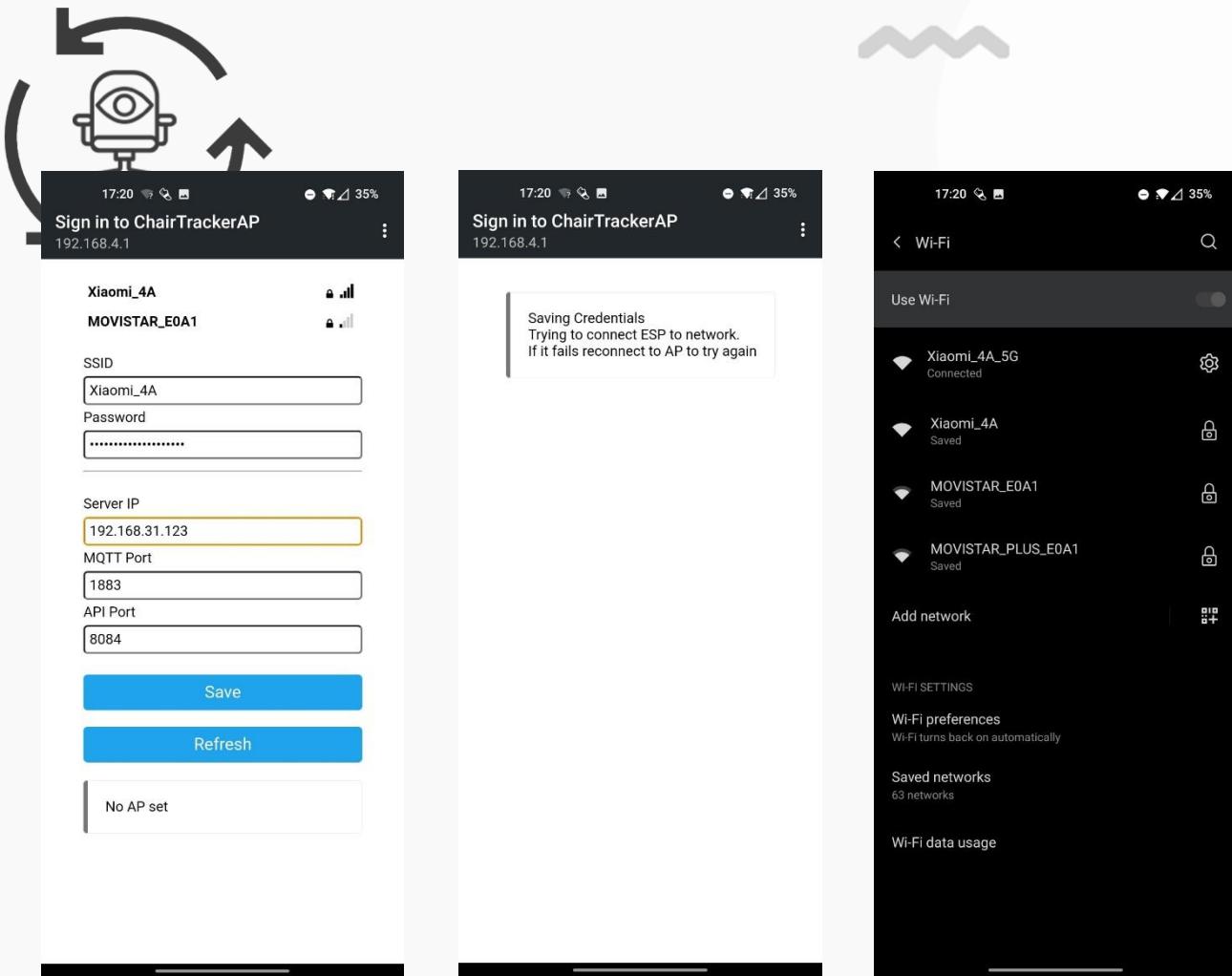


Figura 68. Configuración de credenciales del dispositivo

Una vez realizada la configuración inicial, nada más iniciar la app, nos encontramos con una pantalla de bienvenida en la que tenemos tanto la opción de registrarnos como la de iniciar sesión.

Para registrarnos tendremos que usar el mac (HashMac) que aparece en la caja del dispositivo físico. Una vez tengamos una cuenta, podremos iniciar sesión y empezar a usar la aplicación.

