Grado en Ingeniería Informática

Trabajo Final de Grado

Migración Cloud Control de Costes de Ganadería

Autor:
David Segarra Tejedo

Supervisor:
José Luís Guillén Blasco
Tutor:
Juan Jesús Echagüe Guardiola

Fecha de lectura: 21/07/2015
Curso académico: 2014/2015

Resumen

En el siguiente documento se presenta la el Trabajo Final de Grado de David Segarra Tejedo realizado durante el cuarto curso de Grado en Ingeniería Informática, trabajo realizado durante la estancia en prácticas en la entidad José Luís Guillén Blasco.

El proyecto Migración Cloud Control de Costes de Ganadería surge por la necesidad de la empresa José Luís Guillén Blasco de migrar a otra plataforma cloud una aplicación, ya implementada mediante Google Sheets (Hojas de cálculo de Google) y Google Apps Script, que se encarga de controlar los costes que una ganadería genera por varias vías. El proyecto está orientado a desplegar dicha aplicación en la plataforma Google App Engine y el objetivo es permitir que sea más fácil realizar las operaciones necesarias, operaciones CRUD (Create, Read, Update, Delete) básicamente, para la introducción de datos y la muestra de información extraída de una serie de cálculos. El motivo principal de la migración es que Google Sheets y Google Script tienen un límite de almacenamiento y es mucho más costoso procesar datos y hacer los cálculos.

Las tecnologías utilizadas en el desarrollo del proyecto son HTML5, CSS3 y JSP para el desarrollo de la parte de la interfaz del usuario que es visible al cliente. Por otra parte, se ha utilizado la base de App Engine, base de datos NoSQL llamada Datastore. Finalmente, para que las dos partes nombradas anteriormente se comuniquen se han utilizado Java Servlets.

Índice general

1. INTRODUCCIÓN

1.1. OBJETIVOS DEL PROYECTO

El objetivo de este proyecto es migrar una aplicación web que se encarga de controlar y gestionar los costes generados por una ganadería. Dicha aplicación está soportada por Google Sheets y Google Apps Script y se quiere migrar a una plataforma Cloud para facilitar el trabajo y poder aumentar el volumen de datos y accesos. Google Sheets tiene un límite de almacenamiento y se hace pesada la introducción de datos, y a medida que aumenta la información en las hojas de cálculo a Google Apps Script le es más costoso realizar cálculos. Estos son los principales motivos por los que se pretende migrar la aplicación a una plataforma Cloud.

Se pretende realizar el desarrollo completo de un proyecto desde el inicio hasta el final. Por ello, se realiza en primer lugar un análisis de requisitos de la aplicación, incluyendo requisitos funcionales y de datos. Posteriormente, en la fase de diseño se realiza un estudio de las herramientas que existen en el mercado y las tecnologías disponibles para la realización del mismo. En este caso concreto se realiza un estudio de las plataformas cloud disponibles y finalmente la decisión de una de ellas para alojar nuestra aplicación. A continuación, se realiza la fase de la implementación, las pruebas y la puesta en marcha de la aplicación.

1.2. CONTEXTO Y MOTIVACIÓN DEL PROYECTO

Este proyecto es propuesto y supervisado por la empresa José Luís Guillén Blasco, durante la estancia en prácticas.

La motivación de éste surge por la necesidad de la empresa de ofrecer una mejor calidad del servicio haciendo más cómoda la gestión de los datos ya almacenados y la generación de nuevos datos. Dado que hay una previsión de aumento de la información de la ganadería, la empresa ha decidido migrar la aplicación a una plataforma más robusta, segura y sobretodo fácilmente escalable.

La empresa se sitúa en Castellón y se dedica a ofrecer servicios ya sean hardware, software o servicios orientados a web.

José Luís Guillén Blasco es un Ingeniero Técnico Informático con una larga trayectoria en el sector y que finalmente ha decidido ser él mismo el que ofrece directamente soluciones y servicios a los clientes.

2. DESCRIPCIÓN DEL PROYECTO

El proyecto consiste en implementar en una plataforma cloud una aplicación ya existente mediante Google Sheets y Google Scripts.

La aplicación tiene la finalidad de controlar y gestionar los costes que genera una ganadería, desde la compra de reses hasta el cálculo de los gastos que genera cada res. La aplicación almacena los siguientes datos y se desea que la nueva aplicación mantenga la misma información:

- Materias Primas
- Lotes
- Ficha Vacas
- Ficha Corderos
- Gastos fijos
- Compras
- Consumos
- Incidencias
- Log
- Notas
- Procesos: Cálculo coste consumos
- Procesos: Cálculo y asignación de costes a reses
- Procesos: Cambios de estado
- Consultas

Las acciones básicas que se quiere que desempeñe la aplicación son las funciones CRUD de todos los datos que contiene la base de datos mencionados anteriormente. También se necesita que realice tres cálculos: “Cálculo coste de consumos”, “Cálculo y asignación de costes a reses” y “Cambios de estado”. Dichos cálculos están más detallados en la especificación de requisitos.

Para ello, la primera tarea a realizar es un estudio a fondo de las plataformas cloud disponibles para finalmente elegir la más apropiada o la que más convenga utilizar. Para la implementación se quiere utilizar la tecnología más novedosa. En un primer momento se usa HTML5, CSS3 y JSP (JavaServer Pages) para implementar la interfaz de usuario; Java Servlets para implementar la parte servidora y comunicar la base de datos con la interfaz de usuario; y finalmente se utilizara una base de datos NoSQL para el almacenamiento de la información de la aplicación. Aunque se pretende introducir a la aplicación Google EndPoints y AngularJS.

También se pretende implementar un sistema de registro mediante cuentas de Google, es decir, dar permisos a ciertas cuentas ya existentes en Google para permitir el acceso a la aplicación.

3. PLANIFICACIÓN

3.1. PLANIFICACIÓN DE TAREAS Y TEMPORAL

En el diagrama de Gantt que se muestra en la Ilustración 1 se pueden apreciar las tareas que se han realizado para el desarrollo del proyecto y su planificación temporal inicial. No se ha profundizado más, por ejemplo, en tareas como “Escritura del código del software” porque no sabíamos todavía con exactitud las clases que iba a tener el proyecto, ni por supuesto, todos los métodos que se necesitaba implementar. Por esta razón se ha realizado una estimación de la duración de cada tarea, podríamos decir, global y se ha realizado una planificación final más adelante con todas las tareas realizadas realmente.

La primera fase del proyecto fue la de “Comunicación”. Durante tres días el supervisor dio a conocer la empresa y expuso en qué consistía el proyecto, sus partes y las tecnologías y herramientas que se pretendían utilizar.

Una fase importante del proyecto era realizar un estudio sobre las plataformas cloud existentes, en este caso planificamos seis días para dicho estudio y se cumplieron los plazos. Finalmente, en la fase “Conclusión y elección de la plataforma”, se decidió utilizar Google App Engine.

La siguiente fase, “Análisis del sistema”, consistía en profundizar en el sistema ya implementado para averiguar cuáles eran las necesidades de nuestra futura aplicación. Por tanto, durante dos días se estuvo investigando qué requisitos de datos y funcionales necesitaba el sistema. Una vez detectados estos requisitos, durante cuatro días, se realizó la fase de Especificación de requisitos para detallar más a fondo las necesidades del sistema. También en esta fase se instaló todo el software necesario para implementar el sistema (Eclipse, Plugin de Google para Eclipse, Maven, ...).

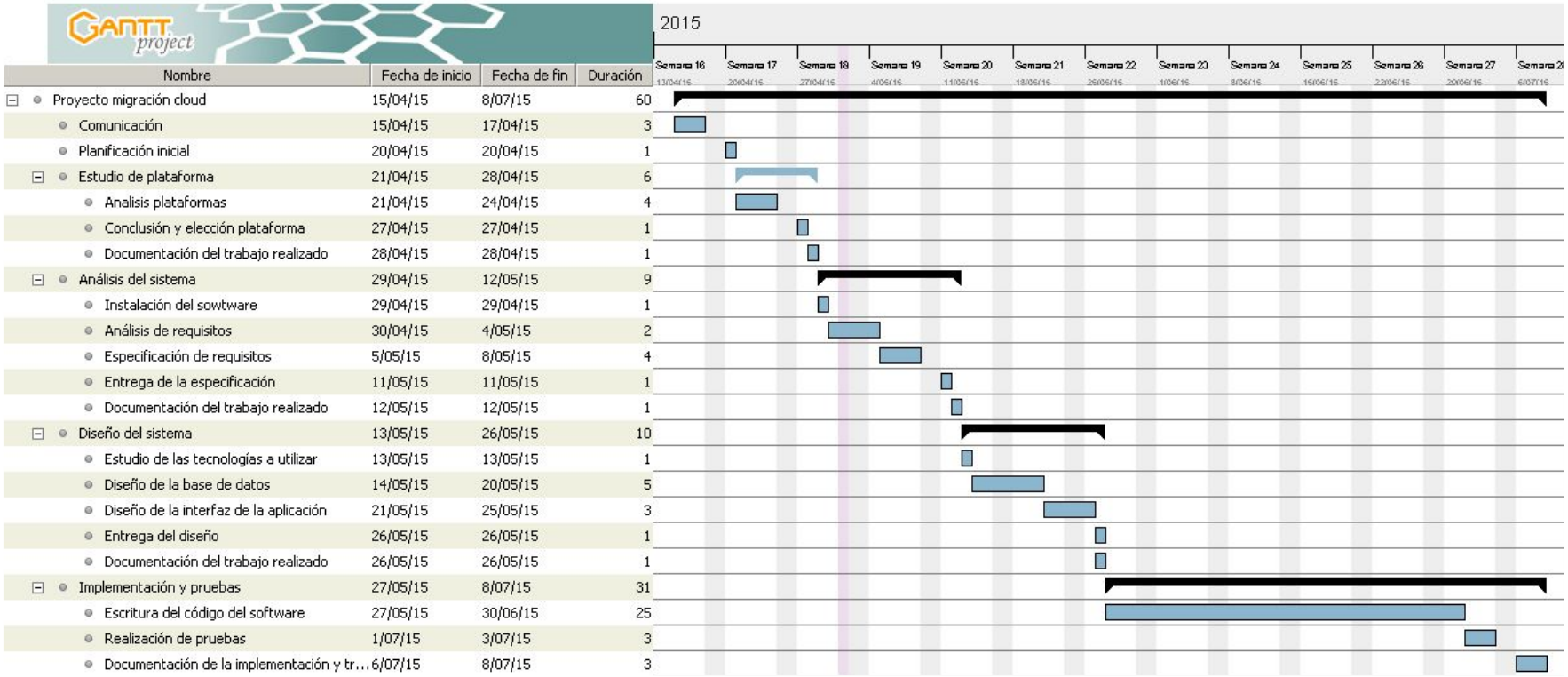
En la fase “Diseño del sistema” se pretendía determinar qué tecnologías se iban a emplear tanto para la base de datos, como para la parte de la interfaz de usuario, como para la parte servidora. En la fase “Estudio de las tecnologías a utilizar” se investigó en la página developers.google.com cuáles eran las tecnologías que sugiere Google para implementar aplicaciones destinada a Google App Engine. En la fase “Diseño de la base de datos” se investigó cuál era la base de datos NoSQL de Google y cuál era su funcionamiento. Y finalmente se diseñó la interfaz de usuario, aunque no se entró en mucho detalle porque el proyecto se orientó más al funcionamiento que al aspecto.

