**Equipo 8 - ciclo 4**

**Integrantes:**

Raúl Balaguera

María Gónzalez

Santiago Jaramillo

Santiago Sanabria

Mauricio Murcia

Daniel Pinto

**Selección de categoría**

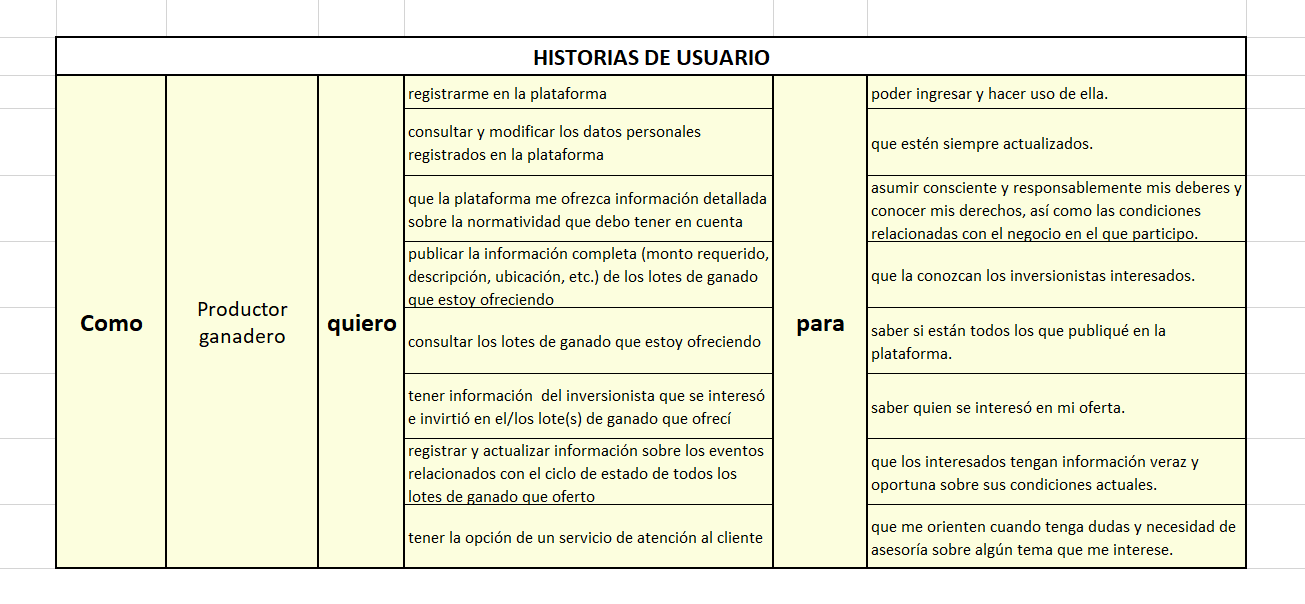
* Tienda virtual

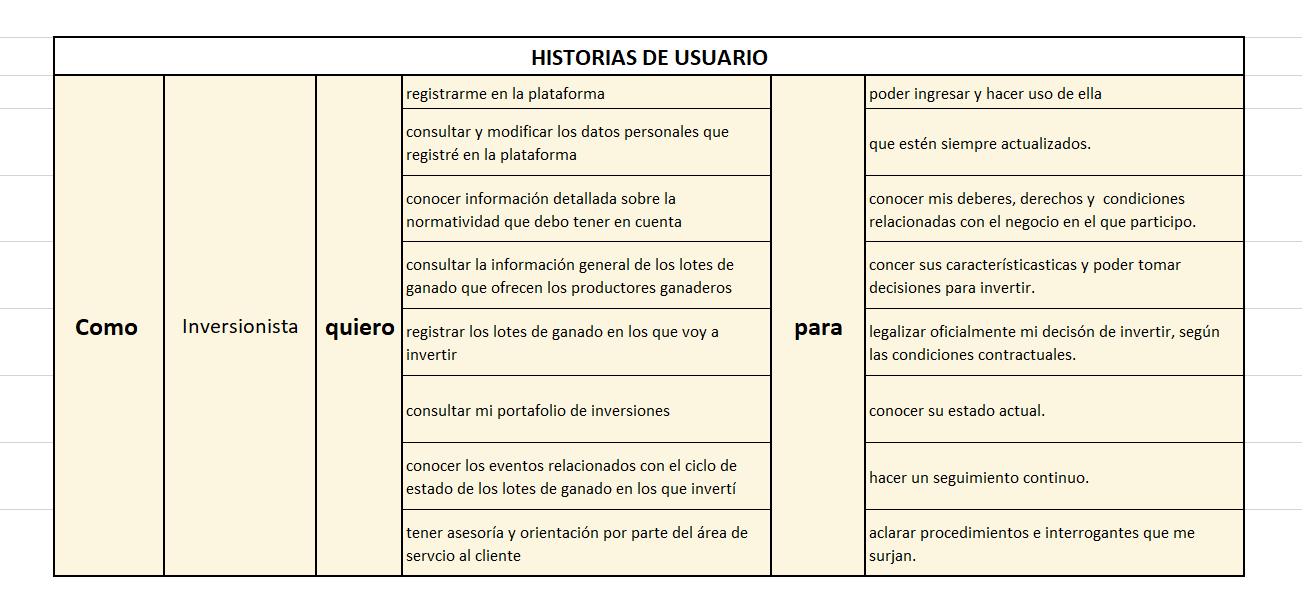
**Justificación**

La producción ganadera con terceros se ha llevado a cabo relacionando el capital dispuesto por uno o más inversores junto con la mano de obra dispuesta por el productor ganadero, quienes acuerdan previamente el reparto porcentual sobre las ganancias generadas tras la venta final de las cabezas de ganado. En ocasiones, el círculo social de los productores no cuenta con recursos financieros para participar en empresas ganaderas, por lo que deben recurrir a otras fuentes de financiación o posponer sus planes productivos. A la par de esta problemática, se encuentran personas con recursos destinados a la inversión que no cuentan con alternativas de inversión en este sector.

