



UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática



TFG del Grado en Ingeniería Informática
GoBees
Monitorización del estado de una
colmena mediante la cámara de un
smartphone.



Presentado por David Miguel Lozano
en la Universidad de Burgos — 8 de febrero de 2017
Tutores: Dr. José Francisco Díez Pastor
y Dr. Raúl Marticorena Sánchez

Índice general

Índice general	I
Índice de figuras	III
Apéndice A Manuales	1
A.1. Introducción	1
A.2. Planificación temporal	2
A.3. Estudio de viabilidad	16
Apéndice B Especificación de Requisitos	23
B.1. Introducción	23
B.2. Objetivos generales	24
B.3. Catálogo de requisitos	24
B.4. Especificación de requisitos	26
Apéndice C Especificación de diseño	53
C.1. Introducción	53
C.2. Diseño de datos	53
C.3. Diseño arquitectónico	55
C.4. Diseño procedimental	63
C.5. Diseño de interfaces	65
Apéndice D Documentación técnica de programación	68
D.1. Introducción	68
D.2. Estructura de directorios	68
D.3. Manual del programador	69
D.4. Pruebas del sistema	83
Apéndice E Documentación de usuario	89
E.1. Introducción	89

ÍNDICE GENERAL

II

E.2. Requisitos de usuarios	89
E.3. Instalación	89
E.4. Manual de usuario	91
Bibliografía	103

Índice de figuras

A.1. Sprint 0	3
A.2. Sprint 1	4
A.3. Sprint 2	4
A.4. Sprint 3	5
A.5. Sprint 4	6
A.6. Sprint 5	6
A.7. Sprint 6	7
A.8. Sprint 7	8
A.9. Sprint 8	8
A.10.Sprint 9	9
A.11.Sprint 10	9
A.12.Sprint 11	10
A.13.Sprint 12	11
A.14.Sprint 13	11
A.15.Sprint 14	12
A.16.Sprint 15	13
A.17.Sprint 16	13
A.18.Sprint 17	14
A.19.Sprint 18	15
A.20.Porcentaje de horas dedicadas por categoría.	15
A.21.Compatibilidad entre licencias.	19
 B.1. Diagrama de casos de uso.	27
C.1. Diagrama E/R.	54
C.2. Diagrama relacional.	54
C.3. Patrón MVP.	55
C.4. Patrón repositorio.	56
C.5. Arquitectura de la aplicación.	57
C.6. Diagrama de paquetes simplificado.	58

C.7. Paquetes de las diferentes características.	58
C.8. Paquete tipo de una característica.	59
C.9. Diagrama de clases general.	60
C.10. Paquete <i>monitoring</i>	60
C.11. Diagrama de clases del paquete <i>camera</i>	61
C.12. Diagrama de clases del paquete <i>algorithm</i>	61
C.13. Paquete <i>data</i>	62
C.14. Paquete <i>model</i>	62
C.15. Diagrama de clases del paquete <i>source</i>	63
C.16. Diagrama de secuencia del algoritmo.	64
C.17. Prototipos iniciales.	65
C.18. Diseños finales de las interfaces.	66
C.19. Diagrama de navegabilidad.	67
C.20. Paleta de colores.	67
D.1. Android Studio.	70
D.2. Terminal Git Bash.	70
D.3. Variable del sistema de OpenCV.	71
D.4. Clonar repositorio de GitHub.	72
D.5. Importar proyecto en Android Studio.	72
D.6. Dependencias del proyecto.	74
D.7. Compilar proyecto.	75
D.8. TravisCI.	78
D.9. Codecov.	79
D.10. CodeClimate.	80
D.11. Configuración de SonarQube en Gradle.	80
D.12. SonarQube.	81
D.13. VersionEye.	82
D.14. Página web generada con ReadTheDocs.	83
D.15. Test unitarios.	84
D.16. Ejecución de los test unitarios.	84
D.17. Ejecución del test de integración del algoritmo.	85
D.18. Aplicación de etiquetado de fotogramas.	86
D.19. Plataforma de desarrollo del algoritmo.	87
E.1. GoBees en Google Play.	90
E.2. Instalación desde Google Play	91
E.3. Colmenar de muestra.	92
E.4. Añadir colmenar.	93
E.5. Información meteorológica.	95
E.6. Colocación del <i>smartphone</i> en la colmena.	97
E.7. Ajustes de monitorización.	99
E.8. Detalle de la grabación.	100
E.9. Sobre GoBees.	102

Apéndice A

Manuales

A.1. Introducción

La fase de planificación es un punto clave en cualquier proyecto. En esta fase se estima el trabajo, el tiempo y el dinero que va a suponer la realización del proyecto. Para ello, se analiza minuciosamente cada una de las partes que componen el proyecto. Este análisis, además de permitir conocer los recursos necesarios, es de gran ayuda en fases posteriores del desarrollo. En este anexo se detalla todo este proceso.

La fase de planificación se puede dividir a su vez en:

- Planificación temporal.
- Estudio de viabilidad.

En la primera parte, se elabora un calendario o un programa de tiempos. En estos se estima el tiempo necesario para la realización de cada una de las partes del proyecto. Se debe establecer una fecha fija de inicio del proyecto y una fecha de finalización estimada. Teniendo en cuenta el peso de cada una de las tareas y los requisitos que se deben cumplir para poder empezar a trabajar en cada una de ellas.

La segunda parte se centra en la viabilidad del proyecto. El estudio de viabilidad se puede dividir a su vez en dos apartados:

- Viabilidad económica: donde se estiman los costes y los beneficios que puede suponer la realización del proyecto.
- Viabilidad legal: el contexto en el que se ejecuta el proyecto está regulado por una serie de leyes. Se deben analizar todas aquellas que afecten al proyecto. En el caso del software, las licencias y la Ley de Protección de Datos pueden ser los temas más relevantes.

A.2. Planificación temporal

Al inicio del proyecto se planteó utilizar una metodología ágil como Scrum para la gestión del proyecto. Aunque no se ha seguido al 100 % la metodología al tratarse de un proyecto educativo (no éramos un equipo de 4 a 8 personas, no hubo reuniones diarias, etc.), sí que se ha aplicado en líneas generales una filosofía ágil:

- Se aplicó una estrategia de desarrollo incremental a través de iteraciones (*sprints*) y revisiones.
- La duración media de los *sprints* fue de una semana.
- Al finalizar cada *sprint* se entregaba una parte del producto operativo (incremento).
- Se realizaban reuniones de revisión al finalizar cada *sprint* y al mismo tiempo de planificación del nuevo *sprint*.
- En la planificación del *sprint* se generaba una pila de tareas a realizar.
- Estas tareas se estimaban y priorizaban en un tablero *canvas*.
- Para monitorizar el progreso del proyecto se utilizó gráficos *burndown*.

Comentar que la estimación se realizó mediante los *story points* que provee ZenHub y, a su vez, se les asignó una estimación temporal (cota superior) que se recoge en la siguiente tabla:

Story points	Estimación temporal
1	15min
2	45min
3	2h
5	5h
8	12h
13	24h
21	2,5 días
40	1 semana

Tabla A.1: Equivalencia entre *story points* y tiempo.

A continuación se describen los diferentes *sprints* que se han realizado.

Sprint 0 (09/09/16 - 16/09/16)

La reunión de planificación de este *sprint* marcó el comienzo del proyecto. En una reunión previa se había planteado la idea del proyecto a Jose Francisco y este había aceptado tutorizarla. En esta nueva reunión se profundizó en la

idea, se incorporó Raúl Marticorena como cotutor y se plantearon los objetivos del primer *sprint*.

Los objetivos fueron: profundizar y formalizar los objetivos del proyecto, investigar el estado del arte en algoritmos de detección y tracking aplicados a la apicultura, establecer el conjunto de herramientas que conformarían el entorno de desarrollo, la gestión del proyecto y la comunicación del equipo y, por último, realizar un esquema rápido de la aplicación que se deseaba desarrollar.

Las tareas en las que se descompusieron los objetivos se pueden ver en: [Sprint 0](#).

Se estimaron 8 horas de trabajo y se invirtieron finalmente 9,25 horas, completando todas las tareas.

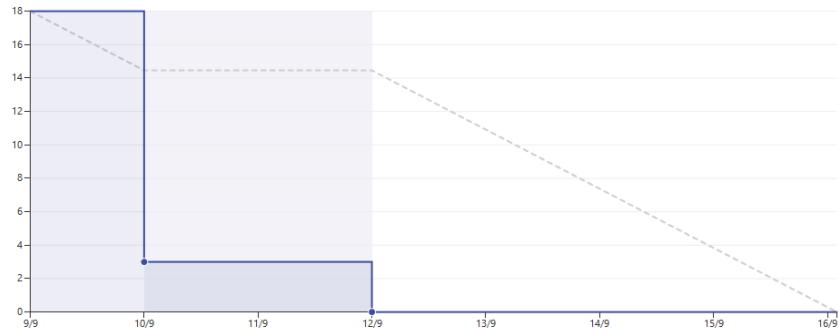


Figura A.1: Sprint 0.

Sprint 1 (17/09/16 - 23/09/16)

Los objetivos de este **sprint** fueron: tomar contacto con OpenCV para Android, realizar un curso *online* de iniciación a Android, investigar qué algoritmos de detección y tracking estaban disponibles en OpenCV para Android y empezar a trabajar en la documentación.

En este *sprint* se tuvo la suerte de hablar sobre el proyecto con Rafael Saracchini, investigador en temas de visión artificial en el Instituto Tecnológico de Castilla y León. Rafael nos propuso una serie de algoritmos que nos podían ser útiles y otros que no funcionarían bajo nuestros requisitos.

Las tareas en las que se descompusieron los objetivos se pueden ver en: [Sprint 1](#).

Se estimaron 7,25 horas de trabajo y se invirtieron finalmente 13,25 horas, completando todas las tareas.

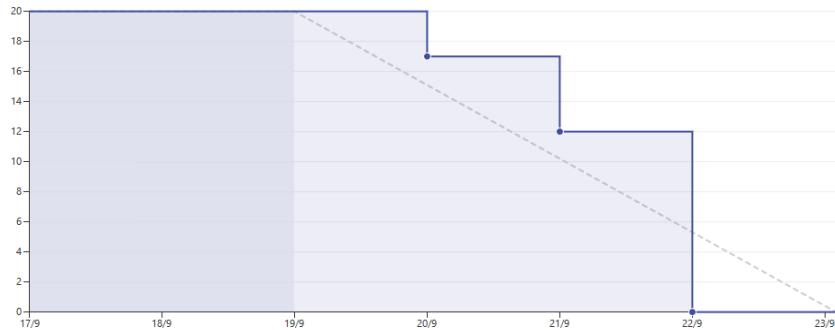


Figura A.2: Sprint 1.

Sprint 2 (24/09/16 - 29/09/16)

Los objetivos de este *sprint* fueron: investigar cómo implementar con OpenCV los algoritmos descritos en el *sprint* anterior, continuar la formación en Android y OpenCV y realizar grabaciones en el colmenar para tener un conjunto de vídeos con los que realizar pruebas.

Las tareas en las que se descompusieron los objetivos se pueden ver en: [Sprint 2](#).

Mientras se realizaba una de las tareas del *sprint*, se encontraron dos *bugs* relacionados con OpenCV y Android ([#26](#) y [#27](#)) que nos impidieron continuar el desarrollo. El investigar su origen y buscar soluciones supuso una gran cantidad de horas y no se lograron resolver hasta el siguiente *sprint*.

Se estimaron 11,75 horas de trabajo y se invirtieron finalmente 33 horas, quedando dos tareas pendientes para terminar durante el siguiente *sprint*.



Figura A.3: Sprint 2.

Sprint 3 (30/09/16 - 06/10/16)

Los objetivos de este *sprint* fueron: intentar resolver los bugs descubiertos en el *sprint* anterior, o si esto fuese imposible, buscar una vía alternativa para continuar el proyecto y continuar investigando las implementaciones de los algoritmos de extracción de fondo en OpenCV.

Las tareas en las que se descompusieron los objetivos se pueden ver en: [Sprint 3](#).

Se estimaron 20,75 horas de trabajo y se invirtieron finalmente 31 horas, quedando una tarea por terminar.

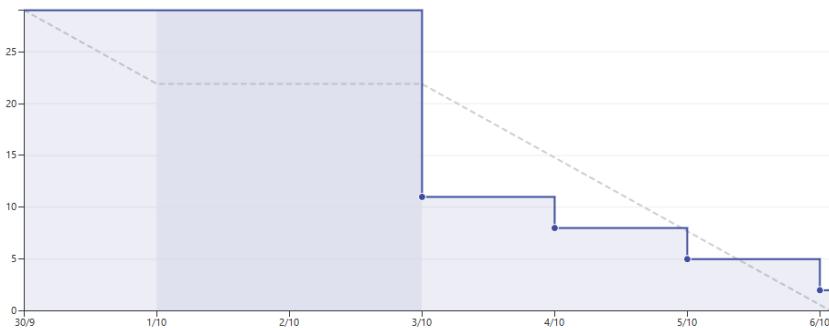


Figura A.4: Sprint 3.

Sprint 4 (07/10/16 - 13/10/16)

Los objetivos de este *sprint* fueron: investigar técnicas de preprocesado y postprocesado para mejorar los resultados de la fase de extracción del fondo. Seleccionar y parametrizar el algoritmo de extracción de fondo que provea los mejores resultados para nuestro problema. Continuar el curso de Android. Integrar los servicios de integración continua y documentación continua en el repositorio.

Las tareas en las que se descompusieron los objetivos se pueden ver en: [Sprint 4](#).

Se estimaron 37 horas de trabajo y se invirtieron finalmente 39,5 horas, completando todas las tareas.

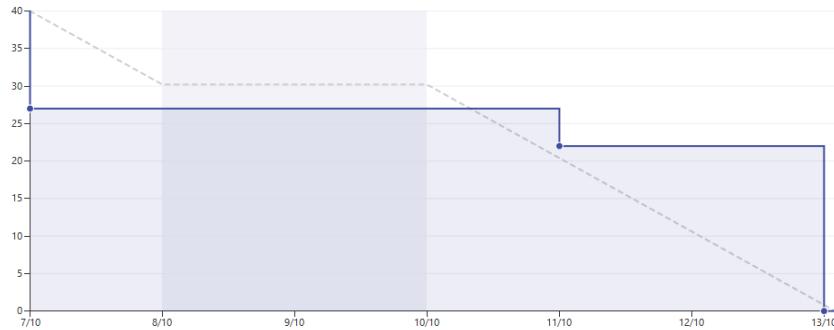


Figura A.5: Sprint 4.

Sprint 5 (14/10/16 - 20/10/16)

Los objetivos de este *sprint* fueron: afinar la parametrización de los algoritmos implementados en el *sprint* anterior. Detectar contornos y contar los pertenecientes a abejas. Pensar algún método que pueda solventar el problema del solapamiento de abejas. Documentar *sprint* anterior. Continuar la formación en Android.

Las tareas en las que se descompusieron los objetivos se pueden ver en: [Sprint 5](#).

Se estimaron 27 horas de trabajo y se invirtieron finalmente 34 horas, completando todas las tareas.

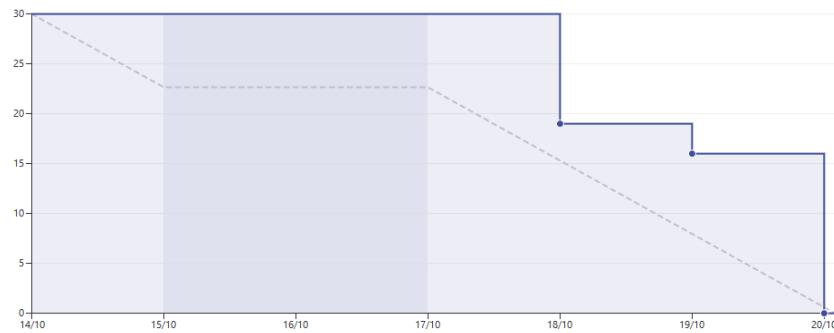


Figura A.6: Sprint 5.

Sprint 6 (21/10/16 - 27/10/16)

Los objetivos de este *sprint* fueron: mudar el algoritmo de visión artificial desarrollado en la plataforma Java a Android. Comenzar a desarrollar una

aplicación de testeo del algoritmo para conocer el error que comete. Investigar si es posible simular el entorno de trabajo filmando a una pantalla.

Las tareas en las que se descompusieron los objetivos se pueden ver en: [Sprint 6](#).

Mientras se mudaba el algoritmo a Android se encontró un *bug* de OpenCV ([#55](#)) que agotaba la memoria del móvil. Este se debía a una mala liberación de recursos por parte de OpenCV y resolvió liberándolos manualmente.

La tarea que más se desvió de su estimación fue la de testeo de los algoritmos. Esto se debió a la dificultad añadida que supuso ejecutar los test unitarios con dependencias de OpenCV en Travis. Finalmente, se solventó instalando OpenCV en la máquina virtual de Travis (compilando desde el código fuente) e inicializando la librería de forma estática (ya que no se deseaba tener que arrancar un emulador para ejecutar los tests unitarios).

Se estimaron 20,75 horas de trabajo y se invirtieron finalmente 41 horas, completando todas las tareas.



Figura A.7: Sprint 6.

Sprint 7 (28/10/16 - 04/11/16)

Los objetivos de este *sprint* fueron: estudiar patrón de arquitectura MVP (*Model-View-Presenter*) y pensar en cómo aplicarlo al proyecto. Diseñar la posible arquitectura de la aplicación. Estudiar el uso de inyección de dependencias en Android con Dagger 2. Documentar las secciones de Introducción y Objetivos.

Las tareas en las que se descompusieron los objetivos se pueden ver en: [Sprint 7](#).

Se estimaron 16 horas de trabajo y se invirtieron finalmente 23 horas, completando todas las tareas.

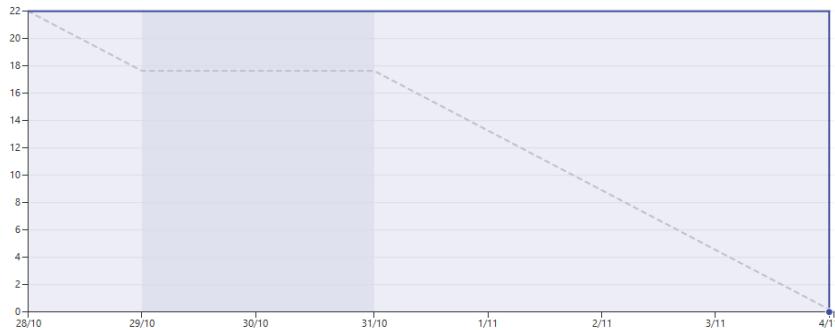


Figura A.8: Sprint 7.

Sprint 8 (05/11/16 - 10/11/16)

Los objetivos de este *sprint* fueron: diseñar el modelo de datos de la aplicación teniendo en cuenta el uso final de estos. Desarrollar una aplicación Java para realizar un conteo manual de un conjunto de frames. Utilizar los datos obtenidos mediante la aplicación de conteo para implementar un test que calcule el error que comete el algoritmo.

Las tareas en las que se descompusieron los objetivos se pueden ver en: [Sprint 8](#).

Se estimaron 46 horas de trabajo y se invirtieron finalmente 53 horas, completando todas las tareas.



Figura A.9: Sprint 8.

Sprint 9 (11/11/16 - 17/11/16)

Los objetivos de este *sprint* fueron: implementar acceso a datos. Inyección de dependencias con los *build variants* de Gradle. Empezar a desarrollar las distintas actividades de la app.

Las tareas en las que se descompusieron los objetivos se pueden ver en: [Sprint 9](#).

Se estimaron 23 horas de trabajo y se invirtieron finalmente 24,25 horas, completando todas las tareas.



Figura A.10: Sprint 9.

Sprint 10 (11/11/16 - 17/11/16)

Los objetivos de este *sprint* fueron: continuar desarrollando las actividades principales de la app. Corregir documentación escrita hasta el momento. Documentar Técnicas y herramientas y Aspectos relevantes.

Las tareas en las que se descompusieron los objetivos se pueden ver en: [Sprint 10](#).

Se estimaron 33,75 horas de trabajo y se invirtieron finalmente 39,25 horas, completando todas las tareas.

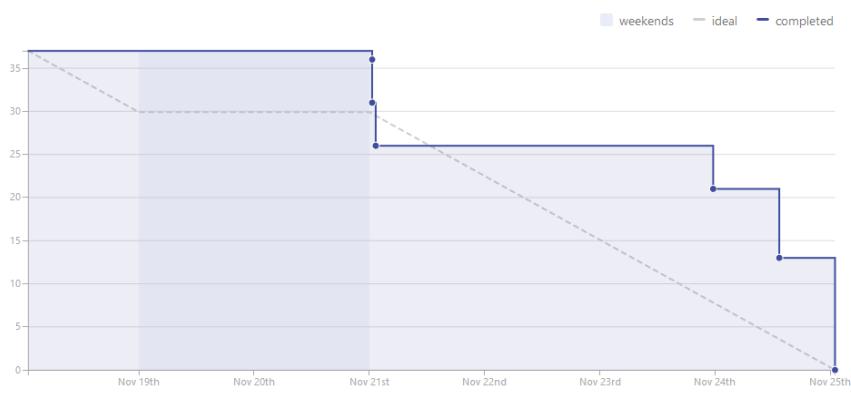


Figura A.11: Sprint 10.

Sprint 11 (26/11/16 - 01/12/16)

Los objetivos de este *sprint* fueron: implementar la vista detalle de una colmena con sus grabaciones, pestañas en las vistas de colmenar y colmena y la sección de ajustes. Corregir los errores en la documentación indicados por los tutores. Continuar la formación en Android.

Las tareas en las que se descompusieron los objetivos se pueden ver en: [Sprint 11](#).

Se estimaron 25,75 horas de trabajo y se invirtieron finalmente 34 horas, completando todas las tareas.



Figura A.12: Sprint 11.

Sprint 12 (02/12/16 - 09/12/16)

Los objetivos de este *sprint* fueron: implementar las partes de visualización de los datos recogidos por la app (gráficos de actividad de vuelo, temperatura, precipitaciones, vientos, etc.) Documentar trabajos relacionados. Empezar a desarrollar la web del producto.

Las tareas en las que se descompusieron los objetivos se pueden ver en: [Sprint 12](#).

Se estimaron 36,25 horas de trabajo y se invirtieron finalmente 50,75 horas, completando todas las tareas.



Figura A.13: Sprint 12.

Sprint 13 (10/12/16 - 14/12/16)

Los objetivos de este *sprint* fueron: agregar opción de localización GPS al añadir colmenar. Incluir una tabla comparativa en la sección Trabajos relacionados.

Las tareas en las que se descompusieron los objetivos se pueden ver en: [Sprint 13](#).

Se estimaron 26,25 horas de trabajo y se invirtieron finalmente 14,25 horas, completando todas las tareas.



Figura A.14: Sprint 13.

Sprint 14 (15/12/16 - 11/01/17)

Se trató del sprint más largo de todos los realizados, con una duración de cuatro semanas debido a las vacaciones de Navidad.

