

# **Diseño y desarrollo de una aplicación móvil para la monitorización de personas de la 3ª edad**

Memoria de Proyecto Final de Máster

**Máster Universitario en Aplicaciones Multimedia**

Área Profesionalizadora

**Autor: José Manuel Castellano Domínguez**

Consultor: Sergio Schvarstein Liuboschetz

Profesor: Laura Porta Sim

03/01/2020

## Créditos/Copyright



Esta obra está sujeta a una licencia de Reconocimiento

[3.0 España de Creative Commons](https://creativecommons.org/licenses/by/3.0/es/)

© (José Manuel Castellano Domínguez)

Reservados todos los derechos. Está prohibido la reproducción total o parcial de esta obra por cualquier medio o procedimiento, comprendidos la impresión, la reprografía, el microfilme, el tratamiento informático o cualquier otro sistema, así como la distribución de ejemplares mediante alquiler y préstamo, sin la autorización escrita del autor o de los límites que autorice la Ley de Propiedad Intelectual.

## FICHA DEL TRABAJO FINAL

<b>Título del trabajo:</b>	<i>Diseño y desarrollo de una aplicación móvil para la monitorización de personas de la 3ª edad.</i>
<b>Nombre del autor:</b>	<i>José Manuel Castellano Domínguez</i>
<b>Nombre del consultor/a:</b>	<i>Sergio Schvarstein Liuboschetz</i>
<b>Nombre del PRA:</b>	<i>Laura Porta Simó</i>
<b>Fecha de entrega (mm/aaaa):</b>	01/2020
<b>Titulación::</b>	<i>Plan de estudios del estudiante</i>
<b>Área del Trabajo Final:</b>	<i>Area TFM Profesionalizadora</i>
<b>Idioma del trabajo:</b>	<i>Castellano</i>
<b>Palabras clave</b>	<i>Webapp, bienestar, camiseta inteligente</i>
<p><b>Resumen del Trabajo (máximo 250 palabras):</b> <i>Con la finalidad, contexto de aplicación, metodología, resultados i conclusiones del trabajo.</i></p> <p>Este proyecto de profesionalización consiste en el diseño y desarrollo de una aplicación móvil, implementada como una Web App, que permita la monitorización de una serie de constantes vitales de los ancianos que lleven puesta una camiseta inteligente que será la encargada de suministrar la información que podrá ser visualizada desde la App.</p> <p>La idea del proyecto viene a raíz de una idea de negocio. El negocio que se quiere llevar cabo tiene 2 líneas de trabajo</p> <ul style="list-style-type: none"> <li>- Diseño y montaje de productos electrónicos usando microcontroladores y sensores basados en <i>Arduino</i>.</li> <li>- Diseño e implementación de aplicaciones móviles que permitan interactuar con el producto electrónico diseñado en la primera línea de negocio.</li> </ul> <p>El primer proyecto que se va a llevar a cabo es diseñar una camiseta inteligente capaz de obtener una serie de constantes vitales y permitir que los datos obtenidos puedan ser consultados posteriormente por una aplicación móvil.</p> <p>El diseño de la camiseta inteligente queda excluido íntegramente del ámbito de este TFM. Queda fuera del ámbito del TFM, tanto el diseño y montaje del circuito electrónico, como la implementación del código necesario para hacer funcionar dicho producto.</p> <p>En este TFM se realizará el diseño y desarrollo de una aplicación móvil, como una Web App, que pueda ser instalada en teléfonos bajo sistemas operativos Android e IOS. A lo largo del TFM se diseñará y desarrollará la aplicación móvil que debe ajustarse a las funcionalidades que permita la camiseta inteligente.</p>	

**Abstract (in English, 250 words or less):**

This professionalization project consists of the design and development of a mobile application, implemented as a Web App. This app will allow the monitoring of a series of vital signs of the elderly who wear a smart shirt that will be responsible for providing the information that can be seen in the App.

The idea of the project comes from a business idea. The business to be carried out has 2 lines of work but they are related to each other.

- Design and assembly of electronic products using microcontrollers and sensors based on *Arduino*.
- Design and implementation of mobile applications that allow interacting with the electronic product designed in the first line of business.

The first project to be carried out is to design a smart shirt capable of obtaining a series of vital signs and allowing the data obtained to be consulted later by a mobile application.

The smart shirt design is completely excluded from the scope of this Master's Thesis. It is outside the scope of the Master's Thesis, both the design and assembly of the electronic circuit, as well as the implementation of the code necessary to operate said product.

In this Master's Thesis we'll perform the design and development of a mobile application, such as a Web App, that can be installed on phones under Android and IOS operating systems. Throughout the Master's Thesis, the mobile application will be designed and developed that must conform to the functionalities that the smart shirt allows.

## **Dedicatoria/Cita**

A mi querida Mariola ..., que tantas noches se ha pasado a mi lado durmiendo mientras realizaba mis sueños.

## Agradecimientos

A Paöla, Isa, Paula e Esther por ayudarme a entender la problemática a solucionar y ayudarme a definir las interfaces y la funcionalidad.

A Juan Carlos por dedicarse al desarrollo de la camiseta inteligente.

## Abstract

Este proyecto de profesionalización consiste en el diseño y desarrollo de una aplicación móvil, implementada como una Web App, que permita la monitorización de una serie de constantes vitales de los ancianos que lleven puesta una camiseta inteligente que será la encargada de suministrar la información que podrá ser visualizada desde la App.

La idea del proyecto viene a raíz de una idea de negocio. El negocio que se quiere llevar cabo tiene 2 líneas de trabajo

- Diseño y montaje de productos electrónicos usando microcontroladores y sensores basados en *Arduino*.
- Diseño e implementación de aplicaciones móviles que permitan interactuar con el producto electrónico diseñado en la primera línea de negocio.

El primer proyecto que se va a llevar a cabo es diseñar una camiseta inteligente capaz de obtener una serie de constantes vitales y permitir que los datos obtenidos puedan ser consultados posteriormente por una aplicación móvil.

El diseño de la camiseta inteligente queda excluido íntegramente del ámbito de este TFM. Queda fuera del ámbito del TFM, tanto el diseño y montaje del circuito electrónico, como la implementación del código necesario para hacer funcionar dicho producto.

En este TFM se realizará el diseño y desarrollo de una aplicación móvil, como una Web App, que pueda ser instalada en teléfonos bajo sistemas operativos Android e IOS. A lo largo del TFM se diseñará y desarrollará la aplicación móvil que debe ajustarse a las funcionalidades que permita la camiseta inteligente.

---

This professionalization project consists of the design and development of a mobile application, implemented as a Web App. This app will allow the monitoring of a series of vital signs of the elderly who wear a smart shirt that will be responsible for providing the information that can be seen in the App.

The idea of the project comes from a business idea. The business to be carried out has 2 lines of work but they are related to each other.

- Design and assembly of electronic products using microcontrollers and sensors based on *Arduino*.
- Design and implementation of mobile applications that allow interacting with the electronic product designed in the first line of business.

The first project to be carried out is to design a smart shirt capable of obtaining a series of vital signs and allowing the data obtained to be consulted later by a mobile application.

The smart shirt design is completely excluded from the scope of this Master's Thesis. It is outside the scope of the Master's Thesis, both the design and assembly of the electronic circuit, as well as the implementation of the code necessary to operate said product.

In this Master's Thesis we'll perform the design and development of a mobile application, such as a Web App, that can be installed on phones under Android and IOS operating systems. Throughout the Master's Thesis, the mobile application will be designed and developed that must conform to the functionalities that the smart shirt allows.

### **Palabras clave**

Android, IOS, Web App, 3ª edad, salud, constantes vitales, camiseta inteligente



## Notaciones y Convenciones

- Las palabras pertenecientes al glosario se escribirán en cursiva
- El código fuente se mostrará en cursiva y con fuente calibri.

# Índice

<b>Capítulo 1: Introducción.....</b>	<b>15</b>
1.Contexto y justificación del trabajo .....	15
2. Definición del trabajo a realizar .....	17
3. Objetivos generales .....	17
3.1 Objetivos principales.....	18
4. Metodología y proceso de trabajo.....	19
5. Planificación.....	19
6. Presupuesto .....	20
7. Productos finales .....	21
8. Estructura del resto del documento .....	22
<b>Capítulo 2: Análisis .....</b>	<b>23</b>
1. Estado del arte.....	23
2. Análisis del mercado .....	24
2.1 Salud Conectada .....	25
2.2 Heart Rate OS – Android Watch.....	26
2.3 Health Wearable.....	26
2.4 Resumen aplicaciones vistas: .....	27
3. Público Objetivo .....	28
4. Especificaciones del producto .....	30
<b>Capítulo 3: Desarrollo.....</b>	<b>31</b>
<b>3.1 Sprint 1 .....</b>	<b>33</b>
Pantalla de Constantes Vitales.....	33
Entrevistas .....	33
Prototipos .....	36
Pantallas finales realizadas .....	37
Notaciones sobre la implementación .....	37
Notaciones sobre el proceso de pruebas o test.....	39
Pantalla de Login .....	39
Entrevistas .....	40
Prototipos .....	41
Pantallas finales realizadas .....	42
Notaciones sobre la implementación .....	42
<b>3.2 Sprint 2 .....</b>	<b>45</b>

Pantalla de Registro .....	45
Entrevistas .....	46
Prototipos .....	47
Pantallas finales realizadas .....	47
Notaciones sobre la implementación .....	47
Pantalla de Listar Camisetas .....	48
Entrevistas .....	48
Prototipos .....	49
Pantallas finales realizadas .....	49
Notaciones sobre la implementación .....	50
<b>3.3 Sprint 3 .....</b>	<b>50</b>
Pantalla de Registrar y Editar Camisetas .....	50
Entrevistas .....	50
Prototipos .....	52
Pantallas finales realizadas .....	53
Notaciones sobre la implementación .....	53
Pantalla de Definición de Umbrales .....	58
Entrevistas .....	58
Prototipos .....	59
Pantallas finales realizadas .....	59
Notaciones sobre la implementación .....	59
<b>3.4 Sprint 4 .....</b>	<b>61</b>
Notificaciones (Frontend) .....	61
Entrevista .....	61
Prototipos .....	63
Pantallas finales realizadas .....	63
Notaciones sobre la implementación .....	63
Constantes Vitales (Generación de Datos) .....	66
<b>3.5 Sprint 5 .....</b>	<b>69</b>
Notificaciones Backend .....	69
Entrevista .....	69
Prototipos .....	70
Notaciones sobre la implementación .....	71
<b>3.6 Sprint 6 .....</b>	<b>75</b>
Datos adicionales en la camiseta .....	75
Prototipos .....	76
Notaciones sobre la implementación .....	77

Valor mínimo y máximo de las constantes vitales .....	77
Prototipos .....	78
Notaciones sobre la implementación .....	79
<b>Capítulo 4: Backend.....</b>	<b>80</b>
1. Características del Servidor Backend .....	80
2. Características App móvil .....	80
2. Características Servidor Web .....	81
<b>Capítulo 5: Pruebas de Test.....</b>	<b>84</b>
1. Periodo y modo de distribución .....	84
2. Obtención de datos y resultados.....	86
<b>Capítulo 6: Pruebas Realizadas .....</b>	<b>88</b>
<b>Capítulo 7: Conclusiones y líneas de futuro .....</b>	<b>94</b>
1. Conclusiones .....	94
2. Líneas de futuro.....	96
<b>Bibliografía .....</b>	<b>98</b>
<b>Anexo A: Glosario .....</b>	<b>102</b>
<b>Anexo B: Entregables del proyecto .....</b>	<b>105</b>
<b>Anexo C: Fichero de planificación inicial.....</b>	<b>106</b>
<b>Anexo D: Colección Pruebas .....</b>	<b>107</b>
<b>Anexo E: Perfiles de los Early Adopters .....</b>	<b>108</b>
Esther Ridaura.....	108
Paula Estivaliz .....	109
Isa Martinez .....	110
Paõla Sousa.....	112
<b>Anexo F: Curriculum Vitae .....</b>	<b>114</b>
<b>Anexo G: Transcripciones de entrevistas .....</b>	<b>115</b>

## Figuras y tablas

### Índice de figuras

Figura 1 - Estadística del Padrón Continuo a 1 de enero de 2018 (Fuente: INE) .....	15
Figura 2 - Afectación de la vida diaria. (Fuente: [3]).....	16
Figura 3 - Curva de mercado por sector (Fuente[20]) .....	29
Figura 4 - Diagrama de Navegación .....	31
Figura 5 - Ejemplo Latido Corazón (Fuente [26]) .....	36
Figura 6 - Prototipos Constantes Vitales.....	36
Figura 7 - Versión final Constantes Vitales .....	37
Figura 8 - Prototipos Login.....	41
Figura 9 - Versión Final Pantalla de Login .....	42
Figura 10 - Solicitud Permisos Facebook.....	44
Figura 11 - Resetear contraseña o recuperar cuenta (Desde PC) .....	45
Figura 12 - Prototipos Registro .....	47
Figura 13 - Versión Final Registro de Cuenta .....	47
Figura 14 - Prototipos Listar Camiseta.....	49
Figura 15 - Versión Final Listar Camisetas .....	49
Figura 16 - Prototipos Crear/Editar camisetas .....	52
Figura 17 - Versión Final Crear Editar Camiseta.....	53
Figura 18 - Combo de Icono con Imagenes .....	57
Figura 19 - Prototipo definición umbrales.....	59
Figura 20 - Versión final definición umbrales .....	59
Figura 21 - Prototipo Notificaciones .....	63
Figura 22 - Versión final Notificaciones .....	63
Figura 23 - Arquitectura de Firebase Cloud Messaging (Fuente: Microsoft) .....	72
Figura 24 - Prototipos nuevos campos.....	76
Figura 25 - Pantallas Nuevos campos implementadas .....	77
Figura 26 - Prototipos Visualizaciones adicionales .....	78
Figura 27 - Pantallas implementadas Visualizaciones adicionales .....	78
Figura 28 - Estructura del servicio Restful .....	81
Figura 29 - Versión Alfa 0.0.5 de Healthshirt .....	85
Figura 30 - Administración de Testers .....	86
Figura 31 - Ejemplo logs de depuración.....	86
Figura 32 - Ejemplo error en Crashlitycs.....	87
Figura 33 - Ejemplo error de bloqueo en Crashlitycs .....	87

## Índice de tablas

Tabla 1 - Distribución de horas por sprint .....	21
Tabla 2 - Distribución de horas a lo largo del proyecto .....	21
Tabla 3 - Pruebas autenticación.....	90
Tabla 4 - Pruebas Listado, Añadir y Editar Camisetas .....	91
Tabla 5 - Pruebas Constantes.....	91
Tabla 6 - Pruebas Notificaciones .....	93

# Capítulo 1: Introducción

## 1.Contexto y justificación del trabajo

Actualmente la sociedad española va envejeciendo lentamente. Según el informe más reciente del CSIC [1] hay más de 8 millones de personas en la tercera edad y se espera que este valor se incremente a medida que pasen los años. Tal y como se muestra en la figura 1, el grupo de edad que abarca la mayor población es el grupo de gente que comprende entre los 40 y los 50 años. Eso significa que en unos escasos 25 años el grupo de población de la 3ª edad sea el más numeroso.

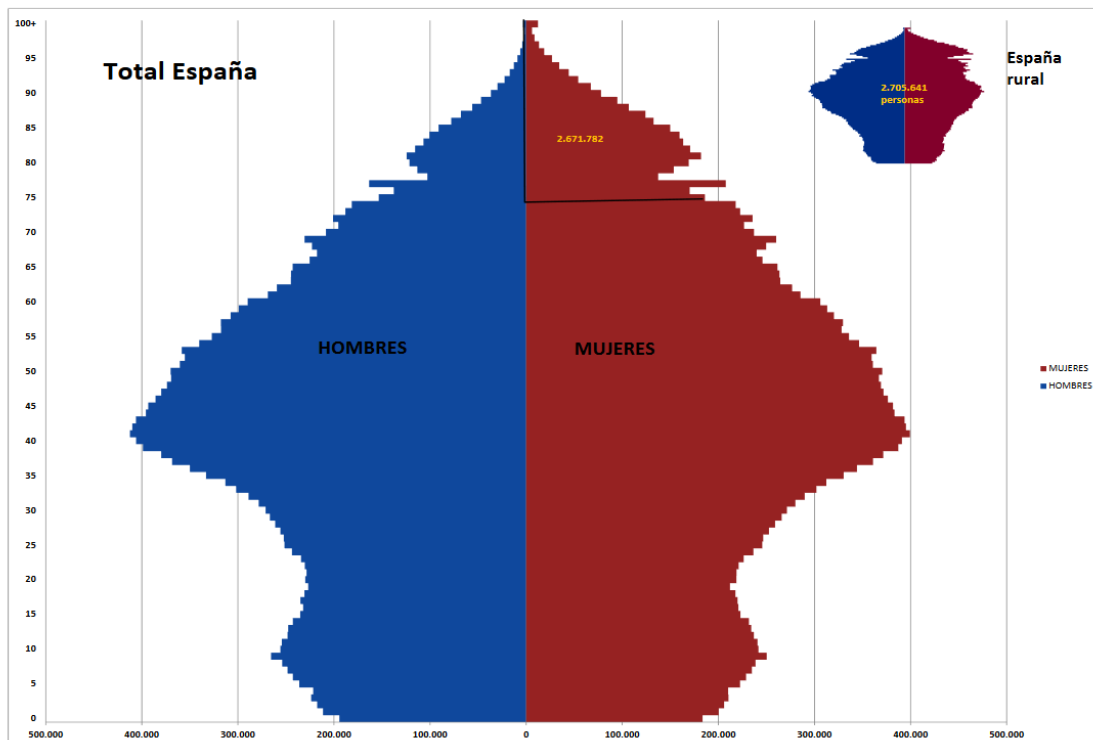


Figura 1 - Estadística del Padrón Continuo a 1 de enero de 2018 (Fuente: INE)

El cuidado de nuestros mayores implica un gran coste monetario que tal y como se indica en un artículo del periódico *El Mundo* en el año 2017 [2] se espera que se incremente a un ritmo de 580 millones de euros por año en los próximos 10 años debido al crecimiento de la población de la 3ª edad. Además del coste monetario, también existe un gran coste en el tiempo invertido en el cuidado de la gente de la 3ª edad. Un estudio realizado por el CSIC [3] en colaboración con SEGG y LINDOR, indica que más del 88% de los cuidadores ven muy afectada su vida por el cuidado de gente mayor. En la figura 2 se puede observar una gráfica del citado informe, donde se desglosa en 9 niveles en cómo afecta a la vida de los cuidadores, el cuidado de la gente mayor (donde 1 es que apenas afecta a su vida y 9 que afecta completamente a su vida). Se observa que más del 88% de los cuidadores están en el nivel 5 o superior de cómo afecta esta actividad a sus vidas.

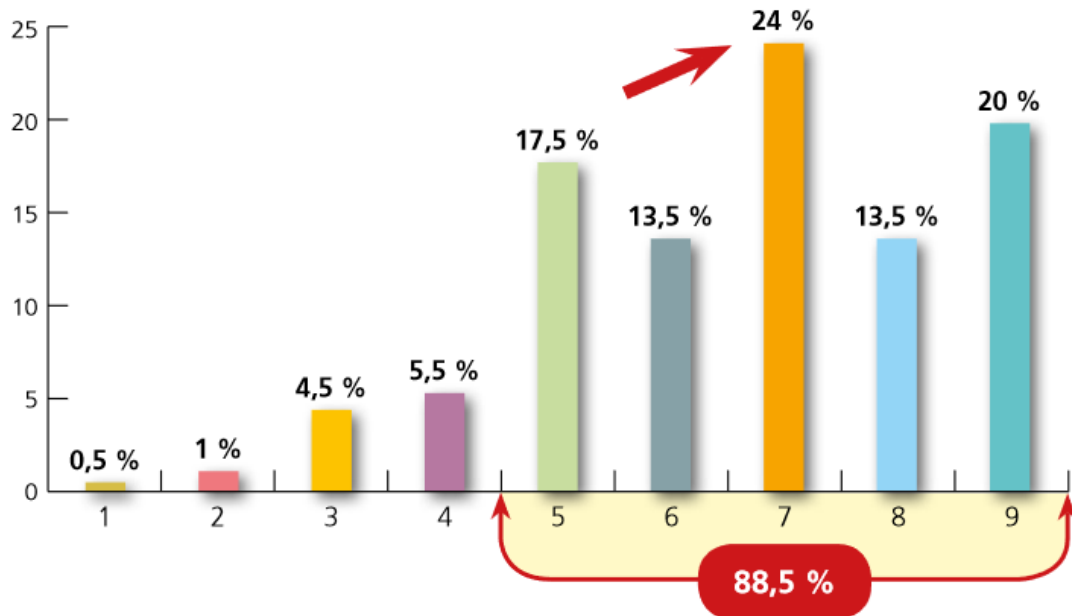


Figura 2 - Afectación de la vida diaria. (Fuente: [3])

La mayoría de la gente que tiene que cuidar de sus mayores, no posee ni dinero ni tiempo para encargarse de sus mayores. Algunas personas se limitan simplemente a enviar a sus padres/abuelos a una residencia, despreocupándose completamente del tema.

Además, aquellos que no pueden o no quieren enviar a sus padres/abuelos o aquellos ancianos que no tienen ninguna familia, se encuentran en una situación de abandono, sin que haya un seguimiento médico. En ocasiones vemos noticias que indican que se ha encontrado un anciano en su vivienda y nadie se ha preocupado de él [4], o incluso situaciones en que los hijos han abandonado por falta de tiempo a sus padres y se han acabado enterando tiempo después de que ha muerto [5].

El motivo con el que se pretende diseñar y desarrollar una aplicación móvil es doble. En primer lugar, es impulsar un nuevo tipo de negocio basado en la venta de productos electrónicos que puedan ser consultados o manipulados mediante el uso de aplicaciones móviles. Este es el primero de los productos que se quiere realizar donde la idea es vender el producto electrónico a un precio, mientras que la aplicación móvil será completamente gratuita para el consumidor, aunque para utilizarla será necesario haber adquirido el producto electrónico.

En segundo lugar, se pretende mejorar el bienestar tanto de la gente de la 3ª edad como de los cuidadores o familiares más directos. En algún momento, todos tenemos algún familiar que envejece y que necesita de cuidados especiales o ser monitorizado. De hecho, el momento de escribir el presente documento, mis padres están rozando casi los 70 años y quiero que pasen una buena vejez y poder saber en todo momento que se encuentran bien y no depender de tener que llevarlos a una residencia de ancianos o de contratar a un cuidador que tenga que estar todo el día con ellos.



## 2. Definición del trabajo a realizar

Tal y como se ha visto en el anterior apartado el número de gente de la 3ª edad crece cada día que pasa, por lo que se necesita encontrar soluciones para que dicha gente mayor este lo mejor atendida posible. Actualmente la mayoría de aplicaciones que se centran en el bienestar de las personas se basan en productos nativos, que se conectan a un dispositivo *wearable* mediante *Bluetooth*. Esto conlleva limitaciones técnicas ya que los dispositivos que se conectan vía *Bluetooth* requieren estar muy cerca ambos dispositivos para que puedan intercambiar los datos.

Por ello en este TFM se va a realizar una aplicación de estas características donde se van a monitorizar una serie de constantes vitales de una persona de la 3ª edad que lleva una camiseta inteligente y se van a poder visualizar dichos datos desde una aplicación móvil, pero se va a utilizar un enfoque distinto. En primer lugar, se va a implementar como una *WebApp* lo que va a permitir que solamente realizando un único desarrollo la aplicación esté disponible para todas las plataformas disponibles. En segundo lugar, para solucionar el problema de la limitación de la cercanía de los dispositivos, los datos generados del producto se subirán a un servidor mediante un servicio *Restful* donde posteriormente podrán ser rescatados los datos desde la aplicación móvil. Esto permitirá a un usuario monitorizar las constantes vitales del usuario que lleva la camiseta inteligente, aunque la misma se encuentre a 100 Km de distancia.

## 3. Objetivos generales

Como se ha indicado en el apartado anterior la mayoría de las aplicaciones para dispositivos móviles que se han desarrollado se han diseñado de forma nativa e utilizan *Bluetooth* para vincularse con el proyecto.

La idea del presente TFM es enfocar el desarrollo de la aplicación bajo otro punto de vista. La idea principal es desarrollar una aplicación para dispositivos móviles que pueda ser ejecutado bajo distintas plataformas utilizando el mismo código, es decir, el desarrollo no va a ser nativo, ya que de lo contrario requeriría de un desarrollo por cada una de las plataformas. Por ello en primer lugar, la nueva aplicación móvil será una Web App que pueda ser fácilmente producida para distintas plataformas. Con ello se consigue que con un solo desarrollo funcione para distintas plataformas. En principio, se desarrollará y se testeará sólo para la plataforma *Android* ya que además de tener una cuenta de desarrollador, se posee gran variedad de teléfonos y tabletas donde probar la misma. No obstante, con el mismo código implementado se podrá producir una versión para la plataforma *IOS*.

La Web App será implementada bajo el framework *IONIC*. La versión de *IONIC* sobre la que se va a trabajar es la 4.0 que trabaja bajo Angular 5.0. Como entorno de ejecución se utilizará *NODE* y gestión de paquetes *NPM*. Además, en la medida de lo posible, se van a utilizar componentes que permitan gestionar de manera eficiente y que permita la reutilización de código, como ejemplo de ello se puede mencionar el uso de los preprocesadores *SCSS* o el uso de *minify* para las entregas finales. Además, para el control de versiones y el reporte de los bugs detectados se utilizará como repositorio de versiones *Git*.

El repositorio de *Github* donde se puede encontrar el código se puede encontrar en [25].

### **3.1 Objetivos principales**

El requisito fundamental de esta app será obtener los datos que se van obteniendo de la camiseta, pero la principal diferencia es que las mismas serán obtenidas a través de un servidor en vez de ser obtenidas a través del *Bluetooth*.

Los principales objetivos que debe cumplir la aplicación son las siguientes:

- Obtener y mostrar las constantes vitales que se la camiseta inteligente ha subido al servidor.
- Avisar mediante una notificación cuando se rebase un umbral definido.

Los usuarios esperan que la aplicación les permita:

- Monitorizar en todo momento las constantes vitales de la persona que lleva puesta la camiseta inteligente independientemente de donde se encuentre cada dispositivo.
- Ser notificado en caso de que el valor de una cierta constante vital supere un umbral que hayan definido previamente.

En cuanto a mis objetivos personales es el siguiente:

- Poner en marcha un nuevo modelo de negocio basado en *wearables* y aplicaciones móviles.

## 4. Metodología y proceso de trabajo

Para desarrollar la aplicación, se va a utilizar una aproximación PMBOK, enfocándonos en una serie de áreas que tal y como se indica en [16] y manteniendo un equilibrio entre todas ellas. Sobre todo, se va a enfocar en los aspectos de *calidad, tiempo y riesgos*, que son las áreas más importantes en el proyecto, pero sin dejar de lado el resto de los aspectos. Para conseguir dicho equilibrio se seguirá una filosofía basada en *SCRUM* [17] en la cual los *sprint* tendrán 2 semanas de duración. En cada uno de los *sprint* se abarcará el desarrollo una o 2 pantallas, según el coste de implementar dichas pantallas. Si durante un *sprint* se desarrolla 2 pantallas, aquellas tareas que requieran interactuar con los *early adopters*, se realizarán en el mismo periodo de tiempo. Estas tareas se indican con un asterisco. El desarrollo de cada una de las pantallas consistirá en los siguientes pasos:

- Entrevista con los *early adopters* para determinar cuáles son las necesidades para dicha pantalla. (\*)
- Realización de un primer boceto en *Balsamiq*.
- Los *early adopters* testean y validan el boceto en *Balsamiq*. (\*)
- Se implementa la pantalla junto con la funcionalidad definida.
- Se realiza un proceso de pruebas, corrigiendo los errores detectados. (\*)
- Documentación

Existirán un total de 6 *sprint* distintos y habrá una entrega de código y documentación cada 2 *sprint*. La única excepción será el 6º *sprint* que no durará exactamente 2 semanas, sino una semana y media para adecuarlo al calendario de la entrega final del proyecto. Además de los 6 *sprint*, habrá un *sprint 0* que se iniciará con anterioridad a los 6 *sprint* y que tan sólo durará una semana. En este *sprint* inicial se hará una colección inicial de todas las tareas y requisitos que debe cumplir el proyecto en general, que permita generar un *product backlog* inicial.

## 5. Planificación

Al inicio del proyecto, no se conocen todas las características que va a tener la aplicación móvil ya que se desconoce todas las constantes vitales que van a medir la camiseta, por lo que se dejará margen de maniobra para poder definir requisitos adicionales, eliminar los que ya no sean necesarios o correcciones del proyecto a medida que se vayan avanzando. Durante los primeros *sprint* se irán especificando e implementando las funcionalidades más críticas del programa (como por ejemplo la ventana en la que se visualizan las constantes vitales) y en los siguientes *sprint* se implementarán los requisitos menos importantes de la aplicación (notificaciones, dar de alta la camiseta, etc.). Se ha dejado hueco en los últimos *sprint* para cubrir posibles desviaciones en el proceso, cubrir nuevos requisitos a cubrir o corregir el *feedback* que provenga del cliente.

Cada 2 *sprint* habrá un hito que coincidirá con una entrega al cliente para que pueda evaluar una aplicación funcional y pueda ir reconduciendo el contenido de las siguientes entregas según su criterio. El 2º *sprint* después de cada hito (el *sprint* previo al siguiente hito), se definirá una tarea que consistirá en aplicar aquellas correcciones o modificaciones que vengan impulsadas por el cliente. El 3º hito (tras 6 *sprint*) se realizará la entrega final del proyecto y por tanto se procederá a pasar al mantenimiento del mismo.

En la figura 3, se expone una pequeña parte del *diagrama de Gantt* inicial con los *sprint* planificados. El mismo se puede encontrar completo para todo el ámbito del proyecto en el anexo A. Los *sprint* están representados en rojo y las tareas principales del proyecto en morado. Las subtareas que conforman las tareas principales se encuentran de color azul. Los hitos del proyecto corresponden con las entregas del proyecto (PEC 2, 3, 4 y 5) se encuentran de color amarillo. Por último, la defensa del TFM se encuentra de color verde, incluyéndose dentro de la planificación del proyecto.

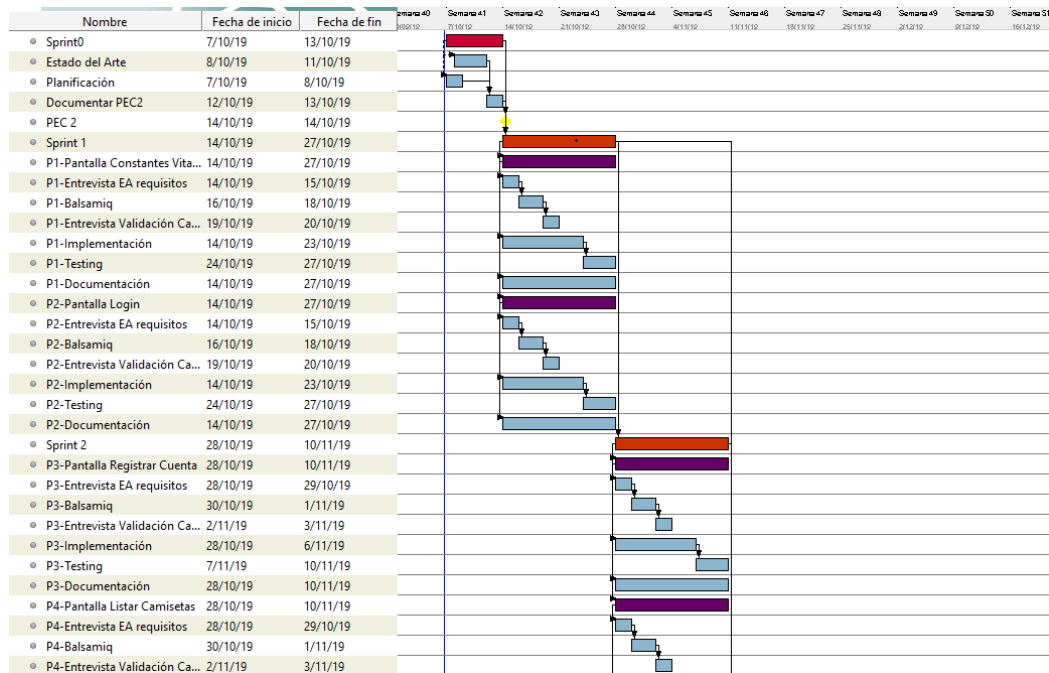


Figura 3 – Previsualización planificación del proyecto

## 6. Presupuesto

Las herramientas que se utilizarán durante el presente proyecto son las siguientes:

- Un PC con un SO Windows 10 o Linux.
- El SDK de IOS.
- Varios dispositivos móviles Android (1 teléfono y 1 tablet) para probar la aplicación en distintas pantallas.
- *IONIC* como framework para desarrollar la Web App.
- *NPM* como gestor de paquetes.
- Visual Studio Code como IDE de desarrollo.
- Hosting para poder subir el *Backend* de la aplicación móvil.
- *Github* como servidor remoto de repositorios y *Git* como herramienta de control de versiones.
- MySQL como servidor de Base de Datos.

Adicionalmente si se desea probar la aplicación en un Iphone se necesitarán las siguientes herramientas, aunque en este proyecto nos enfocaremos en Android.

- Un MAC mini o MacBook con XCode instalado.
- El SDK de IOS.
- Un Iphone o un Ipad.

En cuanto a la planificación por horas se va a distribuir en cada *sprint* de la siguiente forma:

Tarea	Horas por Sprint
Entrevistas Early Adopters	4
Diseño capturas <i>Balsamiq</i>	4
Implementación	16
Despliegue	1
Testing	4
Correcciones	3
Documentación	8

Tabla 1 - Distribución de horas por sprint

En total en cada *sprint* se invertirán cerca de 40 horas en la realización de todas estas tareas. Esta planificación de horas es para los *sprint* 1 al 5. El *sprint* 0 con una semana de duración se invertirán cerca de 20 horas para la planificación del resto del proyecto y el *sprint* 6 debido a la menor duración del mismo se invertirán cerca de 35 horas en la realización del mismo. Con lo que el tiempo invertido queda de la siguiente manera.

Sprint	Horas
0	20
1	40
2	40
3	40
4	40
5	40
6	35
<b>Total</b>	<b>255</b>

Tabla 2 - Distribución de horas a lo largo del proyecto

## 7. Productos finales

Los productos obtenidos al final de este trabajo serán:

- Aplicación Móvil en Android e IOS.
- Manual de Instalación.
- Manual de generación de versiones.
- Manual de Usuario.
- Memoria del Trabajo Final de Master.

