
GoBees Documentation

Publicación 1.0

David Miguel Lozano

16 de February de 2017

1. Resumen	3
1.1. Abstract	3
2. Introducción	5
2.1. Estructura de la memoria	6
2.2. Materiales adjuntos	6
3. Objetivos del proyecto	9
3.1. Objetivos generales	9
3.2. Objetivos técnicos	9
3.3. Objetivos personales	10
4. Conceptos teóricos	11
4.1. Preprocesado	11
4.2. Substracción del fondo	12
4.3. Posprocesado	16
4.4. Detección y conteo de abejas	17
5. Técnicas y herramientas	19
5.1. Metodologías	19
5.2. Patrones de diseño	19
5.3. Control de versiones	20
5.4. Hosting del repositorio	20
5.5. Gestión del proyecto	21
5.6. Comunicación	21
5.7. Entorno de desarrollo integrado (IDE)	21
5.8. Documentación	22
5.9. Servicios de integración continua	22
5.10. Sistemas de construcción automática del software	23
5.11. Librerías	24
5.12. Página web	25
5.13. Otras herramientas	26
6. Aspectos relevantes	27
6.1. Inicio del proyecto	27
6.2. Metodologías	28
6.3. Formación	29
6.4. Desarrollo del algoritmo	30
6.5. Desarrollo de la <i>app</i>	32

6.6.	<i>Testing</i>	34
6.7.	Documentación	35
6.8.	Publicación	36
6.9.	Reconocimientos	37
7.	Trabajos relacionados	39
7.1.	Artículos científicos	39
7.2.	Proyectos	40
7.3.	Fortalezas y debilidades del proyecto	40
8.	Conclusiones y Líneas de trabajo futuras	43
8.1.	Conclusiones	43
8.2.	Líneas de trabajo futuras	43
9.	Plan del proyecto software	45
9.1.	Introducción	45
9.2.	Planificación temporal	45
9.3.	Estudio de viabilidad	57
10.	Especificación de Requisitos	63
10.1.	Introducción	63
10.2.	Objetivos generales	63
10.3.	Catálogo de requisitos	64
10.4.	Especificación de requisitos	65
11.	Especificación de diseño	83
11.1.	Introducción	83
11.2.	Diseño de datos	83
11.3.	Diseño arquitectónico	85
11.4.	Diseño procedimental	94
11.5.	Diseño de interfaces	95
12.	Manual del programador	99
12.1.	Introducción	99
12.2.	Estructura de directorios	99
12.3.	Manual del programador	100
12.4.	Pruebas del sistema	113
13.	Manual de usuario	119
13.1.	Introducción	119
13.2.	Requisitos de usuarios	119
13.3.	Instalación	119
13.4.	Manual de usuario	122
	Bibliografía	137



Resumen

La actividad de vuelo de una colmena es un indicador importante sobre su estado de salud. Sin embargo, la monitorización manual de este parámetro es un proceso muy costoso y puede introducir una tasa de error elevada. Por ello, se han desarrollado diversos métodos que permiten automatizar este proceso.

En este trabajo se propone un nuevo método más accesible al público general que permite la monitorización de la actividad de vuelo de una colmena mediante la cámara de un *smartphone* Android.

Además, se ha desarrollado una aplicación de gestión de colmenares que integra el algoritmo y proporciona las herramientas necesarias para interpretar y organizar toda la información recabada.

GoBees es la aplicación resultante y se encuentra disponible a través de Google Play o de la página oficial del proyecto <http://gobees.io/>.

Descriptores: contador de abejas, actividad de vuelo, monitorización de colmenas, gestión de colmenares, aplicación Android.

Abstract

Flight activity of a honey bee colony is an overall indicator of the state of the hive's health. However, manually monitoring this parameter is a very expensive and time-consuming process and can introduce a high error rate. Thus, several methods to automate this process have been developed over the years.

In this work, we propose a new method more accessible to the general public which allows monitoring the flight activity of a honey bee colony using the built-in camera of an Android smartphone.

In addition, an apiary management application has been developed, incorporating the algorithm and providing the necessary tools to interpret and organize all the information gathered.

GoBees is the resulting application and it is available from Google Play or the official web site <http://gobees.io/>.

Keywords: bee counter, flight activity, hive monitoring, apiary management, Android application.

Introducción

Las abejas son una pieza clave en nuestro ecosistema. La producción de alimentos a nivel mundial y la biodiversidad de nuestro planeta dependen en gran medida de ellas. Son las encargadas de polinizar el 70 % de los cultivos de comida, que suponen un 90 % de la alimentación humana [art:bees_decline]. Sin embargo, la población mundial de abejas está disminuyendo a pasos agigantados en los últimos años debido, entre otras causas, al uso extendido de plaguicidas tóxicos, parásitos como la varroa o la expansión del avispon asiático [art:ccd].

Actualmente los apicultores inspeccionan sus colmenares de forma manual. Uno de los indicadores que más información les proporciona es la actividad de vuelo de la colmena (número de abejas en vuelo a la entrada de la colmena en un determinado instante) [art:campbell2008]. Este dato, 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.

La actividad de vuelo de una colmena varía dependiendo de múltiples factores, tanto externos como internos a la colmena. Entre ellos se encuentran la propia población de la colmena, las condiciones meteorológicas, la presencia de enfermedades, parásitos o depredadores, la exposición a tóxicos, la presencia de fuentes de néctar, etc. A pesar de los numerosos factores que influyen en la actividad de vuelo, su conocimiento es de gran ayuda para la toma de decisiones por parte del apicultor. Ya que este posee información sobre la mayoría de los factores necesarios para su interpretación.

La captación prolongada de la actividad de vuelo de forma manual es muy costosa, tediosa y puede introducir una tasa de error elevada. Es por esto que a lo largo de los años se haya intentado automatizar este proceso de muy diversas maneras. Los primeros intentos se remontan a principios del siglo XX, donde se desarrollaron contadores por contacto eléctrico [art:lundie1925]. Otros métodos posteriores se basan en sensores de infrarrojos [art:struye1994], sensores capacitivos [art:campbell2005], códigos de barras [beebarcodes] o incluso en microchips acoplados a las abejas [art:decourtye_honeybee_2011]. En los últimos años, se han desarrollado numerosos métodos basados en visión artificial [art:campbell2008] [art:chiron2013a] [art:chiron2013] [art:tashakkori2015].

Los métodos basados en contacto, sensores de infrarrojos o capacitivos tienen el inconveniente de que es necesario realizar modificaciones en la colmena, mientras que en los basados en códigos de barras o microchips es necesario manipular las abejas directamente. Estos motivos los convierten en métodos poco prácticos fuera del campo investigador. Por el contrario, la visión artificial aporta un gran potencial, ya que evita tener que realizar ningún tipo de modificación ni en el entorno, ni en las abejas. Además, abre la puerta a la monitorización de nuevos parámetros como la detección de enjambrazón (división de la colmena y salida de un enjambre) o la detección de amenazas (avispones, abejaruco, etc.).

Todos los métodos basados en visión artificial propuestos hasta la fecha utilizan hardware específico con un coste elevado. En este trabajo se propone un método de monitorización de la actividad de vuelo de una colmena mediante la cámara de un *smartphone* con Android.

El método propuesto podría revolucionar el campo de la monitorización de la actividad de vuelo de colmenas, ya que lo hace accesible al gran público. Ya no es necesario contar con costoso hardware, difícil de instalar. Solamente es necesario disponer de un *smartphone* con cámara y la aplicación GoBees. Además, esta facilita la interpretación de los

datos, representándolos gráficamente y añadiendo información adicional como la situación meteorológica. Permitiendo a los apicultores centrar su atención donde realmente es necesaria.

Estructura de la memoria

La memoria sigue la siguiente estructura:

- **Introducción:** breve descripción del problema a resolver y la solución propuesta. Estructura de la memoria y listado de materiales adjuntos.
- **Objetivos del proyecto:** exposición de los objetivos que persigue el proyecto.
- **Conceptos teóricos:** breve explicación de los conceptos teóricos clave para la comprensión de la solución propuesta.
- **Técnicas y herramientas:** listado de técnicas metodológicas y herramientas utilizadas para gestión y desarrollo del proyecto.
- **Aspectos relevantes del desarrollo:** exposición de aspectos destacables que tuvieron lugar durante la realización del proyecto.
- **Trabajos relacionados:** estado del arte en el campo de la monitorización de la actividad de vuelo de colmenas y proyectos relacionados.
- **Conclusiones y líneas de trabajo futuras:** conclusiones obtenidas tras la realización del proyecto y posibilidades de mejora o expansión de la solución aportada.

Junto a la memoria se proporcionan los siguientes anexos:

- **Plan del proyecto software:** planificación temporal y estudio de viabilidad del proyecto.
- **Especificación de requisitos del software:** se describe la fase de análisis; los objetivos generales, el catálogo de requisitos del sistema y la especificación de requisitos funcionales y no funcionales.
- **Especificación de diseño:** se describe la fase de diseño; el ámbito del software, el diseño de datos, el diseño procedimental y el diseño arquitectónico.
- **Manual del programador:** recoge los aspectos más relevantes relacionados con el código fuente (estructura, compilación, instalación, ejecución, pruebas, etc.).
- **Manual de usuario:** guía de usuario para el correcto manejo de la aplicación.

Materiales adjuntos

Los materiales que se adjuntan con la memoria son:

- Aplicación para Android GoBees.
- Aplicación Java para el desarrollo del algoritmo.
- Aplicación Java para el etiquetado de fotogramas.
- JavaDoc.
- *Dataset* de vídeos de prueba.

Además, los siguientes recursos están accesibles a través de internet:

- Página web del proyecto: <http://gobees.io/>
- GoBees en Google Play: <https://play.google.com/store/apps/details?id=com.davidmiguel.gobees>

- Repositorio del proyecto: <https://github.com/davidmigloz/go-bees>
- Repositorio de las herramientas desarrolladas para el proyecto: <https://github.com/davidmigloz/go-bees-prototypes>

Objetivos del proyecto

A continuación, se detallan los diferentes objetivos que han motivado la realización del proyecto.

Objetivos generales

- Desarrollar una aplicación para *smartphone* 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.

Objetivos técnicos

- Desarrollar un algoritmo de visión artificial con OpenCV que permita contar el número de abejas en cada fotograma en tiempo real.
- Desarrollar una aplicación Android con soporte para API 19 y superiores.
- Aplicar la arquitectura MVP (*Model-View-Presenter*) en el desarrollo de la aplicación.
- Utilizar Gradle como herramienta para automatizar el proceso de construcción de software.
- Utilizar Git como sistema de control de versiones distribuido junto con la plataforma GitHub.
- Hacer uso de herramientas de integración continua como Travis, Codecov, Code Climate, SonarQube o VersionEye en el repositorio.
- Aplicar la metodología ágil Scrum junto con TDD (*Test Driven Development*) en el desarrollo del software.
- Realizar test unitarios, de integración y de interfaz.
- Utilizar ZenHub como herramienta de gestión de proyectos.
- Utilizar un sistema de documentación continua como Read the Docs.
- Distribuir la aplicación resultante en la plataforma Google Play.
- Realizar una página web para la difusión de la aplicación.

Objetivos personales

- Realizar una aportación a la modernización de la apicultura.
- Abarcar el máximo número de conocimientos adquiridos durante la carrera.
- Explorar metodologías y herramientas novedosas utilizadas en el mercado laboral.
- Adentrarme en el campo de la visión artificial.
- Profundizar en el desarrollo de aplicaciones Android.

Conceptos teóricos

La parte del proyecto con mayor complejidad teórica radica en el algoritmo de visión artificial, el cual se puede dividir en cuatro fases. En primer lugar, se realiza un preprocesado de la señal de entrada para optimizarla. En segundo lugar, se sustrae el fondo para segmentar los objetos en movimiento. Posteriormente, se realiza un posprocesado de la señal para mejorar los resultados obtenidos de la sustracción del fondo. Y por último, se clasifican y cuentan los contornos que pueden pertenecer a una abeja.

A continuación se exponen los conceptos teóricos que conlleva cada fase.

Preprocesado

Antes de aplicar el algoritmo de sustracción del fondo, es recomendable realizar un preprocesado de los fotogramas para facilitar el procesamiento posterior, minimizar el ruido y optimizar los resultados. A continuación se explican las técnicas utilizadas.

Conversión de RGB a escala de grises

Los fotogramas captados por la cámara se devuelven en formato RGB. En este formato se poseen tres matrices, una por cada canal (*Red-Green-Blue*). La suma aditiva de los tres canales resulta en una imagen a color.

Sin embargo, el color no proporciona ninguna información relevante en nuestra tarea de identificación de abejas. Es por esto que se pueden convertir los fotogramas de RGB a escala de grises. De esta manera, se trabajará solamente con una matriz de píxeles en lugar de tres. Simplificando, en gran medida, el número de operaciones a realizar y por tanto, aumentando el rendimiento de nuestro algoritmo final.

OpenCV utiliza la conversión colométrica a escala de grises [[opencv:color_cvt](#)]. Esta técnica se basa en principios colométricos para ajustar la luminancia de la imagen a color y la imagen resultante. Devolviendo una imagen con la misma luminancia absoluta y la misma percepción de luminosidad [[wiki:grayscale](#)].

Utiliza la siguiente fórmula para calcular la luminancia resultante:

$$\text{RGB[A] to Gray: } Y \leftarrow 0,299 \cdot R + 0,587 \cdot G + 0,114 \cdot B$$

La fórmula calcula la luminancia de una forma no lineal, sin necesidad de realizar una expansión gamma. Los coeficientes intentan imitar la percepción de intensidad medida por un humano tricromático, siendo más sensible al color verde y menos al color azul [[wiki:grayscale](#)].

Desenfoque Gaussiano

Las imágenes captadas pueden contener ruido que puede dificultar su procesamiento. El ruido son variaciones aleatorias del brillo o del color de una imagen. Una técnica que permite reducirlo es el desenfoque.

En nuestro caso hemos utilizado desenfoque Gaussiano, un filtro de paso bajo que reduce las componentes de alta frecuencia de la imagen utilizando para ello una convolución con una función Gaussiana [\[wiki:gaussian\]](#). Se diferencia del desenfoque promedio en que da más peso a los vecinos cercanos, siendo estos más influyentes en el resultado.

El kernel utilizado en la convolución es una muestra discreta de la función Gaussiana. En nuestro caso utilizamos un kernel 3x3 que se corresponde con: [\[book:mastering_opencv\]](#)

$$M = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

En la siguiente imagen se puede ver el resultado de aplicar esta fase a la imagen de entrada:



Substracción del fondo

En un sistema de monitorización por vídeo resulta de gran interés el poder extraer los objetos en movimiento del resto de la imagen. Esto se conoce como extracción del fondo, en inglés *background subtraction* o *foreground detection*, y consiste en clasificar todos los píxeles de un determinado fotograma bien como fondo, o como primer plano [\[wiki:bs\]](#). En primer plano se engloban todos los objetos en movimiento, mientras que en el fondo se encuentran todos los objetos estáticos junto con posibles sombras, cambios de iluminación u otros objetos en movimiento que no son de interés, como puede ser la rama de un árbol balanceándose por el viento. Se trata de un paso crítico, ya que algoritmos posteriores dependen en gran medida de los resultados de este.

En nuestro proyecto, la toma de imágenes se realiza mediante una cámara estática. Esto facilita en parte la detección del fondo, ya que este se corresponderá con todos los píxeles estáticos. Sin embargo, como trabajamos en un entorno al aire libre, tenemos que lidiar también con cambios de iluminación, sombras, u otros objetos móviles que no son de nuestro interés (falsos positivos).

Con OpenCV para Android podemos implementar varios algoritmos básicos de extracción del fondo y además, nos proporciona la implementación de dos algoritmos más sofisticados: `BackgroundSubtractorMOG` y `BackgroundSubtractorKNN`. Tras realizar un estudio de todos ellos, nos decantamos finalmente por `BackgroundSubtractorMOG2`. A continuación, explicamos el funcionamiento de todos los algoritmos que se probaron, así como los resultados que nos proporcionaron.

Substracción con imagen de referencia

Se parte de una imagen de referencia del fondo, en la que no haya ningún objeto en movimiento. A partir de esta, se obtienen los elementos en movimiento substrayendo a cada fotograma la imagen tomada como referencia.

Este método, al tomar un modelo del fondo tan sencillo y estático, es muy vulnerable a cambios en la escena (iluminación, sombras, objetos del fondo con ligeros movimientos, pequeñas oscilaciones de la cámara, etc.). Sin embargo, ofrece muy buenos resultados cuando se trabaja en una escena con la iluminación y los elementos controlados, ya que al ser tan simple, es muy eficiente [\[programarfacil:detmov\]](#).

Para implementar este algoritmo con OpenCV, se hace uso de la función `Core.absdiff()`.

En nuestro problema, al trabajar al aire libre nos es imposible utilizar este algoritmo. Ya que el modelo del fondo cambia constantemente.

Substracción del fotograma anterior

En este método, el modelo del fondo se extrae del fotograma anterior. De tal manera, que a cada nuevo fotograma se le subtrae el anterior.

De esta manera se mejora la respuesta a cambios en la escena, como los cambios de iluminación. Sin embargo, si un objeto en movimiento se queda estático en la imagen, este deja de ser detectado [\[book:opencv_java\]](#).

La implementación se realiza como en la técnica anterior, variando el modelo del fondo.

Tras probarlo en nuestro problema específico, vimos que no nos era de utilidad. Ya que el fotograma resultante de la diferencia contenía las abejas por duplicado (la abeja en el fotograma actual y misma abeja en el fotograma anterior en otra posición).

Substracción del acumulado de los fotogramas anteriores

Una mejora interesante del algoritmo anterior supone tomar como modelo del fondo un acumulado de los fotogramas anteriores de acuerdo a un ratio de aprendizaje. De esta forma, se puede lidiar con cambios en el fondo de la imagen dinámicamente. El modelo se calcula de acuerdo a la siguiente fórmula:

$$u_t = (1 - \alpha)u_{t-1} + \alpha p_t$$

Donde p_t es el nuevo valor del píxel, u_{t-1} es la media del fondo en el instante $t - 1$, u_t es la nueva media del fondo y α es el ratio de aprendizaje (cómo de rápido olvida los frames anteriores) [\[book:opencv_java\]](#).

OpenCV provee la función `Imgproc.accumulateWeighted()` que implementa la fórmula anterior por nosotros. Haciendo uso de esta función y de la utilizada en la sección anterior podemos implementar este algoritmo.

Tras probarlo, vimos que tenía una eficiencia muy buena y se adaptaba a los cambios correctamente. Sin embargo, no era capaz de diferenciar las sombras de las abejas, por lo que se obtenían falsos positivos.

BackgroundSubtractorKNN

Se trata de un método que se basa en el algoritmo de clasificación supervisada *K nearest neighbors* (k-nn). El algoritmo fue propuesto en el artículo [\[art:zivkovic_efficient_2006\]](#). Y de acuerdo con sus conclusiones, es muy eficiente cuando el número de píxeles que se corresponden con el primer plano es bajo.

La clase de OpenCV que lo implementa es `BackgroundSubtractorKNN`. Los parámetros más importantes son:

- `history`: número de fotogramas recientes que afectan al modelo del fondo.
- `dist2Threshold`: umbral de la distancia al cuadrado entre el píxel y la muestra para decidir si un píxel está cerca de esa muestra.
- `detectShadows`: con un valor verdadero detecta las sombras (aumenta considerablemente el tiempo de procesamiento).

En nuestras pruebas, el algoritmo proporcionaba unos resultados buenos pero su tiempo de ejecución era muy elevado (entorno a 25ms/frame). Como el tiempo de ejecución es un factor clave en nuestro proyecto, se descartó el uso de este algoritmo.

BackgroundSubtractorMOG2

BackgroundSubtractorMOG2 es una mejora del algoritmo BackgroundSubtractorMOG. En la versión original de OpenCV se encuentran implementados ambos, sin embargo, en los *wrappers* para Android solo disponemos de la revisión.

BackgroundSubtractorMOG2 está basado en el modelo *Gaussian Mixture* (GMM). Se trata de un modelo compuesto por la suma de varias distribuciones Gaussianas que, correctamente elegidas, permiten modelar cualquier distribución [*coursera:gmm*].

El algoritmo de substracción del fondo fue propuesto en el artículo [*art:yao_improved_2001*] y modela cada píxel del fondo como la mezcla de K distribuciones Gaussianas. Los pesos de la mezcla representan las proporciones de tiempo que el color de ese píxel se ha mantenido en la escena. Siendo los colores de fondo más probables los que más permanezcan y sean más estáticos [*opencv:bs_tutorial*].

BackgroundSubtractorMOG2 se basa en los mismos principios que su antecesor pero implementa una mejora sustancial. Es el propio algoritmo el que selecciona el número adecuado de distribuciones Gaussianas necesarias para modelar cada píxel. De esta manera, se mejora notablemente la adaptabilidad del algoritmo a variaciones en la escena. Fue propuesto en los artículos [*art:zivkovic_improved_2004*] y [*art:zivkovic_efficient_2006*].

El código fuente de este algoritmo está disponible en [*github:background_segm*] (interfaz) y [*github:bgfg_gaussmix2*] (implementación).

La clase de OpenCV que lo implementa es BackgroundSubtractorMOG2. Posee los siguientes parámetros configurables: [*opencv:mog*]

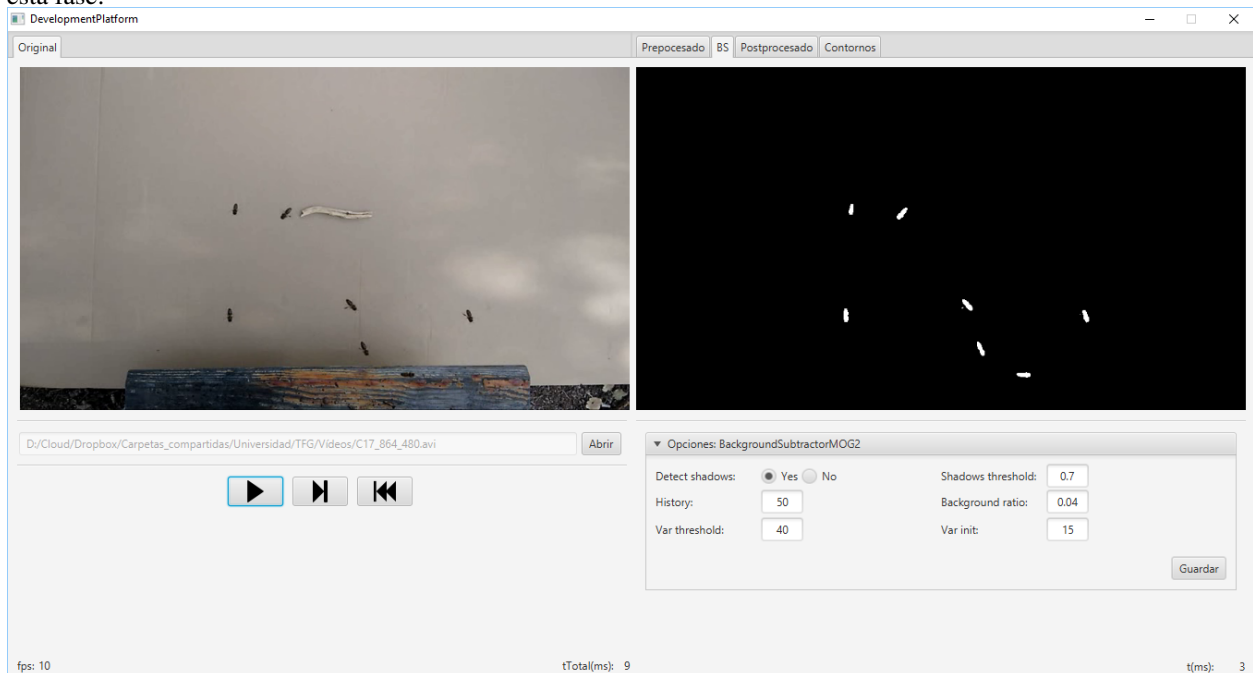
- **history**: número de fotogramas recientes que afectan al modelo del fondo. Se representa en la literatura como T . Por defecto, 500 fotogramas. Nosotros hemos obtenido buenos resultados con valores de entorno a 50.
- **learningRate**: valor entre 0 y 1 que indica como de rápido aprende el modelo. Si se establece un valor de -1 el algoritmo elige automáticamente el ratio. 0 significa que el modelo del fondo no se actualiza para nada, mientras que 1 supone que el modelo del fondo se reinicializa completamente cada nuevo fotograma. En la literatura podemos encontrar este parámetro como α . Si el intervalo que se quiere considerar es **history**, se debe establecer $\alpha=1/\text{history}$ (valor por defecto). También se pueden mejorar los resultados iniciales estableciendo $\alpha=1$ en el instante 0 e ir decreméntándolo hasta $\alpha=1/\text{history}$. De esta manera, en el inicio aprende rápidamente, pero una vez estabilizada la situación las variaciones afectan menos al modelo. En nuestro caso, el valor por defecto ha funcionado correctamente.
- **backgroundRatio**: si un píxel del primer plano permanece con un valor semi-constante durante $\text{backgroundRatio} \cdot \text{history}$ fotogramas, es considerado fondo y se añade al modelo del fondo como centro de una nueva componente Gaussiana. En los artículos se hace referencia a este parámetro como T_B . $T_B=0.9$ es el valor por defecto. Este parámetro nos permite decidir cuando dejar de contar una abeja que se ha quedado inmóvil o un objeto nuevo en la escena como podría ser una hoja que se acaba de caer de un árbol.
- **detectShadows**: con un valor verdadero (valor por defecto) detecta las sombras (aumenta ligeramente el tiempo de procesado). Nos permite despreciar las sombras de las abejas con muy buenos resultados.
- **shadowThreshold**: el algoritmo detecta las sombras comprobando si un píxel es una versión oscurecida del fondo. Este parámetro define cómo de oscura puede ser la sombra como máximo. Por ejemplo, un valor de 0.5 (valor por defecto) significa que si un píxel es más del doble de oscuro, entonces no se considerará sombra. En los artículos se representa como τ .
- **shadowValue**: es el valor utilizado para marcar los píxeles de sombras en la máscara resultante. El valor por defecto es 127. En la máscara devuelta, un valor de 0 siempre se corresponde con un píxel del fondo, mientras que un valor de 255 con un píxel del primer plano.

- **nMixtures**: número máximo de componentes Gaussianas para modelar el modelo del fondo. El número actual se determina dinámicamente para cada píxel. Hemos utilizado el valor por defecto, 5.
- **varThreshold**: umbral utilizado en el cálculo de la distancia cuadrada de Mahalanobis entre el píxel y el modelo del fondo para decidir si una muestra está bien descrita por el modelo o no. Este parámetro no afecta a la actualización del modelo del fondo. Se representa como `Cthr`. Por defecto, 16. Se han obtenido mejores resultados con valores de entorno a 40.
- **varThresholdGen**: umbral sobre la distancia cuadrada de Mahalanobis entre el píxel y el modelo para ayudar a decidir si un píxel está cercano a alguna de las componentes del modelo. Si no es así, es considerado como primer plano o añadido como centro de una nueva componente (dependiendo del `backgroundRatio`). Se representa como `Tg` y su valor por defecto es 9. Un valor menor genera más componentes Gaussianas, mientras que un valor mayor genera menos.
- **complexityReductionThreshold**: este parámetro define el número de muestras necesarias para probar que una componente existe. Se representa como `CT`. Su valor por defecto es `CT=0.05`. Si se establece su valor a 0 se obtiene un algoritmo similar al de Stauffer & Grimson (no se reduce el número de componentes).
- **varInit**: varianza inicial de cada componente Gaussianas. Afecta a la velocidad de adaptación. Se debe ajustar teniendo en cuenta la desviación estandar de las imágenes. Por defecto es 15.
- **varMin**: varianza mínima. Por defecto, 4.
- **varMax**: varianza máxima. Por defecto, $5 \times \text{varInit}$.

De todos ellos, los parámetros más importantes a ajustar son `history` o `learningRate`, `varThreshold` y `detectShadows`.

La parametrización correcta de este algoritmo es clave para su buen funcionamiento. Por ello, durante las pruebas se integró en nuestra aplicación de desarrollo, permitiendo variar todos estos parámetros en tiempo real. De esta manera, se pudo elegir la mejor configuración para nuestro problema concreto.

En la siguiente imagen se puede ver una captura de nuestra plataforma de desarrollo en la pestaña correspondiente a esta fase:



Una vez parametrizado correctamente, vimos como este algoritmo era el que mejores resultados nos proporcionaba. Con un tiempo de ejecución en nuestro equipo de pruebas de entorno a 4ms/frame, mucho menor que el proporcionado por `BackgroundSubtractorKNN`, de entorno a 25ms/frame. El algoritmo detectaba correctamente las abejas, era

resistente al ruido, y además, era capaz de diferenciar una abeja de su sombra. Por todos estos motivos, se seleccionó para la fase de substracción del fondo.

Otros algoritmos

La implementación original de OpenCV implementa otros dos algoritmos más que no están disponibles a través de los *wrappers* de Android.

- `BackgroundSubtractorGMG` es un algoritmo que combina una estimación estadística del fondo de la imagen junto con una segmentación Bayesiana píxel a píxel [[opencv:bs_tutorial](#)].
- `BackgroundSubtractorFGD` está disponible en la versión para CUDA. Utiliza la regla de decisión de Bayes para clasificar los elementos del fondo y los del primer plano atendiendo a sus vectores de características [[art:li_foreground_2003](#)].

En la siguiente imagen se puede ver el resultado de aplicar `BackgroundSubtractorMOG2` a la salida de la fase anterior:



Se puede apreciar como ha descartado correctamente las sombras en movimiento de los árboles y se ha quedado únicamente con los objetos en movimiento.

