Análisis, diseño e implementación de un sistema prototipo basado en un panel de administración web y aplicaciones móviles multiplataforma para la clasificación, seguimiento y trazabilidad de ganado caprino

Alumno
Joel David Delgado Perdomo
Máster Universitario en Ingeniería Informática (Programación de Dispositivos Móviles)

Tutores
Dr. Abraham Rodríguez Rodríguez
José Luis González Santana

Julio 2019 – Las Palmas de Gran canaria

Trabajo de Fin de Máster en Ingeniería Informática (Desarrollo de aplicaciones móviles) de la Universidad de Las Palmas de Gran Canaria presentado por el alumno:

Joel David Delgado Perdomo

Título del proyecto

Análisis, diseño e implementación de un sistema prototipo basado en un panel de administración web y aplicaciones móviles multiplataforma para la clasificación, seguimiento y trazabilidad de ganado caprino

Tutor

Dr. Abraham Rodríguez Rodríguez

Agradecimientos

A mi tutor Abraham Rodríguez Rodríguez, por su apoyo incondicional desde el inicio, con su disponibilidad total desde el primer momento, su atención y cariño.

A mis compañeros del Máster de mi promoción por su ayuda, compañerismo y amistad.

A Ana Guedes, la mejor profesora que he tenido nunca, con infinita paciencia durante todos los años de mi carrera y por subirme los ánimos cada vez que pensé que ni llegaría a ser ingeniero.

A Evee, por aguantarme día sí día también, durante más de un semestre con todas las prácticas y dolores de cabeza que eso implica.

A mis padres, mi padrino y mis amigos (ustedes saben quienes son), que con el amor más grande siempre han mirado por mi y me han ayudado durante mi vida académica y personal.

Resumen

En la ganadería Cabra Palmera, realizan la gestión, trazabilidad, estudios de genética y registro de nacimientos del ganado caprino haciendo uso de un sistema basado en MS-DOS y la toma de datos de campo con lápiz y papel.
Para los técnicos veterinarios puede ser una ardua tarea pues tienen que procesar toda la información tomada a papel por los ganaderos y ellos mismos para posteriormente digitalizarlos en un programa antiguo con el que nadie está familiarizado y que requiere del uso de máquinas virtuales y complejidades informáticas.
Por ello, proponemos una sencilla aplicación móvil multiplataforma que permite la rápida toma de estos datos y su consulta directamente desde el móvil, facilitando sustancialmente el duro trabajo de los técnicos y ganaderos.

Abstract

In Cabra Palmera ranch, they are in charge of the management, traceability, genetic studies and birth registration of goats using a system based on MS-DOS and the data on the field with pencil and paper.
For veterinary technicians it can be an arduous task because they have to process all the information taken by the farmers and themselves from paper and then digitize them into an old program that nobody is familiar with and that requires the use of virtual machines and computer science complexities.
Therefore, we propose a simple multiplatform mobile application that allows the quick taking of this data and its consultation directly from the mobile phone, substantially facilitating the hard work of technicians and ranchers.

Índice

Índice de ilustraciones

Índice de tablas

Introducción

La gestión en las ganaderías hoy en día se podría considerar arcaica para el año que vivimos. Dadas las grandes oportunidades que nos dan las tecnologías de la información actualmente, parece algo raro que aún exista un sector con gran volumen de datos que esté sin apenas explotar en términos de aplicaciones modernas. Por ello, hemos decidido realizar un proyecto que pueda colaborar a remediar este hecho en la medida de lo posible y de lo que nuestros conocimientos en la informática y nuestra capacidad de buscar soluciones nos lo permitan. Nos hemos centrado en una actividad que hasta ahora se realiza con papel y lápiz en muchos de sus pasos y conllevan el uso de aplicaciones antiguas poco intuitivas y en algunos puntos tediosas de utilizar y por que por lo tanto ralentizan el proceso.

Por este motivo, hemos decidido realizar una propuesta para la ganadería de Cabra Palmera en la que lidiar con estas aplicaciones forme parte del pasado y se agilice el trabajo diario de los técnicos veterinarios y ganaderos de esta. La finalidad de nuestra propuesta es simplificar al máximo posible el número de pasos de los diversos procesos que tienen que registrar digitalmente y crear un producto de referencia en este campo.

Para lograrlo, hemos desarrollado una solución que permite a los ganaderos registrar a los nacimientos de los animales, darlos de baja, realizar transferencias a otras ganaderías, identificarlos mediante la lectura de su código identificativo usando uso de la cámara, de su identificador numérico único o del nombre que le hayan dado y visualizar su información completa de una manera rápida y clara. Los técnicos veterinarios por su parte pueden realizar también una identificación de los animales, actualizar el estado de las crías cuando pasan a su etapa adulta y asignarles un identificador numérico único y calificar a los animales en distintos apartados para valorar su condición.

Proveemos además en este mismo proyecto, de un panel de administración en el que se pueden visualizar e importar los datos en formato CSV para su posterior gestión desde la aplicación móvil, simplificando la transición de los sistemas actuales al propuesto.

Para la conexión al *backend* del sistema se ha utilizado la plataforma de *Firebase* con el uso de *Cloud Functions* para la gestión de algunas operaciones.

Estructura del documento

Esta memoria del proyecto está realizada por capítulos, descritos más en detalle a continuación:

1. Competencias específicas cubiertas

   Enumeramos y justificamos como se han cubierto las competencias específicas del título durante la realización de este proyecto.

2. Aportaciones

   Explicamos cuáles son las aportaciones que nuestro proyecto tiene que hacer al mercado actual y al espacio socioeconómico, señalando además cuál ha sido el impacto personal que ha tenido en nosotros.

3. Normativa y legislación

   Indicamos nuestro análisis de la normativa y legislación vigentes en las partes que corresponden a este proyecto y las medidas que se han tomado en este proyecto para cumplir con ellas, incluyendo además información sobre las licencias utilizadas en los diferentes productos software utilizados durante el desarrollo del proyecto y el tipo de licencia que usará nuestro proyecto.

4. Metodología de trabajo y planificación del proyecto

   Señalamos cuál de las distintas metodologías de trabajo para el desarrollo de software hemos escogido y justificamos nuestra elección, mostrando además la planificación del proyecto detenidamente.

5. Análisis

   Establecemos el punto de partida del proyecto, dejando clara la situación actual del problema a resolver, las herramientas utilizadas previamente por el cliente y los objetivos propuestos que queríamos alcanzar con el proyecto.
   Detallamos y señalamos el análisis realizado del proyecto, identificando los requisitos funcionales y la esquematización de estos mediante diagramas de casos de uso.

6. Diseño

   Especificamos el diseño y la arquitectura del sistema escogidas para el sistema, justificándose debidamente y detallando su implementación y funcionamiento. También se incluye una especificación de las colecciones utilizadas en nuestra base de datos *No-SQL*.

7. Desarrollo

   Introducimos cuáles han sido las tecnologías escogidas para el desarrollo e implementación del proyecto y cuáles han sido las herramientas seleccionadas para ello.
   Tenemos un vistazo a cómo ha sido la realización del proyecto, justificando la elección del *framework* escogido, la estructura de archivos del proyecto de la aplicación móvil, la estructura de archivos del proyecto de la aplicación web de escritorio y como se han configurado para trabajar conjuntamente.

8. Pruebas

   Mostramos las pruebas que hemos realizado para validar el funcionamiento de la aplicación, separadas por pruebas de usabilidad y pruebas de integración.

9. Resultados, conclusiones y trabajo futuro

   Finalizamos el conjunto de esta memoria, indicando cuáles han sido nuestras reflexiones, resultados finales, posibles cambios y mejoras que podrían añadirse al proyecto en un futuro no muy lejano.

10. Bibliografía

    Declaramos todas las referencias a las que se ha hecho alusión en algún momento de la memoria del proyecto, sus autores y sus fuentes respectivas.

11. Anexos
    I. Explicamos de manera clara y simple como hacer uso de la aplicación móvil.
    II. Detallamos como instalar la aplicación móvil y el portal en escritorio para su desarrollo y testeo en su versión web.
    III. Destacamos la presentación del proyecto en el Foro Nacional de Caprino.

1. Competencias específicas cubiertas

1.1 Comunes

C01

Capacidad para proyectar, calcular y diseñar productos, procesos e instalaciones en todos los ámbitos de la ingeniería informática.

Durante este proyecto hemos tenido que diseñar, proyectar y calcular el producto completo final a entregar al cliente, teniendo en cuenta sus necesidades, los costes y los procesos de este, asegurando un resultado de calidad.

C08

Capacidad para la aplicación de los conocimientos adquiridos y de resolver problemas en entornos nuevos o poco conocidos dentro de contextos más amplios y multidisciplinares, siendo capaces de integrar estos conocimientos.

Dada la naturaleza del proyecto y la poca explotación del sector en el apartado de la ingeniería informática, es cuanto menos interesante intentar adaptar soluciones a un entorno poco conocido en el mundo de la informática, por eso necesitamos enriquecernos de los conocimientos de este contexto, y una vez adquiridos nos sitúan en una posición pionera en el trabajo de este campo.

1.2 Específicas de tecnologías informáticas

TI01

Capacidad para modelar, diseñar, definir la arquitectura, implantar, gestionar, operar, administrar y mantener aplicaciones, redes, sistemas, servicios y contenidos informáticos.

En este proyecto hemos abarcado las fases de análisis diseño y desarrollo características de un producto informático completo. Hemos diseñado e implementado una arquitectura de sistema informático, con su correspondiente aplicación móvil y web que permite el uso de redes y servicios entre sí y propicia de información al usuario final.

TI05

Capacidad para analizar las necesidades de información que se plantean en un entorno y llevar a cabo en todas sus etapas el proceso de construcción de un sistema de información.

Dada la naturaleza del proyecto hemos realizado un estudio completo de la solución existente ya implementada en la compañía del cliente para mejorarlo en diversos aspectos con nuestra propuesta, especialmente aquellos que han quedado anticuados con la rápida evolución en las tecnologías de la información en los últimos años.

TI10

Capacidad para utilizar y desarrollar metodologías, métodos, técnicas, programas de uso especifico, normas y estándares de computación gráfica.

En el proyecto hemos hecho hincapié en la experiencia de usuario con la aplicación móvil teniendo en cuenta distintos principios y técnicas de diseño de interacción de usuario para asegurar que la misma sea lo mejor posible.

TI11

Capacidad para conceptualizar, diseñar, desarrollar y evaluar la interacción persona–ordenador de productos, sistemas y servicios informáticos.

Para este proyecto se han realizado dos interfaces de usuario, una aplicación móvil y una aplicación web de escritorio, en la que los usuarios del producto puedan utilizar el mismo con comodidad y sencillez. Se han aplicado diversos principios de diseño de interfaces de usuario para asegurar la simplicidad y una suave curva de aprendizaje del sistema.

TI12

Capacidad para la creación y explotación de entornos virtuales, y para la creación y distribución de contenidos multimedia.

Una parte importante del proyecto es el uso de la cámara para la identificación del código de los animales usando la *API* de *Google Vision* así como añadir imágenes de los mismos a la aplicación para su rápida identificación en el apartado de fichas.

2. Aportaciones

2.1 Entorno socioeconómico

El sistema desarrollado se centra en dos aspectos principales: el primero, unificar el uso de tres aplicaciones distintas con distintos propósitos en el mismo sistema, y el segundo, mejorar el flujo de trabajo y los procesos a realizar tanto por los ganaderos como los técnicos veterinarios de la empresa cliente.

La unificación de las tres aplicaciones tiene varias repercusiones, entre ellas destacamos las siguientes:

- Simplificamos la curva de aprendizaje del sistema.
- Ahorramos numerosas horas de trabajo en simplemente gestión de datos entre aplicación y aplicación.
- Eliminamos la necesidad de utilizar máquinas virtuales, sistemas operativos antiguos y software terceros para la realización del trabajo diario.
- Mejoramos en general la cantidad de información que puede ser consultada por el usuario y facilitamos su acceso.

Además, supone numerosas ventajas para los usuarios, de entre las cuáles destacamos:

- Se puede consultar toda la información de los animales en un mismo lugar.
- La identificación de los animales no queda restringida al identificador numérico, muchas veces complicado de recordar, pudiendo realizarla mediante el apodo que se le haya dado al animal, identificándolo visualmente por la imagen que se le haya añadido o usando la cámara para escanear el código que lleva adherido a su cuerpo para extraer su identificador total o parcialmente.
- Ganamos una enorme eficiencia al tener directamente acceso a la información el móvil que siempre acompaña a los usuarios ya no es necesario tomar notas en papel para procesarlas a un archivo *Excel* o *CSV* que posteriormente deba ser procesado por un programa al que hay que pasárselo por un terminal de línea de comandos.
- Mayor seguridad: las copias de la base de datos se realizan de manera periódica en los servidores de *Google* y el hecho de no tener distintas aplicaciones asegura la coherencia de los datos en todo momento.
- Flexibilidad: el hecho de que ahora los ganaderos también puedan dar de baja a los animales y registrar sus nacimientos simplifica enormemente la tarea de los técnicos veterinarios, pues en el pasado los ganaderos tomaban nota a mano, para posteriormente el técnico ir ganadería por ganadería recuperando los datos de sus notas para posteriormente añadirlas a un archivo CSV e importarlos.

2.2 Personal

En el ámbito personal hemos podido a prender a realizar un trabajo original, pasando por cada una de sus etapas, análisis, diseño y desarrollo. Dichas etapas nos han brindado oportunidades de poner en práctica muchos de los contenidos aprendidos durante el Máster, especialmente aplicándolo al desarrollo móvil, foco principal del título.

En el análisis recurrimos a herramientas conocidas como son los diagramas de casos de uso y sus especificaciones para identificar y validar los requisitos de usuario, que posteriormente serían validados con el cliente teniendo en cuenta el *feedback* que pudiera darnos al respecto, mejorando sustancialmente el resultado final.

En el diseño, hemos podido aplicar los distintos patrones de diseño y arquitecturas de software aprendidos durante el máster realizando así un diseño coherente y sencillo con el usuario final. Asimismo, el hecho de poder contrastar el diseño continuamente en cada iteración con el cliente fue una experiencia muy enriquecedora que nos ayudaba bastante a identificarnos con los usuarios finales y tener en cuenta ideas y detalles que de otra manera habrían pasado por alto.

Por último, en el apartado del desarrollo, ha sido una experiencia destacable cuanto menos. Trabajar con un equipo completo que puede aportar ideas y conocimiento a tu aplicación y profundizar en tecnologías conocidas, y algunas nuevas como *Node.JS* o *Firebase* no tiene precio. La cercanía y el compañerismo del equipo y la atención y disponibilidad del cliente nos ofrecieron sin duda lo mejor del proyecto y ha ayudado muchísimo a sobrellevarlo.

Como añadido, destacamos que el hecho de implementar dos aplicaciones completas e independientes que tienen que trabajar de manera cohesiva y fluida entre sí ha ayudado a identificar problemas en los requisitos y en la arquitectura del sistema que de haber sido de otra manera podría no haber sido identificados a tiempo o podrían haber implicado cambios sustanciales en la aplicación con la consiguiente repercusión negativa en el producto final.

3. Normativa y legislación

3.1 Licencias de Software

Una licencia de software es un contrato entre el autor del software y el usuario/consumidor del mismo en el que se estipulan los términos y condiciones bajo los que el creador del software autoriza su uso y explotación bajo un tipo de licencia determinado.
Bajo estas líneas se indican cuáles son las licencias software respectivas para cada uno de los productos Software que hemos utilizado durante nuestro desarrollo:

3.1.1 GNU GPL

[1] La Licencia Pública General de GNU (GNU GPL) *es la licencia de derecho de autor más ampliamente usada en el mundo del software libre y código abierto, y garantiza a los usuarios finales la libertad de usar, estudiar, compartir y modificar el software.*
Su propósito es doble: declarar que el software cubierto por esta licencia es libre, y protegerlo (mediante una práctica conocida como copyleft) de intentos de apropiación que restrinjan esas libertades a nuevos usuarios cada vez que la obra es distribuida, modificada o ampliada.
Esta licencia se ha utilizado para el producto software **[2] StarUML** en su versión 1.
De la versión 1 en adelante, este producto software ya no está sujeto a la GPL. Por lo tanto, su uso requiere la adquisición de una licencia y si su fin es lucrativo, de un acuerdo comercial, por lo que no afecta ni limita a nuestro proyecto en ninguna de sus condiciones de uso.

3.1.2 Licencia MIT

[3] La licencia MIT *es una Licencia de software libre permisiva lo que significa que impone muy pocas limitaciones en la reutilización y por tanto posee una excelente compatibilidad de licencia. La licencia MIT permite reutilizar software dentro de software propietario. Por otro lado, la licencia MIT es compatible con muchas licencias copyleft, como la GNU GPL (software con licencia MIT puede integrarse en software con licencia GPL, pero no al contrario). El texto de la licencia no tiene copyright, lo que permite su modificación.*
Esta licencia permite reutilizar el software así licenciado tanto para ser software libre como para ser software no libre, permitiendo no liberar los cambios realizados al programa original.
La licencia MIT es utilizada por los siguientes productos software usados en el desarrollo: Ionic, Angular, Angular Material y NodeJS. Y teniendo en cuenta sus restricciones tampoco afecta a nuestra implementación ni la limita en sus términos.

3.1.3 Licencia de uso de servicios de pago de Google Firebase

[4] La licencia de Google Firebase tiene un formato de pago por uso dependiendo de la cantidad de datos se consuman en subida y bajada y el número de conexiones simultáneas al servicio, pero no incluye restricciones que afecten directamente a nuestro proyecto.

3.1.4 Licencia de Bitbucket

[5] La licencia de Bitbucket tiene un formato gratuito y otro de pago, independientemente del escogido, dado que no incumplimos ninguno de sus acuerdos a la hora de comercializar el producto, no tendríamos problemas legales con esta licencia en este sentido.

3.1.5 Licencia de JetBrains

[6] La licencia de JetBrains educativa tiene un formato de suscripción anual y es gratuita para los miembros de la comunidad universitaria. Dado el fin comercial del proyecto y a las restricciones de este tipo de licencia incompatibles con el carácter de nuestro sistema, adquirimos una licencia mensual de *Jetbrains Webstorm* para el desarrollo del proyecto.

3.1.6 Licencia para educación de Office 365

[7] La licencia para educación de Office 365 destinada a la Universidad contiene restricciones que incluyen el uso no comercial de sus productos, pero dado que únicamente se ha utilizado dicha licencia en el producto software Microsoft Word 2019 para la redacción de esta memoria, no limita el alcance de nuestro proyecto.

3.1.7 Licencia para Adobe XD

El uso de la licencia de este programa es gratuito y no impone limitaciones comerciales para el uso de herramienta de prototipado que le hemos dado en nuestro proyecto.

3.1.8 Tipo de licencia escogida para el proyecto

Dadas las características y el origen de este proyecto por parte de una empresa cliente, podemos definir nuestro proyecto como un **[8] software propietario** al revisar su definición:
Se denomina con software propietario o privativo al software del cual no existe una forma libre de acceso a su código fuente, el cual solo se encuentra a disposición de su desarrollador y no se permite su libre modificación, adaptación o incluso lectura por parte de terceros.
La persona física o jurídica (compañía, corporación, fundación, etc.), al poseer los derechos de autor sobre un software, tiene la posibilidad de controlar y restringir los derechos del usuario sobre su programa, lo que en el software no libre implica por lo general que el usuario solo tendrá derecho a ejecutar el software bajo ciertas condiciones, comúnmente fijadas por el proveedor, que signifique la restricción de una o varias de las cuatro libertades.
Por lo que los términos de uso y de la licencia, quedarán pendientes de ser definidos por la empresa, entregándole a la misma todos los derechos del proyecto y su explotación.

3.2 Seguridad de los Datos

Primeramente, vamos a definir en base a que ley cumplimentamos los apartados legales del proyecto, siendo la **Ley Orgánica 3/2018 del 5 de diciembre de Protección de Datos Personales [9]**. Para saber cuál es la finalidad de dicha ley podemos consultar su definición citada bajo estas líneas:

1. ***La Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos Personales y garantía de los derechos digitales (LOPD-GDD) [10]*** *es una ley orgánica aprobada por las Cortes Generales de España que tiene por objeto adaptar el Derecho interno español al Reglamento General de Protección de Datos. Esta ley orgánica deroga a la anterior Ley Orgánica 15/1999 de Protección de Datos de Carácter Personal (aunque se mantiene vigente para la regulación de ciertas actividades).*

Así mismo se requiere para el correcto cumplimiento de esta ley, conviene revisar el **Reglamento (UE) 2016/679 [11]** que se define de la siguiente forma:

2. ***El Reglamento General de Protección de Datos (RGPD) [12]*** *es el reglamento europeo relativo a la protección de las personas físicas en lo que respecta al tratamiento de sus datos personales y a la libre circulación de estos datos. Entró en vigor el 25 de mayo de 2016 y fue de aplicación el 25 de mayo de 2018, dos años durante los cuales las empresas, las organizaciones, los organismos y las instituciones se fueron adaptando para su cumplimiento. Es una normativa a nivel de la Unión Europea, por lo que cualquier empresa de la unión, o aquellas*

empresas que tengan negocios en la Unión Europea, que manejen información personal de cualquier tipo, deberán acogerse a la misma.

