Universidad de Valladolid

UNIVERSIDAD DE VALLADOLID

E.T.S.I. TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

GRADO EN INGENIERÍA DE TECNOLOGÍAS ESPECÍFICAS DE TELECOMUNICACIÓN. MENCIÓN EN TELEMÁTICA

CARDIOMANAGER.
Aplicación orientada a personas con problemas del corazón, para dispositivos Android

Autor:
D. Josu Escalada Blanco
Tutor:
Dña. Isabel de la Torre Díez
Valladolid, 14 de Julio de 2014

Agradecimientos

A mi familia, por su apoyo durante estos cuatro años desde la lejanía, a todos los colegiales del CMU Santa Cruz, que tan buenos momentos me han hecho pasar, a mis amigos de ese magnífico pueblo que es Lerma, y a los miembros del Grupo de Telemedicina y eSalud, tanto a mi tutora de TFG, Isabel, como al doctorando Borja, por su ayuda prestada.

Universidad de Valladolid

TÍTULO:	**CARDIOMANAGER. Aplicación orientada a personas con problemas del corazón, para dispositivos Android.**
AUTOR:	**D. Josu Escalada Blanco**
TUTOR:	**D. Isabel de la Torre Díez**
DEPARTAMENTO:	**Departamento de Teoría de la Señal y Comunicaciones e Ingeniería Telemática**

TRIBUNAL

PRESIDENTE:	**D. Miguel López-Coronado Sánchez-Fortún**
VOCAL:	**Dña. Isabel de la Torre Díez**
SECRETARIO	**Dña. Beatriz Sainz de Abajo**
SUPLENTE	**D. Carlos Gómez Peña**
SUPLENTE	**D. Salvador Dueñas Carazo**

FECHA:	**14 de Julio de 2014**
CALIFICACIÓN:	

Resumen de TFG

Debido a los altos niveles de integración de los dispositivos móviles en nuestra sociedad, y a la importancia de las enfermedades cardiovasculares, que son la principal causa de mortalidad y discapacidad en los países desarrollados, surge la necesidad de aprovechar las oportunidades que nos ofrece la tecnología para mejorar las condiciones de vida de los pacientes y ayudarles en lo posible, lo que se conoce como eSalud. Este trabajo consiste en la realización de la aplicación CardioManager, diseñada para funcionar en dispositivos con sistema operativo Android, englobada dentro del campo de la eSalud, que pretende ayudar a personas con problemas del corazón, informándoles sobre diferentes aspectos de su enfermedad, llevando un registro de medicamentos, registrando la realización de diferentes actividades y calculando el riesgo de sufrir un accidente cardiovascular.

Palabras clave

Android
Dispositivos móviles
Smartphone
Eclipse

Java
eSalud
Enfermedades cardiovasculares

Abstract

Owing to the high integration of the mobile devices in our society, and to the importance of the cardiovascular diseases, which are the main cause of death and disability in the developed countries, the necessity of taking the advantages technology offers to us in order to improve the life conditions of the patients and help them as possible arises, which is known as eHealth. This work consists on the development of an application called CardioManager, designed to work in devices with Android operative system embedded. CardioManager is aimed to help people with heart problems, giving them some advice about some aspects of their disease, monitoring the doses of their medicines, saving different activities related to it and calculating their risk of suffering a cardiovascular accident.

Keywords

Android
Mobile device
Smartphone
Eclipse
Java
eHealth
Cardiovascular diseases

TABLA DE CONTENIDOS

BLOQUE 1
INTRODUCCIÓN

Enfermedades cardiovasculares [1]

Las enfermedades cardiovasculares engloban todos los trastornos y enfermedades que afectan al corazón y los vasos sanguíneos (arterias y venas). Dependiendo del órgano a tratar y de la enfermedad, los encargados de tratar las mismas suelen ser cardiólogos, cirujanos vasculares, neurólogos o radiólogos. Este tipo de enfermedades son la principal causa de muerte o discapacidad en los países desarrollados.

Principales enfermedades cardiovasculares

Son muchas las enfermedades cardiovasculares existentes, sin embargo, nos centraremos en algunas de las principales, que estarán también contenidas en la aplicación *CardioManager* que posteriormente será presentada.

Hipertensión

La presión arterial es la fuerza que hace la sangre sobre las paredes de los vasos sanguíneos. Cuando esta presión supera determinado límite, decimos que el paciente padece hipertensión. En concreto, este límite está marcado por los valores 140/90 mmHg (presión sistólica/presión diastólica). Se detecta mediante medidores de presión arterial.

Causas. En la mayoría de los casos, la causa es desconocida, aunque se sabe que hay una fuerte influencia hereditaria. Una minoría de los casos tiene una causa conocida y que puede ser tratada, haciendo desaparecer la enfermedad por completo para siempre. Si bien es verdad, que se conocen algunos factores de riesgo, como son los niveles de sodio, el estado de los riñones y si se padecen enfermedades renales, la apnea durante el sueño, la diabetes o la edad.

Síntomas. No existen síntomas si la enfermedad se ha manifestado de una forma leve. Si la enfermedad es grave, puede producirse náuseas, vómitos, pérdida de visión o sangrado nasal.

Tratamiento. Puede llevarse a cabo un tratamiento a base de hábitos saludables tales como dieta sana y equilibrada, baja en grasas y sales, realizar deporte y evitar el sedentarismo, el tabaco y el alcohol, o a través de medicamentos recetados por el médico.

La cardiopatía isquémica es una enfermedad cardiovascular producida por la arteriosclerosis de las arterias coronarias, que son las encargadas de proporcionar la sangre al músculo cardiaco. La arteriosclerosis consiste en el estrechamiento de las paredes de los vasos sanguíneos debido a la acumulación de grasas y células inflamatorias (linfocitos). Este proceso se inicia desde los primeros años de vida, aunque no es peligroso ni presenta síntomas hasta que este estrechamiento sea lo suficientemente notable. Puede causar una disminución del oxígeno que recibe el músculo del miocardio y provocar desde una angina de pecho, hasta un infarto si es muy acentuada esta falta.

Causas:

- ✓ Edad avanzada
- ✓ Antecedentes familiares

- ✓ Aumento de las cifras de colesterol total, sobre todo LDL, y disminución de los valores de colesterol bueno HDL.
- ✓ Tabaquismo, alcoholismo, diabetes, obesidad, sedentarismo, estrés
- ✓ Hipertensión

Síntomas. Como bien hemos comentado, la enfermedad comienza a desarrollarse desde los primeros años de vida, pero sus síntomas no se hacen evidentes hasta pasados muchos años. Los más comunes son dolor en el pecho, pesadez, fatiga en la respiración.

Tratamiento. Esta enfermedad puede tratarse a través de diferentes medicamentos, reduciendo los factores de riesgo, mediante programas de rehabilitación, o a través de intervención quirúrgica. Se pueden llevar a cabo diversos hábitos saludables, como seguir una dieta sana y equilibrada, baja en sal y grasas, o en caso de haber sufrido un infarto de miocardio grave, no realizando esfuerzos físicos.

Enfermedad valvular

Las enfermedades valvulares o valvulopatías impiden a las válvulas del corazón abrirse o cerrarse correctamente durante el ciclo cardiaco. Las cuatro válvulas pueden verse afectadas, aunque las más peligrosas son las que afectan a la válvula aórtica o a la mitral.

Causas. Las más comunes son la afectación reumática y la degenerativa, malformaciones o infecciones.

Síntomas. Los más graves suceden en el caso de la valvulopatía aórtica, y disnea (sensación de falta de aire), dolor torácico y síncope (pérdida del conocimiento). En el resto de casos, aunque no son tan graves, puede producirse disnea o aparición de arritmias como la fibrilación auricular.

Tratamiento. Sustitución o implante de la válvula que falla por una prótesis metálica o biológica.

Fibrilación auricular

La fibrilación auricular es un ritmo cardiaco irregular y anormal, que se traduce en latidos muy rápidos y puede desembocar en coágulos de sangre que pueden viajar desde el corazón hasta el cerebro, ocasionando un infarto cerebral.

Causas. La principal causa de esta enfermedad es que las aurículas del corazón no se contraen con normalidad, siendo incapaces de mantener el ritmo normal de latido.

Síntomas:

- ✓ Latidos fuertes y rápidos.
- ✓ Dolor en el pecho.
- ✓ Dificultad para respirar.
- ✓ Mareos, desmayos, desorientación o cansancio.

Tratamiento. Mediante medicamentos para recuperar el ritmo cardiaco o anticoagulantes, o mediante cardioversión eléctrica (electrochoques al corazón).

Otras enfermedades

Insuficiencia cardiaca. Se trata de la consecuencia del desequilibrio entre la capacidad del corazón para bombear sangre y la necesidad de oxígeno y otros nutrientes del organismo.

Muerte súbita. Es la parada repentina del corazón en una persona que se encuentra aparentemente sana. Su principal causa es la arritmia cardiaca.

Miocardiopatía. Las miocardiopatías son enfermedades específicas del músculo cardiaco, el miocardio, que puede realizar malas contracciones que no vacíen correctamente el corazón, malas relajaciones que no permitan llenarlo o ambas al mismo tiempo.

Endocarditis infecciosa. Es la inflamación del revestimiento de las paredes internas del corazón, como consecuencia de una infección producida por microorganismos o bacterias que han entrado en la sangre.

Arritmia. La arritmia es la alteración del ritmo cardiaco respecto al ritmo normal.

Cardiopatías congénitas. Son las enfermedades cardiovasculares que engloban malformaciones o defectos en el corazón producidas durante el periodo embrionario como son los cortocircuitos izquierda derecha, cuando la sangre pasa del circuito sistémico al pulmonar.

Enfermedad de Kawasaki. Se trata de una inflamación de las arterias del organismo que aparece generalmente en los niños, y que puede desembocar en la aparición de aneurismas.

Coartación de aorta. Es un estrechamiento de la arteria aorta que provoca una disminución del flujo sanguíneo.

Directrices, hábitos saludables, consejos y prohibiciones

Es recomendable seguir una dieta sana y equilibrada, baja en grasas y sal, para mantener unos niveles de colesterol correctos. Para ello es necesario que la dieta sea rica en frutas, verduras, legumbres y pescado, tratando de evitar alimentos ricos en colesterol como los huevos, lácteos no desnatados, carne, pastelería, bollería industrial, fritos o salsas. Los médicos también recomiendan la realización de actividades físicas de forma moderada y adecuándose a la enfermedad y a la edad del paciente, destacando caminar, la natación o el ciclismo.

Se aconseja a los pacientes que sigan con detalle las especificaciones de los médicos, que cumplan con rigurosidad los horarios de tomas de medicación, respetando la cantidad. También se aconseja controlar los factores que aumentan el riesgo de padecer este tipo de enfermedades, como son la presión arterial y el colesterol, los niveles de glucosa en el caso de los diabéticos, y sobre todo, mantener una vida tranquila libre de estrés.

Se debe evitar a toda costa el tabaco, el alcohol, el sedentarismo y en general los excesos.

Riesgo de sufrir un accidente cardiovascular [2]

Los factores de riesgo cardiovascular son los que tienen relación con la probabilidad de sufrir una enfermedad cardiovascular. Estos factores de riesgo son:

- ✓ Colesterol.
- ✓ Hipertensión.
- ✓ Diabetes.
- ✓ Tabaquismo.
- ✓ Falta de ejercicio, sedentarismo.
- ✓ Mala alimentación.
- ✓ Obesidad.
- ✓ Estrés y ansiedad.
- ✓ Drogas.
- ✓ Frecuencia cardiaca.
- ✓ Edad.
- ✓ Antecedentes familiares.
- ✓ Sexo o género.
- ✓ Anticonceptivos orales.
- ✓ Herencia genética.
- ✓ Gripe.
- ✓ Proteina C reactiva (PCR).
- ✓ Enfermedades cardiovasculares.

Modelos de riesgo en medicina

Es necesaria la creación de modelos de riesgo de un suceso para conocer las variables que intervienen en el proceso, los factores de riesgo en este caso, y analizar cómo se produce y cómo podemos prevenir su aparición. Los modelos creados en medicina no son deterministas, sino probabilísticos, en los que existe cierto grado de incertidumbre. Además, se centran en casos generales, mientras que en la práctica debemos tratar a los pacientes de forma individual, estudiando caso por caso.

Modelos de riesgo cardiovascular

Como bien hemos comentado antes, las enfermedades cardiovasculares son una de las principales causas de mortalidad de los países desarrollados, por lo que surge la necesidad de desarrollar un modelo que nos avise sobre el riesgo de sufrir una de estas enfermedades, con el fin de prevenirlas.

Existen múltiples trabajos sobre el cálculo del riesgo cardiovascular, aunque el método por excelencia es el método de Framingham¸ muy utilizado para la toma de decisiones en base a la estimación que realiza.

Sin embargo, tras diversos estudios realizados, se observó que sobreestimaba el riesgo absoluto de la enfermedad cuando se trataba de países europeos, con diferente estilo de vida al de los habitantes de la ciudad de Framingham, Massachussetts, USA, donde se realizó el estudio. Por

ello, surge de nuevo la necesidad de crear un modelo más adecuado para este nuevo entorno, Europa. El proyecto SCORE, de reciente publicación, sofoca esta necesidad, y permite calcular el riesgo cardiovascular para los habitantes de los países europeos.

Existen otros modelos, entre los que cabe destacar el proyecto INDIANA, adaptado a pacientes tanto europeos como norteamericanos. Todos ellos disponen de calculadoras online disponibles para todos los públicos.

Cálculo del riesgo cardiovascular según el modelo de Framingham

Los factores que debemos tener en cuenta para este modelo son:

- ✓ Sexo.
- ✓ Edad.
- ✓ Colesterol, total y HDL.
- ✓ Presión arterial.
- ✓ Diabetes.
- ✓ Tabaquismo.

En primer lugar se debe calcular el valor de la siguiente expresión:

Para los hombres: $L_H = b_{E1} \cdot Edad + b_C + b_H + b_T + b_D + b_F$

Para las mujeres: $L_M = b_{E1} \cdot Edad + b_{E2} \cdot Edad^2 + b_C + b_H + b_T + b_D + b_F$

Donde los coeficientes *b* son los siguientes:

Coeficiente	**Hombres**	**Mujeres**
b_{E1}	0.04826	0.33766
b_{E2}	0	-0.00268
b_C Coediciente colesterol mg/dl		
< 160	-0.65945	-0.26138
160-199	0	0
200-239	0.17692	0.20771
240-279	0.50539	0.24385
≥ 280	0.65713	0.53513

b_H Coeficiente HDL mg/dl		
< 35	0.49744	0.84312
35 – 44	0.24310	0.37796
45 – 49	0	0.19785
50 – 59	-0.05107	0
≥ 60	-0.48660	-0.42951
b_T Coeficiente tensión arterial (mmHg)		
PAS < 120 PAD < 80	-0.00226	-0.53363
PAS <130 PAD < 85	0	0
PAS <140 PAD < 90	0.28320	-0.06773
PAS < 160 PAD < 100	0.52168	0.26288
PAS ≥160 PAD ≥100	0.61859	0.46573
b_D Coeficiente de diabéticos		
No	0	0
Sí	0.42839	0.59626
b_F Coeficiente fumadores		
No	0	0
Sí	0.52337	0.29246

Una vez calculado el valor *L*, debemos restarle la cantidad *G*.

$$G_H = 3.0975$$

$$G_M = 9.92545$$

Y exponenciamos ese valor, calculando *B=exp(L-G)*.

A continuación determinamos el valor $R = 1-S^{B}$, donde S es:

$$S_H = 0.90015$$

$$S_M = 0.96246$$

R será la probabilidad de sufrir un accidente cardiovascular en los siguientes 10 años.

También existe la posibilidad de calcular el riesgo mediante un modelo que utilice el valor LDL de colesterol, siendo el mismo procedimiento, modificando los coeficientes.

En la siguiente tabla podemos observar los valores medios obtenidos por el estudio para diferentes rangos de edades.

Edad	**Mujeres**	**Hombres**
30 - 34	< 1 %	3 %
35 - 39	< 1 %	5 %
40 - 44	2 %	6 %
45 - 49	5 %	10 %
50 - 54	8 %	14 %
55 - 59	12 %	16 %
60 - 64	13 %	21 %
65 - 69	9 %	30 %
70 - 74	12 %	24 %

Cálculo del riesgo cardiovascular según el modelo SCORE

El proyecto SCORE, como bien hemos comentado, surgió por la necesidad de adaptar un modelo de cálculo de riesgo cardiovascular para pacientes de Europa. En él, se calcula la probabilidad de sufrir un accidente cardiovascular en los próximos 10 años, a través de dos ecuaciones diferentes, una para enfermedades coronarias y otra para las no coronarias, además de dos métodos de evaluación, uno de los cuales utiliza como parámetro el colesterol total y otro en el que se utiliza la relación colesterol/HDL. Se calculan por separado los riesgos de padecer enfermedad coronaria (EC) y no coronaria (ENC), y el riesgo total supondrá la suma de ambos. También se tiene en cuenta si el paciente vive en un país con alto o bajo riesgo de enfermedad cardiovascular. En este último grupo se incluyen España, Italia o Bélgica.

El procedimiento es el siguiente:

En primer lugar, calculamos la probabilidad de supervivencia para los diferentes métodos para la edad actual del paciente y a los 10 años, de acuerdo a las ecuaciones:

$$S_0(Edad) = \exp\{-\exp(a)\cdot(Edad-20)^p\}$$

$$S_0(Edad + 10) = \exp\{-\exp(a)\cdot(Edad-20)^p\}$$

Los valores de los coeficientes *a* y *p* son los siguientes:

		Enf. Coronaria		**Enf. No Coronaria**	
		a	**p**	**a**	**p**
Población de riesgo bajo	**Hombre**	-22.1	4.71	-26.7	5.64
	Mujer	-29.8	6.36	-31.0	6.62
Población de riesgo alto	**Hombre**	-21.0	4.62	-25.7	5.47
	Mujer	-28.7	6.23	-30.0	6.42

A continuación, calculamos el siguiente valor, para EC y ENC,

$$w = b_C(Colesterol - 6) + b_T(TAS-120) + b_F$$ [1]

donde los coeficientes bC y bT son:

[1] Nota: TAS (Tensión Arterial Sistólica)

	Enf.Coronaria	**Enf. No Coronaria**
b_C [mmol/l]	0.24	0.02
b_T [mmHg]	0.018	0.022
b_F (Sólo si es fumador)	0.71	0.63

Debemos tener en cuenta que el colesterol está medido en mg/dl y debemos convertirlo a mmol/l, la medida en la que está el coeficiente. Para ello, basta con multiplicar la cantidad por 0.02586. A continuación, calculamos la probabilidad de supervivencia con estos factores de riesgo a esa edad y a 10 años:

$$S(Edad) = S_0(Edad)^{\exp(w)}$$

$$S(Edad+10) = S_0(Edad+10)^{\exp(w)}$$

Ahora, para cada tipo de enfermedad calculamos la probabilidad de supervivencia a los 10 años dependiendo de la supervivencia a la edad actual:

$$S_{10}(Edad) = S(Edad+10)/S(Edad)$$

El riesgo para cada enfermedad será:

$$Riesgo = 1 - S_{10}(Edad)$$

Obtendremos entonces el riesgo de padecer una EC, R_{EC}, y el de paceder una ENC, R_{ENC}. El riesgo total será la suma de ambos:

$$R_T = R_{EC} + R_{ENC}$$

Cardiomanager

Cardiomanager es una aplicación para dispositivos móviles dotados del sistema operativo Android, englobada dentro del mundo de la *eSalud*, cuya finalidad es ayudar a los pacientes de enfermedades cardiovasculares a mejorar su calidad de vida, conocer más datos sobre su enfermedad y llevar un control exhaustivo sobre la medicación, las actividades que realizan y otras características que puedan tener relación con su enfermedad. Sin embargo, puede ser también utilizada por cualquier usuario para informarse sobre enfermedades cardiovasculares, para llevar un control de las tomas de cualquier medicamento o para averiguar su nivel de riesgo de sufrir un accidente cardiovascular.

Control de actividades

A través de este módulo, el paciente puede introducir diferentes actividades relacionadas con su enfermedad, como son:

- ✓ Rehabilitación.
- ✓ Actividad física.
- ✓ Análisis de sangre.
- ✓ Presión sanguínea.
- ✓ Medida de glucosa.
- ✓ Ataques o crisis.
- ✓ Excesos.

Al registrar una jornada de rehabilitación, puede almacenar la fecha y una breve descripción sobre la misma, mientras que en la actividad física también puede añadir la duración de la misma y una descripción sobre el ejercicio realizado. En la secciones de análisis de sangre, presión sanguínea o medida de glucosa, se pueden introducir los datos obtenidos de diferentes análisis realizados por el médico. Al introducir un ataque o crisis, se puede seleccionar la ubicación para llevar un control de las mismas, y en excesos también se puede especificar el tipo de exceso y una descripción del mismo.

Todos estos datos pueden ser observados en un calendario, que el paciente puede enseñar a su médico para que el mismo compruebe como ha llevado el paciente la enfermedad, o para que el mismo paciente observe las actividades realizadas y cambie sus hábitos si es necesario.

Además, los diferentes análisis, sanguíneo, de presión y de glucosa, pueden ser visualizados en diferentes gráficas que permiten comparar unos resultados con otros y obtener conclusiones.

Control de medicamentos

El usuario puede programar la aplicación para que le avise en el momento que deba tomarse un determinado medicamento. El paciente puede seleccionar entre diversos tipos de tratamiento y programar la toma a cualquier hora. Además, puede escoger si desea que se muestren notificaciones o no. Una vez llegada la hora de la toma, el paciente podrá clasificarla como tomada o no tomada.

Además, la aplicación construirá un calendario con los diferentes medicamentos y las diferentes tomas, se hayan producido o no, permitiendo llevar un control sobre todos los medicamentos y sobre si se ha cumplido con las horas de tomas y el número de dosis.

Información y educación

La aplicación también dispone de amplias secciones en las que se informa sobre las distintas enfermedades cardiovasculares, mostrando qué es lo que son, cuáles son sus síntomas y su tratamiento, y dando directrices sobre hábitos saludables, consejos o prohibiciones.

Información del paciente

El paciente puede introducir su información personal, que será almacenada con seguridad en la aplicación y manteniendo la privacidad del paciente. Además, podrá registrar otro tipo de información, como alergias, hábitos tóxicos, enfermedades, antecedentes quirúrgicos o transfusiones.

Calculadora de riesgo cardiovascular

A través de este módulo, el paciente o cualquier usuario podrán, introduciendo diversos datos, averiguar su riesgo de sufrir un accidente cardiovascular en los próximos 10 años. Podrá escoger entre dos métodos, el de Framingham, para pacientes de Norteamérica, y el método SCORE, para pacientes de Europa. Aun así, se recomienda realizar el cálculo con los dos métodos y compararlos.

Además, se podrá almacenar el resultado y visualizar una lista con los diferentes cálculos, para comprobar si se ha aumentado o disminuido el riesgo cardiovascular.

BLOQUE 2
CARDIOLOGÍA Y TECNOLOGÍAS DE LA INFORMACIÓN Y LA COMUNICACIÓN (TICs)

eSalud

*"La **eSalud** se define como la aplicación de las Tecnologías de Información y Comunicación (TIC) en el amplio rango de aspectos que afectan el cuidado de la salud, desde el diagnóstico hasta el seguimiento de los pacientes, pasando por la gestión de las organizaciones implicadas en estas actividades. En el caso concreto de los ciudadanos, la eSalud les proporciona considerables ventajas en materia de información, incluso favorece la obtención de diagnósticos alternativos. En general, para los profesionales, la eSalud se relaciona con una mejora en el acceso a información relevante, asociada a las principales revistas y asociaciones médicas, con la prescripción electrónica asistida y, finalmente, con la accesibilidad global a los datos médicos personales a través de la Historia Clínica Informatizada".* [3]

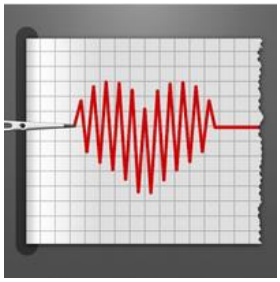
Como bien trata esta definición, podemos definir *eHealth*, o *eSalud* en castellano, como la aplicación de las TIC en la medicina y la salud en general. El objetivo no es otro que mejorar la calidad de vida de los pacientes y ayudar a los profesionales de la salud, bien sean doctores, enfermeros o personal administrativo, a realizar mejor sus funciones. Algunos servicios que engloba la *eSalud* son:

- ✓ **Historiales médicos electrónicos**. Facilitando las labores de administración dentro de un mismo centro o entre diferentes centros.
- ✓ **Telemedicina** o medicina a distancia. Permite a los pacientes realizar consultas o someterse a diversas pruebas en tiempo real sin tener que desplazarse hasta el hospital.
- ✓ **Difusión** de información orientada al ciudadano
- ✓ **Difusión** de información orientada al especialista
- ✓ **Equipos virtuales** de cuidados sanitarios, formados por profesionales de la salud que discuten y colaboran entre sí.
- ✓ **Dispositivos de detección** de caídas o accidentes que avisan rápidamente a los servicios de emergencia.
- ✓ **Dispensador** electrónico de medicamentos.
- ✓ **eDiagnostic**, que permite la detección de enfermedades y la realización de pruebas a distancia, englobando la *telecardiología*, con la recogida de la presión arterial y el pulso del paciente, la *teledermatología*, que permite mostrar al médico lesiones cutáneas a distancia, *telepatología*,…

Otras características reseñables de la *eSalud* son que fomenta la **investigación**, permite que la información sobre enfermedades llegue antes a los pacientes y a otras personas que puedan llegar a sufrir ciertas enfermedades, mejorando así la **educación** de la población en general, mejora la **seguridad** de los pacientes, pues el médico puede realizar un mejor, y en muchos casos, ayuda a una **detección** más rápida de las enfermedades.

Plataformas y aplicaciones para dispositivos móviles en el campo de la *eSalud* [4]

A continuación, se muestran 9 aplicaciones para dispositivos con Android, entre otros, que pueden ser de utilidad para pacientes con cardiopatías.

<table>
<tr><th colspan="2">Cardiógrafo - Cardiograph</th></tr>
<tr><td>Breve descripción</td><td> 
Aplicación para Android (gratuita) y iPhone (de pago) que se encarga de medir la frecuencia cardiaca del usuario.

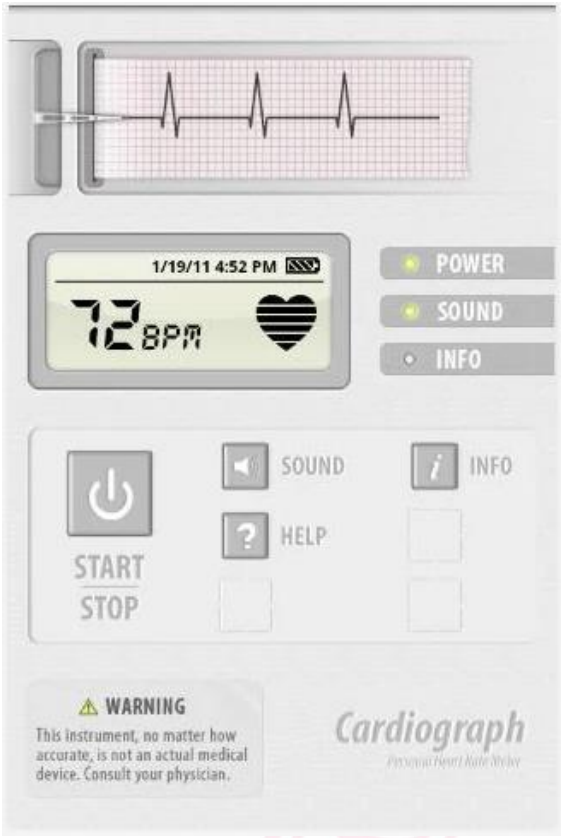
El procedimiento que sigue para ello es realizar fotografías de la yema del dedo colocando la misma en la cámara del dispositivo, observando los cambios en la misma y calculando así el ritmo cardiaco.

Permite registrar diversos usuarios y almacenar, para cada uno de ellos, un historial de mediciones diferente para que los pacientes puedan realizar un seguimiento.</td></tr>
<tr><td>Opinión</td><td>Esta aplicación ha sido descargada más de 10.000.000 veces desde la plataforma Google Play, lo que nos indica que es conocida y aceptada por muchos usuarios.

Dentro de sus fortalezas podemos destacar que su interfaz es muy atractiva, simple y muy lograda. Además de soportar diversos usuarios e historiales.

Como debilidades se encuentran el sonido que se produce mientras se detecta el pulso, que puede llegar a ser un poco desagradable, la carencia de vista apaisada o el banner de publicidad en la parte inferior, que dependiendo de los usuarios, puede llegar a molestar. Sin embargo, el mayor problema que tiene esta aplicación es que su método no sólo es inexacto (detectada una tolerancia del ±10%) sino que es complicado llevarlo a cabo en algunos dispositivos, pues a veces ni se detectan las pulsaciones y es necesario repetir varias veces la medida.

Puede que los pacientes con enfermedades del corazón encuentren utilidad a esta aplicación, pues es importante para ellos tomarse el pulso de vez en cuando y realizar un seguimiento de las mediciones. Sin embargo, podemos calificar esta aplicación de escasa para estos pacientes a la hora de afrontar la enfermedad, pues no sólo se deben realizar seguimientos de</td></tr>
</table>

	la frecuencia cardiaca en las enfermedades cardiovasculares, sino que se suelen tener en cuenta otros muchos parámetros.
Capturas de pantalla	

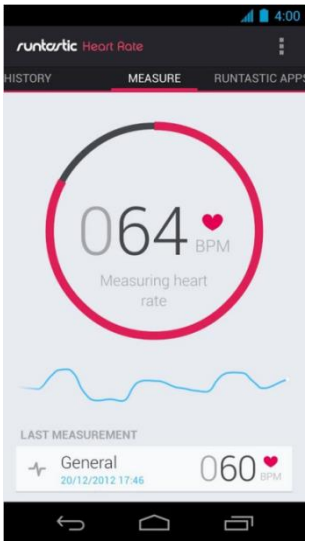
<table>
<tr><th colspan="2">Runtastic Heart Rate</th></tr>
<tr><td>Breve descripción</td><td> 
Aplicación similar a Cardiograph, que permite al paciente medir su frecuencia cardiaca y consultar un historial de mediciones.

El mecanismo utilizado para ello también es el mismo, situando la yema del dedo del paciente en la cámara del dispositivo.

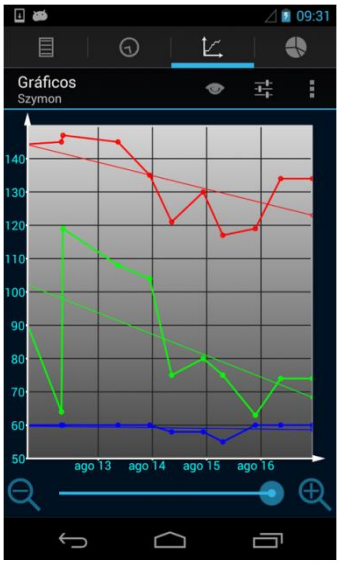
Existe una versión de pago en la que se mejoran los servicios de la misma.</td></tr>
<tr><td>Opinión</td><td>Esta aplicación no ha sido descargada tantas veces como la anterior, principalmente porque supera en debilidades a la misma. Aunque soluciona el molesto sonido, sólo se pueden realizar un número determinado de tomas, por no hablar de que sólo se puede registrar un único paciente. El aspecto más molesto de la aplicación es sin duda, el hecho de que sea necesario registrarse con Facebook o con un correo electrónico, tanto para la versión gratuita como para la versión mejorada de pago, pues no aporta ninguna utilidad a la aplicación.

Podemos obtener las fortalezas de la aplicación en la versión de pago, que nos permite un número ilimitado de medidas, habilita el uso de filtros, muestra notificaciones y suprime los anuncios. Además, la interfaz gráfica de ambas versiones es sencilla e intuitiva.

Al igual que la aplicación anterior, es escasa para la realización de un seguimiento de una enfermedad cardiovascular.</td></tr>
</table>

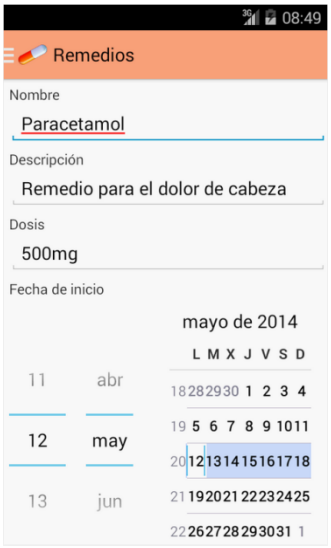
Capturas de pantalla	 
Presión Arterial (My Heart)	
Breve descripción	 Interesante aplicación que permite controlar las presiones arteriales sistólica y diastólica. Está orientada a usuarios con **hipertensión**. A través de un medidor de presión se recogen los datos y se introducen manualmente en la aplicación, que se encarga de analizarlos y almacenarlos. Además, a partir de estos datos, la misma aplicación genera información para el usuario sobre cómo tratar mejor su enfermedad y mejorar su calidad de vida. Estos informes pueden ser enviados a un médico especialista.
Opinión	La única **desventaja** de esta aplicación es que para eliminar el banner de anuncios debemos pagar una pequeña cantidad de dinero. Además, existe la necesidad de un dispositivo externo para medir la presión, algo de lo que no son capaces los dispositivos móviles hoy en día. Por lo demás, se trata de una aplicación muy completa y con grandes **fortalezas**. A la hora de introducir una medición, permite especificar distintas etiquetas informando sobre la misma, y en las secciones de gráficos, estadísticas e historial, además de mostrar estas mediciones, se permite filtrar la información que se muestra por fecha, hora, tipo de presión o pulso. También existe la opción de especificar un email al que enviar un informe con los datos que se deseen o una ruta dentro del teléfono donde almacenar el mismo. Por último, posee un módulo de '*reminders*' para recordar al usuario distintas acciones relacionadas con la

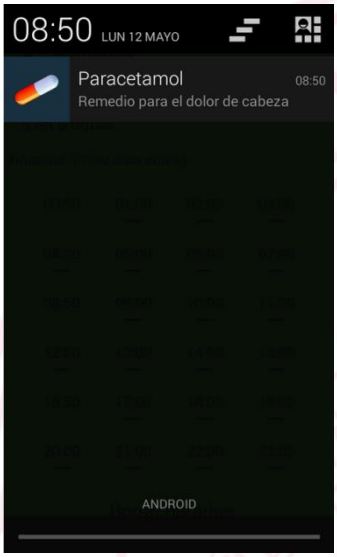
	enfermedad y la aplicación. Por lo tanto, podemos calificar esta aplicación como muy recomendable para usuarios que padezcan *hipertensión*, pues les ayuda a tratar mejor su enfermedad y a informar de manera continuada a su médico, que podrá realizar un seguimiento de la enfermedad mucho más cercano.

Capturas de pantalla	    

Hora de la medicación!	
Breve descripción	 A través de esta aplicación, el usuario introduce una serie de horas en las que deba tomar un determinado medicamento. Cuando llegue el momento, la aplicación le avisará mediante una notificación, indicando qué medicamento debe tomar e información sobre la dosis.
Opinión	La principal **ventaja** de esta aplicación es que su funcionamiento es muy sencillo de entender y que presta un gran servicio. Además, es útil para cualquier paciente de cualquier enfermedad, no se limita a las cardiovasculares. Dentro de los **defectos** que podemos encontrar, el más reseñable es que visualmente, está muy poco trabajada. El estilo utilizado es el estilo por defecto en programación Android. Podemos decir que es una aplicación que nos presta un gran servicio, pues nos avisa de las tomas y previene de posibles olvidos. Sin embargo, no lleva ningún tipo de registro sobre si los medicamentos han sido tomados o no, simplemente se dedica a avisar.

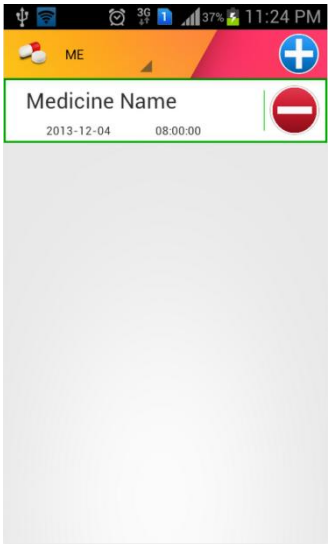
Capturas de pantalla

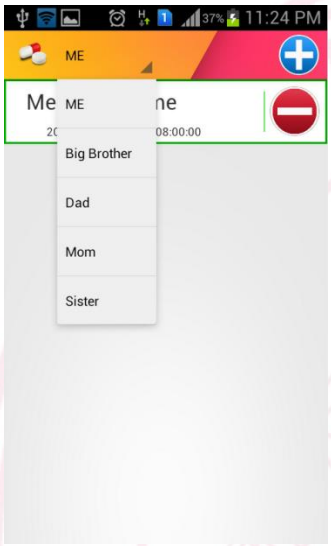


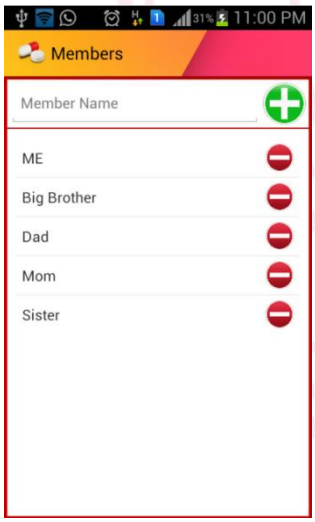


Medicine Tracker	
Breve descripción	 Al igual que la aplicación anterior, nos avisa cuando sea el momento de tomar un medicamento, según lo que hayamos introducido en la aplicación.
Opinión	Respecto a la aplicación anterior, la interfaz gráfica está mucho más trabajada, y es todavía más simple a la hora de programar las tomas. Su gran **ventaja** es que permite seleccionar qué usuarios tienen que tomar cada medicamento, y que se permite elegir si mostrar notificación o no. Sin embargo, su gran **desventaja** sigue siendo la ausencia de un registro de tomas.

Capturas de pantalla







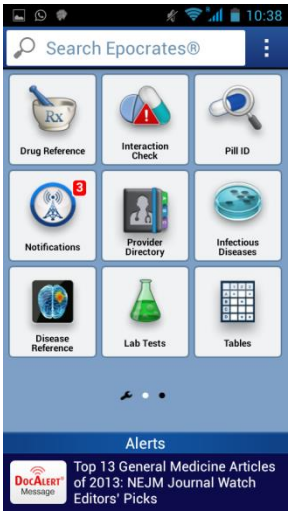
<table>
<tr><th colspan="2">Epocrates</th></tr>
<tr><td>Breve descripción</td><td> 
Esta aplicación, disponible para Android y iPhone, nos proporciona gran cantidad de información sobre cualquier enfermedad, incluyendo por supuesto, las cardiovasculares. Dentro de sus distintas secciones, podemos encontrar un registro de medicamentos, en el que observamos las dosis aconsejadas para adultos, animales, niños, medidas de seguridad, efectos secundarios, etc, todos ellos ordenados por tipo de enfermedad. Además, contiene también un comprobador de compatibilidad entre medicamentos (hasta 30 al mismo tiempo) que nos avise si determinada mezcla puede ser perjudicial, y un identificador de medicamento, que nos dice el nombre de dicho medicamento a partir de una serie de características que el usuario introduzca. Por último, incluye varios módulos con mucha información sobre enfermedades y tipos de pruebas en laboratorio y tablas comparativas entre medicamentos que tratan las mismas enfermedades, además de una calculadora de diferentes índices médicos.</td></tr>
<tr><td>Opinión</td><td>De sus ventajas podemos destacar que todo lo anteriormente citado está disponible en la versión gratuita, y que para los pacientes pueden ser muy útiles tanto el identificador de medicamentos como el comprobador de compatibilidad entre los mismos, ya que el doctor puede no haber tenido en cuenta los medicamentos que está tomando su paciente a la hora de recetarle unos nuevos.

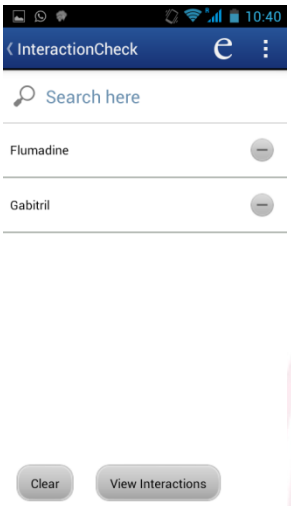
Como desventajas podemos mencionar la larga espera durante la instalación de la aplicación y la descarga de la información, y la necesidad de registrarse para poder hacer uso de la misma.