En este sentido, surge la realización de un medio virtual en el cual se permita comercializar ganado y transar inversiones de negocios, acercando a productores e inversores interesados, con el fin de obtener financiación de terceros y rentabilidades futuras. Así mismo, se plantea la realización futura de una moneda única de transacción denominada: “Vaca Coin”.

**Historias de usuario**







**Versión 1.0 arquitectura web**

**OG:**

Desarrollar e implementar Api web para el control y la administración de la ganadería por intermedio de la Api Web.

**Registro**:

Tener un registro de las granjas de nuestros campesinos con sus respectivos datos de tierra, ganado, tipo de ganado, alimento aplicado, vacuna, venta, animales de producción.

**Productor:**

Plataforma digital de ganado (intermediario)

**Servidor Web:**

Se desarrolla la api en el lado del servidor realizando conexiones bidireccionales y/o unidireccionales, sincrónicas y asincrónicas con el cliente generando así una respuesta de la api para el cliente. (Protocolos, HTTP Modelo OSI)

La Api Web utilizara el protocolo HTTP para comunicarse a través de una conexión TCP, es por esto que se puede entender como un sistema petición/respuesta como el típico Cliente/Servidor y este si o si debe correr como cualquier tipo de red.

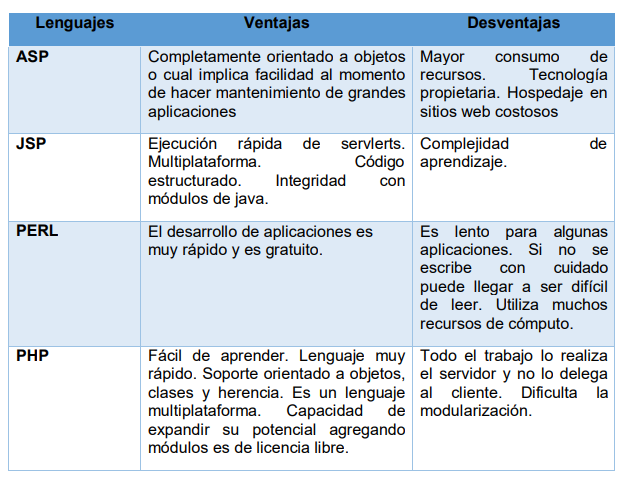
**Lenguajes del lado del Servidor:**

Existen varios lenguajes de programación que nos permite programar del lado del servidor cada uno de estos explota más fondo ciertas características que lo hacen más útil o menos útil para el desarrollo de la Api Web.

La página se ejecuta del lado del servidor pueden realizar acceso a base de datos, conexión en red, y otras funcionalidades para crear la página final que verá el cliente y/o usuario.

Los lenguajes que se pueden usar PHP, Python, Java (cuales más metemos?)

tablita si si o si no coloco tablita



**Bases de datos**

Motores de Bases de datos:

Este es el servidor principal para almacenar, procesar y proteger los datos. El motor de base de datos proporciona acceso controlado y procesamiento de transacciones rápidas para cumplir con los requisitos de las aplicaciones.

Los motores es la creación de bases de datos relacionadas para el procesamiento de transacciones en línea o datos de procesamiento analíticos. Se puede crear tablas para almacenar datos y objetos de base de datos como índices, vistas y procedimientos, estos motores funcionan muy bien para administrar y proteger los datos.

Mongodb

Sistema de base de datos NoSQL, orientado a documentos y de código abierto. En lugar de guardar los datos de tabla, tal como se hacen en las bases de datos relacionadas.

**HERRAMIENTAS TECNOLÓGICAS**

GraphQl

Lenguaje de consulta y manipulación de datos para Apis, su entorno de ejecución es realizar consultas con datos existentes. (Creado por FB)

PHP

Lenguaje de programación de uso general de código del lado del servidor originalmente diseñado para el desarrollo web de contenido dinámico.

HTML 5

Es la quinta revisión importante del lenguaje básico de la World Wide Web, HTML, siglas de HyperText Markup Language (lenguaje de marcas de hipertexto), hace referencia al lenguaje de marcado para la elaboración de páginas web

CSS3

Es un lenguaje de hojas de estilos Hoja de estilo en cascada o CSS (siglas en inglés de cascading style sheets) creado para controlar el aspecto o presentación de los documentos electrónicos definidos con HTML y XHTML.

Bootstrap

Es un framework o conjunto de herramientas de software libre para diseño de sitios y aplicaciones web. Contiene plantillas de diseño con tipografía, formularios, botones, cuadros, menús de navegación y otros elementos de diseño basado en HTML y CSS, así como, extensiones de JavaScript opcionales.

JavaScript

Es un lenguaje de programación que se utiliza principalmente para crear páginas web dinámicas. Una página web dinámica es aquella que incorpora efectos como texto que aparece y desaparece, animaciones, acciones que se activan al pulsar botones y ventanas con mensajes de aviso al usuario.

Jquery

Es una biblioteca de JavaScript, creada inicialmente por John Resig, que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web.

