**Universidad Tecnológica de Tecámac**

**Empresa:** Blue sky

**Proyecto:** Pet Sitting

**Integrantes:**

* Alonso Pérez Antonio
* Jiménez Rodríguez Lizet
* Larios Soto Kay

**Grupo:** 7ITI2

**Fecha de entrega:** 04-Marzo-2019

Índice

[1. Introducción 3](#_Toc4964579)

[2. Análisis de requerimientos 4](#_Toc4964580)

[2.1 Proceso actual 5](#_Toc4964581)

[2.2 Historias de Usuario 6](#_Toc4964582)

[3. Definición de los requerimientos 16](#_Toc4964583)

[3.1 Requerimientos funcionales 17](#_Toc4964584)

[3.2 Requerimientos no funcionales 18](#_Toc4964585)

[4. Modelos del sistema 19](#_Toc4964586)

[4.1 Estructura de sistema 21](#_Toc4964587)

[4.2 Comportamiento del sistema 22](#_Toc4964588)

[5. Propuesta de desarrollo 23](#_Toc4964589)

[6. Plan de pruebas 24](#_Toc4964590)

[6.1 Planeación 25](#_Toc4964591)

[6.2 Planificación 29](#_Toc4964592)

[7. Prototipo del sistema 30](#_Toc4964593)

[7.1 Diseño inicial 32](#_Toc4964594)

[7.2 Funcionalidad 33](#_Toc4964595)

[Anexo I 34](#_Toc4964596)

[Anexo II 35](#_Toc4964597)

[Anexo III 36](#_Toc4964598)

[Glosario 37](#_Toc4964599)

# Introducción

En la actualidad las mascotas son parte fundamental en la vida de las personas, tanto, que consideran a sus mascotas como parte de su familia, lamentablemente las personas no pueden cuidar de una mascota como se debe por falta de tiempo o por su profesión, hay casos en que las familias salen de vacaciones y los dueños dejan a su mascota solos o son encargados a una persona ajena lo que ocasiona desconfianza a los dueños, por estos motivos se ha decidido crear un dispensador de alimento para la mascota, el cual llene el plato de comida del animal de forma automática con información precargada y además definir la información sobre el tipo de mascota a la que va dirigido. La definición del tipo de mascota y los tiempos en los que se debe dispensar el alimento serán guardados por el usuario mediante una aplicación de escritorio, cabe mencionar que esto solo se realizara una sola vez por el usuario.

Para la elaboración del proyecto es importante hacer el levantamiento de los requerimientos del proyecto porque nos permite analizar las necesidades del usuario como también ayuda a definir los requisitos que debe llevar este proyecto. Se definieron los dos tipos de requerimientos: funcionales y no funcionales.

Así mismo se realizó los diferentes diagramas, modelos y comportamientos, además de la estructura del sistema representada con los diferentes modelos UML evaluados mediante pruebas.

# Análisis de requerimientos

En la ingeniería de software, un análisis de requerimientos es una tarea que cubre el hueco entre la definición del software a nivel sistema y el diseño del mismo. Tanto el desarrollador como el cliente tienen un papel activo, pues juntos definen en detalle los requisitos del sistema a desarrollar y los pasos a seguir.

En otras palabras, el análisis de requisitos abarca aquellas tareas que se dirigen a determinar las necesidades o condiciones que se deben cumplir para un producto o proyecto nuevo o modificado, teniendo en cuenta los requisitos posiblemente conflictivos de los distintos interesados, analizando, documentando, validando y administrando software o requisitos del sistema.

El proceso de análisis de requerimientos refina, modela, especifica y verifica las solicitudes de usuario, y con ello genera documentos base para la ejecución de los procesos siguientes.

Características:

* Es un estudio profundo de una necesidad tecnológica que tiene una empresa, organización o negocio.
* Especifica las características operacionales que tendrá el software a desarrollar.
* Se realiza a través de entrevistas, observación, indagación y demás técnicas específicas.
* Describe el plan del proyecto a seguir.
* Es fundamental entregar el proyecto dentro del tiempo y presupuesto acordados y de los objetivos de negocio.

El análisis de requisitos es crítico para el éxito o el fracaso de un proyecto de sistemas o software.

La calidad de los requisitos se puede mejorar a través de estos y otros métodos como: visualización, uso consistente de plantillas, documentación de dependencias.

## Proceso actual

BLUE SKY es una empresa orientada al desarrollo de software, en estos momentos la empresa está elaborando el proyecto “Pet Sitting”, una aplicación de escritorio en el cual su objetivo es realizar un dispensador de alimento automático para las mascotas

La empresa cuenta con tres personas que realizan las siguientes actividades:

* Líder/Documentador
* Realizar la planeación.
* Levantar los requerimientos.
* Documentar el proceso del proyecto.
  + Diseñar el dispensador de alimento.
* Programador/ Diseñador
  + Desarrollo de la base de datos.
  + Desarrollo de los procesos de la aplicación.
  + Diseño de las interfaces de la aplicación.
* Tester
  + Informe de prueba de la aplicación.
  + Informe de prueba del dispensador.
  + Conexión del dispensador con la aplicación.

Los actores que se involucran durante la ejecución del proyecto y además realizan las siguientes actividades:

* Dueño de la mascota
  + Llenar formulario con la información de la mascota.
* Sistema
  + Llenar el plato de comida del animal.
  + Avisar cuando se agota el alimento.

## Historias de Usuario

Las historias de usuario, son pequeñas descripciones de los requerimientos de un cliente. Su utilización es común cuando se aplica marcos de entornos ágiles como Scrum. Al redactar las historias de usuario se debe tener en cuenta describir el Rol, la funcionalidad y el resultado esperado en una frase corta. Debe venir acompañada (al reverso) de los criterios de aceptación, hasta un máximo de 4 por historia, redactado también en una frase que indique el contexto, el evento y el comportamiento esperado ante ese evento.

