E.T.S. de Ingeniería Industrial,
Informática y de Telecomunicación

Desarrollo de nuevas utilidades para el Programa de vigilancia, control y erradicación de las Encefalopatías Espongiformes Transmisibles (EETs)

Grado en Ingeniería Informática

Trabajo Fin de Grado

Alumno: Eneko Lana Maquírriain
Tutora: Edurne Barrenechea Tartas
Pamplona, 18 de Junio de 2018

upna Universidad Pública de Navarra
Nafarroako Unibertsitate Publikoa

ÍNDICE

1. INTRODUCCIÓN

En este documento se va a describir detalladamente el Trabajo Fin de Grado (TFG) realizado durante las prácticas curriculares de ingeniería informática en la empresa Nasertic. En concreto, el desarrollo de tres proyectos centrados en agilizar la gestión de muestras animales que realizan en el laboratorio de dicha empresa.
Los dos primeros proyectos tienen como objetivo la actualización de los sistemas de registro de muestras utilizados durante la recogida de las mismas en los distintos mataderos de Navarra; mientras que el tercero, tiene como finalidad la automatización de la generación de informes con los datos de las muestras tras su análisis.

Nasertic es una empresa pública, que forma parte del grupo de empresas CPEN (Corporación Pública Empresarial de Navarra). El abanico de clientes y funciones de Nasertic es muy amplio, algunas de estas funciones son: instalación y mantenimiento de antenas de telecomunicaciones en montes navarros, servicio de instalación y asistencia del software oficial del Gobierno de Navarra, análisis de aguas tierras y vegetales, análisis de animales sacrificados en mataderos, etc.

Los proyectos de software descritos en este documento han sido realizados para facilitar los procesos de recogida de muestras y generación de informes relacionados con esta última función de análisis de animales. Estos análisis realizados por el laboratorio de Nasertic, tienen como objetivo la detección de Encefalopatías espongiformes transmisibles (EETs), más conocidas como enfermedad de las vacas locas en el caso de la infección en bovinos.

Estos análisis comenzaron a realizarse a raíz de la epidemia que se dio alrededor del año 2000, cuando se comenzaron a dar casos de la enfermedad en humanos. Durante los años que siguieron, se analizaba el 100% de los animales que cumpliesen con ciertos criterios de edad o sintomatología, en la actualidad se ha reducido el número de animales analizados, realizando estos análisis a una muestra aleatoria de animales que cumplen los criterios.

Desde el Gobierno de España se exigen diversos informes periódicos con los resultados de los análisis, además de estar establecido el procedimiento de erradicación del foco de infección en caso de encontrar animales cuyo análisis de un resultado positivo. Cada año se emite un informe público en el que se muestran los resultados de los análisis en las distintas provincias de España [1].

En la Figura 1 tenemos un mapa sacado del informe anual de 2017, en el que vemos que en la actualidad se siguen dando focos de esta enfermedad en explotaciones ganaderas por todo el país.

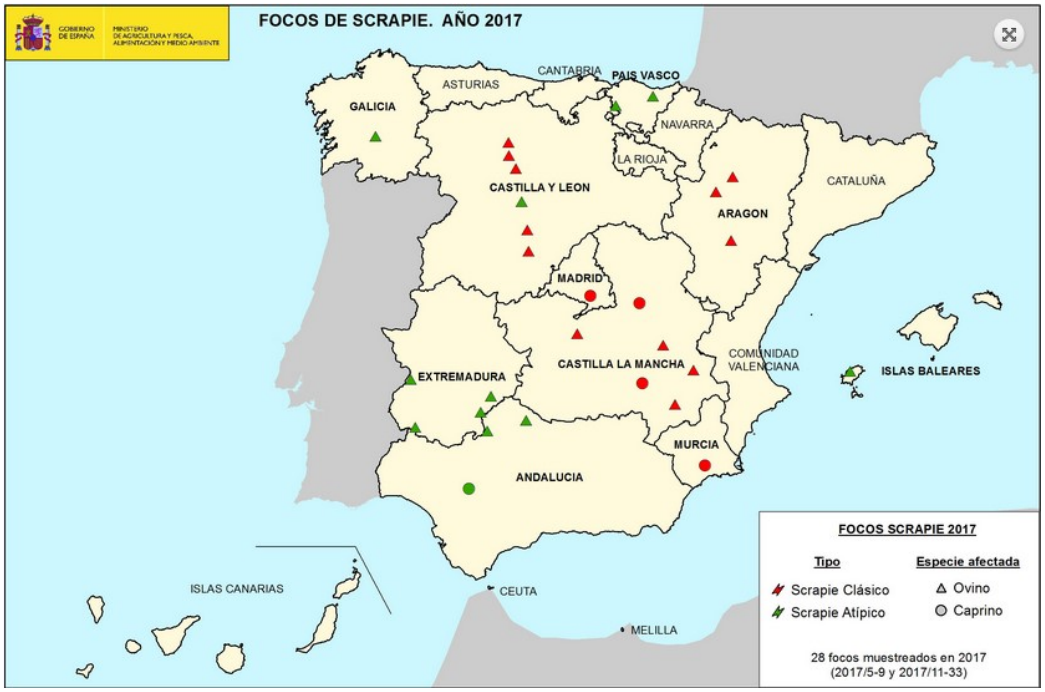

Figura 1. Mapa focos EETs 2017

2. TECNOLOGIAS UTILIZADAS Y ENTORNO DE DESARROLLO

2.1 Access y VBA

Tanto la aplicación de recogida de muestras del portátil, como la aplicación de gestión de muestras para generación de informes, han sido desarrolladas en Access 2003, haciendo uso de la herramienta Visual Basic para Aplicaciones (VBA), que viene integrada en la mayoría de programas de Microsoft Office (Access, Excel, Word…).

Utilizamos Access para gestionar todas las tablas de las distintas bases de datos, así como la mayoría de las consultas sobre estas tablas, además de las vistas de la aplicación (llamadas formularios), y de la vinculación de los eventos que se dan en estas vistas con las distintas macros (o funciones) que se generan mediante código desde VBA.

Se ha desarrollado en Access 2003 porque se va a trabajar sobre aplicaciones desarrolladas con anterioridad en esta misma plataforma, por lo que no ha habido opción de utilizar un entorno más moderno.

Para entender de qué tipo de eventos estamos hablando pongo varios ejemplos sencillos: cargar un formulario, hacer clic en un botón, modificar un campo de texto…

A continuación mostramos unas imágenes de Access para explicar tanto su interfaz como su funcionamiento.

En la Figura 2 tenemos a la izquierda las listas de tablas (tanto las propias de esta aplicación, como las tablas vinculadas desde una base de datos externa a la aplicación, indicadas con una flecha a su izquierda) y consultas de la base de datos, formularios… A la derecha tenemos el modo de edición de una de las tablas, vemos en la parte superior todos los campos de la tabla, y en la parte inferior los distintos parámetros del campo seleccionado.

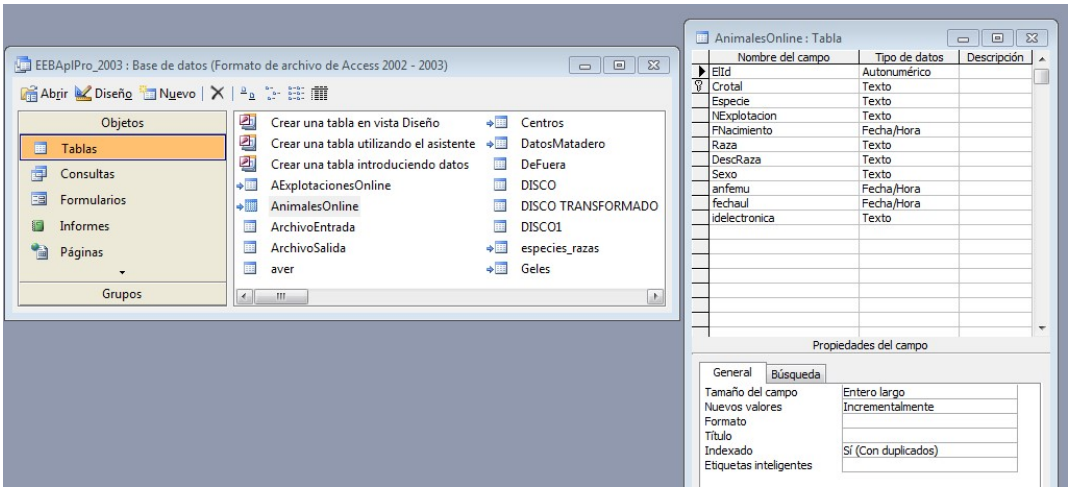

Figura 2. Interfaz de Access

A continuación, en la Figura 3, mostramos una consulta en el modo de diseño, en la parte superior podemos ver la consulta de modo visual. En este ejemplo, observamos tres tablas relacionadas por dos operadores de unión (representados por las líneas que unen las tablas).

En la parte central de la figura tenemos los campos que van a estar en la consulta que se va a realizar, es decir, en la sentencia SELECT. Dichos campos son atributos que se van a mostrar o que son necesarios en las condiciones de dicha consulta, es decir, en el select o en el where (en este caso todos están en el select, por tener el checkbox marcado, y el campo estado estará en el where, por tener un criterio establecido).

En la parte inferior de la figura tenemos el código SQL que se genera a partir de la interfaz visual que se acaba de describir; cualquier modificación en uno de los dos modos de visualizar la consulta se verá reflejada en el otro.

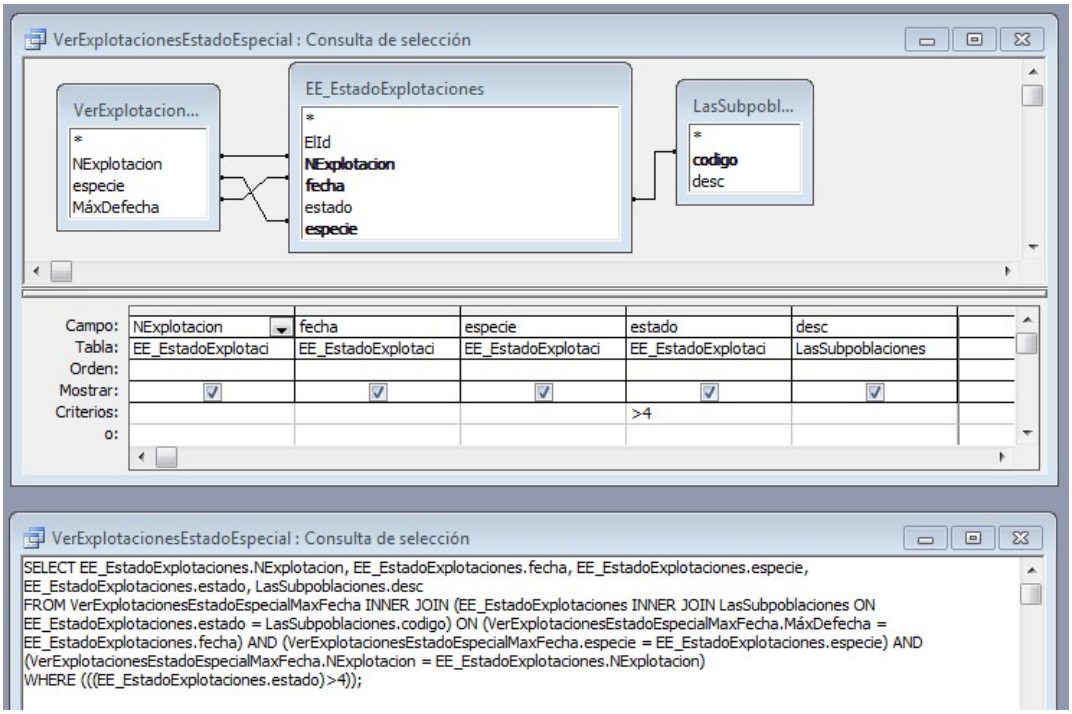

Figura 3. Interfaz de Access

Finalmente, en la Figura 4 visualizamos un formulario en el modo de edición a la izquierda, donde vemos varios textos y botones colocados, estos objetos se pueden mover con facilidad por el formulario, simplemente arrastrándolos. A la derecha tenemos las propiedades de un botón, desde donde modificamos sus parámetros (texto, tamaño, función que se ejecuta al hacer clic…) y bajo estos parámetros vemos el cuadro de herramientas que se utiliza para añadir nuevos objetos al formulario.