Por las especificaciones de dichas leyes, debemos garantizar la protección de los datos personales que figuren en nuestra de base de datos, por lo que el acceso a la base de datos del sistema está protegido bajo contraseña y el inicio de sesión de los usuarios se realiza a través de una verificación de número de teléfono y una contraseña cifrada debidamente que es obtenida en la base de datos y comprobada una vez enviada la verificación. A continuación, citamos aquellos artículos que afectan al proyecto y definimos el motivo y su gestión:

3.2.1 Artículo 6. Tratamiento basado en el consentimiento del afectado

1. *De conformidad con lo dispuesto en el artículo 4.11 del Reglamento (UE) 2016/679, se entiende por consentimiento del afectado toda manifestación de voluntad libre, específica, informada e inequívoca por la que este acepta, ya sea mediante una declaración o una clara acción afirmativa, el tratamiento de datos personales que le conciernen.*
    - Dado que los usuarios facilitarán su nombre y número de teléfono, debemos especificar claramente en los términos de uso de la aplicación su finalidad, en nuestro caso de identificación e inicio de sesión respectivamente y requerir la autorización de los usuarios de manera explícita antes de incluirlos en el sistema y previa utilización del servicio.

3. *No podrá supeditarse la ejecución del contrato a que el afectado consienta el tratamiento de los datos personales para finalidades que no guarden relación con el mantenimiento, desarrollo o control de la relación contractual.*
    - El responsable de datos de la empresa cliente deberá cerciorarse de que el uso de los datos mencionados se limita únicamente a aquellas finalidades dispuestas en los términos de uso.

3.2.2 Artículos 11 - 18. Ejercicio de los derechos de las personas.

Para cumplir con dichos artículos y facilitar sus derechos a los usuarios, el responsable de los datos de la empresa cliente estará disponible para tramitar accesos, rectificaciones, supresiones, limitaciones, portabilidades u oposiciones de los datos en todo momento contactando a su responsable de datos.

3.2.3 Artículos 28 – 33. Responsable y encargado del tratamiento.

Para cumplir con estos artículos, deben existir las figuras de responsable y encargado de tratamiento en la empresa cliente, tal y como se especifica en los mismos en detalle. Incluyendo los datos contacto de las dos figuras, como se recaban los datos, la finalidad del tratamiento de los datos y el plazo que serán preservados.

4. Metodología de trabajo y planificación del proyecto

4. 1 Metodología de trabajo

Para realizar la metodología de trabajo hemos decidido basarnos en el **[13] modelo de ciclo de vida del software basado en prototipos,** ya que será la que mejor se adecue a nuestra forma de trabajo, siempre teniendo presente que el proyecto tiene dos aplicaciones independientes que posibilitará el poder construir nuestras aplicaciones de manera incremental e ir verificando en cada etapa que logramos los objetivos y teníamos una respuesta positiva de los requisitos considerados en el análisis junto a la ayuda de nuestro tutor.
Para transmitir mejor en que consiste este ciclo de vida, procederemos, en primer lugar, a definir que es un prototipo:

- ***[14] Los prototipos*** *son una representación limitada de un producto, el cual posibilita a ambas partes interesadas en el desarrollo (cliente y desarrollador) ir haciendo pruebas en situaciones reales o explorar su uso, creando una interacción directa en el proceso de diseño, lo cual generará una calidad mayor del producto final.*

Sobre esta definición vamos a incidir en la metodología de desarrollo basada en prototipos. Este tipo metodológico pertenece a los modelos que constan de un desarrollo evolutivo, por lo cual tiene lugar una primera implementación del proyecto, y se itera sobre este de manera constante hasta lograr el producto final. Son reconocidos por estar diseñados para ajustarse al cambio durante el desarrollo del proyecto, resultando ideal implementarlos en un trabajo original como este.

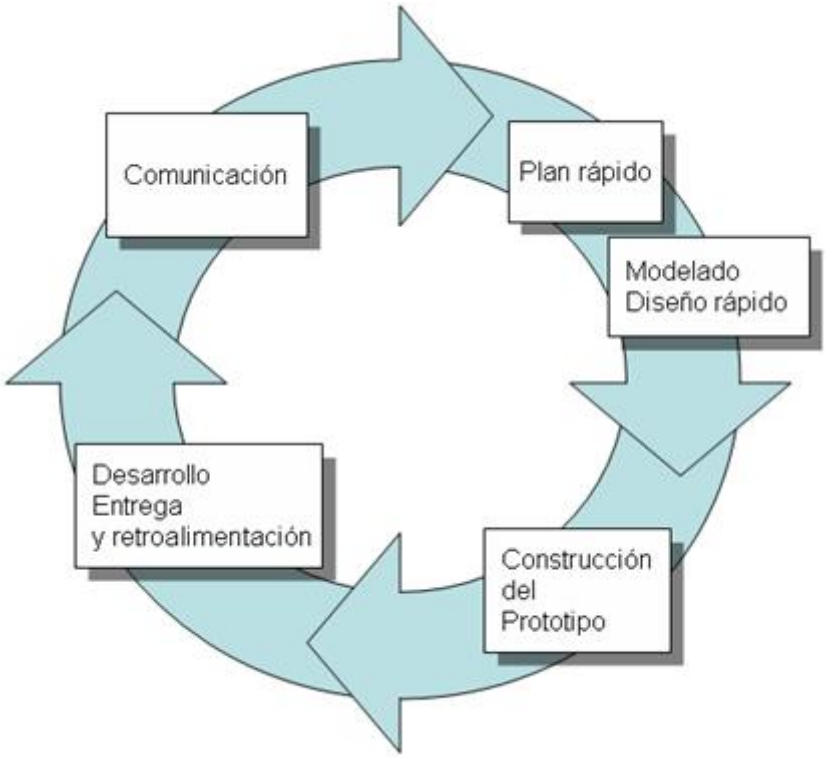

Ilustración 1: Modelo de construcción de prototipos

Como podemos observar en la *Ilustración 1*, este modelo sigue una estructura iterativa, cuyas etapas pasaremos a definir a continuación:

- Plan rápido, Modelado y Diseño rápido: esta etapa se centra en representar aquellos aspectos visibles para el cliente.
- Construcción del prototipo: se construye un prototipo, que podría ser funcional o no, en el que el cliente comprueba si dichas funcionalidades son las deseadas.
- Desarrollo, Entrega y retroalimentación: se lleva a cabo un proceso de desarrollo de las funcionalidades y se entregan al cliente, consiguiendo una retroalimentación de su parte, pudiendo comprobar acerca de si los requisitos han sido validados correctamente o no.
- Comunicación: El cliente transmite cambios que desea realizar en la implementación o posibles mejoras antes de cerrar la iteración.

Por dicho funcionamiento, el modelo basado en la construcción de prototipos disfruta de las siguientes **[15] ventajas e inconvenientes**:

Ventajas

- Será de gran utilidad cuando el cliente conoce los objetivos generales para el software, pero no puede identificar los requisitos detallados de entrada, procesamiento o salida.
    - En nuestro ejemplo concreto el cliente tiene una experiencia negativa con sus herramientas actuales y tiene un conocimiento medio de las herramientas tecnológicas disponibles y de como es el proceso de desarrollo de un software, jugando a nuestro favor en gran medida.
- Contará también con un mejor enfoque cuando el encargado del desarrollo del software está inseguro de la eficacia de un algoritmo, de la adaptabilidad de un sistema operativo e, incluso, de la forma que debería tomar la interacción humano-maquina.
    - Como partimos de una base poco relacionada con el desarrollo de aplicaciones móviles y, en general, todo desarrollo puede ser mejorable, resulta ideal poder validar cada implementación. Esto nos ayudará a ver que avanzamos en la dirección adecuada a la hora de validar los requisitos del producto, conforme se van realizando.
- El código es reutilizable.
    - Mientras hemos ido realizando prototipos, contamos con parte del código funcional que ha sido posible reutilizar, ahorrándonos horas de trabajo, así como tiempo de desarrollo y posibilitándonos cumplir con las entregas.

Inconvenientes

- El usuario puede ver modificadas sus expectativas mientras ve el prototipo, condicionando su opinión con respecto al sistema final. Debido a la creación de un prototipo de manera rápida, tienden a dejarse de lado aspectos importantes, como la calidad o el mantenimiento a largo plazo, obligando en la mayoría de los casos a reconstruir una vez se ha cumplido la función. Es frecuente una negación por parte del cliente a esa reconstrucción y que pida una modificación en el prototipo para obtener en él el sistema final, lo que llevaría a convertirlo en un prototipo evolutivo, pero partiendo de un estado poco recomendado.
    - En esta situación como hemos mencionado anteriormente, no tuvimos problemas con el cliente, siempre se mostró comprensible y entendía el proceso de elaboración del producto, alentándonos a mostrarles nuestras nuevas implementaciones siempre que pudiéramos y facilitando la comunicación en todo momento.

- Con la intención de desarrollar rápidamente el prototipo, el desarrollador tomará, en algunas ocasiones, decisiones de implementación poco convenientes (por ejemplo, usar un lenguaje de programación incorrecto, pero que nos otorgue un desarrollo más rápido). A medida que pase el tiempo, el desarrollador puede olvidar porqué tomó esa decisión inicial, con lo que puede provocar que dichas elecciones pasen a ser parte del prototipo final.
    - En nuestro caso, debido a la experiencia que hemos obtenido y las buenas prácticas aprendidas durante el máster, hemos podido evitar en mayor medida la mala toma de decisiones al realizar el prototipo y escoger las tecnologías para el desarrollo.

Conclusiones

La construcción de prototipos, aún teniendo en cuenta los problemas que surgirían en casos concretos, puede ser un paradigma efectivo para la ingeniería de software. El punto clave estará en fijar desde el principio las líneas en las que nos vamos a mover; es decir, el cliente y el desarrollador se deben poner de acuerdo en:

- Que el prototipo se construya y sirva como un mecanismo para la definición de requisitos.
- Que el prototipo se descarte, al menos en parte.
- Que después de este proceso, se desarrolle el software real con un mayor enfoque hacia la calidad.

Si conseguimos tener estas condiciones claras, siguiendo las indicaciones del cliente, al final esta metodología nos ha proporcionado muchas ventajas y muy pocas desventajas reales. Siempre debemos tener en cuenta la naturaleza de nuestro cliente, por lo que estos resultados no deben tomarse como algo para generalizar en todos los casos, ya que la metodología basada en prototipos aún contempla ciertos riesgos.

4.2 Planificación del proyecto

4.2.1 Planificación inicial

Para realizar una planificación correcta del proyecto, primeramente, realizamos un estudio del estado del arte, para poder identificar cuáles eran los usuarios principales de nuestra aplicación, así como que funcionalidades requerirían los mismos dentro de nuestro sistema. A partir de esta información pudimos identificar mejor cuáles serían los actores de nuestro sistema y que requisitos funcionales precisaba nuestro sistema. Para representar esta información y tenerla a nuestra disposición de manera clara y sencilla, construimos una serie de diagramas de casos de uso, con sus respectivas tablas de especificación, donde se detalla al completo la funcionalidad a implementar.

Una vez finalizadas las representaciones, se contrastaron con el tutor de empresa y con el cliente y se realizaron los cambios que estimamos oportunos para ajustarnos a una solución más correcta. Una vez bien definidas pasamos a realizar una primera representación gráfica de la interfaz de usuario de nuestra aplicación móvil sencilla con únicamente con el flujo de la aplicación usando Adobe XD como podemos ver en la imagen siguiente.

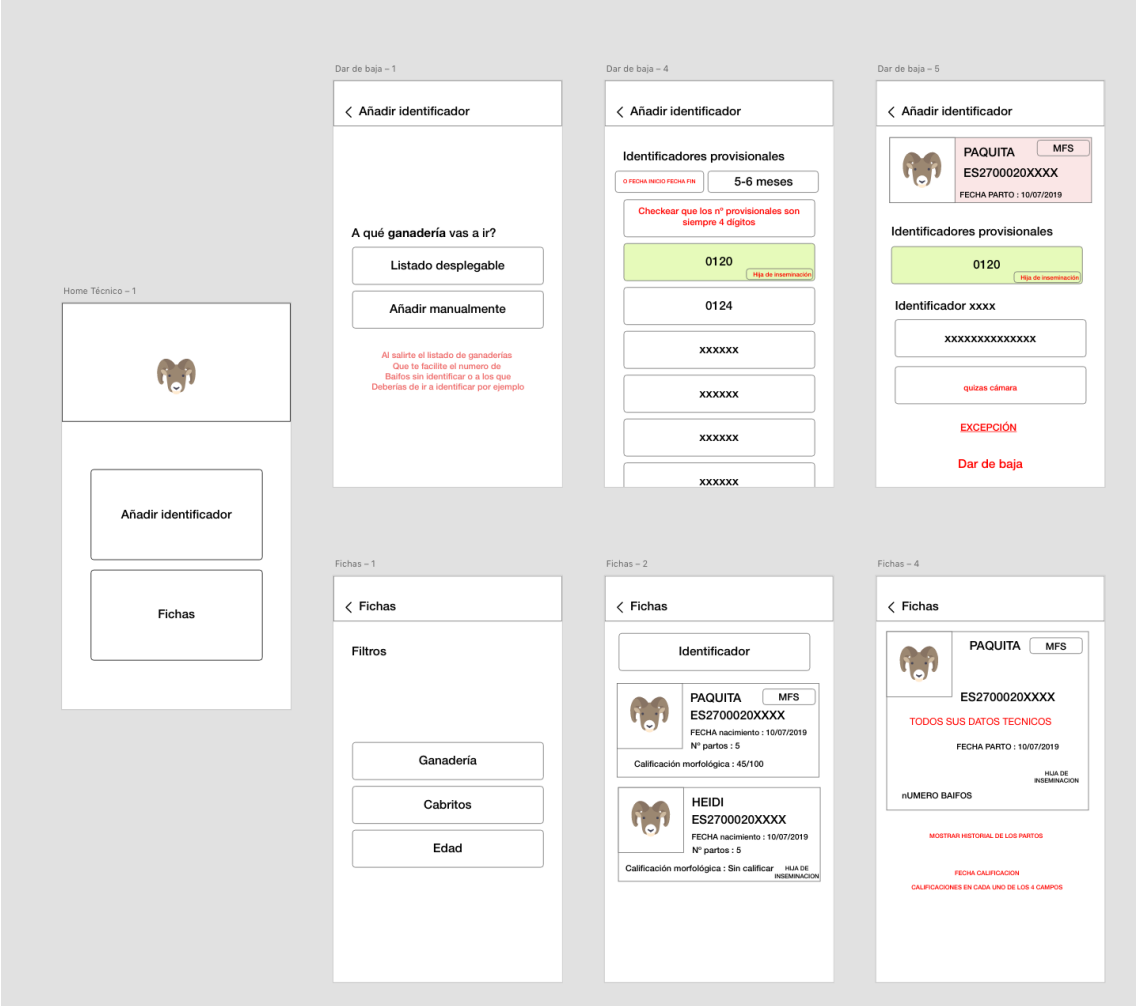

Ilustración 2: Wireframe diseñado en Adobe XD

Esta interfaz nos ayudó mucho a darnos cuenta de que debíamos especificar mejor las funcionalidades que habíamos definido, así como a identificar algunas que no nos habíamos planteado. También nos permitió validar con el cliente si el flujo de los procesos era el correcto y las funcionalidades se habían representado correctamente.

A continuación, debíamos describir bien la arquitectura del proyecto, realizando un análisis y estudio de las tecnologías disponibles para el desarrollo de este, ver sus ventajas e inconvenientes y por último realizar una elección de aquellas que concibiéramos que encajaban mejor con nuestra visión.

Finalmente nos decantamos por las tecnologías de *Ionic* para la aplicación móvil y *Angular* para la aplicación web de escritorio. Escogidas las mismas, comenzó un periodo de aprendizaje de las tecnologías y profundización de ellas, así como de otras tecnologías complementarias que formarían parte del proyecto tales como *Firebase* y *Node.js*.

Cuando consideramos que teníamos la destreza suficiente en estas tecnologías comenzamos el desarrollo primeramente de la aplicación móvil y la aplicación web. Siguiendo la estrategia del modelo de vida basado en prototipos, detallado en el apartado anterior, el proyecto fue dividido en una serie de pequeñas iteraciones y se fueron realizando conforme al ciclo de este mismo modelo. Las iteraciones consistieron en lo siguiente:

1º. Instalamos las herramientas necesarias para el desarrollo de ambas aplicaciones en el equipo y realizamos el menú principal de la aplicación móvil junto al inicio de sesión de la web.
2º. Desarrollamos las vistas de las distintas secciones de la aplicación tanto móvil como web. Se implementó la funcionalidad básica de transición entre las mismas.
3º. Definimos los modelos de nuestra base de datos, para posteriormente poder realizar la funcionalidad de visualización, búsqueda e importación de datos en el portal web.
4º. Construimos la funcionalidad básica del perfil de ganadero: Registro de Nacimientos, Bajas y Fichas. Se añadió además una primera iteración del inicio de sesión en la aplicación móvil.
5º. Implementamos la capacidad de dar de alta a usuarios desde el panel de administración usando su número de teléfono y se añadieron los roles a nuestro modelo de base de datos.
6º. Realizamos las funcionalidades correspondientes al perfil de técnico veterinario, tales como Identificar y Fichas. El último apartado de Calificar quedó pendiente porque faltaba la definición de este por parte del cliente.
7º. Realizamos las maquetaciones del panel web y se añadieron las funciones en la nube correspondientes para realizar el inicio de sesión o registro en el sistema según correspondiera.
8º. Comenzamos con el diseño final de la aplicación móvil, realizando un segundo prototipo con la herramienta Adobe XD añadiendo detalles de la marca, colores y una interfaz más cuidada.
9º. Desplegamos el apartado de Calificar en el rol de técnico con su correspondiente maquetación final. Se establece el inicio de sesión con verificación SMS a través del móvil para cerrar este proceso. Se realiza la publicación de las aplicaciones en las tiendas para la realización de pruebas.

4.2.2 Ajustes de la planificación

El proyecto original contemplaba otros dos apartados menos urgentes para la funcionalidad del técnico y se incluía además la posibilidad de gestionar todo el sistema desde el panel de administración, lo cuál complicaría arduamente el proyecto y quedaría fuera de los límites de tiempo posibles para la realización del trabajo de fin de título siendo funcionalidades no establecidas en el momento del contrato, lo que repercutiría negativamente en la fecha de entrega y perjudicaría a otros proyectos paralelos de la empresa.
Por ello se decidió apartar estas funcionalidades para la siguiente versión del proyecto en un futuro no muy lejano y previa definición de sus límites.

5. Análisis

En este apartado nos centraremos en comentar la etapa de Análisis de nuestro proyecto.

5.1 Estado actual, requisitos y objetivos

5.1.1 Estado actual y Alternativas existentes

Para conseguir que nuestra aplicación satisfaga las necesidades del cliente primeramente debemos realizar un estudio de la situación actual y de las soluciones que se por parte nuestro cliente para realizar su flujo de trabajo. De esta manera podremos identificar cuáles son los principales problemas y poder identificar cuáles serían las soluciones adecuadas para satisfacer sus requerimientos y garantizar la satisfacción del cliente.

Teniendo en cuenta esta premisa realizamos un estudio exhaustivo de las aplicaciones que utilizaba el cliente para la realización de las tareas que tenía que realizar nuestro proyecto, las cuales tuvimos la oportunidad de probar y ver como era el flujo de trabajo del cliente con las mismas, teniendo reuniones posteriormente para establecer que querían modificar, corregir, mejorar o funcionalidades que añadir al proyecto comparándolo con su solución previa. A continuación, enumeramos las dos herramientas que se utilizaban anteriormente y una breve descripción de estas:

- **[16] Geslib**:
  - Permite el registro de los animales y su búsqueda por identificador únicamente, así como ver sus datos correspondientes.
  - Permite la importación de los datos a través de un Excel y la exportación de estos en formato Excel.
  - No permite el registro del resto de características necesarias para el técnico: calificación, registro de nacimientos, consulta de historial de los animales.
  - Requiere una máquina virtual y su uso en escritorio dado que está diseñada para Windows 98.
  - Interfaz anticuada, con más de siete pasos para realizar una única tarea o navegar por los menús.
  - Sólo los técnicos veterinarios tienen acceso a ella.

- **[17] GMR**:
    - Permite el registro de los datos de un nacimiento y sus respectivos datos de lactación.
    - Sólo se permite el registro de estos datos.
    - Sólo existe versión de escritorio.
    - No se permite la exportación de los datos.
- **[18] Progecom**:

    - Permite el registro de la información respectiva a la calificación de los animales.
    - Sólo se permite el registro de estos datos.
    - No se pueden añadir más campos de los estrictamente indicados, ni notas de ningún tipo.
    - Desarrollada en *COBOL*, y únicamente disponible en MS-DOS por lo que requiere de una máquina virtual para su ejecución (Windows 98).
    - Sólo existe versión de escritorio.