Los objetivos de este *sprint* fueron: implementar el servicio de monitoreo en segundo plano, junto con su sección de ajustes, la obtención de información meteorológica, la edición y borrado de colmenares y colmenas y las pestañas de información de colmenar y colmena. Además, realizar un estudio de viabilidad legal y seleccionar la licencia más apropiada para el proyecto.

Las tareas en las que se descompusieron los objetivos se pueden ver en: [Sprint 14](#).

Se estimaron 143 horas de trabajo y se invirtieron finalmente 187,75 horas, completando todas las tareas.



Figura A.15: Sprint 14.

Sprint 15 (12/01/17 - 18/01/17)

Los objetivos de este *sprint* fueron: finalizar el desarrollo principal de la app completando el menú y la internacionalización. Completar los contenidos de la memoria y continuar con los anexos “Plan del proyecto software” y “Requisitos.”

Las tareas en las que se descompusieron los objetivos se pueden ver en: [Sprint 15](#).

Se estimaron 39 horas de trabajo y se invirtieron finalmente 37,75 horas, a falta de terminar los anexos planificados por falta de tiempo.



Figura A.16: Sprint 15.

Sprint 16 (19/01/17 - 25/01/17)

Los objetivos de este *sprint* fueron: completar las tareas pendientes del anterior sprint (Especificación de requisitos y Análisis económico), documentar el diseño de datos, procedimental y arquitectónico y aumentar la cobertura de los test.

Las tareas en las que se descompusieron los objetivos se pueden ver en: [Sprint 16](#).

Se estimaron 45,75 horas de trabajo y se invirtieron finalmente 45,25 horas, completando todas las tareas.



Figura A.17: Sprint 16.

Sprint 17 (26/01/17 - 02/02/17)

Los objetivos de este *sprint* fueron: continuar anexos. Convertir la memoria a formato LaTeX. Pulir los últimos detalles de la aplicación y publicarla en Google Play.

Las tareas en las que se descompusieron los objetivos se pueden ver en: [Sprint 17](#).

Se estimaron 53,50 horas de trabajo y se invirtieron finalmente 56,50 horas, completando todas las tareas.



Figura A.18: Sprint 17.

Sprint 18 (02/02/17 - 07/02/17)

Los objetivos de este *sprint* fueron: imprimir memoria, terminar anexos y corrección de errores.

Las tareas en las que se descompusieron los objetivos se pueden ver en: [Sprint 18](#).

Se estimaron 41 horas de trabajo y se invirtieron finalmente 41 horas, completando todas las tareas.



Figura A.19: Sprint 18

Resumen

En la siguiente tabla se muestra un resumen del tiempo dedicado a los distintos tipos de tareas.

Categoría	Issues	Tiempo (h)
<i>Bug</i>	26	40,75
<i>Documentation</i>	41	106
<i>Feature</i>	63	410
<i>Research</i>	30	128
<i>Testing</i>	7	49
TOTAL	167	794

Tabla A.2: Desglose de las horas dedicadas al proyecto.

■ Bug ■ Documentation ■ Feature ■ Research ■ Testing

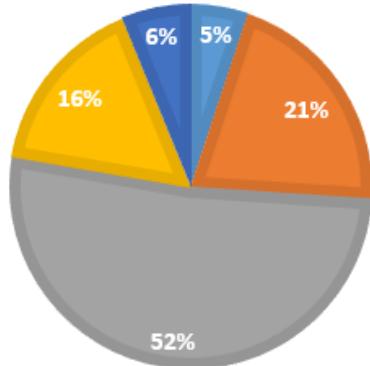


Figura A.20: Porcentaje de horas dedicadas por categoría.

A.3. Estudio de viabilidad

Viabilidad económica

En el siguiente apartado se analizarán los costes y beneficios que podría haber supuesto el proyecto si se hubiese realizado en un entorno empresarial real.

Costes

La estructura de costes del proyecto se puede desglosar en las siguientes categorías.

Costes de personal:

El proyecto ha sido llevado a cabo por un desarrollador empleado a tiempo completo durante cinco meses. Se considera el siguiente salario:

Concepto	Coste
Salario mensual neto	1.000€
Retención IRPF (15 %)	272,23€
Seguridad Social (29,9 %)	542,65€
Salario mensual bruto	1.814,88€
Total 5 meses	9.074,40 €

Tabla A.3: Costes de personal.

La retribución a la Seguridad Social se ha calculado como un 23,60 % por contingencias comunes, más un 5,50 % por desempleo de tipo general, más un 0,20 % para el Fondo de Garantía Salarial y más un 0,60 % de formación profesional. En total un 29,9 % que se aplica al salario bruto [7].

Costes de hardware:

En este apartado se revisan todos los costes en dispositivos *hardware* que se han necesitado para el desarrollo del proyecto. Se considera que la amortización ronda los 5 años y han sido utilizados durante 5 meses.

Concepto	Coste	Coste amortizado
Dispositivo móvil	300€	25€
Ordenador portátil	800€	66,67€
Total	1.100€	91,67€

Tabla A.4: Costes de *hardware*.

Costes de *software*:

En este apartado se revisan todos los costes en licencias de *software* no gratuito. Se considera que la amortización del *software* ronda los 2 años.

Concepto	Coste	Coste amortizado
Windows 10 Pro	279€	58,13€
Createley	5€	1,04€
Total	284€	59,17€

Tabla A.5: Costes de *software*.**Costes varios:**

En este apartado se revisan el resto de costes del proyecto.

Concepto	Coste
Dominio gobees.io	31,90€
Cuenta Google Play	25€
Memoria impresa y cartel	50€
Alquiler de oficina	500€
Internet	150€
Material de apicultura de prueba	150€
Total	906,90€

Tabla A.6: Costes varios.

Costes totales:

El sumatorio de todos los costes es el siguiente:

Concepto	Coste
Personal	9.074,40€
<i>Hardware</i>	91,67€
<i>Software</i>	59,17€
Varios	906,90€
Total	10.132,14€

Tabla A.7: Costes totales.

Beneficios

La aplicación desarrollada se distribuirá de forma gratuita y sin publicidad, por lo que a corto plazo no se obtendrán beneficios.

La forma de monetizar la aplicación será en una segunda fase, cuando se desarrolle una plataforma en la nube que sincronice la información de varios dispositivos y permita el acceso remoto a la información.

Se considerarán tres tipos de suscripciones:

Tipo	Colmenares	Colmenas	Plataformas	Precio
Hobby	1	10	App / Cloud	Gratis
Amateur	5	100	App / Cloud	5€/mes
Profesional	Ilimitados	Ilimitados	App / Cloud	20€/mes

Tabla A.8: Tipos de suscripciones.

Viabilidad legal

En esta sección se discutirán los temas relacionados con las licencias. Tanto del propio *software*, como de su documentación, imágenes y videos.

“En Derecho, una licencia es un contrato mediante el cual una persona recibe de otra el derecho de uso, de copia, de distribución, de estudio y de modificación (en el caso del *Software Libre*) de varios de sus bienes, normalmente de carácter no tangible o intelectual, pudiendo darse a cambio del pago de un monto determinado por el uso de los mismos.” [31]

Software

En primer lugar, vamos a analizar cuál sería la licencia más conveniente para nuestro proyecto. Por un lado, somos nosotros los que podemos elegir qué derechos queremos proporcionar a los usuarios y cuáles no. Sin embargo, estamos limitados por los derechos que nos conceden a nosotros las licencias de las dependencias utilizadas en el proyecto.

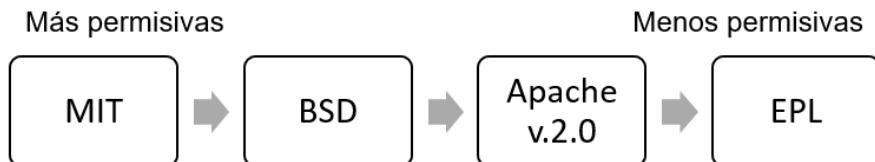
A continuación, se muestran las licencias de las dependencias usadas.

Dependencia	Versión	Descripción	Licencia
Android Support Library	25.1.0	Biblioteca de compatibilidad de Android.	Apache v2.0
OpenCV	3.1.0	Biblioteca de visión artificial.	BSD
Google Play Services	10.0.1	Biblioteca que proporciona acceso a diferentes servicios, entre ellos, localización.	Apache v2.0

Dependencia	Versión	Descripción	Licencia
Guava	20.0	Conjunto de bibliotecas comunes para Java.	Apache v2.0
RoundedImage	2.3.0	Componente para mostrar imágenes redondeadas en Android.	Apache v2.0
MPChart	3.0.1	Biblioteca de gráficos para Android.	Apache v2.0
VNTPicker	1.0.0	Componente para seleccionar valores numéricos.	Apache v2.0
Preference			
Permission	1.0.6	Biblioteca que facilita la gestión de permisos en tiempo de ejecución.	MIT
Utils			
JUnit	4.12	Framework para <i>testing</i> unitario en Java.	EPL
Mockito	2.0.2	Framework para <i>mocking</i> en Java.	MIT
SLF4J	1.7.21	API para <i>logging</i> en Java.	MIT
Apache Log4j	1.7.21	Biblioteca para <i>logging</i> en Java.	Apache v2.0
Android	20160810	Biblioteca para trabajar con JSON.	Apache v2.0
JSON			
Espresso	2.2.2	Framework de <i>testing</i> para Android.	Apache v2.0

Tabla A.9: Dependencias del proyecto.

Por lo tanto, tenemos que escoger una licencia para nuestro proyecto que sea compatible con Apache v2.0, BSD, MIT y EPL. En el siguiente gráfico mostramos la compatibilidad entre estas licencias, así como su grado de permisividad.



*El sentido de la flecha indica compatibilidad.

Figura A.21: Compatibilidad entre licencias.

Podemos observar que la licencia más restrictiva (en el sentido de obligaciones a cumplir) es la *Eclipse Public License* que posee la librería JUnit.

La forma de monetización del proyecto se realizará mediante suscripciones a una plataforma *cloud* que permitirá la sincronización entre varios dispositivos, entre otras funcionalidades. Por lo tanto, la liberación del código del

proyecto no pone en peligro su monetización, sino todo lo contrario, abre la puerta a que la comunidad *Open Source* aporte valor adicional a nuestro proyecto. El permitir la distribución de la app libremente y de forma gratuita también nos es beneficioso, ya que aumenta las posibilidades de recibir nuevas suscripciones de usuarios. Y por último, no nos importaría que otras empresas se basaran en nuestro código fuente para desarrollar sus productos, siempre los liberaran bajo una licencia de código abierto para que nosotros también pudiéramos aprovechar las mejoras que hubieran realizado.

Teniendo en cuenta todo lo anterior, la licencia que más se ajusta a nuestras pretensiones es la *GNU General Public License v3.0*, que, de forma resumida, establece lo siguiente: [9]

Derechos	Condiciones	Limitaciones
Uso comercial.	Liberar código fuente.	Limitación de responsabilidad.
Distribución.	Nota sobre la licencia y copyright.	Sin garantías.
Modificación.	Modificaciones bajo la misma licencia.	
Uso de patentes.	Indicar modificaciones realizadas.	
Uso privado.		

Tabla A.10: Resumen de la licencia GPLv3.

Sin embargo, GPL v3.0 no es compatible con la licencia EPL que posee JUnit. Ya que, la EPL requiere que “cualquier distribución del trabajo conceda a todos los destinatarios una licencia para las patentes que pudieran tener que cubrir las modificaciones que han hecho” [8]. Esto supone que los destinatarios pueden añadir una restricción adicional, hecho que prohíbe rotundamente GPL: “[que el distribuidor] no imponga ninguna restricción más sobre el ejercicio de los derechos concedidos a los beneficiarios” [9].

Tras analizar otras licencias alternativas, no se ha encontrado ninguna compatible con EPL y, a la vez, con nuestras pretensiones. Por lo que finalmente se ha tomado la decisión de utilizar dos licencias para el código fuente del proyecto. Por un lado, todo el código fuente de la aplicación se ha licenciado bajo GPL v3.0. Mientras que el código fuente de testeo, que hace uso de código licenciado bajo EPL (JUnit), se ha liberado bajo licencia Apache v2.0, la cual sí que es compatible con EPL.

Documentación

Aunque se puede utilizar también la licencia GPL v3.0 para licenciar la documentación, no es lo más recomendable. Ya que contiene numerosas cláusulas que solo tienen sentido cuando se habla de código fuente. Por ejemplo, si alguien quisiese distribuir una copia de la documentación de forma impresa, estaría obligado a proporcionar también una copia del código fuente.

Por lo que se ha decidido utilizar una licencia *Creative Commons*, las cuales están más enfocadas a licenciar este tipo de material. En concreto, se ha elegido la *Creative Commons Attribution 4.0 International* (CC-BY-4.0). Que establece lo siguiente: [6]

Derechos	Condiciones	Limitaciones
Uso comercial.	Nota sobre la licencia y copyright.	Limitación de responsabilidad.
Distribución.	Indicar modificaciones realizadas.	Sin garantías.
Modificación.		No proporciona derechos sobre marcas registradas.
Uso privado.		No proporciona derechos sobre patentes.

Tabla A.11: Resumen de la licencia CC-BY-4.0.

Imágenes y vídeos

En la documentación no se ha utilizado ninguna imagen de terceros, todas las imágenes son propias del proyecto y cuentan con la misma licencia que la documentación (CC-BY-4.0).

El *dataset* de vídeos de prueba también se encuentra bajo la misma licencia.

Por otro lado, en la aplicación se han utilizado dos fuentes de imágenes de terceros:

Fuente	Descripción	Licencia
Material design icons	Conjunto de iconos oficial de Google.	Apache v2.0
Simple Weather Icons	Conjunto de iconos meteorológicos.	Apache v2.0

Tabla A.12: Fuentes de imágenes de terceros.

Aunque ambos autores renuncian a la obligación de especificar explícitamente su autoría, se les ha mencionado en la sección ‘Licencias de software

libre” de la aplicación.

El resto de imágenes y gráficos utilizados son de autoría propia y se distribuyen también bajo CC-BY-4.0.3.

Resumen

En la siguiente tabla se resumen las licencias que posee el proyecto.

Recurso	Licencia
Código fuente app	GPLv3
Código fuente tests	Apache v2.0
Documentación	CC-BY-4.0
Imágenes	CC-BY-4.0
Vídeos	CC-BY-4.0

Tabla A.13: Resumen de las licencias del proyecto.

Apéndice B

Especificación de Requisitos

B.1. Introducción

Este anexo recoge la especificación de requisitos que define el comportamiento del sistema desarrollado. Posee un doble objetivo: servir como documento contractual entre el cliente y el equipo de desarrollo y como documentación correspondiente al análisis a la aplicación.

Se han seguido las recomendaciones del estándar IEEE 830-1998, que manifiesta que una buena especificación de requisitos *software* debe ser: [16]

- **Completa:** todos los requerimientos deben estar reflejados en ella y todas las referencias deben estar definidas.
- **Consistente:** debe ser coherente con los propios requerimientos y también con otros documentos de especificación.
- **Inequívoca:** la redacción debe ser clara de modo que no se pueda mal interpretar.
- **Correcta:** el software debe cumplir con los requisitos de la especificación.
- **Trazable:** se refiere a la posibilidad de verificar la historia, ubicación o aplicación de un ítem a través de su identificación almacenada y documentada.
- **Priorizable:** los requisitos deben poder organizarse jerárquicamente según su relevancia para el negocio y clasificándolos en esenciales, condicionales y opcionales.
- **Modificable:** aunque todo requerimiento es modificable, se refiere a que debe ser fácilmente modificable.
- **Verificable:** debe existir un método finito sin costo para poder probarlo.

B.2. Objetivos generales

El proyecto persigue los siguientes objetivos generales:

- Desarrollar una aplicación para *smartphones* que permita la monitorización de la actividad de vuelo de una colmena a través de su cámara.
- Facilitar la interpretación de los datos recogidos mediante representaciones gráficas.
- Aportar información extra a los datos de actividad que ayude en la toma de decisiones.
- Almacenar todos los datos generados de forma estructurada y fácilmente accesible.

B.3. Catálogo de requisitos

A continuación, se enumeran los requisitos específicos derivados de los objetivos generales del proyecto.

Requisitos funcionales

- **RF-1 Gestión de colmenares:** la aplicación tiene que ser capaz de gestionar colmenares.
 - **RF-1.1 Añadir colmenar:** el usuario debe poder añadir un nuevo colmenar con un nombre, una localización y unas notas específicas.
 - **RF-1.1.1: Obtener localización:** la aplicación tiene que ser capaz de obtener la localización actual del usuario.
 - **RF-1.2 Editar colmenar:** el usuario debe poder editar la información de un colmenar ya existente.
 - **RF-1.2.1: Obtener localización:** la aplicación tiene que ser capaz de obtener la localización actual del usuario.
 - **RF-1.3 Eliminar colmenar:** el usuario debe poder eliminar un colmenar ya existente junto con toda su información asociada.
 - **RF-1.4 Listar colmenares:** el usuario debe poder listar todos los colmenares existentes.
 - **RF-1.4.1 Obtención de información meteorológica:** la aplicación tiene que ser capaz de obtener la información meteorológica de cada uno de los colmenares.
 - **RF-1.5 Ver colmenar:** el usuario debe poder visualizar toda la información relativa a un determinado colmenar.
 - **RF-1.5.1 Obtención de información meteorológica:** la aplicación tiene que ser capaz de obtener la información meteorológica relativa a un determinado colmenar.

- **RF-2 Gestión de colmenas:** la aplicación tiene que ser capaz de gestionar colmenas.
 - **RF-2.1 Añadir colmena:** el usuario debe poder añadir una nueva colmena con un nombre y unas notas específicas.
 - **RF-2.2 Editar colmena:** el usuario debe poder editar la información de una colmena ya existente.
 - **RF-2.3 Eliminar colmena:** el usuario debe poder eliminar una colmena ya existente junto con toda su información asociada.
 - **RF-2.4 Listar colmenas:** el usuario debe poder listar todas las colmenas existentes en un determinado colmenar.
 - **RF-2.5 Ver colmena:** el usuario debe poder visualizar toda la información relativa a una determinada colmena.
- **RF-3 Gestión de grabaciones:** la aplicación tiene que ser capaz de gestionar grabaciones.
 - **RF-3.1 Añadir grabación:** la aplicación tiene que ser capaz de crear una nueva grabación a partir de los datos de monitorización.
 - **RF-3.2 Eliminar grabación:** el usuario debe poder eliminar una grabación ya existente junto con toda su información asociada.
 - **RF-3.3 Listar grabaciones:** el usuario debe poder listar todas las grabaciones existentes de una determinada colmena.
 - **RF-3.4 Ver grabación:** el usuario debe poder visualizar toda la información relativa a una determinada grabación.
- **RF-4 Monitorización de la actividad de vuelo:** el usuario tiene que ser capaz de monitorizar la actividad de vuelo de una colmena a partir de una determinada parametrización de esta.
 - **RF-4.1 Previsualización:** el usuario debe poder previsualizar la salida del algoritmo de conteo de abejas.
 - **RF-4.2 Configurar monitorización:** el usuario debe poder configurar todos los parámetros relativos a la monitorización.
 - **RF-4.3 Obtención de información meteorológica:** la aplicación tiene que ser capaz de obtener la información meteorológica relativa a un determinado colmenar.
- **RF-5 Configuración de la aplicación:** el usuario debe poder configurar todos los parámetros disponibles en la aplicación, como el idioma o las unidades meteorológicas.
- **RF-6 Ayuda de la aplicación:** el usuario debe poder obtener ayuda sobre cada una de las funcionalidades de la aplicación.
- **RF-7 Información de la aplicación:** el usuario debe poder obtener información sobre la aplicación, compartirla o enviar sugerencias.

Requisitos no funcionales

- **RNF-1 Usabilidad:** la aplicación debe ser intuitiva, con una curva baja de aprendizaje, errores explicativos y adaptada al entorno de trabajo.
- **RNF-2 Rendimiento:** la aplicación tiene que tener unos tiempos de carga y procesado aceptables en un dispositivo móvil de gama media. La pantalla nunca deberá quedar congelada.
- **RNF-3 Capacidad y Escalabilidad:** la aplicación tiene que estar preparada para una recogida de datos continuada y debe permitir la adición de nuevas funcionalidades de forma sencilla.
- **RNF-4 Disponibilidad:** la aplicación debe estar siempre disponible para su uso, independientemente de la localización, la no disponibilidad de internet, o cualquier otro factor.
- **RNF-5 Seguridad:** la aplicación debe gestionar de forma adecuada todos los datos de carácter sensible, como claves, *tokens*, etc.
- **RNF-6 Mantenibilidad:** la aplicación debe ser desarrollada de acuerdo a algún patrón arquitectónico estándar que asegure escalabilidad, portabilidad, testabilidad, etc. Además, tiene que cumplir los estándares de código de Android.
- **RNF-7 Soporte:** la aplicación debe dar soporte a versiones mayores o iguales a Android 4.4 (*KitKat*).
- **RNF-8 Monitorización:** la aplicación debe monitorizar correctamente la actividad de vuelo de una colmena cuando el dispositivo se coloca en posición cenital a la colmena, sobre un soporte estático y con un fondo claro y uniforme.
- **RNF-9 Internacionalización:** la aplicación deberá estar preparada para soportar varios idiomas, localizando textos, unidades de medida, imágenes, etc.

B.4. Especificación de requisitos

En esta sección se mostrará el diagrama de casos de uso resultante y se desarrollará cada uno de ellos.

Diagrama de casos de uso

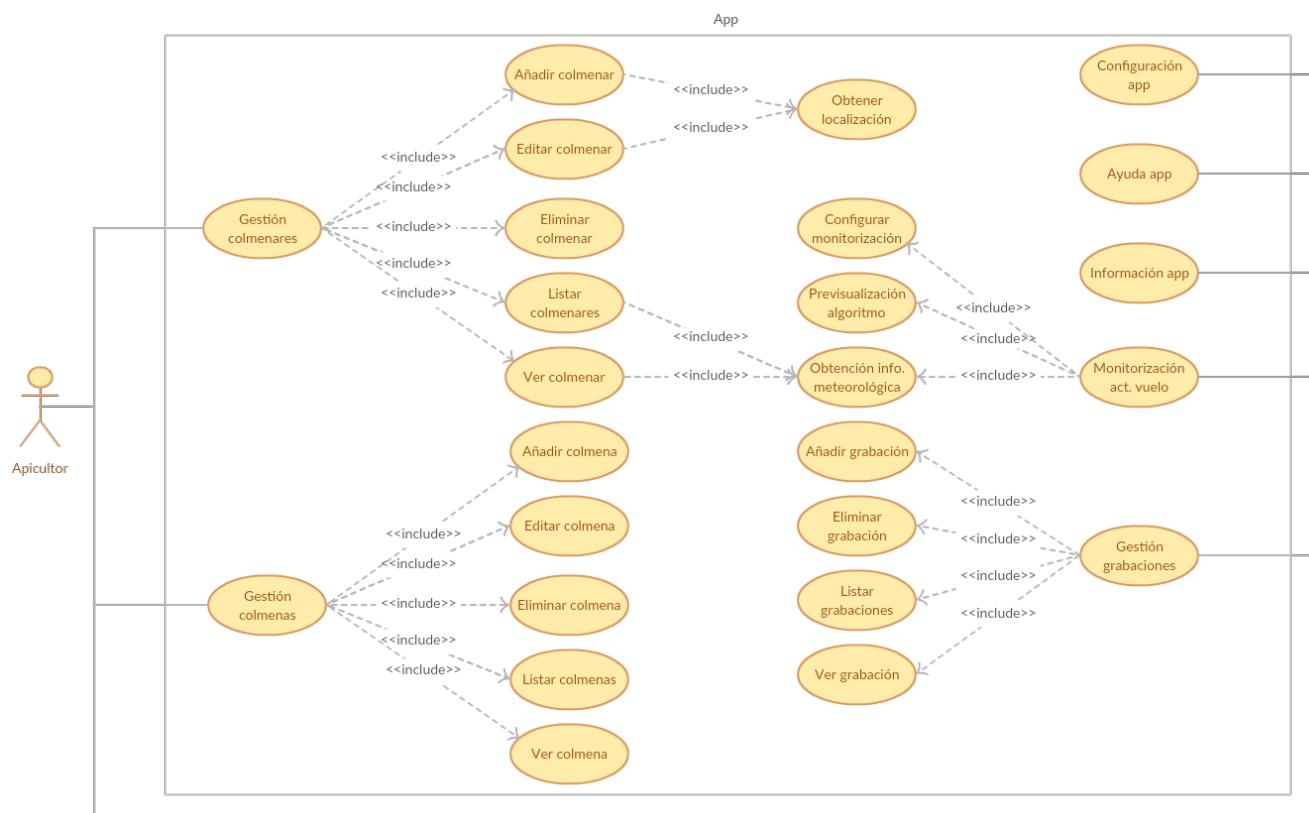


Figura B.1: Diagrama de casos de uso.