Angular

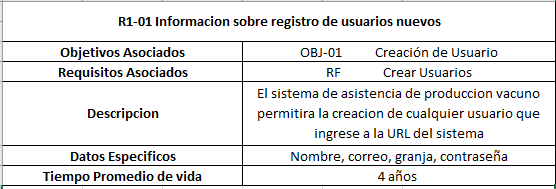
Es un framework de JavaScript de código abierto, mantenido por Google, que ayuda con la gestión de lo que se conoce como aplicaciones de una sola página. Su objetivo es aumentar las aplicaciones basadas en navegador con capacidad de Modelo Vista Controlador (MVC), en un esfuerzo para hacer que el desarrollo y las pruebas sean más fáciles.

JSON

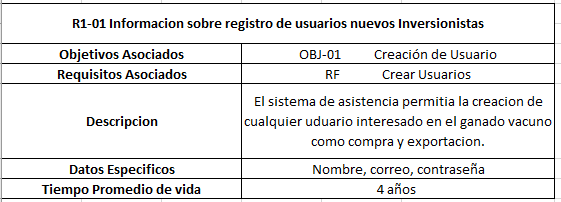
Acrónimo de JavaScript Object Notation- Notación de Objetos de JavaScript, es un formato ligero para el intercambio de datos. JSON es un subconjunto de la notación literal de objetos de JavaScript que no requiere el uso de XML

**Requisitos de la Información:**

**Registro de usuarios nuevos (ganadero)**

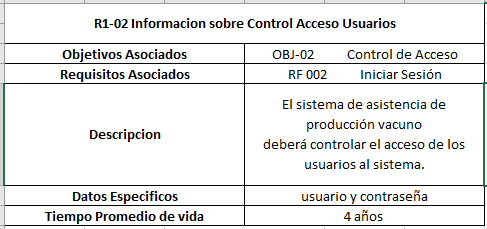
****

**Registro de usuarios nuevos (inversionistas)**

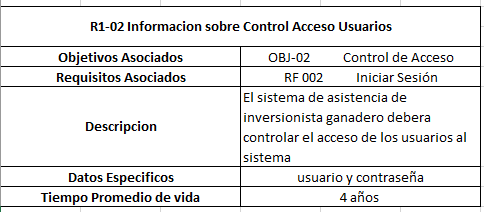
****

**Control de Acceso Usuarios**

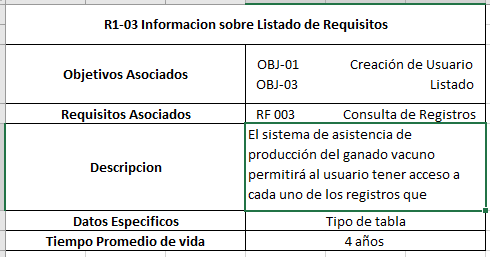
**Control de usuarios (ganadero)**

****

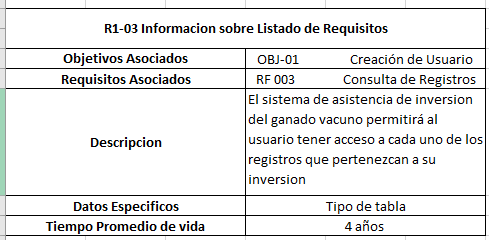
**Control de usuarios (inversionista)**

****

**Listado de Requisitos (ganadero)**

****

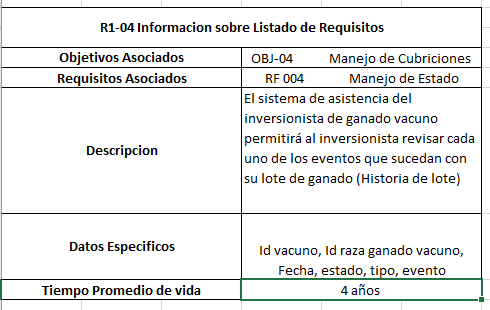
**Listado de Requisitos (inversionista)**

****

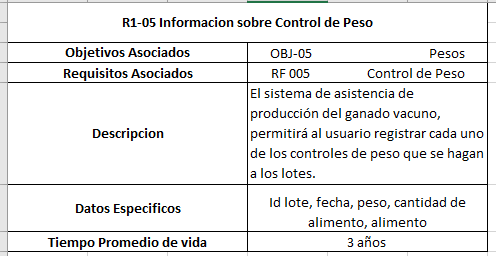
**Gestión de Cubriciones (ganadero)**

****

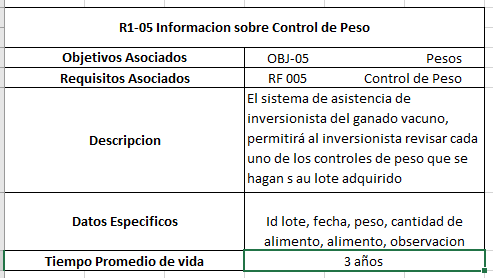
**Gestión de Cubriciones (inversionista)**