Durante el inicio del proyecto intentamos tener acceso a estas herramientas, pero por términos de confidencialidad del cliente con el proveedor no pudimos tener acceso a las mismas. Sin embargo, nos explicaron el flujo de trabajo normal que tenían con dichas aplicaciones y como gestionaban los datos:

1. Cuando se produce un parto, en algunos casos, el ganadero toma los datos del nacimiento a papel y los guarda. Realiza esta operación durante un período de tiempo variable hasta que se realiza la visita por parte del técnico. Además, en este período también anota los datos de lactancia de los animales correspondientes. Esto es suponiendo el caso favorable de que el/la ganadero/a recoja los datos in situ.
2. El/La técnico/a acude a la explotación ganadera y recoge los datos manualmente, verifica el estado de los animales y cambia o actualiza valores que pudieran haberse introducido por error, o aquellos de los que el ganadero no se acordaba o no había tomado notas. Así mismo, el/la técnico/a añade un identificador temporal a los animales para facilitar su próxima visita.
3. El/La técnico/a traspasa la información de los animales a ordenador uno a uno usando el programa *GMR* en un ordenador de escritorio y los datos de lactación de las madres.
4. Pasado un período de tiempo variable, el/la técnico/a vuelve a realizar una visita a la explotación ganadera para recoger nueva información. Se toman los datos mencionados anteriormente si ha habido nuevos partos y se verifica si consulta y verifica también con los ganaderos si hay animales que ya estén en etapa adulta pendientes de una identificación definitiva.

Si existieran dichos animales, el/la técnico/a les asignará un identificador definitivo (normalmente consultan previamente si existen por teléfono, para solicitar los identificadores definitivos a la regulación del gobierno pertinente) y los calificará. Valorará sus diferentes aspectos: capa, apariencia general, carácter lechero, capacidad y sistema mamario. Dicha clasificación de los animales se realiza de manera completa y tomándose la información a mano.

5. El/La técnico/a arranca los programas de *Geslib* y *Progecom* en sus correspondientes máquinas virtuales para traspasar la información tomada en el campo. Traspasa la información de su registro e identificador final de uno en uno al programa *Geslib* para posteriormente en el programa de *Progecom* registrar sus datos de calificación. Una vez en *Progecom* estos datos se exportan y se modifican en un archivo *Excel* para la importación de una muy pequeña parte de la información en *Geslib*.

Así mismo, para evaluar la posibilidad de cuota de mercado que tendría nuestro proyecto, realizamos una investigación de alternativas similares existentes, destacando las siguientes con sus funcionalidades citando a sus respectivas páginas web:

- **[19] TrazaFarm:**
  - *Crea tu libro de registros veterinarios (medicamentos, vacunas, biocidas).*
  - *Crea el registro de movimiento de animales.*
  - *Crea un registro de piensos.*
  - *Crea el registro de bajas de la explotación.*
  - *Dispone de una agenda donde crear notas fácil y rápidamente.*
  - *Controla y alerta de los periodos de supresión en medicamentos.*

- **[20] VacApp**
  - *Gestiona tus rebaños y controla donde ha pastado cada vaca.*
  - *Descubre que vacas tienen más teneros a lo largo de los años y apuntate problemas en el parto.*
  - *Simplifica el papeleo a rellenar al hacer un saneamiento.*
  - *No pierdas datos, guárdalos en la nube y accede desde cualquier sitio.*
  - *VacApp también funciona sin conexión a internet. No siembre hay cobertura en el campo.*
  - *Puedes importar tu ganado leyendo códigos de barras de los DIBs. También puedes importar y exportar Excels.*

Durante esta parte de investigación también encontramos otras aplicaciones similares, pero o bien habían sido retiradas del mercado o bien estaban en producción y no estaban publicadas aún.

5.1.2 Requisitos y Objetivos

Habiendo visto el proceso actual de transferencia de información entre las distintas aplicaciones podemos identificar los siguientes problemas:

- No existe sincronización directa entre las aplicaciones.
- Deben utilizarse distintas versiones de sistemas operativos, haciendo uso además de máquinas virtuales con la complejidad técnica que ello conlleva para el usuario medio.
- La toma de datos por parte de los ganaderos no siempre es in situ, dando posibilidad a errores y olvidos.
- Existen muchos pasos en el proceso, lo que puede dar lugar a más errores, en el traspaso de información, en la copia de los datos del ganadero.
- La calificación de los animales tiene que realizarse de manera completa, de tal manera que los técnicos no pueden abandonar la explotación hasta terminar todas las calificaciones correspondientes con la inflexibilidad que ello conlleva.
- El traspaso de información a las aplicaciones tiene que realizarse de animal en animal, suponiendo un tedioso proceso repetitivo en el caso de que haya decenas o incluso cientos de animales de distintas explotaciones ganaderas en el momento de traspasar la información.
- Los técnicos tienen que gestionar toda esta información manualmente, separarla de alguna manera con las distintas explotaciones ganaderas y consultar previamente mediante llamadas de teléfonos (ceñidos a la disponibilidad de los ganaderos) para solicitar con antelación los números identificativos antes de su visita.

Y teniendo en cuenta las alternativas existentes en el mercado podríamos destacar como interesantes las siguientes funcionalidades derivadas de las suyas y aplicadas a nuestro cliente:

- Gestión de registros veterinarios.
- Registro de la alimentación de los animales.
- Registro de la ubicación de los animales.
- Control y alerta de medicamentos para los animales que lo requieran.
- Información sobre los problemas del parto.
- Estadísticas sobre los animales con más crías sanas y con mejores calificaciones, para preservar mejor la raza de la cabra palmera.
- Sincronización en la nube.
- Funcionamiento *offline*.
- Importación y exportación de los datos en archivos de hojas de cálculo.

Ahora podemos construir un listado de los requisitos ideales que podría tener una aplicación final en el mercado, en función de las necesidades del cliente y de las alternativas existentes. Dicho listado incluiría:

- Una única aplicación con todo el contenido centralizado.
- Diferentes versiones para las distintas plataformas, tanto móviles como de escritorio.
- Registro de los nacimientos de los animales y gestión de sus crías hasta la edad adulta, incluido su proceso de identificación final.
- Calificación de los animales bien parciales o completas.
- Visualización y consulta de la información de los animales, con la posibilidad de añadir sus nombres o fotografías para facilitar su búsqueda.
- Sistema de notificaciones para agilizar la comunicación de ganaderos y técnicos.
- Consulta de un historial de todas las operaciones realizadas por técnicos y ganaderos.
- Trazabilidad de los animales y sus transacciones haciendo uso de tecnologías **[21]Blockchain**.
- Gestión de registros veterinarios.
- Registro de la alimentación de los animales.
- Registro de la ubicación de los animales.
- Control y alerta de medicamentos para los animales que lo requieran.
- Información sobre los problemas del parto.
- Estadísticas sobre los animales con más crías sanas y con mejores calificaciones, para preservar mejor la raza de la cabra palmera.
- Sincronización en la nube.
- Funcionamiento *offline*.
- Importación y exportación de los datos en archivos de hojas de cálculo.
- Compatible con la gestión de distintas empresas ganaderas y diferentes animales.
- Escanear el microchip incorporado en los animales haciendo uso de algún periférico hardware compatible y conexión *Bluetooth* o *NFC*.
- Gestión completa de usuarios: registro, edición, bloqueos, bajas.

Pero dada la limitación temporal de nuestro trabajo de fin de máster y del cliente, hemos decidido focalizar nuestros requisitos que se pueden observar en la tabla a continuación:

Número	Nombre	Descripción
1	Registrar usuario	Se registra al usuario introduciendo su número de teléfono y la ganadería a la que pertenece.
2	Filtrar usuarios	Se muestra el listado de usuarios que coincide con el filtro introducido.
3	Importar usuarios	Se selecciona un archivo .CSV y se realiza el volcado de usuarios de dicho archivo en el sistema.
4	Filtrar animales	Se muestra el listado de animales que coincide con el filtro introducido.
5	Importar animales	Se selecciona un archivo .CSV y se realiza el volcado de animales de dicho archivo en el sistema.
6	Filtrar explotaciones	Se muestra el listado de explotaciones que coincide con el filtro introducido.
7	Importar explotaciones	Se selecciona un archivo .CSV y se realiza el volcado de explotaciones de dicho archivo en el sistema.
8	Filtrar nacimientos	Se muestra el listado de nacimientos que coincide con el filtro introducido.
9	Importar nacimientos	Se selecciona un archivo .CSV y se realiza el volcado de nacimientos de dicho archivo en el sistema.
10	Identificar animal	Se identifica a la cría de un animal como adulta proveyéndola de un número identificativo definitivo.
11	Transferir cría	Se transfiere a la cría a otra ganadería, bien interna o externa.
12	Dar cría de baja	Se da de baja a la cría en el sistema (fallecimiento, pérdida, malformaciones).
13	Ver ficha	Se visualiza la información de un animal específico.
14	Calificar animal	Se califica a un animal en sus diferentes apartados.
15	Transferir cría	Se registra el parto de un animal, indicando la madre, el padre (opcional), fecha, número de crías y sexo de las crías.
16	Añadir código provisional	Se añade el código provisional a una o a todas las crías del parto registrado.
17	Ver ficha	Se visualiza la información de un animal específico.
18	Añadir nombre	Se añade un nombre/apodo al animal seleccionado.
19	Añadir fotografía	Se añade una fotografía al animal seleccionado.
20	Dar animal de baja	Se da de baja al animal, quedando no disponible para su búsqueda manual, registros o calificaciones.

El resto de los requisitos serían deseables en un producto final después de varias iteraciones en el proyecto y se podrían plantear como trabajo futuro.

El principal objetivo de nuestra propuesta es ofrecer una solución sencilla, amigable con los dos tipos de usuario (ganadero y técnico), completa, multiplataforma y portable. Por ello hemos decidido marcarnos los siguientes objetivos:

- Ofrecer una interfaz de usuario sencilla, clara, intuitiva e independiente del sistema operativo o el dispositivo.
    - Para ello utilizaremos dos aplicaciones, una móvil y multiplataforma y una de escritorio
- Unificar las funcionalidades de las tres aplicaciones existentes en un único proyecto y que pueda trabajarse con todas ellas sin sincronizaciones manuales.
    - Usando una única base de datos para las distintas aplicaciones del sistema solucionamos la sincronización manual de los datos y aseguramos su persistencia y cohesión.
- Agilizar los flujos de trabajo y procesos que se realizan a diario en la aplicación.
    - El simple hecho de que la aplicación sea móvil permite reducir bastantes pasos de copia de la misma información en campo y a ordenador y viceversa, así como que los ganaderos hagan parte del trabajo una vez y no tenga que realizarlo tanto los ganaderos como los técnicos.
- Simplificar todos aquellos procesos que se puedan e implementar las nuevas funcionalidades que requiera el cliente.
    - Con nuestra experiencia en el desarrollo de aplicaciones móviles podemos realizar distintas propuestas al cliente sobre nuevas funcionalidades o alternativas a sus problemas que vayan surgiendo durante el proceso de desarrollo del proyecto.
- Asegurarnos de que tanto ganaderos como técnicos se sienten satisfechos con el producto final y mejora sustancialmente su trabajo.
    - La comunicación continua y validación de los requisitos y su implementación entre cliente, usuarios y los desarrolladores garantizará este aspecto con total seguridad.

5.2 Actores

Primeramente, vamos a señalar los actores que hemos identificado basándonos en los requisitos definidos, separando cada uno y explicando el papel que tienen en nuestro proyecto.

5.2.1 Administrador/a

El administrador del sistema es la persona responsable de la aplicación web de escritorio y tiene el control total del proyecto, principalmente, de gestión de datos. Destacando entre sus funcionalidades:

- Importación de datos automática de las siguientes entidades:
    - Animales
    - Explotaciones ganaderas y responsables
    - Nacimientos y datos de lactancia
- Visualización de las mismas entidades y filtrado de las mismas en tablas
- Dar de alta a usuarios usando su número de teléfono

5.2.2 Técnico/a Veterinario/a

Los técnicos de Cabra Palmera son los responsables de la gestión de los animales a nivel profesional. Sus funcionalidades en el sistema son:

- Identificar a las crías de los animales, pudiendo introducir su código identificativo final, darlas de baja o transferirlas directamente.
- Visualizar los datos de cualquier animal independientemente de la explotación ganadera a la que pertenezcan y acceder a sus datos de calificación si los hubiera registrado.
- Calificar a los animales en los distintos apartados descritos por el cliente.

5.2.3 Ganadero/a

Los ganaderos son los responsables directos de las explotaciones ganaderas que tengan asignadas y sus respectivos animales. Las funcionalidades a las que tiene acceso en el proyecto son:

- Registro de partos de los animales, identificando a la madre, el padre, número de crías, la fecha y sexo de estas, pudiendo además asignar un código temporal a los mismos para su facilitar su identificación hasta que lleguen a la edad adulta.

- Visualización de la información de los animales de sus explotaciones ganaderas y los datos que tengan disponibles, salvando sus calificaciones oficiales. Además, pueden añadir una foto a los animales y un apodo para facilitar su identificación en la aplicación.
- Baja del animal en el sistema, bien por fallecimiento, pérdida o transferencia a otra explotación ganadera, pudiendo esta última ser interna (comprendida en el sistema de la aplicación) o externa (a un particular u otra explotación ganadera de otra compañía).

5.3 Casos de uso

Para poder reflejar de manera visual y cómoda los requisitos funcionales del sistema separados por actores hemos decidido hacer uso del producto software *StarUML*. La razón de haber escogido este software es porque permite modelar la aplicación usando UML el cuál es el lenguaje de modelado más utilizado en la actualizad y está reconocido como estándar ISO, lo que permite que independientemente del idioma o del país, las ideas estarán plasmadas de la misma manera y serán reconocidas por cualquier otra persona que esté familiarizada con UML.

5.3.1 Diagramas de casos de uso

En este apartado, mostraremos los diagramas de casos de uso realizados para tener una representación sencilla y esquemática de los requisitos funcionales que fueron identificados en el transcurso del proyecto, separados en diferentes imágenes para facilitar su lectura en un documento con el formato de esta memoria.

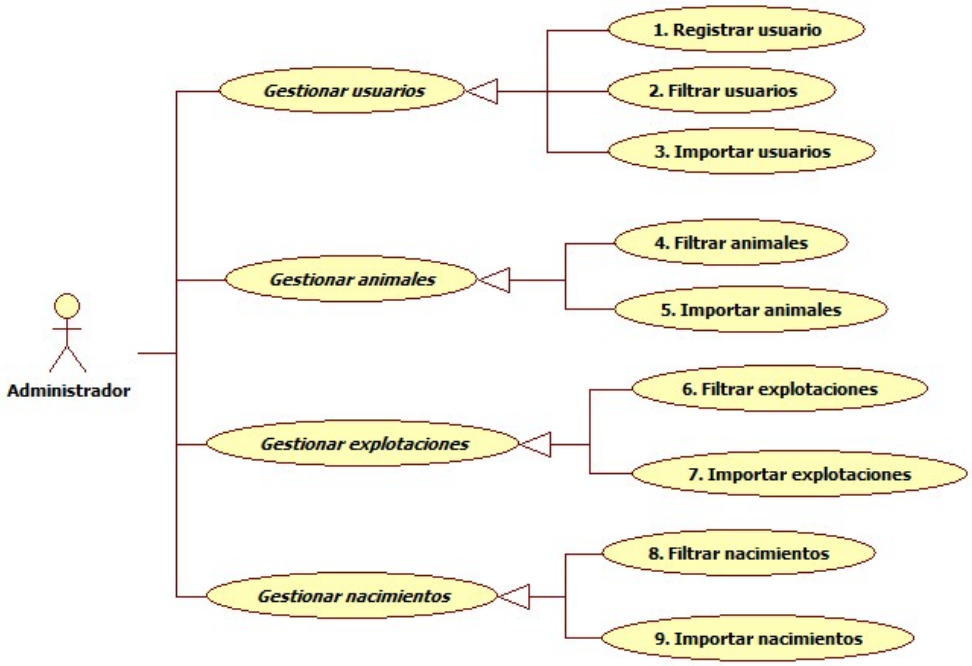

Ilustración 3: Diagrama de casos de uso (Administrador)

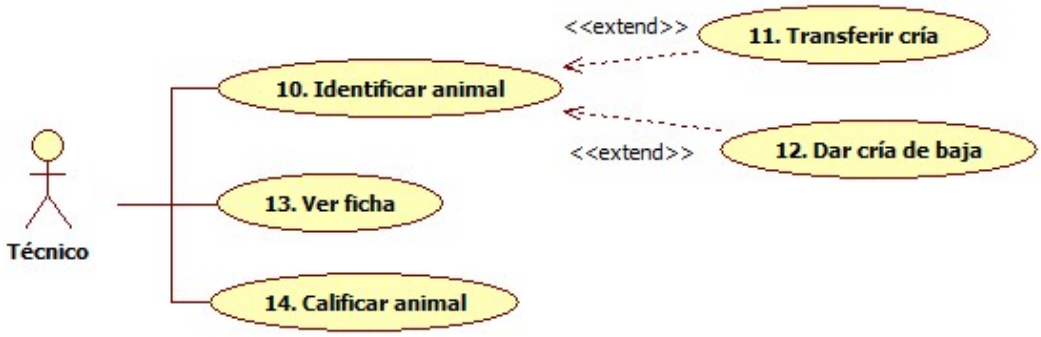

Ilustración 4: Diagrama de casos de uso (Técnico)

Ilustración 5: Diagrama de casos de uso (Ganadero)

5.3.2 Especificación de casos de uso

Para acabar esta sección adjuntamos una serie de casos de uso que consideramos relevantes en la aplicación, especificados siguiendo un modelo tabulado:

CASO DE USO	10	**IDENTIFICAR ANIMAL**	
Descripción	El técnico/a identifica a la cría de un animal como adulta proveyéndola de un número identificativo definitivo.		
Actores	Técnico/a.		
Precondiciones	El técnico/a debe haber iniciado sesión en el sistema.		
Flujo normal	**Paso**	**Acción**	
	1	Se accede a la opción “Identificar”.	
	2	Se selecciona la ganadería en la cuál se desea buscar la cría.	
	3	Se selecciona la cría a identificar.	
	4	Se introduce el código definitivo.	
	5	Se finaliza la operación.	
Postcondiciones	Se registra la cría como animal adulto y aparece en el sistema en la sección de animales.		
Variaciones	**Paso**	**Acción**	
	4 A	Se realiza una transferencia del animal, es decir se cambia su explotación ganadera propietaria a otra, interna si es del sistema y externa si no lo es, incorporando dicho movimiento del animal a su historial.	
	4 B	Se da de baja al animal, por lo que ya no aparecerá en las búsquedas de la aplicación salvando la sección de “Fichas”.	
Extensiones	**Paso**	**Condición**	**Caso de Uso**
Excepciones			
Observaciones			

Tabla 1: Especificaciones de casos de uso (Identificar)

CASO DE USO	15	REGISTRAR PARTO	
Descripción	El ganadero/a registra el parto de un animal, indicando la madre, el padre (opcional), fecha, número de crías y sexo de las crías.		
Actores	Ganadero/a.		
Precondiciones	El ganadero/a debe haber iniciado sesión en el sistema.		
Flujo normal	**Paso**	**Acción**	
	1	Se accede a la opción “Partos”.	
	2	Se identifica a la madre mediante identificador numérico.	
	3	Se identifica al padre con cualquiera de las opciones disponibles.	
	4	Se establece el número de crías, la fecha del parto y el sexo de estas.	
	5	Se completa la operación.	
Postcondiciones	Se registra el parto del animal y sus respectivas crías.		
Variaciones	**Paso**	**Acción**	
	1 A	Se identifica a la madre mediante su nombre.	
	1 B	Se identifica a la madre mediante el escaneo de su código en el animal.	
	3 A	Se salta la identificación del padre por desconocimiento de este.	
Extensiones	**Paso**	**Condición**	**Caso de Uso**
	4		16
Excepciones			
Observaciones			

Tabla 2: Especificaciones de casos de uso (Partos)

CASO DE USO	17	VER FICHA	
Descripción	El ganadero/a visualiza la información de un animal específico.		
Actores	Ganadero/a.		
Precondiciones	El ganadero/a debe haber iniciado sesión en el sistema.		
Flujo normal	**Paso**	**Acción**	
	1	Se accede a la opción “Fichas”.	
	2	Se selecciona el animal deseado, filtrando por identificador o nombre si deseara.	
	3	Se consultan los datos del animal.	
Postcondiciones			
Variaciones	**Paso**	**Acción**	
Extensiones	**Paso**	**Condición**	**Caso de Uso**
	3		18
	3		19
Excepciones			
Observaciones			

Tabla 3: Especificaciones de casos de uso (Fichas)

6. Diseño

En este apartado vamos a describir el diseño al que hemos llegado tras la fase de análisis, describiéndolo detenidamente.