Actores

Solo interactuará con el sistema un actor, que se corresponderá con la figura del apicultor.

Casos de uso

CU-01	Gestión de colmenares
Versión	1.0
Autor	David Miguel Lozano
Requisitos asociados	RF-1, RF-1.1, RF-1.1.1, RF-1.2, RF-1.2.1, RF-1.3, RF-1.4, RF-1.5, RF-1.5.1
Descripción	Permite al usuario gestionar sus colmenares.
Precondición	La base de datos se encuentra disponible.
Acciones	<ol style="list-style-type: none"> 1. El usuario entra en la aplicación. 2. Se listan todos los colmenares. 3. Por cada colmenar se da la opción de ver detalle, editar o eliminar. 4. Se muestra un botón para añadir un colmenar.
Postcondición	El número de colmenares listado es igual al número de colmenares en la base de datos.
Excepciones	<ul style="list-style-type: none"> ■ Error al cargar colmenares (mensaje). ■ No existe ningún colmenar (vista especial).
Importancia	Alta

Tabla B.1: CU-01 Gestión de colmenares.

CU-02	Añadir colmenar
Versión	1.0
Autor	David Miguel Lozano
Requisitos asociados	RF-1.1, RF-1.1.1
Descripción	Permite al usuario añadir un nuevo colmenar.
Precondición	La base de datos se encuentra disponible.

CU-02	Añadir colmenar
Acciones	<ol style="list-style-type: none"> 1. El usuario presiona en el botón de añadir colmenar. 2. Se muestra el formulario para introducir los datos del colmenar. 3. El usuario introduce el nombre. 4. El usuario pulsa obtener localización (opcional). <ol style="list-style-type: none"> a. Se obtiene la localización del usuario. 5. El usuario introduce notas sobre el colmenar (opcional). 6. El usuario pulsa el botón de aceptar. 7. Si no hay ningún error, se guarda un nuevo colmenar con los datos introducidos. 8. Volver a Gestión de colmenares.
Postcondición	Existe un colmenar más en la base de datos.
Excepciones	<ul style="list-style-type: none"> ■ Error al guardar colmenar (mensaje). ■ No se ha introducido nombre del colmenar (resaltar).
Importancia	Alta

Tabla B.2: CU-02 Añadir colmenar.

CU-03	Editar colmenar
Versión	1.0
Autor	David Miguel Lozano
Requisitos asociados	RF-1.2, RF-1.2.1
Descripción	Permite al usuario editar un colmenar ya existente.
Precondición	La base de datos se encuentra disponible. El colmenar a editar existe.

CU-03	Editar colmenar
Acciones	<ol style="list-style-type: none"> 1. El usuario selecciona un colmenar para editar. 2. Se obtienen los datos del colmenar de la base de datos. 3. Se rellena el formulario de edición con los datos del colmenar. 4. El usuario edita alguno de los campos. 5. Si el usuario pulsa obtener localización. <ol style="list-style-type: none"> a. Se obtiene la localización del usuario. 6. El usuario pulsa el botón aceptar. 7. Si no hay ningún error, se actualiza el colmenar en la base de datos.
Postcondición	La información del colmenar en la base de datos ha sido actualizada.
Excepciones	<ul style="list-style-type: none"> ■ Error al guardar colmenar (mensaje). ■ No se ha introducido nombre del colmenar (resaltar).
Importancia	Alta

Tabla B.3: CU-03 Editar colmenar.

CU-04	Eliminar colmenar
Versión	1.0
Autor	David Miguel Lozano
Requisitos asociados	RF-1.3
Descripción	Permite al usuario eliminar un colmenar ya existente.
Precondición	La base de datos se encuentra disponible. El colmenar a eliminar existe.

CU-04	Eliminar colmenar
Acciones	
	<ol style="list-style-type: none"> 1. El usuario selecciona un colmenar para eliminar. 2. Se eliminan los datos de ese colmenar de la base de datos. 3. Se elimina el colmenar de la vista. 4. Se informa al usuario.
Postcondición	Existe un colmenar menos en la base de datos.
Excepciones	<ul style="list-style-type: none"> ■ Error al eliminar colmenar (mensaje).
Importancia	Alta

Tabla B.4: CU-04 Eliminar colmenar.

CU-05	Listar colmenares
Versión	1.0
Autor	David Miguel Lozano
Requisitos asociados	RF-1.4, RF-1.4.1
Descripción	Permite al usuario listar todos sus colmenares. Por cada colmenar se muestra el nombre, el número de colmenas y la condición meteorológica y temperatura actuales.
Precondición	La base de datos se encuentra disponible.

CU-05	Listar colmenares
--------------	--------------------------

Acciones

1. El usuario accede a Gestionar Colmenares.
2. Se obtienen todos los colmenares de la base de datos.
3. Se actualiza su información meteorológica si no se dispone de esta o la que se dispone es de hace más de 15 minutos.
4. Se muestran la lista de colmenares. Cada elemento de la lista posee el nombre del colmenar, el número de colmenas y la condición meteorológica y temperatura de ese colmenar.

Postcondición

-

Excepciones

- Error al cargar colmenares (mensaje).
- No existen colmenares (vista especial).
- No existe conexión a internet (mensaje).
- Error al recuperar la información meteorológica (mensaje).

Importancia

Alta

Tabla B.5: CU-05 Listar colmenares.

CU-06	Ver colmenar
Versión	1.0
Autor	David Miguel Lozano
Requisitos asociados	RF-1.5, RF-1.5.1
Descripción	Permite al usuario visualizar toda la información relativa a un determinado colmenar existente.
Precondición	La base de datos se encuentra disponible. El colmenar a visualizar existe.

CU-06	Ver colmenar
Acciones	
	<ol style="list-style-type: none"> 1. El usuario selecciona un colmenar para visualizar. 2. Se obtienen los datos del colmenar de la base de datos (incluidas sus colmenas). 3. Se actualiza su información meteorológica si no se dispone de esta o la que se dispone es de hace más de 15 minutos. 4. Se muestra una lista con sus colmenas. 5. Se muestra la información general del colmenar (localización, número de colmenas, última revisión y notas). 6. Se muestra la información meteorológica en detalle.
Postcondición	-
Excepciones	<ul style="list-style-type: none"> ■ Error al cargar colmenar (mensaje). ■ No existe conexión a internet (mensaje). ■ Error al recuperar la información meteorológica (mensaje).
Importancia	Alta
Tabla B.6: CU-06 Ver colmenar.	

CU-07	Obtener localización
Versión	1.0
Autor	David Miguel Lozano
Requisitos asociados	RF-1.1.1, RF-1.2.1
Descripción	Permite obtener la localización actual del usuario.
Precondición	Se poseen permisos de acceso a la localización.

CU-07	Obtener localización
Acciones	<ol style="list-style-type: none"> 1. El usuario selecciona obtener localización actual. 2. La aplicación se conecta al servicio de localización. 3. El servicio de localización va devolviendo ubicaciones, cada vez más precisas. 4. Cuando el usuario considera la localización suficientemente buena, vuelve a presionar el botón de localización para detener la localización. Si no lo hace, se detendrá automáticamente al cambiar de actividad. 5. Se devuelve la localización obtenida.
Postcondición	Las coordenadas devueltas son válidas.
Excepciones	<ul style="list-style-type: none"> ■ No se poseen permisos de localización (solicitar). ■ Error de conexión con el GPS (mensaje).
Importancia	Alta

Tabla B.7: CU-07 Obtener localización.

CU-08	Obtener información meteorológica
Versión	1.0
Autor	David Miguel Lozano
Requisitos asociados	RF-1.4.1, RF-1.5.1
Descripción	Permite obtener la información meteorológica actual en un determinado colmenar.
Precondición	<p>Se poseen permisos de acceso a internet.</p> <p>El colmenar existe y posee localización.</p>

CU-08 Obtener información meteorológica

Acciones

1. El sistema ejecuta la orden de actualizar información meteorológica para un determinado colmenar.
2. Se obtiene la ubicación del colmenar de la base de datos.
3. Se realiza una consulta a la API de *OpenWeatherMap*.
4. Se procesan los datos recibidos.
5. Se devuelven los datos recibidos.

Postcondición La información meteorológica devuelta es válida.

Excepciones

- No se poseen permisos de internet (solicitar).
- El colmenar no tiene localización (ignorar petición).

Importancia Alta

 Tabla B.8: CU-08 Obtener información meteorológica.

CU-09 Gestión de colmenas

Versión	1.0
Autor	David Miguel Lozano
Requisitos asociados	RF-2, RF-2.1, RF-2.2, RF-2.3, RF-2.4, RF-2.5
Descripción	Permite al usuario gestionar las colmenas de un determinado colmenar.
Precondición	La base de datos se encuentra disponible. El colmenar existe.

CU-09	Gestión de colmenas
--------------	----------------------------

Acciones

1. El usuario entra en la vista detalle de un colmenar.
2. Se listan todas las colmenas.
3. Por cada colmena se da la opción de ver detalle, editar o eliminar.
4. Se muestra un botón para añadir una colmena.

Postcondición El número de colmenas listado es igual al número de colmenas de ese colmenar en la base de datos.

Excepciones

- Error al cargar colmenas (mensaje).
- No existe ninguna colmena (vista especial).

Importancia	Alta
--------------------	------

Tabla B.9: CU-09 Gestión de colmenas.

CU-10 Añadir colmena	
Versión	1.0
Autor	David Miguel Lozano
Requisitos asociados	RF-2.1
Descripción	Permite al usuario añadir una nueva colmena.
Precondición	La base de datos se encuentra disponible. El colmenar existe.

CU-10	Añadir colmena
Acciones	<ol style="list-style-type: none"> 1. El usuario presiona en el botón de añadir colmena. 2. Se muestra el formulario para introducir los datos de la colmena. 3. El usuario introduce el nombre. 4. El usuario introduce notas sobre el colmenar (opcional). 5. El usuario pulsa el botón de aceptar. 6. Si no hay ningún error, se guarda una nueva colmena con los datos introducidos y se asocia al colmenar. 7. Volver a Gestión de colmenas.
Postcondición	Existe una colmena más para ese colmenar en la base de datos.
Excepciones	<ul style="list-style-type: none"> ■ Error al guardar colmena (mensaje). ■ No se ha introducido nombre de la colmena (resaltar).
Importancia	Alta

Tabla B.10: CU-10 Añadir colmena.

CU-11	Editar colmena
Versión	1.0
Autor	David Miguel Lozano
Requisitos asociados	RF-2.2
Descripción	Permite al usuario editar una colmena ya existente.
Precondición	<p>La base de datos se encuentra disponible.</p> <p>El colmenar existe.</p>

CU-11	Editar colmena
Acciones	<ol style="list-style-type: none"> 1. El usuario selecciona una colmena para editar. 2. Se obtienen los datos de la colmena de la base de datos. 3. Se rellena el formulario de edición con los datos del colmenar. 4. El usuario edita alguno de los campos. 5. El usuario pulsa el botón aceptar. 6. Si no hay ningún error, se actualiza la colmena en la base de datos.
Postcondición	La información de la colmena en la base de datos ha sido actualizada.
Excepciones	<ul style="list-style-type: none"> ■ Error al guardar colmena (mensaje). ■ No se ha introducido nombre de la colmena (resaltar).
Importancia	Alta

Tabla B.11: CU-11 Editar colmena.

CU-12	Eliminar colmena
Versión	1.0
Autor	David Miguel Lozano
Requisitos asociados	RF-2.3
Descripción	Permite al usuario eliminar una colmena ya existente.
Precondición	<p>La base de datos se encuentra disponible.</p> <p>El colmenar existe.</p> <p>La colmena a eliminar existe.</p>

CU-12	Eliminar colmena
Acciones	
	<ol style="list-style-type: none"> 1. El usuario selecciona una colmena para eliminar. 2. Se eliminan los datos de esa colmena de la base de datos. 3. Se elimina la colmena de la vista. 4. Se informa al usuario.
Postcondición	Existe una colmena menos en ese colmenar en la base de datos.
Excepciones	<ul style="list-style-type: none"> ▪ Error al eliminar colmena (mensaje).
Importancia	Alta

Tabla B.12: CU-12 Eliminar colmena.

CU-13	Listar colmenas
Versión	1.0
Autor	David Miguel Lozano
Requisitos asociados	RF-2.4
Descripción	Permite al usuario listar todas las colmenas de un determinado colmenar. Por cada colmena se muestra el nombre y la fecha de la última revisión.
Precondición	<p>La base de datos se encuentra disponible.</p> <p>El colmenar existe.</p>

CU-13	Listar colmenas
--------------	------------------------

Acciones

1. El usuario accede a Gestionar Colmenas de un determinado colmenar.
2. Se obtienen todas las colmenas de ese colmenar de la base de datos.
3. Se muestran la lista de colmenas. Cada elemento de la lista posee el nombre de la colmena y la fecha de la última revisión.

Postcondición

-

Excepciones

- Error al cargar colmenas (mensaje).
- No existen colmenas (vista especial).

Importancia

Alta

Tabla B.13: CU-13 Listar colmenas.

CU-14	Ver colmena
Versión	1.0
Autor	David Miguel Lozano
Requisitos asociados	RF-2.5
Descripción	Permite al usuario visualizar toda la información relativa a una determinada colmena existente.
Precondición	<p>La base de datos se encuentra disponible.</p> <p>El colmenar existe.</p> <p>La colmena a visualizar existe.</p>

CU-14 **Ver colmena**

Acciones

1. El usuario selecciona una colmena de un determinado colmenar para visualizar.
2. Se obtienen los datos de la colmena de la base de datos (incluidas sus grabaciones).
3. Se muestra una lista con sus grabaciones.
4. Se muestra la información general de la colmena (última revisión y notas).

Postcondición -**Excepciones**

- Error al cargar colmena (mensaje).

Importancia Alta

Tabla B.14: CU-14 Ver colmena.

CU-15 **Gestión de grabaciones**

Versión 1.0**Autor** David Miguel Lozano**Requisitos asociados** RF-3, RF-3.1, RF-3.2, RF-3.3, RF-3.4**Descripción** Permite al usuario gestionar las grabaciones de una determinada colmena.**Precondición** La base de datos se encuentra disponible.

El colmenar y la colmena existen.

Acciones

1. El usuario entra en la vista detalle de una colmena.
2. Se listan todas las grabaciones.
3. Por cada grabación se da la opción de ver detalle o eliminar.
4. Se muestra un botón para iniciar una nueva monitorización.

CU-15	Gestión de grabaciones
Postcondición	El número de grabaciones listado es igual al número de grabaciones de esa colmena en la base de datos.
Excepciones	<ul style="list-style-type: none"> ▪ Error al cargar grabaciones (mensaje). ▪ No existe ninguna grabación (vista especial).
Importancia	Alta

Tabla B.15: CU-15 Gestión de grabaciones.

CU-16	Añadir grabación
Versión	1.0
Autor	David Miguel Lozano
Requisitos asociados	RF-3.1
Descripción	Permite añadir una nueva grabación a partir de los datos recogidos durante la monitorización.
Precondición	<p>La base de datos se encuentra disponible.</p> <p>El colmenar y la colmena existen.</p>
Acciones	<ol style="list-style-type: none"> 1. El usuario presiona el botón de finalizar monitorización. 2. Si no hay ningún error, se guarda una nueva grabación con los datos recogidos durante la monitorización (número de abejas e información meteorológica) y se asocia a la colmena. 3. Volver a Gestión de grabaciones.
Postcondición	Existe una grabación más para esa colmena en la base de datos.

CU-16	Añadir grabación
Excepciones	

- Error al guardar grabación (mensaje).
- Grabación demasiado corta (mensaje).

Importancia	Alta
Tabla B.16: CU-16 Añadir grabación.	

CU-17	Eliminar grabación
Versión	1.0
Autor	David Miguel Lozano
Requisitos asociados	RF-3.2
Descripción	Permite al usuario eliminar una grabación ya existente.
Precondición	<p>La base de datos se encuentra disponible.</p> <p>El colmenar y la colmena existen.</p> <p>La grabación a eliminar existe.</p>
Acciones	<ol style="list-style-type: none"> 1. El usuario selecciona una grabación para eliminar. 2. Se eliminan los datos de esa grabación de la base de datos. 3. Se elimina la grabación de la vista. 4. Se informa al usuario.
Postcondición	Existe una grabación menos en esa colmena en la base de datos.
Excepciones	<ul style="list-style-type: none"> ▪ Error al eliminar grabación (mensaje).
Importancia	Alta

Tabla B.17: CU-17 Eliminar grabación.

CU-18	Listar grabaciones
Versión	1.0
Autor	David Miguel Lozano
Requisitos asociados	RF-3.3
Descripción	Permite al usuario listar todas las grabaciones de una determinada colmena. Por cada grabación se muestra la fecha y una previsualización de la actividad de vuelo.
Precondición	La base de datos se encuentra disponible. El colmenar y la colmena existen.
Acciones	<ol style="list-style-type: none"> 1. El usuario accede a Gestionar Grabaciones de una determinada colmena. 2. Se obtienen todas las grabaciones de esa colmena de la base de datos. 3. Se muestran la lista de grabaciones. Cada elemento de la lista posee la fecha de la grabación y una previsualización de la actividad de vuelo.
Postcondición	-
Excepciones	<ul style="list-style-type: none"> ■ Error al cargar grabaciones (mensaje). ■ No existen grabaciones (vista especial).
Importancia	Alta

Tabla B.18: CU-18 Listar grabaciones.

CU-19	Ver grabación
Versión	1.0
Autor	David Miguel Lozano
Requisitos asociados	RF-3.4

CU-19	Ver grabación
Descripción	Permite al usuario visualizar toda la información (actividad de vuelo, temperatura, precipitaciones y viento) relativa a una determinada grabación existente.
Precondición	<p>La base de datos se encuentra disponible.</p> <p>El colmenar y la colmena existen.</p> <p>La grabación a visualizar existe.</p>
Acciones	<ol style="list-style-type: none"> 1. El usuario selecciona una grabación de una determinada colmena para visualizar. 2. Se obtienen los datos de la grabación de la base de datos (actividad de vuelo, temperatura, precipitaciones y viento). 3. Se muestra un gráfico con la actividad de vuelo. 4. Se muestra un gráfico con la evolución de la temperatura. 5. Se muestra un gráfico con la evolución de las precipitaciones. 6. Se muestra un gráfico con la evolución del viento.
Postcondición	-
Excepciones	<ul style="list-style-type: none"> ■ Error al cargar grabación (mensaje).
Importancia	Alta

Tabla B.19: CU-19 Ver grabación.

CU-20	Monitorizar actividad de vuelo
Versión	1.0
Autor	David Miguel Lozano
Requisitos asociados	RF-4, RF-4.1, RF-4.2, RF-4.3

CU-20	Monitorizar actividad de vuelo
Descripción	Permite al usuario monitorizar la actividad de vuelo de una colmena a partir de una determinada parametrización.
Precondición	Se poseen permisos de cámara. La cámara se encuentra disponible. El colmenar y la colmena existen.

CU-20**Monitorizar actividad de vuelo**

Acciones

1. El usuario pulsa el botón de inicializar nueva monitorización.
2. Se muestra una previsualización de la salida del algoritmo.
3. Si el usuario presiona el botón de configurar:
 - a. Abrir ajustes.
 - b. El usuario realiza los ajustes oportunos.
 - c. Actualizar algoritmo y cámara con los ajustes.
 - d. Volver a la previsualización.
4. Si el usuario presiona el botón de iniciar monitorización:
 - a. Se lanza el servicio de monitorización.
 - b. Se realiza una cuenta atrás de 5 segundos antes de empezar a monitorizar.
 - c. Se inicia la cámara.
 - d. Se consumen los 10 primeros fotogramas para crear el modelo del fondo.
 - e. Se comienza a monitorizar.
5. Por cada fotograma recibido:
 - a. Se convierte a escala de grises.
 - b. Se aplica un desenfoque Gaussiano.
 - c. Se aplica BackgroundSubtractorMOG2.
 - d. Se aplican varias fases de erosión y dilatación.
 - e. Se obtienen los contornos de las regiones en movimiento.
 - f. Se contabilizan como abejas aquellos contornos que cumplen las condiciones.
 - g. Se almacena el resultado.
6. Si el colmenar posee localización:
 - a. Cada 15 minutos, obtener información meteorológica.
 - b. Guardarla en la base de datos asociada al colmenar.
7. Cuando se recibe la orden de finalizar:
 - a. Cerrar la cámara.
 - b. Dejar de consultar información meteorológica.
 - c. Devolver datos recolectados.

CU-20	Monitorizar actividad de vuelo
Postcondición	La grabación tiene más de 5 registros.
Excepciones	<ul style="list-style-type: none"> ■ No se tienen permisos de cámara (solicitar). ■ Error de cámara (cancelar). ■ No existe conexión a internet (no obtener información meteorológica). ■ Error al obtener información meteorológica (ignorar).
Importancia	Alta

Tabla B.20: CU-20 Monitorizar actividad de vuelo.

CU-21	Previsualización del algoritmo
Versión	1.0
Autor	David Miguel Lozano
Requisitos asociados	RF-4.1
Descripción	Permite al usuario previsualizar los resultados que está proporcionando el algoritmo de conteo en tiempo real (visualización de los fotogramas de entrada, la máscara de salida y el número de abejas contadas).
Precondición	<p>Se poseen permisos de cámara.</p> <p>La cámara se encuentra disponible.</p> <p>El colmenar y la colmena existen.</p>
Acciones	<ol style="list-style-type: none"> 1. El usuario pulsa el botón de inicializar nueva monitorización. 2. Se muestra en tiempo real los fotogramas (bien los de entrada del algoritmo o los de salida). 3. Se muestra el número de abejas que contabiliza en cada fotograma analizado. 4. Si el usuario modifica algún ajuste, se actualiza en la previsualización.