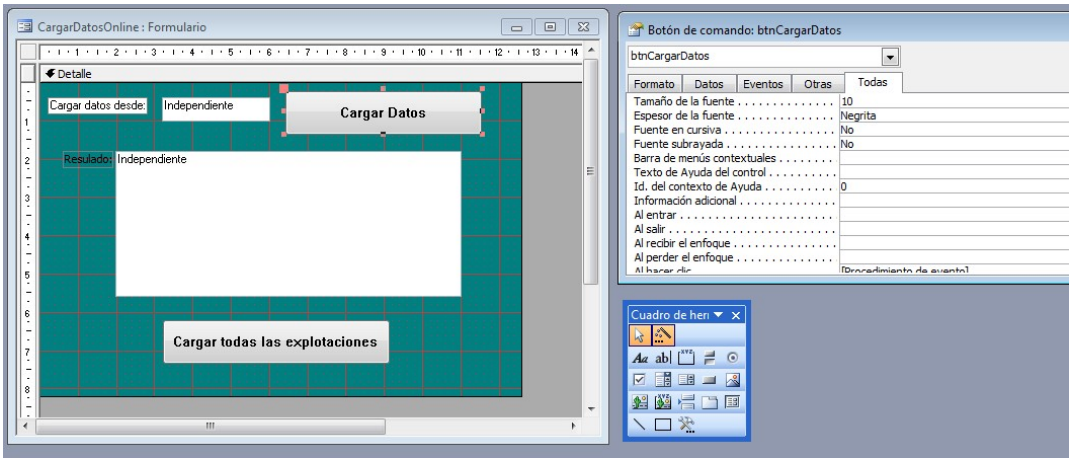

Figura 4. Interfaz de Access

VBA es un dialecto del lenguaje BASIC, lenguaje de alto nivel que surgió en 1964 para iniciar a la programación a estudiantes, que ha evolucionado hasta el punto de tener algunas variantes que han sido muy populares a lo largo de los años, e incluso a día de hoy, como Visual Basic .Net por ejemplo.

VBA es un lenguaje de programación orientado a eventos. Esta denominación se debe a que el flujo del programa está determinado por los distintos eventos que se van a disparar desde los formularios de Access, siendo estas interacciones con los eventos de los formularios, la única forma de ejecutar fragmentos de código. Otro aspecto a destacar de este lenguaje es que al estar desarrollado por Microsoft, ofrece muchas facilidades a la hora de utilizar la API propia de Windows.

Ejemplo de código VBA (Wikipedia)

```
Public Function BUSINESSDAYPRIOR(dt As Date) As Date
    Select Case Weekday(dt, vbMonday)
    Case 1
        BUSINESSDAYPRIOR = dt -3
    Case 7
        BUSINESSDAYPRIOR = dt -2
    Case Else
        BUSINESSDAYPRIOR = dt -1
    End Select
End Function
```

2.2 PHP

La aplicación de Access de recogida de muestras, utiliza un servicio web para actualizar la copia de la base de datos que se almacena para tener los datos de los animales cuando no tenemos acceso a internet. Estos Web services se han programado en PHP.

PHP es un acrónimo recursivo de Hypertext Preprocessor (procesador de hipertexto). Se trata de un lenguaje de propósito general del lado del servidor. Por tanto, el cliente no tiene acceso al código PHP, este simplemente realiza una petición al servidor y este ejecuta un script PHP y envía el resultado de este script al cliente.

En los PHPs desarrollados para esta aplicación, se ha utilizado mssql para acceder a la base de datos (Microsoft SQL Server), y la respuesta de las peticiones se sirve al usuario en formato JSON, utilizando la librería VBA-JSON [2] para realizar la conversión de forma simple. A continuación, mostramos un ejemplo de cómo se mandarían los datos de un animal en este formato:

{"listaOvino":[{"crotal":"ES120000123456","explotacionrega":"ES120000012345","especie":"03","fechanacimiento":"2006-01-01 00:00:00","codigoraza":"3128","descraza":"NAVARRA","sexo":"Hembra","anfemu":"2018-05-14 00:00:00","fechaul":"2018-05-14 10:33:33","idelectronica":"10040000724140000285926"}]}

Los PHPs están alojados en la web y se les ha añadido un sistema de seguridad sencillo para que los datos no estén a disposición de cualquiera que conozca la ubicación de los mismos.

El sistema consiste en utilizar uno de los parámetros de la consulta (la fecha por ejemplo) y realizar una operación con este parámetro y una clave que se encuentra en el código del PHP, y comprobar si el hash md5 del resultado de la operación se corresponde con el código recibido en la petición. Con este sistema, si alguien intercepta un paquete, solo podría replicar la petición que ha interceptado, y no podría realizar una petición con una fecha distinta, a no ser que tenga acceso al código de la aplicación, y por lo tanto pueda ver como se ha generado el hash md5.

2.3 NODE.JS

Node.js es un entorno multiplataforma, de código abierto, orientado principalmente a usarse en la capa del servidor. Es un lenguaje asíncrono, orientado a eventos, que fue desarrollado principalmente para la creación de servidores web altamente escalables.

En este proyecto, se ha utilizado Node para desarrollar el servidor que se ejecuta en el portátil para la recogida de muestras. En concreto, la función de este servidor es gestionar las peticiones que se realizan desde el dispositivo móvil utilizado para la recogida, ejecutando en la base de datos Access alojada en el portátil las inserciones y actualizaciones necesarias.

A continuación, mostramos algunas de las ventajas de utilizar Node frente a otras alternativas, como podría ser por ejemplo Apache + PHP.
En primer lugar, el lenguaje en sí mismo puede considerarse una ventaja, ya que al ser javascript el lenguaje utilizado para desarrollar el servidor Node, nos encontramos con sistemas en los que la parte del cliente y el servidor están desarrolladas en javascript; por tanto, se puede facilitar su integración en algunos casos, además de permitir reutilizar el código sin ninguna modificación.

Instalar un servidor en Node es muy sencillo, no requiere de ninguna configuración previa más allá de instalar Node en el dispositivo que vaya a ser nuestro servidor, independientemente del sistema operativo de este. Además, Node cuenta con una buena comunidad de desarrolladores en la web, que ofrece tutoriales muy buenos y actualizados [3], así como un gestor de paquetes muy completo (npm) que facilita la instalación y actualización de librerías externas.

Node utiliza un solo hilo para gestionar sus eventos asíncronos, esta forma de funcionamiento tan distinta a lo acostumbrado en un servidor web, hace más sencilla la programación de aplicaciones que tienen que actuar en tiempo real, a la vez que da al programador más posibilidades y más control sobre las peticiones web recibidas.

El servidor es capaz de soportar muchas más peticiones de forma concurrente en comparación con Apache, precisamente por esta forma de trabajar en un solo hilo con eventos asíncronos.

Sin embargo, no todo son ventajas, Node presenta algunos inconvenientes, por ejemplo, al ser un lenguaje tan nuevo, algunas actualizaciones pueden contener cambios muy grandes, llegando en algunos casos a que el código de una aplicación de Node deje de funcionar en versiones posteriores. Otro aspecto negativo de Node es que no es eficiente a la hora de realizar tareas que requieran de un uso intensivo de CPU, ya que existe un bucle principal que controla los eventos del sistema corriendo en un único hilo, lo que unido a altos usos de CPU puede producir bloqueos.

Finalmente, el tener un único hilo gestionando todas las peticiones también puede ser un problema grave. Si una de estas peticiones provoca un error de ejecución en el servidor Node, al no estar gestionándose en un hilo aparte, todo el servidor Node se va a detener, y aunque hay varias formas muy sencillas de volver a ejecutar Node automáticamente cuando se produce el error, todas las variables almacenadas en tiempo de ejecución se perderán.

2.4 IONIC, CORDOVA Y ANGULAR

Ionic es un kit de desarrollo de código abierto para desarrollo móvil multiplataforma, se trata de una tecnología muy joven, publicada en 2013. Ionic fue desarrollado utilizando Angular 2 y Apache Cordova, y además necesita de Node.js para funcionar.

Angular 2 es un Framework para desarrollo de aplicaciones web desarrollado en typescript, Angular está desarrollado por Google y es de código abierto, y destaca especialmente en la creación de webs de una sola página, aunque no se limita a este uso.

Apache Cordova es un entorno de desarrollo móvil, los lenguajes que ofrece son CSS3 y HTML 5 para las vistas, y javascript para la lógica, además ofrece librerías y facilidades para hacer uso de las distintas herramientas de los dispositivos móviles como son GPS, cámara, gestión de conexiones wii, etc.

El funcionamiento de Ionic es similar a Cordova en esos aspectos, con la principal diferencia de que el lenguaje utilizado para la lógica de la aplicación es typescript en lugar de javascript.

Internamente, para permitir al usuario el desarrollo en HTML y CSS, lo que hace Ionic es integrar una vista web en la aplicación generada, y comunicar este vista con la parte lógica de la app, de esta forma consigue funcionalidades muy prácticas, como una vista web actualizada en tiempo real con las variables de las clases de typescript.

Para facilitar el desarrollo de estas interfaces web, la página oficial de Ionic ofrece ejemplos de uso de todos los tipos de controles que suelen aparecer comúnmente en una interfaz móvil [4].

Como aspecto negativo de Ionic habría que destacar que al ser un entorno multiplataforma, que funciona sobre plataformas completamente distintas (PC, android, ios…), algunas de las herramientas que hacen uso de las herramientas nativas de uno de estos sistemas operativos, generan errores cuando se ejecutan en otro sistema operativo, por lo que el programador tiene que incluir comprobaciones para controlar la ejecución de ciertas líneas de código en función del sistema que se esté ejecutando.

3. OBJETIVOS

En esta sección vamos a analizar los diferentes problemas que se dan en el laboratorio de Nasertic, y qué medidas se van a tomar para solventarlos.

3.1 Automatización de lectura de datos durante la recogida de muestras

Cuando un animal bovino u ovino es sacrificado en un matadero en Navarra, un veterinario de Nasertic se dirige al matadero para tomar una muestra de ese animal y analizarla en busca de indicios de encefalopatías espongiformes.

El problema que nos encontramos en este proceso de recogida es el siguiente, el veterinario que realiza el registro de la muestra, introduce en un programa el número de identificación del animal (Denominado comúnmente como crotal), pero lo hace mediante un portátil que no tiene acceso a internet, por lo que en el lugar de la recogida no dispone de los datos del animal.

No disponer de los datos del animal en el lugar de recogida da lugar a dos problemas, el primero de ellos es que para que el posterior análisis sea válido, el animal tiene que ser mayor de una determinada edad, si el animal es menor, la muestra no debe ser tomada. Por lo tanto, cuando tienen duda sobre si la edad del animal es correcta, el veterinario tiene que llamar por teléfono a las oficinas del laboratorio para que alguien, utilizando un servicio web, introduzca el crotal del animal en cuestión, y determine si la edad es correcta o si la muestra no debe ser recogida.

El segundo problema es que una vez de vuelta en las oficinas, el veterinario tiene que utilizar el servicio web anteriormente mencionado para obtener ciertos datos del animal, y de ahí pasarlos a mano al programa del portátil, desde el que posteriormente se exportaran los datos de las muestras a la base de datos del laboratorio.

Cualquier solución que implique depender de conexión internet en el lugar de recogida de muestras queda descartada, pues cabe la posibilidad de que en estas zonas no tengamos cobertura.

Por lo tanto, la solución a implementar consiste en mantener en el portátil una copia de los datos necesarios de la base de datos que utilizan los servicios web (base de datos perteneciente al gobierno). De esta forma, en cuanto se introduzca el código de un animal en la aplicación del portátil, tanto con lector de códigos de barras como a mano, todos los datos del animal se completen automáticamente. Este funcionamiento, implica no tener que introducir a mano estos datos, y además, al disponer también de la edad del

animal en los mataderos, en cuanto un animal menor de edad es registrado, el programa informa al veterinario de la incidencia.

Este cambio se va a llevar a cabo a la par que una actualización de los equipos portátiles, ya que tienen aproximadamente 10 años. Seguramente por el hecho de estar completamente aislados (excepto por el uso de un USB para exportar las muestras de los mismos) han llevado el paso del tiempo bastante bien, en especial, teniendo en cuenta el tiempo que pasan en los mataderos, y la suciedad de la que están rodeados cuando están en ellos.

Los sustitutos de estos portátiles antiguos van a ser unos portátiles especiales, especialmente por su resistencia y que van a permitir ser lavados periódicamente. Por otra parte, van a estar provistos de más medidas de seguridad puesto que estos sí que van a disponer de conectividad a internet.

3.2 Mejora del sistema de lectura electrónica para la recogida de muestras

En cuanto a los dispositivos electrónicos utilizados para la lectura de la identificación del animal de forma automática, actualmente utilizan una pistola lectora de códigos de barras, conectada mediante un cable USB al portátil que tiene el programa de registro de muestras.
En algunas ocasiones esto es un problema, debido a la falta de espacio el veterinario que realiza la recogida no puede colocar el portátil siempre donde él quiere, y el animal no queda a rango del cable, por lo que tiene que ir apuntando los crotales en un papel para después pasarlo a mano al portátil.

Para solventar este problema se plantearon dos posibles soluciones. La primera de ellas era sustituir el lector de códigos de barras por uno inalámbrico, la segunda solución, consistía en realizar una aplicación para móvil, para realizar la lectura de crotales desde el móvil.

Se eligió la segunda opción ya que la ventaja más clara que se le vio a esta opción frente a la otra, es la mayor versatilidad y personalización que presenta frente a la simplicidad de utilizar la pistola inalámbrica. Por ejemplo, si lees un animal menor de edad, con la pistola lectora de códigos de barras no puedes informar al usuario, y si este está lejos del portátil podría no enterarse del problema, mientras que desde el móvil se le puede mostrar un aviso sin problemas.

Además se planteó la posibilidad de, en un futuro, poder prescindir del portátil y migrar todas las funcionalidades de la aplicación del portátil al móvil, pero se decidió que por el momento la aplicación dependiese del portátil, comunicándose con este utilizando la funcionalidad de abrir una zona WIFI que tienen todos los dispositivos móviles. Esta

decisión de depender del portátil se debe a que para el correcto funcionamiento de la aplicación necesitamos una copia de la base de datos que contiene los datos de los animales, y en esta base de datos, la tabla que contiene la información de los animales tiene un número de registros muy alto.