6.1 Diseño de la arquitectura del sistema

[22] La arquitectura multi-nivel o por capas: *en dicha arquitectura a cada nivel se le confía una misión simple, lo que permite el diseño de arquitecturas escalables (que pueden ampliarse con facilidad en caso de que las necesidades aumenten).*
El más utilizado actualmente es el diseño en tres niveles (o en tres capas):

- *Capa de presentación: la que ve el usuario (también se la denomina "capa de usuario"), presenta el sistema al usuario, le comunica la información y captura la información del usuario en un mínimo de proceso (realiza un filtrado previo para comprobar que no hay errores de formato). También es conocida como interfaz gráfica y debe tener la característica de ser "amigable" (entendible y fácil de usar) para el usuario. Esta capa se comunica únicamente con la capa de negocio.*
- *Capa de negocio: es donde residen los programas que se ejecutan, se reciben las peticiones del usuario y se envían las respuestas tras el proceso. Se denomina capa de negocio (e incluso de lógica del negocio) porque es aquí donde se establecen todas las reglas que deben cumplirse. Esta capa se comunica con la capa de presentación, para recibir las solicitudes y presentar los resultados, y con la capa de datos, para solicitar al gestor de base de datos almacenar o recuperar datos de él. También se consideran aquí los programas de aplicación.*
- *Capa de datos: es donde residen los datos y es la encargada de acceder a los mismos. Está formada por uno o más gestores de bases de datos que realizan todo el almacenamiento de datos, reciben solicitudes de almacenamiento o recuperación de información desde la capa de negocio.*

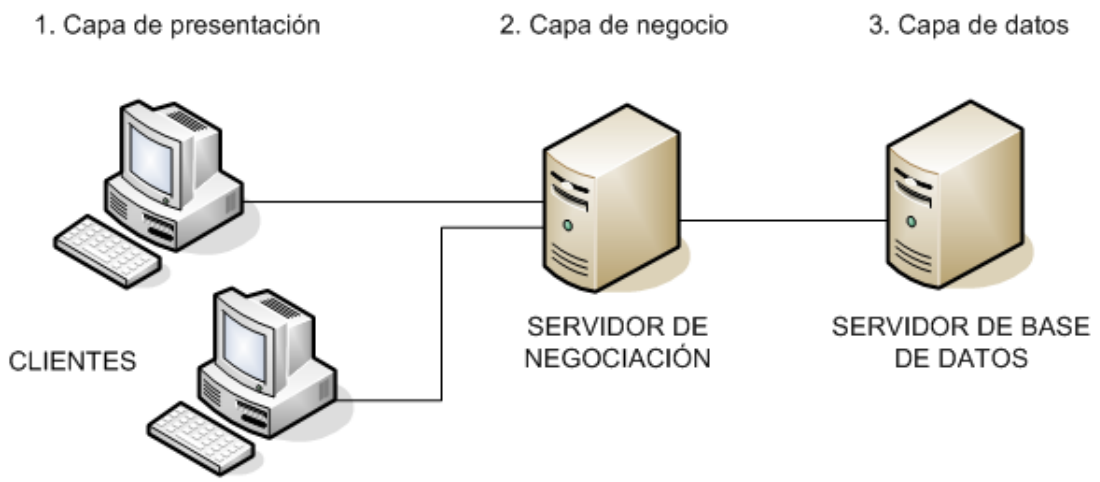

Ilustración 6: Instalación típica de una arquitectura multi-nivel

Sobre estas líneas, en la *Ilustración* 7, podemos apreciar que, la capa de presentación reside en el lado de los clientes, la de negocio suele incluirse en un servidor de negociación y por último la capa de datos en el servidor que alberga la base de datos.
Hemos decidido escoger esta arquitectura porque nos proporciona las **[23] siguientes ventajas:**

- *Centralización del control: los accesos, recursos e integridad de los datos son controlados por el servidor, de forma que un programa cliente defectuoso o no autorizado no pueda dañar el sistema. Esta centralización también facilita la tarea de actualizar dichos datos u otros recursos.*
- *Escalabilidad: se puede aumentar la capacidad de clientes y servidores por separado. Cualquier elemento puede ser aumentado (o mejorado) en cualquier momento.*
- *Fácil mantenimiento: al estar distribuidas las funciones y responsabilidades entre varios ordenadores independientes, es posible reemplazar, reparar, actualizar, o incluso trasladar un servidor, mientras que sus clientes no se verán afectados por ese cambio (o se afectarán mínimamente). Esta independencia de los cambios también se conoce como encapsulación.*
- *Simplifica la seguridad: no es necesario aplicar políticas de seguridad complejas en las aplicaciones del cliente, ya que la lógica de negocio y los datos están alojados en un único servidor. Además, Existen tecnologías, suficientemente desarrolladas, diseñadas para esta arquitectura que aseguran la seguridad en las transacciones, la amigabilidad de la interfaz, y la facilidad de uso.*

En nuestro caso particular, las aplicaciones del cliente sólo tienen la capa de presentación y una capa de red que permite conectarse al servidor, y el *Backend* de la aplicación web y aplicación móvil es el mismo y accedemos a su API conectándonos directamente a *Firebase*. Dicha arquitectura puede observarse en detalle en la *Ilustración 7* en la página siguiente.

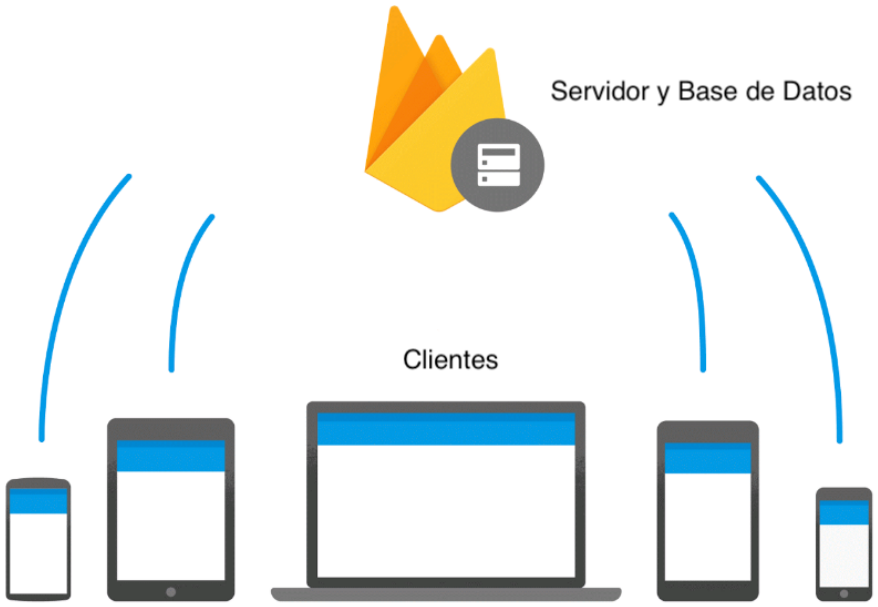

Ilustración 7: Diagrama de despliegue

6.1.1 Frontend

En el *Frontend* del sistema contamos con dos aplicaciones distintas:

- Aplicación móvil: en la cuál los actores principales son el Técnico/a y el Ganadero/a implementando las funcionalidades descritas previamente en el apartado anterior.
- Aplicación web: en esta, el actor principal es el administrador y su funcionalidad se reduce a la gestión de las distintas entidades del sistema como hemos indicado anteriormente.

El uso de la aplicación móvil se restringirá a un único dispositivo móvil que los usuarios tengan, dado el carácter multiplataforma de nuestra aplicación. Por otro lado, el uso de la aplicación web estará restringido un único usuario que la compañía cliente designe para la gestión del sistema.

6.1.2 Backend

Como hemos indicado anteriormente el *Backend* de nuestro proyecto está íntegramente implementado en *Firebase* con sus consiguientes reglas y funciones en la nube y se utiliza la misma API para la conexión desde ambas aplicaciones de *Frontend*, simplificando enormemente el proceso. Así mismo la base de datos es compartida entre ambas aplicaciones y está alojada en los servidores de *Google* empresa ala cuál se paga por dicho servicio.

6.2 Diseño de la Base de Datos

6.2.1 Contenido de la Base de Datos

Como indicamos previamente en la memoria, la base de datos establecida en *Firebase* es de tipo *No-SQL*. El término No-SQL se define como una amplia clase de sistemas de gestión de bases de datos que difieren del modelo clásico de SGBDR (Sistema de Gestión de Bases de Datos Relacionales) en aspectos importantes, siendo el más destacado que no usan SQL como lenguaje principal de consultas. Los datos almacenados no requieren estructuras fijas como tablas, normalmente no soportan operaciones JOIN, ni garantizan completamente ACID (atomicidad, consistencia, aislamiento y durabilidad) y habitualmente escalan bien horizontalmente.
Dada esta característica, no nos es posible representar nuestra base de datos de manera tradicional, pero podemos hacernos una idea de su estructura con los modelos de las entidades, bajo estas líneas, que se guardan en la base de datos.

```
export class User {
    id: string;
    name: string;
    phoneNumber: string;
    ranches: Ranch[] | {
        [x: string]: {
            id: string,
            REGA: string
        }
    };
    role: Roles;
    REGA: string;
    details: any;
}
```

Ilustración 8: Modelos (Usuario)

Podemos observar que en el modelo de usuario se almacenan sus detalles personales, sus ganaderías asociadas y su rol.

```
export class Ranch {
    id: string;
    name: string;
    province: string;
    status: string;
    REGA: string;
}
```

Ilustración 9: Modelos (Ganadería)

En el caso de las ganaderías se almacena su REGA y sus datos personales, así como el estado de actividad de esta en el mercado nacional (de alta o de baja).

```
export class Goat {
    id: string;
    name?: string;
    CEA: string;
    photo?: string;
    crotal: string;
    birthDate: Date;
    registerDate: Date;
    purity: string;
    fatherId: string;
    motherId: string;
    value: number;
    qualificationDate: Date;
    breederRanch: string;
    propietaryRanch: string;
    gender: string;
    isDropped: boolean;
    dropReason: string;
    dropDate: Date;
    numberOfBirths?: number;
    births?: GoatBirth[];
    latestBirthDate?: Date;
    isMFS?: boolean;
    fromInsemination?: boolean;
    history?: Transference[];
    features: Features;
    adnCardena: string;
}
```

Ilustración 10: Modelos (Animal)

En los animales nos encontramos muchísimos campos que son extraídos de los archivos .CSV durante la importación del panel de administración y otros muchos otros del nuestro proyecto en sí. Destacando la fecha de calificación, el motivo de baja, la fecha de baja, el número de partos del animal, un vector que incluye referencias a los nacimientos, la última fecha de parto, si está categorizado con una valoración excelente (*isMFS*), el historial de transferencias del animal y dos campos más que se utilizarán en versiones futuras del proyecto que sirven para indicar si tiene como origen la inseminación artificial y un identificador de ADN.

```
export class GoatBirth {
    id: string;
    date: Date;
    numberOfBabies: number;
    lactationDuration?: number;
    milkKilograms?: number;
    fatRate?: number;
    proteinRate?: number;
    dailyProduction?: number;
}
```

Ilustración 11: Modelos (Nacimiento de animal)

El nacimiento del animal es una versión reducida del modelo completo del nacimiento, pero es aquí donde se incluyen los datos de lactación y crianza. Estos datos son extraídos durante la importación del panel administrativo, y pueden consultarse en la aplicación. Se prevé que en el futuro se puedan añadir también desde la aplicación.

```
export class Transference {
    id: string;
    date: Date;
    originId: string;
    originName: string;
    originREGA: string;
    destinationId: string;
    destinationName: string;
    destinationREGA: string;
}
```

Ilustración 12: Modelos (Transferencia)

Las transferencias recogen la información de que ganadería es la originaria de la transacción y cuál es la de destino para cuando los animales sean trasladados o vendidos.

```
export class Birth {
    id: string;
    date: Date;
    motherId: string;
    fatherId?: string;
    numberOfBabies: number;
    babiesIds?: string[];
    lactationDuration?: number;
    milkKilograms?: number;
    fatRate?: number;
    proteinRate?: number;
    dailyProduction?: number;
}
```

Ilustración 13: Modelos (Nacimiento)

También tenemos los nacimientos, muy similar a los nacimientos de animal, pero con un poco más de información como un vector que contiene los identificadores de las crías o quién es la madre/padre del parto.

```
export class BabyGoat {
    id: string;
    temporalId: string;
    gender: string;
    isDropped: boolean;
    isFromArtificialInsemination: boolean;
    birthDate: Date;
    propietaryRanch: string;
    birthId: string;
    goatId: string;
    nameGoat?: string;
    history?: Transference[];
    mother: Goat
}
```

Ilustración 14: Modelos (Crías)

En el caso de las crías se indica quién es su madre, su ganadería propietaria y su historial por si fuera transferida antes de alcanzar la edad adulta, además de detalles sobre el animal.

```
export class Features {
    baseCoating?: string;
    coat?: any;
    appearance?: any;
    appearanceValue?: number;
    milkProfile?: any;
    milkProfileValue?: number;
    capacity?: any;
    capacityValue?: number;
    mammalSystem?: any;
    mammalSystemValue?: number;
    totalValue?: number;
    observations?: string;
}
```

Ilustración 15: Modelos (Características)

Por último, las características, incluidas como un atributo de los animales, donde destacan las puntuaciones de cada sección y la puntuación total, así como un campo distinto para volcar los datos recogidos en cada sección, apariencia, carácter lechero, capacidad y sistema mamario.

6.2.2 Almacenamiento de imágenes

Para el almacenamiento de imágenes de los animales y las fotografías para la identificación de código usando *Google Vision* hemos decidido usar **[24] Cloudinary** como servicio de almacenamiento de imágenes principalmente por su facilidad para trabajar con *Cloud Functions* de *Firebase* y para evitar almacenar directamente en nuestra base de datos, de por sí bastante extensa, numerosos archivos de manera completa.

6.3 Diseño Arquitectónico

Los *frameworks Ionic* y *Angular* están basados en el patrón arquitectónico de **[25] Desarrollo Basado en Componentes (CBD).**

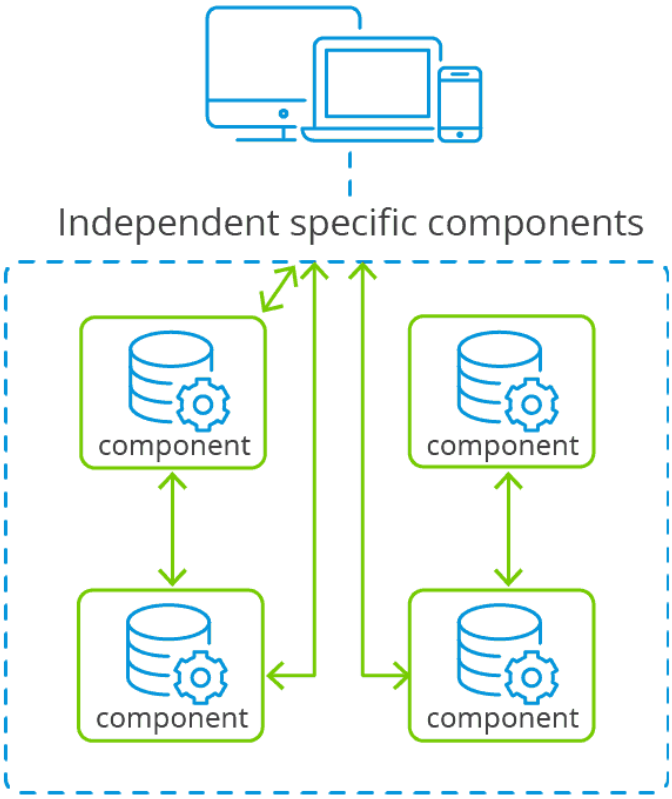

Ilustración 16: Diagrama del Desarrollo Basado en Componentes

Este patrón arquitectónico enfatiza la separación de asuntos, separation of concerns (SoC), por lo que se refiere a la funcionalidad de amplio rango disponible a través de un sistema de software dado. Es un acercamiento basado en la reutilización para definir, implementar, y componer componente de software débilmente acoplados en sistemas. Esta práctica persigue un amplio grado de beneficios tanto en el corto como el largo plazo, para el software en sí mismo y para las organizaciones que patrocinan tal software.

Por este motivo se adapta perfectamente a nuestro proyecto en el que se reutilizará mucho código en la parte de aplicación móvil y la de escritorio, pudiendo desarrollar componentes específicos para ambas plataformas y facilitar su mantenimiento y reusabilidad.

6.4 Diseño de la interfaz de usuario

La interfaz de usuario es el puente que une al usuario y el software, y su importancia es mayor de la que solemos pensar. El buen diseño en las mismas garantiza de sobremanera cierta parte del éxito de una aplicación. Hay muchas aplicaciones excelentes en cuanto a funcionalidades, sin embargo, debido a lo complejo o dificultoso que se vuelve su uso debido a la interfaz, el usuario se siente rechazado por el software y termina por abandonarlo, convirtiéndose en un producto no satisfactorio.
No obstante, existen en el mercado muchísimas aplicaciones que, sin contar las mejores implementaciones atrapan a millones de usuarios porque les hacen sentir cómodos, que las cosas se realizan sin dificultad, no tienen que buscar entre un millón de opciones y la información se muestra de manera clara y precisa, sin excesos ni adornos innecesarios. Por ello hemos realizado una interfaz de usuario móvil que cumpla con los principios de diseño definidos por **[26] Jeff Johnson** en el libro ***GUI Bloopers 2.0*.**

Principio 1

- *Considerar como más importante el usuario y las tareas que debe realizar anteponiéndose a la tecnología.*
- *Comprender al usuario, sus tareas y el entorno de software en las que se desarrolla.*

Para cumplir este principio, en nuestra aplicación hemos tenido en cuenta el tipo de público al que va dirigido, esto que implica muchos de los ganaderos/as no están familiarizados con las últimas interfaces de usuario y sólo les interesa poder ver el contenido de manera sencilla y clara. Por ello se han evitado menús deslizantes, elementos dinámicos, pantallas flotantes y gestos multitáctiles con el fin de evitar acciones erráticas y mantener una interfaz clara.

Principio 2

- *El primer paso en el diseño de una interfaz es desarrollar un modelo conceptual de la interacción y las tareas. Los aspectos de presentación deben abordarse posteriormente.*

Para este principio, previo al diseño final de la interfaz de usuario y a la capa de presentación realizamos ciertos esbozos de cómo sería la interfaz de usuario de los ganaderos y los técnicos, así como prototipos sencillos haciendo uso del producto software Adobe XD.

Principio 3

- *Centrarse en la visión que tiene el usuario respecto a la tarea.*
- *Esforzarse por la naturalidad, ideas y vocabulario de él.*
- *Las cuestiones internas del programa (de la tecnología en general) deben permanecer internas.*
- *Encontrar un equilibrio entre prestaciones y complejidad.*

Para conseguir cumplimentar con estos puntos, pedimos segundas opiniones al resto de compañeros y a algunas técnicas voluntarias sobre qué interfaz de usuario les parecía más clara o más concisa, y comprobar si nuestro diseño casaba con la idea que ellas tenían de cómo debían realizarse ciertas acciones. Siempre hemos intentado evitar mostrar cualquier tipo de información de los procesos internos de la aplicación y que el usuario disponga de funcionalidades extra que puedan resultarle útiles, siempre en pos de realizar de manera más sencilla las tareas principales, sin aumentar la complejidad de los procesos.

Principio 4

- *Diseña pensando en las cosas más comunes y frecuentes, no en las extraordinarias.*
- *Haz que los casos comunes y frecuentes sean muy fáciles de realizar.*

Debida a la naturaleza de la aplicación, hemos centrado nuestros esfuerzos en que las acciones principales no requieran muchos cambios de pantalla para realizarse, ver el contenido de los mensajes no debería estar más allá de tres vistas. De esta manera y muchas otras, hemos intentado simplificar las tareas más comunes de la aplicación, haciendo que la navegación sea rápida y eficiente.

Principio 5

- *No le compliques la vida al usuario.*
- *No le obligues a razonar en demasía.*

Se ha conseguido no mostrar pantallas innecesarias al usuario y no mostrar mensajes de error indescifrables, si ocurre algún error durante la ejecución de alguna funcionalidad, especifica claramente el porqué ha ocurrido y que debe hacer el usuario para ponerle solución.

Principio 6

- *Facilita el aprendizaje.*
- *Consistencia: significado de los controles deducido a partir de ellos mismos o pistas muy fáciles, nunca valiéndonos de nuestra memoria (poner el conocimiento de uso de las coas en el mundo no en nuestro cerebro).*
- *Evita las consecuencias de los errores, el usuario se atreverá a explorar cosas nuevas, ya que no suponen riesgo.*

Hemos propuesto una interfaz de usuario homogénea y muy similar en los distintos procesos, de tal manera de que el usuario siempre esté familiarizado sobre su avance en el mismo, a pesar de que sea la primera vez que realiza un proceso dada su similitud visual con los demás. Además, ninguno de los errores que se muestran en la aplicación conlleva un efecto negativo en el usuario y en su experiencia.