Chair Tracker

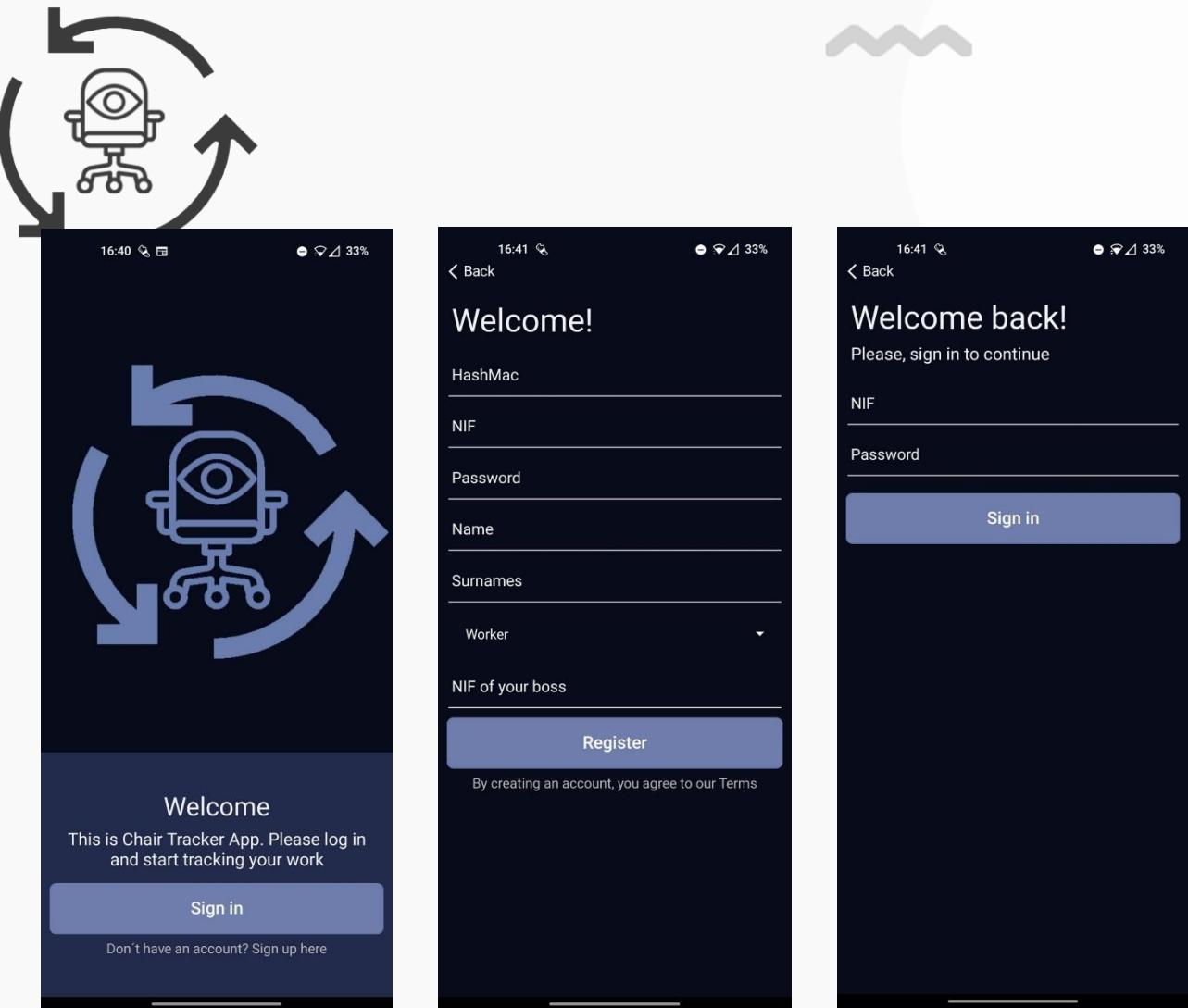
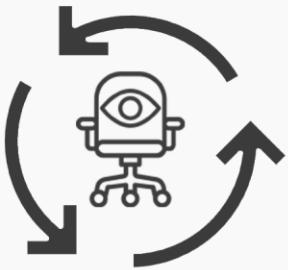


Figura 69. Welcome de la aplicación

Dentro de la app, nos encontramos con una barra de navegación inferior que nos permite movernos fácilmente entre las cuatro pantallas principales (alarmas, contactos, estadísticas y perfil). Por defecto, nos encontraremos en la pantalla de alarmas, que contiene un listado con las alarmas que el usuario tiene planificadas (con un máximo de 5 alarmas que evitarán la saturación del dispositivo físico).