La última fase, “Implementación y pruebas”, no se planificó con detalle porque no se sabía con exactitud las clases que se tenían que implementar ni los métodos que se necesitaban implementar, aunque se le asignaron 31 días ya que es la parte más costosa del proyecto. Para la parte de implementación se ha seguido un tutorial de Google Developers, durante cinco días más o menos, que desarrollaba un libro de visitas, por lo que ha restado días a la implementación de nuestra aplicación, aunque creímos necesario

realizar dicho tutorial para tener un primer contacto y realizar alguna aplicación más sencilla. Esta fase no se ha dedicado entera a implementación y pruebas, ya que cuando se había implementado una parte del proyecto, utilizando JSP y Java Servlets, se vio oportuno sustituir dichas tecnologías por AngularJS y Google EndPoints para facilitar el trabajo y emplear metodologías más nuevas y punteras. Por tanto, como se ve en la Ilustración 2, se le han asignado 21 días a la escritura de código y a las pruebas de la aplicación y 7 días a búsqueda y adquisición de información sobre las nuevas tecnologías que se deseaban incorporar.

Cada fase tiene al final una fase de “Documentación del trabajo realizado”, esta fase consta de un día para redactar el trabajo realizado en un blog creado exclusivamente para publicar todo el desarrollo del proyecto.

Link del blog: sites.google.com/a/uji.es/control-costes-ganaderia



Nombre	Fecha de inicio	Fecha de fin	Duración
Proyecto migración cloud	15/04/15	8/07/15	60
Comunicación	15/04/15	17/04/15	3
Planificación inicial	20/04/15	20/04/15	1
Estudio de plataforma	21/04/15	28/04/15	6
Analisis plataformas	21/04/15	24/04/15	4
Conclusión y elección plataforma	27/04/15	27/04/15	1
Documentación del trabajo realizado	28/04/15	28/04/15	1
Análisis del sistema	29/04/15	12/05/15	9
Instalación del sowtware	29/04/15	29/04/15	1
Análisis de requisitos	30/04/15	4/05/15	2
Especificación de requisitos	5/05/15	8/05/15	4
Entrega de la especificación	11/05/15	11/05/15	1
Documentación del trabajo realizado	12/05/15	12/05/15	1
Diseño del sistema	13/05/15	26/05/15	10
Estudio de las tecnologías a utilizar	13/05/15	13/05/15	1
Diseño de la base de datos	14/05/15	20/05/15	5
Diseño de la interfaz de la aplicación	21/05/15	25/05/15	3
Entrega del diseño	26/05/15	26/05/15	1
Documentación del trabajo realizado	26/05/15	26/05/15	1
Implementación y pruebas	27/05/15	8/07/15	31
Escritura del código del software	27/05/15	30/06/15	25
Realización de pruebas	1/07/15	3/07/15	3
Documentación de la implementación y tr...	6/07/15	8/07/15	3

Ilustración 1. Resumen de tareas y planificación inicial

Nombre	Fecha de inicio	Fecha de fin	Duración
Proyecto migración cloud	15/04/15	8/07/15	60
Comunicación	15/04/15	17/04/15	3
Planificación inicial	20/04/15	20/04/15	1
Estudio de plataforma	21/04/15	28/04/15	6
Analisis plataformas	21/04/15	24/04/15	4
Conclusión y elección plataforma	27/04/15	27/04/15	1
Documentación del trabajo realizado	28/04/15	28/04/15	1
Análisis del sistema	29/04/15	12/05/15	9
Instalación del sowtware	29/04/15	29/04/15	1
Análisis de requisitos	30/04/15	4/05/15	2
Especificación de requisitos	5/05/15	8/05/15	4
Entrega de la especificación	11/05/15	11/05/15	1
Documentación del trabajo realizado	12/05/15	12/05/15	1
Diseño del sistema	13/05/15	26/05/15	10
Estudio de las tecnologías a utilizar	13/05/15	13/05/15	1
Diseño de la base de datos	14/05/15	20/05/15	5
Diseño de la interfaz de la aplicación	21/05/15	25/05/15	3
Entrega del diseño	26/05/15	26/05/15	1
Documentación del trabajo realizado	26/05/15	26/05/15	1
Implementación y pruebas	27/05/15	8/07/15	31
Escritura del código del software	27/05/15	23/06/15	20
Realización de pruebas	2/07/15	2/07/15	1
Aprendizaje de nuevas tecnologías	25/06/15	3/07/15	7
Documentación de la implementación y tr...	6/07/15	8/07/15	3

Ilustración 2. Resumen de tareas y planificación final.

4. ANÁLISIS Y DISEÑO DEL SOFTWARE

4.1. ANÁLISIS DE REQUISITOS

En el apartado que se muestra a continuación se realiza el análisis de todos los requisitos que el sistema ya dispone y debería cumplir después de la migración. Estos requisitos son marcados por la empresa, que en este caso actúa como cliente. Tras una reunión al inicio del desarrollo del proyecto, se fija una lista de funcionalidades y datos que debe cubrir la aplicación.

A continuación se presenta una lista de todos estos requerimientos y en el ANEXO A podemos encontrar la especificación de requisitos en la que se detalla cada uno de ellos de manera individual y en mayor medida.

Requisitos funcionales

RF-01. Autorizar el acceso de cuentas de Google
RF-02. Crear Lotes
RF-03. Buscar Lotes
RF-04. Editar Lotes
RF-05. Eliminar Lotes
RF-06. Crear Materias primas
RF-07. Buscar Materias primas
RF-08. Editar Materias primas
RF-09. Eliminar Materias primas
RF-10. Crear Compras
RF-11. Buscar Compras
RF-12. Editar Compras
RF-13. Eliminar Compras
RF-14. Crear Consumos
RF-15. Buscar Consumos
RF-16. Editar Consumos
RF-17. Eliminar Consumos
RF-18. Crear Fichas Vacas
RF-19. Buscar Fichas Vacas
RF-20. Editar Fichas Vacas
RF-21. Eliminar Fichas Vacas
RF-22. Crear Fichas Corderos
RF-23. Buscar Fichas Corderos
RF-24. Editar Fichas Corderos
RF-25. Eliminar Fichas Corderos
RF-26. Crear Gastos Fijos

RF-27. Buscar Gastos Fijos
RF-28. Editar Gastos Fijos
RF-29. Eliminar Gastos Fijos
RF-30. Recalcular Costes Consumo
RF-31. Recalcular Costes Vacas Tramos
RF-32. Recalcular Costes Corderos Tramos

Requisitos de datos

RD-01. Lotes
RD-02. Materias primas
RD-03. Compras
RD-04. Consumos
RD-05. Fichas Vacas
RD-06. Fichas Corderos
RD-07. Gastos Fijos
RD-08. Incidencias
RD-09. Log
RD-10. Notas

4.2. ESTUDIO COMPARATIVO DE PLATAFORMAS DE CLOUD COMPUTING

Las plataformas Cloud Computing están fundadas en un paradigma tecnológico moderno que ofrece nuevas alternativas a empresas de diversas envergaduras para implementar modelos de negocios innovadores. Con estos nuevos modelos de negocio las empresas pequeñas pueden hacer uso de las plataformas Cloud Computing disponiendo de la posibilidad de incrementar, tanto progresiva como abruptamente, su capacidad de cómputo y almacenamiento de datos en función de las necesidades y en tiempo real, implicando una oportunidad singular para la competencia de mercado.

A continuación se va realizar el análisis de las cuatro posibles plataformas cloud más relevantes en estos momentos y posteriormente se realiza una conclusión y elección de la plataforma a utilizar en el proyecto: App Engine, Amazon Web Services, Microsoft Azure y OpenStack.