Principio 7

- *Proporciona información al usuario, no simplemente soltarle datos.*
- *"La pantalla le pertenece al usuario" es una forma de no alterar ese espacio en el que él se mueve. Respeta su territorio y no cambies las cosas bruscamente (inercia de la pantalla).*

Cuando el usuario realiza alguna actividad que debería de tener una retroalimentación, se ha implementado que tenga algún tipo de respuesta gráfica. Por ejemplo, cuando el usuario da de baja a un animal desde la parte de nacimientos, se le muestra al usuario una ventana de confirmación, y si se confirma la eliminación este desaparece de la pantalla.

Principio 8

- *Diseña para la interactividad entre la aplicación y el usuario.*
- *Respetar el tiempo del usuario.*

El tiempo del usuario es vital, por ello, se ha diseñado la aplicación para que la navegación por la misma sea rápida, con el menor tiempo de espera y mostrando el menor número de pantallas posible sin repercutir en la sencillez de uso.

Principio 9

- *Prueba los diseños con usuarios reales, es decir: Test y Verificación.*
- *Corrige los fallos.*

Además de las pruebas de la aplicación realizadas con el Tutor de Empresa para validar los requisitos funcionales, se hicieron pruebas con usuarios potenciales para verificar si el diseño cumplía con su cometido y si todas las funcionalidades se ejecutaban correctamente. Gracias a esto, se descubrieron algunos errores que no se habían identificado durante las pruebas del desarrollo y se corrigieron debidamente.

7. Desarrollo

7.1 Tecnologías y herramientas utilizadas

En este apartado detallaremos cuáles fueron las herramientas y tecnologías escogidas para el desarrollo de la aplicación dividiéndose en tres categorías: Tecnologías, Herramientas y Servidores.

7.1.1 Tecnologías

- **[27] Ionic:** es un *framework* gratuito y de código abierto para desarrollar aplicaciones híbridas multiplataforma que utiliza las tecnologías de Cordova, HTML5 y CSS como base y hace uso de Angular para gestionar las aplicaciones. Hemos decidido usar la versión 3 del *framework* pese a ya haber sido lanzada la 4 principalmente a que esta última a penas tiene documentación de terceros y era muy reciente en el momento de la elección de las tecnologías, habiendo salido de un estado de pruebas de desarrollo hacía solo unos pocos meses. También incorpora componentes de interfaz de usuario sencillos y personalizables, y el resultado final es muy similar a una aplicación nativa, salvando el hecho de que la aplicación sólo se desarrolla una vez para distintas plataformas.

- **[28] Cordova:** es un entorno de desarrollo de aplicaciones móviles que permite a los programadores construir aplicaciones para dispositivos móviles haciendo uso de CSS3, HTML5 y Javascript en lugar de lidiar con las APIs específicas de cada plataforma, así como encapsular CSS, HTML y código Javascript dependiendo de la plataforma del dispositivo.

- **[29] Angular:** es un *framework* de JavaScript, escrito en *TypeScript* y de código abierto mantenido por Google, que se utiliza para crear aplicaciones web *SPA*. Permite crear aplicaciones web siguiendo el patrón de Modelo-Vista-Controlador de manera mucho más sencilla.

- **[30] JavaScript:** *es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos,3 basado en prototipos, imperativo, débilmente tipado y dinámico.*

- **TypeScript:** transpilador de JavaScript fuertemente tipado con con herramientas muy útiles para asegurar la escalabilidad de las aplicaciones.

- **SPA:** o *Single Page Application* es una aplicación web que cabe en una sola página, cargando todo el contenido de la misma una sola vez y que permite la

recarga dinámica e interactiva de las distintas partes que vaya necesitando el cliente, evitando las cargas manuales y la apertura de nuevas páginas para acceder a diferentes secciones.

- **Ajax:** es una técnica de desarrollo web para crear aplicaciones interactivas. Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre las páginas sin necesidad de recargarlas, mejorando la interactividad, velocidad y usabilidad en las aplicaciones.

- **HTML5:** es un lenguaje de marcas utilizado principalmente para presentar y estructurar contenido web. Es la versión más extendida del estándar e incluye numerosas novedades como la inserción de multimedia, la reducción de las dependencias de los complementos y el acceso offline.

- **SASS [31]:** *es un metalenguaje de Hojas de Estilo en Cascada (CSS). Es un lenguaje de script que es traducido a CSS. SassScript es el lenguaje de script en sí mismo. Sass consiste en dos sintaxis. La sintaxis original, llamada indented syntax («sintaxis con sangrado») que usa una sintaxis similar al Haml.3 Este usa la indentación para separar bloques de código y el carácter de nueva línea para separar reglas. La sintaxis más reciente, SCSS, usa el formato de bloques como CSS. Este usa llaves para denotar bloques de código y punto y coma (;) para separar las líneas dentro de un bloque. La sintaxis indentada y los ficheros SCSS tienen las extensiones. sass y .scss respectivamente.*
  Se utilizó para el diseño visual tanto de las aplicaciones móviles como de la versión en escritorio, simplificando enormemente la escritura del código CSS gracias a la herencia de estilos, la posibilidad de definir herencia de clases CSS y el uso de variables para utilizar el mismo valor de un atributo en distintas partes de la aplicación.

- **[32] CSS3:** *es un lenguaje de diseño gráfico para definir y crear la presentación de un documento estructurado escrito en un lenguaje de marcado.2 Es muy usado para establecer el diseño visual de los documentos web, e interfaces de usuario escritas en HTML o XHTML.*

- **[33] Git:** es un software de control de versiones diseñado por Linus Torvalds, pensando en la eficiencia y la confiabilidad del mantenimiento de versiones de aplicaciones cuando éstas tienen un gran número de archivos de código fuente.

- **Angular Material:** librería visual de desarrollo para Angular desarrollada por *Google* con una excelente documentación que implementa la guía de estilos de *Material Design*.

- **Firebase:** es un servicio de Google que permite alojar nuestra base de datos con sin gestionar la infraestructura de esta, la cuál se adapta dependiendo de nuestras necesidades. También nos provee de muchísima funcionalidad ya implementada como gestión de la base de datos, autentificación, *Web* Hosting seguro, almacenamiento multimedia y de archivos, mensajería, estadísticas, funciones en la nube y registro y trazabilidad de errores.

  Nos interesa principalmente por sus servicios de:

  - Autentificación (ya que implementan un servicio de autentificación por teléfono de manera segura y con verificación SMS muy fácil de usar)
  - El uso de las *Cloud Functions* que nos permite tener funciones que están pendientes de cambios en nuestra base de datos para disparar ciertas operaciones, como por ejemplo la subida de una fotografía a la API de *Cloudinary* cuando un ganadero envía una imagen al sistema, siendo esta procesada en la función, enviada a la API de *Cloudinary* y posteriormente insertando la dirección obtenida de la API para incorporarla en la entidad correspondiente de nuestra base de datos.
  - Disponibilidad offline, ya que el uso de *Firebase* permite que al no tener conexión a internet en nuestro móvil (por ejemplo, en el caso de trabajo de campo de ganaderos y técnicos en ciertas zonas sin cobertura) los cambios se guarden en local y se sincronicen con la base de datos en cuanto volvamos a tener conexión.

- **Node.js**: *framework* de desarrollo *backend* de *Javascript* para la implementación de las funciones en la nube de *Firebase* mencionadas anteriormente.

- **AngularFire**: la librería oficial de *Firebase* para *Angular* que nos permite hacer uso de *Firebase* desde *Angular* de la manera más sencilla, teniendo un sistema basado en observables, autentificación, uso de datos offline, notificaciones push, etc.

- **RxJS:** es una librería de programación reactiva de *Javascript* que utiliza el concepto de observables para realizar llamadas asíncronas de manera mucho más sencillas, especialmente útil para manejar la funcionalidad de *AngularFire*.

- **Cloudinary:** servicio web de almacenamiento de imágenes con herramientas muy útiles disponibles en su *API* para la manipulación de estas.

- **Cloud Vision:** analizador de inteligencia artificial de *Google* que permite realizar identificaciones de imágenes, textos y otros conceptos a partir de una simple imagen. Extremadamente útil para la identificación del código numérico de un animal al enviarle una fotografía a través de su API.

7.1.2 Herramientas

- **[34] JetBrains Webstorm:** *JetBrains Webstorm* es un IDE profesional de JavaScript que es compatible con una amplia gama de tecnologías modernas relacionadas con los lenguajes de programación *JavaScript*, *HTML* y *CCS* especialmente útil para la refactorización de código y con soporte nativo de *Angular* e *Ionic*.

- **[35] Google Chrome:** es un navegador web desarrollado por Google y compilado con base en varios componentes e infraestructuras de desarrollo de aplicaciones de código abierto, como el motor de renderizado Blink. Este explorador cuenta con herramientas para desarrolladores web muy útiles que permiten depurar e incluso realizar cambios en el código en tiempo real.

- **[36] Adobe XD:** una herramienta para diseñadores web y de aplicaciones móviles que permite facilitar el diseño de la experiencia de usuario (UX), aunando las funciones de programas dedicados al desarrollo de aplicaciones web y prototipos en un solo programa.
  Adobe XD ofrece todas las herramientas para la realización de prototipos de páginas web y aplicaciones, permitiendo crear fácilmente prototipos animados, previsualizarlos y compartirlos con otros miembros del equipo. Finalizados o en proceso, los diseños se pueden compartir de manera sencilla a través de enlaces públicos que pueden abrirse incluso desde el móvil, brindando así la posibilidad de experimentar el diseño también en este dispositivo.

- **[37] Bitbucket:** *Bitbucket* es un servicio de alojamiento basado en web, para los proyectos que utilizan el sistema de control de versiones Mercurial y Git.

7.2 Estructura de la aplicación

La estructura de ambos proyectos es muy similar dado que *Ionic* trabaja usando la misma estructura que *Angular* salvando algunas pequeñas diferencias (los *Services* se denominan *Providers* y se genera un componente y módulo de tipo *Page* para cada nueva vista) por lo que detallaremos la estructura general de un proyecto de este tipo y las carpetas que lo componen. Cabe destacar que hemos seguido la guía de directivas propia de Angular que facilita la escalabilidad de la aplicación enormemente.

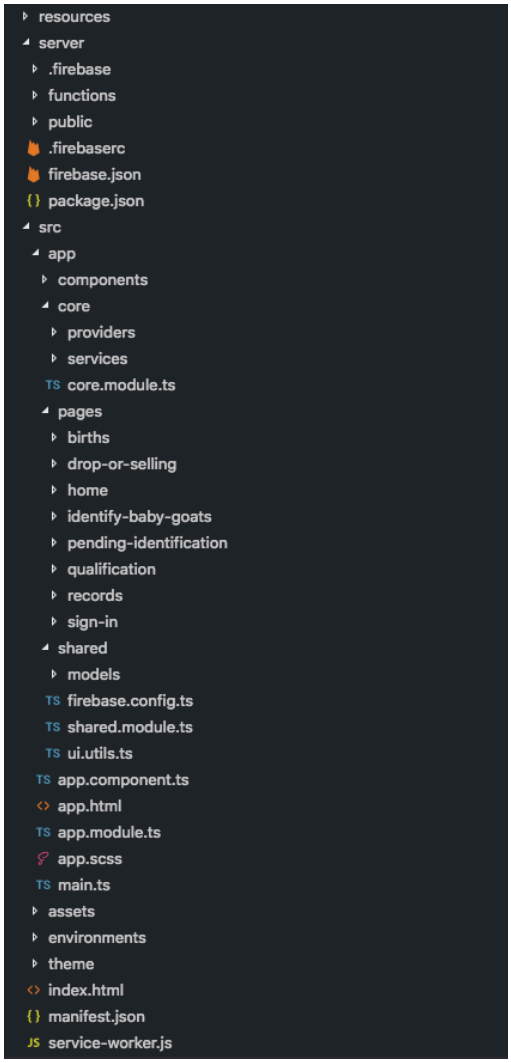

Ilustración 17: Estructura del proyecto

Primeramente, señalar como se ve en la *Ilustración 17* la incorporación de dos módulos a la aplicación: el *CoreModule* y el *SharedModule* los cuales cumplen una función fundamental para garantizar la escalabilidad de cualquier aplicación basada en angular:

- **CoreModule:** contiene todos aquellos servicios *Singleton*, componentes universales y cualquier otra característica de la que se requiera tener una única instancia en toda la aplicación. Para prever la reimportación de este modulo en otros sitios, normalmente se añade un *guard* en su constructor, que permiten controlar el acceso a una sección determinada de la aplicación.
  Además de este *guard* en nuestras aplicaciones hemos incorporado los *services* y *providers* necesarios.

- **SharedModule:** se incluyen todos los componentes compartidos por la aplicación, tales como modelos, *pipes* y cualquier otro elemento utilizado repetidamente por distintos componentes de nuestra aplicación. En el caso de las no hemos utilizado ningún elemento de este tipo, pero no son más que operaciones que puede añadirse al *HTML* de un componente a la hora de mostrar contenido de la ejecución de *TypeScript* (como variables, por ejemplo) que permiten transformar dicho contenido antes de visualizarlo. Un ejemplo claro sería el *pipe* de *date* ya incluido en *Angular* nativamente, que transforma un elemento de tipo *Date* que por defecto se mostraría como una ristra inteligible a un formato determinado especificado en el pipe, como se puede ver bajo estas líneas.

```
<p>Mi fecha de nacimiento es {{ birthday | date:"MM/dd/yy" }} </p>
```

  Resultado: Mi fecha de nacimiento es 11/11/92

  En el hemos incluido por ejemplo el módulo de *Angular Material* para la aplicación de escritorio que incluye todos los componentes de la librería que utilizamos y los modelos de nuestra base de datos en ambas aplicaciones.
  Este componente será reutilizado en los distintos módulos de nuestra aplicación, por lo que es importante que no tenga dependencias de otros módulos, lo cuál podría dar problemas por dependencia circular.

Ahora continuaremos detallando el resto de las carpetas de la aplicación:

- **Components:** carpeta que almacena aquellos componentes que se reutilizan en la aplicación, tales como la barra de progreso en la aplicación móvil o las tablas en la aplicación web.
- **Providers y Services:** se almacenan los servicios de las aplicaciones. En la aplicación de escritorio no existen los **Providers**, son un concepto de *Ionic*, aunque funcionan exactamente igual. Los servicios nos permiten obtener recursos y realizar tareas en los mismos, como peticiones http, acceder al almacenamiento loca, etc.
- **Pages:** se incluyen las distintas vistas del sistema, con un archivo HTML, un archivo SCSS para edición de estilos y un archivo TS para la ejecución de código **TypeScript** donde incorporar la lógica.

- **Models:** se conservan los distintos modelos definidos como clases que referencian a los usados en la base de datos de *Firebase* para simplificar y dejar el código más limpio cuando usamos *TypeScript*.
- **Server:** se incorpora el despliegue de nuestra aplicación en *Firebase* con una subcarpeta *Functions* donde declarar las funciones en la nube que queremos que se disparen cuando se realicen acciones en *Firebase RealTime Database*.

7.2.1 Angular (Aplicación web de escritorio)

Utilizamos la herramienta de angular principalmente durante el desarrollo de la aplicación web de escritorio. Nos decantamos por que es el mismo *framework* de desarrollo que usa *Ionic* por lo que podríamos reutilizar gran parte del código y simplificaba muchísimo el tiempo de aprendizaje y periodo de adaptación al entorno. Así mismo, teníamos la ventaja de seguir el mismo tipo de estructura en el proyecto e integrarlo como un despliegue del mismo proyecto gracias a que *Firebase* así nos lo permite, teniendo ambas aplicaciones contenidas en el mismo proyecto.

Partes destacables de este código serían el *guard* de control de usuario de tipo administrador y la funcionalidad de auto-importación de los documentos CSV al servidor. A continuación, pasaremos a explicarlas un poco más en detalle:

1. ***Guard* de control de usuario:** primeramente, deberíamos definir *guard* que no es más que un concepto de angular que constituye una pequeña función que devuelve un tipo *boolean* diciendo si el acceso a un componente (página del menú en nuestro caso) puede ser o no accesible. Si devuelve verdadero, el acceso se completa de manera transparente al usuario, y en caso contrario simplemente no se accede a esa sección. Para lograrlo establecimos un archivo *admin-auth.gard.ts* en la carpeta *server/public/src/app/core/guards* como un elemento inyectable para que angular pueda incrustarlo usando su inyector de dependencias interno con una función *canActivate* (nombre estándar para la activación de *guards* según la guía de estilos de *Angular*) definida como se ve a continuación:

```
@Injectable()
export class AdminAuthGuard implements CanActivate {

    public constructor(
        private angularFireAuth: AngularFireAuth,
        private db: AngularFireDatabase,
        private router: Router,
        private toastrService: ToastrService
    ) {
    }

    public canActivate(route: ActivatedRouteSnapshot, state: RouterStateSnapshot): Observable<boolean> {
        return this.angularFireAuth.user.pipe(
            switchMap(user => {
                return this.db.object<Admin>(`admins/${user.uid}`).valueChanges();
            }),
            map((admin: Admin) => {
                if (admin != null) {
                    return true;
                } else {
                    this.router.navigateByUrl('/sign-in');
                    this.toastrService.error('This user does not have enough privileges to access this section');
                    return false;
                }
            })
        );
    }
}
```

Ilustración 18: AdminAuthGuard

En ella hacemos uso del *plugin* de *AngularFire* para acceder al servicio de autentificación de nuestro proyecto *Firebase* y accedemos a un observable del usuario que haya iniciado sesión. Haciendo uso de la librería *RxJS* manejamos dicho observable para, una vez recibido el valor de este, se ejecite una petición de tipo promesa a la base de datos a la tabla de administradores con el identificador único del usuario, mapeamos el resultado de esa petición y comprobamos si efectivamente nos ha dado un objeto de tipo usuario. En caso afirmativo devolvemos verdadero permitiendo al usuario acceder a la sección y en caso contrario lo redirigimos al *login* y mostramos un mensaje de error correspondiente.

Incorporamos dicho *guard* en el archivo *app-routing.module.ts* que gestiona las rutas de nuestra aplicación:

```
const routes: Routes = [
  { path: '', component: LandingPageComponent },
  { path: 'sign-in', component: SignInPageComponent },
  { path: 'home', component: NavBarComponent, canActivate: [AdminAuthGuard],
    children: [
      { path: 'pendingUsers', component: PendingUsersPageComponent},
      { path: 'goats', component: GoatsPageComponent},
      { path: 'ranches', component: RanchesPageComponent},
      { path: 'births', component: BirthsPageComponent}
    ] },
];
```

Ilustración 19: App-routing Module

Y de esa manera podemos usar nuestro *guard* para controlar el acceso a la ruta */home*. Nótese que al tener componentes hijos no es necesario añadir dicho *guard* a cada uno de sus hijos.

2. **Autoimporación de CSV**: para ver esta funcionalidad podremos acceder a una de las secciones con importación de ficheros como por ejemplo el componente *GoatsPage* situado en la ruta *server/public/src/app/pages/goats-page* y revisamos el archivo *TypeScript* correspondiente:

```
public convertFile(event: any): void {
    const file = event.files[0];
    this.readFileContent(file).then(content => {
        this.processCsv(content);
    }).catch(error => console.log(error));
}

public readFileContent(file): Promise<any> {
    const reader = new FileReader();
    return new Promise((resolve, reject) => {
        reader.onload = (event: any) => resolve(event.target.result);
        reader.onerror = error => reject(error);
        reader.readAsText(file);
    });
}

public processCsv(content): void {
    const goatsArray = content.split('\n');
    goatsArray.forEach(goatLine => {
        const lineIndex = goatsArray.findIndex(line => line === goatLine);
        if (lineIndex === 0) {
            return;
        }
        goatsArray[lineIndex] = this.getGoatFrom(goatLine);
    });
    goatsArray.shift();
    this.goatsService.setGoats(goatsArray).then(() => {
        this.goats = goatsArray;
    });
}
```

Ilustración 20: Goats Page

Primeramente, convertimos el archivo seleccionado en la función *convertFile* y lo leemos usando el *FileReader* de *JavaScript* en la función *readFileContent*. Una vez leído procesamos su contenido realizando un array de animales separado por saltos de línea (del archivo csv) y para cada línea generaremos un elemento *Goat* en un vector *goatsArray*.

```
public setGoats(goats: Goat[]): Promise<void> {
    const goatsArrayOfFireBaseObjects = {};
    goats.forEach(goat => {
        if (goat.id) {
            goatsArrayOfFireBaseObjects[goat.id] = goat;
        }
    });
    return this.db.object('goats').update(goatsArrayOfFireBaseObjects);
}
```

Ilustración 21: GoatsService

Por último, una vez cada línea se envía el vector completo al servicio de la entidad estableciendo todos los animales importados de manera masiva en nuestra base de datos como se ve en la imagen superior.
Es importante destacar que la operación escogida de *AngularFire* es un *update* porque de lo contrario, si usaramos un *set* por ejemplo estaríamos eliminando por completo el elemento anterior por lo que perderíamos los datos de nuestra aplicación para esos animales si existiera una segunda importación. El *update* solo actualiza aquellos campos que hayan visto modificado su valor.