## 8. Estructura del resto del documento

En los siguientes capítulos de la memoria tendremos:

- Análisis
- Desarrollo
- Instrucciones de uso
- Conclusiones

En el apartado del *Análisis* se hará un resumen de la situación actual sobre los *wereables* y las aplicaciones que han sido desarrolladas para trabajar con ellas. Esto ayudará a entender al lector cuales son las alternativas actuales y por qué se pretende abordar el problema de otra manera. Además, se analizará quienes van a ser nuestro público objetivo o el término que vamos a utilizar a lo largo de este proyecto *Early Adopter*, que son las personas a las que se han entrevistado, explicando sus inquietudes y preocupaciones. Estas personas irán transmitiendo bajo su punto de visto los aspectos que debería solucionar la aplicación e irán validando las diversas pantallas de las que se va a componer nuestra aplicación.

En el apartado de *Desarrollo* se explicará desde el proceso de indicar el porqué del uso de las tecnologías que se van a usar a lo largo del proyecto, así como la estructura que va a tener el proyecto en sí. Incorporando las pantallas que va a tener la aplicación en sí, la navegación entre dichas ventanas y como se ha enfocado la solución para resolver los problemas que los *Early Adopters* nos van comunicando, así como las pruebas que se van realizando.

En el apartado de *Instrucciones de uso* se indicará como se debe utilizar la aplicación, así como las pruebas que se han realizado para probar que se cumplen las especificaciones del programa.

En el apartado de *Conclusiones* se indicarán si los objetivos del proyecto se han cumplido y cuáles serán las posibles líneas de trabajo en un futuro.

## Capítulo 2: Análisis

### 1. Estado del arte

Con la llegada del nuevo milenio, surgió el boom y auge de los dispositivos móviles e inteligentes. Este boom se inició con la aparición del primer *Iphone* lanzado por la compañía Apple y la presentación del sistema operativo Android por parte de Google, ambos durante el año 2007. Actualmente los sistemas operativos de *Android* (Google) e *IOS* (Apple) se ejecutan en incontables dispositivos como son teléfonos móviles, tabletas, pulseras, relojes, automóviles y televisores. Por poner algún ejemplo de la variedad de dispositivos tenemos, por el lado de Android, televisores de marcas como *Philips* y *Sony* o *smartwatches* donde se pueden encontrar cientos de modelos (*Huawei*, *Xaomi*, etc). Por el lado de *IOS*, se tiene la propia televisión diseñada por *Apple* y como *smartwatch* tiene su propio modelo de reloj denominado *Apple Watch*.



Figura 4 - Algunos modelos de Smartwatch (Fuente: Computerhoy)

El boom no se limita únicamente simplemente a los dispositivos descritos anteriormente, sino a cualquier dispositivo que pueda estar conectado a una red. En la última década, el número de dispositivos interconectados entre sí, ha crecido exponencialmente. Este término se ha acuñado con el nombre de *IoT* (Internet of Things) y fue acuñado por Dave Evans perteneciente a Cisco. En [6] se indica en que el número de dispositivos interconectados superaba ya al de personas y que esta diferencia se iría incrementando a medida que fueran pasando los años. En el mencionado artículo, se indica que cada vez habrá más dispositivos con sensores que permitirán enviar transmitir dicha información para que pueda ser analizada o procesada por otros dispositivos.

Aquellos dispositivos que puedan ser sujetos a ser llevados o vestidos por una persona se les conoce como *wearables*. Este término fue acuñado [7] durante la década de los 90s, indicando que un *wearable* es una pieza de vestimenta que tiene capacidad de computación, es decir, es una pieza de ropa que actúa igual que un ordenador y que posee diferentes entradas para capturar eventos del entorno. Los *wearables* más conocidos son los *smartwatches*, aunque existen otros como camisetas, zapatillas, etc.

La mayoría de los *smartwatches* incorporan una serie de sensores que recogen información como el número de pasos, la frecuencia cardíaca o el número de horas de sueño que luego

envían a otro dispositivo mediante una tecnología inalámbrica. La gran mayoría de los *smartwatches* actuales funcionan vinculándose a un *Smartphone* u teléfono inteligente mediante *Bluetooth* (lo que crea una red de área personal, o denominada PAN, formada por el teléfono inteligente y el reloj), aunque en los *smartwatches* más recientes se permite realizar este vínculo gracias a una red *Wifi*. Este comportamiento se extrapola a la gran mayoría de los *wearables*. En el caso del presente proyecto, el *wearable* que se utiliza es una camiseta inteligente en vez de un reloj, aunque ambos tienen el mismo funcionamiento.

Por poner un ejemplo de lo indicado en el párrafo anterior, en el 2014 se presentó un estudio [8] que se pretendía construir una camiseta inteligente y una aplicación móvil, para el seguimiento de pacientes con reumatismo, que permite vincularse vía *Bluetooth* con la camiseta para así obtener y mostrar los datos en la pantalla del teléfono inteligente. En la figura 2 se muestra el esquema de conexión del mismo entre la camiseta y el teléfono inteligente.

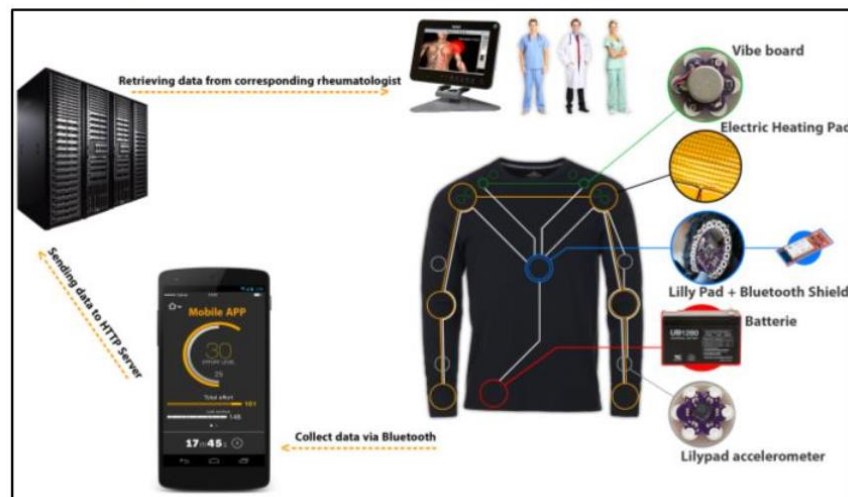


Figura 5 - Esquema de conexión de camiseta inteligente (Fuente: [8])

Como se observa en la figura 5, la camiseta envía, vía *Bluetooth*, los datos obtenidos de los sensores instalados, la aplicación móvil los muestra y actúa de pasarela para publicarlos en la nube, desde donde el médico en cuestión puede consultarlos. Esta arquitectura funciona bien cuando se tiene el teléfono y la camiseta cerca, pero en el momento que se aleje, los datos dejan de ser obtenidos. Adicionalmente existe otro problema; el teléfono inteligente del usuario es el encargado de almacenar y subir los datos al servidor lo que hace que el consumo de batería sea elevado, más teniendo en cuenta de que se tiene que tener el *Bluetooth* activo para poderse vincular con la camiseta.

## 2. Análisis del mercado

Si en vez del hardware, se centra el estudio sobre las aplicaciones para dispositivos móviles, basta con hacer una simple búsqueda en Google Play [9] para observar que existe gran variedad de aplicaciones que pueden ser utilizadas en los *wearable* o dependen de los datos del *wearable* para funcionar. En concreto, si se centra la investigación sobre aplicaciones relacionadas con la salud, se pueden encontrar cientos de las mismas. La gran mayoría de



las aplicaciones que se encuentran en el market que implican el uso de un *wearable* están relacionadas con apps la detección de constantes relacionadas con la actividad deportiva, como son el número de pasos, la frecuencia cardiaca o el número de calorías quemadas. Una investigación de cada una de estas aplicaciones daría para escribir varios libros, por lo que simplemente se limitará a explicar las aplicaciones más relevantes encontradas respecto al ámbito que abarca el presente TFM, que consiste el uso de un *wearable* para obtener constantes vitales de un paciente.

## 2.1 Salud Conectada

Esta aplicación [10], desarrollada por la compañía de seguros médicos Sanitas, permite conectarse a cualquier *wearable* vía *Bluetooth* para obtener las constantes vitales del paciente y las muestra en el teléfono móvil. Adicionalmente permite apuntar las citas, recordatorios y un listado de medicamentos a tomar. El problema que tiene principalmente es la gran dependencia que tiene del *Bluetooth* para funcionar, de hecho, la principal queja de los usuarios es que se desvincula fácilmente el *wearable* del dispositivo móvil. Además, para obtener valores, ambos dispositivos deben de estar cerca. Otro problema se ha detectado mientras se hizo una prueba de la aplicación para el presente estudio, debes de tener una cuenta en Sanitas para poderla utilizar y no se sugiere la creación de una desde la aplicación.

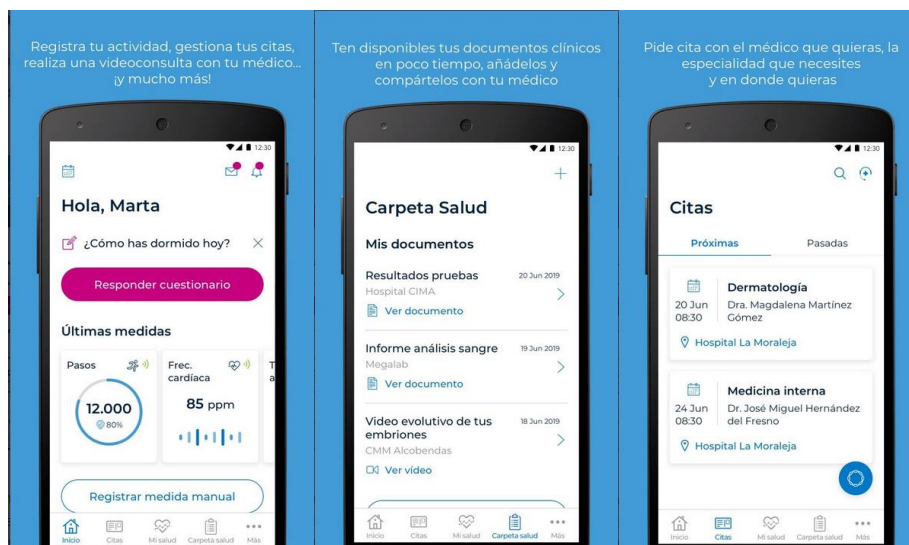


Figura 6 - Capturas aplicación Salud Conectada (Fuente: [10])

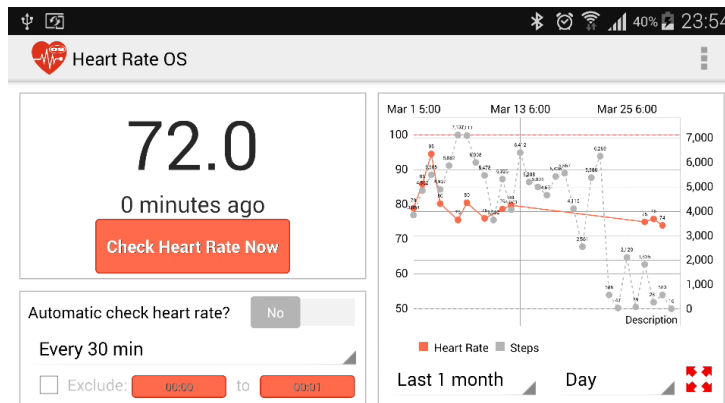


Figura 7 - Capturas aplicación Heart Rate OS (Fuente: [11])

## 2.2 Heart Rate OS – Android Watch

Esta aplicación [11], desarrollada por la *LFApp* y cuya aplicación puede visualizarse en la figura 7, permite medir la frecuencia cardiaca de la persona que lleva el reloj inteligente que esté vinculado al teléfono inteligente. La aplicación permite dos modos de funcionamiento. La primera de ellas es solicitando una medición manualmente, mientras que la segunda es programar la aplicación para que haga mediciones cada media hora. Cada medición dura 1 minuto. Luego, los datos obtenidos se sincronizan con los de *Google Fit*.

El principal inconveniente que tiene esta aplicación es que la monitorización de la frecuencia cardiaca no es permanente y se puede programar únicamente cada 30 minutos. Además, las opciones de sincronización solo permiten sincronizar con *Google Fit*.

## 2.3 Health Wearable

Esta aplicación [12], desarrollada por *Analog Devices Inc* es una aproximación parecida a lo que se pretende abordar en el presente proyecto. Esta aplicación permite conectarse a un *smartwatch* específicamente creado [13] para suministrar una serie de constantes vitales a la aplicación móvil. Esta aplicación permite especificar entre que constantes se quieren realizar mediciones (permite frecuencia cardiaca, conductividad de la piel, temperatura, etc.).

La aplicación permite visualizar en tiempo real y mediante una serie de gráficas, las mediciones obtenidas a lo largo del tiempo y visualizar un histórico del mismo. La única diferencia respecto a las anteriores es que no existe opción para compartir y publicar los datos en la nube.

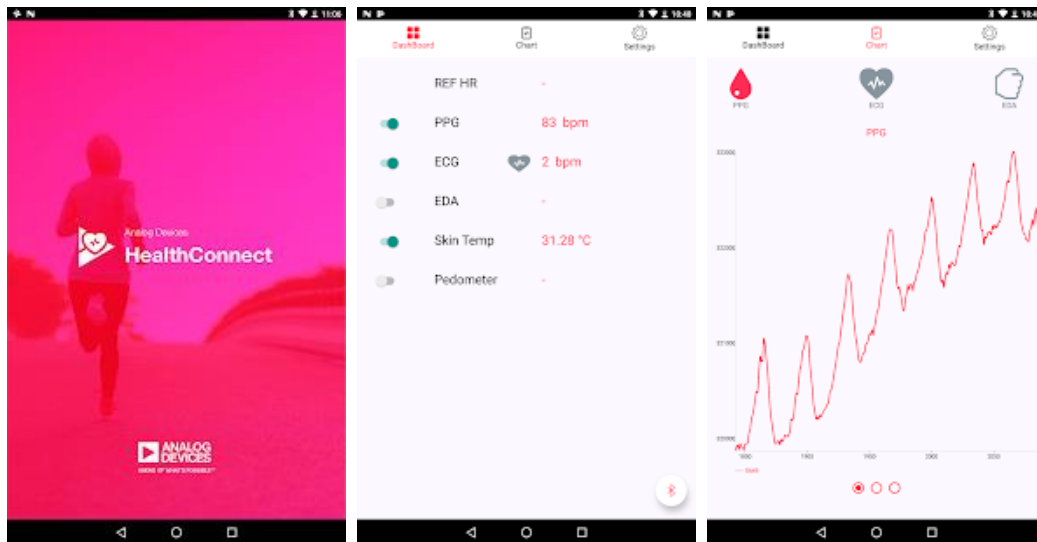


Figura 8 - Capturas aplicación Health Wearable. Fuente ([12])

## 2.4 Resumen aplicaciones vistas:

Las aplicaciones que se han visto presentan 3 características principales:

- Son apps nativas.
- Requieren de *Bluetooth* para vincularse con el *wearable*.
- En caso de detectar alguna anomalía en los datos, no se notifica al usuario.

Esto conlleva a una serie de características que tienen estas aplicaciones con el *wearable* asociado.

- Requieren estar cerca el dispositivo móvil del *wearable*.
- Se debe hacer un desarrollo por cada plataforma que se desee exportar la aplicación. Esto quiere decir que en caso de querer exportarlo a Android e IOS hay que realizar 2 desarrollos distintos.
- Si la app tiene opción de subir datos a la nube, utiliza la conexión a la red del dispositivo móvil, lo que utiliza los datos móviles del usuario e utiliza la batería del dispositivo.
- No existen mecanismos para avisar en caso de detectar un problema grave en las constantes vitales.
- No existen mecanismos que notifique al usuario que el *wearable* está a punto de quedarse sin batería.

Como se verá dentro del apartado de objetivos y el alcance, lo que se pretende con la nueva aplicación móvil es eliminar estas limitaciones que tienen las actuales aplicaciones.

Por último y para acabar el apartado, se abandona el terreno de las aplicaciones móviles para centrarnos en aplicaciones sanitarias relacionadas con el cuidado de pacientes mediante el uso de *wearables*.

En el año 2010, la universidad Rey Juan Carlos impulsó un proyecto para controlar las constantes vitales y tener localizados en todo momento a todos los pacientes del hospital. La idea principal en la que se basa el proyecto LOBIN [14], es tener una camiseta con un conjunto de sensores y que mediante Wifi comunique esta información al sistema central del

Hospital La Paz, situado en Madrid. Principalmente esta camiseta está dirigida a aquellos pacientes con problemas de corazón, aunque la misma es capaz de medir otros factores como la temperatura del paciente o detección y alarma en caso de detectar que el paciente abandona la zona de la que le fue asignada.

El proyecto LOBIN se centra principalmente en la arquitectura del sistema (desde la implementación de la camiseta a la comunicación con el sistema central) y no en la aplicación en sí, pero si hay indicaciones de que como se tenía que desarrollar la aplicación y de lo que tenía que contener.



Figura 9 - Interfaz aplicación proyecto LOBIN. Fuente ([13])

La aplicación consiste en una aplicación Java con varias ventanas, las cuales cada una tiene una función muy específica. En la ventana principal se ve el árbol de pacientes y un mapa del hospital donde se localizan los pacientes. Cuando se pulsa sobre el paciente, se abre una nueva ventana con las constantes del paciente.

### 3. Público Objetivo

El concepto de *Early Adopter* fue definido por Rogers Everest en 1962 [20]. En el citado artículo se describe dicho concepto como aquellas personas que, sin llegar a ser innovadoras, adquieren un producto porque les soluciona un problema que tienen. El *early adopter* es el grupo de gente que adquiere un producto antes de que se haga popular pero que se adelantó a la época en la cual el producto se hizo famoso.

Por poner un ejemplo, cuando surgió la primera generación de *smartwatches* en el año 2012, la gente que adquirió los relojes fueron los primeros *early adopters*. Los mismos buscaban un dispositivo que pudieran llevar fácilmente en el cuerpo y pudiera medir el número de pasos, la frecuencia cardiaca y la velocidad, es decir, que los mismos era gente dedicada al deporte. Gracias a esto, surgieron las *pulseras inteligentes* que fueron una evolución de los *smartwatches* orientados a gente que hacer deporte. Así que en este caso los *early adopters* fueron los deportistas que vieron la necesidad de cuantificar su rendimiento.

En el caso del presente proyecto, la camiseta inteligente va a servir para monitorizar el bienestar de la gente de la 3ª edad. Aquellas personas que están interesadas en controlar dicho bienestar son principalmente sus familiares y los cuidadores que cuidan de estos. Por ello nuestro *Early Adopter* van a ser dos tipos de personas.

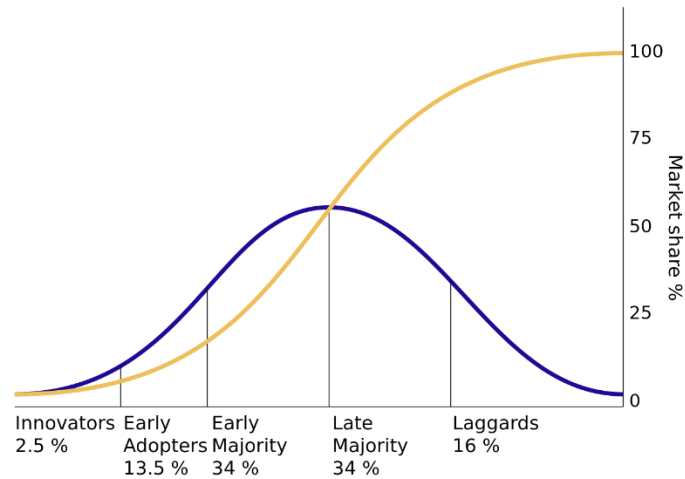


Figura 3 - Curva de mercado por sector (Fuente[20])

El primero de este tipo de personas se basa principalmente en aquellas personas que tienen padres o abuelos ya en la 3ª edad y estos viven solos o no pueden ser vigilados en todo momento ya sea por falta de dinero o de tiempo.

El segundo tipo de personas, son aquellas personas que se dedican a cuidar de las personas mayores en sus hogares. Estas personas suelen cuidar de 2 o 3 personas al mismo tiempo. Puede darse la situación de que en el momento de que se esté cuidando de una de estas personas otra le ocurra una emergencia que requiera de la asistencia del cuidador o incluso su posible traslado a un centro médico para su tratamiento.

Durante el desarrollo del proyecto se va a entrevistar a 4 personas, 2 *Early Adopters* de cada tipo. Aunque las motivaciones por las que cuidan los ancianos son distintas (uno es el bienestar de los ancianos y la otra es la profesión con la que genera ingresos) ambas pretenden lograr el mismo fin; tener monitorizado el bienestar de nuestros mayores y que en caso de necesitar ayuda se le pueda suministrar cuanto antes.

En el anexo B se suministra el perfil de las 4 personas a las que se va a entrevistar durante el desarrollo de todo el proyecto, explicando para cada una de ellas las preocupaciones que tiene en relación a la problemática que se pretende solucionar, así lo que espera que le solucione la aplicación.

Para proteger la privacidad de los *Early Adopters*, los datos personales de estas personas van a ser sustituidos por datos ficticios para mantener el anonimato de estas personas. Solamente se van a mantener los datos relacionados con la problemática a solucionar. Para que los *Early Adopters* tengan la tranquilidad de que solamente sus datos van a ser utilizados para la utilización de este proyecto se les hace firmar durante la primera entrevista un contrato de confidencialidad que indica los únicos datos que van a ser recolectados son los relacionados con el tipo de trabajo que realiza, las preocupaciones que tiene entorno al problema a solucionar y sus opiniones de cómo lo va a solucionar.

Los *Early Adopters* van a ser entrevistados 2 veces cada 2 semanas, al inicio de cada *sprint* y luego se les pedirá que testeen el producto al final del *sprint* y que den una valoración final. Con dichas valoraciones se procederá a corregir los errores encontrados en los colchones de planificación que se han dejado.

En la primera de las entrevistas se va a entrevistar a la persona para abarcar la solución de una ventana o funcionalidad de la aplicación, indicando que datos deberían mostrarse, como le gustaría que se mostrasen y que comportamiento se espera que tengan los mismos. En cada una de estas entrevistas se va a abarcar 2 *features* o funcionalidades al mismo tiempo dejando cualquier otra funcionalidad para el *sprint* que le corresponda. De hecho, si el *Early Adopter* se desvía y comienza a contarnos preocupaciones de una funcionalidad que corresponda a otro *sprint* se intentará reconducir la conversación a la funcionalidad actual. La duración de estas entrevistas rondará entre 10 y 20 minutos. Esta entrevista siempre será la primera tarea del *sprint*.

En la segunda de las entrevistas se mostrará al *Early Adopter* unos prototipos o *mockups* de la aplicación realizados en *Balsamiq* y evaluarán bajo su punto de vista si el prototipo cumple sus expectativas y si puede solucionar la problemática relacionada o si requiere cambios. Si el prototipo es validado o requiere de cambios leves se llevará ya a código. En caso contrario, se planificará una nueva reunión con los *Early Adopters* para readaptar los prototipos a los cambios que sugieran o se vea que necesiten.

## 4. Especificaciones del producto

Las funcionalidades que debe realizar la aplicación móvil son las siguientes:

- Debe permitir al usuario autenticarse en el sistema. Una vez autenticado en el sistema, los datos del resto de apartados deben mostrarse únicamente información de dicho usuario.
- Debe permitir al usuario registrar una cuenta.
- Debe listar el conjunto de camisetas que tiene el usuario registradas en la aplicación.
- Debe permitir añadir, modificar y eliminar camisetas.
- Debe mostrar el último valor obtenido por la camiseta para las siguientes constantes vitales.
  - Frecuencia cardiaca.
  - Sudoración.
  - Temperatura.
- Debe mostrar los datos de las constantes vitales en un periodo de tiempo mediante gráficas de las siguientes constantes vitales.
  - Frecuencia cardiaca.
  - Sudoración.
  - Temperatura.
- Debe permitir definir un umbral de notificación para una o más de las constantes vitales o un tiempo en el que no se reciban datos.
- Debe notificar al usuario si alguno de los umbrales definidos se supera.
- Debe notificar al usuario si la persona que lleva la camiseta ha sufrido una caída.
- Debe notificar al usuario si la batería de la camiseta es baja.

## Capítulo 3: Desarrollo

Antes de abordar una a una cada una de las partes de la aplicación se va a dar una pincelada por encima al conjunto completo de lo que va a ser la aplicación móvil. La aplicación móvil se divide en 2 partes, la parte que no requiere autenticación y la parte autenticada en la misma.

En la figura 11 se muestra el diagrama de navegación del mismo, donde las flechas indican el sentido de la navegación. En algunos casos el sentido es doble, donde desde una ventana se puede ir a otra y desde la misma se puede volver a la anterior ya sea mediante otro botón o pulsando el botón atrás del terminal móvil.

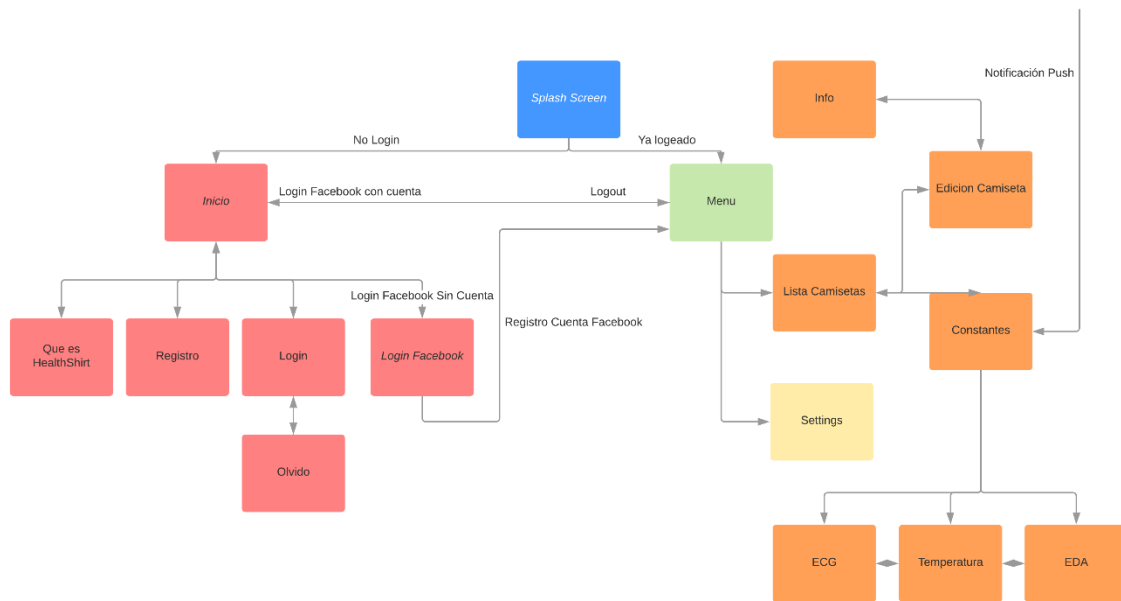


Figura 4 - Diagrama de Navegación

El primero de los grandes bloques está representado por los bloques de color rojo. Estos bloques son el sistema de autenticación a la aplicación. Dado el carácter de los datos que se van a tratar el sistema y de acuerdo al *RGPD* del 2016, tienen que estar autenticado de manera de que los usuarios únicamente vayan a ver los datos que realmente les pertenezcan y no los de otros usuarios. Para que los usuarios puedan utilizar la aplicación deberán confirmar la cuenta aceptando los términos de privacidad que vienen en las mismas. Los mismos al momento de escribir estas líneas no se encuentran escritas y se irán incorporando antes de que salga definitivamente la aplicación en su versión 1.0. Durante los *sprints* 1 y 2 se definirán las necesidades funcionales que deben tener estas pantallas de autenticación, así como las pantallas que definen los mismos.

El segundo de los bloques está representado por los bloques de color naranja, que representan ya a las partes privadas de la aplicación que únicamente aquellos usuarios que se hayan registrado y aceptado los términos de privacidad pueden acceder. Estas pantallas estarán gobernadas por un menú que se puede mostrar u ocultar deslizando hacia la derecha la pantalla, aunque el número de opciones del menú estará limitado únicamente a dos, el listado de las camisetas que tenemos registradas en nuestra cuenta y las opciones de configuración.

Este segundo bloque se divide a su vez en la parte de gestión de las camisetas y en la visualización de las constantes vitales. En la pantalla de constantes vitales se podrán visualizar las constantes vitales que se obtengan a partir de esa camiseta. Por último, se puede observar como si se recibe una notificación push en el dispositivo se procederá a avisar al usuario y cuando se accede a dicha notificación se abrirá automáticamente la ventana de constantes asociadas a la camiseta a la que llegó la notificación.

Tal y como se ha comentado en los apartados anteriores, esta aplicación se va a realizar utilizando el Framework *IONIC* que está basado en *Cordova*, esto nos permitirá que utilizando el mismo código se pueda generar una versión para Android y una versión para IOS, aunque si se quisiera también se podría generar también una versión para web. La versión de *IONIC* que se va a utilizar es la versión 4 con una versión de *Cordova* 8.1.2. Dentro de esta versión de *IONIC* se va a trabajar bajo Angular 5 donde se trabajará utilizando el patrón *MVC*. Cada una de las pantallas de la aplicación se conformará por un fichero con extensión *html* (página web) que será la vista, un fichero con extensión *ts* (typescript) que actuará de controlador y un fichero con extensión *SCSS* (que luego se compilará a *CSS*) que permitirá definir los estilos de la pantalla. Para evitar la duplicidad de código en las clases *SCSS* lo que se hará es que el fichero con extensión *SCSS* para cada una de las vistas importará las reglas más generales que se definirán en el fichero *app.SCSS* que pertenece a la página con la que se arranca la aplicación. Todas las reglas de estilos que vayan a estar definidas en varias de las páginas se incluirán dentro de este fichero que posteriormente se importará para formar el fichero *CSS* resultante. Solamente aquellos estilos que pertenezcan exclusivamente a lo que forma parte de los estilos de la propia página (por ejemplo, un estilo que afecte a un único campo de esa página y que no se vuelva a utilizar en el resto de páginas), se escribirá dentro del propio fichero *SCSS* perteneciente a dicha vista. Esto nos permitirá en la medida de lo posible reutilizar sin necesidad de escribirlo de nuevo las reglas de estilos necesarias y si en un futuro se necesita reescribir dicha regla, el cambio afectará directamente a todas las páginas que necesiten utilizar de dichos estilos.

Para implementar algunas de las funcionalidades más especiales y que requieran ya de controlar el comportamiento del dispositivo donde se esté ejecutando la aplicación, se procederá a la descarga de algunos plugins que están disponibles desde el framework de *IONIC*. En caso de que alguno externo sea necesario será indicado en el apartado y añadido a la bibliografía. El plugin permite simplificar el tiempo de desarrollo abstrayendo al desarrollador de tener que invertir tiempo de desarrollo y pruebas, invirtiendo realmente en programar y probar la funcionalidad que tiene prevista probar y no los aspectos secundarios de la misma. Con ello el plugin se encargará de implementar la parte de lógica dependiente de la plataforma, sin ser necesario para nosotros el implementar el código para dicha plataforma abstrayéndonos completamente y simplemente codificando la funcionalidad que se desee implementar.

Todo esto permitirá tener una aplicación web funcional que pueda generarse tanto para la plataforma Android como para la plataforma IOS.



## 3.1 Sprint 1

### ***Pantalla de Constantes Vitales***

La primera de las funcionalidades que se va a tratar es la visualización de las constantes vitales. Esta pantalla es una de las más importantes de la aplicación ya que es el motivo principal por el que el cliente va a estar interesado en nuestro producto. En la pantalla de las constantes vitales se van a poder visualizar un conjunto de constantes que son obtenidas a partir de la camiseta inteligente y que son enviadas a un servicio en la nube y luego son recuperadas por nuestra aplicación a partir de un servicio *Restful* que se explicará en el apartado de *Backend*.

En dicha pantalla se mostrarán una serie de gráficas, cada una asociada a una constante vital. En las entrevistas, se definieron tanto los requisitos funcionales que debe cumplir esta pantalla (los cuales se definen en la primera reunión de cada uno de los *sprint*), así como deben mostrarse los distintos elementos que forman la pantalla.

### ***Entrevistas***

En las entrevistas realizadas, se introdujeron 4 principales constantes que se desean conocer en tiempo real y que permiten determinar si la persona mayor está bien o no. Las 4 constantes que interesan ser visualizadas desde este menú son las siguientes:

- **ECG:** ECG son las siglas de Electrocardiograma. Esta constante vital se refiere a los procesos de sístole y diástole que realiza el corazón durante una pulsación cardiaca. El mismo permite detectar muchas patologías cardiacas entre las que se pueden encontrar arritmias, infartos, cardiopatías, entre otras. La preocupación principal de los *Early Adopters* respecto a esta constante vital, es que a la gente mayor de la que se preocupa y desea monitorizar, tenga principalmente un infarto o parada cardiaca, ya que en caso de detectarse a tiempo es posible que se le pueda salvar la vida.
- **Temperatura:** Esta constante vital hace referencia a la temperatura corporal que tiene el paciente. Los *Early Adopters* han indicado que en muchas más ocasiones de las que desean les entra un poco de fiebre, pero no lo detectan hasta que acuden a visitarlos. Puede ocurrir el caso contrario, donde la persona coge frio y baja su temperatura corporal, como por ejemplo puede pasar al vivir en un lugar muy frio y la persona mayor no se encuentra completamente abrigada, lo que hace que se reduzca su temperatura corporal. La idea es detectar cuando la persona mayor está teniendo frio o calor y se pueda ponerle remedio, aunque también puede servir como seguimiento de un síntoma de fiebre y ver si hay mejoría de la misma o no.
- **EDA:** Esta constante vital hace referencia a la conductancia eléctrica de la piel. Su uso principalmente es en el detector de mentiras donde si la persona miente la misma empieza a temblar y a sudar, cosa que es detectada por la máquina. También es utilizado en el ámbito deportivo donde permite detectar como de bien funcionan las reacciones del atleta ante situaciones estresantes, un deportista con unos valores muy

bajos de EDA o una oscilación muy baja de la misma implicará que tiene pocos reflejos o reacción ante los imprevistos lo cual no será bueno para el deportista.

El mismo principio se puede aplicar para determinar si una persona está enferma o no. Cuando una persona esta intranquila o le entra un sofoco de calor los valores de esta constante oscilan rápidamente, en cuyo caso se puede detectar que la persona no se encuentra bien. Los *Early Adopters* ven interesantes esta constante ya que permite detectar si la persona que lleva la camiseta está tranquila o está empezando estar intranquila o si sufre algún golpe de calor.

- **FBS:** Son las siglas del nivel de glucosa en sangre. El mismo detecta si los niveles de glucosa en sangre son los correctos o no. Se usa principalmente para detectar que aquellas personas diabéticas tienen un nivel de azúcar en sangre correcto por si hay que pincharles insulina o no. La preocupación de 2 de nuestros *Early Adopters* fue que las personas que estaban a su cargo eran diabéticas por lo que se tenía que vigilar su nivel de azúcar en sangre cada poca hora por si había que aplicar insulina al paciente o no.