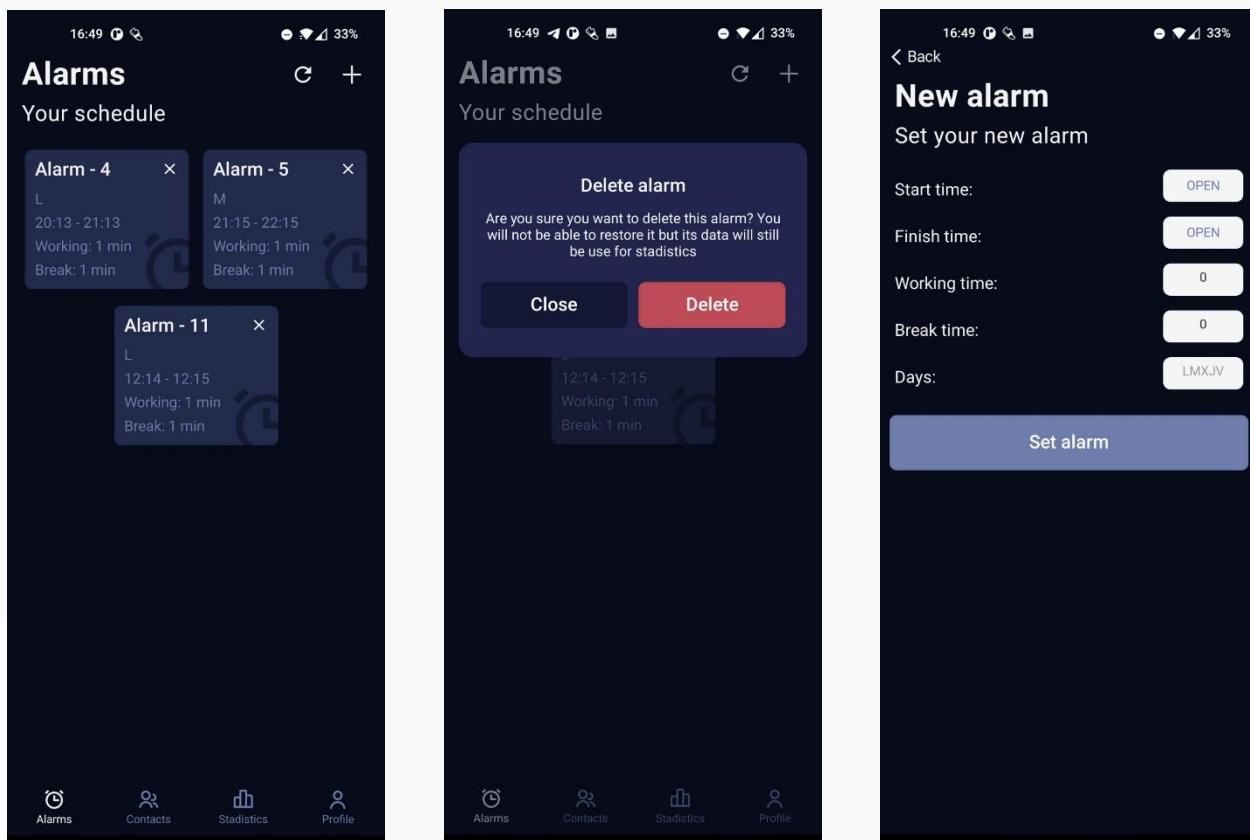
Estas alarmas pueden eliminarse pulsando la cruz que aparece sobre cada tarjeta, mostrándose un mensaje de confirmación para realizar dicho borrado.

Chair Tracker



En caso de necesitar una nueva alarma, pulsaremos el icono de suma, que se encuentra en la esquina superior derecha, que nos llevará a la pantalla de añadir alarma.