Posprocesado

Para mejorar los resultados de la extracción de fondo y preparar la imagen para la búsqueda de contornos, se han aplicado las siguientes técnicas:

Dilatación

Se trata de una operación morfológica por la cual se expanden las regiones luminosas de una imagen. Esto se consigue mediante la sustitución de cada píxel por el más brillante de los vecinos considerados por el *kernel* (matriz utilizada para la convolución). De esta manera se consiguen unir las regiones de abejas que podían haberse roto [[book:mastering_opencv](#)].

Erosión

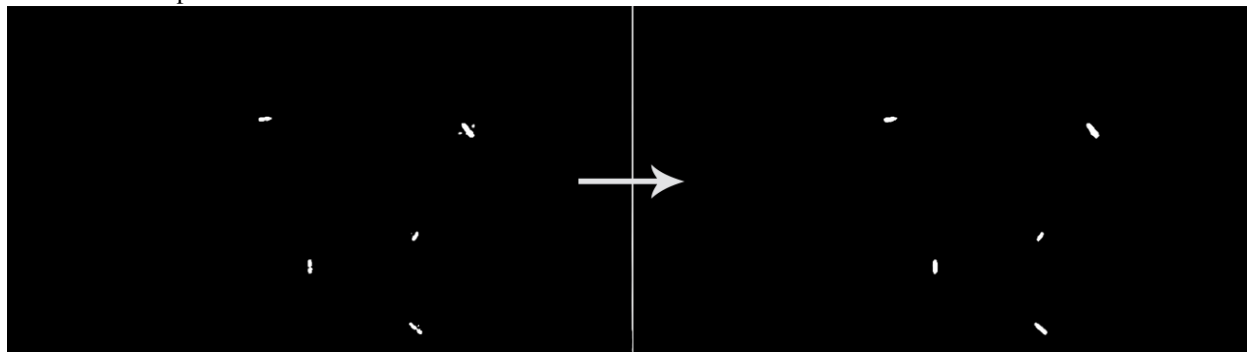
Se trata de la operación contraria a la anterior, expande las regiones oscuras de la imagen. Para ello se coge el valor mínimo de los valores considerados por el *kernel* [[book:mastering_opencv](#)].

La dilatación nos permite reconstruir las abejas, pero también aumenta su tamaño, aumentando el riesgo de solapamientos. Para evitar esto, se vuelve a reducir el tamaño de estas mediante una erosión.

En nuestro algoritmo aplicamos tres operaciones morfológicas seguidas:

1. **Erosión (3x3)**: elimina las piernas de las abejas.
2. **Dilatación (2x2)**: junta la cabeza de las abejas con su cuerpo que en numerosas ocasiones es separado durante la substracción de fondo.
3. **Erosión (3x3)**: recupera el tamaño inicial.

A continuación podemos ver el resultado de esta fase:



Detección y conteo de abejas

El último paso que realiza nuestro algoritmo de visión artificial es detectar cuáles de las regiones obtenidas en la fase anterior se corresponden con abejas. Para ello, se realiza una búsqueda de contornos y se filtran por área.

Entendemos por contorno una línea curva que une todos los puntos continuos del borde de una región de un mismo color o intensidad.

La salida de la fase anterior es una imagen binaria con los objetos en movimiento en blanco y el fondo en negro. Por lo tanto, el objetivo de esta fase es detectar todas las regiones blancas que puedan corresponderse con una abeja.

OpenCV provee la función `Imgproc.findContours()` para realizar la búsqueda de contornos. Esta toma una imagen binaria y devuelve una lista con todos los contornos encontrados. Para entender la función se necesita comprender una serie de conceptos: [\[opencv:contour\]](#)

- **Jerarquía**: los contornos pueden ser independientes unos de otros, o poseer una relación padre-hijo cuando un contorno está dentro de otro. En la jerarquía se especifican las relaciones entre contornos.
- **Modo de obtención del contorno**: define cómo se van a obtener los contornos en cuestión de jerarquía [\[opencv:find_contour\]](#).
 - `RETR_LIST`: devuelve todos los contornos en una lista, sin ninguna información de jerarquía entre ellos.
 - `RETR_EXTERNAL`: devuelve todos los contornos externos. Si algún contorno tiene contornos hijo, estos son ignorados.
 - `RETR_CCOMP`: devuelve los contornos agrupados en dos niveles de jerarquía. Un primer nivel en el que se encuentran todos los contornos exteriores. Y un segundo nivel con los contornos correspondientes a agujeros en los primeros.
 - `RETR_TREE`: devuelve todos los contornos creando un árbol completo con la jerarquía.
- **Método de aproximación de los contornos**: define el método que utiliza la función para almacenar los contornos [\[opencv:find_contour\]](#).
 - `CHAIN_APPROX_NONE`: almacena todos los puntos del borde del contorno.
 - `CHAIN_APPROX_SIMPLE`: almacena sólo los puntos relevantes del contorno. Por ejemplo, si el contorno es una línea no se necesita almacenar todos los puntos de esta, con el punto inicial y el final basta. Esto es

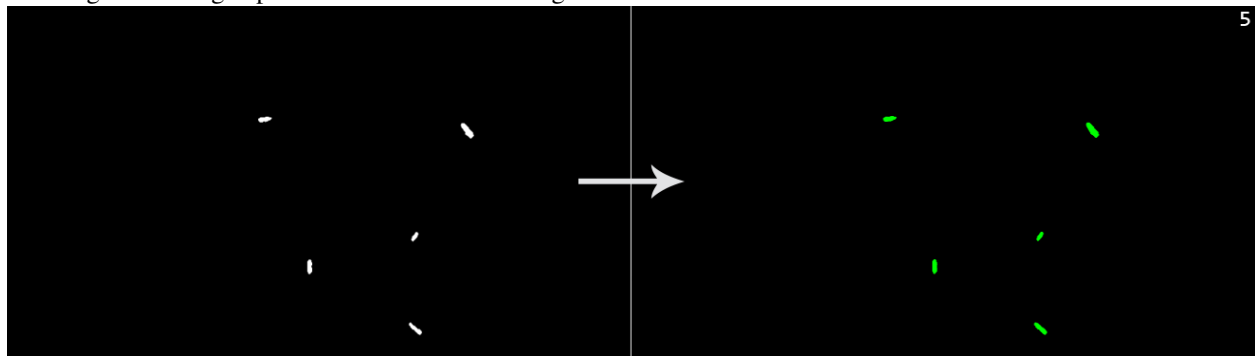
lo que realiza este método, eliminar todos los puntos redundantes y comprimirlos para que ocupe menos espacio.

- `CV_CHAIN_APPROX_TC89_L1` y `CV_CHAIN_APPROX_TC89_KCOS`: aplican el algoritmo de aproximación de cadena de Teh-Chin, simplificando los polígonos que forman los contornos.
- `CV_CHAIN_CODE`: almacena los contornos utilizando el código de cadenas de Freeman.

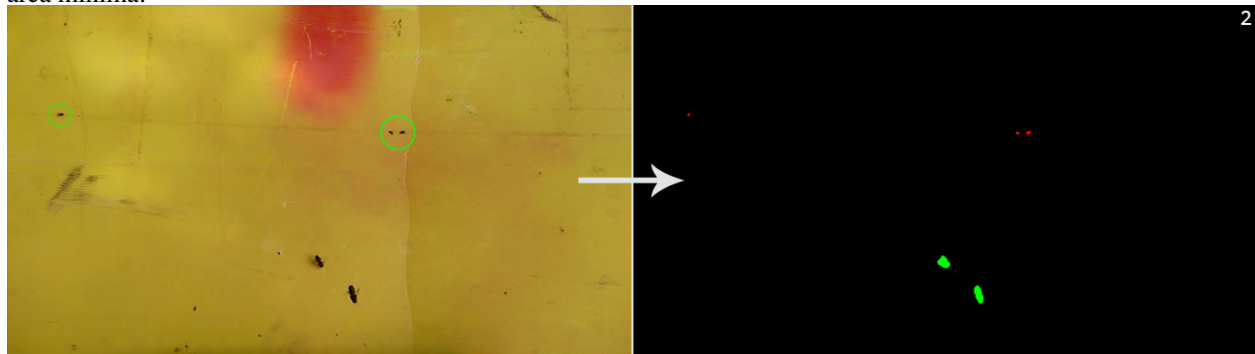
En nuestro caso, la configuración más adecuada es utilizar `RETR_EXTERNAL` y `CHAIN_APPROX_SIMPLE`. Ya que no nos interesa ningún contorno interno que pueda tener la abeja (y que en principio no debería tener) y tampoco nos es relevante el cómo se almacenan estos, sólo nos interesa el número.

Para evitar posibles falsos positivos, establecemos un umbral mínimo y máximo en el área del contorno. De esta manera, evitamos que contornos diminutos o grandes generados por ruidos o por objetos del entorno (moscas, pájaros, roedores...) sean contados como abejas.

En la siguiente imagen podemos ver la salida del algoritmo:



En esta otra se puede apreciar como se descartan las tres moscas que hay en la imagen ya que su área es inferior al área mínima:



Técnicas y herramientas

Metodologías

Scrum

Scrum es un marco de trabajo para el desarrollo de software que se engloba dentro de las metodologías ágiles. Aplica una estrategia de trabajo iterativa e incremental a través de iteraciones (*sprints*) y revisiones.

Test-Driven Development (TDD)

Es una práctica de desarrollo de software que se basa en la repetición de un ciclo corto de desarrollo: transformar requerimientos a test, desarrollar el código necesario para pasar los test y posteriormente refactorizar el código. Esta práctica obliga a los desarrolladores a analizar cuidadosamente las especificaciones antes de empezar a escribir código, fomenta la escritura de test y la simplicidad de código y aumenta la productividad. Como resultado se obtiene software más seguro y de mayor calidad.

Gitflow

Gitflow es un flujo de trabajo con Git que define un modelo estricto de ramas diseñado en torno a los lanzamientos de proyecto. En la rama *main* se hospeda la última versión estable del proyecto. La rama *develop* contiene los últimos desarrollos realizados para el siguiente lanzamiento. Por cada característica que se vaya a implementar se crea una *feature branch*. La preparación del siguiente lanzamiento se realiza en una *release branch*. Si aparece un fallo en producción, este se soluciona en una *hotfix branch*.

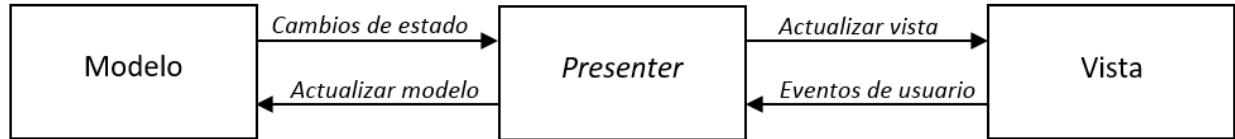
Patrones de diseño

Model-View-Presenter (MVP)

MVP es un patrón de arquitectura derivado del *Model-View-Controller* (MVC). Permite separar los datos internos del modelo de una vista pasiva y enlazarlos mediante el *presenter* que maneja toda la lógica de la aplicación. Posee tres capas:

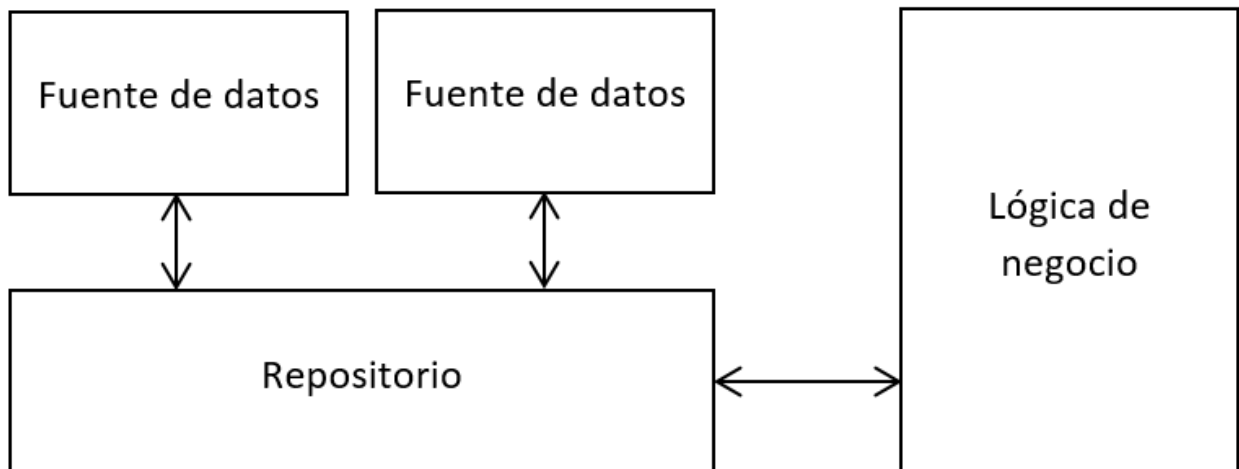
- **Model:** almacena y proporciona los datos internos.
- **View:** maneja la visualización de los datos (del modelo). Propaga las acciones de usuario al *presenter*.

- *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.



Patrón repositorio

El patrón repositorio proporciona una abstracción de la implementación del acceso a datos con el objetivo de que este sea transparente a la lógica de negocio de la aplicación. Por ejemplo, las fuentes de datos pueden ser una base de datos, un *web service*, etc. 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.



Control de versiones

- Herramientas consideradas: [Git](#) y [Subversion](#).
- Herramienta elegida: [Git](#).

Git es un sistema de control de versiones distribuido. A día de hoy, es el sistema con mayor número de usuarios. La principal diferencia con Subversion es su carácter descentralizado, que permite a cada desarrollador tener una copia en local del repositorio completo. Git está licenciado bajo la licencia de software libre GNU LGPL v2.1.

Hosting del repositorio

- Herramientas consideradas: [GitHub](#), [Bitbucket](#) y [GitLab](#).
- Herramienta elegida: [GitHub](#).

GitHub es la plataforma web de hospedaje de repositorios por excelencia. Ofrece todas las funcionalidades de Git, revisión de código, documentación, *bug tracking*, gestión de tareas, *wikis*, red social... y numerosas integraciones con otros servicios. Es gratuita para proyectos *open source*.

Utilizamos GitHub como plataforma principal donde hospedamos el código del proyecto, la gestión de proyecto (gracias a ZenHub) y la documentación. Además, el repositorio está integrado con varios servicios de integración continua.

Gestión del proyecto

- Herramientas consideradas: [ZenHub](#), [Trello](#), [Waffle](#), [VersionOne](#), [XP-Dev](#) y [GitHub Projects](#).
- Herramienta elegida: [ZenHub](#).

ZenHub es una herramienta de gestión de proyectos totalmente integrada en GitHub. Proporciona un tablero canvas en donde cada tarea representada se corresponde con un *issue* nativo de GitHub. Cada tarea se puede priorizar dependiendo de su posición en la lista, se le puede asignar una estimación, uno o varios responsables y el *sprint* al que pertenece. ZenHub también permite visualizar el gráfico *burndown* de cada *sprint*. Es gratuita para proyectos pequeños (máx. 5 colaboradores) o proyectos *open source*.

Comunicación

- Herramientas consideradas: email y [Slack](#).
- Herramienta elegida: [Slack](#).

Slack es una herramienta de colaboración de equipos que ofrece salas de chat, mensajes directos y llamadas VoIP. Posee un buscador que permite encontrar todo el contenido generado dentro de Slack. Además, ofrece un gran número de integraciones con otros servicios. En nuestro proyecto vamos a utilizar la integración con GitHub para crear un canal que sirva de log de todas las acciones realizadas en GitHub. Slack ofrece una versión gratuita que provee las características principales.

Entorno de desarrollo integrado (IDE)

Java

- Herramientas consideradas: [IntelliJ IDEA](#) y [Eclipse](#).
- Herramienta elegida: [IntelliJ IDEA](#).

IntelliJ IDEA es un IDE para Java desarrollado por JetBrains. Posee un gran número de herramientas para facilitar el proceso de escritura, revisión y factorización del código. Además, permite la integración de diferentes herramientas y posee un ecosistema de *plugins* para ampliar su funcionalidad. Su versión *community* está disponible bajo la licencia Apache 2. Aunque también es posible adquirir la versión *Ultimate* gratuitamente si se es estudiante.

Android

- Herramientas consideradas: [Android Studio](#) y [Eclipse](#).
- Herramienta elegida: [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. Se distribuye bajo la licencia Apache 2.

Markdown

- Herramientas consideradas: [StackEdit](#) y [Haroopad](#).
- Herramienta elegida: [Haroopad](#).

Haroopad es un editor de documentos Markdown. Soporta Github Flavored Markdown y Mathematics Expression, además de contar con un gran número de extensiones. Se distribuye bajo licencia GNU GPL v3.0.

LaTeX

- Herramientas consideradas: [ShareLaTeX](#) y [Texmaker](#).
- Herramienta elegida: [Texmaker](#).

Texmaker es un editor gratuito y multiplataforma para LaTeX. Integra la mayoría de herramientas necesarias para la escritura de documentos en LaTeX (PdfLaTeX, BibTeX, makeindex, etc). Además, incluye corrector ortográfico, auto-completado, resaltado de sintaxis, visor de PDFs integrado, etc. Está licenciado bajo GNU GPL v2.

Documentación

- Herramientas consideradas: [Microsoft Word](#), [LibreOffice](#), [LaTeX](#), [Markdown](#), [GitHub Wikis](#).
- Herramienta elegida: [Markdown](#) + [LaTeX](#).

La documentación se ha desarrollado en Markdown para integrarla con el servicio de documentación continua [Read the Docs](#). Una vez terminada, se ha exportado a LaTeX utilizando el conversor [Pandoc](#).

Markdown es un lenguaje de marcado ligero en texto plano que puede ser exportado a numerosos formatos como HTML o PDF. Su filosofía es que el lenguaje de marcado sea fácil de escribir y leer. Markdown es ampliamente utilizado para la escritura de archivos README, en foros como StackOverflow o en herramientas de comunicación como Slack.

LaTeX es un sistema de composición de textos que genera documentos con una alta calidad tipográfica. Es ampliamente utilizado para la generación de artículos y libros científicos, principalmente por su potencia a la hora de representar expresiones matemáticas.

Servicios de integración continua

Compilación y testeo

- Herramientas consideradas: [TravisCI](#) y [CircleCI](#).
- Herramienta elegida: [TravisCI](#).

Travis 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. Es gratuita para proyectos *open source*.

Cobertura de código

- Herramientas consideradas: [Coveralls](#) y [Codecov](#).
- Herramienta elegida: [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. Posee una extensión de navegador para GitHub que permite visualizar por cada archivo de código que líneas están cubiertas por un test y cuáles no. Es gratuita para proyectos *open source*.

Calidad del código

- Herramientas consideradas: [Codeclimate](#), [SonarQube](#) y [Codacy](#).
- Herramientas elegidas: [Codeclimate](#) y [SonarQube](#).

Codeclimate es una herramienta que realiza revisiones de código automáticamente. Es gratuita para proyectos *open source*. En nuestro proyecto hemos activado los siguientes motores de chequeo: [checkstyle](#), [fixme](#), [markdownlint](#) y [pmd](#).

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 tests unitarios, *bugs* potenciales, etc.

Revisión de dependencias

- Herramientas consideradas: [VersionEye](#).
- Herramienta elegida: [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. Posee una versión gratuita con ciertas limitaciones.

Documentación

- Herramientas consideradas: [Read the Docs](#).
- Herramienta elegida: [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. La página web posee un buscador, da soporte para diferentes versiones del proyecto y soporta internacionalización. Además, permite exportar la documentación en varios formatos (pdf, epub, html, etc.). El servicio es totalmente gratuito, sostenido por donaciones y suscripciones *Gold*.

Sistemas de construcción automática del software

Maven

[Maven](#) es una herramienta para automatizar el proceso de construcción del *software* (compilación, testeo, empaquetado, etc.) enfocada a proyectos Java. Básicamente describe cómo se tiene que construir el *software* y cuáles son sus dependencias.

Gradle

Gradle es una herramienta similar a Maven, pero basada en el lenguaje de programación orientado a objetos Groovy. El sistema de construcción de Android Studio está basado en Gradle y es actualmente el único soportado de forma oficial para Android.

Librerías

Android Support Library

La **librería de soporte de Android** facilita algunas características que no se incluyen en el *framework* oficial. Proporciona compatibilidad a versiones antiguas con las últimas características, incluye elementos para la interfaz adicionales y utilidades extra.

Espresso

Espresso es un *framework* de *testing* para Android incluido en la librería de soporte para *testing* en Android. Provee una API para escribir UI test que simulen las interacciones de usuario con la app.

Google Guava

Google Guava agrupa un conjunto de librerías comunes para Java. Proporciona utilidades básicas para tareas cotidianas, una extensión del *Java collections framework* (JCF) y otras extensiones como programación funcional, almacenamiento en caché, objetos de rango o *hashing*.

Google Play Services

Google Play Services es una librería que permite a las aplicaciones de terceros utilizar características de aplicaciones de Google como Maps, Google+, etc. En nuestro caso se ha hecho uso de su servicio de localización, que utiliza varias fuentes de datos (GPS, red y wifi) para ubicar el dispositivo rápidamente.

JavaFX

JavaFX es una librería para la creación de interfaces gráficas en Java.

JUnit

JUnit es un *framework* para Java utilizado para realizar pruebas unitarias.

Material Design

Material Design es una guía de estilos enfocada a la plataforma Android, pero aplicable a cualquier otra plataforma. Fue presentada en el Google I/O 2014 y se adoptó en Android a partir de la versión 5.0 (Lollipop). Se basa en objetos materiales, piezas colocadas en un espacio (lugar) y con un tiempo (movimiento) determinado.

Mockito

Mockito es un *framework* de *mocking* que permite crear objetos *mock* fácilmente. Estos objetos simulan parte del comportamiento de una clase. Mockito está basado en EasyMock, mejorando su sintaxis haciendo los test más simples y fáciles de leer y con mensajes de error descriptivos.

MPAndroidChart

MPAndroidChart es una librería para la creación de gráficos en Android.

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. Se distribuye gratuitamente bajo una licencia *BSD* desde hace más de una década. Posee una comunidad de más de 50.000 usuarios alrededor de todo el mundo y se ha descargado más de 8 millones de veces.

Aunque OpenCV está escrito en C/C++ posee *wrappers* para varias plataformas, entre ellas Android, en donde da soporte a las principales arquitecturas de CPU. Desde hace unos años, también soporta CUDA para el desarrollo en GPU tanto en escritorio como en móvil, aunque en esta última el soporte es todavía reducido.

OpenWeatherMaps

OpenWeatherMap es un servicio online que proporciona información meteorológica. Está inspirado en OpenStreetMap y su filosofía de hacer accesible la información a la gente de forma gratuita. Utiliza distintas fuentes de datos desde estaciones meteorológicas oficiales, de aeropuertos, radares e incentiva a los propietarios de estaciones meteorológicas a conectarlas a su red. Proporciona una API que permite realizar hasta 60 llamadas por segundo de forma gratuita.

PowerMock

PowerMock es una librería de *testing* que permite la creación de *mocks* de métodos estáticos, constructores, clases finales o métodos privados.

Realm

Realm es una base de datos orientada a objetos enfocada a dispositivos móviles. Se definen como la alternativa a SQLite y presumen de ser más rápidos que cualquier ORM e incluso que SQLite puro. Posee una API muy intuitiva que facilita en gran medida el acceso a datos.

Página web

GitHub Pages

GitHub Pages es un servicio de hosting estático que permite hospedar la página del proyecto en su propio repositorio de GitHub. Permite utilizar Jekyll, un generador de sitios estáticos. No soporta tecnologías del lado de servidor como PHP, Ruby, Python, etc.

Bootstrap

[Bootstrap](#) es un *framework* para desarrollo *front-end*. Contiene una serie de componentes ya implementados que facilitan y agilizan diseño. Está desarrollado siguiendo la filosofía *mobile first*.

Otras herramientas

Mendeley

[Mendeley](#) es un gestor de referencias bibliográficas. Permite añadir referencias de varias formas, visualizar los documentos, etiquetarlos, compartirlos, etc. Posteriormente se puede exportar todo el catálogo a un fichero BibTex para ser utilizadas desde LaTeX.

Creately

[Creately](#) es una aplicación web que permite crear todo tipo de diagramas altamente personalizables. Aunque posee una versión gratuita limitada, se optó por pagar un mes de suscripción al valorar que realmente iba a ser utilidad.

Aspectos relevantes

En este apartado se van a recoger los aspectos más importantes del desarrollo del proyecto. Desde las decisiones que se tomaron y sus implicaciones, hasta los numerosos problemas a los que hubo que enfrentarse y cómo se solucionaron.

Inicio del proyecto

La idea del proyecto surgió del afán de encontrar un tema con el que poder aunar mi formación técnica y mis aficiones.

Mi padre me transmitió el interés por la apicultura desde bien pequeño y, posteriormente, llegué a trabajar en una empresa de apicultura profesional. Esto me hizo ser consciente de los problemas con los que lidia día a día un apicultor. Por otro lado, los conocimientos adquiridos durante mis estudios de Ingeniería Informática, me posibilitaron idear soluciones tecnológicas a alguno de estos problemas.

Tras formalizar la idea del proyecto y recibir el visto bueno de los tutores, nos pusimos manos a la obra.



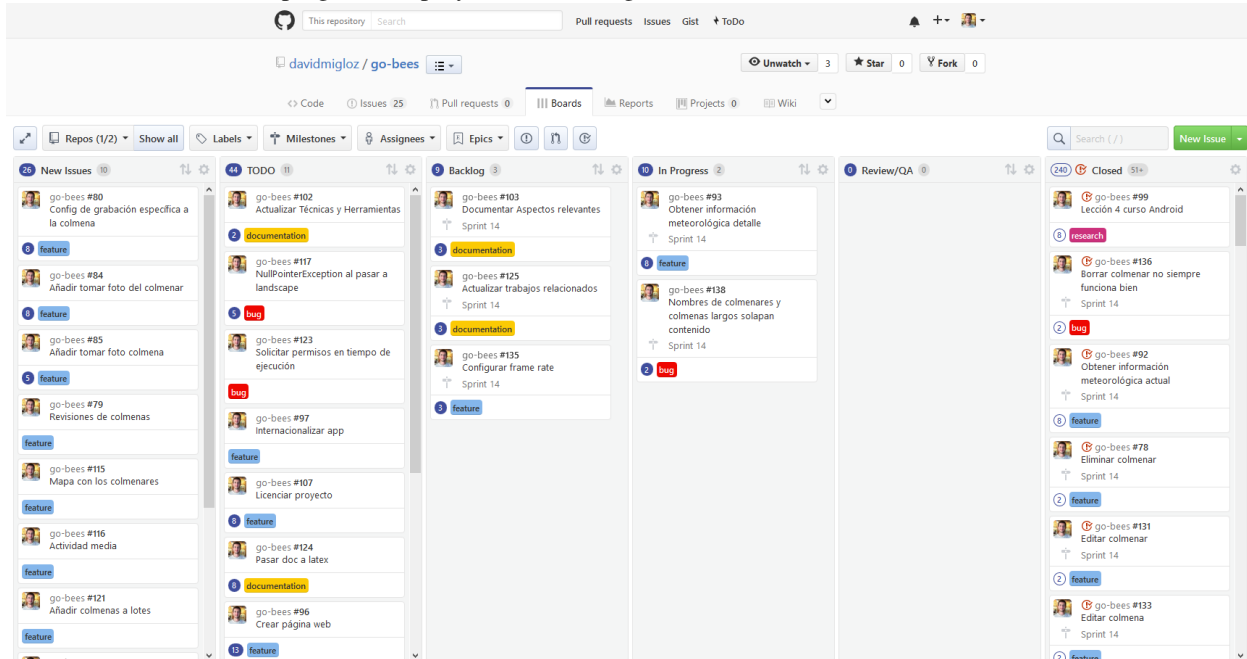
Metodologías

Desde el primer momento se dedicaron grandes esfuerzos para que la realización del proyecto se llevase a cabo de la manera más profesional posible. Para ello, se siguieron varias metodologías y procesos, expuestos a continuación.

Para la gestión del proyecto se utilizó una metodología ágil, en concreto Scrum. Aunque no se siguió al 100 % al tratarse de un proyecto educativo (no éramos un equipo de 4 a 8 personas, no hubo reuniones diarias, etc.), sí que se aplicó una filosofía ágil en líneas generales:

- Se siguió 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* (ZenHub).
- Para monitorizar el progreso del proyecto se utilizó gráficos *burndown*.



Para el diseño del algoritmo de visión artificial se utilizó una metodología de ensayo y error. Se barajaban diferentes alternativas y se iban verificando empíricamente su eficacia y eficiencia.

Para el desarrollo de la aplicación Android se intentó utilizar *Test-Driven Development* (TDD) para fomentar la escritura de test y mejorar la calidad del *software*. La mayoría de los módulos de la aplicación se implementaron exitosamente siguiendo esta metodología. Sin embargo, en las partes más complejas de esta, como el servicio de monitorización, se hacía muy engorroso la escritura de test (ya complicada de por sí en Android), por lo que finalmente se omitieron al penalizar notablemente la productividad.

Formación

El proyecto requería una serie de conocimientos técnicos de los que no se disponía en un principio. Sobre todo, relacionados con visión artificial, OpenCV y Android. A continuación, se enumeran los principales recursos didácticos que se utilizaron.

Para la formación en visión artificial y OpenCV se leyeron los siguientes libros:

- *Android Application Programming with OpenCV 3* (Joseph Howse) [[book:android_opencv](#)].
- *Mastering OpenCV Android Application Programming* (Salil Kapur y Nisarg Thakkar) [[book:mastering_opencv](#)].
- *Learning Image Processing with OpenCV* (Ismael Serrano Gracia, Jesús Salido Tercero y José Luis Espinosa Aranda) [[book:learning_cv](#)].
- *OpenCV 3.0 Computer Vision with Java* (Daniel Lélis Baggio) [[book:opencv_java](#)].

