Sistema de información web en donde se centralizará los datos de las mascotas perdidas, encontradas o recuperadas en el departamento de Cundinamarca

LostPet

Angie Alejandra González

Código: 10892119341

Oscar Daniel Marín Moran

Código: 10892112490

Universidad Antonio Nariño

Programa Especialización en Ingeniería de Software

Facultad de Ingeniería de Sistemas

Bogotá, Colombia

2021

LOSTPET

Sistema de información web en donde se centralizará los datos de las mascotas perdidas, encontradas o recuperadas en el departamento de Cundinamarca

Oscar Daniel Marín Moran

Angie Alejandra González

Proyecto de grado presentado como requisito parcial para optar al título de:

Especialista en Ingeniería de Software

Msc, Dianalin Neme Prada

Universidad Antonio Nariño

Programa Especialización en Ingeniería de Software

Facultad de Ingeniería de Sistemas

Bogotá, Colombia

2021

NOTA DE ACEPTACIÓN

El trabajo de grado titulado
_____, Cumple
con los requisitos para optar
Al título de _____.

_____

Firma del Tutor

_____

Firma Jurado

_____

Firma Jurado

Contenido

1. 11
2. 12
3. 13
4. 14
5. 15
6. 16
7. 17
8. 18
   - 8.1 19
   - 8.2 21
   - 8.3 22
   - 8.4 23
9. 24
10. 27
    - 10.1 29
      - Registrar Usuario 28
      - Recuperar contraseña 31
      - Iniciar Sesión 32
      - Modificar Perfil 34
      - Módulo Administración 35
        - Usuarios 35
        - Modificar Registro Usuario 36
        - Eliminar Registro Usuario 37
      - Razas 37
      - Auditorias 39
      - Aprobar publicaciones 40
      - Módulo de Mascotas 40
      - Notificaciones 46
      - Publicaciones 47
      - Reportar Información Mascota 48
      - Historias de Usuario (Ver Anexos): 50
    - 10.2 49
    - 10.3 50

8.3.1 Diagrama de Despliegue 53
8.3.2 Caso de Uso Arquitecturalmente relevante 54
8.3.3 Diagramas de Secuencia 55
8.3.4 Diagrama de Clases 58
8.3.5 Arquitectura de Alto Nivel 58
11. 58
Proceso de Software 61
12. 61
13. 99
14. 110
15. 111
16. 112

1. Resumen

En el departamento de Cundinamarca no existe actualmente un sitio donde se centralice la información de perros o gatos encontrados o perdidos, haciendo con esto más largo el proceso de búsqueda de la mascota.

Con el proyecto LostPET se pretende mejorar esta búsqueda teniendo en cuenta que una mascota pasa a ser parte de la familia, convirtiendo su pérdida en un momento difícil del que quizá, algunas personas sin escrúpulos se quieran aprovechar para sacar ventaja, motivo por el cual LostPET protege los datos personales de los usuarios.

2. Abstract

In the department of Cundinamarca there is currently no site where the information on found or lost dogs or cats is centralized, thus making the search process for the pet longer.

The LostPET project seeks to help improve this search taking into account that a pet becomes part of the family, turning its loss into a difficult moment that perhaps some unscrupulous people want to take advantage of to take advantage of, which is why LostPET protects the personal data of users.

3. Introducción

LostPET es un sistema de información desarrollado con el fin de centralizar la información de animales (perros y gatos) perdidos y encontrados en Cundinamarca dado que, actualmente, según un estudio publicado por el Tiempo tan solo en Bogotá se pierden diariamente de 10 a 12 mascotas y al no existir una aplicación actualizada que centralice esta información la búsqueda puede llegar a ser extensa.

Para facilitar la búsqueda de la mascota, se generará un PDF tipo cartel para colocar en diferentes puntos físicos o electrónicos. Este cartel incluye algunos datos de la mascota y un código QR que enviará a la persona a un formulario para indicar alguna novedad de la mascota, y, en ningún momento, se expondrán los datos personales de los dueños de las mascotas.

El proyecto fue desarrollado con PHP, BootStrap, CSS, y HTML, utilizando Scrumban como metodología y, diseñando historias de usuario como requerimientos.

A lo largo del documento se verá reflejado más a detalle los diferentes procedimientos realizados para el desarrollo del proyecto, así como sus diferentes desafíos a lo largo de su ejecución.

4. Título

LOSTPET: Sistema de información web en donde se centralizará los datos de las mascotas perdidas, encontradas o recuperadas en el departamento de Cundinamarca.

5. Formulación del Problema

Según un estudio publicado por el periódico El Tiempo, la desaparición de perros y gatos domésticos en el departamento de Cundinamarca ha crecido, en especial en Bogotá en donde se calcula que por día se puede extraviar de 10 a 12 mascotas (Perros o gatos). En las épocas de celebrar festividades no es la excepción, dado que ellos tienen el oído más desarrollado y en consecuencia se puede producir una sobreestimulación del sistema nervioso, generando ansiedad, angustia, miedo y en ocasiones, el instinto de supervivencia de ellos los lleva a buscar desesperadamente un lugar para protegerse. Este tipo de reacciones pueden ocasionar la desorientación del animal y su posible desaparición.

Actualmente no existe una aplicación en la cual, una persona pueda ingresar la información de su mascota sin divulgar sus datos personales y con ello, sin estar exponiéndose a llamadas fraudulentas ya que, aunque existan aplicaciones con un objetivo similar el funcionamiento es diferente y, en algunos casos la aplicación a pesar de estar expuesta al público no permite ingresar datos nuevos ya que está obsoleta o no se han realizado actualizaciones recientes.

Teniendo en cuenta lo anterior, ¿Cómo a través de un sistema de información se puede apoyar a la localización de animales perdidos protegiendo los datos personales de los usuarios?

6. Objetivo General

Implementar un sistema de información que permita el registro, consulta e ingreso de novedades de mascotas pérdidas o encontradas (Perros o gatos) en el departamento de Cundinamarca, gestionando de manera segura la información de los dueños de las mascotas evitando con ello la divulgación de sus datos personales.

7. Objetivos Específicos

- Centralizar la información de los animales perdidos (gatos y perros) a través de una interfaz gráfica que permita registrar, consultar y obtener la información.

- Diseñar una funcionalidad que permita a una persona brindar información sobre un animal reportado como perdido o encontrado a través del escaneo de un código QR.

- Informar a los dueños de mascotas cualquier eventualidad que suceda respecto a la desaparición de su animal de compañía mediante notificaciones vía correo electrónico,

- Construir un mecanismo que permita al dueño de una mascota perdida evitar llamadas fraudulentas protegiendo sus datos personales.

- Salvaguardar los datos proporcionados por los usuarios teniendo en cuenta la normatividad vigente para la protección de datos.

- Automatizar la ejecución de pruebas funcionales aplicadas a la regresión de un flujo exitoso.

8. Marco de Referencia

En el año 2018 se lanzó la primera aplicación móvil del gobierno Distrital de Bogotá llamada Distrito Appnimal desarrollada con el apoyo de MinTIC y Colciencias, que entre sus diferentes módulos se diseñó uno que permitía al usuario registrar un animal de su propiedad que se encuentre perdido con el fin de ubicarlo y recuperarlo de manera rápida; también permitiría registrar un animal encontrado sin dueño con el fin de localizar a su propietario.

Sin embargo, actualmente la aplicación, al ser una iniciativa del gobierno dejó de ser actualizada ya que no continuaron con la licitación para el mantenimiento de los diferentes módulos. La última actualización fue en el mes de Julio de 2018 por lo que muchas de sus funciones ya no sirven o están obsoletas.

Teniendo en cuenta los comentarios hechos por los usuarios de la aplicación, le hace falta filtros para buscar por características como color, tamaño, raza, entre otros, y la opción de dar de baja los anuncios cuando ya se ha encontrado el animalito perdido, además de una forma de contactar por medio de la App a quien publica el anuncio.

En Bogotá, según un estudio publicado en el 2017 por Minsalud, hay un total de 1'277.230 Perros y gatos, de los cuales son 998.986 perros y 278.244 gatos, mientras que en Cundinamarca hay un total de 359.524 siendo 288.144 perros y 71.380 gatos.

8.1 Estado del Arte

Nombre Aplicativo: Distrito Appnimal

Ventajas:

- App funcional durante su año de lanzamiento.
- Módulo en el cual se evidencias las leyes que protegen los animales.
- Software Libre como App móvil para descargas en Ios y Android

Desventajas:

- En el módulo de mascotas perdidas y encontradas no se pueden filtrar por color, raza, tamaño, características particulares, etc.
- No deja actualizar la información si se tiene una mascota perdida, borrar el reporte o generar alguna alerta.
- Constantes fallas de conexión o errores para subir la información de las mascotas encontradas, en adopción o pérdidas.
- No se puede contactar con quien tiene una mascota en adopción,
- perdida o encontrada, ya que no solicita datos de contacto.
- No indica la fecha en la que se realizó una publicación.
- Al momento de registrarse da la opción de hacerlo desde Facebook o Twitter, pero dichas funciones no están disponibles para la App.
- El Módulo de Zoolidario no funciona ya que se debe registrar en la App y no deja registrarse ya que aparecen errores de conexión.
- Falta de actualizaciones sobre la App para corregir errores (13 de Julio 2018).

Análisis:

Actualmente no existe una aplicación funcional en la cual se centre la información de animales perdidos. La aplicación al ser una iniciativa del gobierno dejó de ser actualizada ya que no continuaron con la licitación.

La última actualización fue en el mes de Julio de 2018 por lo que muchas funciones ya no sirven o están obsoletas.

Nombre Aplicativo: Animalitos perdidos en Colombia.

Ventajas:

- Interfaz de búsqueda con filtros variados para hacer más amigable la consulta.
- Cuenta con diversas difusiones a nivel de medios sociales, ya que cuenta página de Facebook, Twitter y un Blog.
- Se pueden reportar desde animales perdidos, robados o encontrados.
- Se puede ver un histórico de años pasados de animales perdidos, encontrados o robados.
- Diversos comunicados sobre temas referentes a mascotas.

Desventajas:

- Algunas funciones de la página web arrojan error (Ver google maps Animalitos perdidos en Bogotá).
- No tiene búsquedas avanzadas de animales perdidos
- A la fecha se encuentra caída la página.

Análisis:

Aunque presenta una interfaz amigable para la búsqueda, no cuenta con notificaciones para avisar de algún evento. Se estima ampliar las búsquedas y utilizar notificaciones por medio de correos electrónicos y QR.

Actualizar permanentemente el sitio web con información útil para los usuarios y sus mascotas.

Teniendo en cuenta los resultados obtenidos en el análisis realizado al estado del arte se concluye que aunque las aplicaciones ofrecen características funcionales a los usuarios, la falta de actualización de los aplicativos hace que esta se vuelva obsoleta y no brinde opciones nuevas e innovadoras.

El aplicativo LostPet ofrecerá actualizaciones continuas a los usuarios y un punto de innovación el cual permitirá utilizar el código QR para reportar información de las mascotas pérdidas o encontradas así como notificaciones en línea vía correo electrónico.