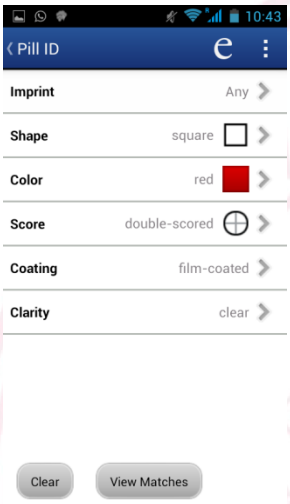
Se trata de una aplicación muy completa, con muchísima información, no solo para los pacientes, sino también para los doctores. Muy recomendable para pacientes de cualquier tipo de enfermedad.</td></tr>
</table>

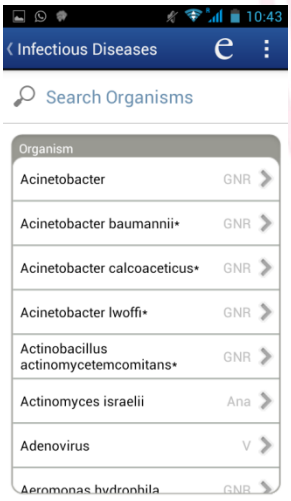
Capturas de pantalla

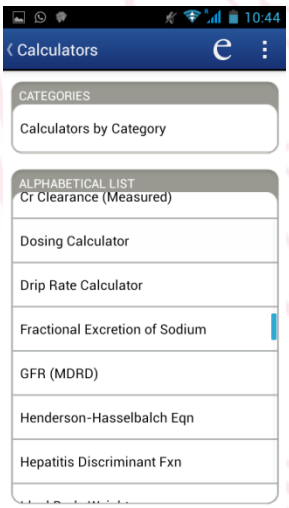
EPOCRATES
an athenahealth company





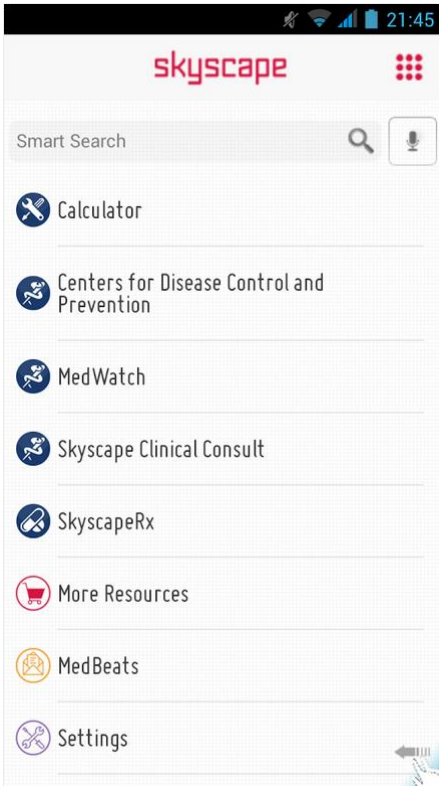


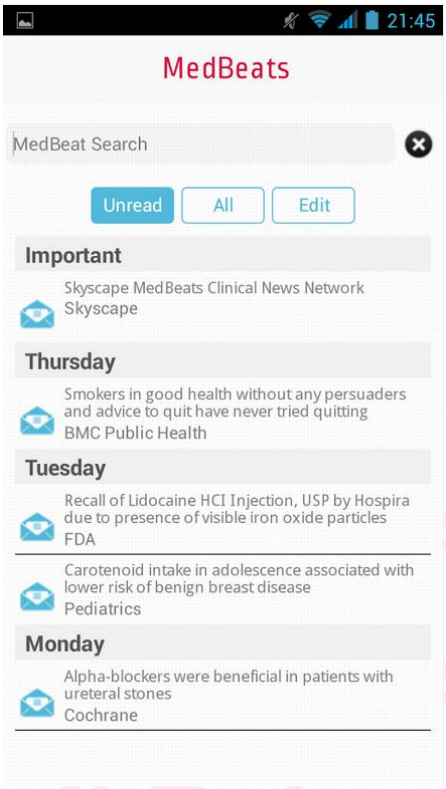


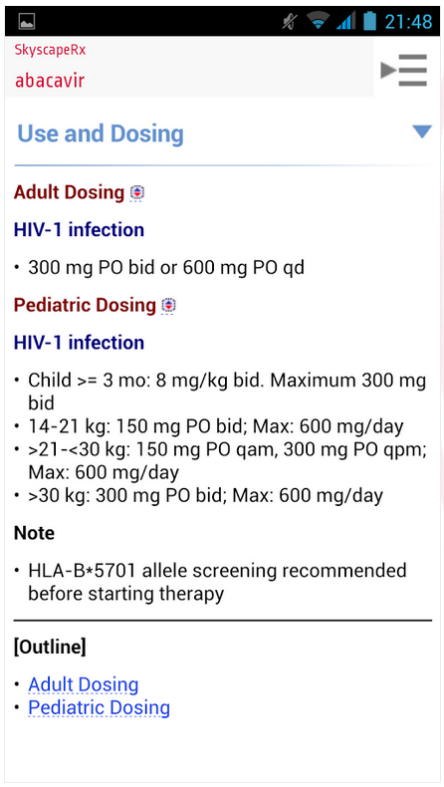


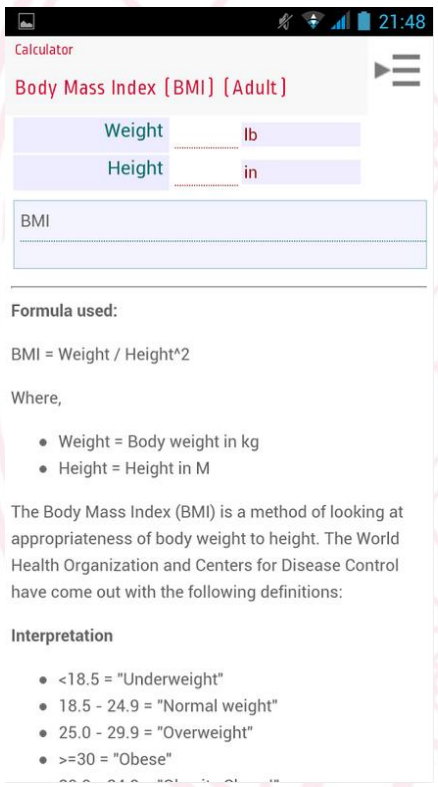
Skyscape Medical Library	
Breve descripción	 Aplicación para Android y iPhone que en su versión gratuita nos proporciona una amplia sección de información sobre medicamentos de distintas marcas, con imagen de los mismos, un comprobador de compatibilidad de medicamentos, una calculadora de dosis, una sección de información sobre enfermedades y una bandeja de noticias relacionadas con la medicina. En su versión de pago añade múltiples servicios como un atlas médico, un manual de enfermería para médicos y enfermeros o el módulo 5-Minute Clinical Consult(5MCC), 5 minutos de consulta clínica, entre otros.
Opinión	Como **puntos débiles** podemos decir que es una aplicación similar a Epocrates, pero que en su versión gratuita se queda un poco escasa de información y no tiene tantas calculadoras. También es necesario registrarse para hacer uso de la aplicación. Los puntos fuertes los podemos encontrar en la versión de pago, pues nos proporciona elementos originales y que pueden ser de gran utilidad, como el 5MCC. En general, si el paciente no desea optar por la versión de pago, no es muy recomendable su uso, ya que Epocrates supera de largo a esta aplicación en su versión gratuita.

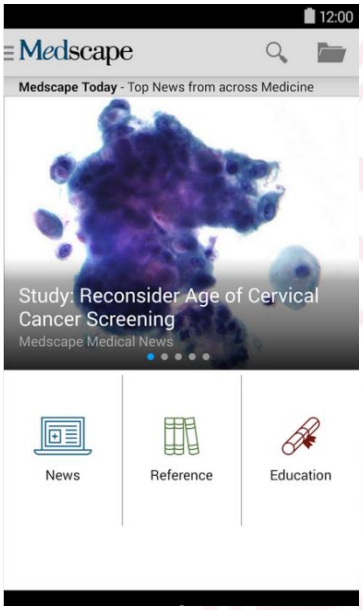
Capturas de pantalla



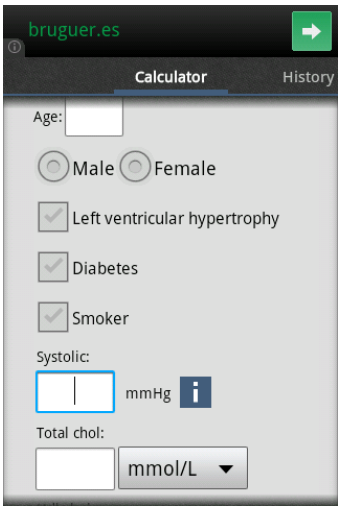
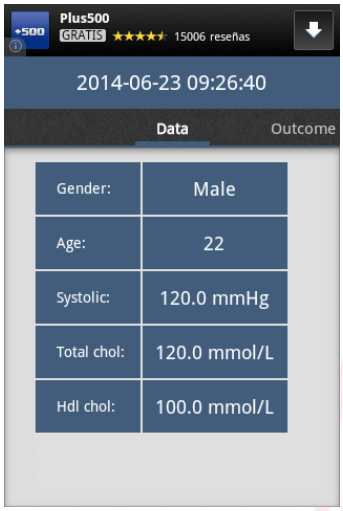
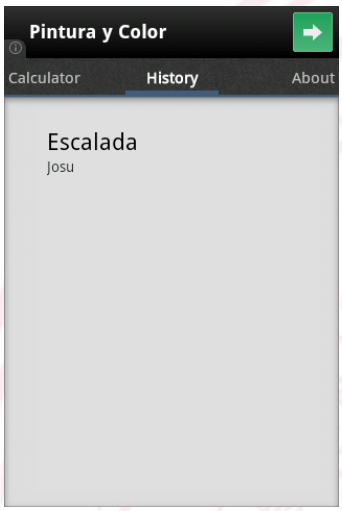
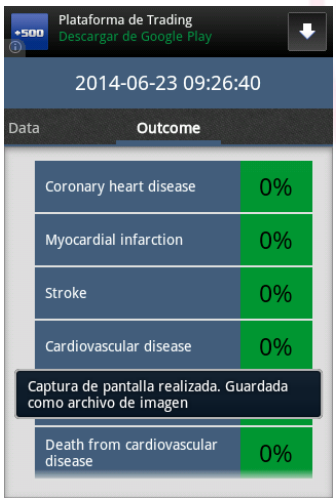
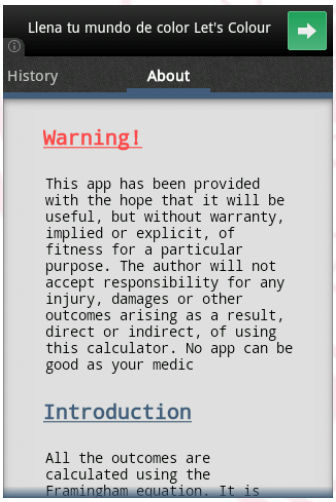






Medscape	
Breve descripción	Aplicación muy similar a las anteriores que nos proporciona noticias sobre medicina, información de enfermedades y medicamentos y varias calculadoras médicas.
Opinión	Es muy similar a las anteriores, por lo tanto, sus ventajas y debilidades son parecidas. Destacar que su interfaz gráfica es la más simple y agradable de las tres.
Capturas de pantalla	

Cardiac risk calculator	
Breve descripción	 Calculadora del riesgo cardiovascular muy simple que permite además almacenar las diferentes pruebas para la persona que queramos.
Opinión	Su principal **ventaja** es que es muy simple y es fácil de entender. Sin embargo, las **desventajas** son múltiples para una aplicación tan pequeña. No especifica el método que sigue para realizar el cálculo de riesgo cardiovascular, la interfaz gráfica en general no está muy lograda y tiene un banner de publicidad en la parte superior algo molesto. Por lo tanto, podemos resumir que para el cálculo del riesgo cardiovascular, es recomendable buscar plataformas web, mucho más acertadas y completas que esta aplicación.

Capturas de pantalla	     

Análisis de los sistemas operativos para dispositivos móviles [5][6]

Recientes estudios sobre el uso de dispositivos móviles, realizados por distintos organismos y plataformas a nivel nacional e internacional, nos conducen hacia la misma conclusión, bastante evidente por otra parte: el uso de los mismos entre la población se ha incrementado en los últimos años de una forma muy importante, y son cada vez más comunes en nuestras vidas. Y con ello, no solo crece el número de terminales, sino que también crecen el número de aplicaciones para los mismos y de personas dedicadas al desarrollo de las mismas, con el objetivo de ofrecer nuevos servicios a los usuarios.

Cada fabricante de móviles y tablets decide qué sistema operativo desea implantar en sus dispositivos. Por esta razón, los desarrolladores de aplicaciones para móviles se deben tener en cuenta cuales son los sistemas operativos más utilizados, para realizar un correcto estudio de mercado de su aplicación.

Actualmente, a nivel internacional, según los últimos informes de la consultora IDC, el dominio de Android es indiscutible. Basta con analizar los dispositivos que tenemos a nuestro alrededor para darnos cuenta de ello, pues el sistema operativo de Google sin duda arrasa, y supera con bastante ventaja al resto de sistemas más comunes, como puedan ser iOS, Windows Phone, Blackberry OS, Firefox OS o Symbian OS. La siguiente tabla muestra algunos datos obtenidos del estudio anteriormente mencionado.

Sistema Operativo	Número de dispositivos (millones de unidades) 2013	Cuota de Mercado 2013	Número de dispositivos (millones de unidades) 2012	Cuota de Mercado 2012	Incremento
Android	793.6	78.6%	500.1	69.0%	58.7%
iOS	153.4	15.2%	135.9	18.7%	12.9%
Windows Phone	33.4	3.3%	17.5	2.4%	90.9%
BlackBerry	19.2	1.9%	32.5	4.5%	-40.9%
Otros	10.0	1.0%	39.3	5.4%	-74.6%
Total	1009.6	100.0%	725.3	100.0%	39.2%

En primer lugar, podemos observar un claro incremento del número de dispositivos. Un 39.2% concretamente, lo cual es bastante significativo y muestra la clara importancia de estos objetos en la vida de las personas.

En segundo lugar, vemos el claro dominio de Android dentro del mercado de los sistemas operativos, con casi el 80% de la cuota de mercado, y un notable aumento respecto a los datos de 2012. En segunda posición del ranking se encuentran los dispositivos fabricados por Apple, que hacen uso de iOS. Mientras que ha bajado la cuota de mercado, el número total de dispositivos ha aumentado. Curioso el crecimiento de Windows Phone, subiendo 0.9 puntos de

cuota de mercado y doblando casi el número de terminales con este sistema. También destacable el tremendo bajón de Blackberry, que sigue perdiendo cuota día a día. Dentro de otros sistemas operativos, cabe destacar el nuevo Firefox OS, de muy reciente creación y que ha cogido fuerza, sobre todo en América Latina, aunque su cuota general de mercado sigue siendo muy baja.

A nivel nacional, según revelan los datos de un informe sobre aplicaciones móviles realizado por The App Date, en España existen unos 23 millones de usuarios que cuentan con smartphones o tablets y se descargan 4 millones de aplicaciones diarias, lo cual está obligando a las empresas y a los desarrolladores de software a adaptar sus productos a las nuevas necesidades y costumbres de los usuarios. En lo que se refiere a sistemas operativos, no hay discusión. El dominio de Android en España es todavía mayor que a nivel mundial, con una cuota de mercado del 90%, muy superior a iOS con 4,8% o Windows Phone, con 3,7%.

Es por ello, por lo que surge la necesidad de crear aplicaciones y plataformas para proporcionar servicios a los usuarios, concretamente servicios relacionados con el campo de la telemedicina. El objetivo es aprovechar el alto grado de uso de los dispositivos móviles para mejorar la calidad de vida de pacientes de distintas enfermedades. Poniéndonos en la piel de un desarrollador, es evidente que a la hora de escoger un sistema operativo para el que construir las aplicaciones, Android es la primera opción a escoger, debido a su alto acoplamiento en el mercado, desarrollando aplicaciones nativas, que nos permiten acceder a los más bajos niveles del sistema y explotar todo lo que nos ofrece. Como segunda opción, existe la posibilidad de desarrollar aplicaciones con HTML5, CSS y JavaScript con PhoneGap. A través de esta opción podemos construir aplicaciones para todo tipo de sistemas operativos: Android, Windows Phone, iOS o Firefox OS, sin tener que modificar muchas líneas de código, algo que no conseguiríamos programando nativamente para cada uno de los sistemas.

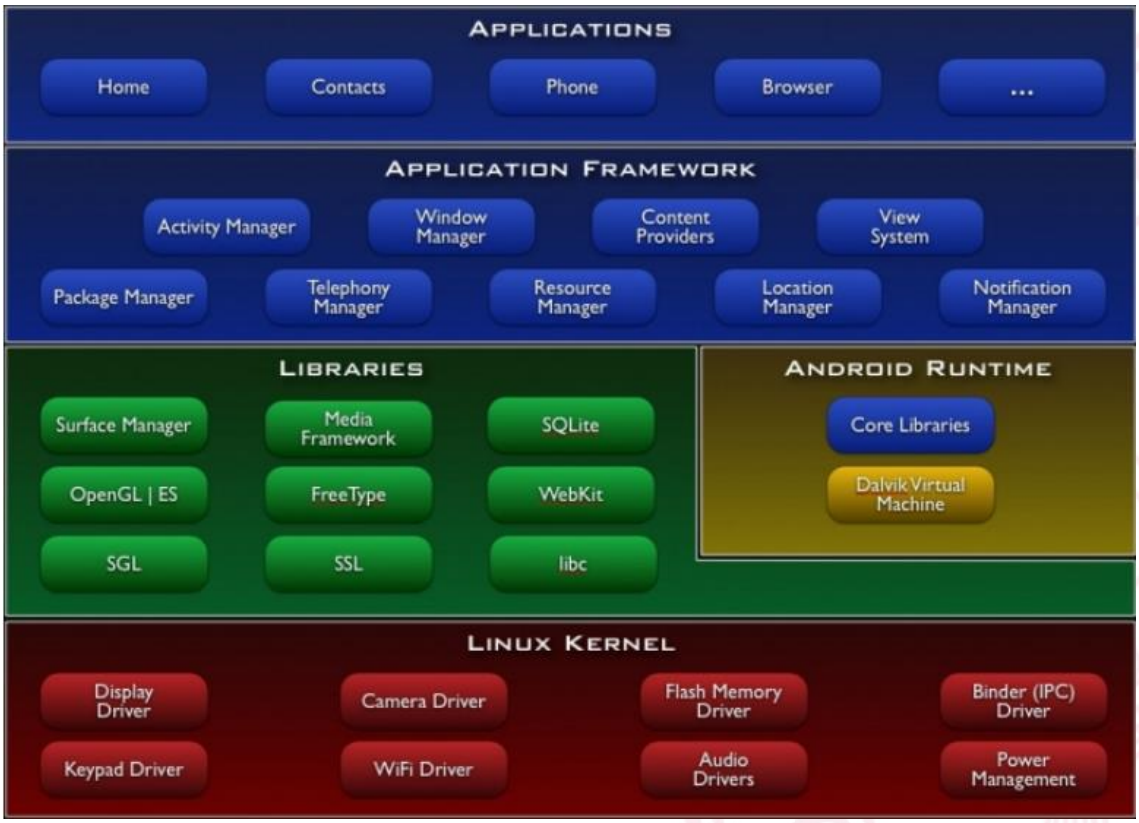
Android OS [7]

Android es un sistema operativo creado para ser implantado en dispositivos móviles con pantalla táctil, como *smartphones* y *tablets*, desarrollado por Android, Inc., y comprado por Google más tarde.

El sistema Android consiste en:

- ✓ Sistema operativo (basado en Linux)
- ✓ Entorno de ejecución de aplicaciones basado en Java
- ✓ Conjunto de bibliotecas de bajo y medio nivel (accesibles por las aplicaciones)
- ✓ Conjunto inicial de aplicaciones para el usuario final.

La siguiente imagen muestra la pila de componentes de Android:



Applications:

- ✓ Aplicaciones básicas escritas en Java como son los contactos, teléfono, navegador, …

Application framework: herramientas que se encargan de la gestión de las aplicaciones.

- ✓ Activity Manager: para el ciclo de vida de las aplicaciones.
- ✓ Window Manager: gestiona que actividad se muestra en primer plano y cuáles en *backgroung*.
- ✓ Content Providers: permite a las aplicaciones hacer uso de datos de otras aplicaciones, como puedan ser los contactos o la agenda.
- ✓ View System: elementos que forman la interfaz gráfica de usuario (GUI).

- ✓ Package Manager: para la obtención de información sobre las aplicaciones instaladas en el dispositivo, como los permisos requeridos o el espacio necesario.
- ✓ Telephony Manager: para la gestión de llamadas y mensajería.
- ✓ Resource Manager: gestión de acceso a recursos.
- ✓ Location Manager: para la obtención de la ubicación geográfica del dispositivo.
- ✓ Notification Manager: permite mostrar al usuario mensajes sobre el dispositivo o sobre otras aplicaciones.
- ✓ Alarm Manager: permite establecer alarmas que avisen a otras aplicaciones o al dispositivo sobre cierto evento.

Libraries:

- ✓ OpenGL ES/ SGL: gráficos 3D y 2D.
- ✓ Media Framework: grabación y reproducción de audio y video.
- ✓ SQLite: gestión de bases de datos.
- ✓ Free Type: soporte para distintos tipos de fuentes.
- ✓ SSL: gestión de comunicaciones seguras.
- ✓ WebKit: soporte para aplicaciones de tipo navegador, es decir, las no desarrolladas nativamente.

Android Runtime:

- ✓ Core libraries: permite que las bibliotecas anteriormente descritas sean accesibles desde el entorno Java.
- ✓ Dalvik VM: máquina virtual Java optimizada para los dispositivos móviles que llevarán Android.

Linux Kernel:

- ✓ Kernel de Linux 2.6.
- ✓ Incluye, entre otros, controladores hardware, gestor de memoria, gestor de batería o un gestor de procesos.

Ventajas del uso y programación de aplicaciones Android

- ✓ El código abierto bajo licencia Apache.
- ✓ Más de 100.000 aplicaciones disponibles para Android, la mayoría de forma gratuita.
- ✓ Es el sistema operativo para dispositivos móviles más utilizado del mundo, como ya vimos en el anterior estudio.
- ✓ Puede ser instalado en cualquier dispositivo, gracias al kernel de Linux y al entorno de ejecución de aplicaciones Java.
- ✓ Independencia del operador. Aunque los operadores diseñen aplicaciones propias para sus usuarios, lo cierto es que el funcionamiento de Android no tiene que ver nada con los mismos, lo cual es un punto a favor de los usuarios, que simplemente escogerán su operador de acuerdo a los servicios de red que ofrezcan, y no de otro tipo.

- ✓ Coste cero. Los usuarios no tienen que pagar costes extras por la instalación de este sistema operativo en sus teléfonos.
- ✓ Personalizable. Al ser de código abierto y libre, cada fabricante de dispositivos puede crear su propia GUI, y además, los propios usuarios pueden elegir entre infinidad de temas, fondos de pantalla, animaciones o widgets.
- ✓ No sólo está disponible para teléfonos móviles o tablets, sino que también está disponible para netbooks, microondas, lavadoras, navegadores GPS o relojes.
- ✓ Multifunción y escalable.
- ✓ Comunidad Android. Este sistema operativo cuenta con la mayor comunidad de desarrolladores del mundo, que incluye eventos, concursos, competiciones, colaboración entre programadores, foros o debates, todos ellos al alcance de todo el mundo. Además, las mejoras del mismo no son fruto del trabajo de una simple empresa, sino de la participación y colaboración de desarrolladores de todo el mundo.
- ✓ Multitarea. Android es capaz de mantener abiertas varias aplicaciones realizando a su vez gestión de memoria, suspendiendo las aplicaciones que no están en uso y si es necesario, cerrándolas tras un determinado tiempo de inactividad.

Desventajas

- ✓ Multitarea. Si bien está dentro de la sección de ventajas, ésta se trata de un arma de doble filo. Android gestiona la memoria y cierra las aplicaciones que no se están usando, sin embargo, no siempre cierra esas aplicaciones, por lo que es necesario el uso de una aplicación extra que las cierre y que por tanto, se añade a la lista de aplicaciones en ejecución. Además, el mantener muchas aplicaciones ejecutándose en segundo plano, aumenta seriamente el consumo de las baterías, problema creciente en los últimos años en el mundo de los dispositivos móviles.
- ✓ Al permitir a los fabricantes añadir su propia GUI, los usuarios tienen que convivir con infinidad de aplicaciones que no desean y que no se pueden desinstalar, o que realizan mal sus funciones. Al menos, podemos ignorarlas y buscar alternativas dentro de las muchas aplicaciones gratuitas que hay en el mercado.
- ✓ Muy fragmentado. Demasiadas versiones en muy pocos años, más concretamente, 35 versiones desde el 23 de septiembre de 2008 al 19 de junio de 2014. Una media de aproximadamente 6 versiones al año. Esto provoca incompatibilidades con algunas aplicaciones, que sólo funcionan para un determinado número de versiones. Por ello, se recomienda a los desarrolladores realizar diseños adaptativos, que exploten lo que nos ofrece cada nueva versión, sin menospreciar las versiones más antiguas con menos servicios.

BLOQUE 3
METODOLOGÍA

Herramientas

En esta sección se mostrarán las cuatro herramientas utilizadas en todo el proceso de desarrollo de la aplicación.

Eclipse ADT Bundle

El paquete Eclipse ADT Bundle incluye todas las herramientas necesarias para desarrollar una aplicación Android de forma nativa. Este paquete ingluye:

- ✓ Eclipse. Programa informático que proporciona una serie de herramientas para el desarrollo de software, utilizado en todo el mundo. No sólo permite desarrollo de aplicaciones en Android, también permite realizar programas en Java, C++ o creación de sitios web.
- ✓ ADT Plugin (*Android Development Tools*). Complemento necesario para que podamos desarrollar aplicaciones Android con Eclipse. Incluye las bibliotecas propias para el desarrollo de Android.
- ✓ Android SDK Tools (*Software Development Kit*). Incluye más herramientas para compilar el código, depurar la aplicación, o probarla en dispositivos virtuales, los llamados *Android Virtual Devices*(ADV). Incluye unas herramientas diferentes para cada versión disponible de Android.

Aunque no forma parte del paquete, es necesario tener instalado un entorno de desarrollo de Java, *Java Development Kit* (JDK), en nuestro ordenador.

GIMP

GIMP es un programa de edición de imágenes, muy similar a Adobe Photoshop, pero libre y gratuito. A través de este programa se han podido modificar los distintos iconos incluidos en la aplicación.

WinMerge

WinMerge nos permite comparar ficheros con código, bien sean de distinta procedencia o de diferentes versiones, y mezclarlos o escoger las líneas adecuadas. Se ha utilizado este programa para la corrección de los ficheros de `strings.xml`, que más tarde serán explicados, y para su traducción al castellano.

Notepad++

Notepad++ es un editor de texto y de código fuente libre que permite escribir ficheros prácticamente en cualquier lenguaje de programación de uso actual. Se ha utilizado para editar algunos ficheros XML y para realizar búsquedas de cadenas en varios ficheros al mismo tiempo y reemplazarlas o eliminarlas.

Diseño de la aplicación

Para individualizar el desarrollo de las distintas funcionalidades de la aplicación y poder avanzar al mismo tiempo en varias de ellas, se ha dividido la aplicación en varios módulos:

- ✓ Actividad principal
- ✓ Introducción de Actividades
- ✓ Registro de Actividades
- ✓ Introducción de Nuevo Medicamento
- ✓ Registro de Medicamentos
- ✓ Notificaciones y aviso de toma
- ✓ Información
- ✓ Directrices
- ✓ Información del paciente
- ✓ Calculadora de Riesgo Cardiovascular
- ✓ Ayuda

A continuación, se detallan las características de cada uno de estos módulos:

Actividad principal

En la página principal se mostrarán distintos iconos identificativos para acceder al resto de módulos. Los módulos de información del paciente y ayuda estarán disponibles a través del menú.

Además, deslizando la pantalla hacia la izquierda, aparecerá el número de la persona de contacto de emergencia y una opción de llamada y otra de envío de mensaje de texto.

Introducción de Actividades

En este módulo, el usuario podrá seleccionar introducir diferentes actividades relacionadas con su enfermedad. Para cada una de ellas, además, se mostrarán diferentes vistas que permitan seleccionar:

- ✓ Rehabilitación
  - o Fecha.
  - o Texto editable, donde indicar observaciones sobre la jornada de rehabilitación.
- ✓ Actividad física
  - o Fecha.

- Duración.
- Texto para describir la actividad realizada.

✓ Análisis de sangre

- Fecha.
- Corpúsculos (millones/uL).
- Hemoglobina (g/dL).
- Hematocritos (%).
- Anticoagulante INR (%).
- Urea (mg/dL).
- Creatinina (mg/dL).
- Colesterol HDL (mg/dL).
- Colesterol LDL (mg/dL).
- Triglicéridos (mg/dL).

✓ Presión sanguínea

- Fecha.
- Medida:
  - Sistólica (mmHg).
  - Diastólica (mmHg).
- Pulso.

✓ Glucosa

- Fecha.
- Medida (mg/dL).
- Hb A1c (%).
- Nombre del medicamento.
- Tipo: lenta, intermedia o rápida.
- Unidades: 0, 1, 2 ó 3.

✓ Ataque/Crisis

- Fecha.

- Duración.
- Texto editable para escribir los síntomas.
- Mapa para seleccionar la ubicación.

✓ Exceso

- Fecha.
- Tipo: alcohol, comida o tabaco.
- Texto editable para escribir diferentes observaciones.

Todas estas actividades serán almacenadas en la base de datos de la aplicación.

Registro de Actividades

En este módulo se mostrará un calendario donde se podrán visualizar las diferentes actividades introducidas por el usuario, además de diversas gráficas en las que comparar los resultados de los distintos análisis de sangre, de presión y de glucosa.

Introducción de Nuevo Medicamento

En este módulo, el usuario podrá introducir diferentes medicamentos que deba tomar. Podrá seleccionar las horas y los días de las tomas, incluso el intervalo de tiempo entre las mismas y el día de finalización del tratamiento.

Registro de Medicamentos

En este módulo, el usuario podrá, en primer lugar, ver toda la información referente a un medicamento. Además, podrá visualizar en un calendario las tomas de los distintos medicamentos, tanto las que ha realizado como las que no, que aparecerán en rojo.

Notificaciones y aviso de toma

La aplicación deberá avisar al usuario cada vez que sea la hora de realizar una toma. Además, pulsando en la notificación se accederá el registro de medicamentos, donde podremos marcar las dosis como tomadas o no tomadas. Si la aplicación es cerrada, deberá recuperar todas las tomas que hayan tenido lugar desde que fue cerrada hasta que fue de nuevo abierta.

Información

En este módulo se mostrará información médica relacionada con diferentes cardiopatías. Las opciones seleccionables de este módulo serán:

- ✓ Exceso de ejercicio físico
- ✓ Relación corazón-embarazo
- ✓ Cardiopatía isquémica
- ✓ Hipertensión

- ✓ Diabetes
- ✓ Enfermedad valvular
- ✓ Fibrilación auricular

Directrices

En este módulo se mostrarán guías y directrices relacionadas con la cardiología y enfermedades cardiovasculares. Las opciones seleccionables de este módulo serán:

- ✓ General.
- ✓ Corazón y deporte.
- ✓ Embarazo.
- ✓ Cardiopatía isquémica.
- ✓ Hipertensión.
- ✓ Diabetes.
- ✓ Enfermedad valvular.
- ✓ Fibrilación auricular.

Información del paciente

En este módulo el paciente introducirá sus datos personales, que deberán almacenarse cifrados para mantener la privacidad del mismo. Para ello, tendrá que registrarse, con un nombre de usuario y una contraseña. Además podrá añadir otra información personal de carácter médico que pueda ayudar en su enfermedad, como son:

- ✓ Alergias, hábitos tóxicos y enfermedades, que incluirán:
  - Nombre.
  - Descripción.
- ✓ Antecedentes quirúrgicos y transfusiones, que incluirán:
  - Nombre.
  - Descripción.
  - Fecha.
  - Lugar.

También en este módulo se podrá elegir cual será el número de contacto en caso de emergencia de entre todos los contactos del teléfono.

Calculadora de Riesgo Cardiovascular

En este módulo el usuario introducirá algunos datos personales que permitan calcular su riesgo de sufrir un accidente cardiovascular:

- ✓ Sexo.
- ✓ Edad.
- ✓ Colesterol.
- ✓ Presión.
- ✓ Diabetes.
- ✓ Tabaquismo.

A continuación, se mostrarán los resultados a los que se ha llegado a través de dos métodos diferentes, el de Framingham, orientado a pacientes de América del Norte, y el método SCORE, para pacientes europeos.

Ayuda

Breve sección donde informar al usuario sobre el uso de la aplicación.

Desarrollo

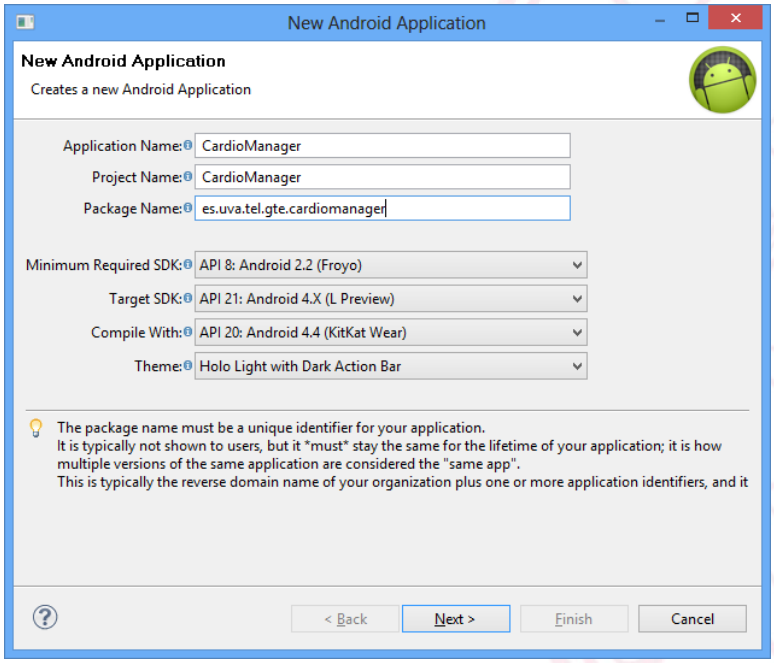
Un proyecto Android en Eclipse

En esta sección, se describirán los distintos procesos a seguir para la creación de una aplicación Android con Eclipse.

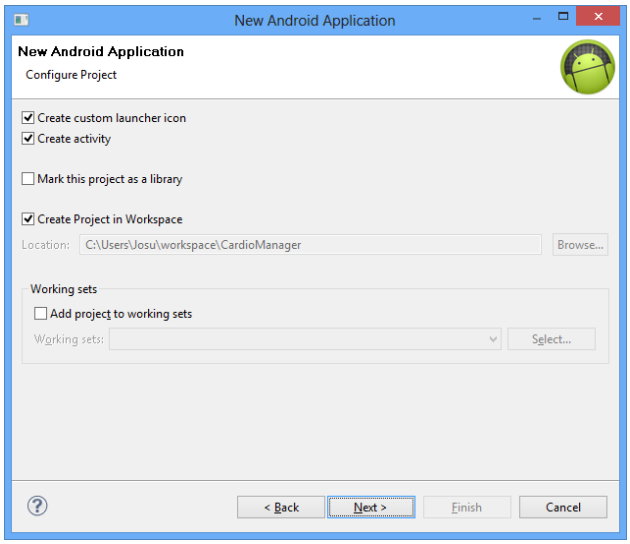
Creación de un proyecto en eclipse

Para crear un proyecto de aplicación en Android, demos pulsar en *File* > *New* > *Android Application Project*. En la ventana que se abrirá rellenaremos los diferentes campos que se corresponden con:

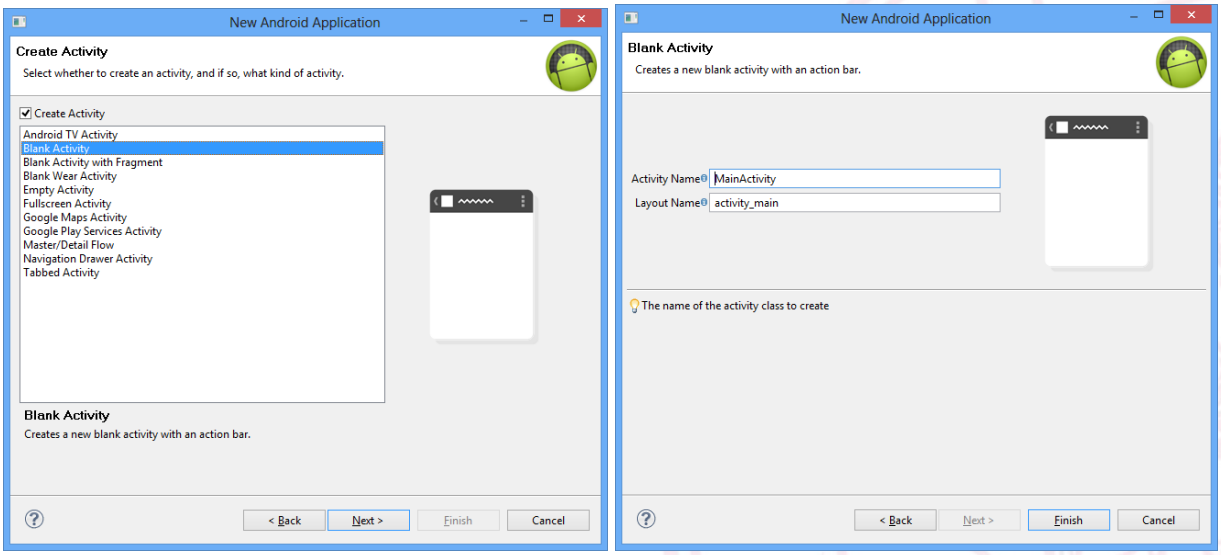
- ✓ *Application Name*: nombre de la aplicación que aparecerá en el dispositivo.
- ✓ *Project Name*: nombre del directorio del proyecto.
- ✓ *Package Name*: espacio de nombres del paquete de la aplicación, siguiendo las normas de los paquetes en programación Java, por ejemplo, *com.example.holamundo*.
- ✓ *Minimum Required SDK*: la versión más baja en la que la aplicación funcionará.
- ✓ *Target SDK*: la versión más alta en la que se ha probado la aplicación.
- ✓ *Compile With*: la versión de Android con la que se ha compilado la aplicación.
- ✓ *Theme*: estilo de la interfaz de usuario que utilizará la aplicación.



Pulsando en *Next*, accederemos a la siguiente ventana, en la que escogeremos diferentes opciones sobre el proyecto, el icono de la aplicación, si se trata de una librería o la carpeta donde deseamos almacenar los ficheros del proyecto.



A continuación, escogemos si deseamos crear una nueva actividad y qué tipo de actividad será. Se recomienda crear una actividad en blanco para mayor flexibilidad a la hora de desarrollar. Además, escogeremos también el nombre de la actividad y su fichero *layout*, cuya función será expuesta más tarde.



Al final de este proceso, ya tendremos nuestro proyecto creado y listo para empezar a programar.

Componentes de una aplicación

Activity

Cada aplicación está formada por una serie de conjuntos de visualización, coloquialmente conocidos como 'pantallas'. Cada una de estas pantallas se denomina Actividad, y proporciona una interfaz con la que el usuario puede interactuar.

Service

Son procesos que se ejecutan en segundo plano, sin necesidad de interactuar con el usuario. Pueden ser locales o remotos.

Intent

Un *Intent* es un objeto que representa el deseo de realizar una acción. Se utiliza para comenzar actividades o arrancar servicios.

Broadcast Receiver

Recibe anuncios de tipo *broadcast* y realizan acciones en consecuencia (batería baja, llamada entrante, etc).

Content Provider

Mecanismo que permite a distintas aplicaciones compartir datos.

View

Se denomina así a cada elemento que forma la interfaz gráfica:

- ✓ *Button, ImageButton, ToggleButton*: para mostrar cualquier tipo de botón.
- ✓ *TextView*: texto fijo.
- ✓ *EditText*: campo de texto editable.
- ✓ *CheckBox*: casilla para marcar con un *tick*.
- ✓ *Spinner*: lista desplegable en la que podemos seleccionar un elemento.
- ✓ *NumberPicker, DatePicker*: selector de números o fechas.
- ✓ *CalendarView*: selector de fecha con vista de un calendario. Sólo en versiones modernas.
- ✓ Otros

Layout

Conjunto de vistas agrupadas en torno a una estructura:

- ✓ *LinearLayout*: estructura lineal, es decir, un elemento detrás de otro.
- ✓ *RelativeLayout*: estructura relativa. Permite especificar la posición de cada elemento de forma relativa a otro.
- ✓ *GridLayout*: estructura de filas y columnas. Cada elemento ocupa una de estas casillas.
- ✓ Otros

Fragment

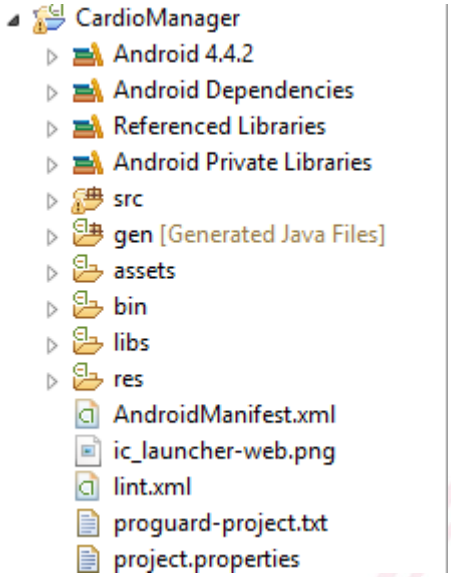
Es una porción de espacio dentro de una actividad que tiene un comportamiento independiente.

Menu

Conjunto de opciones disponibles desde el botón habilitado para ello en los dispositivos o desde la barra de acciones (*ActionBar*) en las versiones más modernas.

Estructura de un proyecto Android

Tras la creación del proyecto, podremos visualizar la estructura de directorios que tiene una aplicación Android como la que aquí se desarrollará.



`src/`

Contiene el conjunto de ficheros `.java` que forman la aplicación, almacenados en un conjunto de directorios según el nombre del paquete que los contiene.

`bin/`

Directorio donde se almacenan los ficheros generados tras compilar la aplicación. Entre ellos destacamos el fichero `.apk`, que se trata del instalador de la aplicación.

`jni/`

Sólo necesario si utilizamos Android NDK (*Native Development Kit*) para desarrollar con C++. No se utiliza para esta aplicación, por eso no aparece en la imagen anterior.

`gen/`

Consta de dos ficheros generados automáticamente por el ADT:

- ✓ `BuildConfig.java`: define la constante `DEBUG`, que indica si la aplicación está en desarrollo.
- ✓ `R.java`: asocia los recursos que definimos en los ficheros XML con elementos a los que podamos acceder desde el código Java.

`assets/`

Carpeta en la que podemos almacenar ficheros arbitrarios que pueda utilizer la aplicación.

`res/`

Contiene los recursos de la aplicación. Dada la importancia de este directorio, es necesario detallar su contenido:

- ✓ `anim/`
  - Ficheros XML relacionados con animaciones.
- ✓ `color/`
  - Ficheros XML que describan colores.
- ✓ `drawable/`
  - Directorio donde almacenaremos todas las imágenes de las que haga uso la aplicación, con formato `.png`, `.jpeg` o `.gif`.
- ✓ `layout/`
  - Ficheros XML que describen la interfaz gráfica de cada actividad.
- ✓ `menu/`
  - Ficheros XML que describen las distintas opciones a mostrar en los menús.
- ✓ `raw/`
  - Para cualquier tipo de fichero que no esté englobado en el resto de directorios y que no sea XML.
- ✓ `values/`
  - Directorio donde almacenamos los ficheros XML que definen cadenas, tamaños, listas, temas, estilos o constantes.
- ✓ `xml/`
  - Ficheros XML que no estén englobados en los demás directorios.

`libs/`

Directorio de las bibliotecas de las que hará uso el proyecto.

`AndroidManifest.xml`

Este fichero, obligatorio para todas las aplicaciones, describe las características de la aplicación y de sus componentes. Desde el número de actividades, la jerarquía de las mismas, los títulos de cada una de ellas, los servicios que utiliza la aplicación o que define, los permisos que necesita tener habilitados, las versiones soportadas y de compilado, o librerías externas utilizadas.

`project.properties`

Contiene la versión más alta con la que se ha probado la aplicación, *Target*, y referencias a librerías externas que se utilizan.

Necesidades de la aplicación

A continuación, se describen algunas características de la aplicación que requieren el uso de elementos externos al desarrollo de una aplicación Android estándar.

Base de datos

La aplicación debe almacenar mucha información referente a actividades, medicamentos, tomas, alarmas, información personal del paciente o medidas realizadas con la calculadora de riesgo cardiovascular.

Android nos ofrece tres posibilidades para almacenar información:

- ✓ `SharedPreferences` o preferencias de Android.
- ✓ Fichero de texto
- ✓ Base de datos SQL

En primer lugar, descartamos el uso de preferencias para almacenar toda esta información, debido a que sólo se recomienda su uso para guardar información referente a la personalización de la aplicación por parte del usuario, como pueda ser el tamaño de letra, el color de las fuentes u otras opciones de presentación, pues se trata de una cantidad muy pequeña de información que no va a crecer con el tiempo y el uso de la aplicación.

En segundo lugar, descartamos también el uso de ficheros de texto, pues el tiempo de creación, escritura, lectura y borrado de ficheros es bastante alto, lo cual no nos conviene. Además, no se sigue un formato estandarizado para la representación de los datos en los ficheros, y no es nada enriquecedor su utilización, desde el punto de vista del aprendizaje de la programación Android.

Por lo tanto, la mejor alternativa para almacenar información parece ser el uso de una base de datos SQL. Android posee una librería para la creación y gestión de bases de datos y el acceso a las mismas desde el código Java. Esta opción solventa los problemas del uso de preferencias, pues la cantidad de información puede crecer fácilmente con el uso de la aplicación, y del uso de ficheros, pues el uso de bases de datos sigue una serie de normas y directrices (impuestos por los propios métodos definidos en la librería de SQL para Android). Además, ser beneficioso con el objetivo de aprender a administrar y utilizar bases de datos, ya que el uso de SQL está muy extendido.

Mapas

Android no posee ninguna opción ni ninguna librería que nos proporcione una base con un mapa, con su sistema de coordenadas geográficas y sobre el cuál podamos obtener información de direcciones, almacenar coordenadas, mostrar puntos u obtener la localización del dispositivo.