**HU01 – Registrar Mascota**

**Como:** Usuario

**Puedo:** Registrar los datos de la mascota al sistema.

**Debido a:** Qué para dispensar el alimento es necesario el registro de la mascota.

**Comprobación:**

Se realiza el registro de la mascota correctamente.

**ROI 10/5=2**

10

5

**HU02 – Modificar datos Mascota**

**Como:** Usuario

**Puedo:** Modificar los datos de la mascota al sistema.

**Debido a:** Si hay un error o ya es necesario el cambio de datos.

**Comprobación:**

Se realiza el cambio de los datos de la mascota correctamente.

**ROI 10/5=2**

10

5

**HU03 – Eliminar Mascota**

**Como:** Usuario

**Puedo:** Eliminar a la mascota

**Debido a:** Si ya no tienen a la mascota a su cuidado.

**Comprobación:**

Se eliminan los datos de la mascota correctamente.

**ROI 10/5=2**

10

5

**HU04 – Mostar Mascota**

**Como:** Usuario

**Puedo:** Mostrar los datos de la mascota

**Debido a:** Si necesito ver los datos o para actualizar es necesario mostrarlos.

**Comprobación:**

Se mostraran los datos de la mascota correctamente.

**ROI 10/5=2**

10

5

**HU05 – Envía datos al dispositivo.**

**Como:** Sistema

**Puedo:** Enviar los datos guardados al dispositivo de dispensar.

**Debido a:** Es necesario los datos para saber en qué tiempo y cuanto se va dispensar el alimento.

**Comprobación:**

Se envía y recibe los datos correctamente.

**ROI 10/4=2.5**

10

4

**HU06 –Envía notificación para actualizar datos.**

**Como:** Arduino

**Puedo:** Enviar notificación mediante SMS al dueño.

**Debido a:** Que en cierto tiempo la mascota crecerá y es necesario actualizar datos.

**Comprobación:**

Se envía y recibe el SMS correctamente.

**ROI 9/3=3**

9

3

**HU07 – Envía notificación de aviso cuando haya poca cantidad de alimento**

**Como:** Arduino

**Puedo:** Notificar al dueño con un mensaje de texto SMS que la cantidad de comida en el dispensador está por terminarse.

**Debido a:** Para dispensar la comida de la mascota es necesario contar con alimento suficiente.

**Comprobación:**

Se enviara un mensaje de aviso cuando quede poco alimento.

**ROI 8/3=2.6**

8

7

10

3

**HU08 – Dispensar la cantidad de alimento a soltar.**

**Como:** Arduino

**Puedo:** Soltar la cantidad de comida que la mascota necesita.

**Debido a:** El dueño configura y coloca los datos de la mascota para que el sistema la proporcione de forma automática.

**Comprobación:**

Dispensar la comida a la mascota dependiendo de las características del ejemplar.

**ROI 10/7=1.4**

3

3

**HU09 –** **Llenar el plato de comida de la mascota**

**Como:** Arduino

**Puedo:** Proporcionar la comida de la mascota en su plato o recipiente correspondiente.

**Debido a:** Que es necesario dispensar la comida de la mascota en una zona, se realiza en su plato personal.

**Comprobación:**

Llenar el plato de comida de la mascota con su comida adecuada.

**ROI 10/7=1.4**

10

7

**HU10 –** **Abrir puerta del alimento.**

**Como:** Arduino

**Puedo:** Abrir la compuesta que permite la salida de comida del contenedor al plato de la mascota.

**Debido a:** Que existirá una compuerta que bloquee el acceso de comida y solo se accione cuando se requiera proporcionar comida.

**Comprobación:**

Se abrirá la compuerta del recipiente para dispensar la comida de la mascota.

**ROI 10/7=1.4**

10

7

10

**HU11 –** **Cerrar puerta del alimento.**

**Como:** Arduino

**Puedo:** Cerrar la compuerta para impedir que se siga dispensando más comida al plato de la mascota.

**Debido a:** Debe proporcionar únicamente la comida suficiente que requiera el animal.

**Comprobación:**

Se cerrara la compuerta cuando el alimento dispensado sea el adecuado para el ejemplar.

**ROI 10/7=1.4**

7

**HU12 – Detectar insuficiencia de alimento del dispensador.**

**Como:** Arduino

**Puedo:** Enviar una alerta que notifique que el alimento está por acabarse

**Debido a:** Cada cierto tiempo la comida se va disminuyendo del dispensador.

**Comprobación:**

Se envía y recibe la notificación correctamente.

**ROI 10/5=2**

10

5

**HU13 – Diseño del prototipo.**

**Como: Diseñador**

**Puedo:** Crear el diseño del prototipo del dispensador.

**Debido a:** Para crear el dispensador se necesita un plano de cómo construirlo.

**Comprobación:**

Se muestra el diseño del dispensador.

**ROI 10/3=3.3**

10

3

**HU14 – Diseño de la base de datos.**

**Como:** Analista

**Puedo:** Crear la base de datos para el sistema.

**Debido a:** Se necesita guardar la información de la mascota.

**Comprobación:**

Se puede guardar la información.

**ROI 10/6=1.6**

10

6

**HU15 –Diagrama entidad-relación.**

**Como:** Analista

**Puedo:** Crear diagrama de la base de datos

**Debido a:** Crear la base de datos se necesita el diagrama.

**Comprobación:**

Se muestra el diagrama entidad-relación.