CU-21	Previsualización del algoritmo
Postcondición	-
Excepciones	<ul style="list-style-type: none"> ■ No se tienen permisos de cámara (solicitar). ■ Error de cámara (cancelar).
Importancia	Alta

Tabla B.21: CU-21 Previsualización del algoritmo.

CU-22	Configuración de la monitorización
Versión	1.0
Autor	David Miguel Lozano
Requisitos asociados	RF-4.2
Descripción	Permite al usuario configurar todos los parámetros relativos a la monitorización (parámetros del algoritmo y parámetros de la cámara).
Precondición	<p>Se poseen permisos de cámara.</p> <p>La cámara se encuentra disponible.</p> <p>El colmenar y la colmena existen.</p>
Acciones	<ol style="list-style-type: none"> 1. El usuario se encuentra en la pantalla de previsualización y pulsa el botón de ajustes. 2. Se abre una ventana con los diferentes parámetros ajustables (mostrar salida o entrada del algoritmo, modificar tamaño de las regiones, ajustar áreas de una abeja, zoom y frecuencia de muestreo). 3. Cuando el usuario realiza alguna modificación, actualizar instantáneamente ese parámetro en la cámara o en el algoritmo.
Postcondición	-

CU-22	Configuración de la monitorización
Excepciones	

- No se tienen permisos de cámara (solicitar).
- Error de cámara (cancelar).

Importancia	Alta
Tabla B.22: CU-22 Configuración de la monitorización.	

CU-23	Configuración de la aplicación
Versión	1.0
Autor	David Miguel Lozano
Requisitos asociados	RF-5
Descripción	Permite al usuario configurar todos los parámetros disponibles en la aplicación, como el idioma o las unidades meteorológicas o realizar determinadas tareas de mantenimiento.
Precondición	La base de datos se encuentra disponible.
Acciones	<ol style="list-style-type: none"> 1. El usuario presiona el botón de ajustes de aplicación. 2. Se abre una ventana con los diferentes parámetros ajustables. 3. Si el usuario modifica cualquier parámetro, se hace efectiva la nueva configuración al instante.
Postcondición	-
Excepciones	<ul style="list-style-type: none"> ▪ Error al guardar configuración (mensaje).

Importancia	Alta
Tabla B.23: CU-23 Configuración de la aplicación.	

CU-24	Ayuda de la aplicación
Versión	1.0
Autor	David Miguel Lozano
Requisitos asociados	RF-6
Descripción	Permite al usuario obtener ayuda sobre cada una de las funcionalidades de la aplicación.
Precondición	Se dispone de permisos de internet. Se dispone de conexión a internet.
Acciones	<ol style="list-style-type: none"> 1. El usuario presiona el botón de ayuda de aplicación. 2. Se abre una ventana que carga una página web con la ayuda de la aplicación categorizada por acciones.
Postcondición	-
Excepciones	<ul style="list-style-type: none"> ■ No se disponen de permisos de internet (solicitar), ■ No hay conexión a internet (mensaje).
Importancia	Alta

Tabla B.24: CU-24 Ayuda de la aplicación.

CU-25	Información de la aplicación
Versión	1.0
Autor	David Miguel Lozano
Requisitos asociados	RF-7
Descripción	Permite al usuario obtener información sobre la aplicación, compartirla o enviar sugerencias.
Precondición	-

CU-25	Información de la aplicación
Acciones	
	<ol style="list-style-type: none">1. Si el usuario presiona el botón de compartir aplicación, se le muestran los diferentes medios soportados por el dispositivo para compartirla.2. Si el usuario presiona sobre el botón de enviar comentarios, se abre la aplicación de email con la información del destinatario rellenada para que el usuario pueda enviar sus sugerencias.3. Si el usuario presiona sobre el botón acerca de GoBees, se abre una ventana con información sobre la versión, autor, licencia, página web, historial de cambios y librerías utilizadas junto con sus licencias.
Postcondición	-
Excepciones	-
Importancia	Media

Tabla B.25: CU-25 Información de la aplicación.

Apéndice C

Especificación de diseño

C.1. Introducción

En este anexo se define cómo se han resuelto los objetivos y especificaciones expuestos con anterioridad. Define los datos que va a manejar la aplicación, su arquitectura, el diseño de sus interfaces, sus detalles procedimentales, etc.

C.2. Diseño de datos

La aplicación cuenta con las siguientes entidades:

- **Colmenar (Apiary)**: tiene un nombre, una imagen, una localización y unas notas. A su vez, guarda un registro del tiempo meteorológico actual y varios registros del tiempo que hacia cuando se realizaron las grabaciones de sus colmenas.
- **Colmena (Hive)**: tiene un nombre, una imagen y unas notas. A su vez, posee varias grabaciones de distintas monitorizaciones de la colmena.
- **Registro (Record)**: se corresponde a la salida del algoritmo de conteo al analizar un fotograma. Tiene un *timestamp* y el número de abejas que había en el fotograma.
- **Registro meteorológico (MeteoRecord)**: guarda información sobre el estado meteorológico en una localización y un momento dado. Tiene un *timestamp*, la localidad, el código correspondiente a la condición meteorológica, el ícono correspondiente, temperatura, presión, humedad, velocidad y dirección del viento, porcentaje de nubes, precipitaciones, y nieve.

Diagrama E/R

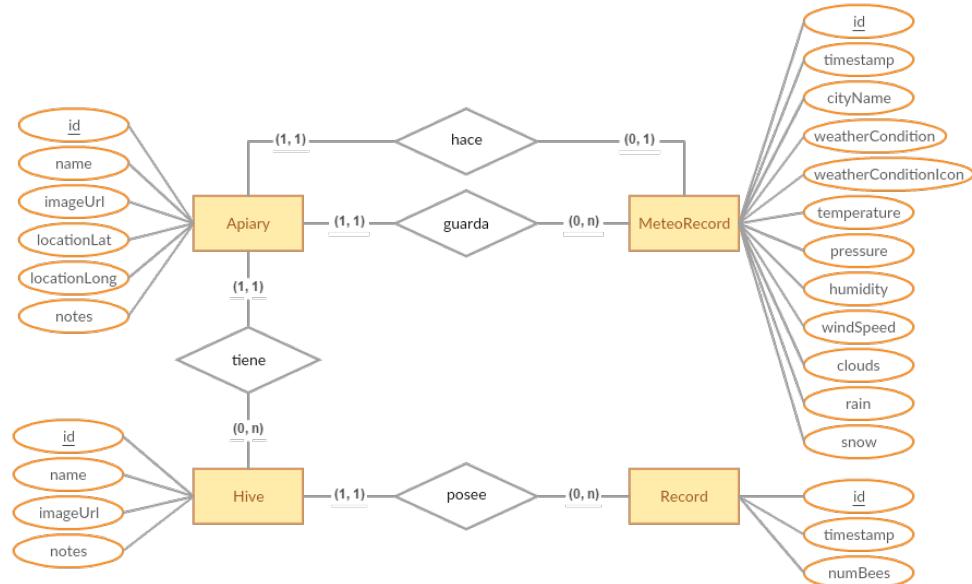


Figura C.1: Diagrama E/R.

Diagrama Relacional

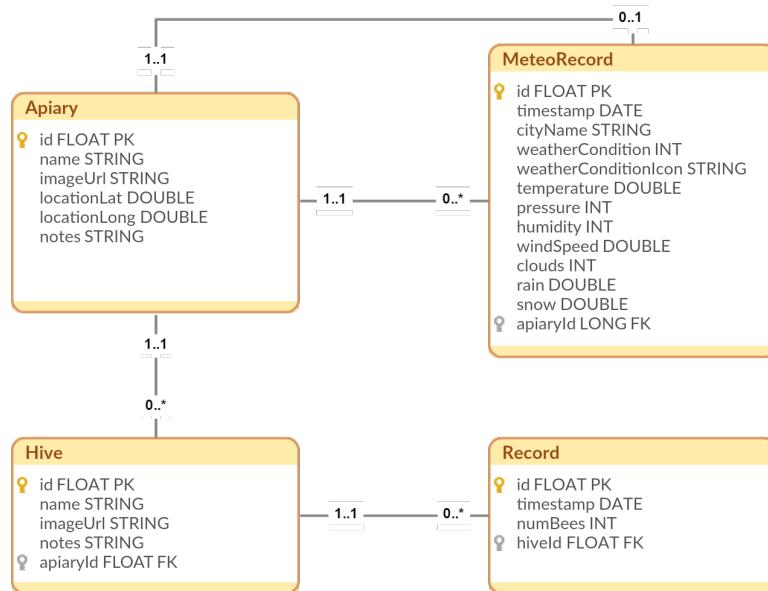


Figura C.2: Diagrama relacional.

C.3. Diseño arquitectónico

El hecho de que el proyecto se haya realizado para la plataforma Android ha condicionado muchas de las decisiones de diseño. Aun así, se han aplicado una serie de patrones para intentar desacoplar el código lo máximo posible y así mejorar su testabilidad y mantenibilidad.

Model-View-Presenter (MVP)

Uno de los patrones arquitectónicos que más relevancia está ganando para el desarrollo de aplicaciones es MVP (*Model-View-Presenter*). Se trata de un patrón derivado del MVC (*Model-View-Controller*) cuyo objetivo es separar la vista del modelo de datos subyacente. MVP introduce la figura del *presenter* que actúa de mediador entre estas dos capas. Su segundo objetivo es maximizar la cantidad de código que se puede testear de forma automática.

MVP divide la aplicación en las siguientes capas:[10]

- *Model*: se corresponde únicamente con el acceso a datos. Se encarga de almacenar y proporcionar los diferentes datos que maneja la aplicación. En nuestra aplicación se corresponde con el Repositorio.
- *View*: se encarga de la visualización de los datos (del modelo). Propaga todas las acciones de usuario al *presenter*. En nuestra aplicación se corresponde con los *Frgments*.
- *Presenter*: enlaza las dos capas anteriores. Sincroniza los datos mostrados en la vista con los almacenados en el modelo y actúa ante los eventos de usuario propagados por la vista. En nuestra aplicación se corresponde con los *Presenters*.



Figura C.3: Patrón MVP.

Existen varias variantes sobre cómo implementar MVP en Android. En nuestro caso, se ha seguido la expuesta Google en Android Architecture Blueprints [18]. En ella se realizan las siguientes consideraciones:

- Se utilizan las *Activity* como controladores globales que se encargan de crear y conectar las vistas con los *presenters*.
- Se utilizan los *Fragment* como vistas ya que proporcionan numerosas ventajas cuando se trabaja con múltiples vistas.

Patrón repositorio

Para la capa del modelo, se ha utilizado el patrón repositorio que proporciona una abstracción de la implementación del acceso a datos con el objetivo de que este sea transparente a la lógica de negocio [21].

En nuestra aplicación existen dos fuentes de datos: por una parte, está la base de datos local implementada con Realm, y por otra, tenemos la API remota que nos da acceso a la información meteorológica. Ambas fuentes son transparentes para los *presenters*.

El repositorio media entre la capa de acceso a datos y la lógica de negocio de tal forma que no existe ninguna dependencia entre ellas. Consiguiendo desacoplar, mantener y testear más fácilmente el código y permitiendo la reutilización del acceso a datos desde cualquier cliente.

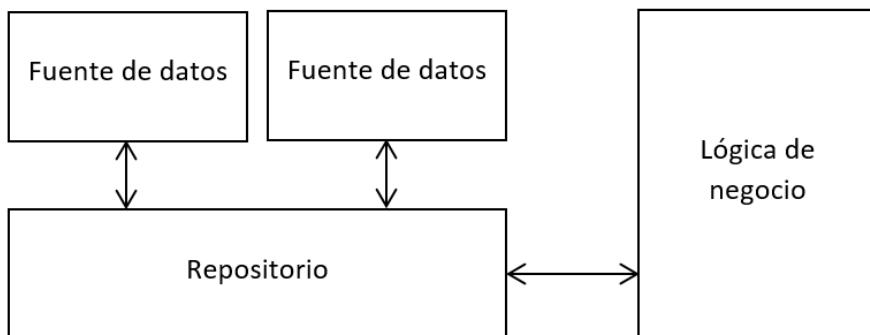


Figura C.4: Patrón repositorio.

Inyección de dependencias

A la hora de testear, es muy frecuente necesitar sustituir la implementación de una clase por otra “falsa” que se comporte de una manera predeterminada para conseguir probar la funcionalidad de manera aislada. En nuestro caso, para facilitar la labor de testeo nos vimos obligados a sustituir la base de datos Realm por una base de datos en memoria. Esta sustitución se realizó mediante la inyección de dependencias.

La inyección de dependencias es un patrón mediante el cual se proporcionan todas las dependencias que una clase necesita para su funcionamiento, en lugar de ser la propia clase quien las cree. Al separar las dependencias de la propia clase, se posibilita la opción de sustituir estas por dobles con un comportamiento definido [30].

Para la implementación de la inyección de dependencias se han utilizado los *build flavors* que proporciona Gradle. Se crearon dos *flavors*:

- **mock:** inyectaba una base de datos en memoria utilizada para el testeo de la aplicación.
- **prod:** inyectaba la base de datos Realm utilizada para producción.

Arquitectura general

El resultado de la arquitectura tras aplicar los patrones explicados es el siguiente:

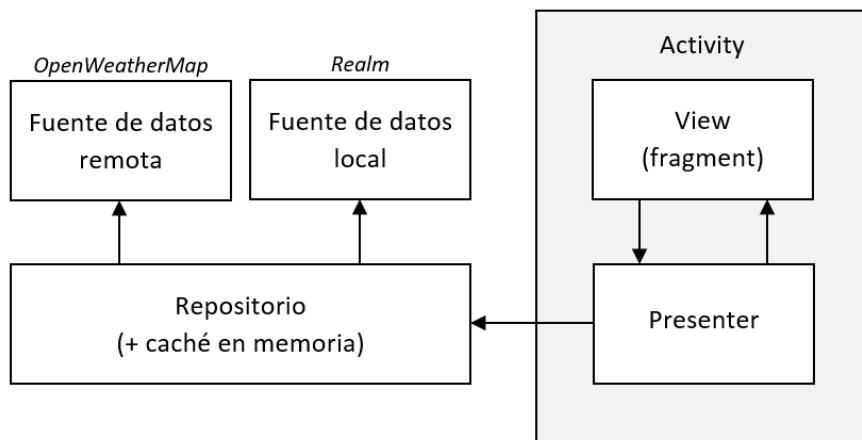


Figura C.5: Arquitectura de la aplicación.

Para agilizar la navegación por la aplicación se implementó una capa de caché en el repositorio.

Diseño de paquetes

Para la organización de los diferentes archivos que componen la aplicación no se utilizó la estrategia convencional de paquete por capa (*package by layer approach*), sino una estrategia de paquete por característica (*package per feature approach*).

Siguiendo esta estrategia se agruparon todos los archivos relacionados cada una de las distintas funcionalidades de la aplicación en un mismo paquete. De esta manera se mejora notablemente la legibilidad y la modularización de la aplicación, ya que se puede modificar cada funcionalidad de forma independiente.

Existen dos paquetes excepcionales que no siguen esta convención:

- Paquete **data**: agrupa toda la capa de modelo.
- Paquete **utils**: reúne un conjunto de clases de utilidad generales que son utilizadas por varias características.

El diagrama de paquetes es el siguiente:

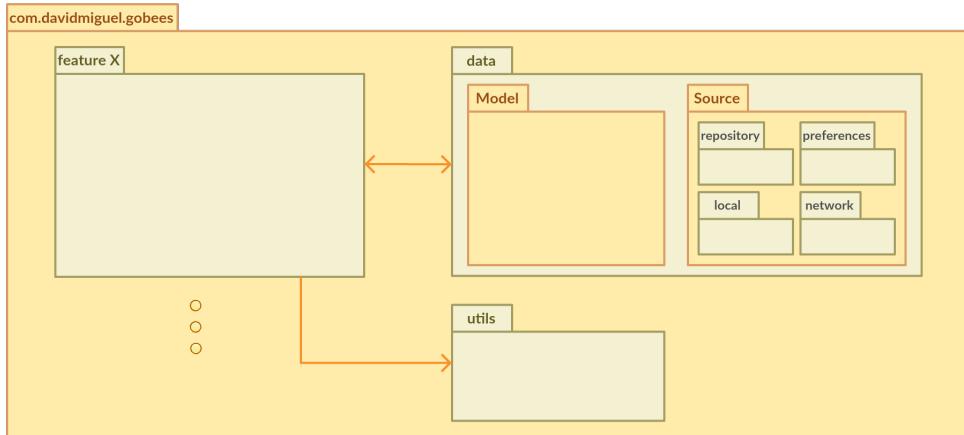


Figura C.6: Diagrama de paquetes simplificado.

El paquete **feature X** se correspondería con cada paquete de cada funcionalidad. Se ha representado de esta manera para simplificar el diagrama.

A continuación, se muestran por separado los paquetes de todas las funcionalidades:

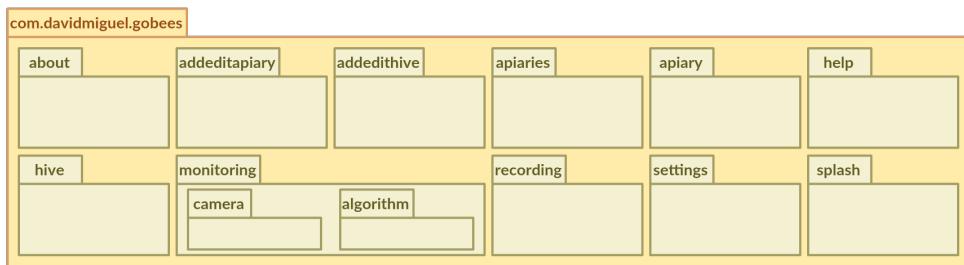


Figura C.7: Paquetes de las diferentes características.

- **about**: contiene la funcionalidad de “Acerca de” de la aplicación. Donde se muestra el autor, licencia, versión de la *app*, sitio web, historial de cambio y todas las dependencias junto con sus licencias.
- **addeditapiary**: permite añadir o editar colmenares.
- **addedithive**: permite añadir o editar colmenas.
- **apiaries**: permite listar los colmenares y gestionarlos.
- **apiary**: permite listar las colmenas de un colmenar, gestionarlas y mostrar la información relativa al colmenar.
- **help**: muestra la ayuda de la aplicación.

- **hive**: permite listar las grabaciones de una colmena, gestionarlas y mostrar la información relativa a la colmena.
- **monitoring**: agrupa toda la funcionalidad de monitorización de la actividad de vuelo de una colmena, desde la configuración hasta la ejecución del algoritmo.
- **recording**: permite visualizar los detalles de una determinada grabación.
- **settings**: permite configurar los distintos parámetros de la aplicación.
- **splash**: muestra una pantalla de inicio mientras la aplicación carga en memoria los recursos necesarios.

Diseño de clases

Aplicando MVP, cada característica clave de la aplicación posee los siguientes componentes:

- **FeatureActivity**: funciona como un controlador global que crea la vista y el *presenter* y los enlaza.
- **FeatureContract**: se trata de una interfaz que establece los siguientes contratos:
 - **FeatureContract.View**: define la capa *view* para esta característica (las únicas funciones que expone a otras capas).
 - **FeatureContract.Presenter**: define la interacción entre las capas *view* y *presenter*. Describe las acciones que pueden ser iniciadas desde la vista.
- **FeatureFragment**: implementación concreta de la capa *view*.
- **FeaturePresenter**: implementación concreta de la capa *presenter*. Escucha las acciones de usuario y actualiza la vista cuando cambia el modelo.

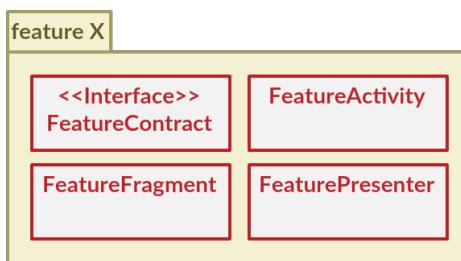


Figura C.8: Paquete tipo de una característica.

El diagrama de clases general que muestra cómo se relacionan todos los componentes de una determinada característica es el siguiente:

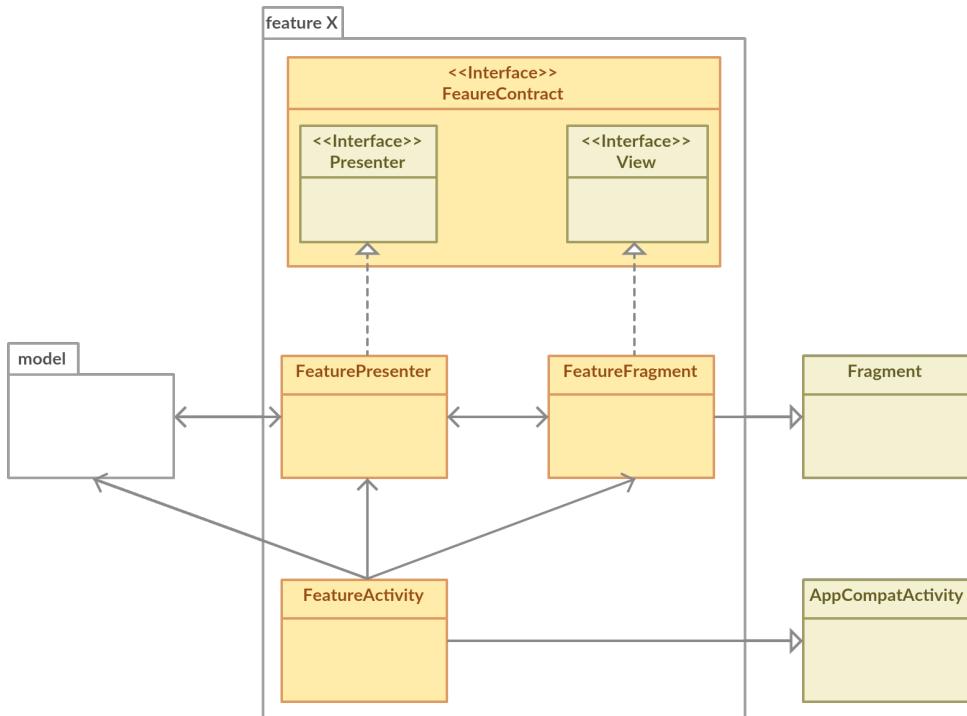
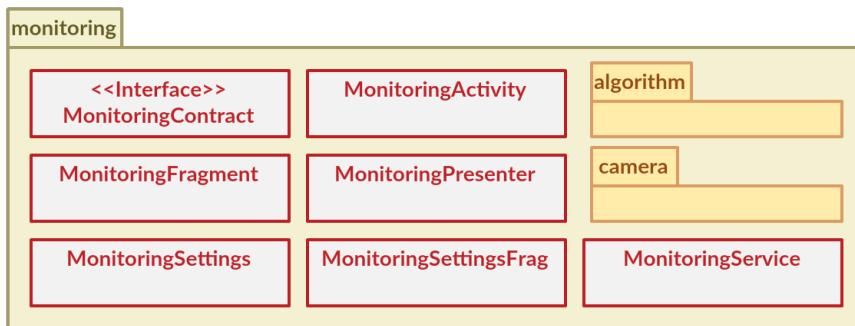
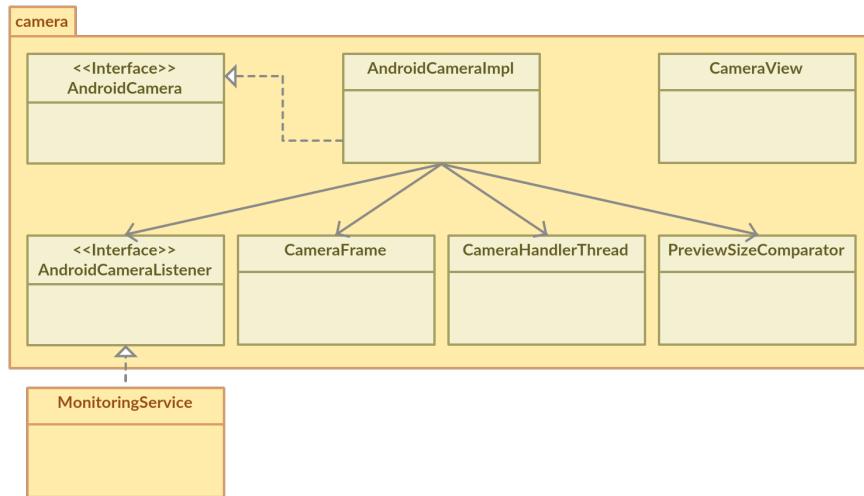


Figura C.9: Diagrama de clases general.