App Engine

Google App Engine es un servicio de alojamiento web que presta Google de forma gratuita hasta determinadas cuotas. Este servicio permite ejecutar aplicaciones sobre la infraestructura de Google. Si no se cuenta con un dominio propio, Google proporciona uno

con la siguiente estructura, midominio.appspot.com. También permite implementar un dominio propio a través de Google Apps.

Actualmente las aplicaciones Google App Engine se implementan mediante los lenguajes de programación Python, Java, Go y PHP.

Google App Engine ofrece varias opciones de almacenamiento para adaptarse a nuestras necesidades: Una base de datos tradicional MySQL usando la nube SQL, una base de datos NoSQL o almacenamiento de objetos haciendo uso de Cloud Storage.

Con App Engine puedes utilizar herramientas conocidas, como Eclipse, IntelliJ, Maven, Git, Jenkins, PyCharm y más. La App Engine SDK permite probar aplicaciones de forma local en un entorno simulado y luego implementar nuestra aplicación con las herramientas de línea de comandos simples o el lanzador en el escritorio.

Una ventaja de App Engine (y de la mayoría de plataformas de este tipo) es que no nos tenemos que preocupar de la administración de bases de datos, servidores, etc. y únicamente nos tenemos que centrar en el desarrollo del código de la aplicación.

- *Pros y contras*

Pros	**Contras**
- Se encarga del manejo de la carga del tráfico irregular. - Pagas los servicios que usas. - Todos los servicios listos para usarse. El usuario no necesita configurar nada. - Servicios gratuitos hasta una determinada cuota. - Posee una red de alta velocidad. - Posibilidad de utilizar gran variedad de lenguajes. - Alta disponibilidad.	- No soporta aplicaciones multihilo. - 30 segundos para completar una solicitud. - Caro para carga constante. - Dificultad de cambiar de App Engine a otra. - No es un servidor estándar. No es posible acceder al sistema operativo, ni ejecutar software de terceros, ni es posible realizar tareas de administración desde línea de comandos, ni servicio SSH.

Amazon Web Services

Amazon Web Services (AWS abreviado) es una colección de servicios de computación en la nube (también llamados servicios web) que en conjunto forman una plataforma de computación en la nube, ofrecidas a través de Internet por Amazon.com. Es usado en aplicaciones populares como Dropbox, Foursquare, HootSuite. Es una de las ofertas internacionales más importantes de la computación en la nube compite y directamente contra servicios como Microsoft Azure y Google Cloud Platform.

Amazon Elastic Compute Cloud (Amazon EC2) es un servicio web que proporciona capacidad informática con tamaño modificable en la nube. Está diseñado para facilitar a los desarrolladores la informática en la nube escalable basada en web.

La sencilla interfaz de servicios web de Amazon EC2 permite obtener y configurar su capacidad con una fricción mínima. Proporciona un control completo sobre sus recursos informáticos y permite ejecutarse en el entorno informático acreditado de Amazon. Amazon EC2 reduce el tiempo necesario para obtener y arrancar nuevas instancias de servidor en minutos, lo que permite escalar rápidamente la capacidad, ya sea aumentándola o reduciéndola, según cambien sus necesidades. Amazon EC2 cambia el modelo económico de la informática, al permitir pagar solo por la capacidad que utiliza realmente. Amazon EC2 proporciona a los desarrolladores las herramientas necesarias para crear aplicaciones resistentes a errores y para aislarse de los casos de error más comunes.

- *Pros y contras*

Pros	**Contras**
- El código puede ser escrito en lenguajes de programación simples como C # , .Net , MVC. - No hay dependencia del proveedor en cuanto a temas del código. - Gran respaldo del equipo de Amazon. - Alta disponibilidad.	- Cobro del precio mínimo aunque no se utilice todos los recursos ofrecidos. - El escalado es una tarea difícil. - Si una instancia falla puede fallar todo el sistema.

Microsoft Azure

Microsoft Azure (anteriormente Windows Azure y Azure Services Platform) es una plataforma ofrecida como servicio y alojada en los Data Centers de Microsoft. Windows Azure utiliza un sistema operativo especializado, llamado de la misma forma, para correr sus "capas", un cluster localizado en los servidores de datos de Microsoft que se encargan

de manejar los recursos almacenados y procesamiento para proveer los recursos (o una parte de ellos) para las aplicaciones que se ejecutan sobre Windows Azure.

Windows ofrece una manera de proteger la información importante con una copia de seguridad automática dentro de un servicio de almacenamiento. Las copias de seguridad quedan cifradas antes de la transmisión y se almacenan cifradas en Windows Azure. Estas copias de seguridad están fuera de sitio, lejos de su centro de datos, lo que reduce la necesidad de asegurar y proteger los medios de copia de seguridad en el lugar.

La administración de copias de seguridad en la nube usa herramientas de copia de seguridad conocidas en Windows Server, Windows Server Essentials, o el Administrador de System Center Data Protection. Estas herramientas proporcionan experiencias similares al configurar, supervisar y recuperar copias de seguridad ya sea en el disco local o el almacenamiento de Windows Azure, o puede utilizar el software propio del agente. Después de que los datos se copian a la nube, los usuarios autorizados pueden recuperar fácilmente copias de seguridad de cualquier servidor. También se pueden utilizar Copias de seguridad incremental para asegurar el uso eficiente de almacenamiento y un menor consumo de ancho de banda, al mismo tiempo que permite la recuperación de punto en el tiempo de varias versiones de los datos.

Dentro de la plataforma, el servicio de Windows Azure es el encargado de proporcionar el alojamiento de las aplicaciones y el almacenamiento no relacional. Dichas aplicaciones deben funcionar sobre Windows Server 2008 R2. Pueden estar desarrolladas en .NET, PHP, C++, Ruby o Java. Además del servicio de ejecución, dispone de diferentes mecanismos de almacenamiento de datos: tablas NoSQL, blobs, blobs para streaming, colas de mensajes o 'drives' NTFS para operaciones de lectura / escritura a disco.

Azure admite cualquier sistema operativo, lenguaje, herramienta y marco, ya sea Windows, Linux, SQL Server, Oracle, C# o Java. Pone a nuestro alcance lo mejor de los ecosistemas de Windows y Linux, por lo que nos da la posibilidad de crear excelentes aplicaciones y servicios que funcionan con cualquier dispositivo.

- *Pros y contras*

Pros	**Contras**
- Pagas por lo que gastas. - Facilidad y rapidez del escalado. - El usuario no se preocupa de las tareas de configuración de los servicios. - Alta disponibilidad.	- No dispone de Memcache. - Coste inicial elevado. - Plataforma con retraso de desarrollo. - Lentitud de la respuesta de las instancias.

- Gestión de recuperación de datos gestionada por Microsoft. - SDK completo, documentado y bien integrado en el IDE.	- No dispone de estructura MapReduce.

OpenStack

OpenStack es un conjunto de proyectos de software de código abierto que las empresas/proveedores de servicios pueden usar para configurar y ejecutar su nube de computación e infraestructura de almacenamiento. Rackspace y la NASA son los contribuyentes iniciales clave para la pila. Rackspace contribuyó con su plataforma "Archivos en la Nube" (código) para alimentar la parte de almacenamiento de objetos de OpenStack, mientras que la NASA aportó su plataforma "Nebulosa" (código) para alimentar la parte Compute. El Consorcio OpenStack ha logrado tener más de 100 miembros, incluyendo Canonical, Dell, Citrix, etc en menos de un año. OpenStack hace que sus servicios se encuentren disponibles por medio de una API compatible con Amazon EC2/S3. Por lo tanto, las herramientas cliente escritas para AWS se pueden utilizar con OpenStack.

Nova es el controlador de fábrica para Cómputo para la nube OpenStack. Todas las actividades necesarias para apoyar el ciclo de vida de las instancias dentro de la nube OpenStack se manejan por Nova. Esto hace que Nova sea una plataforma de gestión que administra los recursos de cómputo, redes, autorización, y las necesidades de escalabilidad de la nube OpenStack. Sin embargo, Nova no proporciona ninguna capacidad de virtualización por sí mismo, sino que utiliza las API de libvirt para interactuar con los hipervisores compatibles. Nova expone todas sus capacidades a través de una API de servicios web que sea compatible con la API de EC2 de Amazon Web Services.

Swift proporciona un almacén de objetos virtuales eventualmente consistentes distribuida para OpenStack. Es análogo a Amazon Web Services - Simple Storage Service (S3). Swift es capaz de almacenar miles de millones de objetos distribuidos en diferentes nodos. Swift ha incorporado redundancia y tolerancia a fallos de gestión y es capaz de transmitir y guardar multimedia. Es sumamente escalable tanto en términos de tamaño (varios petabytes) y de capacidad (Número de objetos).

Heat es un servicio para orquestar múltiples aplicaciones compuestas en la nube utilizando el formato de la plantilla CloudFormation AWS, tanto a través de una API REST OpenStack-nativa como un API de consultas CloudFormation-compatible. Heat proporciona una implementación CloudFormation AWS para OpenStack que orquesta una plantilla CloudFormation AWS que describe una aplicación de nube ejecutando las llamadas apropiadas a la API OpenStack, para generar la ejecución de aplicaciones de nube. El software integra otros componentes básicos de OpenStack en un sistema de plantillas de un archivo. Las plantillas permiten la creación de la mayoría de los tipos de recursos

OpenStack (tales como instancias, IPs flotantes, volúmenes, grupos de seguridad, usuarios, etc), así como también algunas funciones avanzadas tales como una alta disponibilidad de instancias, autoescalabilidad de instancias y pilas anidadas. Al proporcionar una integración tan estrecha con otros proyectos núcleo de OpenStack, todos los proyectos núcleo de OpenStack podrían recibir un mayor número de usuarios.