En la pantalla se van a implementar 3 de estas constantes el *ECG*, la temperatura y el *EDA*. Los motivos de descartar inicialmente el *FBS* son los siguientes.

- Los sensores *FBS* son más intrusivos que los otros 3 (requieren pinchar al paciente cada vez que se pone la camiseta).
- No todas las personas necesitan que se les monitorice el nivel de glucosa en sangre.
- Algunos de los *Early Adopters* no lo indicaron como constante a monitorizar.
- Los sensores *FBS* no están popularizados y todavía son proyectos que tienen pocos años de investigación [24].

En las entrevistas se definió la necesidad de sacar la información de las constantes vitales en 2 ventanas de tiempo. La primera de ellas es, mostrar la información de las constantes en tiempo real, que es la opción que se mostrará por defecto cuando se entre a visualizar los datos de dicha camiseta. La otra de ellas es mostrar la información obtenida durante un periodo de tiempo en el pasado, pero se debe especificar exactamente de qué hora y minuto hasta que hora y minuto se desea visualizar los datos.

Dado que la información del *ECG* y el *EDA* cambia constantemente en pocos milisegundos, produciéndose ciclos en el caso de *ECG*, las constantes vitales de *ECG* y *EDA* no tienen que mostrar el periodo total del tiempo analizado, sino que deben empezar a reproducir la secuencia desde el inicio hasta el fin. En el caso de la temperatura se mostrará 1 punto por cada minuto de temperatura que se tenga en la muestra (si es una hora se tendrá 60 puntos).

Esto quiere decir que se si se especifica los datos de 2 minutos en el pasado, tanto el *ECG* como el *EDA* empezarán a reproducir como si se tratara de una animación dichas constantes vitales en dichos 2 minutos. En el caso del *ECG* se mostrará cómo se reproduce el latido cardiaco y en el caso del *EDA* se visualizará la línea que muestra el nivel de la resistencia eléctrica de la piel, donde de vez en cuando y si el paciente está en una situación tensa se producirá oscilaciones. En el caso de la temperatura se mostrarán únicamente 2 puntos en la gráfica.

En el caso de visualizar los datos en tiempo real, la información mostrada será la información del ECG y EDA en el último minuto y la información de los últimos 10 minutos respecto a la temperatura. Antes de que se termine este minuto, el siguiente minuto de datos estará disponible en el servidor, obteniéndose y mostrándose, si no la hubiera se detendría la animación y no se mostraría ningún dato.

En caso de que se acceda a la pantalla y no haya datos disponibles se informará si ha habido un error en caso de acceder a la información, pero si no ha habido un error y resulta que no existe ninguna información simplemente no se mostrará ninguna información. Esto se hace por que es posible que al intentar acceder a los datos actuales no existan datos, porque la camiseta no haya subido datos recientemente, lo que puede implicar que la camiseta no este encendida (por falta de batería o porque este apagada) o por que no tenga conectividad la red Wifi a la que está conectada la misma. En este caso no se considera que sea un error, ya que no ha habido un error al buscar los datos ya que los mismos no existían. El usuario que vea que no hay datos actuales, conocerá entonces que existe un posible problema con la camiseta y se pondrá a solucionarlo.

En caso de que el error sea por un fallo en la comunicación entre el móvil y el servidor web, entonces si deberá indicarse al usuario del error del mismo, para que sepa que el error ha sido producido por el proceso de recuperación de datos y no por que no existan datos en sí mismo en la Base de Datos.

Por último, para cada una de las constantes vitales se debe mostrar los siguientes valores:

- ECG: Número de pulsaciones medias durante el periodo de tiempo analizado (o último minuto si es en tiempo real). Esto se explicará brevemente más adelante como se calculará.
- EDA: Media de la resistencia eléctrica de la piel durante el periodo de tiempo analizado (o último minuto si es en tiempo real).
- Temperatura: Media de la temperatura durante el periodo de tiempo analizado (o último minuto si es en tiempo real).

Los valores medios de la EDA y temperatura pueden ser fácilmente calculados, simplemente se sumarán el valor todos los datos obtenidos para dicha constante vital en ese periodo de tiempo y se dividirá por el número total de datos obtenidos para dicha constante vital en ese periodo de tiempo.

Esto mismo no se puede aplicar en el ECG. La lectura de los valores de un ECG no mide una pulsación o una media de pulsaciones directamente. Los sensores ECG miden la variación eléctrica del latido del corazón a lo largo del tiempo. En un sensor ECG conectado a una potencia de 3.3V los valores suelen oscilar entre 0 y 600, siendo un valor medio unos 200. El proceso del latido del corazón se puede resumir en la Figura 12



Figura 5 - Ejemplo Latido Corazón (Fuente [26])

Como se puede observar en la imagen, el proceso de un latido de corazón se divide en 2 partes. Uno de ellos se le conoce como *diástole* y es el periodo en el cual el corazón se encuentra relajado y en el cual el mismo se llena de sangre. Esto corresponde con las zonas planas de la imagen y con los picos pequeños que se producen antes del pico más grande. En el momento que se produce este pico, al proceso se le conoce como *sístole*, en el cual el corazón bombea el corazón hacia el resto del cuerpo, el mismo proceso termina tras el segundo pico pequeño que se detecta a la derecha del grande y antes de que vuelva a la zona plana.

Cuando empieza la *sístole*, el sensor ECG mide un valor aproximado entre 500 y 600 y luego posteriormente se reduce hasta un valor cercano a 0. El resto de picos no suelen sobrepasar los 300 a 350, siendo el valor normal plano valores entre 200 y 250. Por tanto, se puede calcular el NPM contando el número de picos superior a 500 (se puede hacer análogamente con picos inferior a 50) durante ese periodo de tiempo y dividirlo por el número total de minutos. Por ejemplo, si en 2 minutos se ha detectado 138 picos, entonces el NPM que calculará el programa es de 69.

## Prototipos



Figura 6 - Prototipos Constantes Vitales

## Pantallas finales realizadas

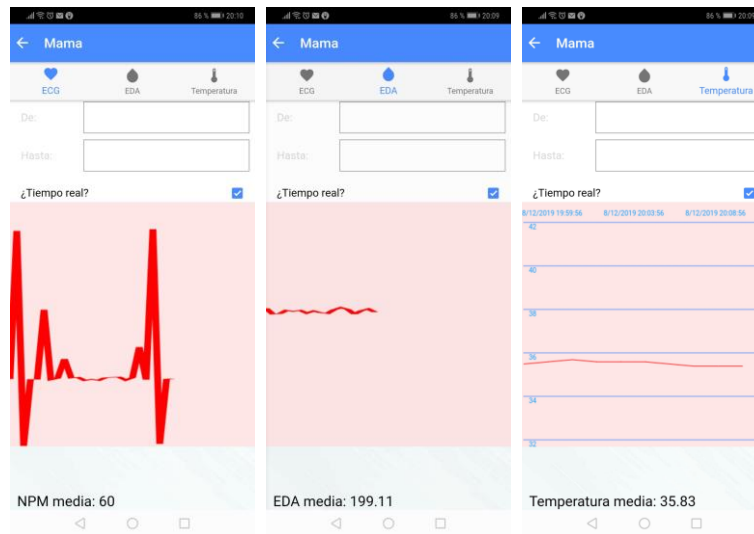


Figura 7 - Versión final Constantes Vitales

## Notaciones sobre la implementación

Esta pantalla simplemente es una pantalla que contiene a su vez otras 3 ventanas. Aunque todas ellas siguen el patrón *MVC* (Modelo vista controlador) donde se tiene un fichero en extensión *html* que representa a la vista y un fichero en extensión *ts* que representa al controlador, el controlador principal es el asociado a la ventana de constantes.

Este controlador es el encargado de inicializar las estructuras y los demás controladores que se conforman en esta pantalla y se encarga de centralizar cualquier petición que provenga de los controladores asociados específicamente a una constante vital antes de redirigirla a quien debe. Es decir, aunque se pulse el campo de Fecha De desde la vista de la temperatura, el controlador de la temperatura no se encarga de realizar la petición, sino que delegará la petición en el controlador de constantes que será la encargada de luego enviar la petición a quien deba tratarla (en este caso, hay un *provider* o proveedores típicos de *Angular*) quien se encargará de realizar todas las operaciones que requieran conexión con el servicio *Restful*, del que se hablará más adelante.

Los controladores de cada una de las constantes se encargarán simplemente en pintar la gráfica o animación asociada a la constante una vez tengan valores disponibles para pintar. Para conocer si hay un valor o no para pintar, existirá un flag que los controladores individuales podrán consultar para ver si se dispone de nuevos datos, en caso afirmativo obtendrán los nuevos datos ya almacenados localmente y procederán a reiniciar la gráfica o la animación. Este mecanismo es utilizado cuando se estén consultando los datos en tiempo real, ya que conocemos con que periodicidad se van a necesitar datos nuevos.

Además, el controlador padre, podrá avisar a los hijos indicándoles que deben refrescar los datos, utilizándose esta funcionalidad para evitar una consulta continua de los datos, que puede darse sobre todo cuando se realiza una búsqueda de datos históricos, ya que desconocemos cada cuanto van a ser dichas consultas.

En cuanto a la visualización de las gráficas, se ha utilizado el componente *Canvas* que viene definido en HTML5. Para cada uno de los canvas se ha definido una serie de transformaciones para visualizar los datos a una escala visible dentro de la pantalla del móvil o desplazar los datos a medida que se van pintando. Cada una de las vistas relacionadas con las constantes vitales tiene su propio canvas. Las constantes ECG y EDA están conformadas por una animación que va transcurriendo a lo largo del tiempo. Para ello lo primero que se hace es calcular el intervalo en milisegundos que debe transcurrir entre actualizaciones, lo que se puede traducir en cuantos frames van a haber por segundo.

```
this.fpsIntervalo = 1000 / this.fps;
```

Donde fps es el número de frames que se desea que haya como máximo por cada segundo. En el caso del ECG y del EDA el valor es de 20 frames por segundo, ya que es el número de datos que los sensores están capturando desde la camiseta y transmiten posteriormente al servidor en lotes de 1200 datos por minuto. Además, es importante cambiar la orientación del canvas, ya que por defecto el punto de origen (0,0) se encuentra situado en la esquina superior izquierda por lo que si la pintamos saldría al revés. Esto requiere dos transformaciones

```
this.ctx.translate(0, this.canvas.height);  
this.ctx.scale(10,-0.5);
```

En primer lugar se debe trasladar el punto de origen a la esquina inferior izquierda del canvas, esto se consigue realizando una traslación al punto 0 de las X y al punto *canvas.height* que corresponde con la altura del canvas, por lo que acabará en la esquina inferior izquierda. Además, se necesita un movimiento de reescalado. Si no se reescala, podrían caber 600 puntos dentro del canvas, pero los mismos se verían muy pequeños y apenas se podría apreciar los datos del ECG o del EDA. Con la altura ocurre el efecto contrario. Los datos del ECG y del EDA van comprendidos entre 0 y 600 que son los valores que obtienen los sensores que hay conectados en la camiseta con un voltaje de 3.3V, pero la altura del canvas es de 300. Por tanto, para reescalar el canvas a una medida que sea fácilmente visualizable se debe cambiar a la mitad la escala en el eje de las Y (dividiéndolo entre 0.5) y el eje de las X debe dividirse entre 10 para aumentar en 10 veces la escala del mismo.

Estas dos transformaciones permitirán visualizar la animación a una escala natural a como si el usuario estuviera visualizando directamente la animación.

Si ha transcurrido el tiempo de pintar un nuevo frame, lo primero que se hace es limpiar todo el canvas utilizando *clearRect*. A continuación, si no llevamos más de 25 frames, simplemente pintamos la lista de valores que llevábamos hasta ahora. Esto lo hace la parte *if* del código que se muestra a continuación. Si llevamos más de 25 frames de animación pintamos siempre los últimos 25 frames que se llevan de la animación. Esto lo hace la parte *else* del código que se muestra a continuación.

```
if (this.x++ < this.panAtX) {  
  for (var xx = 0; xx <= this.x; xx++) {  
    if(xx==0){  
      this.ctx.moveTo(xx,this.data[xx]);
```

```

    }
    else{
        this.ctx.lineTo(xx,this.data[xx]);
    }
}
this.ctx.stroke();

} else {
    for (var xx2 = 0; xx2 < this.panAtX; xx2++) {
        var y = this.data[this.x - this.panAtX + xx2];
        if(xx2==0){
            this.ctx.moveTo(xx2,y);
        }
        else{
            this.ctx.lineTo(xx2,y);
        }
    }
    this.ctx.stroke();
}

```

El motivo de esto es que, si no, no cabría toda la animación en el canvas, viéndose solamente los primeros segundos de la misma e ocultándose el resto de la misma. Finalmente dibujar toda la línea que se ha ido trazando a lo largo de la animación se utilizará el método *stroke* para indicar que pinte el trazo.

### **Notaciones sobre el proceso de pruebas o test**

La pantalla o funcionalidad de las constantes vitales es la parte fundamental de nuestra aplicación. Directamente, si no hubiera dado tiempo a implementar esta parte de la aplicación nadie utilizaría la aplicación ya que se basaría en una simple app de listado, donde conectaríamos una cuenta y añadiríamos un conjunto de elementos a una lista. Por ello es la parte de la aplicación a la que más énfasis y tiempo se le ha dedicado a lo largo del proyecto, no dando tiempo a ser completado durante el primer sprint. Por ello dentro del apartado del sprint 4, se retomará esta funcionalidad, pero para explicar la otra parte que influye en la misma que es de la generación automática de datos para poderla probar ya que la camiseta en el momento de escribir las presentes líneas no se encuentra disponible.

### **Pantalla de Login**

En la pantalla de Login o autenticación se busca que el cliente pueda autenticarse en el sistema antes de acceder a los datos de las camisetas que tiene dado de alta en el sistema. Para ello la idea inicial era de una simple pantalla de login que permitiera al usuario autenticarse en la aplicación. No obstante, como se va a comentar en los próximos párrafos, esta pantalla aumentó muchísimo su complejidad a medida que se iba viendo con los *Early Adopters* durante las entrevistas.

## Entrevistas

Lo primero que mencionaron los *Early Adopters* al visualizar los primeros prototipos realizados en *Balsamiq* es que veían muy abrupto descargar una aplicación y presentarles una pantalla de *login* sin más explicación de lo que era la aplicación en sí, sobre todo si los usuarios todavía no habían adquirido previamente la camiseta asociada a la aplicación. Además, indicaron que con la información que se daba en la pantalla de *login*, no les quedaba claro si tienen que adquirir un producto adicional o si se puede usar directamente la aplicación. Por ello sugirieron todos, que se debería informar al usuario, sin ser tampoco intrusivo, de que va la aplicación y ya que es un producto que requiere ser comprado aparte, indicar donde pueden comprar dicho producto.

La idea principal de la aplicación es que la publicidad que se hace para publicitar el producto sea sobre la propia camiseta y no sobre la propia aplicación. La aplicación no recibiría ningún tipo de inversión o ASO para posicionarla en los buscadores, ya que no tiene sentido publicitar la aplicación si no se tiene la camiseta. Junto con la camiseta vendrían las instrucciones para descargar la aplicación y ponerla en marcha, es decir es una aplicación orientado a producto, tal y como se vio en la asignatura de *Tecnologías y Aplicaciones Multimedia*.

No obstante, este detalle que recalcaron los *Early Adopters* indican que se debe cubrir al menos todas las posibles entradas de un nuevo cliente y si el nuevo cliente aparece por que ha visualizado la aplicación en el *market* ya sea porque ha buscado apps relacionadas con camisetas inteligentes o con el cuidado a la 3ª edad, hay que estar preparados para reconducir a dicho cliente para que este informado de que va nuestro producto. Por ello desde la pantalla inicial de la aplicación debe existir una opción que permita a un usuario al clicarla recibir más información del mismo e incluso la posibilidad de comprar el producto.

No obstante, también se indicó la necesidad de que antes de mostrar la propia ventana de *login* ofreciera la posibilidad de registrar una cuenta o autenticar al usuario correspondiente, pero sin entrar específicamente en componentes o funcionalidades de que corresponden a la pantalla de *login*. En dicha ventana, los usuarios indicaron que debería ser la primera y por tanto la que puede llevar a explicar cómo funciona la aplicación.

Otro punto de los más importantes debatidos fue como permitir la autenticación en el sistema. La mayoría de los usuarios actuales utilizan los sistemas de autenticación que ofrecen Google, Facebook, Twitter, etc. En concreto, los 3 *Early Adopters* más jóvenes indicaron que ellos a día de hoy ya no rellenan ningún formulario de registro ni tan siquiera para registrarse en la aplicación. Salvo que sea algo de extrema necesidad, ellos no rellenan los campos y directamente descartan la aplicación. Si dejan que este sistema se realice por una plataforma como por ejemplo Facebook que es la plataforma que usan los 3 *Early Adopters* utilizarán la misma para identificarse y autenticarse en el sistema sin necesidad de mucho esfuerzo. Además, este esfuerzo se simplifica mucho más si la persona que usa la aplicación tiene la aplicación de Facebook instalada, lo que implicará que con un solo clic podrá registrar la cuenta y autenticarse.

Por último, dado que Facebook conlleva la problemática de conectar la cuenta de HealthShirt con Facebook, en el caso de que un día la quieran desvincular (haciéndolo desde Facebook) no tendrían manera de recuperar la cuenta del mismo porque desconocerían la contraseña



que tienen, por lo que sugirieron que existan los tradicionales métodos de recuperación de contraseña. Para ello en el momento que se conecte la cuenta por Facebook, si no existe una cuenta registrada, se solicitará al usuario que se proporcione una contraseña para crear la cuenta y si la cuenta ya existiera se le pedirá la contraseña de la misma para verificar que es el usuario el que se conecta.

Durante la entrevista de validación de las capturas de *Balsamiq* los usuarios echaron de menos una opción para recuperar la contraseña por lo que se rehízo el modelo para contemplar esta posibilidad. El mismo simplemente debería solicitar el nombre del usuario y enviar un correo indicándole que para recuperar la cuenta o restablecer la contraseña se haga clic sobre una URL y que se permita definir una nueva contraseña siempre siguiendo los criterios de contraseña que se hubieran definido.

## Prototipos



Figura 8 - Prototipos Login

### **Pantallas finales realizadas**

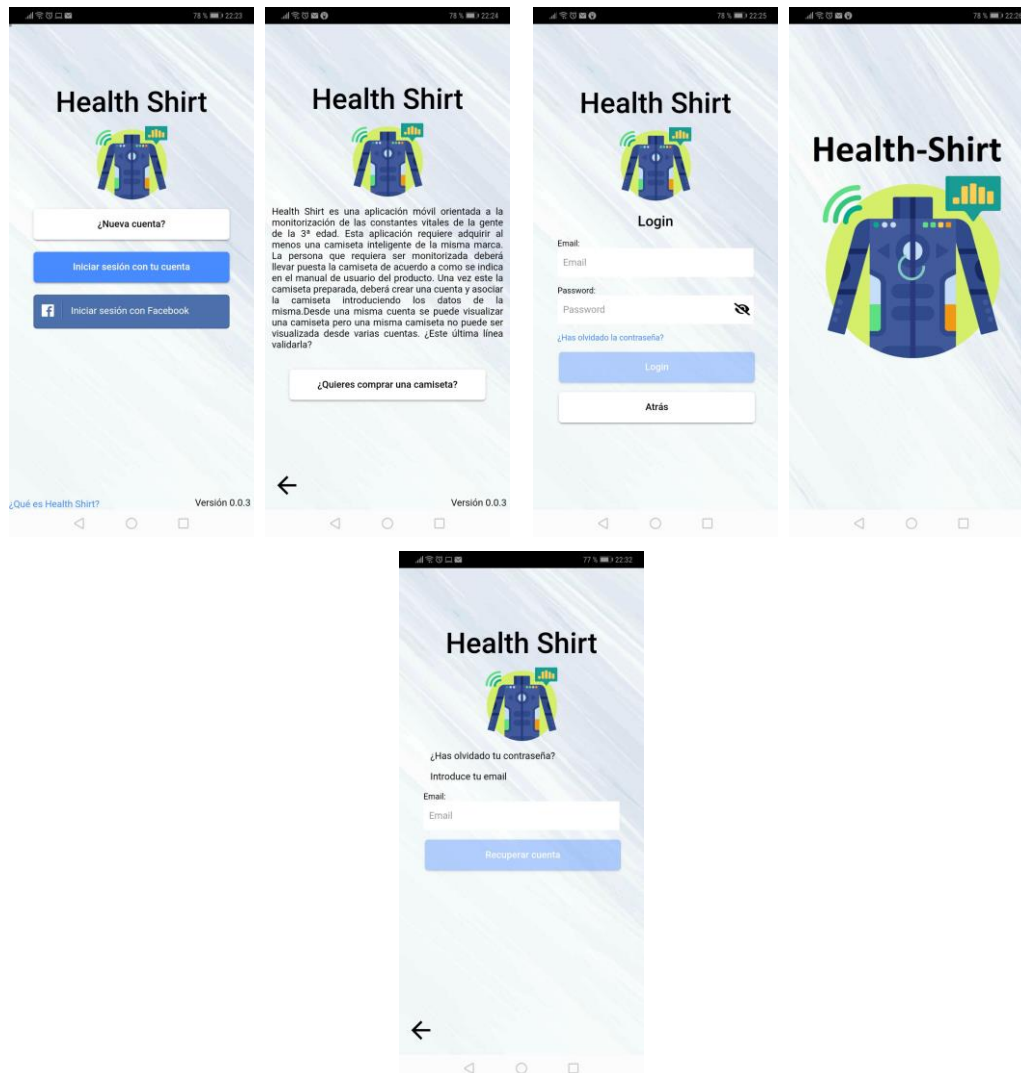


Figura 9 - Versión Final Pantalla de Login

### **Notaciones sobre la implementación**

Para la realización de esta pantalla se ha basado en la realización de distintas pantallas basándonos en el patrón *MVC* (Modelo vista controlador). Cada una de estas pantallas tiene su vista implementada en formato *html* y el controlador con extensión en formato *ts*.

Estas pantallas no tienen mucha complicación que requieran una extensa explicación. Lo principal de estas pantallas son los dos formularios creados, uno de ellos para permitir el login tradicional del usuario introduciendo únicamente un nombre de usuario y una contraseña. Estos formularios tienen controles para validar que todos los datos enviados son correctos, si no se cumplen estas condiciones entonces no se habilita el botón.

Las condiciones para permitir la autenticación de un usuario son las siguientes:

- El campo usuario debe estar relleno.
- El campo password debe estar relleno.
- El campo usuario debe ser una dirección de correo electrónico, la forma del mismo debe cumplir con el formato [direccion@nombredominio.extension](#), para ello se ha utilizado una expresión regular que sólo será cierta cuando se cumpla con el formato especificado arriba.
- El campo password debe contener al menos 6 caracteres.
- El campo password debe contener al menos una minúscula.
- El campo password debe contener al menos una mayúscula.
- El campo password debe contener al menos un número.
- El campo password debe contener al menos un carácter especial.

En el caso de que el usuario requiera recuperar la cuenta por el olvido de la contraseña se procederá a conectar con el servidor *backend* a través del servicio *Restful*. El mismo realiza 2 tareas. En primer lugar, asocia a este usuario un *token* de un solo uso para que pueda cambiar la contraseña. En segundo lugar, utilizando la función *mail* de *php* se envía un email al usuario de dicha cuenta. Hay que recordar que el nombre del usuario es la propia dirección de correo electrónico.

En dicho email, hay un enlace que contendrá el *token*, el cual va codificado. El *token* generado contiene información encriptada de la cuenta que se va a restaurar la contraseña y en ningún momento cuando se haga clic se va a mostrar al usuario a que cuenta pertenece. El motivo es mantener el anonimato de la cuenta en caso de que el enlace sea interceptado por un hacker.

Por último, indicar que el cambio de las pantallas de esta funcionalidad funcionará como una pila, se irán apilando, pero cuando se cierre o finalice en una pantalla se extraerá de la pila. En el momento que se complete el login, se resetea la pila, impidiendo que si se intenta volver atrás no se pueda volver a dichas ventanas, salvo que el usuario inicie un proceso de logout, en cuyo caso volvería a la página inicial.

En cuanto a la autenticación vía Facebook se ha creado una aplicación dentro la consola de desarrolladores de Facebook [27]. En la misma se ha dado de alta una aplicación en la cual sólo se va a solicitar los permisos de email, nombre y foto del usuario. Estos permisos son los que por defecto se proporcionan al crear cualquier app de Facebook. En nuestro caso sólo necesitamos el permiso de email ya que sólo nos interesa dicho dato para poder crear la cuenta o buscar la cuenta en nuestra base de datos para autenticar al usuario.

Para que el usuario pueda usar la autenticación por Facebook deberá tener la aplicación de Facebook o Facebook Lite instalada. Cuando se autentica correctamente en Facebook, el mismo solicitará al usuario a aceptar el permiso de que debe facilitar que se pueda leer su dirección de email. Una vez aceptado este permiso, Facebook enviará un *token* al móvil del cliente. Dicho *token* será remitido al servicio *Restful* para verificar que el usuario se ha autenticado correctamente, para ello se validará el *token* contra el SDK de Facebook, que aparte de utilizar el *token* que se le ha facilitado, se utilizará la API KEY proporcionada por la consola de desarrollador de Facebook para validar este *token*. Si el *token* es correcto entonces se procederá a buscar la cuenta. Si existe la cuenta entonces se validará la autenticación y la aplicación móvil procederá a mostrar el listado de las camisetas. Si no

existiera la cuenta se procederá a solicitar la contraseña para crear la cuenta. De esto se hablará en la creación de la cuenta en el *sprint* 2.

Todo lo explicado en los anteriores apartados se ha desarrollado basándonos en el plugin de Facebook 4 [28], el cual nos da las funcionalidades necesarias para poder realizar el registro de Facebook invocando al método *login* y luego recuperando los datos con el método *api*.

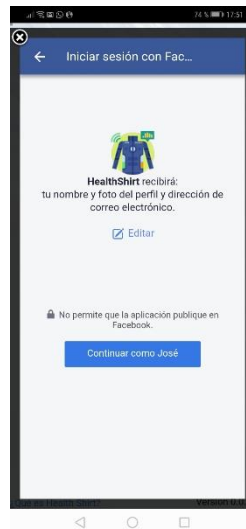


Figura 10 - Solicitud Permisos Facebook

Por último, tal y como se ha comentado en el apartado de las entrevistas, se ha tenido que implementar un mecanismo de recuperación de contraseñas. Cuando un usuario desea recuperar la cuenta, por que se le ha olvidado la contraseña, debe rellenar su nombre de usuario. Este dato se pasa al servicio *Restful* y si el usuario está registrado entonces se procede a generar un *token* que estará asociado al nombre de usuario y se envía un email que contiene una URL que contiene dicho *token* a dicho usuario utilizando la función *mail* de PHP.

Cuando se hace clic sobre la URL de dicho correo entonces el servidor comprobará si el *token* es correcto y si pertenece a un usuario registrado. En caso de que lo sea, se solicitará al usuario que introduzca la nueva contraseña, donde también tendrá que confirmarla para asegurar de que se escribe correctamente. La contraseña tiene los mismos criterios que se han indicado hace un par de párrafos. Si la contraseña cumple las condiciones, entonces se procede al cambio de contraseña. La contraseña se almacena en la base de datos encriptada en *base64*.

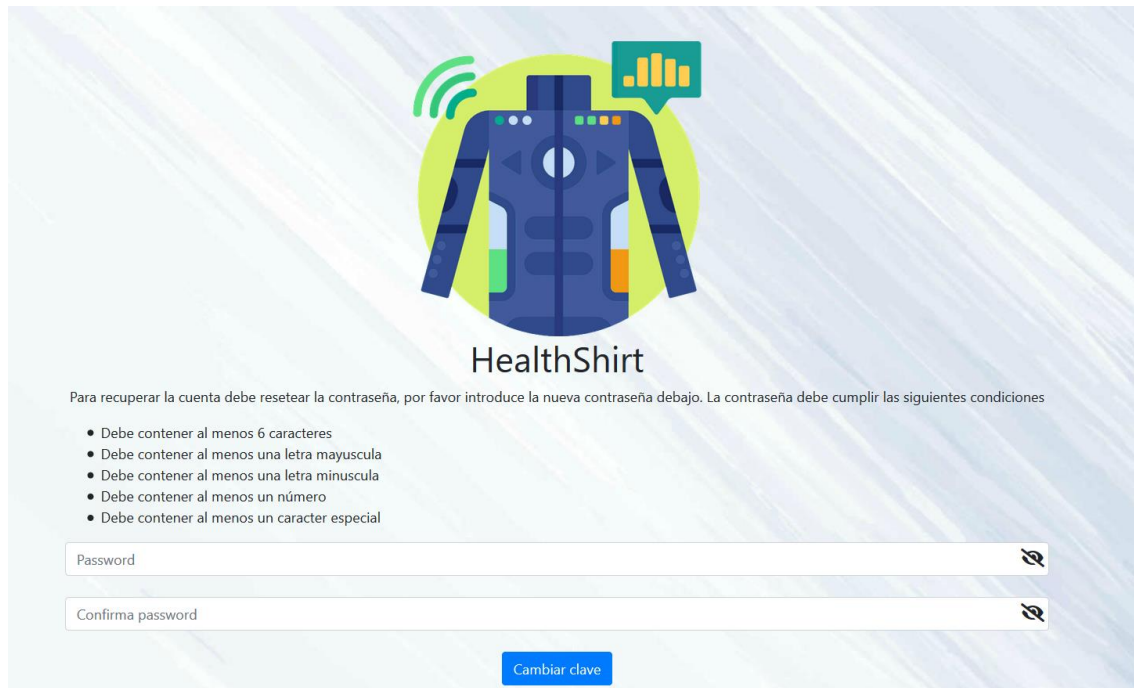


Figura 11 - Resetear contraseña o recuperar cuenta (Desde PC)

## 3.2 Sprint 2

### ***Pantalla de Registro***

En el apartado del *Login* se ha comentado toda la problemática relacionada con el sistema de autenticación. Dado que esta pantalla se ha hecho en el 2º sprint, se ha suavizado mucho la curva de desarrollo necesario para desarrollar esta funcionalidad, ya que se ha realizado gran parte de las pantallas del sistema de autenticación.

Esta pantalla debe permitir registrar una cuenta en el sistema, pero hay que tener en cuenta que en el *sprint* anterior ya se definió la posibilidad de registrar una cuenta una cuenta utilizando Facebook. De Facebook ya pueden obtener algunos datos como son el nombre del usuario, su dirección de correo electrónico o su foto, que son los datos que por defecto cualquier aplicación creada en la consola de desarrollo de Facebook pide por defecto. Como se ha indicado en el apartado de login, no es necesario pedir más datos ya que solamente es necesario la dirección de email para autenticarse en el sistema. Unos de los puntos importantes a analizar en este apartado es el número de datos que son necesarios recabar del usuario para poder registrar la cuenta, ya que es interesante ver si realmente necesitamos muchos datos personales del usuario a la hora de crear la cuenta o no.

Un número muy grande de datos puede hacer que un usuario se eche atrás a la hora de registrar la cuenta, aunque si se recaban muy poquitos datos es posible que no se tenga la suficiente información a la hora de darle correctamente el servicio.

## **Entrevistas**

El primer punto que se trató cuando se empezó a debatir el tema del registro de la autenticación fue precisamente sobre el número de datos y que datos se iba a solicitar a cualquier usuario para registrar la cuenta.

Los *Early Adopters* están completamente de acuerdo en que quieren rellenar la menor cantidad de información posible (si se recuerda, en el apartado del *login* se comentó que los *Early Adopters* comentaron que quieren rellenar la menor cantidad posible de formularios) por lo que dijeron que hay que disminuir al máximo esta información y sólo solicitar aquella que sea completamente imprescindible. Además, existe otro punto por el que es importante reducir esta información al máximo. Hay que recordar que en 2016 se puso en marcha el nuevo *RGPD* en el que hay que especificar a los usuarios de cualquier contenido digital como van a ser tratado sus datos y para que fines van a ser utilizados. Si se reduce al máximo esta información, en caso de riesgo de fuga, se consigue que los problemas derivados del mismo sean menores.

Para poder autenticar al cliente de forma correcta sólo es necesario un dato que permita identificarlo. Dado que de Facebook se va a recuperar la dirección de correo electrónico y el mismo es un dato unipersonal e intransferible, la dirección de correo electrónico es el dato que se utiliza para identificar al usuario. Ya una vez conectado el usuario en la cuenta, el mismo va a subir datos de las camisetas de la gente que está monitorizando la camiseta, pero no datos personales del propio usuario. Además, tampoco se va a solicitar al usuario que registre los datos de las camisetas en el registro, ya que puede ser que tenga 1, 2 o más camisetas e incluso puede que no tenga ninguna todavía.

Así que en principio no nos interesa conocer ninguno de los datos personales del usuario que quiere. Por ello, se acordó con los *Early Adopters* que la única información que va a pedirse o ser recuperada de Facebook es la dirección de correo electrónico. Además, para poder registrar la cuenta y permitir que los datos de la misma estén completamente protegidos se solicitará una contraseña que deberá ser por el usuario en caso de querer acceder a la aplicación.

En el caso de Facebook, si la cuenta no existe, se solicitará una contraseña para crear la cuenta como paso previo a la creación de la cuenta y al primer inicio de sesión.

Una vez registrada la cuenta, se enviará un nuevo correo solicitando la confirmación de la cuenta. Esta necesidad, aunque no fue expresada directamente por los usuarios como algo que necesiten, sí que informaron que, dado que se puede introducir cualquier dirección de correo electrónico, no se debe dejar que un usuario que se haya registrado con una dirección de correo electrónico que sea correcta se mantenga su cuenta, por lo que, si en 48 horas desde la creación de la cuenta no se ha activado la cuenta, la misma debería borrarse del sistema.

## Prototipos



Figura 12 - Prototipos Registro

## Pantallas finales realizadas

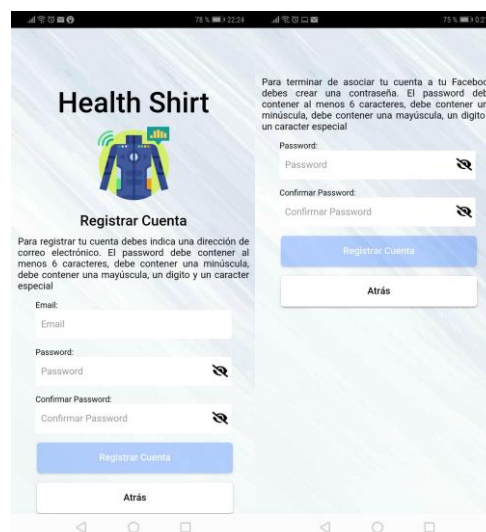


Figura 13 - Versión Final Registro de Cuenta

## Notaciones sobre la implementación

En este apartado se han seguido algunas de las características que ya se siguieron en el desarrollo de la pantalla de *Login*. Se recicla parte del formulario realizado en el *Login* como las expresiones regulares para comprobar el email, como las comprobaciones si se cumplen las condiciones de contraseña.