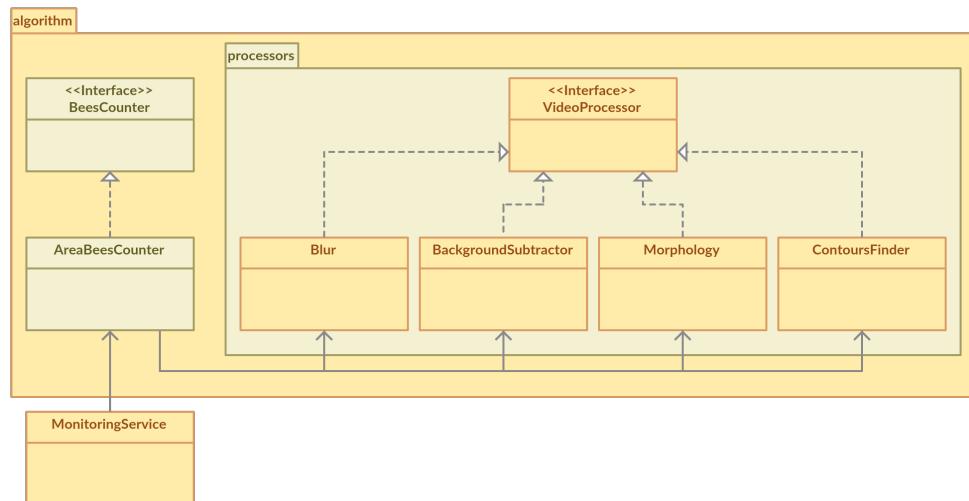
El único paquete que se diferencia de la estructura expuesta es el paquete `monitoring`. Este integra a su vez toda la lógica de acceso a la cámara y todas las clases relacionadas con el algoritmo de conteo.

Figura C.10: Paquete `monitoring`.

El diagrama de clases del paquete `camera` es el siguiente:

Figura C.11: Diagrama de clases del paquete *camera*.

El diagrama de las clases que implementan el algoritmo de conteo es el siguiente:

Figura C.12: Diagrama de clases del paquete *algorithm*.

En la parte del acceso a datos, se poseen dos paquetes como se ha visto en el apartado anterior.

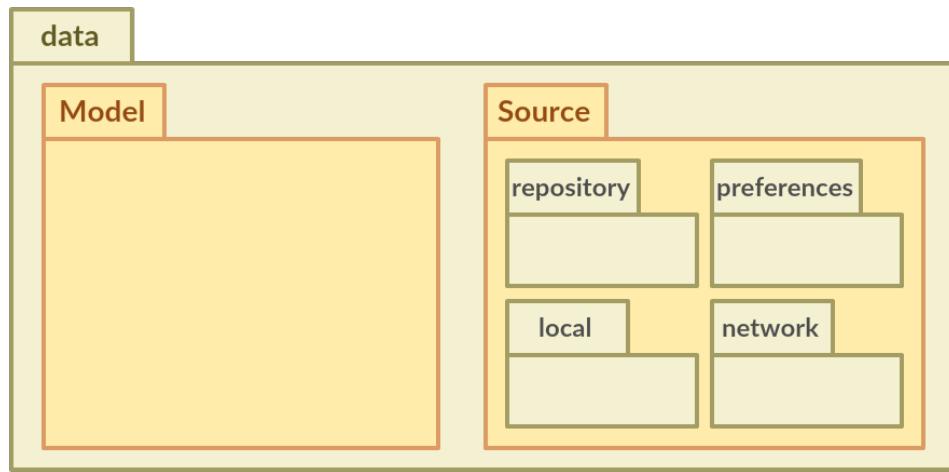


Figura C.13: Paquete *data*.

El paquete `model` contiene todas las clases de modelo que se mapean con la base de datos.

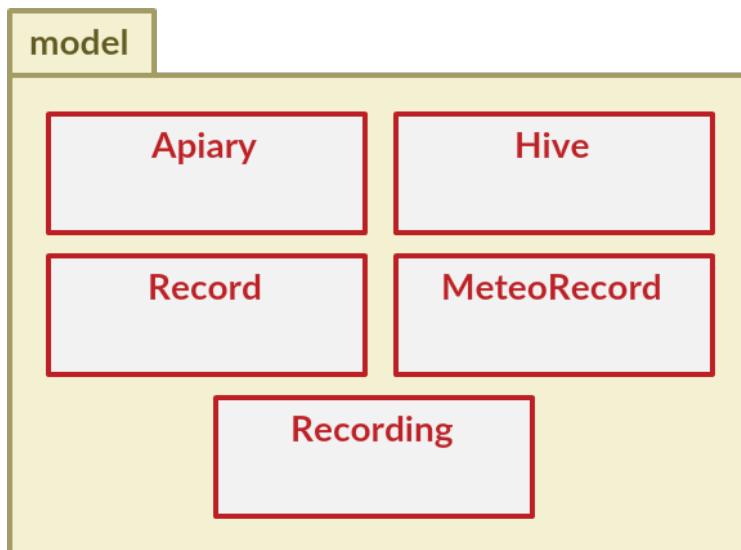


Figura C.14: Paquete *model*.

Nota: la clase `Recording` se utiliza para agrupar a un conjunto de `Records`, pero no se almacena en la base de datos directamente (solo los `Records`).

Por otro lado, el paquete `source` contiene todas las clases correspondientes a los accesos de las diferentes fuentes de datos. Su diagrama de clases es el siguiente:

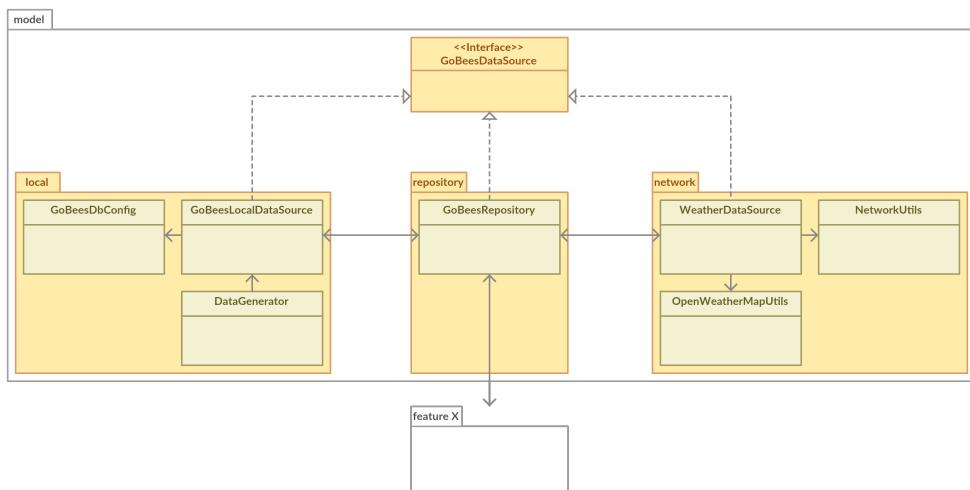


Figura C.15: Diagrama de clases del paquete `source`.

Para conocer a mayor detalle las funciones de cada clase se puede consultar la documentación JavaDoc de la aplicación.

C.4. Diseño procedimental

En este apartado se recogen los detalles más relevantes respecto a la ejecución del algoritmo de monitorización de la actividad de vuelo de una colmena.

En el siguiente diagrama de secuencia se ha representado como es la interacción entre los diferentes objetos que se encargan de la inicialización de la monitorización, la obtención de las imágenes y su posterior procesado por el algoritmo de conteo.

APÉNDICE C. ESPECIFICACIÓN DE DISEÑO

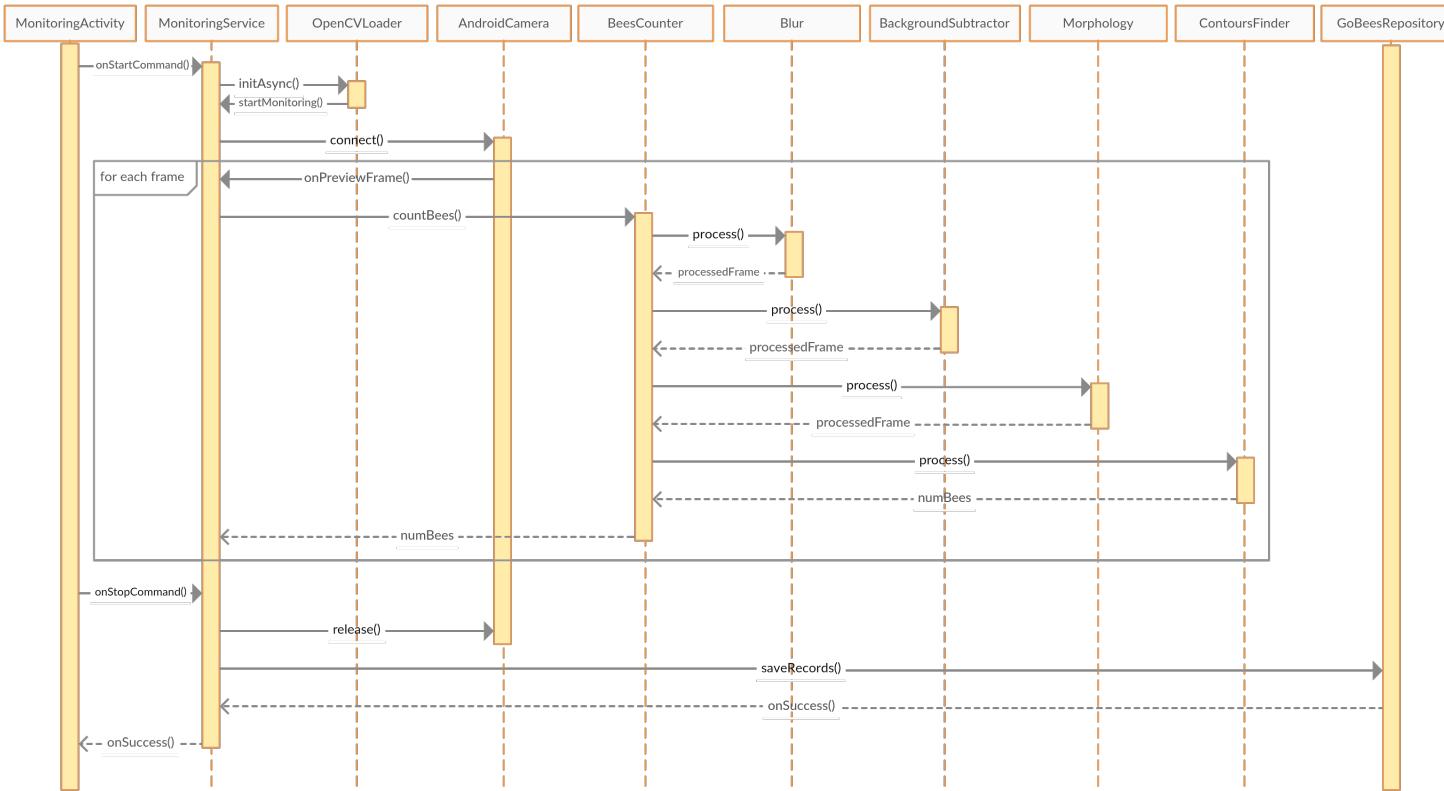


Figura C.16: Diagrama de secuencia del algoritmo.

C.5. Diseño de interfaces

En el diseño de la interfaz se ha seguido la guía de estilos de *Material Design* [13] introducida en el Google I/O 2014 y que se adoptó en Android a partir de la versión 5.0 (*Lollipop*).

En las primeras etapas de proyecto se realizaron una serie de prototipos en los que se plasmaron las principales funcionalidades de la aplicación.

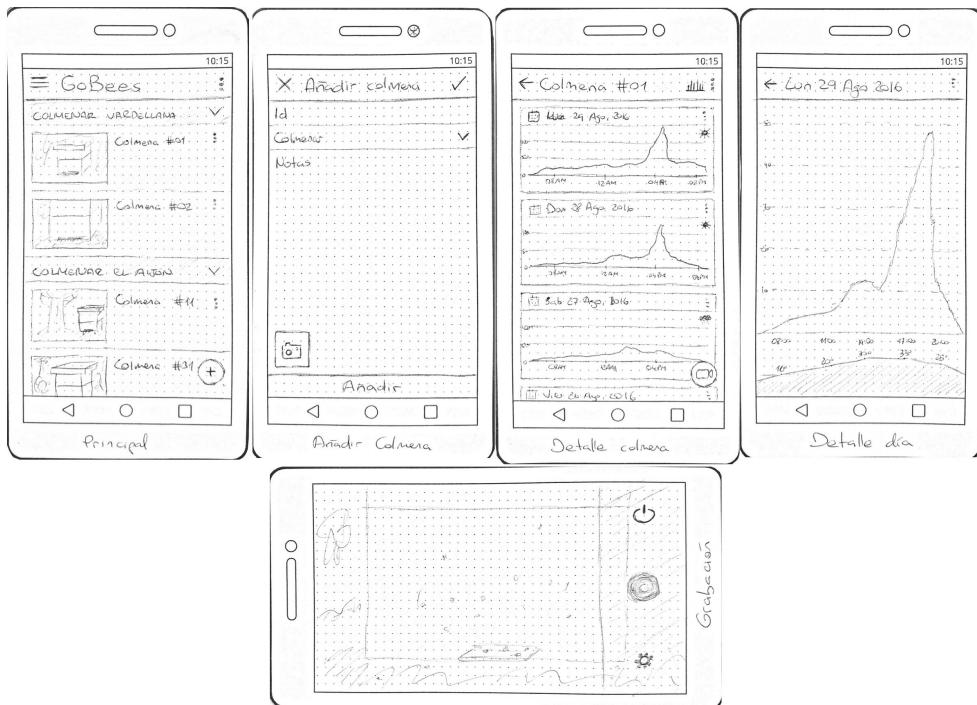


Figura C.17: Prototipos iniciales.

Tras una serie de iteraciones, estos se fueron mejorando hasta obtener las interfaces con las que cuenta hoy en día la *app*.

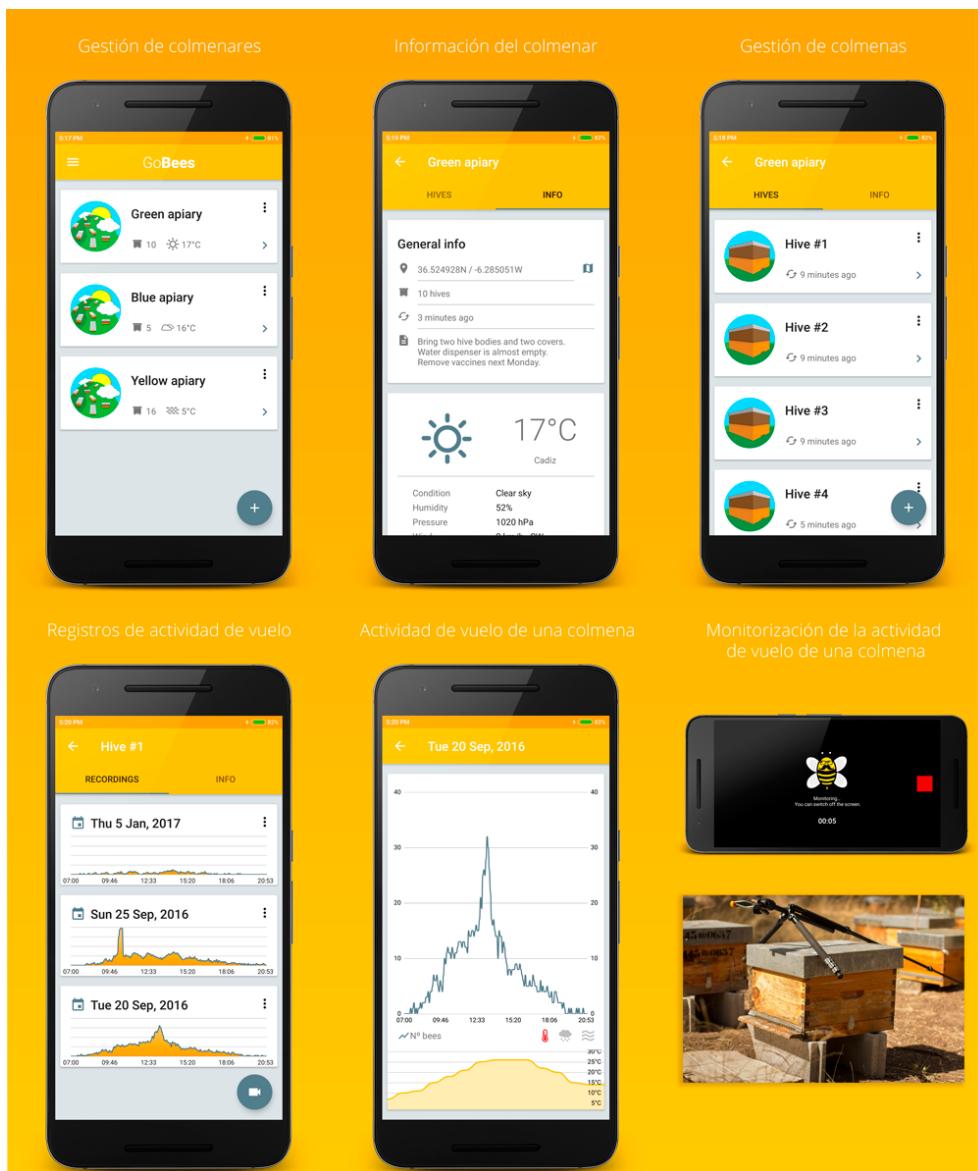


Figura C.18: Diseños finales de las interfaces.

El siguiente diagrama muestra la navegabilidad por la aplicación. Esta ha sido distribuida de acuerdo al tipo de contenido y a las tareas a realizar sobre este.

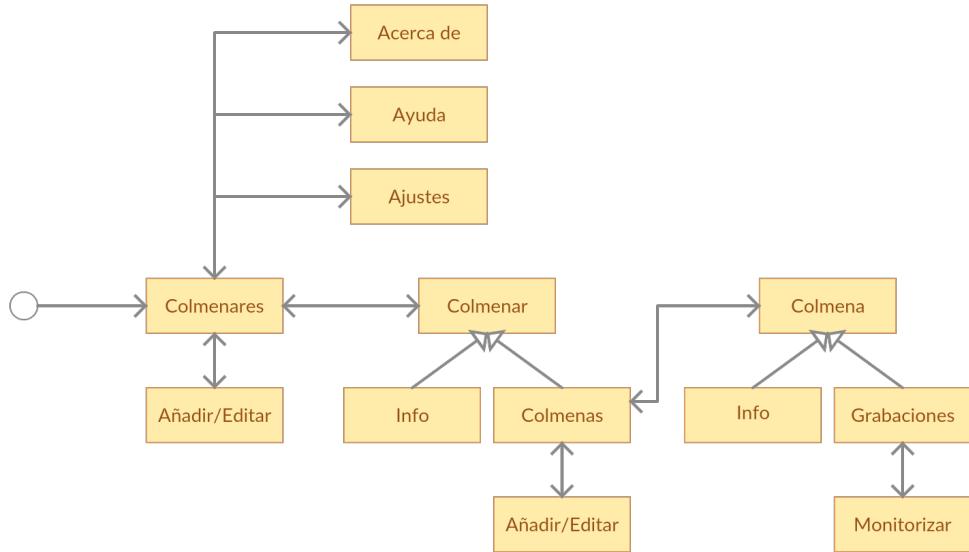


Figura C.19: Diagrama de navegabilidad.

Se ha escogido la paleta de colores entre los recomendados por *Material Design*. Utilizando como principal un color en la gama de los 500, lo que denominan un color *material*, y definiendo otro color que contraste con este para acentuar.

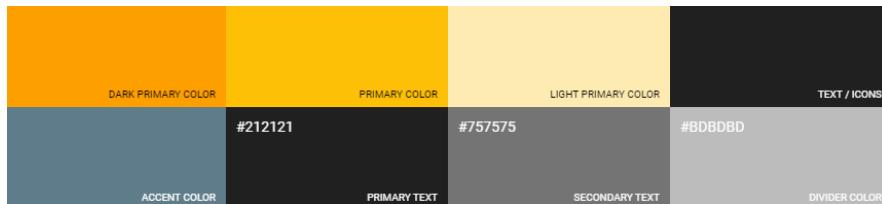


Figura C.20: Paleta de colores.

Apéndice D

Documentación técnica de programación

D.1. Introducción

En este anexo se describe la documentación técnica de programación, incluyendo la instalación del entorno de desarrollo, la estructura de la aplicación, su compilación, la configuración de los diferentes servicios de integración utilizados o las baterías de test realizadas.

D.2. Estructura de directorios

El repositorio del proyecto se distribuye de la siguiente manera:

- /: contiene los ficheros de configuración de Gradle, de los servicios de integración continua, el fichero README y la copia de la licencia.
- /app/: módulo correspondiente a la aplicación.
- /app/src/: código fuente de la aplicación.
- /app/src/main/: contiene todas las clases comunes a todos los *flavours*.
- /app/src/main/res/: recursos de la aplicación (*layouts*, menús, imágenes, cadenas de texto, etc.).
- /app/src/mock/: *mock flavour*, utilizado para injectar componentes alternativos durante los test.
- /app/src/prod/: *prod flavour*, utilizado para injectar los componentes que se utilizan en la versión de producción.
- /app/src/test/: test unitarios.
- /app/src/testMock/: test unitarios y de integración que necesitan injectar componentes falsos.
- /app/src/androidTest/: Android UI test.