Con el fin de mantener continuamente la aplicación, se realizan actualizaciones trimestralmente adaptándose a las necesidades del mercado.

8.2 Impacto

Teniendo en cuenta las necesidades expuestas en el planteamiento del problema, se espera que el impacto social del proyecto sea positivo. Se abordará una necesidad actual de la comunidad, la cual permitirá a las personas de la región de Cundinamarca facilitar la búsqueda de las mascotas extraviadas haciendo uso de la tecnología.

De igual manera se pretende disminuir el tiempo de búsqueda y recuperación de la mascota ya que actualmente esta búsqueda dura meses y hasta años sin encontrar una sola pista del animal perdido. Se espera que esta búsqueda disminuya a máximo un mes o que por lo menos se tenga una pista de la mascota en este rango de tiempo y hacer posible que el dueño se vuelva a encontrar con su animal de compañía.

8.3 Componentes de Innovación

Al momento de registrar un animal perdido, el sistema insertará la información en una base de datos y generará un archivo en formato pdf que se podrá imprimir y visualizar los datos básicos de la mascota. En este documento tipo cartel, se incluirá un código QR el cual almacenará una url que re-direccionará al usuario al sitio web diseñado, se cargará la información previamente registrada con los datos básicos de la mascota para que se pueda ingresar cualquier pista o información del animal perdido. Cada vez que se registre algún tipo de información, el sistema enviará una notificación vía correo electrónico al propietario de la mascota en donde se le mostrará los datos enviados por algún usuario.

En ningún momento se expondrán los datos del dueño de la mascota, sino que, al ser notificaciones vía correo electrónico, se protegerán los mismos; la persona que da la información de la mascota podrá dar sus datos para que el dueño se contacte.

8.4 Marco Teórico

Una mascota es un animal de compañía que pasa a ser parte de la familia, y no es para más, ya que estos animales suelen dar amor incondicional y en muchas ocasiones, ayudan a salir de crisis emocionales.

Miguel de la Torre, director de nuevos negocios de Kantar Worldpanel indicó que, según un estudio realizado por ellos, “En Colombia existen más de 3’500.000 hogares con mascotas, de los cuales el 67 % tienen perro, el 18 % gato y 16 % tienen ambos”, es por ello que al desaparecer deja un gran vacío para los integrantes de la familia.

En la actualidad la tecnología ha tomado una gran influencia en la vida cotidiana y en la forma de comunicarse, es por ello que el avance tecnológico brinda mecanismos para facilitar los quehaceres diarios, cuando se dan casos de pérdidas en las mascotas, es normal visualizar carteles pegados en diferentes postes, o, publicaciones por diferentes redes sociales, sin embargo, no se ha podido centralizar esta información en una misma página.

Dado los beneficios que trae la tecnología consigo, se creará una página web utilizando el lenguaje de programación PHP junto con CSS, Javascript, Bootstrap y Html5, la cual permitirá diseñar un aplicativo que abarque los objetivos pactados para el proyecto.

Se escogió PHP como lenguaje de programación para el desarrollo de la solución ya que además de ser un software libre, corre en cualquier plataforma y permite estructurar de una manera dinámica y fácil la construcción de los módulos de la aplicación. Además se integra muy bien con HTML la cual permite maquetar la parte gráfica del sistema y junto con Css y Bootstrap permite presentar una interfaz llamativa para el usuario.

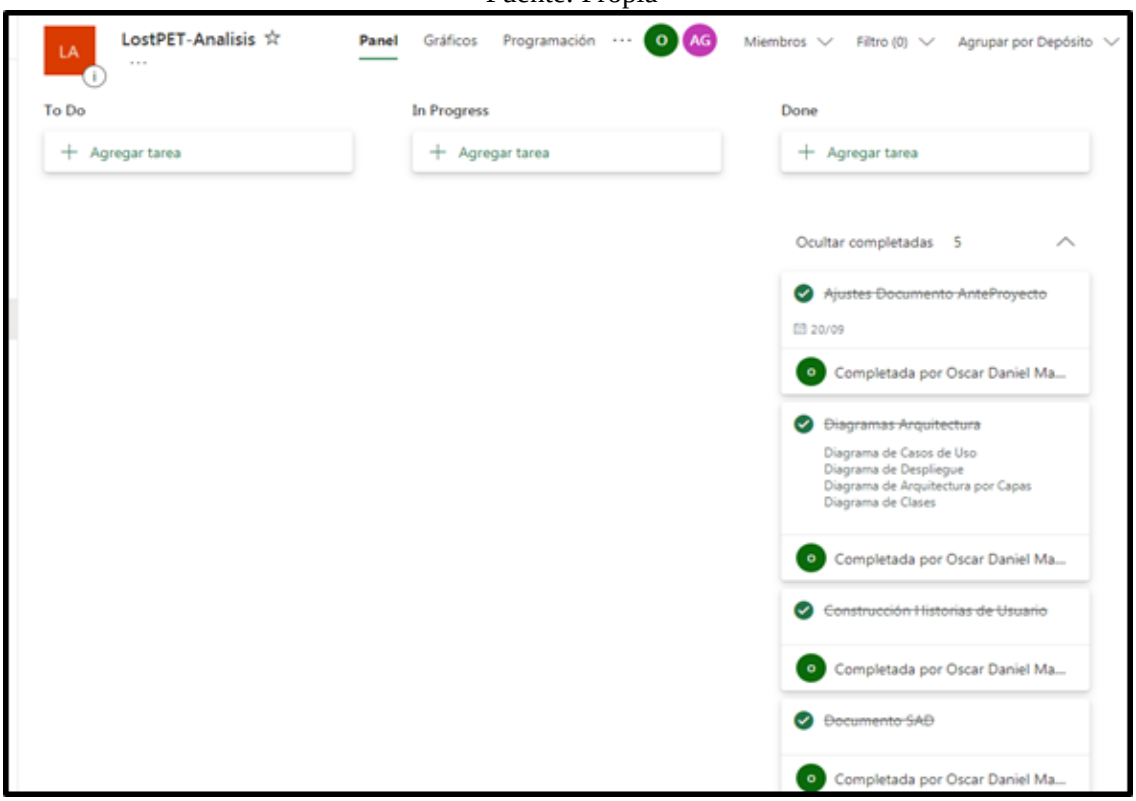
9. Metodología

Para realizar la solución a la problemática planteada, la metodología utilizada para el desarrollo del proyecto es la combinación de Scrum y Kanban, más conocida como Scrumban la cual combina los principios de los métodos ágiles de gestión de proyectos. Esto permitirá conocer el estado real de la ejecución del proyecto, realizar un mayor análisis y seguimiento a las tareas realizadas y mejorar la interacción entre los miembros del grupo cumpliendo con entregas pequeñas pero significativas.

Para la programación de las tareas, se utilizó el tablero Planner proporcionado por la herramienta Office de Microsoft para llevar el seguimiento de las actividades. Estas se dividieron en las siguientes etapas:

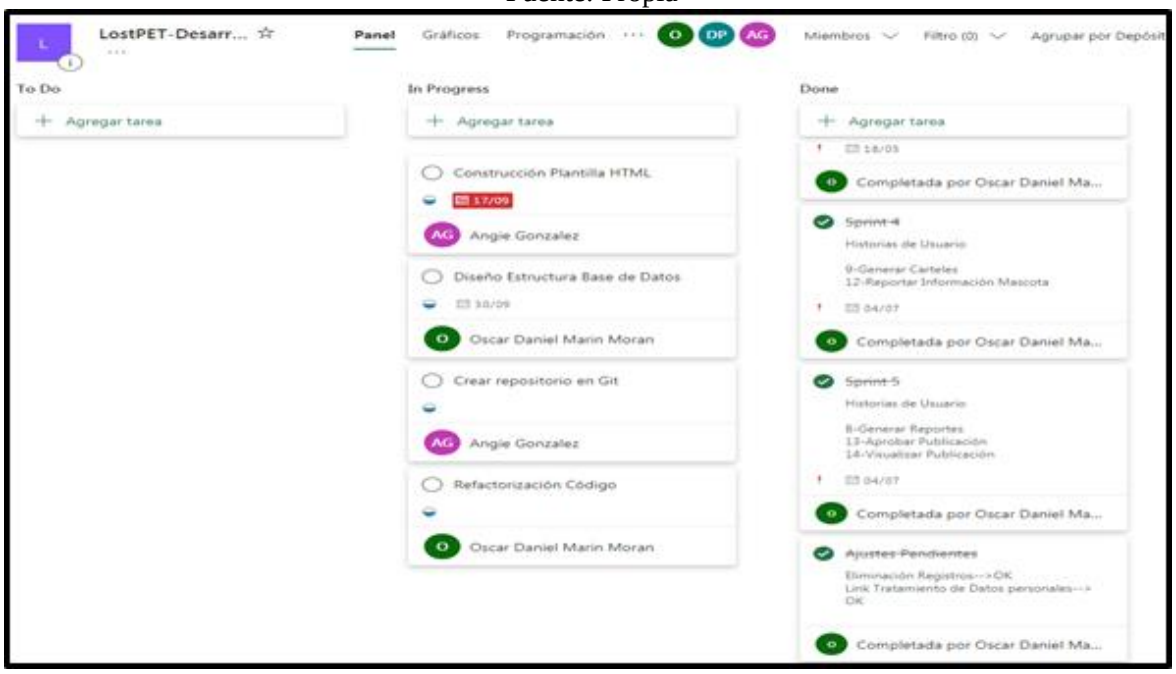
Etapa de Análisis: Es esta etapa se construyó el documento de análisis funcional del requerimiento basado en el levantamiento de información realizado por los estudiantes. Al tener ya definido el requerimiento se empezó a diseñar las historias de usuario, en total 15 que cubrieron el requerimiento planteado. Con lo anterior se pasó a diseñar los diagramas de arquitectura los cuales fueron unificados en el documento SAD.