Lo único a recalcar es que cuando se crea una cuenta, indistintamente de si se ha creado directamente o a través de Facebook, se enviará un correo con un enlace que contiene un *token* codificado. Este *token* se genera en el servicio *Restful* en el momento que se registra



la cuenta y se asocia a la cuenta que se acaba de crear. El correo electrónico contendrá una URL que contendrá este *token* y se enviará al usuario utilizando la función *mail* que proporciona PHP. El usuario debe entrar en esta URL para confirmar la cuenta, ya que sino al cabo de 48 horas la cuenta será borrada de forma automática. Basta con acceder a la URL con el *token* una sola vez para que la cuenta se active.

Tanto si el usuario se ha autenticado como si el usuario ha registrado la cuenta e indistintamente de si se ha realizado a través de la aplicación o por Facebook se guardarán una serie de datos en la memoria del teléfono. Para ello se usará *sqllite-storage* de Cordova [29].

Una vez la cuenta se ha registrado correctamente, se pasará a la pantalla de listar camisetas que fue desarrollado en el mismo *sprint* y se solicitará al servicio *Restful* una solicitud para traer las camisetas pertenecientes al usuario que esta autenticado. Una vez situada en esta ventana el usuario no puede volver atrás, saliéndose de la aplicación si pulsa el botón de atrás. Para hacer este comportamiento se utiliza de la pila de navegación que incluye los sistemas operativos móviles. Cuando se accede a la página de listado de camisetas, se elimina toda la pila de navegación y cualquier nueva página que abra se introducirá en la pila haciendo que cuando se vuelva atrás se vuelva a la página del listado de camisetas y cuando no queda ninguna página en la pila se salga de la aplicación.

### ***Pantalla de Listar Camisetas***

Esta pantalla, es la primera que se visualiza una vez el usuario se ha autenticado en el sistema. Cualquier usuario que previamente estuviera también autenticado en el sistema también accederá directamente a esta pantalla. En la misma se muestra un listado con todas las camisetas que el usuario tiene registrado en el sistema. Hay que recalcar que cada una de estas camisetas hace referencia una camiseta que debe ser adquirida para ser agregada. No obstante, varios usuarios pueden visualizar los datos de la misma camiseta en caso de que estén de acuerdo en visualizar ambos los datos.

### ***Entrevistas***

En una primera reunión, los *Early adopters* indicaron que se debe visualizar rápidamente a quien pertenece cada camiseta, por ello se definió que se mostrara un icono que representara a la camiseta, pero los *early adopters* indicaron que no querían compartir imágenes personales, sino que se seleccionara entre un conjunto de iconos posibles permitiendo así que el usuario elija el icono que más se acerque a sus necesidades.

Además, después de la primera versión de las capturas, en el cual se muestra el icono, el nombre de la camiseta y el familiar al que pertenece; se indicó que sería interesante mostrar el porcentaje de batería que le queda a la camiseta, como así la fecha y hora a la que se ha recibido el último dato, así de esta manera el usuario puede conocer cuándo se ha recibido el último dato y si el mismo es diferente del actual, puede existir un pequeño problema.



Hay que recordar que en la pantalla de constantes vitales no el porcentaje de batería que queda disponible en la camiseta ni la última hora en la que el servidor recibió datos de la camiseta. Este último no es necesario mostrarlo ya que cuando se accede a dicha ventana, la misma entra en mostrar datos en tiempo real y por tanto lo que se está visualizando son los datos realizados en el último minuto.

Por tanto, los *early adopters* solicitaron que, junto con el nombre y parentesco de la camiseta, se indique también el porcentaje de batería disponible en la última actualización que se recibió, así como cuando fue dicha actualización.

### ***Prototipos***

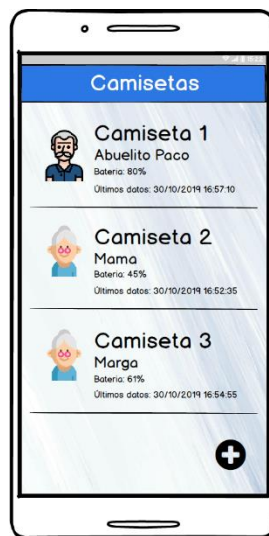


Figura 14 - Prototipos Listar Camiseta

### ***Pantallas finales realizadas***

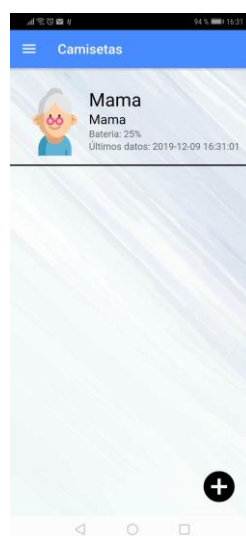


Figura 15 - Versión Final Listar Camisetas

## **Notaciones sobre la implementación**

Para realizar este listado se utiliza un elemento de *IONIC* denominado *ion-list* en el cual definimos un elemento *ion-item* donde montaremos la vista utilizando etiquetas *html*. Por cada camiseta que se recupera del servidor se mostrará en su *ion-item* correspondiente. Cuando va a cargarse la vista de la página del listado de páginas, se llama a un *provider* denominado *rest-camiseta*, el cual actúa de controlador, permitiendo que se lancen las peticiones para recuperar los datos de las camisetas.

Además, para evitar que el usuario necesite cambiar de página y volver a acceder, se utiliza otro componente denominado *ion-refresher* el cual cuando el usuario desplaza con el dedo el listado fuerza la actualización de los datos refrescándose el listado de las camisetas, así como los datos de las mismas.

En cuanto al icono que se muestra en la camiseta, se obtiene junto con los datos de la camiseta. Se obtiene una URL donde se encuentra la imagen (se encuentra almacenada en el mismo servidor que el servicio *Restful*) por lo que al cargarse esa camiseta la misma procede a buscar y mostrar la imagen a partir de dicha URL.

## **3.3 Sprint 3**

### ***Pantalla de Registrar y Editar Camisetas***

En esta pantalla se ha definido la funcionalidad que permite a los usuarios crear una camiseta, así como editar los datos de la misma o borrar una camiseta. Esta funcionalidad permite a los usuarios tras haber adquirido una camiseta, dar de alta la camiseta con un mínimo de datos, editarlos y cuando ya no sea necesario borrar las camisetas. Algunos de los datos que se necesitan definir ya fueron definidos en la pantalla de *listar camisetas* como son el nombre de la camiseta, el grado de parentesco y el icono que representa al icono.

### ***Entrevistas***

Aunque en principio la implementación de esta pantalla iba a ir de forma separada con la pantalla o funcionalidad de definición de los umbrales, la misma se acaba definiendo en la misma pantalla debido a que los usuarios expresaron que los umbrales tienen que ir por camiseta y no de forma global. De todo esto se hablará en el apartado entrevistas de la funcionalidad de definición de los umbrales.

La idea que definieron los usuarios es en que se pueden ir dando de alta las camisetas en las que luego se podrá visualizar el dato de las mismas. Las camisetas requieren una serie de datos mínimo para poder ser dadas de alta. Como se ha comentado en los *sprint* anteriores, los usuarios quieren rellenar la menor cantidad de datos posibles, así que se van a incorporar los menos posibles además de los indicados por los *early adopters*. Los mismos ya indicaron que era importante un icono que serán el que utilicen principalmente para identificar la camiseta y luego el nombre de la camiseta y el nombre del familiar o parentesco

con el mismo. Todos estos datos son definidos por el usuario y son obligatorios, pero no son los únicos datos que se necesitan.

Entre ellos los que más destacan son el número de serie y el código de seguridad. La razón de introducir el segundo, es para evitar que cualquier persona malintencionada pueda acceder a la camiseta probando números de serie al azar. Este código de seguridad es un código que introduce el usuario en una microsd conectada a la camiseta, la cual solamente las personas interesadas conocen dicho valor y pueden añadir la camiseta proporcionando el mismo. En el momento que se desee que una persona no tenga acceso a la camiseta, simplemente deberá cambiarse dicho código, momento que las personas que no tengan agregada la camiseta con dicho código de seguridad, no puedan seguir visualizando los datos.

Para dar de alta la camiseta, un usuario deberá introducir por lo menos un icono, un nombre, el grado de parentesco, el número de serie que viene incluido en la camiseta que han comprado y el código de seguridad que han definido en la microsd de la misma. Si algunos de estos dos últimos datos no son correctos no se permitirá dar de alta la camiseta, indicando que los datos son incorrectos.

Cuando se edite los datos de la camiseta, se podrán editar los mismos datos excepto el número de serie y el código de seguridad, ya que el mismo no puede modificarse ya que son datos invariables de la camiseta. Estos dos datos son únicamente para restringir que se pueda dar de alta la camiseta sin tener acceso realmente a ella, una vez se ha añadido la camiseta, se entiende que las ediciones van a ser siempre sobre la misma camiseta. El resto de datos seguirán siendo obligatorios de rellenar y no podrán ser vaciados tras haber dado de alta la camiseta.

En las entrevistas se habló de situaciones en las que una o más personas pueden querer visualizar los datos de la misma camiseta, pero sin necesidad de compartir la cuenta, por lo que el sistema debe permitir que dos usuarios completamente distintos puedan dar de alta y visualizar los datos de la misma camiseta siempre que ambos usuarios conozcan el número de serie y código de seguridad para dar de alta la misma.

Para dar de alta una camiseta se hará clic sobre un botón de “añadir” que será representado por el botón más y se situará en la parte inferior de la pantalla. Al hacerse clic sobre dicho botón se cambiará a la página de creación de camisetas.

Para poder editar o borrar las camisetas, se incorporarán dos botones por cada una de las camisetas que se muestran en el listado de camisetas. Dichos botones no deben visualizarse directamente, sino que deben mostrarse únicamente cuando el usuario arrastre la camiseta hacia la izquierda. En dicho momento se mostrarán dos botones, uno representado con un lápiz sobre un fondo verde que indicará para editar la camiseta y el otro es de una papelera sobre un fondo rojo que indica el borrado del mismo.

Cualquier acción que se realice de las descritas en esta funcionalidad, deberá refrescar el listado nuevamente para actualizar cualquier cambio que se haya realizado en los mismos. Si se ha dado alta una camiseta nueva, se debe mostrar en el listado junto con las demás. Si

se ha editado una camiseta, si se ha modificado el nombre o el parentesco de la misma, debe mostrarse los cambios. Por último, si se ha borrado la camiseta, debe desaparecer del listado.

Para terminar este apartado, los *Early Adopters* indicaron que, junto con los umbrales, desconocían para que servían todos los campos que aparecían en el formulario de registrar o edición de las camisetas, por lo que sugirieron que debería haber un mecanismo para informar al usuario que se debe rellenar en cada uno de los campos o que información se espera en los mismos. Por ello se propone realizar una nueva página que se puede acceder a través del icono información que aparecerá en la barra superior de ambas páginas. Cuando se pulse sobre dicho icono se abrirá una nueva página (se apilará en la pila de navegación) donde se mostrará un texto donde se incluirá la información para cada uno de los campos que conforman el formulario y ejemplos sobre como rellenar el mismo.

### Prototipos

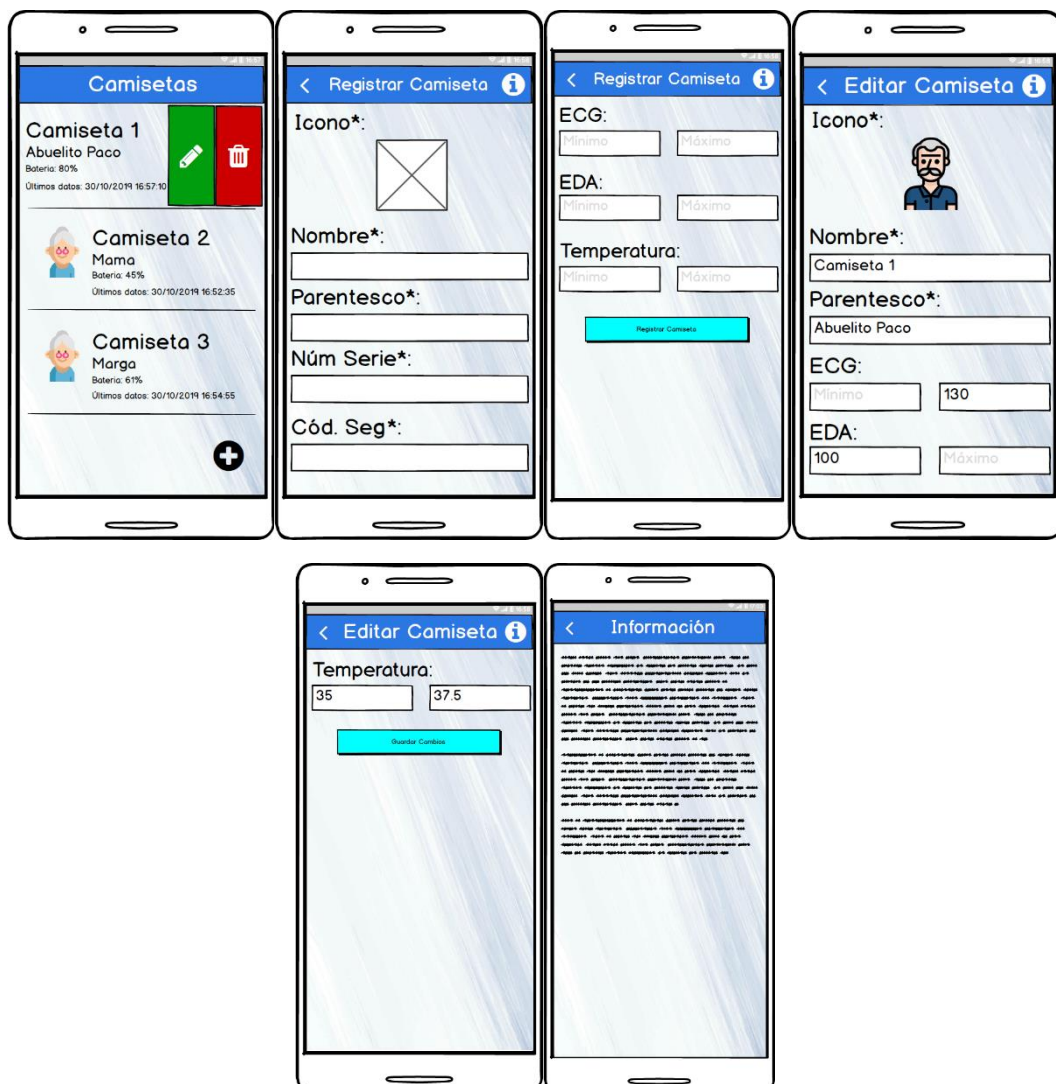


Figura 16 - Prototipos Crear/Editar camisetas

## Pantallas finales realizadas

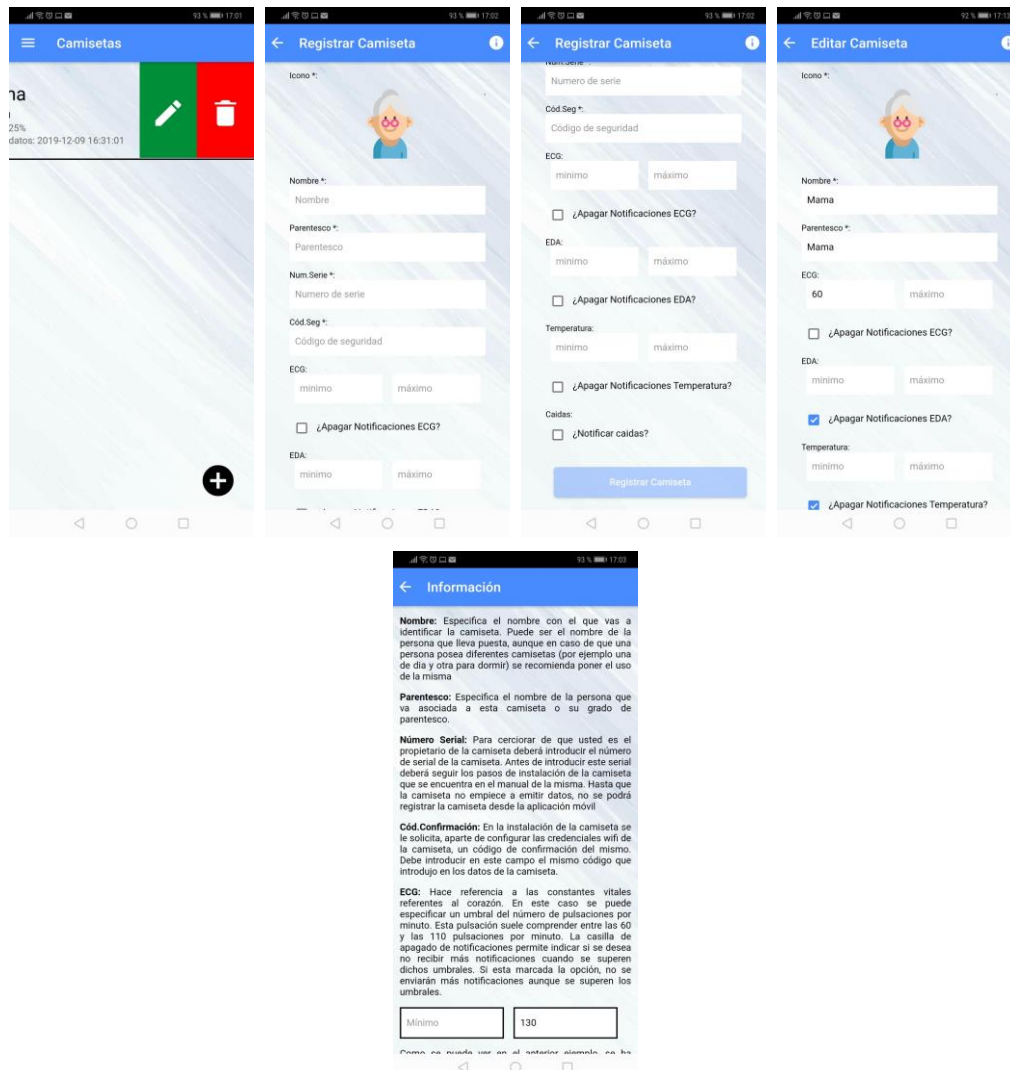


Figura 17 - Versión Final Crear Editar Camiseta

## Notaciones sobre la implementación

Esta funcionalidad es posible junto con la funcionalidad de las constantes vitales una de las que más trabajo requiere para hacerla funcionar. Hay que tener en cuenta que no se pueden visualizar los datos de las constantes vitales si el usuario no ha dado de alta la camiseta, por lo que esta es posiblemente la segunda funcionalidad más importante de la aplicación.

Se va a empezar hablando de la página de creación de las camisetas. La explicación que se va a dar sobre dicha página es análoga a la que se puede hacer sobre la página de edición de las camisetas, pero teniendo en cuenta las siguientes diferencias.

- Los campos Número de serie y Código de seguridad no se visualizan en la edición.
- Los campos aparecen rellenos con la información de la camiseta en la edición.

- El botón de guardar tiene un texto diferente. En la creación de la camiseta pone *registrar camiseta*, mientras que en la edición pone *guardar cambios*.
- En la barra superior el texto es diferente. En la creación de la camiseta pone *Crear camiseta*, mientras que en la edición pone *Editar camiseta*.

En primer lugar, a la hora de crear el formulario, se ha definido una serie de restricciones sobre algunos de los campos del formulario. A continuación, se muestran los únicos campos que tienen algún tipo de restricción para poderse validar el formulario.

```
nombre: new FormControl("", Validators.required),
parentesco: new FormControl("", Validators.required),
numeroserie: new FormControl("", [Validators.required, Validators.minLength(16), Validators.maxLength(16)]),
codseg: new FormControl("", Validators.required),
icono: new FormControl("", Validators.required),
```

Como se ve, se ha indicado que los campos son obligatorios y deben ser rellenados, pero, además, los números de serie de la camiseta están ya definidos y se restringen a valores de 16 caracteres. Para rellenar un número de serie correctamente debe indicarse una secuencia de 16 caracteres (por ejemplo, el 1111111111111111). El resto de los campos que existan en el formulario no tienen ninguna restricción, esto es por ejemplo la definición de los umbrales que se explicará en la próxima sección.

El problema principal que tiene este formulario es precisamente el elemento más importante que consideraron los *Early Adopters* que es el que usaran para identificar la camiseta, que es el icono identificativo de la camiseta. El mismo se tiene que seleccionar y no escribir una URL ni ninguna clave identificativa. Por defecto, un combo de selección permite elegir una opción que se encuentra escrita y no como una imagen propiamente dicha. Por ello se necesita cambiar el comportamiento del combo (*ion-select* en *IONIC*) para que haga 2 acciones.

```
prepararImagenesIcono() {
  setTimeout(() => {
    let buttonElements = document.querySelectorAll('div.alert-radio-group button');
    console.log(buttonElements);
    if (!buttonElements.length) {
      this.prepararImagenesIcono();
    } else {
      for (let index = 0; index < buttonElements.length; index++) {
        let buttonElement = buttonElements[index];
        console.log(buttonElement);
        let optionLabelElement = buttonElement.querySelector('.alert-radio-label');
        let image = optionLabelElement.innerHTML.trim();
        console.log(image);
        buttonElement.classList.add('imageselect', 'image_' + image);
        if (image == this.imagen) {
          buttonElement.classList.add('imageselected');
        }
      }
    }
  });
}
```

```

    }
  }
}, 100);
}

```

En primer lugar, en el momento que el usuario haga clic en el combo, el mismo debe obtener cada una las posibles opciones a mostrar y modificar la estructura interna del combo para añadir una nueva clase CSS que permita pintar como fondo de la opción del combo dicha imagen. Así que a cada opción de la imagen se le añadirá la clase `image_nombreimagen` cuya definición de estilos vía SCSS, permite transparentar el texto de la opción y mostrar como fondo de la imagen y centrado la imagen seleccionada para dicha opción tal y como se muestra en la siguiente regla definida en SCSS.

```

@each $imageOption in $imagesToSelect {
  &.image_#{$imageOption} {
    &, &.select-text {
      color: transparent;
      background-image: url("../assets/images/#{$imageOption}.png");
      background-size: contain;
      background-repeat: no-repeat;
      background-attachment: fixed;
      background-position: center;
    }
  }
}

```

Con ello se consigue que la imagen quede justo en medio, como fondo de la opción y además que el texto de la opción no se visualice.

Aparte existe otro problema. En el momento que se seleccione una opción el comportamiento por defecto del combo es mostrar el texto de la opción seleccionada. Por ello, cuando se detecte que se ha modificado el valor (evento *onchange*) se debe de invocar otro método que permita reemplazar el texto de la opción seleccionada por la imagen de la misma. Para ello el método que se muestra a continuación:

```

ponerImagen(imagen) {
  let buttonElements = document.querySelectorAll('div.alert-radio-group button.imageselect');
  console.log(buttonElements);
  for (let index = 0; index < buttonElements.length; index++) {
    let buttonElement = buttonElements[index];
    buttonElement.classList.remove('imageselected');
    if (buttonElement.classList.contains('image_' + imagen)) {
      buttonElement.classList.add('imageselected');
    }
  }
}

```

```
}
}
```

En el mismo se realizan acciones muy similares a las explicadas anteriormente, aunque ahora la idea es simplemente que solamente la opción del combo que haya sido seleccionada es la lelimine dicho atributo. Si no se eliminara, a medida que el usuario fuera seleccionando opciones, se irían rellenando los campos de opción con la imagen asociada, lo que al final haría que se visualizará incorrectamente todas las opciones con una imagen y sin que las mismas estuvieran ocultas.

Dicha clase lo que permite es que a la opción del combo que haya sido seleccionada, se elimine el texto, tal y como se muestra en el código SCSS que se muestra a continuación. Además, fuerza al contenido asociado a un combo como son el icono de la selección del mismo a que desaparezca, ya que no queda bien cuando se visualiza junto con la imagen.

```
.select-text {
    @include fixed_width(100%);
    border-radius: 5px;
    color: transparent;
}

.alert-radio-inner,
.alert-radio-icon,
.alert-radio-label {
    display: none;
}
```

Luego se utiliza el mismo código SCSS visto anteriormente (ya que las clases de opciones del combo son hijas de la clase del combo padre) para situar como fondo de la opción la imagen que ha sido seleccionada. Aunque esto no es suficiente ya que, al seleccionar la opción del combo, el mismo centra el contenido en la parte izquierda del mismo (si se hablara del texto, el mismo aparecería situado en la parte izquierda del combo), por tanto, se debe indicar una serie más de atributos para corregir este problema. Por tanto, se indica mediante los atributos *position*, *left* y *top* que el mismo se debe situar en mitad del espacio disponible. Para asegurarnos de que el mismo está abarcando el 100% del ancho y altura disponible se le indica mediante los atributos *height* y *width* que el mismo debe ocupar el 100% del espacio disponible.

Estas acciones hacen que el menú quede de la siguiente manera en la que se visualiza en la figura 18.

Se ha hablado de la nueva página de creación de camiseta que se puede acceder desde el botón o icono con el símbolo más situado en la parte inferior del listado de las camisetas. La cuestión es que no se ha hablado de cómo se accede a la manera de editar una camiseta o como borrar la misma. Ambos botones deben aparecer para una camiseta en específico cuando se arrastra hacia la izquierda dicha camiseta, lo que permite a un usuario acceder a la página de edición de una camiseta o al borrado de la misma.



Dado que es difícil por accidente borrar una camiseta (ya que esto implica directamente la acción de arrastrar y de darle al botón de borrado de la misma) no se ha implementado ningún mecanismo de seguridad o de confirmación del borrado, por lo que un usuario no deberá confirmar la acción, ya que el mismo ha procedido a arrastrar hacia la izquierda la camiseta elegida y a seleccionar el borrado de la misma.

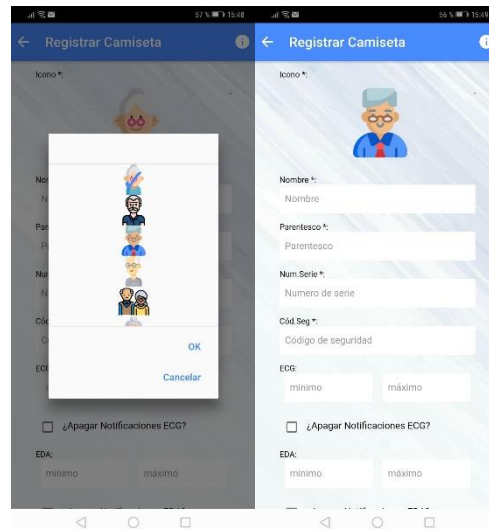


Figura 18 - Combo de Icono con Imágenes

Dado que los botones deben encontrarse ocultos hasta que el usuario decida arrastrar dichos botones, el componente de *ion-list* definido es insuficiente, ya que el mismo, aunque contiene elementos para situar componentes en distintas partes del elemento de la lista (mediante los atributos *start* y *end*) esto haría que simplemente los botones salieran en el lugar designado del elemento. Por tanto, se añade dos componentes adicionales que permiten añadir el componente que nosotros deseamos.

Dado que cada componente se va a comportar como un elemento que va a poderse deslizar con un dedo, se debe utilizar previamente a *ion-item* el componente *ion-item-sliding*. Este componente lo que hace es que cualquier contenido que este incluido dentro del mismo pueda ser desplazado por el usuario en la dirección que se le indique (por defecto hacia la izquierda).

El otro componente necesario es *ion-item-options*. Este componente contiene un grupo de *ion-item-option* donde se podrá definir la vista asociada al elemento que se desea mostrar. En nuestro caso se utiliza *ion-icon* para mostrar un icono que representa el elemento sobre la acción a realizar. Los elementos de *ion-item-options* sólo saldrán cuando el usuario realice una acción sobre el componente *ion-item-sliding* y se mostrarán en la posición indicada siempre y cuando la dirección del deslizamiento sea la apropiada.

```
<ion-item-options side="end" >
  <ion-item-option class="tamano_boton botonEdicion" (click)="editarCamiseta(camiseta)"><ion-
icon class="en_medio" name="create"></ion-icon></ion-item-option>
  <ion-item-option class="tamano_boton botonBorrar" (click)="borrarCamiseta(camiseta)"><ion-
icon class="en_medio" name="trash"></ion-icon></ion-item-option>
```

</ion-item-options>

Luego como se ve en el anterior código se utilizan dos clases CSS para indicar que cada botón es distinto. La clase *botonEdicion* simplemente indica que el color del fondo del componente es de color verde, mientras que la clase *botonBorrar* indica que el color del fondo del componente es de color rojo.

### ***Pantalla de Definición de Umbrales***

En esta funcionalidad se va a definir la manera de especificar los umbrales en los cuales cuando la camiseta inteligente transmita al servidor los datos de nuevas constantes vitales, si para alguna de las constantes se supera alguno de los umbrales definidos para dicha constante se procederá a enviar una notificación a dicho dispositivo.

### ***Entrevistas***

La idea inicial con la surgió esta funcionalidad era definir un menú configuración donde en el mismo se definirían los umbrales de todas las camisetas. En las entrevistas iniciales se indicó la necesidad de que el propio usuario pueda distintos niveles de umbral que permitan controlar cuando las constantes vitales superan un valor mínimo o máximo para tomar medidas. Dichos umbrales en principio se definieron en un principio sobre todo el conjunto de usuarios del mismo usuario, indistintamente de la camiseta sobre la que llegara la notificación. No obstante, cuando se presentó el primer prototipo de *Balsamiq* sobre esta pantalla, los usuarios replicaron rápidamente que no todos los pacientes van a tener los mismos umbrales, ya que lo que para un paciente puede ser normal para otro puede que sea un nivel peligroso.

Por poner un ejemplo a lo especificado en el párrafo anterior, la temperatura de una persona suele rondar entre los 36º y 37º centígrados. Si se definiera un umbral de 36º para la temperatura, si se tuviera una persona con la camiseta puesta que su temperatura media fuera inferior a 36º centígrados, la misma estaría disparando notificaciones todo el tiempo, lo que provocaría que se dispararan situaciones de alerta cuando realmente no las hay.

Este factor lo recalcaron sobre todo los *Early Adopters* que tienen varias personas a su cargo. No todas las personas mayores presentan las mismas necesidades ni tienen los mismos problemas, por lo que no se debe globalizar los umbrales de las constantes. Por tanto, los *Early Adopters* indicaron que para cada una de las camisetas que se de alta en la aplicación debe incorporar sus propios umbrales.

Estos umbrales deben mostrarse después de todos los datos obligatorios a rellenar en el registro y edición de camisetas, ya que los mismos, no son obligatorios. Los usuarios no están obligados a rellenar dichos umbrales. Si un umbral no se rellena, simplemente es como si dicha constante no tiene definido un umbral.

Los usuarios indicaron que hay deberían definirse dos tipos de umbral por cada una las constantes. Un umbral de valor mínimo y un umbral de valor máximo, esto es importante distinguirlos, ya que para ciertas constantes puede ser de información valiosa el obtener un caso por debajo de lo normal que como encima de lo normal. Por tanto, para cada una de las

constantes que existe en el programa se debe permitir definir un umbral máximo y un umbral mínimo, pero ninguno de estos umbrales será obligatorio para poder registrar o editar la camiseta. Estos umbrales deberán por tanto estar situados dentro de las páginas de creación y edición de camisetas.

## Prototipos

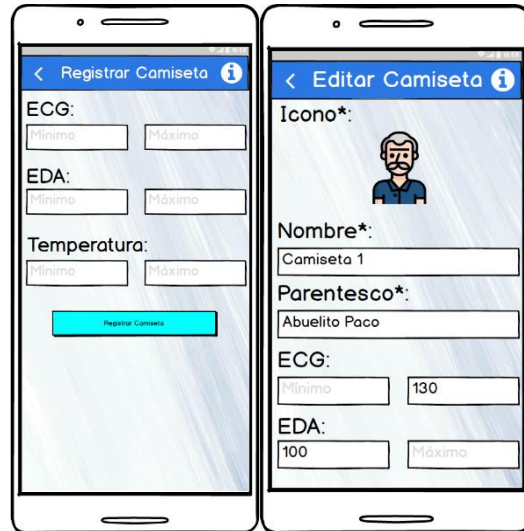


Figura 19 - Prototipo definición umbrales

## Pantallas finales realizadas

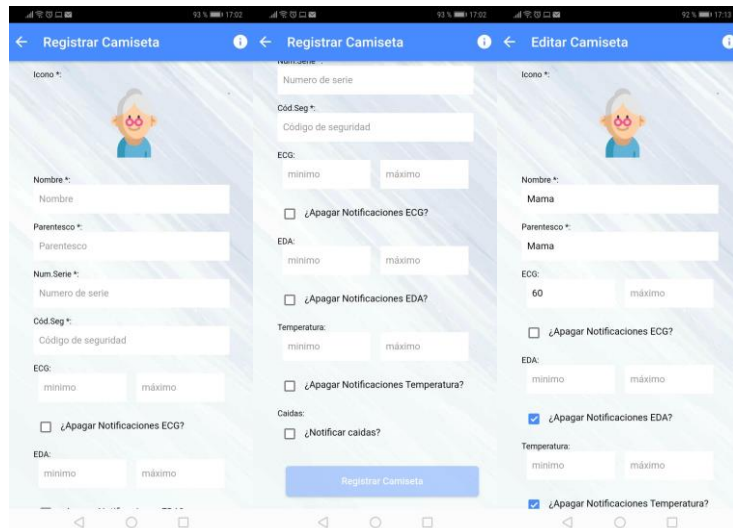


Figura 20 - Versión final definición umbrales

## Notaciones sobre la implementación

La implementación de esta funcionalidad pasa simplemente por añadir nuevos campos a los formularios ya implementados en las páginas de creación y edición de las camisetas. Basta simplemente con añadir dichos campos al formulario que se ha definido en el apartado de creación y edición de camisetas, pero en este caso se debe indicar que no existe ninguna acción de validación asociada a dichos campos. Esto se consigue simplemente dejando el

segundo parámetro del componente *FormControl* un listado vacío, ya que el mismo indica un listado de todas las validaciones que se deben realizar sobre el componente definido.

```
ecgminimo: new FormControl("", []),
ecgmaximo: new FormControl("", []),
edaminimo: new FormControl("", []),
edamaximo: new FormControl("", []),
temperaturaminimo: new FormControl("", []),
temperaturamaximo: new FormControl("", []),
```

Con esto se tienen definidos los campos, pero ahora falta definirlos en la vista, pero esta ocasión tanto el campo que define el umbral mínimo, como el que define el umbral máximo deben aparecer en la misma línea para que sea más fácil asociarlos al mismo componente. Para poder conseguir este efecto se utiliza *FlexBox* [29]. Con *FlexBox*, con unas simples reglas se consigue que los elementos queden alineados siguiendo una orientación que nosotros le indicamos y respetando una separación.

```
.display-inline{
  display:flex;
  @include fixed_width(100%);
  justify-content: space-around;
  flex-wrap: nowrap;
}
```

En este caso le estamos indicando 3 propiedades. La primera de ellas es que dicho componente va a utilizar una visualización *flex* con lo que los elementos que están dentro de dicho componente van a distribuirse de la forma que se indique el resto de atributos que definamos. Con la propiedad *flex-wrap* indicamos si permitimos que los componentes que no quepan en una sola línea puedan moverse a la fila inferior. En este caso se ha definido el valor *nowrap* que fuerza a que todos los componentes deban mostrarse en la misma fila. Finalmente, con *justify-content* se indica la alineación que van a tener los componentes alineados. En este caso se ha utilizado *space-around* con lo cual se puede definir que los espacios entre la esquina y el primer elemento y la esquina y el último elemento sean iguales. Además, se utiliza el resto del espacio disponible para separar los componentes entre ellos. Como sólo existen dos componentes por línea esto permite poner el resto del espacio disponible entre la separación entre el primer campo y el segundo campo.