Sin embargo, si existe una librería que permite el uso de los mapas de Google, Google Maps Android API V2, que nos permite mostrar un mapa base y colocar sobre él diferentes puntos. Además, para la obtención de las direcciones postales, existe la Google Places API, que a través

de un proveedor de contenidos (*ContentProvider*) nos proporciona un servicio de conversión de coordenadas geográficas en direcciones postales con formato JSON.

Gráficas

Android tampoco dispone de ninguna biblioteca que proporcione herramientas para dibujar gráficas. Por eso, necesitamos de bibliotecas elaboradas por otros programadores. En este caso, se ha escogido *afreechart*, bajo licencia GNU. Entre sus principales características destaca que permite realizar zoom sobre la parte concreta de la gráfica que queremos observar, que permite dibujar ejes de cualquier tipo y escala, dispone multitud de tipos de gráficos, como gráficos de puntos, de barras, permite establecer leyendas y es de uso gratuito.

Seguridad

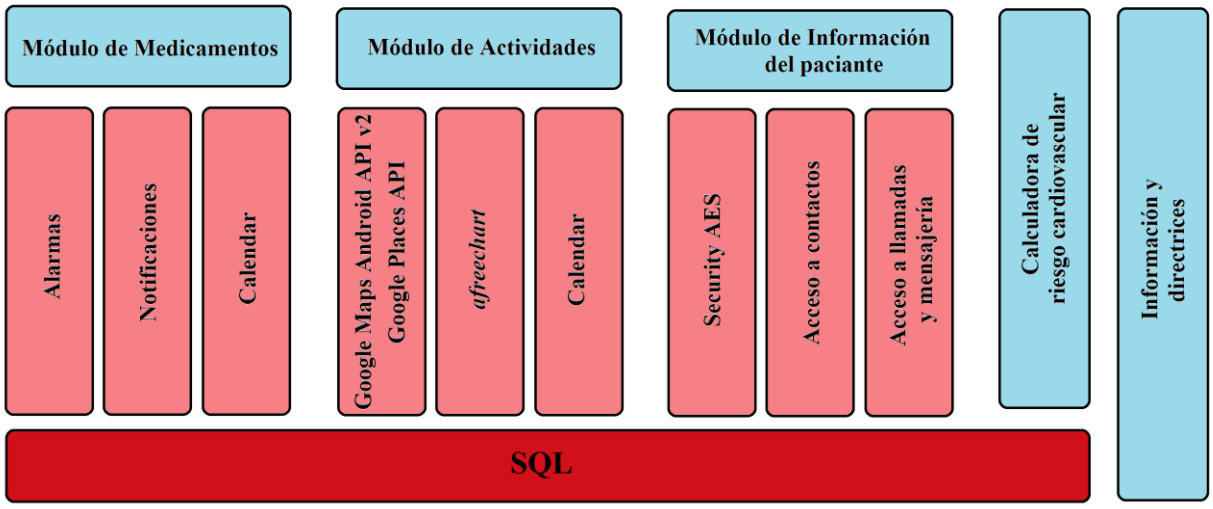
Dado que es necesario almacenar ciertos datos sobre el usuario de forma que mantengamos la privacidad del mismo, es necesario cifrar esos datos antes de almacenarlos. Para ello, se ha utilizado la biblioteca `javax.crypto` para cifrar la información.

Interfaz gráfica

Para mejorar algunos aspectos de la interfaz gráfica, también es necesario hacer uso de bibliotecas externas. En concreto, para mostrar de forma más atractiva y adecuada a la aplicación los selectores o *Pickers* de fecha, número y hora, se ha escogido la biblioteca `android-numberpicker` de SimonVT, que permite editar el estilo de los mismos.

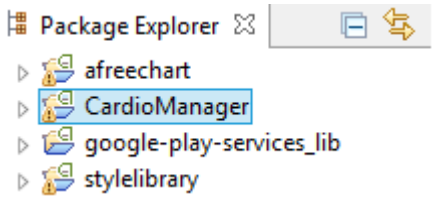
Implementación

En esta sección se desarrolla detalladamente la metodología empleada para alcanzar los distintos objetivos en cada uno de los módulos de la aplicación. En concreto, se detalla el procedimiento seguido para el desarrollo de todo el código en Java, además de los ficheros XML que describen los distintos *layouts*. La siguiente figura muestra un diagrama con la arquitectura de la aplicación. En azul podemos observar los módulos más importantes, acompañados de cada uno de las principales bibliotecas o servicios de los que hace uso en rojo.

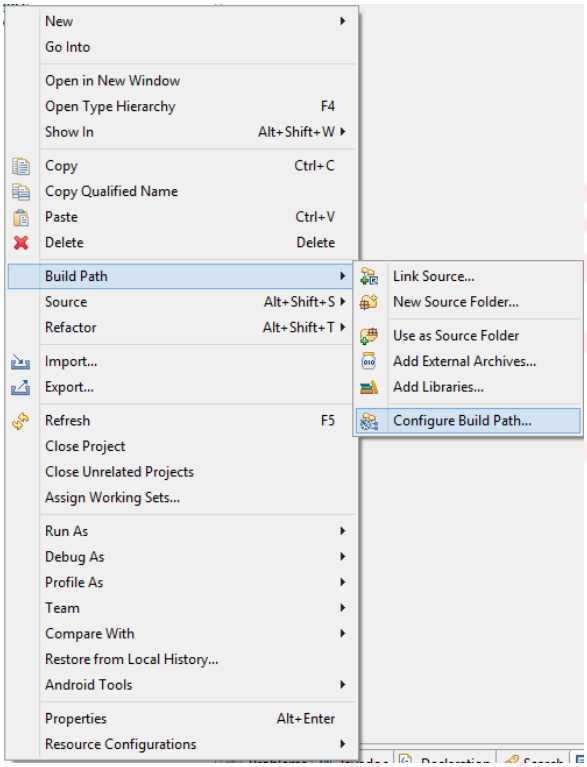


Consideraciones previas

En la siguiente imagen podemos observar los proyectos incluidos en la aplicación. No sólo está el proyecto de la aplicación, CardioManager, sino también las tres bibliotecas necesarias para ampliar funcionalidades, que antes hemos comentado.

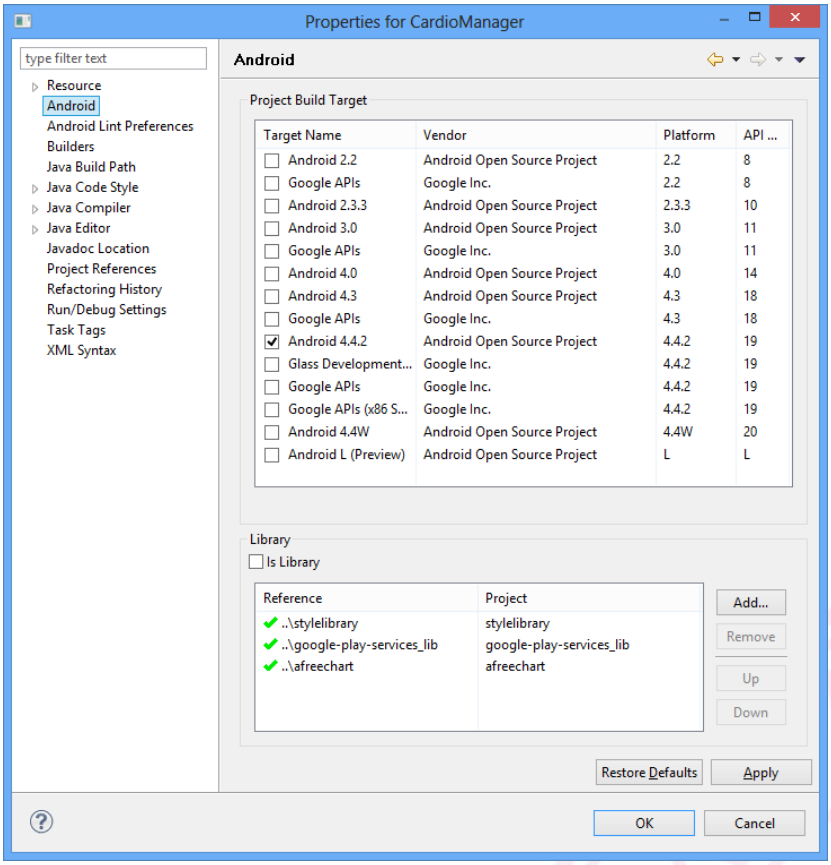


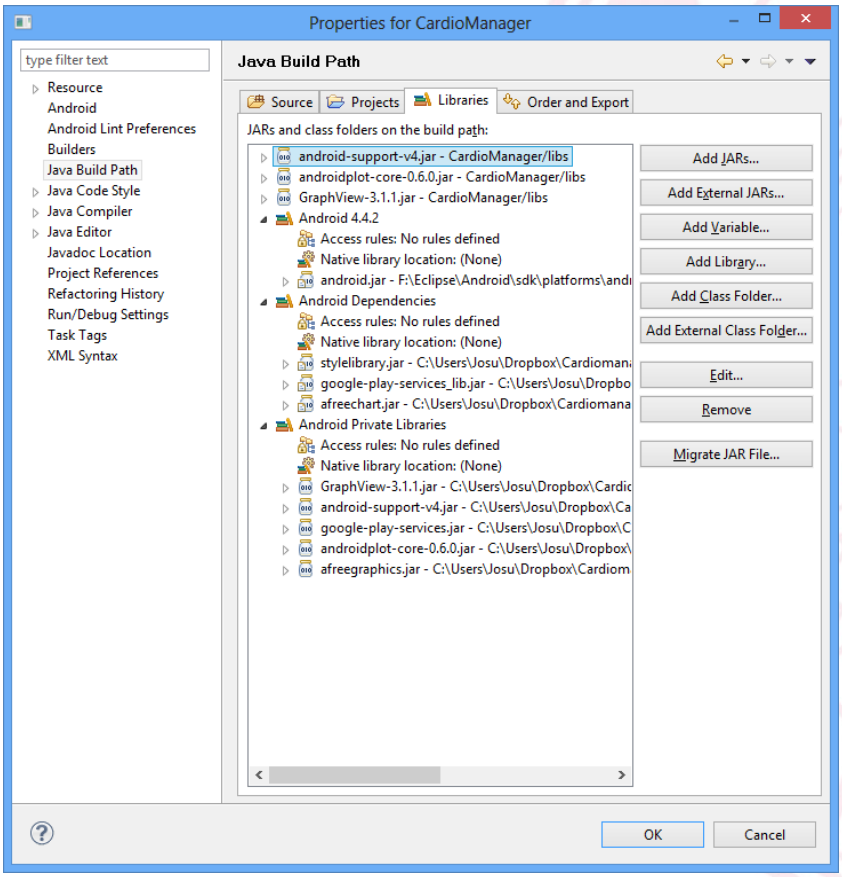
Para añadirlas a nuestro proyecto CardioManager las diferentes bibliotecas debemos configurar el *BuildPath.* Accedemos a él pulsando en el botón derecho del ratón en nuestro proyectom y seleccionando las siguientes opciones.



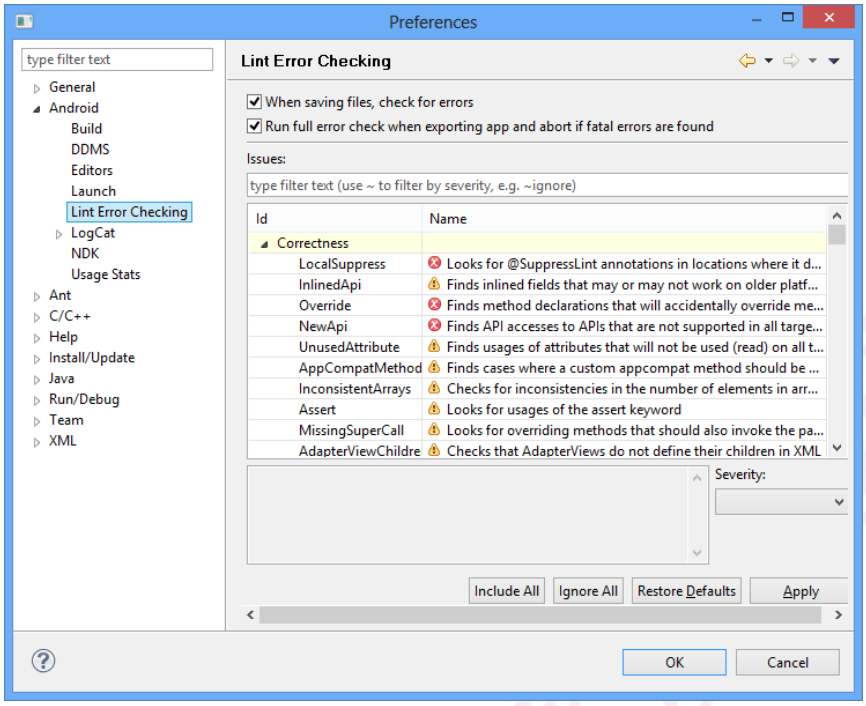
A continuación, en la sección de Android, debemos añadir las tres librerías para poder hacer uso de ellas.

Además, debemos comprobar que las dependencias están configuradas de forma correcta, tal y como aparece en las siguientes imágenes.

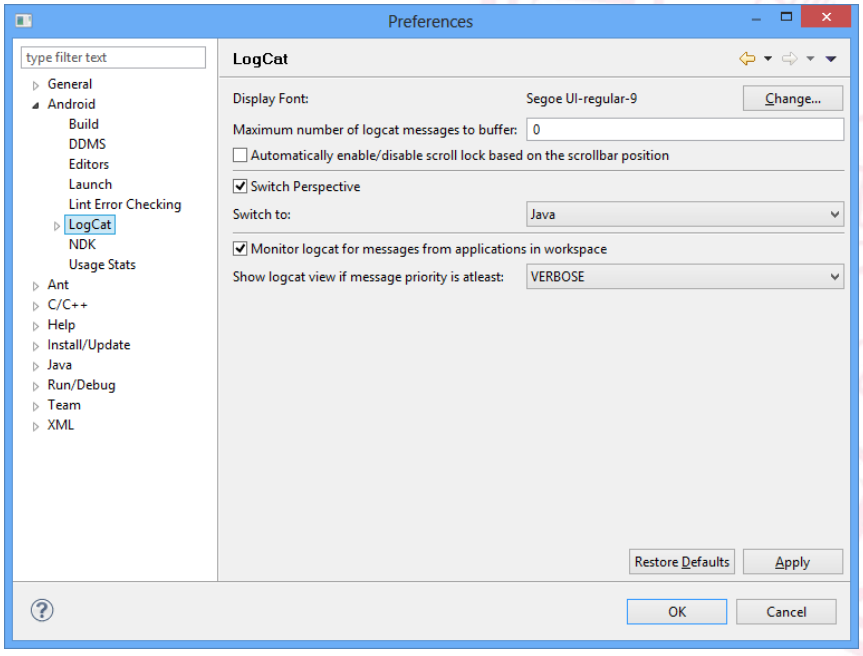

Properties for CardioManager
type filter text
Resource
Android
Android Lint Preferences
Builders
Java Build Path
Java Code Style
Java Compiler
Java Editor
Javadoc Location
Project References
Refactoring History
Run/Debug Settings
Task Tags
XML Syntax
Android
Project Build Target
Target Name
Vendor
Platform
API ...
Android 2.2
Android Open Source Project
2.2
8
Google APIs
Google Inc.
2.2
8
Android 2.3.3
Android Open Source Project
2.3.3
10
Android 3.0
Android Open Source Project
3.0
11
Google APIs
Google Inc.
3.0
11
Android 4.0
Android Open Source Project
4.0
14
Android 4.3
Android Open Source Project
4.3
18
Google APIs
Google Inc.
4.3
18
Android 4.4.2
Android Open Source Project
4.4.2
19
Glass Development...
Google Inc.
4.4.2
19
Google APIs
Google Inc.
4.4.2
19
Google APIs (x86 S...
Google Inc.
4.4.2
19
Android 4.4W
Android Open Source Project
4.4W
20
Android L (Preview)
Android Open Source Project
L
L
Library
Is Library
Reference
Project
..\stylelibrary
stylelibrary
..\google-play-services_lib
google-play-services_lib
..\afreechart
afreechart
Add...
Remove
Up
Down
Restore Defaults
Apply
OK
Cancel


Properties for CardioManager
type filter text
Resource
Android
Android Lint Preferences
Builders
Java Build Path
Java Code Style
Java Compiler
Java Editor
Javadoc Location
Project References
Refactoring History
Run/Debug Settings
Task Tags
XML Syntax
Java Build Path
Source
Projects
Libraries
Order and Export
JARs and class folders on the build path:
android-support-v4.jar - CardioManager/libs
androidplot-core-0.6.0.jar - CardioManager/libs
GraphView-3.1.1.jar - CardioManager/libs
Android 4.4.2
Access rules: No rules defined
Native library location: (None)
android.jar - F:\Eclipse\Android\sdk\platforms\andr
Android Dependencies
Access rules: No rules defined
Native library location: (None)
stylelibrary.jar - C:\Users\Josu\Dropbox\Cardioman
google-play-services_lib.jar - C:\Users\Josu\Dropbo
afreechart.jar - C:\Users\Josu\Dropbox\Cardiomana
Android Private Libraries
Access rules: No rules defined
Native library location: (None)
GraphView-3.1.1.jar - C:\Users\Josu\Dropbox\Cardic
android-support-v4.jar - C:\Users\Josu\Dropbox\Ca
google-play-services.jar - C:\Users\Josu\Dropbox\C
androidplot-core-0.6.0.jar - C:\Users\Josu\Dropbox\
afreegraphics.jar - C:\Users\Josu\Dropbox\Cardiom
Add JARs...
Add External JARs...
Add Variable...
Add Library...
Add Class Folder...
Add External Class Folder...
Edit...
Remove
Migrate JAR File...
OK
Cancel

También es necesario configurar algunas opciones en Eclipse. A la hora de exportar la aplicación, podemos tener algunos problemas relacionados con los ficheros de la carpeta /res, en concreto con los ficheros necesarios para otros idiomas que no sean los idiomas seleccionados para esta aplicación, el inglés y el castellano. Por ello, debemos desactivar la opción *Run ful error check when exporting app and abort if fatal errors are found*, que por defecto, como vemos en la imagen, esta activada. Es accesible desde las preferencias de Eclipse, sección Android, sección Lint Error Checking.



Además, para poder llevar un seguimiento de lo que sucede mientras probamos la aplicación, es conveniente activar la casilla *Monitor logcat for messages from applocations in workspace*, accesible desde las preferencias de Eclipse, sección Android, sección LogCat.



AndroidManifes.xml

Como bien se ha indicado previamente, este fichero es obligatorio para todas las aplicaciones, y su importancia es vital. Por ello, se muestra a continuación el contenido del fichero, para su posterior explicación.

En el primer fragmento de código, vemos como comienza el fichero AndroidManifes.xml. En primer lugar se especifica el nombre del paquete de la aplicación, que introducimos durante la creación del proyecto, aunque puede ser cambiado durante la fase de desarrollo. Además, observamos que se trata de la primera versión de la aplicación.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="es.uva.tel.gte.cardiomanager"
    android:versionCode="1"
    android:versionName="1.0" >
```

A continuación, se muestran las líneas que especifican el número de API correspondiente con la mínima versión de Android en la que funciona la aplicación, en este caso API nivel 8, correspondiente con Android 2.2, y el nivel de API de la versión de Android más alta para la que se ha probado la aplicación, API nivel 18 en este caso, o Android 4.3.

```
    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="18" />
```

El siguiente fragmento de código muestra los permisos que el usuario necesita otorgar a la aplicación para que funcione correctamente y pueda ser instalada. Estos permisos se detallan en la sección Manual de usuario, por lo que no entraremos en más detalles.

```
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
    <uses-permission android:name="android.permission.READ_CALENDAR" />
    <uses-permission android:name="android.permission.READ_CONTACTS" />
    <uses-permission android:name="android.permission.CALL_PHONE" />
    <uses-permission android:name="android.permission.SEND_SMS" />
    <uses-permission android:name="android.permission.VIBRATE" />
    <uses-permission
       android:name="es.uva.tel.gte.cardiomanager.permission.MAPS_RECEIVE" />
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission
       android:name="com.google.android.providers.gsf.permission.READ_GSERVICES" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />

    <!-- Protect the map component of the application using application signature -->
    <permission
        android:name="es.uva.tel.gte.cardiomanager.permission.MAPS_RECEIVE"
        android:protectionLevel="signature" />
```

A continuación, especificamos la versión de OpenGL ES que necesita la aplicación. OpenGL es la API para gráficos de Android.

```
    <uses-feature
        android:glEsVersion="0x00020000"
        android:required="true" />
```

Se especifican también los tipos de pantalla soportados por la aplicación.

```
<supports-screens
    android:largeScreens="true"
    android:normalScreens="true"
    android:smallScreens="true"
    android:xlargeScreens="true" />
```

A continuación se declara la información referente al icono, nombre y tema de la aplicación, además de las distintas actividades, servicios y proveedores de contenido.

```
<application
    android:allowBackup="true"
    android:icon="@drawable/ic_launcher"
    android:label="@string/app_name"
    android:theme="@style/AppTheme" >
```

La primera actividad que declaramos será con la que se inicie la aplicación. Debemos especificarlo en el manifiesto añadiendo el `<intent-filter>` que vemos.

```
<activity
      android:name="es.uva.tel.gte.cardiomanager.main.MainActivity"
      android:label="@string/app_name"
      android:theme="@style/SampleTheme"
      android:configChanges="orientation" >
      <intent-filter>
          <action android:name="android.intent.action.MAIN" />
          <category android:name="android.intent.category.LAUNCHER" />
      </intent-filter>
</activity>
```

A continuación, seguimos declarando las distintas actividades. Es necesario que estén todas presentes, pues si no la aplicación no funcionará correctamente y se producirán cierres inesperados. En el siguiente fragmento se muestran algunas de las actividades en las que se ha declarado algún atributo de carácter especial o que cabe destacar. No se ha incluido el fichero de forma íntegra. Los atributos a los que debemos prestar especial atención son:

- ✓ `android:label="@string/…"`. Este atributo nos permite seleccionar el título que se mostrará cuando la actividad esté en primer plano.

- ✓ `android:parentActivityName="…"`. Con este atributo habilitamos la navegación hacia atrás al pulsar en la barra de acciones de las versiones modernas de Android (versión superior o igual a Android 4.0). En concreto, se volverá a la actividad especificada entre comillas. Una segunda forma de especificar esto es a través del siguiente código es:
  ```
  <meta-data
        android:name="android.support.PARENT_ACTIVITY"
        android:value="…"/>
  ```

- ✓ `android:theme="@style/SampleTheme"`. Este atributo indica que la actividad utiliza el tema especificado entre comillas, desarrollado en un fichero XML, y que será expuesto en la siguiente sección de Interfaz Gráfica.

- ✓ `android:configChanges="orientation"`. Indica que el dispositivo debe gestionar los cambios en la configuración del dispositivo, en concreto cambios de orientación cuando

el usuario gira el dispositivo, cuando la actividad de la que forma parte este atributo está en primer plano.

- ✓ `android:launchMode="singleTop"`. Este atributo con este valor indica que, si la actividad ha sido ya creada y se encuentra en segundo plano, en el caso de que ocurra una llamada hacia la misma, en lugar de crearla de nuevo, se muestra en primer plano.

- ✓ `android:windowSoftInputMode="adjustPan"`. Este atributo con este valor se utiliza para que, en las actividades con vistas de tipo *EditText*, al abrirse el teclado del teléfono, no se oculte el campo donde estemos escribiendo y la pantalle se ajuste automáticamente.

- ✓ Hay que prestar atención a lo especificado en la actividad `MapChooseLocation`. En esta actividad, que más tarde describiremos, se muestra una barra de búsqueda especificada en un fichero XML, por lo que debemos añadir dicho código., que consta de un <intent-filter> indicando que la actividad es una actividad en la que se va a realizar una acción de búsqueda o *search*, e indicando el *layout* que mostrará dicha barra de búsqueda.

```
<activity
    android:name="es.uva.tel.gte.cardiomanager.activity.NewActivity"
    android:label="@string/title_new_activity"
    android:parentActivityName=
     "es.uva.tel.gte.cardiomanager.main.MainActivity"
    android:theme="@style/SampleTheme"
    android:configChanges="orientation" >
</activity>
<activity
    android:name="es.uva.tel.gte.cardiomanager.info.Information"
    android:label="@string/title_info"
    android:launchMode="singleTop"
    android:parentActivityName=
     "es.uva.tel.gte.cardiomanager.main.MainActivity"
    android:theme="@style/SampleTheme" >
</activity>
<activity
    android:name="es.uva.tel.gte.cardiomanager.guide.Guidelines"
    android:label="@string/guide"
    android:parentActivityName=
     "es.uva.tel.gte.cardiomanager.main.MainActivity"
    android:theme="@style/SampleTheme" >
</activity>
<activity
    android:name="es.uva.tel.gte.cardiomanager.medicinenew.MedicineMain"
    android:label="@string/title_activity_new_medicine"
    android:parentActivityName=
     "es.uva.tel.gte.cardiomanager.main.MainActivity"
    android:theme="@style/SampleTheme"
    android:windowSoftInputMode="adjustPan" >
</activity>

...

<activity
    android:name="es.uva.tel.gte.cardiomanager.medicinerecord.MedicineList"
    android:label="@string/title_activity_medicine_list"
    android:parentActivityName=
     "es.uva.tel.gte.cardiomanager.medicinerecord.MedicineRecordMain"
    android:theme="@style/SampleTheme"
    android:configChanges="orientation">
```

```
    </activity>

...

    <activity
        android:name="es.uva.tel.gte.cardiomanager.activityrecord.ChooseGraph"
        android:label="@string/title_activity_choose_graph"
        android:parentActivityName=
         "es.uva.tel.gte.cardiomanager.activityrecord.ActivityRecordMain"
        android:theme="@style/SampleTheme" >
        <meta-data
            android:name="android.support.PARENT_ACTIVITY"
            android:value=
                "es.uva.tel.gte.cardiomanager.activityrecord.ActivityRecordMain"
         />
    </activity>

    <activity
        android:name="es.uva.tel.gte.cardiomanager.maps.MapChooseLocation"
        android:label="@string/title_activity_choose_location"
        android:launchMode="singleTop"
        android:theme="@style/SampleTheme" >
        <intent-filter>
            <action android:name="android.intent.action.SEARCH" />
        </intent-filter>

        <!-- Points to searchable activity -->
        <meta-data
            android:name="android.app.default_searchable"
            android:value=".MapChooseLocation" />

        <!-- Points to searchable meta data -->
        <meta-data
            android:name="android.app.searchable"
            android:resource="@xml/searchable" />
    </activity>


    <activity
        android:name="es.uva.tel.gte.cardiomanager.maps.LoadLocation"
        android:label="@string/title_activity_load_location"
        android:parentActivityName=
         "es.uva.tel.gte.cardiomanager.activitycalendar.CalendarMain"
        android:theme="@style/SampleTheme" >
        <meta-data
            android:name="android.support.PARENT_ACTIVITY"
            android:value=
                "es.uva.tel.gte.cardiomanager.activitycalendar.CalendarMain" />
    </activity>
```

Como ya hemos comentado, en el manifiesto también deben declararse los servicios, los *Broadcast Receivers* y los proveedores de contenidos (*ContentProviders*).

```
<!-- SERVICES -->
    <service
        android:name="es.uva.tel.gte.cardiomanager.medicinealarmservice.Manager"
        android:enabled="true"
        android:exported="false" >
    </service>
    <service
        android:name="es.uva.tel.gte.cardiomanager.main.AppService"
        android:enabled="true"
        android:exported="false" >
    </service>
    <service
```

```
            android:name="es.uva.tel.gte.cardiomanager.main.CheckAlarmsService"
            android:enabled="true"
            android:exported="false" >
        </service>
        <service
            android:name="com.example.android.location.ReceiveUpdatesIntentService"
            android:exported="false"
            android:label="@string/app_name" />


    <!-- BROADCAST RECEIVER -->
      <receiver android:name=
           "es.uva.tel.gte.cardiomanager.medicinealarmservice.Receiver" >
      </receiver>

      <!-- CONTENT PROVIDER -->
      <provider
          android:name="es.uva.tel.gte.cardiomanager.jsonprovider.PlaceProvider"
          android:authorities=
                  "es.uva.tel.gte.cardiomanager.jsonprovider.PlaceProvider"
          android:exported="false" />
```

Además, para el uso de servicios Google, como más tarde comentaremos, se necesita la clave de API registrada para esta aplicación. Esto se especifica añadiendo metadatos dentro de la declaración de la aplicación. Por último, se cierran todas las etiquetas y termina el manifiesto.

```
<!-- META-DATA -->
        <!-- Specifies the Android API Key,
            which is obtained from Google API Console -->
        <meta-data
            android:name="com.google.android.maps.v2.API_KEY"
            android:value="AIzaSyB07pb4ubpDJwOwDywFC6ymFQnxSflEZtM"    />
        <!-- Actual: GTE -->
        <!--  android:value="AIzaSyAByDMTgd8sfOOjq1nkaLG9KFcG7ZzqRic"
        PC Josu - Sobremesa -->
        <meta-data
            android:name="com.google.android.gms.version"
            android:value="@integer/google_play_services_version" />
    </application>
</manifest>
```

Actividad principal

El módulo que se encarga de la gestión de la actividad principal está incluido en el paquete `es.uva.tel.gte.cardiomanager.main`. En la pantalla inicial se muestran los siete botones con su icono incluido que nos permiten acceder a cada uno de los módulos de la aplicación, y si deslizamos el dedo hacia la izquierda, se nos permite consultar nuestro contacto de emergencia y establecer una llamada o enviarle un mensaje. La programación de estos botones es de lo más básica, por lo tanto no merece la pena incluir ningún código identificativo. Sí que es importante mostrar el código y la distribución de las clases que se encargan de la gestión de mostrar la pantalla inicial y la pantalla mostrada tras deslizar el dedo.

Las ficheros con el código que se encarga de esto son:

- ✓ `MainActivity.java`.
    - Define la clase MainActivity, que hereda de FragmentActivity, la clase base para el uso de actividades que contengan fragmentos (*Fragments*).

- En esta clase, además, se inicia el servicio que comprobará si la aplicación fue cerrada y es necesario reestablecer alarmas de medicación y se gestionará si el usuario pulsa en alguno de los botones de acceso a otros módulos.
- Para acceder a los módulos de Información del paciente y Ayuda, es necesario pulsar en los botones del menú.

✓ `slide_main.xml`

- Fichero de *layout* que carga la actividad anterior. Su contenido es:

```
<android.support.v4.view.ViewPager
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:id="@+id/pager"
… />
```

- Como vemos, se trata de un *ViewPager*, un tipo de *layout* que nos permite cargar varias páinas (o pantallas) y navegar a través de ellas deslizando el dedo hacia la izquierda o la derecha de la pantalla.

✓ `main.xml`

- Fichero de menú cuyo contenido es el siguiente:

```
<menu
xmlns:android="http://schemas.android.com/apk/res/android" >
    <item
       android:id="@+id/help"
       android:orderInCategory="100"
       android:showAsAction="ifRoom"
       android:title="@string/help"
       android:icon="@drawable/help"/>
    <item
        android:id="@+id/reg"
        android:orderInCategory="100"
        android:showAsAction="ifRoom"
        android:title="@string/personalinfo"
        android:icon="@drawable/person"/>
</menu>
```

- En este fichero se especifican dos opciones en el menú, una para acceder a la Ayuda y otra para acceder a la Información personal. El menú está implementado de forma que sea compatible con todas las versiones de Android disponibles y se adapte a las mismas. El atributo `android:showAsAction = "ifRoom"`, permite que el menú se muestre en la barra de acciones (*ActionBar*) en las versiones más actuales de Android, y que simplemente aparezca al pulsar en el botón de menú para las versiones antiguas. Además, se carga un icono identificativo para cada opción.
- *Nota: de aquí en adelante no se mostrarán más ficheros de menú, ya que su implementación es similar a esta y en ninguno de ellos se añade ninguna funcionalidad ni característica nueva.*

✓ `MainFragment.java`

- Como hemos comentado, MainActivity.java es la clase que funciona como base para mostrar fragmentos encima. Al 'inflar' un *layout* de tipo *ViewPager*, lo que hacemos es establecer un marco que contenga un conjunto de fragmentos. Por defecto, se mostrará el primer fragmento a pantalla completa, y al deslizar hacia la derecha se mostrarán el resto de fragmentos, uno a uno, algo similar a pasar las páginas de un libro. Por ello, necesitamos una clase que gestione la carga de estos fragmentos, que no es otra que la aquí tratada. Dependiendo del número de página cargará uno u otro fichero de *layout*.

✓ `activity_main.xml`

- Fichero que define el conjunto de botones que debemos pulsar para acceder a los distintos módulos. Define una serie de *Views* de tipo *TableRow* que contienen *ImageButtons*. A continuación se muestra un fragmento de código para mejor comprensión:

```
<TableRow
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:baselineAligned="false"
        android:orientation="horizontal" >
        <LinearLayout
            android:layout_width="0dp"
            android:layout_height="match_parent"
            android:gravity="center"
            android:layout_weight="1"
            android:orientation="horizontal" >
            <ImageButton
                android:id="@+id/buttonRegAct"
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:layout_margin="@dimen/padding_spinner"
                android:background="#00000000"
                android:scaleType="fitCenter"
                android:src="@drawable/button_reg_act"
                android:onClick="clickNewRegister" />
             </LinearLayout>

         <LinearLayout
            android:layout_width="0dp"
            android:layout_height="match_parent"
            android:gravity="center"
            android:layout_weight="1"
            android:orientation="horizontal" >
            <ImageButton
                android:id="@+id/buttonViewAct"
                android:layout_width="fill_parent"
                android:layout_height="fill_parent"
                android:layout_margin="@dimen/padding_spinner"
                android:background="#00000000"
                android:scaleType="fitCenter"
                android:src="@drawable/button_view_act"
                android:onClick="clickActivityRegister" />
        </LinearLayout>
    </TableRow>
```

- ✓ `contact_main_activity.xml`
  - Fichero que carga el *layout* encargado de mostrar el contacto de emergencia y dos botones, uno para realizar una llamada y otro para enviar un mensaje.
- ✓ `PoliticDialog.java`
  - Define un *DialogFragment* que muestra los términos que debe aceptar el usuario para el uso de la aplicación, que se muestran en un diálogo al abrir la aplicación por primera vez.
- ✓ `AppService.java` y `CheckAlarmService.java`
  - Ambas clases definen dos *Services* que se encargan de comprobar si la aplicación fue cerrada y si es necesario reestablecer alarmas. Estas dos clases están incluidas en este paquete, sin embargo, forman parte del módulo de notificaciones y alarmas, por lo que no serán detalladas aquí.

Introducción de Actividades

Este módulo, utilizado para introducir actividades, está implementado por completo en el paquete `es.uva.tel.gte.cardiomanager.activity`, en la clase `NewActivity.java`. Sin embargo, para la selección de la ubicación de los ataques o crisis se hace uso de las clases del paquete `es.uva.tel.gte.cardiomanager.maps`, cuyo contenido será especificado posteriormente. Además se hace uso de la biblioteca `stylelibrary`, para los diferentes *Pickers*, que también será especificada posteriormente.

En esta actividad, se muestra un desplegable o *Spinner* en el que se permite seleccionar entre los distintos tipos de actividades. Al escoger uno de ellos, se actualiza la actividad mostrando distintas vistas en función de los parámetros necesarios para almacenar cada actividad. El código que muestra el *Spinner* y gestiona los eventos de selección de cada actividad es el siguiente.

```
ArrayAdapter<CharSequence> adapter =
      ArrayAdapter.createFromResource(this, R.array.register_array,
      R.layout.spinner_item);
adapter.setDropDownViewResource(R.layout.dropdown_spinner_item);

Spinner typeActivity =
      (Spinner) findViewById(R.id.spinner_type_act);
typeActivity.setAdapter(adapter);
typeActivity.setOnItemSelectedListener(new OnItemSelectedListener() {
      public void onItemSelected(AdapterView<?> parent, View view,
            int pos,long id) {
            hideAllElements();
            if(android.os.Build.VERSION.SDK_INT<11) layout
                  = (LinearLayout) findViewById(R.id.dateLinearLayout);
            else layout  = (LinearLayout) findViewById(R.id.layoutDate);
            layout.setVisibility(View.VISIBLE);
            switch(pos){
                  case 0:
                        TABLE_NAME = "";
                        layout.setVisibility(View.GONE);
```

```
                        break;
                    case 1:     //Rehabilitación
                        et = (EditText) findViewById(R.id.editText_obs);
                        et.setVisibility(View.VISIBLE);
                        TABLE_NAME = DBAdapter.TABLE_NAME_REHAB;
                        break;
                    case 2:     //Actividad física
                        ...
                        TABLE_NAME = DBAdapter.TABLE_NAME_PHYSICAL;
                        break;
                    ...
                }
            }
        @Override
        public void onNothingSelected(AdapterView<?> arg0) {
            // Do nothing
        }
});
```

Dentro de este código cabe destacar el método `hideAllElements()`, que oculta todos los elementos del *layout*, excepto el *Spinner* de selección de actividad.

En lo referente a los *Views* incluidos en el *layout*, para la selección de una fecha, se ha utilizado un *DatePicker* personalizado, editado a través de la biblioteca de SimonVT de la que hemos hablado anteriormente. Para la introducción de texto se hace uso de *EditTexts*, para la selección de duraciones, tipo de insulina o tipo de exceso se muestran diferentes *Spinners* y para el resto de parámetros, de carácter numérico, se utilizan *NumberPickers* editados también a través de la biblioteca de SimonVT. En este módulo también se escoge la ubicación de un ataque o crisis, pero este procedimiento se explica en posteriores secciones, donde se detalla todo lo referente al uso de mapas.

Registro de Actividades

En este módulo podemos consultar información sobre todas las actividades introducidas. Los paquetes y clases encargados de que se lleven a cabo estas funciones son:

- ✓ `es.uva.tel.gte.cardiomanager.activityrecord.`
  - o `ActivityRecordMain.java`
- ✓ `es.uva.tel.gte.cardiomanager.activitycalendar.`
  - o `CalendarMain.java`
  - o `CalendarAdapter.java`
  - o `FragmentCalendar.java`
  - o `FragmentSpec.java`
  - o `FragmentNothingSelected.java`
  - o `DayAdapter.java`
  - o `Row.java`

- ✓ es.uva.tel.gte.cardiomanager.chart.
  - o SelectMonthYear.java
  - o BloodPressureActivity.java
  - o BloodPressureFragment.java
  - o BloodPressureChart.java
  - o GlucoseActivity.java
  - o GlucoseFragment.java
  - o GlucoseChart.java
  - o TestActivity.java
  - o TestFragment.java
  - o TestChart.java
  - o DemoView.java

En el primero de ellos, la única clase que hay se encarga de mostrar dos botones, el primero con acceso al calendario de actividades y el segundo con acceso a las gráficas.

En el segundo, se gestiona todo lo referente a mostrar el calendario y las actividades de cada día. La actividad que contiene el calendario es `CalendarMain.java`, y es necesario mostrar parte de su contenido para ilustrar su funcionalidad.

```
public class CalendarMain extends FragmentActivity
      implements FragmentCalendar.OnItemGridClick{

      DBAdapter dbAdapter;
      Formater formater;
      boolean smallScreen;
      int landscape;
      Cursor cursorRehab;
      Cursor cursorPhysical;
      Cursor cursorTest;
      Cursor cursorBloodPressure;
      Cursor cursorGlucose;
      Cursor cursorAttack;
      Cursor cursorExcess;
      Calendar calendar;
    FragmentSpec fragment;

@Override
protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);

      if(Build.VERSION.SDK_INT > 11){
            ActionBar actionBar = getActionBar();
            actionBar.setDisplayHomeAsUpEnabled(true);
      }
```

```
            // Creamos el fragmento que muestra las descripciones
            fragment = new FragmentSpec();
            // LLamada al méto que permite llenar el fragmento
                  con los datos de la lista (si existen)
            fragment.setFragmentSpec(getApplicationContext(),
                  R.layout.activity_list_item, null, true);
            fragment.setRetainInstance(true);

            // True si estamos en tamaño de pantalla normal o pequeño,false si
            // estamos en una pantalla grande o extra grande
            smallScreen = (getResources().getConfiguration().screenLayout &
                  Configuration.SCREENLAYOUT_SIZE_MASK) <=
                  Configuration.SCREENLAYOUT_SIZE_NORMAL;

            if(Build.VERSION.SDK_INT<9)
            setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);
            else setRequestedOrientation(ActivityInfo
                        .SCREEN_ORIENTATION_SENSOR_PORTRAIT);

              if (smallScreen) {
                  // Si tamaño de pantalla pequeño, mostramos el
                  // fragmento del calendario únicamente
                  setContentView(R.layout.calendar_main_small_normal);
                  // Creamos objeto del fragmento a mostrar
                  FragmentCalendar calendar = new FragmentCalendar();
                  // Añadimos el fragmento al contenedor
                  getSupportFragmentManager().beginTransaction()
                        .add(R.id.fragment_calendar_container, calendar)
                        .commit();
              }
              else {
                  setContentView(R.layout.calendar_main);
              }

            // Leemos todos los cursores de la base de datos una vez,
            // al iniciar la actividad, para mejorar el rendimiento.
            dbAdapter = new DBAdapter(getApplicationContext());
            dbAdapter.open();
            cursorRehab = dbAdapter.fetchAllRows(DBAdapter.TABLE_NAME_REHAB);
            cursorPhysical = dbAdapter.fetchAllRows(DBAdapter.TABLE_NAME_PHYSICAL);
            cursorTest = dbAdapter.fetchAllRows(DBAdapter.TABLE_NAME_BLOOD_TEST);
            cursorBloodPressure =
                  dbAdapter.fetchAllRows(DBAdapter.TABLE_NAME_BLOOD_PRESSURE);
            cursorGlucose = dbAdapter.fetchAllRows(DBAdapter.TABLE_NAME_GLUCOSE);
            cursorAttack = dbAdapter.fetchAllRows(DBAdapter.TABLE_NAME_ATTACK);
            cursorExcess = dbAdapter.fetchAllRows(DBAdapter.TABLE_NAME_EXCESS);
      }

      @Override
      public void onItemGridClick(String selectedDay, int selection){
            try{
                  // Lista de con el tipo de actividad,
                        el icono y una breve descripcion.
                  ArrayList<Row> rowList = update(selectedDay, selection);

                  // LLamada al méto que permite llenar el fragmento
                        con los datos de la lista (si existen)
                  fragment = new FragmentSpec();
                  fragment.setFragmentSpec(getApplicationContext(),
```

```
                R.layout.activity_list_item, rowList, false);
            // Create new fragment and transaction
            FragmentTransaction transaction
                = getSupportFragmentManager().beginTransaction();
            Fragment frg;
            // Para pantallas pequeñas, debemos sustituir
                el fragmento con el calendario
            // por uno con la lista de actividades. Añadimos el fragmento
                retirado a la pila de tareas.

            if (smallScreen) {
                frg =
                    getSupportFragmentManager()
                    .findFragmentById(R.id.fragment_calendar_container);
                // Replace whatever is in the fragment_container
                    view with this fragment,
                // and add the transaction to the back stack
                transaction.remove(frg);
                transaction.replace(R.id.fragment_calendar_container,
                    fragment);
                transaction.addToBackStack(null);
                transaction.commit();
            }
            // Si estamos en tamaños de pantalla grandes,
                actualizamos el segundo fragmento.
            else{
                frg = getSupportFragmentManager()
                    .findFragmentById(R.id.spec_fragment);

                // Replace whatever is in the fragment_container
                    view with this fragment,
                // and add the transaction to the back stack
                transaction.remove(frg);
                transaction.replace(R.id.spec_fragment, fragment);
                transaction.commit();
            }

        }
        catch(Exception e){
            …
        }
}

…
```

Se trata de una actividad que hereda de FragmentActivity pues servirá de marco para mostrar diversos fragmentos A la hora de desarrollar el calendario, y con el objetivo de mejorar la interfaz de usuario se tomó la decisión de, para dispositivos de gran tamaño, mostrar el calendario en la mitad superior de la pantalla, y la descripción de las actividades del día en la mitad inferior de la misma. Mientras que para dispositivos normales o pequeños, el calendario se muestra a pantalla completa, y al pulsar en un día, se muestra a pantalla completa también, sustituyendo al calendario, la descripción de las actividades del día. También se tomó la decisión de obligar al dispositivo a mostrar el calendario en vertical, y no colocar una vista apaisada en el caso de que se gire la pantalla del dispositivo, con el fin de mejorar la visualización.

Para ello se elaboró el anterior código: en primer lugar, se habilita el botón *home* de la barra de acciones que permite volver hacia atrás en la pila de tareas. A continuación se crea el fragmento que vaya a contener la descripción y se carga con valores nulos, para evitar problemas. A continuación debemos averiguar si se trata de un dispositivo con pantalla grande o pequeña, y en función de ello, mostraremos los dos fragmentos al mismo tiempo ocupando cada uno media pantalla o de uno en uno y ocupando la pantalla entera. Se debe observar para esto la variable `smallScreen`, pues en función de su valor se carga un fichero de *layout* u otro. Se obliga también al dispositivo a colocar la pantalla en vertical. El código es diferente entre versiones, pues para versiones antiguas sólo existe una vista en vertical, ya que sólo hay dos orientaciones, mientras que para versiones actuales hay dos vistas verticales, habiendo en total cuatro orientaciones. En el caso de que nos encontremos en una pantalla pequeña, debemos hacer que el fragmento de calendario ocupe la pantalla entera y pueda desaparecer al pulsar sobre uno de los días para dejar paso al fragmento de descripción. Por último, como tarea final antes de dar por terminada la creación de esta actividad, leemos de base de datos todos los datos referentes a las actividades.

El segundo método que se define en el código es `onItemGridClick(String selectedDay, int selection)`, llamado cuando el usuario pulsa sobre uno de los días del calendario. Se pasan como parámetros el día seleccionado y el tipo de actividad seleccionada, que ahora explicaremos cómo se produce. Éste método obtiene una lista con las actividades que cumplan con los dos parámetros introducidos (que se hayan producido el día seleccionado y sean del tipo seleccionado) y las envía al segundo fragmento, que para pantallas grandes será actualizado mientras que para pantallas pequeñas será mostrado en toda la pantalla, ocultando el calendario.

Este es el funcionamiento de la actividad principal, sin embargo, debemos tener en cuenta como se muestra el calendario y los días. Las clases encargadas son `FragmentCalendar.java`, `CalendarAdapter.java` y `DayAdapter.java`. El primero de ellos se encarga de establecer un *layout* que contiene, en primer lugar, dos botones para cambiar de mes, un *GridView*, que no es otra cosa que una casilla con los días del mes y un *Spinner* que permite elegir las actividades a mostrar, si todas, o alguna en concreto. Sin embargo, para poder editar el estilo de cada una de las casillas, debemos ayudarnos de una clase aparte que gestione si el día de cada casilla tiene una actividad registrada. De ello se encarga la segunda clase, en `CalendarAdapter.java`.