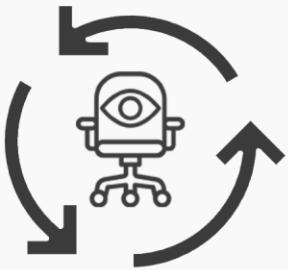
El refresco de la información está programado de forma en la que se realice automáticamente en cuanto haya alguna modificación en los datos, pero el usuario podrá realizar un refresco manual al pulsar el icono de refrescar, también situado en la esquina superior derecha.



Chair Tracker

Figura 70. Sección de alarmas de la aplicación

La siguiente pantalla que encontramos es la de contactos, donde se muestra una lista de usuarios dependiendo del rol que tenga el usuario logeado. Si es jefe dispondrá de una lista con sus empleados y con los demás jefes. Por el contrario, si es empleado, tendrá la lista rellena con su jefe y con sus compañeros de trabajo.



Desde esta pantalla, podremos realizar llamadas a cualquier usuario que aparezca en el listado. Además, si el rol es jefe, tendrá la posibilidad de establecer alarmas a sus empleados.

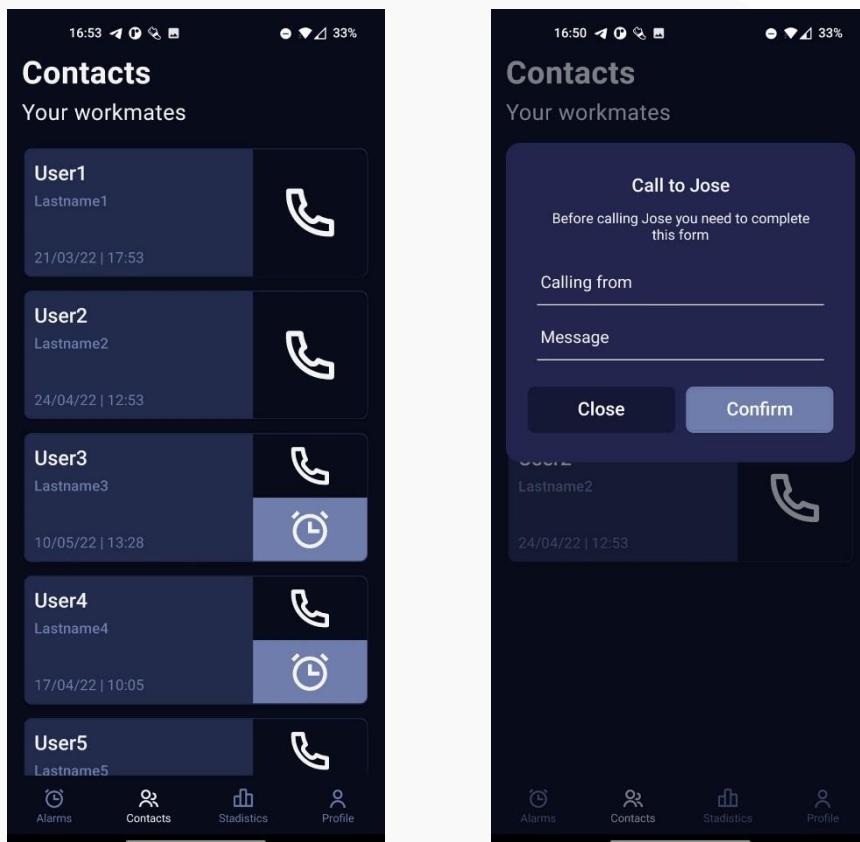


Figura 71. Sección de contactos de la aplicación

La tercera pantalla trata de un listado con las estadísticas de los tiempos de trabajo y descanso del usuario clasificado por alarmas, donde, al igual que en la pantalla de alarmas, existe un refresco automático, pero también se puede realizar manualmente.