- *Pros y contras*

Pros	**Contras**
- Permite gestionar el almacenamiento de datos en varios servidores que trabajen de manera conjunta en clusters. - Permite añadir nuevos nodos y configurarlos de manera autónoma. - Soporte corporativo. - Comunidad en crecimiento.	- Base de código joven. - Futuro incierto. - Configuración inicial.

Conclusión: Decisión de la plataforma.

Para concluir el estudio de la plataforma cloud empleada para el desarrollo del proyecto me dispongo a exponer cual es la plataforma elegida para dicha tarea y los motivos que me han llevado a esta decisión.

La herramienta que se ha decidido utilizar para el desarrollo del proyecto es App Engine.

Son varios los motivos por los que se ha decidido elegir esta plataforma. La primera conclusión es descartar OpenStack por su poca madurez y robustez frente a las otras herramientas expuestas.

La posibilidad de utilizar varios lenguajes de programación conocidos como Java, Python, Go o PHP ha sido una razón para elegir esta herramienta. Esta plataforma también nos da la posibilidad de implementar una base de datos NoSQL, por tanto, es un punto a favor para decantarnos por App Engine ya que son bases de datos fácilmente escalables, pueden manejar grandes cantidades de datos, responden a las necesidades de escalabilidad horizontal que tienen cada vez más empresas, no dispone de lenguajes estándar específicos, por tanto, podemos utilizar el que más nos guste, no generan cuellos de botella. Dado que no se puede proveer el crecimiento de nuestros datos ni las

necesidades de proceso en la aplicación a migrar, la arquitectura NoSQL es una buena decisión para este tipo de “problemas”.

Una ventaja de App Engine es que ofrece el servicio gratuito hasta una determinada cuota, por tanto, nos da un margen de trabajo de manera gratuita.

Una parte de la aplicación de control de costes de la ganadería está implementada en Google App Script y Google Sheet. Dado que ya se utiliza Google para proveer estos servicios, es un pequeño motivo para decantarnos por App Engine.

4.3. DISEÑO DE LA INTERFAZ

En cuanto al diseño de la interfaz, se ha querido realizar un diseño sencillo y limpio. Se ha tratado, en la medida de lo posible, facilitar el uso de la aplicación y minimizar el número de clics para agilizar los procedimientos.

Como vemos en la Ilustración 3, la página principal consta de una cabecera y un menú que comparten todas las páginas. En este menú se encuentran todas las entidades que tiene nuestra aplicación. Como se ha comentado en el apartado 2 (Descripción del proyecto) se pretende integrar en la aplicación el sistema de autenticación de Google, es decir, dar permisos a ciertas cuentas de Google para que puedan acceder a la aplicación. Por tanto, si se implementa esta función, la página principal será distinta, conteniendo únicamente un botón para iniciar sesión con tu cuenta.

Ilustración 3. Pantalla de inicio.

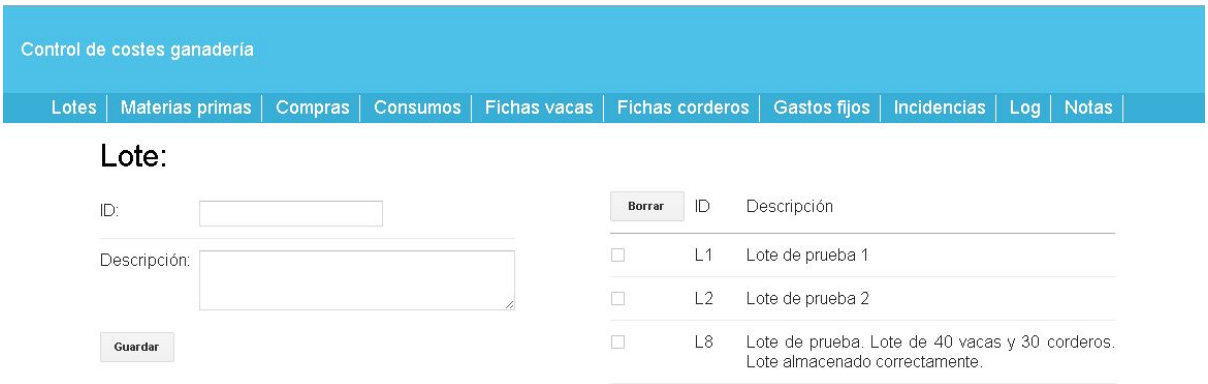


Ilustración 4. Pantalla del apartado de Lotes.

Ilustración 5. Pantalla del apartado de Materias Primas.

Ilustración 6. Pantalla del apartado de Compras.

Como vemos en las ilustraciones anteriores, todas las pantallas siguen un mismo patrón. Todas tienen en la parte superior el menú de navegación donde poder acceder a todos los apartados de la aplicación, en la parte izquierda un formulario para introducir los datos y añadir nuevos datos y en la parte derecha un resumen de todos los datos almacenados donde podemos marcar el checkbox de cada uno de los datos para poder borrarlos con el botón “Borrar” que se encuentra en la parte superior.

5. IMPLEMENTACIÓN, PRUEBAS Y DOCUMENTACIÓN

5.1. TECNOLOGÍAS WEB EMPLEADAS E IMPLEMENTACIÓN

A continuación se van a explicar tanto las herramientas utilizadas para la escritura del código, como plugins necesarios para la implementación, como las tecnologías web empleadas.

Para la implementación del código de nuestra aplicación web se ha utilizado como herramienta principal la versión Luna de Eclipse con una serie de plugins que se mencionan más adelante. Para el trabajo en equipo se ha creado un repositorio en GitHub (https://github.com/dsegarratejedo/Proyecto) para tener un control de versiones y que el supervisor pudiera ver el trabajo realizado.

Para el desarrollo de la aplicación, en un primer momento, se realizó un tutorial de Google Developers para implementar una aplicación web con el objetivo de desplegarla en Google App Engine. Por tanto, se utilizaron las tecnologías que se presentaban en dicho tutorial. Antes de empezar a escribir el código, se debían instalar una serie de plugins que nos proporcionaban lo necesario para el desarrollo. El plugin principal es Google para Eclipse que es la primera suite de herramientas integradas para la nube Google y nos permite desplegar la aplicación en App Engine. Las herramientas que proporciona se centran en la creación de una gran lógica de la aplicación. Otro plugin importante es el Maven de Apache que es una herramienta de software para la gestión y construcción de proyectos Java, en este caso, lo utilizamos orientado a una arquitectura predefinida por Google. También se instaló el plugin Objectify que es el que nos proporciona todo lo necesario para crear y gestionar los aspectos de la base de datos DataStore (Base de datos NoSQL de Google).

Para el desarrollo de la parte del cliente se utiliza básicamente HTML5. Para dar estilo a las páginas se emplea CSS3. Desde un primer momento se le quería dar un estilo similar al que emplea Google en todas sus plataformas y páginas, por tanto, empleamos una hoja de estilo que nos proporciona Google para este fin. Para ello, escribiremos la siguiente línea en todas las páginas de la aplicación:

```
<link rel="stylesheet" href="https://ssl.gstatic.com/docs/script/css/add-ons1.css">
```

Otra tecnología utilizada en la parte cliente es JSP (JavaServer Pages) que ayuda a crear páginas dinámicas basadas en HTML. Mediante las plantillas integradas en los documentos HTML el cliente se comunica con los Servlets para realizar las peticiones y así mostrar la información de manera sencilla y dinámica.

Para realizar la comunicación entre cliente y servidor se han utilizado Java Servlets. Con dichos servlets podemos realizar llamadas a los métodos GET, POST, DELETE y UPDATE dependiendo de la tarea que se desee realizar.

Por último, para almacenar toda la información se decidió utilizar una base de datos NoSQL. En este caso, para emplear este tipo de base de datos en App Engine, Google dispone de un gestor de bases de datos llamado Datastore, y este es el que se ha empleado en el proyecto. Es un gestor bastante fácil de utilizar, ya que para almacenar datos simplemente creamos un objeto de la clase deseada y con apenas una instrucción se almacena la información.

En la Ilustración 7 vemos el esquema donde aparecen todas las tecnologías utilizadas y cómo se comporta cada una de ellas con el resto de elementos del sistema.

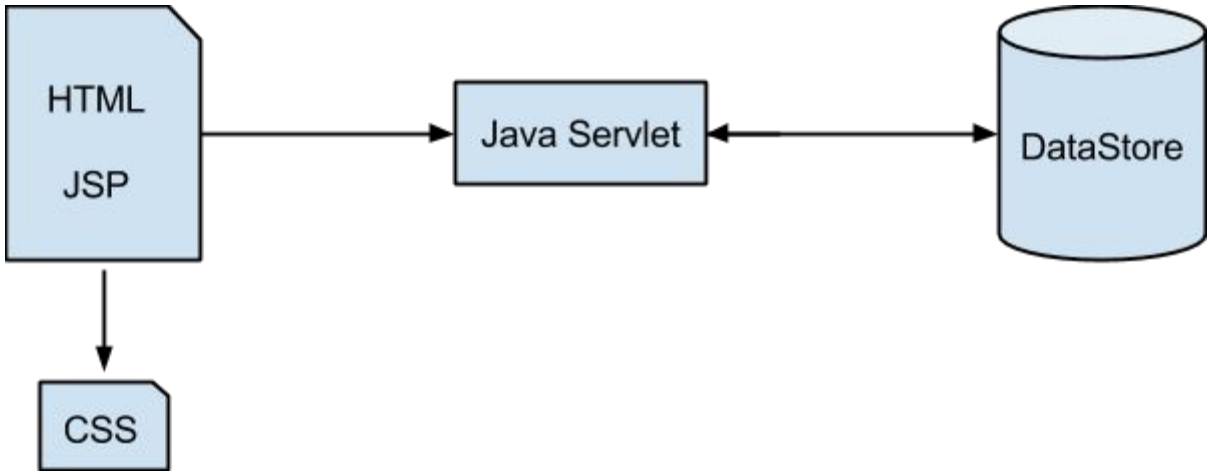


Ilustración 7. Esquema del funcionamiento de la aplicación.