```
/**
 * Create a new view for each item referenced by the Adapter
*/
public View getView(int position, View convertView, ViewGroup parent) {
        View v = convertView;
        boolean smallScreen = (mContext.getResources().getConfiguration().screenLayout
                              & Configuration.SCREENLAYOUT_SIZE_MASK) <=
                              Configuration.SCREENLAYOUT_SIZE_NORMAL;

        if (convertView == null) {
                // If it's not recycled, initialize some attributes
                LayoutInflater vi = (LayoutInflater) mContext
                        .getSystemService(Context.LAYOUT_INFLATER_SERVICE);
                if(smallScreen)         v = vi.inflate(R.layout.calendar_item, null);
                else    v = vi.inflate(R.layout.calendar_item_with_icons, null);
        }

        TextView dayView = (TextView) v.findViewById(R.id.dateCalendarItem);

        String gridValue = dayString.get(position);
        Calendar todayGrid = Calendar.getInstance();
```

```
try{
        try {
                todayGrid = formater.parseToCalendarFromDate(gridValue);
        } catch (ParseException e) {
                Log.e("FALLO1",e.toString());
        }

        if((todayGrid.get(Calendar.DAY_OF_MONTH) > 1) && (position < firstDay)) {
                dayView.setTextColor(Color.GRAY);
                dayView.setClickable(false);
                dayView.setFocusable(false);
        }
        else if((todayGrid.get(Calendar.DAY_OF_MONTH) < 7) && (position > 28)){
                dayView.setTextColor(Color.GRAY);
                dayView.setClickable(false);
                dayView.setFocusable(false);
        }
        else{
                dayView.setTextColor(Color.BLACK);
        }

        v.setBackgroundColor(Color.WHITE);
        v.setPadding(3, 3, 3, 3);
        dayView.setText(String.valueOf(todayGrid.get(Calendar.DAY_OF_MONTH)));
        String monthStr = "" + (month.get(Calendar.MONTH) + 1);
        if (monthStr.length() == 1) monthStr = "0" + monthStr;
        Calendar dateTime = Calendar.getInstance();
        if(smallScreen){
                if(activ!=null && dates!=null){
                for(int a=0;a<activ.length;a++){
                        try {
                                dateTime = formater.parse(dates[a], "00:00 am");
                        } catch (ParseException e) {
                                Log.e("CalendarAdapter", e.toString());
                        }
                        if((dateTime.get(Calendar.YEAR) == todayGrid.get(Calendar.YEAR))
                        && (dateTime.get(Calendar.MONTH) == todayGrid.get(Calendar.MONTH))
                        && (dateTime.get(Calendar.DAY_OF_MONTH) ==
                        todayGrid.get(Calendar.DAY_OF_MONTH))){
                                                v.setBackgroundColor(Color.GREEN);
                        }
                }
                }
        }
        else{
                ImageView [] icons =
                        {(ImageView) v.findViewById(R.id.calendar_item_icon_1),
                         (ImageView) v.findViewById(R.id.calendar_item_icon_2),
                         (ImageView) v.findViewById(R.id.calendar_item_icon_3),
                         (ImageView) v.findViewById(R.id.calendar_item_icon_4),
                         (ImageView) v.findViewById(R.id.calendar_item_icon_5),
                         (ImageView) v.findViewById(R.id.calendar_item_icon_6),
                         (ImageView) v.findViewById(R.id.calendar_item_icon_7)};

                for(int x=0;x<7;x++){
                        icons[x].setImageResource(R.drawable.whitesquare);
                }
                if(activ!=null && dates!=null){
                        int count_icons = 0;
                        for(int a=0;a<activ.length;a++){
                        try {
                                dateTime = formater.parse(dates[a], "00:00 am");
                        } catch (ParseException e) {
                                Log.e("CalendarAdapter", e.toString());
                        }

                        if((dateTime.get(Calendar.YEAR) == todayGrid.get(Calendar.YEAR)) &&
                        (dateTime.get(Calendar.MONTH) == todayGrid.get(Calendar.MONTH)) &&
                        (dateTime.get(Calendar.DAY_OF_MONTH)==
                                todayGrid.get(Calendar.DAY_OF_MONTH))){
                                        if(count_icons<7){
                                                icons[count_icons]
                                                .setImageResource(getSmallIcon(activ[a]));
                                                count_icons ++;
```

```
                                                }
                                        }
                                }
                        }
                }
        }
        catch(Exception e){
                Log.e("CalendarAdapter",e.toString());
        }

        return v;
}
```

Para pantallas de gran tamaño, se ha querido enriquecer visualmente la aplicación, por lo que los días en los que se produzcan actividades, esta clase añadirá a la casilla un icono identificativo de la actividad. Sin embargo, para pantallas pequeñas, se coloreará de verde la casilla, pues no habría espacio suficiente para todos los iconos. Por último, DayAdapter.java define una clase que instanciaremos para obtener un objeto que almacene el día al que corresponde cada casilla, es decir, casilla 1 día 28 de febrero (mes pasado), casilla 2 día 1 de marzo (mes actual), y así sucesivamente hasta completar una cuadrícula entera.

Por último, debemos detallar el proceso de mostrar la descripción de los diferentes días. En primer lugar, se muestra el contenido de la clase `FragmentNothingSelected.java`, utilizada cuando se crea la actividad y no se ha seleccionado ningún día. La clase `FragmentSpec.java` muestra una lista con las actividades del día seleccionado y que recibe desde la actividad principal del calendario, mostrándolas en un *ListView* cuyo formato esta especificado a través de la clase `Row.java` y el fichero `fragment_spec.xml`. La lista con las actividades se irá actualizando dependiendo del día en el que el usuario pulse.

Para acabar este apartado, queda por detallar todo lo referente al dibujo de gráficas. En primer lugar, tras acceder a esta sección de la aplicación, se muestra la actividad SelectMonthYear.java, que permite escoger qué gráficas deseamos mostrar, si de análisis de sangre, de presión sanguínea o de glucosa, a través de un *Spiner*, y el mes del que consultar los datos a través de dos *NumberPicker*, uno para el mes y otro para el año. El procedimiento utilizado para las tres es el mismo, por lo que simplemente se mostrará el detalle para una de ellas, en concreto, para la representación de las gráficas de glucosa.

El método utilizado para ello es similar al de la actividad principal de la aplicación. Una actividad con un *ViewPager* que muestre varias páginas, cada una con una gráfica distinta. La actividad principal que contiene esta vista es GlucosePlotActivity.java, y sus similares para los otros dos tipos de gráficas. La clase GlucoseFragment.java especifica el fragmento que contendrá las diferentes páginas, que cargarán gráficas diferentes. La clase GlucoseChart.java proporciona las vistas que forman la gráfica. Para ello simplemente se hace uso de la biblioteca *afreechart*. Mediente diversos comandos, cargando la información desde la base de datos, se construyen las gráficas. El contenido de dicho fichero es el siguiente, y muestra cómo devuelve los distintos objetos que contienen las gráficas.

```
public class GlucoseChart extends DemoView
{
        Context context;
        int month, year;
        private Formatter fmt;
        AFreeChart [] chart;
        int chartNum;
        int maxNumOfDays;
```

```
public GlucoseChart(Context context, int month, int year, int chartNum)
{
	super(context);
	this.context = context;
	this.month = month;
	this.year = year;
	this.chartNum = chartNum;
	chart = createChart();
	setChart(chart[chartNum]);
}


public AFreeChart [] createChart() {
	DateAxis domainAxis = new DateAxis(context.getString(R.string.date));
	domainAxis.setTickMarkPosition(DateTickMarkPosition.MIDDLE);
	ValueAxis rangeAxis = new NumberAxis("Value");

	IntervalXYDataset [] dataSet = createDataset();
	XYItemRenderer renderer1 = new XYBarRenderer(0.20);

	XYPlot plot1 = new XYPlot(dataSet[0],
		domainAxis, rangeAxis, renderer1);
	XYPlot plot2 = new XYPlot(dataSet[1],
		domainAxis, rangeAxis, renderer1);

	plot1.setDatasetRenderingOrder(DatasetRenderingOrder.FORWARD);
	plot1.setOrientation(PlotOrientation.VERTICAL);

	plot2.setDatasetRenderingOrder(DatasetRenderingOrder.FORWARD);
	plot2.setOrientation(PlotOrientation.VERTICAL);

	AFreeChart [] chart = {new AFreeChart("Glucosa",
		AFreeChart.DEFAULT_TITLE_FONT, plot1, false),
		new AFreeChart("HB1Ac",
		AFreeChart.DEFAULT_TITLE_FONT, plot2, false)};

	return chart;
    }

/**
 * Creates a sample dataset.
 * @return The dataset.
 */
public IntervalXYDataset [] createDataset() {
	fmt = new Formatter();
	Cursor cursor;
	TimeSeries seriesGlucose;
	TimeSeries seriesHb;

	Calendar calendar = Calendar.getInstance();
	calendar.set(Calendar.MONTH, month);
	maxNumOfDays = calendar.getMaximum(Calendar.DAY_OF_MONTH);

	DBAdapter dbAdapter = new DBAdapter(context);
	dbAdapter.open();
	// Cadena que usaremos para la petición a la base de datos
	// Queremos solo las medidas de glucosa
```

```
        realizadas en el mes introducido por el usuario.
        String selection =
                    DBAdapter.COLUMN_DATE + " LIKE '%" +
                    fmt.format("%1$02d",month)
                    + "-"  + String.valueOf(year) + "%';";

        cursor = dbAdapter.query(DBAdapter.TABLE_NAME_GLUCOSE,
                    null, selection, null, null, null, null, null);

        if(cursor.moveToFirst()){
              seriesGlucose = new TimeSeries("Glucose");
              seriesHb = new TimeSeries("Hb1Ac");

              for(int a=1;a<maxNumOfDays;a++){
                    cursor.moveToFirst();
                    do{
                          String [] date = cursor.getString(1).split("-");
                          if(Integer.parseInt(date[0])==a){
                                seriesGlucose.addOrUpdate(
                                      new Day(a, month, year),
                                Integer.parseInt(cursor.getString(2)));
                                seriesHb.addOrUpdate(
                                      new Day(a, month, year),
                                Double.parseDouble(cursor.getString(3)));
                                break;
                          }
                          else{
                                seriesGlucose.addOrUpdate(
                                      new Day(a, month, year),0);
                                seriesHb.addOrUpdate(
                                      new Day(a, month, year),0);
                                }
                          }while(cursor.moveToNext());
                    }
          }
          else{
              cursor.close();
              dbAdapter.close();
              return null;
          }
          dbAdapter.close();
          fmt.close();

          TimeSeriesCollection [] result = {
              new TimeSeriesCollection(seriesGlucose),
              new TimeSeriesCollection(seriesHb)};

          return result;
    }

}
```

Mapas

Los paquetes que se encargan de la gestión de los mapas son `es.uva.tel.gte.cardiomanager.jsonprovider` y `es.uva.tel.gte.cardiomanager.maps`.

El primero de los paquetes se trata de un conjunto de clases que nos ayudan a convertir coordenadas geográficas en direcciones:

- ✓ `PlaceProvider.java`: se trata del proveedor de contenidos para obtener las direcciones de la Google Places API.
- ✓ `PlaceJSONParser.java`: convierte el resultado recibido de Google Places Autocomplete API en formato JSON a formato accesible desde Java.
- ✓ `PlaceDetailsJSONParser.java`: convierte el resultado de Google Places Details API, en formato JSON, a formato accesible desde Java.

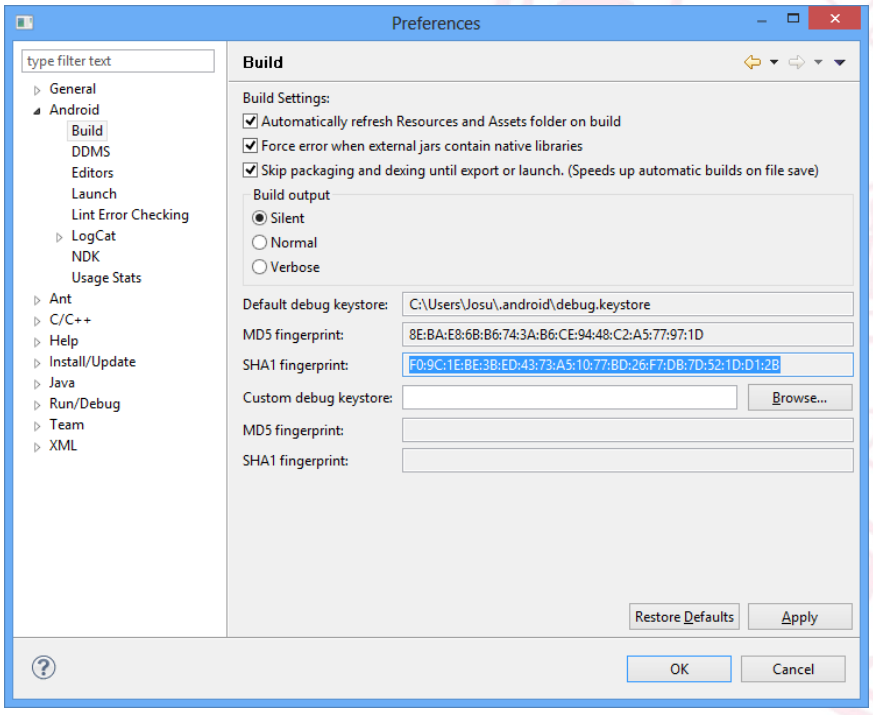
El desarrollo de estas clases ha sido realizado por George Mathew, experto en desarrollo de aplicaciones Android, que publicó las mismas para que todo el mundo tuviera acceso a ellas. Como no forman parte de este trabajo, no se muestra su contenido.

El segundo paquete se encarga de mostrar los mapas y gestionar los eventos de localización y mostrar ubicaciones. Las clases de este paquete son MapChooseLocation.java, accesible desde el registro de actividades para obtener una localización y guardarla, y en segundo lugar, LoadLocation.java, para mostrar una localización previamente guardada. A continuación, se muestra un fragmento de código de esta última clase:

```
GoogleMap mGoogleMap;

@Override
protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.map_main);

      Bundle bundle = getIntent().getExtras();
      address = bundle.getString("LOCATION");
      latitude = bundle.getDouble("LATITUDE");
      longitude = bundle.getDouble("LONGITUDE");

      SupportMapFragment fragment
            = (SupportMapFragment) getSupportFragmentManager()
                  .findFragmentById(R.id.map);
      mGoogleMap = fragment.getMap();
      mGoogleMap.setMyLocationEnabled(true);
      mGoogleMap.getUiSettings().setZoomControlsEnabled(true);
      mGoogleMap.getUiSettings().setCompassEnabled(true);
      mGoogleMap.getUiSettings().setMyLocationButtonEnabled(true);
      mGoogleMap.getUiSettings().setAllGesturesEnabled(true);

      position = showLocations(latitude, longitude, address);

      if (mGoogleMap != null) {
            mGoogleMap.setOnMyLocationChangeListener(
                  new GoogleMap.OnMyLocationChangeListener() {
                        @Override
                        public void onMyLocationChange(Location arg0) {
                              if(first_open){
                                    LatLng latlng
                                    = new LatLng(latitude, longitude);
```

```
                                        mGoogleMap.animateCamera(CameraUpdateFact
                                        ory.newLatLngZoom(latlng, 15));
                                        first_open = false;
                                    }
                                }});
                        }
                }

        ...

private LatLng showLocations(double lat, double lng, String address){
        MarkerOptions markerOptions = null;
        LatLng position = null;
        mGoogleMap.clear();
        markerOptions = new MarkerOptions();
        position = new LatLng(lat,lng);
        markerOptions.position(position);
        mGoogleMap.addMarker(markerOptions);
              mGoogleMap.animateCamera(CameraUpdateFactory
                    .newLatLngZoom(position, 15));
        return position;
}
...
```

Para poder hacer uso de los mapas de Google, es necesario registra la aplicación en la consola de Google para desarrolladores. Tras registrar la aplicación, debemos habilitar el uso de Google Maps Android API V2, y obtener una clave para la aplicación. Esta clave es generada automáticamente, tras introducir la clave SHA1 de nuestra herramienta Eclipse, utilizada para firmar digitalmente las aplicaciones, y el nombre del paquete. Esta clave puede verse en la imagen inferior, accesible desde las preferencias de Eclipse, y el nombre del paquete está especificado en la `AndroidManiufest.xml`. Tras obtener esta clave, debemos añadirla a este último fichero, tal y como se comentó en la sección referente al mismo.



Introducción de Nuevo Medicamento

Este módulo, en el que el usuario guardará un nuevo medicamento en la base de datos, está desarrollado en el paquete `es.uva.tel.gte.cardiomanager.medicinenew`. Debido a que es necesaria mucha información para establecer alarmas y guardar el medicamento, es necesario dividir la introducción de esta información en varias actividades.

La primera de ellas está implementada en el fichero MedicineMain.java. En él, el usuario introducirá el nombre del medicamento y notas que considere oportunas a través de dos vistas de tipo *EditText* y podrá escoger el tipo de tratamiento a través de un *Spinner* con las distintas opciones posibles. Los tipos de tratamiento son explicados en el manual de uso, por lo que no es necesario comentar nada más en esta sección, salvo que según se van seleccionando las opciones de este *Spinner*, se actualiza un *WebView* con la descripción del tipo de tratamiento seleccionado.

En la segunda actividad se elegirán los parámetros relacionados con las alarmas, las horas, el número de dosis o las fechas de inicio y fin. Dependiendo del tipo de tratamiento, se pasará a una actividad u otra, implementadas en las siguientes clases:

- ✓ `MedicineEveryday.java`. Muestra diferentes vistas para la selección de estos parámetros.
  - *Spinner* de duración. Permite escoger si el tratamiento es indefinido o si tiene fechas de inicio y fin. En el caso de seleccionar la segunda opción, se muestran dos *DatePickers* personalizados de la bilbioteca de SimonVT para escoger estas fechas.
  - *Spinner* de número de dosis. Permite escoger de 1 a 5 dosis al día. Según el número seleccionado, se muestra un pequeño *layout* para cada una de las dosis que consiste en un botón de selección de hora y un texto con la misma. Al pulsar sobre los botones, se abre un diálogo que permite seleccionar la hora.
  - *CheckBox* que permite elegir si se desea mostrar una notificación y avisar al usuario cuando llegue la toma o no.
- ✓ `MedicineSelectPeriod.java`. En esta actividad se seleccionan los parámetros a través de:
  - *DatePicker* personalizado, que permite escoger la fecha de inicio de tratamiento.
  - *Button* que muestra el diálogo de selección de hora. Será la hora de inicio del tratamiento en este caso.
  - Dos *EditTexts* que permitan, uno escoger el número total de dosis a tomar, y el segundo las horas que deben pasar entre una dosis y otra.
  - *CheckBox* que permite elegir si se desea mostrar una notificación y avisar al usuario cuando llegue la toma o no.
- ✓ `MedicineOneDose.java`. En esta actividad se seleccionan los parámetros a través de:

- *DatePicker* personalizado, que permite escoger la fecha de la toma.
- *Button* que muestra el diálogo de selección de hora.
- *CheckBox* que permite elegir si se desea mostrar una notificación y avisar al usuario cuando llegue la toma o no.

✓ `MedicineSelectDay.java`. Similar a la clase `MedicineEveryday.java`. Simplemente añade un pequeo *layout* que permita escoger los días. Esto se realiza a través de una serie de *ToggleButtons*. Si aparecen marcados, se entiende que ese mismo día se deberá tomar la medicación, y si no lo están, no se deberá tomar.

El diálogo de selección de hora está implementado en la clase `TimePickerFragment.java`. Se trata de un diálogo que muestra dos *NumberPicker*, uno para la hora y otro para los minutos, además de dos botones propios de la aplicación.

La tercera pantalla o actividad es `MedicineSummary.java`, y en ella se muestra un resumen de los datos introducidos por el usuario, con el objetivo de que el paciente los revise y si es necesario, vuelva hacia atrás y los corrija. Si está de acuerdo con lo introducido, pulsando sobre la flecha superior derecha, que forma parte de la navegación de todo el módulo, se guardará en base de datos y se establecerá la primera alarma. El código para el establecimiento de alarmas se especifica más adelante.

Dado que este módulo no realiza ninguna función especial ni tiene aspectos reseñables, no se muestran fragmentos de código.

Registro de Medicamentos

Este módulo está formado por los siguientes paquetes y clases:

✓ `es.uva.tel.gte.cardiomanager.medicinerecord`
- `MedicineRecordMain.java`
- `MedicineList.java`
- `FragmentMedicine.java`
- `FragmentDescription.java`
- `PendingMedicines.java`
- `NotTakenMedicines.java`

✓ `es.uva.tel.gte.cardiomanager.medicinecalendar`
- `CalendarMedicineMain.java`
- `FragmentCalengarMedicines.java`
- `FragmentSpecMedicines.java`
- `CalendarMedicineAdapter.java`

- Row.java

En el primer paquete se encuentra la actividad principal del módulo, MedicineRecordMain.java, que contiene cuatro botones para acceder a las diferentes partes del módulo. En primer lugar tenemos la lista de medicamentos y su descripción. MedicineList.java es una clase que hereda de FragmentActivity, FragmentMedicine.java es un fragmento que carga la lista de los nombres de los medicamentos, mientras que FragmentDescription.java se trata de otro fragment que cargará la descripción del fragmento seleccionado en la lista. La actividad principal gestiona cómo, mostrando primero la lista de medicamentos a pantalla completa, se muestra también, a pantalla completa, la descripción del medicamento en un nuevo fragmento. Para el caso de grandes pantallas, los dos fragmentos se muestran al mismo tiempo, y no se sustituyen uno por otro. Esta filosofía es la misma empleada para los calendarios.

Además, la actividad PendingMedicines.java muestra una lista de las tomas de medicamentos pendientes de marcar. Cada elemento de esta lista tiene un *CheckBox* que el usuario podrá marcar. A continuación, tras seleccionar los elementos que desee, podrá marcarlos como tomados o no tomados pulsando uno de los dos botones habilitados para ello. La actividad NotTakenMedicines.java, por el contrario, muestra una lista con las tomas no realizadas.

Por último, el segundo paquete se encarga de mostrar en un calendario, las distintas tomas de los medicamentos. El funcionamiento es similar al del calendario de actividades, por lo que no se detallará más esta sección, resaltando simplemente que en este caso no se dibujan iconos, y que los días simplemente se colorean de verde o de rojo dependiendo de si hay dosis no tomadas para ese día.

Notificaciones y aviso de toma

Este módulo está formado por las clases Manager.java y Receiver.java del paquete , y por las clases AppService.java y CheckAlarmService.java del paquete . Sin embargo, la primera alarma de cada medicamento se establece a través del siguiente código, en el módulo de introducción de medicamento, en la actividad de MedicineSummary.java.

```
public void start(long time, int id, int postnotification){
      aManager = (AlarmManager) getSystemService(Context.ALARM_SERVICE);
      intent = new Intent(this, Receiver.class);
      dbAdapter = new DBAdapter(getApplicationContext());
      dbAdapter.open();

      // Código de la alarma
      // id -> Identificador de medicamento
      // time -> Tiempo en ms en el que debe saltar la primera alarma
      // 0 -> Valor 'false' para 'marked'. No ha sido marcada la dosis
      // 1 -> is_first=true, pues se trata de la
            primera alarma del tratamiento
      // Devuelve el ID_Pending, que será el REQUEST_CODE
            para las alarmas de este medicamento
      REQUEST_CODE = dbAdapter.insertPending(id, time, 0, 1);

      // Agregamos algunos datos extra al intent
      intent.putExtra("ID", (int)id);
      intent.putExtra("PostNotification", postnotification);
```

```
        intent.putExtra("Time", time);
        intent.putExtra("REQUEST_CODE", (int) REQUEST_CODE);

        // Creamos el PendingIntent que lanzará el anuncio Broadcast
        pendingIntent = PendingIntent.getBroadcast(this, (int) REQUEST_CODE,
              intent, PendingIntent.FLAG_UPDATE_CURRENT);

        // Establecemos la alarma
        aManager.set(AlarmManager.RTC_WAKEUP, time, pendingIntent);

        dbAdapter.close();

        startService(new Intent(this, AppService.class));
}
```

Como vemos, se hace uso del servicio de alarmas de Android, que permite avisar al dispositivo a cierta hora. El código utilizado para identificar la alarma será el ID que se nos devuelva al introducir la información del medicamento en la base de datos. A continuación se crea un *Intent* al receptor de anuncios *broadcast* y se le pasa información extra, como el código de alarma, la hora, o si se desea mostrar una notificación. Por último, se programa la alarma a la hora de la toma, y se inicia el servicio definido en `AppService.java`.

Este servicio no realiza ninguna función, simplemente está activo tras introducir un medicamento. Cuando la aplicación se cierra por completo, tanto de primer como de segundo plano, el servicio también termina. Este es el mecanismo utilizado para averiguar si la aplicación ha sido cerrada o no. Al iniciar la actividad principal, se comprueba si este servicio está activo. Si está activo, es que no se cerró la aplicación, si no está activo, puede ser que no se hayan introducido todavía medicamentos, o que la aplicación se cerró por completo y haya que reprogramar las alarmas.

`CheckAlarmServcie.java` es el servicio encargado de recuperar el funcionamiento normal de las alarmas tras un cierre completo de la aplicación. En el caso de que la aplicación permaneciera mucho tiempo parada y en este periodo de tiempo hayan tenido que tener lugar ciertas tomas, la aplicación las calcula y las coloca como pendientes de marcar. Además, se calcula cuál será la siguiente alarma en sonar, y se reprogramará, para poder retomar el funcionamiento normal de la aplicación.

Durante el funcionamiento normal de la aplicación, cuando llega la hora de tomar la medicación, el sistema realiza una llamada al *BroadcastReceiver* especificado en `Receiver.class`. Este receptor inicia el servicio `Manager.class`, que se encarga de postear una notificación y reprogramar la siguiente alarma si es necesario.

Aunque el código de estas clases es bastante largo y complejo, no aporta ninguna información especial, por lo que no se mostrarán más fragmentos de código de este módulo.

Información y Directrices

Estos dos módulos son diferentes en cuanto a objetivos y contenidos, pero muy similares en cuanto a implementación y programación se refiere. Se corresponden con los paquetes `es.uva.tel.gte.cardiomanager.info` y `es.uva.tel.gte.cardiomanager.guide` respectivamente.

En ambos casos se muestra un *Spinner* en el que se pueden seleccionar las distintas opciones de información y directrices, según lo especificado en el anterior apartado de diseño de la aplicación. Además, se muestran tres botones, referentes a '¿Qué es?', 'Síntomas' y 'Tratamiento' para Información y 'Hábitos saludables', 'Consejos' y 'Prohibiciones' para Directrices, que al pulsarlos, despliegan un *WebView* con el texto correspondiente.

Información del paciente

Este módulo está desarrollado por completo en el paquete `es.uva.tel.gte.cardiomanager.patient`, con las clases:

- ✓ `Login.java`
- ✓ `NewPatientActivity.java`
- ✓ `PatientMainActivity.java`
- ✓ `PatientInfoList.java`
- ✓ `NewPatientInfo.java`
- ✓ `ContactActivity.java`
- ✓ `ChoosePhoneActivity.java`
- ✓ `ContactAdapter.java`
- ✓ `ContactBean.java`

Al acceder a este módulo de la aplicación, se abre la actividad de `Login.java`, que permite al usuario introducir su nombre de usuario y contraseña. Si no se ha registrado previamente un usuario, no se podrá acceder al módulo, y se tendrá que registrar un nuevo usuario a través de `NewPatientActivity.java`. En esta actividad se seleccionará un nombre de usuario, contraseña, y otros datos de información personal que permanecerán cifrados en la base de datos. Más tarde se especificará el proceso seguido para cifrar y descifrar información. Habiendo registrado ya un usuario, se muestra la página principal de usuario, `PatientMainActivity.java`. En esta actividad se muestra, en primer lugar un botón que al ser pulsado despliega un *WebView* con la información personal de usuario. El segundo botón, permite acceder a la página principal de contacto de emergencia, y debajo, se muestra un *ListView* con cinco tipos elementos, detallados en la sección de diseño de la aplicación, que amplían la información personal de usuario. Al pulsar, por ejemplo, en el elemento de alergias, se abre una nueva actividad, `PatientInfolist.java`, con una lista de las alergias introducidas por el usuario. Al mantener pulsado el dedo sobre una de ellas se permite editarlas. Además, a través del menú se permiten añadir nuevas. Tanto a la hora de añadir como de editar, se utiliza la actividad `NewPatientInfo.java`.

En la actividad de contacto de emergencia, `ContactActivity.java`, se muestra el contacto del teléfono que el usuario ha marcado como contacto de emergencia. Además, se muestran dos botones, uno para realizar llamadas y otro para enviar un mensaje a dicho contacto. El código utilizado para ello es bastante interesante, por lo que queda plasmado a continuación.

Para realizar llamadas:

```
btnCall.setOnClickListener(new View.OnClickListener() {
      public void onClick(View v) {
            Toast.makeText(getApplicationContext(),
                        getResources().getString(R.string.calling),
                        Toast.LENGTH_SHORT).show();
            String phoneNumber = "tel:" + phoneNo;
            Intent intent = new Intent(Intent.ACTION_CALL,
                  Uri.parse(phoneNumber));
            startActivity(intent);
      }
});
```

Y para enviar mensajes:

```
btnMessage.setOnClickListener(new View.OnClickListener() {
      public void onClick(View v) {
            Uri uri = Uri.parse("smsto:" + phoneNo);
            Intent intent = new Intent(Intent.ACTION_SENDTO, uri);
            intent.putExtra("sms_body",
                  getString(R.string.app_name) + " " +
                  getString(R.string.help));
            startActivity(intent);
      }
});
```

Como vemos, sólo es necesario obtener el nombre que hemos almacenado como contacto e iniciar una actividad de llamada o de envío de mensajes, añadiendo cómo información extra los datos del contacto.

En el caso de que no se haya seleccionado un contacto de emergencia, se dispone de un botón para dicha función. Al pulsar sobre él, se abre una actividad, `ChoosePhoneActivity.java`, con una lista con todos los contactos almacenados en el dispositivo. Al pulsar sobre uno de ellos, se abrirá un diálogo, que preguntará si desea que el contacto seleccionado sea el contacto de emergencia. Las clases `ContactAdapter.java` y `ContactBean.java` se utilizan para mostrar la lista de contactos. Para obtener la información de los contactos, se emplea el siguiente fragmento de código:

```
Cursor mCursor = getContentResolver().query(
      ContactsContract.CommonDataKinds.Phone.CONTENT_URI,
      null, null, null, null);

if(mCursor.moveToFirst()){
      while (mCursor.moveToNext()) {
            String name = mCursor.getString(mCursor
                  .getColumnIndex(ContactsContract.CommonDataKinds
                        .Phone.DISPLAY_NAME));
            String phoneNumber = mCursor
                  .getString(mCursor
                        .getColumnIndex(ContactsContract.CommonDataKinds.
                        Phone.NUMBER));
            ContactBean objContact = new ContactBean();
            objContact.setName(name);
            objContact.setPhoneNo(phoneNumber);
            list.add(objContact);
      }
```

```
}
```

Capa de seguridad AES

En la clase `SecurityUtils.java`, del paquete `es.uva.tel.gte.cardiomanager.security`, se encuentra desarrollada toda la sección de seguridad. El código utilizado para cifrar datos es el siguiente:

```
public String encrypt(String password, String plainText){
	int saltLength = keyLength / 8;		// same size as key output
	try{
		//Final key creation process
		SecureRandom random = new SecureRandom();
		byte[] salt = new byte[saltLength];
		random.nextBytes(salt);
		KeySpec keySpec = new PBEKeySpec(password.toCharArray(),
			salt, iterationCount, keyLength);
		//Check times with PBEWITHSHA256AND256BITAES-CBC-BC
		SecretKeyFactory keyFactory =
			SecretKeyFactory.getInstance("PBKDF2WithHmacSHA1");
		byte[] keyBytes =
			keyFactory.generateSecret(keySpec).getEncoded();
		SecretKey key = new SecretKeySpec(keyBytes, "AES");

		//Creation of the cipher to encrypt the the text
		Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
		//Encryption algorithm selection
		byte[] iv = new byte[cipher.getBlockSize()];
		random.nextBytes(iv);
		IvParameterSpec ivParams = new IvParameterSpec(iv);
		//Initialization Vector (iv) introduction
		cipher.init(Cipher.ENCRYPT_MODE, key, ivParams);
		byte[] ciphertext = cipher.doFinal(plainText.getBytes("UTF-8"));
		//Execution of the encryption
		String encodeText
			= Base64.encodeToString(ciphertext, Base64.DEFAULT);
			//Conversion bytes to string
		String encodeFinal
			= encodeText + separator + Base64.encodeToString(salt,
			 Base64.DEFAULT) + separator + Base64.encodeToString(iv,
			 Base64.DEFAULT);
			// Addition to the encrypted text of the
				salt and the iv (string), which will be
				stored in the database
		return encodeFinal;
	}
	catch (Exception e) {
		Log.e("Error SecurityUtils", "Encrypt: " + e.toString());
		return null;
	}
}
```

Y el código para descifrar es:

```
public String decrypt(String encodeFinal, String password){
	try{
		//Extraction of the coded text, the salt and the iv separately
		String[] fields = encodeFinal.split(separator);
```

```
                byte[] cipherBytes = Base64.decode(fields[0], Base64.DEFAULT);
                byte[] salt = Base64.decode(fields[1], Base64.DEFAULT);
                byte[] iv = Base64.decode(fields[2], Base64.DEFAULT);

                //Obtaining the final key the same way as in the encryption part
                KeySpec keySpec = new PBEKeySpec(password.toCharArray(),
                      salt, iterationCount, keyLength);
                //Check times with PBEWITHSHA256AND256BITAES-CBC-BC
                SecretKeyFactory keyFactory =
                      SecretKeyFactory.getInstance("PBKDF2WithHmacSHA1");
                byte[] keyBytes =
                      keyFactory.generateSecret(keySpec).getEncoded();
                SecretKey key = new SecretKeySpec(keyBytes, "AES");

                //Creation of the cipher to decrypt the text
                //Decryption algorithm selection
                Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
                IvParameterSpec ivParams = new IvParameterSpec(iv);
                //Initialization Vector (iv) introduction
                cipher.init(Cipher.DECRYPT_MODE, key, ivParams);
                //Execution of the decryption
                byte[] plaintext = cipher.doFinal(cipherBytes);
                String plainrStr = new String(plaintext , "UTF-8");
                return plainrStr;
          }
          catch (Exception e) {
                Log.e("Error SecurityUtils", "Decrypt: " + e.toString());
                return null;
          }
    }
```

Calculadora de Riesgo Cardiovascular

Este módulo se encuentra recogido por completo en el paquete `es.uva.tel.gte.cardiomanager.cardiorisk`. Su página principal está gestionada por la actividad `CardioRiskMain.java`. En ella, se muestran dos botones con un estilo similar al de la aplicación. El primero de ellos nos conduce a la calculadora en sí, y el segundo hacia el registro de cálculos. Además, a través del menú se puede consultar la ayuda de esta sección, gestionada por la clase `CardioRiskHelp.java`.

Al pulsar sobre el primer botón, accedemos a la actividad `CardioRiskInputData.java`, que miestra un *layout* compuesto por:

- ✓ Una serie de *Spinners* para la selección de sexo, colesterol, HDL, presión sanguínea, diabetes, fumador o país de alto o bajo riesgo.
- ✓ *EditText* para la elección de la edad.

Los dos métodos a través de los cuales se calcula el riesgo cardiovascular tienen distintos parámetros de entrada, sin embargo, todos ellos se recogen en esta actividad. Cuando el usuario pulsa en el botón de la parte inferior derecha de la pantalla, se mostrará el resultado en una nueva actividad, `CardioRiskResult.java`. Los resultados de los dos métodos se muestran tanto en texto, a través de *TextViews*, como en una barra de progreso, *ProgressBar*, para verlo de forma más atractiva visualmente. En la parte inferior de la pantalla se sitúa un botón para guardar la información del cálculo en la base de datos.

Para consultar las diferentes medidas, basta con pulsar en el segundo botón de la actividad principal de este módulo y se cargara la actividad `CardioRiskRecord.java`, que muestra una lista con las diferentes medidas, sus resultados, y los parámetros introducidos.

Base de datos

En Android es necesaria la creación de una clase que interactúe con la base de datos. En concreto, en este caso, se trata de la clase DBAdapter.java, dentro del paquete es.uva.tel.gte.cardiomanager.databasehelper. Para observar mejor cómo se ha realizado el diseño de la base de datos, se adjunta un fragmento de código con las cadenas que definen las tablas:

```
//These are the column names.
public static final String COLUMN_ID = "_id";

//Auxiliar text elements
public static final String TEXT_TYPE = " TEXT";
public static final String DATE_TYPE = " DATE";
public static final String INT_TYPE = " INT";
public static final String LONG_TYPE = " LONG";
public static final String DOUBLE_TYPE = " DOUBLE";
public static final String COMMA_SEP = ",";
public static final String EQUAL = "=";

  //Initial version and DATABASE NAME.
public static final int DATABASE_VERSION = 1;
public static final String DATABASE_NAME = "CardiomanagerDatabase.db";

 /*--------------------------------------------------------------------*
                            PATIENT REGISTER
*--------------------------------------------------------------------*/

  //Estos serán los nombres de las columnas
public static final String COLUMN_USER = "User";
public static final String COLUMN_NAME = "Name";
public static final String COLUMN_DESCRIPTION = "Description";
public static final String COLUMN_DATE = "Date";
public static final String COLUMN_PLACE = "Place";
public static final String COLUMN_SURNAME = "Surname";
public static final String COLUMN_ADDRESS = "Address";
public static final String COLUMN_CITY = "City";
public static final String COLUMN_BIRTH = "Date_of_birth";
public static final String COLUMN_NUM = "Social_security_number";
public static final String COLUMN_DOC = "Doctor";
public static final String COLUMN_ACCEPTED1 = "Accepted1";
public static final String COLUMN_ACCEPTED2 = "Accepted2";

//Nombres de las tablas dentro de la BD
public static final String TABLE_NAME_POLITICS_ACCEPTED = "Politics";
public static final String TABLE_NAME_MAP_MESSAGE = "MapMessage";
public static final String TABLE_NAME_USER = "User";
public static final String TABLE_NAME_ALLERGIES = "Allergies";
public static final String TABLE_NAME_ANTECEDENTS = "Antecedents";
public static final String TABLE_NAME_DISEASES = "Diseases";
public static final String TABLE_NAME_HABITS = "Toxic_habits";
public static final String TABLE_NAME_TRANSFUSIONS = "Transfusions";
```

```
public static final String TABLE_NAME_DATA = "Personal_data";
public static final String TABLE_NAME_CONTACT= "Contact";