Para la formación en Android se realizaron los siguientes cursos online:

- *Android Development for Beginners: How to Make Apps* (Udacity) [*course:android_beginners*].
- *Developing Android Apps* (Udacity) [*course:developing_android*].
- *Android Testing Codelab* (Google) [*course:testing*].

También cabe destacar la importancia que tuvo la comunidad [StackOverflow](#) para la resolución de los diferentes problemas que surgieron durante el desarrollo.

Desarrollo del algoritmo

Una gran parte de los recursos del proyecto se dedicaron al desarrollo del algoritmo de visión artificial para la monitorización de la actividad de vuelo de una colmena.

El problema a resolver poseía una serie de condicionantes que dificultaban el análisis:

- Cada abeja ocupa una porción muy pequeña de la imagen.
- Las condiciones lumínicas varían a lo largo del día o de la época del año.
- Existen sombras producidas por la cámara o por las propias abejas.
- A 20 fps una abeja volando puede recorrer una distancia significativa entre fotogramas.
- Un grupo de abejas puede estar confinado, dificultando su segmentación.

El desarrollo comenzó con una búsqueda bibliográfica sobre el tema. En esta, se encontraron varios artículos relacionados que nos dieron una idea de las técnicas que podíamos probar. También se tuvo una reunión con un experto en visión artificial que nos proporcionó su punto de vista sobre cómo abordar el problema.

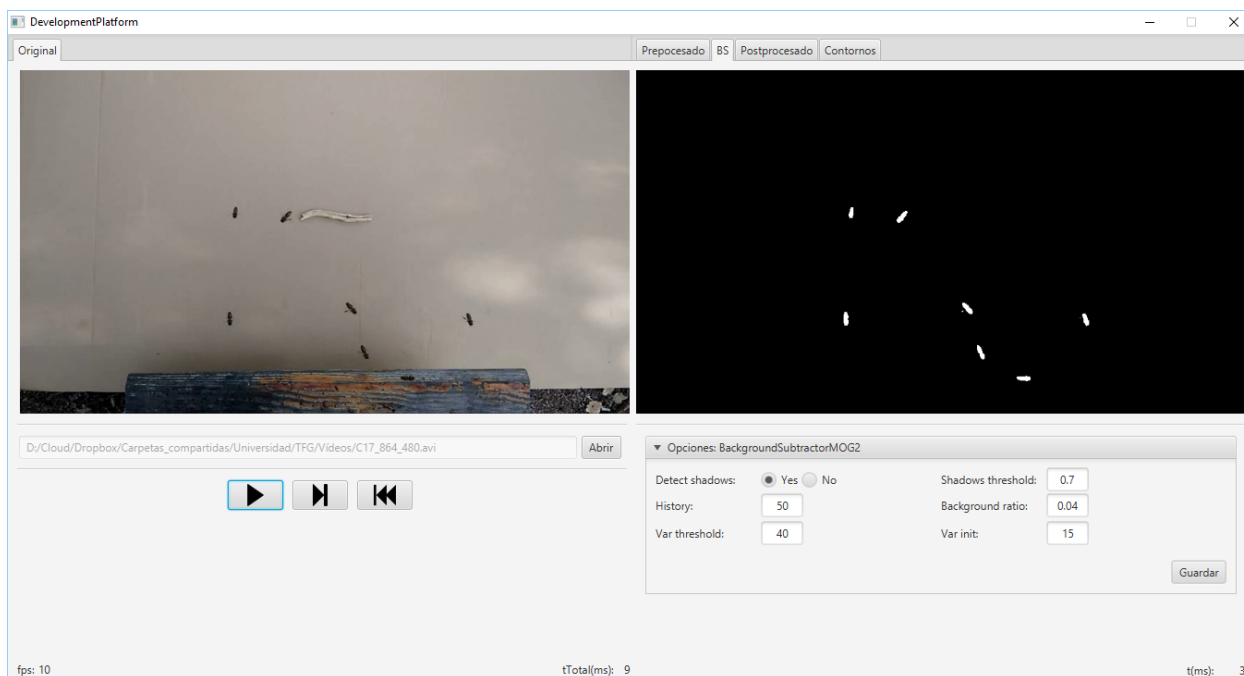
El primer punto a abordar fue la toma de las imágenes. Se decidió que la toma se debía de hacer con un trípode en posición cenital a la colmena. De esta manera, se disminuía las diferencias de tamaño por perspectiva, no se obstaculizaba el vuelo de las abejas y se facilitaba el análisis. Además, era aconsejable cubrir el suelo con alguna superficie uniforme para mejorar la segmentación de las abejas.



En un primer momento, el desarrollo del algoritmo se iba a realizar directamente sobre la plataforma Android. Sin embargo, rápidamente nos dimos cuenta que no era lo más adecuado si queríamos seguir una metodología basada en el ensayo y error, debido a los elevados tiempos de compilación y a la poca flexibilidad de la plataforma.

Finalmente se decidió desarrollar el algoritmo directamente en Java, ya que nos proporcionaba una mayor flexibilidad y, además, migrar el código a Android posteriormente sería una tarea bastante trivial.

Para facilitar el desarrollo del algoritmo y del testeo de las diferentes alternativas se realizó una aplicación Java que permitía parametrizar en tiempo real las diferentes etapas del algoritmo.



Además de permitir variar la parametrización, la aplicación nos permitía observar la salida de cada etapa del algoritmo y su tiempo de computación.

Es importante recalcar que los tiempos de computación obtenidos en la plataforma había que multiplicarlos por cuatro si se quería obtener una estimación de lo que equivaldría ejecutarlo en el dispositivo móvil. Por lo que los márgenes con los que se jugaba no eran demasiado grandes.

Tras varias iteraciones, se consiguió obtener un algoritmo bastante robusto que poseía las siguientes etapas:

1. **Preprocesado:** conversión de RGB a escala de grises y desenfoque Gaussiano para facilitar el procesamiento.
2. **Substracción del fondo:** extracción de los elementos en movimiento utilizando el algoritmo BackgroundSubtractorMOG2.
3. **Postprocesado:** mejora de la salida de la etapa anterior mediante varias iteraciones de erosión y dilatación.
4. **Detección y conteo de abejas:** localización de los contornos pertenecientes a abejas en base al área y conteo de los mismos.

Desarrollo de la app

El desarrollo de la aplicación Android se realizó de forma incremental, publicando una *release* al finalizar cada *sprint*.

Se decidió dar soporte a partir de la versión KitKat (API 19), de tal forma que es cubría un 86,3 % de los dispositivos Android del mercado [[android:versions](#)]. Dar soporte a versiones anteriores suponía una sobrecarga importante, por lo que se descartó al no constituir una cuota de mercado importante.

La primera tarea consistió en migrar el algoritmo de visión artificial a la plataforma Android. A primera vista, no parecía una tarea complicada. Sin embargo, nos encontramos con varios *bugs* que, unidos a la mala documentación de OpenCV para Android, dificultó considerablemente la labor.

El primer *bug* (*issue #26*) consistía en que al llamar a algunas funciones del núcleo de OpenCV, se producía un error con un mensaje que hacía referencia a una señal que variaba entre ejecuciones, lo cual hizo muy difícil encontrar el origen. Finalmente, se dio con la fuente del problema, el cual tenía su origen en que la versión de la aplicación OpenCV Manager distribuida en el Google Play contaba con una versión corrupta de la librería OpenCV 3.1.

Se notificó al equipo de OpenCV a través de su gestor de incidencias y finalmente solucionaron el problema en la nueva versión OpenCV 3.2.

El segundo *bug* (*issue* #27) hacía que la aplicación fallase cuando se compilaba para dispositivos con una versión de Android inferior a Lollipop, debido a la API de la cámara. Tras publicar el caso en [StackOverflow](#), un usuario sugirió que podía estar relacionado con el *Instant Run* de Android Studio. Y así fue; desactivando esta característica, la aplicación no fallaba. Se describió el caso en el gestor de incidencias de Android y, a día de hoy (enero 2017), el *bug* se encuentra resuelto y está a la espera de ser incorporado en futuras *releases* del *framework*.

Una vez solventados ambos *bugs*, se logró que el algoritmo funcionase sin problemas en la nueva plataforma.

Cabe destacar que como OpenCV no distribuía oficialmente la librería a través de ningún repositorio que permitiese utilizarla directamente como dependencia Gradle, se creó uno propio [*gobees:prototipes*].

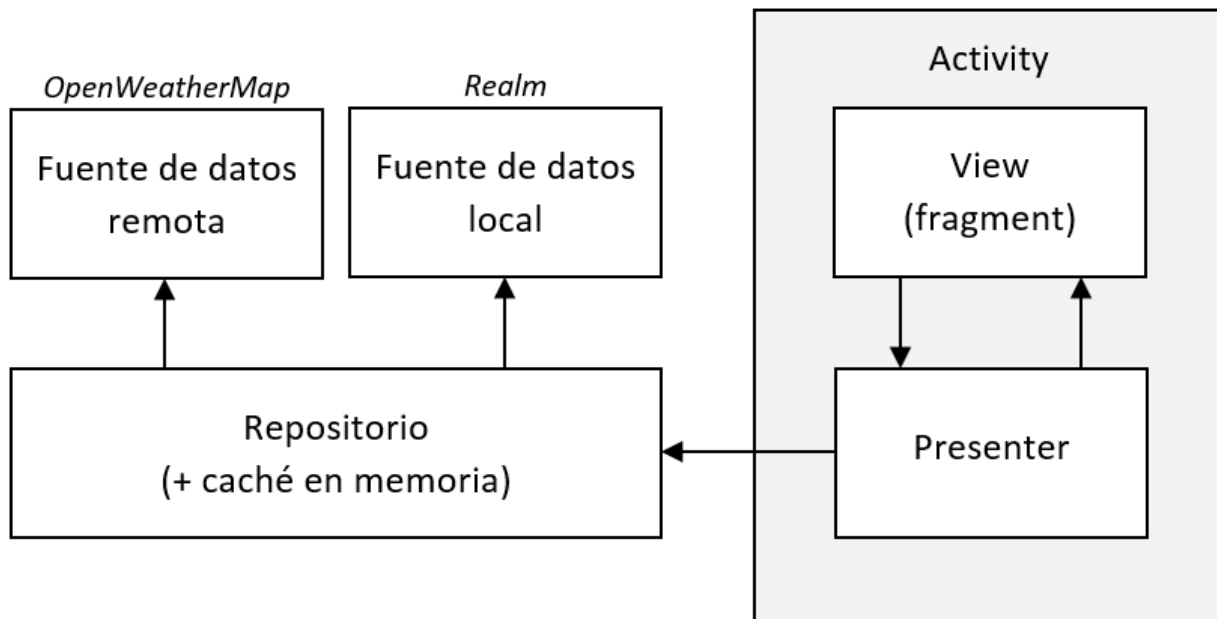
Para el diseño de la arquitectura de la *app* se siguió el patrón de arquitectura *Model-View-Presenter* (MVP), que permite separar los datos internos del modelo de una vista pasiva y agrupar toda la lógica de la aplicación en una capa intermedia llamada *presenter*. De esta manera se consigue un código muy desacoplado, haciendo que este sea más fácil de testear y mantener.

En cuanto a la persistencia de datos, se optó por utilizar Realm. Se trata de una base de datos orientada a objetos que proporciona una API para trabajar directamente con la capa de persistencia. Es multiplataforma y presume de ser más rápida que SQLite.

Para la obtención de la información meteorológica se escogió la API proporcionada por *OpenWeatherMaps*, la cual nos permitía realizar hasta 60 llamadas por minuto de forma gratuita.

El acceso a datos se centralizó utilizando el patrón repositorio, que abstrae la lógica de negocio de la fuente de datos. Todo el acceso a datos se centraliza en el repositorio y es este quien decide de qué fuente los obtiene (base de datos, internet, etc.). Además, se incorporó una capa de caché en este punto con el fin de agilizar la navegación por la *app*.

El siguiente esquema resume la arquitectura de la aplicación:



Surgieron problemas al implementar el algoritmo de monitorización como un servicio de Android para que el usuario pudiese apagar la pantalla durante la monitorización. El acceso a la cámara proporcionado por OpenCV era en sí una vista, y las vistas no pueden ser utilizadas en servicios. Finalmente se optó por realizar una implementación propia que accediese directamente a la API de la cámara y convirtiese los fotogramas al formato de OpenCV.

En cuanto al diseño de la aplicación, se dedicaron grandes esfuerzos a la usabilidad y accesibilidad de la misma. Se siguieron las directrices de diseño recogidas en la guía de Material Design en cuanto a estilos, disposición de los

elementos, tipos de componentes, patrones de navegación, gestos, etc.

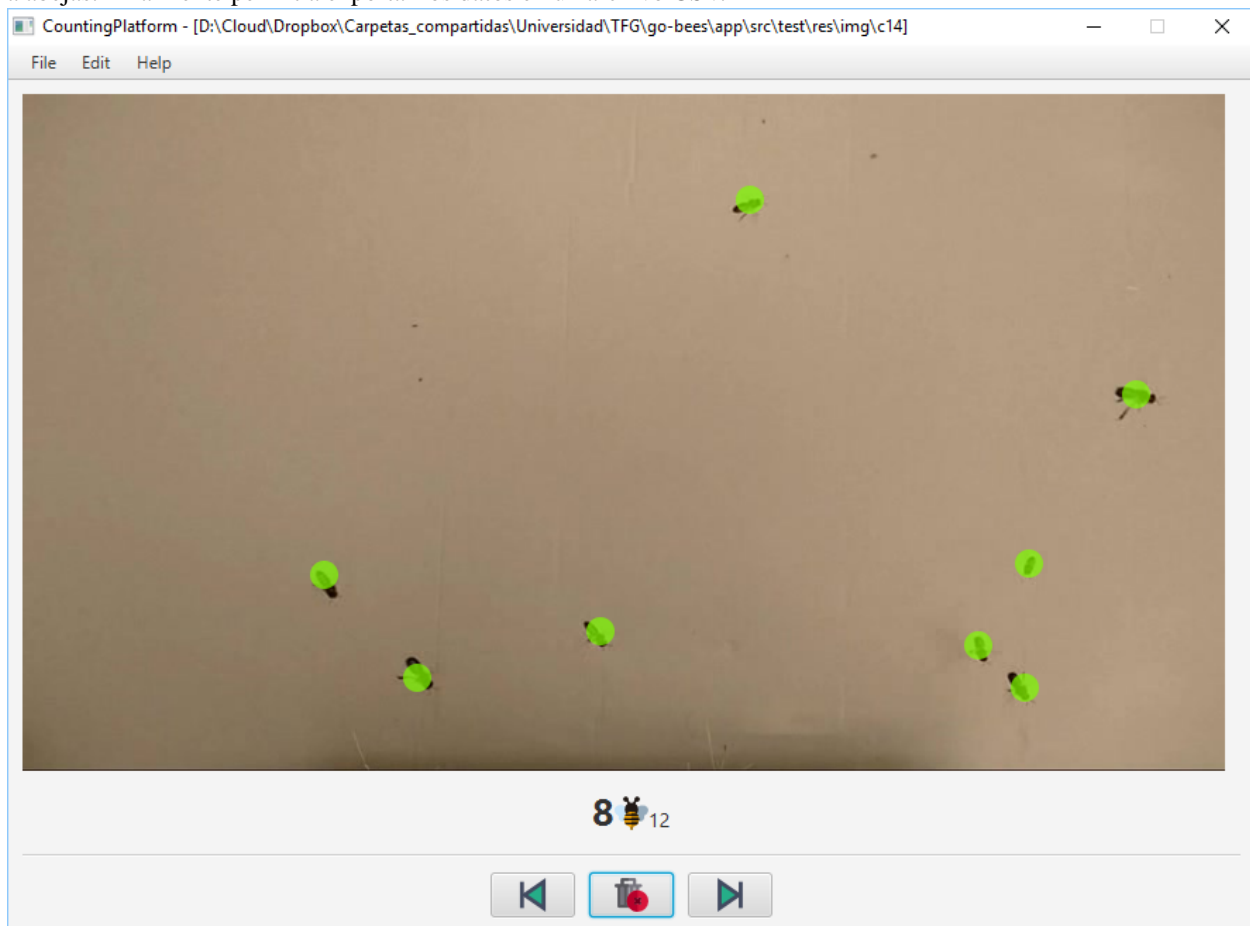
Por último, se internacionalizó la aplicación a los siguientes idiomas: español, inglés, catalán, polaco y árabe. Para ello se utilizó la herramienta Toolkit del Traductor de Google que permite realizar una primera traducción automática de los diferentes textos de la aplicación y posteriormente una revisión colaborativa de los resultados de esta. Para la revisión se recurrió a amistades nativas en los diferentes idiomas.

Testing

En lo relativo al *testing*, podemos diferenciar el testeo del algoritmo del testeo de la *app*.

Para testear el error cometido por el algoritmo era necesario contar con fragmentos de vídeo etiquetados con el número de abejas presentes en cada fotograma. Como esta labor era muy tediosa, se desarrolló una aplicación Java para agilizar el etiquetado de los fotogramas.

La aplicación iba mostrando los diferentes fotogramas al usuario y este indicaba con el ratón los píxeles pertenecientes a abejas. Finalmente permitía exportar los datos en un archivo CSV.



Se etiquetaron fragmentos de vídeo correspondientes a tres situaciones distintas:

- **Caso 1** (vídeo #C14): actividad media, sombras de abejas y moscas.
- **Caso 2** (vídeo #C17): actividad media, cambios de iluminación y sombras de árboles.
- **Caso 3** (vídeo #C5): alta actividad, solapamiento, sombras y fondo no uniforme.

Tras ejecutar el algoritmo se obtuvieron los siguientes errores relativos (considerando como error absoluto la distancia entre la medida esperada y la obtenida):

- **Caso 1:** 2,43 %.
- **Caso 2:** 0,89 %.
- **Caso 3:** 4,48 %.

Se puede observar que en la situación menos favorable el error es menor a un 5 %, precisión más que suficiente para la finalidad de los datos.

En cuanto al tiempo medio de ejecución del algoritmo para cada fotograma era de 25 ms en el equipo de pruebas (Intel i7-3610QM) y de 100 ms cuando se ejecutaba en un *smartphone* (Xiaomi Mi4).

Por otra parte, la aplicación se testeó mediante test unitarios, test de integración y test de interfaz. La mayor parte de los test unitarios se realizaron contra los *presenters*, que son los que poseen la lógica de negocio de la *app*. Los test de interfaz se ejecutaron en siete dispositivos diferentes, uno por cada versión de Android que soporta la *app* (API 19-25).

Además, se configuraron una serie de servicios de integración continua, de tal forma, que cada vez que se realizaba un *commit* en el repositorio, se ejecutaban las siguientes tareas:

1. **Travis:** realizaba una compilación del proyecto, ejecutaba los test unitarios, ejecutaba *Lint*, ponía en marcha un emulador de Android, y ejecutaba los Android test sobre este. Al finalizar, enviaba los resultados a Codecov y SonarQube.
2. **Codecov:** realizaba un análisis sobre la cobertura de los test unitarios.
3. **CodeClimate:** ejecutaba cuatro motores de chequeo (*checkstyle*, *fixme*, *pmd* y *markdownlint*) sobre el código para detectar posibles problemas o vulnerabilidades en él.
4. **SonarQube:** analizaba código duplicado, violaciones de estándares, cobertura de tests unitarios, bugs potenciales, etc.
5. **VersionEye:** chequeaba todas las dependencias utilizadas en la aplicación y comprobaba si estaban actualizadas, si tenían algún problema de seguridad conocido, o si violaban la licencia del proyecto.

Shipping faster with ZenHub build passing codecov 33% code climate 4.0 quality gate passing dependencies up to date docs develop

Por último, cabe comentar algunas estadísticas del proyecto:

Concepto	Valor
Total de líneas	38.068
Líneas de código	57 %
Comentarios	35 %
Líneas en blanco	9 %
Número de clases	126
Número de XML	86
Número de test unitarios	x
Número de test de interfaz	x
Cobertura total test unitarios	14 %
Cobertura test unitarios algoritmo	100 %
Cobertura test unitarios <i>presenters</i>	100 %

Documentación

En un primer momento, se decidió escribir la documentación en formato Markdown, utilizando las *wikis* que proporciona GitHub en el repositorio. De esta forma, la documentación podía ser visualizada directamente desde GitHub, sin

necesidad de tener que compilarla con cada modificación.

Posteriormente, se optó por implementar un sistema de documentación continua integrado en el repositorio, en concreto ReadTheDocs. De tal forma, que la documentación se escribía en archivos Markdown dentro del repositorio y este servicio generaba una página web (go-bees.readthedocs.io) que se actualizaba cada vez que se realizaba un *commit*.

No obstante, los tutores preferían obtener la documentación en formato PDF para su corrección, por lo que finalmente se optó por utilizar Sphinx junto con ReadTheDocs.

Sphinx es un generador de documentación que permite exportar la documentación en varios formatos, entre ellos HTML y PDF. Desafortunadamente, no soportaba Markdown como formato de entrada, por lo que hubo que migrar la documentación al formato reStructuredText. Esta conversión se realizó con la herramienta Pandoc.

Con todo configurado, ahora ReadTheDocs generaba automáticamente la página web y un PDF actualizado con los últimos cambios realizados en la documentación.



Para la exportación final de la memoria se utilizó el conversor Pandoc, con objeto de transformar la documentación del formato reStructuredText a LaTeX. Algunos elementos, como las citas o las imágenes, no eran convertidos correctamente, por lo que se tuvo que hacer uso de expresiones regulares.

La totalidad de este tedioso proceso se realizó bajo la idea de que cualquier proyecto comercial tiene su documentación accesible desde una página web, y que además necesita estar acorde con la versión del proyecto. Sin embargo, para este caso concreto en el que el entregable final es un PDF con un formato determinado, el montar todo este sistema ha supuesto una sobrecarga notable.

Publicación

En cuanto la aplicación estuvo lista para pasar a producción, se publicó en Google Play.

TODO meter captura Google Play.

Además, se desarrolló una página web promocional (gobees.io), donde se describen las características de la aplicación, los manuales de usuario, y el enlace de descarga, entre otras cosas.

TODO meter captura página web.

Por último, se crearon perfiles en las principales redes sociales para promocionar la aplicación.

Reconocimientos

Durante el desarrollo del proyecto se obtuvieron varios reconocimientos:

- **Beca de colaboración con departamentos:** se me concedió esta beca para profundizar en el algoritmo desarrollado y publicar un artículo científico sobre él.
- **Prototipos Orientados al Mercado:** GoBees resultó ganador de uno de los tres premios de la convocatoria de Prototipos Orientados al Mercado realizada por el Vicerrectorado de Investigación y Transferencia del Conocimiento de la Universidad de Burgos.
- **YUZZ:** GoBees fue elegido para participar en el programa YUZZ 2017, patrocinado por el Banco Santander para el impulso del talento joven y el espíritu emprendedor.

Trabajos relacionados

Como se comentó en la introducción, los intentos de automatizar el proceso de monitorización de la actividad de una colmena se remontan hasta principios del siglo pasado. Sin embargo, no es hasta 2008 cuando se introduce la visión artificial en este campo. A continuación, se exponen los artículos científicos relacionados publicados hasta la fecha, así como proyectos con objetivos similares.

Artículos científicos

Video Monitoring of Honey Bee Colonies at the Hive Entrance

Se trata del primer artículo publicado sobre el tema (año 2008). Los autores fueron Jason Campbell, Lily Mummert y Rahul Sukthankar del *Intel Research Pittsburgh*. En él proponen un método de visión artificial para monitorizar las entradas y salidas de abejas en una colmena, consiguiendo diferenciar las que entran de las que salen. Se describen los desafíos técnicos que supuso y la solución a la que llegaron finalmente [\[art:campbell2008\]](#).

Detecting and tracking honeybees in 3D at the beehive entrance using stereo vision

En 2013, Guillaume Chiron, Petra Gomez-Krämer y Ménard Michel publicaron un artículo en *EURASIP Journal on Image and Video Processing*, donde proponían un método para la monitorización de abejas a la entrada de una colmena basado en un sistema de tiempo real con visión estereoscópica. Gracias al cual podían obtener una representación en tres dimensiones de las trayectorias de las abejas [\[art:chiron2013\]](#).

Image Processing for Honey Bee hive Health Monitoring

El último artículo publicado data del año 2015 por Rahman Tashakkori y Ahmad Ghadiri de la *Appalachian State University*. En él, mejoran el método de detección propuesto en [\[art:campbell2008\]](#) y lo utilizan para estimar el número de abejas que habrá en un instante de tiempo dado [\[art:tashakkori2015\]](#).

Comparación

Artículo	Año	Citas	Detección de movimiento	Conteo de abejas	Tracking
<i>Video Monitoring of Honey Bee Colonies at the Hive Entrance</i>	2008	24	<i>Adaptative background subtraction.</i>	<i>Template-based method.</i>	<i>Maximum weighted bipartite graph matching.</i>
<i>Detecting and tracking honeybees in 3D at the beehive entrance using stereo vision</i>	2013	8	<i>Adaptative background subtraction with depth information.</i>	<i>Hybrid 3D intensity depth segmentation .</i>	<i>Kalman filter y Global Nearest Neighbor.</i>
<i>Image Processing for Honey Bee hive Health Monitoring</i>	2015	0	<i>Averaging a background with illumination invariant method.</i>	<i>Area-based method.</i>	No
GoBees	2017	0	<i>Mixture of Gaussians method (BackgroundSubtractorMOG2).</i>	<i>Area-based method.</i>	No

Proyectos

EyesOnHives

EyesOnHives es el principal competidor del proyecto. Se trata de un producto comercial cuyo fin es la monitorización del estado de salud de las colmenas mediante su actividad de vuelo. Integra un *hardware* específico que se encarga de la captación de imágenes y una plataforma en la nube que las procesa y permite el acceso a los datos.

- Web del proyecto: <http://www.keltronixinc.com>

HiveTool

Se trata de un proyecto *OpenSource* que ofrece un conjunto de herramientas para monitorizar distintos parámetros de una colmena. Una de estas herramientas es “Bee Counter”, un contador de abejas por visión artificial desarrollado sobre una Raspberry Pi.

- Web del proyecto: <http://hivetool.org/>

Fortalezas y debilidades del proyecto

Características	GoBees	EyesOnHives	HiveTool
No requiere <i>Hardware</i> específico			
Instalación sencilla			
Procesamiento en local		Parcial	
No requiere wifi			
No requiere red eléctrica			
Localización GPS			
Plataformas	Android	Web App	Linux

Las principales fortalezas del proyecto son:

- No se necesita adquirir ningún *hardware* específico como en el resto de proyectos, simplemente se necesita un *smartphone* con Android. Esto hace el proyecto mucho más accesible a los potenciales usuarios.
- La instalación es muy sencilla. Únicamente se requiere un trípode o cualquier otro tipo de soporte que permita sujetar el *smartphone* en posición cenital.
- El procesamiento de las imágenes se realiza en local no en un servidor. Considerando que los colmenares suelen estar en medio del monte, no podemos requerir una conexión *wifi* como necesita EyesOnHives y el envío de vídeo mediante tecnologías 3G/4G supondría un coste económico muy elevado.
- No requiere estar conectado a la red eléctrica. El *smartphone* cuenta con su propia batería. El consumo de la aplicación no es muy elevado al estar la pantalla apagada durante la monitorización. Aun así, se pueden utilizar *powerbanks* (baterías portátiles) en caso de ser necesarios.
- El *smartphone* tiene integradas varias tecnologías de transmisión de información. Lo da la posibilidad de crear una plataforma que centralice la recogida de datos de varios dispositivos sin importar su localización.
- Relacionado con el punto anterior, el *smartphone* nos permite estar conectados a internet, posibilitándonos ampliar la información que maneja nuestra aplicación. Por ejemplo, podemos acceder a la información meteorológica en tiempo real.
- El GPS del *smartphone* nos permite localizar geográficamente la monitorización y, por tanto, la información meteorológica. Además, puede ser de utilidad en caso de robo, gracias a aplicaciones como *Android Device Manager*, Cerberus, etc. que permiten localizar el dispositivo de forma remota.

Las principales debilidades son:

- Actualmente solo se encuentra disponible para Android. Aunque en una segunda fase del proyecto se creará una plataforma en la nube que centralice todos los datos y una aplicación web que permita acceder a ellos.
- El utilizar un *smartphone* como soporte *hardware* tiene sus ventajas, pero también sus inconvenientes. La cámara no tiene el mismo rendimiento que una cámara diseñada específicamente para esta tarea. Esto nos ha limitado en las técnicas de visión artificial que hemos podido aplicar, por no disponer de imágenes con la suficiente nitidez.

Conclusiones y Líneas de trabajo futuras

En esta sección se exponen las conclusiones derivadas del trabajo, así como las posibles líneas de trabajo futuras por las que se puede dar continuidad al proyecto.

Conclusiones

Tras el desarrollo del proyecto podemos extraer las siguientes conclusiones:

- El objetivo general del proyecto se ha cumplido satisfactoriamente. Ahora los apicultores cuentan con una aplicación Android que no solo les permite gestionar sus colmenares, sino que además les brinda la posibilidad de monitorizar la actividad de vuelo de sus colmenas e interpretar los datos recogidos.
- El haber utilizado el ecosistema Android para la realización del proyecto ha aportado ciertas ventajas tanto en herramientas de desarrollo, como en la distribución de la aplicación. Sin embargo, también ha supuesto un desafío al tener que desarrollar un algoritmo de una cierta complejidad para dispositivos con recursos bastante limitados.
- El proyecto ha abarcado gran parte de los conocimientos adquiridos durante el grado. Además, ha requerido el aprendizaje de muchos otros como la visión artificial, OpenCV, Android, etc.
- Durante el proyecto se han utilizado un gran número de tecnologías y herramientas. La mayoría de ellas han contribuido a mejorar la calidad del producto final o de los procesos intermedios. No obstante, algunas de ellas han supuesto una sobrecarga importante, como sucedió en el caso de la documentación. Aun así, el conocimiento adquirido de todas ellas será de mucha utilizada en proyectos futuros.
- Gracias a la parte de investigación que posee el proyecto, se ha aprendido a realizar búsquedas bibliográficas y a familiarizarse con la lectura de artículos científicos.
- La utilización de varios servicios de integración continua nos ha permitido la detección temprana de defectos en el *software*, reduciendo el impacto de estos y dando lugar a un código de mayor calidad.
- Es muy difícil estimar la duración de tareas de investigación o tareas sobre las que no se posee un conocimiento previo. Sin embargo, el haber aplicado una metodología ágil nos ha permitido ser más flexibles ante los cambios. Y finalmente, se ha completado satisfactoriamente el proyecto en el plazo establecido.