**Chair
Tracker**

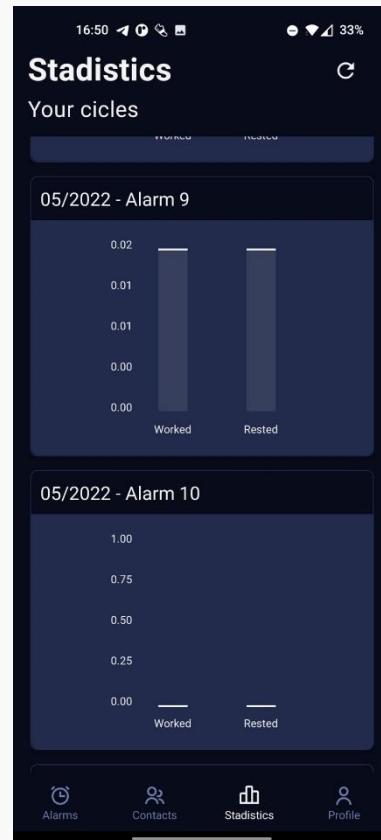
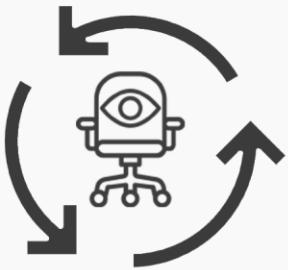


Figura 72. Sección de estadísticas de la aplicación

La última pantalla principal es la del perfil, donde el usuario puede consultar su información. Dispone de un calendario el cual permite, al pulsar cualquier día, ver las alarmas programadas para ese día. También aparece un botón principal y dos secundarios.

El principal nos lleva al registro de llamadas entrantes y salientes del usuario (posee refresco automático). Para movernos entre las llamadas entrantes y salientes, podemos usar tanto los botones de la parte superior, como el gesto de deslizar hacia un lado.

Uno de los botones secundarios nos ofrece el “Acerca de” de la aplicación, mientras que el otro contiene los “Términos y condiciones”.

En la esquina superior derecha encontraremos un botón que permitirá cerrar la sesión del usuario.

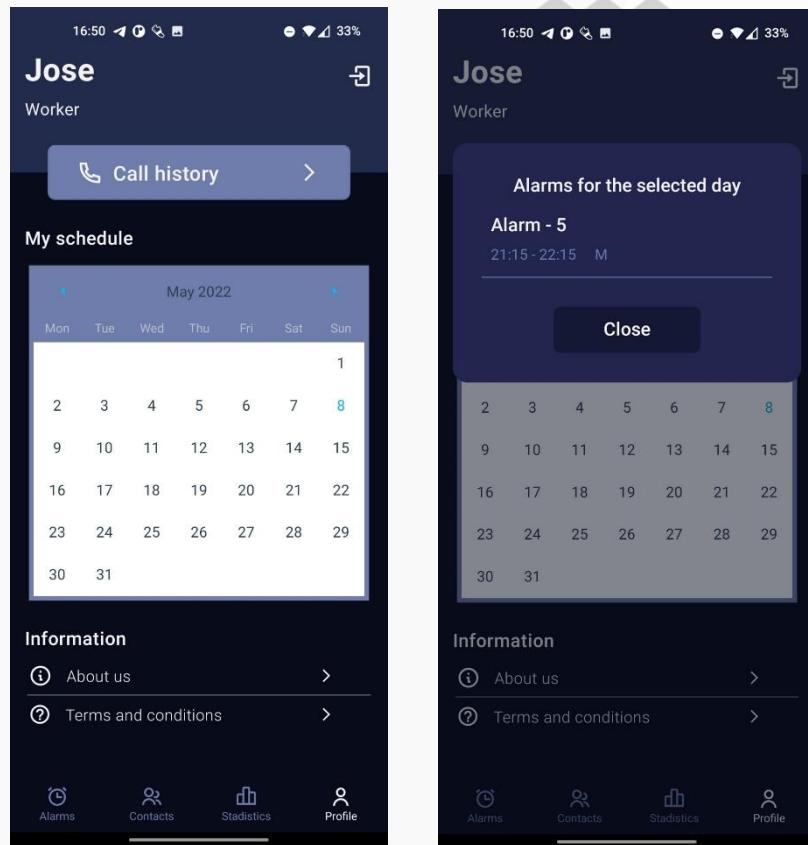
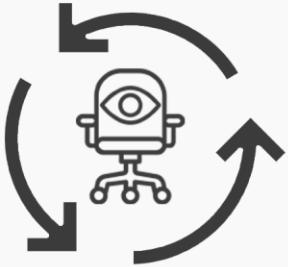
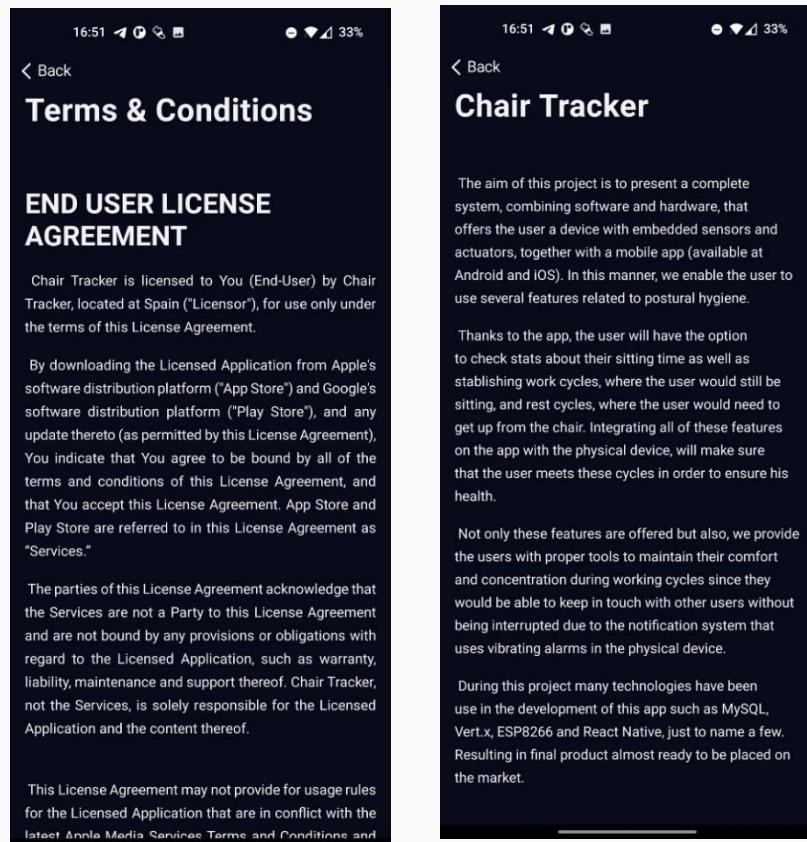
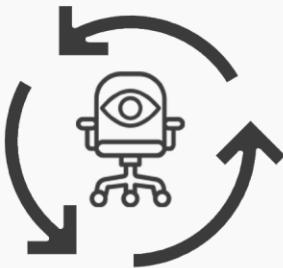