7.2.2 Ionic (Aplicación móvil)

Escogimos el uso de *Ionic* como *framework* de desarrollo para la aplicación móvil por diferentes motivos:
El primero ya se señaló en el apartado anterior y es por que utiliza *Angular* para su estructura e implementación, por lo que podemos reutilizar bastante código y simplificar el trabajo.
El segundo es que este *framework* es multiplataforma, es decir, sólo debemos realizar el código en un único lenguaje y podremos exportarlo a las diferentes plataformas disponibles, en este caso *iOS iOS* y *Android*.
El tercero y más importante era nuestra experiencia con este *framework* siendo el más popular dentro de la empresa desarrolladora y habiendo tenido personalmente experiencia con el mismo en las asignaturas del máster.

Partes destacables de las implementaciones de la aplicación móvil serían: el inicio de sesión con verificación *SMS* y el reconocimiento de dígitos en una fotografía haciendo uso de una *Cloud Function* de *Firebase* y *Cloudinary*. Para verlas un poco más de cerca las explicaremos en los siguientes puntos:

1. **Inicio de sesión con verificación SMS:** primeramente accedemos al archivo *TypeScript* de la página *sign-in* ubicada en *src/app/pages/sign-in* y veremos la función que se lanza cuando se completa el formulario de inicio de sesión.

```
public async signIn() {
    this.loadingService.presentLoading();
    if (!this.verificationID) {
        let number = this.signInForm.get('phoneNumber').value;
        const isValid = await this.authProvider.isValidPhone(number).catch(console.error);
        if (isValid) {
            this.verificationID = await this.authProvider.askForConfirmNumber(number).catch(e => { console.error(e); return 'Error' })
        } else {
            this.loadingService.dissmissLoading();
            return this.alertCtrl.create({ title: 'Este teléfono no está registrado' }).present();
        }
    } else {
        await this.authProvider.confirm(this.verificationID, this.signInForm.get('code').value).catch(e => console.error(e));
    }
    this.loadingService.dissmissLoading();
}
```

Ilustración 22: SignIn

Vemos que se inicia un servicio que nos presenta una pantalla de carga y cogemos el valor del formulario *phoneNumber* y lo enviamos a la función *isValidPhone* del servicio *authProvider*. En caso de no estar registrado se mostrará una alerta de error, pero en caso de tener el teléfono dado de alta (lo tiene que hacer el Administrador desde el panel web previamente por requisitos del cliente) se solicita un código de verificación, que si es introducido correctamente por el usuario solicitará la confirmación del código al servicio y accederá al sistema correctamente.
En el servicio, como vemos abajo, utilizamos el *plugin* de *AngularFire* para realizar estas operaciones, que, en caso satisfactorio, actualizarán el *BehaviorSubject* del usuario al que nos suscribimos al arrancar la aplicación en y dará acceso al usuario al sistema.

```
async logIn(number: string) {
  let verificationID = "";
  if (this.platform.is('cordova')) {
    verificationID = await this.fireNative.verifyPhoneNumber(number, 120).catch(e => { console.error(e) });
  }
}

async confirm(verificationID: string, code: number) {
  return this.fireNative.signInWithVerificationId(verificationID, code);
}
```

Ilustración 23: AuthProvider

2. **Reconocimiento de dígitos en una fotografía haciendo uso de una Cloud Function de Firebase y Cloudinary:** primeramente, para ver esta implementación deberemos acceder a cualquier pantalla que tenga esta implementación como por ejemplo el archivo *TypeScript* situado en */src/app/pages/births/mother-identification*.

```
public async scanWithCamera(): Promise<void> {
    const cameraOptions: CameraOptions = this.utilsService.onlyCameraOption();
    const base64Image = await this.camera.getPicture(cameraOptions);
    this.loadingService.presentLoading();
    this.utilsService.searchForCrotalInImage(base64Image).subscribe((result: any) => {
        this.showCrotalSearch = true;
        this.motherForm.get('id').setValue(result.recognizedText);
        this.loadingService.dissmissLoading();
        this.selectedMethod = 'id';
    }, error => {
        this.loadingService.dissmissLoading();
    });
}
```

Ilustración 24: MotherIdentification

En dicho archivo podemos observar la función *scanWithCamera* que usando un *plugin* nativo de *Ionic* recoge la imagen desde la cámara o desde el álbum. Una vez adquirida la imagen, en base 64, la enviamos al método *searchForCrotalInImage* y en caso de que la *API* de *Cloud Vision* reconozca el texto nos lo enviará en el resultado y lo estableceremos debidamente en el campo del formulario.

```
public searchForCrotalInImage(base64Image: string): Observable<String> {
    return this.http.post<string>(`${this.googleVisionAPIUrl}/searchForCrotalInImage`, { image: base64Image });
}
```

Ilustración 25: UtilsService

En el servicio, como podemos observar arriba, simplemente se hace una petición a la dirección de nuestra *Cloud Function* añadiendo como contenido nuestra imagen en base 64. Para ver como opera nuestra *Cloud Function* deberemos echar un vistazo al archivo *index.js* situado en *server/functions* que es desde donde hemos desplegado las funciones relacionadas con nuestro proyecto *Firebase* enlazado previamente.

```
const functions = require('firebase-functions');
const admin = require('firebase-admin');
admin.initializeApp();
var db = admin.database();

const cors = require('cors')({origin: true});

exports.searchForCrotalInImage = functions.https.onRequest((request, response) => {
    cors(request, response, () => {
        const vision = require('@google-cloud/vision');
        const client = new vision.ImageAnnotatorClient();

        const base64EncodedImage = request.body.image;
        const imageToSend = {
            image: {content: base64EncodedImage}
        };

        client.textDetection(imageToSend)
            .then(results => {
                const result = results[0];

                const detections = result.textAnnotations;
                if (detections.length === 0) {
                    return response.send({recognizedText: null});
                }
                const dirtyRecognizedText = detections[0].description;
                const recognizedText = dirtyRecognizedText.replace('\n', '');
                return response.send({recognizedText});

            })
    });
});
```

Ilustración 26: Index.js

En la *Ilustración 26* podemos observar como requerimos la librería de *Cloud Vision* y enviándole a un cliente de anotaciones la imagen recogemos el reconocimiento de texto si lo hubiera, eliminando posibles saltos de línea que introduce a veces el software.

7.2.3 Conexión a la API de Firebase

Para la conexión de nuestras aplicaciones móvil y web a nuestro *Backend* en *Firebase* sólo necesitamos aplicar la siguiente configuración mostrada en la *Ilustración 18* que nosotros hemos incorporado en un archivo *firebase.config.ts*.

```
export const FIREBASE_CONFIG = {
    apiKey: '111111111111111111',
    authDomain: 'cabra-palmera-gescan.firebaseapp.com',
    databaseURL: 'https://cabra-palmera-gescan.firebaseio.com',
    projectId: 'cabra-palmera-gescan',
    storageBucket: 'cabra-palmera-gescan.appspot.com',
    messagingSenderId: '1111111111'
};
```

Ilustración 27: Configuración de la API de Firebase

Una vez hecho, simplemente importamos nuestra configuración en nuestro *CoreModule* tal y como muestra la siguiente imagen:

```
firebase.initializeApp(FIREBASE_CONFIG);

@NgModule({
    imports: [
        IonicModule,
        AngularFireModule.initializeApp(FIREBASE_CONFIG),
        AngularFireAuthModule,
        AngularFireDatabaseModule,
        HttpClientModule
```

Ilustración 28: Configuración de Firebase en CoreModule

Con ello ya tendríamos el acceso a nuestra base de datos de *Firebase* configurado y podríamos acceder a ella habiendo instalado el *plugin AngularFire* usando módulos tales como *AngularFireAuth* (para los servicios de autentificación) o *AngularFireDatabase* para acceder a la base de datos.

8. Pruebas

8.1 Pruebas de usabilidad

Para verificar que nuestra aplicación había cumplido los objetivos que nos habíamos marcado, y más importante aún, cubría las necesidades del cliente realizamos una serie de pruebas del prototipo, solicitando a distintos usuarios finales (técnicos y ganaderos) que hicieran uso de la aplicación y nos dieran su opinión acerca de los siguientes apartados:

- Verificar si la interfaz era clara, sencilla de usar, intuitiva.
- Comprobar si el proceso de inicio de sesión y registro resultaba tedioso o largo de realizar.
- Verificar si los usuarios, por sí mismos y sin indicaciones, eran capaces de navegar sin problemas por la aplicación y realizar las acciones que deseaban.
- Asegurar que cuando se mostraba un mensaje de error, éste era claro y conciso y ayudaba al usuario a subsanarlo por si mismo.
- Medir y cuantificar si los tiempos de carga resultaban o no molestos para el usuario.
- Preguntar si realmente notaban alguna diferencia respecto a aplicaciones nativas en cuanto a rendimiento, fluidez o diseño.
- Preguntar si recomendarían dicha aplicación por encima de la aplicación existente y si estarían dispuestos a migrar por completo a nuestro sistema.

Tuvimos bastantes pruebas en numerosas reuniones. De entre ellas destacamos:

1. Primera reunión con el cliente para identificar cuáles serían los requisitos de su aplicación y para que nos cuenten un poco su experiencia con las herramientas actuales. No tuvimos acceso directo a las aplicaciones, pero nos realizaron una pequeña explicación de las herramientas y su flujo de trabajo recogido y explicado anteriormente en esta memoria en el apartado de Análisis.
2. Segunda reunión para verificar el apartado visual con los *mockups* realizados para el sistema. Gracias a esta segunda reunión recogimos bastante información sobre como eran identificados los animales por parte de los ganaderos y que la opción de búsqueda por nombre sería una genial adición no contemplada inicialmente. Así mismo se validó la estructura de la aplicación y se añadieron funcionalidades que inicialmente no estaban contempladas, como poder dar de baja a las crías directamente desde la ventana de nacimiento o la sección completa de identificar a crías pendientes por parte del ganadero.
3. Reunión para validar los cambios añadidos en función de la anterior reunión, así como debatir las primeras instancias del diseño final de la aplicación, el estilo e identificar si la aplicación podría tener cuota de mercado significativa si se ampliara a otros tipos de animales o ganaderías.

4. Se debate el diseño final de la aplicación, en los que se identifican ciertas alertas que deberían aparecer en el sistema al realizar ciertas operaciones, como cuando no se ha introducido el código temporal de ninguna cría y se menciona en detalle el apartado de calificar, el cuál estaba pendiente porque el cliente no tenía claro como sincronizar la nueva aplicación con su herramienta actual.
5. Verificación de la aplicación desarrollada a modo de prototipo y publicación en las tiendas correspondientes. Además, se tomó nota de cuáles serían los siguientes pasos para la próxima versión de la aplicación y se nos informó de que había sido presentada en un foro nacional de ganaderías, incluyéndonos en la autoría lo cuál nos hizo sentir muy orgullosos.

   Una vez recibidas las respuestas por su parte pudimos verificar los siguientes puntos:

   - La aplicación es sencilla y clara de usar, limpia y minimalista y se indican claramente sus diferentes apartados.
   - El proceso de inicio de sesión es muy similar al de otras aplicaciones de mensajería, por lo que parece estándar, sencillo y rápido.
   - No tuvimos que aclarar a los usuarios como navegar por la aplicación, ni mucho menos que acciones podían realizarse o qué significaba algún componente de la interfaz.
   - Los mensajes de error son claros y ayudan mucho a darnos cuenta de qué errores se han cometido y hay muy pocos.
   - La aplicación es rápida, con un rendimiento muy similar al de otras aplicaciones de mensajería.
   - La mayoría pensó que se trataba de una aplicación nativa. Los compañeros del trabajo con conocimientos en informática notaban la diferencia, pero nos comentaban que era casi inapreciable para un usuario estándar.
   - Todos los usuarios de prueba recomendarían el uso de la aplicación.

8.2 Pruebas de interacción

Por último, probamos a realizar todos los posibles flujos, correctos e incorrectos de ambas aplicaciones para detectar posibles fallos, interaccionando la una con la otra en una base de datos de prueba creada para la ocasión. Detectamos varios fallos de maquetado y algunos de diseño de la base de datos que posteriormente fueron solucionados y trajeron a la mesa nuevas ideas de funcionalidades.
Así mismo, facilitamos las aplicaciones a algunos usuarios de prueba ganaderos y técnicos para que nos dieran más información, con lo que detectamos algunos problemas de visualización en dispositivos *iOS* o la ocultación de información cuando aparecía el teclado del sistema operativo en las secciones con introducción de texto. Se realizaron también pequeñas mejoras, como mostrar el sexo de los animales o cambiar la imagen de perfil de los animales por defecto, añadir notas y puntuaciones al finalizar una calificación y cambiar todos los selectores del apartado Calificar a un selector de rueda cuando fueran más de 3 elementos.

9. Resultados, conclusiones y trabajo futuro

9.1 Resultados y conclusiones

Por la parte académica creo que ha sido un hito en mi carrera profesional, nunca hasta ahora había tenido que lidiar con la gestión completa de un proyecto por mi cuenta, siendo completamente independiente en todas las etapas tales como Análisis, Diseño y Desarrollo, habiendo alcanzado un nivel de madurez en este sentido que ni me habría planteado antes de entrar en el máster.

E igual que pasó con mi trabajo de fin de grado, fue duro al principio, pero la satisfacción de ver que tantos estudios han merecido la pena y que realmente, pesa las dificultades eres perfectamente capaz de sobrellevar la situación y alcanzar los objetivos es una sensación incomparable.

Por la parte de colaboración con la empresa, sólo puedo decir que en Squaads he tenido una experiencia muy gratificante, he visto como he podido profundizar en conocimientos que ya poseía hasta niveles que no me habría planteado y descubierto la densidad y dimensión de los *frameworks* de *Angular* y *Ionic*. Además, he descubierto nuevas utilidades y *frameworks*, teniendo un acercamiento real a los mismos por las características del proyecto, como son *Node.js* y *Firebase*, dejando a un lado los ejemplos típicos que podremos encontrar por internet.

Ha sido un placer y una experiencia muy gratificante poder compartir estos meses de trabajo con el equipo de José Luis. Valoro muchísimo que me hayan dado la capacidad de trabajar en un proyecto de esta envergadura y que la gestión de este corra por mi parte, siendo alguien nuevo en la empresa denota mucha confianza en mi persona y espero haber sabido corresponder con mi trabajo.

En lo que respecta al prototipo entregado al cliente, las opiniones que tuvimos durante la fase de pruebas fueron mayoritariamente positivas, señalando nuestro buen trabajo y diseño, habiendo marcado un antes y un después en el uso de los sistemas informáticos en la empresa. De hecho, presentaron nuestra aplicación en un fórum especializado y se realizó un *paper* de la misma, adjunto en el último anexo de esta memoria. Repasando la lista de objetivos iniciales de la aplicación:

- Ofrecer una interfaz de usuario sencilla, clara, intuitiva e independiente del sistema operativo o el dispositivo.
    - Desarrollamos dos aplicaciones completas de escritorio y móvil que se comunican entre ellas. La interfaz es sencilla e intuitiva, y ha sido validada por el cliente.
- Unificar las funcionalidades de las tres aplicaciones existentes en un único proyecto y que pueda trabajarse con todas ellas sin sincronizaciones manuales.
    - Usamos una única base de datos en *Firebase* que permite la sincronización entre las dos aplicaciones sin problemas que además permite su uso offline cuando no hay cobertura suficiente en entornos como el campo.
- Agilizar los flujos de trabajo y procesos que se realizan a diario en la aplicación.
    - La aplicación está disponible en el móvil que poseen todos los usuarios del sistema facilitando mucho su disponibilidad e inmediatez del uso y no requiere un traspaso de información duplicada entre ganaderos y técnicos.
- Simplificar todos aquellos procesos que se puedan e implementar las nuevas funcionalidades que requiera el cliente.
    - También se han reducido durante la fase de análisis y con el cliente muchos de sus procesos por sugerencias tanto suyas como nuestras que no se habían planteado en el inicio, reduciendo el número de pasos al máximo.
- Asegurarnos de que tanto ganaderos como técnicos se sienten satisfechos con el producto final y mejora sustancialmente su trabajo.
    - El número de reuniones con el cliente para ir validando el proyecto así como las numerosas pruebas realizadas por nuestra parte y con usuarios de prueba por parte del cliente, nos ha garantizado un producto final solvente, preciso y que satisface con creces las necesidades del usuario.

9.2 Trabajo futuro

En lo que respecta a nuestro sistema prototipo, durante su desarrollo ya adelantamos que se percibieron nuevas funcionalidades que implementar en futuras versiones, pero deberían ser integradas en una segunda iteración del proyecto por límites de tiempo. Por ello las destacaremos a continuación, haciendo patente que este proyecto seguirá adelante por cuenta de la empresa colaboradora. Las funcionalidades futuras ya fueron mencionadas anteriormente en los requisitos ideales del proyecto en el apartado de Análisis y la continuación de este proyecto y sus funcionalidades correrá a cargo de la empresa desarrolladora.

10. Bibliografía

[1] Wikipedia. "GNU General Public License", [en línea]. Disponible en: https://es.wikipedia.org/wiki/GNU_General_Public_License

[2] StarUML. "About StarUML", [en línea]. Disponible en: http://staruml.sourceforge.net/v1/license.php

[3] Wikipedia. "Licencia MIT", [en línea]. Disponible en: https://es.wikipedia.org/wiki/Licencia_MIT

[4] Google. "Licencia de Google Firebase", [en línea]. Disponible en: https://firebase.google.com/terms/billing

[5] Atlassian. "Licencia de Bitbucket", [en línea]. Disponible en: https://es.atlassian.com/licensing/bitbucket-server

[6] JetBrains s.r.o. "Toolbox Subscription License Agreement For Education And Training", [en línea]. Disponible en: https://www.jetbrains.com/store/license_classroom.html

[7] Microsoft Corporation. "Términos de licencia de Microsoft Office 365", [en línea]. Disponible en: http://download.microsoft.com/Documents/UseTerms/Office%20365_University_Spanish_e3b1384a-d4f2-415e-9674-67f1997fa630.pdf

[8] Wikipedia. "Software propietario", [en línea]. Disponible en: https://es.wikipedia.org/wiki/Software_propietario

[9] Wikipedia. "Ley Orgánica de Protección de Datos Personales y garantía de los derechos digitales", [en línea]. Disponible en: https://es.wikipedia.org/wiki/Ley_Org%C3%A1nica_de_Protecci%C3%B3n_de_Datos_Personales_y_garant%C3%ADa_de_los_derechos_digitales

[10] Jefatura del estado. "Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos Personales y garantía de los derechos digitales.", [en línea]. Disponible en: https://www.boe.es/buscar/doc.php?id=BOE-A-2018-16673

[11] Wikipedia. "Reglamento (UE) 2016/679", [en línea]. Disponible en: https://es.wikipedia.org/wiki/Reglamento_General_de_Protecci%C3%B3n_de_Datos

[12] Jefatura del estado. "Reglamento (UE) 2016/679", [en línea]. Disponible en: https://www.boe.es/doue/2016/119/L00001-00088.pdf

[13] Wikipedia. "Modelo de prototipos", [en línea]. Disponible en: https://es.wikipedia.org/wiki/Modelo_de_prototipos

[14] Alberto Lacalle. "Prototipos", [en línea]. Disponible en: http://albertolacalle.com/hci_prototipos.htm

[15] Jorge Trejos. "Modelo Prototipado" [en línea]. Disponible en: http://jorgetrejos.blogspot.com/2010/08/modelo-prototipado.html

[16] Geslib S.L. "Geslib", [en línea]. Disponible en: https://editorial.trevenque.es/productos/geslib/

[17] Gestión del Medio Rural de Canarias, “GMR”, [en línea]. Disponible en: https://www.gmrcanarias.com/servicios/
[18] Programación General de Computadoras, “Progecom”, [en línea] Disponible en: http://www.asinte.org/
[19] TrazaFarm. “TrazaFarm”, [en línea]. Disponible en: https://trazafarm.com/
[20] VacApp. “VacApp”, [en línea]. Disponible en: https://es.wikipedia.org/wiki/Cadena_de_bloques
[21] Wikipedia. “Cadena de bloques”, [en línea]. Disponible en: https://vacapp.net/es
[22] Wikipedia. “Programación por capas”, [en línea]. Disponible en: https://es.wikipedia.org/wiki/Programaci%C3%B3n_por_capas
[23] Ropbby Quintero. “Ventajas y Desventajas de Cliente – Servidor” [en línea]. Disponible en: https://robiniclienteservidor.weebly.com/ventajas---desventajas.html
[24] Cloudinary. “Cloudinary”, [en línea]. Disponible en: https://cloudinary.com/
[25] Wikipedia. “Desarrollo Basado en Componentes”, [en línea]. Disponible en: https://es.wikipedia.org/wiki/Ingenier%C3%ADa_de_software_basada_en_componentes
[26] Juan Méndez Rodríguez. “Reglas Empíricas de Diseño de Interfaces”, [en línea]. Disponible en: http://telepresencial1617.ulpgc.es/cv/ulpgctp17/pluginfile.php/80359/mod_resource/content/1/Tema_I_7.pdf
[27] Drifty Co. “Ionic”, [en línea]. Disponible en: https://ionicframework.com/
[28] The Apache Software Foundation. “Apache Cordova”, [en línea]. Disponible en: https://cordova.apache.org/
[29] Google. “Angular”, [en línea]. Disponible en: https://angular.io/
[30] Wikipedia. “JavaScript”, [en línea]. Disponible en: https://es.wikipedia.org/wiki/JavaScript
[31] Wikipedia. “SASS”, [en línea]. Disponible en: https://es.wikipedia.org/wiki/Sass
[32] Wikipedia. “CSS”, [en línea]. Disponible en: https://es.wikipedia.org/wiki/Hoja_de_estilos_en_cascada
[33] Linus Torvalds “Git”, [en línea]. Disponible en: https://git-scm.com/
[34] JetBrains, “JetBrains Webstorm”, [en línea]. Disponible en: https://www.jetbrains.com/webstorm/

[35] Google. “Un navegador web rápido y gratuito”, [en línea]. Disponible en: https://www.google.es/chrome/browser/desktop/index.html
[36] Adobe. “Adobe XD”, [en línea]. Disponible en: https://www.adobe.com/es/products/xd.html
[37] Atlassian. “Bitbucket”, [en línea]. Disponible en: https://bitbucket.org/

Anexo I: Manual de usuario

Aplicación móvil

Inicio de sesión

Ilustración 29: Inicio de sesión de la aplicación móvil

En las pantallas sobre estas líneas podemos identificar el inicio de sesión de la aplicación móvil. Para iniciar sesión, introduciremos nuestro número de teléfono móvil (que previamente habrán tenido que dar de alta en el panel de administración) y posteriormente, en la segunda pantalla introduciremos el código de verificación que recibiremos mediante SMS para validar nuestro teléfono.