**ROI 10/2=5**

10

2

**HU16 –Storyboards de la aplicación de escritorio.**

**Como:** Analista

**Puedo:** Usar los storyboards para crear la aplicación del sistema.

**Debido a:** necesitar una aplicación para controlar el dispensador.

**Comprobación:**

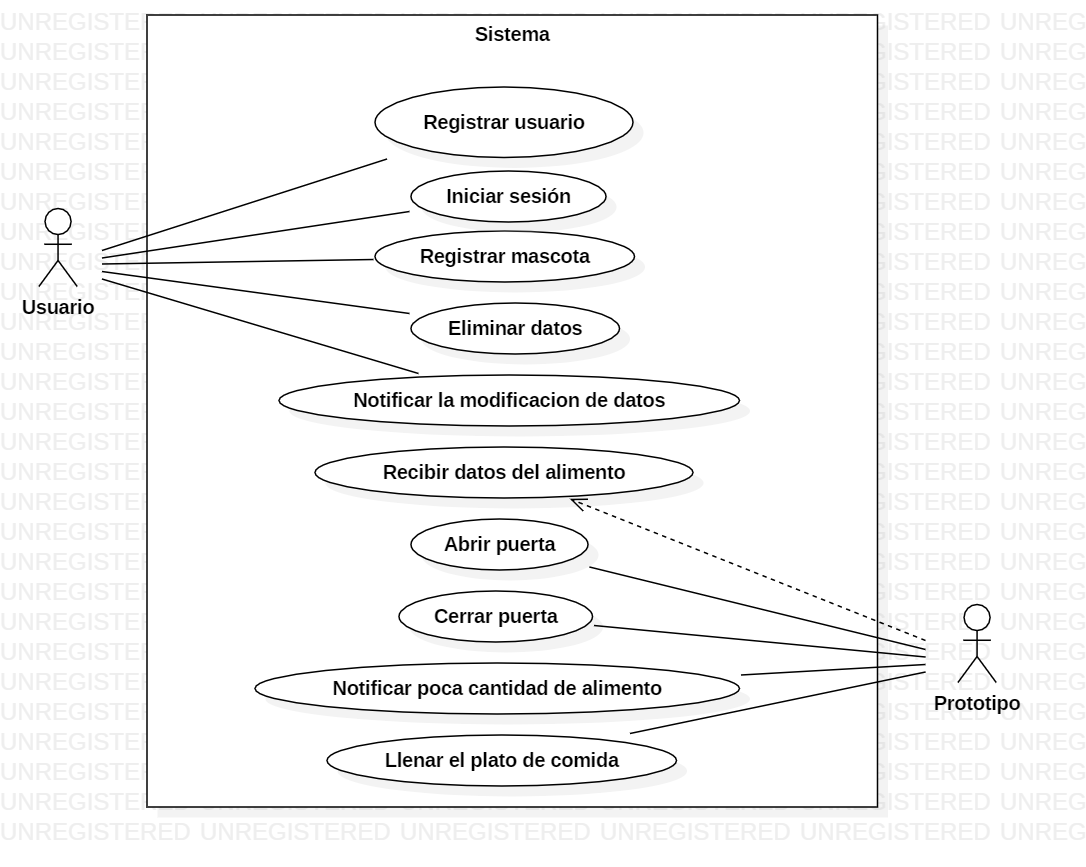
Se usará la aplicación para el manejo del sistema.

**ROI 10/2=5**

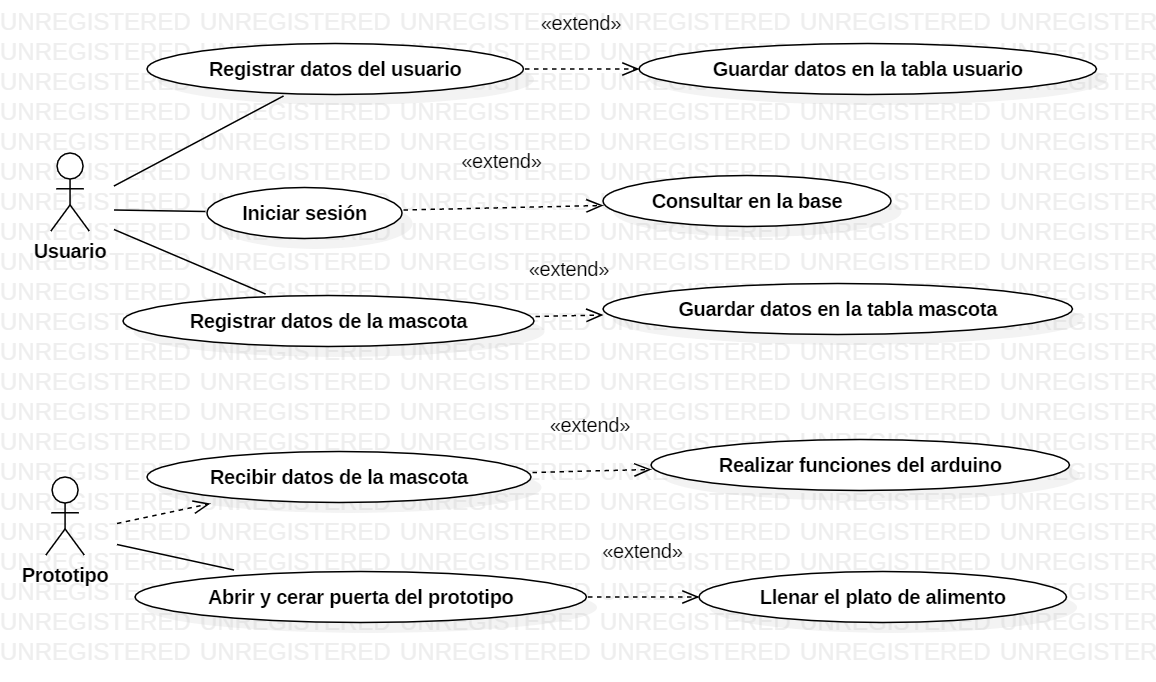
10

2

También se muestran los diagramas de casos de usos con los actores que intervienen en petsitting. (Ver figura 2.1 y 2.2)



**Figura 2.1** Diagrama de casos de uso de sistema



**Figura 2.2** Diagrama de casos de uso extendido del sistema

# Definición de los requerimientos

La definición de requerimientos es el **conjunto de técnicas y procedimientos que nos permiten conocer los elementos necesarios para definir un proyecto de software.** Es una tarea de ingeniería del software que permite especificar las características operacionales del software, indicar la interfaz del software con otros elementos del sistema y establecer las restricciones que debe cumplir el software.