También encontramos otro problema relacionado con la lectura electrónica y ocurre únicamente en animales ovinos y caprinos. Surge porque en estos animales es muy común que los crotales, que los llevan colgando de las orejas, acaben perdidos porque se los quitan entre ellos por algún mordisco en las orejas.

Por otra parte, las ovejas además de los crotales tienen un segundo dispositivo de identificación, se trata de una bola de cerámica que tragan y queda almacenada en su estómago. Estas bolas de cerámica, llamadas bolos, tienen un dispositivo capaz de emitir el código electrónico que identifica al animal a corto alcance.

Se propuso como solución al problema de los crotales perdidos identificar a las ovejas mediante un aparato capaz de leer estos bolos, pero tras hacer unas preguntas para ver si se trataba de una solución viable, se tuvo que descartar, pues en las principales explotaciones de Navarra, los animales no llevan bolo y además estos daban problemas de interferencias.

No se ha podido solucionar el problema con estos animales que no tienen ni crotal ni bolo, siendo necesario registrar el animal metiendo a mano su código de explotaron, y generando un código único para identificar la muestra.

3.3 Generación automática de informes mensuales

Tras el proceso de recogida de muestras, los datos se exportan desde el portátil a la base de datos del laboratorio, En este proceso es donde entra en juego la aplicación que gestiona las muestras durante y después del proceso de analizarlas.

Una vez al mes, el laboratorio de Nasertic, debe generar y enviar varios informes sobre los animales que han sido analizados durante dicho mes. Estos informes están compuestos por varios ficheros. Primero tenemos tres tablas (una para cada tipo de animal, bovino caprino y ovino) con los números de animales que cumplen ciertas condiciones. Por ejemplo, animales que se van a exportar fuera de la comunidad foral, o animales pertenecientes a explotaciones con mayor riesgo de enfermedad, determinando este riesgo en función de casos de análisis positivos en los últimos años en esa misma explotación.
Además de estas tablas, los informes contienen varias listas de animales. En concreto, los animales que cumplen ciertas condiciones poco comunes, como pueden ser, animales menores que una edad concreta, o animales que han sido sacrificados de urgencia.

Todos estos informes, con la única excepción de la tabla de animales bovinos, son generados a mano por un trabajador del laboratorio, que tiene que mirar animal por animal, sumándolo en la casilla de la tabla que corresponda, y añadiéndolo a una de las listas de animales especiales si fuese necesario. Si consideramos que el número de animales analizados al mes ronda los 300 o 400 animales, vemos que el volumen de trabajo que supone generar estos informes a mano es muy alto, alrededor de dos días de trabajo mensual. Añadiendo las funcionalidades adecuadas al programa de gestión de muestras, este trabajo se podría realizar en menos de una hora. Por todo ello, estas funcionalidades son precisamente las que se van a implementar y describir a lo largo de este proyecto.

4. ANTECEDENTES

Lo primero que cabe destacar sobre las aplicaciones que van a ser modificadas (tanto la aplicación de recogida de muestras como la de gestión de muestras y análisis) es que ambas aplicaciones fueron desarrolladas hace algo más de 15 años, y en el transcurso de estos años las necesidades que cubrían estas aplicaciones han evolucionado. Por lo que muchas de las funcionalidades de estas aplicaciones han dejado de ser necesarias.

Además, los cambios que se han realizado a estas aplicaciones durante estos 15 años, han sido cambios que se han hecho de forma rápida y sin ninguna consideración por la mantenibilidad ni la legibilidad del código. Por todo ello, nos hemos encontrado clases enteras que no tienen ni un solo comentario explicando el código, campos redundantes en las tablas, campos cuyo nombre no tiene nada que ver con lo que representar pues los criterios de clasificación han cambiado, entre otras cosas.

5. METODOLOGÍA

En el desarrollo de este proyecto la metodología seguida es un punto intermedio entre el proceso unificado y la programación extrema, ya que por una parte, el proceso de desarrollo ha estado guiado por los casos de uso y los requisitos funcionales derivados de los mismos. Pero por otra parte no se ha seguido el modelo de desarrollo iterativo propio del proceso unificado.

Las características propias de la programación extrema que se han dado durante el desarrollo han sido la comunicación constante con el cliente, y la modificación sobre la marcha de algunos requisitos funcionales.

En cuando al sistema de control de versiones, se utiliza un método rudimentario, todos los proyectos están almacenados en las unidades de red a las que se tiene acceso en la oficina, y cuando un desarrollador realiza cambios en un proyecto, hace una copia del mismo, indicando en el nombre del archivo a que fecha corresponde la modificación.

En cuanto al sistema de publicación de versiones de cara a los usuarios del laboratorio, estos tienen un fichero .bat para la ejecución de cada programa, cuya ejecución realiza una copia a su ordenador de la versión que se encuentra en la carpeta de programas publicados de la unidad de red del laboratorio.

Durante el proceso de desarrollo, la comunicación con el cliente ha sido directa e interrumpida. La primera reunión fue la más extensa de todas, en ella se explicó como trabajaban hasta ahora en el laboratorio, los problemas que tenían, y los requisitos que debía cumplir el programa encargado de solventar esos problemas. A lo largo del desarrollo además se realizaron numerosas llamadas telefónicas, y se mandaron muchos correos electrónicos para realizar aclaraciones y discutir algunas mejoras que surgieron sobre la marcha.

Al finalizar el proceso de desarrollo, se organizó una reunión final, en la que se realizó una demostración de las funcionalidades del programa, se dieron las explicaciones de uso que se consideraron necesarias, y se mostró el manual de usuario en el caso de la aplicación móvil. Después, los clientes evaluaron brevemente el producto e indicaron su grado de satisfacción con el mismo.

6. ANÁLISIS

6.1. REQUISITOS FUNCIONALES

En esta sección vamos a describir los requisitos funcionales concluidos a partir de las reuniones con los clientes.

6.1.1 Aplicación recogida de muestras (portátil)

- Se podrán descargar a una base de datos local los datos de los animales y explotaciones que hayan sido modificados después de una fecha dada.

- Se podrán descargar a una base de datos local los datos de todas las explotaciones.

- Se debe almacenar la fecha hasta la que los datos están actualizados.

- Se podrán consultar los datos del animal y su explotación de cualquier muestra ya registrada.

- Se podrán consultar los datos de todos los animales de una explotación determinada.

- Al registrar una muestra por su número de crotal, se deben rellenar automáticamente todos los datos del animal, y en caso de no disponer de estos datos, mostrar una alerta al usuario.

- Se debe permitir al usuario registrar una muestra utilizando la pistola lectora de códigos de barras, en cuyo caso habrá que traducir los crotales al código habitual si el código leído no tiene el formato correcto.

- Se debe permitir al usuario registrar una muestra indicando su código de explotación, en estos casos se generara un número de crotal automáticamente en función de la fecha actual y el número de sacrificio.

- En el caso de registrar un animal con una edad inferior 48 meses en el caso de los bovinos, y de 18 meses para ovinos y caprinos, se debe mostrar una alerta al usuario, y cancelar el registro de la muestra.

- Se debe poder modificar, si el usuario lo desea, el crotal, número de sacrificio, especie, fecha de nacimiento y código de explotación de un animal ya registrado.

6.1.2 Aplicación recogida de muestras (móvil)

- La lista de centros se debe cargar dinámicamente desde la base de datos del portátil

- El usuario debe poder elegir el centro desde el que se realiza la recogida

- Al registrar una muestra por su número de crotal, se deben registrar la muestra con todos los campos rellenados automáticamente, en caso de no disponer de estos datos, mostrar un formulario al usuario, para que cancele el registro, o lo acepte añadiendo los datos fecha de nacimiento, especie y explotación si lo desea.

- Se debe permitir al usuario registrar una muestra utilizando la cámara del móvil como lector de códigos de barras, en cuyo caso habrá que traducir los crotales al código habitual si el código leído no tiene el formato correcto.

- Se debe permitir al usuario registrar una muestra indicando su código de explotación, en estos casos se generara un número de crotal automáticamente en función de la fecha actual y el número de sacrificio, y se le mostrara un formulario al usuario para que complete los campos fecha nacimiento, especie y explotación si el usuario lo desea.

- En el caso de registrar un animal con una edad inferior 48 meses en el caso de los bovinos, y de 18 meses para ovinos y caprinos, se debe mostrar una alerta al usuario, y cancelar el registro de la muestra.

- La aplicación debe disponer de un sistema de actualizaciones automáticas, que compruebe periódicamente si la versión instalada coincide con la última versión publicada por el departamento de producción.

6.1.3 Aplicación generación de informes mensuales

- La aplicación debe permitir establecer estados para las distintas explotaciones, pudiendo tener cada explotación un estado concreto en un intervalo de fechas determinado, las explotaciones no pueden tener dos estados durante una misma fecha.

- La aplicación debe permitir eliminar estados, para corregir errores si se introduce un estado incorrectamente.

- Se debe poder visualizar un historial con todos los estados por los que ha pasado una explotación.

- Se deben poder visualizar los estados actuales de todas las explotaciones que tengan estados especiales.

- En el caso de los informes de bovinos, se deben poder listar las vacas analizadas entre dos fechas que cumplan una de las siguientes condiciones: ser menores de 48 meses, haber sido sacrificadas de urgencia, haber sido sacrificadas para consumo humano, o presentar sintomatología de enfermedad.

- Para los animales ovinos y caprinos, se deben poder listar los animales analizados entre dos fechas, que pertenezcan a explotaciones con un estado especial, por ejemplo, animales en explotaciones de comercio intracomunitario, o de explotaciones en vigilancia intensificada.

- Además, para los animales ovinos y caprinos, se debe de poder generar un informe en Word, en el que se muestre el número de animales analizados entre dos fechas pertenecientes a cada uno de los estados de las explotaciones, así como que explotaciones están en cada estado, en el ANEXO A se muestra un ejemplo de informe a generar.

6.2. REQUISITOS NO FUNCIONALES

A continuación vamos a ver los requisitos no funcionales deducidos de las reuniones con los clientes.

6.2.1 Aplicación recogida de muestras (portátil)

- Los cambios realizados no pueden afectar al formato de las etiquetas que se van a imprimir, ni al formato de los ficheros de textos generados para la exportación de los datos de las muestras a la base de datos del laboratorio.

- La aplicación debe ser lo más sencilla posible, intentando mantener su simplicidad, o reducirla si fuese posible.

- Los servicios web utilizados por la aplicación no deben ser accesibles sin conocer una contraseña (La contraseña se debe encontrar en el código de la aplicación, y no tiene que ser introducida por el usuario)

6.2.2 Aplicación recogida de muestras (móvil)

- La comunicación entre el portátil y el móvil se debe establecer requiriendo la mínima interacción por parte del usuario.

- La aplicación debe ser lo más sencilla posible, y el registro de muestras debe ser ágil e intuitivo.

6.2.3 Aplicación generación de informes mensuales

- La aplicación debe impedir que un usuario que no tenga permisos elevados tenga la capacidad de eliminar los estados de las explotaciones.

6.3. CASOS DE USO

En la Figura 5 tenemos los casos de uso de las herramientas añadidas a la aplicación de gestión de muestras para la generación de informes.