5.2. REALIZACIÓN DE PRUEBAS

La realización de pruebas del proyecto ha sido una tarea más bien sencilla. Como se ha realizado el proyecto en local ha sido muy fácil realizar las pruebas a medida que se iban implementando las funcionalidades e incidir en los casos especiales.

Las acciones básicas que tiene la aplicación son: Crear , Mostrar, Actualizar y Borrar. Por tanto, para cada apartado las pruebas son iguales.

Para la acción de crear se han probado dos casos especiales: Insertar un elemento con igual ID y dejar algún campo vacío del formulario.

Las pruebas para la acción de borrar han sido eliminar más de un elemento a la vez y no marcar ningún checkbox y pulsar el botón borrar.

La acción de mostrar no debe dar ningún error si no existen datos en la base de datos, por tanto, debe dejar la tabla vacía.

Por último, no se ha podido implementar la acción de editar un elemento.

5.3. ORGANIZACIÓN DEL CÓDIGO

En este apartado se incluye una breve explicación sobre la estructura de ficheros del proyecto. En primer lugar se muestra en la Ilustración 8 el árbol del proyecto donde se representa su estructura de ficheros y carpetas.

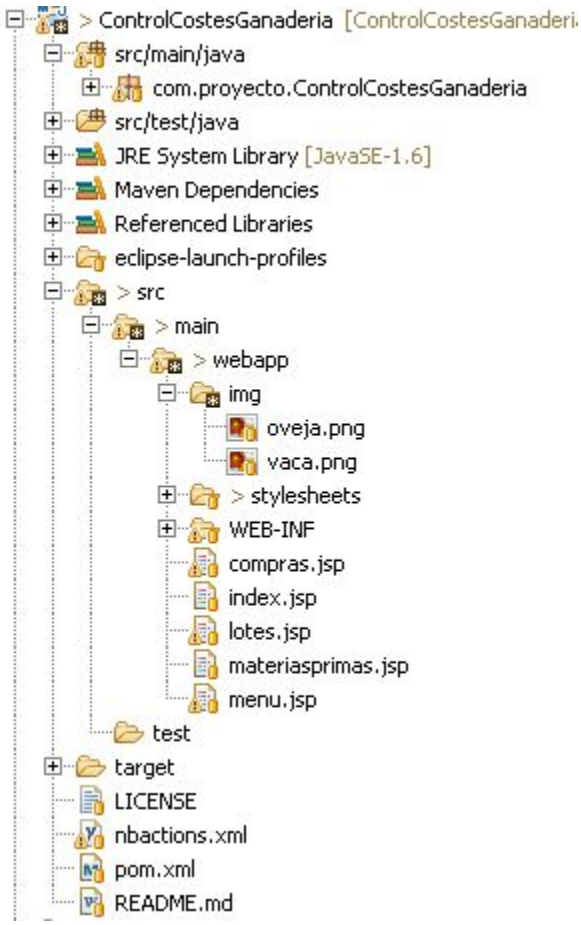


Ilustración 8. Árbol del proyecto.

A continuación se va a explicar el contenido de los directorios más importantes y relevantes del proyecto.

Los dos directorios que contienen los ficheros más relevantes son el directorio “java” y el directorio “src”, aunque dentro de este último hay varias carpetas con archivos de mucha importancia del proyecto.

En primer lugar, el directorio “java” contiene el paquete “com.proyecto.ControlCostesGanaderia” donde se encuentran, de forma general, los ficheros .java. Más concretamente encontramos dos tipos de ficheros: las clases de las entidades y los servlets. Como cada entidad está aislada del resto, almacenamos cada una en un fichero.

Dentro del directorio “src/main” encontramos la carpeta “webapp”. En este directorio encontramos tanto otros directorios como ficheros y son los que componen la parte visible de la aplicación. Contiene todas las páginas .jsp, donde se implementa la parte HTML y JSP y también se encuentran los directorios “img”, “stylesheets” y “WEB-INF”. En “img” se ubican todas las imágenes que contiene la aplicación, “stylesheets” contiene los archivos .css que necesitan los ficheros .jsp para dar estilo a la interfaz y en “WEB-INF” encontramos algunos archivos de configuración donde definimos los servlets, por ejemplo.

6. MEJORAS

En el siguiente apartado se nombrarán algunas mejoras y ampliaciones que se podrían introducir en la aplicación.

Como se ha mencionado en algún apartado anterior, se pretendía crear un sistema de autenticación para que solo puedan acceder a la aplicación aquellas personas que se desee. Una de las mejoras que se incorporará en un futuro es la autenticación mediante la cuenta de Google. Esto consiste en dar permiso a ciertas cuentas ya existentes de Google.

Otra mejora para nuestro sistema es incluir Endpoints de Google, esto nos permite, a grandes rasgos, declarar una serie de métodos que se puedan invocar desde varios clientes. Esto nos permitiría crear un back-end de manera más sencilla para clientes web y clientes móviles. En la Ilustración 9 vemos un esquema de cómo quedaría estructurado el sistema.

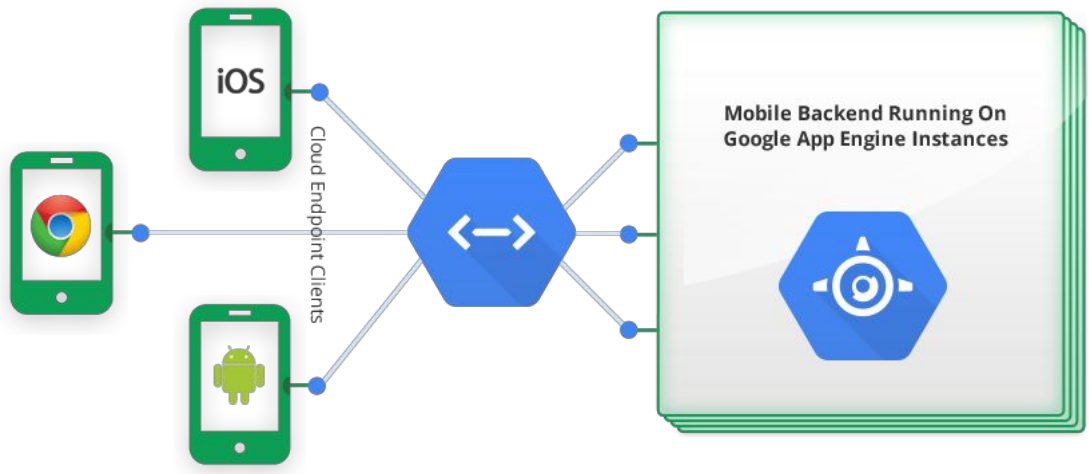


Ilustración 9. Esquema de sistema con Endpoints.

Incorporar AngularJS a nuestra aplicación es otro de los objetivos que se desea realizar. AngularJS es un framework de JavaScript de código abierto que ayuda con la gestión de lo que se conoce como aplicaciones de una sola página. Esto nos ahorraría mucho trabajo a la hora de crear webs dinámicas ya que simplifica mucho operaciones e instrucciones realizadas en JavaScript.

7. CONCLUSIONES

La aplicación Control de Costes Ganadería tiene la finalidad de controlar todos los costes que genera una ganadería a través de diferentes vías. El proyecto Migración Cloud Control de Costes Ganadería consiste en migrar una aplicación implementada mediante Google Sheets y Google App Script a una plataforma cloud, en este caso hemos elegido App Engine. La aplicación consta de varios apartados donde se se muestra, añade, modifica y borra información como por ejemplo información de los corderos, información de las vacas, consumo de las reses, etc.

El primer paso que se realizó fue un estudio de plataformas cloud para decidir dónde se desplegaría la aplicación. Las cuatro plataformas estudiadas más a fondo son App Engine, Microsoft Azure, Amazon Web Services y OpenStack. Una vez realizado el estudio, se decidió desplegar la aplicación en App Engine. Como no se finalizó la implementación del proyecto en su totalidad, la aplicación no ha sido desplegada todavía.

Una vez finalizada la estancia en prácticas en la empresa desarrollando el Trabajo Final de Grado se presenta la aplicación con tres apartados implementados: Lotes, Materias primas y Compras. Los otros apartados se implementarían de forma muy similar. Se han utilizando HTML5, CSS3 y JSP para la parte cliente; la parte de almacenamiento de datos está formada por la base de datos Datastore de Google con ayuda del plugin Objectify; y para comunicar ambas partes utilizamos Java Servlets con los métodos que nos ofrece: GET, POST, DELETE y UPDATE.

Cuando se implementaron los apartados nombrados anteriormente, se vio oportuno introducir dos tecnologías que facilitarían el trabajo y mejorarían la aplicación: AngularJS y Endpoints de Google. Por tanto, se empleó tiempo en investigación de ambas tecnologías en el tiempo que correspondía a “Implementación” establecido en la planificación inicial.