- `/docs/`: documentación del proyecto.
- `/docs/img/`: imágenes utilizadas en la documentación.
- `/docs/javadoc/`: documentación *javadoc*.
- `/docs/latex/`: documentación en formato L^AT_EX.
- `/docs/rst/`: documentación en formato reStructuredText.

Para saber más sobre la organización de un proyecto Android consultar la documentación [14].

D.3. Manual del programador

El siguiente manual tiene como objetivo servir de referencia a futuros programadores que trabajen en la aplicación. En él se explica cómo montar el entorno de desarrollo, obtener el código fuente del proyecto, compilarlo, ejecutarlo, testearlo y exportarlo.

Entorno de desarrollo

Para trabajar con el proyecto se necesita tener instalados los siguientes programas y dependencias:

- Java JDK 7.
- Android Studio.
- Git.
- OpenCV.

A continuación, se indica como instalar y configurar correctamente cada uno de ellos.

Java JDK 7

El lenguaje de programación más popular para realizar aplicaciones Android es Java. A día de hoy, Android no soporta la versión 8 de Java, por lo que tenemos que trabajar con la versión 7. Podemos obtener esta versión desde [17]. Se debe elegir correctamente el sistema operativo y la arquitectura del ordenador y, posteriormente, seguir el asistente de instalación.

Android Studio

Android Studio es el IDE oficial para el desarrollo de aplicaciones Android. Está basado en IntelliJ IDEA de JetBrains. Proporciona soporte para Gradle, emulador, editor de *layouts*, refactorizaciones específicas de Android, herramientas Lint para detectar problemas de rendimiento, uso, compatibilidad de versión, etc.

APÉNDICE D. DOCUMENTACIÓN TÉCNICA DE PROGRAMACIÓN70

Se puede obtener desde [15]. Junto con Android Studio se instala también el Android SDK y *Android Virtual Device* (AVD).

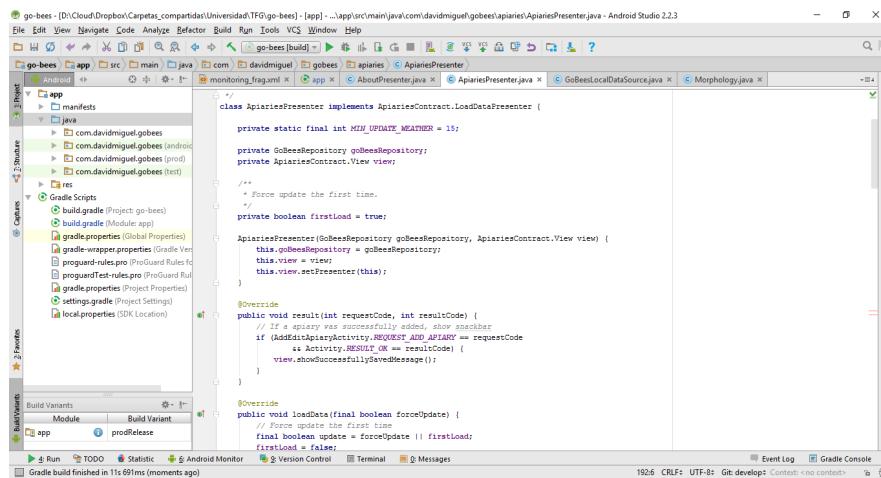


Figura D.1: Android Studio.

Git

Para hacer uso del repositorio se necesita tener instalado el gestor de versiones Git. Este programa nos permitirá clonar el repositorio, movernos por sus diferentes ramas, etiquetas, etc. Se puede obtener desde [11]. Una vez instalado, trabajaremos con Git Bash.

 A screenshot of a terminal window titled "MINGW64:/c/Users/ddmll". The user is cloning a GitHub repository into a local directory. The command entered is "git clone https://github.com/davidmigloz/go-bees.git". The output shows the progress of the cloning process, including object counting, compression, and receiving objects. The terminal window has a dark background with white text and a light gray scroll bar.

Figura D.2: Terminal Git Bash.

OpenCv

OpenCV es un paquete *Open Source* de visión artificial que contiene más de 2500 librerías de procesamiento de imágenes y visión artificial, escritas en C/C++ a bajo/medio nivel.

Para ejecutar OpenCV en un dispositivo Android se necesita tener instalado la aplicación [OpenCV Manager](#). Sin embargo, para el desarrollo de la aplicación también debemos instalar la versión de escritorio de OpenCV para poder ejecutar los test de integración del algoritmo en local.

Podemos obtener OpenCV desde la página oficial [22]. En este proyecto hemos utilizado la versión 3.2.

Una vez instalada, tenemos que añadir al *path* de Windows el directorio donde se encuentran los ejecutables (`opencv/build/java/x64` o `x86`).

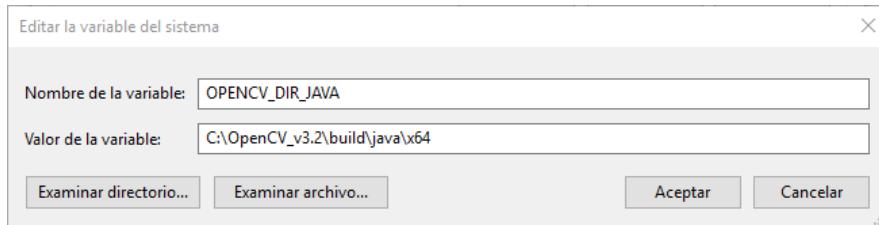


Figura D.3: Variable del sistema de OpenCV.

En la página web oficial se puede obtener información más detallada sobre el proceso de instalación.

Obtención del código fuente

Para el desarrollo de la aplicación se ha utilizado un repositorio Git hospedado en GitHub. Para obtener una copia de este hay que proceder de la siguiente manera:

1. Abrir la terminal Git Bash.
2. Desplazarse al directorio donde se desee copiar el repositorio (utilizando el comando `cd`).
3. Introducir el siguiente comando:
`git clone https://github.com/davidmigloz/go-bees.git`
4. Se iniciará la descarga del repositorio, cuando finalice se dispondrá de una copia completa de este.

```
davidmigloz@ddmlls MINGW64 ~
$ cd ~
davidmigloz@ddmlls MINGW64 ~
$ git clone https://github.com/davidmigloz/go-bees.git
Cloning into 'go-bees'...
remote: Counting objects: 7202, done.
remote: Compressing objects: 100% (2284/2284), done.
remote: Total 7202 (delta 3887), reused 7170 (delta 3855), pack-reused 0
Receiving objects: 100% (7202/7202), 68.17 MiB | 1.86 MiB/s, done.
Resolving deltas: 100% (3887/3887), done.
Checking connectivity... done.

davidmigloz@ddmlls MINGW64 ~
$ |
```

Figura D.4: Clonar repositorio de GitHub.

Para conocer el proceso detalladamente consultar [12].

Importar proyecto en Android Studio

Una vez obtenido el código fuente de la aplicación, tenemos que importarlo como proyecto de Android Studio. Para ello, hay que seguir los siguientes pasos:

1. Abrir Android Studio.
2. Menú **File > Open...**
3. Buscamos el directorio donde hemos clonado el repositorio.
4. Dentro del repositorio, seleccionamos el archivo **build.gradle**.
5. Android Studio detectará que es un proyecto Android y lo importará automáticamente.
6. Si alguna característica de las que hace uso la aplicación no se encuentra instalada, Android Studio mostrará un mensaje de error con un enlace para instalar la característica en cuestión.

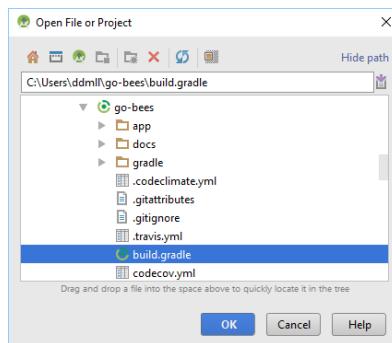


Figura D.5: Importar proyecto en Android Studio.

Para conocer el proceso detalladamente consultar [2].

Añadir nuevas características a la aplicación

Tras importar el proyecto en Android Studio, ya estamos en disposición de realizar modificaciones de la aplicación.

Para añadir una nueva característica siguiendo la arquitectura MVP, la convención de paquete por característica y las metodologías TDD y GitFlow, se deben seguir los siguientes pasos generales.

1. Crear una nueva rama (*feature branch*) desde la rama *develop*:
`git checkout -b export-data develop`
2. Crear un nuevo paquete con el nombre de la característica que se desea añadir (ej. `exportdata`).
3. Crear una interfaz (ej. `ExportDataContract.java`) que contenga a su vez dos interfaces. En una se deben definir las responsabilidades del *presenter* y en la otra las de la vista. Hacer *commit*:
`git add -A`
`git commit -m ".^dd export data contract #x"`
4. Crear una clase para el *presenter* (ej. `ExportDataPresenter.java`) que implemente su correspondiente interfaz anterior (no añadir ninguna lógica todavía). Hacer *commit*.
5. Crear una clase para la vista (ej. `ExportDataFragment`) que descienda de `Fragment` e implemente su correspondiente interfaz anterior (no añadir ninguna lógica todavía). Hacer *commit*.
6. Crear una clase que descienda de `AppCompatActivity` (ej. `ExportDataActivity.java`) y que enlace el modelo, el *presenter* y la vista. Hacer *commit*.
7. Crear un test sobre el *presenter* de acuerdo a los requisitos. Hacer *commit*.
8. Ejecutar el test y comprobar que no pasa.
9. Implementar las clases anteriores hasta conseguir que pasen el test. Hacer *commit*.
10. Refactorizar el código para mejorar su calidad. Hacer *commit*.
11. Añadir un *intent* desde donde se quiera acceder a esa característica. Hacer *commit*.
12. Una vez que se ha implementado correctamente la característica, se debe incorporar a la rama *develop* y sincronizar con GitHub:
`git checkout develop`
`git merge --no-ff export-data`
`git branch -d myfeature`
`git push origin develop`

Actualizar dependencias

Una tarea de mantenimiento común es la actualización de las dependencias de la aplicación. Es importante tenerlas actualizadas para evitar problemas de seguridad o funcionalidad que pudiesen tener en versiones anteriores.

El proyecto utiliza Gradle como sistemas de construcción automática del *software*. Una de sus funcionalidades es la gestión de dependencias. Esta permite al desarrollador definir las dependencias de su aplicación, sus versiones y los repositorios donde se hospedan y Gradle se encarga de descargarlas e importarlas al proyecto automáticamente.

Las dependencias se definen en el fichero `build.gradle` del módulo de la aplicación (`go-bees/app/build.gradle`):

```
dependencies {

    // App's dependencies, including test
    compile 'com.android.support:recyclerview-v7:25.1.1'
    compile 'com.android.support:appcompat-v7:25.1.1'
    compile 'com.android.support:design:25.1.1'
    compile 'com.android.support:cardview-v7:25.1.1'
    compile 'com.android.support:support-v4:25.1.1'
    compile 'com.github.davidmigloz:opencv-android-gradle-repo:3.2.0'
    compile 'com.google.android.gms:play-services-location:10.0.1'
    compile 'com.google.guava:guava:20.0'
    compile 'com.makeramen:roundedimageview:2.3.0'
    compile 'com.github.PhilJay:MPAndroidChart:v3.0.1'
    compile 'com.vanniktech:vnnumberpickerpreference:1.0.0'
    compile 'rebus:permission-utils:1.0.6'
    // Dependencies for local unit tests
    testCompile 'junit:junit:4.12'
    testCompile 'org.mockito:mockito-all:2.0.2-beta'
    testCompile 'org.powermock:powermock-module-junit4:1.6.6'
    testCompile 'org.powermock:powermock-api-mockito:1.6.6'
    testCompile 'org.slf4j:slf4j-api:1.7.22'
    testCompile 'org.slf4j:slf4j-log4j12:1.7.22'
    testCompile 'log4j:log4j:1.2.17'
    testCompile 'org.json:json:20160810'
    // Android Testing Support Library's runner and rules
    androidTestCompile 'com.android.support.test:runner:0.5'
    androidTestCompile 'com.android.support.test:rules:0.5'
    // Dependencies for Android unit tests
    androidTestCompile 'junit:junit:4.12'
    androidTestCompile 'org.mockito:mockito-core:2.6.2'
    // Espresso UI Testing
    androidTestCompile 'com.android.support.test.espresso:espresso-core:2.2.2'
    androidTestCompile 'com.android.support.test.espresso:espresso-contrib:2.2.2'
    androidTestCompile 'com.android.support.test.espresso:espresso-intents:2.2.2'
    // Resolve conflicts between main and test APK:
    androidTestCompile 'com.android.support:support-annotations:25.1.1'
    androidTestCompile 'com.android.support:support-v4:25.1.1'
    androidTestCompile 'com.android.support:recyclerview-v7:25.1.1'
    androidTestCompile 'com.android.support:appcompat-v7:25.1.1'
    androidTestCompile 'com.android.support:design:25.1.1'
}
```

Figura D.6: Dependencias del proyecto.

Se puede observar que existen tres formas de importar las dependencias, cada una define con un ámbito de aplicación distinto:

- **Compile**: estará disponible para el código de la aplicación.
- **testCompile**: estará disponible en los test unitarios de la aplicación.
- **androidTestCompile**: estará disponible en los test de instrumentación de la aplicación.

Para actualizar la versión de una dependencia, solamente hay que actualizar el número de la versión que figura en la importación. Posteriormente, se debe sincronizar Gradle (*Sync Project with Gradle Files*).

Compilar código fuente

La compilación del proyecto se realiza mediante la tarea **build** de Gradle. Podemos ejecutarla por línea de comandos (`./gradlew build`) o mediante la interfaz de Android Studio.

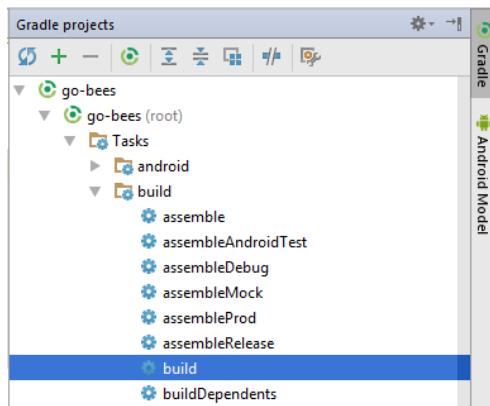


Figura D.7: Compilar proyecto.

Todos los ficheros generados durante la compilación se guardan en la carpeta **build** del proyecto.

Para conocer el proceso detalladamente consultar [1].

Ejecutar aplicación

La aplicación se puede ejecutar en un dispositivo real o en un emulador.

Dispositivo real

Para ejecutar la aplicación en un dispositivo real, se debe conectar este al equipo de desarrollo mediante un cable USB. El equipo debe tener los *drivers* del dispositivo instalado, sino no lo reconocerá.

Una vez conectado el dispositivo:

1. Presionar el botón *Run*.
2. Si el equipo reconoce el dispositivo se mostrará su nombre debajo de “*Connected Devices*”.
3. Seleccionar el dispositivo y pulsa *Ok*.
4. Se transferirá el ejecutable de la aplicación y se instalará.
5. Una vez instalada, se podrá utilizar la aplicación desde el dispositivo.

Emulador

Un emulador (denominados *Android Virtual Device - AVD*) es una aplicación que simula el funcionamiento de un dispositivo real Android. La creación y gestión de los emuladores se hace a través de *AVD Manager*.

Para ejecutar la aplicación en un emulador:

1. Presionar el botón de Run.
2. Si ya se posee algún emulador instalado, se mostrará en la lista de *Android Virtual Devices*.
3. Si no, presionar el botón “*Create New Virtual Device*”.
4. Seleccionar las características que se deseen para el emulador y pulsa finalizar.
5. Seleccionar el emulador creado y pulsar *Ok*.
6. Se iniciará el emulador y se instalará la aplicación en él.
7. Una vez instalada, se podrá utilizar la aplicación desde el emulador.

Para conocer el proceso detalladamente consultar [1].

Exportar aplicación

Para exportar la aplicación como un fichero .apk:

1. Menú *Build > Generate APK*.
2. Se generará un archivo apk y se guardará en `build/output/apk`.

Si el apk que se desea generar es para distribuirlo en Google Play, este debe estar firmado. Para ello:

1. Menú *Build > Generate Signed APK*.
2. Se debe seleccionar el archivo `.jks` con la clave e introducir su contraseña. Si no se dispone de una clave, se puede generar siguiendo el asistente.
3. Se generará un archivo `apk` firmado apto para subir al Google Play.

Para conocer el proceso detalladamente consultar [1].

Servicios de integración continua

En el repositorio se han integrado varios servicios de integración continua para detectar fallos en el software lo antes posible, reduciendo el impacto de estos y aumentando la calidad del código.

A continuación, se describe cada servicio y se indica cómo configurarlo.

TravisCI

TravisCI es una plataforma de integración continua en la nube para proyectos alojados en GitHub. Permite realizar una *build* del proyecto y testearla automáticamente cada vez que se realiza un *commit*, devolviendo un informe con los resultados.

Para integrar Travis en el repositorio hospedado en GitHub se debe crear una cuenta en su página web y dar permisos de acceso al repositorio. Una vez asociado el servicio, este se configura mediante el fichero `travis.yml`.

Las secciones más importantes de este fichero son:

- `sudo`: permite definir si el usuario de la máquina virtual tendrá privilegios o no.
- `language`: permite definir el lenguaje de programación del proyecto.
- `jdk`: permite definir la versión del JDK.
- `compiler`: permite definir el compilador.
- `addons`: permite configurar *plugins* instalados en Travis (como, por ejemplo, el *plugin* de SonarQube).
- `env`: permite definir variables de entorno.
- `android`: permite definir las dependencias Android del proyecto.
- `licenses`: permite aceptar las licencias de las dependencias.
- `before_install`: en esta sección se pueden definir comandos a ejecutar antes de los comandos de la sección `install` (por ejemplo, actualizar la lista de paquetes).
- `install`: en esta sección se deben definir aquellos comandos que instalen alguna dependencia (en nuestro caso `python-numpy`, necesaria para compilar OpenCV).

- **before_script:** en esta sección se pueden definir comandos a ejecutar antes de la sección script. En nuestro caso, nos descargamos el código fuente de OpenCV y lo compilamos.
- **script:** en esta sección se realiza la compilación del proyecto y se ejecutan los diferentes test unitarios y de integración. Además, lanza un emulador y ejecuta los test de interfaz. También ejecuta el motor de chequeo de SonarQube.
- **after_success:** esta sección se utiliza para recolectar datos generados en las secciones anteriores. En nuestro caso, se envían los diferentes informes de ejecución de los test a el servicio Codecov.
- **cache:** permite definir los directorios a cachear entre ejecuciones.

Los *log* de ejecución de Travis son accesibles desde [27].

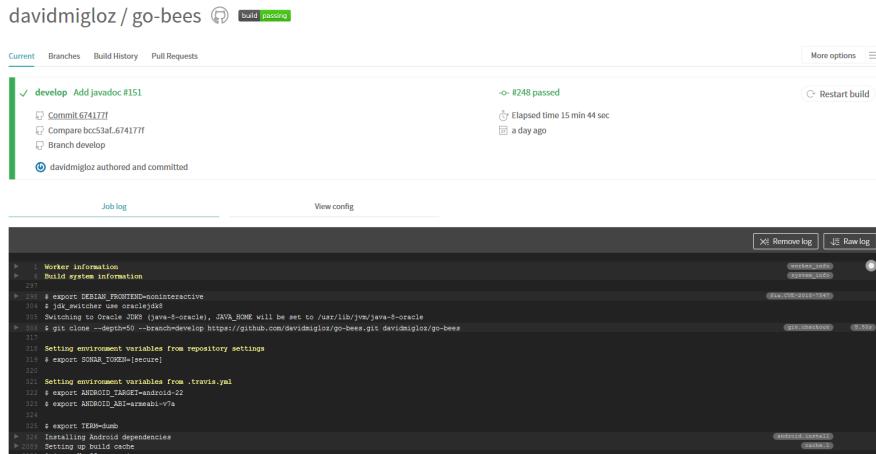


Figura D.8: TravisCI.

Para saber más, acceder a su documentación [26].

Codecov

Codecov es una herramienta que permite medir el porcentaje de código que está cubierto por un test. Además, realiza representaciones visuales de la cobertura y gráficos de su evolución.

La forma de integrarlo en el repositorio es idéntica a cómo se hizo con Travis. Adicionalmente, hay que configurar el *script* que ejecuta Travis para que al finalizar su ejecución envíe los resultados a Codecov.

```
after_success: bash <(curl -s https://codecov.io/bash)
```

La configuración de Codecov se define en el archivo `codecov.yml`.

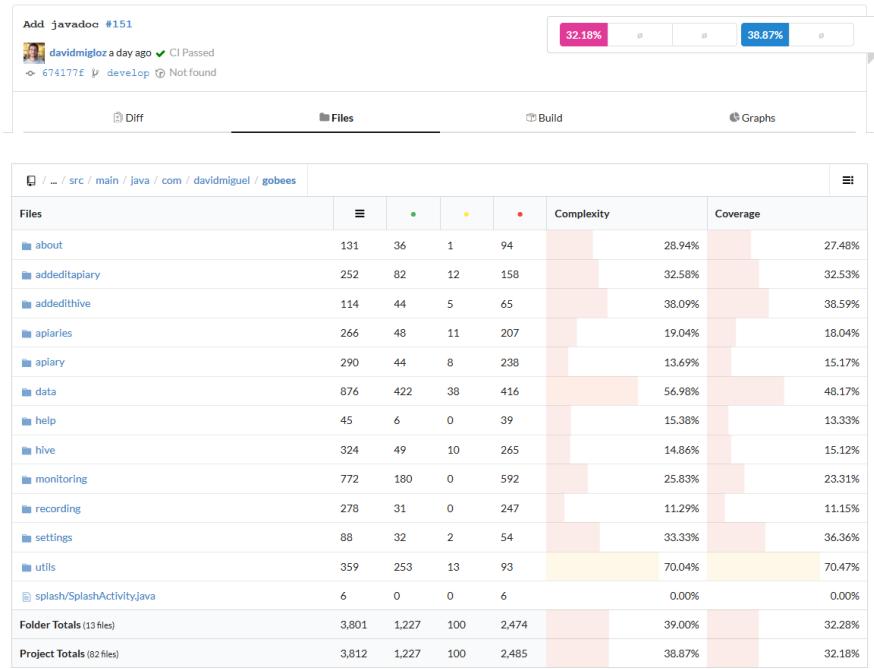


Figura D.9: Codecov.

Para saber más, acceder a su documentación [5].

CodeClimate

Codeclimate es una herramienta que realiza revisiones de código automáticamente.

La integración se realiza de forma similar a Travis. Su fichero de configuración es `.codeclimate.yml`.

En nuestro proyecto hemos activado los siguientes motores de chequeo: `checkstyle`, `fixme`, `markdownlint` y `pmd`.

CodeClimate utiliza el sistema de puntuación GPA (*Grade Point Average*) para indicar el rendimiento general del proyecto. La nota máxima se corresponde con un 4.0.