//Cadenas para la creación de las tablas
public static final String SQL_COMMAND_CREATE_POLITICS_ACCEPTED
        = "CREATE TABLE IF NOT EXISTS "
        + TABLE_NAME_POLITICS_ACCEPTED + " (" +
        COLUMN_ID +  " INTEGER PRIMARY KEY AUTOINCREMENT" + COMMA_SEP +
        COLUMN_ACCEPTED1 + INT_TYPE + COMMA_SEP +
          COLUMN_ACCEPTED2 + INT_TYPE + " );";

public static final String SQL_COMMAND_CREATE_MAP_MESSAGE
        = "CREATE TABLE IF NOT EXISTS " + TABLE_NAME_MAP_MESSAGE + " (" +
          COLUMN_ACCEPTED1 + INT_TYPE + " );";

public static final String SQL_COMMAND_CREATE_USER
        = "CREATE TABLE IF NOT EXISTS " + TABLE_NAME_USER + " (" +
          COLUMN_ID +  " INTEGER PRIMARY KEY AUTOINCREMENT" + COMMA_SEP +
          COLUMN_USER + TEXT_TYPE + " );";

public static final String SQL_COMMAND_CREATE_ALLERGY
        = "CREATE TABLE IF NOT EXISTS " + TABLE_NAME_ALLERGIES + " (" +
          COLUMN_ID +  " INTEGER PRIMARY KEY AUTOINCREMENT" + COMMA_SEP +
          COLUMN_NAME + TEXT_TYPE + COMMA_SEP +
          COLUMN_DESCRIPTION + TEXT_TYPE + " );";

public static final String SQL_COMMAND_CREATE_DISEASE
        = "CREATE TABLE IF NOT EXISTS " + TABLE_NAME_DISEASES + " (" +
          COLUMN_ID +  " INTEGER PRIMARY KEY AUTOINCREMENT" + COMMA_SEP +
          COLUMN_NAME + TEXT_TYPE + COMMA_SEP +
          COLUMN_DESCRIPTION + TEXT_TYPE  + " );";

public static final String SQL_COMMAND_CREATE_ANTECEDENTS
        = "CREATE TABLE IF NOT EXISTS " + TABLE_NAME_ANTECEDENTS + " (" +
          COLUMN_ID +  " INTEGER PRIMARY KEY AUTOINCREMENT" + COMMA_SEP +
          COLUMN_NAME + TEXT_TYPE + COMMA_SEP +
          COLUMN_DATE + TEXT_TYPE + COMMA_SEP +
          COLUMN_PLACE + TEXT_TYPE + COMMA_SEP +
          COLUMN_DESCRIPTION + TEXT_TYPE + " );";

public static final String SQL_COMMAND_CREATE_HABITS
        = "CREATE TABLE IF NOT EXISTS " + TABLE_NAME_HABITS + " (" +
          COLUMN_ID +  " INTEGER PRIMARY KEY AUTOINCREMENT" + COMMA_SEP +
          COLUMN_NAME + TEXT_TYPE + COMMA_SEP +
          COLUMN_DESCRIPTION + TEXT_TYPE + " );";

public static final String SQL_COMMAND_CREATE_TRANSFUSIONS
        = "CREATE TABLE IF NOT EXISTS "
        + TABLE_NAME_TRANSFUSIONS + " (" +
          COLUMN_ID +  " INTEGER PRIMARY KEY AUTOINCREMENT" + COMMA_SEP +
          COLUMN_NAME + TEXT_TYPE + COMMA_SEP +
          COLUMN_DATE + TEXT_TYPE + COMMA_SEP +
          COLUMN_PLACE + TEXT_TYPE + COMMA_SEP +
          COLUMN_DESCRIPTION + TEXT_TYPE + " );";

public static final String SQL_COMMAND_CREATE_DATA
        = "CREATE TABLE IF NOT EXISTS " + TABLE_NAME_DATA + " (" +
          COLUMN_ID +  " INTEGER PRIMARY KEY AUTOINCREMENT" + COMMA_SEP +
          COLUMN_NAME + TEXT_TYPE + COMMA_SEP +
```

```
            COLUMN_SURNAME + TEXT_TYPE + COMMA_SEP +
            COLUMN_ADDRESS + TEXT_TYPE + COMMA_SEP +
            COLUMN_CITY + TEXT_TYPE + COMMA_SEP +
            COLUMN_BIRTH + TEXT_TYPE + COMMA_SEP +
            COLUMN_NUM + TEXT_TYPE + COMMA_SEP +
            COLUMN_DOC + TEXT_TYPE + " );";

    public static final String SQL_COMMAND_CREATE_CONTACT
            = "CREATE TABLE IF NOT EXISTS " + TABLE_NAME_CONTACT + " (" +
            COLUMN_ID +  " INTEGER PRIMARY KEY AUTOINCREMENT" + COMMA_SEP +
            COLUMN_NAME + TEXT_TYPE + COMMA_SEP +
            COLUMN_NUM + TEXT_TYPE + " );";


        /*-------------------------------------------------------*
                              MEDICINES TABLES
         *-------------------------------------------------------*/

      /* --- STRINGS FOR THE DATABASE --- */
      //COLUMN NAMES
    public static final String COLUMN_MEDICINE_NAME = "MedicineName";
    public static final String COLUMN_NOTES = "Notes";
    public static final String COLUMN_KIND_OF_TREATMENT = "Kind";
    public static final String COLUMN_FROM_DATE = "FromDate";
    public static final String COLUMN_TO_DATE = "ToDate";
    public static final String COLUMN_NUMBER_OF_DOSES = "NumberOfDoses";
    public static final String COLUMN_TOTAL_NUMBER_OF_DOSES
            = "TotalNumberOfDoses";
    public static final String COLUMN_DAYS = "Days";
    public static final String COLUMN_PERIOD = "Period";
    public static final String COLUMN_REG_DATE = "RegDate";
    public static final String COLUMN_DATETIME = "DateTime";
    public static final String COLUMN_NOTIF = "Notification";
    public static final String COLUMN_TABLE_NAME = "TableName";
    public static final String COLUMN_CANCELABLE = "Cancelable";
    public static final String COLUMN_TIME = "Time";
    public static final String COLUMN_REMAINING = "RemainingDoses";
    public static final String COLUMN_ID_PENDING = "ID_Pending";
    public static final String COLUMN_MARKED = "Marked";
    public static final String COLUMN_IS_ALARM = "IsAlarm";
    public static final String COLUMN_IS_FIRST = "IsFirst";
    public static final String COLUMN_D = "Sun";
    public static final String COLUMN_L = "Mon";
    public static final String COLUMN_M = "Tue";
    public static final String COLUMN_X = "Wed";
    public static final String COLUMN_J = "Thu";
    public static final String COLUMN_V = "Fri";
    public static final String COLUMN_S = "Sat";


    //TABLE NAMES
    public static final String TABLE_NAME_MEDICINES= "Medicines";
    public static final String TABLE_NAME_PENDING = "Pending";
    public static final String TABLE_NAME_NOT_TAKEN = "NotTaken";
    public static final String TABLE_NAME_TAKEN = "Taken";
    public static final String TABLE_NAME_TIMES = "Times";
    public static final String TABLE_NAME_DATES = "Dates";
    public static final String TABLE_NAME_PERIOD = "Period";
    public static final String TABLE_NAME_DAYS = "Days";
```

```
//TABLE CREATION STRINGS
public static final String SQL_COMMAND_CREATE_MEDICINES =
        "CREATE TABLE IF NOT EXISTS " + TABLE_NAME_MEDICINES + " (" +
        COLUMN_ID +  " INTEGER PRIMARY KEY AUTOINCREMENT" + COMMA_SEP +
        COLUMN_MEDICINE_NAME + TEXT_TYPE + COMMA_SEP +
        COLUMN_NOTES + TEXT_TYPE + COMMA_SEP +
        COLUMN_KIND_OF_TREATMENT + INT_TYPE + COMMA_SEP +
        COLUMN_NOTIF + INT_TYPE + COMMA_SEP +
        COLUMN_REG_DATE + LONG_TYPE + " );";

public static final String SQL_COMMAND_CREATE_PENDING =
        "CREATE TABLE IF NOT EXISTS " + TABLE_NAME_PENDING + " (" +
        COLUMN_ID_PENDING
              +  " INTEGER PRIMARY KEY AUTOINCREMENT" + COMMA_SEP +
        COLUMN_ID +  INT_TYPE + COMMA_SEP +
        COLUMN_DATETIME + LONG_TYPE + COMMA_SEP +
        COLUMN_MARKED + INT_TYPE + COMMA_SEP +
        COLUMN_IS_ALARM + INT_TYPE + COMMA_SEP +
        COLUMN_IS_FIRST + INT_TYPE + " );";

public static final String SQL_COMMAND_CREATE_NOT_TAKEN =
        "CREATE TABLE IF NOT EXISTS " + TABLE_NAME_NOT_TAKEN + " (" +
        COLUMN_ID +  INT_TYPE + COMMA_SEP +
        COLUMN_DATETIME + LONG_TYPE + " );";

public static final String SQL_COMMAND_CREATE_TAKEN =
        "CREATE TABLE IF NOT EXISTS " + TABLE_NAME_TAKEN + " (" +
        COLUMN_ID +  INT_TYPE + COMMA_SEP +
        COLUMN_DATETIME + LONG_TYPE + " );";

public static final String SQL_COMMAND_CREATE_TIMES =
        "CREATE TABLE IF NOT EXISTS " + TABLE_NAME_TIMES + " (" +
        COLUMN_ID +  INT_TYPE + COMMA_SEP +
        COLUMN_TIME + TEXT_TYPE + " );";

public static final String SQL_COMMAND_CREATE_DATES =
        "CREATE TABLE IF NOT EXISTS " + TABLE_NAME_DATES + " (" +
        COLUMN_ID +  INT_TYPE + COMMA_SEP +
        COLUMN_FROM_DATE + TEXT_TYPE + COMMA_SEP +
        COLUMN_TO_DATE + TEXT_TYPE + " );";

public static final String SQL_COMMAND_CREATE_PERIOD =
        "CREATE TABLE IF NOT EXISTS " + TABLE_NAME_PERIOD + " (" +
        COLUMN_ID +  INT_TYPE + COMMA_SEP +
        COLUMN_NUMBER_OF_DOSES + INT_TYPE + COMMA_SEP +
        COLUMN_REMAINING + INT_TYPE + COMMA_SEP +
        COLUMN_PERIOD+ INT_TYPE + " );";

public static final String SQL_COMMAND_CREATE_DAYS =
        "CREATE TABLE IF NOT EXISTS " + TABLE_NAME_DAYS + " (" +
        COLUMN_ID +  INT_TYPE + COMMA_SEP +
        COLUMN_D + INT_TYPE + COMMA_SEP +
        COLUMN_L + INT_TYPE + COMMA_SEP +
        COLUMN_M + INT_TYPE + COMMA_SEP +
        COLUMN_X + INT_TYPE + COMMA_SEP +
        COLUMN_J + INT_TYPE + COMMA_SEP +
        COLUMN_V + INT_TYPE + COMMA_SEP +
```

```
            COLUMN_S + INT_TYPE + " );";


    /*-------------------------------------------------------------*
                            ACTIVITIES TABLES
    *-------------------------------------------------------------*/

    //COLUMN NAMES
    public static final String COLUMN_OBSERVATIONS = "Observations";
    public static final String COLUMN_EXERCISE = "Exercise";
    public static final String COLUMN_MEASUREMENT_1 = "Measurement_1";
    public static final String COLUMN_MEASUREMENT_2 = "Measurement_2";
    public static final String COLUMN_INSULIN = "Insulin";
    public static final String COLUMN_KIND = "Kind";
    public static final String COLUMN_UNITS = "Units";
    public static final String COLUMN_SYMPTOM = "Symptom";
    public static final String COLUMN_CORPUSCLE = "Corpuscle";
    public static final String COLUMN_HEMOGLOBINE = "Hemoglobine";
    public static final String COLUMN_HEMATOCRIT = "Hematocrit";
    public static final String COLUMN_INR = "INR";
    public static final String COLUMN_UREA = "Urea";
    public static final String COLUMN_CREATININE = "Creatinine";
    public static final String COLUMN_HDL_CHORESTEROL = "HDL";
    public static final String COLUMN_LDL_CHORESTEROL = "LDL";
    public static final String COLUMN_TRIGLYCERIDE = "Triglycerides";
    public static final String COLUMN_PULSE = "Pulse";
    public static final String COLUMN_HB = "HbA1c";
    public static final String COLUMN_LOCATION_ADDRESS = "LocationAddres";
    public static final String COLUMN_LATITUDE = "Latitude";
    public static final String COLUMN_LONGITUDE = "Longitude";
    public static final String COLUMN_DURATION = "Duration";


    //TABLE NAMES
    public static final String TABLE_NAME_REHAB = "Rehab";
    public static final String TABLE_NAME_PHYSICAL = "PhysicalActivity";
    public static final String TABLE_NAME_BLOOD_TEST = "BloodTest";
    public static final String TABLE_NAME_BLOOD_PRESSURE = "BloodPressure";
    public static final String TABLE_NAME_GLUCOSE = "Glucosa";
    public static final String TABLE_NAME_ATTACK = "Attack";
    public static final String TABLE_NAME_EXCESS = "Excess";

    //TABLE CREATION STRINGS
    public static final String SQL_COMMAND_CREATE_REHAB =
            "CREATE TABLE IF NOT EXISTS " + TABLE_NAME_REHAB + " (" +
            COLUMN_ID +  " INTEGER PRIMARY KEY AUTOINCREMENT" + COMMA_SEP +
            COLUMN_DATE + TEXT_TYPE + COMMA_SEP +
            COLUMN_OBSERVATIONS + TEXT_TYPE + " );";

    public static final String SQL_COMMAND_CREATE_PHYSICAL =
            "CREATE TABLE IF NOT EXISTS " + TABLE_NAME_PHYSICAL + " (" +
            COLUMN_ID +  " INTEGER PRIMARY KEY AUTOINCREMENT" + COMMA_SEP +
            COLUMN_DATE + TEXT_TYPE + COMMA_SEP +
            COLUMN_EXERCISE + TEXT_TYPE + COMMA_SEP +
            COLUMN_DURATION + TEXT_TYPE + " );";

    public static final String SQL_COMMAND_CREATE_BLOOD_TEST =
            "CREATE TABLE IF NOT EXISTS " + TABLE_NAME_BLOOD_TEST + " (" +
```

```
            COLUMN_ID +  " INTEGER PRIMARY KEY AUTOINCREMENT" + COMMA_SEP +
            COLUMN_DATE + TEXT_TYPE  +  COMMA_SEP +
            COLUMN_CORPUSCLE + TEXT_TYPE  +  COMMA_SEP +
            COLUMN_HEMOGLOBINE + TEXT_TYPE  +  COMMA_SEP +
            COLUMN_HEMATOCRIT + TEXT_TYPE  +  COMMA_SEP +
            COLUMN_INR + TEXT_TYPE  +  COMMA_SEP +
            COLUMN_UREA + TEXT_TYPE  +  COMMA_SEP +
            COLUMN_CREATININE + TEXT_TYPE  +  COMMA_SEP +
            COLUMN_HDL_CHORESTEROL + TEXT_TYPE  +  COMMA_SEP +
            COLUMN_LDL_CHORESTEROL + TEXT_TYPE  + COMMA_SEP +
            COLUMN_TRIGLYCERIDE + TEXT_TYPE  +  " );";

    public static final String SQL_COMMAND_CREATE_BLOOD_PRESSURE =
            "CREATE TABLE IF NOT EXISTS "
            + TABLE_NAME_BLOOD_PRESSURE + " ("
            + COLUMN_ID +  " INTEGER PRIMARY KEY AUTOINCREMENT" + COMMA_SEP +
            COLUMN_DATE + TEXT_TYPE + COMMA_SEP +
            COLUMN_MEASUREMENT_1 + TEXT_TYPE + COMMA_SEP +
            COLUMN_MEASUREMENT_2 + TEXT_TYPE + COMMA_SEP +
            COLUMN_PULSE + TEXT_TYPE + " );";

    public static final String SQL_COMMAND_CREATE_GLUCOSE =
            "CREATE TABLE IF NOT EXISTS " + TABLE_NAME_GLUCOSE + " (" +
            COLUMN_ID +  " INTEGER PRIMARY KEY AUTOINCREMENT" + COMMA_SEP +
            COLUMN_DATE + TEXT_TYPE + COMMA_SEP +
            COLUMN_MEASUREMENT_1 + TEXT_TYPE + COMMA_SEP +
            COLUMN_HB + TEXT_TYPE +  COMMA_SEP +
            COLUMN_INSULIN + TEXT_TYPE + COMMA_SEP +
            COLUMN_KIND + TEXT_TYPE + COMMA_SEP +
            COLUMN_UNITS + TEXT_TYPE + " );";

    public static final String SQL_COMMAND_CREATE_ATTACK =
            "CREATE TABLE IF NOT EXISTS " + TABLE_NAME_ATTACK + " (" +
            COLUMN_ID +  " INTEGER PRIMARY KEY AUTOINCREMENT" + COMMA_SEP +
            COLUMN_DATE + TEXT_TYPE + COMMA_SEP +
            COLUMN_DURATION + TEXT_TYPE + COMMA_SEP +
            COLUMN_SYMPTOM + TEXT_TYPE + COMMA_SEP +
            COLUMN_LOCATION_ADDRESS + TEXT_TYPE + COMMA_SEP +
            COLUMN_LATITUDE + DOUBLE_TYPE + COMMA_SEP +
            COLUMN_LONGITUDE + DOUBLE_TYPE + ");";

    public static final String SQL_COMMAND_CREATE_EXCESS =
            "CREATE TABLE IF NOT EXISTS " + TABLE_NAME_EXCESS + " (" +
            COLUMN_ID +  " INTEGER PRIMARY KEY AUTOINCREMENT" + COMMA_SEP +
            COLUMN_DATE + TEXT_TYPE + COMMA_SEP +
            COLUMN_OBSERVATIONS + TEXT_TYPE + COMMA_SEP +
            COLUMN_KIND + TEXT_TYPE + " );";


    /*-------------------------------------------------------------------*
                                CARDIO RISK
    *-------------------------------------------------------------------*/
    //COLUMN NAMES
    public static final String COLUMN_SEX = "Sex";
    public static final String COLUMN_AGE = "Age";
    public static final String COLUMN_CHOLESTEROL = "Cholesterol";
    public static final String COLUMN_BLOOD_PRESSURE = "Pressure";
    public static final String COLUMN_DIABETIC = "Diabetic";
    public static final String COLUMN_SMOKER = "Smoker";
```

```
public static final String COLUMN_COUNTRY = "Country";
public static final String COLUMN_RESULT_FRAMINGHAM = "ResultFramingham";
public static final String COLUMN_RESULT_SCORE = "ResultScore";

//TABLE NAMES
public static final String TABLE_NAME_CARDIORISK = "Cardiorisk";

//TABLE CREATION STRINGS
public static final String SQL_COMMAND_CREATE_CARDIORISK =
        "CREATE TABLE IF NOT EXISTS " + TABLE_NAME_CARDIORISK + " (" +
        COLUMN_ID +  " INTEGER PRIMARY KEY AUTOINCREMENT" + COMMA_SEP +
        COLUMN_DATE + TEXT_TYPE + COMMA_SEP +
        COLUMN_RESULT_FRAMINGHAM + DOUBLE_TYPE + COMMA_SEP +
        COLUMN_RESULT_SCORE + DOUBLE_TYPE + COMMA_SEP +
        COLUMN_SEX + INT_TYPE + COMMA_SEP +
        COLUMN_AGE + INT_TYPE + COMMA_SEP +
        COLUMN_CHOLESTEROL + INT_TYPE + COMMA_SEP +
        COLUMN_HDL_CHORESTEROL + INT_TYPE + COMMA_SEP +
        COLUMN_BLOOD_PRESSURE + INT_TYPE + COMMA_SEP +
        COLUMN_DIABETIC + INT_TYPE + COMMA_SEP +
        COLUMN_SMOKER + INT_TYPE + COMMA_SEP +
        COLUMN_COUNTRY + INT_TYPE + " );";


// Creating INDEXES
public static final String SQL_COMMAND_CREATE_INDEX_REHAB =
        "CREATE INDEX IDX_REHAB ON " + TABLE_NAME_REHAB
              + "(" + COLUMN_DATE +")";

public static final String SQL_COMMAND_CREATE_INDEX_PHYSICAL =
        "CREATE INDEX IDX_PHYSICAL ON " + TABLE_NAME_PHYSICAL
              + "(" + COLUMN_DATE +")";

public static final String SQL_COMMAND_CREATE_INDEX_TEST =
        "CREATE INDEX IDX_TEST ON " + TABLE_NAME_BLOOD_TEST
              + "(" + COLUMN_DATE +")";

public static final String SQL_COMMAND_CREATE_INDEX_PRESSURE =
        "CREATE INDEX IDX_PRESSURE ON " + TABLE_NAME_BLOOD_PRESSURE
              + "(" + COLUMN_DATE +")";

public static final String SQL_COMMAND_CREATE_INDEX_GLUCOSE =
        "CREATE INDEX IDX_GLUCOSE ON " + TABLE_NAME_GLUCOSE
              + "(" + COLUMN_DATE +")";

public static final String SQL_COMMAND_CREATE_INDEX_ATTACK =
        "CREATE INDEX IDX_ATTACK ON " + TABLE_NAME_ATTACK
              + "(" + COLUMN_DATE +")";

public static final String SQL_COMMAND_CREATE_INDEX_EXCESS =
        "CREATE INDEX IDX_EXCESS ON " + TABLE_NAME_EXCESS
              + "(" + COLUMN_DATE +")";

public static final String SQL_COMMAND_CREATE_INDEX_TAKEN =
        "CREATE INDEX IDX_TAKEN ON " + TABLE_NAME_TAKEN
              + "(" + COLUMN_DATETIME +")";

public static final String SQL_COMMAND_CREATE_INDEX_NOT_TAKEN =
        "CREATE INDEX IDX_NOT_TAKEN ON " + TABLE_NAME_NOT_TAKEN
```

```
+ "(" + COLUMN_DATETIME +")";
```

En primer lugar, se han creado varias tablas recogiendo la información del usuario. Si ha acepatdo los términos de uso de la aplicación, de mapas, si ha introducido nombre de usuario y por lo tanto información peronal, y además, si haseleccionado un contacto de emergencia e introducido alergias, antecedentes quirúrgicos, etc.

En segundo lugar, para almacenar los medicamentos, se han creado diversas tablas, que detallaremos, ya que su entendimiento no es tan sencillo como en el resto de casos, y ha tenido que ser rediseñada varias veces.

- ✓ Tabla “Medicines”. En ella se almacena toda la información referente a un medicamento, como es el tipo de tratamiento, nombre, notas, fecha de registro o si se desea mostrar una notificación.
- ✓ Tabla “Pending”. Almacena las tomas que el usuario tiene pendientes de marcar.
- ✓ Tabla “NotTaken”. Almacena todas las dosis de los medicamentos que el usuario no toma.
- ✓ Tabla “Taken”. Almacena las dosis tomadas.
- ✓ Tabla “Times”. Almacena los tiempos en los que se debe tomar las dosis, en horas.
- ✓ Tabla “Dates”. Almacena las fechas de inicio y/o fin en caso de que existan y el tratamiento no sea indefinido.
- ✓ Tabla “Period”. Sólo para los tratamientos de tipo seleccionar periodo, guarda el periodo, número de dosis totales y número de dosis restantes.
- ✓ Tabla “Days”. Para el caso de tipo de tratamiento seleccionar días, guarda los días de la semana en los que se debe tomar el medicamento.

Para almacenar las actividades, se ha creado una tabla para cada una de ellas, pues los datos que almacenan son muy diferentes.

Y por último, parra la calculadora, basta con una tabla que almacene todos los cálculos. En ella guardamos tanto el resultado como los parámetros introducidos.

Paquete de clases auxiliares

El paquete `es.uva.tel.gte.cardiomanager.auxiliar` contiene varias clases auxiliares que son utilizadas durante toda la aplicación. Entre ellas cabe destacar:

- ✓ `Checker.java`: define métodos para comprobar si la hora o fecha que pasamos como parámetros es correcta, posterior o anterior a la actual.
- ✓ `Formater.java`: permite realizar conversiones entre Strings y objetos de tipo Calendar, Date o Integer.

Ayuda

Este módulo en el que se muestra la ayuda, consta simplemente de un *WebView* con el texto y la imagen correspondiente, explicando las funcionalidades de la aplicación.

Interfaz gráfica

Para el desarrollo de la interfaz gráfica se ha tenido en cuenta que es necesaria la compatibilidad entre diferentes pantallas, por ello se ha tenido en cuenta que es necesario utilizar diferentes *layouts* e imágenes de diferentes resoluciones. Android acepta esto gracias a la flexibilidad que nos ofrece la carpeta /res. En ella hemos podido especificar:

- ✓ Imágenes de distinta resolución, a través de los directorios:
  - res/drawable. Para imágenes que no necesitan ajustar su resolución.
  - res/drawable-ldpi. Para panatallas de baja resolución.
  - res/drawable-mdpi. Para panatallas de resolución media.
  - res/drawable-hdpi. Para panatallas de alta resolución.
  - res/drawable-xhdpi. Para panatallas de muy alta resolución.

  La nomenclatura dpi hace referencia a Density-independent Pixel, medida utilizada para la densidad de píxeles de una pantalla.
- ✓ *Layouts* que se ajustan dependiendo del tamaño de la pantalla y de la orientación. Se han utilizado los directorios:
  - res/layout: para *layouts* que no necesitan adaptarse a las pantallas ni a la orientación.
  - res/layout-land: para *layouts* que necesitan adaptarse a la orientación del dispositivo apaisada.
  - res/layout-large: para adaptar las vistas a pantallas más grandes.
  - res/layout-large-land: para adaptar las vistas a pantallas más grandes y con el dispositivo en apaisado.
  - res/layout-normal: para adaptar las vistas a pantallas normales.
  - res/layout-normal-land: para pantallas normales con el dispositivo en apaisado.
  - res/layout-small: para pantallas pequeñas.
- ✓ También se han adaptado el tamaño de los texto y de los márgenes estableciendo unas dimensiones para los mismos a través del fichero dimens.xml, en la carpeta /res/values. Además, para adaptarse a todas las pantallas, se ha creado un fichero dimens.xml diferente en las carpetas:
  - /res/values
  - /res/values-large

- /res/values-normal
- /res/values-small

De forma que se adapten los textos y los márgenes a todas las pantallas.

Por último, los botones han sido diseñados a través de imágenes y a través de ficheros XML, que especifican cuál es su apariencia sin ser pulsados y cuando son pulsados. Un ejemplo de esto es el siguiente código, referente al botón de educación:

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android" >
    <item android:state_pressed="true"
            android:drawable="@drawable/nurse_pressed" >
        <shape><stroke android:width="1dp"
                  android:color="#af1016" />
            </shape>
    </item>

      <item android:state_focused="true"
              android:drawable="@drawable/nurse_pressed" > <!-- focused -->
            <shape><stroke android:width="1dp"
                  android:color="#af1016" />
            </shape>
    </item>
    <item android:drawable="@drawable/nurse" /> <!-- default -->
</selector>
```

A continuación se muestra el contenido del fichero que define la textura de fondo de la aplicación:

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
android:shape="rectangle">
    <gradient
        android:startColor="#a60d13"
        android:endColor="#00000000"
        android:angle="45"/>
</shape>
```

BLOQUE 4
DESARROLLO DE LA APLICACIÓN

Requisitos de la aplicación

Al igual que todas las aplicaciones y programas software que se crean, *Cardiomanager* también tiene unos requisitos mínimos que debe cumplir el dispositivo en el que se ejecuta. Dichos requisitos han sido obtenidos bien sea a través de pruebas y tests en dispositivos reales, como son el caso de los requisitos a nivel técnico, o bien por decisiones tomadas durante la fase de desarrollo del software, requisitos más orientados hacia la funcionalidad y apariencia de la aplicación.

Dispositivo	Móvil o Tablet
Sistema operativo	Android OS
Versión de sistema operativo	> 2.2 *Froyo*
Espacio libre en almacenamiento interno	de 7 a 10 megabytes (MB)
Memoria RAM	al menos 384 MB
CPU	Tipo de CPU: Single (1 núcleo) Velocidad: al menos 800 megahercios (MHz)
Aceptación de permisos	(Puesto que se aprueban durante la instalación, serán especificados en la siguiente sección)

A continuación, se detallan y justifican los anteriores puntos:

Cardiomanager es una aplicación desarrollada de forma nativa para Android, por lo que el principal requisito es que el dispositivo disponga de un **sistema operativo Android**.

Como ya se comentó anteriormente, dentro de Android OS tenemos distintas versiones que han ido saliendo al mercado con el paso del tiempo. Actualmente todos los dispositivos con Android están equipados con al menos la versión 2.2 *Froyo*, que se corresponde con el nivel de API 8. A la hora de programar la aplicación se ha tenido en cuenta la compatibilidad entre las distintas versiones, adaptando el código y el uso de las bibliotecas para conseguir un correcto funcionamiento. Por lo tanto, *Cardiomanager* está preparada para funcionar en todos aquellos dispositivos con Android, **independientemente de la versión**.

Se requieren unos 7MB de espacio para poder copiar los ficheros correspondientes a la aplicación. Sin embargo, según se va haciendo uso de la aplicación y se van introduciendo medicamentos, alarmas y actividades, este espacio necesario aumenta levemente. Por lo tanto, unos **10 MB** serán suficientes para albergar esta aplicación en nuestro dispositivo, algo perfectamente asumible por cualquier dispositivo.

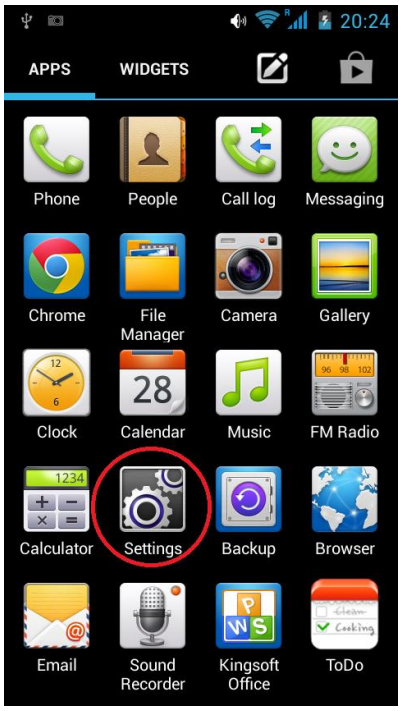
Cardiomanager hace uso de los servicios de mapas de Google, necesita dibujar mapas, WebViews y dos calendarios que se construyen elemento a elemento y programar alarmas y mostrar notificaciones. Por lo tanto, en algunos momentos, la carga computacional puede ser

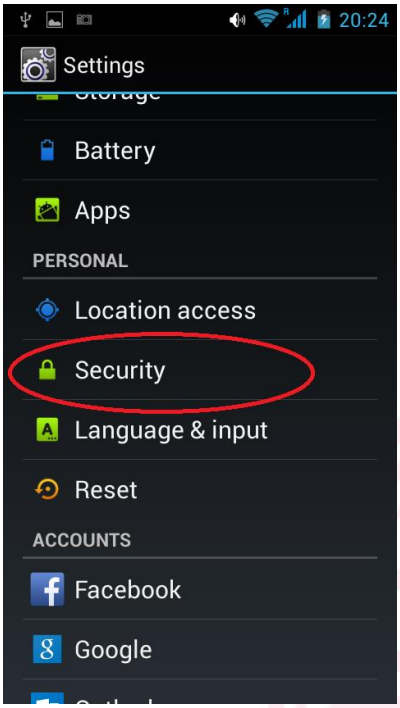
alta. La aplicación ha sido probada, entre otros, con un dispositivo con un procesador de un núcleo y una velocidad de reloj de 800 MHz, y una memoria RAM de 384 MB. El funcionamiento de la aplicación fue correcto, con algunas esperas a la hora de mostrar los calendarios y los mapas, dependientes también de la velocidad de la red. Al ser probada en dispositivos de mayor calidad y velocidad, el rendimiento mejoraba. Por lo tanto, para garantizar un correcto funcionamiento de la aplicación, se indican las características del peor dispositivo con el que se han conseguido buenos resultados, desconociendo el rendimiento de la aplicación para dispositivos con menor memoria o velocidad de procesador. Sin embargo, actualmente, la mayoría de los dispositivos disponen de una memoria RAM de al menos 512 MB, y de un procesador de dos núcleos con al menos 1 GHz de velocidad de reloj, por lo que podemos asegurar un correcto funcionamiento para todos los dispositivos.

No se necesita cumplir **ningún requisito en cuanto a la resolución, profundidad de color o tamaño de pantalla**, pues como ya se ha comentado previamente, la aplicación ha sido diseñada y programada para adaptarse a todos los tipos de resoluciones y tamaños de pantalla disponibles en el mercado (y que Android soporta), mientras que es el propio sistema operativo el que se encarga de la adaptación de colores.

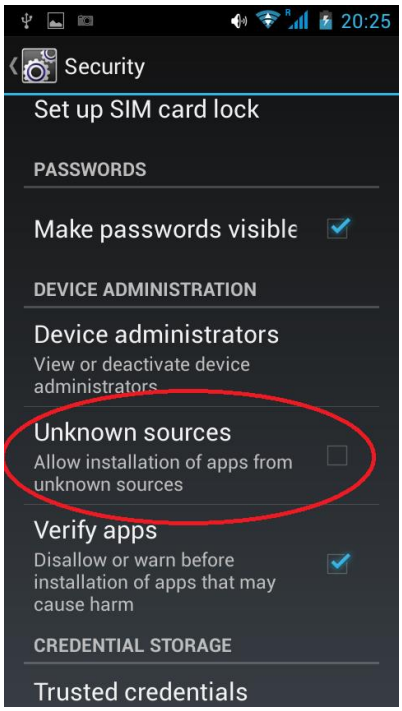
Instalación

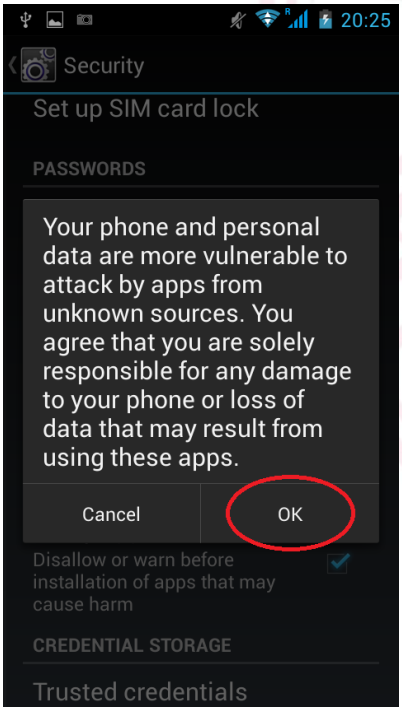
Podemos instalar la aplicación de dos formas diferentes. La primera consiste en descargarnos la aplicación desde el Google Play. Automáticamente el sistema ejecutará el instalador de la aplicación. La segunda consiste en guardar en el dispositivo el fichero `Cardiomanager.apk` proporcionado en el CD, que no es sino el instalador de la aplicación, y ejecutarlo. La desventaja de esta segunda opción es que debemos habilitar la instalación de aplicaciones desde un origen distinto a Google Play.

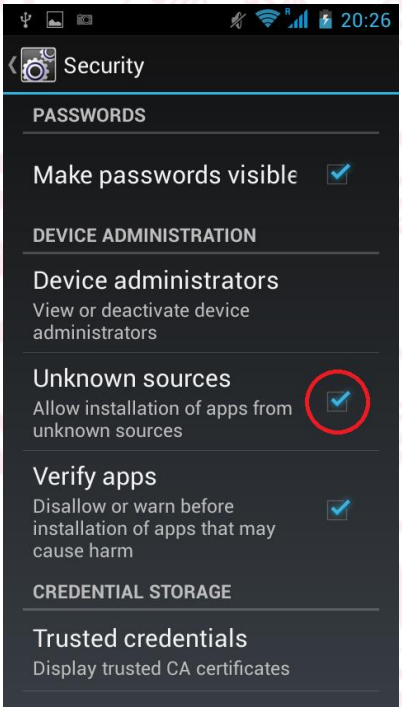




Para ello, debemos marcar la opción **Orígenes desconocidos** en **Ajustes** > **Seguridad** > **Administración de dispositivo**.
En inglés Settings > Security > Device administration > Unknown sources.





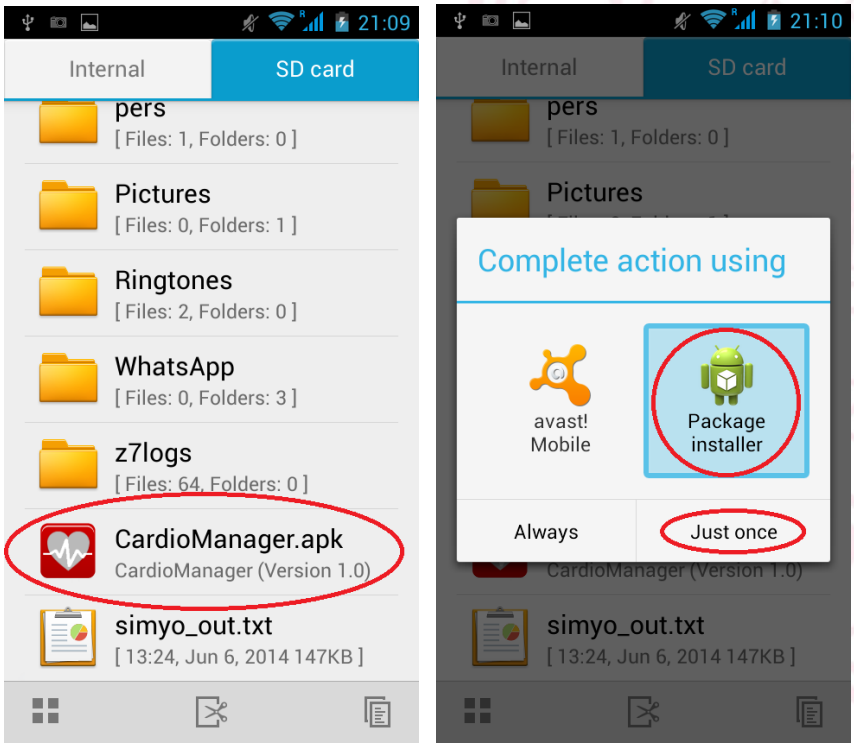


A continuación se muestra cómo proceder con la instalación de la aplicación para el

segundo caso expuesto.

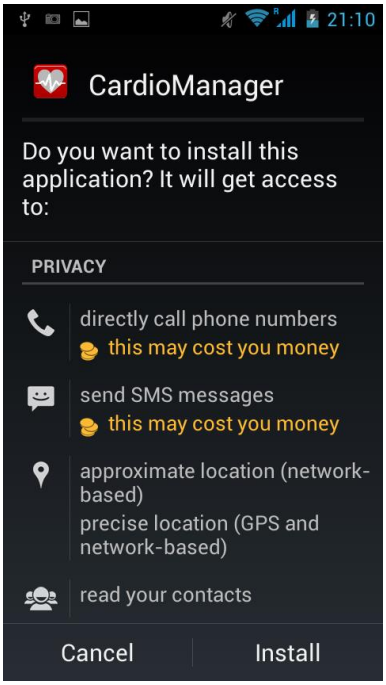
En primer lugar, debemos abrir el **Gestor de ficheros** de nuestro teléfono. Como cada fabricante implanta su UI (*User Interface*) propia, el nombre puede variar entre unos dispositivos y otros, pero su función de explorador de ficheros sigue siendo la misma.

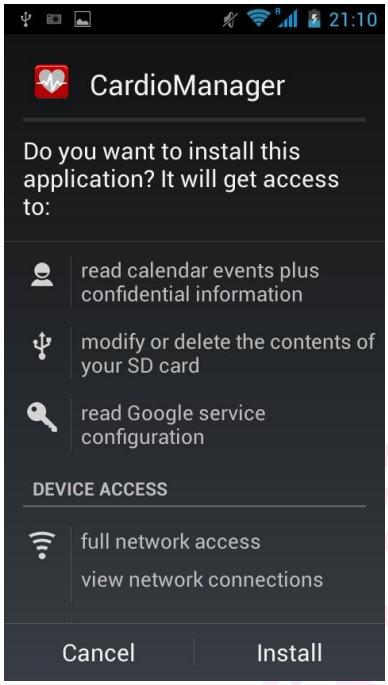
Una vez abierto este explorador, seleccionamos la ubicación donde hemos copiado nuestro instalados `CardioManager.apk` y pulsamos sobre el mismo.

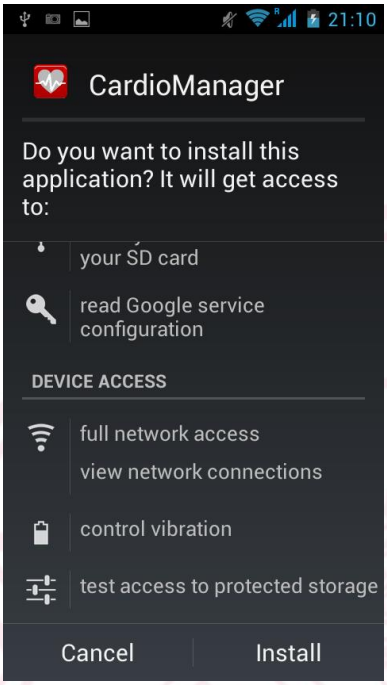


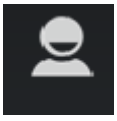
Seleccionamos el **Instalador de paquetes** de Android como el encargado de instalar la aplicación. Esta es la opción más común, aunque si tenemos instalado algún antivirus u otro programa similar también lo podemos utilizar, aunque será diferente la instalación.

A continuación se nos mostrarán en pantalla una serie de elementos que nos indican los elementos a los que accederá la aplicación. Debemos aceptar estos permisos para poder seguir con la instalación. Si el usuario no desea aceptar alguno de los permisos, no podrá hacer uso de la aplicación. A continuación se describen los distintos permisos que necesita que sean aprobados:

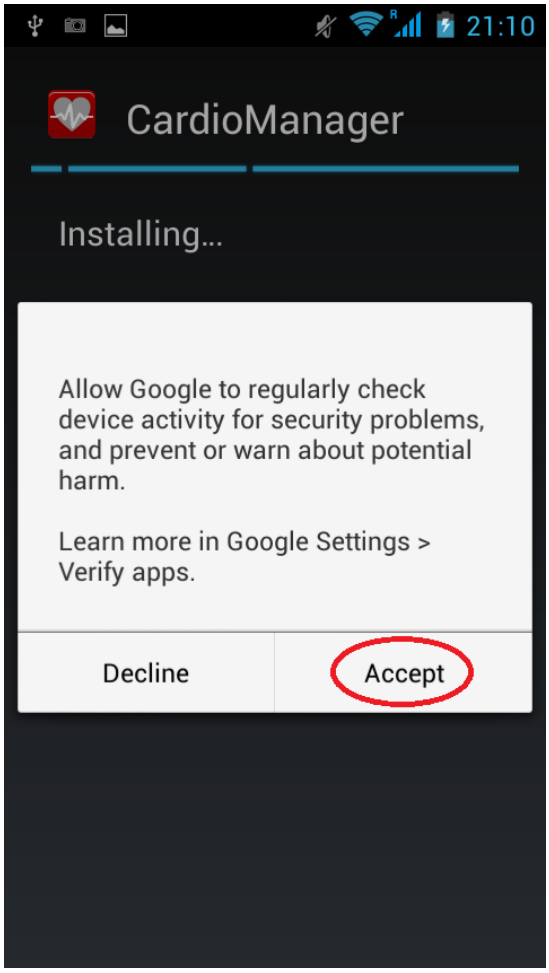


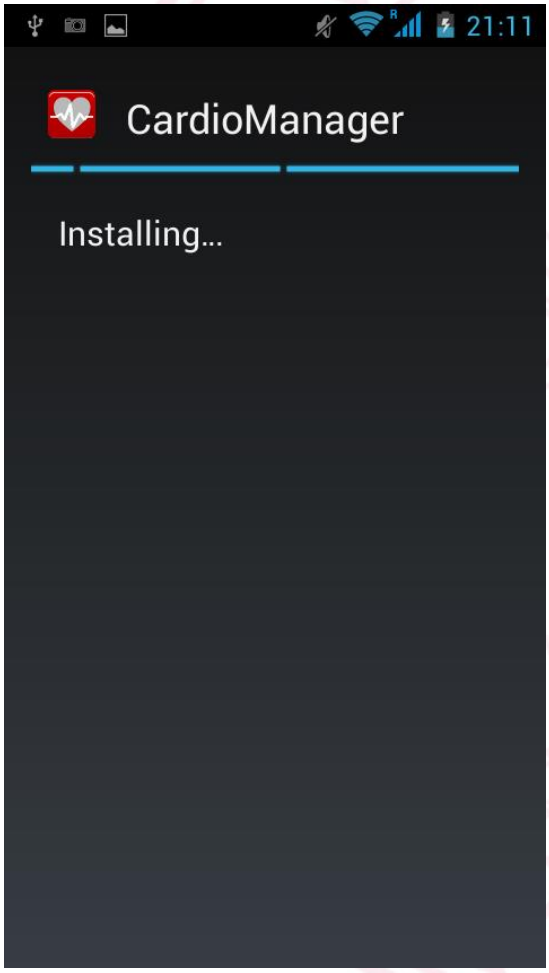




	Llamar directamente a números de teléfono	Necesario para poder realizar una llamada telefónica a nuestro contacto de emergencia
	Enviar mensajes SMS	Necesario para poder enviar un mensaje de texto a nuestro contacto de emergencia
	Ubicación aproximada (basada en red) Ubicación precisa (basada en red y GPS)	Permite localizar el dispositivo a la hora de introducir una actividad de tipo ataque o crisis
	Consultar tus contactos	Necesario para elegir el contacto de emergencia
	Leer eventos de calendario e información confidencial	Necesario para mostrar el calendario de actividades y medicamentos

	Modificar o eliminar el contenido de la tarjeta SD	Necesario para almacenar las distintas actividades y medicamentos, pues la base de datos puede ser almacenada en la tarjeta SD si el usuario así lo desea
	Leer la configuración de servicios de Google	Necesario para utilizar los servicios de mapas de Google
	Acceso completo a red Ver conexiones de red	Necesario para utilizar los servicios de mapas de Google y para poder realizar las llamadas y enviar mensajes de texto SMS
	Controlar la vibración	Permite que el dispositivo vibre cuando se muestre una notificación de la aplicación, es decir, cuando sea la hora de tomar cierto medicamento
	Probar acceso a almacenamiento protegido	Similar a "Modificar o eliminar el contenido de la tarjeta SD"



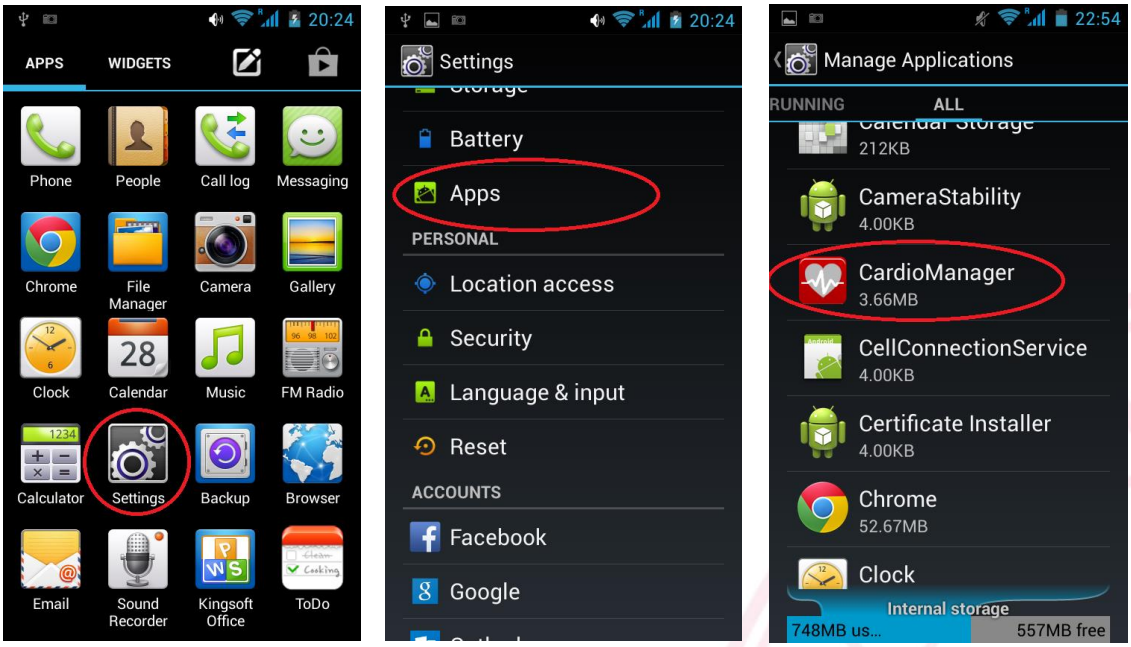


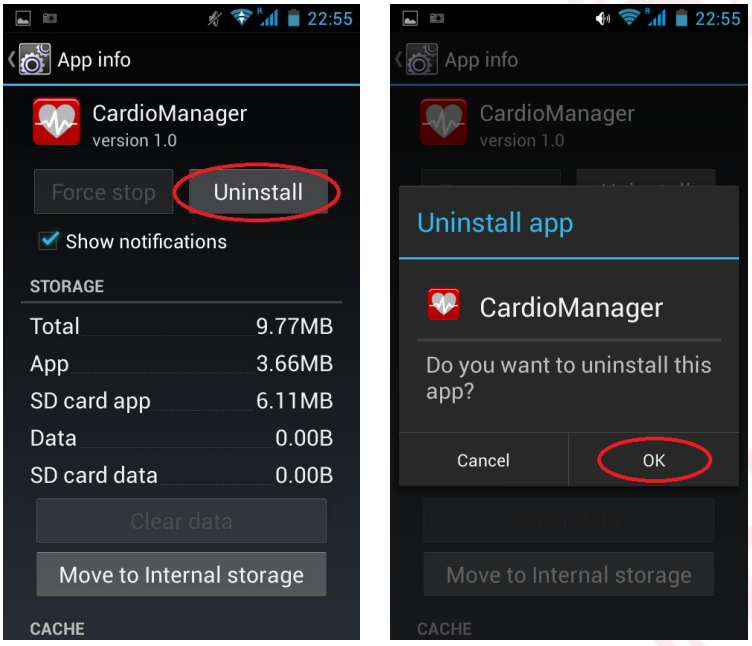
Una vez aceptados estos permisos pulsando en la opción **Instalar**, se procederá con la instalación de la aplicación. Se nos mostrará un diálogo en el que nos preguntan si deseamos que Google compruebe periódicamente el funcionamiento de la aplicación, para averiguar si se está comprometiendo la seguridad del dispositivo y del usuario. Se recomienda aceptar estas comprobaciones, pero siempre es decisión propia del usuario.

Una vez finalizado este proceso, podremos ya disfrutar del uso de la aplicación.

Desinstalación

La desinstalación de la aplicación es similar a la de cualquier aplicación en Android. En primer lugar nos dirigimos a **Ajustes** > **Aplicaciones**. En inglés Settings > Apps. Dentro de la lista de aplicaciones, seleccionamos **Cardiomanager**. En antiguas versiones de Android puede que las opciones sean **Ajustes** > **Administrar aplicaciones**.





Una vez seleccionada nuestra aplicación, pulsamos en **Desinstalar** (Uninstall).

Manual de uso

1. *Actividad principal*
2. *Introducción de Actividades*
3. *Registro de Actividades*
4. *Introducción de Nuevo Medicamento*
5. *Registro de Medicamentos*
6. *Notificaciones y aviso de toma*
7. *Información*
8. *Directrices*
9. *Información del paciente*
10. *Calculadora de Riesgo Cardiovascular*
11. *Ayuda*

Actividad principal

La actividad principal muestra una serie de botones que permiten acceder a las distintas secciones de la aplicación. Desde el menú, disponible en la barra de acciones para Android mayor que 4.0, y tras pulsar el botón de menú en versiones anteriores, se accede a otras secciones. Esto queda especificado en sucesivas páginas.

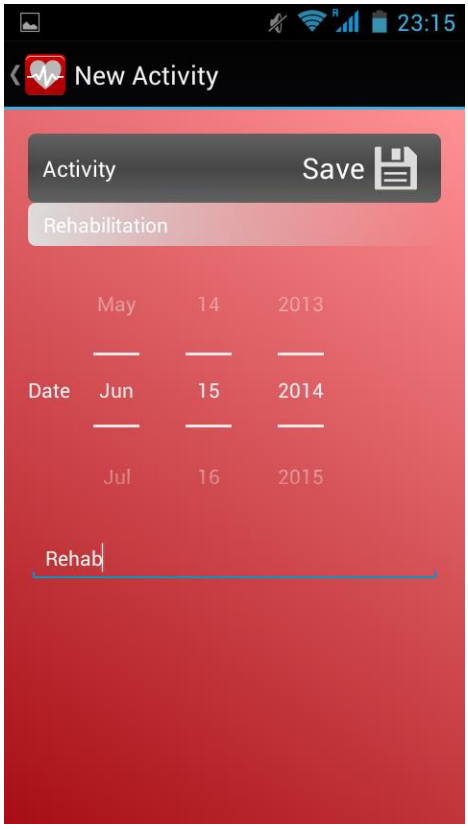
Además, deslizando la pantalla hacia la derecha, se podrá consultar el contacto de emergencia y realizar una llamada o enviarle un mensaje.

Introducción de Actividades

A través de este módulo, el paciente puede introducir diferentes actividades relacionadas con su enfermedad, como son una jornada de rehabilitación, una actividad física como correr o practicar cualquier deporte, introducir los datos de un análisis de sangre, presión sanguínea o un análisis de glucosa, y señalar ataques o crisis que haya sufrido, a la par que excesos que haya podido tener, como puedan ser grandes comidas o ingestas de alcohol.

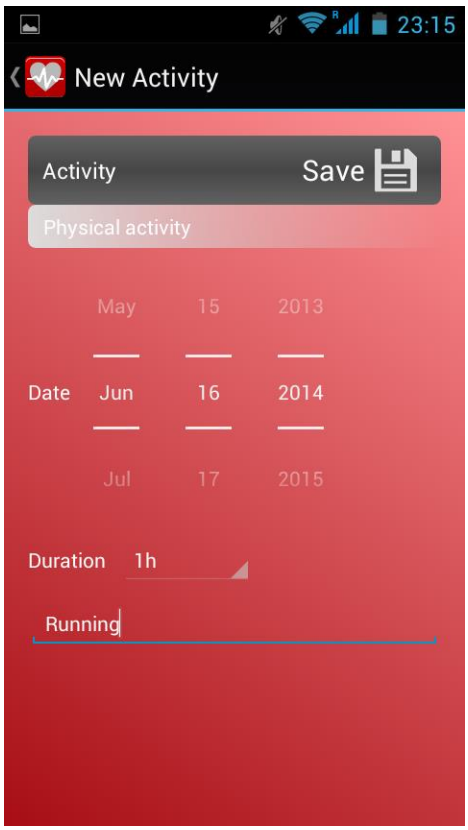
Todas estas actividades son almacenadas en la base de datos de la aplicación, pudiendo ser comprobadas en el módulo Registro de Actividades, que será comentado posteriormente.

A continuación se especifican cada uno de los tipos de actividades que permite la aplicación.



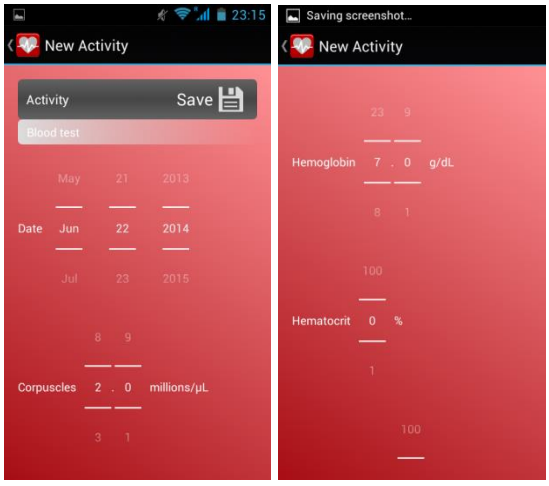
Rehabilitación

Permite seleccionar una **fecha** (por defecto la fecha actual) e introducir un **texto** libre, para indicar la información que el usuario considere oportuna, como los ejercicios realizados, el día, el lugar, los comentarios del médico, etc.



Actividad física

Permite seleccionar una **fecha** (por defecto la fecha actual), la **duración** e introducir un **texto** libre, para indicar la información que el usuario considere oportuna, como el ejercicio realizado, como se ha encontrado durante la realización del mismo, indicar una ruta seguida, etc.



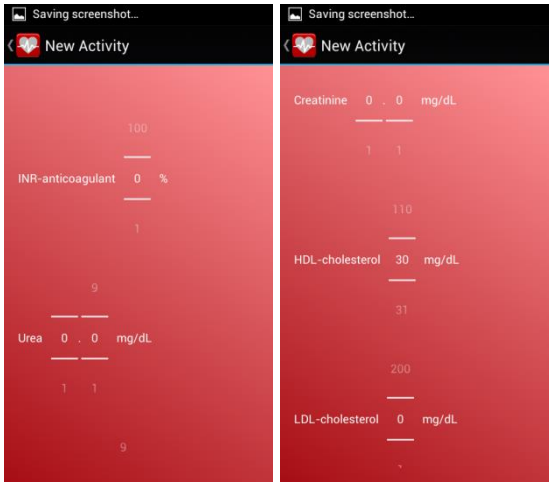
Análisis de sangre

Permite seleccionar una **fecha** (por defecto la fecha actual) y diferentes aspectos relacionados con un análisis de sangre.

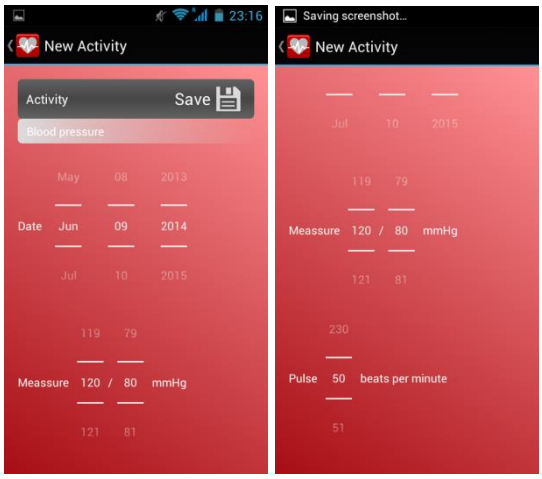
El usuario debe consultar los datos del análisis proporcionados por el doctor, e introducirlos manualmente en la aplicación.

El paciente podrá introducir los siguientes datos del análisis:

- Corpúsculos (millones/uL)
- Hemoglobina (g/dL)
- Hematocritos (%)
- Anticoagulante INR (%)
- Urea (mg/dL)
- Creatinina (mg/dL)
- Colesterol HDL (mg/dL)
- Colesterol LDL (mg/dL)
- Triglicéridos (mg/dL)



Presión sanguínea

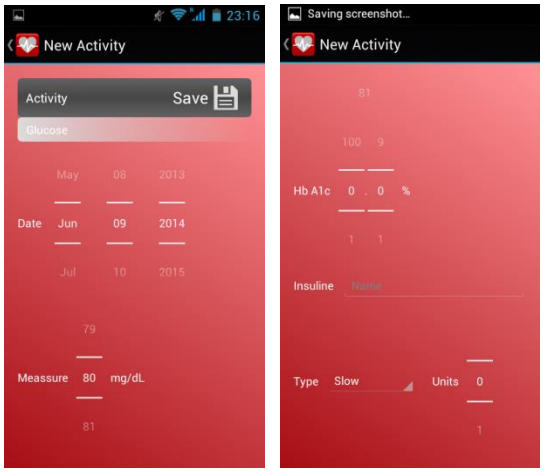


Permite seleccionar una **fecha** (por defecto la fecha actual) y diferentes aspectos relacionados con una medida de presión sanguínea.

El paciente podrá introducir los siguientes datos:

- Medida: hace referencia a la presión arterial *sistólica* y a la presión arterial *diastólica*.
- Pulso (latidos/minuto)

Medida de glucosa

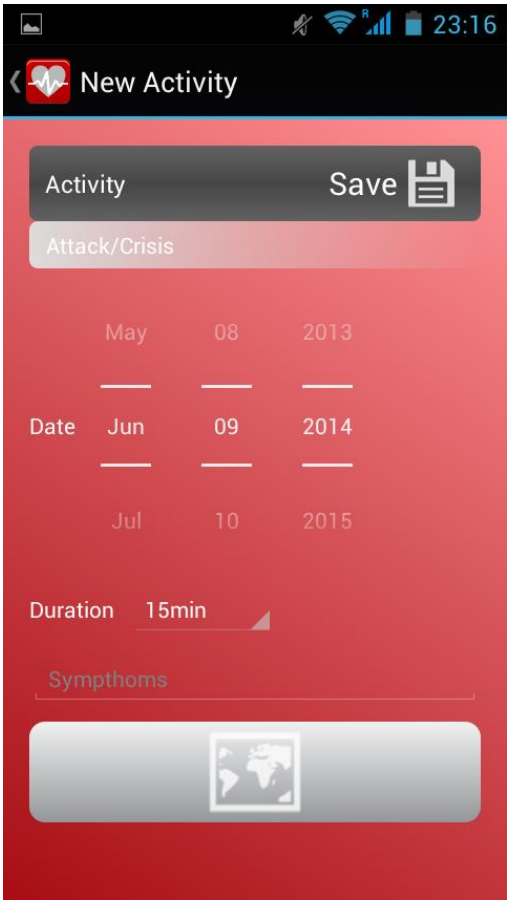


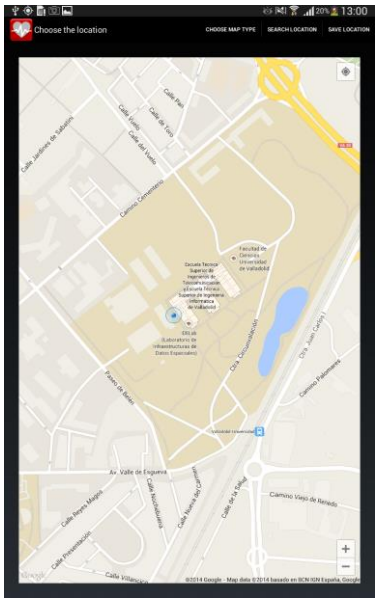
Permite seleccionar una **fecha** (por defecto la fecha actual) y diferentes aspectos relacionados con una medida de glucosa y una toma

El paciente podrá introducir los siguientes datos:

- Insulina inyectada (medida) en mg/dL.
- Hb A1c (%)
- Nombre del medicamento

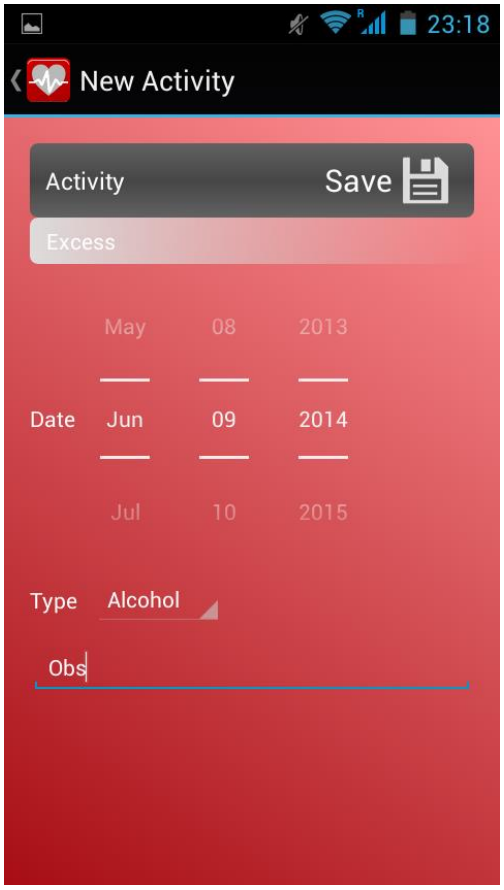
- Tipo: lenta, intermedia o rápida.
- Unidades: de 0 a 3.





Ataques / Crisis

Permite seleccionar una **fecha** (por defecto la fecha actual), la **duración** del ataque o crisis, un **texto** en el que describir los síntomas, las posibles causas y lo que el paciente ha hecho para solucionarlo o sentirse mejor. Además, pulsado el botón del mapamundi, se puede escoger la ubicación que se desee.



Excesos

Permite seleccionar una **fecha** (por defecto la fecha actual), el **tipo** de exceso que ha tenido lugar, como puedan ser una gran ingesta de alcohol, fumar o una comida copiosa y poco sana, y un **texto** en el que describir el exceso y su importancia.

Registro de Actividades

A través de este módulo, el paciente podrá:

- Visualizar en un calendario todas las actividades que haya realizado.
- Visualizar los resultados de los distintos análisis en diferentes gráficas.

Calendario de actividades

Debemos distinguir dos funcionamientos dentro del calendario, dependiendo del tamaño de la pantalla del dispositivo en el que se esté ejecutando la aplicación.

Para dispositivos con pantalla normal o pequeña (hasta 5”).

Se muestra un calendario en el que los días en los que el paciente ha realizado alguna actividad, son de color verde, y los días en los que no se ha producido ninguna actividad, en blanco. Por defecto, se mostrarán el mes y año correspondientes con la fecha actual.

Debajo del calendario, se muestra un desplegable en el que podemos escoger qué tipo de actividades queremos que se muestre en el calendario. Es decir, si queremos ver simplemente los días en los que hemos llevado a cabo una actividad física, basta con seleccionar este tipo de actividad en el desplegable y el calendario se actualizará automáticamente.

Para ver los detalles de cada día, basta con pulsar en la casilla correspondiente. Se mostrará en pantalla una lista con las distintas actividades y la información que el usuario ha introducido para cada una de ellas.