Líneas de trabajo futuras

En primer lugar, comentar que la entrega del Trabajo de Final de Grado solo es un hito en el camino del proyecto, ya que su desarrollo prosigue. A continuación, se resume el *roadmap* del proyecto:

- En la columna “*New issues*” del gestor de tareas se encuentran las nuevas funcionalidades en las que se trabajarán en los próximos meses. Entre ellas se encuentran: dar soporte a operaciones por lotes orientadas a apicultores profesionales con un gran número de colmenas, permitir la exportación de los datos almacenados, añadir informes de revisión de colmenas, varias mejoras de usabilidad y diseño, etc.
- Por otro lado, para la beca de colaboración con departamentos se trabajará en la mejora del algoritmo de monitorización actual. Se probarán nuevas técnicas de detección y *tracking*, y se estudiará su viabilidad para ser ejecutadas en dispositivos móviles.
- Para finales de año se planea tener desarrollada una plataforma en la nube que sincronice los datos de varios dispositivos y permita el acceso a estos mediante una aplicación web. Se planea monetizar el proyecto mediante la suscripción a esta plataforma.
- Se considerará la opción de migrar la aplicación a otras plataformas.
- En el futuro, se podrían explotar los datos almacenados en la plataforma para intentar desarrollar algoritmos que predigan ciertos eventos (enfermedades, enjambrazón, etc.) y que permitan al apicultor anticiparse a estos.

Plan del proyecto software

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.

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

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

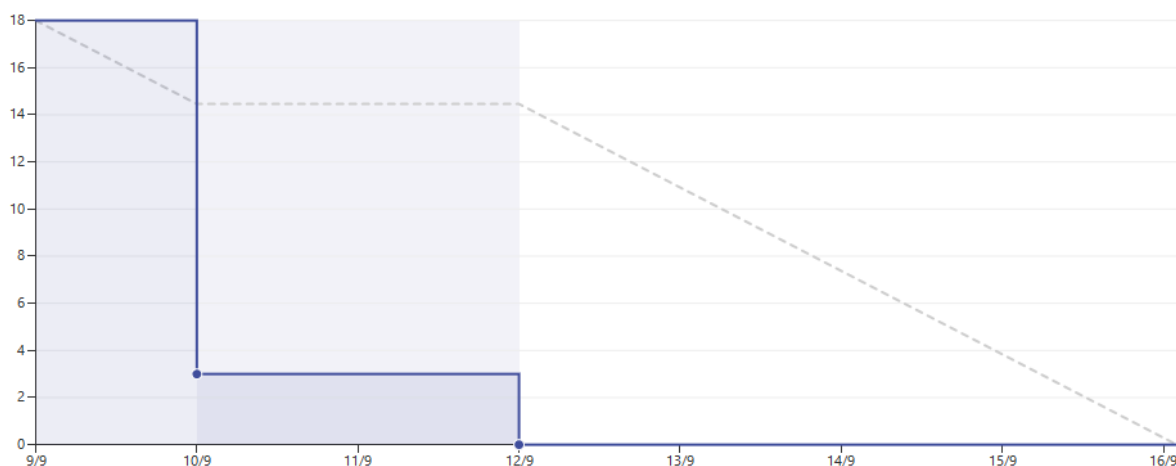
Sprint 0 (09/09/16 - 16/09/16)

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.



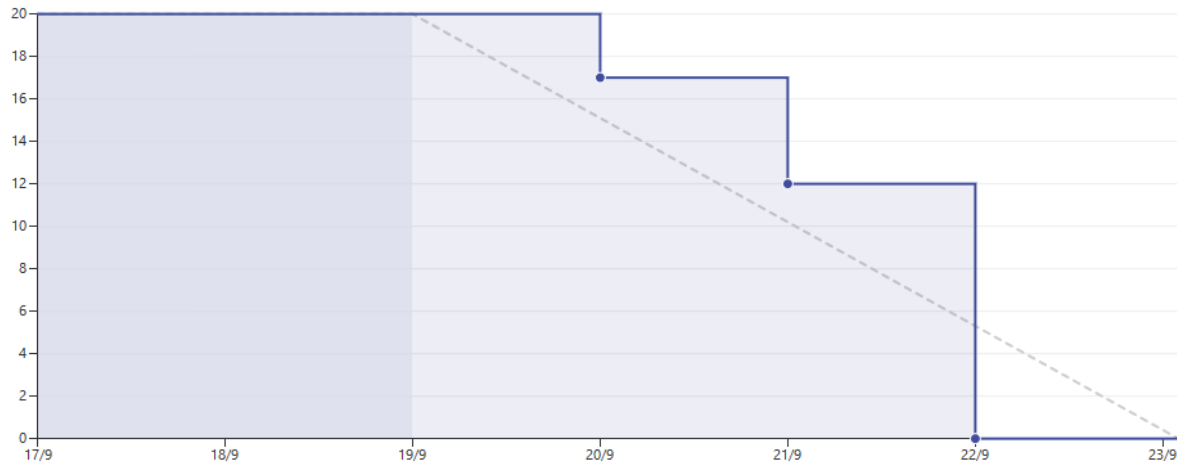
Sprint 1 (17/09/16 - 23/09/16)

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.



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*.

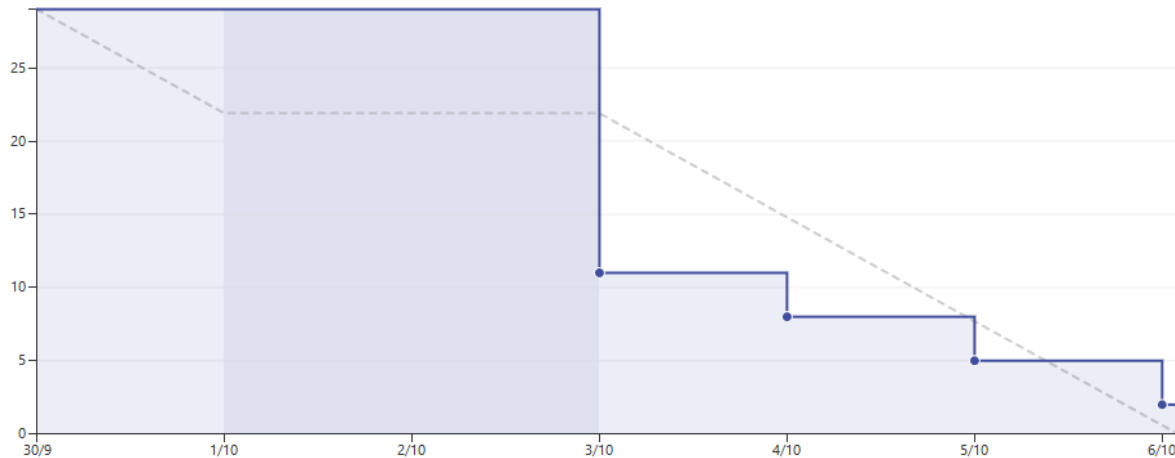


Sprint 3 (30/09/16 - 06/10/16)

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.

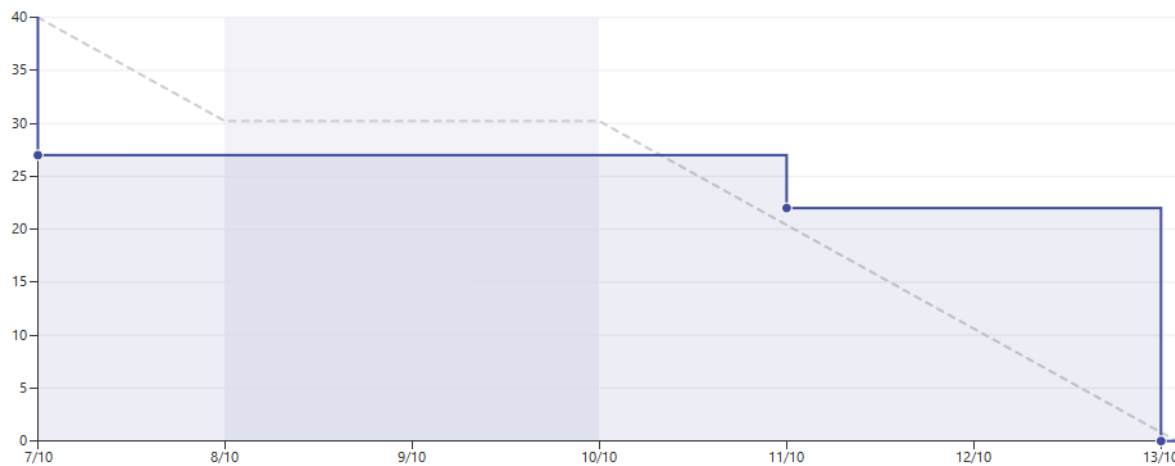


Sprint 4 (07/10/16 - 13/10/16)

Los objetivos de este *sprint* fueron: investigar técnicas de preprocesado y potprocesado 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.

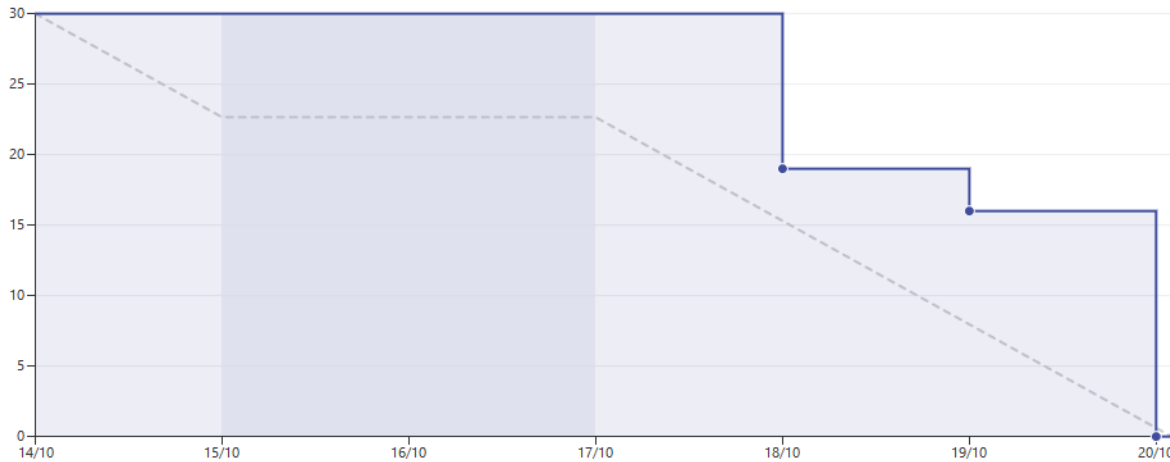


Sprint 5 (14/10/16 - 20/10/16)

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.



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

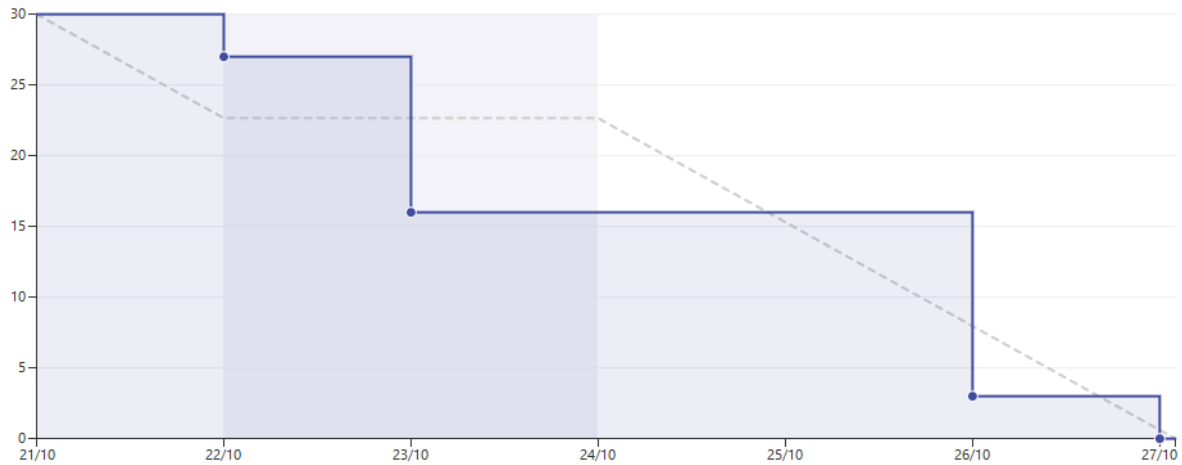
Los objetivos de este *sprint* fueron: mudar el algoritmo de visión artificial desarrollado en la plataforma Java a Android. Comenzar a desarrollar una aplicación de testeo del algoritmo para conocer el error que comete. Investigar si es posible simular el entorno de trabajo filmando a una pantalla.

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.

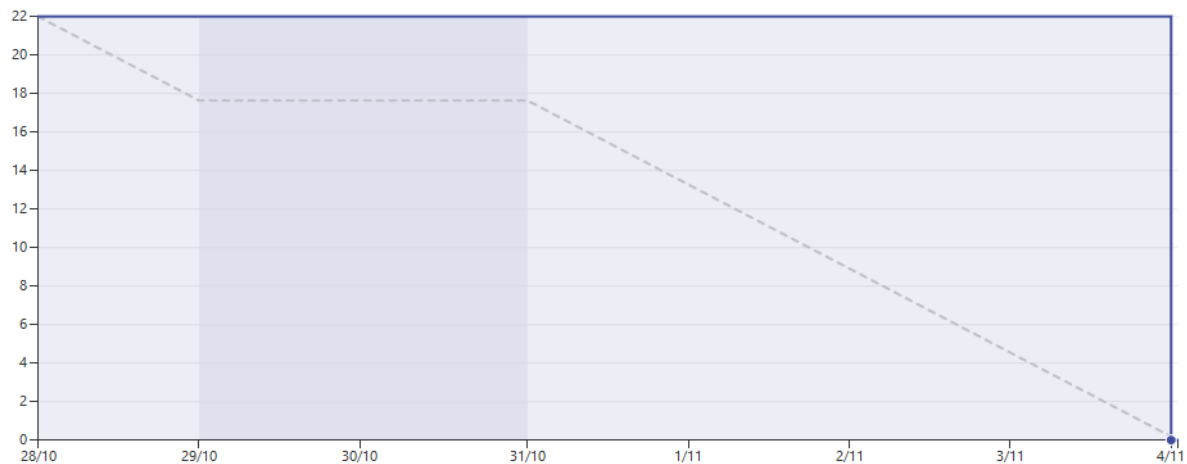


Sprint 7 (28/10/16 - 04/11/16)

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.



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.

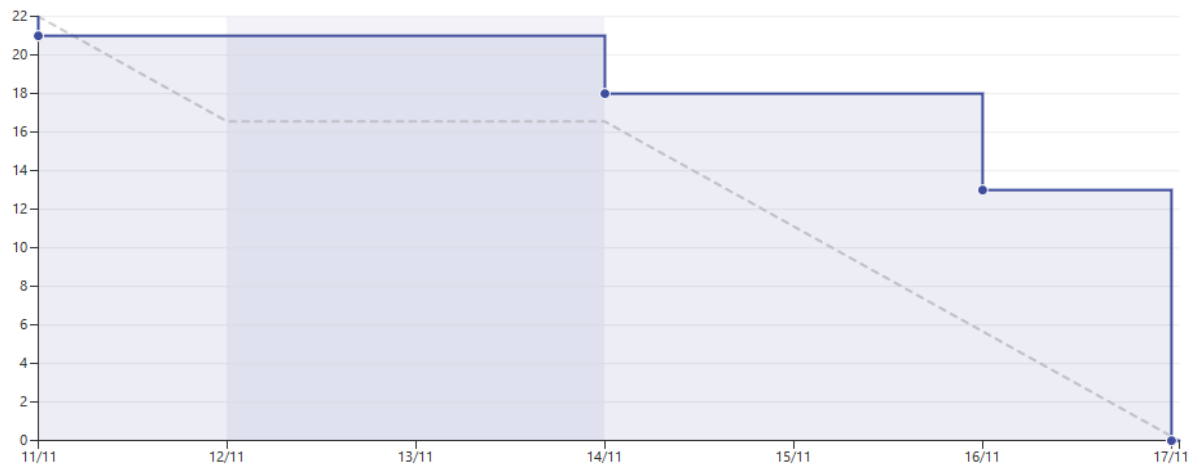


Sprint 9 (11/11/16 - 17/11/16)

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.

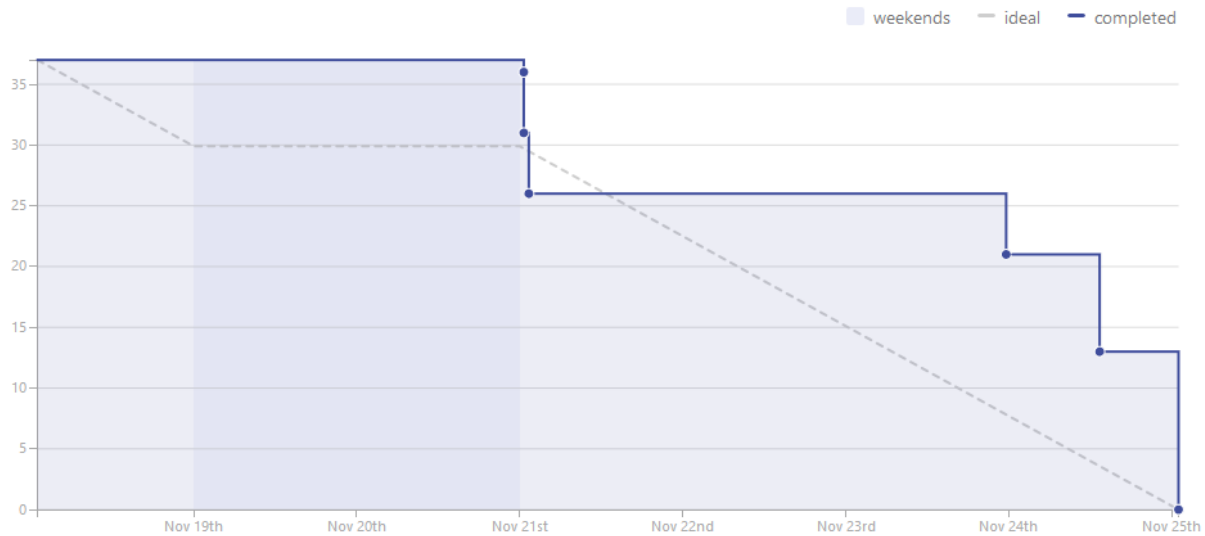


Sprint 10 (11/11/16 - 17/11/16)

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.

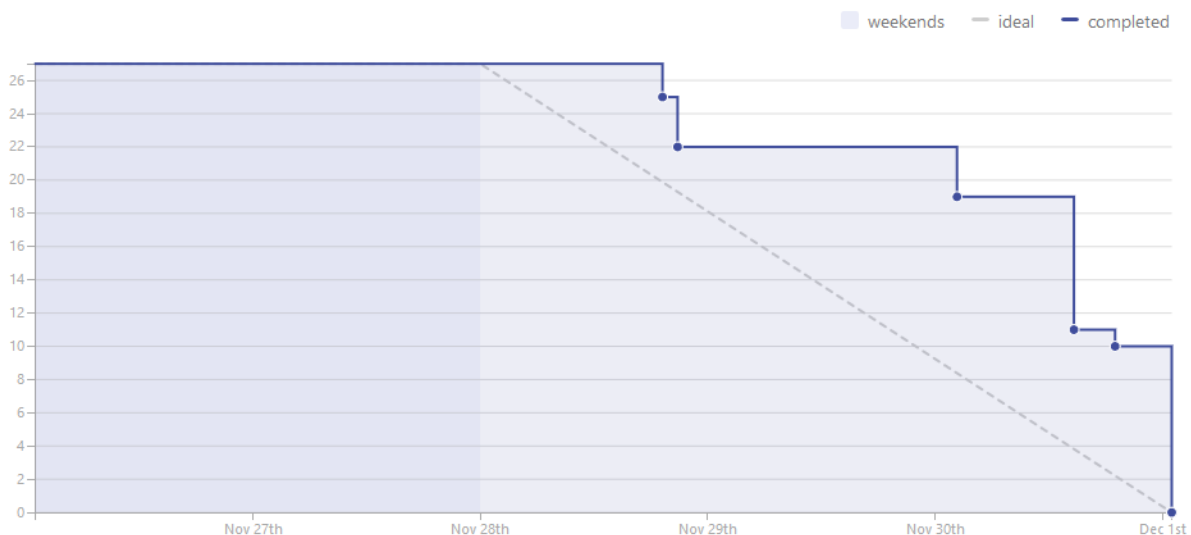


Sprint 11 (26/11/16 - 01/12/16)

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.

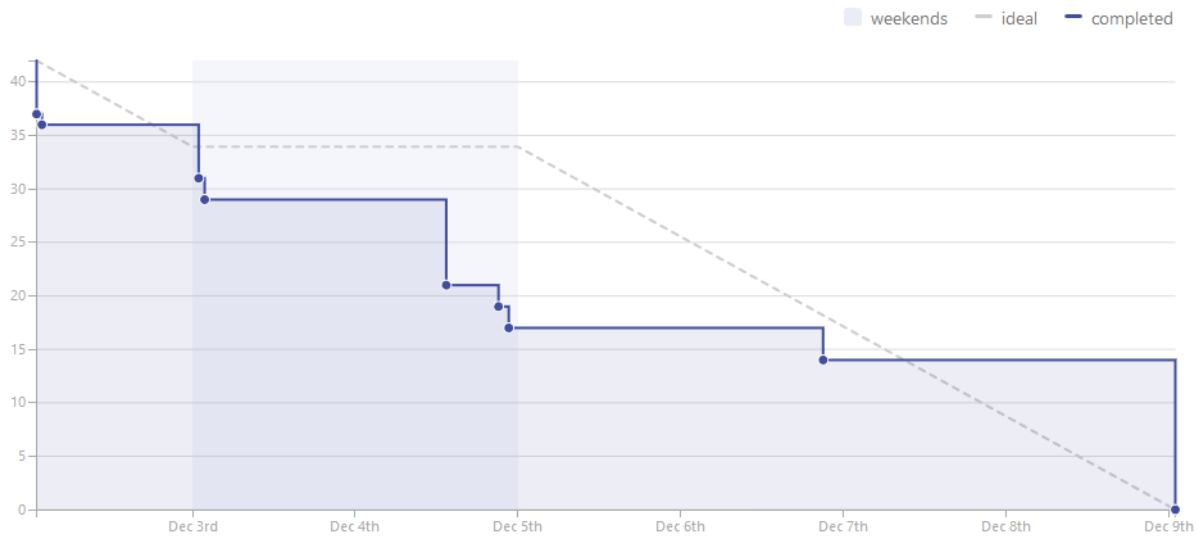


Sprint 12 (02/12/16 - 09/12/16)

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.



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.



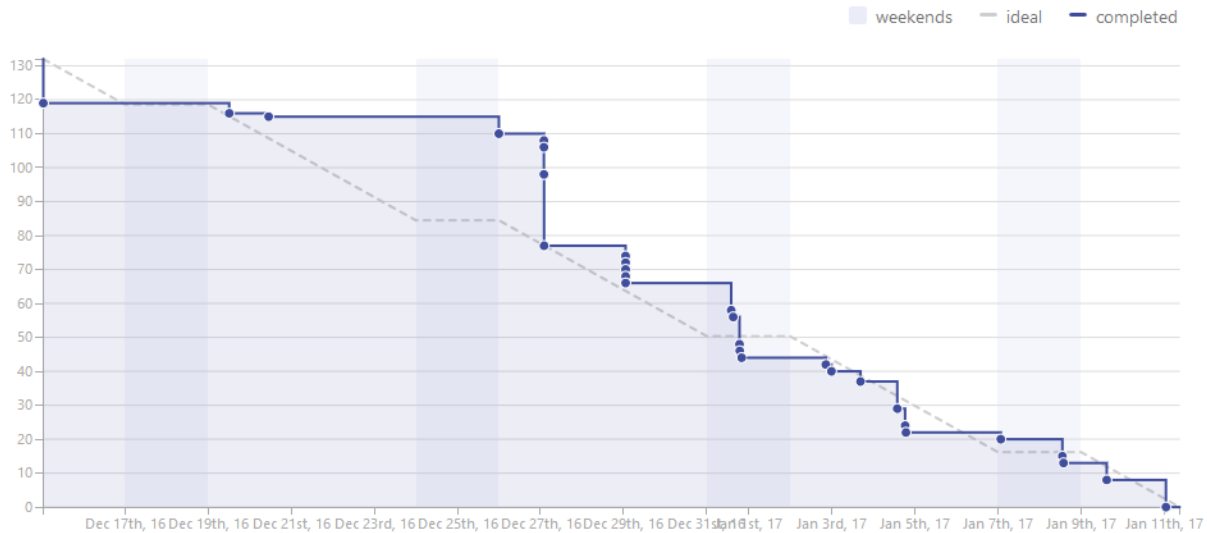
Sprint 14 (15/12/16 - 11/01/17)

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 monitorización 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.

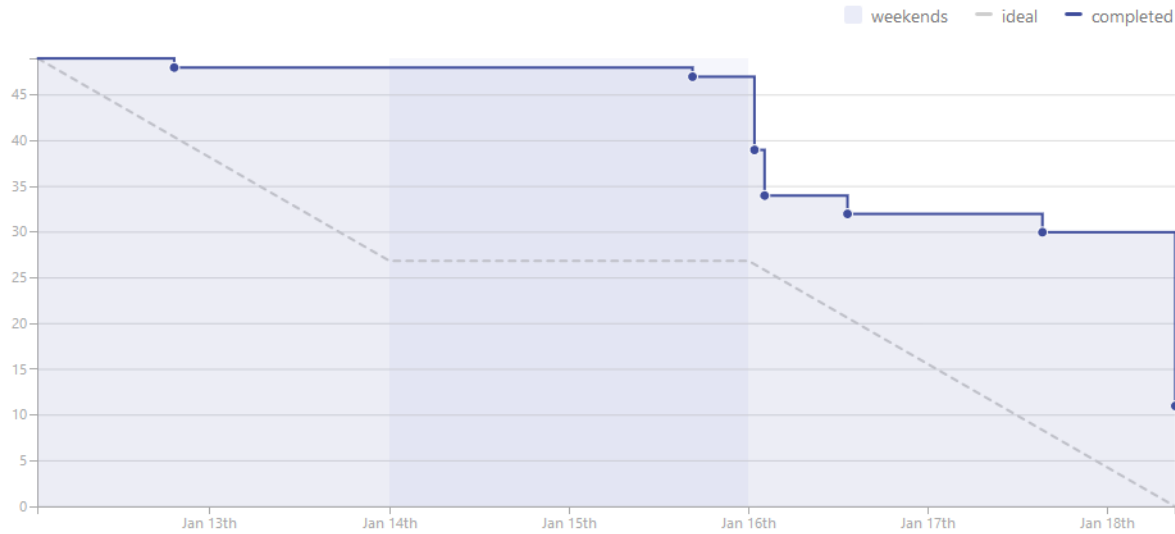


Sprint 15 (12/01/17 - 18/01/17)

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.



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.

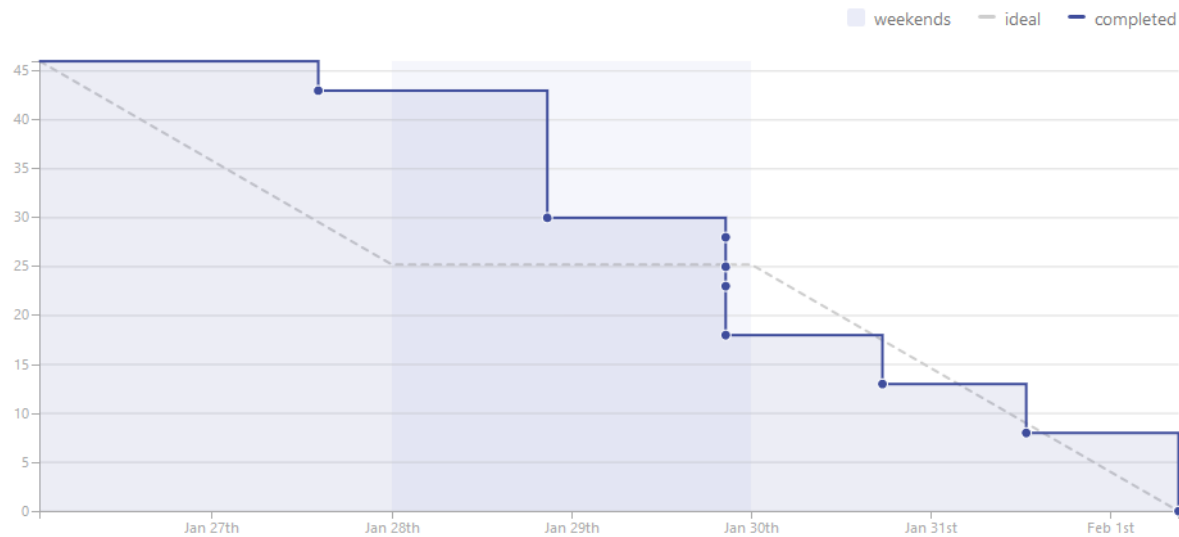


Sprint 17 (26/01/17 - 02/02/17)

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.