Además, aunque se ha hecho para el resto de campos de los formularios de todas las páginas que tienen login, para los campos de umbral se ha definido el atributo *placeholder* lo que permite indicar un texto mientras el campo no haya sido rellenado con ningún valor. Aquí es importante definir qué campo es el mínimo y qué campo es el máximo, para que el usuario sepa en todo momento qué campo debe rellenar.

### 3.4 Sprint 4

Cuando se inició el sprint 4 se tenía previsto realizar la funcionalidad de las notificaciones de las constantes vitales a los usuarios. No obstante, tal y como se ha indicado en la funcionalidad de visualización de las constantes vitales, esta ventana se encontraba incompleta, en primer lugar, porque el desarrollo del mismo todavía no había terminado durante el primer *sprint* y en segundo lugar porque los *testers* de la aplicación no habían podido probar si funcionaba ya que no existe al día que se escriben estas líneas, ningún prototipo de la camiseta que envíe los datos de los sensores al servidor, por lo que nunca hay valores para recuperar y se espera que el prototipo del mismo esté disponible para mediados de marzo.

Por tanto, el *sprint 4* se rehízo para abarcar la parte del *frontend* de las notificaciones, es decir, incluir aquellas cosas necesarias que deben permitir a los usuarios activar o desactivar las notificaciones que se reciban de las constantes vitales y en segundo lugar para terminar el apartado de la visualización de las constantes vitales y adelantar el desarrollo del 5º sprint en realizar un script o programa que se encargue de generar los datos de constantes vitales para una camiseta a modo de prueba.

#### **Notificaciones (Frontend)**

Cuando el valor de una constante vital supera un cierto umbral definido se debe notificar al usuario que está monitorizando la camiseta que se ha superado dicho umbral y el valor del mismo para que el usuario pueda tomar medidas para corregir la situación en caso de que sea necesario.

En este apartado se va a explicar la realización de la parte que permite a los usuarios indicar si quieren recibir o no las notificaciones relacionadas con las constantes vitales, así como otras notificaciones relacionadas con la detección de la caída de la persona que lleva la camiseta, así como el nivel de batería del mismo

#### **Entrevista**

Los *Early Adopters* indicaron que debe existir una notificación por cada una de las constantes vitales que se están visualizando en la aplicación, por lo que la aplicación debe permitir apagar o encender las notificaciones referentes a cada una de las constantes vitales. Adicionalmente, hay un par de eventos que se deben notificar al usuario que tiene registrada la camiseta en caso de que se produzcan en caso de que el usuario solicite que se le envíen dichas notificaciones.

El primero de estos eventos es si la persona que lleva la camiseta se cae. Hay que tener en cuenta que el prototipo está orientado a que lo lleven las personas mayores. La gente mayor suele tener problemas a la hora de andar, por lo que suelen llevar bastones para apoyarse o ayuda de terceras personas para poderse desplazar. En bastantes ocasiones se produce la situación de que la persona mayor, que esta acostada en la cama, se levanta en plena noche, sin depender de nadie más, por alguna necesidad y se cae. Dicha caída puede ser grave y

más todavía si no se puede asistir a dicha persona a tiempo. Por tanto, la aplicación debe permitir encender o apagar las notificaciones relacionadas con las caídas de cada camiseta. Este evento debe poderse configurar por camiseta.

El segundo de estos eventos, es el nivel de batería. Cuando el nivel de batería de la camiseta inteligente disminuye por debajo de un 20%, se debe notificar al usuario indicando que la camiseta le queda poca batería, indicando el porcentaje del mismo. Esto permite al usuario conocer que la camiseta no le queda mucha batería y que la misma necesita un reemplazo lo antes posible. A diferencia de las notificaciones anteriores, esta notificación debe definirse a nivel global. La razón es, que a diferencia de las constantes vitales o las caídas las cuales no se necesitan monitorizar todas al mismo tiempo, el nivel de batería es crítico y que afecta a todas las camisetas, por lo que dicha configuración debe hacerse a nivel global y no a nivel de camiseta.

Además, se puede dar situaciones en las que una persona a cargo de varios usuarios necesita ser notificadas de distintas constantes para distintas personas. Puede ocurrir que, en algún momento, la persona cuidadora a cargo necesite que no se le notifiquen más eventos hasta pasado un periodo de tiempo. Para solucionar esto, el usuario deberá poder apagar desde la aplicación las notificaciones a nivel global, tanto las notificaciones de las constantes vitales, como la de la caída.

Dado que existen varios tipos de notificaciones distintas, se debe habilitar una opción que permita apagar todas las notificaciones del programa. Si esta opción es marcada, no se recibirá ninguna notificación más hasta que la opción sea desactivada.

Por último, dado que se debe configurar el apagado o encendido de las notificaciones a nivel global, se debe definir una nueva página que contenga dicha opción. Tal y como se ha descrito la aplicación, simplemente se tiene un listado de camisetas y en la misma página se puede insertar una camiseta, editarla, borrarla, o si se hace clic encima de una de las camisetas acceder a la página de visualización de las constantes vitales. Hay que añadir una nueva página y se debe acceder desde algún punto de la aplicación.

Para solucionar esto se va agregar un menú que se podrá acceder desde cualquier parte de la aplicación en la que el usuario este autenticado. Para acceder a dicho menú se deberá arrastras hacia la derecha desde la parte izquierda del dispositivo móvil y aparecerá un menú flotante sobre el contenido actual. En dicho menú se deberá visualizar el nombre del usuario, un botón de logout que los usuarios en las pruebas de los *sprint 1* y *2* solicitaron que hacía falta y luego dos opciones de menú. La primera de ellas se denominará camiseta y cuando se haga clic se visualizará la página del listado de las camisetas. La segunda de ellas se denominará *Configuración*. En la misma se debe mostrar al usuario una página que permita configurar las diversas opciones que permita el programa. En este caso las únicas opciones que se van a aparecer de momento en esta página es la configuración de las notificaciones globales. Dicho menú fue hablado durante el *sprint 3* pero se procedió a su implementación durante el *sprint 4*.

## Prototipos

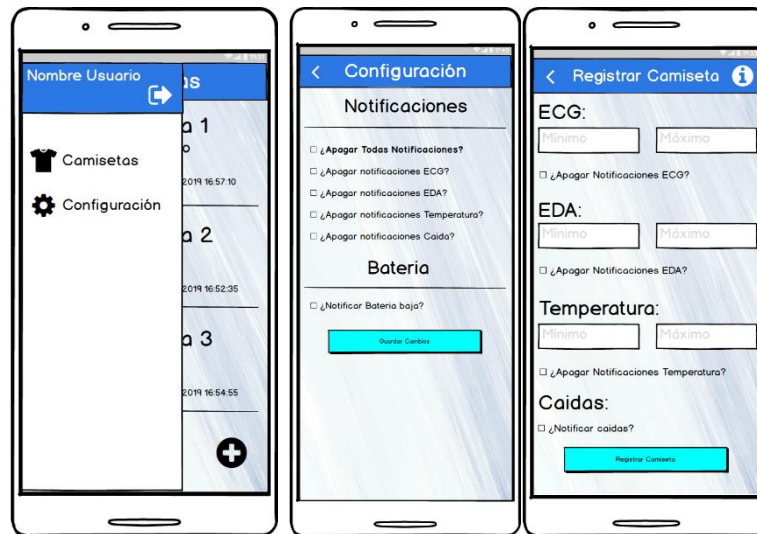


Figura 21 - Prototipo Notificaciones

## Pantallas finales realizadas

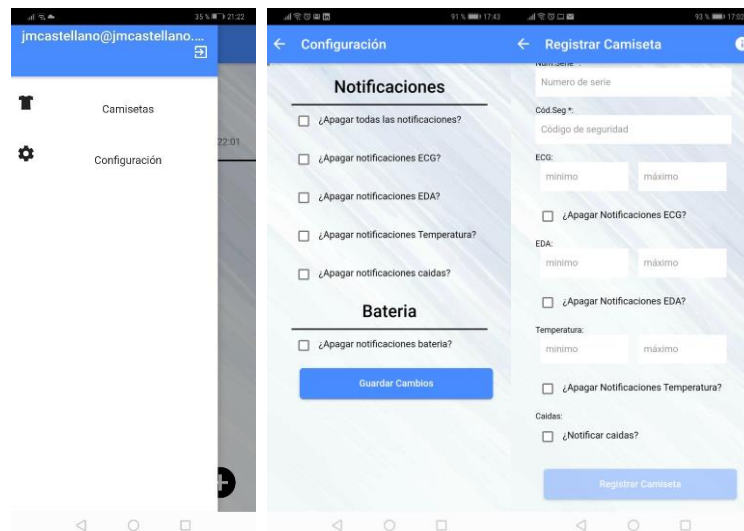


Figura 22 - Versión final Notificaciones

## Notaciones sobre la implementación

La implementación de esta funcionalidad se ha dividido en 3 partes. En primer lugar, se debe crear una nueva página que almacenará un formulario que contendrá 6 *checkbox* o cajas de selección. Las mismas se dividen en 2 partes. La primera de las partes estará conformada por 5 *checkbox* y se solicitará por el encendido o el apagado de las notificaciones relacionadas con el bienestar de la persona que lleva la camiseta. El último de los *checkbox* servirá para apagar o encender las notificaciones relacionadas con la batería. Dado que ninguno de estos campos es obligatorio, el formulario definido no requiere de ninguna validación y por tanto el botón siempre estará habilitado, por tanto, en la lista de condiciones a validar por componente, la misma se encontrará vacía para todos los componentes.

```

notificacioneseceg: new FormControl("", []),
notificacioneseda: new FormControl("", []),
notificacionestemperatura: new FormControl("", []),
notificacionestodas: new FormControl("", []),
notificacionescaida: new FormControl("", []),
notificacionesbateria: new FormControl("", [])

```

Adicionalmente, el comportamiento por defecto de los formularios de *IONIC* es colocar el texto o label del componente encima del campo. Aunque esto queda bien para campos de tipo texto no queda bien para campos de tipo *checkbox* ya que el mismo ocupa muy poquito en espacio y por tanto es preferible que ambos componentes estén en la misma línea. En este caso se vuelve a utilizar la clase CSS de *display-inline* que se definió y explicó el apartado de creación de los umbrales para los mínimos y máximos, aunque ahora la diferencia viene en que se utiliza una propiedad adicional que es *flex-direction* que permite definir en qué dirección se van a definir los componentes. En este caso se ha definido como *row-reverse* que los coloca en fila en modo inverso ya que en la vista se ha colocado primero el label y luego el componente, cuando en realidad en la captura de *Balsamiq* que se ha realizado el *checkbox* se encuentra primero y luego el texto asociado al *checkbox*. Si se cambiara en la vista el orden de los componentes, entonces se debería modificar el valor de la propiedad a *row* que es el valor por defecto del campo *flex-direction*. Una vez que se pulsa el botón que se incorpora, se ejecuta una acción que procederá a obtener los campos que han sido seleccionados y se invocará un *provider* denominado *rest-configuracion* que se encarga de llamar al servicio *Restful* para actualizar los datos de configuración en el cliente. Estos datos se deben de almacenar en el servidor, para que así el servidor tenga constancia de que notificaciones debe o no enviar. Además, así se permite mantener los datos en caso de que el usuario desinstale la aplicación.

La segunda parte requiere de un menú que se pueda acceder desde todas las páginas de la parte autenticada de la aplicación y que se encuentre deshabilitada para las demás. Para hacer este comportamiento se utilizará de un *navigation drawer* el cual es un menú lateral que se puede mostrar al accionar un icono o al realizar un desplazamiento con el dedo sobre la pantalla. Dado que es un elemento en común a la mayoría de las pantallas de aplicación se pondrá sobre la página genérica de la aplicación, que es aquella con la que arranca la aplicación en *IONIC* y es la que se mantiene en memoria hasta que la aplicación finaliza su ejecución. Dado que esta página no se le había definido una vista, se procede a definir una vista para la misma, pero el código mismo se comportará como un menú *navigation drawer*.

```

ion-menu [content]="content">
  <ion-header>
    <ion-toolbar color="primary">
      <ion-title class="margenabajo">
        {{ staticnombreusuario }}
        <ion-icon (click)="go('init')" class="derecha" name="exit"></ion-icon>
      </ion-title>
    </ion-toolbar>
  </ion-header>

```



```

<ion-content class="menu">
  <ion-list class="listado-menu">
    <ion-item (click)="go('camiseta')" class="item-menu">
      <ion-icon name="shirt" class="icon_menu" item-left></ion-icon>
      <ion-label>Camisetas</ion-label>
    </ion-item>
    <ion-item (click)="go('configuracion')" class="item-menu segundomenu">
      <ion-icon name="settings" class="icon_menu" item-left></ion-icon>
      <ion-label>Configuración</ion-label>
    </ion-item>
  </ion-list>
</ion-content>

</ion-menu>

<ion-nav #myNav [root]="rootPage" #content swipeBackEnabled="false"></ion-nav>

```

Como se ve en el código anterior se puede observar que el menú está englobado con el elemento *ion-menu*. El menú está formado por una cabecera donde se ubicará una barra (o *toolbar*) donde irá el nombre del usuario y el icono o botón asociado para que el usuario se pueda desconectar de la sesión. Si se pulsa este botón las credenciales se borrarán del storage interno de la aplicación (utilizando el mismo plugin *sqlite-storage* que se habló en los procesos de login y registro del usuario).

A continuación, vienen los componentes principales del menú de navegación. Los mismos están formados por un listado (definido con *ion-list*) de opciones (definidos con *ion-item*) que permite a los usuarios que al ser pulsados se pueda acceder a la página correspondiente. Por último, para conseguir el efecto del menú lateral se debe especificar que este contenido debe anclarse como un contenido adicional de la página principal (la que se encuentra más al fondo de la pila). Esto se consigue con la última línea donde ancla el menú a la página principal e indica que este contenido debe aparecer cuando se desliza el dedo. La propiedad *swipeBackEnabled* impide que una vez que el usuario ha termina el deslizamiento, si suelta el dedo, el contenido vuelva a desaparecer. En nuestro caso interesa mantenerlo para que el usuario pueda hacer las acciones deseadas.

La última de las partes que se ha necesitado realizar introducir 4 nuevos *checkboxes* dentro de las páginas de creación y edición de camisetas. Esto sigue una explicación parecida a la explicada para la pantalla de configuración. Se han añadido los nuevos campos asociados al campo y dado que los campos no son obligatorios no requieren de ningún tipo de validación.

```

notificacioneseceg: new FormControl("", []),
notificacioneseda: new FormControl("", []),
notificacionestemperatura: new FormControl("", []),
notificacionestodas: new FormControl("", []),
notificacionescaida: new FormControl("", []),

```

Para colocarlos se sigue la misma estructura que se realice para la configuración, se vuelve a utilizar componentes CSS FlexBox para situar tanto la descripción del *checkbox* como el propio *checkbox* en la misma línea respetando una separación y ocupando en la medida de lo posible todo el ancho disponible.

Cuando el usuario envíe el formulario, los valores de dichos campos se enviaron junto con los datos de la camiseta al *provider* rest-camisetas que se encargará de contactar con el servicio *Restful* para actualizar dichos datos.

### **Constantes Vitales (Generación de Datos)**

En el desarrollo de este apartado se realizaron los cambios que todavía necesitaba el sistema para hacerlo funcionar, además tal y como se ha comentado en el apartado de visualización de las constantes vitales, se necesitan disponer de datos para verificar que el funcionamiento es correcto. Para ello en este apartado nos vamos a centrar exclusivamente en la generación automática de los datos para una camiseta (aunque la misma se podría extrapolar fácilmente a un montón de camisetas).

Para ello, en primer lugar, se va a encargar el servidor *Apache* que actúa de *backend* el que se va a encargar de generar estos datos ficticios. Por ello el servidor dentro de la carpeta *generador* se tiene disponible un script denominado *generarDatos*. Este script simula en gran parte el comportamiento de los sensores de la camiseta inteligente el cual envía un paquete de datos al servidor para que se almacene cada minuto. Para empezar, deben empezar generándose los datos para las 3 constantes vitales. Los sensores de las camisetas cogen 1200 medidas por segundo para los datos de ECG y EDA y sólo una para la temperatura. La misma debe enviarse al servidor.

Para el cálculo del ECG hay que recordar bien como era el funcionamiento del corazón. Cuando un sensor de ECG está conectado a una potencia de 3.3V, la potencia en la que el corazón en esta de reposo devuelve es de 200, los pequeños picos que se producen no suelen oscilar sobrepasar los 300 o 350 y por debajo no suele disminuir en menos de los 100. Cuando se produce el pico grande, significa que ha empezado el proceso de la *sístole* proceso que lleva a su valor máximo (o cercano a él) que es de 650, para luego disminuir el mismo hasta casi 0. Cuando finaliza el siguiente pico (que no suele rebasar los 350), acaba el proceso *sístole* y vuelve a la *diástole* que es la zona plana con un valor sobre 200. Todo este proceso es una simple pulsación cardíaca. Dado que son 1200 capturas por minuto, si se quiere simular que se han tenido unas 60 NPM se debe reproducir el anterior 60 veces a lo largo de las 1200 capturas por minuto, lo que implica que cada pulsación debe durar 20 muestras. No obstante, no todo el mundo tiene las mismas pulsaciones cardíacas, así que para simplificarlo se ha construido que se puedan dar 3 número de pulsaciones distintas, las cuales son 60,90 y 120. Con 90 pulsaciones por minuto se necesita que las mismas vayan ubicadas en unas 13 mediciones, aunque algunas intercalándose con otras. Con 120 pulsaciones se necesita que cada pulsación se encuentre únicamente en 10 mediciones.

Por tanto, se define una función en *PHP* que construya una array que contenga las 1200 características. No obstante, los sensores están sujetos a errores y es muy raro incluso en la zona plana que siempre se obtengan los mismos valores, lo mismo ocurre con el valor de los picos. Los picos no tienen siempre por qué ser de la misma potencia, el corazón en ocasiones

hace pulsaciones más fuertes o pulsaciones más débiles según la cantidad de sangre que este circulando en dicho momento por el corazón, por tanto, se definen una serie de funciones que se encargan de generar un valor posible en el rango posible de valores para dicho tramo

Obtener valor plano: Genera un valor entre 195 y 205 que suele representar el corazón en reposo.

Pico300: Genera un valor entre 290 y 310 que representa al pico que se produce al pico previo antes de que empiece la *sístole*.

Pico250: Genera un valor entre 240 y 260. Este representa al primer pico que se produce al iniciarse un nuevo ciclo de pulsación.

Pico400: Genera un valor entre 390 y 410. Este representa al pico final tras la *sístole*, y en el cual empieza el proceso *diástole*.

PicoMáximo: Genera un valor entre 630 y 650, representa el pico en el cual empieza la *sístole*.

PicoMínimo: Genera un valor entre 0 y 20, representa el pico mínimo que se produce tras empezar la *sístole*.

Se utilizan estas funciones siguiendo el orden del latido cardiaco, teniendo en cuenta más rápidas son las pulsaciones menos veces se debe llamar a la función de *obtener\_valor\_plano* que en pulsaciones más pausadas se debe invocar la misma varias veces seguidas. Para simplificar las posibles pruebas, simplemente se han implementado 3 posibles número de pulsaciones (60,90 y 120), aunque se podría implementar fácilmente simplemente determinando ante un nuevo ciclo si la misma debe acelerarse o ralentizarse.

Ya que estamos hablando del ECG, hay que recordar que la aplicación móvil en la página de *Constantes* en la pestaña de ECG se tiene que visualizar el número de pulsaciones medias en la animación con los datos que se tienen actualmente. Los datos que nos llegan son valores entre 0 y 650 que no representan al número de pulsaciones, sino a los distintos voltajes obtenidos. Lo que se desea obtener son el número de pulsaciones para poder calcular la media, ¿Cómo se consigue esto? Si se ha fijado el lector en el diagrama de un electrocardiograma existen 2 procesos en el proceso de un latido. Justo cuando el proceso de la *sístole* empieza, se produce el mayor pico que tienen el mayor voltaje y a continuación se produce el menor pico que tiene el menor voltaje. Con esto basta simplemente contar el número de picos máximos (por lo menos aquellos valores que superan 600 de voltaje) y dividirlos por el número de minutos que abarcan el periodo de tiempo que se esté analizando. En el caso de los datos en tiempo real que se obtienen los datos del último minuto la simple cuenta de estos picos máximos nos dará ya el número de pulsaciones medias, pero en caso de que estemos observando un histórico, el mismo se tendrá que dividir entre el número de minutos que abarque el periodo seleccionado.

Para la generación de los datos del EDA el mismo suele tener un valor plano y continuo (variando en unas pocas unidades debido al error de lectura del sensor) a lo largo del tiempo. El valor medio leído en los sensores que irán en la camiseta suele rondar sobre las 200 unidades, modificando estos valores cuando varíe el nivel de conductancia de la piel dependiendo de cómo de tranquilo o intranquilo este el usuario. Solamente cuando se detectará una situación de intranquilidad o nerviosismo o un estado de mucha relajación se

debería variar la situación del mismo. Por ello se ha definido que exista un 1% de probabilidades de que el valor leído se desplome y un 1º de probabilidades de que el valor leído aumente. Por tanto, se elige un valor entre 0 y 100 y si el valor es 0 entonces se genera un valor entre 70 y 110. Si el valor elegido es 1, entonces se genera un valor entre 290 y 350 y si el valor elegido es distinto a los dos anteriores entonces se genera un valor entre 190 y 210.

Para calcular el valor de EDA medio, simplemente basta con calcular la media de los 1200 datos recibidos.

Para generar la temperatura solamente necesitamos leer el dato una vez por minuto y por tanto sólo debe generarse un dato de temperatura. Dado que este valor no debería cambiar entre un minuto y el minuto siguiente, se debe leer de la base de datos el último valor de temperatura generado en el sistema y se puede producir 3 situaciones:

- Aumentar en 0.1º la temperatura.
- Disminuir en 0.1º la temperatura.
- Mantener la temperatura.

Para ello se genera un número random entre 1 y 10 y si el mismo es del 1 al 3 se disminuye el valor, si el valor es del 8 al 10 se procede a aumentar el valor y para el resto de valores se procede a mantener la temperatura. Este enfoque, tiene un problema que no debería darse si se estuvieran obteniendo los datos desde la camiseta inteligente. Se puede dar situaciones en las que el valor de temperatura decremente mucho hasta el punto en que las temperaturas no serían reales a los que se obtendrían del ser humano, como por ejemplo temperaturas corporales de 10º. Para ello se limita la temperatura a un rango entre 34º y 38º de temperatura (se pueden obtener temperaturas más altas en un ser humano si se poseen fiebres muy altas, pero para este estudio es suficiente).

Para el cálculo de la media de la temperatura basta simplemente con sumar todas las temperaturas obtenidas en el periodo de tiempo analizado y dividirlo entre el número de minutos que existe en el intervalo definido. En el caso de que se estén visualizando los datos en tiempo real, se visualizan las temperaturas obtenidas en los últimos 10 minutos.

Por último, se ha hablado de que se tiene que realizar las notificaciones de que la batería se agotando. Por tanto, en cada nueva ejecución del script (que se ejecuta una vez cada minuto) se obtendrá el porcentaje de batería que tiene actualmente la camiseta y se le restará un 1%. Cuando el valor llegue a 0, se procederá a volver a poner el porcentaje de la batería al 100%.

Una vez generados todos los datos, se procede a insertarlos en la base de datos como un único registro de datos. A partir de este punto lo último que le faltaría hacer a nuestro script generador es comprobar si debe lanzar alguna notificación a los usuarios que tienen asociadas dicha camiseta. Esto se explicará en el apartado de *notificaciones backend* del *sprint 5*.

### 3.5 Sprint 5

Durante este *sprint* se ha ido preparando ya la documentación de la aplicación (manuales de usuario, de instalación, etc) como la presente memoria. No obstante, como se ha podido ver a lo largo del *sprint 4* el desarrollo de la funcionalidad de las notificaciones se ha dividido en 2 partes. La primera de las partes o la parte *frontend* fue desarrollada durante el *sprint 4*, mientras que la segunda de las partes o la parte *backend* es desarrollada durante el *sprint 5*.

#### **Notificaciones Backend**

En esta ocasión vamos a trabajar con el mecanismo para poder comunicar al cliente que tenga las notificaciones activas, una notificación que contiene información con el evento u umbral que se ha superado con los datos actuales que se han leído desde la camiseta inteligente. Para ello, se va a estudiar que necesidades se tienen que abordar para que los usuarios estén informados de cuando llegue una notificación.

#### **Entrevista**

Los *Early Adopters* indicaron que las notificaciones recibidas deberían incluir principalmente 3 datos. La constante vital sobre la que se ha superado el umbral, que umbral ha sido superado, el valor del umbral que ha rebasado dicho valor y el nombre de la camiseta sobre la que se produce dicho umbral. La información debe aparecer de la siguiente forma.

**Título:** Aviso de *ConstanteVital Umbral* superado – *NombreCamiseta*

**Subtítulo:** Se ha alcanzado *ValorUmbral*

Adicionalmente existen 2 tipos más de notificaciones que se pueden recibir en la aplicación. La primera de ellas es si se detecta una caída. En caso de recibirse una notificación de este tipo la misma debe mostrarse de la siguiente forma:

**Título:** Aviso de *caida* – *NombreCamiseta*

**Subtítulo:** El usuario con la camiseta *NombreCamiseta* se ha caído

El último tipo de notificación que puede recibirse es la referente a que la batería se está agotando. En caso de recibirse una notificación de este tipo la misma debe mostrarse de la siguiente forma:

**Título:** Aviso de *batería baja* – *NombreCamiseta*

**Subtítulo:** La camiseta *NombreCamiseta* le queda un *ValorBateria* % de bacteria

Si las notificaciones se reciben teniendo la aplicación abierta y en primer plano la acción asociada a la notificación se realizará automáticamente, mientras que si la notificación se recibe en segundo plano entonces se presentará una notificación en el dispositivo del usuario y la acción sólo se realizará cuando se pulse sobre la notificación (abriendo previamente la aplicación). La acción a realizar por cada una de las notificaciones será la siguiente:

- Si la notificación es referente a una constante vital, se abrirá la pantalla de visualización de las constantes vitales sobre la camiseta a la que hace referencia la

notificación. Se abrirá directamente sobre la pestaña que haga referencia la constante vital.

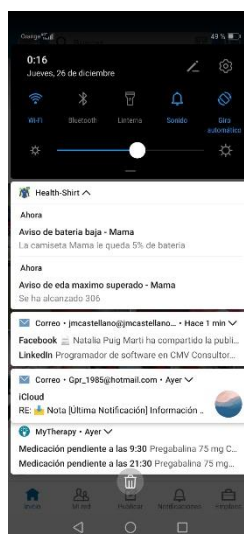
- Si la notificación es referente a la caída de la persona, se abrirá la pantalla de visualización de las constantes vitales sobre la camiseta a la que se hace referencia la notificación. Se abrirá sobre la pestaña ECG.
- Si la notificación es referente a la batería. Se abrirá el listado de camisetas.

Por último y para no interferir la actividad del usuario. Si el usuario está visualizando la pantalla de constantes vitales y llega una nueva notificación referente a unas constantes vitales no se recargará la pantalla, se mantendrá en la actual pantalla, aunque la misma no haga referencia a la camiseta sobre la que ha llegado las notificaciones.

### Prototipos



### Pantallas finales realizadas



### **Notaciones sobre la implementación**

Para enviar las notificaciones a los usuarios cuando un umbral sea superado por una constante vital se necesita de dos medios. El primero de los medios es el servidor web donde se están recibiendo los datos provenientes de la camiseta inteligente y que están almacenando dichos valores en la base de datos.

Cuando la camiseta inteligente genera el paquete de datos a enviar el servidor, será recibido por este mismo y procederá a analizar la información que contiene la misma para determinar si los umbrales definidos por alguno de los usuarios han sido superados por alguno de los valores que se ha recibido en el servidor.

Por poner un ejemplo, si se ha definido un umbral mínimo para el EDA de 90 y entre los valores obtenidos para el EDA se detecta que el mismo ha alcanzado un valor de 85, entonces se procederá a disparar una notificación, siempre y cuando el usuario tenga habilitado las notificaciones en el sistema.

Para enviar la notificación al usuario se necesita un servicio adicional de mensajería. Los dispositivos móviles son capaces de recibir mensajes de tipo FCM y para ello se puede utilizar el servicio de *Firebase Cloud Messaging* que se puede consultar en [30]. Este servicio permite enviar a los usuarios registrados en una aplicación enviarles notificaciones con las que comunicarles avisos u ofertas relacionadas. En cada mensaje FCM se pueden enviar dos tipos de mensajes a los clientes

- Mensajes de notificación, que simplemente son informativos.
- Mensajes de datos, que contienen datos que puede ser utilizado por la aplicación.

En nuestro caso se van a enviar notificaciones o mensajes del segundo tipo ya que se deben comunicar diversos datos a la aplicación móvil del cliente para que el mismo pueda realizar las acciones pertinentes. Solamente cuatro datos se deben comunicar para poder realizar las acciones asociadas a la llegada de la notificación.

- Tipo de notificación o constante (Si es ECG, EDA, temperatura, caída o batería)
- El identificador de la camiseta.
- El nombre de la camiseta.
- El número de serie de la camiseta.

Aunque los dos últimos se podrían acceder a partir del identificador de la camiseta, en la página de visualización de las constantes vitales no se rescatan los datos de la camiseta, estos datos ya se pasan por parámetro desde la página del listado de camisetas a la página de visualización de las constantes vitales. Para evitar tener que hacer una búsqueda adicional para recuperar datos que ya se encuentran disponibles, se enviará dicha información desde el servidor para evitar una búsqueda adicional para que se puedan mostrar en pantalla directamente, ni sobrecargar las búsquedas sobre el servidor.

Aunque esto mismo conlleva otro problema, que va relacionado con el cómo se conoce a quien se tiene que enviar dichas notificaciones. Cuando un usuario se registre y autentique en la aplicación es un potencial usuario para recibir las notificaciones. Hay que recordar

también que dos usuarios pueden estar visualizando cada uno los datos de la misma camiseta pero que ambos hayan definido distintos umbrales e incluso que uno de estos usuarios tenga las notificaciones encendidas mientras que el otro las tiene apagadas. Para solucionar esto, el sistema *Firebase Cloud Messaging* asocia a cada usuario que se registra en la aplicación un *token* identificativo bajo el nombre de *registration\_id*. Este *token* se genera en el momento que el usuario lo solicita por primera vez y es renovado cada cierto tiempo.

De todos modos, aunque se genere este *token*, es insuficiente para poder hacer funcionar nuestro sistema de notificaciones, ya que el servidor no tiene acceso a estos *token*, por lo que en el momento de que nuestros usuarios obtengan o vuelva a obtener el *token*, deberá notificarlo al servidor junto con las credenciales del usuario para asociarlo. Creando esta asociación entonces el servidor si tendrá acceso a los datos del *token* asociado al usuario de la camiseta que acaba de superarse cierto umbral y por tanto facilitando el *id de la aplicación* y el *token* asociado al usuario enviar una notificación al mismo.

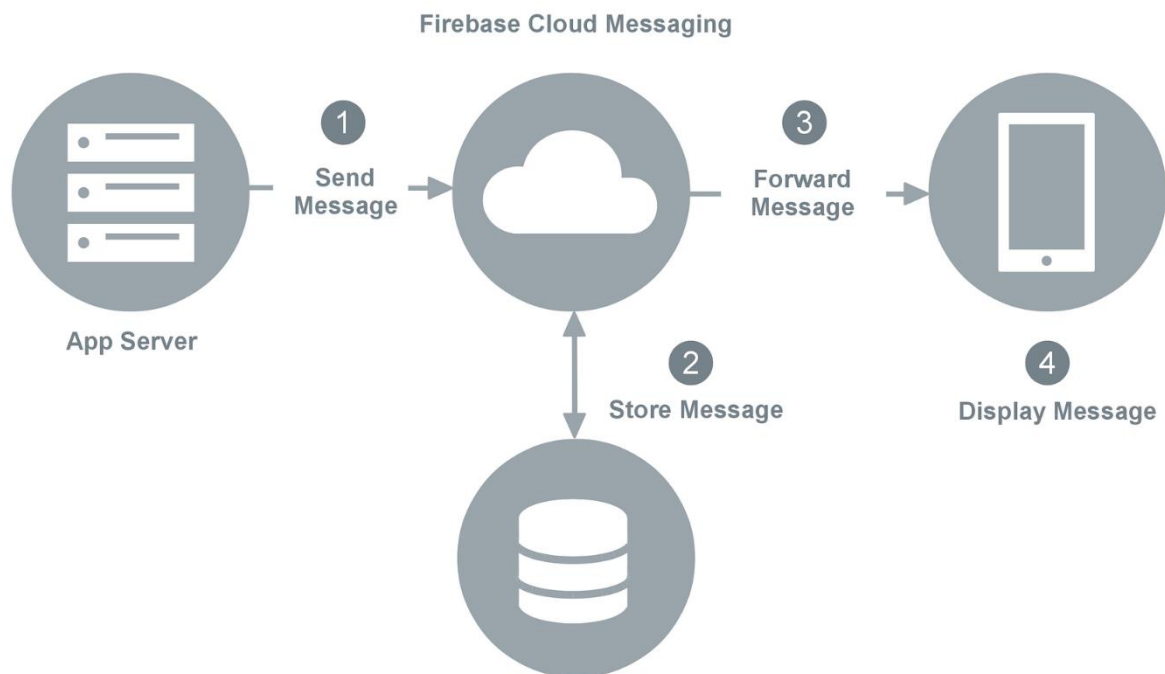


Figura 23 - Arquitectura de Firebase Cloud Messaging (Fuente: Microsoft)

En la figura 23 se observa, como funciona este sistema de notificación. En primer lugar, el servidor web que quiere comunicar el mensaje o notificación a la aplicación móvil destino procede a enviar un mensaje indicándole una serie de parámetros a través de una API que nos proporciona el propio sistema de Firebase (del cual se explicará más adelante). El mismo lo primero que hace es comprobar si dicho mensaje es correcto (y existe tanto la aplicación como al usuario/s a los que se dirige) y almacena en su base de datos el mensaje, por si no fuera posible enviarlo en dicho momento proceder a enviarlo más tarde. Por último, se encarga de mandarle el mensaje o notificación vía push al dispositivo el cual desplegará la notificación en pantalla en el cual, la aplicación asociada a la notificación se encuentra abierta desplegará automáticamente los cambios que estén programadas para ella y si la aplicación no se encuentra abierta o se encuentra en segundo plano entonces se mostrará una notificación.