Por limitaciones temporales, no se ha conseguido el desarrollo total de la aplicación. Los motivos son principalmente dos. El primer motivo es que ha habido una fase de estudio de plataformas Cloud bastante extensa. Y el segundo motivo es que se empezó la implementación utilizando las tecnologías nombradas anteriormente y más adelante se decidió que sería interesante introducir dos tecnologías nuevas que facilitaran el trabajo: Google EndPoints y AngularJS.

En cuanto a mi valoración personal sobre la estancia en prácticas y el Trabajo Final de Grado, estoy muy contento de haber realizado este trabajo y de haber podido participar en el mundo laboral de un proyecto real. Uno de los aspectos más satisfactorios para mí es haber podido emplear mis conocimientos para desarrollar este trabajo, es más, haber aprendido nuevos conocimientos relacionados con la programación web que no sabía, como por ejemplo todo el desarrollo relacionado con la nube o aprender a utilizar bases de datos NoSQL.

Ha sido de gran utilidad para mí, para tener una visión global de la realización de un proyecto, porque he podido realizar todas las fases del desarrollo, desde el inicio, es decir, desde que surge la idea de realizarlo, hasta el fin, tras el diseño de todas sus partes, la implementación, la resolución de problemas y toma de decisiones, las entregas y plazos y todo lo que supone.

Estoy bastante satisfecho con el trabajo realizado aunque me hubiera gustado tener más tiempo para añadir nuevas funcionalidades e incorporar nuevas tecnologías más punteras relacionadas con la nube y la programación web.

También estoy contento con la estancia en prácticas y el Trabajo Final de Grado porque he dedicado mucho tiempo a aprender conceptos nuevos más que realizar un proyecto de principio a fin. Como ya he comentado, no se ha finalizado por completo ya que todo lo que necesitaba saber para realizar la aplicación lo he tenido que aprender por mi cuenta porque el supervisor vio más útil ésto que explicarme lo necesario para que me pusiera a implementar. Para mí esto ha sido un punto a favor.

A todo esto debo añadir que me siento afortunado porque el ambiente de trabajo era muy bueno, tanto a nivel profesional como a nivel personal. Por tanto, pienso que ha sido una muy buena experiencia y me ha ayudado a coger confianza en mí mismo y a verme capacitado para entrar en el mundo laboral.

8. BIBLIOGRAFÍA

[1] Google App Engine
https://**appengine.google.com/**

[2] Google Developers
https://**developers.google.com/?hl=es**

[3] Amazon Web Services
http://**aws.amazon**.com/es

[4] Microsoft Azure: plataforma y servicios de informática en la nube
http://**azure.microsoft**.com/es-es/

[5] OpenStack Open Source Cloud Computing Software
https://www.**openstack.org**

[6] Manual de AngularJS - DesarrolloWeb.com
http://www.desarrolloweb.com/manuales/manual-angularjs.html

[7] W3Schools Online Web Tutorials
http://www.**w3schools.com**

ANEXOS

ANEXO A. ESPECIFICACIÓN DE REQUISITOS

Requisitos de datos

Requisito de datos

Código	RD-01
Nombre	Lotes
Versión	1.0 (07-05-2015)
Autores	David Segarra Tejedo
Datos específicos	Nlote Descripción
Comentarios	Agrupación de reses del mismo tipo y misma fecha de compra.

Requisito de datos

Código	RD-02
Nombre	Materias primas
Versión	1.0 (07-05-2015)
Autores	David Segarra Tejedo
Datos específicos	ID Descripción Existencias
Comentarios	Diferentes elementos de consumo. Se calcula el stock actual mediante consultas de compras/consumos.

Requisito de datos

Código	RD-03
Nombre	Compras
Versión	1.0 (07-05-2015)
Autores	David Segarra Tejedo
Datos específicos	ID Concepto

	Fecha Kgs Importe Consumido Preciokg
Comentarios	Almacena los eventos de adquisición de materias primas en el tiempo.

Requisito de datos

Código	RD-04
Nombre	Consumos
Versión	1.0 (07-05-2015)
Autores	David Segarra Tejedo
Datos específicos	IDMT Concepto Lote Fecha Kgs Importe Asignado No recalcular
Comentarios	Almacena los eventos de consumo de materias primas en el tiempo, por lote y tipo de res.

Requisito de datos

Código	RD-05
Nombre	Fichas Vacas
Versión	1.0 (07-05-2015)
Autores	David Segarra Tejedo
Datos específicos	VacaID Lote Fecha de compra Gastos fijos Importe de compra Coste consumo Coste específico Importe mínimo Margen Importe venta propuesto

	Fecha de venta Fecha de pérdida Importe venta real No recalcular Peso compra Peso Venta
Comentarios	Maestro de información referente a las vacas de nuestra ganadería. Datos manualmente introducidos y otros calculados como son los costes de consumo y gastos fijos.

Requisito de datos

Código	RD-06
Nombre	Fichas Corderos
Versión	1.0 (07-05-2015)
Autores	David Segarra Tejedo
Datos específicos	CorderoID Lote Fecha de compra Importe de compra Costes consumo Coste específico Importe mínimo Margen Importe de venta propuesto Importe de venta real Fecha de venta Fecha de pérdida Ncabezas Ncabezasrestantes No recalcular
Comentarios	Maestro de información referente a los corderos de nuestra ganadería. Datos manualmente introducidos y otros calculados como son los costes de consumo y gastos fijos. Los datos están introducidos en conjunto de número de reses que componen el lote.

Requisito de datos

Código	RD-07
Nombre	Gastos Fijos

Versión	1.0 (07-05-2015)
Autores	David Segarra Tejedo
Datos específicos	Fecha Concepto Importe Alfalfa
Comentarios	Almacena los diferentes conceptos de gasto fijo referidos al conjunto de la ganadería.

Requisito de datos

Código	RD-08
Nombre	Incidencias
Versión	1.0 (07-05-2015)
Autores	David Segarra Tejedo
Datos específicos	Número de vaca Fecha Incidencia
Comentarios	Ficha de anotaciones manuales sobre incidencias ocurridas en el tiempo.

Requisito de datos

Código	RD-09
Nombre	Log
Versión	1.0 (07-05-2015)
Autores	David Segarra Tejedo
Datos específicos	Concepto
Comentarios	Ficha de anotaciones manuales sobre incidencias ocurridas en el tiempo.

Requisito de datos

Código	RD-10
Nombre	Notas
Versión	1.0 (07-05-2015)
Autores	David Segarra Tejedo

Datos específicos	--------
Comentarios	Ficha de anotaciones manuales sobre funcionalidades nuevas, sugerencias, etc.

Requisitos funcionales

Requisito Funcional

Código	RF-01
Nombre	Autorizar el acceso de cuentas de Google
Versión	1.0 (07-05-2015)
Autores	David Segarra Tejedo
Descripción	Sistema para autorizar el acceso a la aplicación a las cuentas de Google a las que se les de permiso.
Precondiciones^{o}	El usuario debe tener permiso para acceder al sistema.
Pasos secuencia normal	
1	Recoger los datos del formulario de autenticación.
2	Comprobar que tiene permiso para acceder a la aplicación.
3	Acceder al sistema.
Postcondiciones	El usuario entra en la aplicación.
Exenciones	Se le deniega el acceso al sistema porque no tiene permiso. No se puede dar de alta el lote porque no se han rellenado todos los campos obligatorios.

Requisito Funcional

Código	RF-02
Nombre	Crear Lotes
Versión	1.0 (07-05-2015)
Autores	David Segarra Tejedo
Descripción	El sistema debe permitir dar de alta lotes, así como su posterior modificación o eliminación.

Pasos secuencia normal

1	Opcionalmente se puede comprobar con las opciones de búsqueda si el lote estaba dado de alta previamente.
2	Introducir los datos del nuevo lote.
3	Guardar datos.

Postcondiciones	El lote ha sido dado de alta.
Exenciones	No se puede registrar el lote porque ya estaba dado de alta. No se puede dar de alta el lote porque no se han rellenado todos los campos obligatorios.

Requisito Funcional

Código	RF-03
Nombre	Buscar Lotes
Versión	1.0 (07-05-2015)
Autores	David Segarra Tejedo
Descripción	El sistema debe permitir realizar consultas de los lotes.
Precondiciones	El lote debe existir para que la búsqueda tenga éxito.

Pasos secuencia normal

1	Introducir el nombre del lote que se desea buscar
2	Realizar consulta

Postcondiciones	Obtención de un listado con los lotes encontrados
Exenciones	No hay lotes que cumplan los requisitos.

Requisito Funcional

Código	RF-04
Nombre	Editar Lotes
Versión	1.0 (07-05-2015)
Autores	David Segarra Tejedo
Descripción	El sistema debe permitir modificar los datos de los lotes ya registrados
Precondiciones	El lote debe existir para que la edición tenga éxito.

Pasos secuencia normal

1	Realizar búsqueda del lote que se desea editar.
2	Seleccionar el lote.
3	Seleccionar la/las características que se desean modificar.
4	Cambiar los datos.
5	Guardar los cambios.

Postcondiciones	El lote queda modificado
Exenciones	No se puede editar el lote porque no estaba dado de alta.

Requisito Funcional

Código	RF-05
Nombre	Eliminar Lotes
Versión	1.0 (07-05-2015)
Autores	David Segarra Tejedo
Descripción	El sistema debe permitir eliminar lotes ya registrados.
Precondiciones	El lote debe existir para que la eliminación tenga éxito.