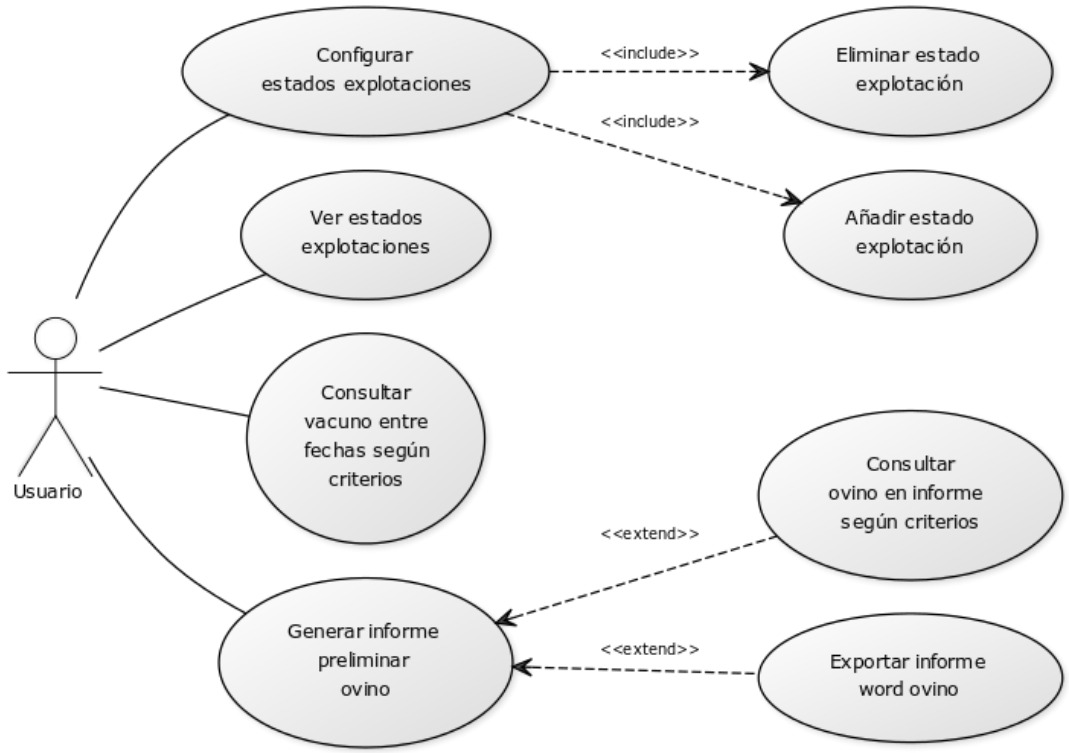

Figura 5. Casos de uso generación de informes

A continuación, Figura 6, mostramos el diagrama de casos de uso de las herramientas añadidas a la aplicación de recogida de muestras

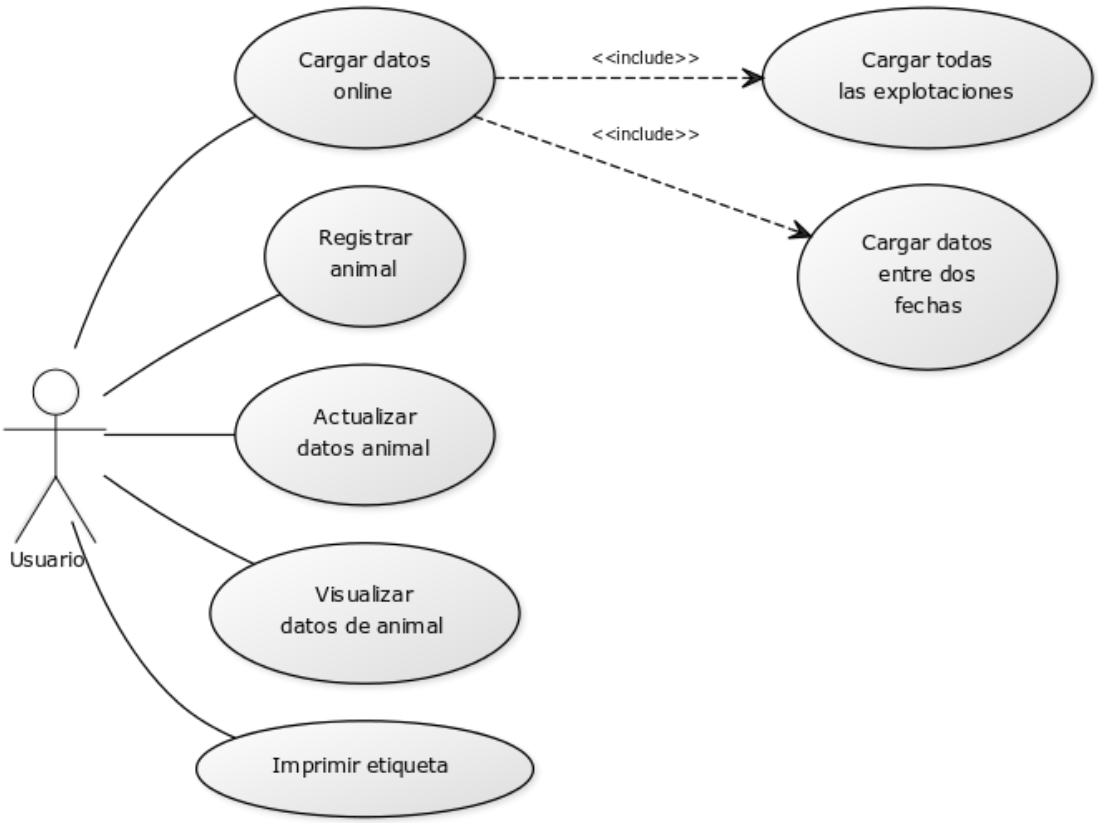

Figura 6. Casos de uso recogida de muestras (portátil)

Vemos en la Figura 7 los casos de uso para la aplicación móvil de recogida de muestras

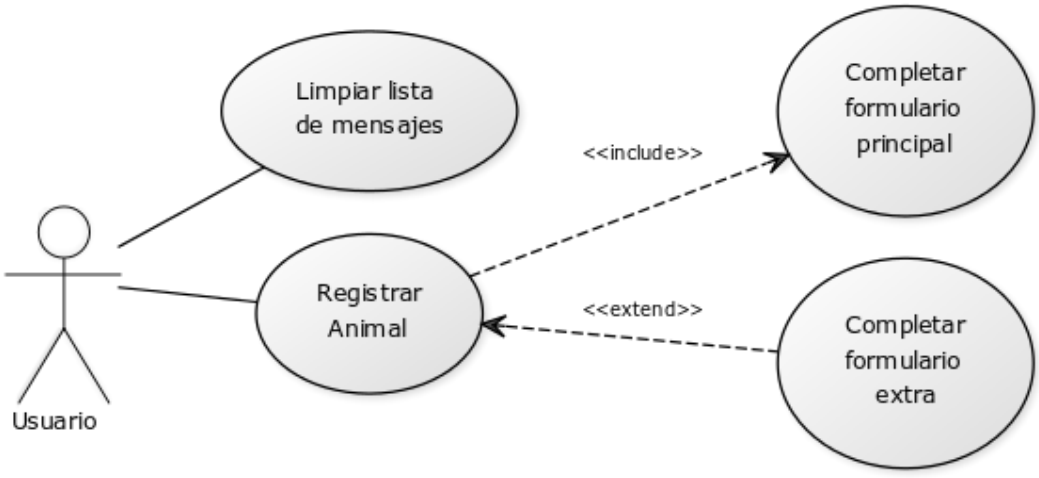

Figura 7. Casos de uso recogida de muestras (móvil)

6.4 MODELO DE DATOS

Durante el desarrollo de estas aplicaciones se han utilizado diversas bases de datos sin realizar apenas cambios en las mismas. Se han creado algunas tablas nuevas pero no se ha modificado ningún diseño de tablas ya existentes.

Las aplicaciones de recogida de muestras, van a hacer uso de la base de datos Access que almacena el portátil en su disco duro. Esta base de datos ya estaba diseñada con anterioridad, y contiene tanto muchas tablas destinadas a funciones de la aplicación que no se van a modificar en este proyecto, como las tablas necesarias para realizar el registro de animales.

Se han tenido que añadir dos tablas a dicha base de datos, estas tablas son las que almacenan los datos descargados desde la web, para tener acceso a los animales cuando el portátil esta en las explotaciones, una de ellas almacena los datos de los animales, y la segunda los de las explotaciones, y ninguna de ellas se ha relacionado con las tablas que ya existían en esta base de datos, pues los datos cargados son independientes de cualquier otro dato que ya existiese en esta base de datos.

La aplicación de generación de informes, hace uso de una base de datos gestionada a través de SQL Server, que almacena la información de todas las muestras que recibe el laboratorio, e información sobre los análisis de las mismas, ha sido necesaria la creación de una tabla que almacena información sobre los estados por los que ha pasado cada explotación de animales. El diseño de esta tabla es muy simple, tenemos el código de la explotación, la fecha en la que la explotación pasa a un determinado estado, y el estado al que pasa.

7. DISEÑO

En esta sección tenemos los distintos diagramas realizados durante el diseño de las aplicaciones, así como los motivos que han llevado a elegir las tecnologías utilizadas. Para la realización de los diagramas se han utilizado diversas páginas web. [5,6,7]

7.1 Diagramas de clases

En cuanto a los diagramas de clases, se han realizado los diagramas de clases necesarios para el desarrollo de la aplicación de recogida en el dispositivo móvil, así pues vamos a tener dos diagramas de clases. El primer diagrama (Figura 8) corresponde con el diagrama de clases de la aplicación desarrollada con Node.js que se alojara en el portátil de recogida y su función consiste en permitir la comunicación de la aplicación móvil con la base de datos del portátil. El diagrama de la Figura 9, es el diagrama de clases de la aplicación del móvil.

No se han realizado diagramas de clases para los proyectos realizados en Access, puesto que la mayoría de los cambios realizados en estos se han hecho en formularios que ya estaban implementados.

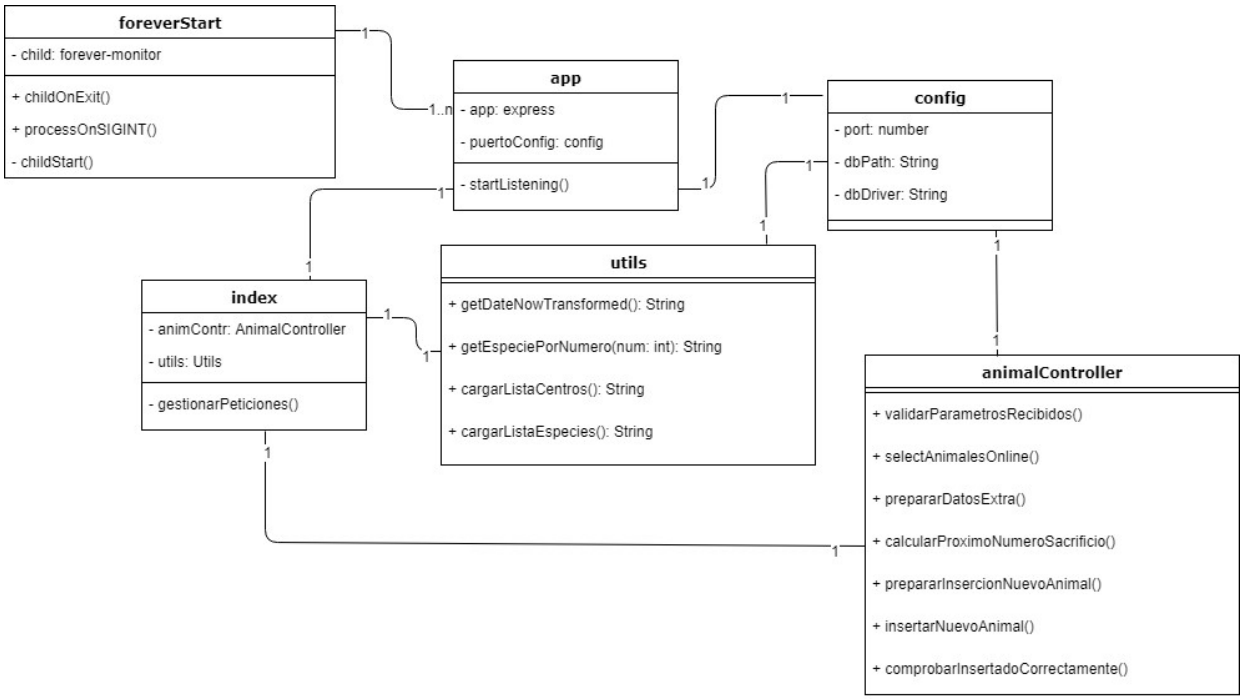

Figura 8. Diagrama de clases servidor Node

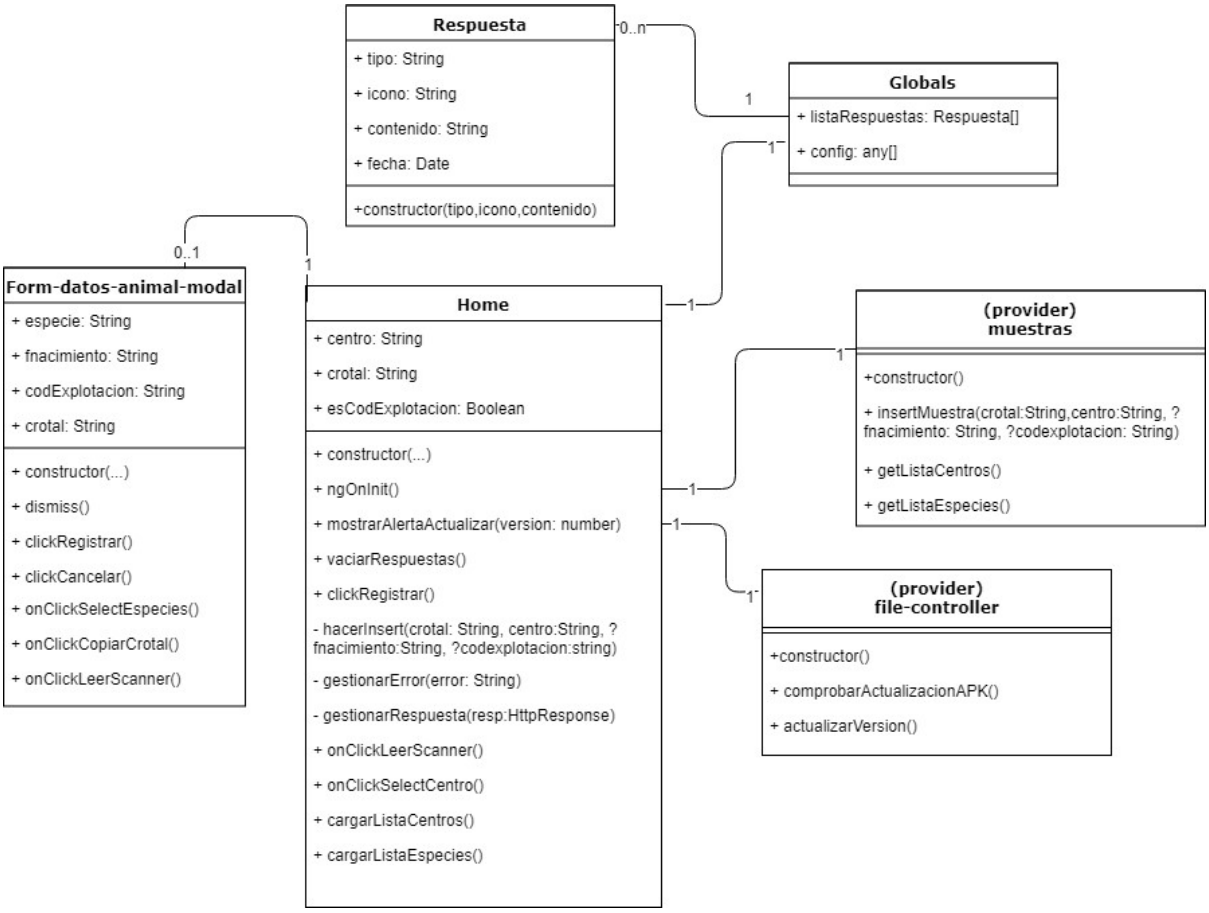

Figura 9. Diagrama de clases aplicación móvil

7.2 Diagramas de Secuencia

A continuación vamos a ver los diagramas de secuencia correspondientes a los casos de uso principales de las aplicaciones desarrolladas.

