



UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática



**TFG del Grado en Ingeniería
Informática**

**UBUDiabetes 2.0
Documentación Técnica**



Presentado por Luis Miguel Inapanta Oyana
en Universidad de Burgos — 27 de junio de 2018
Tutor: Dr. Raúl Marticorena Sánchez

Índice general

Índice general	I
Índice de figuras	III
Índice de tablas	VII
Apéndice A Plan de Proyecto Software	1
A.1. Introducción	1
A.2. Planificación temporal	1
A.3. Estudio de viabilidad	18
Apéndice B Especificación de Requisitos	25
B.1. Introducción	25
B.2. Objetivos generales	25
B.3. Catálogo de requisitos	25
B.4. Especificación de requisitos	28
Apéndice C Especificación de diseño	37
C.1. Introducción	37
C.2. Diseño de datos	37
C.3. Diseño procedimental	40
C.4. Diseño arquitectónico	45
C.5. Diseño de pruebas	68
Apéndice D Documentación técnica de programación	81
D.1. Introducción	81
D.2. Estructura de directorios	81

D.3. Manual del programador	83
D.4. Compilación, instalación y ejecución del proyecto	86
D.5. Pruebas del sistema	88
Apéndice E Documentación de usuario	91
E.1. Introducción	91
E.2. Requisitos de usuarios	91
E.3. Instalación	91
E.4. Manual del usuario	92
Bibliografía	109

Índice de figuras

A.1. Progreso en el <i>sprint</i> 0.	2
A.2. Progreso en el <i>sprint</i> 1.	3
A.3. Progreso en el <i>sprint</i> 2.	4
A.4. Progreso en el <i>sprint</i> 3.	5
A.5. Progreso en el <i>sprint</i> 4.	6
A.6. Progreso en el <i>sprint</i> 5.	7
A.7. Progreso en el <i>sprint</i> 6.	8
A.8. Progreso en el <i>sprint</i> 7.	9
A.9. Progreso en el <i>sprint</i> 8.	10
A.10. Progreso en el <i>sprint</i> 9.	11
A.11. Progreso en el <i>sprint</i> 10.	12
A.12. Progreso en el <i>sprint</i> 11.	13
A.13. Progreso en el <i>sprint</i> 12.	14
A.14. Progreso en el <i>sprint</i> 13.	15
A.15. Progreso en el <i>sprint</i> 14.	16
A.16. Progreso en el <i>sprint</i> 15.	17
A.17. Progreso en el <i>sprint</i> 16.	18
A.18. Datos Globales de Prevalencia (%)	21
 B.1. Diagrama general de casos de uso	 29
 C.1. Proceso de <i>registro</i>	 41
C.2. Registro de un nivel de glucemia	42
C.3. Calcular bolo corrector	42
C.4. Consultar historial de glucemias	43
C.5. Consultar registros	44
C.6. Ajustes	45

C.7. Ajustes	46
C.8. Paquetes Java 1	47
C.9. Paquetes Java 2	48
C.10. Paquetes Java 3	48
C.11. Clase CalculaBolo	49
C.12. Clase Alimento	50
C.13. Clase DetallesIngesta	50
C.14. Clase RegistroIngestas	51
C.15. Clase DataBaseHelper	51
C.16. Clase DataBaseManager	52
C.17. Clase ValoresPOJO	53
C.18. Clase AjustesListAdapter	53
C.19. Clase ChartListAdapter	54
C.20. Clase IngestaAlimentoListAdapter	54
C.21. Clase IngestaDetallesListAdapter	55
C.22. Clase IngestaRegistroListAdapter	55
C.23. Clase ListViewUtility	55
C.24. Activity Ajustes	56
C.25. Fragment Ajustes	56
C.26. Activity Carbohidratos A	57
C.27. Activity Carbohidratos B	58
C.28. Fragment Carbohidratos	58
C.29. Activity ConsultaDetallesListaIngesta	59
C.30. Fragment ConsultaDetallesListaIngesta	59
C.31. Activity Historial	60
C.32. Fragment Historial	60
C.33. Activity Incidencias	61
C.34. Fragment Incidencias	61
C.35. Clase Launcher	61
C.36. Activity MenuPrincipal	62
C.37. Fragment MenuPrincipal	62
C.38. Activity Registro	63
C.39. Fragment Registro	64
C.40. Activity RegistroGlucemias	64
C.41. Fragment RegistroGlucemias	64
C.42. Activity RegistroListaIngesta	65
C.43. Fragment RegistroListaIngesta	65
C.44. Activity SplashScreen	66
C.45. Test: Caso clínico	66
C.46. Test: Sin alimentos 1	67
C.47. Test: Alimentos de 1 en 1	68

C.48.Caso de prueba 1	70
C.49.Caso de prueba 2	70
C.50.Caso de prueba 3	71
C.51.Caso de prueba 4	71
C.52.Caso de prueba 5: Datos paciente	72
C.53.Caso de prueba 5: Datos alimentos 1	72
C.54.Caso de prueba 5: Datos alimentos 2	73
C.55.Caso de prueba 5: Datos alimentos 3	73
C.56.Caso de prueba 5: Datos alimentos 4	74
C.57.Caso de prueba 5: Datos alimentos 5	74
C.58.Caso de prueba 5: Datos alimentos 6	75
C.59.Caso de prueba 5: Datos alimentos 7	75
C.60.Caso de prueba 5: Datos alimentos 8	76
C.61.Caso de prueba 5: Datos alimentos 9	76
C.62.Caso de prueba 5: Datos alimentos 10	77
C.63.Caso de prueba 5: Datos alimentos 11	77
C.64.Caso de prueba 5: Datos alimentos 12	78
C.65.Caso de prueba 5: Datos alimentos 13	78
D.1. Código fuente: GitHub	82
D.2. Android Studio: Descarga	83
D.3. Android Studio: Path de instalación	84
D.4. Android Studio: Dependencias	85
D.5. Android Studio: Plataformas	86
D.6. Android Studio: Tools	87
D.7. Android Studio: Cargar Proyecto	88
D.8. Android Studio: Nuevo Test Unitario A	89
D.9. Android Studio: Nuevo Test Unitario B	90
E.1. SplashScreen	93
E.2. Registro-Perfil de Usuario 1	94
E.3. Registro-Perfil de Usuario 2	95
E.4. Menú principal	96
E.5. Registro de glucemias A	97
E.6. Registro de glucemias B	98
E.7. Incidencias A	99
E.8. Incidencias B	100
E.9. Carbohidratos A	102
E.10. Carbohidratos B	103
E.11. Carbohidratos C	104
E.12. Historial de glucemias C	105

E.13.Consultar registros	106
E.14.Consultar detalles de registros	107
E.15.Ajustes	108

Índice de tablas

A.1. Costes totales	20
B.1. CU-1 Registrar datos de perfil	30
B.2. CU-2 Registrar niveles de glucemia	31
B.3. CU-3 Calcular Bolo Corrector	32
B.4. CU-4 Consultar historial de glucemias	33
B.5. CU-5 Consultar registros diarios	34
B.6. CU-6 Ajustes	35
C.1. Base de datos. Glucemias	37
C.2. Base de datos. Incidencias	38
C.3. Base de datos. Alimentos	38
C.4. Base de datos. Registros diarios	38
C.5. Base de datos. Detalles Registros diarios	39
C.6. SharedPreferences. PreferenciasUsuario	40

Apéndice A

Plan de Proyecto Software

A.1. Introducción

En este apartado se pretende estimar el trabajo, el tiempo y los costes necesarios para la realización del proyecto. Cabe mencionar que se trabajaron con 3 repositorios:

- UbuDiabetes2.0_GII_14b: Repositorio inicial de partida. En él se realizaron las primeras configuraciones y ediciones para el nuevo proyecto.
- UBUDiabetes2.0_GII17.0J: Repositorio nuevo creado para continuar con el proyecto. Se puede consultar más detalles del motivo de la creación de un nuevo repositorio para el proyecto en la sección **Aspectos relevantes del desarrollo del proyecto**.
- Documentation_UBUDiabetes2.0_GII17.0J: Repositorio creado únicamente para la realización de la documentación.

A.2. Planificación temporal

A continuación, se detallarán los diferentes objetivos que se han establecido para cada *sprint* y, a su vez, el progreso obtenido en los mismos. Para ello se ha utilizado una metodología ágil denominada *SCRUM* [3]. Se empleará a su vez el gestor de tareas provisto por GitHub y se generarán gráficos *burndown* para el seguimiento de los *sprint*, los cuales son provistos por la extensión *ZenHub*.

Sprint 0 - Febrero 19-2018 - Marzo 01-2018

En este sprint se pretenden instalar las herramientas necesarias para la realización del proyecto, así como la lectura y comprensión de las memorias y anexos de los proyectos previos relacionados con el mismo. A su vez se pretende ejecutar la aplicación en un emulador para comprobar su correcto funcionamiento. En este sprint se crearon algunas tareas que, como se puede observar quedaron sin finalizar ya que, como se mencionó en la sección **Aspectos relevantes del desarrollo del proyecto**, se creó un nuevo proyecto en el que se continúa con la realización del proyecto. Podemos observar el progreso del *sprint* en la figura A.1

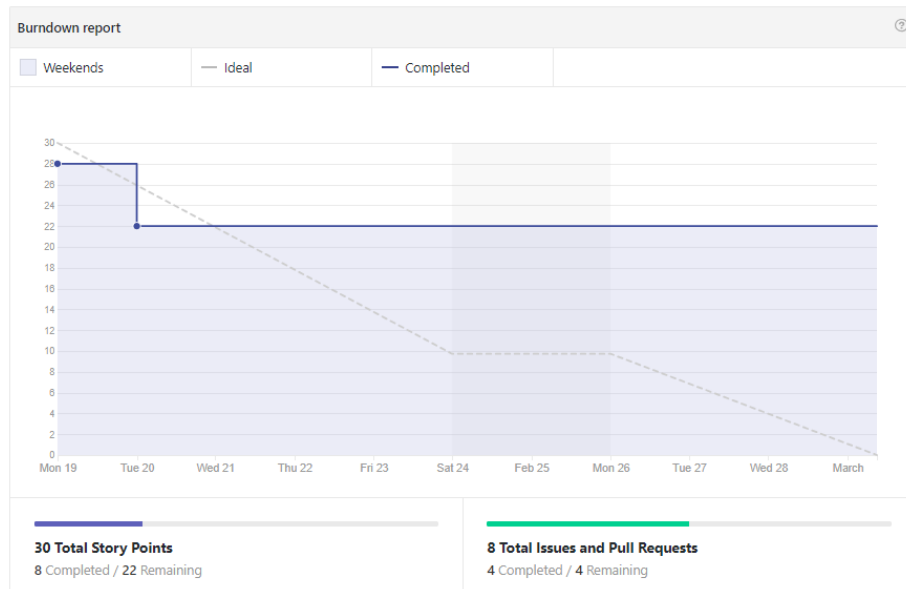
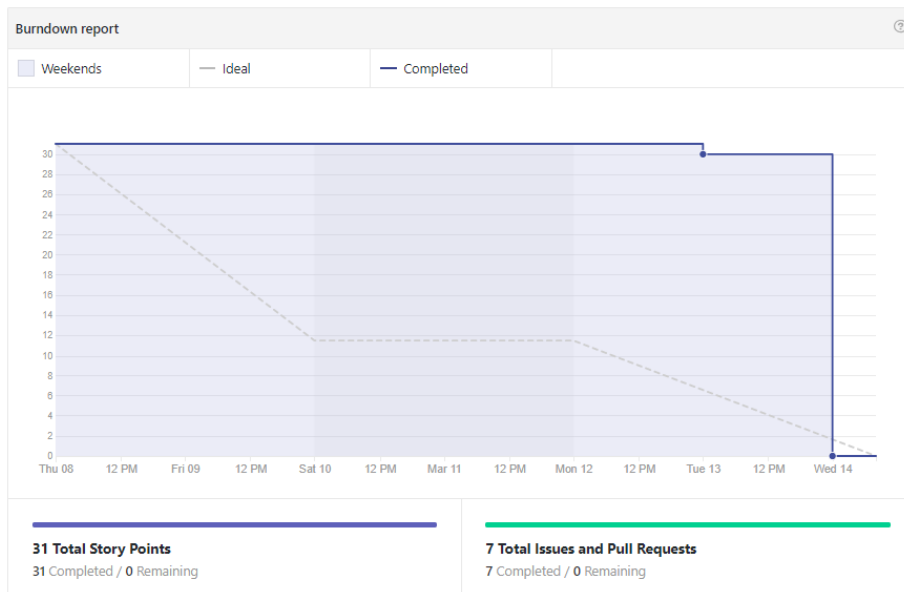


Figura A.1: Progreso en el *sprint* 0.

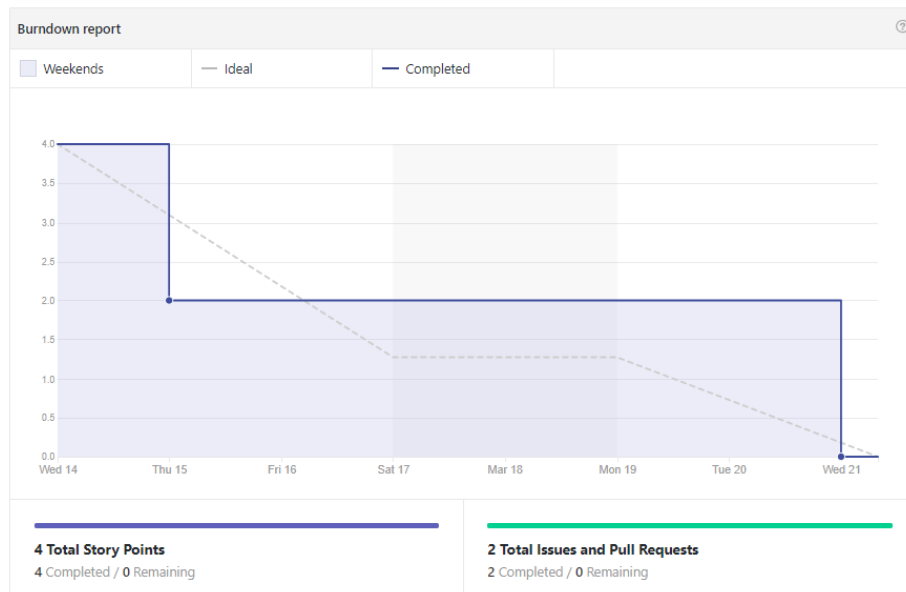
Sprint 1 - Marzo 8-2018 - Marzo 14-2018

En este *sprint* se pretende leer la documentación relacionada con el *testing* con *Espresso* y empezar con las configuraciones necesarias para la creación de los primeros test instrumentados. Podemos observar el progreso del *sprint* en la figura A.2

Figura A.2: Progreso en el *sprint* 1.

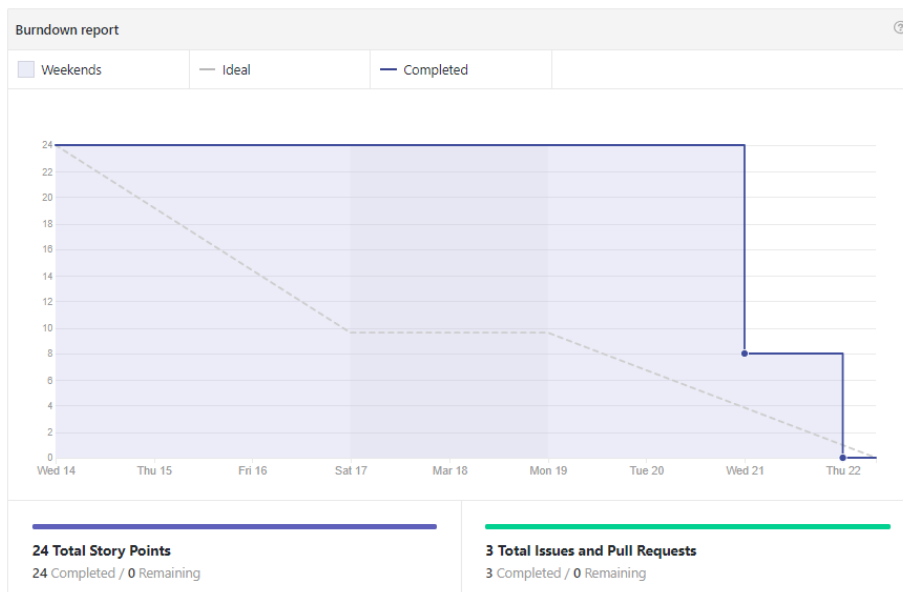
Sprint 2 - Marzo 14-2018 - Marzo 21-2018

En este *sprint* se pretende entender el funcionamiento de la herramienta *Texmaker* para poder empezar a realizar las primeras modificaciones para la documentación del proyecto. Así mismo, se pretende crear un nuevo proyecto en Android Studio, con su repositorio en *GitHub*, con las configuraciones necesarias para poder continuar con el mismo. Podemos observar el progreso del *sprint* en la figura [A.3](#)

Figura A.3: Progreso en el *sprint* 2.

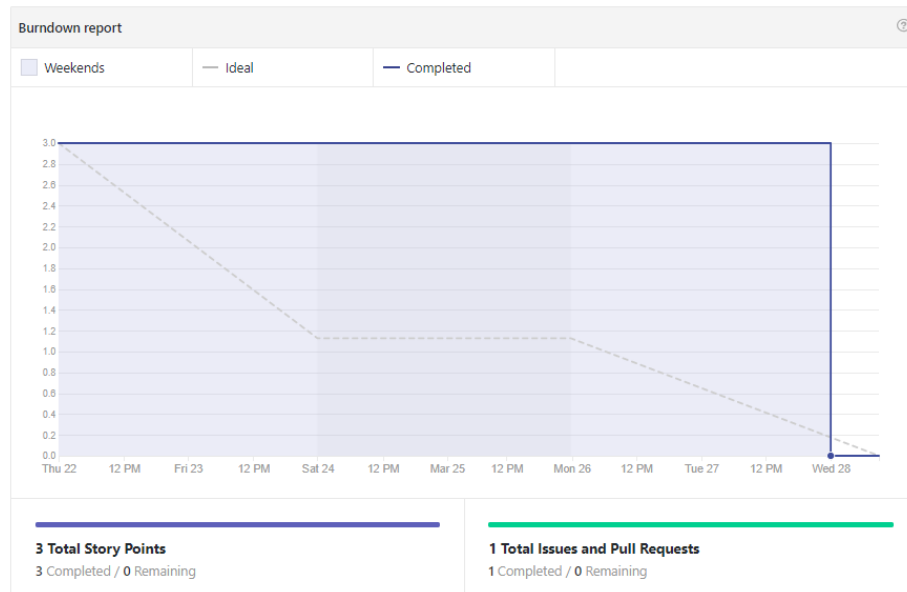
Sprint 3 - Marzo 15-2018 - Marzo 22-2018

Este *sprint* se realizó en la misma semana del *sprint* anterior, pero en el nuevo repositorio creado. En él se pretende migrar todos los datos de UBUDiabetes 1.0 al nuevo proyecto creado, realizando las configuraciones y modificaciones necesarias para la correcta compilación y ejecución del proyecto ya que este nuevo proyecto se realizó en una nueva versión de Android Studio, lo que requería configuraciones diferentes a las del proyecto anterior, por ejemplo, nuevas librerías, diferentes dependencias de Android, contenedores diferentes en los diseños, etc. Así mismo se pretende seguir con la lectura de la documentación relacionada al *testing*. Podemos observar el progreso del *sprint* en la figura [A.4](#)

Figura A.4: Progreso en el *sprint* 3.

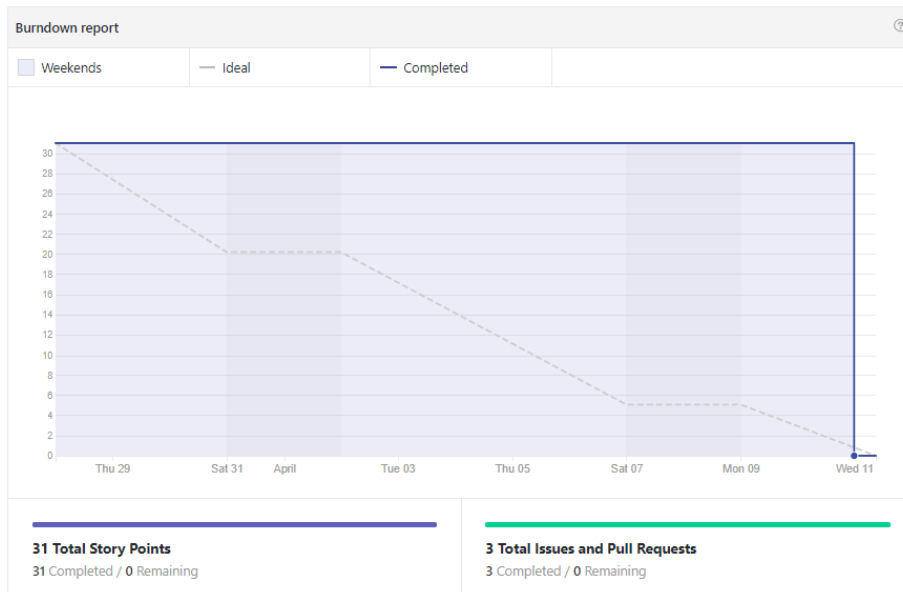
Sprint 4 - Marzo 22-2018 - Marzo 28-2018

En este *sprint* se pretende realizar y ejecutar los primeros test instrumentados con *Espresso*. Podemos observar el progreso del *sprint* en la figura [A.5](#)

Figura A.5: Progreso en el *sprint* 4.

Sprint 5 - Marzo 28-2018 - Abril 11-2018

En este *sprint* se pretende crear las primeras líneas para la documentación del proyecto, crear un nuevo diseño para la pantalla “Perfil-Registro” y crear un buscador de alimentos para la pantalla “Carbohidratos”. Podemos observar el progreso del *sprint* en la figura [A.6](#)

Figura A.6: Progreso en el *sprint* 5.

Sprint 6 - Abril 12-2018 - Abril 19-2018

En este *sprint* se realizaron varias tareas, la mayoría relacionadas con modificaciones en la interfaz gráfica del usuario:

- Empezar con la configuración y ejecución para *sonarQube*.
- Introducir en la tabla “alimentos” el índice glucémico de cada alimento.
- Mostrar en la lista de alimentos ingeridos por el usuario, la categoría según el índice glucémico de cada alimento.
- Buscar e implementar una manera de guardar en un registro los alimentos ingeridos por el usuario con su Bolo corrector final.
- Añadir opción de configuración (en el Perfil) de número de decimales a devolver en el cálculo del bolo corrector.
- Implementar la selección y eliminación de los *items* de la lista de ingesta.
- Cambiar la forma de almacenamiento de las raciones en gramos para cada alimento.

- Implementar la selección de una imagen de galería para el Perfil del usuario.