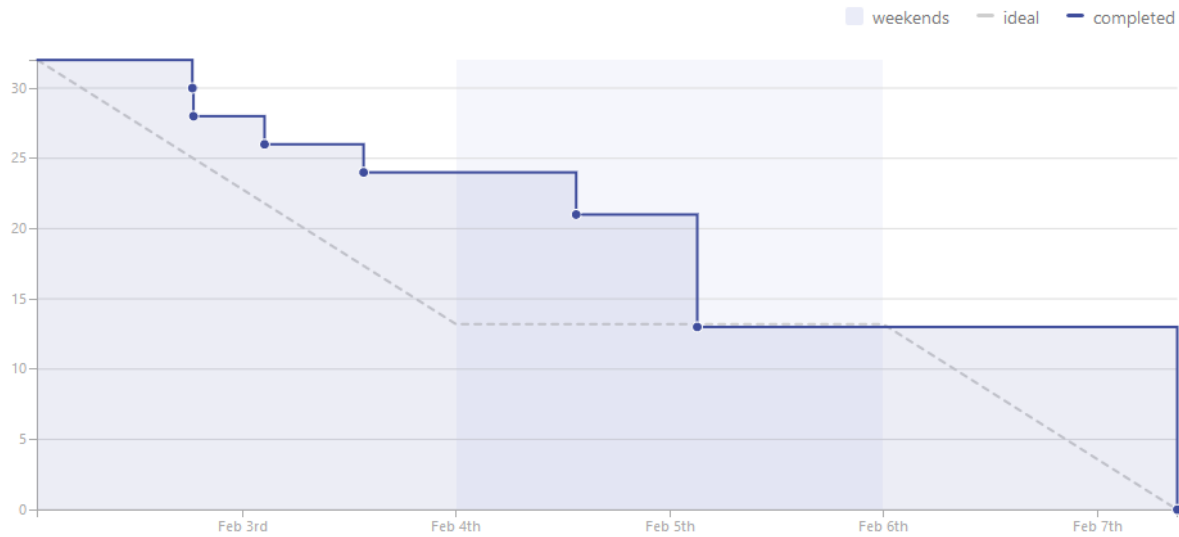


Sprint 18 (02/02/17 - 07/02/17)

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.

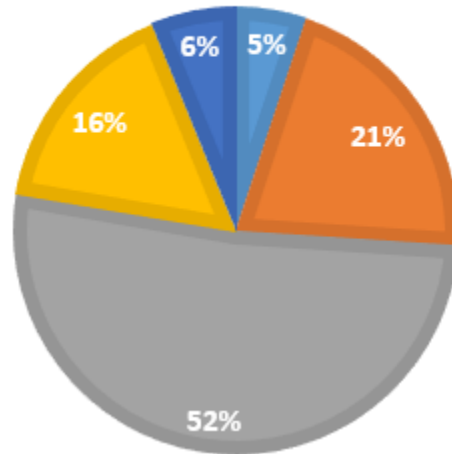


Resumen

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

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

■ Bug ■ Documentation ■ Feature ■ Research ■ Testing



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 €

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 [\[ss_cotizacion\]](#).

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€

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€
Creately	5€	1,04€
Total	284€	59,17€

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€

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€

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

Viabilidad legal

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

“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.” [\[wiki:licencia\]](#)

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	Ver-sión	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
Guava	20.0	Conjunto de bibliotecas comunes para Java.	Apache v2.0
RoundedImageView	2.3.0	Componente para mostrar imágenes redondeadas en Android.	Apache v2.0
MPAndroidChart	3.0.1	Biblioteca de gráficos para Android.	Apache v2.0
VNTNumberPicker Preference	1.0.0	Componente para seleccionar valores numéricos.	Apache v2.0
Permission Utils	1.0.6	Biblioteca que facilita la gestión de permisos en tiempo de ejecución.	MIT
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 JSON	20160810	Biblioteca para trabajar con JSON.	Apache v2.0
Espresso	2.2.2	Framework de <i>testing</i> para Android.	Apache v2.0

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.

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: [\[license:gplv3\]](#)

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.		

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” [\[license:epl\]](#). 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” [\[license:gplv3\]](#).

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: [\[license:cby4\]](#)

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.

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

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

Especificación de Requisitos

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 ha realizado siguiendo las recomendaciones del estándar IEEE 830-1998, que manifiesta que una buena especificación de requisitos *software* debe ser: [\[ieee_830_1998\]](#)

- **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.

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.

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.

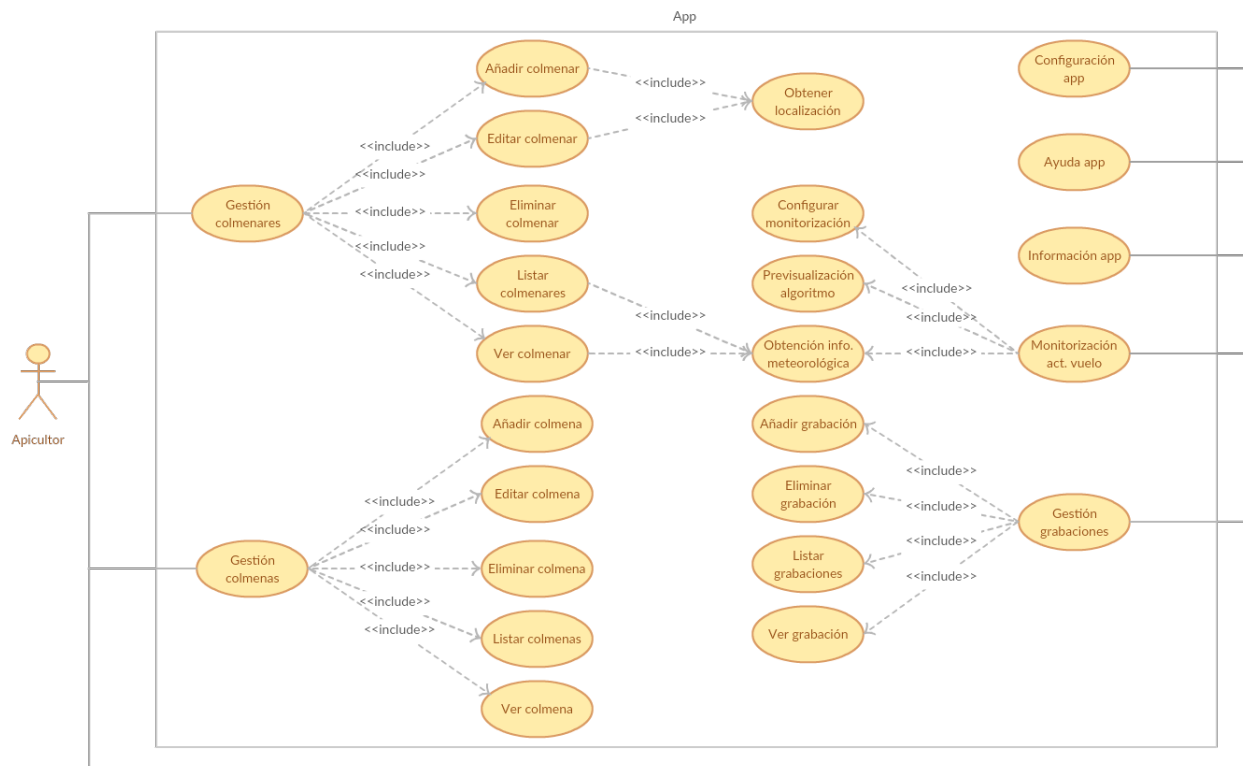
Requisitos no funcionales

- **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.

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



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	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	■ Error al cargar colmenares (mensaje). ■ No existe ningún colmenar (vista especial).
Importancia	Alta

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.
Acciones	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). 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	■ Error al guardar colmenar (mensaje). ■ No se ha introducido nombre del colmenar (resaltar).
Importancia	Alta

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.
Acciones	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. 1. Si el usuario pulsa obtener localización. a) Se obtiene la localización del usuario. 1. El usuario pulsa el botón aceptar. 2. 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	■ Error al guardar colmenar (mensaje). ■ No se ha introducido nombre del colmenar (resaltar).
Importancia	Alta

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.
Acciones	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	■ Error al eliminar colmenar (mensaje).
Importancia	Alta

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.
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

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.
Acciones	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	■ Error al cargar colmenar (mensaje). ■ No existe conexión a internet (mensaje). ■ Error al recuperar la información meteorológica (mensaje).
Importancia	Alta

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.
Acciones	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	■ No se poseen permisos de localización (solicitar). ■ Error de conexión con el GPS (mensaje).
Importancia	Alta

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	Se poseen permisos de acceso a internet. El colmenar existe y posee localización.
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 <i>OpenWeather-Map</i>. 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

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.
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

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.
Acciones	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	■ Error al guardar colmena (mensaje). ■ No se ha introducido nombre de la colmena (resaltar).
Importancia	Alta

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	La base de datos se encuentra disponible. El colmenar existe.
Acciones	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	■ Error al guardar colmena (mensaje). ■ No se ha introducido nombre de la colmena (resaltar).
Importancia	Alta

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	La base de datos se encuentra disponible. El colmenar existe. La colmena a eliminar existe.
Acciones	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	■ Error al eliminar colmena (mensaje).
Importancia	Alta

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	La base de datos se encuentra disponible. El colmenar existe.
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

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	La base de datos se encuentra disponible. El colmenar existe. La colmena a visualizar existe.
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

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.
Postcondición	El número de grabaciones listado es igual al número de grabaciones de esa colmena en la base de datos.
Excepciones	■ Error al cargar grabaciones (mensaje). ■ No existe ninguna grabación (vista especial).
Importancia	Alta

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	La base de datos se encuentra disponible. El colmenar y la colmena existen.
Acciones	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.
Excepciones	■ Error al guardar grabación (mensaje). ■ Grabación demasiado corta (mensaje).
Importancia	Alta

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	La base de datos se encuentra disponible. El colmenar y la colmena existen. La grabación a eliminar existe.
Acciones	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	■ Error al eliminar grabación (mensaje).
Importancia	Alta

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	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	■ Error al cargar grabaciones (mensaje). ■ No existen grabaciones (vista especial).
Importancia	Alta

CU-19	Ver grabación
Versión	1.0
Autor	David Miguel Lozano
Requisitos asociados	RF-3.4
Descripción	Permite al usuario visualizar toda la información (actividad de vuelo, temperatura, precipitaciones y viento) relativa a una determinada grabación existente.
Precondición	La base de datos se encuentra disponible. El colmenar y la colmena existen. La grabación a visualizar existe.
Acciones	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	■ Error al cargar grabación (mensaje).
Importancia	Alta

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
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.
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.
Postcondición	La grabación tiene más de 5 registros.
Excepciones	■ 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).

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	Se poseen permisos de cámara. La cámara se encuentra disponible. El colmenar y la colmena existen.
Acciones	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.
Postcondición	■
Excepciones	■ No se tienen permisos de cámara (solicitar). ■ Error de cámara (cancelar).
Importancia	Alta

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	Se poseen permisos de cámara. La cámara se encuentra disponible. El colmenar y la colmena existen.
Acciones	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	■
Excepciones	■ No se tienen permisos de cámara (solicitar). ■ Error de cámara (cancelar).
Importancia	Alta

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	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	■ Error al guardar configuración (mensaje).
Importancia	Alta

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	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	■ No se disponen de permisos de internet (solicitar), ■ No hay conexión a internet (mensaje).
Importancia	Alta

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	■
Acciones	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 rellena 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

Especificación de diseño

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.

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 hacía 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 icono correspondiente, temperatura, presión, humedad, velocidad y dirección del viento, porcentaje de nubes, precipitaciones, y nieve.

Diagrama E/R

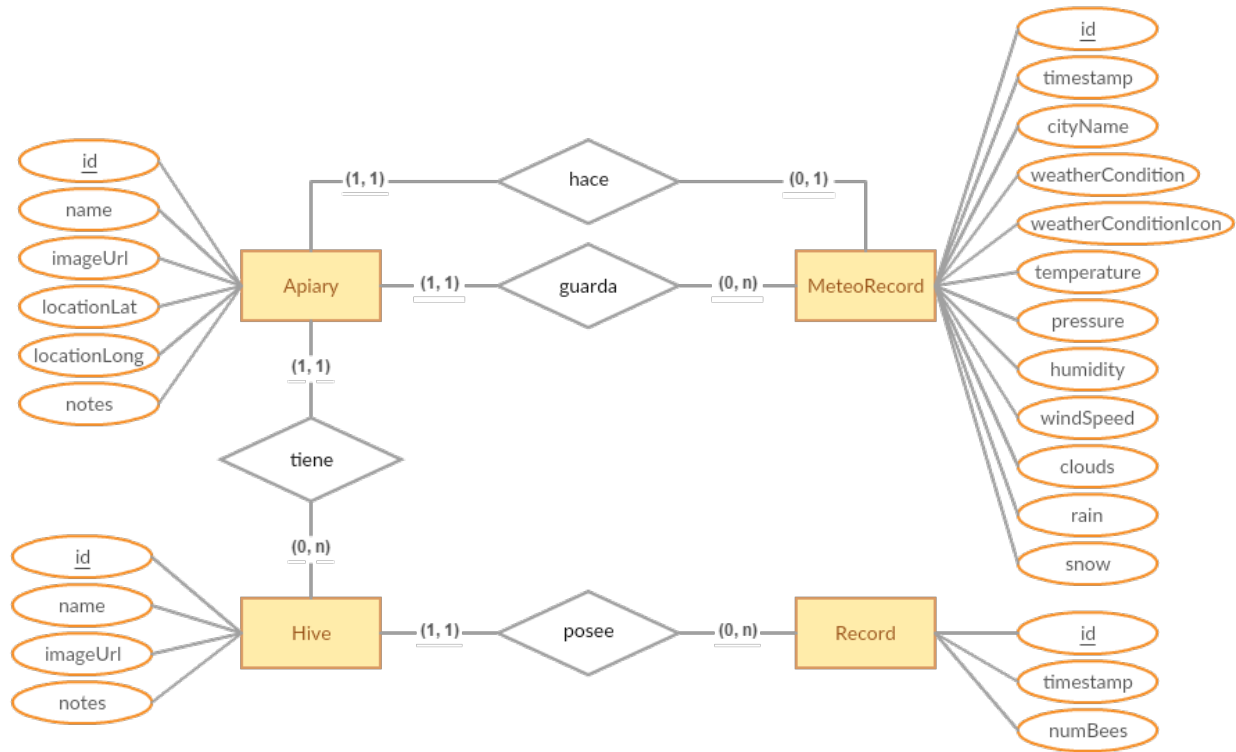
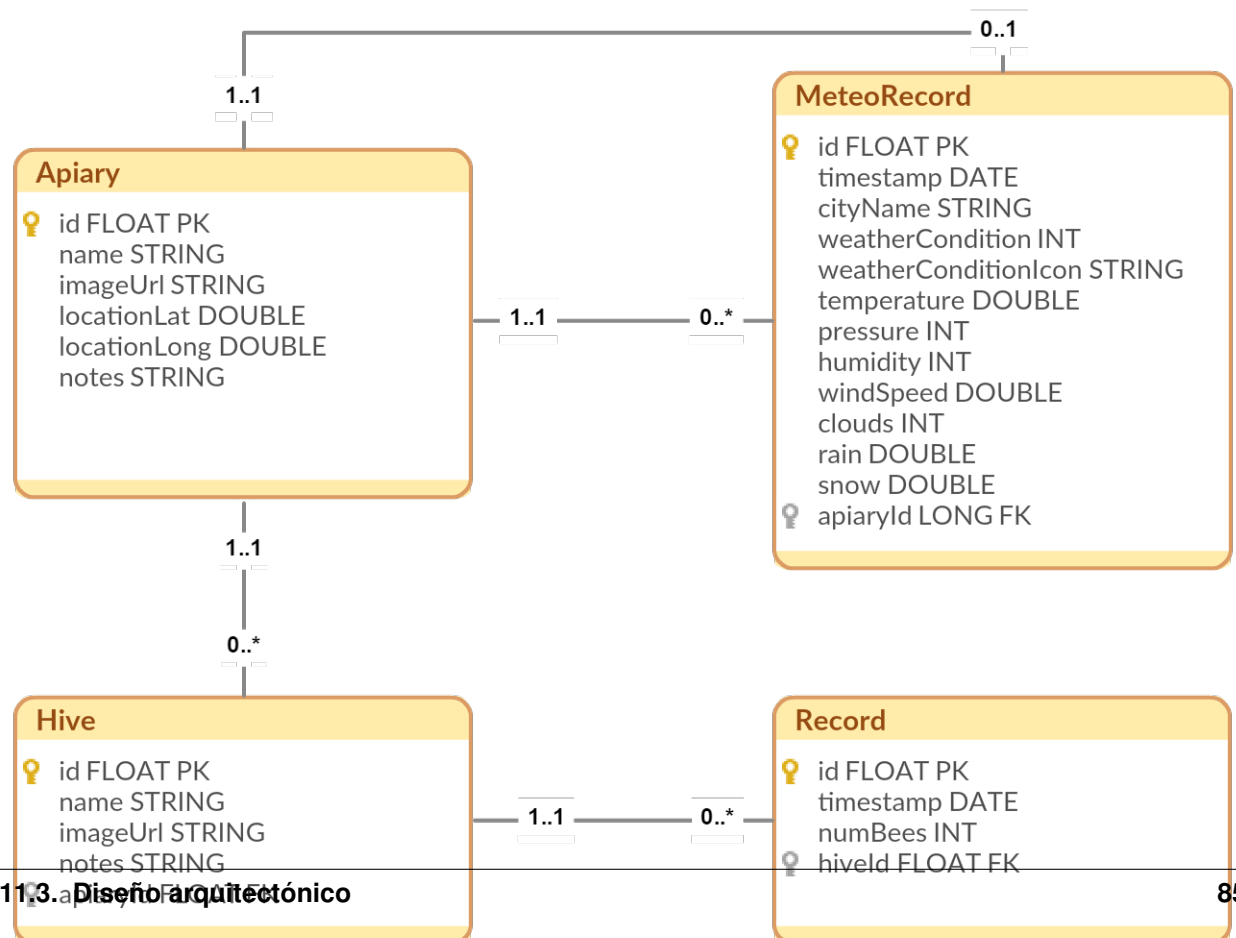


Diagrama Relacional



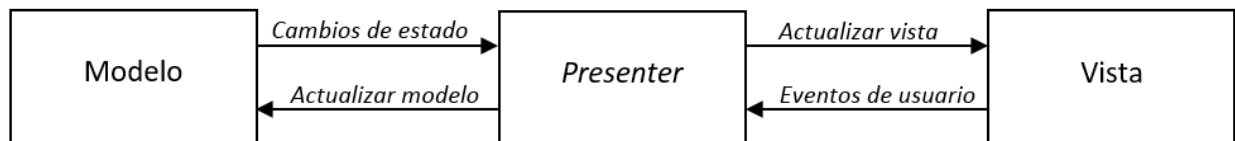
mejorar su testabilidad y mantenibilidad.

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:[[pattern:mvp](#)]

- *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 *Fragments*.
- *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*.



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

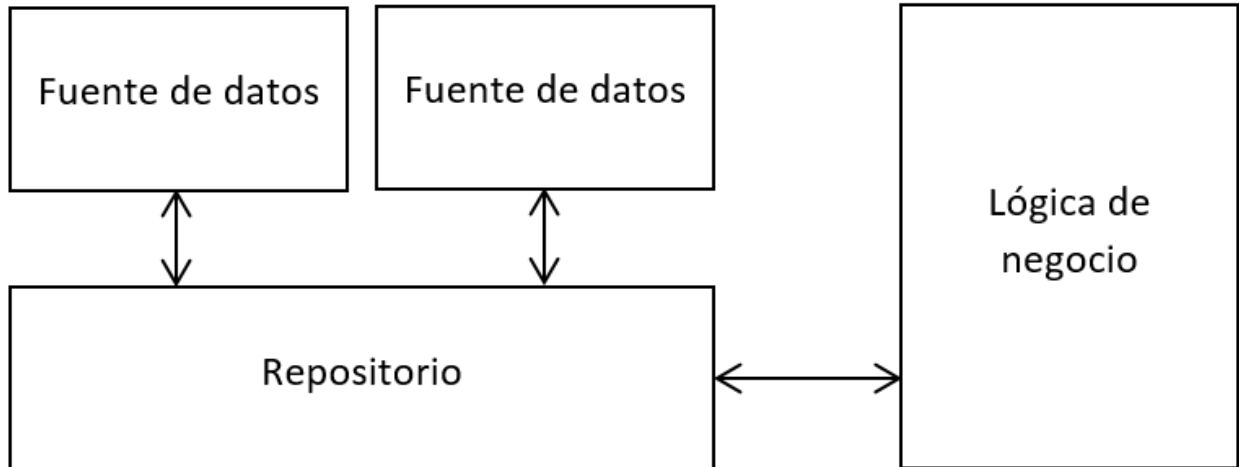
- Se utilizan las *Activity* como controladores globales que se encargan de crear y conectar las vistas con los *presenters*.
- Se utilizan los *Fragment* como vistas ya que proporcionan numerosas ventajas cuando se trabaja con múltiples vistas.

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 [[pattern:repository](#)].

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.



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.

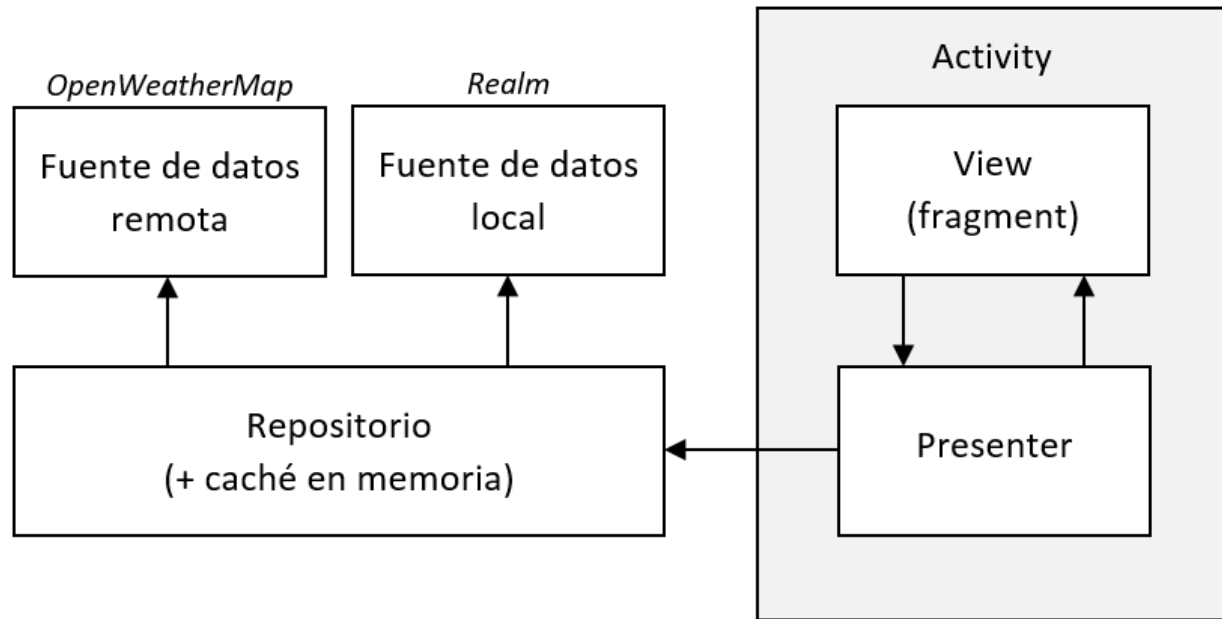
La inyección de dependencias es un patrón mediante el cual se proporcionan todas las dependencias que una clase necesita para su funcionamiento, en lugar de ser la propia clase quien las cree. Al separar las dependencias de la propia clase, se posibilita la opción de sustituir estas por dobles con un comportamiento definido [\[wiki:injection\]](https://es.wikipedia.org/wiki/Inyecci%C3%B3n_de_dependencias).

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:



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

Diseño de paquetes

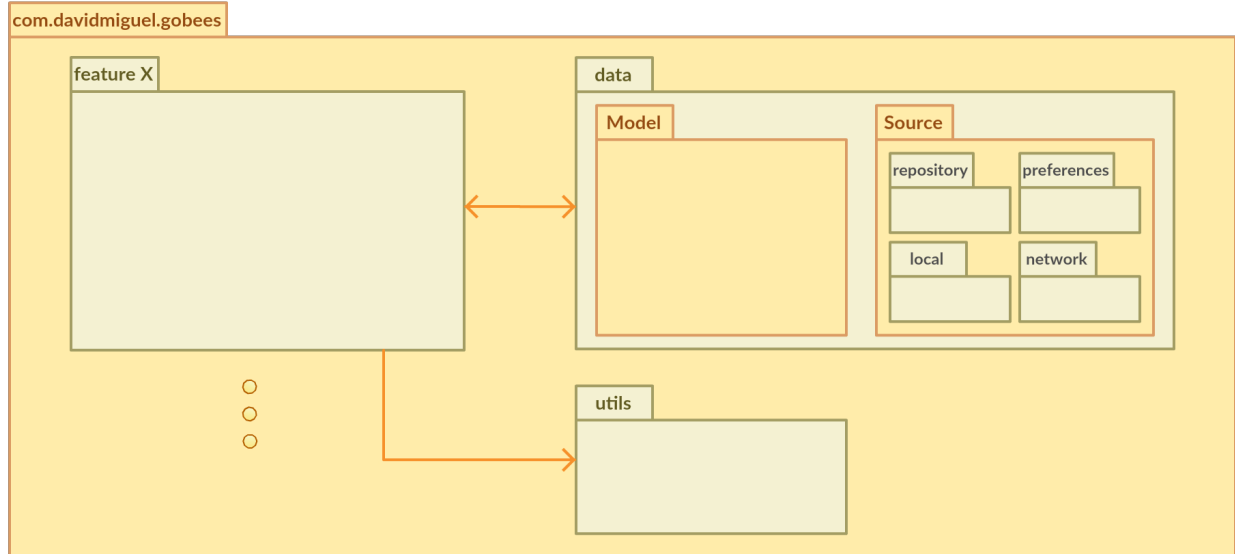
Para la organización de los diferentes archivos que componen la aplicación no se utilizó la estrategia convencional de paquete por capa (*package by layer approach*), sino una estrategia de paquete por característica (*package per feature approach*).

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:

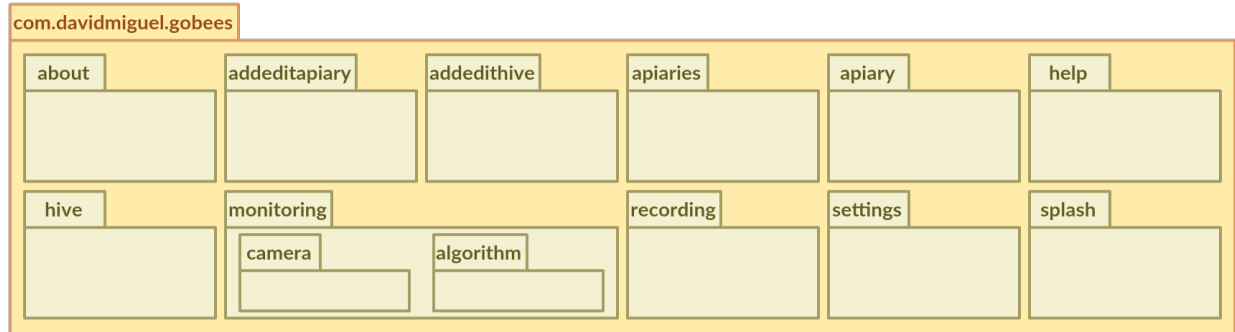
- Paquete *data*: agrupa toda la capa de modelo.
- Paquete *utils*: reúne un conjunto de clases de utilidad generales que son utilizadas por varias características.

El diagrama de paquetes es el siguiente:



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:

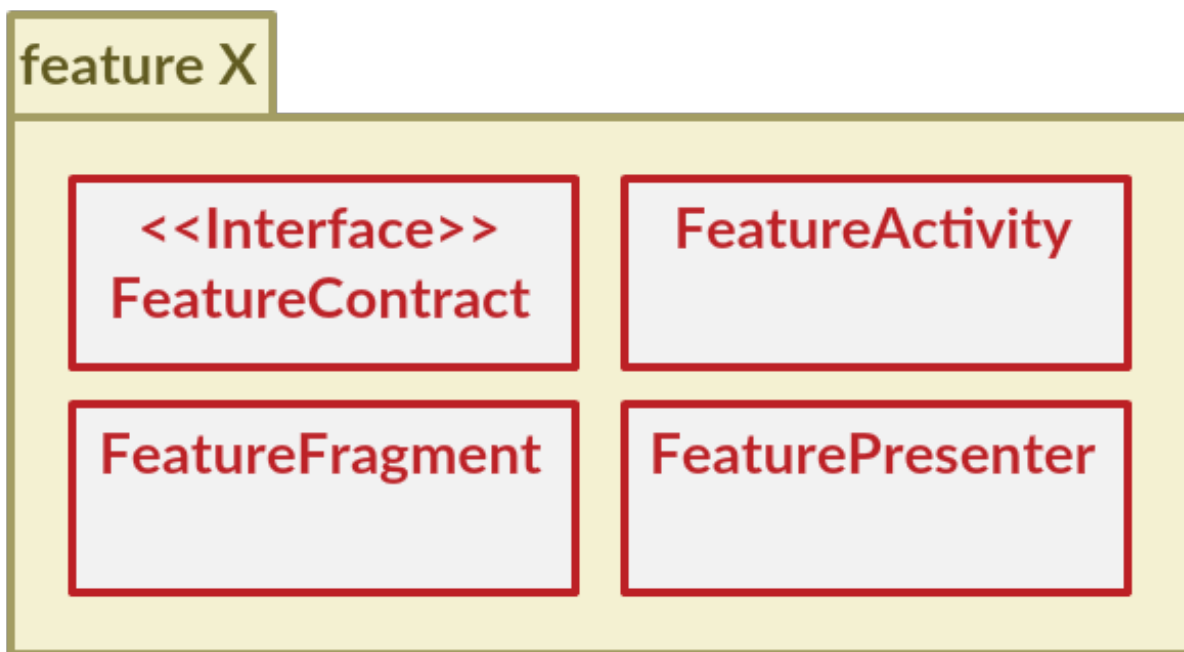


- **About:** contiene la funcionalidad de “Acerca de” de la aplicación. Donde se muestra el autor, licencia, versión de la *app*, sitio web, historial de cambio y todas las dependencias junto con sus licencias.
- **Addeditapiary:** permite añadir o editar colmenares.
- **Addedithive:** permite añadir o editar colmenas.
- **Apiaries:** permite listar los colmenares y gestionarlos.
- **Apiary:** permite listar las colmenas de un colmenar, gestionarlasy mostrar la información relativa al colmenar.
- **Help:** muestra la ayuda de la aplicación.
- **Hive:** permite listar las grabaciones de una colmena, gestionarlasy 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.