Por tanto, se necesitan realizar 2 modificaciones, una sobre nuestro servidor web y la otra sobre nuestra aplicación móvil. Nuestra aplicación móvil necesita instalar un nuevo plugin denominado *cordova-plugin-fcm-with-dependency\_update* [31] que permite tanto registrar el *token* en el servicio de *Firebase Cloud Messaging* como de recibir las notificaciones en el sistema.

Dentro del plugin y dado que se desea que es un comportamiento que debe detectarse en toda la aplicación, incluso si la misma no se encuentra en funcionamiento o se encuentra en segundo plano es importante registrarla nada más se inicia por primera vez la aplicación. Por tanto, lo primero que se debe de hacer es invocar al método *getToken* del plugin para pedirle recuperar un plugin y en el momento en que se reciba la contestación, se procederá a subir dicha información al servidor para que pueda asociar al usuario a un número de *token*. Luego el usuario es asociado en el momento en que se crea la camiseta por lo que al final a partir de la camiseta que ha recibido la constante vital que supera el umbral permitido se puede obtener el *token* del usuario asociado.

Con el método *onTokenRefresh* se puede suscribir para detectar que se modifica el valor del *token*. Los *token* proporcionados por Firebase no son eternos y acaban caducando, por lo que en el momento que dichos *token* caducan Firebase le comunica que le ha sido proporcionado un nuevo *token*. Cuando ocurra dicho evento se debe realizar las mismas acciones que se produjeron la primera vez que se le asignó el *token*, es decir, subir el *token* al servidor para asociarlo al usuario, para ello se utiliza el mismo método del *provider* para indicar dicha información.

A continuación, hay que suscribirse al evento proporcionado por *onNotification* en el cual es el evento que se disparará en el momento en que se reciba una notificación nueva en el sistema si la aplicación está abierta, o si el usuario ha hecho clic sobre alguna de las notificaciones mientras la aplicación estaba en segundo plano o cerrada. En el parámetro *data* que llega junto con la notificación proporciona la información que llega junto con la notificación, entre ellas llega la información que hemos asociado a la notificación (los 4 campos que se han indicado) así como datos sobre la notificación (título de la notificación, cuerpo de la misma o si ha sido pulsada o no).

En cuanto al servidor, cuando el mismo detecta una situación en la cual se supera alguna de los umbrales definidos por dicho usuario para dicha camiseta se comprueba que dicho usuario no tenga las notificaciones apagadas (tanto para dicha camiseta como para el nivel global). Si las mismas no se encuentran apagadas entonces se procede a obtener el *token* del usuario asociado a dicha camiseta. En dicho momento se construyen los parámetros que se van a enviar junto con la notificación y se envía una petición POST a la siguiente URL.

<https://fcm.googleapis.com/fcm/send>

Dicha petición POST debe incluir los siguientes campos (entre otros que no son necesarios para nuestros propósitos):

- **To:** Indica los destinatarios del mensaje. Es decir, el listado de *tokens* a los que debe llegar el mensaje. En este caso, las camisetas se van a tramitar una a una por lo que solamente puede ser uno el destinatario de la notificación. Si este campo no se rellena, el mensaje llegaría a todos los usuarios de la aplicación.

- **Notification:** Indica la información que forma la notificación, así como las acciones asociadas a la misma. No confundir con la información adicional que se desea enviar a la aplicación para que haga el trabajo necesario. En este campo se pasará una array indicando 3 campos. El título de la notificación, el cuerpo de la notificación y el evento relacionado a cuando se haga clic.
- **Data:** Indica la información adicional que se le pasa a la notificación. Aquí se le pasará un array con la información que necesita la aplicación para tramitar la notificación.

Por tanto, lo que hace el servidor web al detectar que debe enviarse una nueva notificación por que se ha superado un umbral definido por el usuario para una camiseta son los siguientes pasos:

- Obtener el *token* del usuario
- Generar la estructura de la notificación, utilizando una array, indicando el título de la notificación, el cuerpo de la misma y la acción relaciona con la misma.
- Generar la información adicional que se va a enviar junto con la notificación, utilizando una array.

Los dos últimos pasos anteriores se pueden encontrar implementados en el fichero *notificaciones.php* de la carpeta utilidades, ya que son funciones auxiliares que utilizará el modulo principal de inserción de los datos para notificar al cliente que se han recibido datos que superan los umbrales ya definidos.

Para acabar este apartado hay que hablar de las notificaciones de la batería y las caídas, ya que todo lo que hemos estado hablando hasta ahora es de que se rebase el umbral de las constantes vitales.

En cuanto a la batería, la detección de la misma es muy simple. Cuando el valor de la batería descienda por debajo de un 20% de la carga se lanzará una notificación siguiendo los mismos mecanismos indicados anteriormente. Para ello buscará los *tokens* de las personas que tienen asociada dicha camiseta y si no tienen deshabilitadas las notificaciones de la batería o todas las notificaciones del sistema a nivel global se procederá a enviarles una notificación indicándoles que el nivel de batería es bajo. La acción de esta notificación es distinta a las demás ya que la misma redirige al usuario a la página de listados de camiseta y no al de constantes vitales como hacen todos los demás.

En nuestro generador de datos, dado que decrementa cada vez que se ejecuta un 1% el nivel de batería, las notificaciones saltarán en el momento que el valor disminuya del 20% y dará un nuevo aviso, hasta que el nivel de carga vuelva a incrementar nuevamente al 100%.

En cuanto a la detección de las caídas. El dato es enviado mediante un flag (0 o 1) por la camiseta al servidor. Si el servidor detecta que este flag vale 1, procederá a obtener el *token* de los usuarios que tengan dicha camiseta asociada y si no tienen desactivadas las notificaciones de caída o las notificaciones a nivel global se procederá a enviarle un mensaje comunicando la caída del usuario.

En nuestro generador de datos, simplemente activamos este flag el 10% de las ocasiones. Si se ha activado se procederá al envío de la notificación por la caída.

## 3.6 Sprint 6

Durante todo este sprint, se ha dedicado principalmente a la documentación de los procesos realizados, así como a la corrección de los errores encontrados por los usuarios. No obstante, el desarrollo de la aplicación ha seguido produciéndose y en esta ocasión se les ha preguntado a los usuarios que echan de menos en la aplicación o que aspectos de la misma les gustaría que tuviera. En este *sprint* se ha procedido a corregir dos de estos puntos.

### ***Datos adicionales en la camiseta***

Los *Early Adopters* que principalmente necesitan la aplicación para su profesión y necesitan tener monitorizados a los usuarios que tienen las camisetas puestas, necesitan añadir más información adicional que les permita gestionar mejor a la gente que esta a su cargo. Por tanto, una de las mejoras que se solicitan es justamente el agregar más campos al formulario de creación y edición de camisetas, pero sin que esto implique rellenar más información adicional, ya que como se ha mencionado a lo largo del documento en repetidas ocasiones los usuarios no quieren rellenar demasiados datos para poder utilizar la aplicación.

### ***Entrevistas***

Aquí todos los *Early Adopters* estuvieron de acuerdo que se podría añadir más campos para tener luego más información disponible en caso de que necesiten consultarla. Por tanto, durante las entrevistas se indicaron que serie de campos adicionales son los que los usuarios necesitan gestionar para el uso de la aplicación.

Los *Early Adopters* que simplemente requieren el uso de la aplicación para el cuidado de sus padres o abuelos, no necesitan realmente introducir ningún dato adicional. Realmente sólo necesitarían dos datos que verían necesario en caso de que tuvieran que indicárselo posteriormente a un médico. El primero de los datos es la edad de la persona. Muchas veces se duda de la edad que tiene realmente la persona. Se sabe más o menos que edad tiene, pero no exactamente cuantos años tiene. El segundo de los datos es un campo de la medicación que toma o las necesidades que tiene el usuario. En ocasiones es necesario añadir información extra relacionada con el tratamiento del cliente o necesidades, para luego comprobar que la misma se esta respetando o se necesita recordar al usuario. Por ello se debería introducir campos que permitan añadir esta información extra para luego pueda ser consultada.

En cambio, los *Early Adopters* que necesitan la aplicación para su profesión necesitan añadir bastante más información extra, como son los teléfonos de contacto, la dirección del usuario, la edad del mismo, el sexo y datos relacionados con la medicación o tratamiento del mismo.

Dado que no todos los usuarios requieren toda esta información se acuerda que:

- Sexo
- Fecha de nacimiento (para calcular la edad)

Sean campos que aparezcan siempre y a continuación del parentesco, ya que dichos datos son datos que todos los usuarios pueden que acaben rellenando al ser común para todos los usuarios.

En cambio, para los usuarios que están a cargo o al cuidado de gente mayor se necesitan datos adicionales a rellenar. Se acuerda que dado que estos datos no son realmente importantes para todos los usuarios los mismos aparezcan únicamente tras pulsar un botón para que aparezcan los nuevos campos a continuación de los datos de los umbrales. Los datos que se acuerdan que aparezcan son los siguientes:

- Telefono (El del propio usuario de la camiseta)
- Telefono de Contacto (El del familiar a cargo)
- Dirección
- Notas u observaciones

### Prototipos

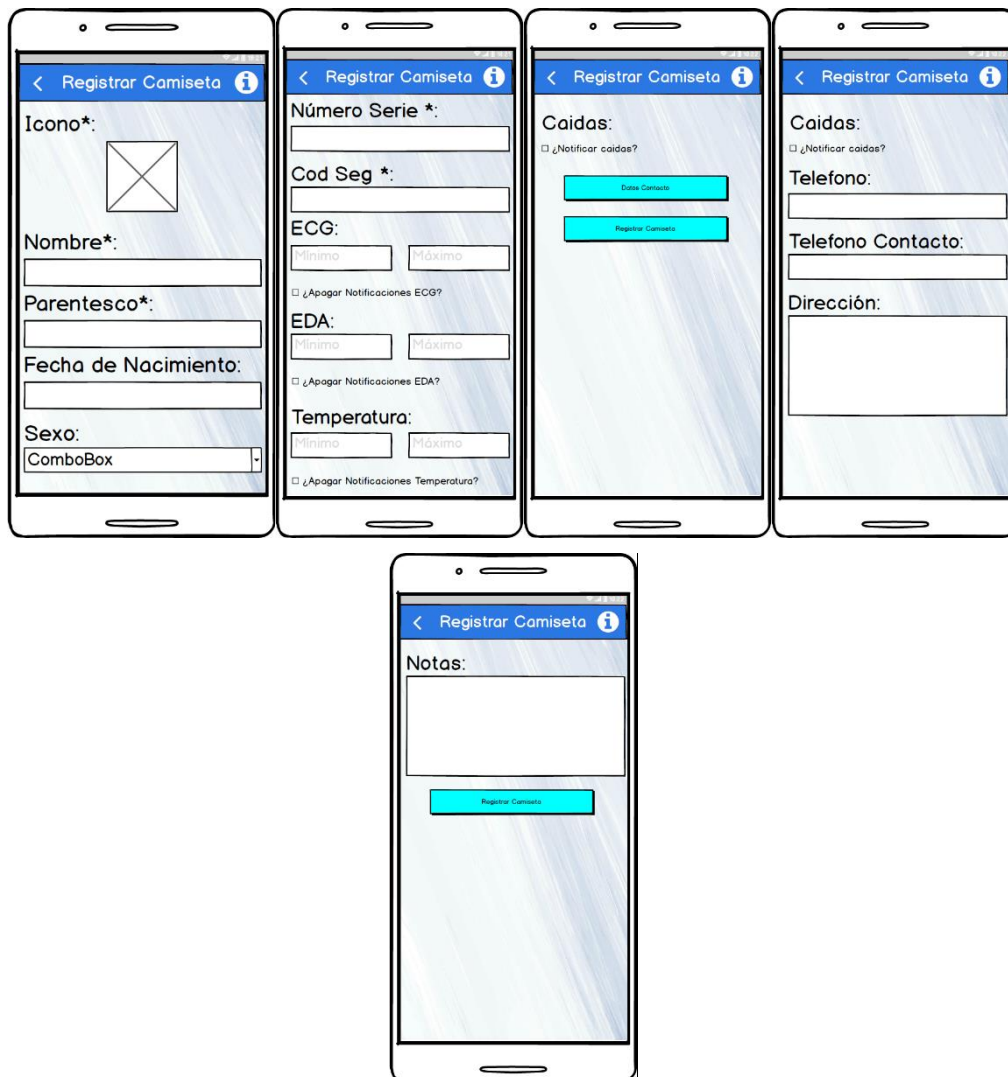


Figura 24 - Prototipos nuevos campos

## Pantallas finales realizadas

The figure shows three screenshots of a mobile application interface titled 'Editar Camiseta'. The interface is designed for editing a 'Camiseta' (shirt) record. It features a blue header with a back arrow and a title. The main content area is divided into several sections: a profile section with a placeholder icon and fields for 'Nombre \*', 'Camiseta', 'Parentesco \*', 'Fecha de nacimiento', 'Sexo', 'ECG', 'EDA', and checkboxes for '¿Apagar Notificaciones ECG?' and '¿Apagar Notificaciones Temperatura?'; a 'Datos Contacto' section with fields for 'Teléfono', 'Teléfono Contacto', 'Dirección', and 'Notas'; and a 'Guardar Cambios' button. The interface is clean and modern, with a light blue background and white text.

Figura 25 - Pantallas Nuevos campos implementadas

## Notaciones sobre la implementación

La implementación de los nuevos campos no requiere prácticamente ninguna explicación adicional a la que se dio durante el *sprint 3* en el apartado de creación y edición de las camisetas. Simplemente se han agregado 6 campos nuevos y en los métodos del *provider rest-camiseta* se han añadido como parámetros esos 6 campos nuevos. Lo único importante a mencionar durante este apartado es la opción de desplegar los 4 campos relacionados con el contacto en el momento que el usuario pulse el botón de *Datos Contacto*.

Inicialmente el *div* que contiene estos 4 campos se asocia a la propiedad del atributo en Angular *escondeme*.

```
<div *ngIf="escondeme">
```

Si la misma es *true*, se visualiza el valor de los campos y si es *false* el valor se esconde. Cuando se pulsa el botón de *Datos Contacto* se invoca un método que lo único que hace es poner esta propiedad a un valor de cierto y por tanto se visualizan los campos. Esta propiedad también se utiliza con el botón, utilizando el atributo *hidden*, en el cual si es cierto se esconde el mismo.

```
[hidden]="escondeme"
```

## Valor mínimo y máximo de las constantes vitales

En este apartado se va a analizar la necesidad de mostrar información adicional en la pantalla de visualización de constantes vitales, ya que el mostrar únicamente la media del valor de las constantes vitales es para algunos casos insuficiente.

## Entrevistas

Los *Early Adopters* durante la última semana han reportado que necesitan conocer además de la información media de la constante durante el periodo de tiempo analizado, el valor más alto y mínimo obtenido. Estos valores deben mostrarse junto con la media, todos en una línea y empezando en primer lugar por el valor mínimo, a continuación, la media y luego el valor máximo del mismo.

## Prototipos



Figura 26 - Prototipos Visualizaciones adicionales

## Pantallas finales realizadas



Figura 27 - Pantallas implementadas Visualizaciones adicionales

### ***Notaciones sobre la implementación***

En este punto los únicos cambios que se han tenido que realizar sobre la implementación es añadir 2 variables más para cada una de las constantes vitales. El calculo del mínimo y máximo para el EDA y la temperatura es sencillo por que simplemente se necesita ver cual es el valor más alto y más bajo.

Para el calculo del ECG es un poco más complicado ya que este valor debe hacerse por minuto y ver cuantos picos hay en 1 minuto de datos y luego comprobar si es el mínimo o el máximo del grupo.

## Capítulo 4: Backend

La mayoría de las implementaciones que se han visto durante el capítulo anterior están relacionadas con la aplicación móvil realizada o más la parte *Frontend* del desarrollo realizado. Es decir, se han estado viendo cómo se han realizado gran parte del contenido visual que visualizará e interactuará el usuario con nuestra aplicación y sólo se ha remitido al *backend* específicamente cuando se ha hablado de como mostrar la visualización de las constantes vitales, ya que en el mismo se han generado los valores de las constantes vitales a modo de prueba. También nos hemos referido al *backend* durante la realización de las notificaciones, cuando el servidor se ha encargado de comunicar con el servicio de *Firebase Cloud Messaging* las pruebas del mismo

En el presente capítulo se va a abordar las comunicaciones que se realizan desde la aplicación móvil desarrollada al *backend*. En este caso el *backend* está organizado como un servicio *Restful* en el cual las peticiones se realizan utilizando URLs de acuerdo a un formato en específico que indica la información que se desea tramitar o realizar. Además de utilizar las URLs en las mismas se especifica el tipo de petición que se desea realizar la cual puede ser principalmente los métodos GET, POST, PUT y DELETE, para realizar una acción muy específica sobre la acción.

### 1. Características del Servidor Backend

- Servidor PHP versión 7.3.
- Servidor Base de Datos MySQL 5.7.28.
- Hosting en 1&1 (actualmente IONOS).

### 2. Características App móvil

Todas las acciones que requiera la aplicación móvil realizar con el servidor serán realizadas a través de un *provider* o un servicio que se encargue de ejecutar dichas peticiones. Las peticiones son todas asíncronas y funcionan como “promesas”, donde se compromete a devolver la respuesta de la petición como máximo al cabo de un periodo de tiempo que en este caso se ha definido en 10 segundos.

En total se han configurado 4 *providers* distintos en toda la aplicación que son los siguientes:

- **Rest:** Se encarga de realizar todas las peticiones relacionadas con el usuario en relación a la autenticación del mismo y a los datos relacionados con cualquier *token* (como el de notificación) que se deba generar o transmitir.
- **Rest-camiseta:** Se encarga de realizar las peticiones relacionadas con el listado de las camisetas, así como la obtención y actualización de los datos de las mismas.
- **Rest-configuracion:** Se encarga de realizar las peticiones de obtención y actualización de las opciones de configuración del usuario. En este caso se trata de obtener el listado de notificaciones encendidas o apagadas a nivel global como cualquier actualización que se haga sobre las mismas.
- **Rest-constantes:** Se encarga de obtener en tiempo real las constantes de una camiseta, así como realizar las búsquedas temporales de las mismas.



Todos estos *providers* tienen definidas la URL base a la que tienen que realizar estas peticiones y luego según la operación a realizar ya especifican en la parte final de la URL exactamente que recurso quieren obtener o actualizar.

## 2. Características Servidor Web

Tal y como se ha estado describiendo a lo largo de todo este capítulo, en el servidor existirá un servicio *Restful* que estará conformado por muchos scripts organizados en distintas carpetas. Para simplificar la implementación del servicio *Restful* la estructura de carpetas formará la URL a la que se tenga que acceder y posteriormente existirá un único fichero dentro de esta carpeta que será un script escrito en *php* llamado *index.php* que contendrá todas las posibles acciones que se pueden realizar sobre dicha ruta.

Además, para evitar la compatibilidad entre versiones de la aplicación en las que en una versión se utilice un método que se esté esperando un resultado y en otra versión se esté esperando un resultado distinto (por ejemplo, que una devuelva un objeto en formato *JSON* y en otra devuelva una simple array) las urls de los servicios *Restful* irán precedidos por el número de versión de la misma. Así se consigue que si un usuario utiliza la versión antigua de la aplicación pueda seguir utilizándola sin peligro de que una nueva versión fuerce a que deje de ir la versión antigua.

En el anexo D se encuentra un fichero *JSON*, en el que se puede importar desde la aplicación Postman, la cual es una colección de pruebas que se han hecho para testear el funcionamiento del servicio *Restful*. En el mismo se puede utilizar para validar que el comportamiento del servicio *Restful* es correcto y el recurso es obtenido, creado, actualizado o borrado.

El servicio *Restful* construido se estructura principalmente en 5 partes

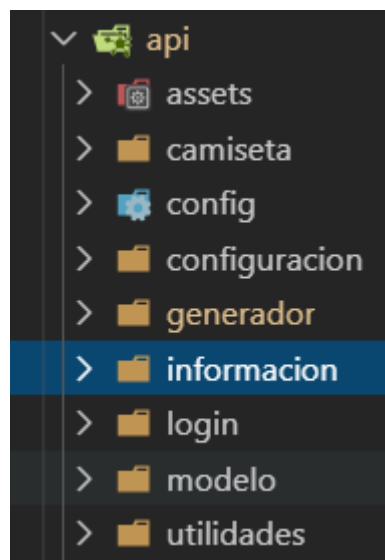


Figura 28 - Estructura del servicio Restful

- **Assets:** Esta carpeta contiene las imágenes y los ficheros CSS que se utilizan tanto en el contenido que se visualiza desde el *backend* como el que se hace referencia desde la aplicación móvil.
- **Config:** Esta carpeta contiene la configuración de acceso a la base de datos.
- **Utilidades:** Esta carpeta contiene funciones auxiliares que son utilizadas por el servicio *Restful*.
- **Modelo:** Esta carpeta contiene los modelos de las tablas creadas en base de datos, así como las funciones encargadas de realizar operaciones sobre estas tablas.
- **Otras Carpetas:** El resto de las carpetas son las carpetas que serán las que contendrán los scripts a los que se les hará las peticiones *Restful*. Son los ficheros de entrada a cada una de las peticiones que se van a realizar.

En total hay rutas (y luego con sus posibles subrutas) a las que se puede acceder:

- **Login:** Se encarga de realizar todas las operaciones relacionadas con el registro y la autenticación de los usuarios en el sistema.
- **Configuración:** Se encarga de realizar las operaciones relacionadas con la actualización de las opciones de configuración
- **Información:** Se encarga de las operaciones de insertar o consultar los valores de las constantes vitales.
- **Camiseta:** Se encarga de realizar las operaciones relacionadas con la creación, actualización, borrado y listado de las camisetas asociadas a los usuarios.

Dado que escribir todo el código que se maneje los distintos métodos hace que sea difícilmente legible el código y dificulta la corrección de la misma, se ha procedido a dividir en más subcarpetas las peticiones, haciendo que en cada fichero *PHP* se tramite un único tipo de petición. Por ejemplo, en las camisetas existen 4 subcarpetas que son *borrar*, *crear*, *editar* y *lista* donde cada una hace referencia a cada una de las 4 posibles operaciones.

En cuanto a la implementación del código fuente *php* la estructura de todos los ficheros PHP es muy similar y pasa a ser explícita a continuación.

Cualquier fichero que sea una creación, edición o borrado de datos necesitará que se le suministre una cadena en formato *JSON* la relación de campos que deben sufrir la actualización de dichos datos. Para que un script de *PHP* pueda leer dicha cadena debe utilizarse la función *file\_get\_contents* sobre el campo de entrada de *PHP*.

```
file_get_contents('php://input');
```

A continuación, lo siguiente es comprobar si existen el mínimo número de datos necesarios para poder realizar la operación. En caso de que no sea así se contesta con un código de error 400.

Si todos los parámetros que hacen falta están disponibles, entonces se inicializa la conexión con la base de datos y se inicializa el objeto en el que se van a realizar las operaciones. A continuación, se realiza sobre el propio objeto creado las operaciones relacionadas con la petición que se está tramitando, realizando las comprobaciones necesarias para ello.

Si en cambio la petición es de obtención de datos, se utiliza la operación GET sobre el fichero, por lo que en vez de necesitarse un objeto *JSON* simplemente se le pasaran los parámetros vía URL, por lo que por ejemplo una correcta petición sería como sigue a continuación

*<http://www.jmcastellano.eu/healthshirt/api/v1/informacion/?numeroserie=1111111111111111>*

Donde en este caso la petición está solicitando un dato de información de la camiseta con número de serie 1111111111111111.

La base de datos que se utiliza es una base de datos en MySQL. Para la conexión con la misma se utiliza PDO [32] para la que se puede dejar preparadas sentencias en SQL para posteriormente indicar fácilmente los parámetros asociadas a la misma y ejecutar la sentencia de la misma abstrayéndonos de toda la implementación relacionada con el conexionado con la base de datos. La configuración y los métodos para el acceso a la base de datos se encuentra en la carpeta *config* del servidor.

Por último, cualquier petición que se realice al servidor puede devolver un código de respuesta o junto con el código de respuesta una respuesta en formato *JSON*. Los códigos de respuesta que puede devolver el servidor son las siguientes (se suministra un ejemplo de cada código de respuesta):

- **200:** Operación realizada correctamente. Un ejemplo sería cuando se ha obtenido correctamente el listado de las camisetas de un usuario.
- **201:** El registro se ha creado. Un ejemplo sería cuando se ha creado una nueva camiseta.
- **400:** A la petición le faltan parámetros. Un ejemplo sería cuando no existe la cadena *JSON* asociado a los datos de la nueva camiseta.
- **403:** Acceso no autorizado al recurso. Un ejemplo sería un usuario intentando actualizar los datos de una camiseta que no es suya.
- **404:** No encontrado. Un ejemplo sería actualizar los datos de una camiseta inexistente.
- **409:** Ha habido un conflicto a la hora de tramitar la petición. Un ejemplo sería un usuario que intenta crear otra camiseta idéntica a una que ya tiene asociada.
- **500:** Error del servidor. Un ejemplo sería que no ha podido realizarse la conexión con la base de datos.

## Capítulo 5: Pruebas de Test

En el presente capítulo se va a explicar que procesos se han seguido para la realización de las pruebas o procesos de test. Como se ha indicado a lo largo de todo el documento se ha estado recibiendo los consejos y las opiniones de todos los *Early Adopters* tanto en el apartado de cuáles son las funcionalidades que esperan los usuarios que realice la aplicación para solucionar su problema, como el contenido y la distribución de los elementos que conforman las distintas pantallas de la aplicación.

En esta ocasión se ha solicitado a los *Early Adopters*, al final de cada *sprint*, que probaran la aplicación, no solamente para detectar si había errores en la aplicación o no, sino también para detectar si la funcionalidad implementada es correcta de acuerdo a las especificaciones que se han ido indicando.

En este capítulo se van a estudiar dos puntos. En el primero de ellos se va a indicar como se ha procedido a realizar las pruebas con los *Early Adopters*, los periodos y los métodos de distribución de la aplicación para que estos usuarios pudieran probar la aplicación. En el segundo de los puntos se va a indicar que mecanismos se han utilizado para la obtención de los resultados de las pruebas, así como la visualización de los posibles errores que se han podido dar durante la ejecución.

### 1. Periodo y modo de distribución

Como se ha comentado durante el apartado de planificación, una de las tareas que se realizan durante cada uno de los *sprints* es un periodo de pruebas en el cual los *Early Adopters* pueden probar la aplicación y proceder a indicar cuales son los errores detectados, que funcionalidades se encuentran de forma incompleta o son correctas de acuerdo a las especificaciones habladas o posibles mejoras que se pueden realizar sobre la aplicación. Dichas pruebas han tenido lugar durante los últimos 3 días de cada *sprint* correspondiente al fin de semana de la semana 2, que aparte de tener más tiempo disponible para la tarea de implementación era el momento de la semana más viable para los *Early Adopters* para la realización de las pruebas.

Los 4 *Early Adopters* con los que se ha hablado poseen dispositivos móviles con sistema operativo Android, por lo que todas las pruebas que pueden realizarse con estos usuarios directamente tienen que ser bajo terminales de este sistema operativo. Por tanto, se debe distribuir la aplicación en su versión Android para las pruebas. Esto obviamente no sería suficiente para verificar su funcionamiento en IOS, se debería hacer pruebas adicionales en su versión de IOS, pero no se dispone de cuenta de desarrollador para IOS, ni usuario con dispositivo compatible (Iphone).

En primer lugar, la tienda de aplicaciones de Google (la Play Store [9]) permite a los desarrolladores subir una aplicación en distintas versiones, pero por defecto existen 3 tipos, versiones Alfa, versiones Beta y versiones en producción. Dentro de las versiones alfa y beta

se pueden definir listas de usuarios que pueden acceder a la aplicación y luego enviar a los correos de dichos usuarios.

Se puede subir una aplicación como una versión alfa, tal y como se muestra en la figura 29, y dejar que la prueben un número muy pequeño de usuarios. Si el número de errores es pequeño o son asumibles, se puede proceder a publicar fácilmente la versión a producción para que este disponible para todos los usuarios como si de cualquier aplicación publicada en la tienda se tratara.



Figura 29 - Versión Alfa 0.0.5 de Healthshirt

Para estas versiones alfa se tiene que configurar el listado de usuarios que van a poder acceder a la aplicación, indicando usuarios que tengan cuentas de google tal y como se indica en la figura 30. Solamente estos usuarios tendrán acceso a la aplicación y podrán descargarse la aplicación. Para ello los usuarios tendrán que acceder a la aplicación mediante la URL indicada en la figura 30 desde el navegador de su móvil (o clicando el enlace en el email que se les envía cuando se les da de alta en la lista de testers).

Utilizando la tienda de aplicaciones en vez de utilizar otros canales de distribución distintos nos da una serie de ventajas que son las siguientes:

- Instalación como si fuera cualquier otra aplicación de la tienda. Lo que permite no tener que depender de explicar a los usuarios como instalar la aplicación y como permitir la instalación de aplicaciones ajenas a la tienda [33].
- Actualización automática cuando hay una nueva versión.
- Restringir quien tiene acceso a la aplicación o no.
- Recopilación de estadísticas.

**Administrar testers**
Pruebas cerradas para determinadas listas de testers

Elige como quieres ejecutar el programa de testing. [Más información](#)

Elige un método de testing
Pruebas cerradas para determinadas listas de testers

Usuarios	Nombre de la lista	Usuarios
<input checked="" type="checkbox"/>	Lista Test	6 <a href="#">EDITAR</a>

[CREAR LISTA](#)

Canal de sugerencias
jmcastellano@jmcastellano.eu

URL de invitación
<https://play.google.com/apps/testing/eu.jmcastellano.healthshirt>

[DESACTIVAR CANAL](#)
[GUARDADO](#)

Figura 30 - Administración de Testers

## 2. Obtención de datos y resultados

Uno de los problemas que se puede observar con la utilización de las aplicaciones móviles, es que no se dispone acceso directamente a los terminales o dispositivos móviles en los que se esta ejecutando la aplicación desarrollada. Los usuarios que realizan las pruebas no tienen conocimientos técnicos de lo que pasa en la aplicación y por tanto simplemente nos van a informar de que existe un error o que el comportamiento del mismo no es correcto sin proporcionarnos información técnica que indique que error se ha producido o como solucionar el mismo.

Si el error se produce por la interacción de la aplicación móvil con en el servidor si se puede obtener información relacionada con el error, ya que dicha información se encuentra almacenada en el servidor. Para ello simplemente en caso de que se produzca un error se procede a escribirlo en un fichero indicando los datos de la operación. Para ello se utiliza el método `file_put_contents` en el cual podemos poner el contenido que le digamos dentro de un fichero. Para evitar tener un fichero log excesivamente grande, se procederá a simplemente a generar un fichero diario con todas las trazas del día.

En la figura 31 se muestra un ejemplo de los logs que se generan en el servidor que sirven para depurar cualquier problema que exista relacionado con el servidor.

```

DEBUG-URLPetición: 85.52.202.180 - December 28, 2019, 1:43 pm- Registrado correctamente jmcastellano@jmcastellano.eu
DEBUG-URLPetición: 85.52.202.180 - December 28, 2019, 1:43 pm- Recuperando camisetas de jmcastellano@jmcastellano.eu
DEBUG-URLPetición: 85.52.202.180 - December 28, 2019, 1:43 pm- Camisetas de jmcastellano@jmcastellano.eu [{"id": "25", "nombre": "Mama", "parentesco": "Mama", "bateria": "58", "horadatos": "2019-12-28T13:43:00"}]
DEBUG-URLPetición: 85.52.202.180 - December 28, 2019, 1:43 pm- Petición datos actuales camiseta 11111111111111111111
DEBUG-URLPetición: 85.52.202.180 - December 28, 2019, 1:43 pm- Constantes de 1111111111111111 [{"ecg": "204,202,296,195,633,9,200,197,402,202,200,240,199,202,202,205,197,197,199,201,200"}]
DEBUG-URLPetición: 85.52.202.180 - December 28, 2019, 1:44 pm- Petición datos historicos camiseta 11111111111111111111
DEBUG-URLPetición: 85.52.202.180 - December 28, 2019, 1:44 pm- Constantes de 1111111111111111 [{"ecg": "201,199,297,197,634,13,204,198,390,199,198,247,196,204,201,201,199,197,203,200,200"}]
DEBUG-URLPetición: 85.52.202.180 - December 28, 2019, 1:44 pm- Petición datos historicos camiseta 11111111111111111111
DEBUG-URLPetición: 85.52.202.180 - December 28, 2019, 1:44 pm- Constantes de 1111111111111111 [{"ecg": "201,199,297,197,634,13,204,198,390,199,198,247,196,204,201,201,199,197,203,200,200"}]
DEBUG-URLPetición: 85.52.202.180 - December 28, 2019, 1:44 pm- Petición datos historicos camiseta 11111111111111111111
DEBUG-URLPetición: 85.52.202.180 - December 28, 2019, 1:44 pm- Constantes de 1111111111111111 [{"ecg": "201,199,297,197,634,13,204,198,390,199,198,247,196,204,201,201,199,197,203,200,200"}]
DEBUG-URLPetición: 85.52.202.180 - December 28, 2019, 1:45 pm- Recuperando camisetas de jmcastellano@jmcastellano.eu
DEBUG-URLPetición: 85.52.202.180 - December 28, 2019, 1:45 pm- Camisetas de jmcastellano@jmcastellano.eu [{"id": "25", "nombre": "Mama", "parentesco": "Mama", "bateria": "56", "horadatos": "2019-12-28T13:45:00"}]
DEBUG-URLPetición: 85.52.202.180 - December 28, 2019, 1:45 pm- Crear camisetas datos para jmcastellano@jmcastellano.eu
ERROR-URLPetición: 85.52.202.180 - December 28, 2019, 1:45 pm- El número de serie no existe o no es válido
DEBUG-URLPetición: 85.52.202.180 - December 28, 2019, 1:46 pm- Crear camisetas datos para jmcastellano@jmcastellano.eu

```

Figura 31 - Ejemplo logs de depuración

Como se puede ver en el ejemplo facilitado, se puede ver como el usuario [jmcastellano@jmcastellano.eu](mailto:jmcastellano@jmcastellano.eu) se ha registrado correctamente en el sistema y posteriormente se recuperan las camisetas del mismo, obteniéndose una camiseta llamada Mama. A continuación, el usuario accede a la pantalla de las constantes vitales, donde primero se solicitan las constantes vitales actuales y luego se hacen 4 peticiones de datos históricos. Luego el usuario vuelve a la pantalla del listado de listado de las camisetas y a continuación intenta crear una nueva camiseta donde el número de serie no existe o no es válido.