Podemos observar el progreso del *sprint* en la figura A.7

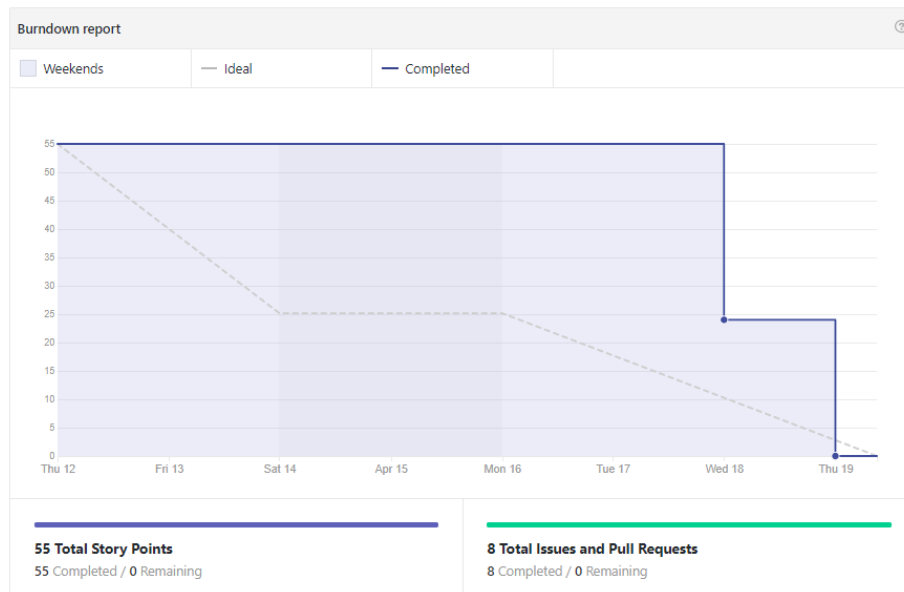


Figura A.7: Progreso en el *sprint* 6.

Sprint 7 - Abril 19-2018 - Abril 26-2018

En este *sprint* se pretende continuar con la memoria del proyecto y realizar nuevas modificaciones para la interfaz gráfica del usuario. Las tareas para este *sprint* fueron:

- Asignar permisos de uso de la cámara y permisos de escritura en la tarjeta SD para Android 6.0 y superior.
- Cambiar la carga de los datos de la tabla “Alimentos” de la pantalla “Carbohidratos” a la pantalla “SplashScreen”.
- Implementar orden inverso para las listas.
- Mostrar el sumatorio de los Hidratos de Carbono y del Bolo corrector en la pantalla “Carbohidratos”.
- Continuar con la documentación para la memoria del proyecto.

Podemos observar el progreso del *sprint* en la figura A.8

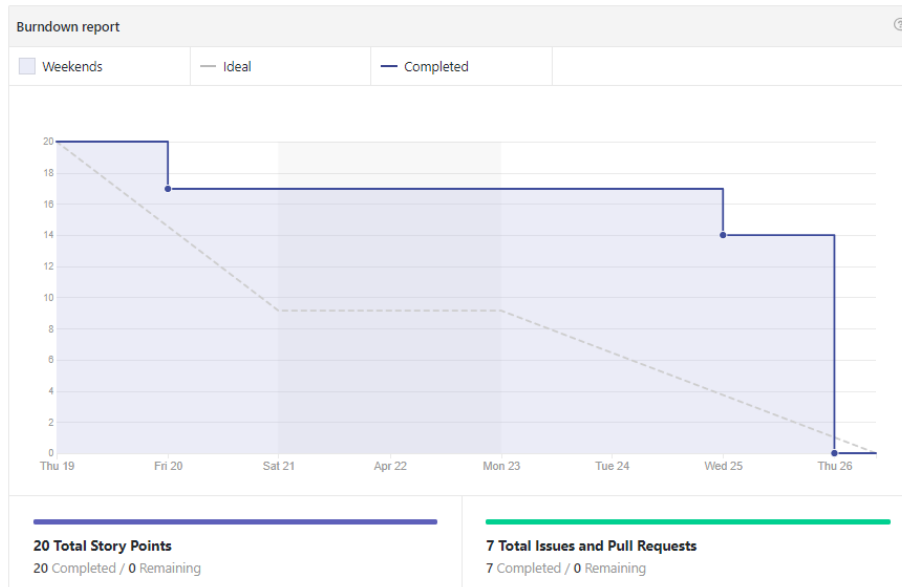
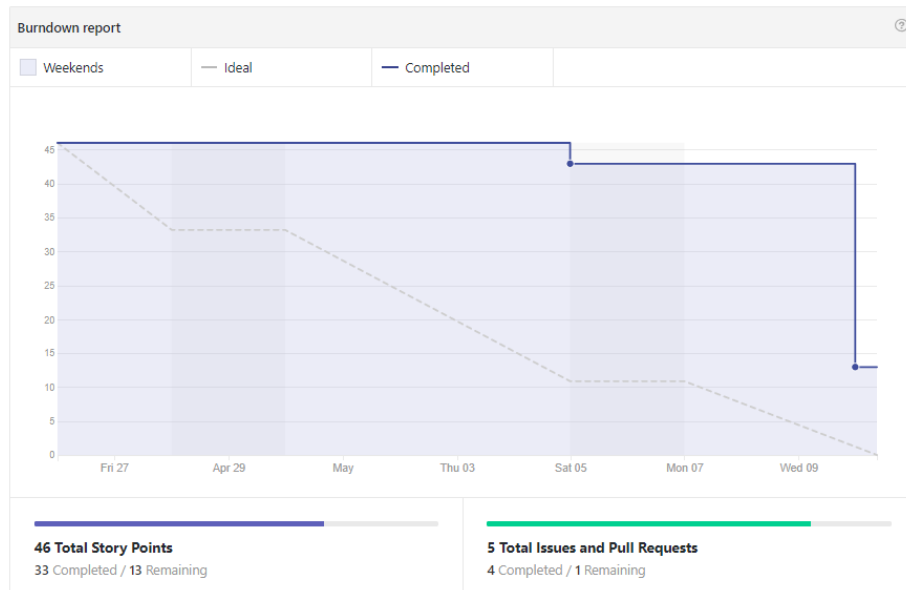


Figura A.8: Progreso en el *sprint* 7.

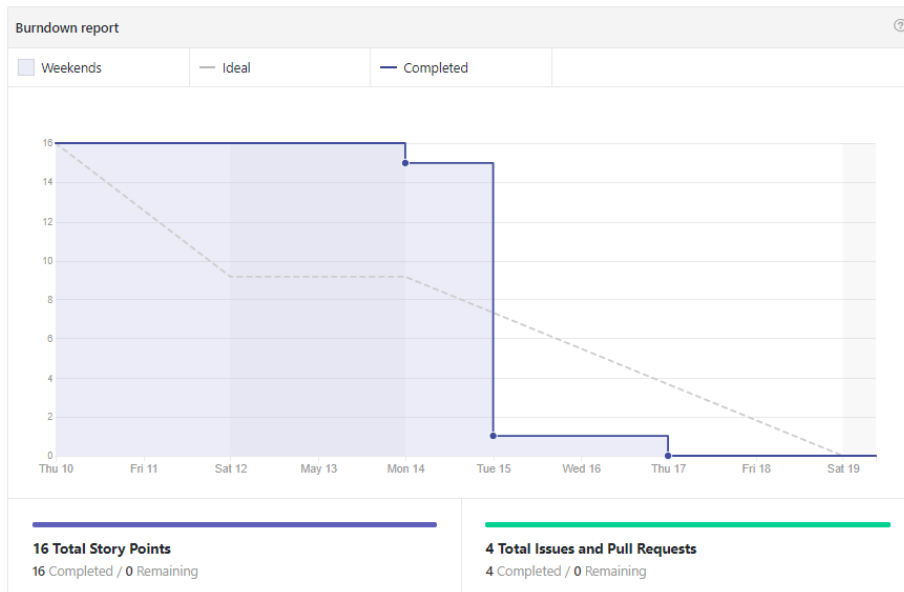
Sprint 8 - Abril 26-2018 - Mayo 10-2018

En este *sprint* se pretende continuar con la realización y/o modificación de nuevos test tanto unitarios como con *Espresso*. A su vez, se pretende seguir con la documentación, pero en un nuevo repositorio creado solo con esa finalidad. Por otro lado, se continuó también con la modificación de la interfaz gráfica del usuario. En este *sprint* se creó la primera **Apk** del proyecto en una *release*. Podemos observar el progreso del *sprint* en la figura A.9

Figura A.9: Progreso en el *sprint* 8.

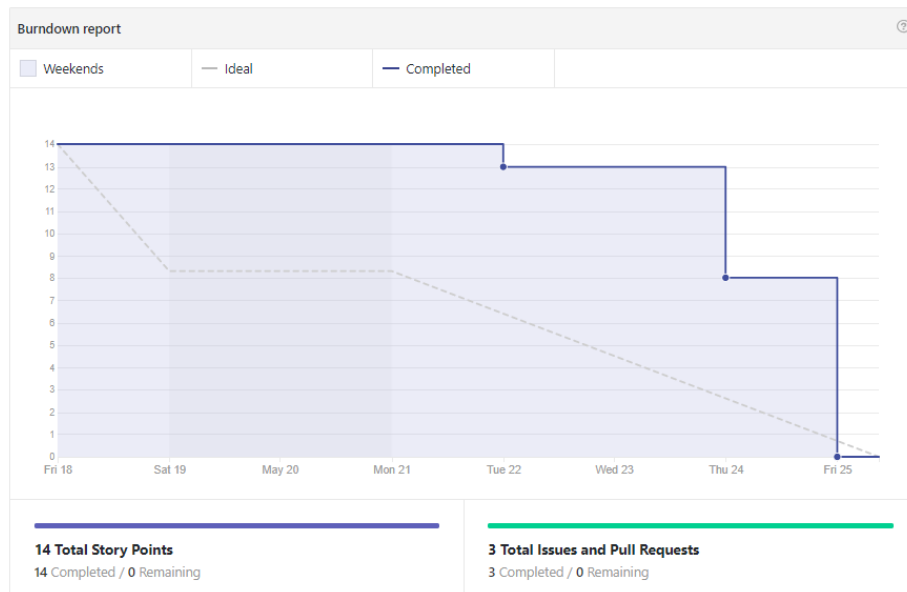
Sprint 9 - Mayo 10-2018 - Mayo 18-2018

En este *sprint* se pretende corregir pequeños detalles que se observaron que fallaban tras instalar la primera **Apk** creada en el *sprint* anterior en un dispositivo móvil real. Tras corregir esos detalles y añadir nuevas modificaciones, se crearon dos nuevas versiones **Apk** para la aplicación. Por otro lado, se corrigieron pequeños fallos en la memoria del proyecto. Podemos observar el progreso del *sprint* en la figura A.10

Figura A.10: Progreso en el *sprint* 9.

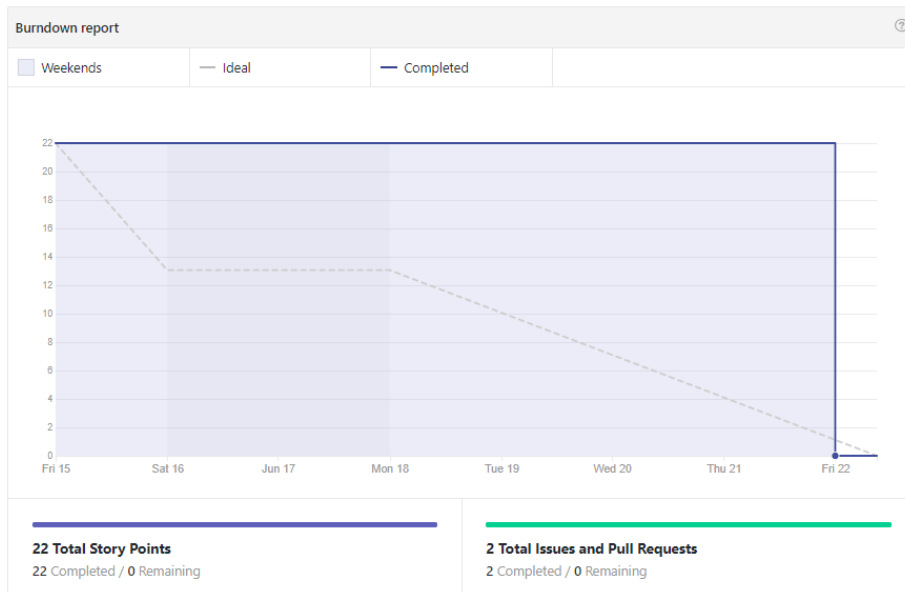
Sprint 10 - Mayo 18-2018 - Mayo 25-2018

En este *sprint* se pretende seguir con la modificación de la interfaz gráfica del usuario. A su vez, se pretende crear una nueva estructura para los test unitarios. Se pretende separar en distintas clases los test que son precisos/exactos, de los que tienen cierto margen de error. Podemos observar el progreso del *sprint* en la figura [A.11](#)

Figura A.11: Progreso en el *sprint* 10.

Sprint 11 - Junio 15-2018 - Junio 22-2018

En este *sprint* se pretende crear una nueva estructura jerárquica de los paquetes java de la aplicación. Así mismo se pretende crear un nuevo diseño de gráficas para consultar el historial de glucemias del usuario. Podemos observar el progreso del *sprint* en la figura [A.12](#)

Figura A.12: Progreso en el *sprint* 11.

Sprint 12 - Junio 22-2018 - Junio 26-2018

En este *sprint* se pretende corregir los últimos detalles de la aplicación y ejecutar la herramienta **SonarLint** de Android Studio en el proyecto final. Podemos observar el progreso del *sprint* en la figura [A.13](#)

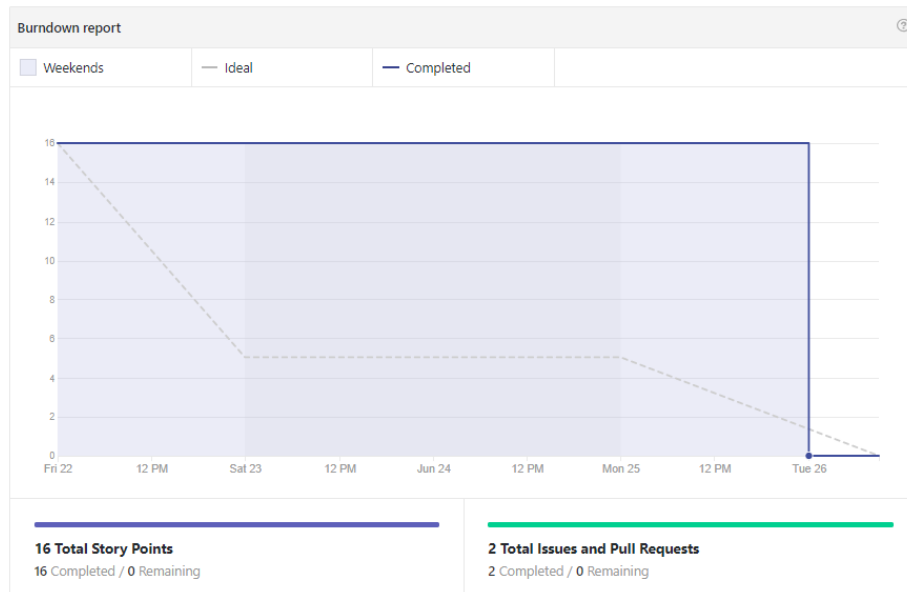
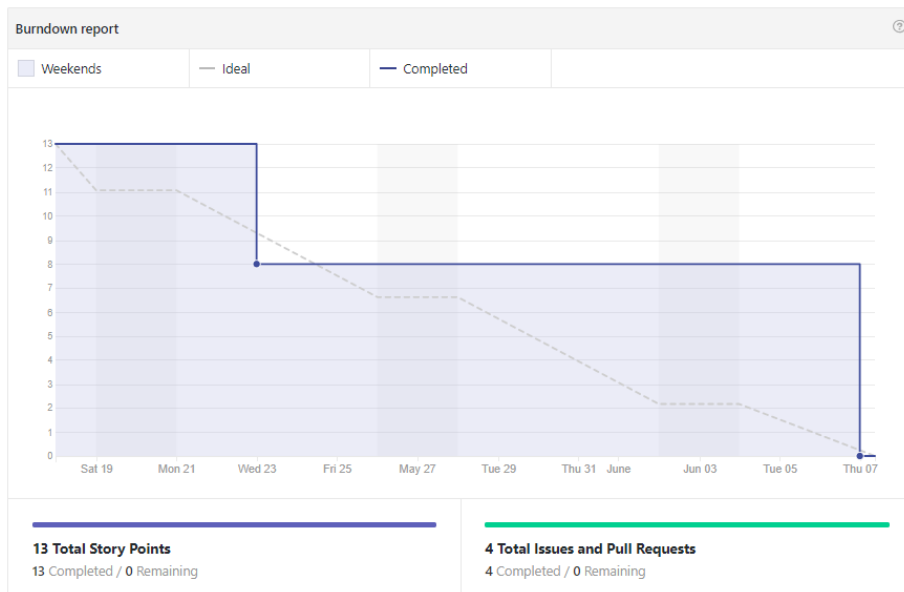


Figura A.13: Progreso en el *sprint* 12.

Los *sprint* que se detallarán a continuación son referentes a la documentación del proyecto, es decir, son creados desde el repositorio creado para la documentación. Las fechas pueden coincidir con alguno de los *sprints* anteriores o pueden estar dentro de las fechas anteriores, su explicación es que se realizaba la documentación y la aplicación en paralelo.

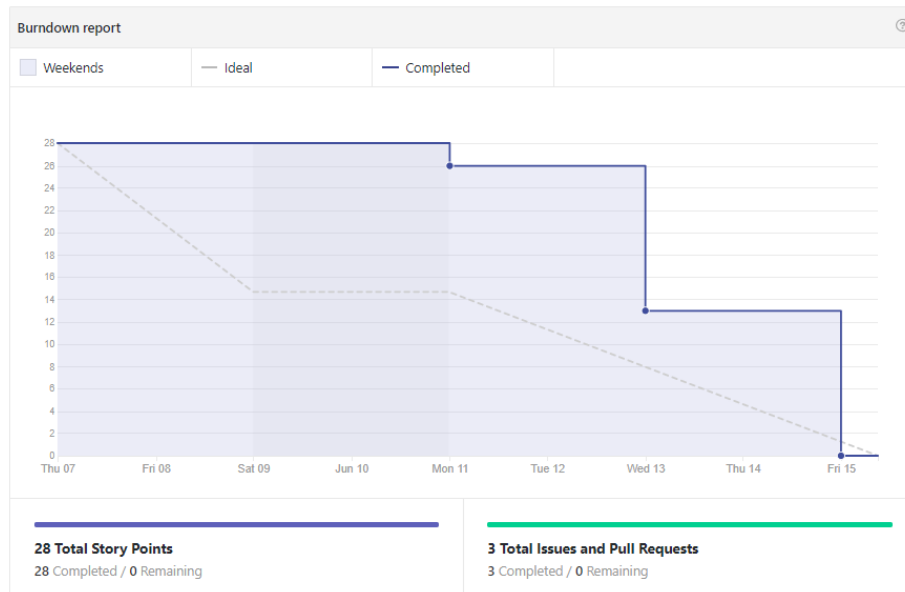
Sprint 13 - Mayo 18-2018 - Junio 07-2018

En este *sprint* se pretende continuar con la documentación del proyecto. Cabe mencionar que no he realizado una dedicación a tiempo completo a este *sprint*, sino que ha sido una dedicación parcial. Esto es debido a que me encuentro cursando dos asignaturas al mismo tiempo, por lo tanto, este *sprint* ha tenido una duración superior a los primeros ya que por estas fechas los alumnos tenemos cerca los exámenes finales. Podemos observar el progreso del *sprint* en la figura [A.14](#)

Figura A.14: Progreso en el *sprint* 13.

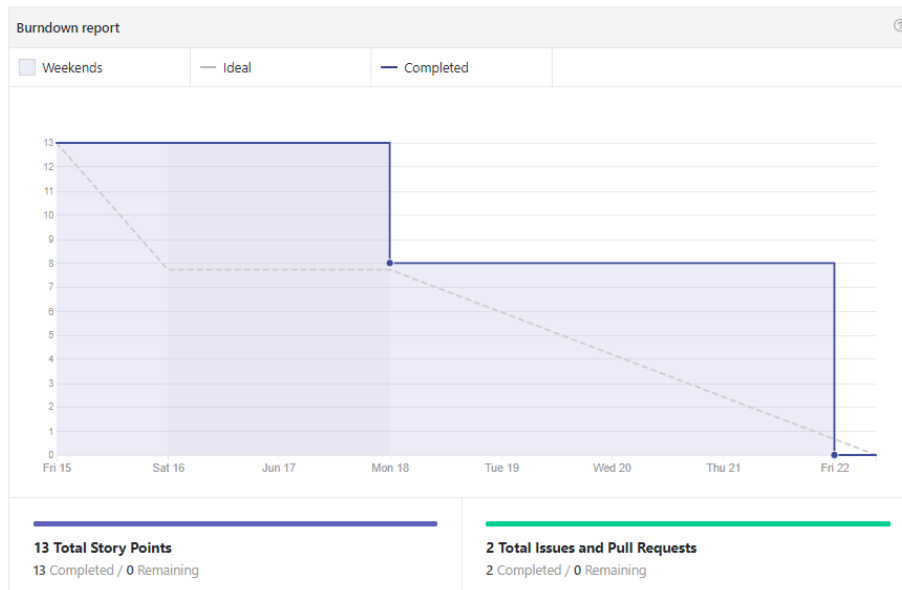
Sprint 14 - Junio 07-2018 - Junio 15-2018

En este *sprint* se pretende corregir los errores de la primera versión parcial corregida por Raúl Marticorena Sánchez. Así mismo se pretende continuar con la documentación. Podemos observar el progreso del *sprint* en la figura [A.15](#)

Figura A.15: Progreso en el *sprint* 14.

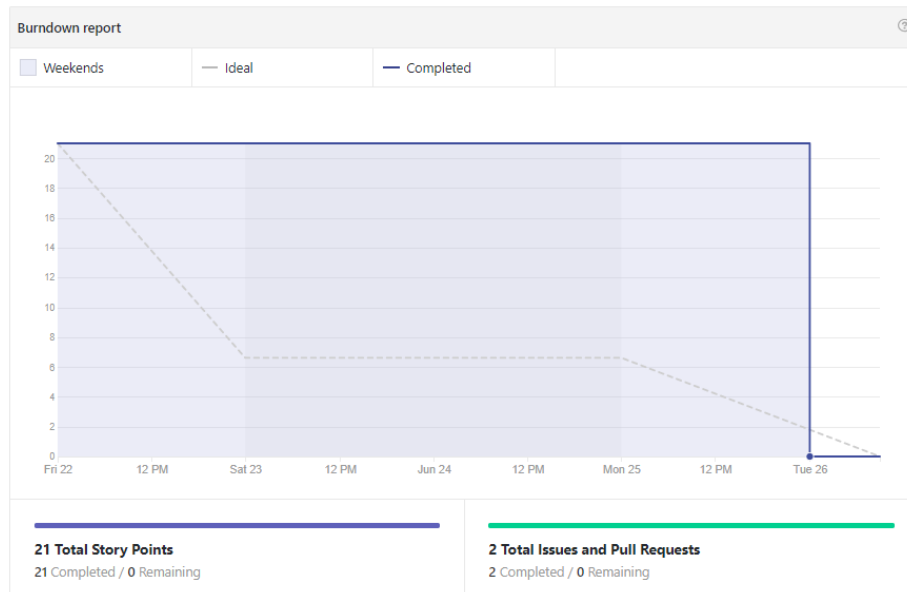
Sprint 15 - Junio 15-2018 - Junio 22-2018

Continuamos con la documentación del proyecto. Podemos observar el progreso del *sprint* en la figura [A.16](#)

Figura A.16: Progreso en el *sprint* 15.