Pasos secuencia normal

1	Realizar búsqueda del lote que se desea eliminar.
2	Seleccionar el lote.
3	Eliminar lote.

Postcondiciones	El lote queda eliminado de la base de datos.
Exenciones	No se puede eliminar el lote porque no estaba dado de alta.

Requisito Funcional

Código	RF-06
Nombre	Crear Materias primas
Versión	1.0 (07-05-2015)
Autores	David Segarra Tejedo
Descripción	El sistema debe permitir dar de alta materias primas, así como su posterior modificación o eliminación.

Pasos secuencia normal

1	Opcionalmente se puede comprobar con las opciones de búsqueda si la materia prima estaba dada de alta previamente.
2	Introducir los datos de la nueva materia prima.
3	Guardar datos.

Postcondiciones	La materia prima ha sido dada de alta.
Exenciones	No se puede registrar la materia prima porque ya estaba dada de alta. No se puede dar de alta la materia prima porque no se han rellenado todos los campos obligatorios.

Requisito Funcional

Código	RF-07
Nombre	Buscar Materias Primas
Versión	1.0 (07-05-2015)
Autores	David Segarra Tejedo
Descripción	El sistema debe permitir realizar consultas de las materias primas.
Precondiciones	La materia prima debe existir para que la búsqueda tenga éxito.

Pasos secuencia normal

1	Introducir el nombre de la materia prima que se desea buscar
2	Realizar consulta

Postcondiciones	Obtención de un listado con las materias primas encontradas.
Exenciones	No hay materias primas que cumplan los requisitos.

Requisito Funcional

Código	RF-08
Nombre	Editar Materias primas
Versión	1.0 (07-05-2015)
Autores	David Segarra Tejedo

Descripción	El sistema debe permitir modificar los datos de las materias primas ya registradas.
Precondiciones	La materia prima debe existir para que la edición tenga éxito.

Pasos secuencia normal

1	Realizar búsqueda de la materia prima que se desea editar.
2	Seleccionar el materia prima.
3	Seleccionar la/las características que se desean modificar.
4	Cambiar los datos.
5	Guardar los cambios.

Postcondiciones	La materia prima queda modificada.
Exenciones	No se puede editar la materia prima porque no estaba dada de alta.

Requisito Funcional

Código	RF-09
Nombre	Eliminar Materias Primas
Versión	1.0 (07-05-2015)
Autores	David Segarra Tejedo
Descripción	El sistema debe permitir eliminar materias primas ya registradas.
Precondiciones	La materia prima debe existir para que la eliminación tenga éxito.

Pasos secuencia normal

1	Realizar búsqueda de la materia prima que se desea eliminar.
2	Seleccionar la materia prima.
3	Eliminar materia prima.

Postcondiciones	La materia prima queda eliminada de la base de datos.
Exenciones	No se puede eliminar la materia prima porque no estaba dado de alta.

Requisito Funcional

Código	RF-10
Nombre	Crear Compras
Versión	1.0 (07-05-2015)
Autores	David Segarra Tejedo
Descripción	El sistema debe permitir dar de alta compras, así como su posterior modificación o eliminación.
Pasos secuencia normal	
1	Opcionalmente se puede comprobar con las opciones de búsqueda si la compra estaba dada de alta previamente.
2	Introducir los datos de la nueva compra.
3	Guardar datos.
Postcondiciones	La compra ha sido dada de alta.
Exenciones	No se puede registrar la compra porque ya estaba dada de alta. No se puede dar de alta la compra porque no se han rellenado todos los campos obligatorios.

Requisito Funcional

Código	RF-11
Nombre	Buscar Compras
Versión	1.0 (07-05-2015)
Autores	David Segarra Tejedo
Descripción	El sistema debe permitir realizar consultas de las compras.
Precondiciones	La compra debe existir para que la búsqueda tenga éxito.
Pasos secuencia normal	
1	Introducir el nombre de la compra que se desea buscar.
2	Realizar consulta.
Postcondiciones	Obtención de un listado con las compras encontradas.
Exenciones	No hay compras que cumplan los requisitos.

Requisito Funcional

Código	RF-12
Nombre	Editar Compras
Versión	1.0 (07-05-2015)
Autores	David Segarra Tejedo
Descripción	El sistema debe permitir modificar los datos de las compras ya registradas.
Precondiciones	La compra debe existir para que la edición tenga éxito.

Pasos secuencia normal

1	Realizar búsqueda de la compra que se desea editar.
2	Seleccionar la compra.
3	Seleccionar la/las características que se desean modificar.
4	Cambiar los datos.
5	Guardar los cambios.
Postcondiciones	La compra queda modificada.
Exenciones	No se puede editar la compra porque no estaba dada de alta.

Requisito Funcional

Código	RF-13
Nombre	Eliminar Compras
Versión	1.0 (07-05-2015)
Autores	David Segarra Tejedo
Descripción	El sistema debe permitir eliminar compras ya registradas.
Precondiciones	La compra debe existir para que la eliminación tenga éxito.

Pasos secuencia normal

1	Realizar búsqueda de la compra que se desea eliminar.
2	Seleccionar la compra.
3	Eliminar compra.

Postcondiciones	La compra queda eliminada de la base de datos.
Exenciones	No se puede eliminar la compra porque no estaba dado de alta.

Requisito Funcional

Código	RF-14
Nombre	Crear Consumos
Versión	1.0 (07-05-2015)
Autores	David Segarra Tejedo
Descripción	El sistema debe permitir dar de alta consumos, así como su posterior modificación o eliminación.

Pasos secuencia normal	
1	Introducir los datos del nuevo consumo.
2	Guardar datos.

Postcondiciones	El consumo ha sido dado de alta.
Exenciones	No se puede dar de alta el consumo porque no se han rellenado todos los campos obligatorios.

Requisito Funcional

Código	RF-15
Nombre	Buscar Consumos
Versión	1.0 (07-05-2015)
Autores	David Segarra Tejedo
Descripción	El sistema debe permitir realizar consultas de los consumos.
Precondiciones	El consumo debe existir para que la búsqueda tenga éxito.

Pasos secuencia normal	
1	Introducir el nombre del consumo que se desea buscar.
2	Realizar consulta.

Postcondiciones	Obtención de un listado con los consumos encontrados
Exenciones	No hay consumos que cumplan los requisitos.

Requisito Funcional

Código	RF-16
Nombre	Editar Consumos
Versión	1.0 (07-05-2015)
Autores	David Segarra Tejedo
Descripción	El sistema debe permitir modificar los datos de los consumo ya registrados.
Precondiciones	El consumo debe existir para que la edición tenga éxito.

Pasos secuencia normal

1	Realizar búsqueda del consumo que se desea editar.
2	Seleccionar el consumo.
3	Seleccionar la/las características que se desean modificar.
4	Cambiar los datos.
5	Guardar los cambios.
Postcondiciones	El consumo queda modificado
Exenciones	No se puede editar el consumo porque no estaba dado de alta.

Requisito Funcional

Código	RF-17
Nombre	Eliminar Consumos
Versión	1.0 (07-05-2015)
Autores	David Segarra Tejedo
Descripción	El sistema debe permitir eliminar consumos ya registrados.
Precondiciones	El consumo debe existir para que la eliminación tenga éxito.

Pasos secuencia normal

1	Realizar búsqueda del consumo que se desea eliminar.
2	Seleccionar el consumo.
3	Eliminar consumo.

Postcondiciones	El consumo queda eliminado de la base de datos.
Exenciones	No se puede eliminar el consumo porque no estaba dado de alta.

Requisito Funcional

Código	RF-18
Nombre	Crear Fichas Vacas

Versión	1.0 (07-05-2015)
Autores	David Segarra Tejedo
Descripción	El sistema debe permitir dar de alta fichas de vacas, así como su posterior modificación o eliminación.

Pasos secuencia normal

1	Opcionalmente se puede comprobar con las opciones de búsqueda si la ficha estaba dada de alta previamente.
2	Introducir los datos de la nueva ficha.
3	Guardar datos.

Postcondiciones	La ficha ha sido dada de alta.
Exenciones	No se puede registrar la ficha porque ya estaba dada de alta. No se puede dar de alta la ficha porque no se han rellenado todos los campos obligatorios.

Requisito Funcional

Código	RF-19
Nombre	Buscar Fichas Vacas

Versión	1.0 (07-05-2015)
Autores	David Segarra Tejedo
Descripción	El sistema debe permitir realizar consultas de las fichas.
Precondiciones	La ficha debe existir para que la búsqueda tenga éxito.

Pasos secuencia normal	
1	Introducir el nombre de la ficha que se desea buscar.
2	Realizar consulta.
Postcondiciones	Obtención de un listado con las fichas encontradas.
Exenciones	No hay fichas que cumplan los requisitos.

Requisito Funcional

Código	RF-20
Nombre	Editar Fichas Vacas
Versión	1.0 (07-05-2015)
Autores	David Segarra Tejedo
Descripción	El sistema debe permitir modificar los datos de las fichas ya registradas.
Precondiciones	La ficha debe existir para que la edición tenga éxito.

Pasos secuencia normal	
1	Realizar búsqueda de la ficha que se desea editar.
2	Seleccionar la ficha.
3	Seleccionar la/las características que se desean modificar.
4	Cambiar los datos.
5	Guardar los cambios.
Postcondiciones	La ficha queda modificada.
Exenciones	No se puede editar la ficha porque no estaba dada de alta.

Requisito Funcional

Código	RF-21
Nombre	Eliminar Fichas Vacas
Versión	1.0 (07-05-2015)
Autores	David Segarra Tejedo
Descripción	El sistema debe permitir eliminar fichas ya registradas.
Precondiciones	La ficha debe existir para que la eliminación tenga éxito.