En la Figura 10, tenemos el diagrama de secuencia para el caso de uso cargar datos online, esto es, para generar la copia de la base de datos en el portátil.

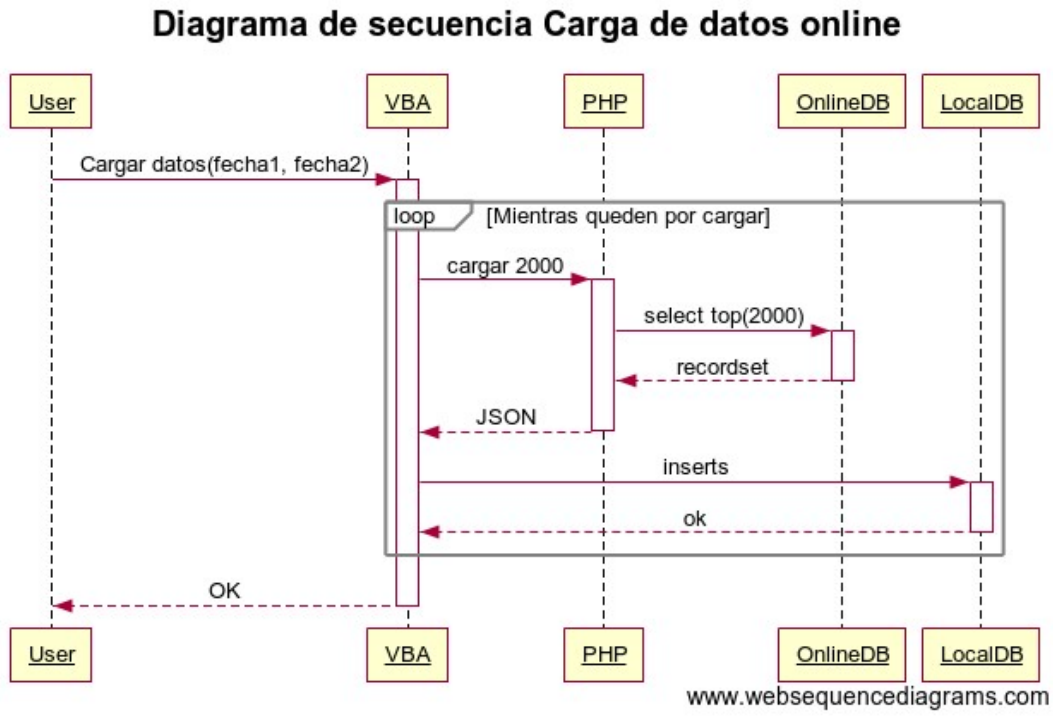

Figura 10. Diagrama de secuencia carga de datos

A continuación tenemos la Figura 11, en la que vemos el diagrama de secuencia para el caso de uso generar informe.

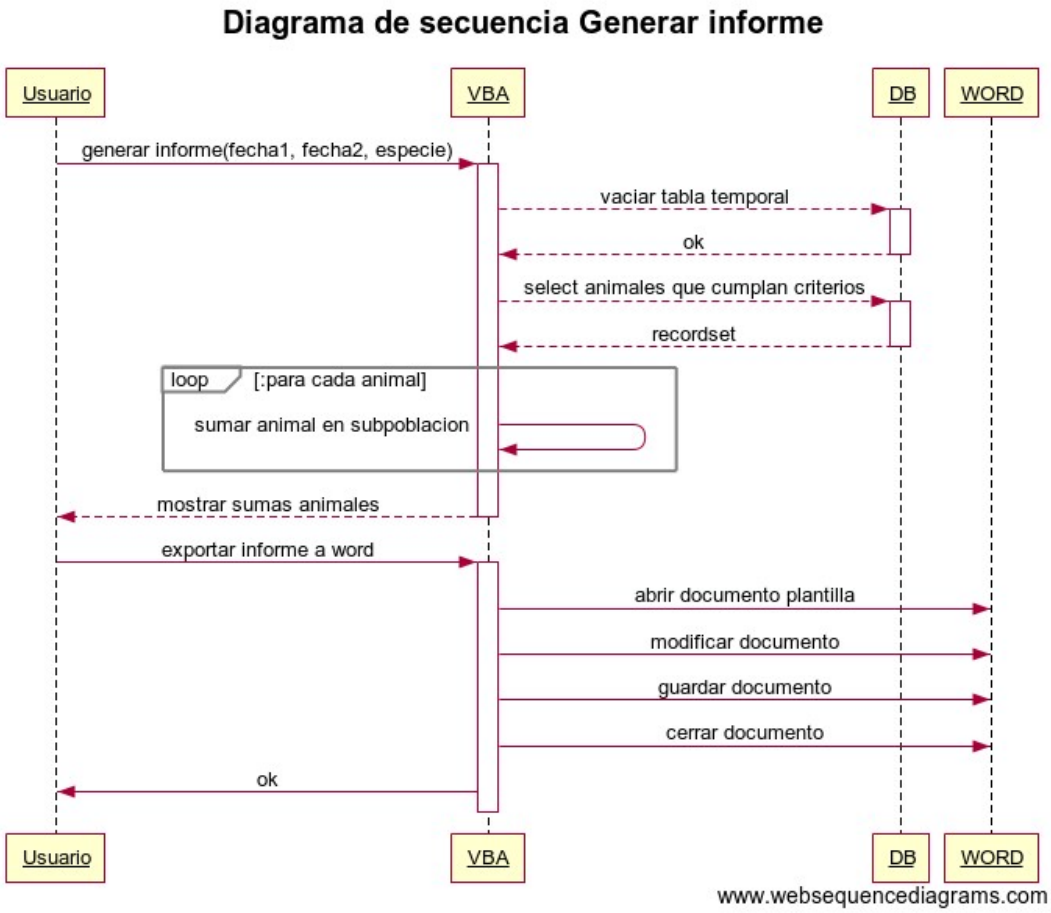

Figura 11. Diagrama de secuencia generar informe

Como último diagrama de secuencia, vemos en la Figura 12 el diagrama correspondiente al caso de uso registrar animal en la aplicación móvil.

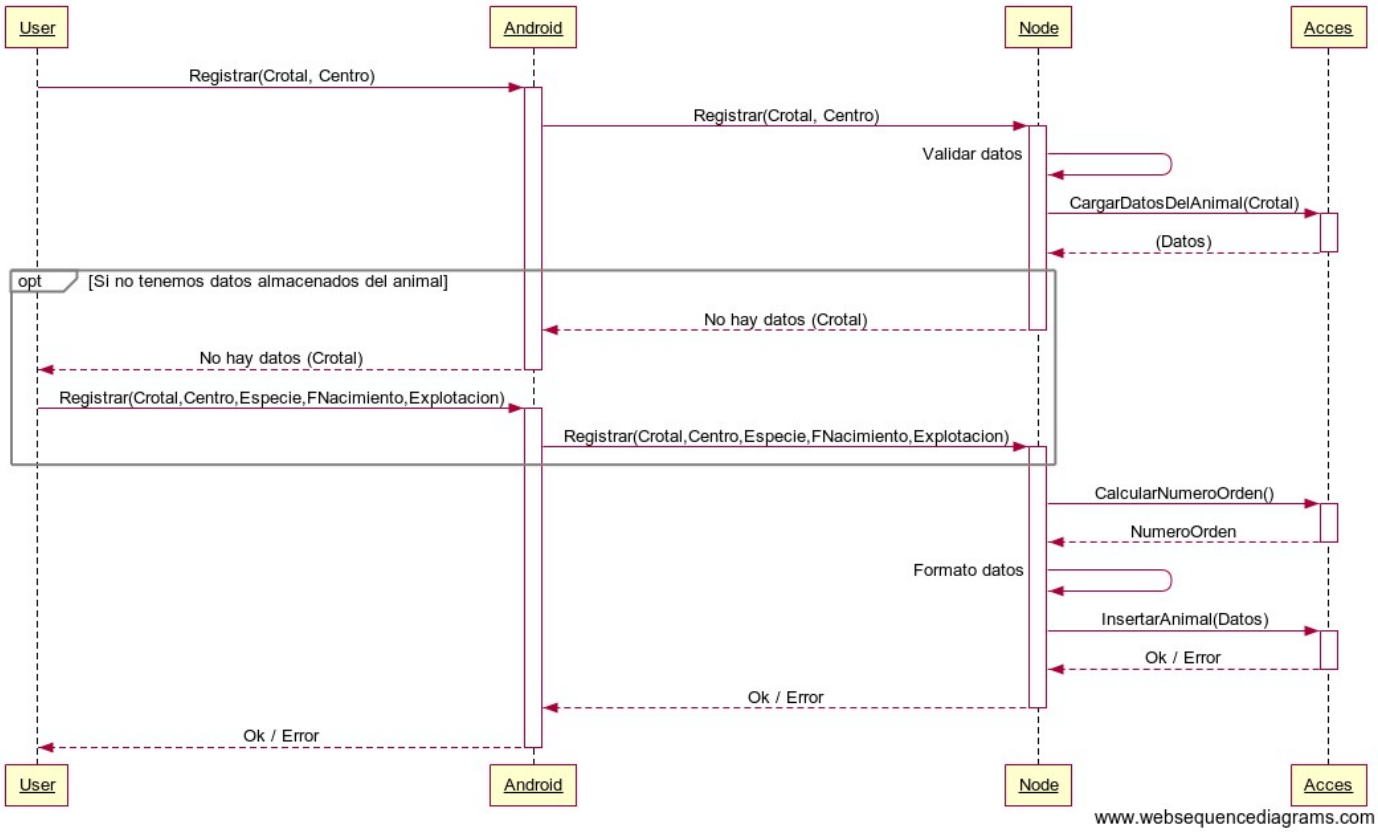

Figura 12. Diagrama de secuencia generar informe

7.3 Elección de tecnologías

En el caso de la aplicación de recogida de muestras para el portátil y la de generación de informes, se ha utilizado Access 2003 y VBA porque se ha trabajado sobre aplicaciones que ya estaban desarrolladas en este entorno.

Sin amargo en la aplicación de recogida de muestras para dispositivos móviles si que se han tomado más decisiones en lo que a las tecnologías se refiere.

Se tomó la decisión de utilizar Ionic y Node.js por los siguientes motivos. Para empezar, se valoró positivamente que los proyectos desarrollados con ambas tecnologías sean multiplataforma, así, si en un futuro se cambian los equipos a otro sistema, como pasar a utilizar un IPhone en vez de un Android, el proyecto se pueda adaptar con poco esfuerzo.

Como segundo motivo, la sencillez. Node.js proporciona una forma muy simple de implementar un servidor web, para instalar el servidor en un ordenador basta con copiar la carpeta con el proyecto, y automatizar la ejecución de un comando de consola en el arranque de Windows.

Un motivo más que lleva a la elección de esta tecnología, consiste en los lenguajes de desarrollo, JavaScript para el servidor y TypeScript en el caso del cliente. La similitud entre los mismos agiliza el proceso de desarrollo, además al estar tan ligadas ambas tecnologías (Ionic está construida sobre Node), en Internet hay mucha información sobre el uso de ambas tecnologías en un mismo proyecto.

La última razón de peso por la que se ha utilizado Node consiste en que el equipo de desarrollo de Nasertic está muy interesado en que Node se acepte en el estándar de software de Gobierno, y el desarrollo de esta aplicación con la herramienta, sirve como demostración de las capacidades de Node, para que en un futuro su aceptación en el estándar suponga menos problemas.

8. IMPLEMENTACIÓN

En la sección que sigue, vamos a ver como se han ido desarrollando los distintos proyectos, comentando algunos problemas que han surgido y como se han afrontado.

8.1 Aplicación de recogida de muestras (portátil)

El desarrollo de esta aplicación se comienza con el sistema de carga de datos desde la base de datos online a la base de datos local alojada en el portátil.