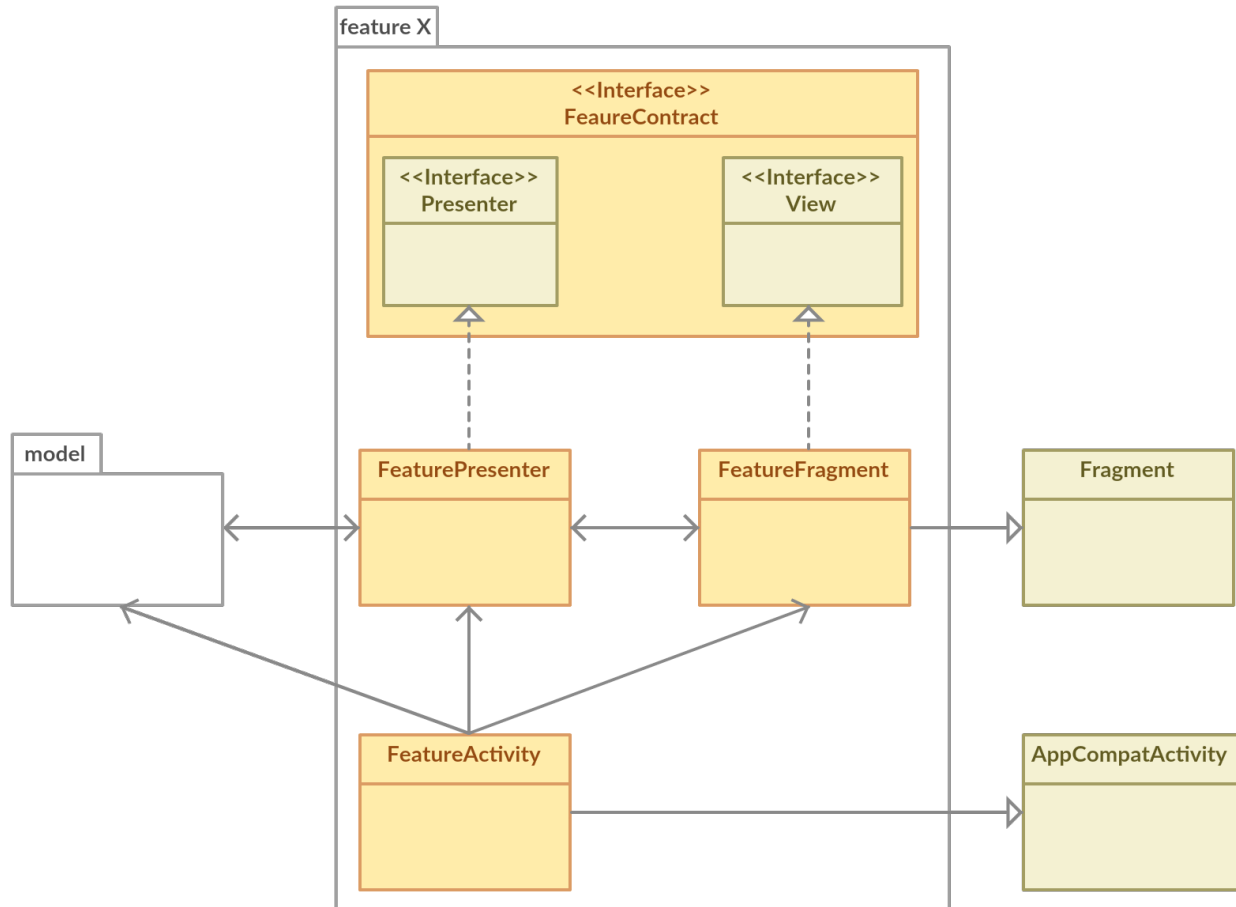
Diseño de clases

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

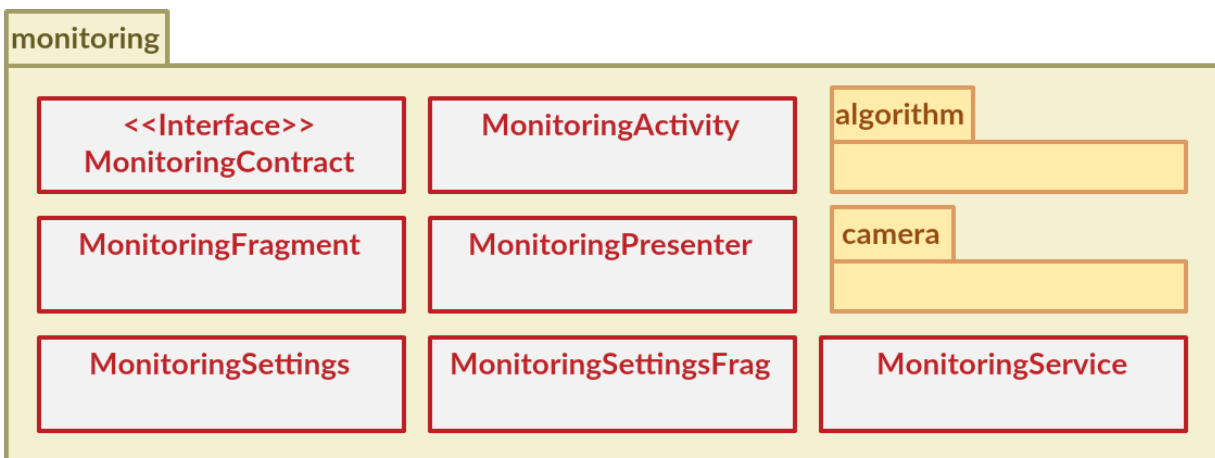
- FeatureActivity: funciona como un controlador global que crea la vista y el *presenter* y los enlaza.
- FeatureContract: se trata de una interfaz que establece los siguientes contratos:
 - FeatureContract.View: define la capa *view* para esta característica (las únicas funciones que expone a otras 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.



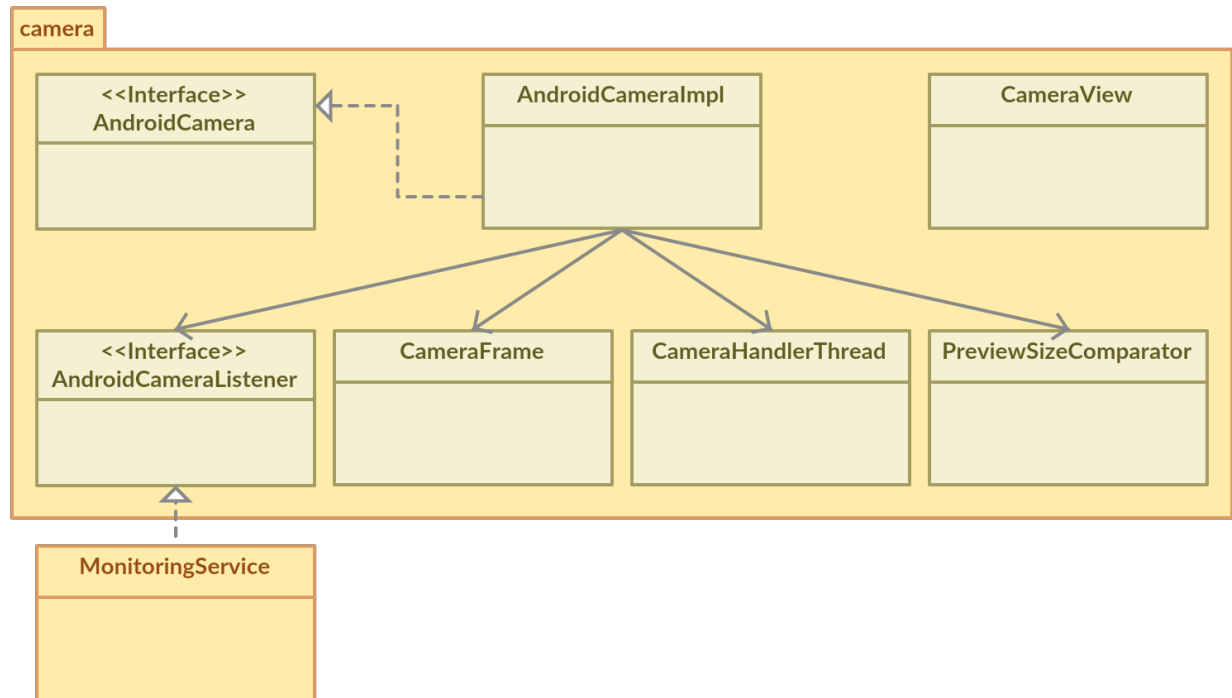
El diagrama de clases general que muestra cómo se relacionan todos los componentes de una determinada característica es el siguiente:



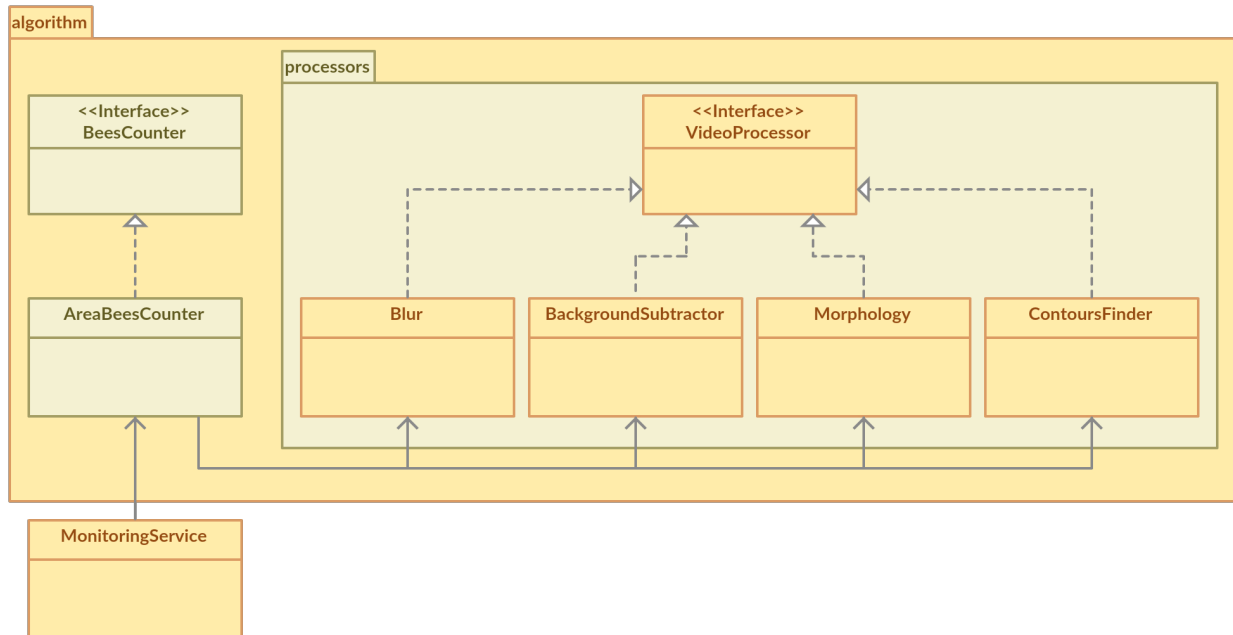
El único paquete que se diferencia de la estructura expuesta es el paquete *monitoring*. Este integra a su vez toda la lógica de acceso a la cámara y todas las clases relacionadas con el algoritmo de conteo.



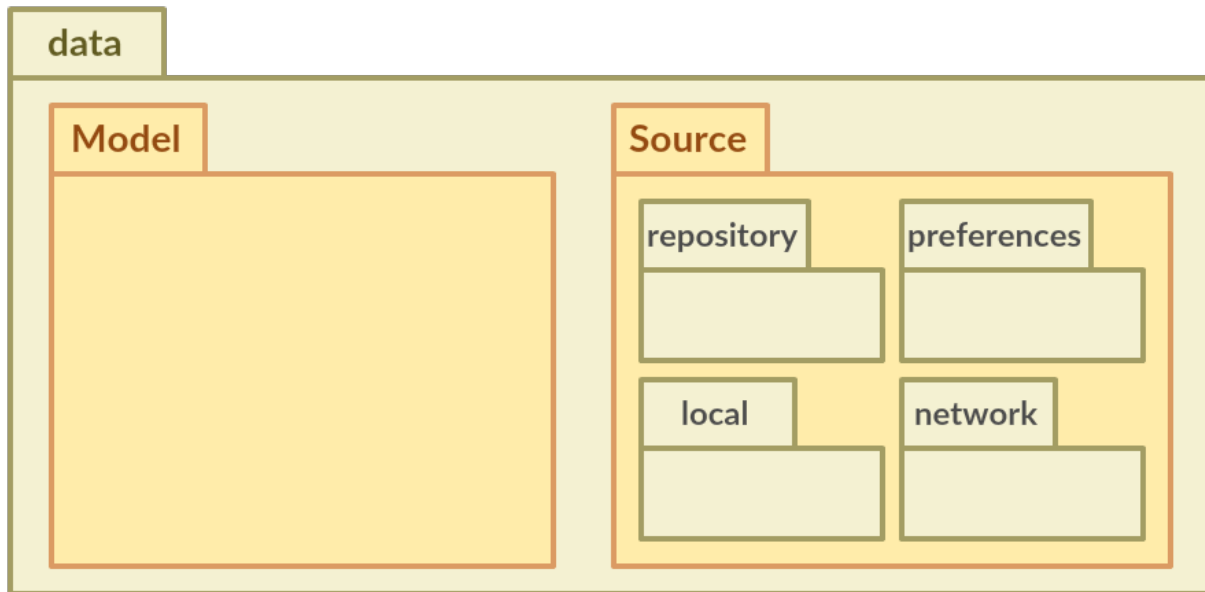
El diagrama de clases del paquete cámara es el siguiente:



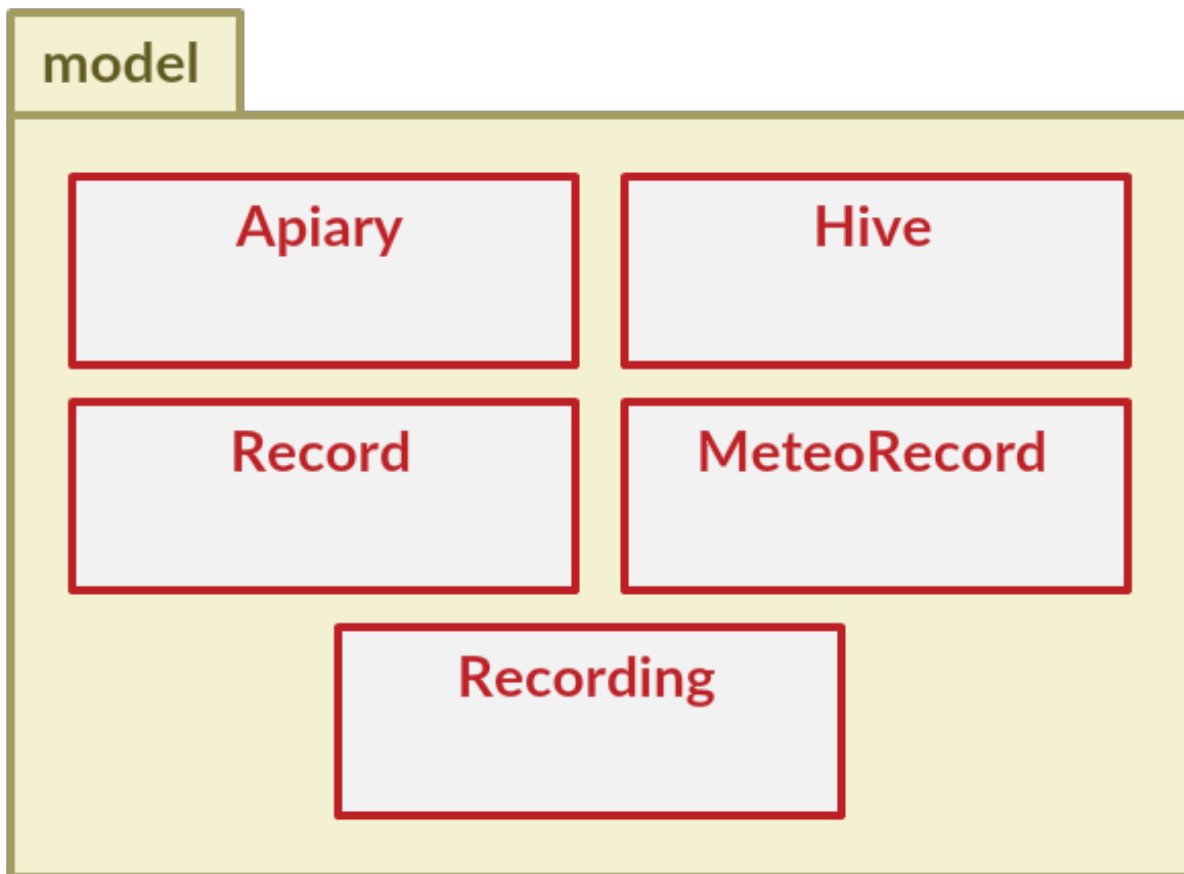
El diagrama de las clases que implementan el algoritmo de conteo es el siguiente:



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

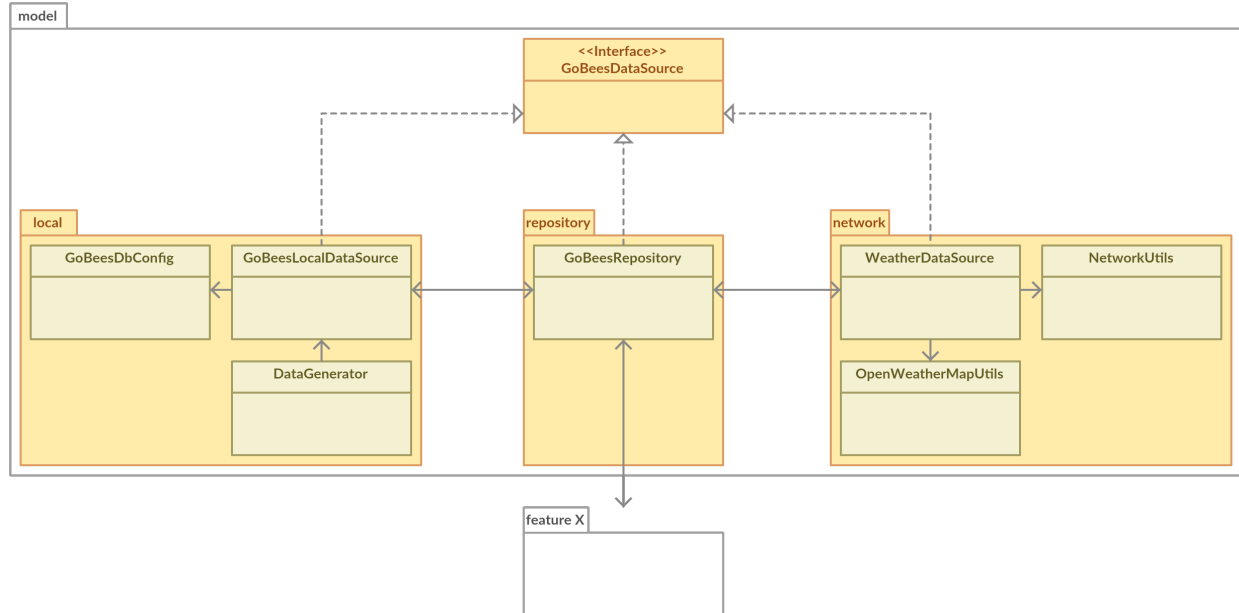


El paquete *model* contiene todas las clases de modelo que se mapean con la base de datos.



*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:

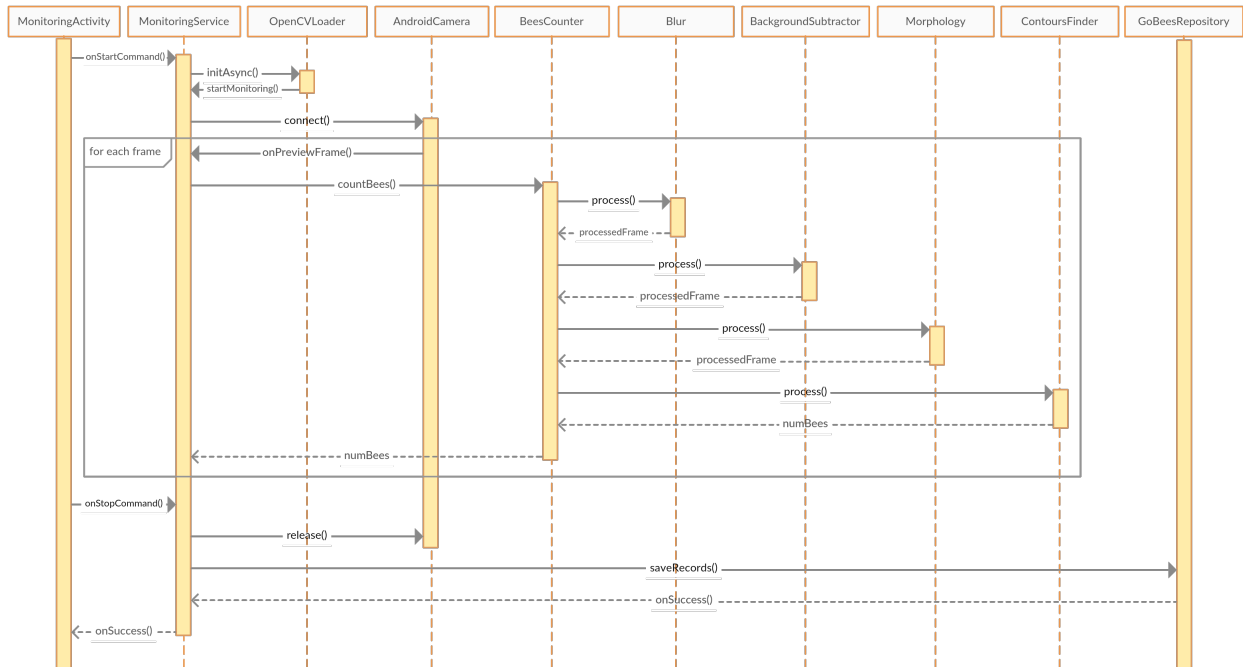


Para conocer a mayor detalle las funciones de cada clase se puede consultar la documentación JavaDoc de la aplicación.

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.

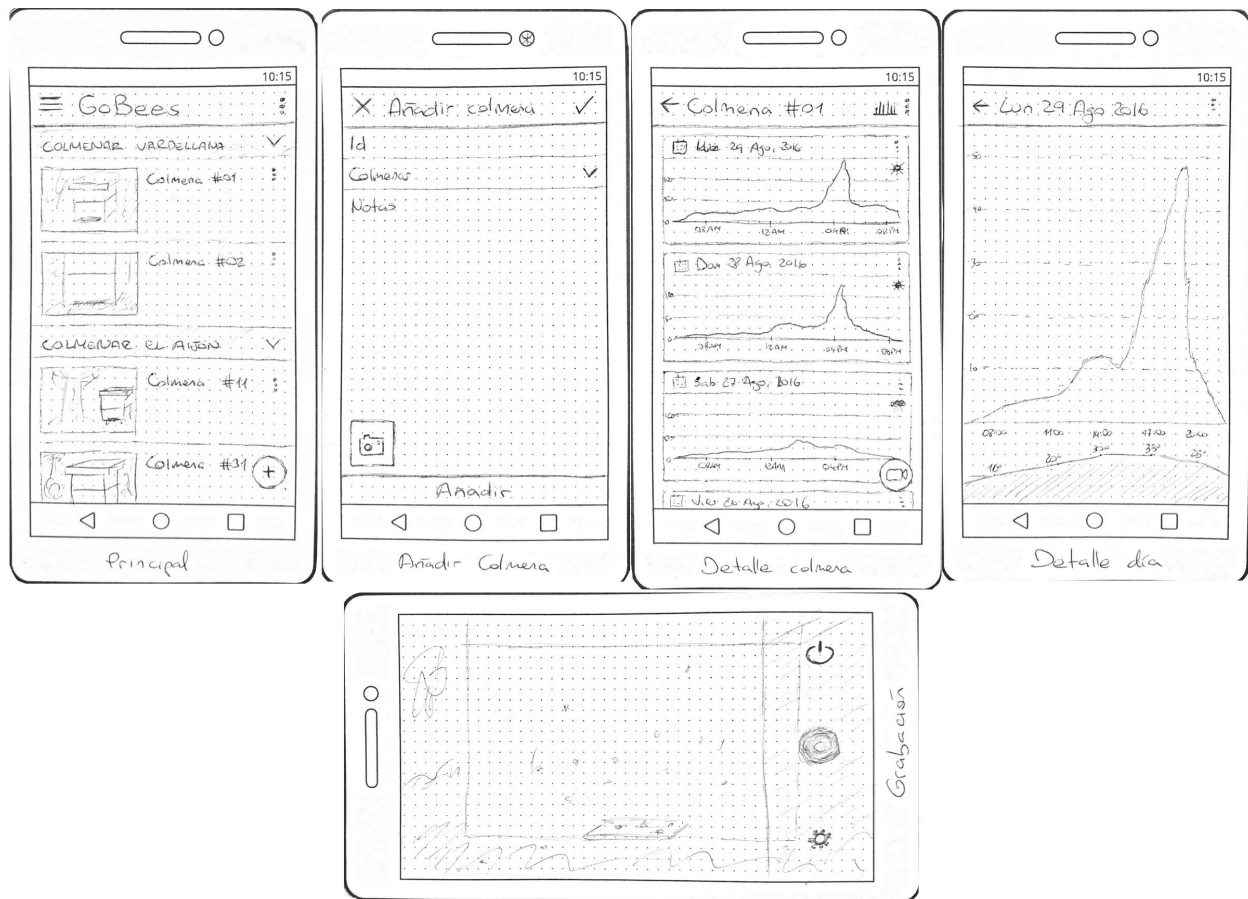
En el siguiente diagrama de secuencia se ha representado como es la interacción entre los diferentes objetos que se encargan de la inicialización de la monitorización, la obtención de las imágenes y su posterior procesado por el algoritmo de conteo.