Imagen 1 – Tablero Planner Etapa de Análisis
Fuente: Propia



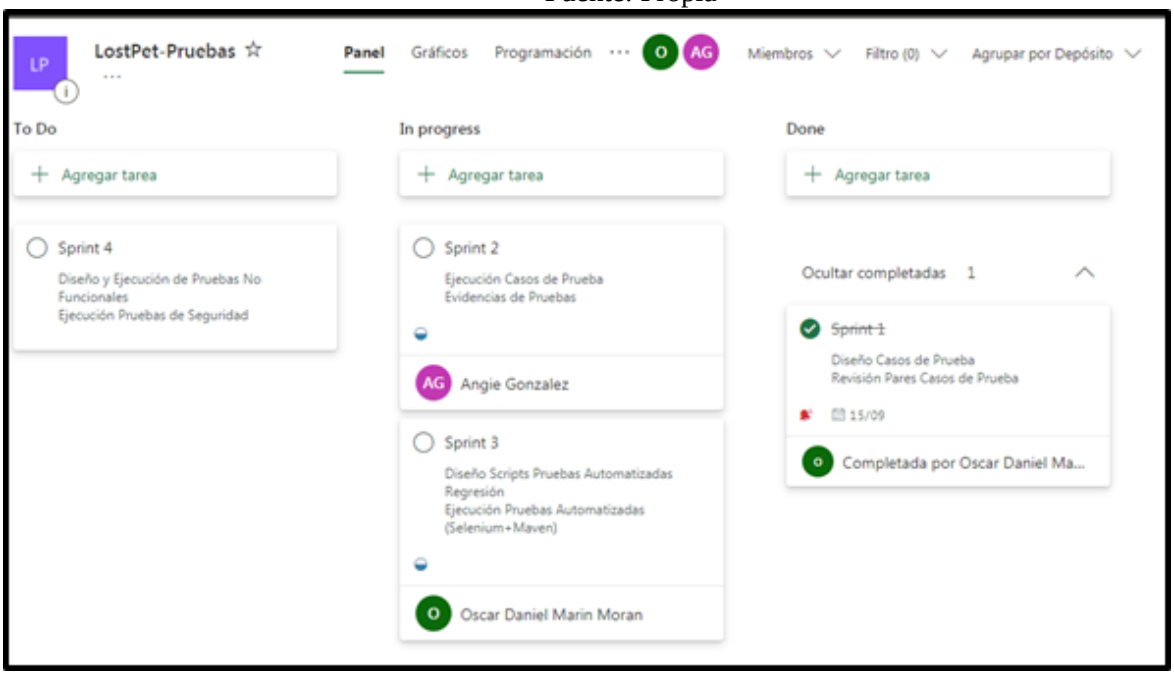
Etapa de Desarrollo: Se dividieron las historias de usuario por Sprint para tener un control sobre las mismas y presentar avances significativos del aplicativo. En total se trabajó con 5 Sprints y además se utilizó parte de estos Sprint para realizar ajustes, correcciones y mejoras a los desarrollos. También se implementaron las validaciones de campos, longitud y seguridad del aplicativo.

Imagen 2 – Tablero Planner Etapa de Desarrollo
Fuente: Propia



Etapa de Pruebas: En esta etapa inicialmente se diseñaron los casos de prueba que cubrirán las funcionalidades del requerimiento. Posterior a esto, estos casos de prueba se dividieron por los mismos Sprint de desarrollo para su posterior ejecución. También se diseñaron los scripts para la ejecución de pruebas automatizadas y no funcionales.

Imagen 3 – Tablero Planner Etapa de Pruebas
Fuente: Propia



10. Proceso de Software

Inicialmente se construyó un documento de **requerimientos** el cual contenía las especificaciones funcionales a implementar. Estas se dividieron por módulos para poder posteriormente crear las historias de usuario. Luego de revisarlo y ajustarlo se pasó a **diseñar** las historias de usuario en donde se detallan de manera clara los módulos a desarrollar. En total se crearon 15 historias de usuario las cuales fueron divididas en 5 Sprints.

Teniendo ya los requerimientos funcionales divididos en historias de usuario, se decidió construir la aplicación en un modelo por capas, utilizando la modelo vista controlador la cual se ajustaba bien al lenguaje de programación que se iba a utilizar, Php.

Definido lo anterior, se empezó con la construcción de la base de datos, iniciando con el modelo entidad relación, que incluía el diseño de las tablas, campos, relaciones, llaves primarias y foráneas, entre otros.

Ya con la base de datos construida, se pasó con la implementación del código fuente y el desarrollo de las historias de usuario. Para ello se utilizó un entorno de desarrollo en las máquinas locales y un versionamiento de código utilizando GitHub. Cada vez que se realizaban ajustes y correcciones al código, por lo menos una vez al día se actualizaba en el repositorio y se realizaban unas pruebas unitarias. Se aplicaron las buenas prácticas de código limpio y en varias ocasiones se tuvo que hacer la refactorización.

A medida que se iban finalizando cada sprint, se empezaban con las pruebas funcionales por parte del grupo QA. Si se encontraban errores, se reportaban en un Excel para llevar el seguimiento y control de los bugs. Al finalizar se tuvieron que corregir 30 incidencias divididas en las siguientes categorías: Ajustes 6, Desarrollo 12, Gráficos 7 y de Texto 5.

Cuando ya las incidencias fueron corregidas y verificadas, se pasó a realizar unas pruebas de regresión para verificar que el comportamiento del aplicativo no hubiese sido alterado.

Posterior a esto se decidió darle un aspecto llamativo al Front de la aplicación, por ello se ajustaron los estilos y las fuentes y se implementaron funciones adicionales en las validaciones. Como el aplicativo sufrió alteraciones nuevamente, se realizaron otras pruebas de regresión que permitiría validar de nuevo el funcionamiento del sistema. En esta fase se ejecutaron también las pruebas de desempeño y de seguridad arrojando resultados positivos.

Como ya no se presentaba ningún error, se pasó a automatizar las pruebas funcionales la cual incluía un flujo completo de la aplicación, estas se realizaron utilizando la metodología BDD (Behavior Driven Development), desarrollo guiado por comportamiento, la cual ejecutaba automáticamente las pruebas de regresión y dejaba un informe con el resultado y las evidencias de la ejecución

Finalmente, se instaló el aplicativo en un ambiente de pruebas, se realizaron las pruebas QA y se certificó para poder colocarlo en ambiente productivo.

Lecciones Aprendidas

- Un buen análisis del proyecto desde su fase inicial permitirá entregar un producto estable.
- Versionar el código fuente permitirá llevar un control de las modificaciones que se hagan al código
- Las pruebas de software desde el inicio del desarrollo del proyecto evitarán reducir los errores antes del paso a producción.
- La automatización de pruebas es una gran ayuda para agilizar la etapa de pruebas del proyecto.
- La refactorización del código permite tener un código fuente entendible y fácil de modificar.
- Las buenas prácticas de código limpio ayudan a que se tenga un código bien estructurado.

Dificultades Presentadas en la Ejecución del Proyecto

- Poco conocimiento de los estudiantes en el lenguaje de programación PHP
- Rediseño del aplicativo en el Front lo cual afectó las pruebas automatizadas ya que se tuvo que realizar correcciones de nuevo y el tiempo que se tenía era limitado.
- Tiempo reducido del grupo por atención a compromisos laborales y académicos.
- No se tenía experiencia en la automatización y ejecución de pruebas funcionales.

10.1 Requerimientos Funcionales

Registrar Usuario

Teniendo en cuenta que el usuario desea registrarse en el aplicativo, el sistema web debe presentar una opción en forma de link en el inicio de sesión que permita registrarse.

Al dar clic sobre el vínculo, se debe dirigir al usuario a un formulario para realizar el registro. Antes de mostrar el formulario, se debe cargar una ventana emergente con los pasos requeridos para poder realizar el registro. Los pasos para realizar el registro deberán ser los siguientes:

- Descargar en el celular la aplicación Google Authenticator
- Registrar los datos en el formulario web y pulsar el botón Registrarse
- Abrir la aplicación Google Authenticator y escanear el código QR que aparece en pantalla
- Digitar el código que aparece en pantalla del celular en el formulario Web

El formulario se presenta en pantalla con los siguientes campos. Se deberá cumplir con la longitud y obligatoriedad y validaciones de cada campo:

Tabla 1-Campos Formulario de Registro de usuarios
Fuente: Propia

Campos	Descripción de los campos	Longitud	Obligatorio SÍ/NO	Validaciones
Correo Electrónico	Texto	100	SI	Estructura del correo valido No debe permitir almacenar el correo si este ya está siendo utilizado por otro usuario
Nombres y Apellidos	Texto	100	Si	N/A
Password	Contraseña	15	Si	Al almacenarse en base de datos debe cifrarse y cumplir con las siguientes restricciones: • La clave debe tener al menos 6 caracteres • La clave no puede tener más de 15 caracteres

				• La clave debe tener al menos una letra minúscula • La clave debe tener al menos una letra mayúscula • La clave debe tener al menos un dato numérico
Campos	**Descripción de los campos**	**Longitud**	**Obligatorio SÍ/NO**	**Validaciones**
Confirmar Password	Contraseña	15	Si	Debe ser igual al password
Registrarme	Button	N/A	N/A	N/A

Luego de realizar las validaciones correspondientes y pulsar el botón Registrarme, el sistema redirige al usuario a un formulario en donde mostrará un código que será escaneado desde el dispositivo móvil del usuario y un campo de texto donde el usuario podrá introducir el código arrojado por el aplicativo Google Authenticator.

Si el código digitado en el campo de texto es correcto, el sistema redirige al usuario al módulo de Perfil de Usuario para que se realicen las actualizaciones correspondientes. De lo contrario, el sistema seguirá solicitando el código hasta que este coincida con el correcto.

Tabla 2-Campos Formulario Factor Doble Autenticación
Fuente: Propia

Campos	Descripción de los campos	Longitud	Obligatorio SÍ/NO	Validaciones
Código Generado	Numérico	6	SI	Validar que el código ingresado sea el mismo arrojado por la aplicación Google Authenticator.
Validar	Button	N/A	N/A	N/A

Recuperar contraseña

Teniendo en cuenta que el usuario olvidó su contraseña y desea recuperarla para poder acceder al aplicativo, el sistema web debe presentar una opción en forma de link en el inicio de sesión que permita recuperar la contraseña.

Al dar clic sobre el vínculo, se debe dirigir al usuario a un formulario para realizar la recuperación de la contraseña.

El formulario se presenta en pantalla con los siguientes campos. Se deberá cumplir con la longitud y obligatoriedad y validaciones de cada campo:

Tabla 3 – Campos Formulario Recuperar Contraseña I
Fuente: Propia

Campos	Descripción de los campos	Longitud	Obligatorio SÍ/NO	Validaciones
Correo Electrónico	Texto	100	SI	Estructura del correo valido Validar que el correo exista en el sistema Validar que el usuario se encuentre activo
Enviar	Button	N/A	N/A	N/A

Luego de realizar las validaciones correspondientes, el sistema envía un correo electrónico a la dirección registrada del usuario con la siguiente información:

Asunto: Recuperar Password
Cuerpo del Correo: Estimado Usuario. Se ha solicitado el restablecimiento de su contraseña. Para restaurarla, por favor visite el siguiente link Cambiar Password. Muchas gracias.

El enlace debe ser enviado cifrado al usuario y verificar que sea válido y no haya sido utilizado.

Luego que el usuario ingresa al vínculo enviado al correo, el sistema o dirige a un formulario donde podrá realizar el cambio de la contraseña:

Tabla 4 - Campos Formulario Recuperar Contraseña II
Fuente: Propia

Campos	Descripción de los campos	Longitud	Obligatorio SÍ/NO	Validaciones
Nueva Contraseña	Contraseña	15	SI	N/A
Confirmar Contraseña	Contraseña	15	SI	Al almacenarse en base de datos debe cifrarse.
Cambiar Contraseña	Button	N/A	N/A	N/A
Iniciar Sesión	Vínculo	N/A	N/A	N/A

Luego de realizar las validaciones correspondientes, el sistema informa al usuario que se cambió la contraseña exitosamente. El usuario podrá iniciar sesión inmediatamente ingresando al vínculo Iniciar Sesión utilizando la nueva clave.