Sprint 16 - Junio 22-2018 - Junio 26-2018

En este *sprint* se pretende finalizar con la documentación del proyecto así como corregir los últimos detalles de la misma. Podemos observar el progreso del *sprint* en la figura [A.17](#)

Figura A.17: Progreso en el *sprint* 16.

A.3. Estudio de viabilidad

En esta sección vamos a realizar un estudio de viabilidad económica de nuestro proyecto y su viabilidad legal, dos aspectos importantes y relevantes.

Viabilidad económica

A continuación, analizaremos la viabilidad económica de nuestro proyecto detallando los costes que hubiera supuesto en un caso real, es decir, en los diferentes niveles de una empresa.

Coste de personal

Los recursos humanos empleados en este proyecto han sido 2: el propio alumno quien realizó el proyecto y el tutor del mismo. Se puede estimar que el alumno ha empleado 309 horas (aun por estimar) a 20€ la hora:

$$20 \text{ €/Hora} * 309 \text{ Horas} = 6180 \text{ €}$$

Para calcular el coste de las reuniones con el tutor se conoce que en total se imparten 32 créditos con un coste anual aproximado por crédito de 600€, por lo que tendríamos:

$$600 \text{ €} * 0,5 \text{ créditos} = 300 \text{ €}$$

A esta cuantía debemos añadirle el valor que debería llevarse la Seguridad Social, lo cual ha sido obtenido a partir de¹:

- 1,55 %: Desempleo.
- 0,03 %: Formación profesional.
- 4,7 %: Contingencias comunes
- Lo que hace un total de: 6,28

Por lo que sería:

$$(6180 + 300) \text{ €} * 0,0628 = 407 \text{ €}$$

Coste de hardware

Cabe mencionar que no se necesitó de ningún equipo informático adicional para la realización del proyecto. No obstante, se deberá calcular el coste del *hardware* utilizado en el proyecto. En este caso al tratarse de una aplicación móvil, únicamente se necesitó un ordenador portátil cuyo valor se estima en 600€ y suponiendo una vida útil de 3-4 años y 15 semanas de proyecto, se tiene un coste de amortización de 43€ aproximadamente.

$$\frac{600\text{€}}{4 \text{ años}} * \frac{1 \text{ año}}{52 \text{ semanas}} * 15 \text{ semanas} \approx 43\text{€}$$

Coste de software

Este proyecto es una segunda versión y se ha desarrollado empleando plataformas *software* libre para todo lo imprescindible, por lo que el coste *software* es cero.

Coste total

Así, el coste total del proyecto ha sido **A.1**:

Beneficios

Para poder obtener ingresos por nuestra aplicación se piensa en sacar una versión comercializable, es decir, convertirla en una aplicación de pago(sin incluir publicidad de terceros).

¹http://www.seg-social.es/Internet_1/Trabajadores/CotizacionRecaudaci10777/Basesytiposdecotiza36537/index.htm

Tabla A.1: Costes totales

Costes	Importe
Personal	6480 €
Seguridad Social	407 €
Material	43 €
Totales	6930 €

Al tratarse de una aplicación móvil para el sistema operativo Android, la fuente principal de descarga sería la tienda oficial **Google Play Store** para que de esta manera esté disponible para su compra. Antes de publicar nuestra aplicación se debe tener en cuenta que debemos tener una cuenta de desarrollador en Google. Esto se puede conseguir a través de **Google Play Developer Console**.

Una vez se haya accedido a la consola de desarrollador de Google Play, se deberá proceder al pago de la tasa que Google pone a todos los desarrolladores para que puedan incluir sus apps en la tienda. Una vez finalizado todos los pagos, ya se puede gestionar nuestra aplicación desde el **centro de gestión e información como desarrolladores**, que nos permitirá añadir nuestra aplicación entre otras opciones. Una vez completada toda la información necesaria para poder publicar nuestra aplicación, habiendo marcado que nuestra aplicación será de pago podremos vender nuestra aplicación a través de **Google Play Store**.

Antes de poner un precio a nuestra aplicación debemos tener en cuenta lo siguiente:

- Google Play [5].
 - Google añade una **Comisión de Transacción** al precio de venta de cada producto. Esta Comisión de Transacción, Google la fija en un 30 % del precio de la aplicación, recibiendo de esta manera un 70 % el desarrollador y el 30 % restante se destina al *partner* de distribución.
 - El precio por darse de alta como desarrollador en Google ronda los 20€.
 - El propio desarrollador es quien pone el precio de la app a partir de 0.50€.
- Diabetes en España [4].

- Según datos recogidos por la **Fundación para la diabetes**, el 13.8 % de la población Española mayores de 18 años tienen Diabetes. Dentro de este porcentaje, las personas con Diabetes tipo 1 supone entre el 5 % y el 1 % del total. Podemos observar estos datos en la siguiente imagen: **A.18**:

DATOS GLOBALES DE PREVALENCIA (%) ESTUDIO di@bet.es		
	% Personas afectadas mayores de 18 años	Núm. de personas afectadas mayores de 18 años
Diabetes Mellitus (DM) Total	13.8	5.301.314
DM Conocida	7.8	2.996.395
DM no conocida	6.0	2.304.919
Tolerancia Anormal de la Glucemia	9.2	3.534.210
Glucemia Basal alterada	3.4	1.306.121
Obesidad (IMC > 30 kg/m ²)	28.2	10.863.431

Figura A.18: Datos Globales de Prevalencia (%)

Por lo tanto, teniendo en cuenta los datos anteriores, y con una previsión de ventas para el **primer año**, se decide:

- Lugar de distribución: España.
- Medio de distribución: Google Play Store.
- Precio de venta: **0.99€**.
- Numero de descargas: 1 % de 5301314 personas con diabetes.

Por lo tanto, el primer año obtendremos unos ingresos de

▪ **Caso A:**

- **Retribución total:**

$$(0,01 * 5301314) * 0,99 \text{ €} = 52483,0086 \text{ €}$$

- **Retribución desarrollador:** 70 % de 52483.0086€ = 36738.10602€.
- **Retribución *partner* de distribución:** 30 % de 52483.0086€ = 15744.903€.

- **Caso B:** Nuestro competidor directo y una de las aplicaciones mejores valoradas y que se asemeja a nuestra aplicación es: **Social Diabetes**. Esta aplicación cuenta con aproximadamente 100Mil descargas en **Google Play Store** y su primera versión Android apareció en 2012. Si estimamos su promedio de ventas en [1]

$$\frac{100000 \text{ descargas}}{6 \text{ años}} \approx 16667$$

Observamos que SocialDiabetes tiene un promedio de 16667 descargas anuales. Si estimamos el mismo número de descargas para el primer año, nuestra aplicación obtendrá:

- **Retribución total:**

$$16667 * 0,99 \text{ €} = 16500,33 \text{ €}$$

- **Retribución desarrollador:** 70 % de 16500.33€ = 11550.231€.
- **Retribución *partner* de distribución:** 30 % de 16500.33€ = 4950.099€.

Así, si el coste total de nuestro proyecto ha sido **6930€**, los **beneficios** obtenidos serían:

- Beneficio según caso A:

- Coste Proyecto:

$$6930 \text{ €} + 20 \text{ €}(\text{GooglePlay}) = 6950 \text{ €}$$

- Retribución Proyecto: 36738.10602€.
- Beneficio:

$$36738,10602 \text{ €} - 6950 \text{ €} = 29788,106 \text{ €}$$

- Beneficio según caso B:

- Coste Proyecto:

$$6930 \text{ €} + 20 \text{ €}(\text{GooglePlay}) = 6950 \text{ €}$$

- Retribución Proyecto: 11550.231€.
- Beneficio:

$$11550,231 \text{ €} - 6950 \text{ €} = 4600,23 \text{ €}$$

Viabilidad legal

En este apartado pasamos a analizar la viabilidad legal del proyecto. Para ello vamos a describir las licencias de elementos o librerías que se hayan podido utilizar.

El primer caso sujeto a estudio es el de la librería **Hitomis/CircleMenu v1.0.2**. Esta librería se distribuye bajo la licencia Apache License v2.0.

El segundo caso sujeto a estudio es el de la librería (**MPAndroidChart v3.0.3**). Esta librería se distribuye bajo la licencia Apache License v2.0.

Apache License V2.0

La licencia *Apache* (Apache License o Apache Software License para versiones anteriores a 2.0) es una licencia de software libre permisiva creada por la *Apache Software Foundation (ASF)*. La licencia *Apache* (con versiones 1.0, 1.1 y 2.0) requiere la conservación del aviso de derecho de autor y el descargo de responsabilidad, pero no es una licencia copyleft, ya que no requiere la redistribución del código fuente cuando se distribuyen versiones modificadas. Al igual que otras licencias de software libre, todo el software producido por la *ASF* o cualquiera de sus proyectos está desarrollado bajo los términos de esta licencia, es decir, la licencia permite al usuario del software de la libertad de usar el software para cualquier propósito, para distribuirlo, modificarlo y distribuir versiones modificadas del software, bajo los términos de la licencia, sin preocuparse de las regalías [6].

Apéndice B

Especificación de Requisitos

B.1. Introducción

En esta sección se expondrán los distintos objetivos que la aplicación debe lograr, así como los requisitos que debe cumplir.

B.2. Objetivos generales

Como objetivos generales en este proyecto tenemos:

- Crear y/o mejorar la interfaz gráfica del usuario.
- Mostrar todos los registros diarios del usuario.
- Facilitar la navegabilidad durante el uso de la aplicación.
- Corregir los métodos utilizados para el cálculo del bolo corrector.
- Mejorar la robustez de la aplicación.
- Incluir pruebas unitarias e instrumentales.

B.3. Catálogo de requisitos

A continuación, listaremos el conjunto de requisitos funcionales extraídos a partir de los objetivos generales del proyecto.

Requisitos funcionales

Los requisitos funcionales se han obtenido a partir de un diagrama de casos de uso hecho desde cero, aunque ciertos requisitos coinciden con los de la versión anterior, por lo que serán iguales.

- **RF-1 Gestión del perfil de usuario:** la aplicación debe ser capaz de registrar y modificar los datos personales y médicos del usuario.
 - **RF-1.1 Comprobar datos:** El sistema deberá comprobar que se han rellenado todos los campos de registro.
 - **RF-1.2 Validar valores de Glucemia:** El sistema debe comprobar que se han introducido los valores de glucemia máximo y mínimo dentro de unos márgenes establecidos.
 - **RF-1.3 Registrar datos del perfil:** El sistema deberá registrar los datos del perfil del usuario.
 - **RF-1.4 Modificar datos del perfil:** El sistema deberá permitir la modificación de los datos del perfil del usuario.
- **RF-2 Cálculo del bolo corrector:** El usuario podrá calcular las unidades de insulina que necesita a partir de su nivel de glucemia.
 - **RF-2.1':** El sistema deberá registrar el nivel de glucemia haciendo uso de *Gestión de glucemias*.
 - **RF-2.2 Cantidad de alimento:** El sistema deberá permitir al usuario introducir y/o eliminar los gramos de uno o más alimentos a consumir. Así mismo, el sistema deberá registrar dichos alimentos con sus respectivas cantidades.
 - **RF-2.3 Bolo corrector:** El sistema deberá mostrar por pantalla el resultado del bolo corrector obtenido a partir de los datos introducidos.
- **RF-3 Gestión de glucemias:** El usuario podrá llevar un registro de sus niveles de glucemia en diferentes etapas del día, así como registrar incidencias en los casos en que corresponda.
 - **RF-3.1 Registrar niveles de glucemia:** El sistema deberá registrar en una base de datos local los valores de glucemia introducidos por el usuario.
 - **RF-3.2 Comprobar límites:** El sistema deberá comprobar si el valor introducido se encuentra dentro de los límites mínimo-máximo introducidos por el usuario en su perfil.

- **RF-3.2.1 Registrar incidencia:** Si el valor introducido está fuera de los límites establecidos, el sistema permitirá al usuario registrar una incidencia que podrá tener opcionalmente una observación por parte del usuario.
- **RF-4 Gestión del historial:** El usuario podrá comprobar de manera gráfica sus niveles de glucemia registrados.
- **RF-5 Gestión de registros diarios:** El usuario podrá comprobar todos sus registros de manera simplificada o detallada.
 - **RF-5.1 Resumen registros:** El sistema deberá mostrar por pantalla los registros diarios del usuario:
 - Id: Identificador del registro.
 - Fecha: Momento del registro.
 - HC: Sumatorio de hidratos de carbono en el cálculo del bolo corrector.
 - Bolo corrector: Unidades de insulina obtenidas en el cálculo del bolo corrector.
 - **RF-5.2 Detalle registros:** El sistema deberá mostrar por pantalla los detalles de un registro previamente seleccionado por el usuario:
 - Tipo: Categoría del alimento.
 - Alimento: Nombre del alimento consumido.
 - Cantidad: Gramos consumidos del alimento.
 - IG: Índice glucémico.
- **RF-6 Gestión de ajustes:** El usuario podrá realizar pequeños ajustes en la aplicación.
 - **RF-6.1:** El sistema deberá permitir configurar los datos del perfil del usuario haciendo uso de *Gestión del perfil de usuario*.
 - **RF-6.2 Copia de seguridad:** El sistema deberá permitir realizar copias de seguridad de los datos registrados por el usuario.
 - **RF-6.3 Liberar espacio:** El sistema deberá permitir eliminar los datos registrados por el usuario.

Requisitos no funcionales

Alguno de estos requisitos coinciden con los de la versión anterior. Aun así, mencionaremos todos para que quede más claro.

- **RNF-1 Apariencia:** La representación de los datos deberá ser atractiva para el usuario.
- **RNF-2 Usabilidad:** El conjunto de elementos visuales de la interfaz deben ser claros e intuitivos, permitiendo un rápido aprendizaje.
- **RNF-3 Mantenibilidad, Portabilidad y Escalabilidad:** La aplicación debe desarrollarse siguiendo alguna técnica que permita facilidad de mantenimiento e incorporación de nuevas características, así como corrección de errores.
- **RNF-4 Eficiencia:** La aplicación debe ser capaz operar adecuadamente con los datos introducidos por el usuario.
- **RNF-5 Seguridad:** La aplicación debe ser capaz de realizar copias de seguridad de los datos registrados por el usuario.
- **RNF-6 Rendimiento:** La aplicación debe ser capaz de soportar el manejo de gran cantidad de datos durante su proceso.
- **RNF-7 Testabilidad:** Capacidad de que la aplicación puede ser probada.

B.4. Especificación de requisitos

Actores

La aplicación **UBU**Diabetes está orientada a usuarios diabéticos del tipo I. En este caso los clientes de la aplicación son miembros del Área de enfermería (Grado de Enfermería) de la facultad de Ciencias de la Salud, como Diego Serrano Gómez y Jesús Puente Alcaraz, y Raúl Marticorena Sánchez, tutor del proyecto. Por lo tanto:

- **Desarrollador:** Luis Miguel Inapanta Oyana.
- **Clientes:** Diego Serrano Gómez, Jesús Puente Alcaraz, y Raúl Marticorena Sánchez.

Diagrama de casos de uso

A continuación se mostrara el diagrama de casos de uso del proyecto.

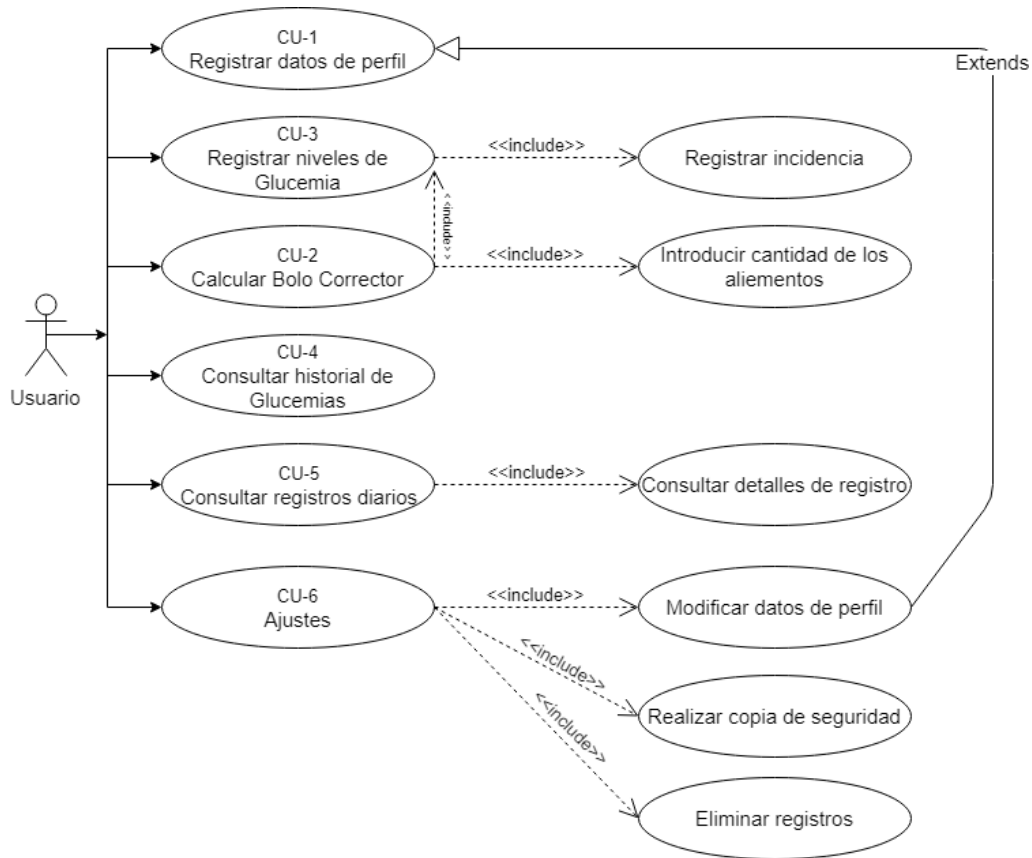


Figura B.1: Diagrama general de casos de uso

CU-1	Registrar datos de perfil
Versión	2.0
Autor	Luis Miguel Inapanta Oyana
Requisitos asociados	RF-1.1, RF-1.2, RF-1.3, RF-1.4
Descripción	Permite al usuario registrar sus datos, tanto personales como médicos.
Precondiciones	<ul style="list-style-type: none"> • Consultar los datos relacionados con la diabetes con su médico.
Acciones	<ol style="list-style-type: none"> 1. El usuario rellena todos los campos del registro. 2. Se cargan los datos de la tabla de alimentos de la federación española de la diabetes. 3. Se visualiza el menú principal.
Postcondiciones	Visualizar menú principal.
Excepciones	No rellenar algún campo del registro
Importancia	Muy Alta

Tabla B.1: CU-1 Registrar datos de perfil

CU-2	Registrar niveles de glucemia
Versión	2.0
Autor	Luis Miguel Inapanta Oyana
Requisitos asociados	RF-3.1, RF-3.2, RF-3.2.1
Descripción	Permite al usuario registrar sus niveles de glucemia en diferentes etapas del día, así como registrar incidencias en los casos en que corresponda.
Precondiciones	<ul style="list-style-type: none"> • Estar registrado. • Seleccionar la opción Registrar glucemia del menú principal - icono de lápiz.
Acciones	<ol style="list-style-type: none"> 1. El usuario introduce su nivel de glucemia. 2. El usuario introduce una incidencia si corresponde.
Postcondiciones	Se registran ambos datos en la Base de Datos.
Excepciones	No introducir nivel de glucemia o incidencia
Importancia	Alta

Tabla B.2: CU-2 Registrar niveles de glucemia

CU-3	Calcular Bolo Corrector
Versión	2.0
Autor	Luis Miguel Inapanta Oyana
Requisitos asociados	RF-2.1, RF-2.2, RF-2.3
Descripción	Permite al usuario calcular las unidades de insulina que necesita a partir de su nivel de glucemia.
Precondiciones	<ul style="list-style-type: none"> • Estar registrado. • Seleccionar la opción Calcular Bolo del menú principal - icono de calculadora.
Acciones	<ol style="list-style-type: none"> 1. El usuario introduce su nivel de glucemia. 2. El usuario introduce una incidencia si corresponde. 3. El usuario introduce introduce o elimina uno o varios alimentos a consumir con sus respectivas cantidades (gramos).
Postcondiciones	<ul style="list-style-type: none"> • Se muestra la usuario el bolo corrector calculado por el sistema. • Se registran los alimentos con sus respectivas cantidades, el sumatorio de los hidratos de carbono consumidos y el bolo corrector calculado en la Base de Datos.
Excepciones	<ol style="list-style-type: none"> 1. No introducir nivel de glucemia o incidencia. 2. No introducir cantidad del alimento.
Importancia	Muy Alta

Tabla B.3: CU-3 Calcular Bolo Corrector

CU-4	Consultar historial de glucemias
Versión	2.0
Autor	Luis Miguel Inapanta Oyana
Requisitos asociados	RF-4
Descripción	Permite al usuario consultar sus niveles de glucemia de manera gráfica.
Precondiciones	<ul style="list-style-type: none"> • Estar registrado. • Seleccionar la opción Historial de glucemias del menú principal - icono de gráfica.
Acciones	1. pendiente aún.
Postcondiciones	• —Pendiente aún.
Excepciones	1. pendiente aún.
Importancia	Media

Tabla B.4: CU-4 Consultar historial de glucemias

CU-5	Consultar registros diarios
Versión	2.0
Autor	Luis Miguel Inapanta Oyana
Requisitos asociados	RF-5.1, RF-5.2
Descripción	Permite al usuario consultar sus registros de manera simplificada o detallada
Precondiciones	<ul style="list-style-type: none"> • Estar registrado. • Seleccionar la opción Consultar registros del menú principal - icono de lupa.
Acciones	<ol style="list-style-type: none"> 1. Seleccionar un <i>item</i> de la lista de registros. 2. Pulsar el botón consultar.
Postcondiciones	• Se muestran los detalles del registro
Excepciones	1. No seleccionar ningún <i>item</i> de la lista.
Importancia	Alta

Tabla B.5: CU-5 Consultar registros diarios

CU-6	Ajustes
Versión	2.0
Autor	Luis Miguel Inapanta Oyana
Requisitos asociados	RF-6.1, RF-6.2, RF-6.3
Descripción	Permite al usuario realizar pequeños ajustes en la aplicación.
Precondiciones	<ul style="list-style-type: none"> • Estar registrado. • Seleccionar la opción Ajustes del sub-menú del menú principal.
Acciones	1. Seleccionar un <i>ítem</i> de la lista de ajustes.
Postcondiciones	<ul style="list-style-type: none"> • Permite al usuario modificar sus datos de perfil. • Permite al usuario realizar una copia de seguridad de sus registros. • Permite al usuario eliminar sus registros.
Excepciones	<ol style="list-style-type: none"> 1. No seleccionar ningún <i>ítem</i> de la lista. 2. Intentar eliminar registros.
Importancia	Media

Tabla B.6: CU-6 Ajustes

Apéndice C

Especificación de diseño

C.1. Introducción

A continuación, especificaremos la manera en la que hemos organizado cada uno de los elementos del proyecto y detallaremos las razones de por qué lo hemos hecho así. Algunos datos son similares a los del TFG de Mario López Jiménez [2].