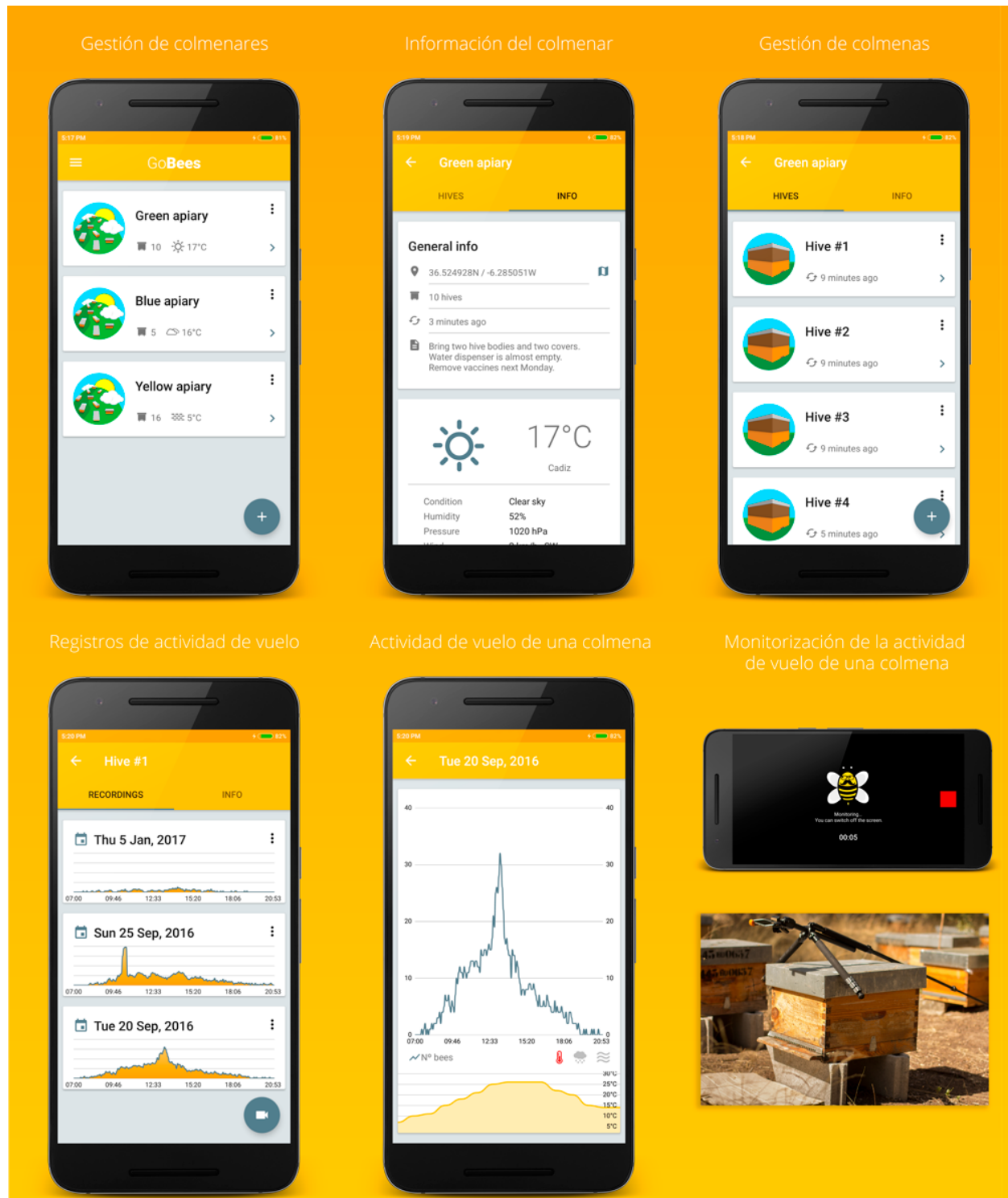
Diseño de interfaces

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

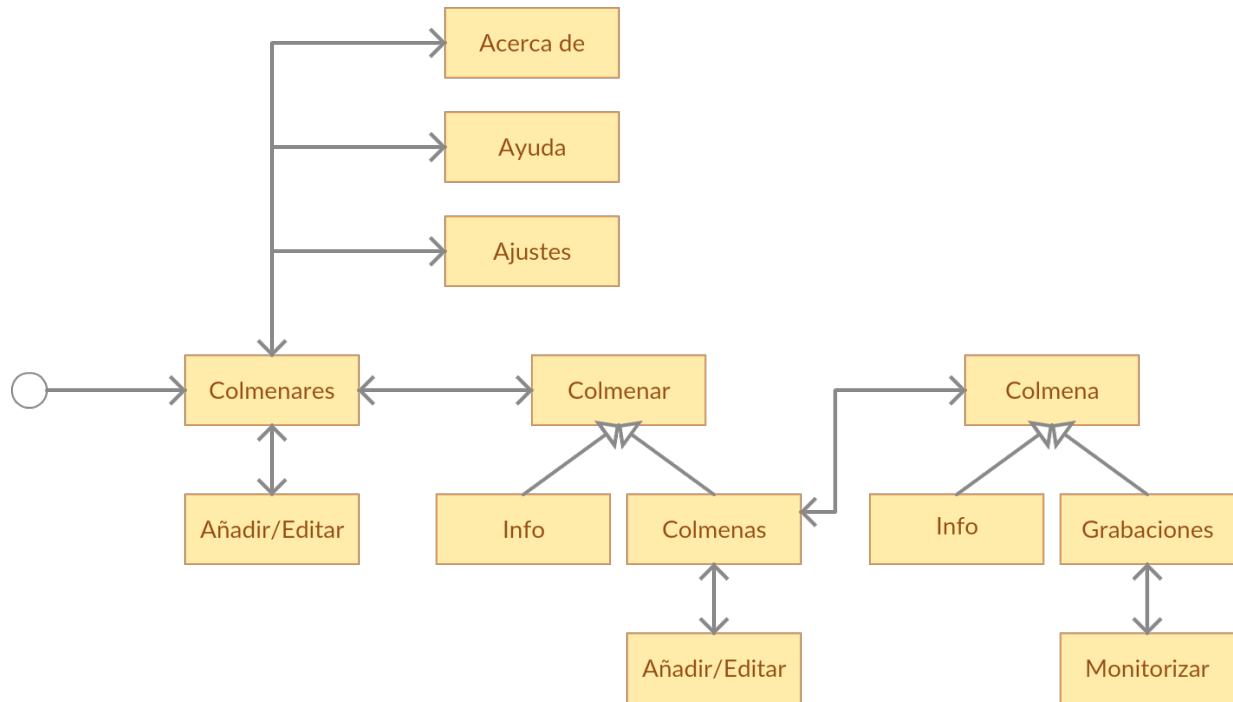
En las primeras etapas de proyecto se realizaron una serie de prototipos en los que se plasmaron las principales funcionalidades de la aplicación.



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



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.



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.

DARK PRIMARY COLOR	PRIMARY COLOR	LIGHT PRIMARY COLOR	TEXT / ICONS
ACCENT COLOR	#212121	#757575	#BDBDBD
	PRIMARY TEXT	SECONDARY TEXT	DIVIDER COLOR

Manual del programador

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.

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 inyectar componentes alternativos durante los test.
- `/app/src/prod/`: *prod flavour*, utilizado para inyectar 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 inyectar 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 LaTeX.
- `/docs/rst/`: documentación en formato reStructuredText.

Para saber más sobre la organización de un proyecto Android consultar [\[android:folders\]](#).

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.

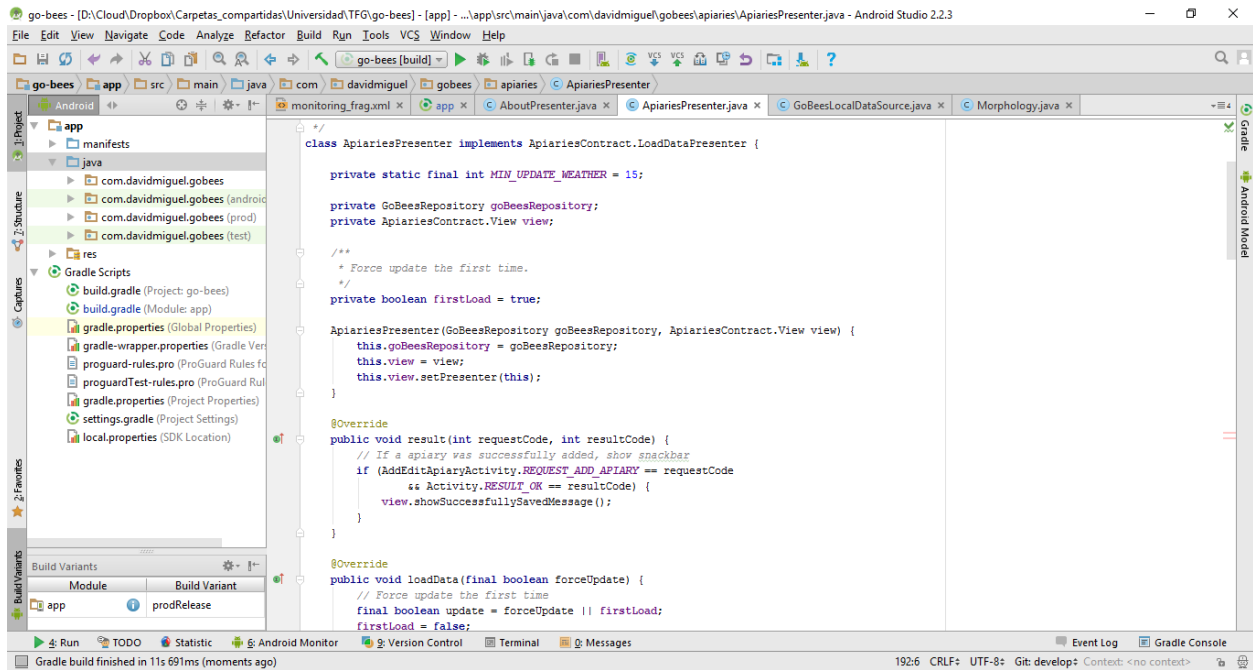
Java JDK 7

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 [\[java:jdk7\]](#). Se debe elegir correctamente el sistema operativo y la arquitectura del ordenador y posteriormente seguir el asistente de instalación.

Android Studio

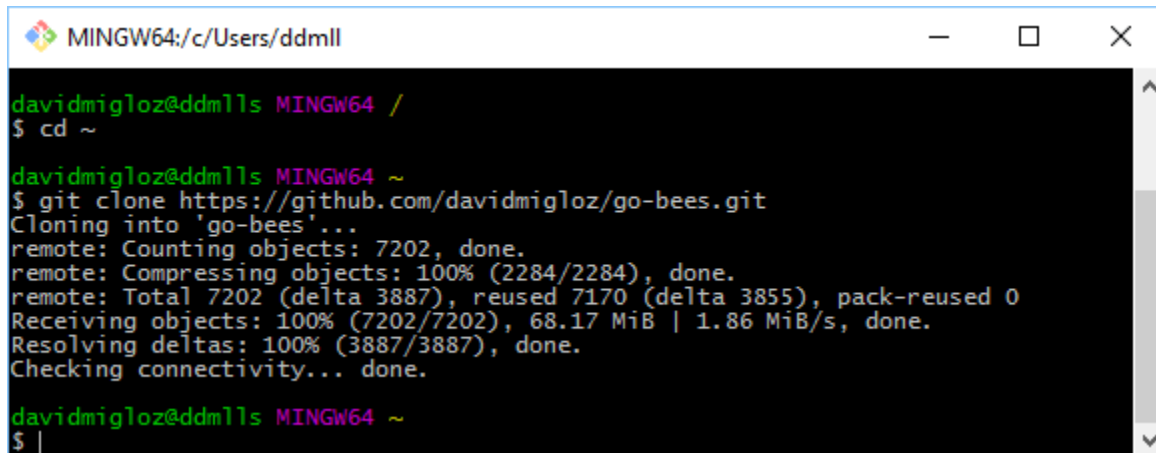
Android Studio es el IDE oficial para el desarrollo de aplicaciones Android. Está basado en IntelliJ IDEA de JetBrains. Proporciona soporte para Gradle, emulador, editor de *layouts*, refactorizaciones específicas de Android, herramientas Lint para detectar problemas de rendimiento, uso, compatibilidad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

Se puede obtener desde [\[android:androidstudio\]](#). Junto con Android Studio se instala también el Android SDK y *Android Virtual Device* (AVD).



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 [\[git:scm\]](https://git-scm.com). Una vez instalado, trabajaremos con Git Bash.



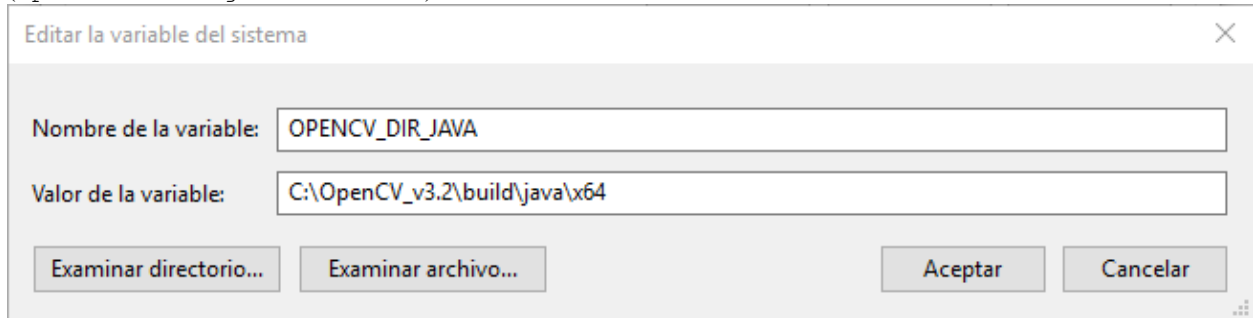
OpenCv

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 [\[opencv:web\]](https://opencv.org). 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).



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.

```
MINGW64:/c/Users/ddmll
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 ~
$ |
```

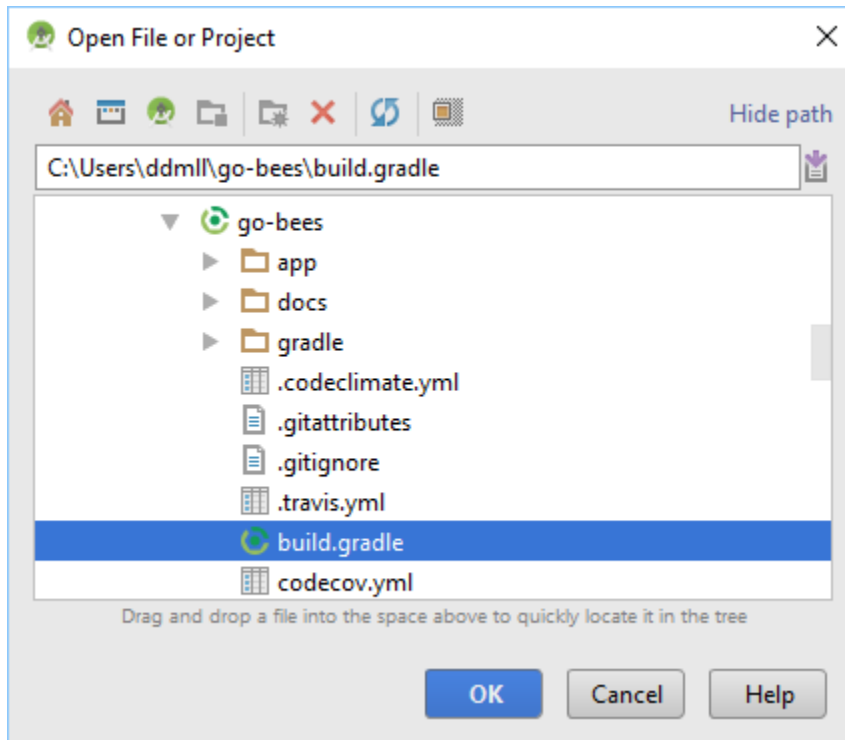
Para conocer el proceso detalladamente consultar [\[github:clone\]](#).

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.



Para conocer el proceso detalladamente consultar [\[android:import\]](#).

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 "Add 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:vntnumberpickerpreference: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'
}
```

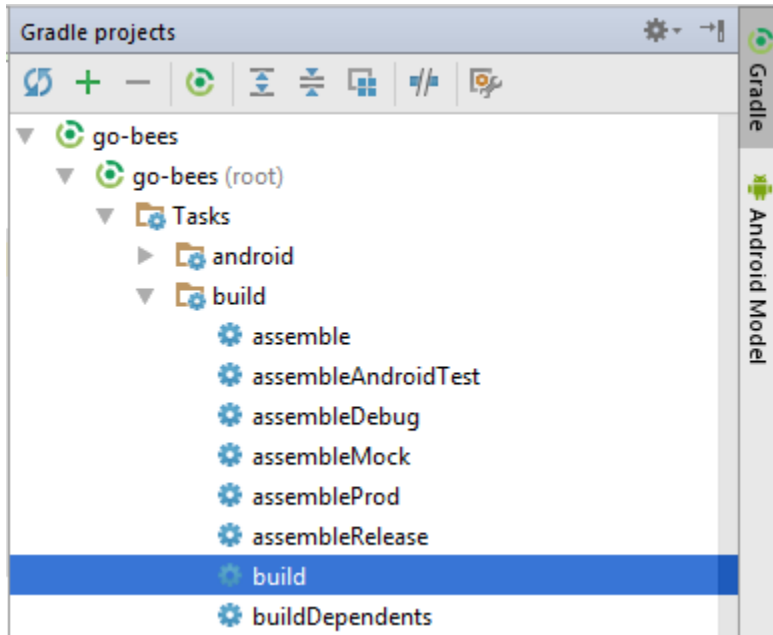
Se puede observar que existen tres formas de importar las dependencias, cada una define con un ámbito de aplicación distinto:

- `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.



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

Para conocer el proceso detalladamente consultar [\[android:compilerun\]](#).

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 [\[android:compilerun\]](#).

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 [\[android:compilerun\]](#).

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`.

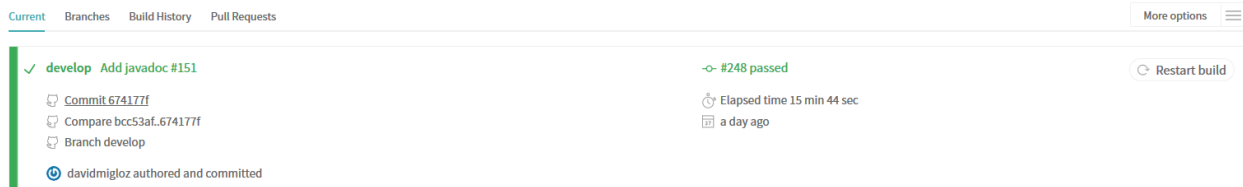
Las secciones más importantes de este fichero son:

- `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 [\[travis:gobees\]](#).

davidmigloz / go-bees  



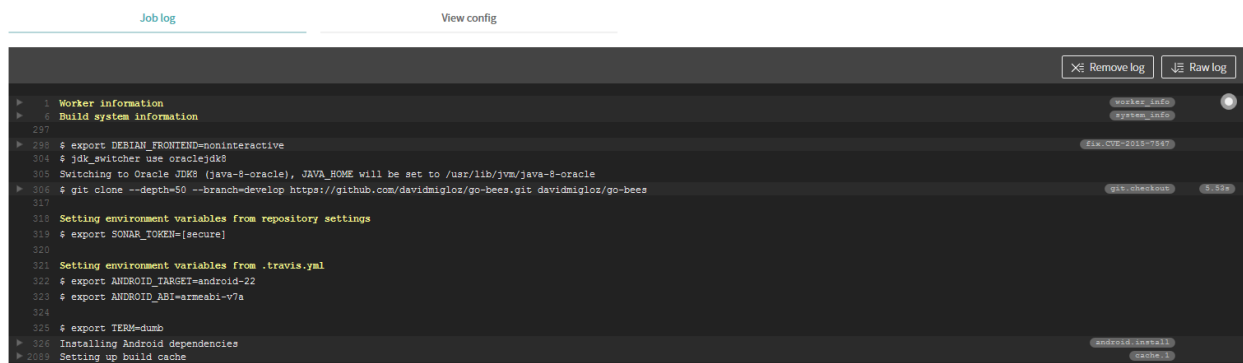
Current Branches Build History Pull Requests More options

✓ develop Add javadoc #151 -> #248 passed Restart build

Commit 674177f
Compare bcc53af..674177f
Branch develop

Elapsed time 15 min 44 sec
a day ago

davidmigloz authored and committed



Job log View config

1 Worker information
2 Build system information
297
298 \$ export DEBIAN_FRONTEND=noninteractive
304 \$ jdk_switcher use oraclejdk8
305 Switching to Oracle JDK8 (java-8-oracle), JAVA_HOME will be set to /usr/lib/jvm/java-8-oracle
306 \$ git clone --depth=50 --branch=develop https://github.com/davidmigloz/go-bees.git davidmigloz/go-bees
317
318 Setting environment variables from repository settings
319 \$ export SONAR_TOKEN=[secure]
320
321 Setting environment variables from .travis.yml
322 \$ export ANDROID_TARGET=android-22
323 \$ export ANDROID_ABI=armeabi-v7a
324
325 \$ export TERM=dumb
326 Installing Android dependencies
327 Setting up build cache

Para saber más, acceder a su documentación [\[travis:doc\]](#).

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`.

Add javadoc #151

32.18%

38.87%

 davidmigloz a day ago ✓ CI Passed
674177f ↗ develop 🔒 Not found

Diff
Files
Build
Graphs

/ ... / src / main / java / com / davidmiguel / gobees								
Files					Complexity		Coverage	
about	131	36	1	94		28.94%		27.48%
addeditapiary	252	82	12	158		32.58%		32.53%
addedithive	114	44	5	65		38.09%		38.59%
apiaries	266	48	11	207		19.04%		18.04%
apiary	290	44	8	238		13.69%		15.17%
data	876	422	38	416		56.98%		48.17%
help	45	6	0	39		15.38%		13.33%
hive	324	49	10	265		14.86%		15.12%
monitoring	772	180	0	592		25.83%		23.31%
recording	278	31	0	247		11.29%		11.15%
settings	88	32	2	54		33.33%		36.36%
utils	359	253	13	93		70.04%		70.47%
splash/SplashActivity.java	6	0	0	6		0.00%		0.00%
Folder Totals (13 files)	3,801	1,227	100	2,474		39.00%		32.28%
Project Totals (82 files)	3,812	1,227	100	2,485		38.87%		32.18%

Para saber más, acceder a su documentación [\[codecov:doc\]](#).

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 [\[codeclimate:gobees\]](#).

Feed
Code
Issues
Branches
Trends
Builds


Summary of January 16th - 22nd
9 files changed, 1,035 insertions, 16 deletions

+
Ten classes/modules were added. 19 days ago
Compare

A

app/src/main/java/com/davidmiguel/gobees/help/HelpActivity.java

A

app/src/main/java/com/davidmiguel/gobees/help/HelpContract.java

A

app/src/main/java/com/davidmiguel/gobees/help/HelpFragment.java

A

app/src/main/java/com/davidmiguel/gobees/help/HelpPresenter.java

A

app/src/main/java/com/davidmiguel/gobees/about/Library.java

A

app/src/main/java/com/davidmiguel/gobees/about/AboutActivity.java

A

app/src/main/java/com/davidmiguel/gobees/about/AboutPresenter.java

A

app/src/main/java/com/davidmiguel/gobees/about/AboutContract.java

code climate

4.0

coverage

unknown

Link to Code Climate from your README.

Hotspots

Huzzah! This repo has no classes or modules worse than a B.

Para saber más, acceder a su documentación `[codeclimate:doc]`.

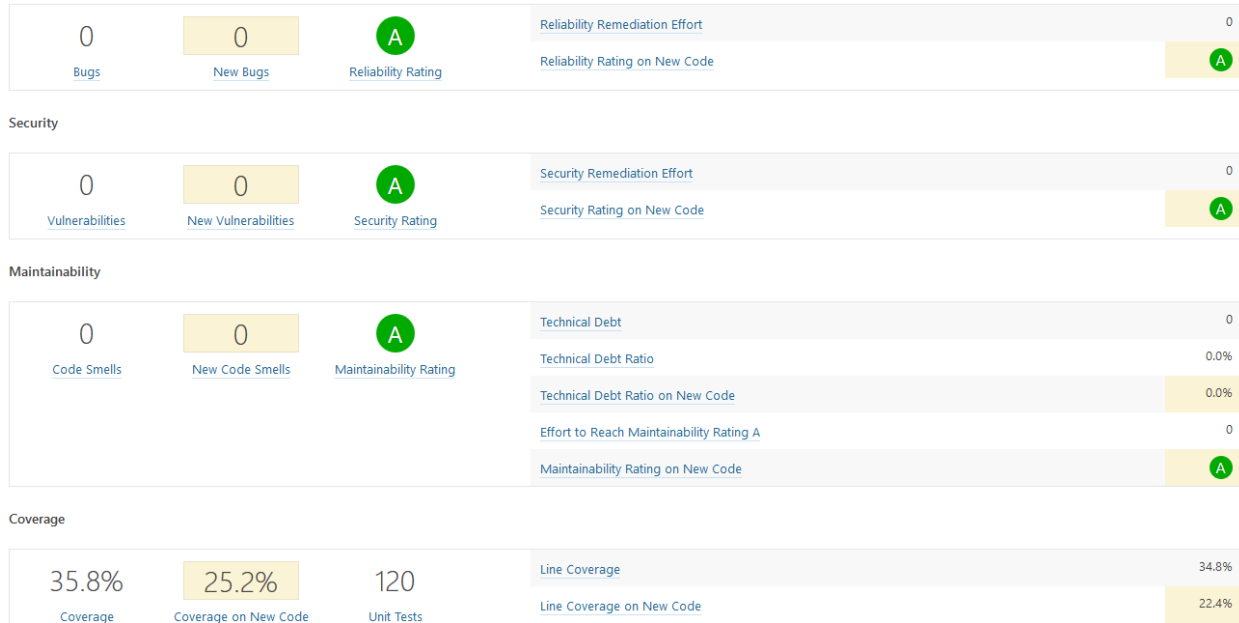
SonarQube es una plataforma de código abierto para la revisión continua de la calidad de código. Permite detectar código duplicado, violaciones de estándares, cobertura de test unitarios, *bugs* potenciales, etc.

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
    }
}
```

Los resultados de los análisis son accesibles desde `[sonarqube:gobees]`.



Para saber más, acceder a su documentación [[sonarqube:doc](#)].

VersionEye

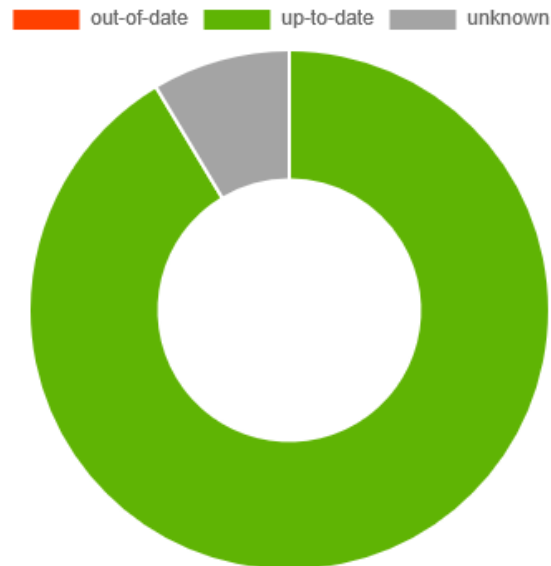
VersionEye es una herramienta que monitoriza las dependencias del proyecto y envía notificaciones cuando alguna de estas está desactualizada, es vulnerable o viola la licencia del proyecto.

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 [[versioneye:gobees](#)].

Dependency	Resolved	Newest	License
junit	4.12	4.12	EPL-1.0
mockito-all	2.0.2-beta	2.0.2-beta	MIT
powermock-module-junit4	1.6.6	1.6.6	Apache-2.0
powermock-api-mockito	1.6.6	1.6.6	Apache-2.0
slf4j-api	1.7.22	1.7.22	MIT
slf4j-log4j12	1.7.22	1.7.22	MIT
log4j	1.2.17	1.2.17	Apache-2.0

Visual Dependencies



Para saber más, acceder a su documentación [\[versioneye:doc\]](#).

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`.



Para saber más, acceder a su documentación [[readthedocs:doc](#)].

Pruebas del sistema

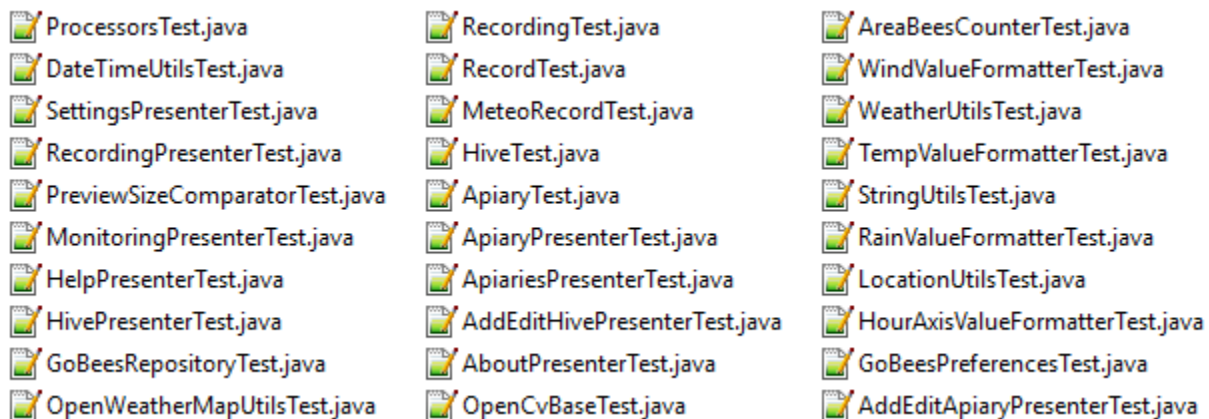
Para verificar el funcionamiento de cada uno de los módulos de la aplicación, su integración y la interacción con estos desde la interfaz, se han desarrollado una serie de baterías de test.

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.

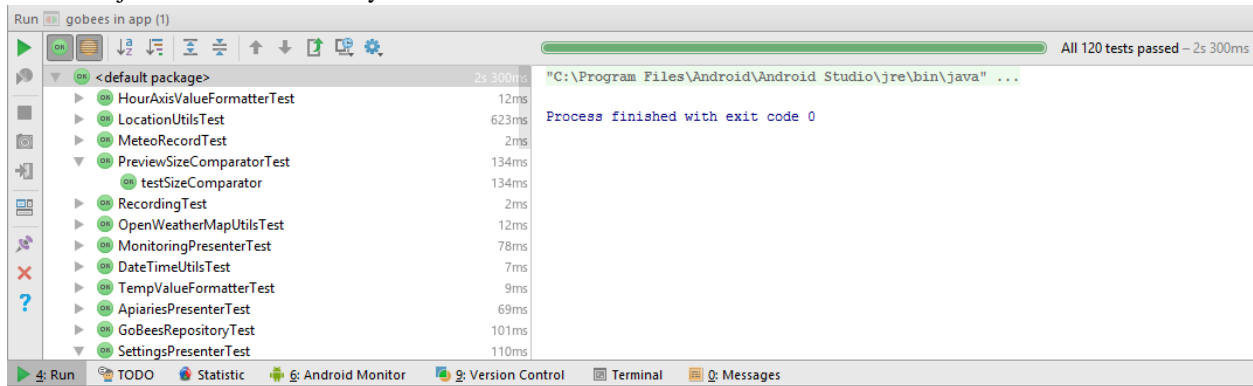
Se han escrito 120 test unitarios que testean 30 clases distintas. Se han testeado en su mayoría los *presenters* que son los que poseen la lógica de la aplicación y no tienen ninguna dependencia al *framework* de Android. Lo que permite ejecutarlos sin necesidad de lanzar un emulador.



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.