Iniciar Sesión

Dado que el usuario ya realizó el registro en el aplicativo, el sistema deberá presentar un formulario para el inicio de sesión. El formulario se presenta en pantalla con los siguientes campos. Se deberá cumplir con la longitud y obligatoriedad y validaciones de cada campo:

Tabla 5 - Campos Formulario de Registro de usuarios
Fuente: Propia

Campos	Descripción de los campos	Longitud	Obligatorio SÍ/NO	Validaciones
Correo Electrónico	Texto	100	SI	Estructura del correo valido El correo debe existir en base de datos.
Contraseña	Contraseña	15	SI	Debe coincidir con la registrada en la base de datos
Ingresar	Button	N/A	N/A	N/A
Link Registrarme	Vínculo	N/A	N/A	N/A
Link Recuperar Contraseña	Vínculo	N/A	N/A	N/A

Además de las validaciones anteriores, el sistema deberá verificar que el usuario esté activo en el sistema y que la contraseña sea correcta.

Luego que el sistema verifica que la información es correcta, el sistema dirige al usuario a un formulario en donde mostrará un campo de texto donde el usuario podrá introducir el código arrojado por el aplicativo Google Authenticator.

Tabla 6 – Campos Formulario Factor Doble Autenticación
Fuente: Propia

Campos	Descripción de los campos	Longitud	Obligatorio SÍ/NO	Validaciones
Código Generado	Numérico	6	SI	Validar que el código ingresado sea el mismo arrojado por la aplicación Google Authenticator.
Validar	Button	N/A	N/A	N/A

Si el código digitado en el campo de texto es correcto, el sistema deberá verificar si el Perfil ya está actualizado. De lo contrario, el sistema seguirá solicitando el código hasta que este coincida con el correcto. Si el perfil no ha sido actualizado el sistema redirige al usuario al módulo de Perfil de Usuario para que se realicen las actualizaciones correspondientes. Si este ya ha sido actualizado, el sistema redirige al usuario a la página de bienvenida del aplicativo.

El sistema deberá guardar una auditoría en donde se almacena un registro por cada vez que el usuario inicie sesión exitosamente.

Modificar Perfil

El sistema deberá presentar una opción para que el usuario pueda modificar los datos del perfil. Al ingresar a esta opción se presentará un formulario en pantalla con los siguientes campos.

Se deberá cumplir con la longitud y obligatoriedad y validaciones de cada campo:

Tabla 7 - Campos Formulario de Modificación de Perfil
Fuente: Propia

Campos	Descripción de los campos	Longitud	Obligatorio SÍ/NO	Validaciones
Correo Electrónico	N/A	N/A	SI	Campo de solo lectura. No se podrá modificar
Nombres y Apellidos	Texto	100	Si	N/A
Dirección	Texto	200	No	N/A

Numero Teléfono	Numérico	10	No	N/A
Departamento	Lista	N/A	Si	Los datos a cargar son los proporcionados por los códigos DANE para cada departamento
Municipio	Lista	N/A	Si	Los datos a cargar son los proporcionados por los códigos DANE para cada municipio. Al elegir el departamento, en este campo sólo se deberán cargar los municipios asociados al departamento elegido.
Tipo Usuario	Lista	N/A	Si	Los datos a mostrar en la lista serán: • Administrador • Propietario de Mascota • Fundación Animal • Informador Mascota Solo se debe mostrar la opción Administrador al usuario que tenga este perfil. De lo contrario no se debe mostrar.
Acepto envío medios electrónicos	CheckBox	N/A	Si	N/A
Acepto tratamiento de datos	CheckBox	N/A	Si	Junto al CheckBox se deberá presentar un vínculo para poder leer la política de tratamiento de datos. (Documento Adjunto)
Campos	**Descripción de los campos**	**Longitud**	**Obligatorio SÍ/NO**	**Validaciones**
Actualizar Perfil	Button	N/A	N/A	N/A

Luego de realizar las validaciones correspondientes, el sistema actualizará la información correspondiente siempre y cuando se haya modificado algún campo y mostrará un mensaje en pantalla indicando que el perfil se actualizó exitosamente. De lo contrario no ejecutará ninguna acción.

Módulo Administración

El sistema mostrará un módulo de Administración, el cual SOLO podrá ser visualizado por los usuarios que tengan perfil de Administrador. De lo contrario este módulo NO se mostrará. El módulo presentará las siguientes opciones:

Usuarios

Al ingresar a esta opción el sistema presentará un listado en forma de filas y columnas con la siguiente información:

Tabla 8 – Campos Listado Usuarios
Fuente: Propia

Nombre de Columna	**Observaciones**
Correo	N/A
Nombres	N/A
Dirección	N/A
Municipio	Se debe presentar el nombre del municipio
Teléfono	N/A
Tipo Usuario	Se debe mostrar la descripción del Tipo de Usuario
Estado	Se debe mostrar la descripción del Estado
Acción	Debe mostrar 2 opciones. Una para modificar el registro y otra para poder eliminarlo.

Modificar Registro Usuario

Esta opción debe visualizarse por cada registro listado. Al momento de dar clic sobre éste, el sistema remite al usuario a un formulario para poder modificar la información del usuario. Se deberá cumplir con la longitud y obligatoriedad y validaciones de cada campo:

Tabla 9 - Campos Formulario de Modificación Usuario
Fuente: Propia

Campos	**Descripción de los campos**	**Longitud**	**Obligatorio SÍ/NO**	**Validaciones**
Correo Electrónico	N/A	N/A	SI	Campo de solo lectura. No se podrá modificar
Nombres y Apellidos	Texto	100	Si	Se debe cargar el dato almacenado en base de datos para el registro seleccionado
Dirección	Texto	200	No	Se debe cargar el dato almacenado en base de datos para el registro seleccionado

Número Teléfono	Numérico	10	No	Se debe cargar el dato almacenado en base de datos para el registro seleccionado
Tipo Usuario	Lista	N/A	Si	Los datos a mostrar en la lista serán: • Administrador • Propietario de Mascota • Fundación Animal • Informador Mascota Solo se debe mostrar la opción Administrador al usuario que tenga este perfil. De lo contrario no se debe mostrar.
Estado	Lista	N/A	Si	Se debe listar los valores almacenados en la tabla Estados de usuario: • Activo • Inactivo
Campos	**Descripción de los campos**	**Longitud**	**Obligatorio SÍ/NO**	**Validaciones**
Acepto envío medios electrónicos	CheckBox	N/A	Si	N/A
Acepto tratamiento de datos	CheckBox	N/A	Si	Junto al CheckBox se deberá presentar un vínculo para poder leer la política de tratamiento de datos. (Documento Adjunto)
Actualizar Perfil	Button	N/A	N/A	N/A

Al momento de pulsar sobre el botón actualizar perfil, el sistema notifica al usuario de la modificación exitosa del registro. Si no se modifica ningún campo, el sistema no realizará ninguna acción.

Eliminar Registro Usuario

Esta opción debe visualizarse por cada registro listado. Al momento de dar clic sobre éste, el sistema valida si no existen registros relacionados a este usuario para poder eliminarlo. Si existen registros relacionados, el sistema NO debe permitir la eliminación, de lo contrario el registro quedará eliminado e informa al usuario en pantalla el resultado de la acción.

Razas

Al ingresar a esta opción el sistema presentará un listado en forma de filas y columnas con la siguiente información:

Tabla 10 – Listado Razas
Fuente: Propia

Nombre de Columna	**Observaciones**
Id Raza	Se debe presentar el código de la raza
Nombre Raza	Se muestra el nombre de la raza
Tipo Mascota	Se debe listar el nombre del tipo de mascota de la raza correspondiente.
Acción	Debe mostrar 2 opciones. Una para modificar el registro y otra para poder eliminarlo.

Se debe presentar una opción que permite agregar una nueva raza. Al pulsar sobre esta opción, se presentará un formulario para agregar una nueva raza. Los campos para realizar la inserción de una nueva raza son las siguientes:

Tabla 11 – Campos Formulario Crear Raza
Fuente: Propia

Campos	**Descripción de los campos**	**Longitud**	**Obligatorio SÍ/NO**	**Validaciones**
Nombre Raza	Texto	50	SI	Campo Obligatorio
Tipo de Mascota	Lista Desplegable	N/A	Si	Se deben cargar el nombre de los tipos de mascota.

Modificar Raza

Esta opción debe visualizarse por cada registro listado. Al momento de dar clic sobre este, el sistema remite al usuario a un formulario para poder modificar la información de la raza. Se deberá cumplir con la longitud y obligatoriedad y validaciones de cada campo:

Tabla 12 - Campos Formulario Modificar Raza
Fuente: Propia

Campos	Descripción de los campos	Longitud	Obligatorio SÍ/NO	Validaciones
Nombre Raza	Texto	50	SI	Campo Obligatorio
Tipo de Mascota	Lista Desplegable	N/A	Si	Se deben cargar el nombre de los tipos de mascota.

Eliminar Raza

Esta opción debe visualizarse por cada registro listado. Al momento de dar clic sobre éste, el sistema valida si no existen registros relacionados a esta raza para poder eliminarla. Si existen registros relacionados, el sistema NO debe permitir la eliminación, de lo contrario el registro quedará eliminado e informa al usuario en pantalla el resultado de la acción.

Auditorias

Al momento de ingresar a esta opción, se debe visualizar un listado en forma de filas y columnas con la información del acceso al aplicativo de los usuarios registrados. El listado debe contener la siguiente información:

- Id usuario
- Nombre Usuario
- Correo Usuario
- Fecha Ingreso
- Hora Ingreso

De igual manera debe existir un buscador dinámico, el cual permite buscar y filtrar la información por cada una de las columnas y mostrar el resultado en pantalla.

Esta opción solo debe ser permitida acceder a los usuarios que tengan el perfil de administrador.

Reportes

Este módulo permitirá descargar solo a usuarios con perfil administrador, reportes en formato CSV con la siguiente información:

- Reporte de Usuarios Registrados
- Reporte de Auditoría
- Reporte de Razas
- Reporte de Mascotas

Cada reporte se descarga en la máquina local del usuario y permite abrirlos en formato CSV, separado por comas. La información mostrada permitirá visualizar los registros alojados en las tablas de usuarios, auditoría, razas y mascotas respectivamente y podrán ser exportados a Excel para realizar diferentes filtros.

Carteles

El sistema presentará en pantalla un listado en forma de columnas y filas con la siguiente información:

- Usuario Registrado
- Id Mascota
- Nombres
- Estado Mascota
- Tipo Mascota
- Sexo
- Edad
- Acción

En la columna Acción deberá presentar un icono por cada registro presentado que permitirá generar el cartel en formato PDF con el código QR de la mascota seleccionada.

Este cartel solo será generado para las mascotas que estén en estado Perdida o Encontrada. Para las mascotas en estado Recuperado se deberá mostrar una carita feliz y un mensaje indicando que no es necesario generar el cartel.