Con este fin se desarrollan los distintos webservices en lenguaje php, que comunicaran la base de datos con el ordenador portátil, así como una interfaz muy sencilla, desde la que se van a satisfacer los dos casos de uso relacionados con la carga de datos (cargar datos entre una fecha pasada y la fecha actual y cargar datos de todas las explotaciones)

Figura 13. Formulario de carga de datos

El primer problema en surgir consiste en el volumen de los datos cargados, como se puede ver en la Figura 13, el número de animales modificados o insertados en 8 días superan los 4000, y al utilizar la aplicación en un nuevo ordenador portátil, se deben cargar los datos de aproximadamente 20 años, pudiendo llegar a superar el millón de animales.

Estos volúmenes impiden realizar la carga de datos en una única llamada php, por lo que la solución a este problema consiste en cargar los datos en varias peticiones, limitando el número de animales por petición, para este caso el número de animales por petición se ha fijado a 2000 animales.

Otro problema relacionado con el tamaño de los datos cargados es el tiempo que se requiere para su descarga e inserción en la tabla local, siendo la inserción bastante más lenta que la descarga. Por este motivo se decidió investigar sobre los distintos métodos para utilizar múltiples hilos en Access, la ventaja de utilizar múltiples hilos queda clara, pudiendo realizar cada petición de 2000 animales en un hilo independiente.

Access por sí mismo no soporta procesos multihilo, la solución que ofrece para funciones asíncronas es el método DoEvents, de forma que si en tu función asíncrona llamas periódicamente a DoEvents, la interfaz de usuario no se bloqueara esperando a que esta termine, sino que cada vez que este método sea llamado, gestionara los distintos eventos de la interfaz, y después se continuará ejecutando la función asíncrona hasta la siguiente llamada a DoEvents.

Si esta solución no es suficiente, hay varias formas de ejecutar hilos desde Access utilizando herramientas externas al propio Access, siendo los siguientes los tres métodos más utilizados para ese fin.

El primer método consiste en lo siguiente, desde el propio Access, generar una copia del archivo ejecutable, generar un fichero con extensión vbs (Visual Basic script), y ejecutar el script. Este script lo que hace es ejecutar fichero Access generado, así como la función que queremos que se ejecute en el nuevo hilo, y después borrar los dos archivos que se han generado para su correcto funcionamiento, tal y como se muestra en la Figura 14.

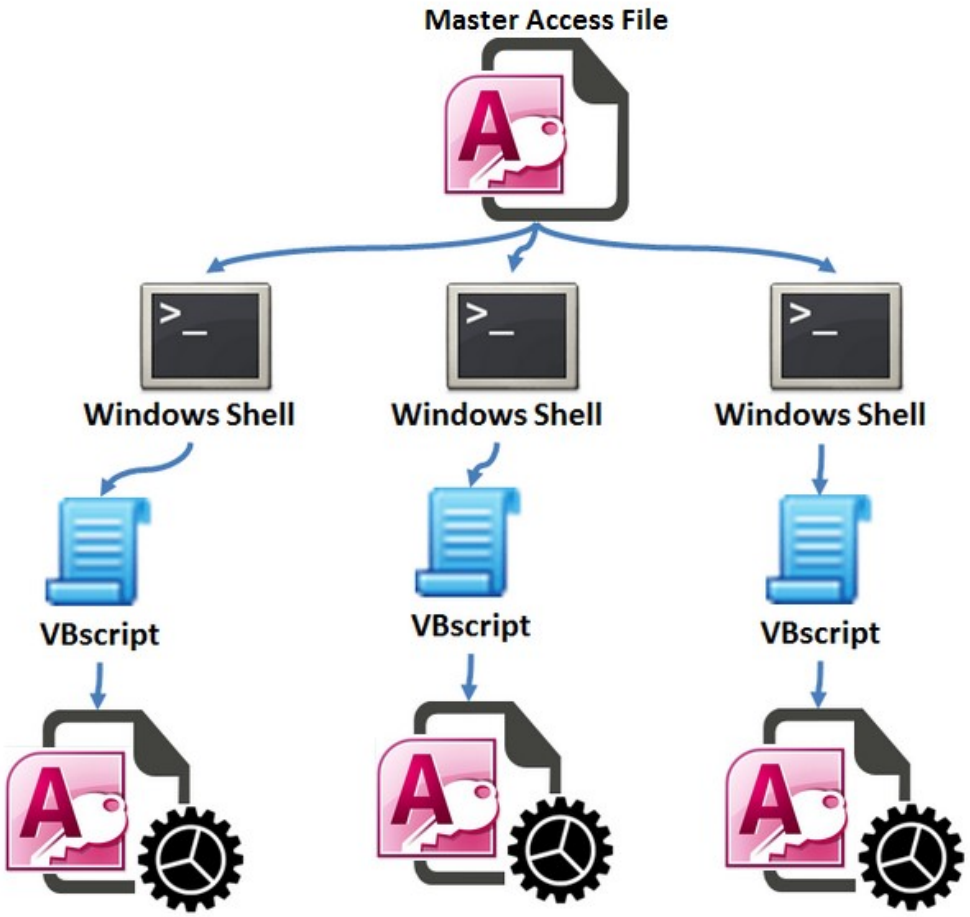

Figura 14. Esquema multihilo VBS

Este método fue nuestra primera opción para solucionar el problema, pero tras su implementación y diversas pruebas, se llegó a la conclusión de que no era un método viable en nuestro caso, pues al abrir un archivo de Access, Windows muestra un aviso de advertencia por cada hilo que se ejecute de esta forma, informando de que el archivo podría ser dañino para el equipo, y a pesar de poderse desactivar este mensaje para un ordenador, no se encontró la forma de desactivar el aviso para cualquier ordenador en el que vaya a ejecutarse el programa.

El segundo método, al igual que el primero, va a generar un fichero de VisualBasicScript y ejecutarlo, pero la diferencia es que este fichero va a contener el código que queremos que ejecute nuestro hilo, en vez de ejecutar otro proceso de Access.

Descartamos este sistema por ser altamente ineficiente (VBS es mucho menos eficiente que VBA), pudiendo llegar en algunos casos a ser el proceso multihilo más lento que el proceso original.

El tercer método es con diferencia el más eficiente de los tres, consiste en programar una librería externa en .NET (ya sea C#, Basic, C++…), exportar el dll correspondiente, y cargarlo desde un módulo de Access para tener acceso a los métodos de dicha librería, los cuales, al estar programadas en .NET, pueden inicializar procesos hijos en otros hilos sin problema.

Se decidió investigar más sobre este método, y descubrir un modo de utilizarlo sin necesidad de registrar la dll en el sistema [8], pues registrarlo es la forma más común de utilizar estas librerías, pero requiere de una configuración previa en los equipos de los usuarios finales.

Tras varias pruebas, utilizando una librería llamada UnmanagedExports [9], se consiguió una implementación correcta del sistema, sin tener que realizar apenas modificaciones en los códigos de las clases .NET a exportar. En el ANEXO B encontraremos un manual realizado para que el equipo de desarrollo de Nasertic, pueda implementar procesos multihilo en sus aplicaciones de VBA en un futuro.

Una vez completado el proceso de carga de datos, se procede a la implementar la automatización del registro de nuevas muestras, el procedimiento es el siguiente, cuando se introduce el campo crotal de un nuevo registro, se comprueba si ese animal existe y si la edad del animal es correcta, y en caso contrario se muestra una alerta al usuario. Si el registro del animal es correcto, se rellenan los campos necesarios de forma automática. Podemos ver el formulario que se encarga del proceso en la Figura 15.

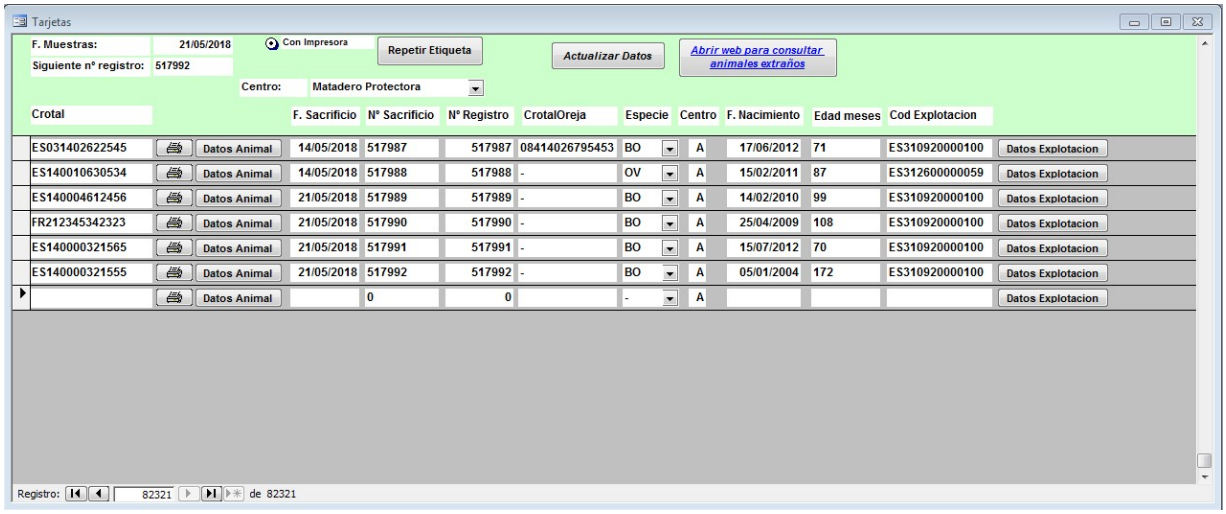

Figura 15. Formulario de registro de animales

Además de este proceso de autocompletado, al formulario de la Figura 15 se le han añadido diversas utilidades para hacer agilizar algunos procesos, por ejemplo, la posibilidad de imprimir la etiqueta de cualquier animal a través de un botón, poder ver los datos de cualquier animal registrado, su explotación, y todos los animales de su explotación, o poder modificar algunos valores desde el formulario.

8.2 Aplicación de recogida de muestras (móvil)

En cuanto a la aplicación de Ionic, los primeros pasos consistieron en implementar las interfaces de la aplicación, en la Figura 16 podemos ver tanto la interfaz principal en la parte central y derecha de la figura, como la interfaz para introducir los datos cuando la automatización no es posible a la izquierda.

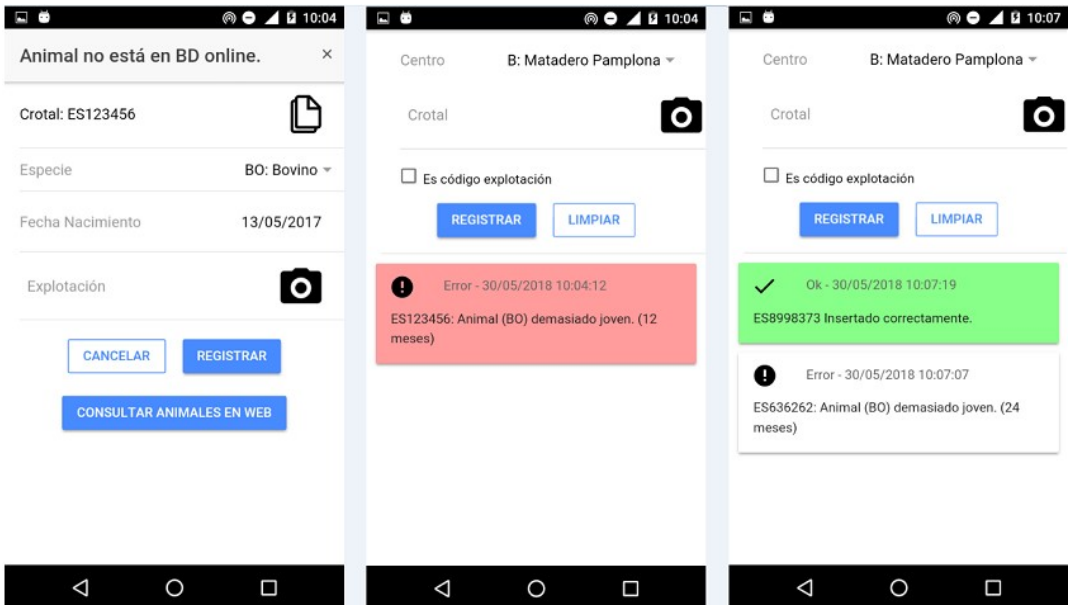

Figura 16. Interfaz de la aplicación Android

Una vez completadas las interfaces, y probados los sistemas de errores y alertas, se ha procedido a desarrollar el servidor Node, y desde el mismo gestionar las distintas peticiones web que se realizan desde la aplicación Android.

La implementación de la parte del servidor no fue demasiado complicada, el flujo del programa es idéntico al que encontramos en la aplicación de recogida de muestras en el portátil. Por lo que la programación del servidor ha consistido en una traducción de VBA a JavasScript. Con la excepción de algunos aspectos de gestión de errores y la comunicación con el móvil.

Los últimos pasos del desarrollo consistieron en la implementación de funcionalidades extras, como la opción de utilizar la cámara del móvil para leer los crotales desde códigos de barras, o el sistema de actualización automática de la aplicación.

Para esta aplicación se han realizado distintos manuales de uso tanto de instalación y uso para los usuarios, como para hacer uso del sistema automático de actualización para los desarrolladores, encontramos los manuales en los ANEXOS C, D, E de este documento.