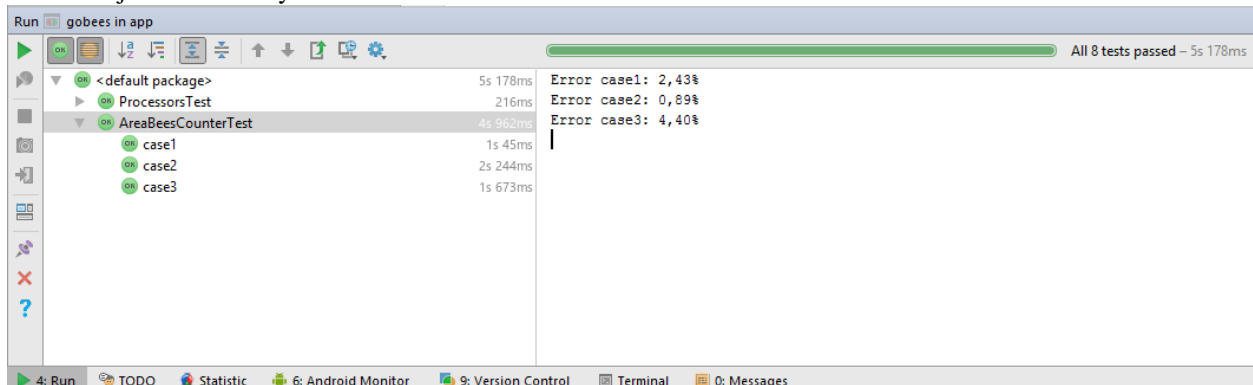
Test del algoritmo

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.

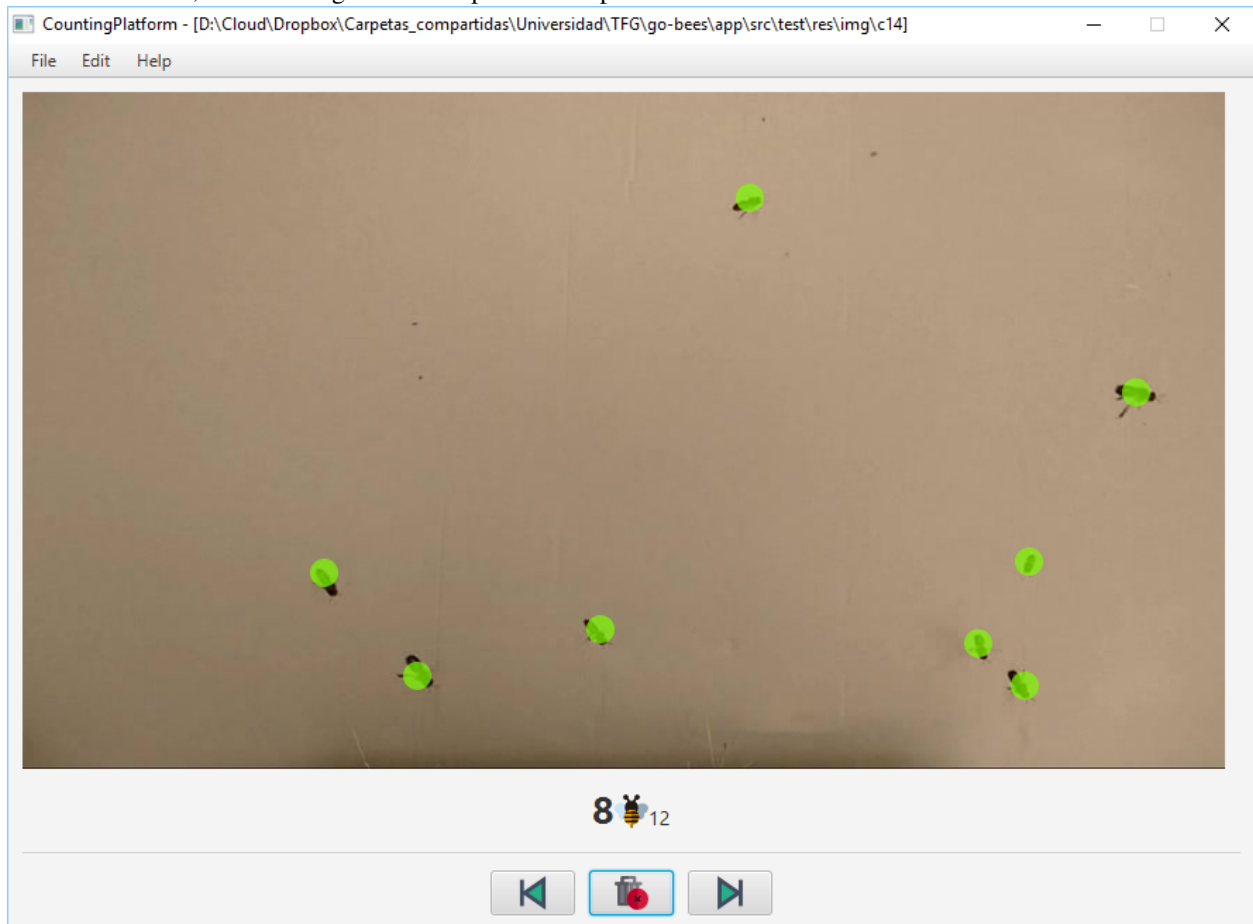


Etiquetado de nuevos conjuntos de fotogramas

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 [\[github:extraapps\]](#)).
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.



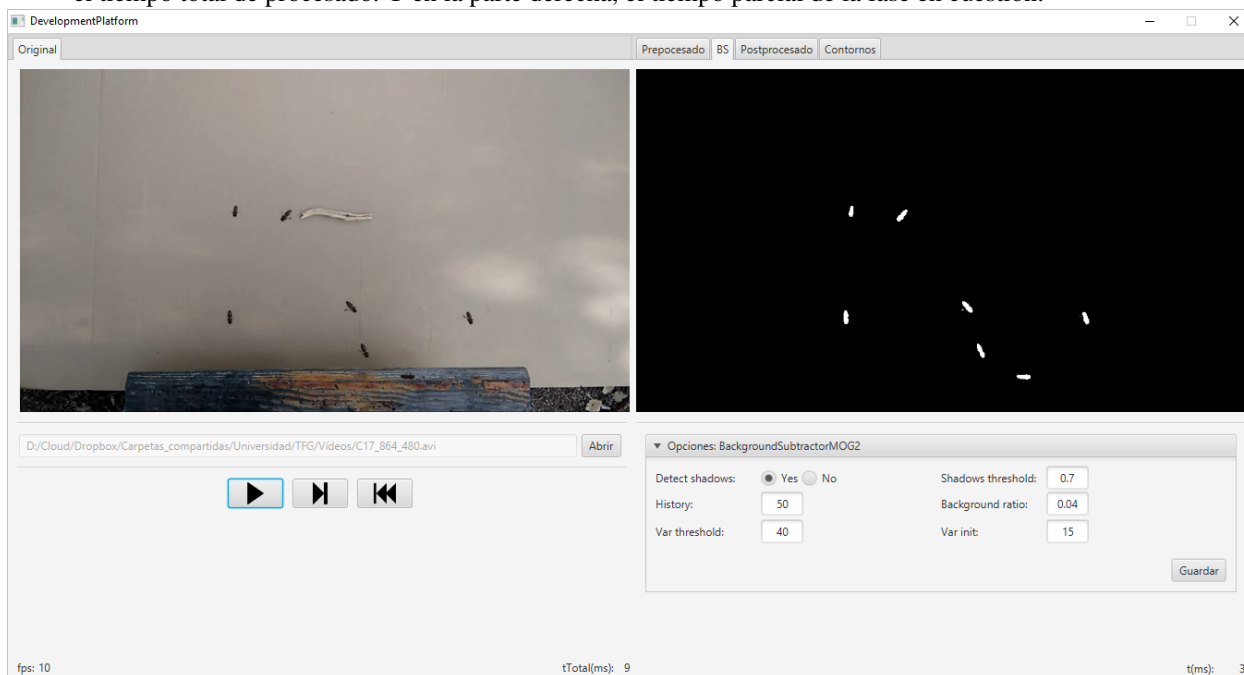
Testeo de la parametrización del algoritmo

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 [\[github:extraapps\]](#). 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.



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	Requisito
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

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.

Manual de usuario

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.

Requisitos de usuarios

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

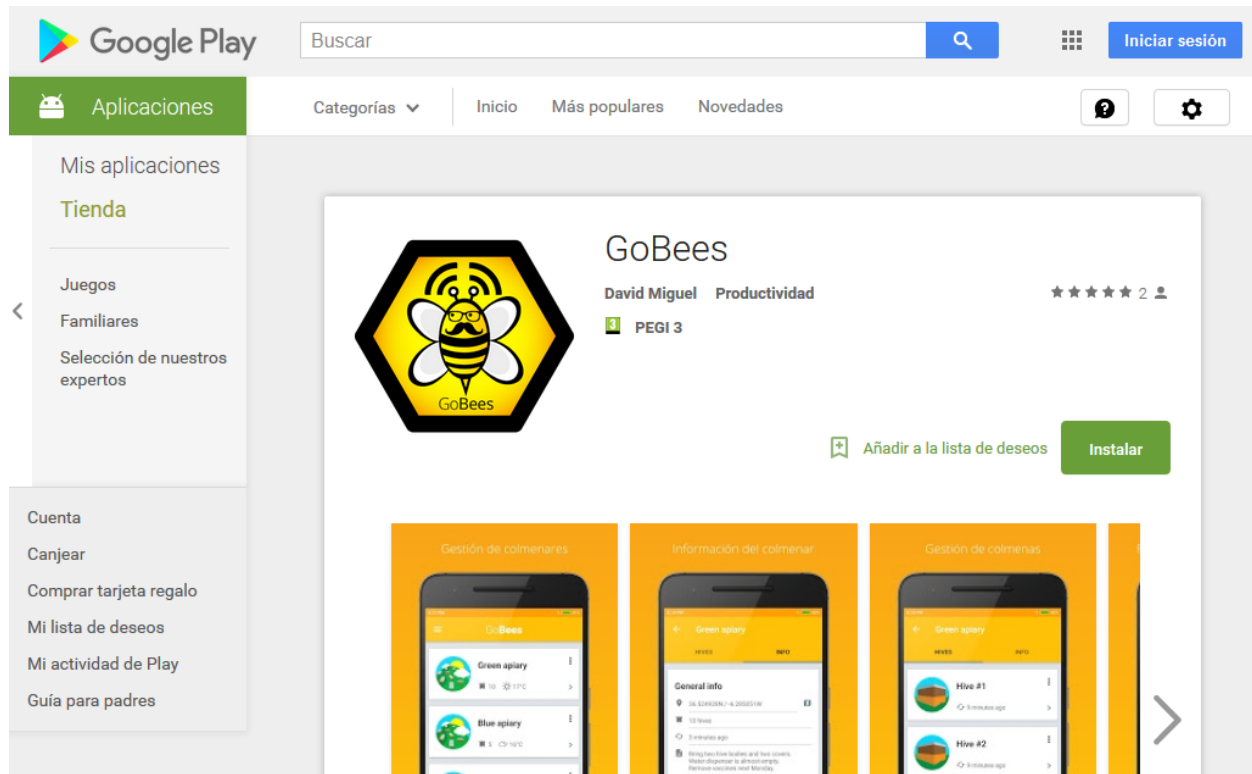
- Contar con un dispositivo que posea Android 4.4 (*KitKat* – API 19) o superior.
- Para utilizar la característica de monitorización de la actividad, es necesario tener instalada la aplicación [OpenCV Manager](#).
- También se necesita contar con permiso para acceder a la cámara del dispositivo.
- Si 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.

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.



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



GoBees

David Miguel



PEGI 3

3,97 MB/9,08 MB

43%



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 en el siguiente enlace:

<https://github.com/davidmigloz/go-bees/releases>

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.

Manual de usuario

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

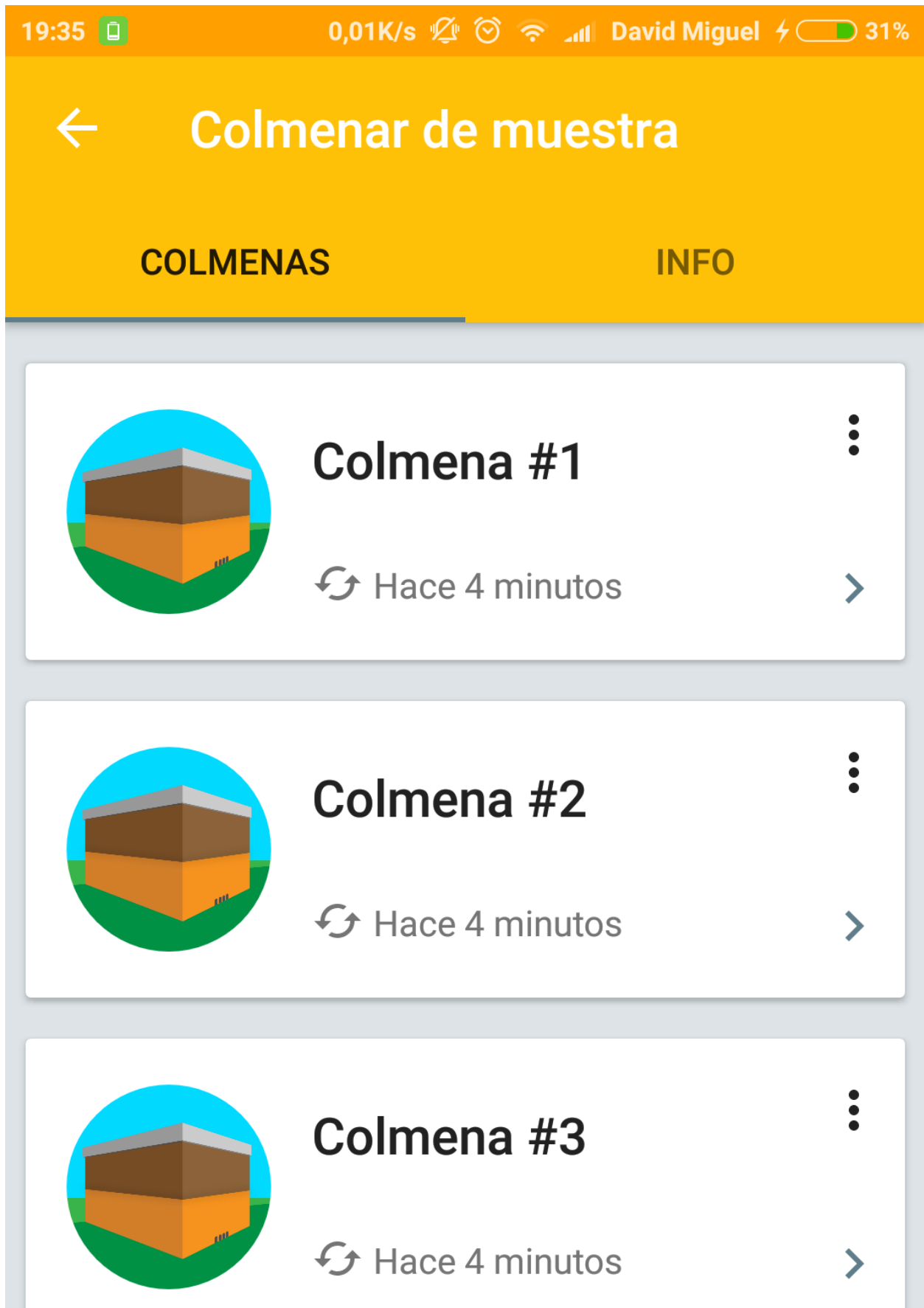
Generar datos de muestra

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.



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.

14:23 0,00K/s    David Miguel   31%

Añadir colmenar



Nombre

"Latitud (ej. 42.352083)"



"Longitud (e.g. -3.697586)"



Notas...

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.

14:25

0,00K/s David Miguel 32%

Colmenar de muestra

COLMENAS

INFO

Hace 0 minutos

Traer dos cajones y dos tapas.
El bebedero está casi vacío.
Quitar vacunas el próximo lunes.

13°C

Sobrón

Tiempo

Llovizna ligera

Humedad

62%

Presión

1015 hPa

Viento

4 km/h - SW

Lluvia

0,00 mm

Nieve

0,00 mm

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:



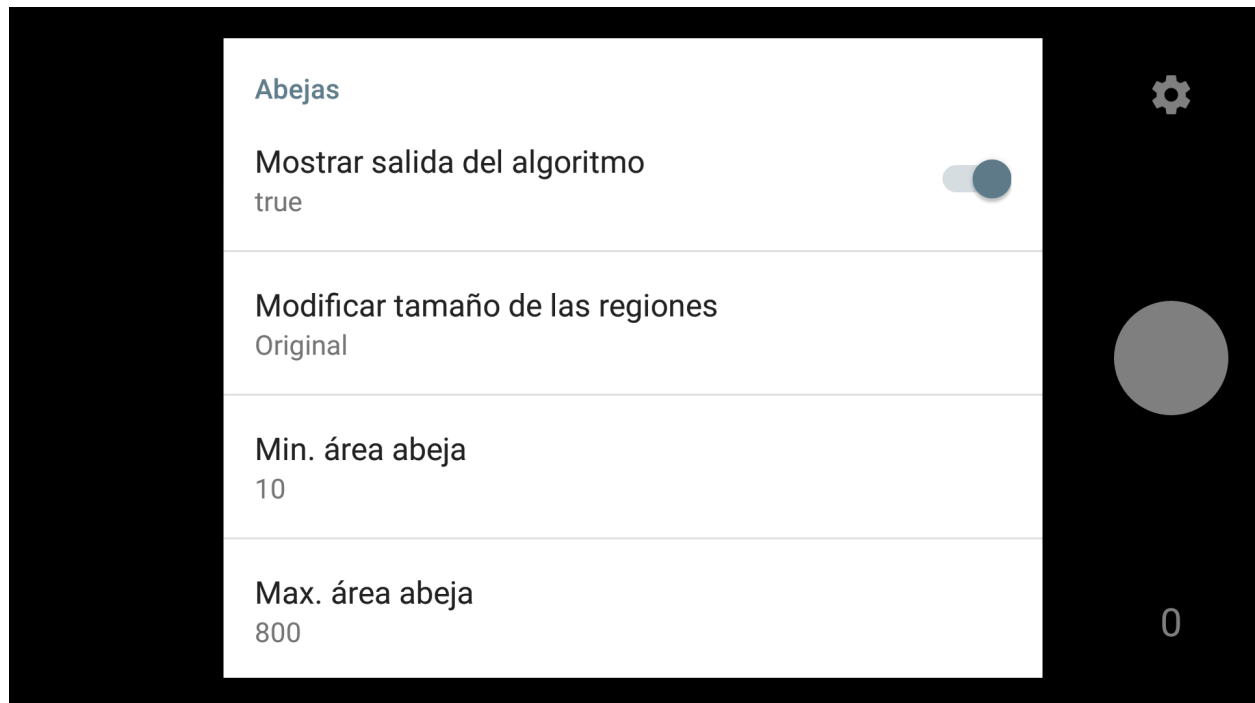
Para mejorar los resultados de la monitorización, es recomendable que el suelo sea de un color claro y uniforme. Si posee maleza, se puede colocar un cartón o similar, como se muestra en la imagen.

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.

*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.



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.

14:27



0,00K/s



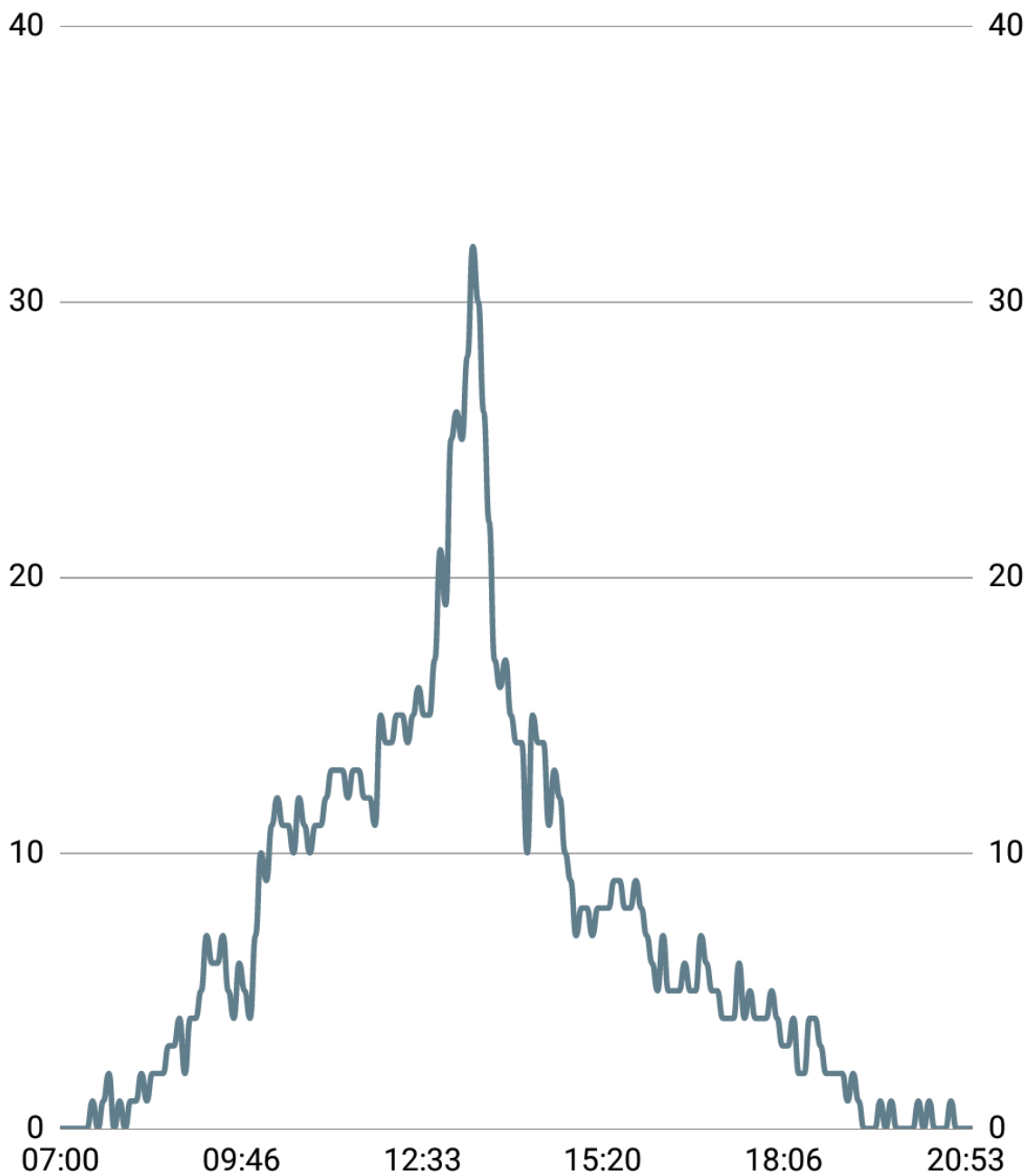
David Miguel



32%



Mar. 20 sept., 2016



N° abejas



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.

14:28  0,00K/s  David Miguel  32%



Acerca de GoBees



GoBees App

Versión v1.0

[WEBSITE](#) [LICENCIA](#) [CHANGELOG](#)

Desarrollado por David Miguel Lozano.

Bibliotecas Open Source:

Android Support Library	Apache v2.0
Google Play Services	Apache v2.0
Guava	Apache v2.0
Material Design Icons	Apache v2.0

- [art:bees_decline] <http://www.greenpeace.org/switzerland/Global/international/publications/agriculture/2013/BeesInDecline.pdf>
- [art:ccd] <https://agresearchmag.ars.usda.gov/AR/archive/2008/May/colony0508.pdf>
- [art:campbell2008] http://homepages.inf.ed.ac.uk/rbf/VAIB08PAPERS/vaib9_mummert.pdf
- [art:lundie1925] <https://archive.org/details/flightactivities1328lund>
- [art:struye1994] <https://hal.archives-ouvertes.fr/hal-00891170/document>
- [art:campbell2005] <http://stacks.iop.org/0957-0233/16/i=12/a=015>
- [beebarcodes] <http://www.uprintlabels.com/barcode-labels-uses.html>
- [art:decourtye_honeybee_2011] <https://www.ncbi.nlm.nih.gov/pubmed/21267650>
- [art:chiron2013a] http://link.springer.com/chapter/10.1007%2F978-3-642-41181-6_71
- [art:chiron2013] <http://jivp.eurasipjournals.springeropen.com/articles/10.1186/1687-5281-2013-59>
- [art:tashakkori2015] <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7133029>
- [opencv:color_cvt] http://docs.opencv.org/3.1.0/de/d25/imgproc_color_conversions.html
- [wiki:grayscale] <https://en.wikipedia.org/wiki/Grayscale>
- [wiki:gaussian] https://en.wikipedia.org/wiki/Gaussian_blur
- [book:mastering_opencv] <https://www.packtpub.com/application-development/mastering-opencv-android-application-programming>
- [wiki:bs] https://en.wikipedia.org/wiki/Background_subtraction
- [programarfacil:detmov] <http://programarfacil.com/blog/vision-artificial/deteccion-de-movimiento-con-opencv-python/>
- [book:opencv_java] <https://www.packtpub.com/application-development/opencv-30-computer-vision-java>
- [coursera:gmm] <https://www.coursera.org/learn/robotics-learning/lecture/XG0WD/1-4-1-gaussian-mixture-model-gmm/>
- [art:yao_improved_2001] <http://www.ee.surrey.ac.uk/CVSSP/Publications/papers/KaewTraKulPong-AVBS01.pdf>
- [opencv:bs_tutorial] http://docs.opencv.org/master/db/d5c/tutorial_py_bg_subtraction.html
- [art:zivkovic_improved_2004] <http://www.zoranz.net/Publications/zivkovic2004ICPR.pdf>
- [art:zivkovic_efficient_2006] <http://www.zoranz.net/Publications/zivkovicPRL2006.pdf>

[github:background_segm] https://github.com/opencv/opencv/blob/master/modules/video/include/opencv2/video/background_segm.hp

[github:bgfg_gaussmix2] https://github.com/opencv/opencv/blob/master/modules/video/src/bgfg_gaussmix2.cpp

[opencv:mog] http://docs.opencv.org/3.1.0/d7/d7b/classcv_1_1BackgroundSubtractorMOG2.html

[art:li_foreground_2003] <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.62.8313&rep=rep1&type=pdf>

[opencv:contour] http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_imgproc/py_contours/py_table_of_contents_contours/py_table_o

[opencv:find_contour] docs.opencv.org/3.0-beta/modules/imgproc/doc/structural_analysis_and_shape_descriptors.html

[book:android_opencv] <https://www.packtpub.com/application-development/android-application-programming-opencv-3>

[book:mastering_opencv] <https://www.packtpub.com/application-development/mastering-opencv-android-application-programming>

[book:opencv_java] <https://www.packtpub.com/application-development/opencv-30-computer-vision-java>

[book:learning_cv] <https://www.packtpub.com/application-development/learning-image-processing-opencv>

[course:android_beginners] <https://www.udacity.com/course/android-development-for-beginners-ud837>

[course:developing_android] <https://www.udacity.com/course/new-android-fundamentals-ud851>

[course:testing] <https://codelabs.developers.google.com/codelabs/android-testing/>

[gobees:prototypes] <https://github.com/davidmigloz/opencv-android-gradle-repo/>

[android:versions] <https://developer.android.com/about/dashboards/>

[art:campbell2008] http://homepages.inf.ed.ac.uk/rbf/VAIB08PAPERS/vaib9_mummert.pdf

[art:chiron2013] <http://jivp.eurasipjournals.springeropen.com/articles/10.1186/1687-5281-2013-59>

[art:tashakkori2015] <http://ieeexplore.ieee.org/document/7133029/?arnumber=7133029>

[wiki:licencia] <https://es.wikipedia.org/w/index.php?title=Licencia&oldid=94243114>

[license:gplv3] <https://www.gnu.org/licenses/gpl-3.0.txt>

[license:epl] <https://www.eclipse.org/legal/epl-v10.html>

[license:ccby4] <https://creativecommons.org/licenses/by/4.0>

[ss_cotizacion] http://www.seg-social.es/Internet_1/Trabajadores/CotizacionRecaudaci10777/Basesytiposdecotiza36537/index.htm

[ieee_830_1998] <https://standards.ieee.org/findstds/standard/830-1998.html>

[pattern:mvp] <https://martinfowler.com/eaDev/uiArchs.html>

[pattern:repository] <https://martinfowler.com/eaCatalog/repository.html>

[pattern:android_architecture] <https://github.com/googlesamples/android-architecture>

[wiki:injection] https://en.wikipedia.org/wiki/Dependency_injection

[design:material] <https://material.io/guidelines/>

[android:folders] <https://developer.android.com/studio/projects/index.html?hl=es>

[java:jdk7] <http://www.oracle.com/technetwork/es/java/javase/downloads/jdk7-downloads-1880260.html>

[android:androidstudio] <https://developer.android.com/studio/index.html?hl=es>

[git:scm] <https://git-scm.com/>

[opencv:web] <http://opencv.org/>

[github:clone] <https://help.github.com/articles/cloning-a-repository/>

[android:import] <https://www.jetbrains.com/help/idea/2016.3/importing-an-existing-android-project.html>
[android:compilerun] <https://developer.android.com/studio/run/index.html?hl=es-419>
[travis:gobees] <https://travis-ci.org/davidmigloz/go-bees>
[travis:doc] <https://docs.travis-ci.com/>
[codecov:doc] <https://docs.codecov.io/>
[codeclimate:gobees] <https://codeclimate.com/github/davidmigloz/go-bees>
[codeclimate:doc] <https://docs.codeclimate.com/>
[sonarqube:gobees] <https://sonarqube.com/dashboard?id=go-bees>
[sonarqube:doc] <https://docs.sonarqube.org/>
[versioneye:gobees] <https://www.versioneye.com/user/projects/57f7b19e823b88004e06ad33>
[versioneye:doc] <https://www.versioneye.com/documentation>
[readthedocs:doc] <http://docs.readthedocs.io/en/latest/>
[github:extraapps] <https://github.com/davidmigloz/go-bees-prototypes/releases>