Aprobar publicaciones

El sistema deberá cargar en pantalla un listado en forma de filas y columnas con la siguiente información:

- **Aprobar** (Se cargará un CheckBox por cada registro listado)
- **Nombres** (Nombre de la mascota)
- **Estado Mascota** (Perdida, Encontrada, Recuperada)
- **Publicación** (Leyenda con la misma información publicada en el cartel)
- **Fecha Publicación**
- **Foto** (Foto de la mascota cargada en el registro)

Solo se debe cargar la información de las publicaciones que aún no han sido aprobadas por el usuario administrador.

Se debe presentar un botón “Aprobar Seleccionadas” para que el sistema pueda actualizar una o varios registros al mismo tiempo. Si no elige ningún registro el sistema debe informar al usuario que no ha seleccionado ninguno, de lo contrario, si elige al menos uno, el sistema actualiza el estado de la publicación e informa al usuario que los registros fueron actualizados exitosamente.

Módulo de Mascotas

Esta opción será mostrada a todos los usuarios registrados en el aplicativo y permitirá realizar las siguientes acciones:

Mis Mascotas

En este módulo se listarán en forma de filas y columnas todas las mascotas que el usuario haya registrado y se presentarán los siguientes campos:

- Nombres
- Estado mascota
- Tipo Mascota
- Raza
- Color
- Manchas
- Tamaño
- Sexo
- Edad
- Observaciones
- Acción
- Cartel

En cada campo deberá mostrarse la descripción más no los códigos con que se almacenan en la base de datos.

En la columna de acción, se presentarán dos iconos, uno para modificar el registro y otro para eliminarlo.

Modificar Mascota

Al pulsar sobre este icono, el sistema dirigirá al usuario a una pantalla en forma de formulario en donde se deberá precargar la información en cada campo correspondiente al registro seleccionado. Se presentará la siguiente información:

Tabla 13 - Campos Formulario Modificar Mascota
Fuente: Propia

Campos	Descripción de los campos	Longitud	Obligatorio SÍ/NO	Validaciones
Estado Mascota	Lista Desplegable	N/A	SI	Debe cargar los estados de las mascotas. Inicialmente deberá mostrar: • Encontrada • Pérdida • Recuperada Por defecto, debe cargar el registro que tiene almacenado para esta mascota
Tipo Mascota	Lista Desplegable	N/A	SI	• Este campo es de solo lectura y debe cargar el registro que tiene almacenado para esta mascota.
Raza	Lista Desplegable	N/A	Sí	Deberá cargar el nombre de las razas registrados en el módulo de Raza. Solo debe cargar según la relación que tiene con el campo tipo mascota. Por defecto, debe cargar el registro que tiene almacenado para esta mascota

Nombres	Texto	30	Si	Deberá cargar el nombre almacenado en la base de datos para este registro
Color	Lista Desplegable	N/A	Si	Deberá cargar el nombre de los colores almacenados en la base de datos. Por defecto, debe cargar el registro que tiene almacenado para esta mascota.
Campos	**Descripción de los campos**	**Longitud**	**Obligatorio SÍ/NO**	**Validaciones**
Manchas	Lista Desplegable	N/A	Si	Deberá cargar el nombre de los colores almacenados en la base de datos. Por defecto, debe cargar el registro que tiene almacenado para esta mascota.
Tamaño	Lista Desplegable	N/A	SI	Deberá cargar el nombre de los tamaños almacenados en la base de datos. Por defecto, debe cargar el registro que tiene almacenado para esta mascota.
Sexo	Lista Desplegable	N/A	SI	Deberá cargar el nombre de los sexos almacenados en la base de datos. Por defecto, debe cargar el registro que tiene almacenado para esta mascota
Edad Mascota	Numérico	N/A	2	Solo debe permitir almacenar valores numéricos.
Fecha Evento	Calendario	N/A	SI	Se debe mostrar un campo tipo calendar para que el usuario elija la fecha correspondiente
Dirección Ultima Vez Visto / Encontrada	Texto	100	SI	Permite caracteres especiales como #

Observaciones	Área de Texto	100	No	Permite escribir una descripción detallada acerca de la mascota u otra información relevante.
Imagen	Imagen	N/A	Si	Los formatos permitidos son JPG y PNG
Fecha Recuperada	Calendario	N/A	No	Este campo se habilitará solo cuando en el campo Estado Mascota el usuario haya elegido la opción RECUPERADA. Al habilitarse se volverá obligatorio.
Campos	**Descripción de los campos**	**Longitud**	**Obligatorio SÍ/NO**	**Validaciones**
Observaciones Recuperación	Área de Texto		No	Este campo se habilitará solo cuando en el campo Estado Mascota el usuario haya elegido la opción RECUPERADA. Al habilitarse se volverá obligatorio.
Modificar Mascota	Button	N/A	N/A	

Cuando el usuario pulsa el botón Modificar, el sistema verifica que todos los campos obligatorios estén correctamente diligenciados. Si son correctos, el sistema actualiza el registro e informa al usuario de la ejecución exitosa del proceso. De lo contrario, si el usuario no actualiza ningún campo, el sistema no realizará ninguna modificación. Al terminar de ejecutar esta acción, el sistema remitirá al usuario al módulo de mis mascotas.

Reportar Mascota

Cuando el usuario ingrese a esta opción, el sistema presenta un formulario para el diligenciamiento de los campos con la siguiente información:

Tabla 14 - Campos Formulario Reportar Mascota
Fuente: Propia

Campos	Descripción de los campos	Longitud	Obligatorio SÍ/NO	Validaciones
Estado Mascota	Lista Desplegable	N/A	SI	Solo debe mostrar las opciones Perdida y Encontrada
Tipo Mascota	Lista Desplegable	N/A	SI	Debe cargar los tipos de mascota. Inicialmente deberá mostrar:

				• Perro • Gato
Raza	Lista Desplegable	N/A	Sí	Deberá cargar el nombre de las razas registrados en el módulo de Raza. Solo debe cargar según la relación que tiene con el campo tipo mascota.
Nombres	Texto		Si	N/A
Color	Lista Desplegable	N/A	Si	Deberá cargar el nombre de los colores almacenados en la base de datos.
Manchas	Lista Desplegable	N/A	Si	Deberá cargar el nombre de los colores almacenados en la base de datos.
Tamaño	Lista Desplegable	N/A	SI	Deberá cargar el nombre de los tamaños almacenados en la base de datos.
Sexo	Lista Desplegable	N/A	SI	Deberá cargar el nombre de los sexos almacenados en la base de datos.
Edad Mascota	Numérico	N/A	2	Solo debe permitir almacenar valores numéricos.
Fecha Evento	Calendario	N/A	SI	Se debe mostrar un campo tipo calendar para que el usuario elija la fecha correspondiente
Campos	**Descripción de los campos**	**Longitud**	**Obligatorio SÍ/NO**	**Validaciones**
Dirección Última Vez Visto / Encontrada	Texto	200	SI	Permite caracteres especiales como #
Observaciones	Área de Texto	100	No	Permite escribir una descripción detallada acerca de la mascota u otra información relevante.
Imagen	Imagen	N/A	Si	Los formatos permitidos son JPG y PNG
Reportar Mascota	Button	N/A	N/A	

Cuando el usuario pulsa el botón Reportar Mascota, el sistema verifica que todos los campos obligatorios estén correctamente diligenciados. Si son correctos, el sistema inserta el registro e informa al usuario de la

ejecución exitosa del proceso. Al terminar de ejecutar esta acción, el sistema remitirá al usuario al módulo de mis mascotas en donde podrá visualizar la mascota previamente creada.

Notificaciones

En esta sección se cargarán todas las notificaciones que los usuarios externos e internos del aplicativo han realizado para la mascota del usuario que está registrado. Deberá mostrar la siguiente información en forma de columnas y filas:

- Nombre de la mascota
- Fecha Evento
- Observaciones
- Teléfono de Contacto
- Icono (Cargará de color azul inicialmente)

Cuando una notificación es leída, el usuario podrá dar clic sobre el icono y este cambiará de color, de azul a gris lo que indicará que la notificación ya fue leída por el usuario.

Consultas

Esta sección permitirá realizar diferentes consultas sobre la información registrada en el aplicativo. Se mostrará a todos los usuarios registrados en el sistema.

Buscar Mascota

Esta sección deberá permitir realizar una búsqueda avanzada con los siguientes filtros, que combinados o no deberán buscar la información correspondiente:

- Estado Mascota
- Tipo Mascota
- Raza
- Color
- Manchas
- Tamaño
- Departamento (Si se elige Bogotá, deberá desplegar una lista desplegable con las localidades
- Municipio (Se deben cargar los municipios según el departamento escogido)

Si no se selecciona ningún campo como filtro de búsqueda, se deberá mostrar toda la información almacenada para las mascotas.

Si el usuario selecciona un campo de búsqueda como filtro o campos combinados, se deberá mostrar la siguiente información en forma de filas y columnas:

- Nombres Mascota
- Estado Mascota
- Tipo Mascota
- Raza
- Color
- Manchas
- Tamaño
- Sexo
- Edad
- Municipio
- Localidad
- Imagen (Se muestra una imagen de la mascota)
- Icono (El icono permitirá dirigirse al formulario para reportar información sobre la mascota visualizada). Solo se debe mostrar este icono a las mascotas en estado Perdida o Encontrada. Mascotas en estado Recuperado deben mostrar una carita feliz y un mensaje indicando que ya no es necesario reportar información.

Publicaciones

El sistema deberá cargar en pantalla un listado en forma de filas y columnas con la siguiente información:

- **Nombres** (Nombre de la mascota)
- **Estado Mascota** (Perdida, Encontrada, Recuperada)
- **Publicación** (Leyenda con la misma información publicada en el cartel)
- **Fecha Publicación**
- **Foto** (Foto de la mascota cargada en el registro)
- **Registrar Novedad** (Icono para registrar información sobre la mascota). Solo se debe mostrar esta opción a las mascotas en estado Perdida o Encontrada. Mascotas en estado Recuperado deben mostrar una carita feliz y un mensaje indicando que ya no es necesario reportar información.

Solo se debe cargar la información de las publicaciones que han sido aprobadas por el usuario administrador.

El icono de registrar novedad solo se debe habilitar a las mascotas que estén reportadas como Perdida o Encontrada. Cuando se pulse sobre este icono, el sistema carga en otra pestaña del navegador el formulario para reportar información sobre esta mascota.

Para las que están reportadas como Recuperadas, se debe mostrar una carita feliz y mostrar un mensaje informando que ya no es necesario reportar información para esta mascota.

En esta pantalla, debe existir un buscador que permite realizar filtros sobre la consulta realizada. Se debe filtrar por las columnas de nombres, estado mascota, publicación y fecha publicación.

Reportar Información Mascota

Luego que el usuario escanee el código QR, se debe presentar un link que el usuario podrá visitar para registrar información sobre la mascota Perdida o Encontrada.