****

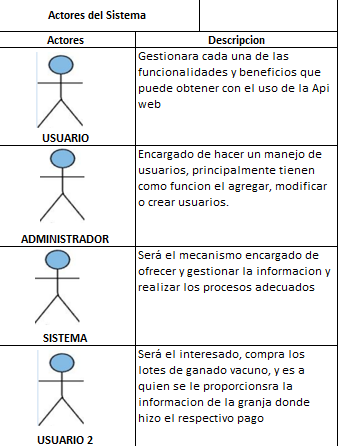
**Control de peso (ganadero)**

****

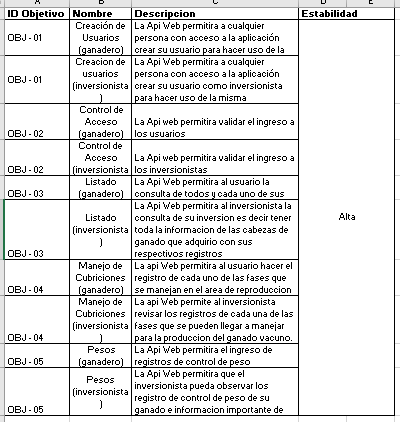
**Control de Peso (inversionista)**

****

**Actores**

****

**Requisitos Funcionales**

****

Arquitectura

Servidor web.. (posible back

BD… (Motores de la BD)

Mongodb

Django

Java

Python

postman

spring

Otras herramientas (posible front):

grhap   
html 5

php

css 3

scss \*

Javascript

Angular \*

Json

Bootstrap

Jquery

**Desarrollo**

**Arquitectura del Sistema:**

El sistema está basado en la arquitectura Usuario/Servidor el cual es el modelo de desarrollo que se emplea en las aplicaciones web

**Capas de los sistemas:**

El primer nivel consiste en la capa de presentación que incluye no solo el navegador o cliente, es decir el equipo que solicita los recursos, sino también el servidor web que es el responsable de presentar los datos de forma adecuada. Para la presentación de la información al usuario es requerido el uso de HTML5, CSS3 Y JavaScript, para modelar cada una de las vistas presentadas.

En el segundo nivel de esta arquitectura se encuentra lo relacionado a los script o programas, también conocidos como servidor de aplicación, cuya tarea es proporcionar los recursos solicitados, esta capa lógica se trabajó con el lenguaje PHP el cual es uno de los más populares en el mundo del desarrollo web y con este se implementó un modelo de desarrollo llamado MVC (Modelo-Vista-Controlador) con el fin de facilitar el acceso a clases y métodos que se implementan a lo largo del desarrollo de la aplicación. 38 Capa 1. Capa 2. Modelo Controlador Vista El último nivel es el encargado de proporcionarle los datos al segundo nivel para su ejecución, esta etapa empieza por el servidor o servicio de gestión de bases de datos. Una aplicación Web trabaja recopilando datos del usuario alojados en el primer nivel y los envía al servidor, que ejecuta un programa, este se encuentra en el segundo nivel procesa los datos y presenta los resultados al usuario el cual se encuentra en el primer nivel. Se presentaría de la siguiente forma. Capa 1: Cliente de Aplicación. Navegador Web Capa 2: Servidor de Aplicaciones. Capa 3: Servidor de Datos

**Arquitectura de Sistema 2:**

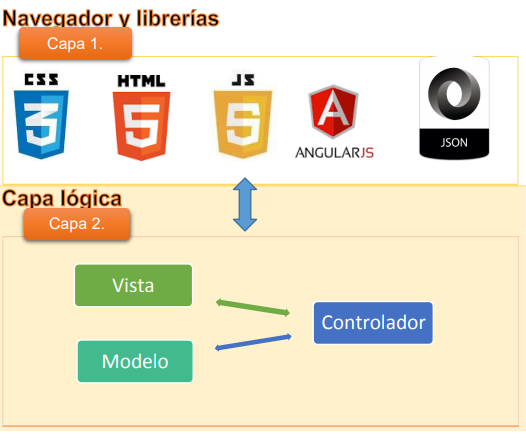
El sistema está basado en la arquitectura Inversionista/Servidor el cual es el modelo de desarrollo que se emplea en las aplicaciones web

**Capas:** Las mismas anteriormente descritas

**Arquitectura de Sistema 3:**

El sistema está basado en la arquitectura Iusuario/Servidor/inversionista el cual es el modelo de desarrollo que se emplea en las aplicaciones web

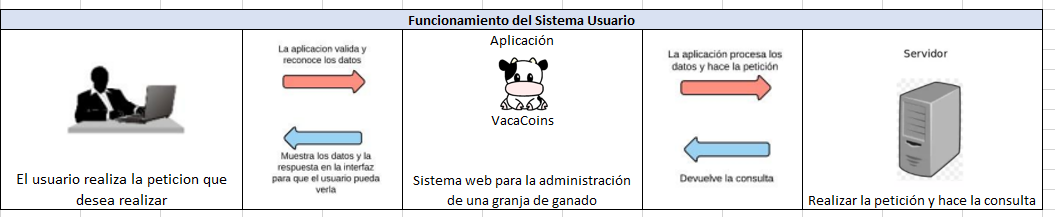
**Modelo de la Arquitectura del sistema compacto:**