Esta implementación soluciona los problemas relacionados con las interacciones con el servidor, pero no soluciona cualquier error que se pueda producir en la aplicación móvil. Para solucionar este inconveniente se utilizará el servicio *Crashlytics de Firebase* [15]. Lo normal para utilizar esta funcionalidad es utilizar unos SDKs que proporciona la propia herramienta, pero dado que se está trabajando en *IONIC* no se pueden utilizar estos SDKs directamente. Por ello se utiliza el plugin en [34]. El mismo se encargará de abstraernos de la implementación para enviar los errores al servicio de *Crashlytics de Firebase* con lo que simplemente invocando el método *logException* se podrá enviar una excepción al sistema. Además, este *plugin* proporciona otra funcionalidad. Si la aplicación da un error que provoca la terminación de la misma, el plugin enviará un evento de bloqueo a *Firebase* y podremos visualizar la información relacionada con la misma. En la la figura 32 se puede visualizar un ejemplo de los errores que se han recibido en *Firebase* y en la figura 33 un ejemplo en caso de que se produzca un error que provoque la terminación prematura de la aplicación.

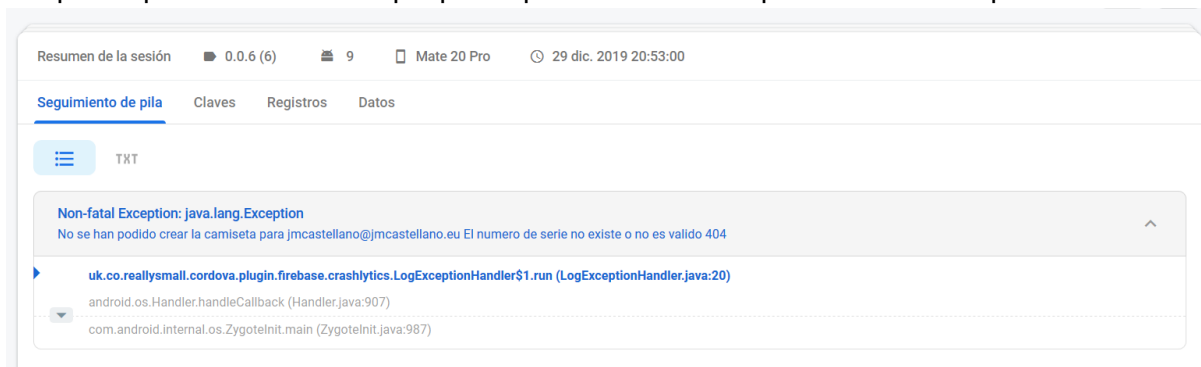


Figura 32 - Ejemplo error en Crashlytics

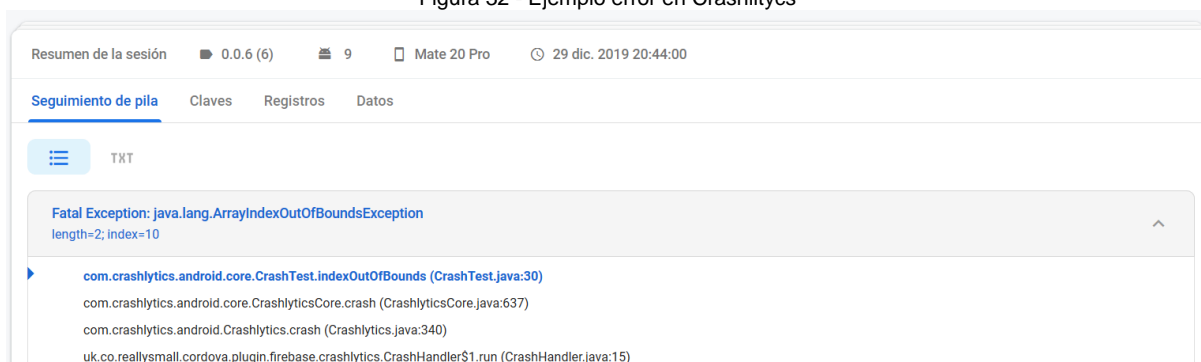


Figura 33 - Ejemplo error de bloqueo en Crashlytics

Con estos dos sistemas se puede monitorizar toda la actividad que hacen nuestros usuarios en nuestra aplicación y observar si se producen errores en nuestra aplicación y poder así detectarlos y corregirlos sin necesidad de tener acceso a los propios terminales de los usuarios ni que estos nos tengan que reportar los errores.

## Capítulo 6: Pruebas Realizadas

En este capítulo se detallará brevemente que pruebas y como se han realizado para probar y verificar que la aplicación funciona correctamente y de acuerdo a las especificaciones que se han ido detallando con los *Early Adopters*.

En primer lugar, hay que indicar que en el momento de escribir estas líneas existen 6 cuentas creadas en la aplicación (1 por cada *Early Adopter*, la cuenta del autor y una cuenta creada para la realización de las pruebas). Los datos para acceder a la cuenta creada para la realización de las pruebas es la siguiente:

Usuario: [healthshirt@jmcastellano.eu](mailto:healthshirt@jmcastellano.eu)

Password: Healthshirt2019&

Aunque se facilite la cuenta para pruebas, se puede crear una prueba desde cero para la realización de las pruebas de registro y autenticación de las pruebas, de hecho las primeras pruebas realizadas en la aplicación van relacionadas con la creación y autenticación de las cuentas de la aplicación. La explicación de los pasos seguidos para poder utilizar la aplicación se encuentra en el manual de usuario que se suministra junto con la entrega. Se detalla a continuación el conjunto de pruebas que se han realizado sobre la parte de autenticación del sistema:

Prueba realizada	Parametros	Resultado	¿Correcto?
Acceder a la pantalla de que va la aplicación	-----	Se visualiza la pantalla de que va la aplicación	Sí
Acceder a la pantalla de registro	-----	Se visualiza la pantalla de registro de cuenta nueva	Sí
Registrar cuenta sin datos	Email: No rellenado Password: No rellenado Confirmar Password: No rellenado	No se habilita el botón de registrar	Sí
Registrar cuenta email no válido	Email: estoesunemail Password:Manolete1985. Confirmar Password: Manolete1985.	No se habilita el botón de registrar.	Sí
Registrar cuenta password no coincide	Email:prueba@prueba.es Password:Manolete1985. Confirmar Password: Manolete1986.	No se habilita el botón de registrar.	Sí
Registrar cuenta password	Email: <a href="mailto:prueba@prueba.es">prueba@prueba.es</a> Password:Manolete1985 Confirmar Password: Manolete1985	No se habilita el botón de registrar.	Sí



formato correcto	no			
Registrar cuenta existente	ya	Email:jmcastellano@jmcastellano.eu Password: Manolete1985. Confirmar Password: Manolete1985.	Se indica que ya existe una cuenta	Sí
Registrar cuenta existente	no	Email:prueba@prueba.es Password: Manolete1985. Confirmar Password: Manolete1985.	La cuenta se registra y se envia un email pidiendo confirmación	Sí
Iniciar Sesión Facebook sin cuenta previa		-----	Se solicitan las credenciales de Facebook via la APP de Facebook y posteriormente se solicita la contraseña	Sí
Iniciar Sesión Facebook con cuenta previa		-----	Se solicitan las credenciales de Facebook via la APP de Facebook y posteriormente pasa al listado de camisetas.	Sí
Acceder a la pantalla de Login		-----	Se accede a la pantalla de login	Sí
Login cuenta inexistente		Usuario: Pruebanoexiste Password: Manolete1985.	Se indica que los datos de autenticación no son correctos	Sí
Login cuenta password incorrecto		Usuario: <a href="mailto:healthshirt@jmcastellano.eu">healthshirt@jmcastellano.eu</a> Password: Healthshirt2020&	Se indica que los datos de autenticación no son correctos.	Sí
Login sin parametros		-----	No se habilita el botón de Login	Sí
Login correcto		Usuario: <a href="mailto:healthshirt@jmcastellano.eu">healthshirt@jmcastellano.eu</a>	Se muestra la página de listado de camisetas	Sí
Acceder a la pantalla de Olvido		-----	Se muestra la página de Olvido de contraseña	Sí
Olvido de Contraseña		Usuario: <a href="mailto:healthshirt@jmcastellano.eu">healthshirt@jmcastellano.eu</a>	Se envia un correo con un	Sí

		token para recuperar la cuenta	
--	--	--------------------------------	--

Tabla 3 - Pruebas autenticación

Para probar las camisetas se han dado de alta dos camisetas en el sistema, ya que como se ha mencionado en el *sprint* 3 en la creación y edición de las camisetas, las camisetas necesitan suministrarse un número de serie y un código de seguridad que tiene que haber definido previamente el usuario. Dado que el producto en sí todavía no existe (por que esta pendiente de fabricarse) se han creado dos camisetas virtuales. Los datos de las camisetas son las siguientes:

Camiseta 1 (Es la que camiseta que actualmente se generan datos)

Número de Serie: 1111111111111111

Código Seguridad: Healthshirt2019

Número de Serie: 2222222222222222

Código Seguridad: Healthshirt2019

Se recomienda para las pruebas utilizar la camiseta 1, ya que la misma es la única camiseta para la que se están generando datos y por tanto las pruebas sobre la visualización de las constantes vitales se han hecho. A continuación, se indican las pruebas realizadas sobre la creación y edición de las camisetas.

Prueba realizada	Parametros	Resultado	¿Correcto?
Listar camisetas	Usuario: <a href="mailto:healthshirt@jmcastellano.eu">healthshirt@jmcastellano.eu</a>	Se lista una camiseta llamada Paco	Sí
Añadir Camiseta falta algún campo obligatorio	No se rellena por ejemplo Número de Serie	No se habilita el botón Registrar Camiseta	Sí
Añadir Camiseta no existe número de serie	Se rellenan todos los campos y en el número de serie se pone 3333333333333333	Se indica que el número de serie o el código de seguridad no son correctos	Sí
Añadir Camiseta Código de Seguridad incorrecto	Se rellenan todos los campos y en el código de seguridad se pone Healthshirt	Se indica que el número de serie o el código de seguridad no son correctos	Sí
Añadir Camiseta ya añadida	Se utilizan como datos: Número de Seguridad: 1111111111111111 Código de Seguridad: Healthshirt2019	Se indica que la camiseta ya se ha añadido previamente	Sí

Añadir Camiseta no añadida previamente	Se utilizan como datos: Número de Seguridad: 2222222222222222 Código de Seguridad: Healthshirt2019	Se vuelve al listado de camisetas y se visualiza la camiseta añadida	Sí
Añadir Camiseta visualizar datos contacto	Pulsar botón datos contacto	Se visualizan los campos teléfono, teléfono contacto, dirección y notas	Sí
Borrado de camiseta	Se pulsa el botón de la papelera roja de la camiseta recién creada	Se borra la camiseta del listado y no se puede recuperar	Sí
Visualizar página de info	-----	Se visualiza la página de información	Sí
Editar camiseta removiendo eliminando algún parámetro obligatorio	Se elimina el valor del campo nombre	El botón de Guardar cambios se deshabilita	Sí
Editar camiseta modificando un campo	Se modifica el valor de cualquier campo pero sin dejarlo vacío	Los cambios se guardan	Sí

Tabla 4 - Pruebas Listado, Añadir y Editar Camisetas

En cuanto a las pruebas realizadas sobre la visualización de las constantes vitales han sido las siguientes:

Prueba realizada	Parametros	Resultado	¿Correcto?
Visualizar datos actuales	Camiseta con número de serie 1111111111111111	Se visualizan los datos de las constantes vitales del último minuto	Sí
Visualizar datos históricos	De: 2019-12-29 22:36 Hasta: 2019-12-29 22:47	Se visualizan los datos. Datos NPM: 87.27 EDA: 200.02 Temperatura: 35.22	Sí

Tabla 5 - Pruebas Constantes

Por último, se realizan pruebas sobre el sistema de notificaciones (hay que tener en cuenta que como los datos generados en el servidor son aleatorios puede ser más fácil o difícil reproducir las siguientes pruebas)

<b>Prueba realizada</b>	<b>Parametros</b>	<b>Resultado</b>	<b>¿Correcto?</b>
Notificación falta de batería	No tener apagadas las notificaciones a nivel global.	Cuando le queda menos de un 20% de batería se avisa mediante una notificación	Sí
Notificación EDA mínimo	Indicar un umbral mínimo de EDA de 200 y no tener las notificaciones EDA a nivel camiseta (1111111111111111) o a nivel global.	Cuando se obtiene un valor inferior de EDA de 200 se recibe una notificación	Sí
Notificación EDA máximo	Indicar un umbral máximo de EDA de 200 y no tener las notificaciones EDA a nivel camiseta (1111111111111111) o a nivel global.	Cuando se obtiene un valor superior de EDA de 200 se recibe una notificación	Sí
Notificación ECG mínimo	Indicar un umbral mínimo de ECG de 90 y no tener las notificaciones ECG a nivel camiseta (1111111111111111) o a nivel global.	Cuando se generan los valores de 60 pulsaciones se recibe una notificación	Sí
Notificación ECG máximo	Indicar un umbral máximo de ECG de 90 y no tener las notificaciones ECG a nivel camiseta (1111111111111111) o a nivel global.	Cuando se generan los valores de 120 pulsaciones se recibe una notificación	Sí

Notificación Temperatura mínimo	Indicar un umbral mínimo de Temperatura de 36 y no tener las notificaciones Temperatura a nivel camiseta (1111111111111111) o a nivel global.	Cuando la temperatura es inferior a 36º se genera una notificación	Sí
Notificación Temperatura máximo	Indicar un umbral máximo de Temperatura de 36 y no tener las notificaciones Temperatura a nivel camiseta (1111111111111111) o a nivel global.	Cuando la temperatura es superior a 36º se genera una notificación	Sí
Notificaciones constante apagadas	Apagar las notificaciones de una constante que estén definidas un umbral	Cuando se rebasa el umbral de la constante no se recibe la notificación	Sí

Tabla 6 - Pruebas Notificaciones

Además, para poder realizar las pruebas contra el servicio *Restful* del servidor se deja disponible el anexo D el cual contiene un conjunto de pruebas realizadas utilizando la aplicación de Postman. En el mismo programa se pueden modificar fácilmente los parámetros o cadena JSON asociada a la misma y se puede realizar la petición. Aunque el servidor actualmente tenga disponible las versiones *v1* y *v2* el fichero de pruebas facilitado en el anexoD se utiliza la *v2* del mismo y se recomienda que para las pruebas que se tengan que realizar se hagan sobre esta misma versión. La versión *v1* de la aplicación es la que utilizan las versiones *0.0.5* o inferior. A partir de la versión *0.0.6* se utiliza ya la versión *v2* del servicio *Restful*.

# Capítulo 7: Conclusiones y líneas de futuro

## 1. Conclusiones

A lo largo del presente documento se ha ido desarrollando una Web App multiplataforma orientada a un producto que debe desarrollarse en los próximos meses. Como se ha visto el desarrollo de la misma continua ya que es una aplicación que no se puede terminar de desarrollar completamente dentro de los meses que hay disponibles para la realización del TFM.

Aunque la aplicación todavía siga en desarrollo, la misma ahora mismo ya es funcional para el principal uso que se le va a dar que es la monitorización de las constantes vitales de los pacientes que lleven las camisetas inteligentes. No obstante, todavía existen muchos puntos que se pueden mejorar en la aplicación y que van a desarrollarse en el apartado de líneas de futuro.

En primer lugar, se ha aprendido que interactuar con solamente 4 *Early Adopters* es realmente un número muy pequeño y más teniendo en cuenta que posteriormente no se han realizado otros métodos de investigación como encuestas o pruebas de laboratorio para observar como se comportan los usuarios ante nuestra aplicación. Generalmente estos procesos se realizan con un número mayor de usuarios y previamente se debe realizar un problema de si el problema a solucionar (en este caso la monitorización de los usuarios) existe o no tal y como se especifica en la metodología LEAN [35].

Además, se ha observado que la realización de una aplicación web únicamente realizado por un solo usuario es realmente difícil, aunque se hayan utilizado metodologías apropiadas para la entrega temprana a cliente. La cantidad de trabajo abarcado es posiblemente sea demasiado excesivo para lo que se planteaba a lo largo de todo el trimestre, pudiéndose quedar cojo en algunos aspectos que se deben ir mejorando en las próximas entregas de la aplicación. Si bien es cierto, que gracias a la utilización de *SCRUM* se ha evitado los problemas de que se entregue la aplicación muy tarde y esto requiera una gran cantidad de cambios. Aunque *SCRUM* se utiliza principalmente en grupos de desarrollo de varias personas se han utilizado algunas de los principios ágiles en los que se basa para realizar una entrega temprana al cliente y realizar ciclos de desarrollo iterativos lo cual ha ayudado a que el proceso de desarrollo fuera más ameno y sencillo.

Los objetivos que se han planteado en el proyecto se recuerdan a continuación. El principal objetivo del proyecto es la realización de una Web App para las plataformas Android e IOS. La misma debe permitir monitorizar las constantes vitales de una serie de usuarios que llevan puesta una camiseta inteligente y que comunican a un servidor remoto vía wifi los datos que se reciben de los sensores y que son recuperadas posteriormente por la aplicación móvil y que son visualizadas por los usuarios. En resumidas cuentas, las funcionalidades que requiere la aplicación inicialmente son:

- Creación y autenticación de una cuenta de usuario.
- Creación, edición y borrado de las camisetas.
- Visualización de los datos de las camisetas.

- Notificación cuando se supera un cierto umbral.

Los 4 objetivos se han cumplido y se tiene una aplicación que es funcional y que en principio los usuarios pueden utilizar para monitorizar las constantes vitales de las personas mayores, no obstante, hay que tener en cuenta que no se ha podido probar con una camiseta de verdad por lo que puede que la generación de constantes vitales que se ha realizado se escape de la realidad y que se tenga que rehacer o modificar la parte relacionada con la visualización de las constantes vitales. Esta parte se solucionará cuando el prototipo de la camiseta inteligente este disponible, que se espera que sea sobre marzo o abril.

Es posible que ciertas partes de la aplicación puedan ser mejoradas ya que algunos *Early Adopters* han indicado que aún se necesitan funcionalidades adicionales por lo que dichos apartados se deben mejorar en la medida de lo posible. De esto se hablará en el apartado de *líneas de futuro*.

La planificación ha sufrido varios percances y retrasos. En primer lugar, no se pudo acabar la parte de la visualización de las constantes vitales durante el *sprint 1*, además se vio gravemente afectado el desarrollo de la parte de la autenticación ya que los usuarios indicaron pantallas adicionales que requerían para hacer las aplicaciones (por ejemplo, una pantalla que indicara de que iba la aplicación y como adquirir la camiseta asociada al mismo). Los *sprint 2* y *3* en general fueron mejor y se pudo acabar el desarrollo del mismo en el tiempo establecido. No obstante, dado que se para mitad de noviembre se vio que el prototipo de la camiseta y que se tenía que finalizar el desarrollo de la pantalla de visualización de las constantes vitales y se tenía que tener información para que los *testers* pudieran verificar el desarrollo del mismo se tuvo que adelantar el desarrollo de la parte de generación de datos un *sprint* y dado que la carga de trabajo era demasiado ya en el *sprint 4* con el desarrollo de las notificaciones se tuvo que dividir la carga de las notificaciones en 2 partes. La primera de ellas la parte que afectaba a la parte visual de la aplicación (añadir o quitar las notificaciones) y la segunda de ellas desarrollada durante el *sprint 5* con la realización del propio envío de las notificaciones desde el servidor a la aplicación móvil utilizando el servicio de *Firebase Cloud Messaging*.

Por tanto, se puede decir que la planificación se ha cumplido parcialmente, aunque se ha desviado menos de lo que parecía en un principio. Si los cambios introducidos no se hubieran realizado, posiblemente se hubiera probado demasiado tarde la parte de la visualización de las constantes vitales, que hay que recordar que es parte fundamental de la aplicación y sin que la posiblemente nadie utilizaría nuestra aplicación. Posiblemente, el error más gordo realizado es que no ha sido suficiente el tiempo de desarrollo invertido en la parte de la visualización de las constantes donde posiblemente se tendría que haber invertido 2 *sprints* enteros en la realización del mismo. En el próximo apartado se indicaron que cambios se van a llevar a cabo en los próximos *sprint*

## 2. Líneas de futuro

Como se ha indicado a lo largo del documento, esta aplicación está orientada a un producto que todavía sea en desarrollo, por lo que en los próximos meses la aplicación va a seguir desarrollándose, añadiendo nuevas funcionalidades y corrigiendo las existentes.

En primer lugar, una vez que se haya finalizado el desarrollo de los prototipos de la camiseta, se deberá realizar un ciclo de prueba para verificar que la pantalla de visualización que se ha realizado es el más idóneo o si requiere cambios, ya que nuestro generador de pruebas es una simulación del mismo, pero con casos muy sencillos de obtención de constantes vitales. Los corazones son sistemas mucho más complejos de lo que se ha realizado y no siempre se limitan básicamente a exactamente al electrocardiograma visto. Existen enfermedades cardíacas que gracias a la visualización de la aplicación deberían visualizarse o reconocerse fácilmente como son posibles arritmias o incluso un paro cardíaco (que representaría una línea plana continua muy por debajo del valor medio de 200).

Lo mismo ocurre con el EDA, el cual es un valor muy continuo y cuya alteración puede durar más de una sola toma de constante (puede durar varios segundos seguidos el susto o la alteración nerviosa del cuerpo). Por lo tanto, se debe verificar si tanto los datos obtenidos y la visualización de las constantes vitales que hay implementado actualmente es suficiente o deben realizarse ajustes para mejorarlo.

En general ahora mismo se necesitan mejorar 4 aspectos de la aplicación:

En la pantalla de visualización de las constantes vitales, sigue siendo un diseño pobre y poco manejable. Actualmente, dado un periodo de tiempo sólo se conoce para cada constante, su valor medio, el mínimo y el máximo. Aunque esta información está bien, se desconoce en que periodo de tiempo se ha producido el mismo y por tanto no puede indicar al usuario cuando se ha producido el mismo. Además, como se ha observado existe un problema con las animaciones en las cuales se reproducen desde el principio del tiempo y van avanzando. Esto presenta una serie de problemas:

- No se puede avanzar, atrasar o pausar la animación.
- No se visualiza en que fecha y hora es de dicho trozo de la animación.
- Siempre se muestra en la misma escala.
- 

Aunque el último punto puede ser un problema para las constantes vitales de ECG y EDA ya que los cambios se producen a lo largo de unos pocos milisegundos y por tanto si se cambia a una escala muy grande la animación no se visualizará. No obstante, si la escala es lo suficientemente grande se puede visualizar más información dentro del canvas, por lo que el usuario no necesita adelantar la animación para ver toda la gráfica. Por tanto, en un nuevo *sprint* (que se llamará *sprint 7*) se van a plantear a los *Early Adopters* los siguientes cambios:

- Añadir botones que permitan avanzar, volver a atrás o pausar la animación de una constante vital.
- Mostrar en todo momento en el canvas a que fecha y hora pertenece ese trozo de la animación
- Permitir mediante unos botones cambiar la escala de la animación.



En segundo lugar, durante el *sprint 6* se han añadido nuevos campos en la creación y edición de las camisetas inteligentes. No obstante, durante las pruebas de *test* los *Early Adopters* se han quejado de que ahora en el formulario existen demasiados campos, aunque exista una opción para desbloquear parte de los campos en que de forma normal los usuarios no rellenan. De hecho, los *Early Adopters* indican que se debe el menú de creación y edición debería dividirse en una pantalla con 3 tabs. En la primera sólo se incluirían los campos obligatorios para rellenar la camiseta (Icono, nombre, parentesco, número de serie y código de seguridad), en la segunda sólo debería ir información relacionada con las constantes vitales (umbrales y notificaciones) y en la tercera sólo debería ir la información extra o de contacto.

Por tanto, en el *sprint 8* se va a plantear un remodelado de las ventanas de creación y edición de las camisetas, proponiendo que sean 3 pestañas cada una con sus campos y un botón común para enviar el formulario. Los usuarios como mínimo tendrán que rellenar los botones de la primera pestaña que son los datos obligatorios y a continuación rellenar los campos que deseen de las otras dos pestañas, según las necesidades que tengan.

Por último, de poco sirve recuperar y visualizar los datos de las constantes vitales si no se pueden exportar para su estudio por parte de un especialista. Esta opción ha sido solicitada por todos los *Early Adopters* desde el principio del desarrollo de la aplicación. La aplicación debería permitir dos funcionalidades respecto a la gestión de los datos de las constantes vitales que deberían desarrollarse durante el *sprint 9*:

- Almacenar los datos obtenidos en la pantalla de visualización de las constantes vitales o recuperar y visualizar los datos los mismos en dicha pantalla. Es decir, una opción para guardar y cargar datos almacenados en local.
- Compartir los datos utilizando medios distintos e incluso formateados de acuerdo a como lo necesiten sistemas externos al nuestro, aquí se podría utilizar tanto los sistemas tradicionales de compartición que permiten las aplicaciones móviles (utilizando otras aplicaciones que tenga instaladas el usuario), así como integración a otros sistemas sanitarios que lo requieran (por ejemplo que hubiera un servicio web implementado por el ejemplo con el Servicio Cantabro de Salud que nos permitiera añadir los datos.

Y el último punto de mejora va más relacionado con la camiseta que con la aplicación móvil, aunque también afecta a la misma. Los *Early Adopters* han solicitado que las personas a su cargo pueden necesitar un botón del pánico que en ocasiones esta situado en una parte de la casa y no lo tienen directamente accesible. Al pulsar este botón debería enviarse una notificación a los usuarios que tengan asociado esta camiseta y a la vez debería realizarse una llamada a un servicio de emergencia que puede consistir en simplemente en contactar con un servicio web lo que permitirá a los operadores de emergencia ponerse en marcha para acudir a la emergencia. Esto se tendría que ver durante un *sprint 10*.

## Bibliografía

- [1] - Bellán García, Antonio; Aceituno Nieto, Pilar; Pérez Díaz, Julio; Ramiro Fariñas, Diego; Ayala García, Alba; Pujol Rodríguez, Rogelio. Un perfil de las personas mayores en España, 2019. Indicadores estadísticos básicos. Madrid, Informes Envejecimiento en red nº 22, 38p. [Fecha de publicación: 06/03/2019]. Disponible en <http://envejecimiento.csic.es/documentos/documentos/enred-indicadoresbasicos2019.pdf>.
- [2] – Viaña, Daniel. El gasto sanitario se incrementará más de 580 millones al año por el envejecimiento de la población. El Mundo [Internet], 2017 [Fecha de publicación: 04/05/2017]. Disponible en <https://www.elmundo.es/economia/macroeconomia/2017/05/04/590a3281e2704e9e178b46b1.html>
- [3] – Artaza Artabe, Iñaki; Ramos Cordero, Primitivo; Gonzalez Nuñez, José; Martínez Hernández, David. Cuidadores de Personas mayores independientes. Madrid, Informes Envejecimiento en red. [Fecha de publicación: 03/11/2016]. Disponible en <http://envejecimiento.csic.es/documentos/documentos/Estudio-Cuidadores-segg.pdf>.
- [4] – Agencia Atlas. Encuentran el cadáver de una anciana que llevaba cinco años muerta. La Provincia, Diario de Las Palmas [Internet], 2019 [Fecha de publicación: 24/04/2019]. Disponible en <https://www.laprovincia.es/multimedia/videos/sucesos/2019-04-24-172706-encuentran-cadaver-anciana-llevaba-cinco-muerta.html>
- [5] – Sanz, J. Hallan un anciano que llevaba más de una semana muerto y rodeado de basura en Valladolid. El Norte de Castilla [Internet], 2019 [Fecha de publicación: 09/07/2019]. Disponible en <https://www.elnortedecastilla.es/valladolid/hallan-anciano-llevaba-20190709201004-nt.html>
- [6] - *Dave Evans*. The Internet of Things: How the Next Evolution of the Internet Is Changing Everything. *CISCO White Paper* [Internet], 2011 [Fecha de publicación 01/04/2011]. Disponible en [https://www.cisco.com/c/dam/en\\_us/about/ac79/docs/innov/IoT\\_IBSG\\_0411FINAL.pdf](https://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf)
- [7] - Man, S. Wearable computing: a first step toward personal imaging. *Computer*, vol. 30, no. 2, pp. 25-32, Feb. 1997.
- [8] - Conor O’Quigley, C. ; Sabourin, M. ; Coyle, S. ; Connolly, J. ; Condall, J. ; Curran, K. ; Corcoran, B. ; Diamond, D. Characteristics of a Piezo-Resistive Fabric Stretch Sensor Glove for Home-Monitoring of Rheumatoid Arthritis. 2014 11th International Conference on Wearable and Implantable Body Sensor Networks Workshops, pp. 23-26, June 2014
- [9] - Google LLC [Internet]. 2019. Google Play. [Consultado el: 07/10/2019]. Disponible en <https://play.google.com/>

- [10] - Google LLC [Internet]. 2019. Salud Conectada. [Consultado el: 07/10/2019]. Disponible en <https://play.google.com/store/apps/details?id=es.sanitas.hosp.saludconectada&gl=ES>
- [11] - Google LLC [Internet]. 2019. Heart Rate OS. Android Watch. [Consultado el: 07/10/2019]. Disponible en <https://play.google.com/store/apps/details?id=com.iwork.wearable.heartratesync2&gl=ES>
- [12] - Google LLC [Internet]. 2019. Health Wereable. [Consultado el: 07/10/2019]. Disponible en <https://play.google.com/store/apps/details?id=com.analogdevices.healthwearable&gl=ES>
- [13] – Broeders J. Transition from Wearable to Medical Devices. Analog Devices [Internet]. [Consultado el: 08/10/2019]. Disponible en <https://www.analog.com/en/technical-articles/transition-from-wearable-to-medical-device.html>
- [14] – López, Gregorio; Custodia, Victor; Moreno, José Ignacio. LOBIN: E-Textile and Wireless-Sensor-Network-Based Platform for Healthcare Monitoring in Future Hospital Environments. IEEE Transactions on Information Technology in Biomedicine, vol.14, issue 6, Nov. 2010
- [15] - Google LLC [Internet]. 2019. Crashlytics Firebase. [Consultado el: 10/10/2019]. Disponible en <https://firebase.google.com/docs/crashlytics/?hl=es-419>
- [16] - Rodríguez, José Ramón. El trabajo final como proyecto. FUOC. Fundació para la Universitat Oberta de Catalunya. [Consultado el: 02/10/2019]
- [17] – Scrum.org [Internet]. 2019. The home of the Scrum. [Consultado el: 09/10/2019]. Disponible en <https://www.scrum.org/>
- [18] – Clarisó, Robert. Introducción al trabajo final. FUOC. Fundació para la Universitat Oberta de Catalunya [Consultado el 03/10/2019].
- [19] – Rodriguez, José Ramón. La gestión del proyecto a lo largo del trabajo final. FUOC. Fundació para la Universitat Oberta de Catalunya [Consultado el 04/10/2019]
- [20] – Rogers, E. Diffutions of Innovation 5th Edition. New York: Free Press of Glencoe, 1962
- [21] – Iconos Flaticon [Internet]. 2019. [Consultado el: 20/10/2019]. Disponible en <http://www.flaticon.com>
- [22] – Iconos Freepik [Internet]. 2019. [Consultado el: 20/10/2019]. Disponible en <http://www.freepik.es>

[23] – Autenticación en Facebook via PHP [Internet]. 2019. [Consultado el: 23/10/2019]. Disponible en <https://github.com/facebook/php-graph-sdk>

[24] - El-Hamid, Ahmed & Fetohi, Amani & Amin, Rabab & Hameed, R. (2015). Design of Digital Blood Glucose Meter Based on Arduino UNO. Volume 3 Issue 8 August, 2015

[25] – Github [Internet]. 2019 [Creado el 02/10/2019]. Disponible en <https://github.com/rpgdragon/loP/https://github.com/rpgdragon/loP/>

[26] – Que es un electrocardiograma [Internet]. 2019. [Consultado el: 30/11/2019]. Disponible en <http://medicinageneraluniversal.blogspot.com/2017/01/que-es-un-electrocardiograma.html>

[27] – Consola de Desarrollador Facebook [Internet]. 2019. [Consultado el 18/10/2019] Disponible en [https://developers.facebook.com/?locale=es\\_ES](https://developers.facebook.com/?locale=es_ES)

[28] – Plugin Login Facebook [Internet]. 2019. [Consultado el 18/10/2019] Disponible en <https://github.com/jeduan/cordova-plugin-facebook4>

[29] – A complete guide to Flexbox [Internet]. 2019. [Consultado el 22/11/2019] Disponible en <https://CSS-tricks.com/snippets/CSS/a-guide-to-flexbox/>

[30] – Firebase Cloud Messaging [Internet]. 2019. [Consultado el 18/12/2019] Disponible en <https://firebase.google.com/docs/cloud-messaging?hl=es>

[31] – Plugin Cordova FMC with Dependency Updated [Internet]. 2019. [Consultado el 18/12/2019]. Disponible en <https://github.com/andrehtissot/cordova-plugin-fcm-with-dependency-updated>

[32] – PDO PHP Mysql [Internet]. 2019. [Consultado el 14/10/2019]. Disponible en <https://www.php.net/manual/es/ref.pdo-mysql.php>

[33] – Instalar APP Android Fuera del Play Sotre [Internet]. 2019. [Consultado el 20/12/2019]. Disponible en <https://es.digitaltrends.com/guias/instalar-app-android-fuera-play-store/>

[34] – Plugin Cordova Firebase Crashlytics [Internet]. 2019. [Consultado el 20/12/2019] Disponible en <https://github.com/ReallySmallSoftware/cordova-plugin-firebase-crashlytics>

[35] - Ries, E. (2011). The lean startup: How today's entrepreneurs use continuous innovation to create radically successful businesses. New York: Crown Business.

[36] - Freepik [Internet]. 2019. [Consultado el 20/10/2019]. Disponible en <https://www.freepik.es> (Fondo de pincel acrílico abstracto azul con textura fondo by rawpixel from [www.freepik.es](http://www.freepik.es))

[37] - Flaticon [Internet]. 2019. [Consultado el 20/10/2019]. Disponible en <https://www.flaticon.es> (Smart clothing icon, older people icons by [Freepik](http://Freepik) from [www.flaticon.com](http://www.flaticon.com))

## Anexos

### Anexo A: Glosario

**Apache:** Servidor web que permite la publicación de contenido web utilizando el protocolo HTTP. El mismo es de código abierto y de los más utilizados del mundo.

**Arduino:** Compañía que desarrolla hardware y software basado en la creación de dispositivos electrónicos capaces de detectar y procesar información relacionada con el entorno que rodea al dispositivo.