Los resultados de los chequeos se encuentran disponibles en [4].

APÉNDICE D. DOCUMENTACIÓN TÉCNICA DE PROGRAMACIÓN80

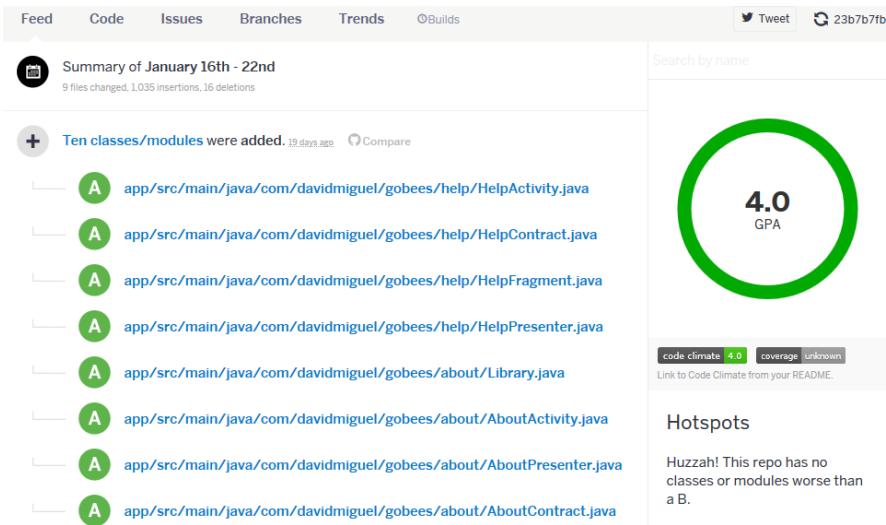


Figura D.10: CodeClimate.

Para saber más, acceder a su documentación [3].

SonarQube es una plataforma de código abierto para la revisión continua de la calidad de código. Permite detectar código duplicado, violaciones de estándares, cobertura de test unitarios, *bugs* potenciales, etc.

Para integrar el servicio hay que seguir los siguientes pasos:

1. Crear una cuenta en www.sonarqube.com.
2. Generar un *token* de autenticación.
3. Instalar el plugin de SonarQube para Gradle (`org.sonarqube`).
4. Configurar SonarQube en el fichero de configuración de Gradle (`build.gradle`).
5. Ejecutar la nueva tarea `sonarqube` de Gradle desde Travis.

```
sonarqube {
    properties {
        property "sonar.projectName", "GoBees"
        property "sonar.projectKey", "com.davidmiguel.gobees"
        property "sonar.language", "java"
        property "sonar.projectVersion", "${android.defaultConfig.versionName}"
        property "sonar.exclusions", "**/*.png,**/*.jpg"
        property "sonar.android.lint.report", "./build/outputs/lint-results-mockDebug.xml"
        property "sonar.junit.reportsPath", "./build/test-results/mockDebug"
        property "sonar.jacoco.reportPath", "./build/jacoco/testMockDebugUnitTest.exec"
        property "sonar.java.coveragePlugin", "jacoco"
        property "sonar.jacoco.reportMissing.force.zero", true
    }
}
```

Figura D.11: Configuración de SonarQube en Gradle.

Los resultados de los análisis son accesibles desde [24].

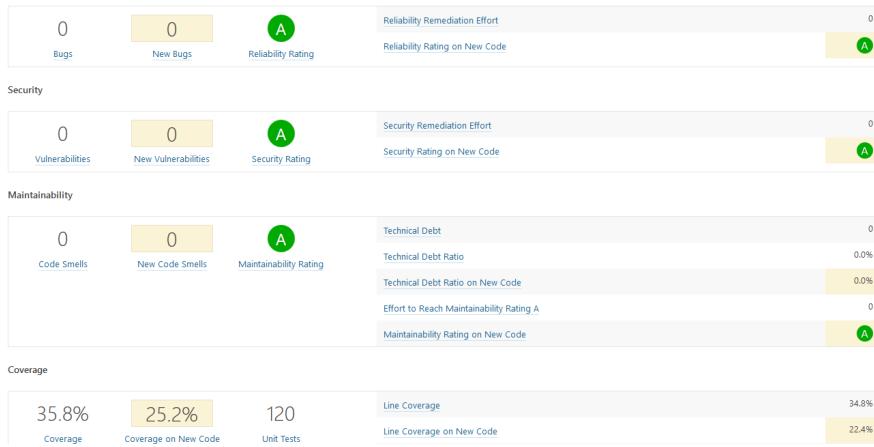


Figura D.12: SonarQube.

Para saber más, acceder a su documentación [23].

VersionEye

VersionEye es una herramienta que monitoriza las dependencias del proyecto y envía notificaciones cuando alguna de estas está desactualizada, es vulnerable o viola la licencia del proyecto.

El servicio se integra de forma similar a Travis. No necesita fichero de configuración.

Cuando se libera una nueva versión de alguna dependencia o se publica alguna vulnerabilidad, VersionEye manda una notificación. Se puede acceder a los informes desde [29].

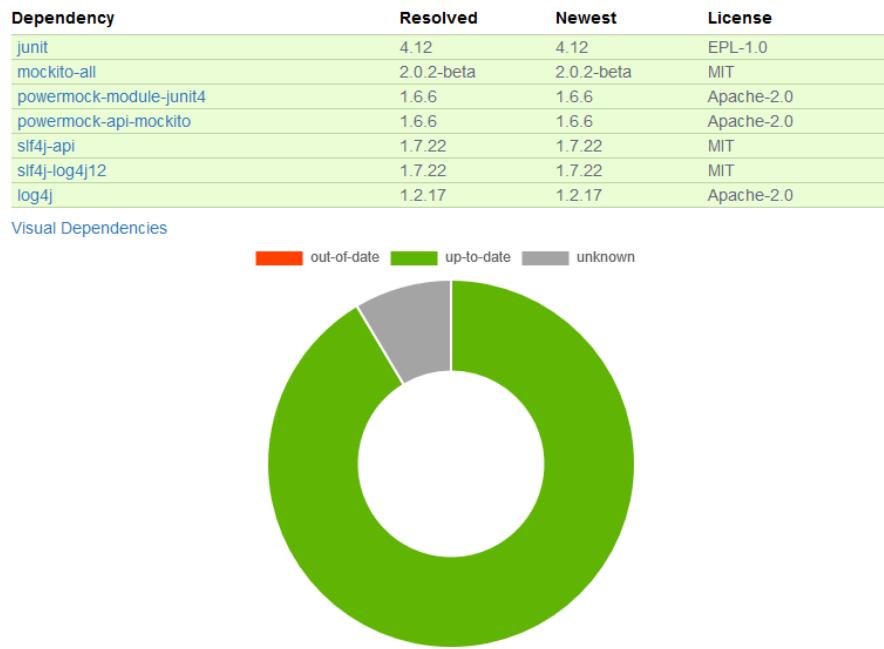


Figura D.13: VersionEye.

Para saber más, acceder a su documentación [28].

Read the Docs

Read the Docs es un servicio de documentación continua que permite crear y hospedar una página web generada a partir de los distintos ficheros Markdown o reStructuredText de la documentación. Cada vez que se realiza un *commit* en el repositorio se actualiza la versión hospedada.

Se integra en el repositorio de la misma manera que Travis. Y se configura mediante el archivo `conf.py` ubicado en `go-bees/docs/rst`.

Actualmente, se encuentra configurado para generar una sección en la página web por cada archivo reStructuredText que encuentre dentro del directorio `rst`.



Figura D.14: Página web generada con ReadTheDocs.

Para saber más, acceder a su documentación [25].

D.4. Pruebas del sistema

Para verificar el funcionamiento de cada uno de los módulos de la aplicación, su integración y la interacción con estos desde la interfaz, se han desarrollado una serie de baterías de test.

Test unitarios

Los test unitarios comprueban la funcionalidad de un único módulo trabajando de forma aislada. Para su escritura se han utilizado las dependencias jUnit y Mockito.

jUnit es un *framework* de Java utilizado para realizar pruebas unitarias. Mockito es un *framework* de *mocking* que permite crear objetos *mock* fácilmente. Estos objetos simulan parte del comportamiento de una clase. De esta manera, podemos aislar el módulo a testear para que los módulos de los que depende no interfieran en los resultados del test.

Se han escrito 120 test unitarios que testean 30 clases distintas. Se han testeado en su mayoría los *presenters* que son los que poseen la lógica de la aplicación y no tienen ninguna dependencia al *framework* de Android. Lo que permite ejecutarlos sin necesidad de lanzar un emulador.

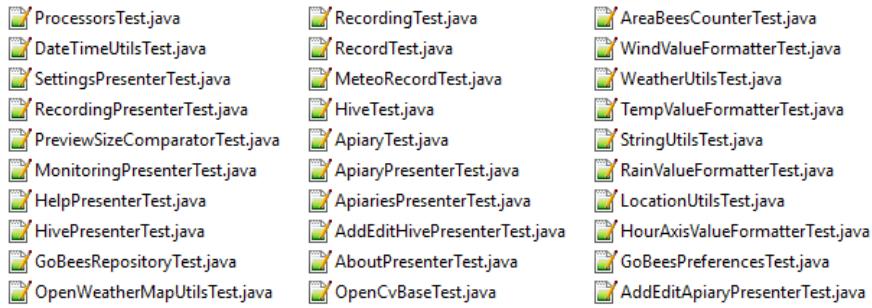


Figura D.15: Test unitarios.

Ejecución de los test unitarios

Los test unitarios se ejecutan automáticamente en Travis cada vez que se realiza un *commit* y se hace un *push* a GitHub. Pero también se pueden ejecutar en local. Para ello:

1. Seleccionar el *Build Variants mockDebug*.
2. Seleccionar como tipo de vista Android.
3. Pulsar botón derecho en el paquete **test** > *Run test in go-bees*.
4. Se ejecutarán todos los test y se obtendrá un informe de resultados.

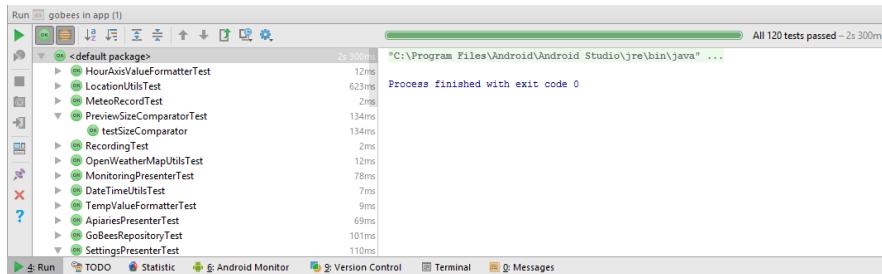


Figura D.16: Ejecución de los test unitarios.

Test del algoritmo

Para testear el algoritmo se han escrito varios test unitarios que prueban cada uno de sus módulos y un test de integración (**AreaBeesCounterTest.java**) que lo testea en su totalidad contra tres conjuntos de fotogramas etiquetados manualmente. De esta manera, se obtiene el error que comete el algoritmo en cada caso y se compara con unos límites prefijados. Si por alguna modificación accidental el error supera el límite, el test falla.

Ejecución del test del algoritmo

El test de integración se ejecuta automáticamente en Travis junto con los test unitarios. También puede ser ejecutado en local, pero es imprescindible tener instalado OpenCV en el equipo. Los pasos a seguir son:

1. Seleccionar el *Build Variants mockDebug*.
2. Seleccionar como tipo de vista Android.
3. Pulsar botón derecho en el paquete `testMock > Run test in go-bees`.
4. Se ejecutará el test y se obtendrá un informe de resultados.

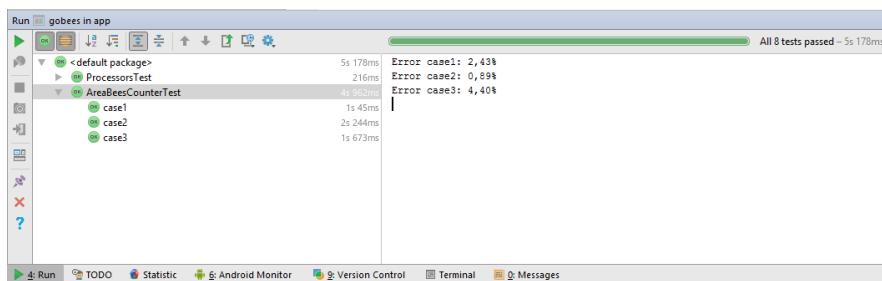


Figura D.17: Ejecución del test de integración del algoritmo.

Etiquetado de nuevos conjuntos de fotogramas

Para etiquetar videos manualmente se ha desarrollado una aplicación en Java que facilita esta labor. La aplicación va mostrando cada fotograma y el usuario solo tiene que pinchar encima de cada abeja existente. Finalmente, la aplicación permite exportar los datos en un archivo CSV con el formato que utiliza el test del algoritmo.

Los pasos a seguir son:

1. Ejecutar la aplicación (Disponible en [19]).
2. Abrir el directorio que posee los fotogramas.
3. Marcar las abejas presentes en cada fotograma con el ratón. La aplicación mostrará el número del fotograma y el número de abejas marcadas.
4. Al finalizar, seleccionar guardar. La aplicación exportará los datos en un archivo CSV.

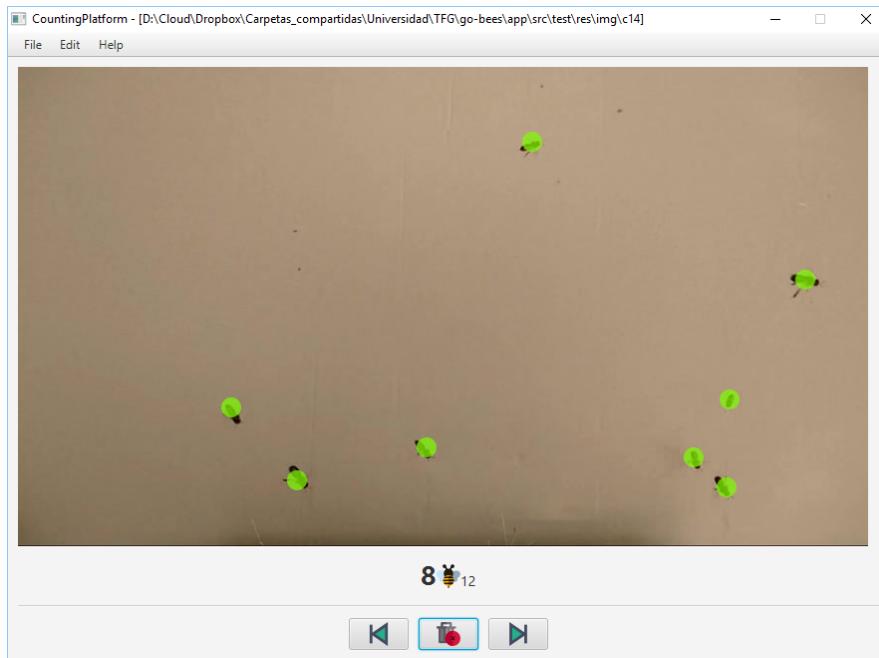


Figura D.18: Aplicación de etiquetado de fotogramas.

Testeo de la parametrización del algoritmo

Para desarrollar el algoritmo y parametrizarlo de forma óptima, se desarrolló una aplicación Java que permite modificar los diferentes parámetros de cada fase en tiempo real y calcular sus tiempos de cómputo.

Si se desea probar nuevas parametrizaciones:

1. Ejecutar la aplicación (Disponible en [19]. Es necesario tener instalado OpenCV en el equipo).
2. Seleccionar un archivo de vídeo de prueba.
3. En la ventana izquierda se visualiza la entrada del algoritmo y a la derecha existe una pestaña por cada fase de este.
4. En cada pestaña, a parte de la salida del algoritmo para esa fase, se poseen una serie de controles para parametrizar el algoritmo.
5. En la parte inferior izquierda se muestra los fotogramas por segundo que se están procesando. En la parte central, el tiempo total de procesado. Y en la parte derecha, el tiempo parcial de la fase en cuestión.

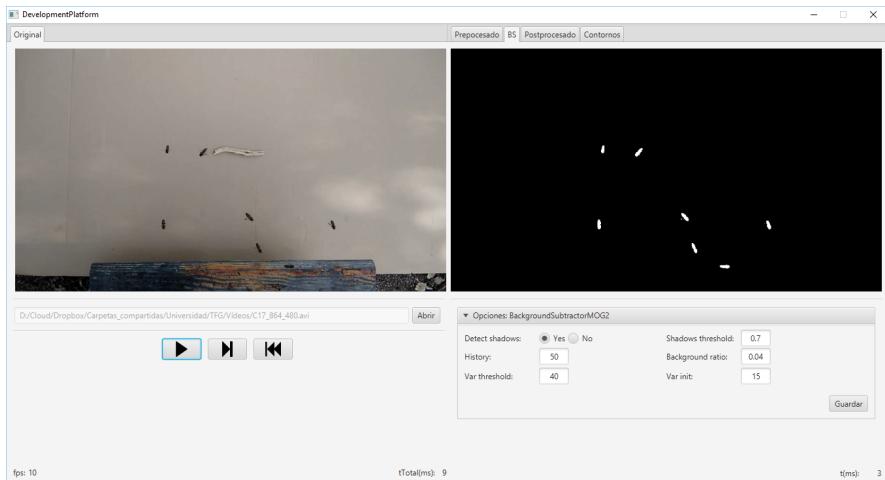


Figura D.19: Plataforma de desarrollo del algoritmo.

Test de interfaz

Por último, se han desarrollado 17 test de interfaz que testean cada uno de los requisitos de la aplicación, a excepción del requisito de monitorización que no fue posible testearlo en un emulador (no se puede utilizar como *feed* de la cámara de un emulador un archivo de vídeo).

Para desarrollar los test se ha utilizado Espresso, un *framework* de *testing* para Android que provee una API para escribir UI test que simulen las interacciones de usuario con la *app*.

En la siguiente tabla se relaciona cada test con el requisito que comprueba.

Test	Requisitos
AddApiaryTest.java	RF-1.1 Añadir colmenar.
EditApiaryTest.java	RF-1.2 Editar colmenar.
DeleteApiaryTest.java	RF-1.3 Eliminar colmenar.
ListApiariesTest.java	RF-1.4 Listar colmenares.
ViewApiaryTest.java	RF-1.5 Ver colmenar.
AddHiveTest.java	RF-2.1 Añadir colmena.
EditHiveTest.java	RF-2.2 Editar colmena.
DeleteHiveTest.java	RF-2.3 Eliminar colmena.
ListHivesTest.java	RF-2.4 Listar colmenas.
ViewHiveTest.java	RF-2.5 Ver colmena.
AddRecordingTest.java	RF-3.1 Añadir grabación.
DeleteRecordingTest.java	RF-3.2 Eliminar grabación.
ListRecordingsTest.java	RF-3.3 Listar grabaciones.
ViewRecordingTest.java	RF-3.4 Ver grabación.
SettingsTest.java	RF-5 Configuración de la aplicación.
HelpTest.java	RF-6 Ayuda de la aplicación.
AboutTest.java	RF-7 Información de la aplicación.

Tabla D.1: Requisitos testeados.

Ejecución de los test de interfaz

Para ejecutar los test de interfaz es imprescindible contar con un dispositivo físico o un emulador. Una vez conectado, se siguen los siguientes pasos:

1. Seleccionar el *Build Variants* `mockDebug`.
2. Seleccionar como tipo de vista Android.
3. Pulsar botón derecho en el paquete `androidTest > Run test in go-bees`.
4. Se ejecutarán cada uno de los test en el dispositivo (Android Studio instala una aplicación adicional que instrumenta a la aplicación a testear).
5. Al finalizar, se obtiene un informe con los resultados.

Apéndice E

Documentación de usuario

E.1. Introducción

En este manual se detallan los requerimientos de la aplicación, cómo instalarla en un dispositivo Android e indicaciones sobre cómo utilizarla correctamente. Todos los procedimientos aquí descritos se encuentran también disponibles en formato video.

E.2. Requisitos de usuarios

Los requisitos mínimos para poder hacer uso de la aplicación son:

- Contar con un dispositivo que posea Android 4.4 (*KitKat* – API 19) o superior.
- Para utilizar la característica de monitorización de la actividad, es necesario tener instalada la aplicación [OpenCV Manager](#).
- También se necesita contar con permiso para acceder a la cámara del dispositivo.
- Si se desea localizar los colmenares mediante GPS, es necesario contar con un dispositivo que lo soporte y conceder el permiso de localización a la aplicación.
- Para acceder a la información meteorológica se requiere conexión a internet.

E.3. Instalación

La instalación se puede realizar de dos maneras: a través de Google Play o instalando directamente el ejecutable de la aplicación en nuestro dispositivo.

Desde Google Play

Google Play es una plataforma de distribución digital de aplicaciones móviles para los dispositivos Android. GoBees se distribuye por esta plataforma desde su versión 1.0.



Figura E.1: GoBees en Google Play.

Video-tutorial: <http://gobees.io/help/videos/installacion-google-play>

Para instalar la aplicación debemos realizar los siguientes pasos:

1. Acceder a la aplicación Google Play.
2. Buscar el término “GoBees”.
3. Entrar en la sección correspondiente a la aplicación.
4. Pulsar el botón instalar.
5. Cuando la instalación haya finalizado, pulsar sobre el botón abrir.
6. La instalación habrá finalizado y la aplicación estará lista para su uso.



Figura E.2: Instalación desde Google Play

Desde fichero ejecutable

La otra opción, es realizar la instalación directamente desde el fichero ejecutable de la aplicación. Estos ficheros poseen la extensión .apk. Podemos conseguir la última versión del .apk de GoBees desde [20].

Video-tutorial: <http://gobees.io/help/videos/instalacion-apk>

Una vez descargado, tenemos que seguir los siguientes pasos:

1. En primer lugar, hay que permitir la instalación de “aplicaciones con orígenes desconocidos”. Para ello:
 - a. Ir a ajustes del dispositivo.
 - b. Seguridad (o Privacidad).
 - c. Activar “Orígenes desconocidos”.
2. Ejecutar el fichero descargado.
3. Pulsar el botón instalar.
4. Cuando la instalación haya finalizado, pulsar sobre el botón abrir.
5. La instalación habrá finalizado y la aplicación estará lista para su uso.

E.4. Manual de usuario

En esta sección se describe el uso de las diferentes funcionalidades de la aplicación.

Generar datos de muestra

Una de las mejores maneras de aprender a utilizar una aplicación es indagando en ella. GoBees permite generar un colmenar de prueba, de tal manera, que podemos explorar las diferentes secciones con datos reales.

Video-tutorial: <http://gobees.io/help/videos/generar-colmenar-prueba>

Para generar los datos de prueba:

1. Pulsar el botón menú.
2. Entrar en la sección “Ajustes”.
3. Seleccionar la opción “Generar datos de muestra”.
4. Se generará un colmenar con tres colmenas y tres grabaciones por colmena.