Al visitar este link debe aparecer un formulario con la información general de la mascota para la cual se requiere registrar información junto con los siguientes campos:

Tabla 15 – Campos Formulario Reportar Información Mascota
Fuente: Propia

Campos	Descripción de los campos	Longitud	Obligatorio SÍ/NO	Validaciones
Fecha Evento	Fecha	N/A	SI	Obligatorio
Teléfono Contacto	Numérico	10	Si	Obligatorio
Observaciones	Área de Texto	500	Si	Obligatorio
Enviar Información	Button	N/A	N/A	N/A

Al momento de enviar la información, el sistema informa al usuario y agradece por el registro informando sobre la mascota.

Se debe enviar un correo electrónico al usuario dueño de la mascota, informando que una notificación fue enviada y para poder ver los detalles debe pulsar un link que se envía dentro del cuerpo del correo y visitar el sitio web. Esta notificación debe ser cargada en el módulo de notificaciones.

Contáctenos

Esta opción permitirá a los usuarios enviar mensajes al administrador del aplicativo para registrar alguna duda, sugerencia, reclamo, entre otros.

Tabla 16 - Campos Formulario Contáctenos
Fuente: Propia

Campos	Descripción de los campos	Longitud	Obligatorio SÍ/NO	Validaciones
Asunto	Varchar	50	Si	Obligatorio
Mensaje	Varchar	200	SI	Obligatorio
Enviar	Button	N/A	N/A	N/A

El sistema deberá validar la obligatoriedad y longitud de campos. Al pulsar el botón enviar, se envía un mensaje al usuario indicando que el mensaje se envió exitosamente y pronto será contactado por el administrador del sistema.

Cuando el administrador del aplicativo ingrese al sistema, en la página principal, se mostrará un mensaje indicando si tiene o no mensaje por leer y la cantidad. Debe presentar un link que lo dirigirá al módulo de Mensajes el cual solo puede ser accedido por el usuario administrador.

Cuando ingrese a dicho módulo se presentarán las siguientes columnas:

- Fecha de mensaje
- Usuario
- Correo Electrónico
- Asunto
- Mensaje
- Leído (Aquí se mostrará en forma de icono azul las notificaciones que aún no han sido leídas)

Si el usuario administrador desea colocar un mensaje como leído, solo basta con pararse y dar clic en el icono y el mensaje quedará como leído.

Historias de Usuario (Ver Anexos):

1. Registrar Usuario
2. Iniciar sesión
3. Recuperar Contraseña
4. Actualizar Perfil Usuario
5. Gestionar Usuarios
6. Gestionar Razas
7. Visualizar Auditoría
8. Generar Reportes
9. Generar Carteles
10. Reportar Mascota
11. Visualizar Mis Mascotas
12. Reportar Información Mascota
13. Aprobar Publicación
14. Visualizar Publicación
15. Contáctenos

10.2 Requerimientos No Funcionales

Pruebas de Desempeño

Se deberán ejecutar este tipo de pruebas sobre el aplicativo desarrollado para evaluar las características relacionadas con el rendimiento, flujo de ejecución, tiempos de respuesta y confiabilidad operativa.

Tiempo de Respuesta: es el intervalo de tiempo que transcurre entre la solicitud de un usuario al sistema y la respuesta de este último.

Rendimiento: es la tasa a la cual el sistema puede atender las peticiones.

Capacidad: Máxima cantidad de trabajo útil que se puede realizar por unidad de tiempo.

Para evaluar lo anterior, se simulará la iteración de 100 usuarios tratando de ejecutar las siguientes acciones en un periodo de tiempo de 1 segundo al mismo tiempo:

- Inicio de sesión en el aplicativo.
- Búsqueda de Mascotas (Desde el Front)
- Búsqueda de Mascotas (Desde la base de datos)

Se deberán mostrar los resultados de cada una de las pruebas en donde se visualice que el desempeño de la aplicación es óptima al recibir una concurrencia de 100 usuarios ejecutando las diferentes acciones antes mencionadas.

10.3 Diseño y Arquitectura

Motivadores de Negocio

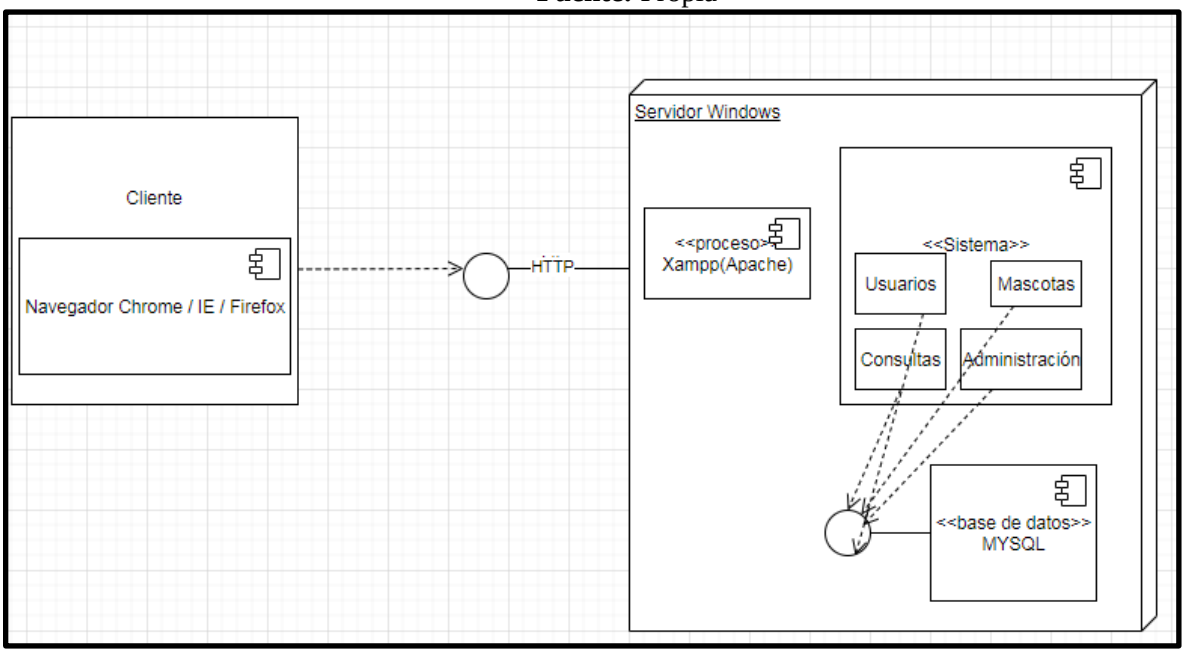
- Posicionamiento de la marca LostPet en el departamento de Cundinamarca como número 1 en la centralización de animales perdidos (perros y gatos)
- Reducir en un 70% el tiempo en la recuperación de una mascota perdida.
- Salvaguardar al 90% los datos personales de los usuarios para evitar llamadas fraudulentas.

Lo anterior se tuvo en cuenta debido a que la desaparición de perros y gatos domésticos en el departamento de Cundinamarca ha crecido, en especial en Bogotá en donde se calcula que por día se puede extraviar de 10 a 12 mascotas. También se ha utilizado esto como método para estafar a las personas pidiendo gran cantidad de dinero para devolver a su animal de compañía.

Para el desarrollo de lo anteriormente mencionado, se definió por parte del equipo LostPet construir un aplicativo web que permitiera registrar la información de animales perdidos o encontrados. Mediante la generación de un cartel PDF el cual contiene un código QR, los informadores de mascotas podrán reportar alguna pista de la mascota extraviada que permitirá al dueño poder recuperarla. En ningún momento se expondrán datos personales de los dueños de las mascotas ya que cualquier información proporcionada será canalizada por correo electrónico.

8.3.1 Diagrama de Despliegue

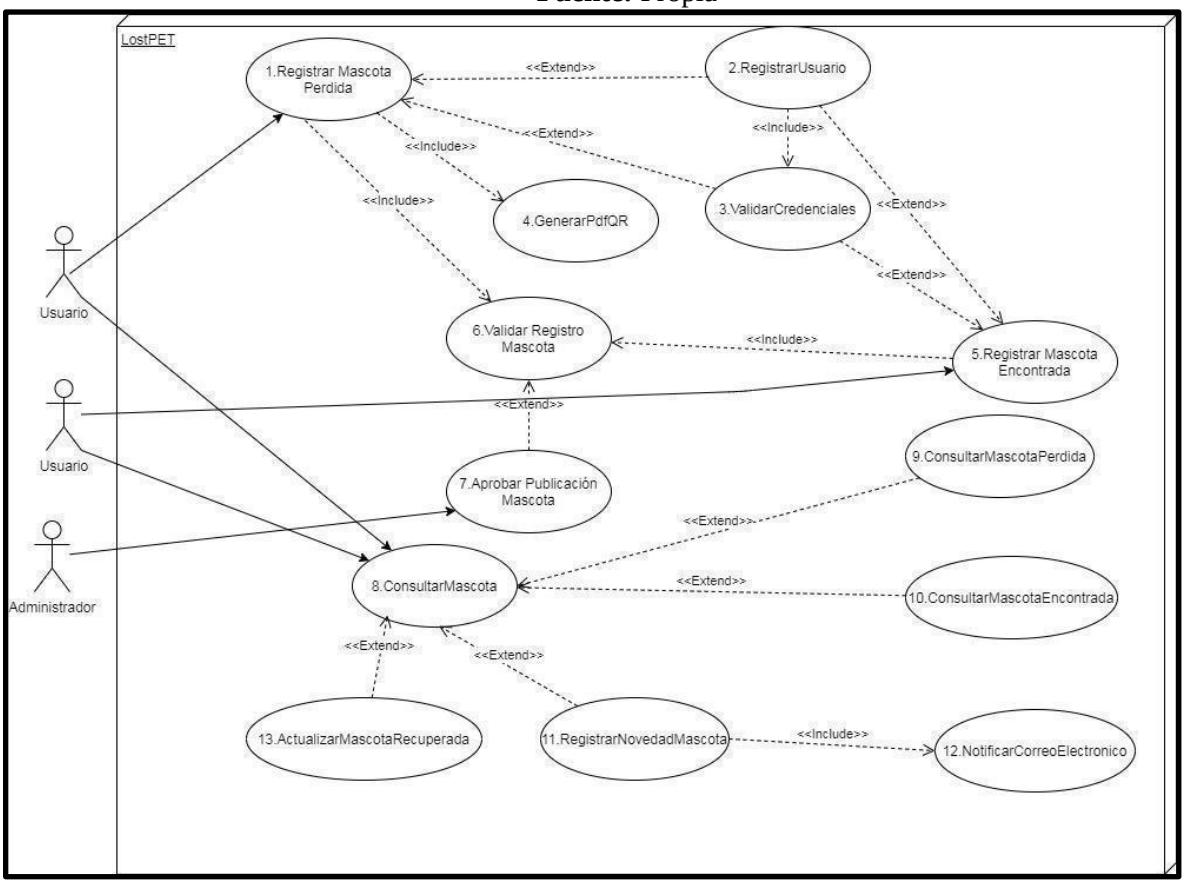
Imagen 4: Diagrama de despliegue
Fuente: Propia



Se realizará el despliegue en una máquina “servidor” con el SO de Windows, mediante la instalación y ejecución de un Xampp, inicializando el servicio de Apache y de MySql como servicio de Windows para así evitar que en dado caso de un reinicio el servicio quede abajo.