Menú principal

A continuación, se nos mostrará el menú principal de la aplicación, mostrado en la *Ilustración 21*.

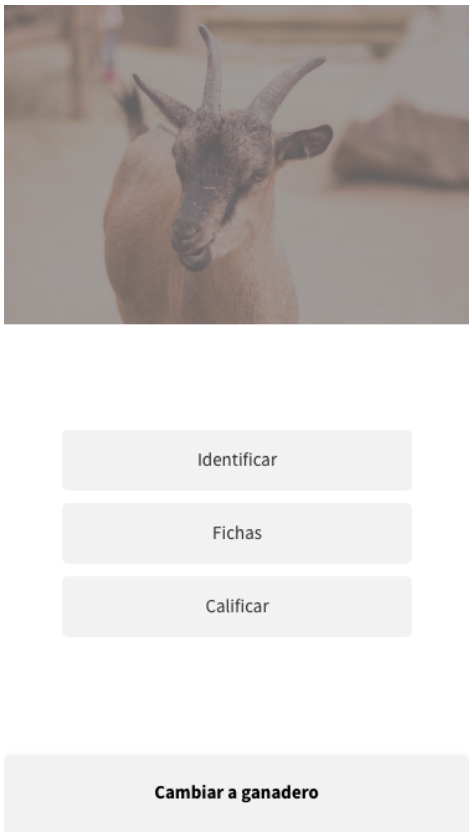

Ilustración 30: Menú principal (técnico)

La única diferencia entre este menú principal y el del técnico es la lista de opciones disponibles. Apareciendo en lugar de Identificar y Calificar las opciones de Partos y Bajas respectivamente.

Partos (Ganadero/a)

Para acceder a esta sección de la aplicación, simplemente tocar en el botón con su nombre en el menú principal y accederemos a ella.

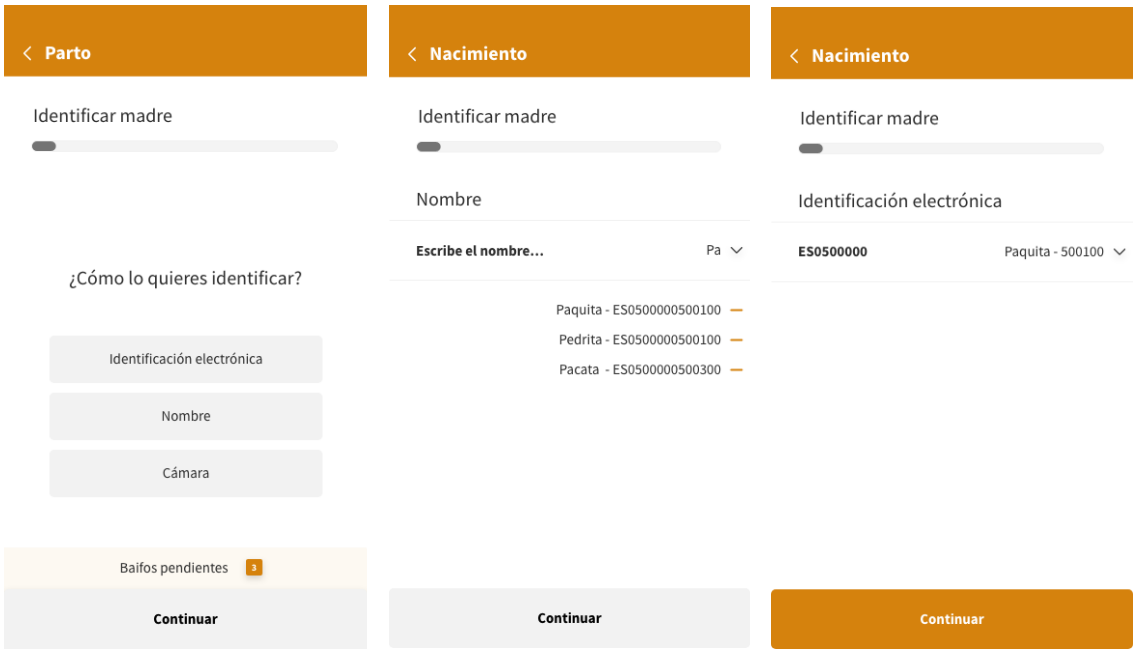

Ilustración 31: Partos (1)

Una vez seleccionada la sección, identificamos al animal utilizando cualquiera de los métodos disponibles: identificación electrónica (introduciendo el identificador numérico del animal), por nombre (búsqueda en los apodos de los animales) y cámara, que escaneará el número de la etiqueta del animal para su introducción automática.

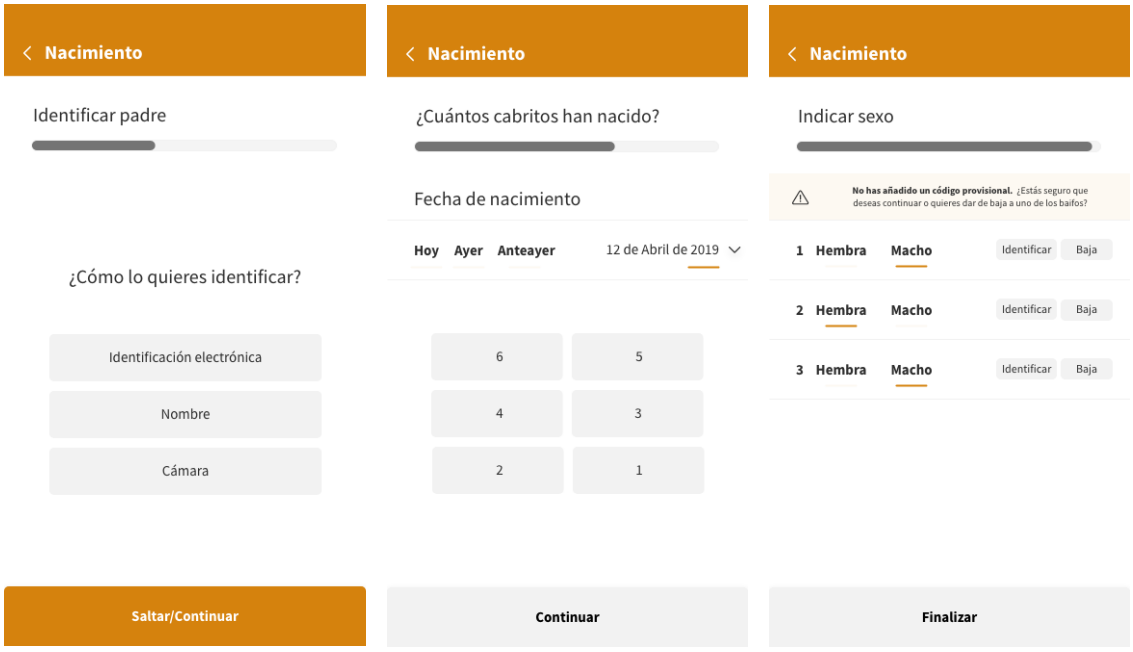

Ilustración 32: Partos (2)

Podemos añadir los detalles del padre si fuera necesario o saltar esta parte si lo desconocemos, en la siguiente vista debemos seleccionar el número de crías y la fecha del parto, por defecto seleccionando el día de hoy.
Por último, debemos indicar el sexo como se puede apreciar en el flujo de la *Ilustración 23* y añadir un código temporal a las crías si se desea, como se muestra en la última imagen de este apartado:

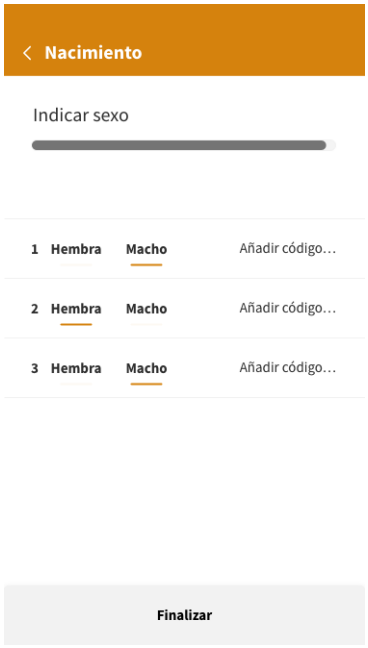

Ilustración 33: Partos (3)

Identificación de chivos pendientes (Ganadero/a)

Esta funcionalidad sólo aparecerá cuando existan partos registrados en nuestra explotación ganadera que no tengan un código temporal asociado a sus respectivas crías.

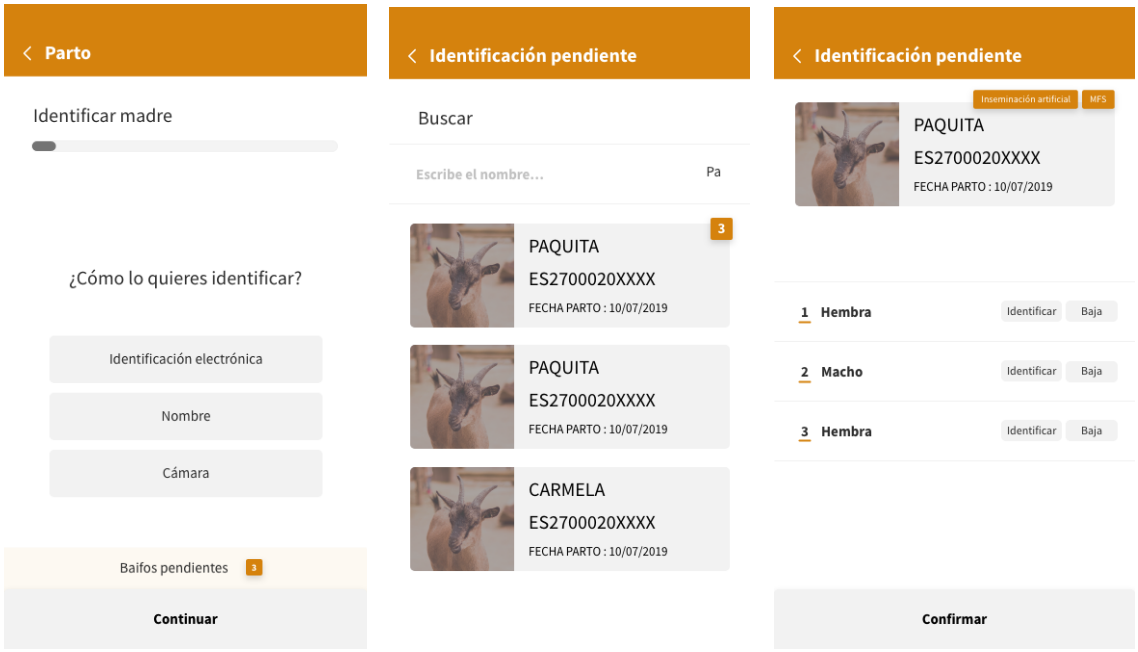

Ilustración 34: Identificación de chivos pendientes

Primeramente, accederemos a la sección de Partos como se puede observar en la *Ilustración 25*, donde aparecerá destacado un acceso a esta funcionalidad cuando sea necesaria nuestra acción al respecto. Al acceder se nos muestran las madres con su correspondiente parto pendiente y un número de crías pendientes por tener código temporal identificativo. Accedemos al detalle de la madre y podremos ver la información completa del parto. Por último, resolvemos las crías pendientes, o bien añadiéndoles un número identificativo temporal o bien dándoles de baja en el sistema directamente.

Identificar (Técnico/a)

Para acceder a esta sección simplemente tocaremos en el botón correspondiente del menú principal. El flujo inicial de esta sección lo podemos apreciar a continuación:

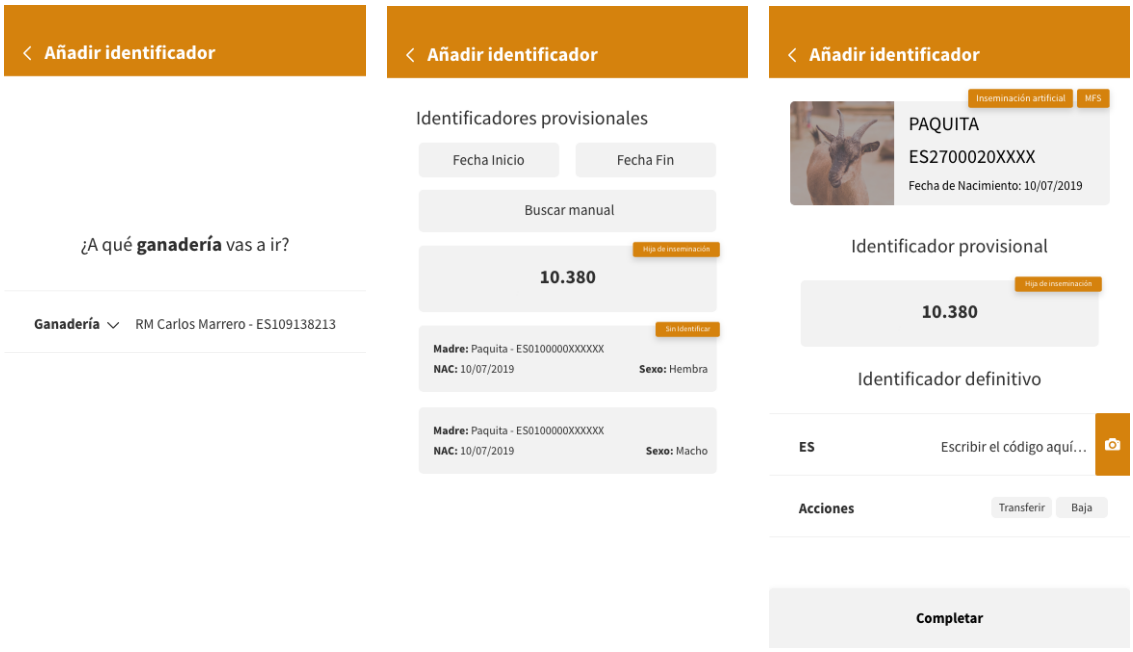

Ilustración 35: Identificar (1)

En este apartado de la aplicación podremos identificar a una cría de animal para convertirla en adulta. Para ello primeramente accedemos a la sección desde el botón del menú principal, después buscamos aquella cría que queramos modificar usando los filtros que prefiramos y ya en el detalle se brindan diferentes opciones que podemos ver en la *Ilustración 27*. Podemos identificar al animal simplemente escribiendo el código final o bien podemos transferirlo a otra ganadería o darle de baja en el sistema por completo.

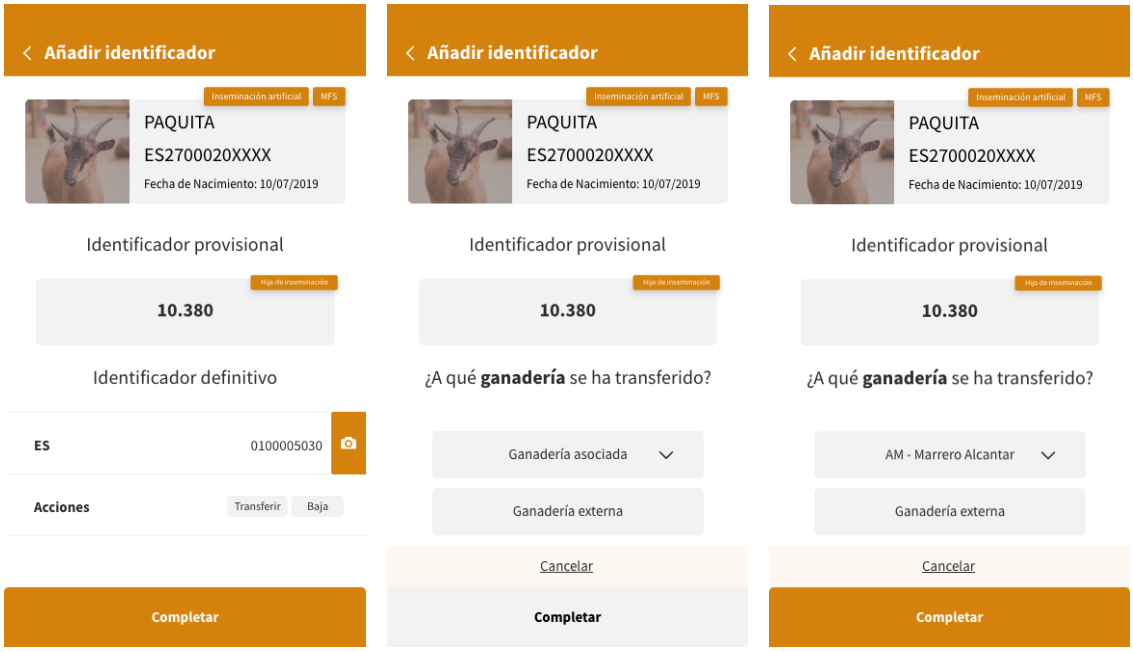

Ilustración 36: Identificar (2)

Calificar (Técnico/a)

Esta funcionalidad de la aplicación móvil permite al técnico realizar una calificación completa del animal en los diferentes apartados que se valoran en la especie: capa, apariencia general, carácter lechero, capacidad y sistema mamario.

Ilustración 37: Calificar (1)

Para continuar por esta sección, una vez seleccionada en el menú principal de la aplicación, indicaremos la ganadería en la que queremos calificar animales, identificaremos al animal con alguna de las opciones previamente explicadas en esta

memoria y llegaremos a la visión general de la sección, siguiendo el flujo de pantallas que podemos apreciar en la *Ilustración 28.*

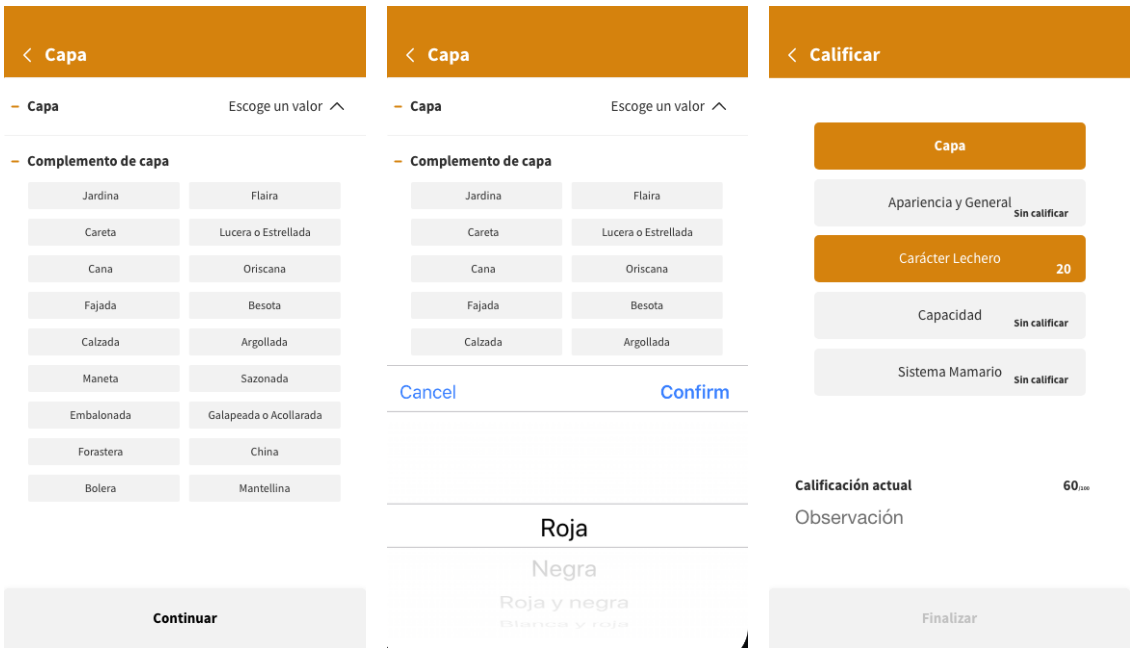

Ilustración 38: Calificar (2)

Procederemos a clasificar el animal entrando en cada uno de los correspondientes apartados y asignando una valoración del apartado según corresponda, haciendo uso de los selectores de rueda y botones habilitados para ello. Una vez completada un apartado, se actualizará la información del animal destacando qué secciones han sido completadas en la vista general de esta sección, tal y como se muestra en la última pantalla de la *Ilustración 29*.