C.2. Diseño de datos

Gran parte de este apartado coincide con la primera versión de nuestro proyecto. Aún así, mostraremos todos los datos.

Tabla C.1: Base de datos. Glucemias

Campo	Tipo	Descripción
Id	Integer Primary Key Autoincrement	Clave primaria.
Fecha	Text Not Null	Fecha de registro.
Periodo	Text Not Null	Periodo del registro del valor.
Valor	Integer Not Null	Valor de Glucemia.

Tabla C.2: Base de datos. Incidencias

Campo	Tipo	Descripción
Id	Integer Primary Key Autoincrement	Clave primaria.
Tipo	Text Not Null	Tipo de incidencia.
Fecha	Text Not Null	Fecha de la incidencia.
Observaciones	Text Not Null	Observaciones asociadas a la incidencia.
Glucemia	Integer Not Null	Clave primaria de la glucemia asociada.

Tabla C.3: Base de datos. Alimentos

Campo	Tipo	Descripción
Tipo Alimento	Tex Primary Key Autoincrement	Tipo de alimento.
Alimento	Integer Primary Key	Nombre del alimento.
Ración	Integer Not Null	Ración media del alimento.
Índice glucémico	Integer Not Null	Índice glucémico del alimento.

Tabla C.4: Base de datos. Registros diarios

Campo	Tipo	Descripción
Id	Integer Primary Key Autoincrement	Clave primaria de la tabla.
Fecha	Text Not Null	Fecha de registro.
Sum_HC	Real Not Null	Total Hidratos de carbono registrado.
Bolo_Corrector	Real Not Null	Bolo Corrector registrado.

Tabla C.5: Base de datos. Detalles Registros diarios

Campo	Tipo	Descripción
Id	Integer Primary Key Autoincrement	Clave primaria de la tabla.
Id_lista	Integer Not Null	Clave primaria del registro asociado.
Alimento	Text Not Null	Alimento registrado.
Cantidad	Real Not Null	Cantidad en gramos del alimento registrado.

Por último, aunque no se trata de una tabla de la base de datos, mencionaremos otro de los métodos utilizados para administrar los datos. En este caso estamos hablando de la interfaz *SharedPreferences* que proporciona Android para acceder y modificar los datos de preferencia del usuario. Este objeto constará de la siguiente colección de datos:

Tabla C.6: SharedPreferences. PreferenciasUsuario

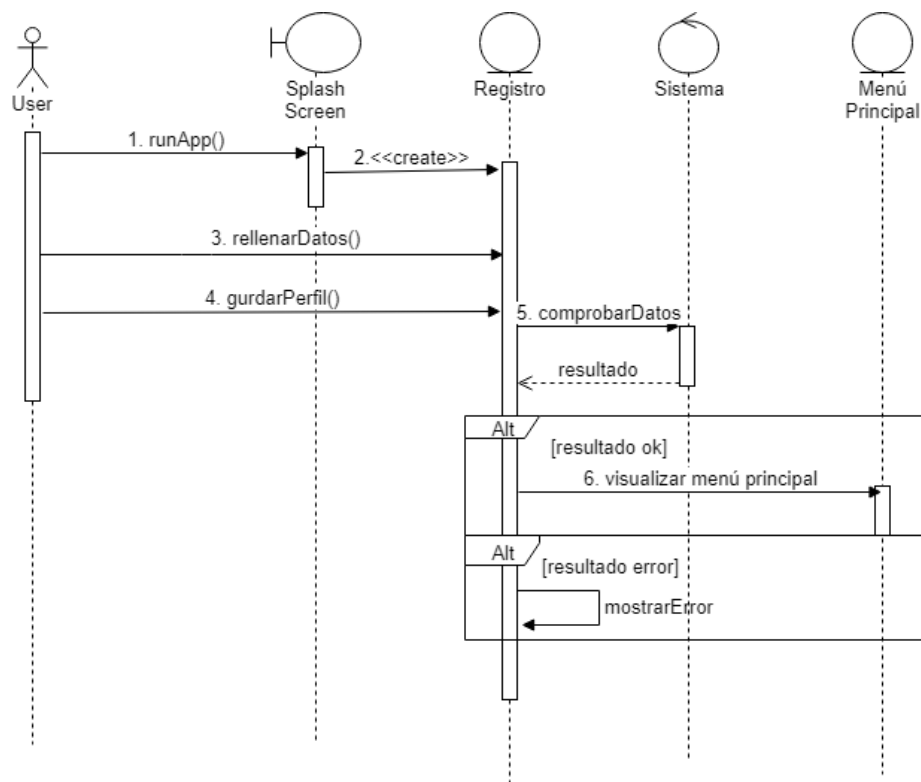
Campo	Tipo	Descripción
primerejecución	Boolean	Si la aplicación se ejecuta por primera vez.
nombre	String	Nombre del usuario.
edad	String	Edad del usuario.
estatura	String	Estatura del usuario.
peso	String	Peso del usuario.
min	String	Valor de glucemia mínimo deseado.
max	String	Valor de glucemia máximo deseado.
udsBasal	String	Uds de insulina basal.
udsRapida	String	Uds de insulina rápida.
rapida	Boolean	Insulina del bolo.
ultrarrapida	Boolean	Insulina del bolo.
Cero	Boolean	Cero decimales en el bolo corrector.
Uno	Boolean	Un decimal en el bolo corrector.
Dos	Boolean	Dos decimales en el bolo corrector.
Imagen	String	Path de la imagen de perfil del usuario.

C.3. Diseño procedimental

En el momento en el que el usuario ejecuta la aplicación, el sistema lanzará la pantalla de bienvenida (Splash Screen) y comprobará si es la primera vez que se ejecuta la aplicación. Si es así, se mostrará la pantalla de registro para que el usuario pueda registrarse en la aplicación, de lo contrario se mostrará la pantalla de Menú Principal. Desde esta pantalla el usuario podrá realizar cualquier acción que desee y que se encuentre en la pantalla Menú Principal.

A continuación, con los diagramas de secuencias, explicaremos el funcionamiento interno del sistema.

En la figura C.1 se muestra como sucede el proceso de registro.

Figura C.1: Proceso de *registro*

Si el proceso que queremos realizar es registrar un nivel de glucemia, se seguirá el diagrama de la figura [C.2](#).

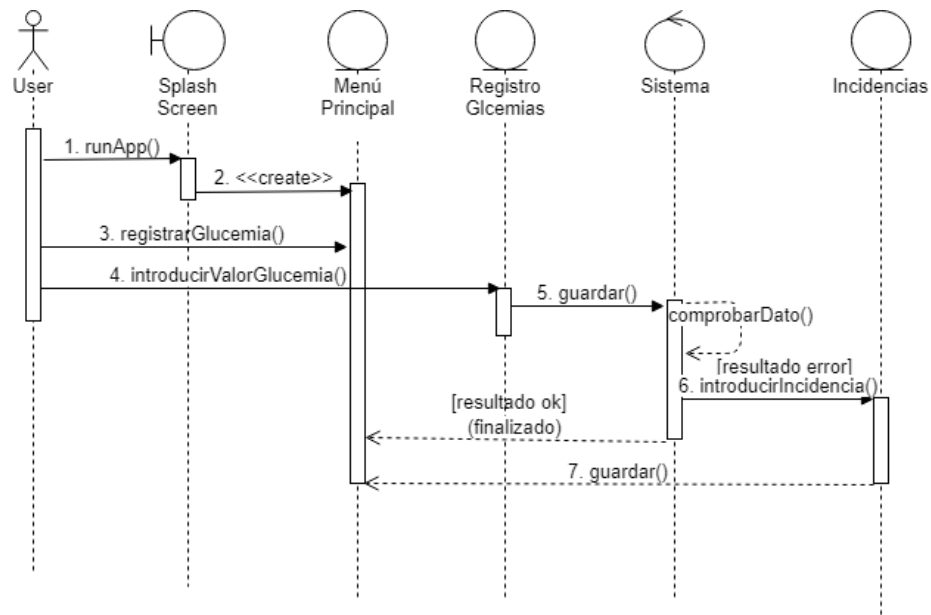


Figura C.2: Registro de un nivel de glucemia

Si el proceso que queremos realizar es calcular el bolo corrector, se seguirá el diagrama de la figura C.3.

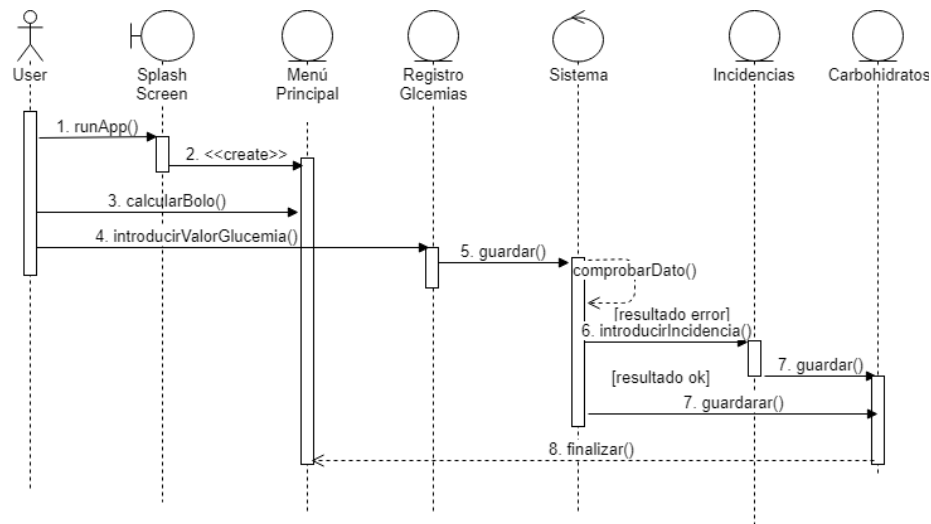


Figura C.3: Calcular bolo corrector

Si el proceso que queremos realizar es consultar el historial de glucemias, se seguirá el diagrama de la figura C.4.

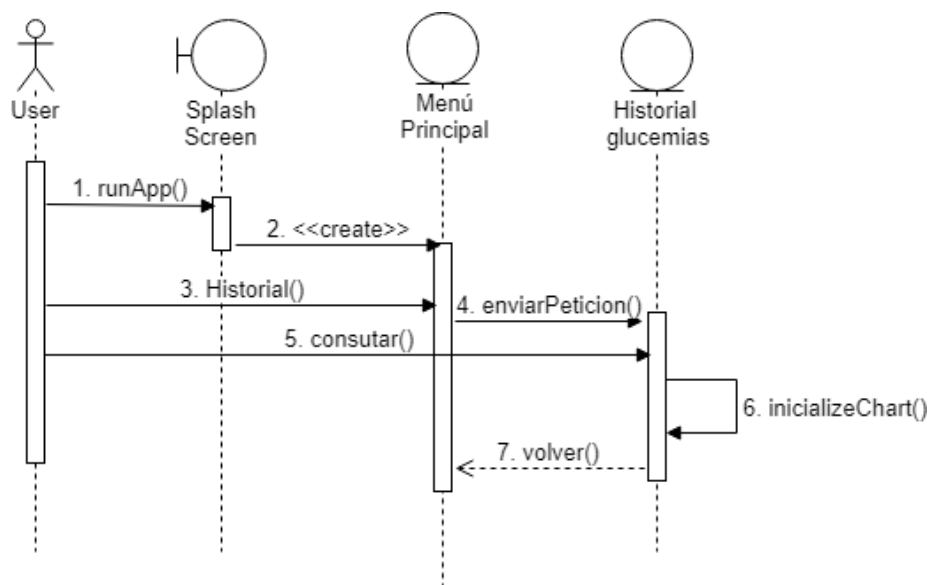


Figura C.4: Consultar historial de glucemias

Si el proceso que queremos realizar es consultar los registros (listas de ingestas), se seguirá el diagrama de la figura [C.5](#).

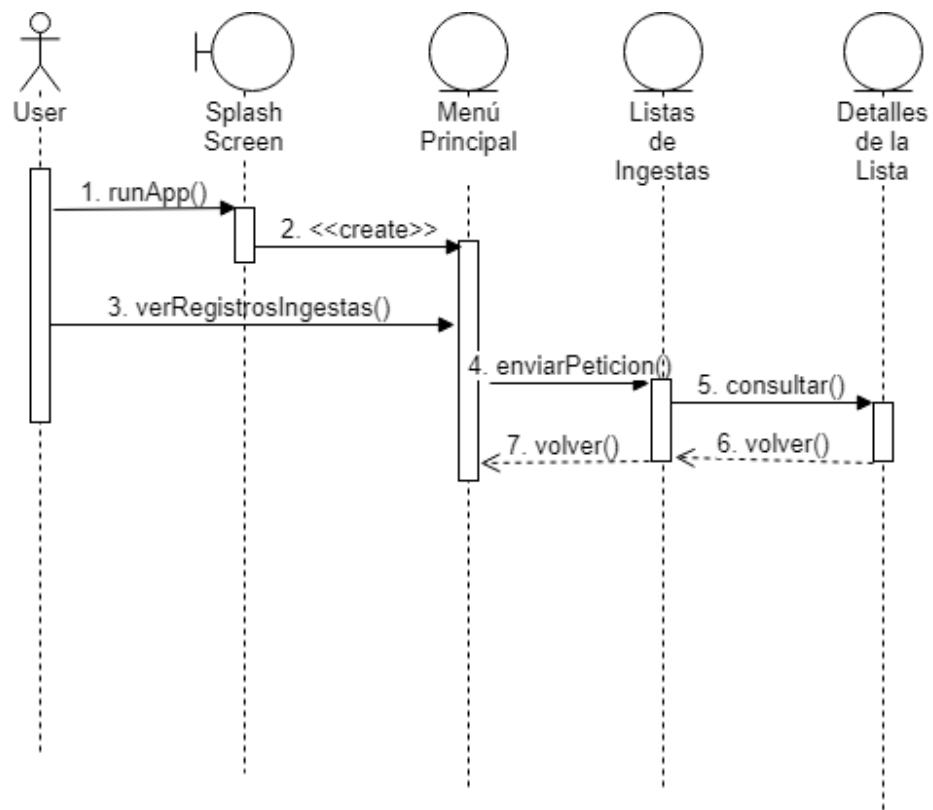


Figura C.5: Consultar registros

Finalmente, si el proceso que queremos realizar es Ajustes de la aplicación, se seguirá el diagrama de la figura [C.6](#).

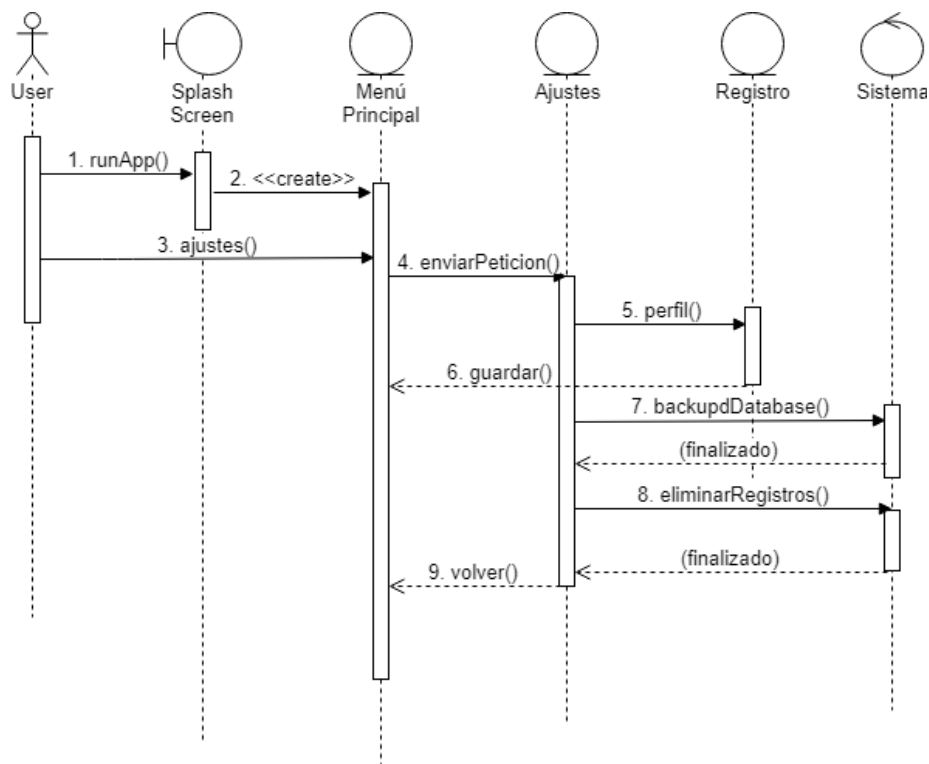


Figura C.6: Ajustes

C.4. Diseño arquitectónico

Diseño de paquetes

Podemos observar en la figura [C.7](#) que la aplicación está estructurada principalmente en 3 paquetes. Estos paquetes o estructura de paquetes la crea Android Studio automáticamente al crear un nuevo proyecto.

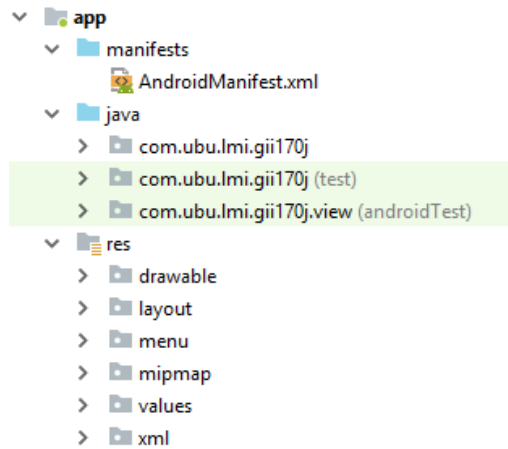


Figura C.7: Ajustes

En este apartado hablaremos de los paquetes en los que hemos modificado y/o añadido nuevos elementos: *java* y *res*.

java

Este paquete se crea por defecto y en él se generan los 3 paquetes de mayor importancia en la aplicación:

- *com.ubu.lmi.gii170j*: Este paquete ha sido definido para separar las *clases* y *activitys* de la aplicación según su jerarquía en ella. La estructura dentro de este paquete ha sido redefinida por completo, es decir, su estructura es diferente a la de la versión anterior. Su nueva estructura es la siguiente C.8:
 - *calculations*: Este paquete contiene únicamente la clase que se encarga de realizar el proceso necesario para el cálculo del bolo corrector de insulina y devolver su resultado.
 - *model*: Este paquete es nuevo y contiene las clases necesarias que se utilizan en los *ArrayAdapter* utilizados en cada una de las listas personalizadas que se visualizan en la aplicación.
 - *persistence*: Este paquete mantiene la misma estructura de la versión anterior y en él se encuentran las clases que implementan todo lo necesario para mantener la persistencia de la aplicación.
 - *util*: En este paquete se encuentran las clases que implementan los *ArrayAdapter* personalizados para la visualización de las distintas listas en la aplicación.

- *view*: Este paquete es similar al utilizado en la versión anterior: **interfaz**. En el se encuentran únicamente las *activities* y los *fragments* que dan funcionalidad a las diferentes pantallas de la aplicación.
- *com.ubu.lmi.gii170j(test)*: Este paquete ha sido definido para separar los *test unitarios* de la aplicación. La estructura de este paquete es nueva C.9:
 - *calculations*: Este paquete contiene únicamente las clases que realizan los test unitarios para los casos clínicos reales aportados por Diego Serrano Gómez.
 - *view*: Este paquete contiene la clase que testea la *activity* **Carbohidratos**. Únicamente testea que esta actividad realiza correctamente los cálculos de gramos de hidratos de carbono. Para estos test, se utilizan los datos que se recogen en el documento pdf “Indicaciones para la corrección de UBUDiabetes 1.0”. Este documento fue proporcionado por Raúl Marticorena Sánchez
- *com.ubu.lmi.gii170j(androidTest)*: Este paquete ha sido definido para separar los *test instrumentales* (*Test en Espresso*) de la aplicación. La estructura de este paquete es nueva C.10:
 - *view*: Este paquete contiene las clases que realizan los test para la interfaz de usuario. Estos tests se realizaron con el *framework* Espresso.

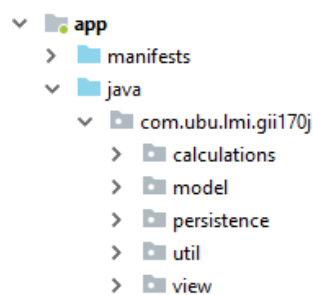


Figura C.8: Paquetes Java 1

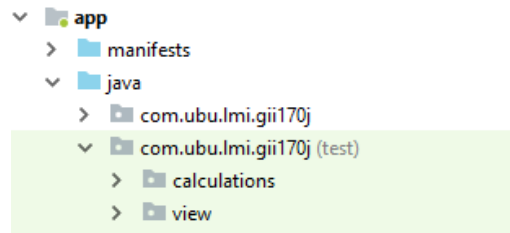


Figura C.9: Paquetes Java 2

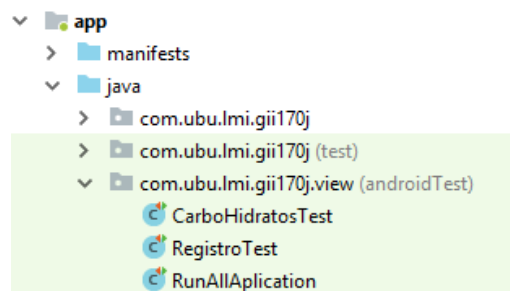


Figura C.10: Paquetes Java 3

res

Este paquete contiene los recursos utilizados por la aplicación.

- *drawable*: Este paquete contiene los archivos *xml* con cada uno de los diseños de las imágenes, botones e iconos utilizados en la aplicación.
- *layout*: Este paquete contiene los archivos *xml* con cada uno de los diseños que definen la estructura visual para la interfaz de usuario.
- *menu*: Este paquete contiene los archivos *xml* con cada uno de los diseños que definen los sub-menús de la aplicación.
- *mipmap*: Este paquete contiene las imágenes de los iconos personalizados utilizados en la *activity* **Ajustes**. A su vez contiene las imágenes del icono de la aplicación en sus diferentes resoluciones.
- *values*: Este paquete contiene varios archivos de recursos:
 - *arrays*: Archivo *xml* que contiene cada uno de los elementos utilizado en los objetos *spinner* e índice glucemico y ración de HC de los alimentos.

- *colors*: Archivo *xml* que contiene los colores definidos para la aplicación.
- *strings*: Archivo *xml* que contiene todos los *strings* (textos) utilizados para la aplicación. En caso de querer añadir un idioma nuevo, habría que duplicar estos archivos, traduciendo estos *Strings* al idioma deseado.
- *styles*: Archivo *xml* que contiene los estilos definidos para la aplicación.

Diseño de clases

Paquete *calculations*

- **CalculaBolo C.11**: Esta clase realiza los pasos necesarios para el cálculo del bolo corrector.