****

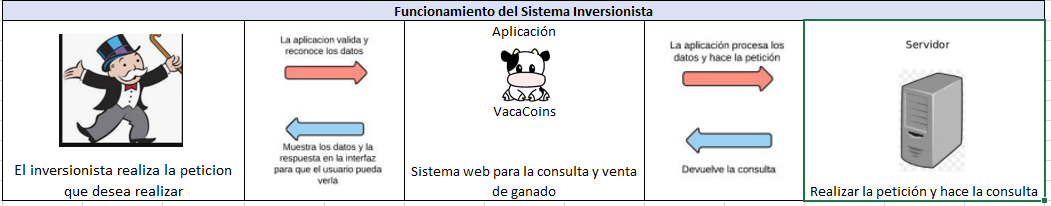
****

**Funcionamiento del Sistema**

**Usuario**

****

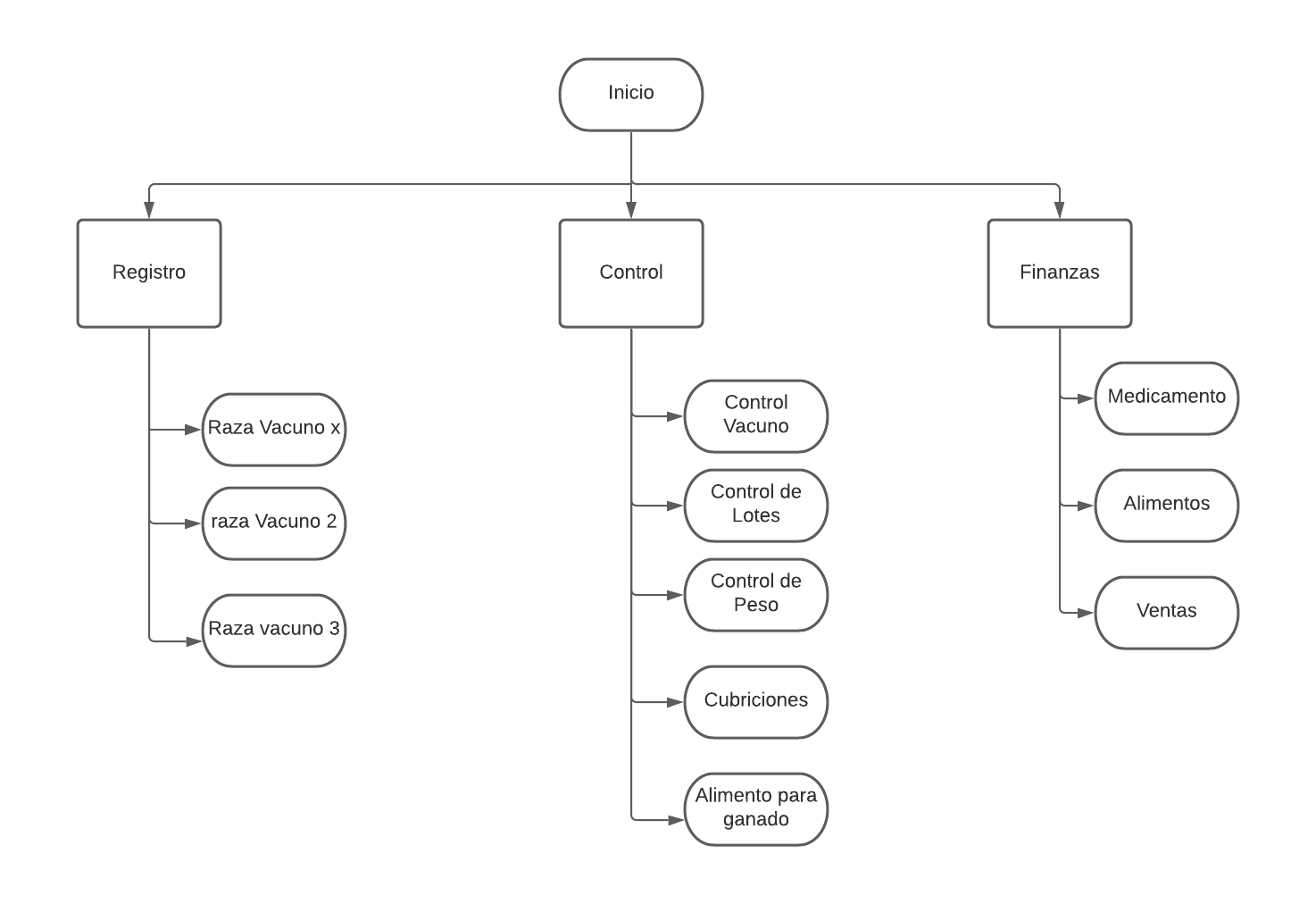
**Inversionista**

****

**Diagrama del Sistema**

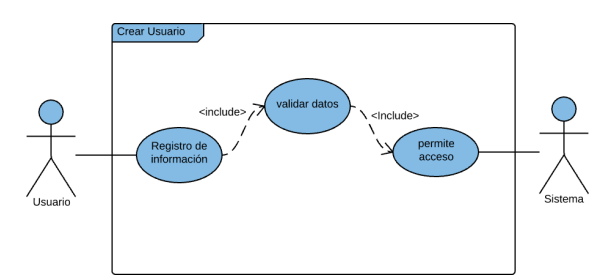
Para la realización y buen entendimiento del sistema se debe tener en cuenta todas las funciones con las que cuenta el sistema, para la realización del proyecto se realizaron diagramas UML para describir el funcionamiento, y así hacerle fácil al usuario conocer las características de la aplicación. Los diagramas los encuentra en anexos.