**Clasificación de los requerimientos:**

* Requerimientos funcionales: qué debe hacer el sistema o software.
* Requerimientos no funcionales: cómo debe funcionar el sistema o software (no su implementación), por ej. calidad, rendimiento, facilidad de uso, etc.

**Características que deberían cumplir los requerimientos:**

* Actual: el requerimiento no debe volverse obsoleto con el paso del tiempo.
* Cohesión: el requerimiento debe dirigirse a solo una única cosa.
* Completo: el requerimiento debe estar completamente declarado en un único lugar, sin información faltante.
* Consistente: el requerimiento no debe contradecir ningún otro requerimiento y debe ser completamente consistente con toda la documentación.
* Correcto/necesario: el requerimiento debe cumplir con la necesidad declarada por los interesados en el sistema/software.
* No ambiguo: el requerimiento debe estar concisamente declarado. Debe expresar hechos objetivos, no opiniones subjetivas. Debe poder ser interpretado de una única manera.
* Verificable/demostrable: la implementación del requerimiento debe poder ser resuelta en alguno de estos cuatro métodos: inspección, análisis, demostración o prueba.

## Requerimientos funcionales

|  |  |
| --- | --- |
| # | Requerimientos funcionales |
| 1 | Registrar mascota. |
| 2 | Modificar datos de la mascota. |
| 3 | Eliminar datos de la mascota. |
| 4 | Realizar el cálculo correspondiente de comida a dispensar. |
| 5 | Mostrar datos de la mascota. |
| 6 | Envía datos al dispositivo. |
| 7 | Envía notificación para actualizar datos de la mascota. |
| 8 | Envía notificación de aviso cuando haya poca cantidad de alimento. |
| 9 | Dispensar la cantidad de alimento a soltar. |
| 10 | Llenar el plato de comida de la mascota. |
| 11 | Abrir puerta del alimento. |
| 12 | Cerrar puerta del alimento. |
| 13 | Detectar insuficiencia de alimento del dispensador. |

Los requerimientos funcionales son declaraciones de los servicios que proveerá el sistema, de la manera en que éste reaccionará a entradas particulares. En algunos casos, los requerimientos funcionales de los sistemas también declaran explícitamente lo que el sistema no debe hacer.(Ver tabla 3.1)

**Tabla 3.1** Lista de requerimientos funcionales del sistema

## Requerimientos no funcionales

Los requerimientos no funcionales representan características generales y restricciones de la aplicación o sistema que se esté desarrollando.

Suelen presentar dificultades en su definición dado que su conformidad o no conformidad podría ser sujeto de libre interpretación, por lo cual es recomendable acompañar su definición con criterios de aceptación que se puedan medir. (Ver tabla 3.2)

|  |  |
| --- | --- |
| # | Requerimientos no funcionales |
| 1 | Realizar la aplicación de control compatible con la plataforma de Windows. |
| 2 | El lenguaje de la aplicación de escritorio debe ser Java y se debe realizar conexión con C++ para el uso del dispensador. |
| 3 | El cálculo de comida y tiempo necesario se debe calcular en menos de 3 segundos. |
| 4 | La información y datos del usuario se protegerán con un sistema de encriptación en la base de datos. |
| 5 | El sistema se centenera en una carpeta comprimida para facilitar la instalación. |

**Tabla 3.2** Lista de requerimientos no funcionales del sistema

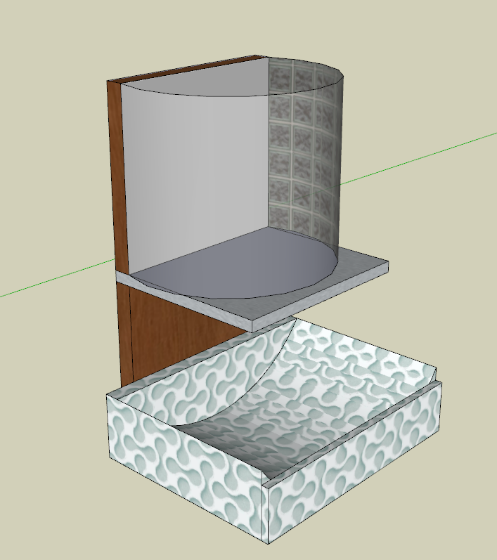
# Modelos del sistema

Un sistema es una colección de entidades que actúan y se relacionan hacia un fin lógico o gráfico. Mientras que un modelo es una representación simplificada de un sistema elaborado para comprender, predecir y controlar el comportamiento o vista de dicho sistema.

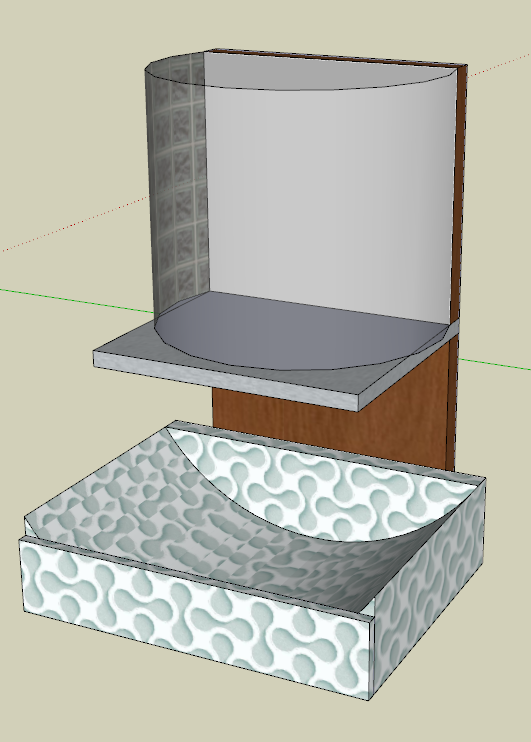
La importancia de realizar este tipo de entidades en la metodología scrum es:

* Nos permite un desarrollo ágil.
* Alta calidad del software
* Colaboradores satisfechos

A continuación se muestran los primeros “prototipos” del sistema (figura 4.1 y figura 4.1)



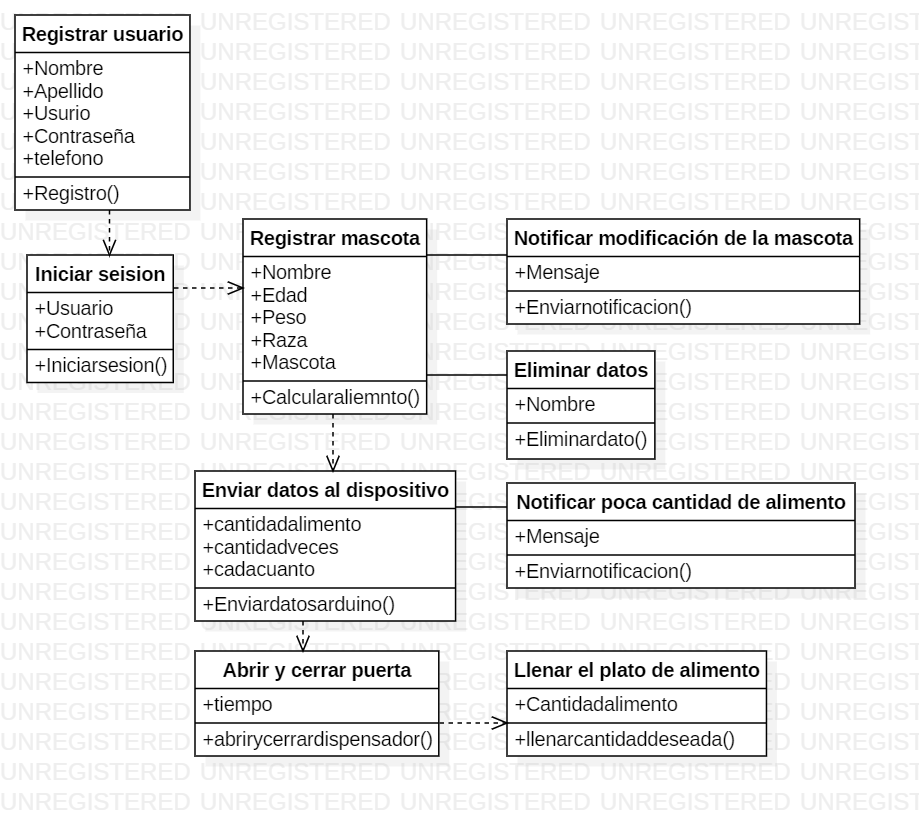
**Figura 4.1** Prototipo P001 del sistema



**Figura 4.2** Prototipo P002 del sistema

## Estructura de sistema

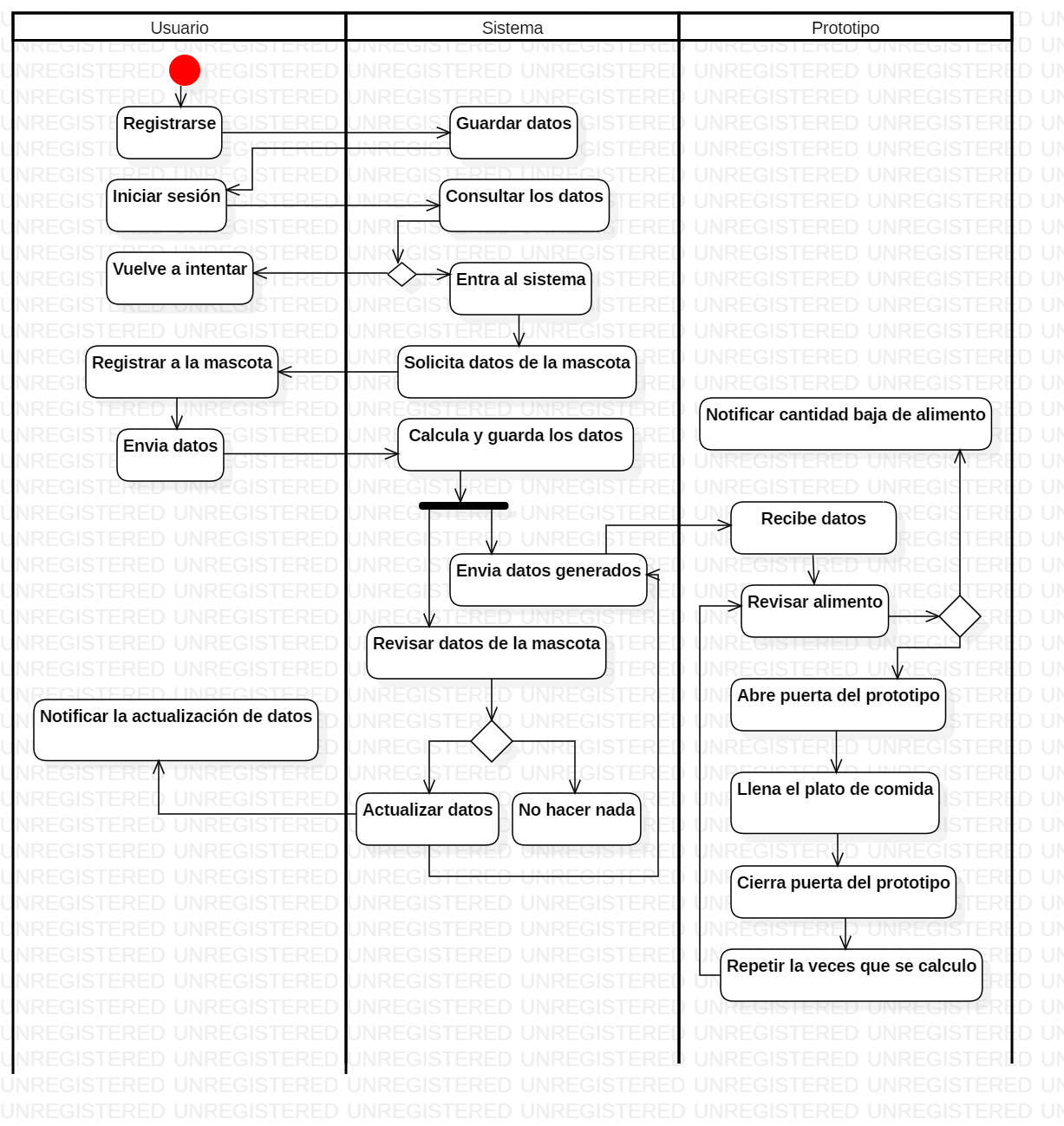
La estructura del sistema es todo el contenido de las características de las funciones necesarias para el desarrollo del sistema como se muestra en la figura 4.3