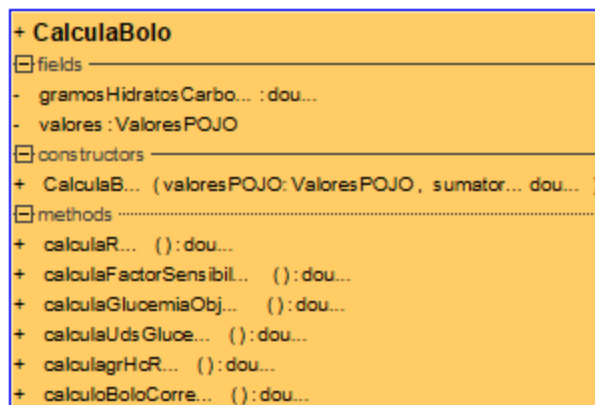


Figura C.11: Clase CalculaBolo

Paquete *model*

- **Alimento C.12**: Esta clase se utiliza para crear el objeto que irá dentro de la lista **Lista de Ingesta** de la pantalla *Carbohidratos*.

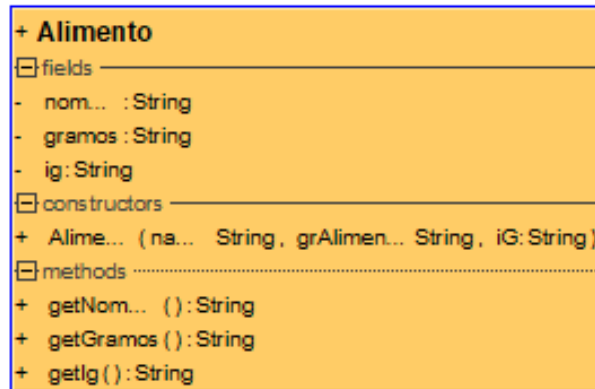


Figura C.12: Clase Alimento

- **DetallesIngesta** C.13: Esta clase se utiliza para crear el objeto que irá dentro de la lista **Detalles de la lista** de la pantalla *Detalles*.

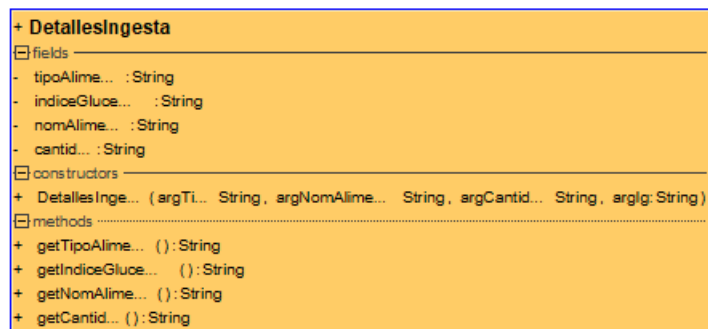


Figura C.13: Clase DetallesIngesta

- **RegistroIngestas** C.14: Esta clase se utiliza para crear el objeto que irá dentro de la lista **Registros diarios** de la pantalla *Listas de Ingestas*.

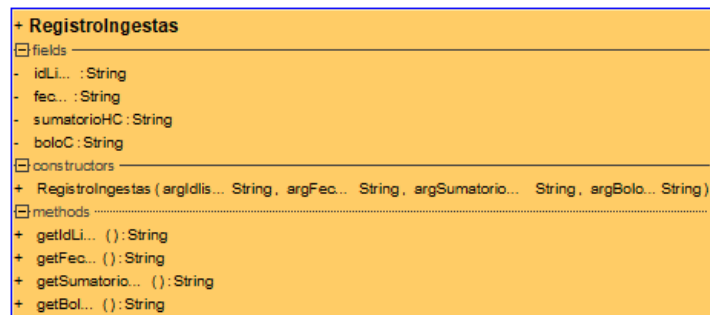


Figura C.14: Clase RegistroIngestas

Paquete *persistence*

- DataBaseHelper [C.15](#): Esta clase se utiliza para crear la base de datos.

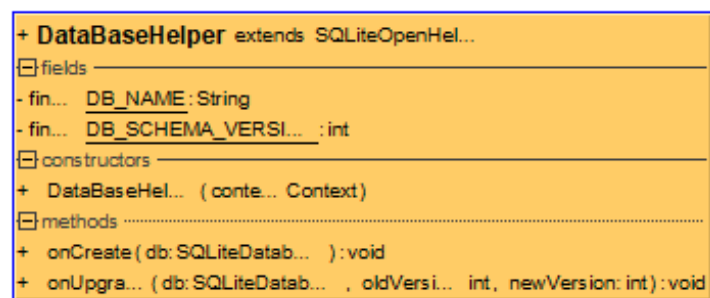


Figura C.15: Clase DataBaseHelper

- DataBaseManager [C.16](#): Esta clase se utiliza para crear la parte lógica de la base de datos.

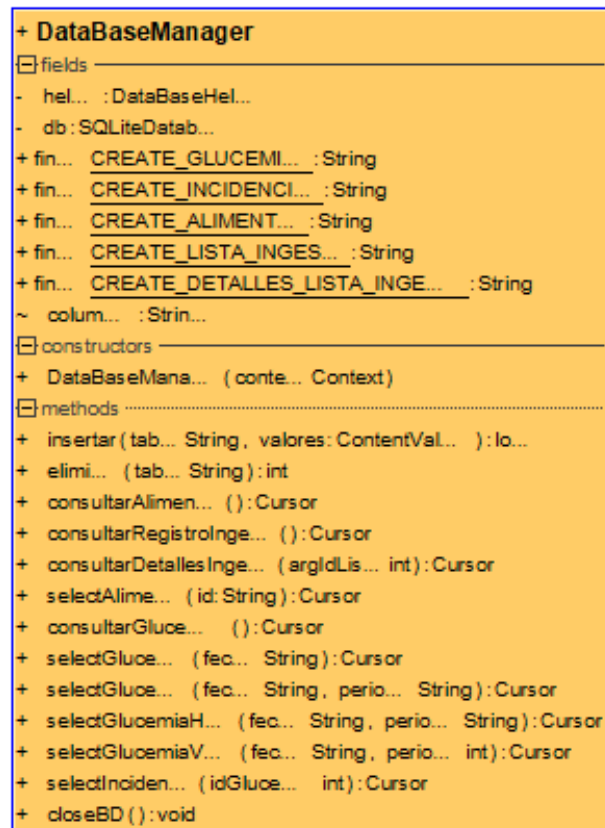


Figura C.16: Clase DataBaseManager

- ValoresPOJO **C.17**: Esta clase se utiliza para crear el objeto *POJO* con los datos médicos del usuario.

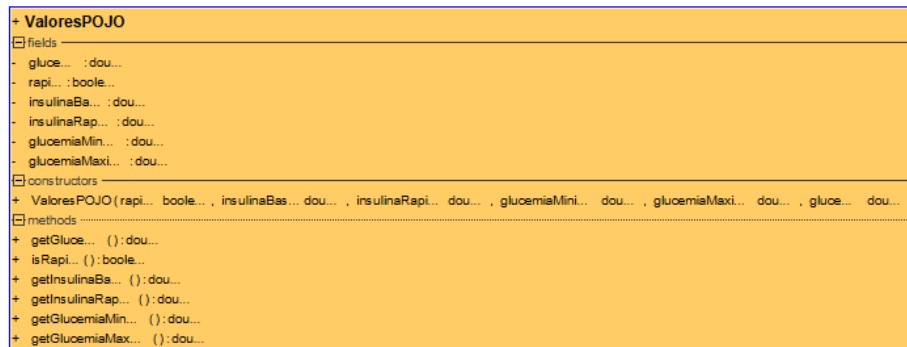


Figura C.17: Clase ValoresPOJO

Paquete *util*

- AjustesListAdapter C.18: Esta clase se utiliza para crear el *ArrayAdapter* personalizado para la lista de **Ajustes**.



Figura C.18: Clase AjustesListAdapter

- ChartListAdapter C.19: Esta clase se utiliza para crear el *ArrayAdapter* personalizado para la lista de **Historial de glucemias**.

```

+ ChartListAdapter extends ArrayAdapter...
  fields
  - mCont... : Context
  ~ mResource : int
  ~ fin... semanas : Strin...
  - valM... : int
  - val... : int
  constructors
  + ChartListAdap... ( conte... Context, resource: int, objec... List<LineDat... , seman... Strin... )
  methods
  + getVi... ( positi... int, convertVi... View, parent: ViewGro... ): View

```

Figura C.19: Clase ChartListAdapter

- IngestaAlimentoListAdapter **C.20**: Esta clase se utiliza para crear el *ArrayAdapter* personalizado para la lista **Lista de Ingesta**.

```

+ IngestaAlimentoListAdapter extends ArrayAdapter...
  fields
  - mCont... : Context
  ~ mResource : int
  ~ fin... ELEVADO : int
  ~ fin... MODERADO_M... : int
  ~ fin... MODERADO_M... : int
  ~ fin... BAJO : int
  constructors
  + IngestaAlimentoListAda... ( conte... Context, resource: int, objec... List<Aliment... )
  methods
  + getVi... ( positi... int, convertVi... View, parent: ViewGro... ): View

```

Figura C.20: Clase IngestaAlimentoListAdapter

- IngestaDetallesListAdapter **C.21**: Esta clase se utiliza para crear el *ArrayAdapter* personalizado para la lista **Detalles de la lista**.



Figura C.21: Clase IngestaDetallesListAdapter

- IngestaRegistroListAdapter [C.22](#): Esta clase se utiliza para crear el *ArrayAdapter* personalizado para la lista **Registros diarios**.

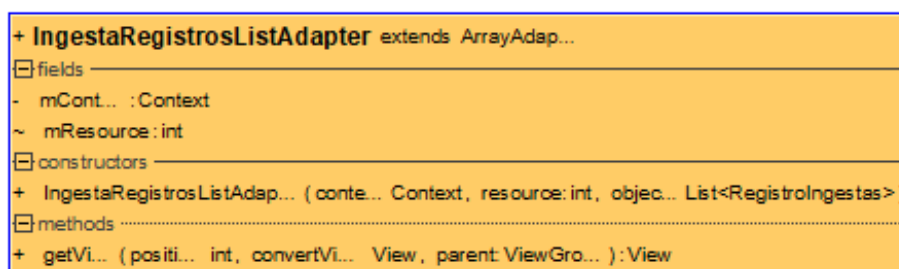


Figura C.22: Clase IngestaRegistroListAdapter

- ListViewUtility [C.23](#): Esta clase se utiliza para que nos permita adaptar el tamaño de una *list View* según el número de *items* que contenga.

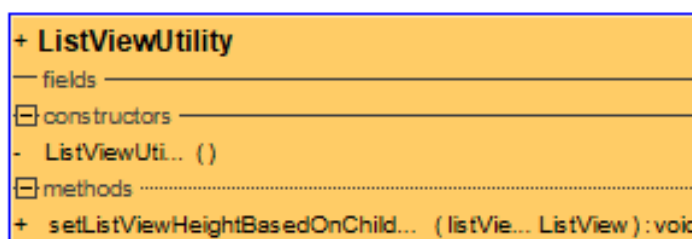


Figura C.23: Clase ListViewUtility

Paquete *view*

Como se ha mencionado en la sección C.4 este paquete contiene las *activitys* que dan funcionalidad a las distintas pantallas de la aplicación. Así mismo, cada una de las *activitys* contiene su propio *fragment*.

- Ajustes C.24: *Activity* encargada de gestionar el menú de Ajustes de la aplicación.

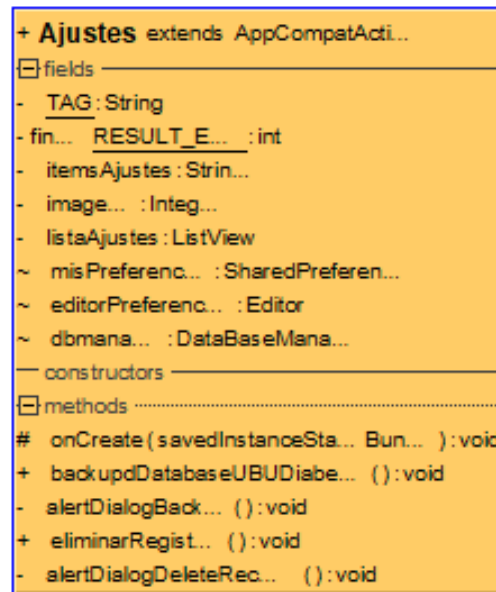


Figura C.24: *Activity* Ajustes



Figura C.25: *Fragment* Ajustes

- Carbohidratos C.26 C.27: Esta *activity* gestiona el registro de los alimentos a ingerir por parte del usuario. Además es la encargada de llamar a la clase CalculaBolo para obtener el resultado y mostrárselo al usuario por pantalla.

```

+ Carbohidratos extends AppCompatActivity
  fields
  - fin... GRAMOS_DE_... : dou...
  - TAG: String
  + fin... COLUMNA_RACION: int
  + fin... COLUMNA_IG: int
  - fin... RESULT_E... : int
  - misPreferenc... : SharedPreferences
  - rapi... : boole...
  - insulinaBa... : dou...
  - insulinaRap... : dou...
  - glucemiaMin... : dou...
  - glucemiaMaxi... : dou...
  - gluce... : dou...
  - bc_c... : Boole...
  - bcU... : Boole...
  - bcDos: Boole...
  - valoresPOJO: ValoresPOJO
  - cb: CalculaB...
  - sumatorioRacio... : dou...
  - actualSumatorioBo... : dou...
  - boloRes... : dou...
  ~ arrayAlimen... : ArrayList<Strin...
  ~ arrayRacion... : ArrayList<Integ...
  - listViewInge... : ListView
  - buttonAddAlime... : Butt...
  - buttonRemoveAlim... : Butt...
  - editTextGra... : EditT...
  - autoCompleteTextViewBusc... : AutoCompleteTextV...
  - listaComi... : Spin...
  - totalAlimentosOrdena... : Strin...
  - editTextActualSumator... : TextVi...
  - editTextActualBo... : TextVi...
  - alimentoInge... : Alime...
  - userListIngesta: ArrayList<Aliment...
  - adpListaInge... : IngestaAlimentoListAda...

```

Figura C.26: *Activity* Carbohidratos A

```

- itemSe... :int
- previousPositi... :int
- co... :int
- previous... :lo...
----- constructors -----
[ ] methods -----
# onCreate(savedInstanceSta... Bun... ):void
+ añadirOtroOnCl... (view:View):void
+ removeOnCl... (view:View):void
+ consulta_grHC_... (alimen... String):Strin...
+ calcularGramosDeHidratosDeCarb... (numeroGram... int, gramosPorRaci... String):dou...
+ finalizarOnCl... (view:View):void
+ calculaBoloCorre... (sumatorio... Dou... ):dou...
- generaComentarioB... (bo... dou... , nDecim... int):String

```

Figura C.27: *Activity* Carbohidratos B

```

+ CarbohidratosFragment extends Fragm...
----- fields -----
[ ] constructors -----
+ CarbohidratosFragm... ()
[ ] methods -----
+ onCreateVi... (inflat... LayoutInfla... , contain... ViewGro... , savedInstanceSta... Bun... ):View

```

Figura C.28: *Fragment* Carbohidratos

- ConsultaDetallesListaIngestas **C.29**: Esta *activity* es la encargada de gestionar los elementos necesarios para que el usuario pueda consultar **los detalles** de sus registros diarios referentes a los alimentos ingeridos y al cálculo del bolo corrector.

```

+ ConsultaDetallesListaIngesta extends AppCompatActivity
└─ fields
+ fin... COLUMNA_TI... : int
+ fin... COLUMNA_IG : int
+ fin... COLUMNA_ALIMEN... : int
+ fin... COLUMNA_CANTID... : int
+ idListaSel... : int
~ detallesListaInge... : Detalles Inge...
- detalleListaInge... : ArrayList<Detalles Ingest...
- adpDetalleListaInge... : IngestaDetallesListaAda...
~ lvDetalleInge... : ListView
└─ constructors
└─ methods
# onCreate ( savedInstanceState... Bun... ) : void

```

Figura C.29: *Activity* ConsultaDetallesListaIngesta

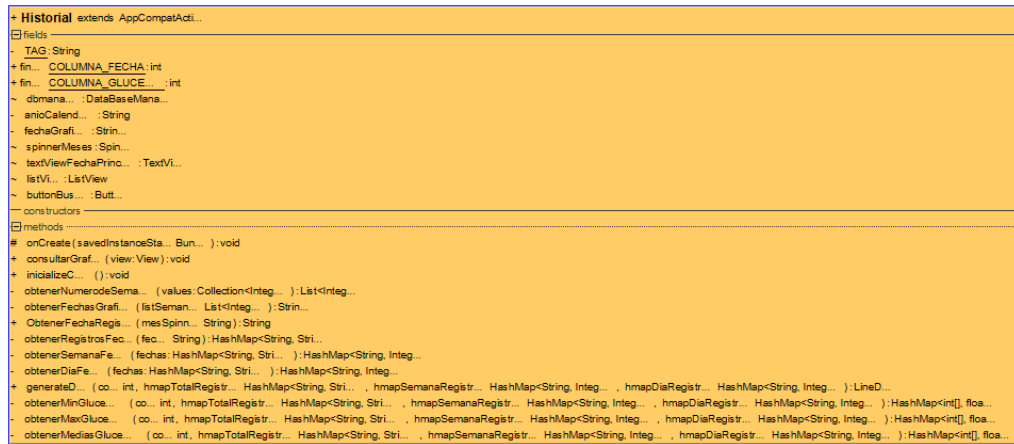
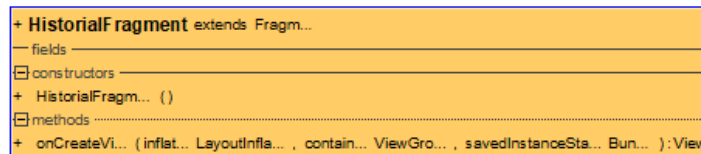
```

+ ConsultaDetallesListaIngestaFragment extends Fragment
└─ fields
└─ constructors
+ ConsultaDetallesListaIngestaFragm... ()
└─ methods
+ onCreateView ( inflater... LayoutInfla... , contain... ViewGro... , savedInstanceState... Bun... ) : View

```

Figura C.30: *Fragment* ConsultaDetallesListaIngesta

- Historial C.29: Esta *activity* es la encargada de gestionar los elementos necesarios para que el usuario pueda consultar el historial de sus registros de glucemias de manera gráfica.

Figura C.31: *Activity* HistorialFigura C.32: *Fragment* Historial

- Incidencias **C.33**: Esta *activity* es la encargada de gestionar los elementos necesarios para que el usuario pueda registrar una incidencia en los casos que corresponda. Es lanzada desde RegistroGlucemias si el valor introducido en ella por el usuario es superior o inferior al margen marcado por los valores mínimo y máximo de glucemia del perfil de usuario.

```

+ Incidencias extends AppCompatActivity
  fields
  - fin... RESULT_E... :int
  - inciden... :String
  constructors
  methods
  # onCreate(savedInstanceState... Bundle... ):void
  + incidenciasOnC... (view:View):void
  + generarContentVal... (inciden... String, observacion... String, id:Integer):ContentVal...
  + getDateTi... ():String

```

Figura C.33: *Activity* Incidencias

```

+ IncidenciasFragment extends Fragment
  fields
  constructors
  + IncidenciasFragm... ()
  methods
  + onCreateView... (inflat... LayoutInfla... , contain... ViewGroup... , savedInstanceState... Bundle... ):View

```

Figura C.34: *Fragment* Incidencias

- Launcher [C.35](#): Esta clase es la encargada de marcar el comportamiento de la aplicación cada vez que la abrimos. Si es la primera vez que se ejecuta la aplicación, lanzará la *activity* Registro, sino, lanza directamente la *activity* MenuPrincipal.

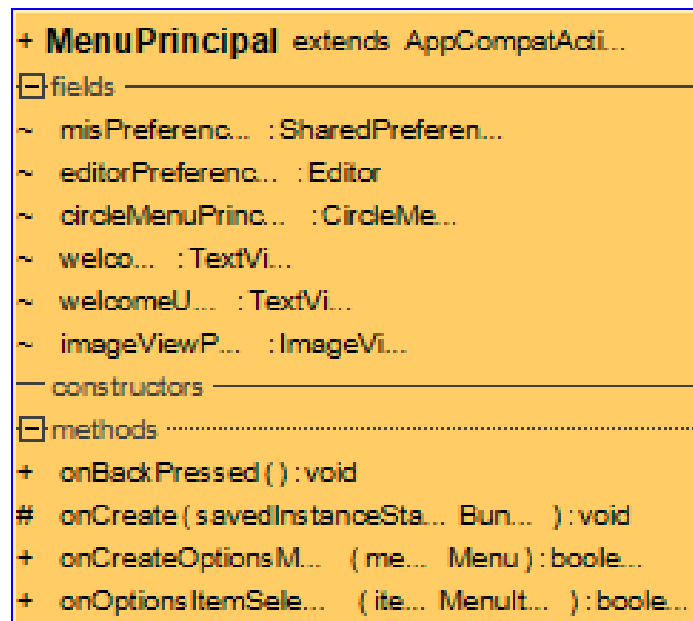
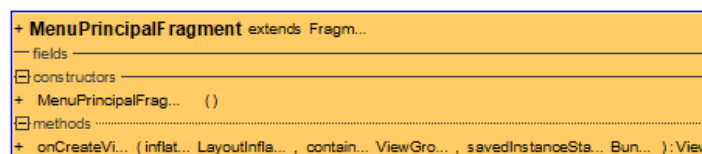
```

+ Launcher extends AppCompatActivity
  fields
  constructors
  methods
  # onCreate(savedInstanceState... Bundle... ):void

```

Figura C.35: *Clase* Launcher

- MenuPrincipal [C.36](#): Esta es la *activity* principal de la aplicación. Gestiona el menú principal de la aplicación, donde el usuario selecciona que funciones de la aplicación quiere utilizar.