8.3.2 Caso de Uso Arquitecturalmente relevante

Imagen 5: Diagrama de Caso de Uso
Fuente: Propia



El usuario podrá (1,5) registrar una mascota (perdida o encontrada) siempre y cuando se haya (2) registrado y (3) validado sus credenciales de acceso al aplicativo.

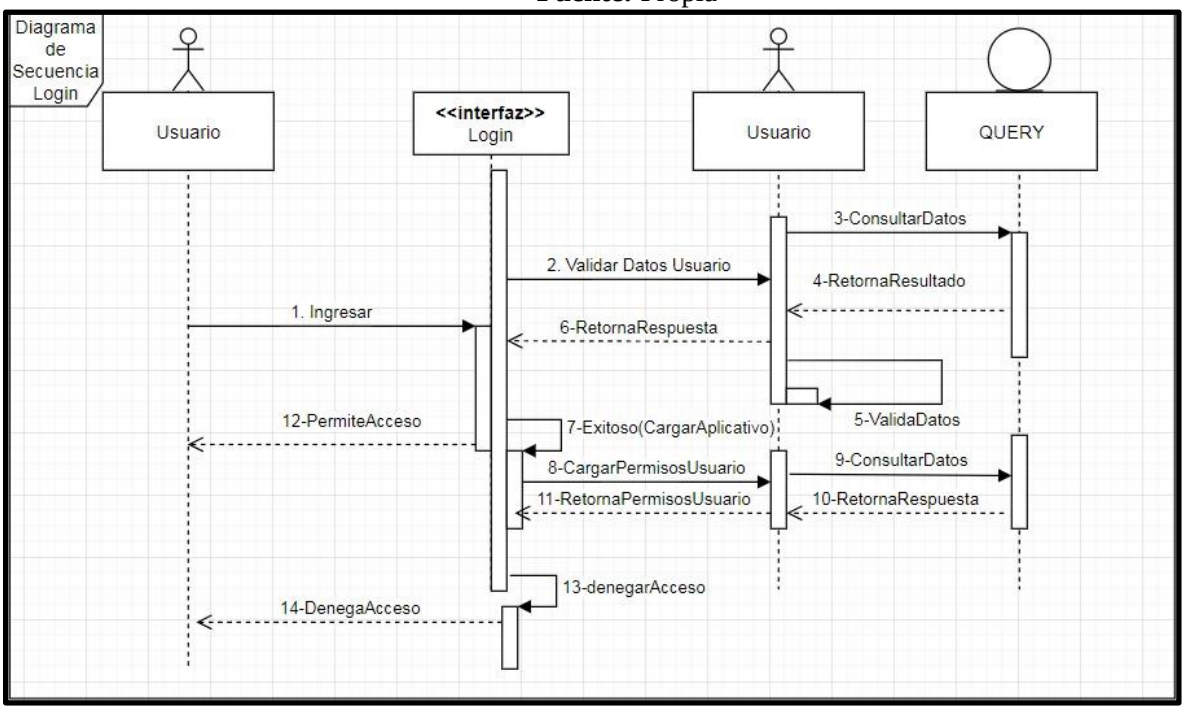
Cuando la mascota se registra, el sistema (4) genera automáticamente un cartel PDF que contiene un código QR con información de la mascota previamente registrada. Este registro queda en un estado para que el Administrador (6) valide el registro creado y si es el caso (7) apruebe la aplicación.

Posterior a esto, se podrá (8, 9,10) consultar la información de la mascota para (11) registrar alguna novedad y si es así se (12) notificará por correo al dueño de la mascota.

Por último, si una mascota es recuperada, permitirá (13) actualizar su estado y no aparecerá en el listado de publicaciones.

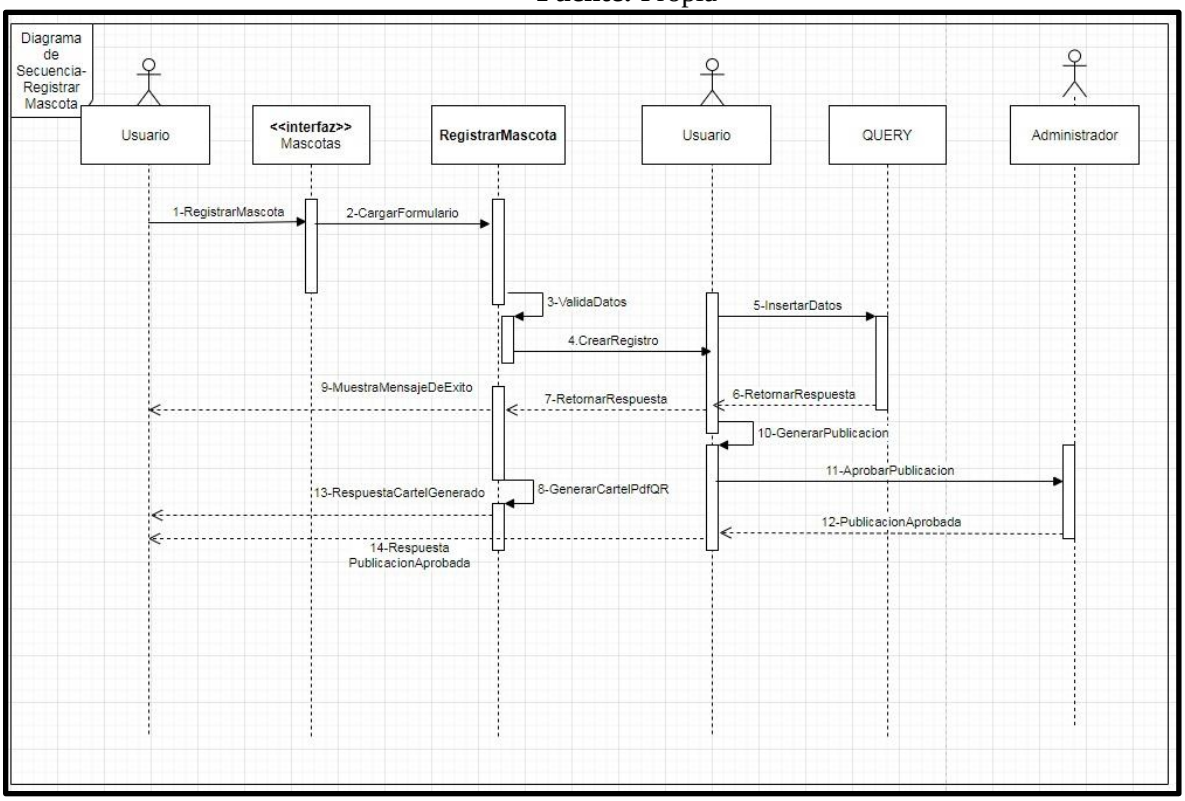
8.3.3 Diagramas de Secuencia

Imagen 6: Diagrama de Secuencia-Login
Fuente: Propia



Mediante la interfaz gráfica de inicio de sesión, el usuario realizará la petición para poder ingresar al sistema. Allí se validaron los datos mediante una consulta realizada a la base de datos. Esta retornará el resultado, si es exitoso, aplicará los permisos correspondientes al perfil que tenga en la base de datos, y lo permitirá ingresar al aplicativo. De lo contrario, el acceso será denegado y se informará al usuario el motivo por el cual no puedo ingresar.

Imagen 6: Diagrama de Secuencia – Registrar Mascota
Fuente: Propia

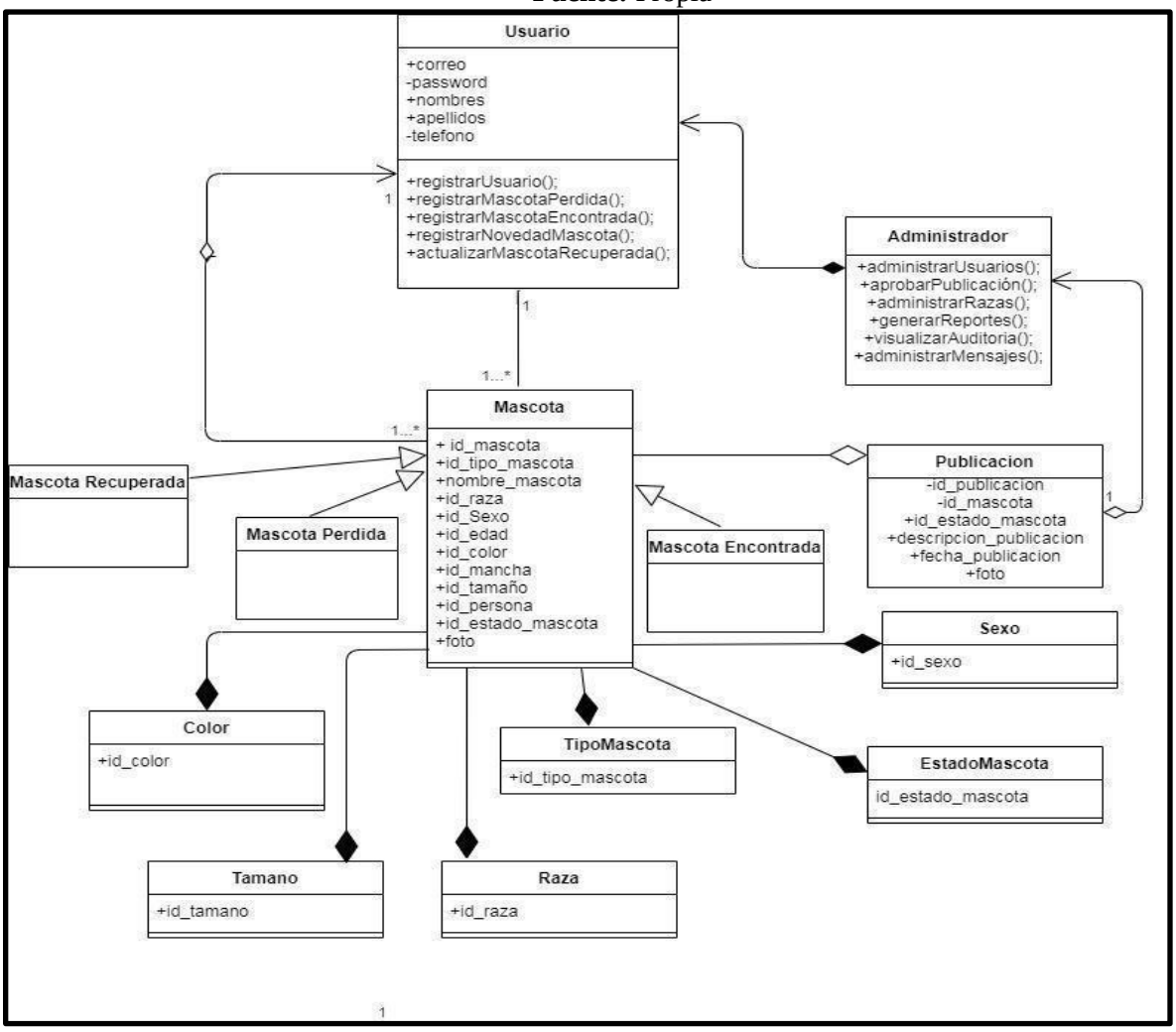


El usuario registrará una mascota utilizando el formulario para realizar esta acción. Se validará que los datos obligatorios estén correctamente diligenciados y posterior a esto se creará el registro insertando la información en la base de datos.