Figura 73. Sección de perfil de la aplicación



Chair Tracker

Figura 74. Sección de términos y condiciones de la aplicación



3. Actas de reuniones con el tutor

17/09/2021

Acotación de los requisitos generales del proyecto.

07/10/2021

Discusión y elección de las tecnologías usadas en el proyecto. Selección del curso de React Native a realizar.

24/10/2021

Reunión donde se mostraron los requisitos funcionales y acotación sobre el índice a trabajar en la documentación correspondiente.

16/11/2021

Actualización del estado del curso y documentación.

13/01/2022

Discusión sobre tecnologías a usar en el desarrollo del dispositivo físico.

31/01/2022

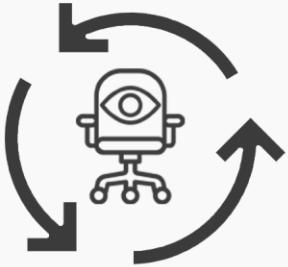
Corrección de la memoria por parte del tutor e indicaciones para la continuación de la documentación, así como la selección de herramientas para el primer mock-up de la app.

15/02/2022

Corrección de los mock-ups y actualización del estado del proyecto hardware y memoria. Discusión sobre activar el modo punto de acceso en el ESP.

07/03/2022

Decisión sobre librería a utilizar para conectar a la API Rest la aplicación y decisión definitiva sobre la librería del ESP y activar el punto de acceso.



07/04/2022

Reunión sobre diversas decisiones relativas a la conexión del smartphone del usuario, dispositivo físico y la app.

03/05/2022

Actualización sobre la primera iteración completamente funcional de la aplicación y feedback sobre tareas aún por implementar.

07/05/2022

Resolución de dudas sobre problemas derivados del uso de librería MQTT que fueron resueltas gracias al uso de un fork de la misma.

18/05/2022

Pequeñas correcciones y feedback sobre el sistema completo en funcionamiento y sobre la sección de implementación de la aplicación en la documentación.

Chair Tracker