Fichas (común)

En esta parte de la aplicación se puede consultar la información de los animales de manera rápida y sencilla. Para acceder simplemente seleccionamos dicha opción en el menú principal de la aplicación.

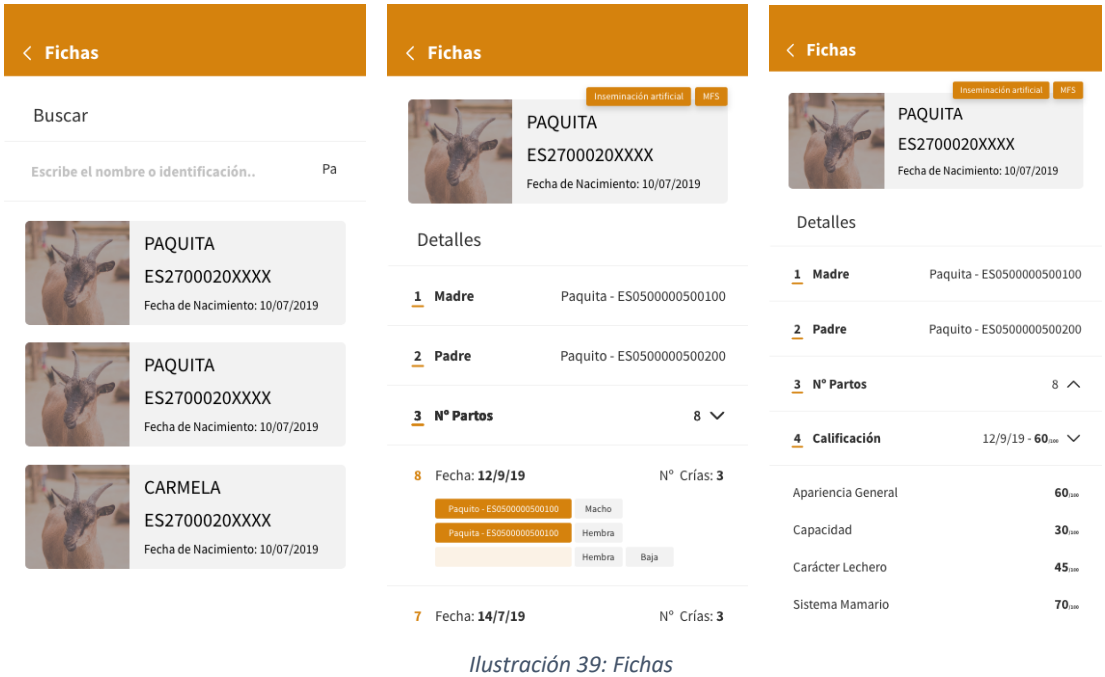

Ilustración 39: Fichas

Siguiendo el flujo de la ilustración superior, una vez accedida a esta sección desde el menú principal (y seleccionando la ganadería deseada en caso de ser técnico/a) filtramos la lista de animales por nombre o identificación hasta obtener la deseada.
Ingresamos en su pantalla de detalle donde podremos consultar su información, cambiar su nombre al tocar en el mismo o actualizar la foto del animal al tocar en la misma. Algunas partes de la visualización de sus datos quedan restringidas para usuarios de privilegios superiores en el sistema, como son las calificaciones detalladas, sólo visibles para los técnicos/as.

Baja (Ganadero/a)

En esta última sección de la aplicación móvil permitimos dar de baja a un animal o transferirlo a otra ganadería, ya sea del sistema (interna) o de fuera del mismo (externa), registrando estos cambios de propietario en la información del animal.

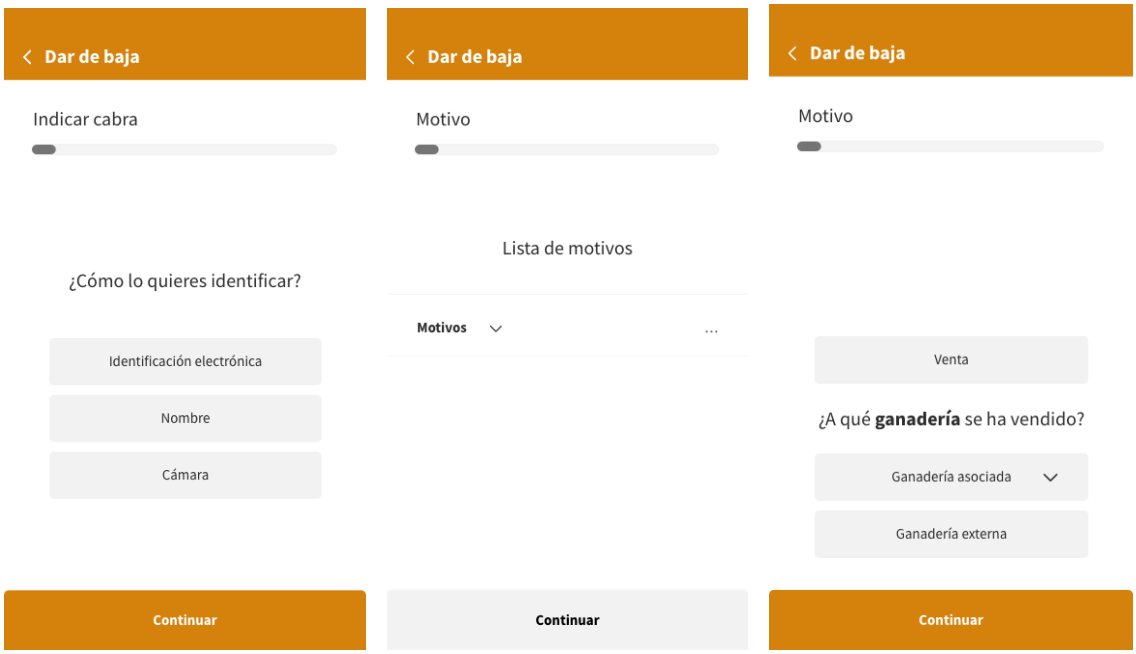

Ilustración 40: Baja

Siguiendo el flujo de las pantallas de la ilustración superior, podemos acceder la sección Baja desde el menú principal, para posteriormente realizar una identificación típica del animal como ya hemos descrito previamente para luego tramitar el motivo de la baja. En caso de que este motivo sea una transferencia/venta podemos registrar una transacción de un animal entre una ganadería y otra, bien sea interna o externa, guardando un historial en el animal para su posterior consulta.

Aplicación web de escritorio

Inicio de sesión

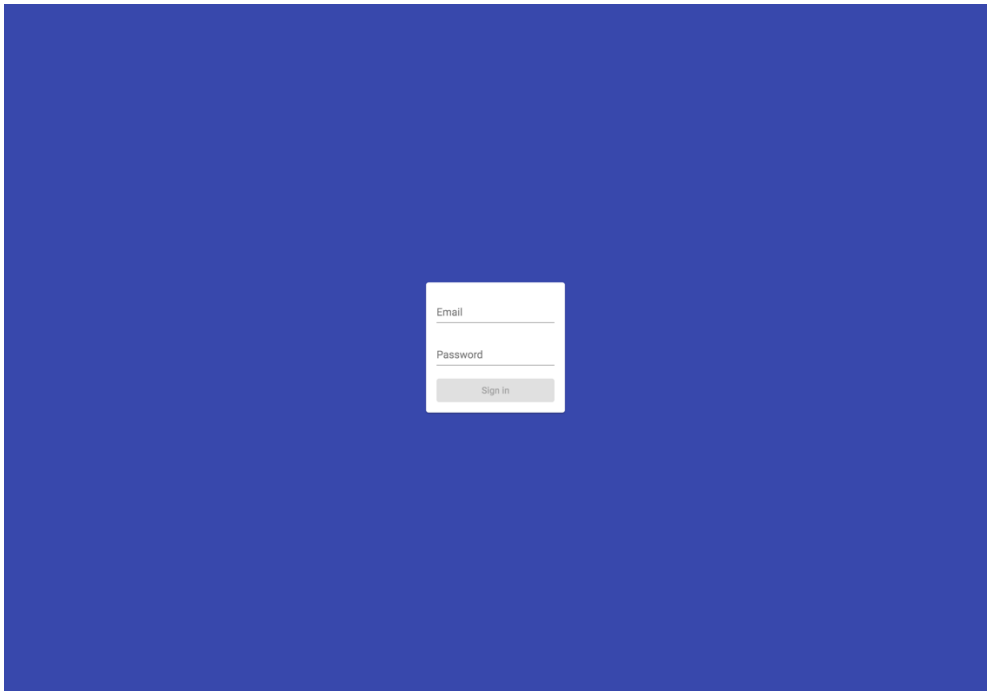

Ilustración 41: Inicio de sesión (panel)

En el inicio de sesión del panel simplemente introduciremos el correo electrónico de usuario y la contraseña correspondiente y pulsaremos el botón *Sign In*.

Sección de entidades

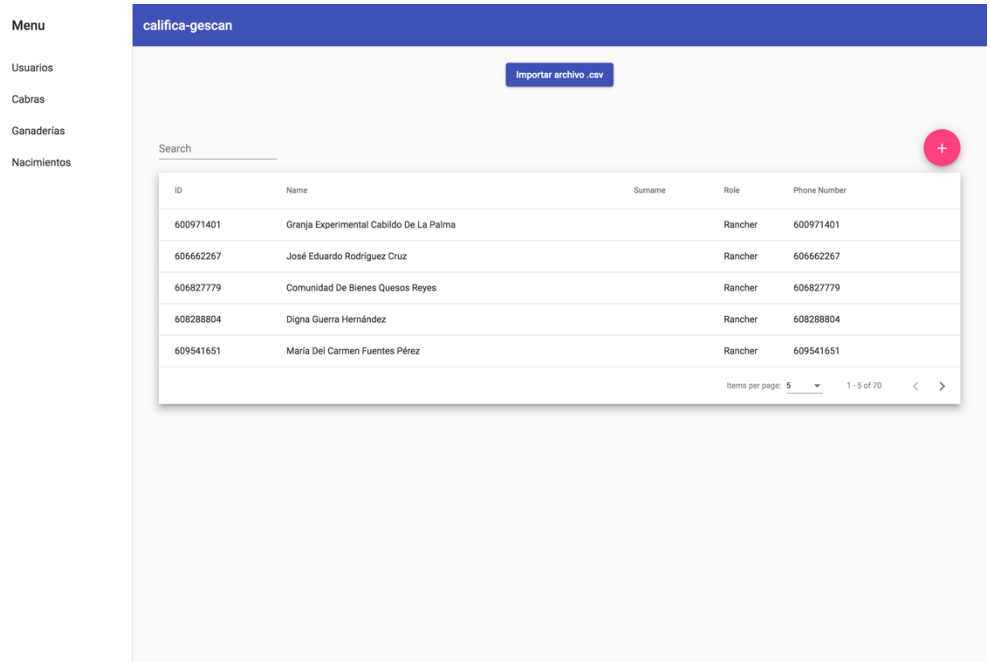

Ilustración 42: Usuarios

Una vez se haya iniciado sesión en la aplicación nos encontramos con el menú lateral y la tabla de usuarios, pudiendo filtrarlos, importarlos y añadir usuarios manualmente. El resto de vistas de la aplicación son exactamente iguales salvando la opción de añadir individualmente que sólo se encuentra disponible para usuarios.

Para añadir un usuario simplemente pulsaremos en el botón magenta de la parte superior derecha y se nos abrirá el siguiente diálogo:

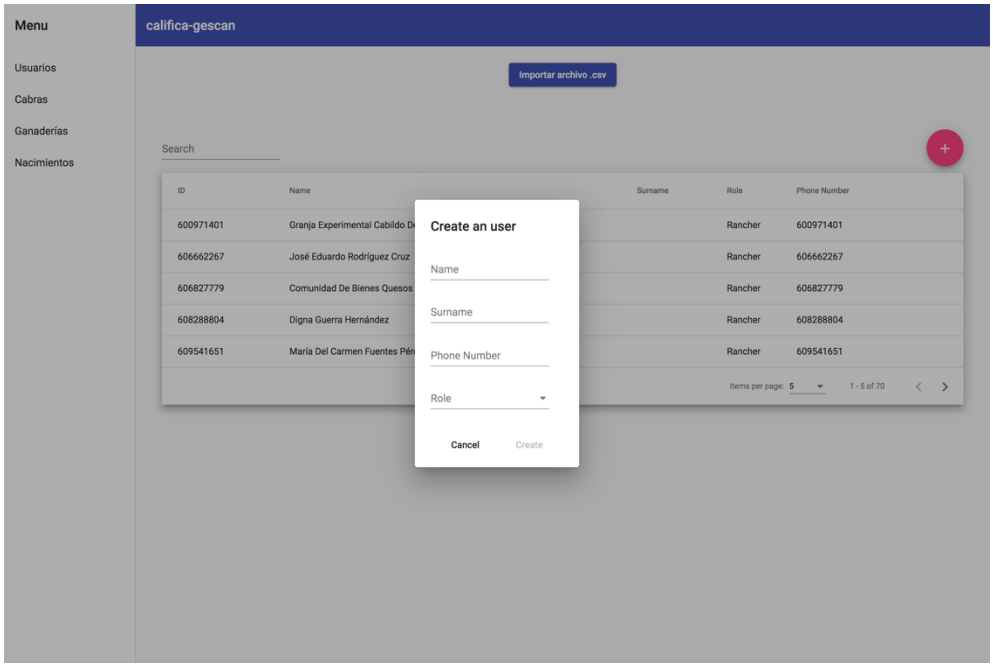

Ilustración 43: Crear usuario

Donde indicaremos los datos necesarios del usuario tales como Nombre, Apellidos, Número de teléfono y Rol.

Para la auto-importación de usuarios, animales, explotaciones y nacimientos simplemente pulsaremos en el botón *Importar archivo .csv* y seleccionaremos un archivo con esa extensión con la cabecera especificada por el cliente y se procesará y actualizará la tabla que representa la colección correspondiente de la base de datos.

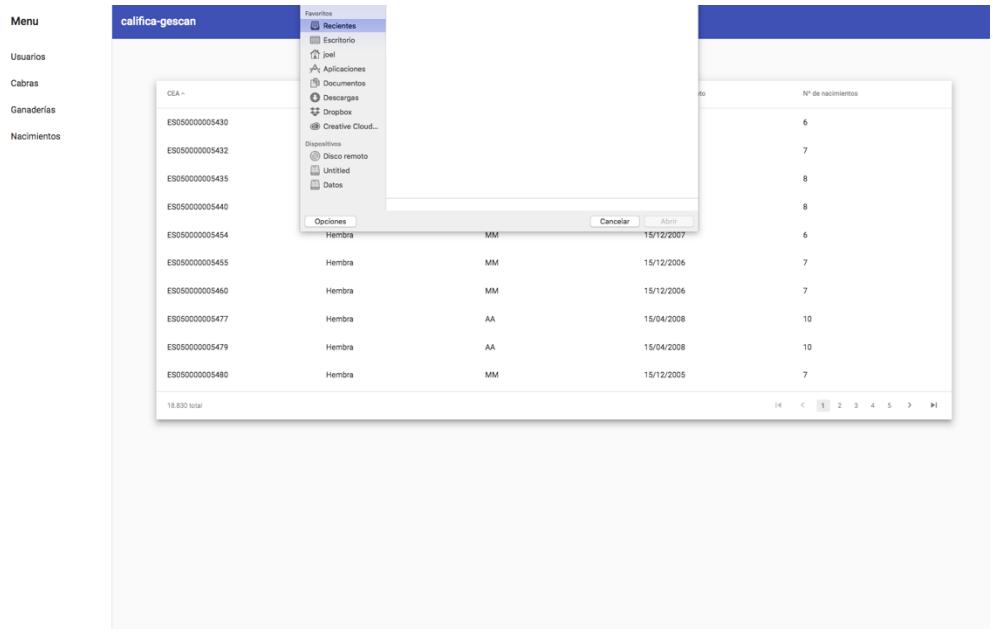

Ilustración 44: Autoimportación de entidades

Anexo 2: Instalación y ejecución de las aplicaciones

Para instalar y ejecutar las aplicaciones en modo de desarrollo deberemos seguir los siguientes pasos:

Pre-requisitos:

- Instalar el entorno de ejecución *JavaScript* Node.js para su sistema operativo.
- Instalar *Ionic CLI* usando el gestor de paquetes *npm*. Abrir un terminal de su sistema operativo y ejecutar la siguiente funcionalidad tras haber instalado *Node.js*.

```
sudo* npm install -g ionic
```

*Sólo si su sistema operativo está basado en Unix (Linux u OS X)

- Instalar *Angular CLI* usando el gestor de paquetes *npm*. Abrir un terminal de su sistema operativo y ejecutar la siguiente funcionalidad tras haber instalado *Node.js*.

```
sudo* npm install -g @angular/cli
```

*Sólo si su sistema operativo está basado en Unix (Linux u OS X)

Instrucciones:

1. Descomprimir el archivo .zip adjunto con esta memoria en la carpeta deseada.
2. Situarnos desde el terminal del sistema operativo en uso dentro de la carpeta.
3. Ejecutar el siguiente comando en la raíz de la carpeta:

```
npm install
```

4. En caso de querer ejecutar la aplicación móvil, simplemente ejecutar:

```
ionic serve
```

5. En caso de querer ejecutar la aplicación de escritorio, acceder desde su terminal a la carpeta */server/public*.
6. Ejecutar el siguiente comando:

```
ng serve --open
```

7. En ambos casos se abrirá el proyecto en su explorador web predeterminado, aunque recomendamos activar las herramientas de desarrollo y seleccionar la vista móvil en su explorador en el caso de la aplicación móvil.

Anexo 3: Presentación del proyecto en el Foro Nacional de Caprino

DESARROLLO DE APLICACIÓN MÓVIL PARA GESTIÓN DE DATOS EN CAMPO

Autores

Muñoz-Mejías E (1), Delgado J (2), González JL (2), Hernández B (3)

(1) Gescan Gestión de Programas de Cría SL.
(2) Bisnes Promoción y Venta 2016.
(3) Asociación de Criadores de Cabras de Raza Palmera.

Objetivos

Incrementar el nivel de digitalización de la Asociación de Criadores de Cabras de Raza Palmera.
Mejorar la eficiencia en la recogida de datos en campo: nacimientos, bajas, altas de animales, calificaciones morfológicas, mediante una app muy fácil de usar tanto para el ganadero como por los técnicos de la asociación.

Desarrollo

1º Análisis del diagrama de flujo de procesos de la asociación

2º Propuesta de mejora y digitalización de los procesos, añadiendo funciones novedosas como la lectura a través de cámara de los identificadores del animal

3º Desarrollo de una aplicación móvil que permita al ganadero recoger los datos de nacimientos y bajas de sus animales, que relacione la maternidad y paternidad de los animales nacidos a través de un identificador, y al técnico digitalizar los procesos de inscripción de animales y calificación morfológica de las cabras.

4º Desarrollo de un panel web de administración para la consulta y modificación de cualquier dato comunicado con el mismo sistema de información que la app.

Tecnologías usadas. Aplicación híbrida (ionic), Angular y Firebase RealTime Database. Especial mención al reconocimiento de texto en imágenes con técnicas de machine Learning.

Resultados

Como resultado ya está operativa una aplicación que reduce considerablemente el tiempo empleado para los procesos ordinarios que realiza la Asociación y sobre todo garantiza una mayor calidad de los datos recogidos tanto por el ganadero como por los técnicos de la Asociación.

Financiación

Proyecto financiado a través del Programa de Cría de la Raza caprina Palmera. Orden de 18 de diciembre de 2018 del Gobierno de Canarias.

Financia

Gobierno de Canarias

Desarrollado por

squaads. Gescan accrp

Organizadores

IGA International Goat Association

X Foro Nacional del Caprino
30 y 31 de Mayo 2019
Antequera

La organización del X Foro Nacional de Caprino celebrado en la ciudad de Antequera (Málaga) los días 30 y 31 de Mayo de 2019:

CERTIFICA

Que el siguiente trabajo científico-técnico ha sido presentado en la sesión de poster de dicho Foro:

DESARROLLO DE APLICACIÓN MÓVIL PARA GESTIÓN DE DATOS EN CAMPO

Y cuyos autores han sido:

Muñoz-Mejías E.; Delgado J.; González J.L.; Hernández B.

Y para que así conste y surta los efectos oportunos se firma la presente en Córdoba a 6 de Junio de 2019.

D. Manuel Gutiérrez Vázquez
Presidente de Cabrandalucia

Dª María Eva Muñoz Mejías
Representante de la Asociación Internacional de Caprino en España