Cabe destacar el caso especial de los Ataques/Crisis, pues manteniendo el dedo pulsado en la actividad en concreto, se mostrará un mapa con la ubicación en la que se produjo el ataque o crisis.

Para dispositivos con pantalla grande o muy grande (7” o más).

En este caso, los días del calendario no se mostrarán en verde si se produjo una actividad, sino que se mostrará un pequeño icono identificativo de la actividad dentro de la casilla del mismo día. El resto del funcionamiento es similar al caso anteriormente expuesto.

Gráficas

Esta sección nos permite visualizar las actividades de análisis de sangre, presión sanguínea y glucosa en gráficas donde podemos comparar los datos introducidos mes a mes.

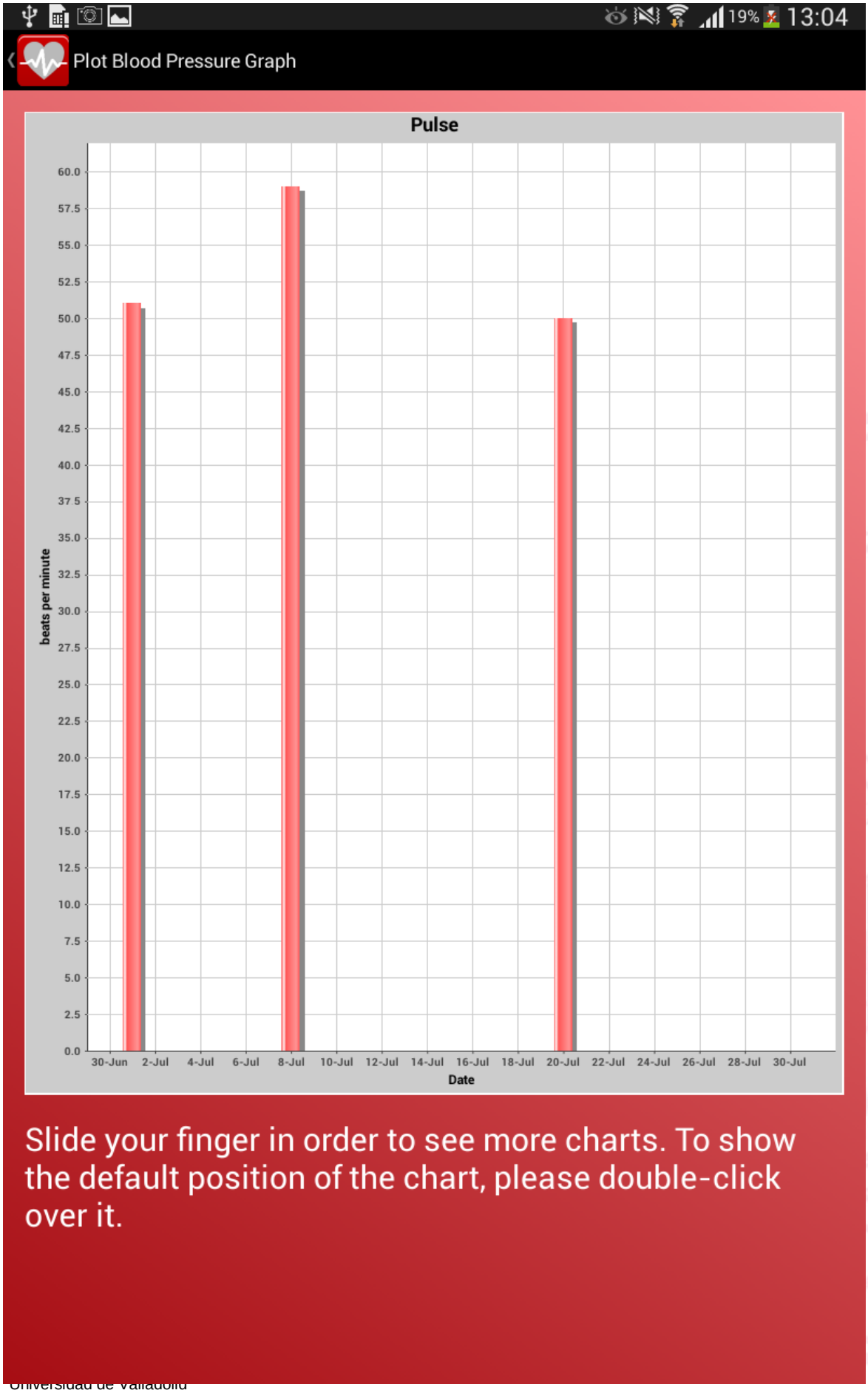
En primer lugar debemos escoger el tipo de actividad y el mes y año que queremos mostrar en las gráficas. Dependiendo de la opción escogida, se mostrarán distintas gráficas, una por cada elemento introducido en la actividad (ver sección anterior).

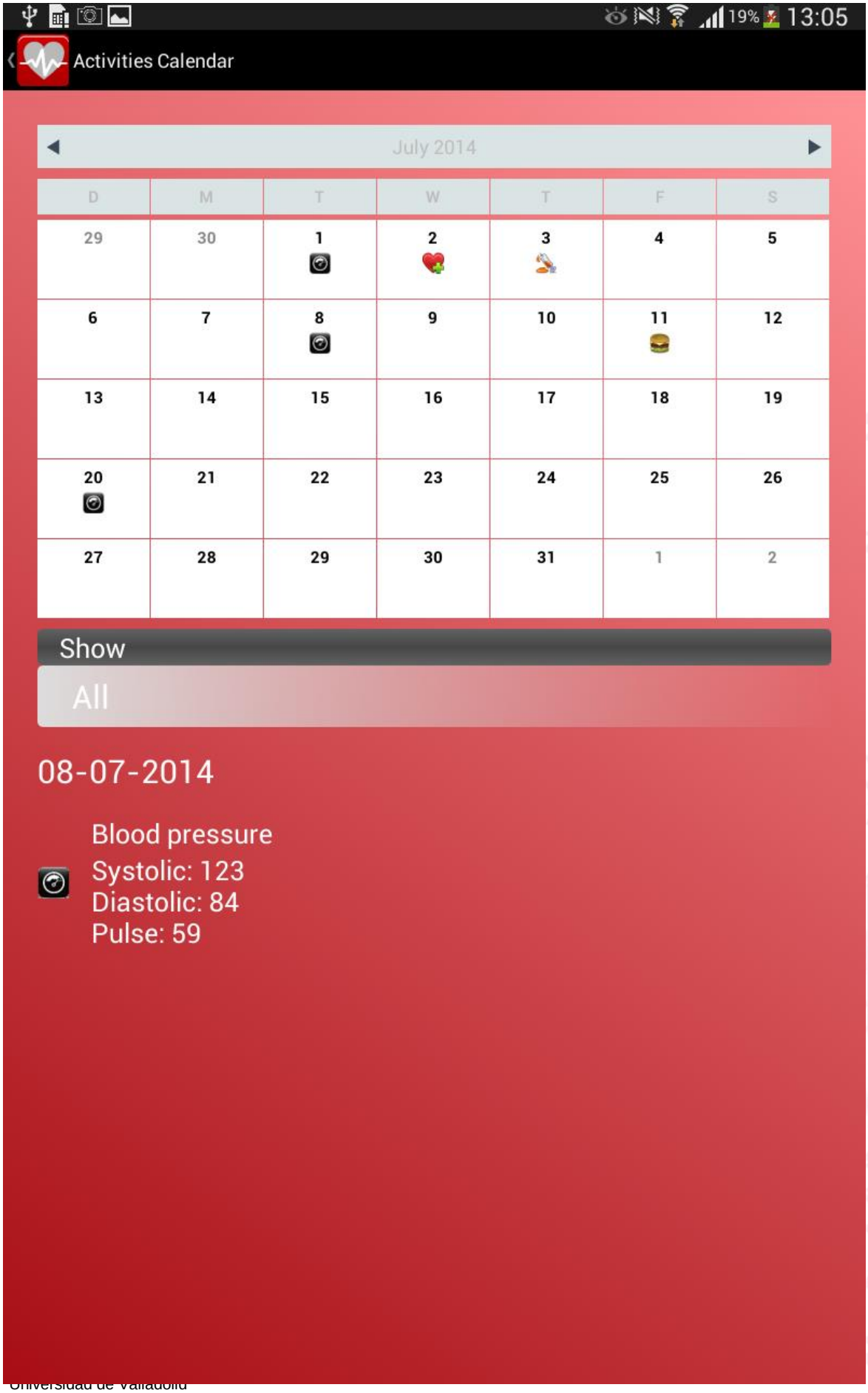
Dentro del mismo mes y actividad, deslizando el dedo horizontalmente podemos ir viendo las distintas gráficas, y dentro de las mismas gráficas podemos hacer zoom para comparar determinados días o ver la cantidad exacta mostrada en la gráfica.

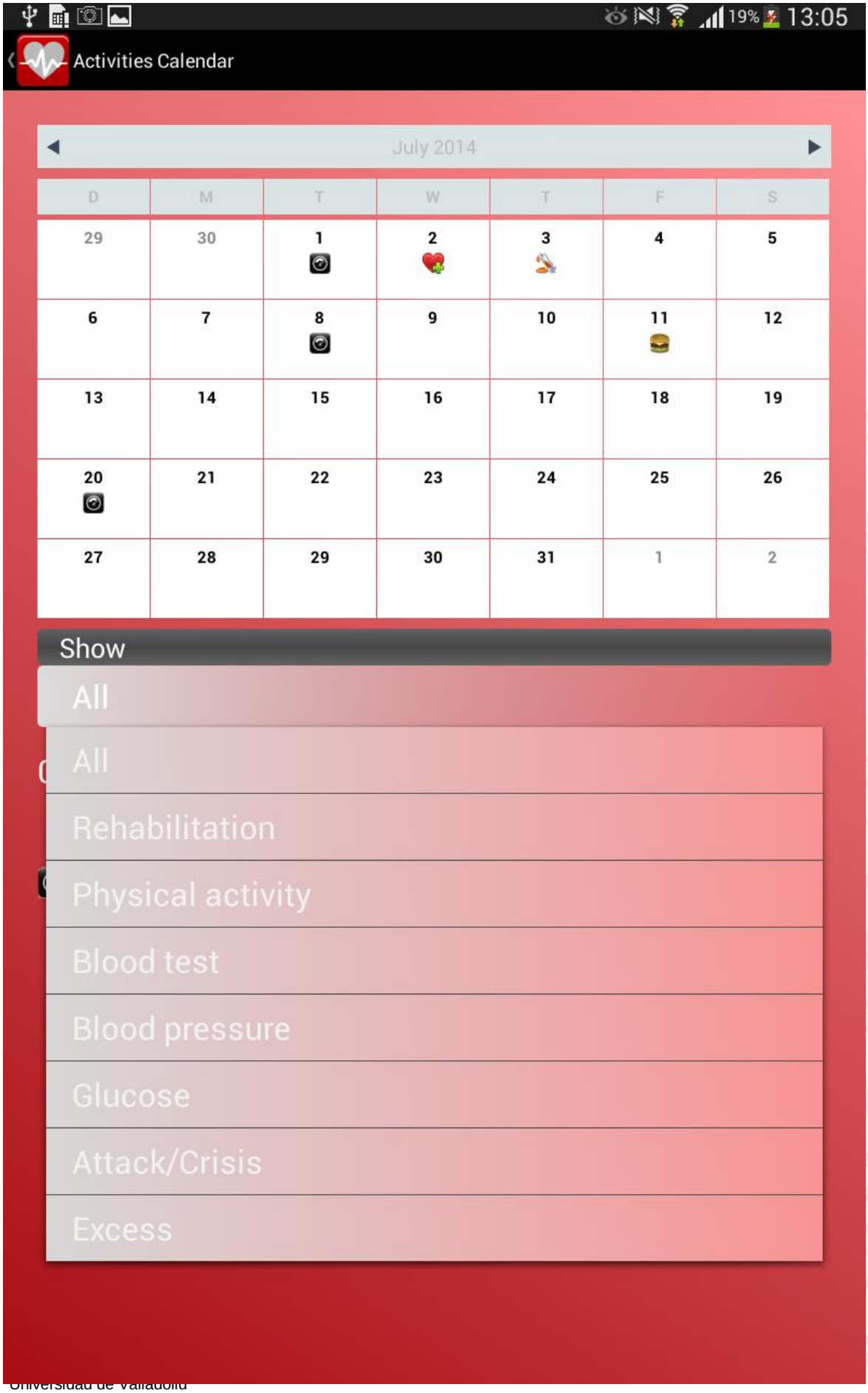
A continuación se muestran imágenes del uso de este módulo.


19% 13:04
Plot Blood Pressure Graph
Systolic
mmHg
130.0
127.5
125.0
122.5
120.0
117.5
115.0
112.5
110.0
107.5
105.0
102.5
100.0
97.5
95.0
92.5
90.0
87.5
85.0
82.5
80.0
77.5
75.0
72.5
70.0
67.5
65.0
62.5
60.0
57.5
55.0
52.5
50.0
47.5
45.0
42.5
40.0
37.5
35.0
32.5
30.0
27.5
25.0
22.5
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
5-Jul
12-Jul
19-Jul
26-Jul
Date
Slide your finger in order to see more charts. To show the default position of the chart, please double-click over it.


19% 13:04
Plot Blood Pressure Graph
26-Jul
Diastolic
87.5
85.0
82.5
80.0
77.5
75.0
72.5
70.0
67.5
65.0
62.5
60.0
57.5
55.0
52.5
50.0
47.5
45 0
42.5
40.0
37.5
35.0
32.5
30 0
27.5
25.0
22.5
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
mmHg
30-Jun
2-Jul
4-Jul
6-Jul
8-Jul
10-Jul
12-Jul
14-Jul
16-Jul
18-Jul
Date
e charts. To show
ase double-click
Slide your finger in order to see more
the default position of the chart, plea
over it.


19% 13:04
Plot Blood Pressure Graph
Pulse
beats per minute
60.0
57.5
55.0
52.5
50.0
47.5
45.0
42.5
40.0
37.5
35.0
32.5
30.0
27.5
25.0
22.5
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
30-Jun
2-Jul
4-Jul
6-Jul
8-Jul
10-Jul
12-Jul
14-Jul
16-Jul
18-Jul
20-Jul
22-Jul
24-Jul
26-Jul
28-Jul
30-Jul
Date
Slide your finger in order to see more charts. To show the default position of the chart, please double-click over it.


19% 13:05
Activities Calendar
July 2014
D M T W T F S
29 30 1 2 3 4 5
6 7 8 9 10 11 12
13 14 15 16 17 18 19
20 21 22 23 24 25 26
27 28 29 30 31 1 2
Show
All
08-07-2014
Blood pressure
Systolic: 123
Diastolic: 84
Pulse: 59


19% 13:05
Activities Calendar
July 2014
D M T W T F S
29 30 1 2 3 4 5
6 7 8 9 10 11 12
13 14 15 16 17 18 19
20 21 22 23 24 25 26
27 28 29 30 31 1 2
Show
All
All
Rehabilitation
Physical activity
Blood test
Blood pressure
Glucose
Attack/Crisis
Excess

Introducción de Nuevo Medicamento

Este módulo permite al usuario almacenar en la aplicación distintos medicamentos relacionados con su enfermedad e indicar a la aplicación que le avise cuando sea la hora de tomar las distintas dosis.

En primer lugar, se muestra una actividad en la que el usuario puede introducir el **nombre del medicamento**, **notas** que considere oportunas sobre cómo debe ingerir el medicamento o aspectos a tener en cuenta del mismo y seleccionar el **tipo de tratamiento** que desea seguir.

Tipos de tratamiento

- ✓ **Todos los días:** el medicamento debe tomarse todos los días de la semana y a las mismas horas.
- ✓ **Selecciona periodo:** cuando el medicamento debe ser tomado cada ciertas horas y cierto número de dosis, por ejemplo, cada 6 horas y un total de 15 dosis o hasta que termine el paquete.
- ✓ **Una dosis:** en el caso de que sea necesario tomar una sola dosis de algún medicamento especial.
- ✓ **Selecciona días:** permite seleccionar los días de la semana en los que se desea tomar el medicamento, por ejemplo, los sábados y los domingos.

Justo debajo del elemento que permite seleccionar entre los distintos tipos de tratamientos, se encuentran las respectivas descripciones para ayudar al usuario a entender mejor el procedimiento. Una vez introducidos estos datos, para seguir, se debe pulsar la flecha situada arriba a la derecha.

Dependiendo de lo seleccionado en el tipo de tratamiento, en la segunda actividad se podrá:

- ✓ Caso **todos los días.**
  - Seleccionar la **duración**:
    - Indefinido: en este caso, nuestro medicamento deberá ser tomado todos los días, empezando por el día en el que se introduce en la aplicación, y se desconoce cuándo se debe dejar de tomar, o si se dejará de tomar algún día. Para indicar que finaliza este medicamento, es necesario borrarlo de la aplicación.
    - Desde/hasta: en el caso de que debamos tomar el medicamento todos los días, pero conozcamos una fecha de inicio y una fecha de fin de tratamiento. Se realizará la comprobación de que la fecha de fin del tratamiento es posterior a la de inicio del tratamiento.
  - Seleccionar el **número de dosis**: se permite programar entre una y cinco dosis para el mismo día. Se mostrarán tantos botones y textos con la hora programada como dosis elegidas haya. El botón con el reloj permite mostrar un diálogo en el que el usuario elegirá la hora de la dosis.

- **Mostrar una notificación**: si se marca la opción de mostrar una notificación, la aplicación añadirá a la bandeja de notificaciones del dispositivo una más, indicando los medicamentos pendientes por tomar, acompañado de un breve sonido de notificación escogido por el usuario a través de los ajustes del teléfono y un breve patrón de vibración.

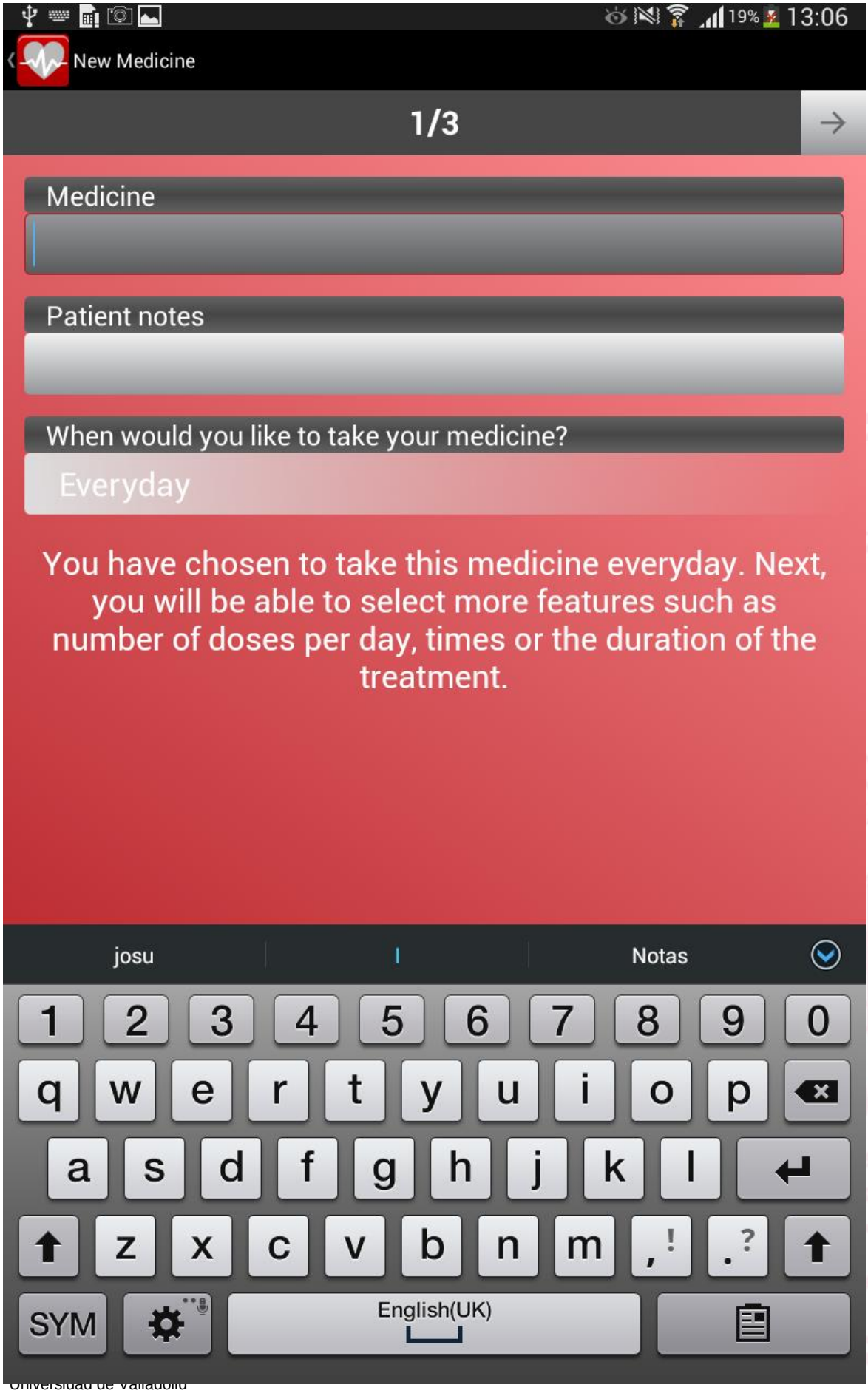
✓ Caso **selecciona periodo.**
- Seleccionar la **fecha de comienzo del tratamiento**.
- Seleccionar la **hora de la primera dosis.**
- Seleccionar el **periodo en horas**, o intervalo de tiempo que debe pasar entre una dosis y otra.
- Seleccionar el **número de dosis** total para este tratamiento. Una vez pasadas estas tomas, se dejarán de mostrar las notificaciones, y se entenderá que el tratamiento ha terminado.
- **Mostrar una notificación**: similar al caso 'todos los días'.

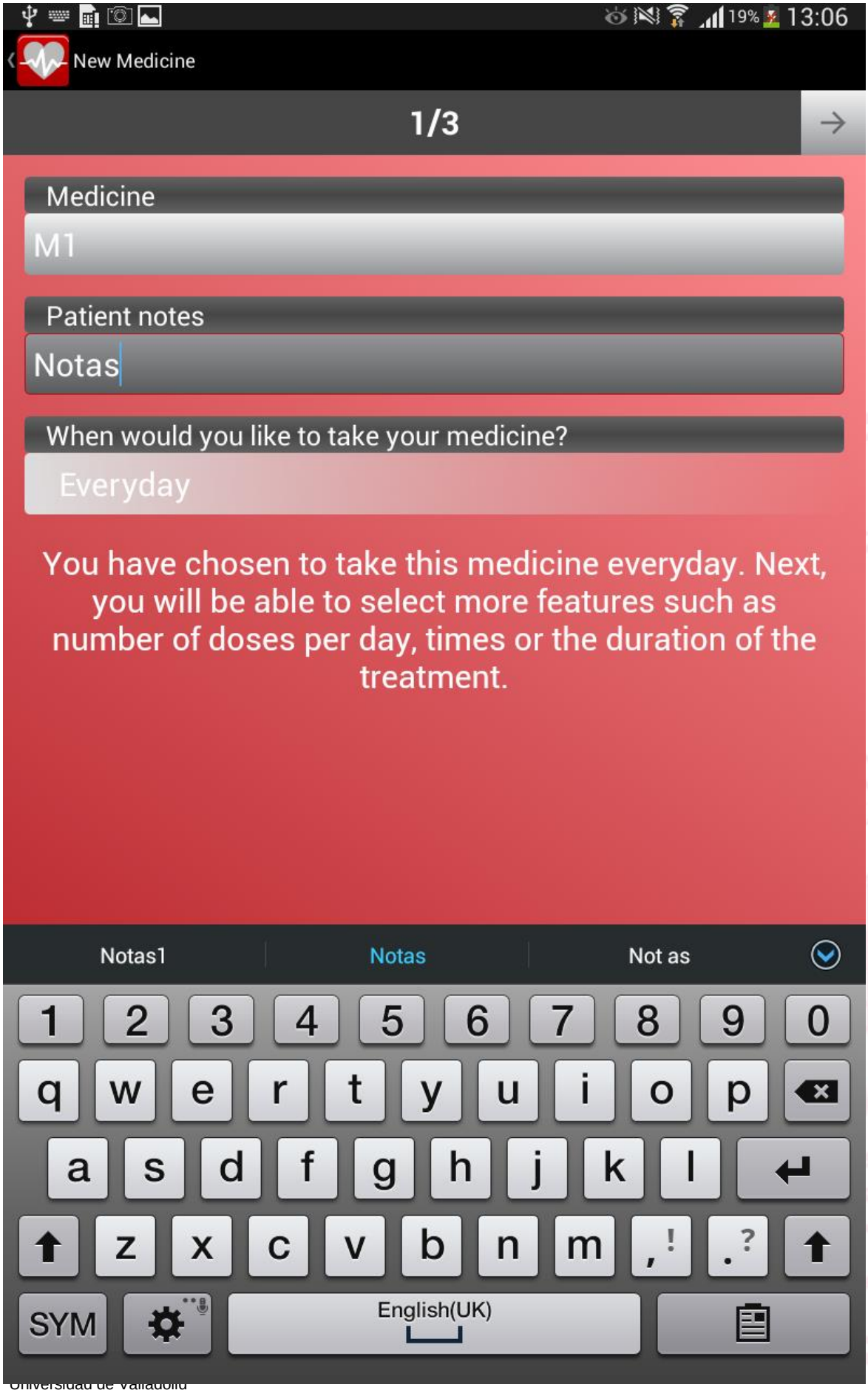
✓ Caso **una dosis**
- Seleccionar la **fecha de la dosis**
- Seleccionar la **hora de la dosis.**
- **Mostrar una notificación**: similar al caso 'todos los días'.

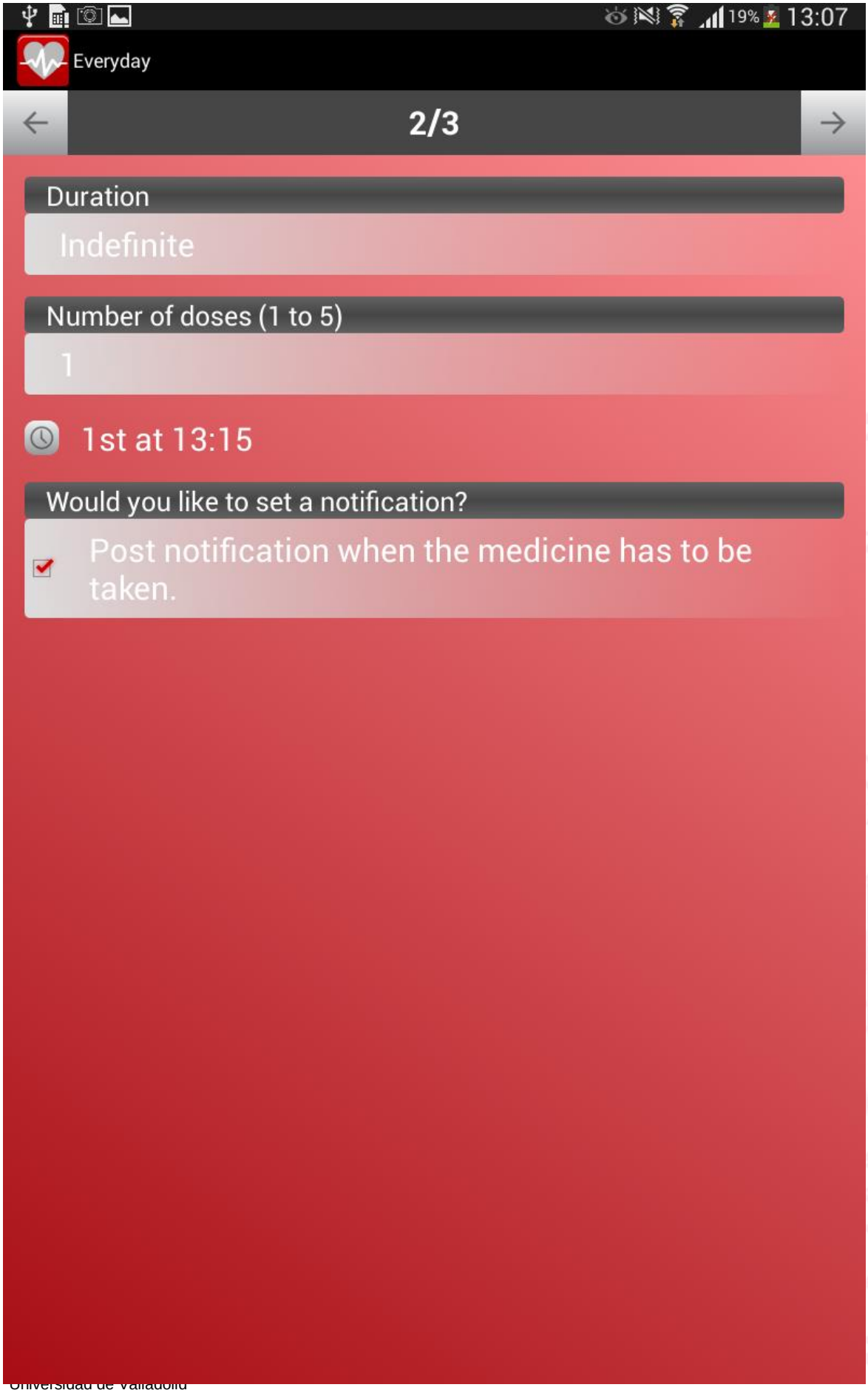
✓ Caso **selección los días.**
- Seleccionar los **días** en los que se desea tomar la medicación. Simplemente es necesario marcar los días deseados (en oscuro).
- Seleccionar la **duración**: similar al caso 'todos los días'.
- Seleccionar el **número de dosis**: similar al caso 'todos los días'.
- **Mostrar una notificación**: similar al caso 'todos los días'.

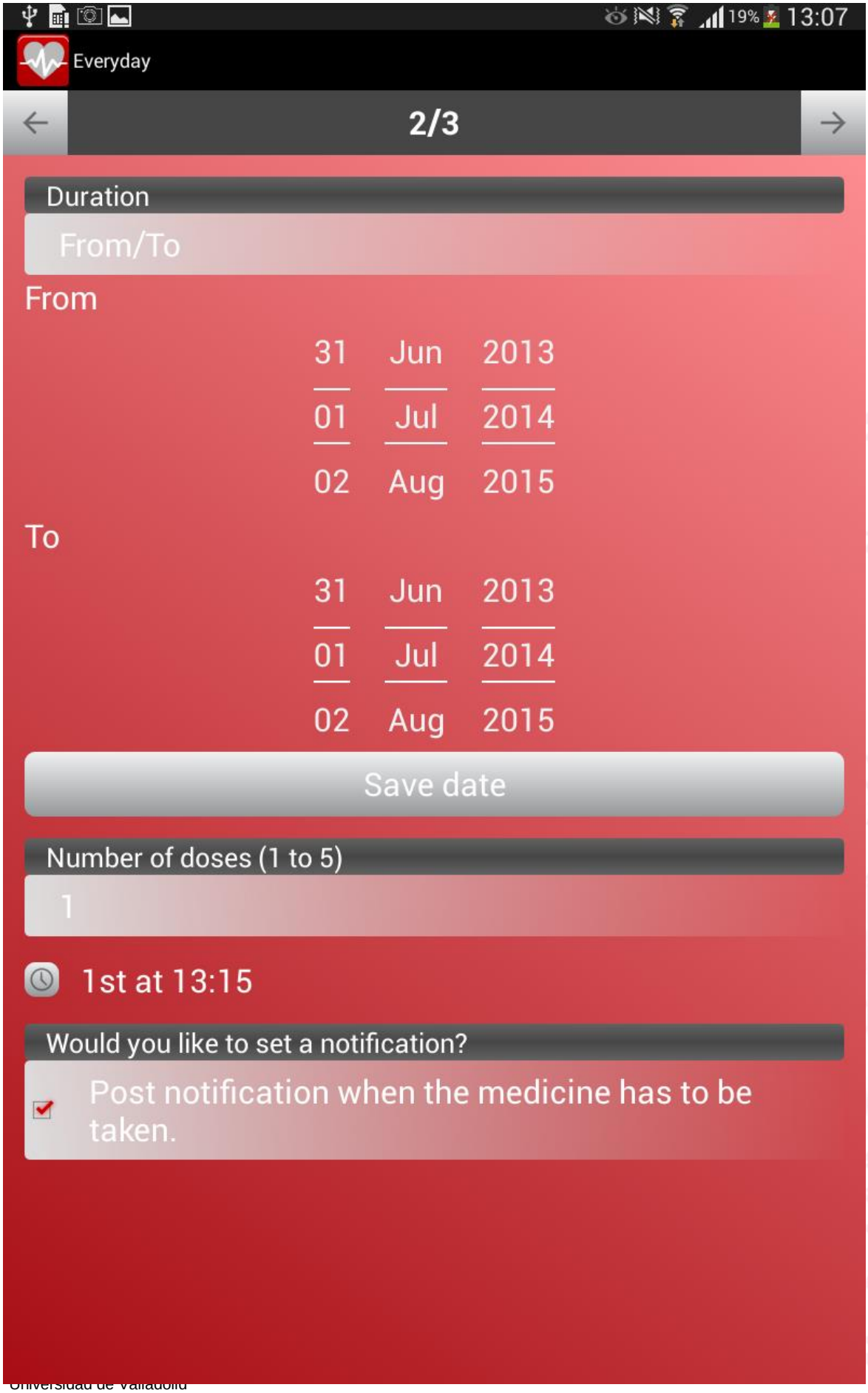
Una vez personalizado el tratamiento, para seguir, se debe pulsar la flecha situada arriba a la derecha.

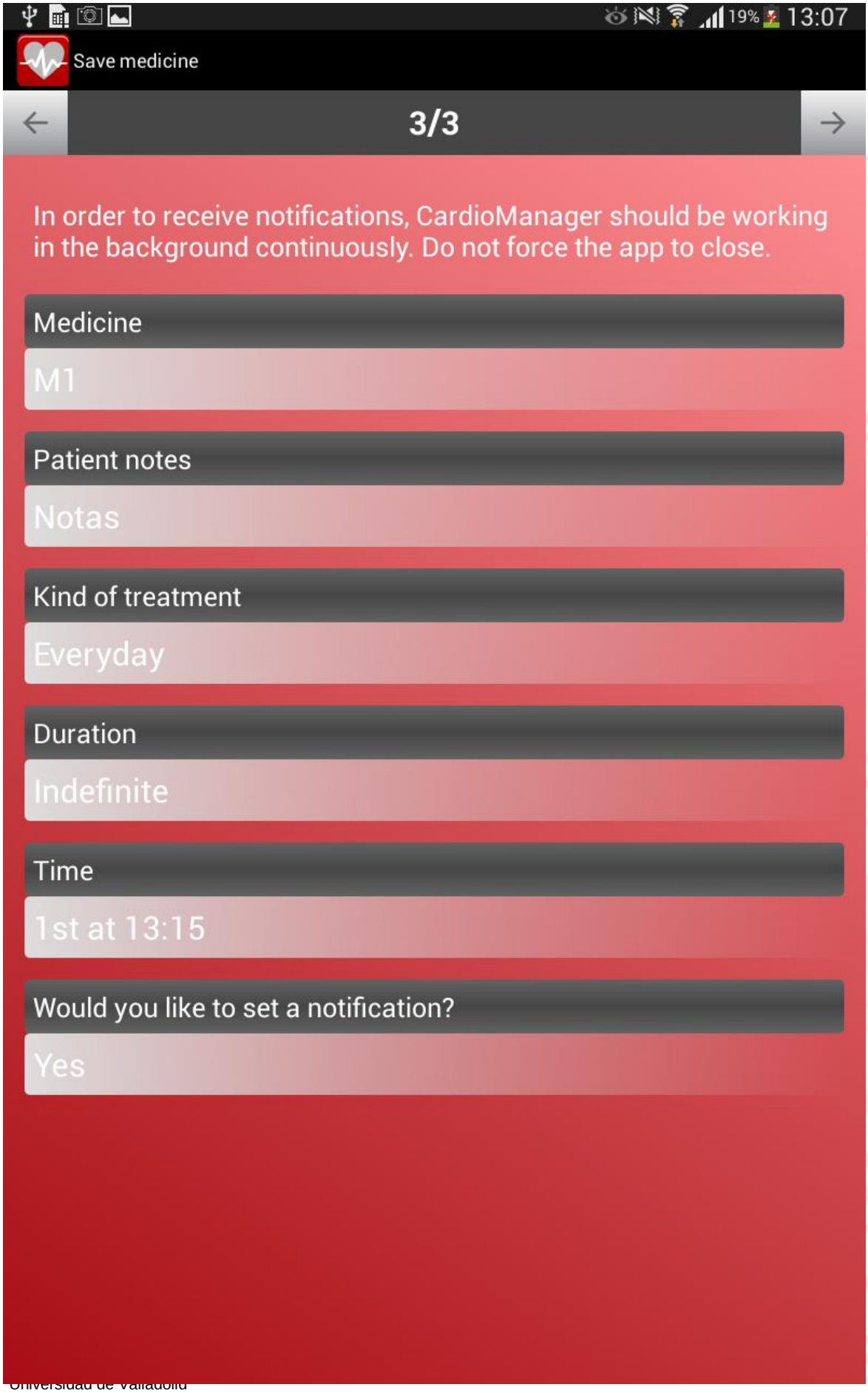
Por último, en la última actividad antes de programar las alarmas, se mostrarán de nuevo todos estos datos, con el objetivo de que el usuario revise bien lo introducido, y vuelva atrás si es necesario modificar algo. Al pulsar la flecha situada arriba a la derecha, se programará la primera dosis y el tratamiento comenzará.


19% 13:06
New Medicine
1/3
Medicine
Patient notes
When would you like to take your medicine?
Everyday
You have chosen to take this medicine everyday. Next, you will be able to select more features such as number of doses per day, times or the duration of the treatment.
josu
Notas
SYM
English(UK)


19% 13:06
New Medicine
1/3
Medicine
M1
Patient notes
Notas
When would you like to take your medicine?
Everyday
You have chosen to take this medicine everyday. Next, you will be able to select more features such as number of doses per day, times or the duration of the treatment.
Notas1
Notas
Not as
1 2 3 4 5 6 7 8 9 0
q w e r t y u i o p
a s d f g h j k l
z x c v b n m , ! . ?
SYM
English(UK)


13:07
Everyday
2/3
Duration
Indefinite
Number of doses (1 to 5)
1
1st at 13:15
Would you like to set a notification?
Post notification when the medicine has to be taken.


19% 13:07
Everyday
2/3
Duration
From/To
From
31 Jun 2013
01 Jul 2014
02 Aug 2015
To
31 Jun 2013
01 Jul 2014
02 Aug 2015
Save date
Number of doses (1 to 5)
1
1st at 13:15
Would you like to set a notification?
Post notification when the medicine has to be taken.


19% 13:07
Save medicine
3/3
In order to receive notifications, CardioManager should be working in the background continuously. Do not force the app to close.
Medicine
M1
Patient notes
Notas
Kind of treatment
Everyday
Duration
Indefinite
Time
1st at 13:15
Would you like to set a notification?
Yes

Registro de Medicamentos

A través de este módulo, el paciente podrá:

- Consultar la información sobre los medicamentos almacenados en la aplicación.
- Visualizar las ‘tomas’ o ‘no tomas’ de los medicamentos en un calendario.
- Marcar como ‘tomado’ o ‘no tomado’ las distintas dosis pendientes de clasificar.
- Consultar una lista con las dosis no tomadas de todos los medicamentos.

Lista de medicamentos

En esta sección se muestra, a la izquierda, una lista con los medicamentos guardados en la aplicación. En la parte derecha de la pantalla, al pulsar sobre un medicamento, se muestra la descripción del medicamento, es decir, nombre, notas, tipo de tratamiento, horas de tomas, fechas, si se mostrará o no una notificación, etc.

En esta sección se permite también borrar de la aplicación cualquiera de los medicamentos. Para ello, basta con mantener el dedo pulsado en uno de los medicamentos de la lista de la parte izquierda. Al eliminar medicamento debemos tener en cuenta que se borrarán todos los datos sobre el mismo, tanto la información del medicamento como las tomas pendientes, tomadas y no tomadas.

Calendario de medicamentos

Este calendario es similar al Calendario de Actividades. En verde, muestra los días en los que las todas las dosis programadas para ese día han sido tomadas, mientras que en rojo se mostrarán los días en los que haya fallado alguna dosis. En blanco, los días en los que no hubo ninguna dosis programada.

Al igual que el Calendario de Actividades, se permite filtrar cuanta información queremos ver. En este caso, podemos elegir dos parámetros, los medicamentos a mostrar y las dosis tomadas o no tomadas. Esto nos ayuda a comprobar, por ejemplo, las dosis no tomadas en cierto mes de cierto medicamento, y sacar las respectivas conclusiones.

Además, al pulsar sobre cualquiera de los días, se mostrará una lista con el desglose de dosis para dicho día: nombre del medicamento, hora de la dosis y si ha sido tomado o no.

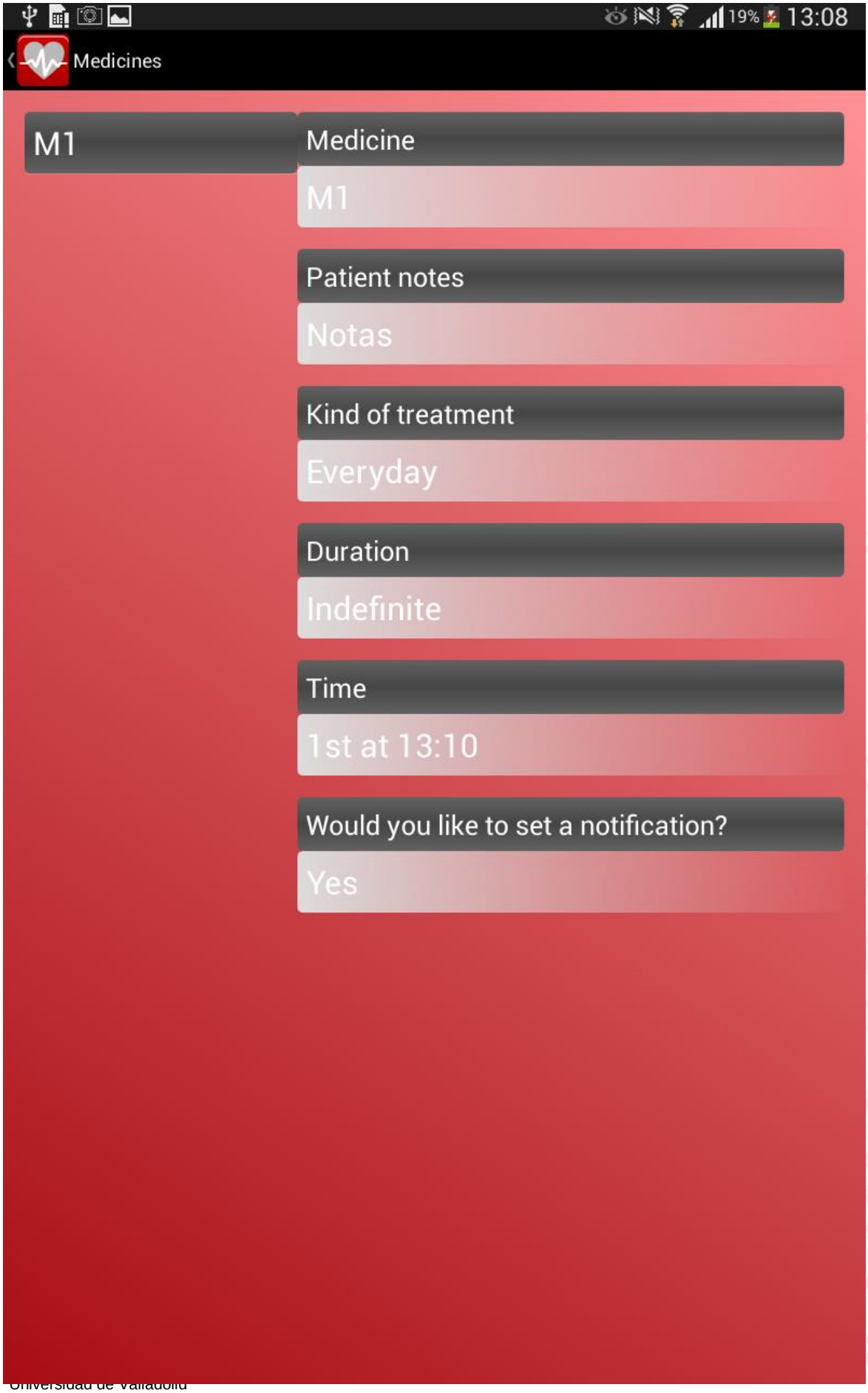
Medicamentos pendientes

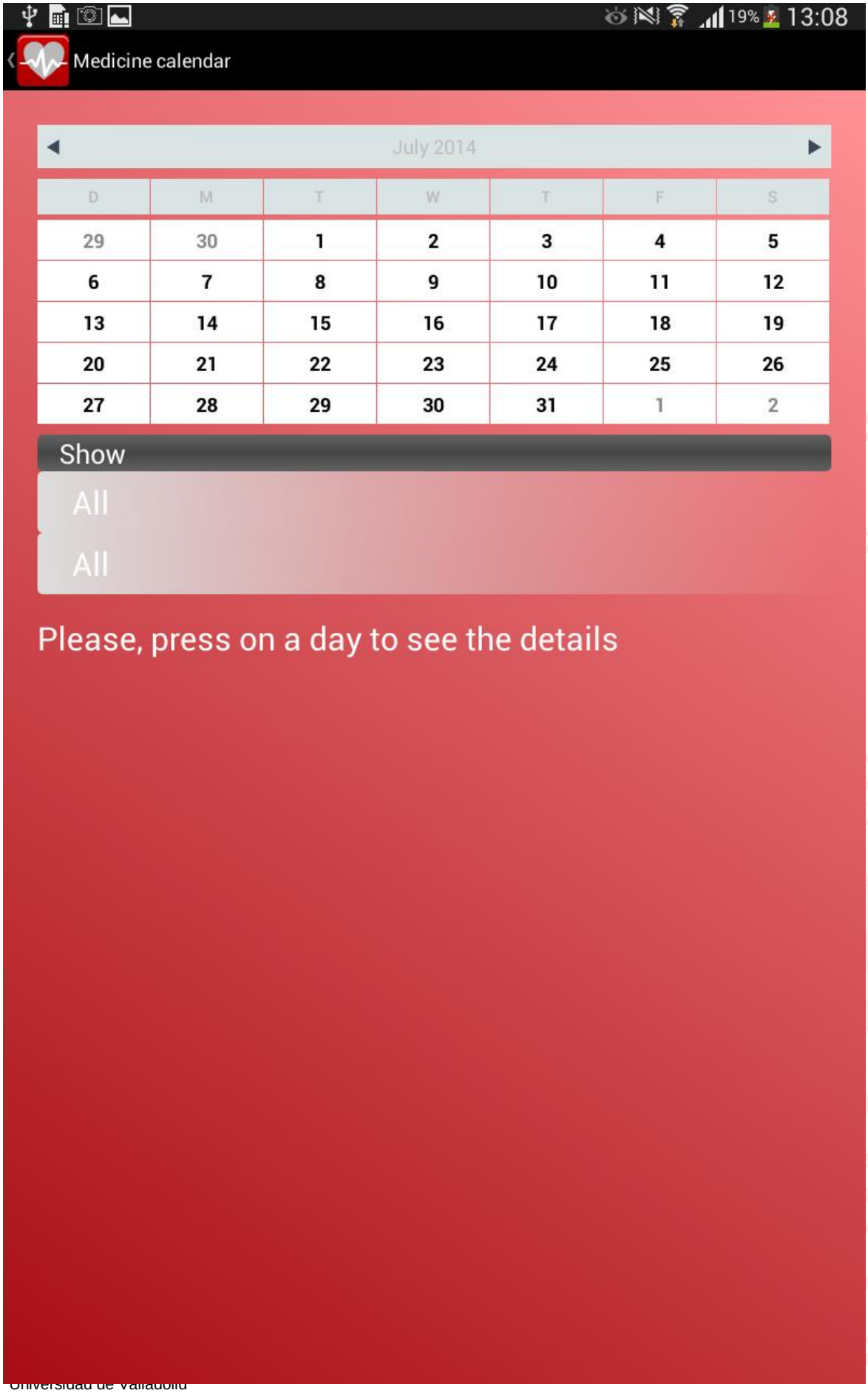
Se muestra una lista con las dosis pendientes de marcar como ‘tomadas’ o ‘no tomadas’. Para proceder a la clasificación, se deben seleccionar las dosis de la lista que deseamos añadir a un grupo (se mostrará un tick rojo a la izquierda) y pulsar el botón de ‘Tomado’ o ‘No tomado’ para guardar las dosis en la categoría correspondiente.

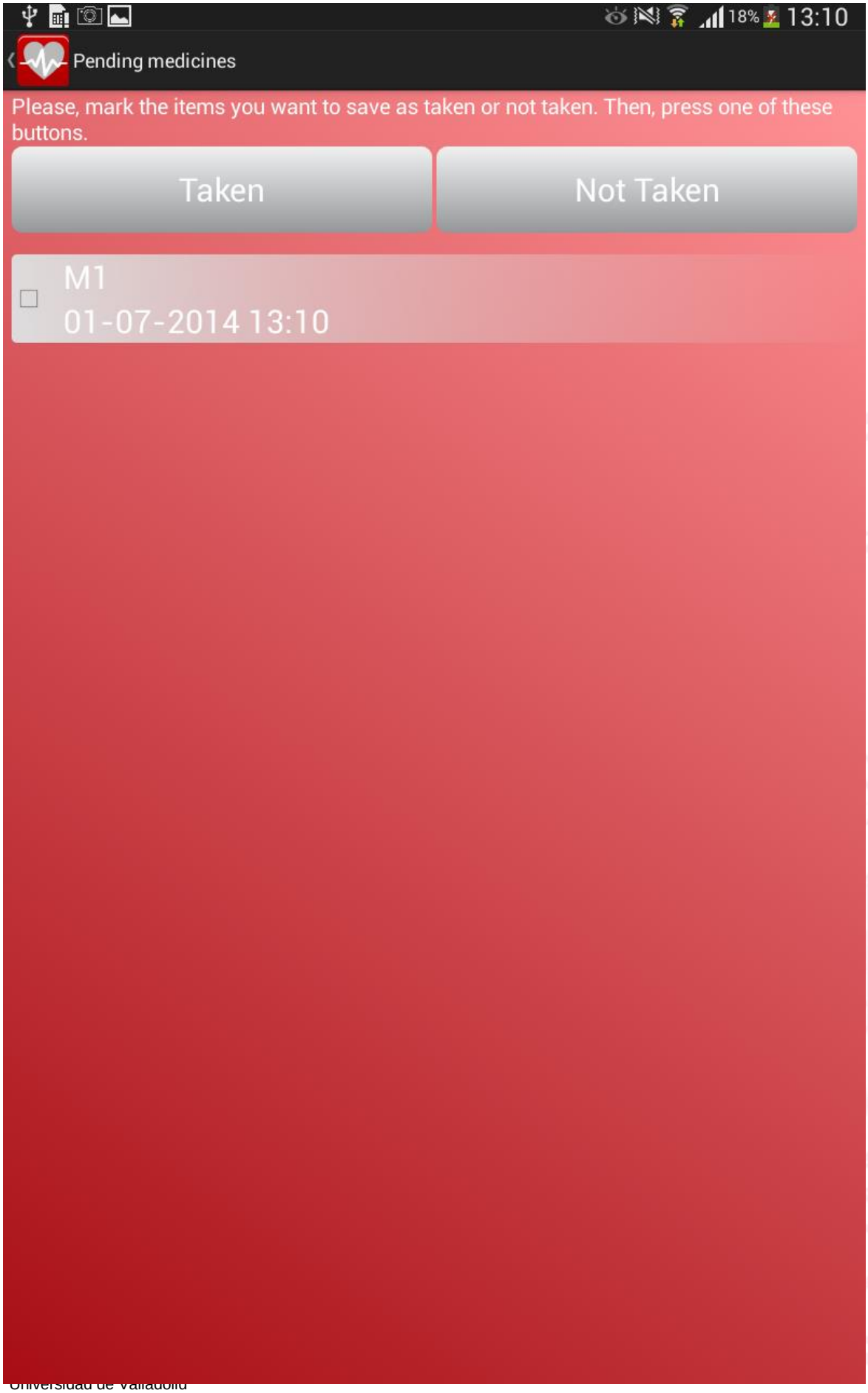
Medicamentos no tomados

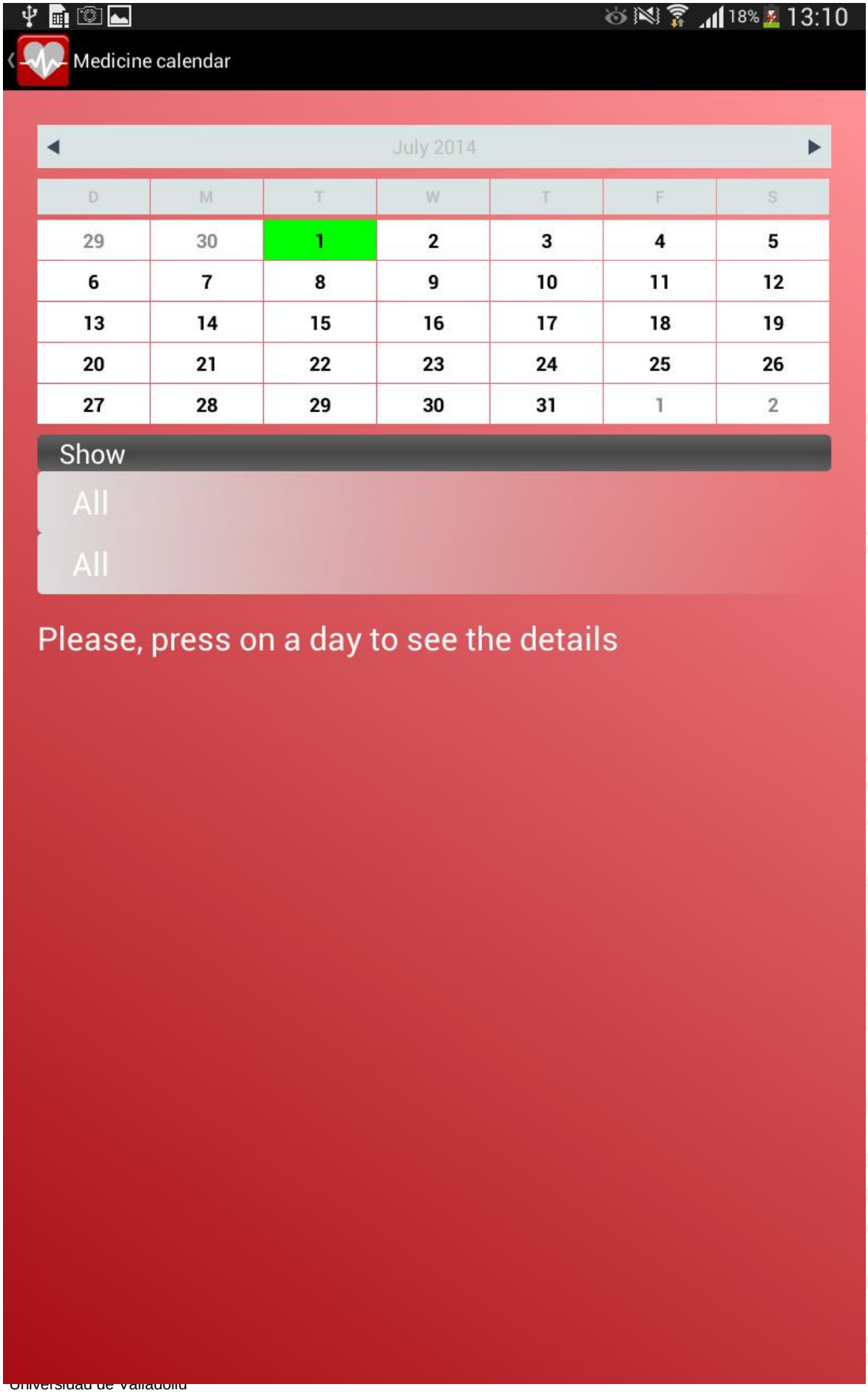
Se muestra una lista con las dosis de los medicamentos no tomadas, ordenados por fecha (los últimos primeros).

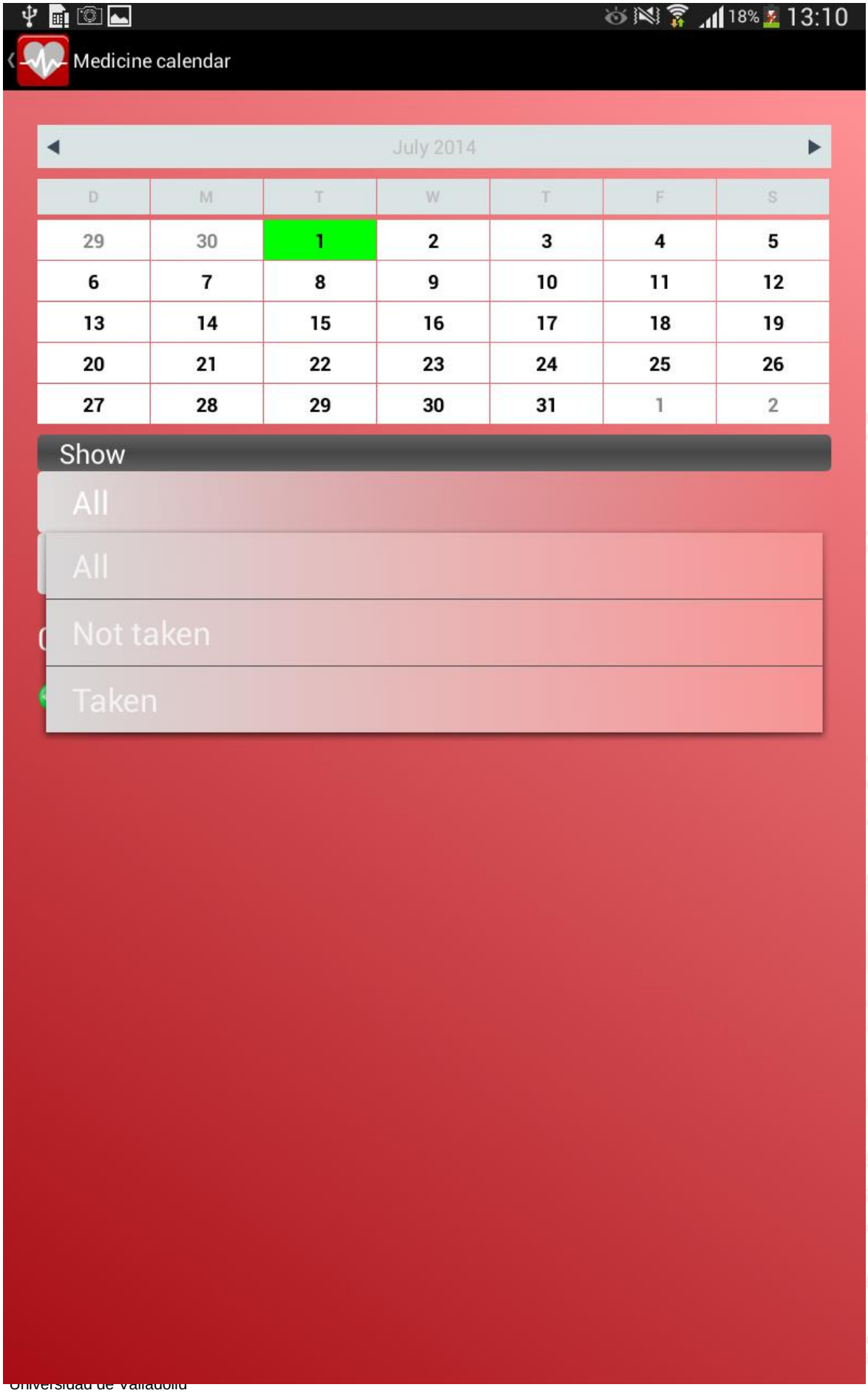
Por defecto se mostrarán todos, pero se podrá escoger el medicamento cuyas dosis no tomadas se desea mostrar a través del seleccionable.

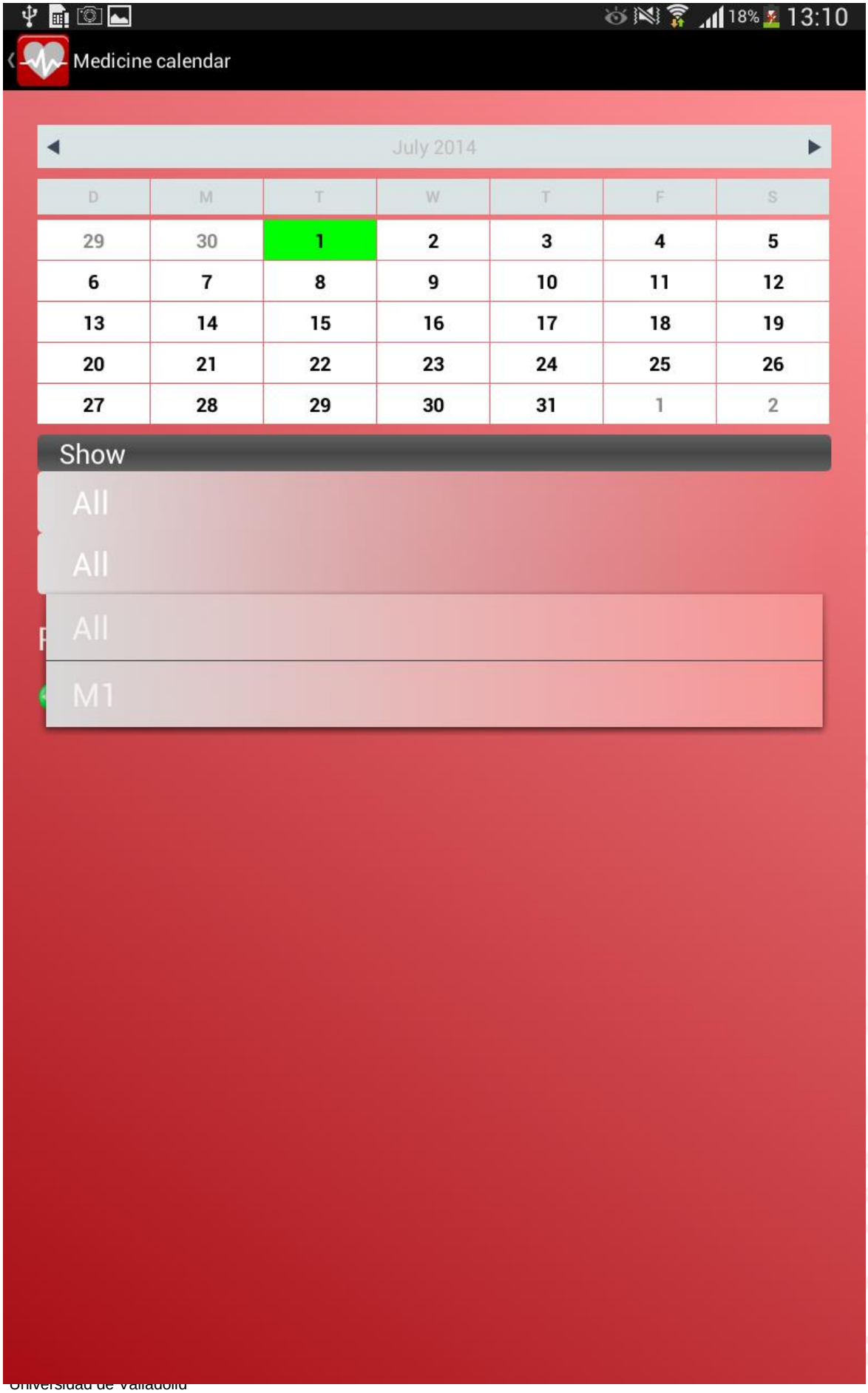

19% 13:08
Medicines
M1
Medicine
M1
Patient notes
Notas
Kind of treatment
Everyday
Duration
Indefinite
Time
1st at 13:10
Would you like to set a notification?
Yes


19% 13:08
Medicine calendar
July 2014
D M T W T F S
29 30 1 2 3 4 5
6 7 8 9 10 11 12
13 14 15 16 17 18 19
20 21 22 23 24 25 26
27 28 29 30 31 1 2
Show
All
All
Please, press on a day to see the details


18% 13:10
Pending medicines
Please, mark the items you want to save as taken or not taken. Then, press one of these buttons.
Taken
Not Taken
M1
01-07-2014 13:10


13:10
Medicine calendar
July 2014
D M T W T F S
29 30 1 2 3 4 5
6 7 8 9 10 11 12
13 14 15 16 17 18 19
20 21 22 23 24 25 26
27 28 29 30 31 1 2
Show
All
All
Please, press on a day to see the details


18% 13:10
Medicine calendar
July 2014
D M T W T F S
29 30 1 2 3 4 5
6 7 8 9 10 11 12
13 14 15 16 17 18 19
20 21 22 23 24 25 26
27 28 29 30 31 1 2
Show
All
All
Not taken
Taken


13:10
Medicine calendar
July 2014
D M T W T F S
29 30 1 2 3 4 5
6 7 8 9 10 11 12
13 14 15 16 17 18 19
20 21 22 23 24 25 26
27 28 29 30 31 1 2
Show
All
All
All
M1

Notificaciones y aviso de toma

Cuando llegue el momento en que se deba tomar un medicamento, si el usuario así lo ha querido, se añadirá una notificación a la bandeja de notificaciones del dispositivo, indicando los medicamentos pendientes por señalar como tomados o no tomados, a la vez que suena la melodía de notificación y el dispositivo vibra con el patrón programado.

Al pulsar en la notificación, se abrirá el **Registro de Medicamentos**, para permitir que el usuario pueda marcar los medicamentos pendientes.

Se deben destacar algunas características de este sistema, pues no es tan simple como parece:

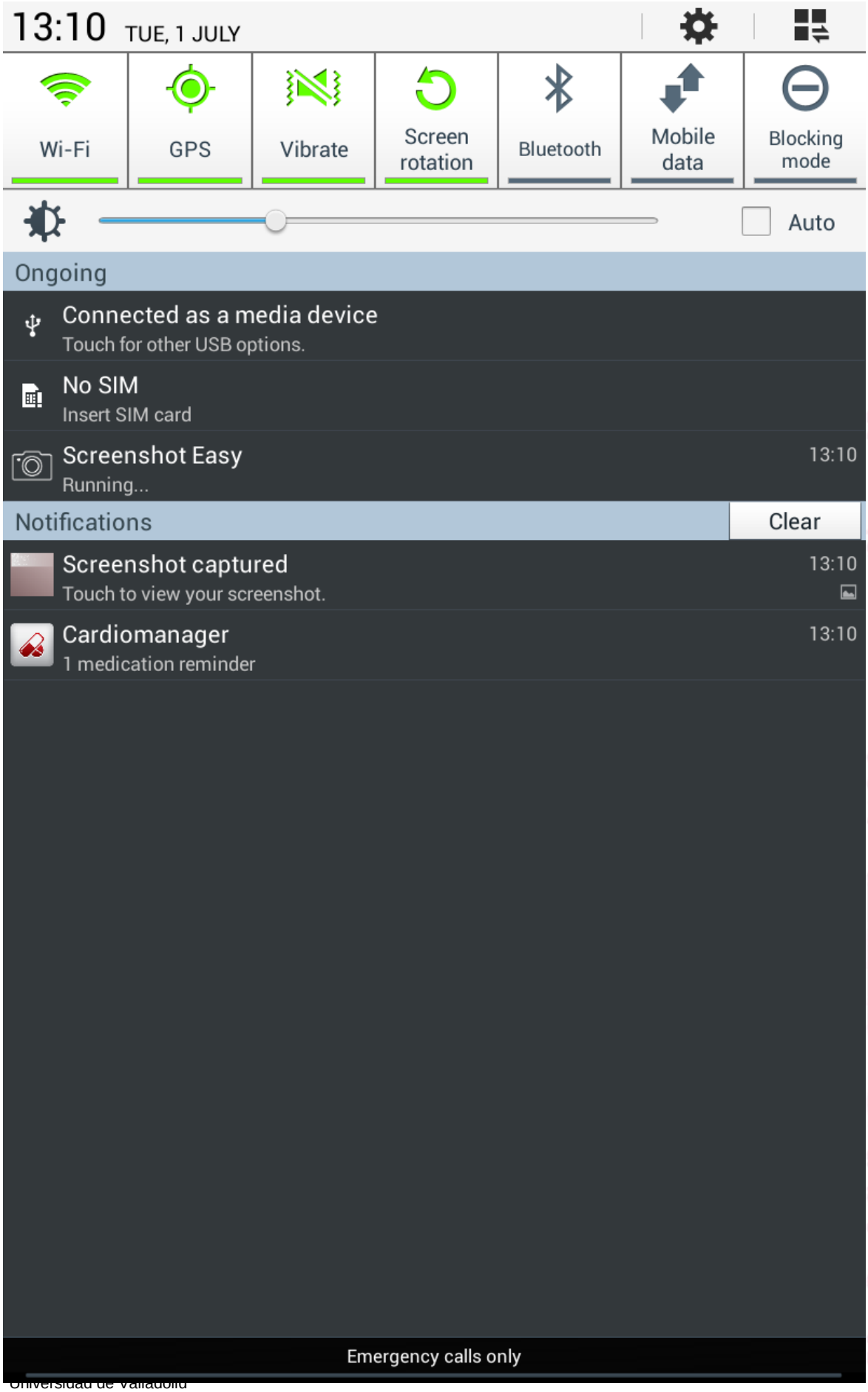
- ✓ Cada vez que llega el momento de tomar una dosis, se comprueba si la dosis inmediatamente anterior fue tomada. En caso de que no haya sido clasificada, se añade automáticamente a la lista de dosis no tomadas.
- ✓ Se recomienda al usuario no cerrar nunca la aplicación y mantenerla en *background*. Sin embargo, puede darse el caso que se fuerce su detención. En este caso, las alarmas programadas para mostrar notificaciones y avisar al usuario serán borradas. Android no dispone de ningún mecanismo para no borrar estas alarmas, por lo tanto, se perderán definitivamente. Sin embargo, *Cardiomanager* dispone de un mecanismo de reactivación de alarmas. Este mecanismo se activa cuando, tras ser cerrada la aplicación, se vuelve a ejecutar. Este mecanismo comprueba cuáles fueron las últimas dosis en las que se pudo avisar al usuario, y calcula cuántas dosis tendrían que haber saltado durante el periodo en que la aplicación ha permanecido cerrada, además de reprogramar las alarmas pertinentes para avisar sobre las dosis venideras. El objetivo de este mecanismo no es solo el restablecimiento de las alarmas, sino también mantener la continuidad de las dosis y poder llevar un seguimiento cien por cien completo del tratamiento.