Pasos secuencia normal	
1	Realizar búsqueda de la ficha que se desea eliminar.
2	Seleccionar la ficha.
3	Eliminar ficha.
Postcondiciones	La ficha queda eliminada de la base de datos.
Exenciones	No se puede eliminar la ficha porque no estaba dado de alta.

Requisito Funcional

Código	RF-22
Nombre	Crear Fichas Corderos
Versión	1.0 (07-05-2015)
Autores	David Segarra Tejedo
Descripción	El sistema debe permitir dar de alta fichas de corderos, así como su posterior modificación o eliminación.
Pasos secuencia normal	
1	Opcionalmente se puede comprobar con las opciones de búsqueda si la ficha estaba dada de alta previamente.
2	Introducir los datos de la nueva ficha.
3	Guardar datos.
Postcondiciones	La ficha ha sido dada de alta.
Exenciones	No se puede registrar la ficha porque ya estaba dada de alta. No se puede dar de alta la ficha porque no se han rellenado todos los campos obligatorios.

Requisito Funcional

Código	RF-23
Nombre	Buscar Fichas Corderos
Versión	1.0 (07-05-2015)
Autores	David Segarra Tejedo

Descripción	El sistema debe permitir realizar consultas de las fichas.
Precondiciones	La ficha debe existir para que la búsqueda tenga éxito.

Pasos secuencia normal	
1	Introducir el nombre de la ficha que se desea buscar.
2	Realizar consulta.

Postcondiciones	Obtención de un listado con las fichas encontradas.
Exenciones	No hay fichas que cumplan los requisitos.

Requisito Funcional

Código	RF-24
Nombre	Editar Fichas Corderos
Versión	1.0 (07-05-2015)
Autores	David Segarra Tejedo
Descripción	El sistema debe permitir modificar los datos de las fichas ya registradas.
Precondiciones	La ficha debe existir para que la edición tenga éxito.

Pasos secuencia normal	
1	Realizar búsqueda de la ficha que se desea editar.
2	Seleccionar la ficha.
3	Seleccionar la/las características que se desean modificar.
4	Cambiar los datos.
5	Guardar los cambios.

Postcondiciones	La ficha queda modificada.
Exenciones	No se puede editar la ficha porque no estaba dada de alta.

Requisito Funcional

Código	RF-25
Nombre	Eliminar Fichas Corderos
Versión	1.0 (07-05-2015)
Autores	David Segarra Tejedo

Descripción	El sistema debe permitir eliminar fichas ya registradas.
Precondiciones	La ficha debe existir para que la eliminación tenga éxito.

Pasos secuencia normal

1	Realizar búsqueda de la ficha que se desea eliminar.
2	Seleccionar la ficha.
3	Eliminar ficha.

Postcondiciones	La ficha queda eliminada de la base de datos.
Exenciones	No se puede eliminar la ficha porque no estaba dado de alta.

Requisito Funcional

Código	RF-26
Nombre	Crear Gastos Fijos
Versión	1.0 (07-05-2015)
Autores	David Segarra Tejedo
Descripción	El sistema debe permitir dar de alta gastos fijos, así como su posterior modificación o eliminación.

Pasos secuencia normal

1	Introducir los datos del nuevo gasto fijo.
2	Guardar datos.

Postcondiciones	El gasto fijo ha sido dado de alta.
Exenciones	No se puede dar de alta el gasto fijo porque no se han rellenado todos los campos obligatorios.

Requisito Funcional

Código	RF-27
Nombre	Buscar Gastos Fijos
Versión	1.0 (07-05-2015)
Autores	David Segarra Tejedo
Descripción	El sistema debe permitir realizar consultas de los gastos fijos.

Precondiciones	El gasto fijo debe existir para que la búsqueda tenga éxito.

Pasos secuencia normal	
1	Introducir el nombre del gasto fijo que se desea buscar.
2	Realizar consulta.

Postcondiciones	Obtención de un listado con los gastos fijos encontrados
Exenciones	No hay gastos fijos que cumplan los requisitos.

Requisito Funcional

Código	RF-28
Nombre	Editar Gastos fijos

Versión	1.0 (07-05-2015)
Autores	David Segarra Tejedo
Descripción	El sistema debe permitir modificar los datos de los gastos fijos ya registrados.
Precondiciones	El gasto fijo debe existir para que la edición tenga éxito.

Pasos secuencia normal	
1	Realizar búsqueda del gasto fijo que se desea editar.
2	Seleccionar el gasto fijo.
3	Seleccionar la/las características que se desean modificar.
4	Cambiar los datos.
5	Guardar los cambios.

Postcondiciones	El gasto fijo queda modificado
Exenciones	No se puede editar el gasto fijo porque no estaba dado de alta.

Requisito Funcional

Código	RF-29
Nombre	Eliminar Gastos Fijos

Versión	1.0 (07-05-2015)
Autores	David Segarra Tejedo
Descripción	El sistema debe permitir eliminar gastos fijos ya registrados.
Precondiciones	El gasto fijo debe existir para que la eliminación tenga éxito.

Pasos secuencia normal	
1	Realizar búsqueda del gasto fijo que se desea eliminar.
2	Seleccionar el gasto fijo.
3	Eliminar gasto fijo.

Postcondiciones	El gasto fijo queda eliminado de la base de datos.
Exenciones	No se puede eliminar el gasto fijo porque no estaba dado de alta.

Requisito Funcional

Código	RF-30
Nombre	Re-calcular Costes Consumos

Versión	1.0 (07-05-2015)
Autores	David Segarra Tejedo
Descripción	Calcular el valor de la columna “importe” para cada registro de “consumos”. Se calcula a partir de los “kg” para el “lote”, “t” tipo y “fecha”, con respecto al valor de compra de la materia prima “idmt” a esa fecha.
Algoritmo	Se trata de calcular la valoración de los “kg” introducidos en “consumos” que dependiendo de la “consumos.fecha” introducida iremos a “compras” y para la materia prima obtendremos el importe que corresponde a “consumos.kg” x “compras.preciokg”. La complejidad reside en que las compras se van introduciendo periódicamente y para una misma materia prima el precio va cambiando dependiendo de la fecha de compra. Con esta variación el “compras.preciokg” va variando con lo que la valoración de los consumos también. Para obtener el “preciokg” que corresponde a la fecha “consumos.fecha” se obtienen la relación de “compras” ordenados por fecha de la materia prima y se hace uso de “compras.asignado” para descontar los “consumos.kg”. Cuando el valor de

	“compras.asignado” es igual a “compras.Kgs” se pasa al siguiente registro de “compras” para esa materia prima. El valor de “consumos.norecalcular” indica que el registro ya ha sido valorado y no es necesario realizar el recálculo. Con ello se agiliza el proceso de valoración sólo para los que “consumos.norecalcular” es vacío.

Requisito Funcional

Código	RF-31
Nombre	Re-calcular Costes Vacas Tramos
Versión	1.0 (07-05-2015)
Autores	David Segarra Tejedo
Descripción	Calcular el valor de las columnas “gastosfijos” y “costesconsumo” para cada registro de “FichasVacas”. Intervienen “lote”, “fechacompra”, “fechaventa” y “fechaperdida”. Se calcula el consumo en valor de importe para cada registro de “FichasVacas” desde “fechacompra” hasta “fechaproceso” (=“fechaventa” o =“fechaperdida” o =fechaactual) teniendo en cuenta el lote al que corresponde y el número de vacas “activas” (“fechaventa” y “fechaperdida” vacías) a fecha “fechaproceso”.
Algoritmo	El valor de “costeconsumo”, referido a una vaca y lote, se calcula como la suma de consumos de ese lote desde “fechacompra” hasta la “fechaproceso”. En este periodo cada consumo se ha de dividir entre el número de vacas “activas” a fecha consumo. Cada vez que “fechaventa” o “fechaperdida” se rellenan el número de vacas activas a esa fecha cambian, por lo que los consumos a esa fecha también. Para este cálculo se construye un array de “fechasevento”, esto es cada vez que una fecha influye en el número de vacas activas. Para cada “fechaventa” o “fechaperdida” (en su defecto fechaproceso=fechaactual” habrá un registro en este array. Estos registros llevarán los cálculos parciales (tramos) de cada cálculo de consumo realizado entre el número de vacas a esa fecha. La suma de los tramos anteriores dará la valoración de

	“costesconsumo”. “gastosfijos” se va calculando de igual forma y a la vez que “costesconsumo” pero referido al número de vacas y corderos activos a fecha de cálculo en global, independientemente del lote al que pertenezcan, ya que en la ficha “GastosFijos” los valores se refieren al global de vacas y corderos.

Requisito Funcional

Código	RF-32
Nombre	Re-calcular Costes Corderos Tramos
Versión	1.0 (07-05-2015)
Autores	David Segarra Tejedo
Descripción	Calcular el valor de las columnas “gastosfijos” y “costesconsumo” para cada registro de “FichasCorderos”. Intervienen “lote”, “fechacompra”, “fechaventa” y “fechaperdida”. Se calcula el consumo en valor de importe para cada registro de “FichasCorderos ” desde “fechacompra” hasta “fechaproceso” (=“fechaventa” o =“fechaperdida” o =fechaactual) teniendo en cuenta el lote al que corresponde y el número de vacas “activas” (“fechaventa” y “fechaperdida” vacías) a fecha “fechaproceso”.
Algoritmo	Igual que **Re-calcular Costes Vacas Tramos** pero para “FichasCorderos”.