Figura C.36: *Activity* MenuPrincipalFigura C.37: *Fragment* MenuPrincipal

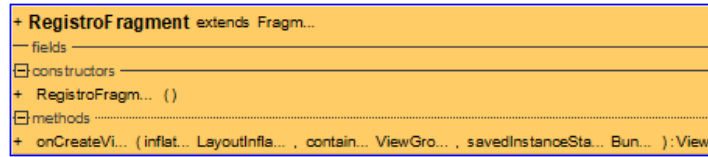
- Registro C.38: Esta *activity* es la encargada de gestionar los elementos necesarios para que el usuario pueda validar y registrar sus datos de perfil.


```

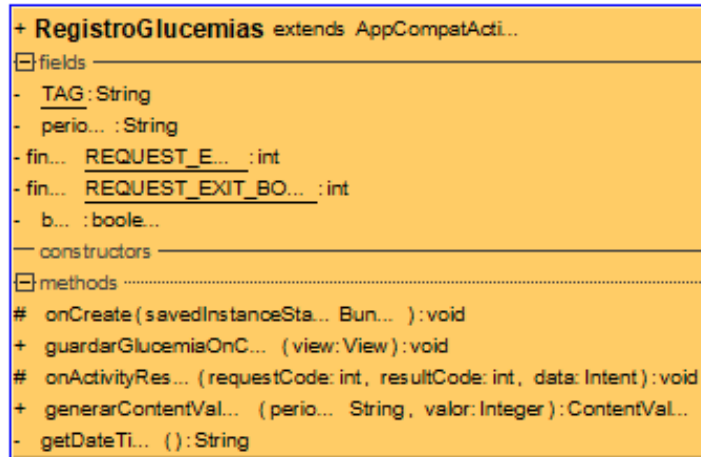
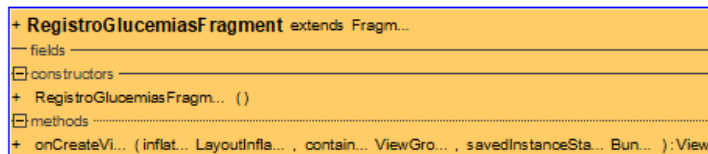
+ Registro extends AppCompatActivity
  fields
  - fin... TAG : String
  - fin... PREFERENCES_USER : String
  - misPreferenc... : SharedPreferences
  ~ nombre... : EditText
  ~ edad... : EditText
  ~ estaturaEt : EditText
  ~ pesoEt : EditText
  ~ maxEt : EditText
  ~ min... : EditText
  ~ udsBasalEt : EditText
  ~ udsRapida... : EditText
  ~ rapidaChe... : RadioButt...
  ~ ultrarrapidaCh... : RadioButt...
  ~ decimalBCCeroCh... : RadioButt...
  ~ decimalBCDosCh... : RadioButt...
  ~ decimalBcTresCh... : RadioButt...
  ~ radioGroupInsulinaB... : RadioGro...
  ~ radioGroupDecimales... : RadioGro...
  ~ imagePe... : ImageVi...
  ~ path : String
  ~ finalP... : String
  - fin... COD_SELECCIONA : int
  constructors
  methods
  # onCreate(savedInstanceState... Bundle) : void
  + fin... cargarPreferenc... () : void
  - validaPermi... () : boole...
  + onRequestPermissionsResult... (requestCode: int, permissions: String[], grantResults: int[]): void
  - solicitarPermisosMan... () : void
  - cargarDialogoRecomenda... () : void
  + clickIma... (view: View) : void
  + loadIma... () : void
  # onActivityResult... (requestCode: int, resultCode: int, data: Intent) : void
  + guardarperfilOnCl... (view: View) : void

```

Figura C.38: *Activity* Registro

Figura C.39: *Fragment* Registro

- RegistroGlucemias C.40: Esta *activity* es la encargada de gestionar los elementos necesarios para que el usuario pueda seleccionar el periodo en que ha realizado la medición de glucemia, y registre el valor correspondiente en la aplicación. Es lanzada al seleccionar la opción **Registro** de glucemias en el MenuPrincipal, o como primer paso al seleccionar la opción de **Calcular bolo**.

Figura C.40: *Activity* RegistroGlucemiasFigura C.41: *Fragment* RegistroGlucemias

- RegistroListaIngesta C.42: Esta *activity* es la encargada de gestionar los elementos necesarios para que el usuario pueda consultar sus registros diarios referentes a los alimentos ingeridos y al cálculo del bolo corrector.

```

+ RegistroListaIngesta extends AppCompatActivity
fields
+ fin... COLUMNA_ID : int
+ fin... COLUMNA_FECHA : int
+ fin... COLUMNA_SUMATORIO... : int
+ fin... COLUMNA_BOL... : int
- idListSeleco... : int
~ registroIngestas : RegistroIngestas
- registroListaInge... : ArrayList<RegistroIngestas>
- adpRegistroListaInge... : IngestaRegistrosListAdapter...
~ lvRegistroIngestas : ListView
constructors
methods
+ onBackPressed() : void
# onCreate(savedInstanceState... Bundle... ) : void
+ consultarItemLi... (view: View) : void

```

Figura C.42: *Activity* RegistroListaIngesta

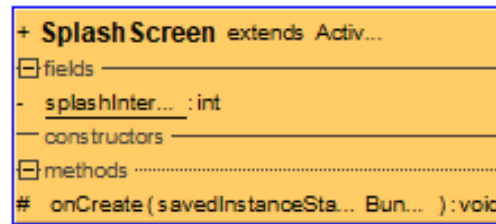
```

+ RegistroListaIngestaFragment extends Fragment
fields
constructors
+ RegistroListaIngestaFragm... ()
methods
+ onCreateView... (inflat... LayoutInfla... , contain... ViewGroup... , savedInstanceState... Bundle... ) : View

```

Figura C.43: *Fragment* RegistroListaIngesta

- SplashScreen C.44: Esta *activity* es la encargada de dar la “bienvenida” al usuario mostrando una imagen con el nombre y color de la aplicación. Esta *activity* es la primera en lanzarse al abrir la aplicación, posteriormente se lanza la *activity* Launcher.

Figura C.44: *Activity* SplashScreen

Paquete que contiene los test Unitarios

- CalculaBolo_CasoClinico_FalloTest C.45: Esta *clase* contiene las pruebas para el caso clínico que se recoge en la hoja “Casos clínicos” del documento *Excel* proporcionado por Diego Serrano Gómez.

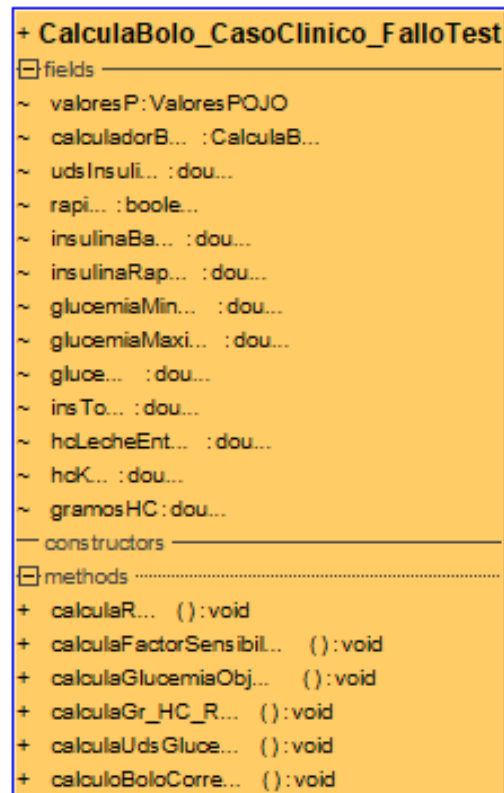


Figura C.45: Test: Caso clínico

- `CalculaBolo_NoPreciso_Test` C.46: Esta *clase* contiene las pruebas para el caso clínico que se recoge en la hoja “Sin alimentos 1” del documento *Excel* proporcionado por Diego Serrano Gómez.

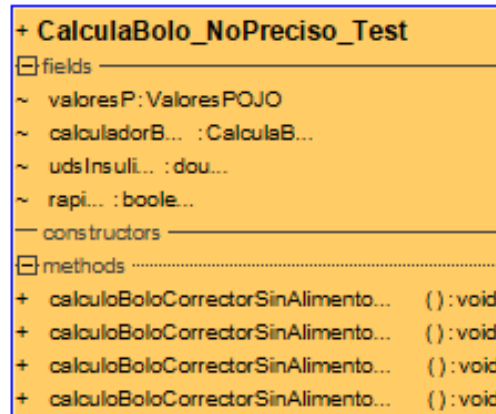


Figura C.46: Test: Sin alimentos 1

- `CalculaBolo_Preciso_Tes` C.47: Esta *clase* contiene las pruebas para el caso clínico que se recoge en la hoja “Alimentos de 1 en 1” del documento *Excel* proporcionado por Diego Serrano Gómez.

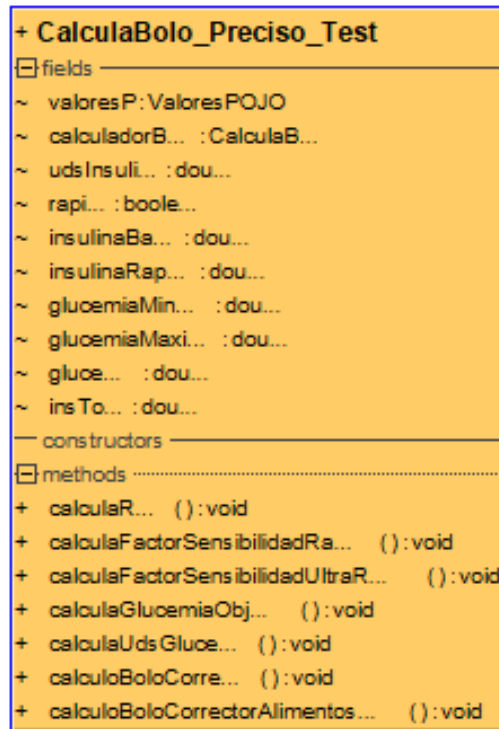


Figura C.47: Test: Alimentos de 1 en 1

C.5. Diseño de pruebas

En cuanto al diseño de pruebas, se han llevado a cabo una serie de *Unit Tests* y *User Interactions Tests* en sus correspondientes paquetes C.4. Para ello se han realizado las siguientes tareas para probar la aplicación:

Diseño de las pruebas

En este apartado se pretende identificar la o las técnicas de pruebas que se utilizarán para probar la aplicación.

Estas técnicas se agrupan en:

- *Técnicas de caja blanca o estructurales*: Esta técnica se basa en un examen minucioso de los detalles procedimentales del código a evaluar. En este caso, hemos llevado a cabo una serie de pruebas relacionadas con el cálculo del bolo de insulina y el cálculo de gramos de hidratos de carbono (HC). Para ello hemos utilizado la estrategia de **Pruebas Unitarias (Unit Tests)**.

- *Técnicas de caja negra o funcionales*: Esta técnica se basa en realizar pruebas sobre la interfaz de la aplicación, es decir, sobre la funcionalidad que debe realizar la misma. En este caso, por falta de tiempo no hemos probado toda la funcionalidad de la aplicación, únicamente se han probado un par de interfaces o pantallas. Para ello hemos utilizado la estrategia de **Espresso UI tests**.

Generación de los casos de prueba

En este apartado se pretende generar los datos que se utilizarán como entrada para ejecutar nuestra aplicación. Más concretamente, se trata de determinar el conjunto de entradas, condiciones de ejecución y resultados esperados para un objetivo particular.

En nuestro caso, el conjunto de datos los obtuvimos por parte de Bruno Martín Gómez y Diego Serrano Gómez. Estos datos se recogen en un documento *Excel* en el que se pueden observar un conjunto de hojas en las que se recogen distintos casos clínicos reales, y un documento *pdf* en el que se recogen diferentes fórmulas para el cálculo del bolo de insulina y el cálculo de gramos de HC. Los datos utilizados para la realización de las distintas pruebas unitarias son:

- Hoja “Sin alimentos 1”: Cálculo del bolo de insulina para 14 pacientes diferentes con distintas necesidades de insulina diaria. Glucemia objetivo de 120 mg/dl y una glucemia de 250, 225, 200 y 175.

Calculo de insulina				
Glucemia: 250 mg/dl				
Glucemia objetivo: 120 mg/dl				
Rápida	Lenta	Insulina total	UbuDiabetes	Teórico
10	8	18		1
12	10	22		2
14	12	26		2
16	14	30		2
18	16	34		2
20	18	38		3
22	20	42		3
24	22	46		3
26	24	50		4
28	26	54		4
30	28	58		4
32	30	62		4
34	32	66		4
36	34	70		5
Media UBU Diabetes			#¡NUM!	
Media Teórica			2,84	
Desviación			#¡NUM!	

Figura C.48: Caso de prueba 1

Calculo de insulina				
Glucemia: 225 mg/dl				
Glucemia objetivo: 120 mg/dl				
Rápida	Lenta	Insulina total	UbuDiabetes 1.0	Teórico
10	8	18		1
12	10	22		1
14	12	26		2
16	14	30		2
18	16	34		2
20	18	38		2
22	20	42		2
24	22	46		3
26	24	50		3
28	26	54		3
30	28	58		3
32	30	62		4
34	32	66		4
36	34	70		4
Media UBU Diabetes			#¡NUM!	
Media Teórica			2,36	
Desviación			#¡NUM!	

Figura C.49: Caso de prueba 2

Calculo de insulina				
Glucemia 200 mg/dl				
Glucemia objetivo: 120 mg/dl				
Rápida	Lenta	Insulina total	UbuDiabetes	Teórico
10	8	18		1
12	10	22		1
14	12	26		1
16	14	30		1
18	16	34		2
20	18	38		2
22	20	42		2
24	22	46		2
26	24	50		2
28	26	54		2
30	28	58		3
32	30	62		3
34	32	66		3
36	34	70		3
Media UBU Diabetes			#iNUM!	
Media Teórica			1,84	
Desviación			#iNUM!	

Figura C.50: Caso de prueba 3

Calculo de insulina				
Glucemia 175 mg/dl				
Glucemia objetivo: 120 mg/dl				
Rápida	Lenta	Insulina total	UbuDiabetes	Teórico
10	8	18		1
12	10	22		1
14	12	26		1
16	14	30		1
18	16	34		1
20	18	38		1
22	20	42		1
24	22	46		1
26	24	50		2
28	26	54		2
30	28	58		2
32	30	62		2
34	32	66		2
36	34	70		2
Media UBU Diabetes			#iNUM!	
Media Teórica			1,35	
Desviación			#iNUM!	

Figura C.51: Caso de prueba 4

- Hoja “Alimentos de 1 en 1”: Comprobación del cálculo de bolo de insulina para un paciente con una glucemia igual a la glucemia objetivo para distintos alimentos.

Perfil	
Glucemia	100 mg/dl
	140 mg/dl
Glucemia inicial	120 mg/dl
Insulina	Rápida
Insulina rápida	15 U
Insulina basal	15 U
Ratio	16,66

Figura C.52: Caso de prueba 5: Datos paciente

Alimentos	Una ración de HC son en gramos	Gramos	Modelo teórico
			Unidades de insulina
Cuajada	200	250	0,75
Flan	50	250	3
Helado de crema	50	250	3
Helado de Hielo	50	250	3
Helado sin azúcar añadido	100	250	1,5
Kéfir	200	250	0,75
Leche desnatada	200	250	0,75
Leche semidesnatada	200	250	0,75
Leche entera	200	250	0,75
Leche condensada	20	250	7,5
Leche en polvo	25	250	6
Nata líquida	300	250	0,5
Natillas	50	250	3
Petit Suisse	70	250	2,14
Queso fresco	250	250	0,6
Queso fresco de pasta, semis	*		
Yogur natural entero o desnatado	200	250	0,75
Yogur desnatado sabores o fruta	125	250	1,2
Yogur entero sabores o fruta	70	250	2,14
Yogur líquido	70	250	2,14
Yogur tipo actimel	100	250	1,5
Yogur tipo actimet 0%	100	250	1,5

Figura C.53: Caso de prueba 5: Datos alimentos 1

Arroz crudo	13	250	11,54
Arroz cocido	38	250	3,95
Arroz integral crudo	13	250	11,54
Arroz integral cocido	40	250	3,75
Arroz hinchado para desayuno	12	250	12,51
Arroz salvaje crudo	13	250	11,54
Arroz salvaje cocido	34	250	4,41
Avena crudo	17	250	8,83
Avena cocido	34	250	4,41
Avena copos	15	250	10
Boniato	50	250	3
Cebada crudo	14	250	10,72
Cebada cocido	42	250	3,57
Centeno crudo	15	250	10
Centeno cocido	38	250	3,95
Cereales desayuno	15	250	10
Cereales desayunos ricos en fibra	20	250	7,5
Cuscús crudo	15	250	10
Cuscús cocido	65	250	2,31
Fideos de arroz, udon, cocido	50	250	3

Figura C.54: Caso de prueba 5: Datos alimentos 2

Fideos de soja	40	250	3,75
Galletas tipo Digestiva	16	250	9,38
Galletas tipo María	15	250	10
Galletas tipo Principe	14	250	10,72
Galleta sin azúcar	18	250	8,34
Garbanzo crudo	20	250	7,5
Garbanzo cocido	50	250	3
Guisantes	100	250	1,5
Harina de trigo o maíz	15	250	10
Harina de centeno	17	250	8,83
Harina de soja	70	250	2,14
Hojaldre crudo	30	250	5
Hojaldre horneado	24	250	6,25
Judías blancas crudo	20	250	7,5
Judías blancas cocido	50	250	3
Lentejas crudo	20	250	7,5
Lentejas cocido	50	250	3
Maíz en lata	50	250	3
Maíz en lata sin azúcar añadido	90	250	1,67
Mijo crudo	15	250	10
Mijo cocido	53	250	2,83
Muesli	15	250	10
Pan blanco	20	250	7,5
Pan de centeno	20	250	7,5
Pan de molde	20	250	7,5

Figura C.55: Caso de prueba 5: Datos alimentos 3

Pan de hamburguesa o frankfurt	18	250	8,34
Pan de trigo integral	23	250	6,52
Pan rallado	15	250	10
Pan tostado o biscote	15	250	10
Pan en bastoncillos	15	250	10
Pasta alimenticia crudo	15	250	10
Pasta alimenticia cocido	50	250	3
Pasta al huevo	16	250	9,38
Patata cocida hervida	50	250	3
Patata horno o asada	35	250	4,29
Patatas fritas	30	250	5
Patatas chips	20	250	7,5
Puré de patatas, copos	15	250	10
Puré de patatas elaborado con leche	80	250	1,88
Quinoa cruda	19	250	7,9
Quinoa cocida	48	250	3,13
Sémola de trigo crudo	14	250	10,72
Sémola de trigo cocido	90	250	1,67
Soja seca crudo	30	250	5
Soja seca cocido	100	250	1,5

Figura C.56: Caso de prueba 5: Datos alimentos 4

Sushi	45	250	3,33
Tapioca cruda	12	250	12,51
Tapioca cocida	33	250	4,55
Trigo sarraceno crudo	12	250	12,51
Trigo sarraceno cocido	42	250	3,57
Trigo tierno crudo	16	250	9,38
Trigo tierno cocido	39	250	3,85
Yuca cocida	33	250	4,55
Aguacate	0		
Albaricoque	150	250	1
Arándano	100	250	1,5
Castaña cruda	30	250	5
Castaña tostada	25	250	6
Cereza	100	250	1,5
Chirimoya	50	250	3
Ciruela	100	250	1,5
Coco fresco	200	250	0,75
Coco seco	150	250	1
Dátil	15	250	10
Frambuesa	150	250	1
Fresones	200	250	0,75
Granada	70	250	2,14
Grosella	200	250	0,75
Grosella negra	140	250	1,07
Higos	100	250	1,5

Figura C.57: Caso de prueba 5: Datos alimentos 5

Kiwi	100	250	1,5
Limón			
Lichi	70	250	2,14
Mandarina	100	250	1,5
Mango	100	250	1,5
Manzana	100	250	1,5
Manzana asada	50	250	3
Melocotón	100	250	1,5
Melocotón en conserva	50	250	3
Melón	200	250	0,75
Membrillo	150	250	1
Membrillo dulce	20	250	7,5
Moras	150	250	1
Naranja	100	250	1,5
Nectarina	100	250	1,5
Nispero	100	250	1,5
Pera	100	250	1,5
Papaya	125	250	1,2
Paraguayo	100	250	1,5
Piña	100	250	1,5

Figura C.58: Caso de prueba 5: Datos alimentos 6

Piña en conserva	85	250	1,77
Piña en su jugo	60	250	2,5
Plátano	50	250	3
Sandía	200	250	0,75
Uva	50	250	3
Acelga	300	250	0,5
Ajo	40	250	3,75
Alcachofa	300	250	0,5
Apio	300	250	0,5
Apio-nabo	500	250	0,3
Berenjena	300	250	0,5
Berro	0		
Berza	0		
Borrajá	0		
Brócoli	300	250	0,5
Calabacín	300	250	0,5
Calabaza	200	250	0,75
Cardo	300	250	0,5
Cebolla	150	250	1
Cebolla frita en aros	100	250	1,5
Champiñón	0		
Col ácida	0		
Col Bruselas, Coliflor	300	250	0,5
Escarola	0		
Endibia	300	250	0,5

Figura C.59: Caso de prueba 5: Datos alimentos 7

Esparrago verde	0		
Espinaca	0		
Grellos	0		
Judía verde	250	250	0,6
Lechuga	300	250	0,5
Lombarda	0		
Nabo	300	250	0,5
Palmitos	200	250	0,75
Pepino	300	250	0,5
Pimiento verde/rojo	300	250	0,5
Puerro	300	250	0,5
Rábano	300	250	0,5
Remolacha	150	250	1
Repollo	300	250	0,5
Ruibarbo	0		
Setas	300	250	0,5
Soja en brotes	300	250	0,5
Tomate	300	250	0,5
Zanahoria	150	250	1
Zanahoria hervida	200	250	0,75

Figura C.60: Caso de prueba 5: Datos alimentos 8

Zanahoria en conserva	225	250	0,67
Aceituna	250	250	0,6
Albaricoque seco	15	250	10
Almendra	150	250	1
Almendra tostada	140	250	1,07
Avellana	150	250	1
Cacahuete	100	250	1,5
Ciruela pasa	15	250	10
Dátil seco	15	250	10
Higo	15	250	10
Nuez	300	250	0,5
Piñón	300	250	0,5
Pipas	80	250	1,88
Pistacho	80	250	1,88
Sésamo	100	250	1,5
Uva pasa	15	250	10
Bebida isotónica	130	250	1,15
Bebida refrescante tipo cola o sabores	100	250	1,5
Bebida refrescante tipo cola o sabores light	0		
Bebida de cacao	100	250	1,5
Bebida de soja	250	250	0,6
Bebida energética	80	250	1,88

Figura C.61: Caso de prueba 5: Datos alimentos 9

Bitter	100	250	1,5
Cava brut	0		
Cava seco o semiseco	250	250	0,6
Cerveza	250	250	0,6
cerveza light	300	250	0,5
Cerveza sin	250	250	0,6
Destilados (dinebre, whisky, ron, vodka)	0		
Gaseosa	0		
Horchata	75	250	2
Horchata light	300	250	0,5
Licor de melocotón o manzana	30	250	5
Mosto	70	250	2,14
Sangría	100	250	1,5
Sidra	200	250	0,75
Tónica	100	250	1,5
Vermut	75	250	2
Vino blanco o tinto	0		
Vino dulce	75	250	2
Zumo de fruta comercial	100	250	1,5

Figura C.62: Caso de prueba 5: Datos alimentos 10

Zumo de fruta natural o sin azúcar añadido	200	250	0,75
Azúcar blanco	10	250	15,01
Azúcar moreno	10	250	15,01
Barrita energética (cereales)	20	250	7,5
Bizcocho o melindo	20	250	7,5
Bollería en general	20	250	7,5
Cruasán	20	250	7,5
Cacao en polvo	12	250	12,51
Cacao en polvo sin azúcar	22	250	6,82
Calamares a la romana	120	250	1,25
Canelones con bechamel	100	250	1,5
Caramelo	12	250	12,51
Chocolate blanco o con leche	17	250	8,83
Chocolate negro	25	250	6
Churros	25	250	6
Crema de cacao	25	250	6
Crema de cacahuete	100	250	1,5
Crema pastelera	40	250	3,75
Croquetas	50	250	3
Donuts	23	250	6,52
Empanadilla de carne	50	250	3
Ensamada	23	250	6,52
Fructosa (edulcorante)	10	250	15,01
Gazpacho comercial	150	250	1

Figura C.63: Caso de prueba 5: Datos alimentos 11

Glucosa (líquida o en pastillas)	10	250	15,01
Golosinas	18	250	8,34
Kétchup	50	250	3
Lasaña	100	250	1,5
Levadura	130	250	1,15
Magdalena	25	250	6
Mazapán	25	250	6
Merengue	11	250	13,64
Mermelada	20	250	7,5
Mermelada light	0		
Miel	13	250	11,54
Mostaza	0		
Palomitas	20	250	7,5
Pastel de chocolate	25	250	6
Pastel de crema	35	250	4,29
Pepinillos en vinagre	0		
Pizza	40	250	3,75
Regaliz	15	250	10
Salsa barbacoa	100	250	1,5
Salsa bechamel	100	250	1,5

Figura C.64: Caso de prueba 5: Datos alimentos 12

Salsa boloñesa	150	250	1
Salsa carbonara	0		
Salsa de tomate comercial	100	250	1,5
Salsa de soja	0		
Sucedaneo de café, EKO.	0		
Surimi (palitos de cangrejo)	100	250	1,5
Tarta de manzana	25	250	6
Tofú	0		
Tortilla de patata	120	250	1,25
Turrón de Alicante	25	250	6
Turrón de Jijona	25	250	6
Vinagre	0		
Vinagre tipo Módena	15	250	10

Figura C.65: Caso de prueba 5: Datos alimentos 13

- Hoja “Casos clínicos”: Un único caso clínico en el que se recogen sus datos médicos, dos alimentos y se bolo de insulina esperado. Este caso de prueba fue de gran utilidad ya que se sabía que la versión anterior de nuestra aplicación obtenía ciertos resultados erróneos en este caso concreto.
 - Perfil de usuario:
 - Glucemia min: 100.
 - Glucemia max: 120.
 - Tipo de insulina: Rápida.
 - Insulina basal: 20.
 - Insulina rápida: 20.
 - Momento del cálculo: Antes de desayunar.
 - Valor de glucemia: 130.