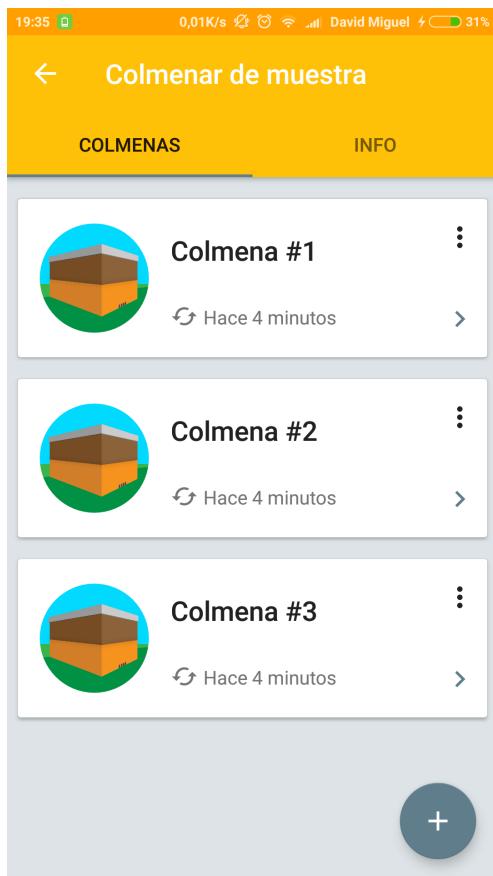


Figura E.3: Colmenar de muestra.

Añadir un colmenar

Un colmenar hace referencia al lugar o recinto donde se poseen un conjunto de colmenas. Un colmenar posee un nombre, una localización y unas notas.

Video-tutorial: <http://gobees.io/help/videos/anadir-colmenar>

Para añadir un nuevo colmenar:

1. Desde la pantalla principal.
2. Pulsar el botón “+”.
3. Definir el nombre del colmenar (obligatorio).
4. Definir la localización del colmenar (opcional).
 - a. Se pueden introducir manualmente las coordenadas, indicando la latitud y la longitud en el sistema de coordenadas geográficas.
 - b. Alternativamente, se puede obtener la localización actual automáticamente pulsando el botón situado en la parte derecha (se necesitan permisos de localización para utilizar esta característica).
5. Definir unas notas sobre el colmenar (opcional). En las notas se puede apuntar cualquier cosa relacionada con el colmenar en general.
6. Pulsar el botón ✓ para guardar el nuevo colmenar.

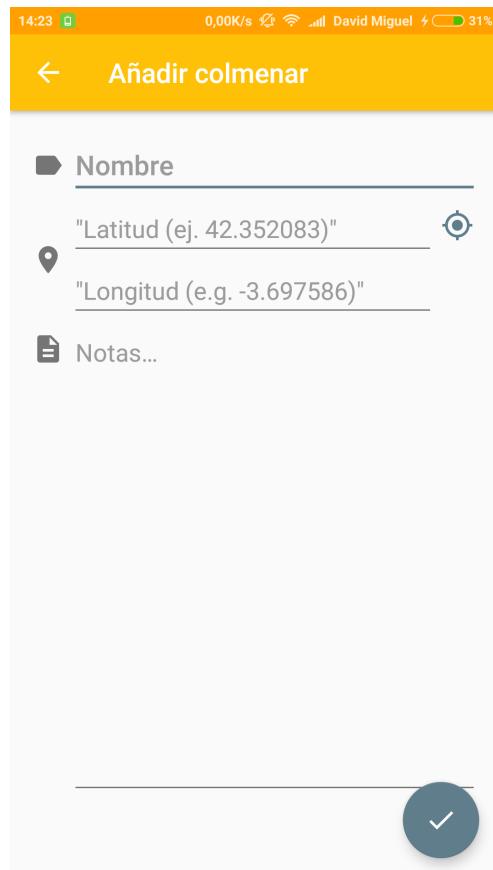


Figura E.4: Añadir colmenar.

Editar un colmenar

Los detalles de un colmenar se pueden editar en cualquier momento.

Video-tutorial: <http://gobees.io/help/videos/editar-colmenar>

Para editar un colmenar existente:

1. Desde la pantalla principal.
2. Pulsar el botón de menú asociado al colmenar a editar (tres puntos verticales situados en la esquina superior derecha).
3. Seleccionar la opción de editar.
4. Se abrirá la pantalla de edición, donde se podrán modificar los datos que se deseen.
5. Pulsar el botón ✓ para actualizar los datos editados.

Eliminar un colmenar

Al eliminar un colmenar, se eliminan también todos los datos asociados a este (información del colmenar, colmenas, grabaciones e información meteorológica).

Video-tutorial: <http://gobees.io/help/videos/eliminar-colmenar>

Para eliminar un colmenar existente:

1. Desde la pantalla principal.
2. Pulsar el botón de menú asociado al colmenar a eliminar (tres puntos verticales situados en la esquina superior derecha).
3. Seleccionar la opción de eliminar.
4. El colmenar se eliminará junto con toda su información.

Consultar la información meteorológica de un colmenar

Para poder consultar la información meteorológica de un colmenar se necesita que este posea una localización y que el dispositivo esté conectado a internet. Si se cumplen estos dos requisitos, la información meteorológica del colmenar se actualizará automáticamente de forma periódica.

Video-tutorial: <http://gobees.io/help/videos/consultar-info-meteo-colmenar>

Para consultar la información meteorológica:

1. Asegurarse de que el colmenar tiene definida una localización y que se posee conexión a internet.
2. En la lista de colmenares, se puede visualizar un resumen con la temperatura y situación meteorológica en cada colmenar.

3. Si se desea consultar la información en detalle, entrar en el colmenar a consultar.
4. Desplazarse a la pestaña “info”.
5. En la parte inferior podremos visualizar todos los detalles de la situación meteorológica actual en ese colmenar.

Se pueden cambiar las unidades meteorológicas, para ello:

1. En la pantalla principal.
2. Pulsar el botón menú.
3. Entrar en la sección “Ajustes” .
4. Seleccionar “Unidades meteorológicas” .
 - a. Sistema métrico: °C y km/h.
 - b. Sistema imperial: °F y mph.

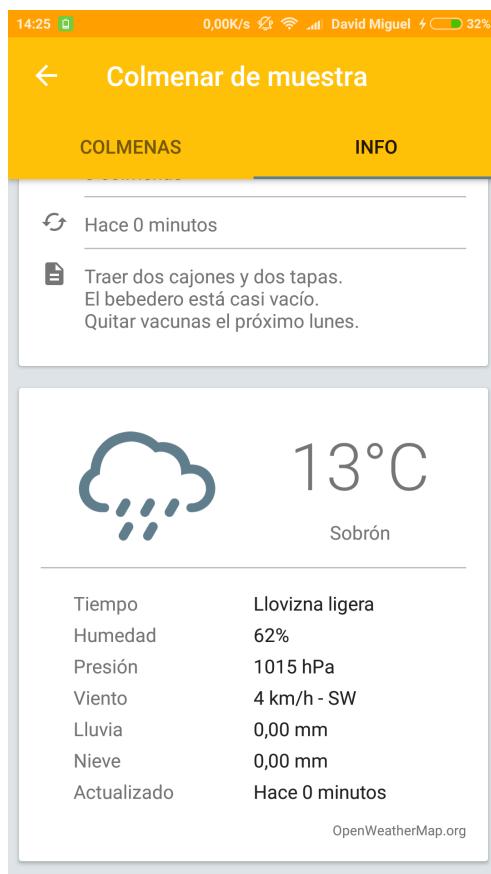


Figura E.5: Información meteorológica.

Visualizar un colmenar en el mapa

GoBees nos permite visualizar fácilmente un determinado colmenar en un mapa utilizando nuestra aplicación de mapas favorita. De esta manera, podemos navegar hacia él o consultar cualquier detalle cartográfico.

Video-tutorial: <http://gobees.io/help/videos/ver-colmenar-mapa>

Para visualizar un colmenar en el mapa:

1. Entrar en el colmenar a visualizar.
2. Desplazarse a la pestaña “info”.
3. Pulsar el botón “mapa” situado a la derecha de la localización del colmenar.
4. Seleccionar la aplicación con la que se desea visualizar el colmenar.

Añadir una colmena

Cada colmena pertenece a un colmenar y tiene un nombre y unas notas. Además, se puede monitorizar su actividad de vuelo, dando lugar a grabaciones.

Video-tutorial: <http://gobees.io/help/videos/anadir-colmena>

Para añadir una colmena en un determinado colmenar:

1. Entrar en el colmenar al que pertenecerá.
2. Definir el nombre de la colmena (obligatorio).
3. Definir unas notas sobre la colmena (opcional). En las notas se puede apuntar cualquier cosa relacionada con la colmena en concreto.
4. Pulsar el botón ✓ para guardar la nueva colmena.

Editar una colmena

Los detalles de una colmena se pueden editar en cualquier momento.

Video-tutorial: <http://gobees.io/help/videos/editar-colmena>

Para editar una colmena existente:

1. Entrar en el colmenar al que pertenece la colmena.
2. Pulsar el botón de menú asociado a la colmena a editar (tres puntos verticales situados en la esquina superior derecha).
3. Seleccionar la opción de editar.
4. Se abrirá la pantalla de edición, donde se podrán modificar los datos que se deseen.
5. Pulsar el botón ✓ para actualizar los datos editados.

Eliminar una colmena

Al eliminar una colmena, se eliminan también todos los datos asociados a esta (información de la colmena y sus grabaciones).

Video-tutorial: <http://gobees.io/help/videos/eliminar-colmena>

Para eliminar una colmena existente:

1. Entrar en el colmenar al que pertenece la colmena.
2. Pulsar el botón de menú asociado a la colmena a editar (tres puntos verticales situados en la esquina superior derecha).
3. Seleccionar la opción de eliminar.
4. La colmena se eliminará junto con toda su información.

Monitorizar la actividad de vuelo de una colmena

La actividad de vuelo, junto con información previa de la colmena y conocimiento de las condiciones locales, permite conocer al apicultor el estado de la colmena con bastante seguridad, pudiendo determinar si esta necesita o no una intervención.

GoBees permite monitorizar este parámetro utilizando la cámara del *smartphone*.

Video-tutorial: <http://gobees.io/help/videos/monitorizacion-act-vuelo>

Para monitorizar la actividad de vuelo es necesario colocar el *smartphone* de forma fija en posición cenital a la colmena. Para esto, se puede utilizar un trípode o un soporte similar. En la siguiente imagen se puede ver un ejemplo de colocación:



Figura E.6: Colocación del *smartphone* en la colmena.

Para mejorar los resultados de la monitorización, es recomendable que el suelo sea de un color claro y uniforme. Si posee maleza, se puede colocar un cartón o similar, como se muestra en la imagen.

Una vez realizado en montaje, hay que seguir los siguientes pasos dentro de la aplicación:

1. Entrar en el colmenar al que pertenece la colmena a monitorizar.
2. Entrar en la colmena.
3. Pulsar en el botón de “monitorización” (situado en la parte inferior derecha con un icono de una cámara).
4. Se abrirá una ventana que permite previsualizar la monitorización.
5. Para configurar los parámetros de la monitorización, pulsar el botón “ajustes” (situado en la parte superior derecha). Se abrirá una pantalla con los siguientes ajustes:
 - **Mostrar salida del algoritmo:** si no se encuentra activado se previsualiza la imagen proveniente de la cámara. Si se activa, se muestran en verde las abejas detectadas y en rojo otros objetos en movimiento que el algoritmo no considera abejas. Además, en la esquina inferior derecha se puede visualizar el número total de abejas contadas en cada fotograma.
 - **Modificar el tamaño de las regiones:** dependiendo de la distancia a la que esté situada la cámara, es posible que las abejas se visualicen demasiado pequeñas o demasiado grandes. Con esta opción, se puede agrandar o disminuir su silueta.
 - **Min. área abeja:** la detección de una abeja se realiza por área. Si el contorno en movimiento detectado posee un área dentro de unos límites se considera una abeja. Este parámetro configura la cota inferior del área. Bien ajustado, permite descartar moscas y mosquitos.
 - **Max. área abeja:** configura la cota superior del área. Permite descartar la mayoría de animales que pueden habitar en el colmenar (avispones, roedores, lagartos o cualquier animal de mayor tamaño).
 - **Zoom:** permite configurar el zoom de la cámara para encuadrar la superficie deseada.
 - **Frecuencia de muestreo:** determina el intervalo de tiempo entre un fotograma analizado y el siguiente a analizar. Es decir, si se establece en 1 segundo, la aplicación captará y analizará un fotograma cada segundo. Cuanto mayor sea el intervalo menor será el consumo de batería.
6. Una vez configurados los parámetros correctamente, se puede iniciar la monitorización pulsado el botón blanco.

7. Se iniciará una cuenta atrás y comenzará la monitorización. Durante esta, la pantalla puede estar apagada para ahorrar batería. Se puede aprovechar la cuenta atrás para apagarla sin influir en la monitorización (al manipular el móvil siempre se producen trepidaciones).
8. Cuando se desee detener la monitorización, se debe pulsar el botón cuadrado rojo. Una vez pulsado, se guardará la grabación y se podrá acceder a los detalles de esta.

Nota: Si se posee alguna aplicación de ahorro de batería es imprescindible añadir una excepción a la aplicación GoBees para que esta se pueda ejecutar en segundo plano sin restricciones. Si no, la aplicación puede ser cerrada durante la monitorización.



Figura E.7: Ajustes de monitorización.

Ver los detalles de una grabación

Al monitorizar una colmena se genera lo que denominamos una grabación. Una grabación contiene los datos de actividad de vuelo de la colmena.

Video-tutorial: <http://gobees.io/help/videos/ver-grabacion>

Para ver los detalles de una grabación:

1. Entrar en el colmenar al que pertenece la colmena monitorizada.
2. Entrar en la colmena.
3. Pulsar en la grabación sobre la que se está interesado.
4. Se mostrará una pantalla con dos gráficos.

- a. El gráfico principal muestra la actividad de vuelo. En el eje de las Y se representa el número de abejas en vuelo y en las X los instantes de tiempo. Si se pulsa sobre un punto del gráfico, se obtiene la medida exacta en ese punto.
 - b. El gráfico inferior muestra la información meteorológica. Existe un selector con tres botones: temperatura, precipitaciones y viento. Según se presione en uno u otro, se muestra su gráfico correspondiente.
5. Con ambos gráficos se puede interpretar la actividad de vuelo de la colmena y determinar si es una actividad normal o la colmena necesita una intervención.

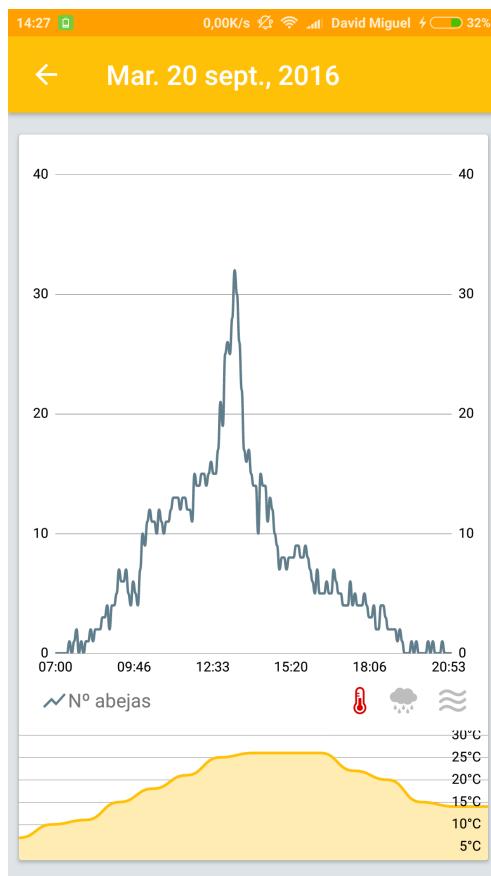


Figura E.8: Detalle de la grabación.

Eliminar una grabación

Al eliminar una grabación, se eliminan también todos los datos asociados a esta.

Video-tutorial: <http://gobees.io/help/videos/eliminar-grabacion>

Para eliminar una grabación existente:

1. Entrar en el colmenar al que pertenece la colmena monitorizada.
2. Entrar en la colmena.
3. Localizar la grabación y pulsar el botón de menú asociado a esta (tres puntos verticales situados en la esquina superior derecha).
4. Seleccionar la opción de eliminar.
5. La grabación se eliminará junto con toda su información.

Eliminar toda la información de la aplicación

Si por algún motivo se desea resetear toda la información almacenada en la aplicación, esta cuenta una opción para ello.

Video-tutorial: <http://gobees.io/help/videos/eliminar-datos>

Para eliminar toda la información de la aplicación:

1. Pulsar el botón menú.
2. Entrar en la sección “Ajustes”.
3. Seleccionar la opción “Borrar todos los datos”.
4. Todos los datos de la aplicación serán borrados. La aplicación volverá al mismo estado que cuando se instaló.

Consultar la información sobre la aplicación

Para conocer la versión instalada de la aplicación, los cambios introducidos en las diferentes versiones, la licencia o el autor de esta hay que acceder a la sección “Acerca de GoBees”.

Video-tutorial: <http://gobees.io/help/videos/acerca-gobees>

Para acceder a la sección “Acerca de GoBees”:

1. Pulsar el botón menú.
2. Entrar en la sección “Acerca de GoBees”.
3. En ella se puede visualizar la versión de la aplicación, el autor y las bibliotecas utilizadas para su desarrollo.
4. Si se presiona el botón “Website” se accede a la página web de GoBees.
5. Si se presiona el botón “Licencia” se visualiza una copia de la licencia de la aplicación.
6. Si se presiona el botón “Changelog” se visualizan los cambios introducidos en cada versión.



Figura E.9: Sobre GoBees.

Bibliografía

- [1] Android. Compilar y ejecutar tu app, 2017. URL <https://developer.android.com/studio/run/index.html?hl=es-419>. [Online; Accedido 24-Enero-2017].
- [2] Android. Importing an existing android project, 2017. URL <https://www.jetbrains.com/help/idea/2016.3/importing-an-existing-android-project.html>. [Online; Accedido 24-Enero-2017].
- [3] CodeClimate. User documentation, 2017. URL <https://docs.codeclimate.com/>. [Online; Accedido 24-Enero-2017].
- [4] CodeClimate. /davidmigloz/go-bees, 2017. URL <https://codeclimate.com/github/davidmigloz/go-bees>. [Online; Accedido 24-Enero-2017].
- [5] Codecov. User documentation, 2017. URL <https://docs codecov io/>. [Online; Accedido 24-Enero-2017].
- [6] Creative Commons. Attribution 4.0 international (cc by 4.0), 2016. URL <https://creativecommons.org/licenses/by/4.0/>. [Online; Accedido 07-Enero-2017].
- [7] Ministerio de Empleo y Seguridad Social. Bases y tipos de cotización 2016. URL http://www.seg-social.es/Internet_1/Trabajadores/CotizacionRecaudaci10777/Basesytiposdecotiza36537/index.htm. [Online; Accedido 20-Enero-2017].
- [8] Eclipse Foundation. Eclipse public license (epl), 2004. URL <https://www.eclipse.org/legal/epl-v10.html>. [Online; Accedido 07-Enero-2017].
- [9] Free Software Foundation. Gnu general public license v3.0, 2007. URL <https://www.gnu.org/licenses/gpl-3.0.txt>. [Online; Accedido 07-Enero-2017].

- [10] Martin Fowler. Gui architectures - mvc and mvp, 2006. URL <https://martinfowler.com/eaaDev/uiArchs.html>. [Online; Accedido 22-Enero-2017].
- [11] Git. Git, 2017. URL <https://git-scm.com/>. [Online; Accedido 24-Enero-2017].
- [12] GitHub. Cloning a repository, 2017. URL <https://help.github.com/articles/cloning-a-repository/>. [Online; Accedido 24-Enero-2017].
- [13] Google. Material design guidelines, 2014. URL <https://material.io/guidelines/>. [Online; Accedido 24-Enero-2017].
- [14] Google. Manage your project, 2016. URL <https://developer.android.com/studio/projects/index.html?hl=es>. [Online; Accedido 24-Enero-2017].
- [15] Google. Android studio, 2017. URL <https://developer.android.com/studio/index.html?hl=es>. [Online; Accedido 24-Enero-2017].
- [16] IEEE. IEEE SA - 830-1998 - IEEE Recommended Practice for Software Requirements Specifications. URL <https://standards.ieee.org/findstds/standard/830-1998.html>. [Online; Accedido 20-Enero-2017].
- [17] Java. Java se development kit 7, 2017. URL <http://www.oracle.com/technetwork/es/java/javase/downloads/jdk7-downloads-1880260.html>. [Online; Accedido 24-Enero-2017].
- [18] Stephan Linzner y Mustafa Kurtuldu Jose Alcérreca. Google en android architecture blueprints. URL <https://github.com/googlesamples/android-architecture>. [Online; Accedido 22-Enero-2017].
- [19] David Miguel Lozano. Aplicaciones de etiquetado y desarrollo del algoritmo, 2017. URL <https://github.com/davidmigloz/go-bees-prototypes/releases>. [Online; Accedido 24-Enero-2017].
- [20] David Miguel Lozano. Releases gobees, 2017. URL <https://github.com/davidmigloz/go-bees/releases>. [Online; Accedido 24-Enero-2017].
- [21] Edward Hieatt y Rob Mee Martin Fowler. Repository pattern. URL <https://martinfowler.com/eaaCatalog/repository.html>. [Online; Accedido 22-Enero-2017].
- [22] OpenCV. Opencv official website, 2017. URL <http://opencv.org/>. [Online; Accedido 24-Enero-2017].
- [23] SonarQube. User documentation, 2017. URL <https://docs.sonarqube.org/>. [Online; Accedido 24-Enero-2017].

- [24] SonarQube. /davidmigloz/go-bees, 2017. URL <https://sonarqube.com/dashboard?id=go-bees>. [Online; Accedido 24-Enero-2017].
- [25] Read the Docs. User documentation, 2017. URL <http://docs.readthedocs.io/en/latest/>. [Online; Accedido 24-Enero-2017].
- [26] Travis. User documentation, 2017. URL <https://docs.travis-ci.com/>. [Online; Accedido 24-Enero-2017].
- [27] Travis. davidmigloz / go-bees, 2017. URL <https://travis-ci.org/davidmigloz/go-bees>. [Online; Accedido 24-Enero-2017].
- [28] VersionEye. User documentation, 2017. URL <https://www.versioneye.com/documentation>. [Online; Accedido 24-Enero-2017].
- [29] VersionEye. /davidmigloz/go-bees, 2017. URL <https://www.versioneye.com/user/projects/57f7b19e823b88004e06ad33>. [Online; Accedido 24-Enero-2017].
- [30] Wikipedia. Dependency injection — wikipedia, the free encyclopedia, 2016. URL https://en.wikipedia.org/w/index.php?title=Dependency_injection&oldid=761366923. [Online; Accedido 07-Enero-2017].
- [31] Wikipedia. Licencia — wikipedia, the free encyclopedia, 2016. URL <https://es.wikipedia.org/w/index.php?title=Licencia&oldid=94243114>. [Online; Accedido 07-Enero-2017].



Este obra está bajo una licencia Creative Commons Reconocimiento 4.0 Internacional ([CC-BY-4.0](#)).