**Mapa de Navegación**

****

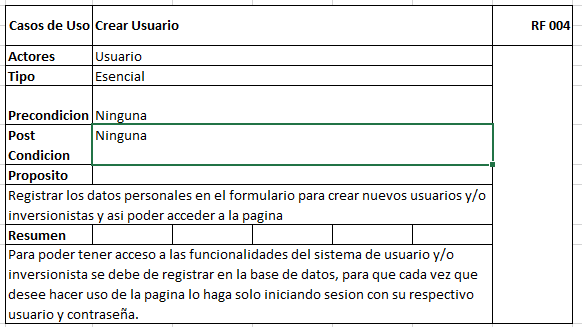
**Diagramas UML**

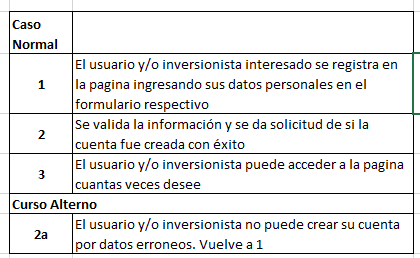
**Diagrama de Caso de Uso**

****

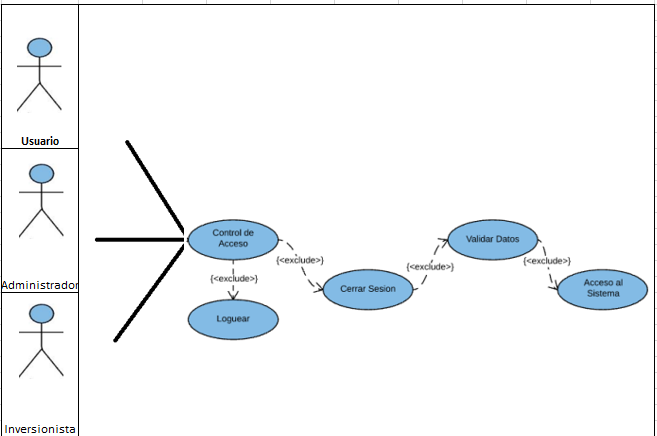
**Casos de Uso**

**Usuario y/o Inversionista**

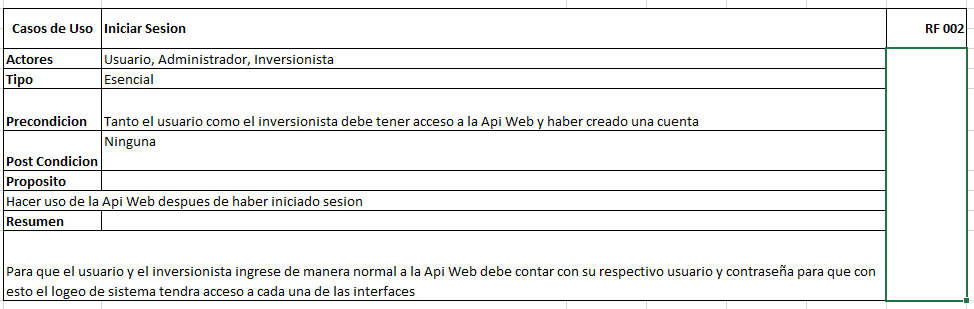


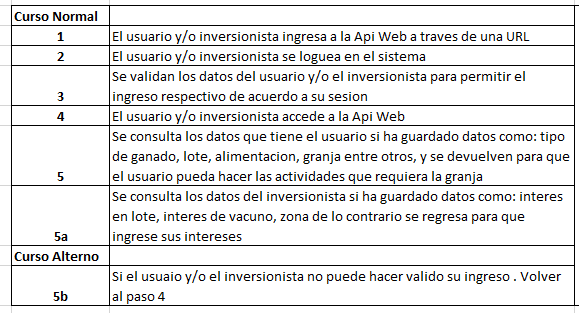


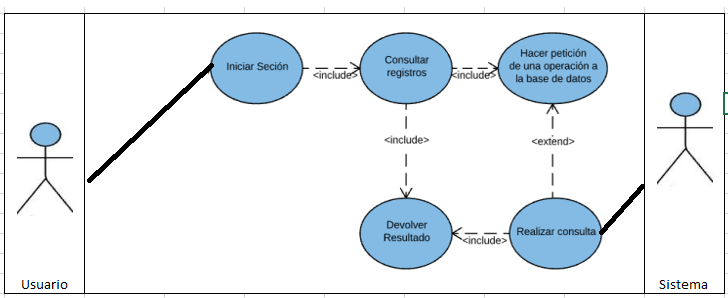
Caso de Uso Al iniciar sesión

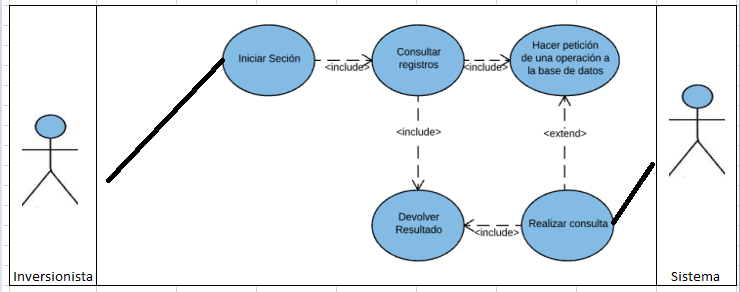
****

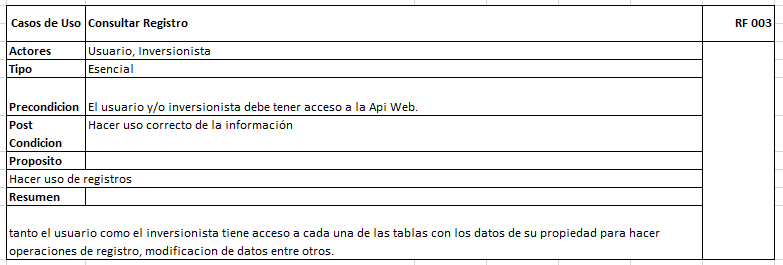
**Caso de Uso al Iniciar Sesión**

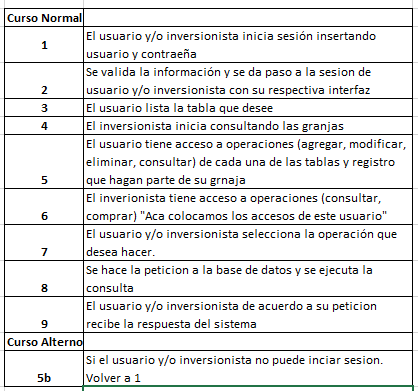