8.3 Aplicación de generación de informes mensuales

El último de los proyectos consiste en automatizar los informes de análisis de muestras que se realizan en el laboratorio.

Para generar los informes de los animales ovinos, es necesario conocer el estado de vigilancia de las explotaciones a las que pertenecen, pero esta información no está almacenada en la base de datos del laboratorio, por lo que el primer paso en este proyecto consiste en implementar un sistema de seguimiento para las explotaciones que se encuentran en un estado de vigilancia excepcional.

Con este propósito ha sido creada una tabla en la que se almacenen los códigos de las explotaciones en estados especiales, el estado al que cambian, y la fecha en la que cambian a dicho estado. Y se ha desarrollado el interfaz de la Figura 17 en el que los usuarios pueden añadir registros en esta tabla, así como realizar consultas.

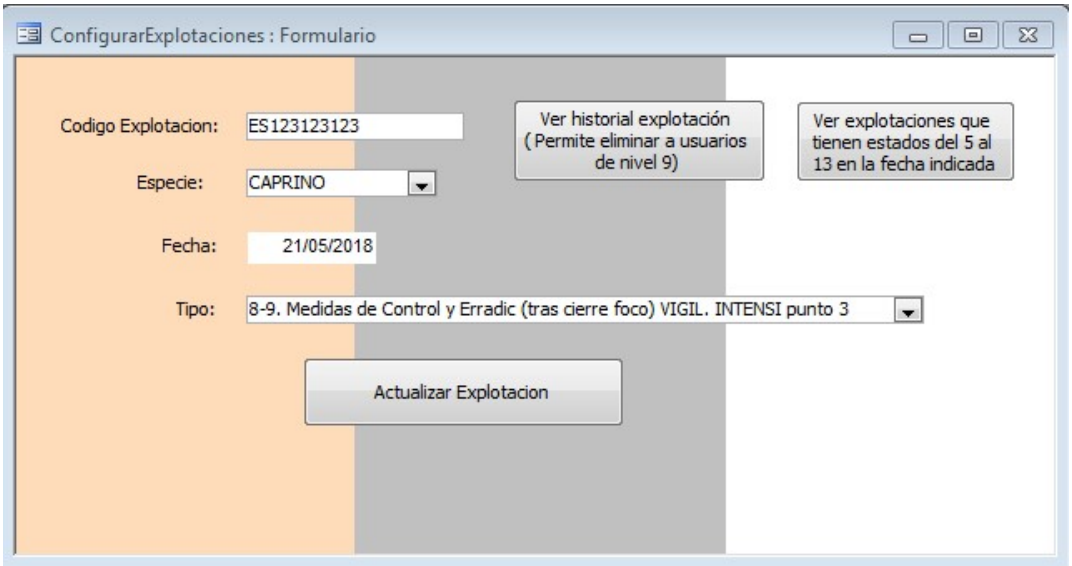

Figura 17. Interfaz de configuración de explotaciones

Una vez terminado el sistema de configuración de explotaciones, se pudo proceder a automatizar la generación de informes. Debido a los cambios requeridos en el informe de ovinos, el sistema de generación de informes que se desarrollado en el pasado había quedado completamente obsoleto, por lo se ha decidido implementar estos informes desde cero.

Para ello se generó el interfaz que vemos en la Figura 18, en el que el usuario tiene la opción de generar el informe en formato Word (Ejemplo de informe generado en el ANEXO A), así como de ver los datos de los animales encontrados en cada casilla del informe.

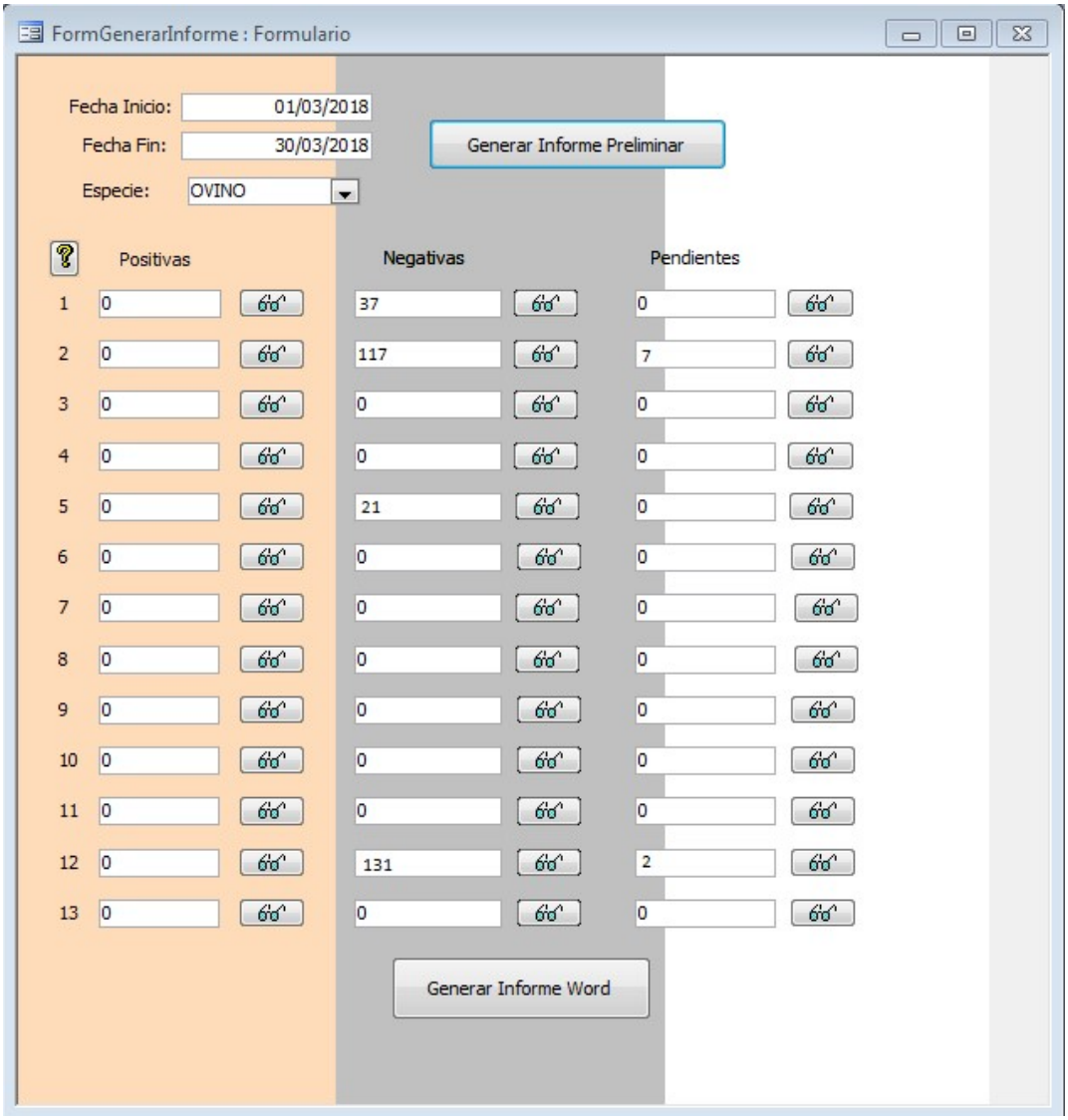

Figura 18. Interfaz de generación de informes ovinos

En cuanto a la generación de informes bovinos, el informe principal no necesitaba de ningún cambio, sin embargo sí que se automatizo la generación de algunas listas de animales que cumplían ciertas condiciones de edad o sintomatología.

9. CONCLUSIONES Y LÍNEAS FUTURAS

9.1 CONCLUSIONES

A continuación voy a detallar las distintas conclusiones a las que he llegado durante y después de la realización de este trabajo.

Sobre los proyectos de Access, destacaría que me han enseñado a valorar las buenas prácticas de programación en las que tanto se insiste a lo largo de la carrera. Pues comprender el funcionamiento previo de los programas fue un proceso más lento y complicado de lo esperado, por la falta de comentarios en el código y por ser los nombres de las variables nada descriptivos. Esto además afecta directamente a la confianza con la cual realizas modificaciones en el código, ralentizando todo el proceso de desarrollo.

Por otra parte, a pesar de ser Access 2003 un entorno de desarrollo con más de 15 años de antigüedad, es realmente potente, y tras adquirir cierta soltura, no da apenas problemas. Aunque sí tiene algunas limitaciones importantes, como ya hemos visto, por ejemplo, la dificultad de implementar procesos multihilo.

En cuanto a la metodología ágil de desarrollo que se ha seguido, similar en muchos aspectos a la programación extrema, me ha parecido una metodología estupenda para proyectos de pequeño o mediano tamaño como ha sido el caso de los distintos programas realizados. Da buenos resultados a corto plazo, y además mantiene al cliente satisfecho pues este ve un progreso continuo, y asegura que al final del desarrollo el producto no se va a alejar de lo que él desea.

9.2 LÍNEAS FUTURAS

El cambio más importante que se ha planteado para un futuro, consiste en modificar la aplicación de móvil para hacerla independiente del ordenador portátil.

En cuanto a la aplicación de recogida del ordenador portátil, se planea en un futuro automatizar el proceso de exportar las muestras registradas en el portátil a la base de datos del laboratorio.

Por su parte, la aplicación de generación de informes no tiene previsto ningún cambio, por lo menos mientras desde el gobierno no se exijan modificaciones en los informes mensuales.

10. BIBLIOGRAFÍA

[1] http://www.mapama.gob.es/es/ganaderia/temas/sanidad-animal-higiene-ganadera/sanidad-animal/enfermedades/encefalopatias-espongiformes-transmisibles/EETs.aspx

[2] https://github.com/VBA-tools/VBA-JSON

[3] https://www.codecademy.com/learn/learn-express

[4] https://ionicframework.com/docs/components/

[5] https://www.websequencediagrams.com

[6] https://yuml.me/diagram/nofunky/usecase/draw

[7] https://www.draw.io/

[8] http://analystcave.com/excel-multithreading-vba-vs-vbscript-vs-c-net/

[9]https://sites.google.com/site/robertgiesecke/Home/uploads/unmanagedexports#TOC-VB.Net

[10] https://www.wikipedia.org/

ANEXO A. EJEMPLO DE INFORME MENSUAL

CUADRO MENSUAL OVINO
PROGRAMA DE VIGILANCIA EETs
01/04/2018 – 30/04/2018

Reb año	Tipo	Fase	Subpoblación en RASVE	Nº muestras *	Positivas	Negativ.	muestras pendient
Otro rebaño: Animal de rebaño NO infectado			1. Sano sacrificado para Consumo Humano.	37		37	
			2. Muerto o Sacrificado No para consumo humano	124		117	7
			3. Sacrificado en campaña de erradicación de enfermedades sin síntomas de enfermedad				
			4. Sospechoso **				
			5. Muerto analizado para cumplir los requisitos de Comercio Intracomunitario , 123NA321	21		21	
Rebaño infectado	CLASICO	Erradicación del Rto 999/2001 (durante erradicación del foco)	6. Sacrificado y Destruido en aplicación de las opciones 1 y 2 de Erradicación				
			7. Sacrificado Consumo Humano en aplicación de las opciones 1 y 2 de Erradicación				
		Medidas de Control y Erradic (tras cierre foco)	8. Muerto explotación sometida a VIGILANCIA INTENSIFICADA pto 3 Anexo VII.				
			9. Sacrificado para Consumo Humano procedente de explotación sometida a VIGILANCIA INTENSIFICADA según punto 3 Anexo VII.				
			10. Muerto explotación sometida a VIGILANCIA INTENSIFICADA pto 4 Anexo VII. , 789NA987				
			11. Sacrificado para Consumo Humano procedente de explotación sometida a VIGILANCIA INTENSIFICADA según punto 4 Anexo VII. , 789NA987				
	ATIPICO	VIGILANCIA INTENSIFICADA TEMBLADERA ATIPICA	12. Muerto explotación con Tembladera Atípica VIGILANCIA INTENSIFICADA. , 122NA777, 010NA333, 878NA787	135		131	2
			13. Sacrificado Consumo Humano explotación con Tembladera Atípica sometida a VIGILANCIA INTENSIFICADA. , 122NA777, 010NA333, 878NA787				

- *Muestras: animales >18 meses de edad o en cuya encía hayan hecho erupción dos incisivos definitivos.
- **: Animal sospechoso EET por sospecha clínica.
- Ampliación de las definiciones "1" en la aplicación RASVE.

ANEXO B. MANUAL DE DESARROLLO DE DLL EN .NET

En este manual se explica cómo generar una librería DLL en .NET (VB y C#) que pueda ser usada desde VBA sin ser registrada en el sistema operativo.