**Figura 4.3** Diagrama de control del sistema

## Comportamiento del sistema

El comportamiento del sistema conlleva los pasos que se debe seguir la funcionalidad del proyecto cuando este se implementa. (Ver figura 4.4)



**Figura 4.4** Diagrama de secuencia del sistema

# Propuesta de desarrollo

En este proyecto se desarrolla un dispensador de alimento automático para ciertas las mascotas con el fin que el dueño (usuario) pueda dejar a su mascota solo en casa cuando sea necesario o lo requiera.

Para el desarrollo del proyecto en cuestión del sistema se necesitar usar los siguientes lenguajes de programación:

* Java
* C++

Java nos permite crear el sistema del proyecto donde contendrá las funciones que conlleva con el dispensador de alimento para la mascota y el siguiente lenguaje C++ este nos ayuda en las funciones de los mecanismos o funciones del dispensador de alimento. También se utiliza una base de datos para almacenar toda la información que se requiera de la mascota y además del usuario está base de datos se crea con MySQL.

Las herramientas que se utilizan son los siguientes programas son:

* NetBeans
* Wampserver
* Star UML
* Gant Project
* REM

La herramienta NetBeans se utiliza para el manejo del lenguaje Java, el programa Wampserver se utiliza para la creación de la base de datos su motor es MySQL y los últimos 3 programas son para la creación de diagramas, casos de uso, etc. Para la documentación del proyecto.

Para la elaboración de este proyecto se estableció la metodología Scrum para la elaboración dicho proyecto la cual es mas apropiada ya que es flexible, ágil, iterativa y eficaz.

# Plan de pruebas

Es el proceso de prueba básico que comprende la planificación y el control de pruebas, análisis y diseño de pruebas, implementación y ejecución de pruebas, evaluación de criterios de salida y elaboración de informes, y actividades de cierre de pruebas.

Caso de prueba:

* Entrada + acciones + salida
* Salida obtenida== Salida esperada
* Salida obtenida!=Salida esperada

**¿Qué es un plan de pruebas?**

Es un producto formal que define los objetivos de la prueba de un sistema, establece y coordina una estrategia de trabajo, y provee del marco adecuado para elaborar una planificación paso a paso de las actividades de prueba.

**Clasificación**

**Fases de proceso de prueba**

* Planificación de las pruebas
* Diseño de las pruebas
* Implementación de las pruebas
* Evaluación de las pruebas

**Fases de plan de pruebas**

* Pruebas unitarias
* Pruebas de integración
* Pruebas del sistema
* Pruebas de implementación
* Pruebas de aceptación

## Planeación

Se describen casos de pruebas junto con su descripción y validación correspondiente para posteriormente ser evaluados en la matriz de pruebas.(Ver tablas 6.1, 6.2, 6.3, 6.4, 6.5, 6.6, 6.7, 6.8)

|  |  |
| --- | --- |
| ID: CP001 | Caso de prueba “Registrar mascota” |
| Objetivo | Verificar que el registro de la mascota haga la inserción correcta con datos válidos. |
| Descripción | Se realizará la evaluación por cada una de las casillas donde se tomara en cuenta que los datos tengan un formato correcto y sus debidas validaciones al dejar casillas en blanco, etc. |
| Recursos necesarios | Tester, equipo de cómputo, bitácora de evaluación. |
| Datos de prueba | Registro exitoso en base de datos y mensaje en pantalla. |

**Tabla 6.1** Caso de prueba 001

|  |  |
| --- | --- |
| ID: CP002 | Caso de prueba “Consultar mascota” |
| Objetivo | Mostrar en pantalla los datos de la mascota a buscar siempre y cuando corresponda al usuario ingresado. |
| Descripción | Se realizará la validación para mostrar solo la mascota que el dueño tenga registrada. |
| Recursos necesarios | Tester, equipo de cómputo, bitácora de evaluación y registro previo de algún ejemplar. |
| Datos de prueba | Consulta exitosa y mostrar datos en pantalla. |

**Tabla 6.2** Caso de prueba 002

|  |  |
| --- | --- |
| ID: CP003 | Caso de prueba “Eliminar mascota” |
| Objetivo | Permitir eliminar la mascota dentro del sistema. |
| Descripción | Se realizará la eliminación de la mascota en caso de que el dueño ya no requiera utilizar el dispensador o sufra la pérdida de su ejemplar y requiera ingresar una nueva mascota. |
| Recursos necesarios | Tester, equipo de cómputo, bitácora de evaluación y mascota registrada. |
| Datos de prueba | Registro eliminado tanto del sistema como la base mostrando mensaje de exitoso. Búsqueda de la mascota eliminada errónea. |