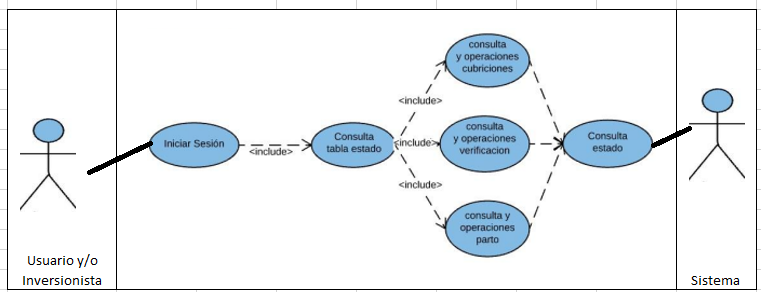




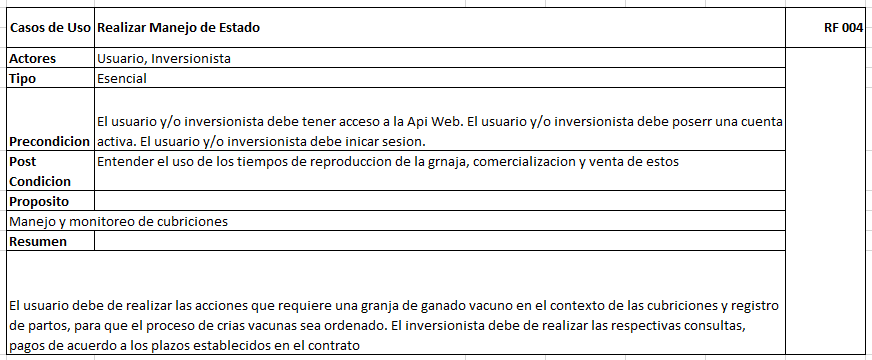
****

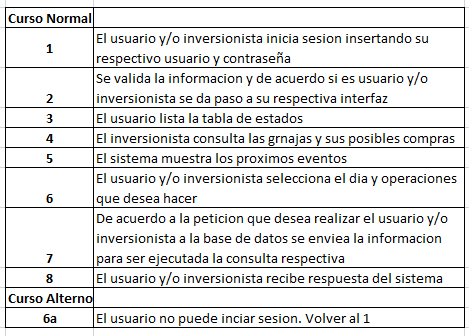
****

**Manejo de estado**

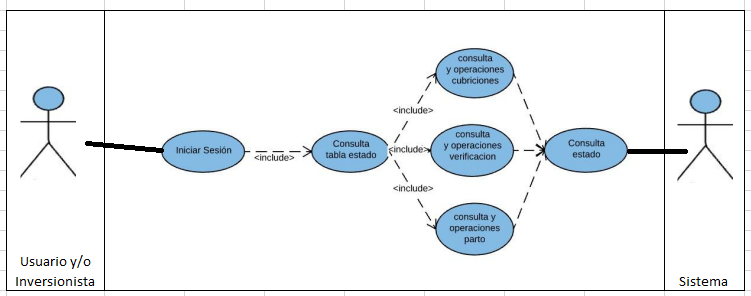
****

**Documentación Caso de Uso realizar Manejo de Estado**

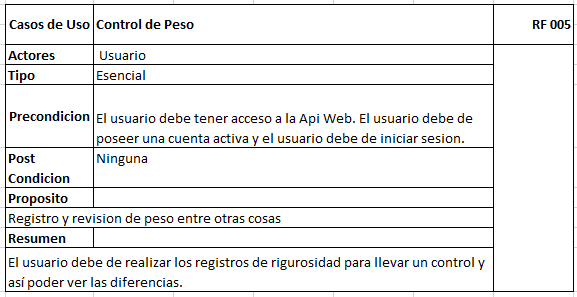


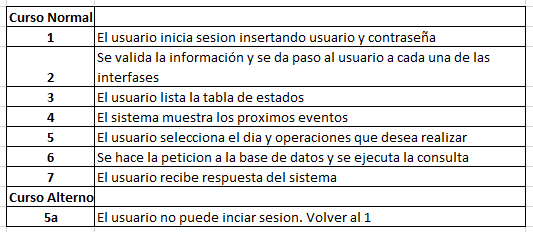


**Caso de uso Control Peso**



**Documentación caso de uso Realizar Manejo de Estado**

****

****

Relaciones y posibles fuentes de información :

Gaviar, Finagro.