**ASO:** Es una rama del marketing que busca como mejorar el posicionamiento de una aplicación móvil en las tiendas de aplicaciones para móviles.

**Backend:** Son los procesos o tareas que no son directamente accesibles por el usuario, como son por ejemplo el acceso a un servidor remoto o a una base de datos.

**Balsamiq:** Aplicación que permite crear prototipos de interfaces de aplicaciones informáticas.

**Base64:** Es un sistema de numeración que usa 64 caracteres distintos como base.

**Bluetooth:** Tecnología de comunicación que permite crear una red personal pequeña muy cercana entre dos dispositivos, lo que permite la transmisión de datos entre ambos dispositivos.

**Cordova:** Popular entorno de desarrollo para desarrollar aplicaciones móviles utilizando configuraciones distintas para cada uno de las posibles plataformas móviles. La misma funciona mediante la utilización de código web utilizando CSS, HTML y Javascript.

**CSS:** Permite mediante la especificación de reglas y atributos indicar como deben visualizarse los componentes de una página web.

**Diagrama de Gantt:** Diagrama que muestra la planificación de tareas e hitos de un proyecto a lo largo del tiempo.

**Early Adopters:** Es el grupo de gente, después de los innovadores, que suele empezar a usar un producto, por que el mismo soluciona un problema que tienen.

**Git:** Es un software que permite gestionar versiones de un conjunto de ficheros de forma descentralizada.

**Github:** Repositorio remoto de gestión de versiones, posiblemente el más conocido que utiliza Git.

**IONIC:** Entorno de desarrollo basada en AngularJS y en Cordova para el desarrollo de aplicaciones móviles multiplataforma.

**MVC:** Patrón utilizado en el desarrollo software basado en 3 principios denominados *modelo*, *vista*, *controlador*. El *modelo* representa al dato o conjunto de información con el que se está trabajando. La *vista* es la parte visual que ve el usuario y el controlador es la parte que realiza las operaciones lógicas necesarias para actualizar la vista y el modelo de acuerdo como uno u otro cambien.

**Product Backlog:** Dentro de la tecnología SCRUM es el conjunto de tareas que hay que realizar para desarrollar todas las funcionalidades de un producto que se irán realizando en los distintos sprint.

**Provider:** Mecanismo en una aplicación web que permite la realización de una tarea o de un servicio dentro de una página web.

**Restful:** Servicio web que implementa la arquitectura REST. La arquitectura REST se basa en que en vez de utilizar los típicos servicios de 1 petición 1 servicio, se basa en que una petición se basa en que una petición obtiene, crea, actualiza o borra un registro en concreto.

**RGPD:** Reglamento General de Protección de Datos. Ley que permite proteger los derechos de los usuarios al uso ilícito de los datos personales de los usuarios por parte de empresas que adquieran dichos datos sin su consentimiento.

**SCRUM:** Metodología ágil que se basa en la rápida entrega y continua de producto respecto a procesos largos de desarrollo y documentación.

**SCSS:** Es un lenguaje que permite preprocesar el contenido del fichero SCSS directamente a una hoja de estilos CSS.

**Smartwatch:** Reloj inteligente que permite tomar una serie de valores, procesarlos e intercambiarlos con un dispositivo vinculado, normalmente un teléfono móvil, lo que permite entre otras cosas visualizar los datos obtenidos desde el reloj en el móvil.

**Sprint:** Ciclo en SCRUM en la que suele ocurrir un ciclo de desarrollo, de duración entre 2 a 4 semanas.

**Testers:** Usuarios que se encargan de probar que una aplicación está libre de errores o bugs.

**Token:** Palabra de texto muy larga y encriptada que contiene información

**Wearable:** Artículo inteligente y con capacidad de procesamiento que puede ser llevado como si fuera una prenda de ropa.



## **Anexo B: Entregables del proyecto**

- Código fuente del proyecto.
- Manual de generación de versiones.
- APK aplicación firma debug.
- Manual de instalación.
- Manual de usuario.
- Ficheros *Balsamiq* con los prototipos realizados.


## **Anexo C: Fichero de planificación inicial**

Junto a esta memoria, se suministra una imagen con la planificación entera inicial que se tiene prevista a la hora de desarrollar la aplicación móvil. La misma se puede encontrar en el fichero anexoC.png

## **Anexo D: Colección Pruebas**

En el fichero anexoD, se encuentra un fichero en formato *JSON* que permite realizar pruebas de peticiones con el programa Postman.


## Anexo E: Perfiles de los Early Adopters

	<p><b>Esther Ridaura</b></p> <p>Claves:</p> <ul style="list-style-type: none"><li>Sus abuelos viven solos y solamente va una asistente social 1 hora al día. El resto del tiempo no hay nadie vigilándolos.</li><li>Trabaja de 8 a 10 horas al día y el resto del tiempo los tiene que dedicar a cuidar de la casa y de sus hijos.</li><li>Su abuela es diabética.</li></ul>	<p><b>HealthShirt quiere que Esther</b></p> <ul style="list-style-type: none"><li>Pueda ver que sus abuelos estén bien.</li><li>No tenga que estar 100% del tiempo vigilando a sus abuelos.</li><li>Avisar a alguien inmediatamente en caso de que sus abuelos no estén bien</li></ul>				
<p><b><u>Perfil personal</u></b></p> <p>Esther es natural de Alcoy, tiene 36 años y vive en Málaga. Actualmente trabaja de lunes a sábado en una pequeña empresa de control de plagas por lo que apenas tiene tiempo libre. El poco tiempo libre que tiene lo tiene que dedicar entre cuidar a sus 2 hijos, realizar las labores del hogar y cuidar a sus 2 abuelos.</p> <p>Los abuelos de Esther se llaman Paco y Marga y tienen 93 y 95 años respectivamente. Desde hace varios años viven solos y solamente son atendidos por una asistente social 1 hora al día que va de lunes a viernes. El resto del tiempo se encuentran completamente solos. Su abuelo hace unos años sufrió un infarto y su abuela es diabética.</p> <p>Debido a los ingresos de Esther y al poco dinero que reciben de pensión sus abuelos, no es posible contratar a un cuidador que este la gran parte del tiempo y por tanto sus abuelos no están cuidados.</p> <p><b><u>Objetivos y Motivaciones</u></b></p> <table><tr><td><p><i>Esther necesita...</i></p><ul style="list-style-type: none"><li>Tener controlada la salud de sus abuelos.</li><li>Vigilar el nivel de glucosa de su abuela.</li></ul></td><td><p><i>Esther teme...</i></p><ul style="list-style-type: none"><li>Que a alguno de sus abuelos le dé un infarto y no se le pueda salvar la vida por no tenerlo vigilado.</li></ul></td></tr><tr><td><p><i>Esther usa HealthShirt para...</i></p><ul style="list-style-type: none"><li>Monitorizar la salud de sus abuelos.</li><li>Poder avisar a algún servicio de emergencia si detecta algún</li></ul></td><td><p><i>Esther presta atención a...</i></p><ul style="list-style-type: none"><li>Al bienestar de sus</li></ul></td></tr></table>		<p><i>Esther necesita...</i></p> <ul style="list-style-type: none"><li>Tener controlada la salud de sus abuelos.</li><li>Vigilar el nivel de glucosa de su abuela.</li></ul>	<p><i>Esther teme...</i></p> <ul style="list-style-type: none"><li>Que a alguno de sus abuelos le dé un infarto y no se le pueda salvar la vida por no tenerlo vigilado.</li></ul>	<p><i>Esther usa HealthShirt para...</i></p> <ul style="list-style-type: none"><li>Monitorizar la salud de sus abuelos.</li><li>Poder avisar a algún servicio de emergencia si detecta algún</li></ul>	<p><i>Esther presta atención a...</i></p> <ul style="list-style-type: none"><li>Al bienestar de sus</li></ul>	<p><b><u>Demografía</u></b></p> <p>Edad: 36 años</p> <p>Estudios: Licenciada en Ciencias Ambientales</p> <p>Trabajo: Técnico de Plaguicidas</p> <p>Sueldo: Aprox – 14.000€ anuales</p> <p>Estado civil: Casada con hijos</p> <p>Hobbies: Naturaleza</p> <p>Personalidad: Activa, Simpática, Ordenada</p> <p><b><u>Capacidades tecnológicas</u></b></p> <p>Ordenador: Usuario medio</p> <p>Internet:</p> <ul style="list-style-type: none"><li>Email</li></ul> <p>Smartphone:</p> <ul style="list-style-type: none"><li>Email</li><li>WhatsApp</li><li>Facebook</li><li>Llamar a sus amigos</li></ul>
<p><i>Esther necesita...</i></p> <ul style="list-style-type: none"><li>Tener controlada la salud de sus abuelos.</li><li>Vigilar el nivel de glucosa de su abuela.</li></ul>	<p><i>Esther teme...</i></p> <ul style="list-style-type: none"><li>Que a alguno de sus abuelos le dé un infarto y no se le pueda salvar la vida por no tenerlo vigilado.</li></ul>					
<p><i>Esther usa HealthShirt para...</i></p> <ul style="list-style-type: none"><li>Monitorizar la salud de sus abuelos.</li><li>Poder avisar a algún servicio de emergencia si detecta algún</li></ul>	<p><i>Esther presta atención a...</i></p> <ul style="list-style-type: none"><li>Al bienestar de sus</li></ul>					


problema en la salud de sus abuelos.	abuelos sin que pierdan su libertad	
--------------------------------------	-------------------------------------	--

### **SCENARIO:**


Es un martes por la tarde y Esther está trabajando cuando la aplicación de HealthShirt notifica a Esther que su abuelo Paco tiene un numero de pulsaciones más elevado de lo habitual. Esther al recibir la notificación la abre y observa que el número de pulsaciones medias excede en 40 a sus pulsaciones normales, por lo que es posible que tenga un problema de arritmia. Esther coge en ese momento y llama a los sistemas de emergencia para que puedan atender a su abuelo.

	<p><b>Paula Estivaliz</b></p> <p>Claves:</p> <ul style="list-style-type: none"> <li>• Su abuela, después de fallecer su marido, se niega a abandonar el hogar</li> <li>• Su abuela está mal de una cadera y se suele caer a menudo. A veces no puede ni levantarse.</li> <li>• Su abuela sufrió un infarto hace 5 años y está en seguimiento.</li> </ul>	<p><b>HealthShirt quiere que Paula</b></p> <ul style="list-style-type: none"> <li>• Pueda ver que su abuela está bien.</li> <li>• Pueda saber cada vez que su abuela se cae por si necesita ir a ayudarla.</li> <li>• Detectar un posible nuevo infarto de su abuela</li> </ul>
<p><b><u>Perfil personal</u></b></p> <p>Paula tiene 34 años y vive en Málaga. Actualmente es cajera y reponedora de fruta en Mercadona. Su horario es un poco complicado ya que 1 semana tiene turno de mañana (de 7:00 a 15:00) y otra semana tiene turno de tarde (14:00 a 22:00) de lunes a sábado.</p> <p>Hace unos años el abuelo de Paula murió y Paula se planteó junto con sus padres y su hermano, el internar a su abuela en una residencia para ancianos. La abuela de Paula se negó en absoluto a irse de su casa donde ha estado viviendo toda su vida. Poco antes de que se muriera su abuelo, a su abuela le entró un infarto. Dicho infarto ha provocado que la movilidad de su abuela se haya reducido, haciendo que en bastantes ocasiones su abuela se caiga a menudo y en ocasiones no se pueda levantar de nuevo.</p>		<p><b><u>Demografía</u></b></p> <p>Edad: 34 años</p> <p>Estudios: Licenciada en Bellas Artes</p> <p>Trabajo: Dependienta de supermercado</p> <p>Sueldo: Aprox – 16.000€ anuales</p> <p>Estado civil: Soltera</p> <p>Hobbies: Museos y galerías de arte</p> <p>Personalidad: Intranquila, Aventurera</p> <p><b><u>Capacidades tecnológicas</u></b></p> <p>Ordenador: Usuario medio</p>

<p><b><u>Objetivos y Motivaciones</u></b></p> <p><i>Paula necesita...</i></p> <ul style="list-style-type: none"> <li>• Tener controlada la salud de su abuela.</li> <li>• Ser notificada si su abuela se cae.</li> <li>• Ser notificado si su abuela puede tener un infarto.</li> </ul> <p><i>Paula teme...</i></p> <ul style="list-style-type: none"> <li>• Que un día su abuela se caiga, no se pueda levantar de nuevo y luego sea demasiado tarde.</li> </ul> <p><i>Paula usa HealthShirt para...</i></p> <ul style="list-style-type: none"> <li>• Monitorizar la salud de su abuela.</li> <li>• Recibir notificaciones de cada vez que su abuela se cae</li> </ul> <p><i>Paula presta atención a...</i></p> <ul style="list-style-type: none"> <li>• A los deseos de su abuela.</li> </ul>	<p>Internet:</p> <ul style="list-style-type: none"> <li>• Email</li> </ul> <p>Smartphone:</p> <ul style="list-style-type: none"> <li>• Email</li> <li>• WhatsApp</li> <li>• Facebook</li> <li>• Llamar a sus amigos</li> </ul>
<p><b><u>SCENARIO:</u></b></p> <p>Es un día cualquiera entre semana y Paula recibe una notificación en su móvil de HealthShirt. La aplicación le notifica que su abuela acaba de caerse. En este momento Paula puede consultar las constantes vitales actuales de su abuela, así como las previas a la caída y ver si está bien o no y actuar en consecuencia.</p>	

	<p><b><i>Isa Martinez</i></b></p> <p>Claves:</p> <ul style="list-style-type: none"> <li>• Es cuidadora social y tiene que dedicar su jornada laboral de 5 personas mayores.</li> <li>• Cada uno de sus pacientes tiene diversas patologías que vigilar, pero se centran principalmente en el corazón, la temperatura, los sofocos y el nivel de azúcar en sangre.</li> </ul>	<p><b><i>HealthShirt quiere que Isa</i></b></p> <ul style="list-style-type: none"> <li>• Monitorizar el bienestar de toda la gente que está a su cargo.</li> <li>• Optimizar el tiempo e ir a cuidar o ayudar a la persona que más lo necesite.</li> <li>• Ante una caída, ir a socorrerla en el menor tiempo posible.</li> </ul>
--	--	---

<p><b><u>Perfil personal</u></b></p> <p>Isa tiene 44 años y aunque estudió pedagogía, la iniciativa en la que se encuentra se dedica principalmente al cuidado de la gente mayor que tiene necesidades especiales. Actualmente Isa está cuidando de 5 ancianos que debe distribuir a razón de 1 o 2 horas al día los 7 días de la semana. Uno de los problemas que tiene es que pierde entre 2 y 3 horas de desplazamiento por Málaga para visitar a la gente que tiene asignada.</p> <p>El problema es que no toda la gente mayor requiere la misma cantidad de cuidados o necesidades todos los días. En ocasiones, la persona sólo necesita hablar un poco, pero a veces requiere necesidades más especiales relacionadas con su salud. Todo esto conlleva a un número grande de horas extras que no son pagadas ni devueltas.</p> <p><b><u>Objetivos y Motivaciones</u></b></p> <div> <div> <p><i>Isa necesita...</i></p> <ul style="list-style-type: none"> <li>• Optimizar su tiempo en el cuidado de cada persona a su cargo.</li> <li>• Ser alertada si alguna persona se cae.</li> <li>• Ser alertada si alguna persona tiene alguna necesidad.</li> </ul> </div> <div> <p><i>Isa teme...</i></p> <ul style="list-style-type: none"> <li>• Ir a cuidar a una persona mientras otra tiene mayor necesidad.</li> </ul> </div> </div> <div> <div> <p><i>Isa usa HealthShirt para...</i></p> <ul style="list-style-type: none"> <li>• Vigilar que toda la gente a su cargo está bien.</li> <li>• Recibir notificaciones de cada vez que alguna se cae para ir a socorrerla.</li> </ul> </div> <div> <p><i>Isa presta atención a...</i></p> <ul style="list-style-type: none"> <li>• Al bienestar y felicidad de todos a los que cuida.</li> </ul> </div> </div>	<p><b><u>Demografía</u></b></p> <p>Edad: 44 años</p> <p>Estudios: Pedagoga</p> <p>Trabajo: Cuidadora en iniciativa social</p> <p>Sueldo: Aprox – 18.000€ anuales</p> <p>Estado civil: Casada con hijos</p> <p>Hobbies: Series y películas</p> <p>Personalidad: Seria y carismática</p> <p><b><u>Capacidades tecnológicas</u></b></p> <p>Ordenador: Usuario medio</p> <p>Internet:</p> <ul style="list-style-type: none"> <li>• Email</li> </ul> <p>Smartphone:</p> <ul style="list-style-type: none"> <li>• Email</li> <li>• WhatsApp</li> <li>• Facebook</li> <li>• Llamar a sus amigos</li> </ul>
<p><b><u>SCENARIO:</u></b></p> <p>Isa empieza su jornada laboral y mientras decide a que persona visita primero, observa que una de sus pacientes se ha caído mientras se levantaba de la cama, en dicho momento se dirige hacia allí para ayudarla. Una vez incorporada, ve que otro de sus pacientes ha estado intranquilo desde esta mañana por lo que se dirige hasta allí</p>	

	<div><div><b>Paöla Sousa</b></div><div>Claves:<ul style="list-style-type: none"><li>• Pasa 3 horas por la mañana y 3 horas por la tarde cuidando a un matrimonio mayor</li><li>• Por la mañana les ayuda a levantarse, cambiarse y les hace la comida</li><li>• Por la tarde les hace la cena, les cambia y les acuesta.</li></ul></div></div>	<div><div><b>HealthShirt quiere que Paöla</b></div><div><ul style="list-style-type: none"><li>• Pueda ver si alguno de los señores mayores de los que cuida necesita ayuda,</li><li>• Se entere de alguna necesidad o enfermedad sin necesidad de que los señores mayores se lo comuniquen.</li></ul></div></div>
<div><div><b><u>Perfil personal</u></b></div><div><p>Paöla lleva más de 5 años cuidando a un matrimonio mayor que vive a escasos 3 minutos de su casa. Ella se encarga de levantarlos, cambiarlos, asearlos, hacerles la comida/cena y luego acostarlos.</p><p>Para todo invierte unas 3 a 4 horas al día, pero ella tiene contratada unas 6 horas que debe permanecer allí (3 por la mañana y 3 por la tarde/noche). El problema es que dichas personas requieren de más cuidados, normalmente fuera de las horas en las que ella esta presente ya que en ocasiones se levantan por la noche y se caen o simplemente no se pueden levantar. En ocasiones a alguno de los dos les entra golpes de calor que requieren de asistencia inmediata.</p><p>Ella ya ha sugerido en más de una ocasión en hacer 2 horas por la mañana y 2 por la tarde y dejar 2 horas al día para posibles emergencias, ya que vive cerca y se puede acercar en cualquier momento que lo necesiten, pero el matrimonio mayor tiene problemas para llamar por teléfono.</p><div><div><b><u>Objetivos y Motivaciones</u></b></div><div><div><div><div><i>Paöla necesita...</i><ul style="list-style-type: none"><li>• Tener controlada la salud de su abuela.</li><li>• Ser notificada si alguno de los dos se cae.</li><li>• Ser notificada si alguno de los dos sufre un golpe de calor.</li></ul></div><div><i>Paöla teme...</i><ul style="list-style-type: none"><li>• Desperdiciar horas que podrían venir bien cuando fueran realmente necesarias.</li></ul></div></div><div><div><i>Paöla usa HealthShirt para...</i><ul style="list-style-type: none"><li>• Monitorizar la salud del matrimonio mayor.</li><li>• Recibir notificaciones cuando alguno de los dos necesita ayuda.</li></ul></div><div><i>Paöla presta atención a...</i><ul style="list-style-type: none"><li>• Al bienestar y a la privacidad del matrimonio mayor y al suyo propio.</li></ul></div></div></div></div></div><div><div><b><u>Demografía</u></b></div><div><p>Edad: 53 años</p><p>Estudios: Graduado Escolar</p><p>Trabajo: Cuidadora</p><p>Sueldo: Aprox – 9.000€ anuales</p><p>Estado civil: Casada con hijos</p><p>Hobbies: Museos y galerías de arte</p><p>Personalidad: Serena, Tranquila</p><div><div><b><u>Capacidades tecnológicas</u></b></div><div><p>Ordenador: Usuario medio</p><p>Internet:</p><ul style="list-style-type: none"><li>• Email</li></ul><p>Smartphone:</p><ul style="list-style-type: none"><li>• Email</li><li>• WhatsApp</li><li>• Llamar a sus amigos</li></ul></div></div></div></div></div></div>		



--	--

**SCENARIO:**

Paöla va a las 08:00 AM al piso del matrimonio mayor, los levanta, los asea, los cambia y les deja la comida preparada. A las 10:00 AM ha terminado y les comenta que si ven que necesitan alguna cosa que en 3 minutos estará allí para lo que necesiten.

A las 16:00 PM le llega a Paöla un aviso al móvil indicándole que el hombre mayor no se encuentra bien y que tiene sudores y temblores. Ella sobre las 16:05 PM llega allí y observa que el hombre no se ha llegado a tomar la medicación que le correspondía a mediodía. Se la hace tomar y empieza a preparar la cena para tenerla lista para luego. Mientras el matrimonio cena y ve un poco de televisión, Paöla vuelve a su casa para preparar su cena. Luego vuelve sobre las 21:00PM para acostarlos en la cama.

A las 02:00 AM detecta que la señora mayor ha intentado levantarse y se ha caído de bruces contra el suelo. Paöla se acerca un momento al domicilio del matrimonio y ayuda a la señora a levantarse y acostarse de nuevo.

## **Anexo F: Curriculum Vitae**

José Manuel Castellano Domínguez, graduado en Informática especializado en tecnologías J2EE, PHP y con un Máster en el desarrollo de aplicaciones móviles (Android, IOS y Windows Phone), ha realizado, colaborado y dirigido varios proyectos de sistemas web y de aplicaciones para dispositivos móviles. Actualmente trabaja como programador Senior en la línea de emergencias y sanidad en la empresa Ingenia S.A.

Durante los últimos 9 años ha estado utilizado diferentes frameworks como son CodeIgniter y Laravel para tecnologías PHP y Struts, JSF y Spring para Java compaginando con el desarrollo de aplicaciones móviles para los sistemas operativos Android e IOS.

## Anexo G: Transcripciones de entrevistas

En este anexo se van a transcribir las notas realizadas en las entrevistas. Dado que las entrevistas fueron telefónicas y con una duración aproximada de 10 minutos, no se realizaron transcripciones ni grabaciones de la totalidad de la misma. Únicamente apunté todas aquellas ideas relevantes que me iban comentando en cada una de las entrevistas que luego utilicé para la realización de las capturas de Balsamiq. Además, a las personas entrevistadas no se les indico que podrían ser grabadas, sólo limite a indicarles que solamente las ideas más importantes que me fueran comentando serían anotadas y que me comprometía a que ningún dato personal viera la luz. Como se ha indicado en el anexo E, los datos personales de los *Early Adopters* se han ocultado.

A lo largo de este anexo se van a ir transcribiendo todas las cosas que fui apuntando en las entrevistas y en los procesos de validación.

### **Sprint 1:**

#### **Entrevistas:**

*Esther:*

Abuela diabética, Abuelo sufrió infarto. Abuela se ha caído varias veces en el último mes. Abuela coge frío por las noches. Ver gráfica nivel azúcar, Ver temperatura actual y por la noche. Si falla recibir dato indicarlo sin rellenar información. Ver separado estado actual e histórico.

Login sencillo, rápido. No volver a pedirlo, no gusta rellenar datos. No pedir login directo, poner pantalla presentando la aplicación.

*Paula:*

Abuela infarto 5 años. Problemas cadera, sufre caídas de continuo. Ver electrocardiograma en directo. Nerviosismo, sudores fríos. Ver de un plumazo si esta bien o mal. Páginas distintas por fecha visualización

Credencial simple, sencillo y rápido y los mínimos posibles. No pedir datos. Recordar automáticamente

*Isa:*

Cuidado varios ancianos iniciativa. Tratamiento de sofocos, detectar rápido pacientes nerviosos. Visualización, customizar por constantes vitales. Prioridad ECG, EDA y temperatura. Dejar poner FBS si necesito. Una gráfica por constante separadas.

Login no necesario, removerlo, si necesario una única vez. Esquiva página si tiene que rellenar datos. Usar email. Presentar app antes de acceder. Ver contraseña, se equivoca siempre.

Paöla:

Matrimonio mayor propenso a caerse y a tener sofocos y ataques de calor. Visualizar temperatura últimos 10 minutos. Ver temperatura en rango de tiempo. Ver ritmo cardiaco actual y si posible ataque al corazón. Avisar al médico en caso de posible ataque.

Poder escribir la contraseña sin necesidad de ocultarla. Presentar la app, saber que la estoy abriendo.

### **Validación 1:**

*Esther:*

Feo, sin imágenes, poner iconos. Demasiado larga página, agrupar por partes o páginas.

Idem visualización, ¿De que va la app? Más info. ¿Recordar contraseña? Facebook.

*Paula:*

No dice nada, poner imágenes finales. Poner constante por página. Dejar elegir constantes.

Idem visualización, Facebook, ni idea de que va la app. Facebook o Twitter. ¿Y si pierdo contraseña?

*Isa:*

Necesita más colorido. Separar constantes. Gráficas más grandes.

Idem visualización, Facebook, ¿Y donde compro la camiseta? Siempre se olvida contraseña

*Paöla:*

No gusta, separar, lioso

Idem visualización, no explica para que es la app. ¿Necesito camiseta?

### **Validación 2:**

*Esther:*

Agrupar formulario constantes actuales o históricos. Eliminar 2ª pagina.

Ok

*Paula:*

Páginas visualización demasiado iguales, diferenciarlas o agrupar.

Ok

*Isa:*

Dejar histórico, hora actual y si cambian modificar la 2ª página

Ok

*Paöla:*

Ok

Ok

## **Sprint 2:**

### **Entrevistas:**

*Esther:*

No pedir datos para el registro. Pedir pocos. No gusta rellenar formularios. Permitir registro por Facebook, pero pedir los minimos. Si no existe, pedir contraseña. Se puede introducir cualquier email, debe asegurarse de que es quien lo solicita.

Se fija en imágenes, por tanto, la camiseta debe indentificarse por un icono. Iconos del programa y no personales. Si datos escritos, indicar nombre de la camiseta

*Paula:*

Mínimos datos. Email y contraseña suficientes. Si puede ser usar login teléfono a lo Orange. Permitir registro por Facebook o por Twitter y pedir lo minimo posible

Indicar camiseta y quien la lleva. Al lado una imagen que identifique fácilmente a quien pertenece la camiseta y dato auxiliar.

*Isa:*

Reducir campos, protección RGPD. Registro por Facebook en 2 clicks. Pedir confirmación vía email

Tiene varios abuelos que se llaman igual, usar un campo adicional para indicar quien es o cliente al que pertenece (parentesco). Usar iconos identificativos, pero no del programa

*Paöla:*

Pedir un email o usuario y la contraseña, luego asegurarse de que soy yo quien lo ha pedido ya que se puede introducir cualquier correo

Iconos de hombre, mujer o de ambos (compra 1 camiseta, usan ambos). Disponer de 2 campos de información

### **Validación:**

*Esther:*

¿Puedo meter cualquier contraseña? Indicar condiciones si hay. Si registro por Facebook indicar si Facebook o normal.

Indicar última vez conectado

*Paula:*

Indicar antes de meter contraseña restricciones, no informar después. Odio reescribir.

Si camiseta eléctrica indicar batería en listado. Indicar hora últimos datos

*Isa:*

¿Se informa después condiciones contraseña? Poner antes

Añadir más datos nivel de batería, último dato cada constante y cuando fueron

*Paöla:*

Saber como debo escribir contraseña.

Indicar en listado si usuario esta bien o no.

### **Sprint 3:**

#### **Entrevistas:**

*Esther:*

Definir menor cantidad de campos posibles. Pedir icono (lista), nombre, parentesco y el código camiseta identificativo como obligatorio. ¿Y si un familiar o un desconocido mira el código camiseta, poder acceder? No pedir más datos si posible. En editar, evitar pedir código camiseta (¿no cambia se entiende no?). Editar y borrar arrastrar desde camiseta.

¿Umbral por debajo o por arriba? Depende de constante no

*Paula:*

Sólo campos obligatorios y pocos. Mismos que lista, icono lista y visualizable, interactuar con icono. Indicar como introducir código camiseta. No subir camiseta si formulario no es correcto. Borrado sin tener que acceder a camiseta

Definir umbral lo que se salga entre 2 valores. Ejemplo si temperatura 35º a 37º normal, resto aviso.

*Isa:*

No más de 5 campos. Permitir que varios usuarios puedan acceder a misma camiseta, pero impedir acceso a otros. Mecanismo eliminar a viejos empleadores de la camiseta. Campos nombre, dato adicional, icono, código camiseta y alguno de teléfono o dirección. Borrar camiseta con un solo clic. Evitar modificación de camiseta, campo inhabilitado.

Un umbral por debajo y otro por arriba, no obligatorios y pudiendo rellenar ambos.

*Paöla:*

Fácil de rellenar, formularios pequeños. Que nadie más vea los datos de los pacientes. Lista amplia de iconos y que haya parejas.

Un umbral por debajo de un valor normal de una constante y un valor por encima de lo normal de una constante vital. Total 6 umbrales.

#### **Validación:**

*Esther:*

No, umbrales a nivel de camiseta no, a nivel global. Abuelo no es igual a abuela.

¿Y que valores introduzco como umbral?

*Paula:*

¿Y si defino un umbral de ECG y el otro siempre supera dicho valor, siempre lanzaría notificación?

¿Qué valor de ECG es bueno como umbral, el NPM o un valor de latido?

*Isa:*

¿Esto es a nivel de camiseta? ¿No debería estar junto en cada definición de camiseta? 2 pacientes distintos para uno temperatura normal 37º y para otro ya fiebre.

¿Qué valor se considera para cada constante como normal? ¿El visualizado en la pantalla de visualización o el que elija?

*Paöla:*

Uniría todo en la misma ventana, no ir a dos ventanas distintas.

Ok (No indica nada relacionado con la ayuda).

### **Validación2:**

*Esther:*

Quizás el icono quedaría mejor arriba

*Paula:*

Ok

*Isa:*

No me gusta donde está lo del icono, debería ir junto con el resto de campos obligatorios.

*Paöla:*

Bien

### **Sprint 4:**

#### **Entrevistas:**



*Esther:*

Notificaciones por constante o caída separadas. Fáciles de diferenciar de un vistazo. Permitir apagar o encender notificaciones por camiseta o todas de golpe. Poner nuevo menú separado. No moleste resto menús.

*Paula:*

Separar notificaciones. Añadir opción notificaciones caída en formulario camiseta. Poner algo para poner modo apagado o avión.

*Isa:*

Notificación por constante, notificación de batería, poner valores por camiseta y a nivel global. Nivel global separado del resto. Nuevo menú global separado del resto.

*Paöla:*

Cada constante su notificación, cada umbral su notificación. Informar si batería baja, <- esto nivel global no por camiseta. Ajustes en barra aplicación.

### **Validación:**

*Esther:*

Se ve claro.

*Paula:*

Todo claro.

*Isa:*

No ve problema.

*Paöla:*

Ok.

### **Sprint 5:**

#### **Entrevistas:**

*Esther:*

Notificación por umbral rebasado, indicar que umbral, constante y valor recibido. Poner 2 frases del tipo Aviso Constante Vital superado Nombre camiseta, lo mismo para las caídas y la batería. Acceder a valor actual si pulso notificación, si app abierta abrir pantalla y mostrar dato.

*Paula:*

No agrupar avisos. Si varios problemas ver todos. Mostrar información de umbral y camiseta con valor con el que se ha superado el mismo. Recibir notificación cada valor nuevo que se supere.

*Isa:*

1 aviso por constante y 1 aviso por rebaso. Indicar tipo de aviso que es indicando en 2 líneas que se ha recibido, de que camiseta se ha recibido y en segunda línea indicar por el valor que se ha superado el mismo. Poder visualizar todas las notificaciones en móvil. Si app abierta redirigir

*Paöla:*

Avisos sonoros y vistosos. No eliminar notificación. Indicar por cada uno información de que es la notificación, la camiseta que lo ha hecho y valor recibido del umbral rebasado. Ver notificación en vivo si app abierta y sino aguardar a en el teléfono para verla más tarde. Insistir en caso de no recepción o leído.

### **Validación:**

*Esther:*

Todo ok.

*Paula:*

Ok.

*Isa:*

Nada.

*Paöla:*

No ve nada raro.

### **Sprint 6:**

### **Entrevistas:**

*Esther:*

Si, es cierto que si quiero ver el mínimo y el máximo en un periodo de tiempo ahora mismo no es práctico. Si abuelo le suben pulsaciones, tengo que ver minuto a minuto. Sería practico ver dicha información, al menos en histórico.

Agregar campos nuevos no necesario. No obstante, Esther nunca se acuerda de la edad de su abuelo. Sería interesante edad o fecha de nacimiento. Si es posible evitarlos y si tienen que aparecer, ocultarlos en la medida de lo posible.

*Paula:*

Echa de menos ver mínimo y máximo fácilmente. Ver todo en la misma línea que la media. Localizar el tiempo exacto donde se produce mínimo y máximo.

No tiene necesidad, pero indica que edad y sexo son campos viables. Intentar que sean los menos posibles. Ubicarlos donde no se suela acceder.

*Isa:*

Mínimo y máximo al menos, pero se necesita detectar anomalías en datos estudiando gráfica, tiempo transcurrido de anomalía, tiempo duración anomalía. Mostrar datos junto a media.

Necesito tener fácil acceso teléfono y dirección de los usuarios, al menos necesita 2 telefonos y campos dirección. Además, necesita campos medicación paciente o citas medico.

*Paöla:*

Necesita mínimos y máximos temperatura, máximos ECG y mínimos EDA, resto opcionales. Mostrar misma línea que media

Dos teléfonos, señor y señora o fijo y móvil. Campos para apuntar observaciones o notas.

### **Validación:**

*Esther:*

Nada que destacar.

No me sirve, además hace demasiado largo el formulario, aunque el mismo se encuentre oculto. Habría que hacerlo en varios modos, uno simple para usuarios como yo y otro avanzado para otras necesidades.

*Paula:*

Bien, me ayuda a ver en un periodo largo de tiempo si ha habido algún problema.

No creo que lo utilice, pero siempre esta bien tenerlo. No obstante, tengo que tirar muy abajo para guardar los cambios. Hacer más pequeño (formulario entiendo).

*Isa:*

Genial. Veo rápidamente en un periodo de tiempo si hay alteraciones o no.

Perfecto, ahora puedo tenerlo ahí a modo recordatorio toda la información. De todos modos, no se ve práctico el formulario

*Paöla:*

Todo bien, se ve mejor.

Me viene bien para guardar la información de la medicación y también si pierdo el teléfono, recuperar el número de teléfono del matrimonio. Estaría bien que no estuviera escondido y estuviera más arriba.