**Tabla 6.3** Caso de prueba 003

|  |  |
| --- | --- |
| ID: CP004 | Caso de prueba “Modificar mascota” |
| Objetivo | Verificar que la modificación de la mascota sea correcta. |
| Descripción | Se realizará la modificación de la mascota cuando pase una cierta cantidad de tiempo y requiera un cambio en la dispensación de su alimento. |
| Recursos necesarios | Tester, equipo de cómputo, bitácora de evaluación y mascota registrada. |
| Datos de prueba | Actualización correcta, actualizar los datos de forma correcta y mandar los nuevos valores al dispositivo de control. |

**Tabla 6.4** Caso de prueba 004

|  |  |
| --- | --- |
| ID: CP005 | Caso de prueba “Envía notificación de aviso cuando haya poca cantidad de alimento.” |
| Objetivo | Enviar mensaje SMS al número del dueño. |
| Descripción | Se realizará el envió de un mensaje de texto sms cuando el sistema detecte que la cantidad de alimento se esta agotando. |
| Recursos necesarios | Tester, equipo de cómputo, bitácora de evaluación. |
| Datos de prueba | Mensaje recibido a un numero celular que el dueño previamente registro al inicio. |

**Tabla 6.5** Caso de prueba 005

|  |  |
| --- | --- |
| ID: CP006 | Caso de prueba “Realizar el cálculo correspondiente de comida a dispensar.” |
| Objetivo | Calcular la cantidad de alimento y el tiempo a dispensar de cada ejemplar. |
| Descripción | Se realizará el cálculo de la cantidad de comida que requiere cada ejemplar, basándose en la investigación previa que se realizo para mantener una buena salud para la mascota. |
| Recursos necesarios | Tester, equipo de cómputo, bitácora de evaluación. |
| Datos de prueba | Una ves realizado el registro el sistema mostrara los datos de control correspondientes (cantidad y tiempo de dispensación) y validara que el dispositivo este conectado al equipo de cómputo. |

**Tabla 6.6** Caso de prueba 006

|  |  |
| --- | --- |
| ID: CP007 | Caso de prueba “Envía notificación para actualizar datos de la mascota.” |
| Objetivo | Enviar mensaje SMS al dueño cuando sea necesario una actualización del ejemplar. |
| Descripción | Se realizará el envío de un mensaje de texto SMS al dueño, cuando el sistema detecte un crecimiento de tiempo del ejemplar y requiere actualizar los datos para una mayor o menor dispensación de comida. |
| Recursos necesarios | Tester, equipo de cómputo, bitácora de evaluación y mascota en crecimiento. |
| Datos de prueba | Envió de mensaje SMS al dueño y reseteo del sistema de dispensación con los nuevos datos correspondientes. |

**Tabla 6.7** Caso de prueba 007

|  |  |
| --- | --- |
| ID: CP008 | Caso de prueba “Envía notificación para actualizar datos de la mascota.” |
| Objetivo | Enviar mensaje SMS al dueño cuando sea necesario una actualización del ejemplar. |
| Descripción | Se realizará el envío de un mensaje de texto SMS al dueño, cuando el sistema detecte un crecimiento de tiempo del ejemplar y requiere actualizar los datos para una mayor o menor dispensación de comida. |
| Recursos necesarios | Tester, equipo de cómputo, bitácora de evaluación y mascota en crecimiento. |
| Datos de prueba | Envió de mensaje SMS al dueño y reseteo del sistema de dispensación con los nuevos datos correspondientes. |

**Tabla 6.8** Caso de prueba 008

## Planificación

Se planifican las pruebas a realizar a cada proceso como se muestra en la tabla 6.9.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Matriz de pruebas. | | | | |
| Código CP | Fecha | Tiempo (horas) | Encargado | Resultado |
| ID: CP001 | 2-Sep-2019 | 1 | Kary Larios | (En espera) |
| ID: CP002 | 2-Sep-2019 | 1 | Kary Larios | (En espera) |
| ID: CP003 | 3-Sep-2019 | 1 | Kary Larios | (En espera) |
| ID: CP004 | 3-Sep-2019 | 1 | Kary Larios | (En espera) |
| ID: CP005 | 5-Sep-2019 | 2 | Kary Larios | (En espera) |
| ID: CP006 | 9-Sep-2019 | 2 | Antonio Alonso | (En espera) |
| ID: CP007 | 9-Sep-2019 | 3 | Antonio Alonso | (En espera) |
| ID: CP008 | 9-Sep-2019 | 3 | Antonio Alonso | (En espera) |

**Tabla 6.9** Plan de pruebas detallado

# Prototipo del sistema

Un prototipo es un modelo (representación, demostración o simulación) fácilmente ampliable y modificable de un sistema planificado, probablemente incluyendo su interfaz y su funcionalidad de entradas y salidas.

Existen varios tipos de prototipos, cada uno de los cuales permite la realización de un tipo determinado de pruebas y con un determinado nivel de realismo. En ingeniería de requisitos, los prototipos más comunes son los siguientes:

* Mock-ups. Se trata de pantallas, típicamente dibujadas a mano en papel, que representan un aspecto concreto del sistema. El soporte que proporcionan a la validación es muy limitado, con la excepción, quizás, de aclarar el interfaz gráfico deseado en casos complejos.
* Storyboards. Son una evolución de los mock-ups, ya que además del interfaz, se muestra la secuencia de acciones, o escenarios, que se deben realizar con el programa. Por ejemplo es habitual que, antes de cerrar un programa, se pregunte si se desea cerrar el fichero de trabajo.
* Maquetas. Una maqueta es una versión simplificada del sistema software deseado. Típicamente, una maqueta representa únicamente el interfaz del sistema y, opcionalmente, las conexiones entre pantallas mediante la utilización de elementos activos como los botones. Si fuera necesaria mayor fidelidad, podrían codificarse partes del sistema, de tal modo que además, del interfaz, el software pudiera ofrecer algunos resultados reales. Ello es lo que se conoce como “prototipo funcional”.