- Cálculo del bolo:
 - Tipo de alimento 1: Lácteos.
 - Alimento 1: Leche entera.
 - Gramos alimento 1: 250.
 - Tipo de alimento 2: Fruta.
 - Alimento 2: Kiwi.
 - Gramos alimento 2: 100.
- Resultados:
 - Uds insulina teóricas: 2.33
 - Uds insulina UBUDiabetes: 2 (resultado versión anterior)
- Documento “Indicaciones para la corrección de UBUDiabetes”:

Unidades de insulina:

$$\text{Unidades de insulina} = \frac{\text{Gramos HC}}{\text{Ratio de insulina}} + \frac{\text{Glucemia inicial} - \text{Glucemia objetivo}}{\text{Factor de sensibilidad}}$$

Ratio de insulina:

$$\text{Ratio de insulina} = \frac{500}{\text{Insulina basal} + \text{Insulina rápida}}$$

Factor de sensibilidad a la insulina (FSI):

$$\text{FSI para la insulina rápida} = \frac{1500}{\text{Insulina basal} + \text{Insulina rápida}}$$

$$\text{FSI para la insulina ultrarrápida} = \frac{1800}{\text{Insulina basal} + \text{Insulina rápida}}$$

Glucemia objetivo:

$$\text{Glucemia objetivo} = \frac{\text{Glucemia máxima} + \text{Glucemia mínima}}{2}$$

En cuanto a los datos para las pruebas con *Espresso*, se utilizaron los mismos que para los test unitarios.

Definición de los procedimientos de las pruebas

Antes de empezar con la realización de las pruebas unitarias, cabe mencionar, que primero se tuvo que corregir los fallos en los cálculos de bolo de insulina en la versión anterior. Tras corregir dichos fallos se empezó a crear los primeros test unitarios. El proceso que se llevó a cabo para la realización de las pruebas unitarias fue el siguiente:

- Creamos una *clase* para cada documento u hoja que contenga los datos a probar. En este caso, disponemos de 3 hojas de *Excel* y un documento *pdf*, con lo cual, creamos 4 *clases* en su correspondiente paquete.
- Creamos los *métodos* necesarios para probar el cálculo del bolo de insulina.
- Introducimos los datos y realizamos los cálculos de forma manual.
- Comprobamos que los resultados obtenidos de forma manual son iguales a los esperados (resultados obtenidos en cada documento).

El proceso que se llevó a cabo para la realización de las pruebas de interacción de usuario fue el siguiente:

- Creamos una *clase* por cada pantalla que se desee probar.
- Establecemos la *@Rule* (Regla) necesaria para cada clase.
- Creamos los *métodos* necesarios para probar la interacción del usuario.
- Introducimos los datos de forma manual en cada elemento que se desee probar para establecer una acción del usuario.
- Comprobamos que la acción obtenida manualmente es la obtenida por la aplicación.

Estos tests serán llevados a cabo por el desarrollador y pueden ser modificados o añadidos en cualquier momento.

Ejecución de las pruebas

Los test unitarios se ejecutarán en el **JVM** local.

En cuanto a los UI tests, se ejecutarán en un emulador creado previamente.

Apéndice *D*

Documentación técnica de programación

D.1. Introducción

En este anexo se detalla tanto la información relativa al código de la aplicación, como el manual del programador con todos los pasos a seguir para poder desplegar y cargar el proyecto en una nueva máquina y seguir trabajando sobre el mismo.

Cabe mencionar que la mayoría de los siguientes apartados se han sacado del Manual del programador del proyecto predecesor (consultar [2]) ya que, de no ser así tendríamos que referenciar continuamente y podría causar problemas a la persona interesada.

D.2. Estructura de directorios

Nuestro proyecto se divide en dos repositorios principales, uno donde tendremos el código fuente, y otro con la documentación.

Código fuente

El código fuente de la aplicación se encuentra en *Github*. El enlace del repositorio es el siguiente: https://github.com/luismiguel17/UBUDiabetes2.0_GII17.0J

El código fuente se puede obtener mediante cualquiera de las opciones que se nos ofrecen en la plataforma [D.1](#):

APÉNDICE D. DOCUMENTACIÓN TÉCNICA DE PROGRAMACIÓN

- **A**: Descargándolo en formato *.zip*.
- **B**: Copiándolo directamente al ordenador para ser utilizado desde GitHub Desktop si se tiene instalado.
- **C**: Clonando el repositorio utilizando para ello la URL que nos proporciona GitHub.

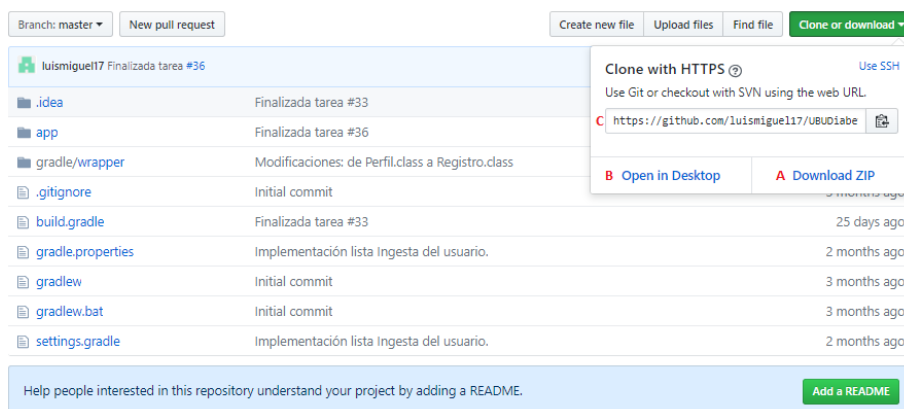


Figura D.1: Código fuente: GitHub

En el repositorio mencionado en el apartado **D.2** podemos encontrar todas las carpetas relacionadas con el código fuente del proyecto, es decir, en él podemos encontrar todos los ficheros relacionados únicamente con el proyecto creado en Android Studio.

Documentación

La documentación relacionada con el proyecto la podemos recoger en *GitHub*. El enlace del repositorio es el siguiente: https://github.com/luismiguel17/Documentation_UBUdiabete2.0_GII17.0J.

La documentación se puede obtener de la misma manera que el código fuente y en ella tenemos las siguientes carpetas:

- *img*: Contiene todas las imágenes utilizadas en la memoria y anexos.
- *tex*: Contiene los distintos archivos en los que se dividen la memoria y los anexos.
- Los *.pdf* de los Anexos y Memoria.

D.3. Manual del programador

En este apartado vamos a describir como instalar las diferentes herramientas necesarias para realizar el proyecto.

Android Studio

Requisitos mínimos

Para Instalar Android Studio en Windows 10, es fundamental comprobar los siguientes requisitos mínimos:

- Sistema operativo Windows 7/8/10 (32 o 64 bits).
- 2GB de RAM (8GB de memoria RAM recomendado).
- 2GB de espacio libre en el disco (4GB recomendado).
- Resolución mínima de 1280 x 800.
- JDK 8 de java.
- 64 bits y procesador Intel (emulador).

Instalación

- Paso 1: Ir a la página web principal de desarrolladores de Android y descargar Android Studio: <https://developer.android.com/studio/>.

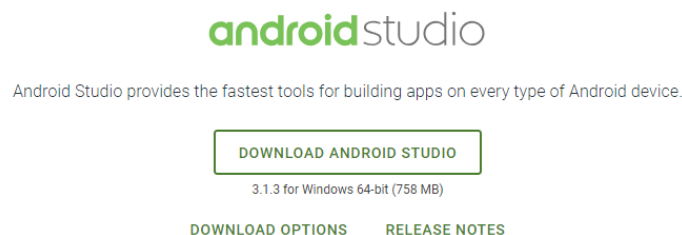


Figura D.2: Android Studio: Descarga

- Paso 2: Ejecutar el instalador en nuestro equipo de trabajo y comenzar con la instalación.

APÉNDICE D. DOCUMENTACIÓN TÉCNICA DE PROGRAMACIÓN

- Paso 3: Seguir el asistente de instalación de Android Studio hasta completar la instalación. El asistente nos marcará por defecto aquellas configuraciones necesarias para la correcta instalación de android Studio. Se recomienda no realizar ningún cambio en él, excepto si se desea cambiar la ruta de instalación.

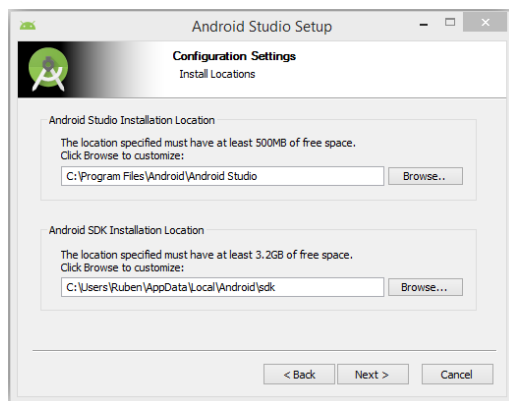


Figura D.3: Android Studio: Path de instalación

- Paso 4: Una vez finalice la instalación, Android Studio se conectará a internet y descargará los elementos del *SDK* necesarios para su correcto funcionamiento.
- Paso 5: Una vez finalizada la copia de datos ya podremos ejecutar la plataforma de desarrollo Android Studio.

Dependencias

Posteriormente a la instalación de Android Studio, se recomienda instalar las dependencias necesarias para el correcto funcionamiento del proyecto. Para ello, únicamente hay que añadir las siguientes líneas de código en el archivo *build.gradle* del módulo de la app:

```
dependencies {

    implementation 'com.android.support:support-v4:26.1.0'
    implementation 'com.android.support:multidex:1.0.1'
    implementation 'com.android.support:design:26.1.0'

    implementation fileTree(include: ['*.jar'], dir: 'libs')
    implementation 'com.android.support:appcompat-v7:26.1.0'
    implementation 'com.android.support.constraint:constraint-layout:1.0.2'
    implementation 'com.android.support:design:26.1.0'
    implementation 'io.reactivex.rxjava2:rxjava:2.1.3'
    implementation 'io.reactivex.rxjava2:rxandroid:2.0.1'
    implementation 'com.android.support:cardview-v7:26.1.0'
    implementation 'com.android.support:recyclerview-v7:26.1.0'

    //test Implementation
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'com.android.support.test:runner:1.0.2'
    androidTestImplementation 'com.android.support.test.espresso:espresso-core:3.0.2'
    androidTestImplementation 'com.android.support.test.espresso:espresso-accessibility:3.0.2'
    androidTestImplementation 'com.google.guava:guava:22.0'
    androidTestImplementation 'com.android.support.test.espresso:espresso-intents:3.0.2'

    //Circle Menu
    implementation 'com.github.Hitomis:CircleMenu:v1.0.2'
    //MPAndroidChart
    implementation files('libs/MPAndroidChart-v3.0.1.jar')

}
```

Figura D.4: Android Studio: Dependencias

DB Broser for SQLite

- Paso 1: Ir a la página web principal descargar DB Browser for SQLite:
<https://sqlitebrowser.org/>.
- Paso 2: Ejecutar el instalador en nuestro equipo de trabajo y comenzar con la instalación.
- Paso 3: Seguir el asistente de instalación hasta completar la instalación. El asistente nos marcará por defecto aquellas configuraciones necesarias para su correcta instalación. Se recomienda no realizar ningún cambio en él, excepto si se desea cambiar la ruta de instalación.
- Paso 4: Una vez finalice la instalación ya podremos ejecutar DB Browser for SQLite.

SonarLint

Para analizar la calidad del código se ha comprobado mediante la herramienta *SonarLint* de Android Studio:

- Paso 1: Desde Android Studio abrimos: File\settings\Plugins

- Paso 2: Buscar e instalar el *plugin* **SonarLint**.
- Paso 3: Para analizar nuestro proyecto debemos abrir: Analyze\Analyze with SonarLint

Emuladores

Para poder ejecutar nuestra proyecto, en nuestro caso, nuestra aplicación, hemos creado distintos emuladores de prueba con distintas versiones del sistema operativo de Android.

La creación de un emulador se detalla en la memoria del proyecto, apartado **Técnicas y Herramientas\Android Studio\AVD Manager**

D.4. Compilación, instalación y ejecución del proyecto

El primer paso, antes de cargar el proyecto, será descargar algunos paquetes para ahorrarnos instalaciones futuras:

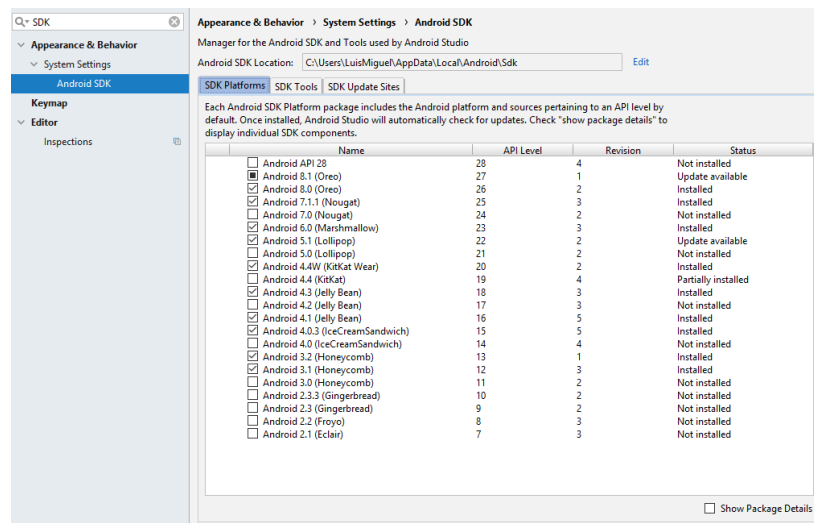


Figura D.5: Android Studio: Plataformas

D.4. COMPILACIÓN, INSTALACIÓN Y EJECUCIÓN DEL PROYECTO

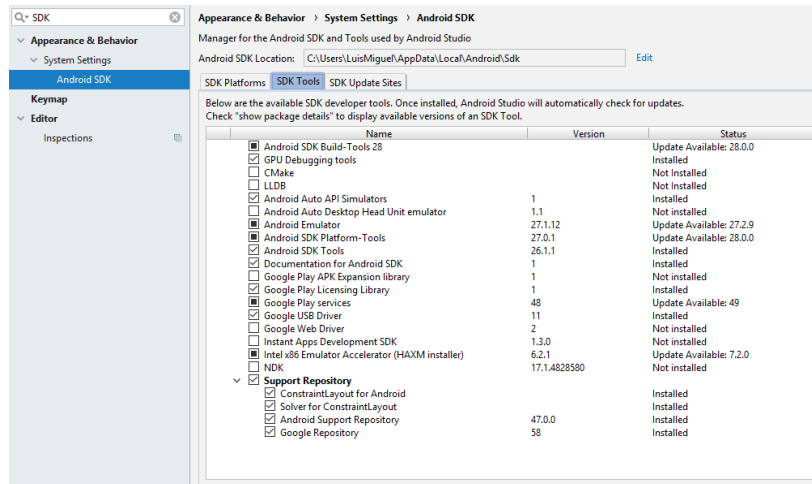


Figura D.6: Android Studio: Tools

Una vez instalados estos paquetes, pasamos a cargar el proyecto. La forma en la que hemos cargado el proyecto inicial ha sido la opción: Open an existing Android Studio Project [D.7](#). Para ello debemos descargar previamente el archivo *.zip* de *GitHub*.

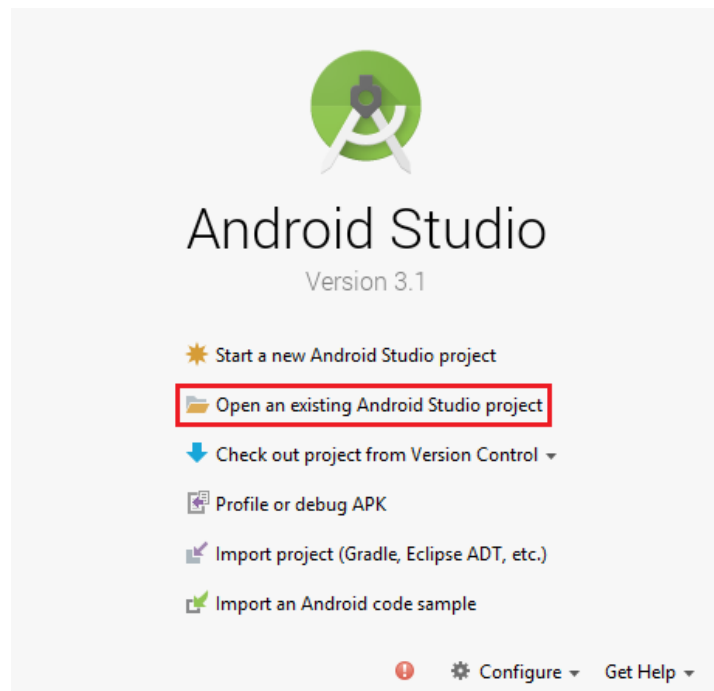


Figura D.7: Android Studio: Cargar Proyecto

Seleccionamos el directorio en el que tenemos el proyecto y procedemos a cargarlo. Una vez importado, ya podemos comenzar a trabajar sobre el mismo.

Para ejecutar el proyecto debemos:

- Paso 1: Ejecutar un emulador creado previamente: **Tools\AVD Manager**
- Paso 2: Desde la barra de herramientas de Android Studio: clic en el botón “Run app” (icono de color verde con el símbolo play).
- Paso 3: Elegimos el emulador ejecutado previamente y clic en OK.

D.5. Pruebas del sistema

Para la realización de las pruebas unitarias se ha seguido en siguiente proceso:

- Paso 1: Abrimos el archivo Java que contiene el código que deseemos probar.
- Paso 2: Clic en la clase o método que deseemos probar, luego presionamos *Ctrl+Shift+T*.
- Paso 3: En el menú que aparece, hacemos clic en *Create New Test*.
- Paso 4: En el diálogo *Create Test*, editamos los campos necesarios, seleccionamos los métodos que queremos generar y luego clic en OK D.8.
- Paso 5: En el diálogo *Choose Destination Directory*, hacemos clic en el conjunto de orígenes correspondiente al tipo de prueba que deseemos crear, en este caso, test para una prueba de unidad local. Luego clic en OK D.9.

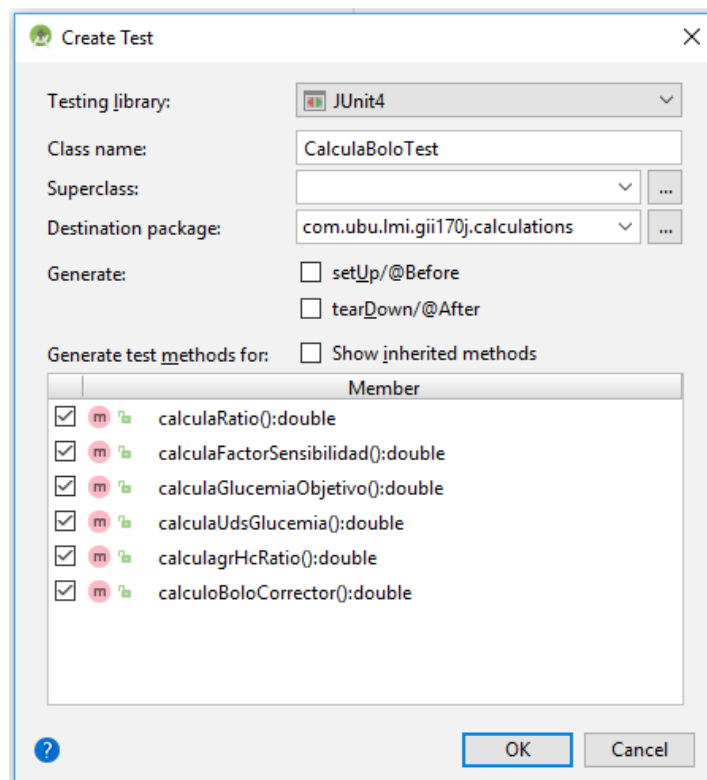


Figura D.8: Android Studio: Nuevo Test Unitario A

APÉNDICE D. DOCUMENTACIÓN TÉCNICA DE PROGRAMACIÓN

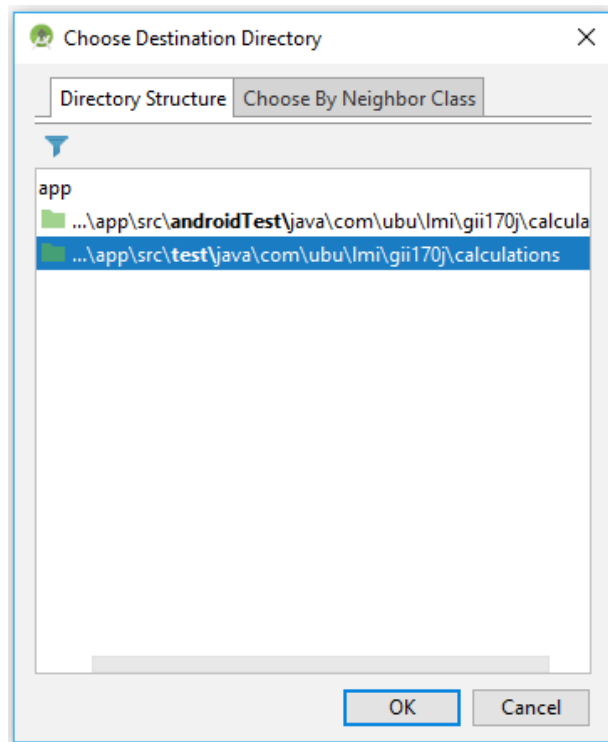


Figura D.9: Android Studio: Nuevo Test Unitario B

Para la realización de las pruebas de interacción de usuario se siguieron los mismos pasos anteriores hasta el paso 4.