Todas las pruebas durante el desarrollo de este manual se han realizado en VISUAL STUDIO 2010 ultimate

Generamos un nuevo proyecto:

file-> new -> project -> Elegir lenguaje -> Class Library -> OK

Marcamos el proyecto como COM-Visible:

project -> properties -> application -> assembly information... -> make assembly COM-Visible (marcar)

Instalamos el instalador de packetes NuGet (Este paso se puede saltar si ya está instalado)

tools -> extension manager... -> NuGet package manager (instalar)

Instalamos la librería UnmanagedExports (Hay que instalarlo en cada proyecto)

tools -> NuGet package manager -> Package Manager Console
Se nos abrirá en la parte inferior la consola de NuGet
Ejecutar el siguiente comando
Install-Package UnmanagedExports

Cambiamos la plataforma objetivo de la librería:

build -> configuration manager -> Active solution platform -> new -> type or select the new platform -> x86 -> Ok
* La configuración por defecto (any cpu) da problemas, en la guía en la que me he basado recomiendan cambiarla por 86
* Las pruebas han dado error al cambiarla a x64

Programamos las funciones de la librería

Ejemplos de código más abajo
*Importante que el nombre de la función .net con la etiqueta [DllExport] coincida con el nombre de la variable de tipo function que se carga de la librería dll en VBA (CreateTestClass en los ejemplos)

Generamos el fichero dll

build -> build solution
* el fichero se encontrará en la carpeta "nombre proyecto"/"nombre proyecto"/bin/x86/debug (o release)

Copiamos el fichero DLL en la carpeta donde se encuentre nuestro proyecto Access

Añadir un módulo en el proyecto Access con un código similar al ejemplo (ejemplo más abajo)

GUIA SACADA DE:
https://analystcave.com/excel-use-c-sharp-in-excel-vba/

OTROS EJEMPLOS DEL USO DE LA LIBRERIA EN:
https://sites.google.com/site/robertgiesecke/Home/uploads/unmanagedexports#TOC-VB.Net:

EJEMPLOS CÓDIGO:
En estos ejemplos, encontramos una clase principal, marcada con la etiqueta com visible en esta clase podremos introducir cualquier código del lenguaje correspondiente sin problemas y estas funciones podrán ser llamadas desde el código VBA

Después vemos una clase (UnmanagedExports) cuya única función es exportar la clase principal a nuestro código VBA
*La etiqueta Marshal es necesaria, para que el SO haga compatible el tipo de el objeto

>>>>>>>>>>>>>>>>>>>>>>>> CODIGO C# <<<<<<<<<<<<<<<<<<<<<<<<
*** SIMPLIFICACIÓN DEL CÓDIGO ORIGINAL DE LA GUIA

```
using RGiesecke.DllExport;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Runtime.InteropServices;
using System.Text;
using System.Threading.Tasks;
   [ComVisible(true)]
   public class Class1{
      public string Text;
      public int Numbers;
      public int GetRandomNumber(){
         Random x = new Random();
         return x.Next(100);
      }
   }
   static class UnmanagedExports{
      [DllExport]
      [return: MarshalAs(UnmanagedType.IDispatch)]
      static Object CreateTestClass(){
         return new Class1();
      }
   }
```

>>>>>>>>>>>>>>>>>>>>>>>> CODIGO VBA <<<<<<<<<<<<<<<<<<<<<<<<<<<<

```
Declare Function CreateTestClass Lib "ClassLibrary1.dll" () As Object
Sub TestTheTestClass()
  Dim testClass As Object
  Set testClass = CreateTestClass() 'Creates an instance of Class1
```

```
  Debug.Print testClass.GetRandomNumber 'Executes the method
  testClass.Text = "Some text" 'Set the value of the Text property
  testClass.Numbers = 23 'Set the value of the Number property
  Debug.Print testClass.Text
  Debug.Print testClass.Numbers
End Sub
```

>>>>>>>>>>>>>>>>>> CODIGO VB .NET <<<<<<<<<<<<<<<<<<<<<<<<

```
Imports System
Imports System.Collections.Generic
Imports System.Runtime.InteropServices
Imports System.Text
Imports System.Threading
Imports System.IO
Imports RGiesecke.DllExport

<ComVisible(True)>
Public Class Class1
   Public text As String
End Class

Public Module UnmanagedExports
   <DllExport()> _
   Public Function CreateTestClass() As <MarshalAs(UnmanagedType.IDispatch)>
Object
      Return New Class1
   End Function
End Module
```

>>>>>>>>>>>>>>> CODIGO C# OTRO EJEMPLO CON HILOS <<<<<<<<<<<<<<<<

```
using RGiesecke.DllExport;
using System;
using System.Collections.Generic;
using System.Runtime.InteropServices;
using System.Text;
using System.Threading;
using System.IO;
   [ComVisible(true)]
   public class Class1{
      private static ReaderWriterLockSlim _readWriteLock = new
ReaderWriterLockSlim();
      public string Text;
      public int Numbers;
      public int GetRandomNumber(){
         Random x = new Random();
         return x.Next(100);
      }
      public int startThreads(int numeroHilos)  {
```

```
        for (int i = 0; i < numeroHilos; i++)      {
            ThreadStart childref = new ThreadStart(CallToChildThread);
            Thread childThread = new Thread(childref);
            childThread.Start();
        }
            return 0;
    }
    public void CallToChildThread()     {
                writeWaitOnLocked("hola " + DateTime.Now,"asd.txt");
        Thread.Sleep(5000);
        writeWaitOnLocked("adios " + DateTime.Now, "asd.txt");
    }
    public void writeWaitOnLocked(String text, String path){
        // Set Status to Locked
        _readWriteLock.EnterWriteLock();
        try     {
            // Append text to the file
            using (StreamWriter sw = File.AppendText(path))    {
                sw.WriteLine(text);
                sw.Close();
            }
        }
        finally  {
            // Release lock
            _readWriteLock.ExitWriteLock();
        }
    }
}
static class UnmanagedExports{
    [DllExport]
    [return: MarshalAs(UnmanagedType.IDispatch)]
    static Object CreateTestClass(){
        return new Class1();
    }
}
```

ANEXO C. MANUAL DE INSTALACIÓN Y CONFIGURACIÓN APLICACIÓN MOVIL

Instalaciones necesarias en los ordenadores portátiles

- Para el funcionamiento de la aplicación, es necesario que el móvil se conecte mediante wifi a un ordenador portátil, en este ordenador tenemos que realizar las siguientes instalaciones.

- Instalar Node.js desde su web oficial
  La versión de node utilizada durante el desarrollo de la aplicación es la 8.11.1
  Versiones superiores deberían ser compatibles.

Configuración de ordenadores portátiles

- Una vez realizadas la instalación de node, tenemos que realizar los siguientes pasos.

- Copiar el contenido de la carpeta RecogidaMuestras-api al portátil
  * no es necesario copiar la carpeta node_modules

- Desde la consola de comandos, situándonos dentro de la carpeta copiada, ejecutamos el siguiente comando
  npm install
  (este comando descargara de internet las librerías necesarias a la carpeta node_modules)

- Solo nos queda configurar el ordenador para que inicie el servidor de node en el arranque
  El sistema utilizado hasta el momento es crear un acceso directo al script "start.vbs" encontrado en la carpeta RecogidaMuestras-api
  en la carpeta de inicio de windows, la carpeta de inicio se encuentra en la siguiente ruta
  C:\Users\>>>NOMBRE_USUARIO<<<\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup

*** la siguiente línea es el contenido del script (por si se perdiese)
CreateObject("Wscript.Shell").Run "node foreverStart.js", 0

Instalación de la app en el dispositivo android

- Para instalar la aplicación en el móvil, solo hay que acceder a la siguiente url desde el navegador.
  http://XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

- La versión de android debe ser 4.3 o superior
  *** Si hay errores durante la instalación, activar desde android la opción de instalar aplicaciones de origen desconocido
  *** Permite instalarse en 4.1 o superior pero con el único dispositivo probado con 4.1 hubo problemas

- Una vez instalada, la aplicación se actualizara automáticamente al iniciarse con conexión a internet

- Para facilitar el uso de la aplicación al usuario final, se recomienda crear un acceso rápido a la configuración de zona wifi del dispositivo móvil si la versión de android no la proporcionase por defecto.

ANEXO D. MANUAL DE DESARROLLO APLICACIÓN MOVIL

Instalaciones necesarias para desarrollar sobre el proyecto

- Instalar Node.js desde su web oficial
  La versión de node utilizada durante el desarrollo de la aplicación es la 8.11.1
  Versiones superiores deberían ser compatibles.

- Con la instalación de Node, habremos instalado también el instalador de paquetes npm
  utilizando npm vamos a instalar tanto cordova como ionic, necesarias para compilar la app móvil y generar el apk correspondiente, ejecutaremos los siguientes comandos de consola.

- npm install -g cordova

- npm install -g ionic

Server node

1. El servidor está en la carpeta RecogidaMuestras-api

2. Para instalar las librerías necesarias hay que ejecutar el siguiente comando dentro de la carpeta
   npm install

3. Para arrancar el servidor hay que ejecutar el comando:
   node foreverStart.js

4. El dispositivo en el que está el servidor y el móvil deben estar en la misma red (zona wifi compartida por el móvil de android)

el móvil ira probando a realizar peticiones a los dispositivos conectados a su zona wifi hasta que uno le responda correctamente, y así sabrá que se ha establecido conexión con éxito

---

App ionic

1. La aplicación está en la carpeta RecogidaMuestras-ionic
2. Para instalar las dependencias necesarias
   Desde la consola, en la carpeta -> npm install
3. Para generar el proyecto android
   cordova platform add android
4. para generar app-debug.apk -> ionic cordova run android
   El apk se genera en la carpeta platforms/android/app/build/outputs/apk/debug

   * A la hora de copiar la aplicación a otra carpeta o disco, se pueden borrar las carpetas
   node_modules y platforms (que son las que hemos generado con los comandos anteriores)

Para actualizar la aplicación y hacer uso de la actualización automática

1. En el archivo config.xml (en el proyecto ionic), en la segunda línea <widget id="nasertic.recogida.muestras" version="0.0.4"
   Hay que modificar el código de la versión por uno superior
2. En el fichero version.txt (en la web), hay que cambiar el número por la versión correspondiente del archivo config.xml
   Esto es, por cada número entre puntos, corresponden dos dígitos, ej: 1.34.9 -> 013409
3. Sustituir el archivo APK viejo por el nuevo APK >> ¡FIRMADO Y MANTENIENDO EL MISMO NOMBRE Y RUTA! <<

Generar el APK firmado

*** El paso 1 no hay que realizarlo para actualizar, sino que en el paso 3 usaremos la firma que ya se usó para firmar la versión antigua de la app

** La clave para el fichero de firma será la misma que el nombre del propio fichero de firma

1. Para generar un nuevo archivo de firma (keytool viene con el jdk de java):
   keytool -genkey -v -keystore XXXXXXXXXXXXXXXXX -alias XXXXXXXXXXXXXXXXXXXXX -keyalg RSA -keysize 2048 -validity 10000
2. Para generar el nuevo apk:
   ionic cordova build --release android
   El apk se genera en la carpeta platforms/android/app/build/outputs/apk/release
3. Firmar el apk (jarsigner viene con el jdk de java):
   jarsigner -verbose -sigalg SHA1withRSA -digestalg SHA1 -keystore XXXXXXXXXXXXXXXXX app-release-unsigned.apk XXXXXXXXXXXXXXXXXXXXX
4. Optimizar el apk (zipaligner viene con el sdk de android):
   zipalign -v 4 app-release-unsigned.apk RecogidaMuestras.apk

ANEXO E. MANUAL DE USO APLICACIÓN MOVIL

Manual de uso de la aplicación Android

Conectando los dispositivos

Se presupone que dispones de un móvil android con la aplicación ya instalada así como un ordenador portátil ya configurado, se pasan a listar los pasos hacerla funcionar.

- El primer paso es activar la opción de compartir wifi (zona wifi) desde el dispositivo Android

- A continuación, conectar el portátil al wifi compartido desde el móvil (si no se conectase automáticamente)

- Una vez hecho esto, al abrir la aplicación, sabremos que se ha establecido la comunicación correctamente si la rueda de carga que se encuentra arriba a la izquierda en la vista principal, a la izquierda del selector de centros, desaparece.

  * si tras varios intentos no se consigue establecer comunicación, la rueda se sustituye por un icono de error. Clic en el mismo para volver a intentar establecer comunicación (aun así si el portátil está en la red del móvil el tiempo de conexión debería ser de unos pocos segundos).

Registrando muestras

Una vez que se ha establecido la conexión entre el portátil y el móvil, podemos proceder a registrar muestras

- Seleccionamos el centro de recogida.

- Rellenamos el campo del crotal, (a mano o utilizando el lector de códigos de barras(icono de la cámara de fotos))

- Pulsamos registrar

- Si tenemos datos que coincidan con el crotal que se está registrando, el proceso será totalmente automático y solo veremos un mensaje de "registrado correctamente" o un error por ejemplo de "animal menor de edad"

- Si por el contrario no tenemos esos datos, se nos abrirá una nueva ventana, en la que podremos establecer los datos extras del animal (fecha nacimiento, explotación y especie), todos estos datos son OPCIONALES, y podremos modificarlos posteriormente desde el programa de recogida en el portátil.

- Cuando queremos introducir un animal por código de explotación en vez de por crotal (no tenemos forma de conocer el crotal) en la pantalla principal de la aplicación hay que marcar la casilla de "es código explotación" antes de darle a registrar.
  Se nos abrirá una nueva ventana, en la que podremos establecer los datos extras del animal (fecha nacimiento, explotación y especie), todos estos datos son OPCIONALES, y podremos modificarlos posteriormente desde el programa de recogida en el portátil.

Otros

- En cualquier momento podemos eliminar todo el historial de mensajes de ok y errores haciendo clic en el botón limpiar.