Otros tipos de prototipos:

* Baja Fidelidad: conjunto de dibujos (por ejemplo, una presentación de escenarios) que constituye una maqueta estática, no computarizada no operativa de una interfaz de usuario para un sistema en planificación.
* Alta Fidelidad: conjunto de pantallas que proporcionan un modelo dinámico, computarizado y operativo de un sistema en planificación.
* Exploratorio: prototipo no reutilizable utilizado para clarificar las metas del proyecto, identificar requerimientos, examinar alternativas de diseño o investigar un sistema extenso y complejo.
* Experimental: prototipo utilizado para la validación de especificaciones de sistema
* Operacional: prototipo iterativo que es progresivamente refinado hasta que se convierte en el sistema final.
* Horizontal: prototipo que modela muchas características de un sistema pero con poco detalle. Dicho detalle alcanzará una profundidad determinada, va a resultar especialmente útil en las etapas tempranas de diseño y tiene como objetivo el test del modo de interacción global, al contemplar funciones comunes que el usuario va a utilizar frecuentemente.
* Vertical: prototipo que modela pocas características de un sistema pero con mucho detalle. Va a resultar especialmente útil en etapas más avanzadas del diseño y tiene como objetivo el test de detalles del diseño
* Diagonal: prototipo horizontal hasta un cierto nivel, a partir del cual se puede considerar
* Global: prototipo del sistema completo. Prototipo horizontal expandido que modela una gran cantidad de características y cubre un amplio rango de funcionalidades. Va a resultar muy útil a lo largo de todo el proceso de diseño.
* Local: prototipo de un único componente o característica del sistema de usabilidad crítica. Va a resultar de utilidad en algunas etapas específicas del proceso de diseño.

**¿Que se logra con su desarrollo?**

Modificación del Sistema en Etapas tempranas de su desarrollo, el éxito del uso del prototipo depende de qué tan pronto y con qué frecuencia se reciba la retroalimentación del usuario para hacer cambios y adecuarlos a las necesidades actuales.

## Diseño inicial

## Funcionalidad

Se muestra en computadora la funcionalidad de 3 requerimientos funcionales.

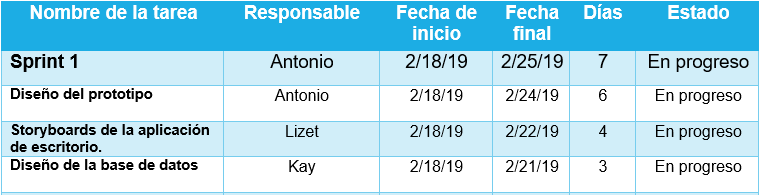
# Anexo I

# Anexo II

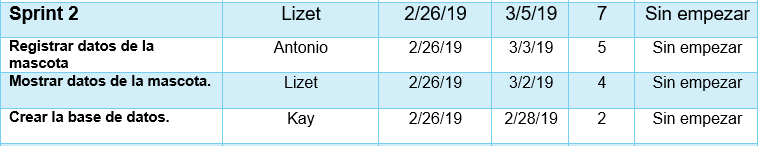
# Anexo III

En este apartado se muestra la planeación del sprint, con sus respectivas observaciones cada uno como se muestra a continuación:

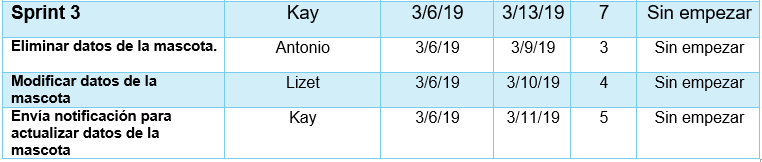
En el primer sprint las tareas se cumplieron en tiempo y forma, sin tener retardos, cumpliendo la planeación asignada.



En el segundo sprint de igual forma las tareas se cumplieron en tiempo y forma, solo se cuentan con algunas observaciones que se realizaron por parte del cliente.



En el tercer sprint las tareas se cumplieron en tiempo y forma, solo se cuentan con algunas observaciones que se realizaron por parte del cliente.



# Glosario

Ambiguo: Que puede entenderse o interpretarse de diversas maneras.

Arduino: Es una plataforma de desarrollo basada en una placa electrónica de hardware libre que incorpora un micro controlador re-programable.

C++: Es un lenguaje de programación

Cohesión: Unión o relación estrecha entre personas o cosas.

Diagrama de caso de uso: Modelan la funcionalidad del sistema usando actores y casos de uso.

Diagrama secuencial: Es un tipo de **diagrama** usado para modelar interacción entre objetos en un sistema.

Entidades: Asociación de personas de cualquier tipo, en especial la que se dedica a una actividad laboral.

Java: Es un lenguaje de programación

Mock-ups: Es un diseño digital de una web y/o aplicación.

MySQL: Es un sistema de gestión de bases de datos relacional desarrollado bajo licencia dual: Licencia pública general/Licencia comercial por Oracle.

Requerimientos funcionales: Define una función del sistema de software o sus componentes.

Requerimientos no funcionales: Son requisitos que imponen restricciones en el diseño o la implementación.

UML: Es un lenguaje estándar para escribir planos de software

Scrum: Es una de las metodologías ágiles más conocidas en la actualidad

Storyboards: Es un conjunto de ilustraciones mostradas en secuencia con el objetivo de servir de guía para entender una historia.