ICA, FEDEGAN, Ministerio de Agricultura y desarrollo rural

Requisitos Funcionales

* Generar carrito de compras alusivo al mercadeo de ganado
* Construir pool de proyectos que le permita a los usuarios seleccionar el más conveniente para su prospecto de inversión
* Notificar al usuario respecto a las inversiones realizadas por medio de un correo electrónico
* Realizar transferencias electrónicas
* Registrar cuenta bancaria
* generar ventas por cabeza de ganado
* generar ventas por volumen carne procesada
* Registrar perfil de campesino(productor)
* Validar perfil de campesino
* Generar validación y compromiso de datos en los términos y condiciones de la app
* Genar peril de inverionesionas
* Generar sistema de administración

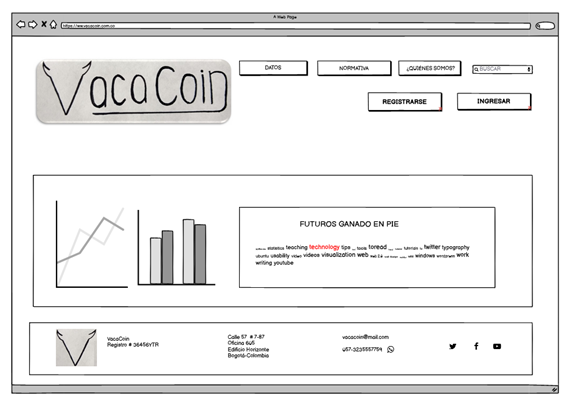
Requisitos no funcionales(FURPS)

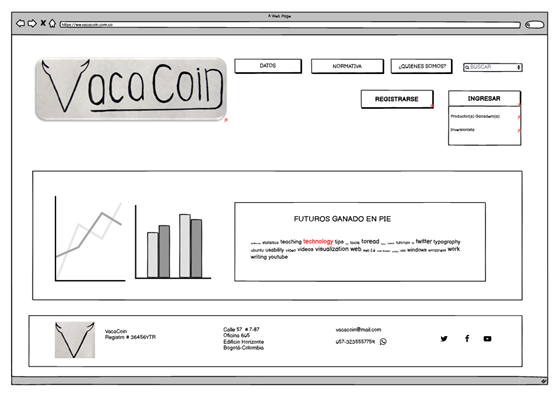
| Rendimiento | Usabilidad | Confiabilidad | Seguridad |
| --- | --- | --- | --- |
| El sistema debe mantener un tiempo de respuesta entre transacciones no mayor a 5 minutos | El sistema debe contar con un interfaz amigable al usuario, tal como: iconos intuitivos , imágenes y gráficos agradables a la vista. | El sistema de mantener consistencia en los datos en todo momento ,esto con la finalidad de que ante fallos en la plataforma no se pierda la información | El sistema debe contar con un control de transacciones es ,decir de no haber comunicación durante una inversión este deberá anular reversa o anular la transacción según sea el caso |
|  |  | el sistema de contar con medidas de tolerancia a fallos |  |

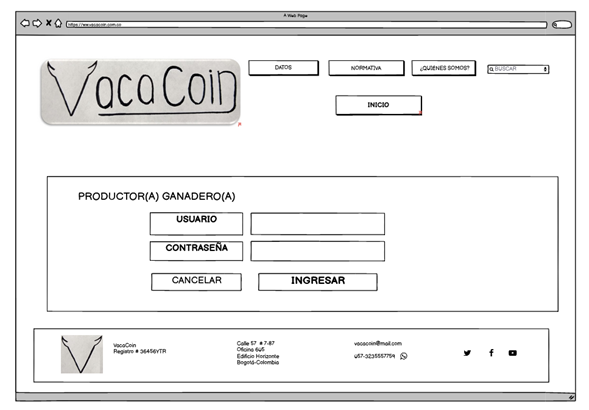
| Mantenibilidad | escalabilidad |
| --- | --- |
| El sistema debe | El sistema ha de estar diseñado para soportar hasta 100 usuarios no concurrentes |

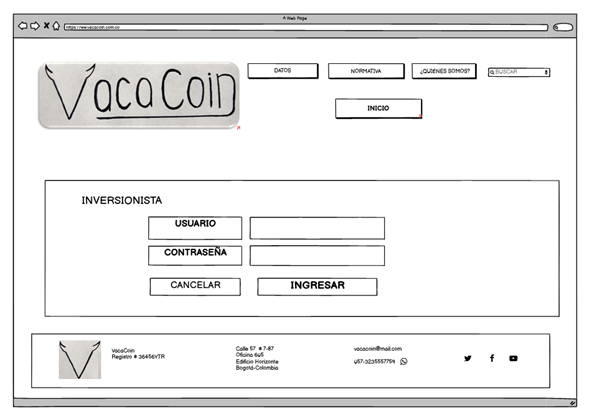
Diagrama UML

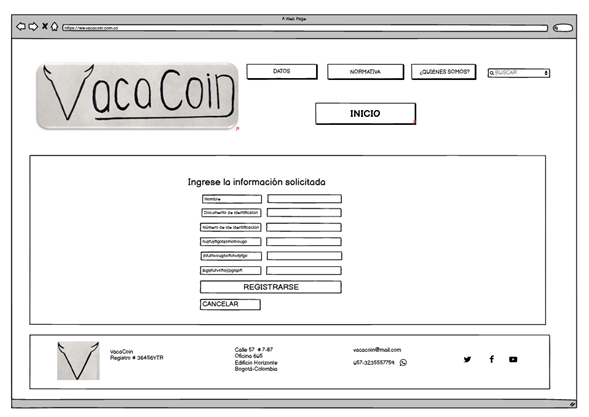
Mockups











Enlace para ver la presentación de los Mockups de VacaCoin en Balsamiq Cloud.

https://balsamiq.cloud/s6o1qkk/pgtxspr/r2278?f=N4IgUiBcAMA0IDkpxAYWfAMlAjPAQrvALJEgBaZA0lANoC6AvkA%3D