- Paso 5: En el diálogo *Choose Destination Directory*, hacemos clic en el conjunto de orígenes correspondiente al tipo de prueba que deseamos crear, en este caso, *androidTest* para una prueba instrumentada.

Para ejecutar una prueba podemos hacerlo de cualquiera de las siguientes maneras:

- Paso 1: Hacemos clic con el botón secundario en una prueba y luego con el primario en *Run*.
- Paso 2: Para ejecutar todas las pruebas, hacemos clic con el botón secundario en el directorio de pruebas y luego con el primario en *Run tests*.

IMPORTANTE: Nos aseguramos de que nuestro proyecto esté sincronizado con *Gradle* haciendo clic en *Sync Project* en la barra de herramientas.

Apéndice *E*

Documentación de usuario

E.1. Introducción

En este apartado se detallarán tanto el manual de instalación como los pasos necesarios para aprender a utilizar la aplicación. Alguno de los puntos detallados a continuación son similares a la versión anterior, aunque en su mayoría son nuevos ya que la interfaz de usuario ha sido redefinida casi por completo.

E.2. Requisitos de usuarios

Para poder utilizar la aplicación en nuestro dispositivo móvil o tablet, los requisitos mínimos necesarios son:

- Sistema Operativo: Android.
- Versión SO: min 4.1 (Jelly Bean) max 8.0 (Oreo).
- Espacio libre: 2GB.

E.3. Instalación

La instalación de una aplicación en un dispositivo Android es muy sencilla. Necesitaremos realiza los siguientes pasos:

- Descargar el archivo *apk* de la aplicación. Esto lo podemos hacer de las siguientes maneras:

- Mover el archivo a nuestro dispositivo: Si disponemos del archivo en un ordenador, debemos conectar el dispositivo a nuestro ordenador mediante cable USB en modo **Transferencia de medios (MTP)**. Una vez conectado copiamos el archivo *apk* en la tarjeta SD (o memoria interna) de nuestro dispositivo. Se recomienda crear una carpeta adicional con el nombre de la aplicación para poder ubicarla mejor.
- Descargar el archivo *apk* de internet: Lo único que debemos hacer es dirigirnos a la página: https://github.com/luismiguel17/UBUDiabetes2.0_GII17.0J/releases y descargar la última versión *apk* de la aplicación.
- Habilitar la opción **Orígenes desconocidos** en nuestro dispositivo. Esta opción suele encontrarse en el menú: **Ajustes\Seguridad**.
- Buscar el archivo *apk* en nuestro dispositivo y ejecutarlo.
- Durante la instalación de la aplicación, el asistente nos preguntará si queremos instalar la aplicación, damos clic en Instalar (Aceptar).
- Al abrir la aplicación instalada, el asistente de Android nos pedirá aceptar los permisos que la aplicación requiera, debemos aceptarlos.

E.4. Manual del usuario

A continuación, se explica el funcionamiento de la aplicación con un ejemplo de un caso real. Este caso lo hemos tomado de los casos de estudio propuestos por Diego Serrano Gómez (los utilizados para el *testing* de la aplicación).

Datos del usuario:

- Datos personales (Datos cualesquiera):
 - Nombre y Apellidos: David Pérez González.
 - Edad: 24
 - Altura(cm): 175
 - Peso(Kg): 80
- Dato médicos (Datos reales):
 - Glucemia min deseada: 100.

- Glucemia max deseada: 120.
- Insulina del bolo: Rápida.
- Unidades de insulina basal: 20.
- Unidades de insulina rápida: 20.
- Numero de decimales en el bolo corrector. 2.

SplashScreen

Al arrancar la aplicación, siempre se mostrará una pantalla de bienvenida con el nombre y colores de la aplicación.



Figura E.1: SplashScreen

Registro-Perfil de usuario

En la ventana de Registro-Perfil de usuario debemos rellenar los datos referentes a las características del usuario. Es necesario rellenar todos los campos, en caso contrario, al darle a GUARDAR la aplicación nos mostrará

un mensaje para avisarnos del error. Además, debemos introducir los valores de glucemia entre los límites expuestos (80-250) y en el orden mín-máx.

Los datos más importantes de esta pantalla son los valores mínimo y máximo de glucemia deseados, y las unidades de insulina basal y rápidas diarias. Los primeros se utilizarán para registrar o no incidencias en caso de salirnos de esos límites, mientras que los segundos se utilizan directamente para el cálculo del bolo corrector. El resto de los valores son importantes para mantener un perfil del usuario y para futuras funcionalidades de la aplicación, pero no son necesarios para el cálculo del bolo ni para otras funcionalidades de la aplicación actual. En cuanto al dato Decimales en el Bolo corrector, únicamente sirve para indicar al sistema cuantos decimales debe mostrar al final en el cálculo del bolo corrector.

Debemos mencionar también, que el usuario tiene la opción de añadir una foto a su perfil.

Datos Personales			
Nombre y Apellidos			
David Pérez González			
Edad	Altura (cm)	Peso (kg)	
24	175	80	

Figura E.2: Registro-Perfil de Usuario 1

Figura E.3: Registro-Perfil de Usuario 2

Menú principal

Una vez introducidos los datos del perfil, la pantalla principal de la aplicación será la del menú principal.

En este menú principal disponemos de 4 opciones:

- **1: Calcular bolo E.4.** Nos permite calcular el bolo corrector después de realizar dos pasos: Registro de glucemias E.4 y Recuento de carbohidratos E.4.
- **2: Registro de glucemias E.4.** Nos lleva a la pantalla de registro de glucemias donde podemos introducir el valor de glucemia actual para guardarlo en la base de datos.

- **3: Historial de glucemias** E.4. Nos permite consultar gráficamente los valores medios, máximos y mínimos de glucemias según el mes seleccionado.
- **4: Consultar registros** E.4. Nos permite consultar los registros diarios relacionados con el cálculo del bolo corrector y la ingesta de alimentos del usuario.

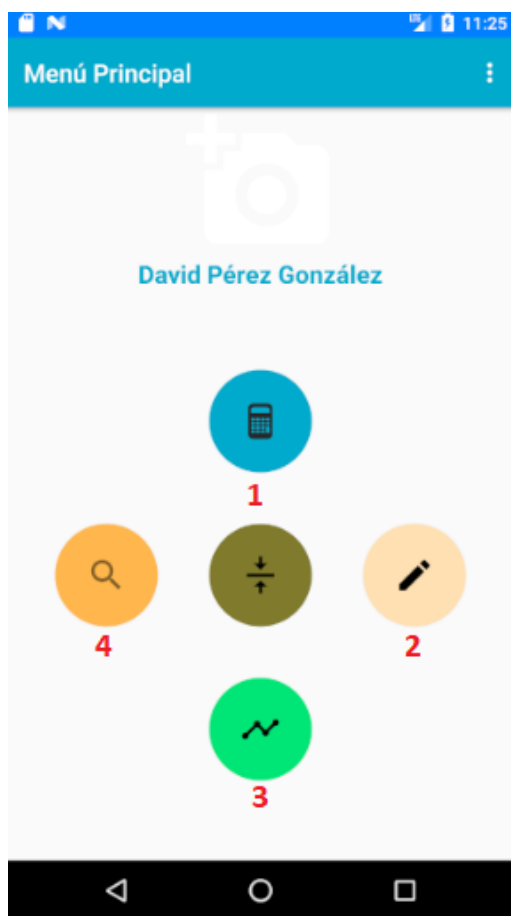


Figura E.4: Menú principal

Registro de glucemias

Esta pantalla podemos seleccionar el periodo del día entre 10 opciones (antes y después de cada ingesta) e introducir nuestro valor de glucemia actual. Una vez conformes con los datos, pulsamos en GUARDAR.

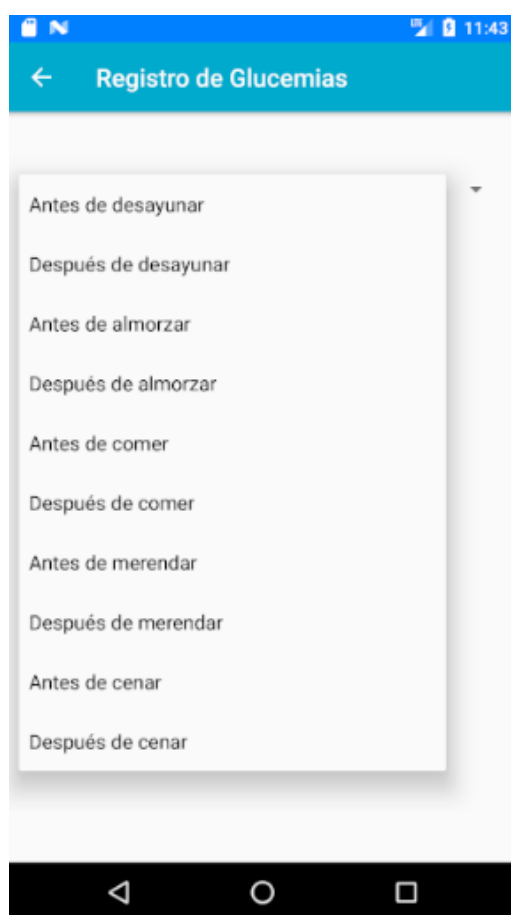


Figura E.5: Registro de glucemias A

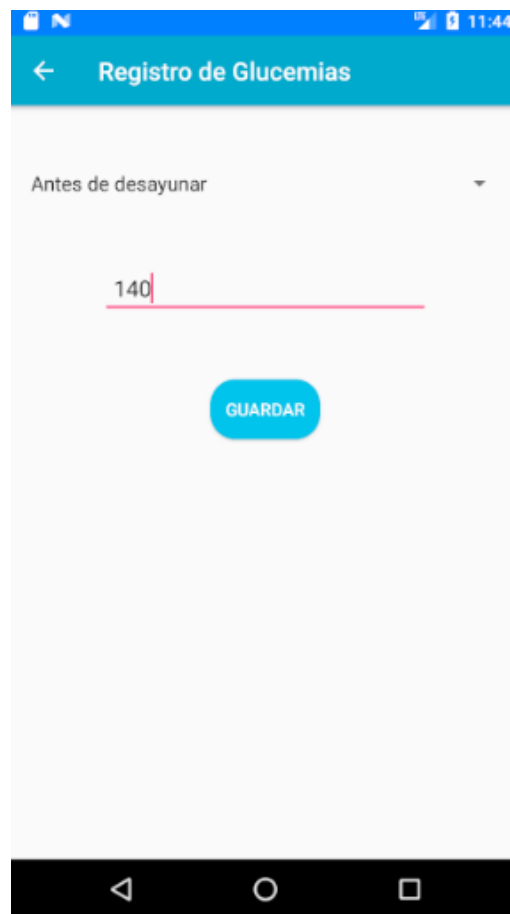


Figura E.6: Registro de glucemias B

Incidencias

En el caso de haber introducido un valor fuera de los límites marcados en el perfil de usuario, pasaremos a esta pantalla en la que podemos seleccionar el tipo de incidencia (si es que sabemos cuál es) e introducir opcionalmente unas observaciones sobre la misma. Los tipos de incidencia mostrados varían según el valor de glucemia guardado este fuera de los límites por exceso o por defecto.

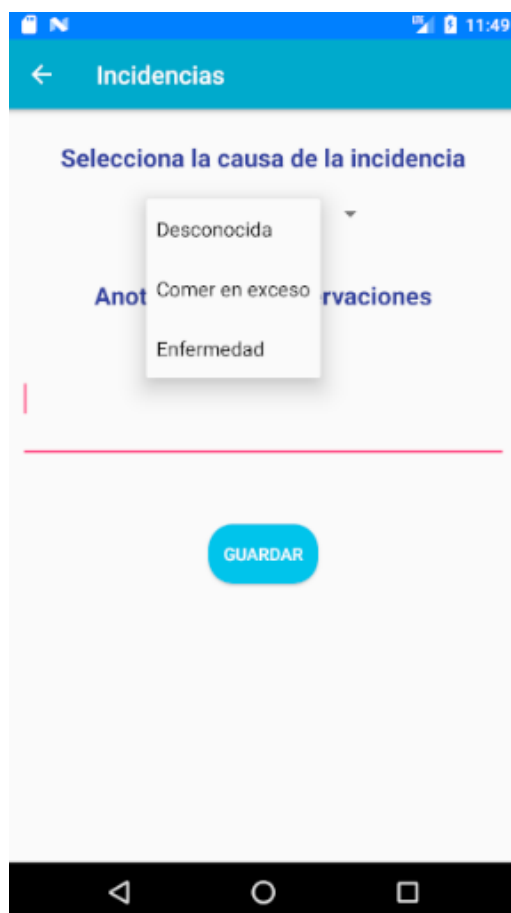


Figura E.7: Incidencias A

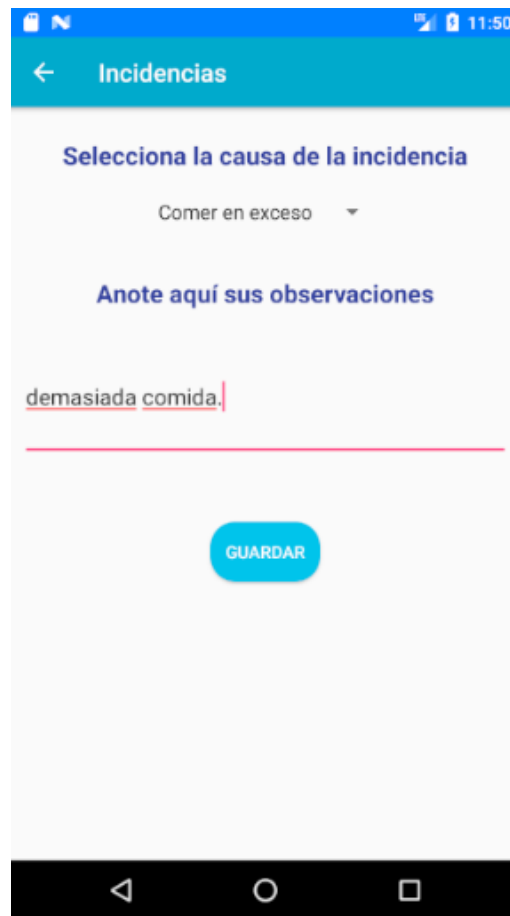


Figura E.8: Incidencias B

Cálculo del bolo corrector

A continuación, se describen los pasos a seguir para el cálculo del bolo corrector.

- Paso 1: Seleccionar la opción **Calcular bolo** del menú principal.
- Paso 2: Registrar el valor de glucemia actual **E.4.**
- Paso 3: Registrar la incidencia si procede **E.4.**
- Paso 4: Introducir los alimentos a ingerir para el recuento de carbohidratos **E.4.**
- Paso 5: Clic en el botón FINALIZAR para mostrar por pantalla el resultado.

Carbohidratos

A continuación, se describen los pasos a seguir para el recuento de carbohidratos con los siguientes alimentos:

Alimentos a ingerir:

- Leche entera, 250 grs.
- Kiwi, 100 grs.

Pasos para introducir los alimentos.

- Paso 1: Buscar el alimento a ingerir en el buscador.
- Paso 2: Seleccionar el alimento de la lista filtrada por el buscador.
- Paso 3: Introducir los gramos a ingerir del alimento seleccionado.
- Paso 4: Añadir el alimento y su cantidad a la lista de ingesta pulsando el botón “+” de color verde.
- Paso 5: Si se desea eliminar un alimento de la lista, se mantiene pulsado el alimento y se pulsa el botón “x” de color rojo.
- Paso 6: Una vez finalizada la lista de alimentos, pulsar el botón FINALIZAR para obtener el bolo corrector. Dependiendo del resultado obtenido, la aplicación mostrará ciertas recomendaciones al usuario.

Carbohidratos

kiwi

Seleccione alimento

Kiwi

Introduce la cantidad del alimento (Grs)

100 +

Lista de Ingesta:

Alimento	Cantidad(gr)	I.G.
Leche entera	250	27

Actual Sum HC: 12.50 Actual Bolo C.:1.00

FINALIZAR

Figura E.9: Carbohidratos A



Figura E.10: Carbohidratos B

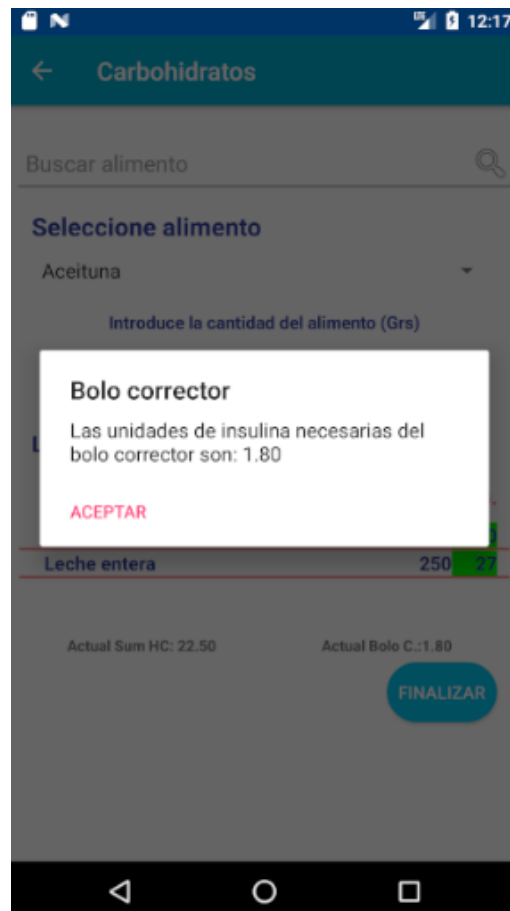


Figura E.11: Carbohidratos C

Historial de glucemias

Si el usuario quiere consultar su historial del mes de Junio, una vez seleccionada dicha opción del menú principal, debe seleccionar el mes de la lista de meses disponible en la pantalla y posteriormente pulsar el botón BUSCAR. A continuación, se mostrará una lista con las gráficas de cada semana del mes donde el usuario ha registrado sus valores de glucemia.

Si el mes seleccionado no tiene registros, la aplicación mostrará un mensaje informando al usuario de ello.

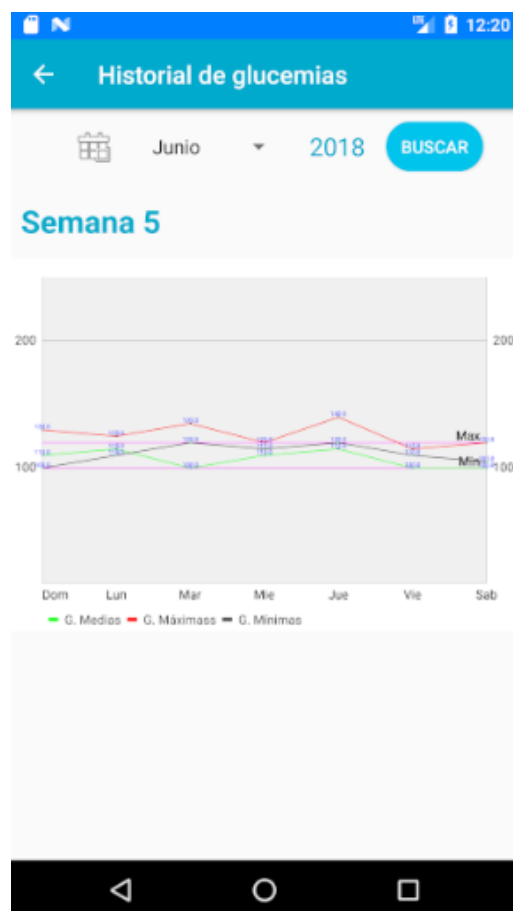


Figura E.12: Historial de glucemias C

Consultar registros

Si el usuario quiere consultar sus registros diarios, simplemente debe seleccionar la cuarta opción del menú principal y podrá ver en una lista (resumida) sus registros diarios relacionados con el cálculo del bolo corrector y los alimentos ingeridos.

Tras esto, el usuario puede consultar los detalles de cada registro seleccionando de la lista el registro a consultar y pulsar el botón “->” (flecha hacia la derecha).



← Listas de Ingestas

Registros diarios

Id	Fecha	HC	Bolo C.
3	26/06/2018 12:17	22.5	1.8
2	26/06/2018 11:32	25	2.4
1	26/06/2018 11:27	22.5	2.33333

→

Figura E.13: Consultar registros



Figura E.14: Consultar detalles de registros

Ajustes

Este sub-menú permite al usuario realizar ciertas configuraciones. Por ejemplo, modificar los datos de su perfil. Para ello deberá ir al sub-menú del menú principal y seleccionar la opción Ajustes. En esta pantalla, seleccionamos la acción que queremos realizar, en nuestro caso, Perfil para modificar nuestros datos. Esto nos llevará a la pantalla de Registro-Perfil [E.4](#). Tras modificar nuestros datos, pulsamos el botón GUARDAR.

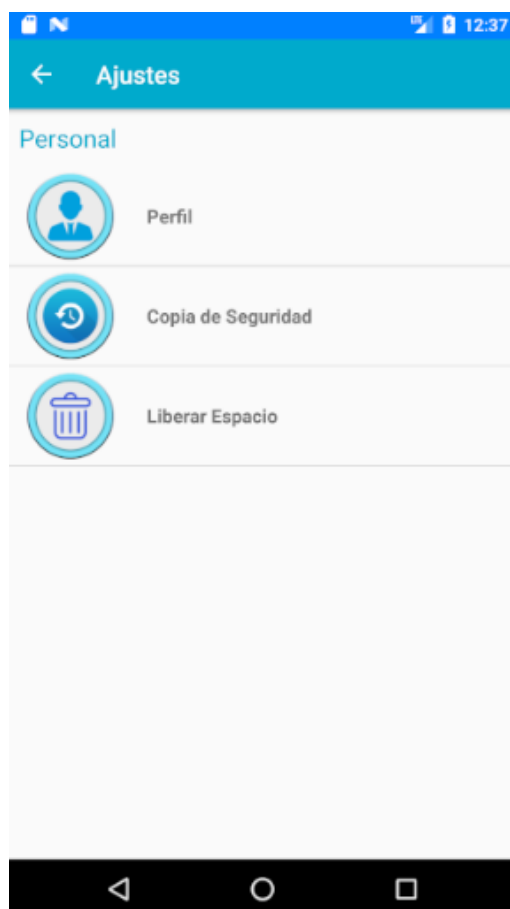


Figura E.15: Ajustes

Acerca de

Esta opción mostrara por pantalla información relacionada con la aplicación.

Bibliografía

- [1] Víctor Bautista. Socialdiabetes. <https://www.socialdiabetes.com/>, 2008.
- [2] Mario López Jiménez. Ubudiabetes: Aplicación de control de diabetes en dispositivos móviles., 2016.
- [3] Ken Schwaber and Mike Beedle. *Agile software development with Scrum*, volume 1. Prentice Hall Upper Saddle River, 2002.
- [4] Unknown. Fundación para la diabetes. <http://www.fundaciondiabetes.org/prensa/297/la-diabetes-en-espana>, Unknown.
- [5] Unknown. Publicar app en google play. https://cincodias.elpais.com/cincodias/2015/02/01/lifestyle/1422792260_243066.html, Unknown.
- [6] Wikipedia. Apache license — wikipedia, la enciclopedia libre, 2017. [Internet; descargado 13-junio-2018].