Información

En este módulo se muestra información médica relacionada con diferentes cardiopatías con el objetivo de solucionar las dudas y preguntas que pueda tener el paciente y su doctor no pueda solucionar en ese momento. Las secciones tratadas en este módulo son:

- ✓ Exceso de ejercicio físico
- ✓ Relación corazón-embarazo
- ✓ Cardiopatía isquémica
- ✓ Hipertensión
- ✓ Diabetes
- ✓ Enfermedad valvular
- ✓ Fibrilación auricular

Pulsando en los diferentes botones, podemos ver qué son las distintas cardiopatías y los principales síntomas y tratamientos.

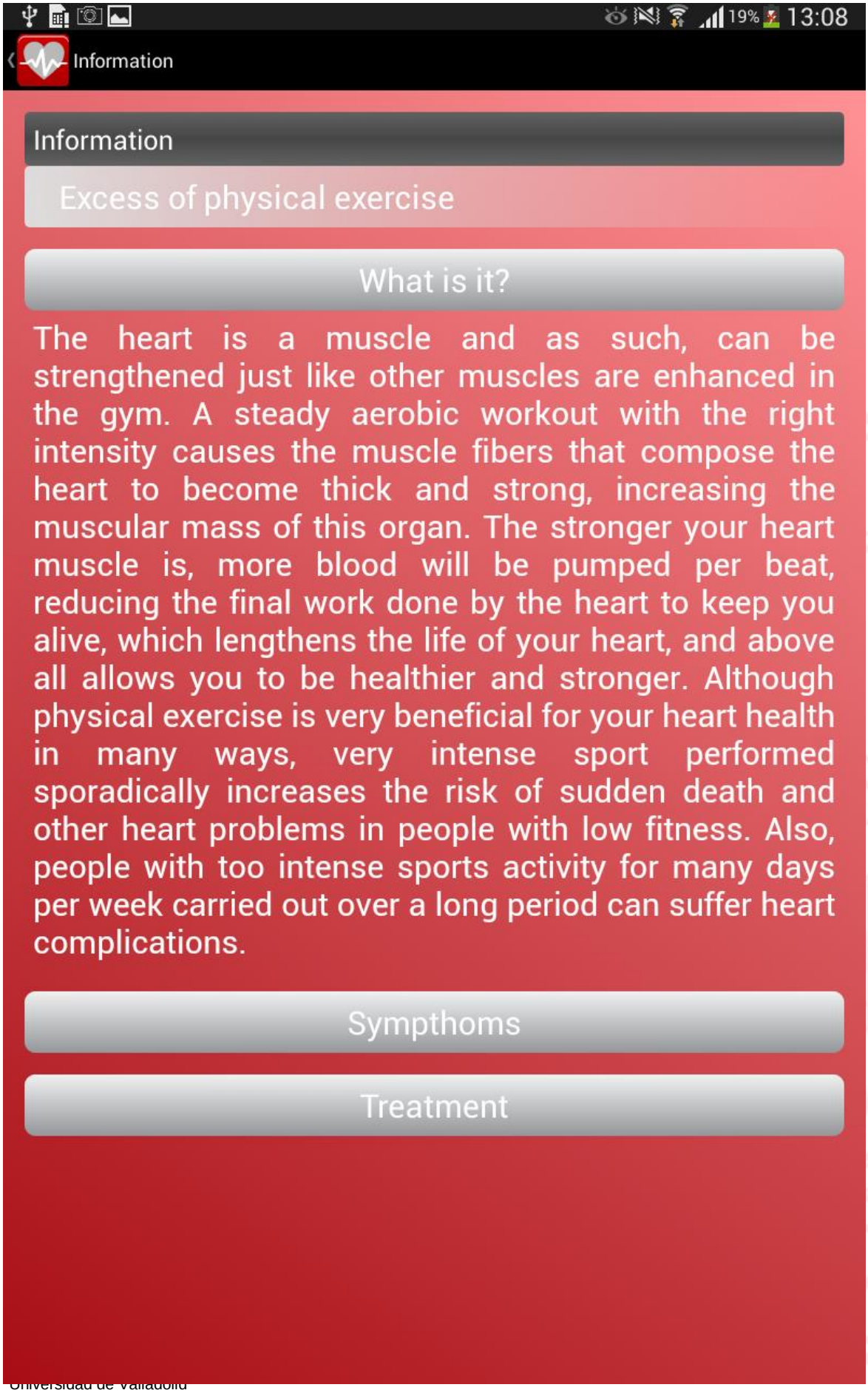

13:10 TUE, 1 JULY
Wi-Fi
GPS
Vibrate
Screen rotation
Bluetooth
Mobile data
Blocking mode
Auto
Ongoing
Connected as a media device
Touch for other USB options.
No SIM
Insert SIM card
Screenshot Easy
Running...
13:10
Notifications
Clear
Screenshot captured
Touch to view your screenshot.
13:10
Cardiomanager
1 medication reminder
13:10
Emergency calls only

Directrices

En este módulo se darán una serie de guías y directrices al paciente sobre cómo debe comportarse en lo respectivo a su enfermedad, con el fin de ayudarle a convivir con su enfermedad y mejorar su calidad de vida. Las secciones tratadas en este módulo son:

- ✓ General.
- ✓ Corazón y deporte.
- ✓ Embarazo.
- ✓ Cardiopatía isquémica.
- ✓ Hipertensión.
- ✓ Diabetes.
- ✓ Enfermedad valvular.
- ✓ Fibrilación auricular.

Pulsando en los diferentes botones, podemos ver qué hábitos saludables se recomienda seguir, que consejos suelen dar los doctores y que suelen prohibir.


19% 13:08
Information
Information
Excess of physical exercise
What is it?
The heart is a muscle and as such, can be strengthened just like other muscles are enhanced in the gym. A steady aerobic workout with the right intensity causes the muscle fibers that compose the heart to become thick and strong, increasing the muscular mass of this organ. The stronger your heart muscle is, more blood will be pumped per beat, reducing the final work done by the heart to keep you alive, which lengthens the life of your heart, and above all allows you to be healthier and stronger. Although physical exercise is very beneficial for your heart health in many ways, very intense sport performed sporadically increases the risk of sudden death and other heart problems in people with low fitness. Also, people with too intense sports activity for many days per week carried out over a long period can suffer heart complications.
Sympthoms
Treatment

Información del paciente

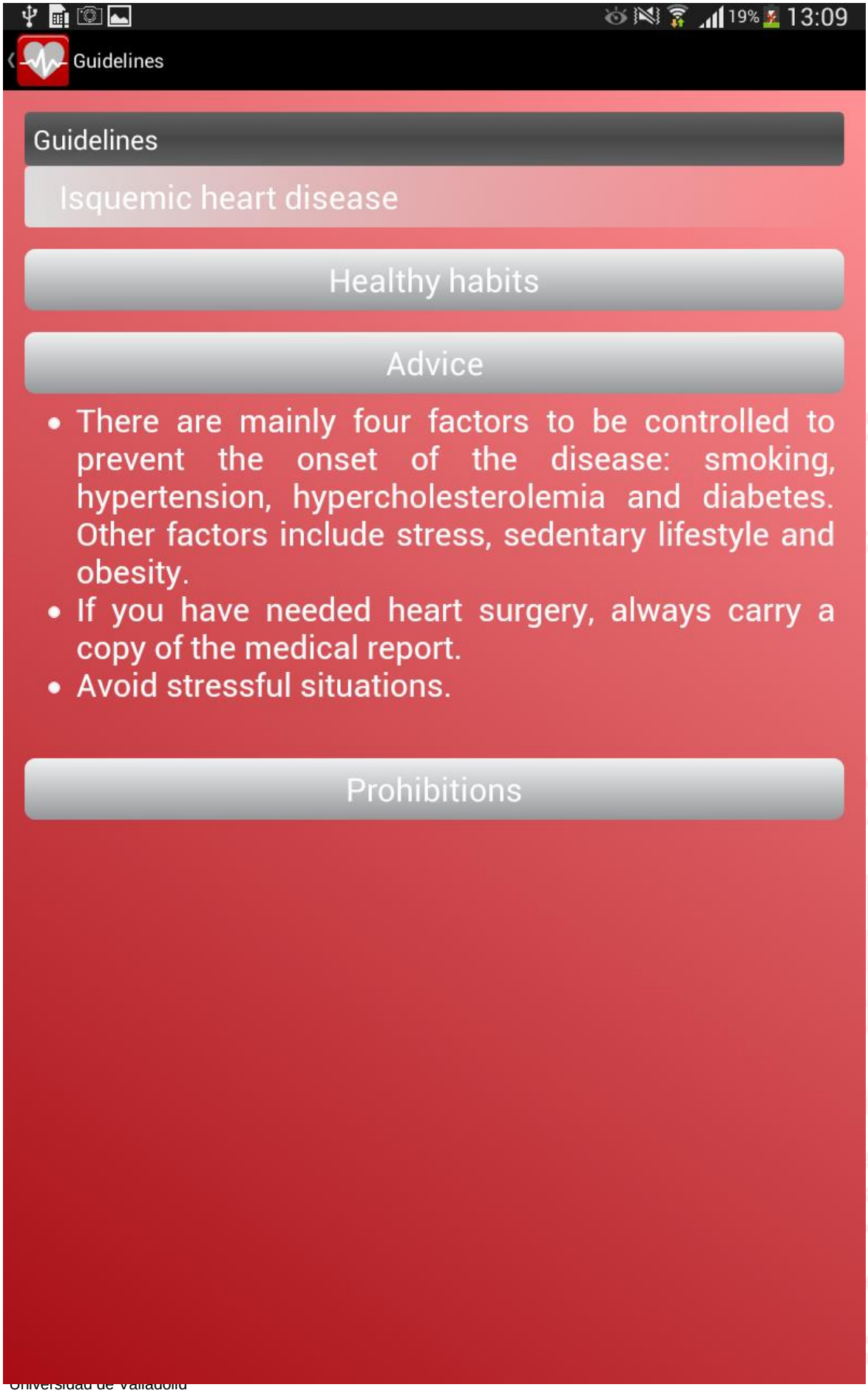
En este módulo, el usuario deberá registrar un nuevo usuario la primera vez, pulsando en el botón de la derecha de pantalla de Login. Tras introducir datos de información personal, podrá acceder a este módulo.

En él se podrá consultar la información personal introducida, y además:

- ✓ Escoger el contacto de emergencia

- ✓ Añadir información sobre el paciente que tenga que ver con su enfermedad, como son:
  - Alergias
  - Hábitos tóxicos
  - Transfusiones
  - Enfermedades
  - Antecedentes quirúrgicos

  y consultarlos en las diferentes listas, además de editarlos o eliminarlos, manteniendo pulsado el dedo sobre el elemento que se desee editar o eliminar.


19% 13:09
Guidelines
Guidelines
Isquemic heart disease
Healthy habits
Advice
• There are mainly four factors to be controlled to prevent the onset of the disease: smoking, hypertension, hypercholesterolemia and diabetes. Other factors include stress, sedentary lifestyle and obesity.
• If you have needed heart surgery, always carry a copy of the medical report.
• Avoid stressful situations.
Prohibitions

Calculadora de Riesgo Cardiovascular

En esta sección, el paciente podrá calcular su riesgo de sufrir un accidente cardiovascular en los próximos años. Además podrá consultar el historial de cálculos.

A través del botón superior de este módulo, se accede a la calculadora. El usuario introducirá los siguientes datos:

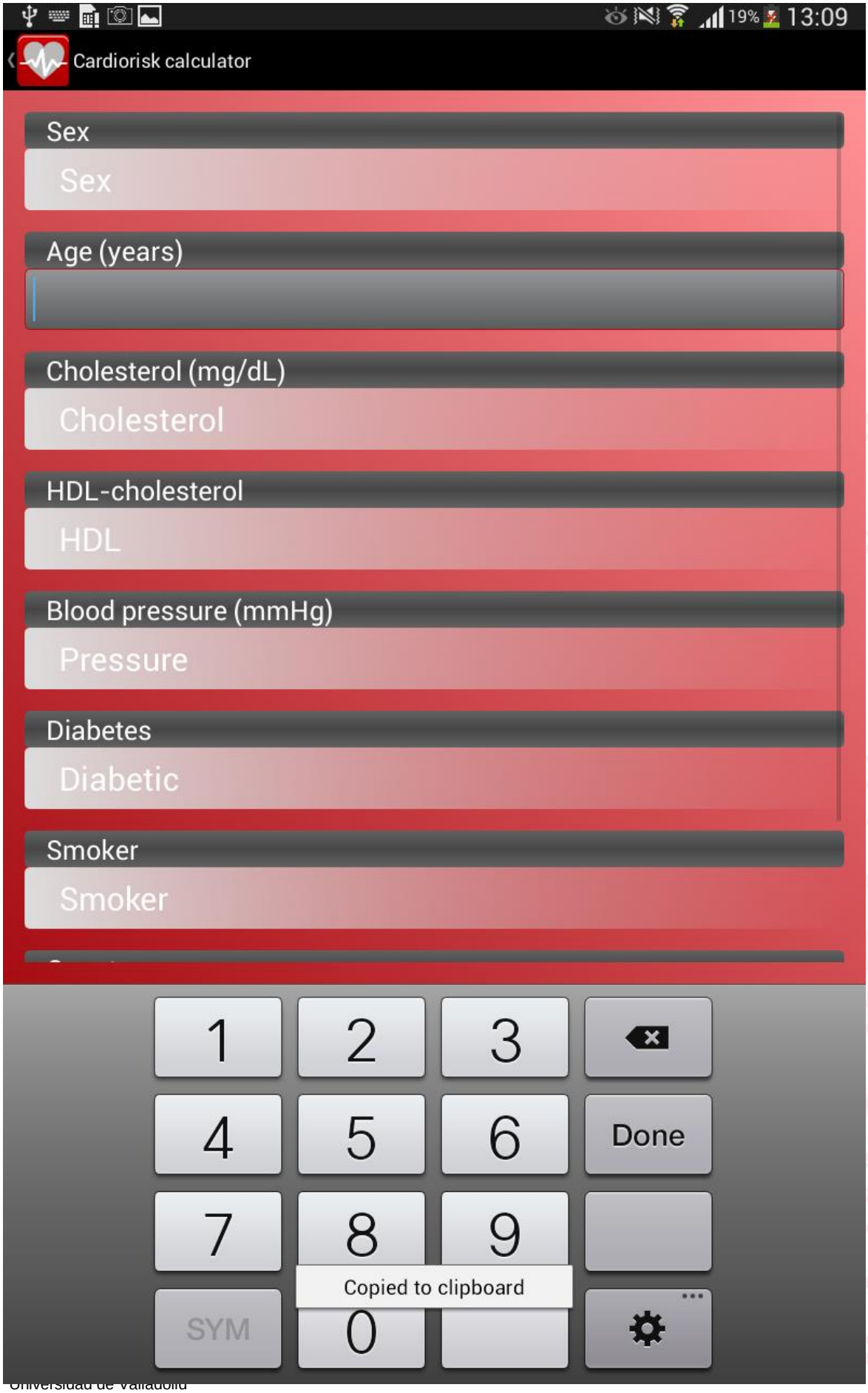
- ✓ Sexo
- ✓ Edad
- ✓ Nivel de colesterol
- ✓ Nivel de colesterol HDL
- ✓ Presión arterial
- ✓ Indicar si es diabétco
- ✓ Indicar si es fumador
- ✓ Indicar si vive en un país de alto riesgo o de bajo riesgo de sufrir un accidente cardiovascular

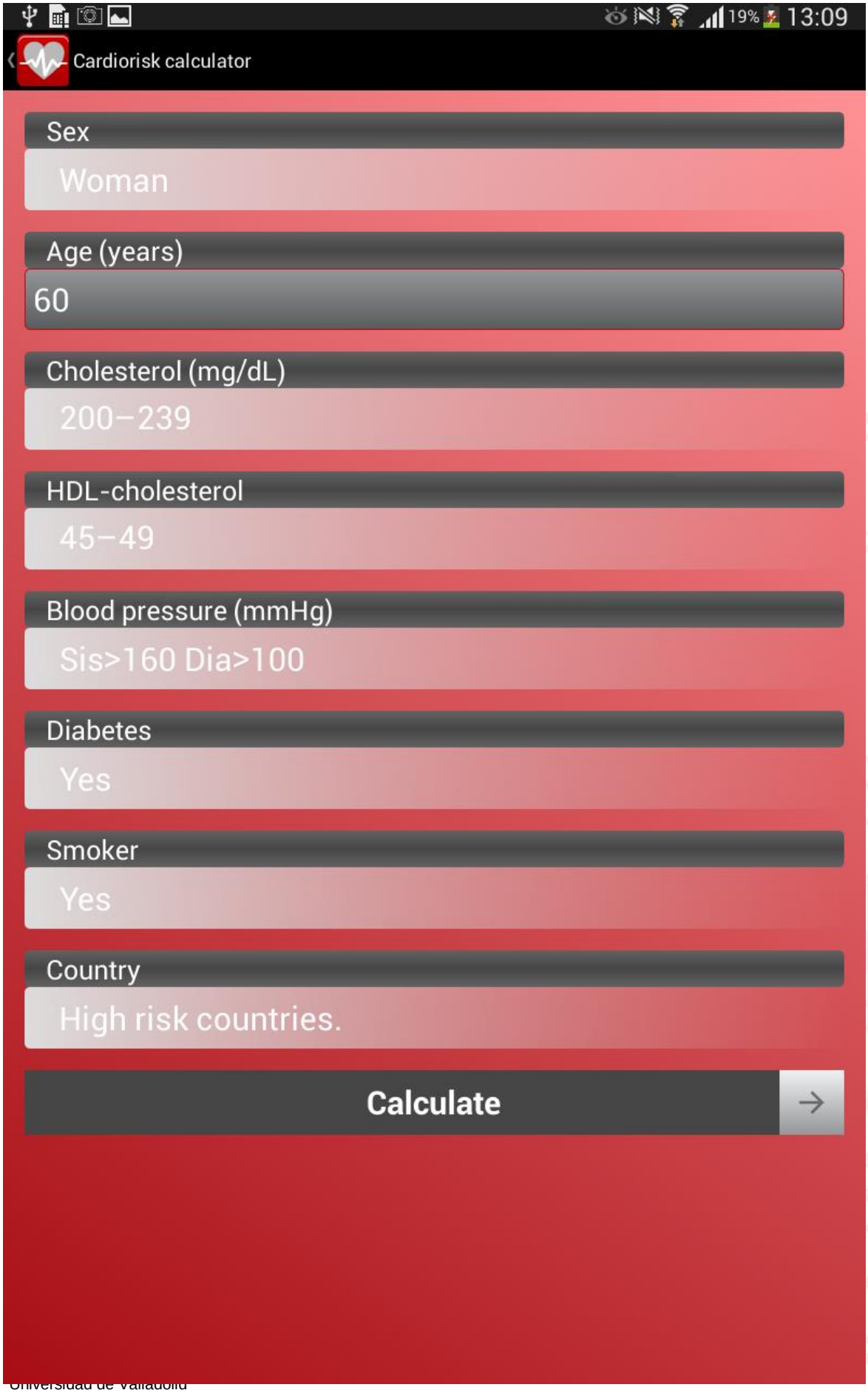
En caso de que tenga alguna duda, a través del menú de la página principal del módulo se puede consultar una breve sección de ayuda para este módulo.

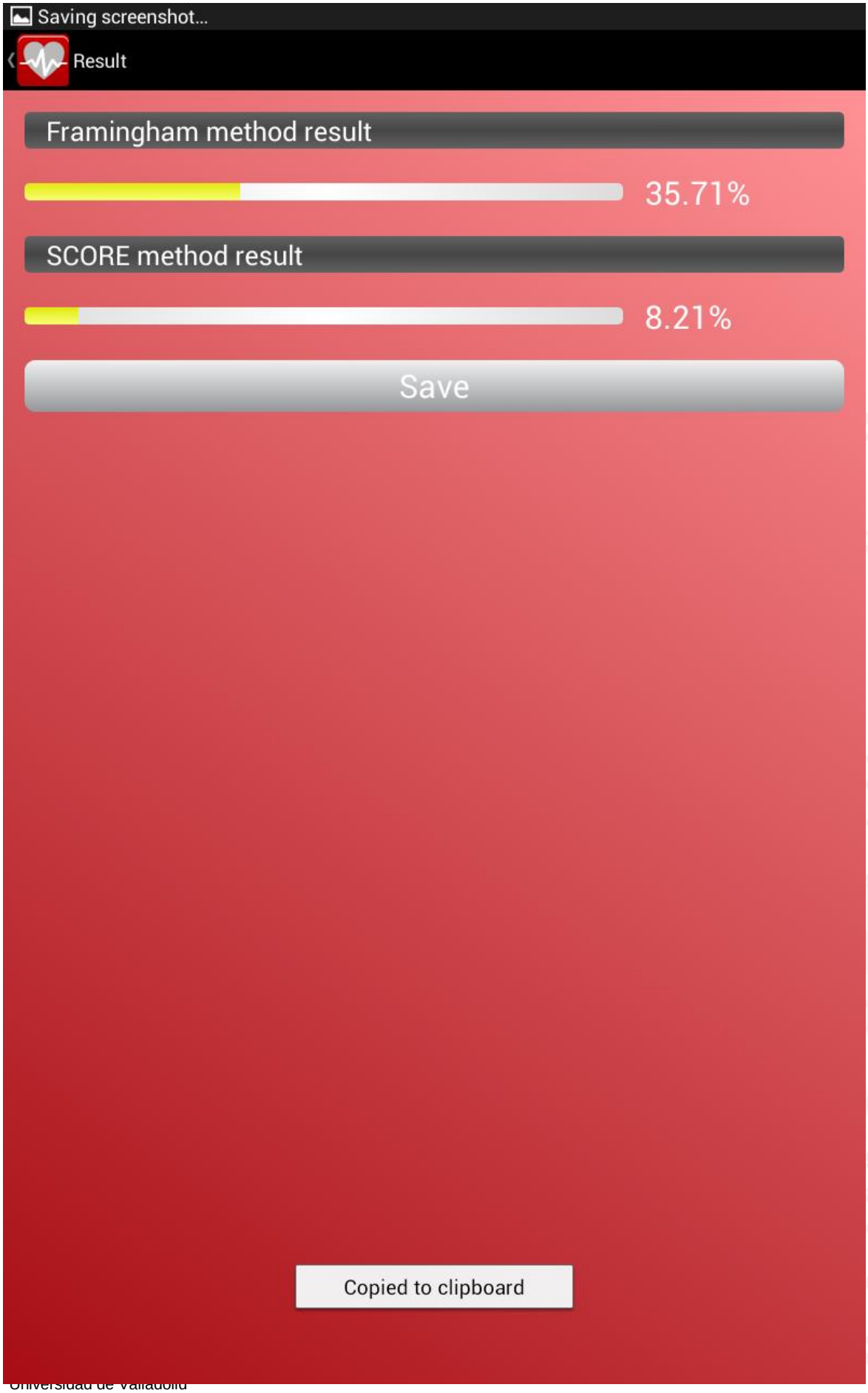
Tras introducir los datos y pulsar en la flecha de la parte inferior derecha de la pantalla, se mostrarán los resultados de este cálculo según dos métodos, y se almacenarán en la base de datos cuando el usuario pulse en guardar.

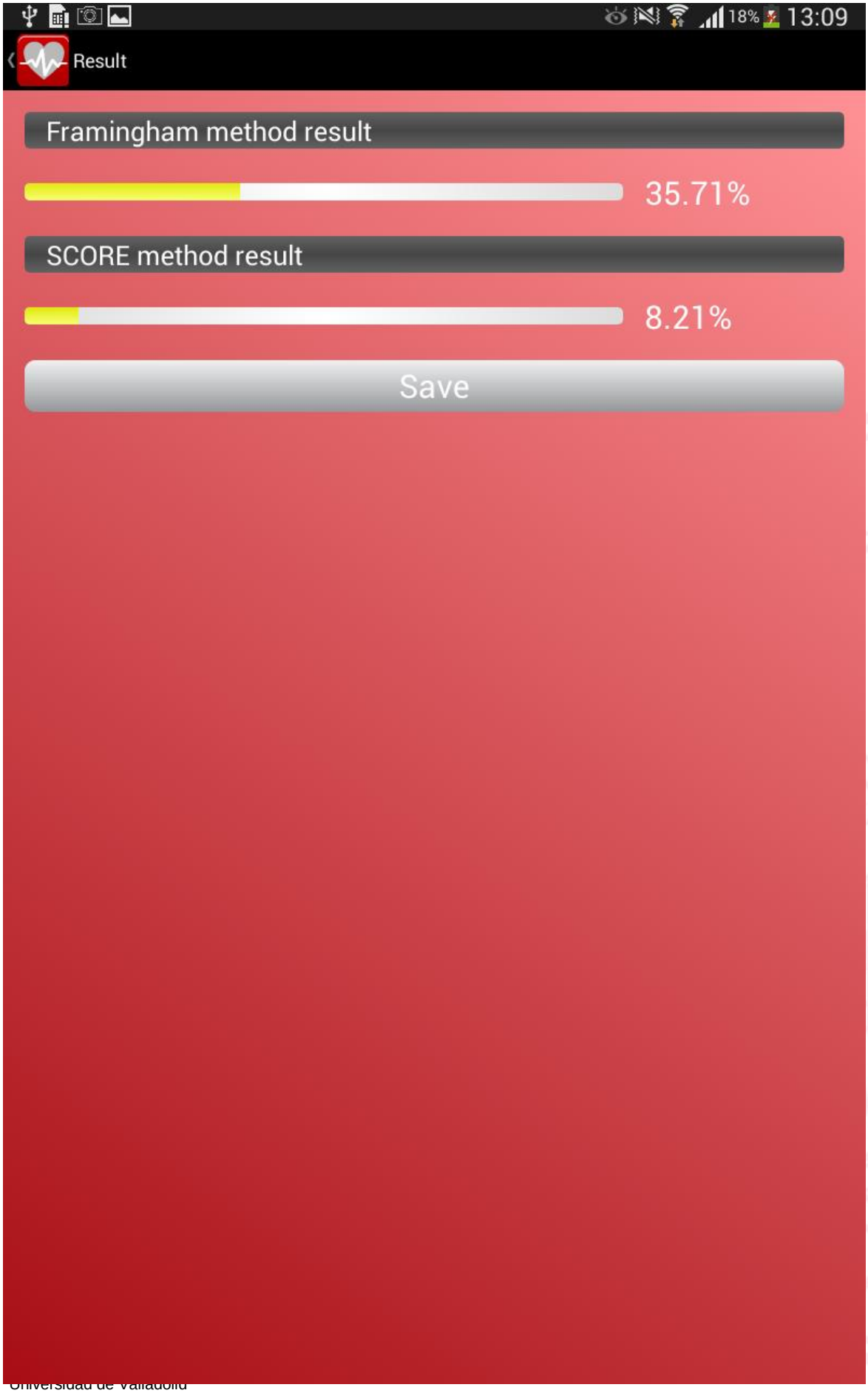
EL botón inferior de la actividad principal permite acceder al historial de cálculos.

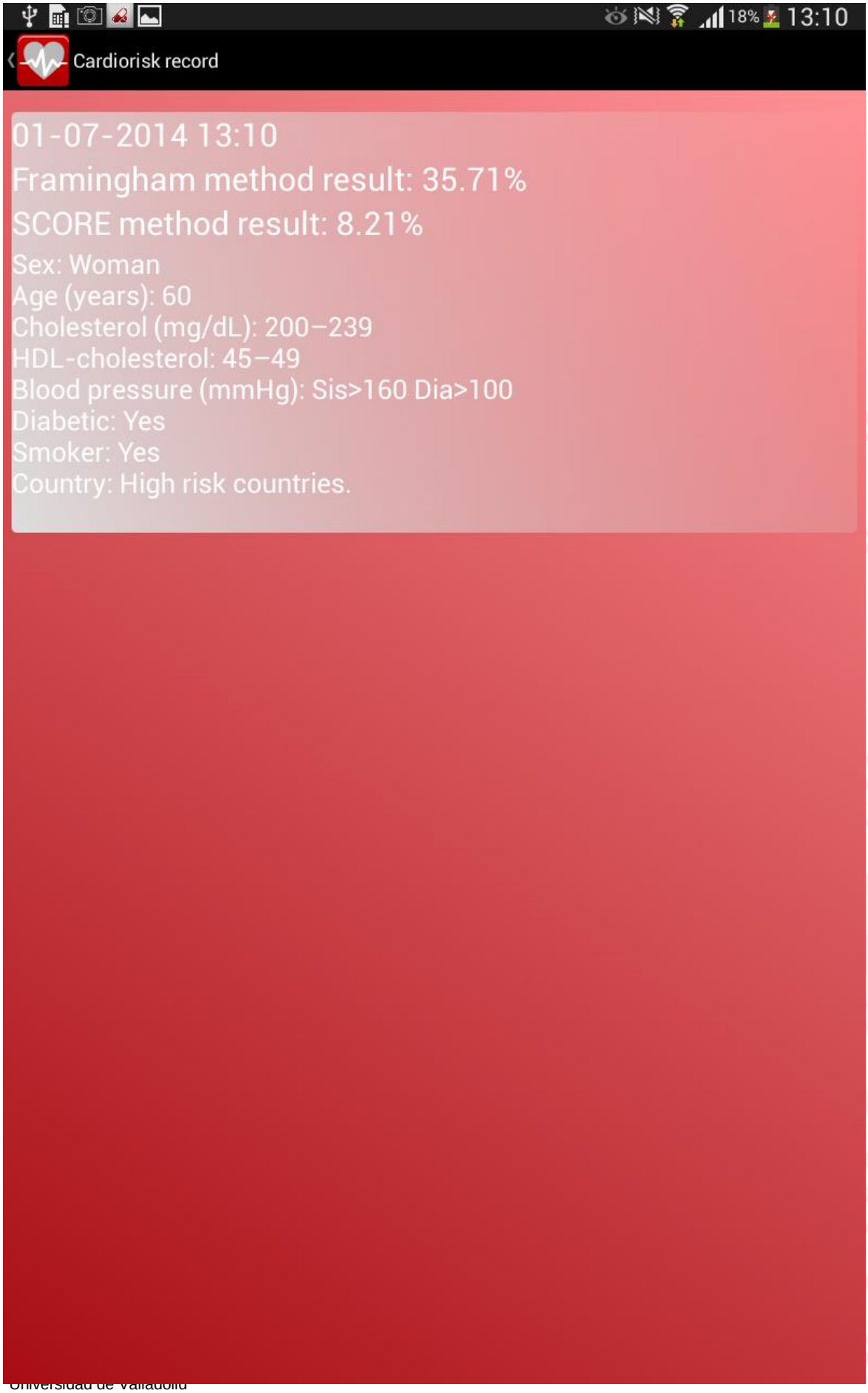

Saving screenshot...
Cardiorisk calculator
?
Copied to clipboard


19% 13:09
Cardiorisk calculator
Sex
Sex
Age (years)
Cholesterol (mg/dL)
Cholesterol
HDL-cholesterol
HDL
Blood pressure (mmHg)
Pressure
Diabetes
Diabetic
Smoker
Smoker
1
2
3
4
5
6
Done
7
8
9
Copied to clipboard
SYM
0


19% 13:09
Cardiorisk calculator
Sex
Woman
Age (years)
60
Cholesterol (mg/dL)
200–239
HDL-cholesterol
45–49
Blood pressure (mmHg)
Sis>160 Dia>100
Diabetes
Yes
Smoker
Yes
Country
High risk countries.
Calculate


Saving screenshot...
Result
Framingham method result
35.71%
SCORE method result
8.21%
Save
Copied to clipboard


18% 13:09
Result
Framingham method result
35.71%
SCORE method result
8.21%
Save


18% 13:10
Cardiorisk record
01-07-2014 13:10
Framingham method result: 35.71%
SCORE method result: 8.21%
Sex: Woman
Age (years): 60
Cholesterol (mg/dL): 200–239
HDL-cholesterol: 45–49
Blood pressure (mmHg): Sis>160 Dia>100
Diabetic: Yes
Smoker: Yes
Country: High risk countries.

Ayuda

Además de a través de este breve manual de uso, el usuario también puede acudir a este módulo, accesible a través del menú desde la actividad principal, para consultar información sobre la aplicación y sobre cómo utilizarla.

Evaluación

Para la evaluación de la aplicación se han utilizado los siguientes dispositivos:

- ✓ Samsung Galaxy Mini 2 GT-6500. Pantalla de 3,5".
- ✓ Tablet Samsung GT-P5200. Pantalla de 10,1".
- ✓ Nexus S y Huawei Ascend Y530. Pantallas de 5" y 4,5". Sin embargo, mismas características de procesador y RAM.
- ✓ Dispositivos virtuales.

Como observamos, se ha probado la aplicación en dispositivos de diferentes capacidades computacionales y de diferentes pantallas, llegando a la conclusión de que funciona correctamente en todas ellas.

Si bien es verdad que se ha detectado un mejor funcionamiento para dispositivos de gran capacidad de procesamiento como son el 2 y el 3, sobre todo a la hora de mostrar determinadas actividades que requieran de mayor potencia gráfica y a la hora de realizar operaciones de lectura y escritura en base de datos. Sin embargo, como se comenta, el funcionamiento ha sido adecuado y rápido en todos los dispositivos, por lo que los usuarios no tendrán ningún problema.

BLOQUE 6
CONCLUSIONES Y LÍNEAS FUTURAS

Conclusiones

El desarrollo de esta aplicación ha supuesto un gran reto a nivel personal debido a diversas razones. En primer lugar, esta aplicación se encuentra uno o dos peldaños por encima de lo cursado en la carrera en lo referente al desarrollo de software, tanto por su tamaño, en torno a unas 10.000 líneas de código Java y unas 6.000 de código XML, como por su importancia tras su realización, pues no pretende quedar en el olvido, sino ser publicada en GooglePlay y estar disponible para todos los usuarios, lo que supone además un punto extra de presión por realizar un buen trabajo.

En segundo lugar, supone un gran reto por el mero hecho de ser mi Trabajo Fin de Grado, y ser una aplicación para enseñar a otras personas, bien sea con el objetivo de conocer su opinión o con el objetivo de realizar una demostración sobre mi trabajo.

También supuso un reto importante el aprender desde cero a programar aplicaciones en Android, pues a pesar de tener experiencia con Java, y al contrario de lo que piensa la mayoría de la gente, programar en Android, no es sólo programar en Java. Si bien es cierto que es necesario haber adquirido conceptos sobre la programación orientada a objetos y a eventos, así como de Java, es necesario indagar en la librería Java de Android para poder considerarse un programador de aplicaciones móviles.

Dentro de los aspectos positivos a resaltar sobre la realización de este proyecto, debo destacar, en primer lugar, el haber llegado a dominar en parte la programación de Android. Aunque no he explotado todas las oportunidades que nos ofrece este sistema operativo, he intentado hacer uso de la mayor cantidad de características posibles, con el fin de enriquecer la aplicación y mis conocimientos. También he aprendido la importancia de realizar una buena planificación y cumplir con los plazos fijados en la misma, aunque creo que no he sido capaz de fijar esos plazos de forma correcta, algo que espero aprender en los próximos años. Mientras he cursado diversas asignaturas de programación, he llegado a aborrecer el uso de herramientas de control de versiones, pero tras la realización de esta aplicación he visto sus puntos positivos y por qué es tan necesario ser riguroso en este aspecto.

Otros aspectos de programación importantes a tener en cuenta es que he aprendido el lenguaje SQL por mi cuenta para elaborar la base de datos, algo de lo cual estoy bastante orgulloso y que me sirvió para llevar mejor mi asignatura de Tecnologías de Aplicaciones Web, donde uno de los apartados era la realización de una base de datos. Además, la programación en Android me ha ayudado a obtener una muy buena nota en la asignatura optativa de Desarrollo de Aplicaciones para Dispositivos Móviles.

Además de los aspectos de programación, debo reseñar que durante la realización de este Trabajo Fin de Grado he tenido una toma de contacto con lo que es la investigación en un departamento de la universidad, algo que es importante de cara a plantearme mi futuro, tanto a corto como a largo plazo.

El único aspecto negativo reseñable de la realización de este TFG es el tiempo empleado. Aun siendo consciente de que tendría que emplear más de las 450 horas que propone el plan, incluyendo las prácticas en empresa, desconocía que tendría que utilizar más de 600, lo que supone un aumento del 33% aproximadamente, dejando claro que la falta de experiencia, el tiempo empleado en aprender lenguajes, pruebas, solucionar problemas, es siempre importante en proyectos software, y más si se trata de programadores junior, como es mi caso.

Por último, me gustaría recomendar a cualquier alumno la realización de una aplicación para Android como TFG, pues, desde mi punto de vista, es motivante, es sencillo aprender, ya que la comunidad mundial de Android proporciona información de calidad y abundante, y es muy útil de cara al futuro laboral.

Líneas futuras

Desarrollo de la aplicación para Windows Phone, iOS y Firefox OS

Si bien es cierto que Android es el sistema operativo más utilizado para teléfonos móviles, existe un pequeño porcentaje de usuarios a los que la aplicación no tiene acceso. Por lo tanto, sería interesante el diseño de una aplicación tipo web con HTML5, CSS y JavaScript, que puede funcionar en cualquiera de las tres plataformas sin apenas tener que adaptar código, haciendo uso de PhoneGap, actualmente Apache Cordova, o plataformas similares que permitan este desarrollo. También se podrían desarrollar nativamente para cada uno de los sistemas operativos, pero la cantidad de tiempo invertida en ello sería muy alta, y la verdad, no sería rentable, con los costes de publicación que supone, sobre todo para iOS.

Módulo con *e-Health Sensor Platform V2.0 for Arduino*

Este conjunto de sensores, disponibles para Arduino y para Raspberry Pi, nos permite realizar medidas del cuerpo del paciente, como son la temperatura, la presión sanguínea o el pulso. Por lo tanto, a través de esta plataforma, se podría diseñar un módulo que recibiera toda esta información y construyera diferentes actividades almacenando estos datos. El problema de esto es, sin ninguna duda, el coste que supone adquirir un equipo completo de este tipo. Además, el usuario tendría que adquirir también estos dispositivos, por lo tanto, no sería rentable, aunque desde el punto de vista de la aplicación, sería muy enriquecedor.

Aplicación inteligente

También sería interesante la creación de un módulo que, a partir de los datos introducidos de actividades y medicación, construya un informe en el que se recomiende practicar más determinada actividad o evitarla, que se proporcione al usuario determinados consejos o prohibiciones a partir de sus hábitos, calcular periódicamente el riesgo cardiovascular o avisar al usuario de que se está descuidando en la toma de cierto medicamento. El reto de esta mejora sin duda está en investigar cómo añadir los hábitos, y a partir de los mismos, mostrar los consejos y prohibiciones. Para ello habría que consultar estudios o libros de referencia que ayudaran a la implementación de este módulo.

Consulta virtual

Aunque existen múltiples aplicaciones que informan sobre enfermedades, no existe ninguna que, introduciendo diferentes datos como edad, sexo y síntomas que el paciente esté sintiendo en ese momento, muestre al usuario qué enfermedades puede estar padeciendo, ordenada de menor a mayor gravedad, recomendando la asistencia al médico con mayor o menos urgencia, y mostrando información sobre las enfermedades resultantes.

Podría resultar muy útil y enriquecedor, y sólo se tendría que investigar determinados síntomas de habituales enfermedades cardiacas, aunque puede realizarse para otras enfermedades más comunes también.

REFERENCIAS BIBLIOGRÁFICAS

[1] http://www.fundaciondelcorazon.com/

[2] http://www.seh-lelha.org/modelries.htm

[3] Informe anual sobre el desarrollo de la Sociedad de la Información en España, eEspaña 2006, pág. 199

[4] Google Play

[5] http://www.ibertronica.es/

[6] http://www.xatakamovil.com/

[7] http://developer.android.com/index.html