Se retornará la respuesta al usuario informando que el registro fue exitoso. De igual manera el sistema generará el cartel Pdf y creará la publicación para que el usuario administrador la apruebe y pueda ser visualizada por los demás usuarios.

8.3.4 Diagrama de Clases

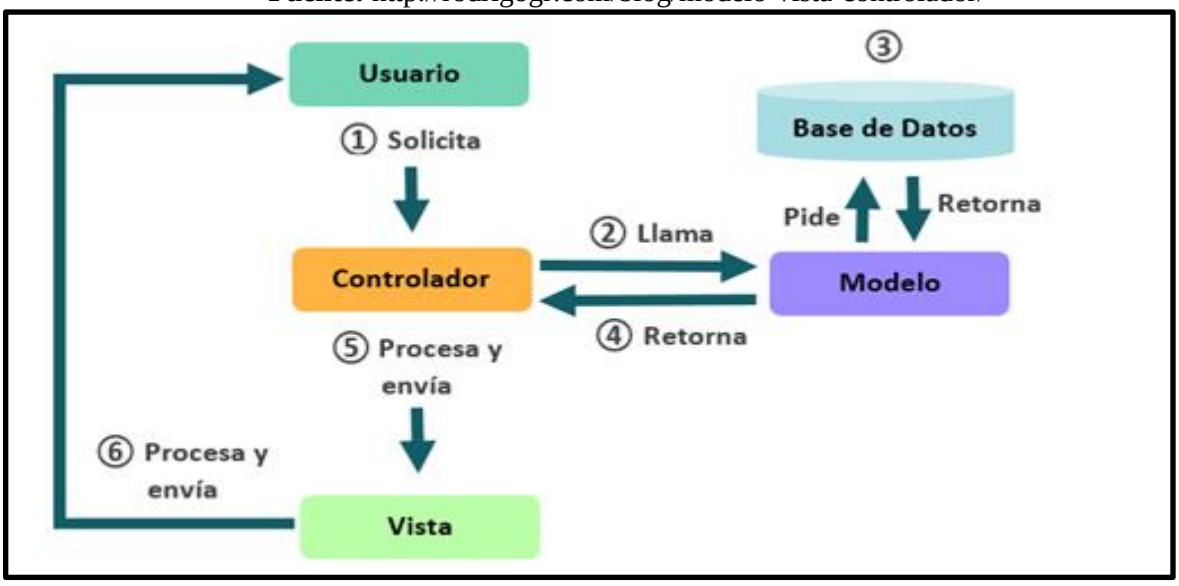
Imagen 7: Diagrama de clases
Fuente: Propia



Se identificaron 4 clases principales en el modelo presentado, así como 9 subclases que son hijas de la clase principal mascota, la cual permitirá gestionar la información y persistencia en la base de datos.

8.3.5 Arquitectura de Alto Nivel

Imagen 8: Modelo Vista Controlador
Fuente: http://rodrigogr.com/blog/modelo-vista-controlador/

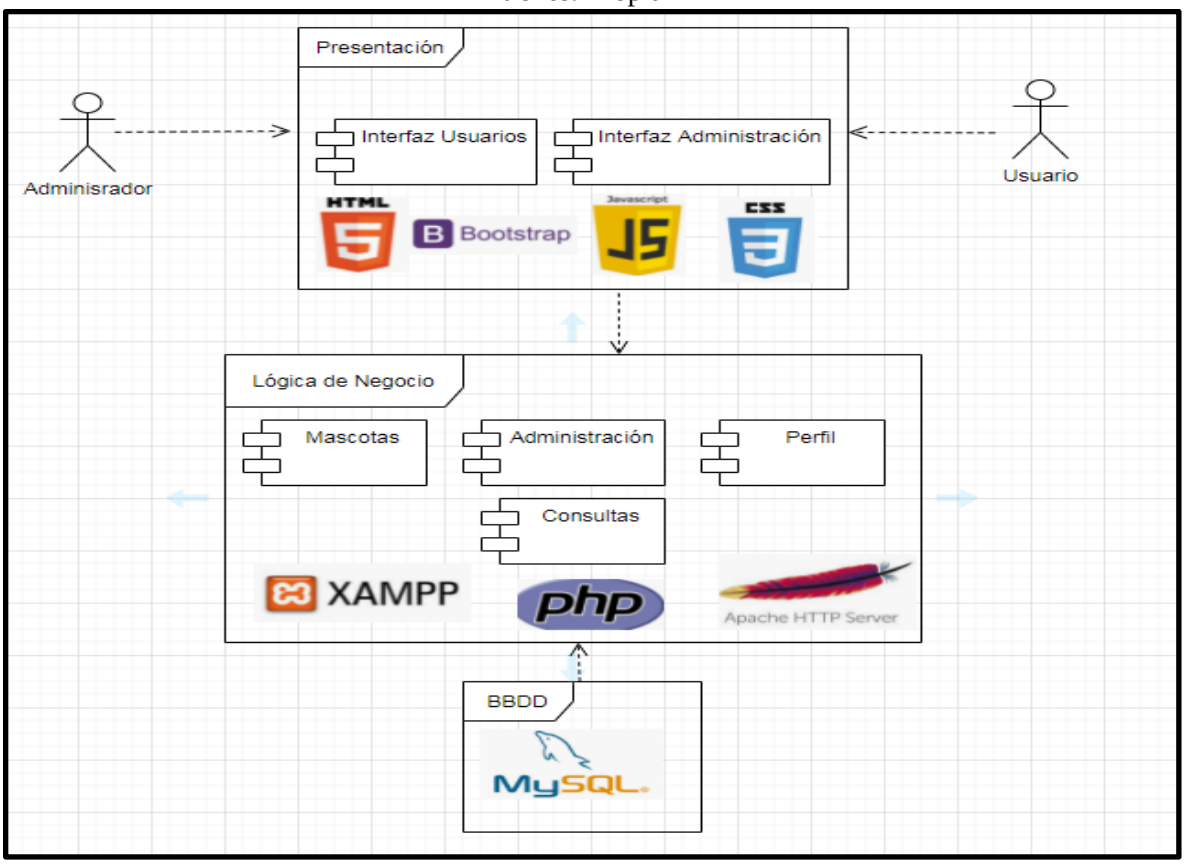


Modelo: Abstracción de los datos. Normalmente se almacenan los datos gestionados por un motor de base de datos. Cada tabla se podría representar en una clase del modelo. Si la aplicación debe obtener la información en una base de datos, el código deberá estar almacenado en el modelo.

Vista: Se encontrarán todos los elementos de la interfaz de usuario de una aplicación, esta puede contener código HTML, hojas de estilo CSS y archivos Javascript. En PHP, las vistas solo tienen etiquetas HTML y aperturas de etiquetas PHP para imprimir los datos del modelo. Las vistas podrían tener funcionalidad con JavaScript al lado del cliente para manipular y validar los datos en el dominio.

Controlador: Está en un punto intermedio entre la vista y el modelo. En el controlador se definen acciones para poder hacer un llamado al modelo correspondiente y enviarle los datos a la vista pertinente.

Imagen 9: Modelo Vista Controlador LostPet
Fuente: Propia



Para la capa de presentación al usuario final se utilizará HTML5, CSS y BootStrap para la maquetación y diseño de la página web. Java Script se utilizará para hacer validaciones sobre los campos de los formularios, así como para el llenado de listas desplegables dinámicamente.

En la lógica del negocio se utilizará el lenguaje de programación PHP para construir todas las funciones y poder interactuar con la base de datos.

Esto será administrado desde la herramienta Xampp la cual permite subir los servicios del servidor apache para poner en funcionamiento el aplicativo.

11. Construcción

Proceso de Software

Se estableció para el desarrollo del requerimiento una arquitectura por capas en donde existiría un **modelo** encargado de la abstracción de los datos, la **vista** para la capa de presentación que se mostraría al usuario y finalmente el **controlador** para poder hacer el llamado al modelo correspondiente y enviarle los datos a la vista pertinente.

Lo anterior se conoce como el patrón MVC, es un patrón arquitectural que tiene como objetivo separar la interfaz gráfica de la lógica que es donde se manejan las peticiones a la persistencia.

El usuario interactúa inicialmente con la vista para conocer el estado del modelo, pero la interacción para gestionar los datos es mediante el controlador, el cual se encarga de invocar servicios en el modelo con sus debidos atributos con datos de la persistencia, para ser actualizados en la vista correspondiente. Cada vez que el usuario requiere una acción, a través del controlador, el modelo actualiza su estado y la vista lo despliega.

Para empezar con la implementación del proyecto se diseñó el modelo Entidad Relación de lo que sería la base de datos. Para ello se utilizó MySqlWorkbench el cual permitió generar lo que sería las tablas, campos, estructura, relaciones, entre otros, y posteriormente los scripts para la creación del modelo de la BD.

Estos scripts fueron ejecutados desde la consola del Xampp y desde allí se continuó ajustando y agregando tablas o campos tan pronto el proyecto lo fuera necesitando.

Ya teniendo la base de datos estable con las relaciones correspondientes, se pasó a alimentar las tablas que estaban como llaves foráneas de las tablas maestras, esto se hizo desde la consola del Xampp (phpMyAdmin).

Teniendo lo anterior, se pasó a diseñar lo que sería el módulo de registro de usuarios, recuperación de contraseña e inicio de sesión.

A partir de esto, se fue construyendo el FrontEnd de la aplicación con HTML, CSS y BootStrap, dándole un diseño llamativo, intuitivo y fácil de utilizar.

Ya teniendo estos módulos construidos, se pasó a diseñar lo que sería el menú del aplicativo y la estructura que este llevaría (header y footer); de esta manera se estableció una plantilla para ser utilizada en todas las páginas.

Luego de esto, se pasaron a construir todas las funcionalidades correspondientes al perfil de administrador y aquí se empezó a trabajar el tema de los roles, los cuales ocultaban o mostrarían las opciones dependiendo del tipo de usuario que se estaría logueando en el aplicativo.

Teniendo estas opciones construidas y diseñadas, se pasó a desarrollar todas las opciones correspondientes a los usuarios que no fuesen administradores del sistema. Aquí se implementó el módulo de mascotas, el cual permitiría registrar, modificar, generar un cartel en formato pdf con el respectivo código QR de las mascotas de cada usuario. Posterior a esto, se implementó el módulo de consultas, el cual dejaría aplicar diferentes filtros para la búsqueda de mascotas y visualizar las publicaciones aprobadas por el usuario administrador.

También se implementó el manejo de sesiones, como la expiración de las sesiones a los 10 minutos de inactividad, la destrucción de la sesión cuando realiza logout y validar que para poder visitar una página el usuario tiene que haberse logueado.

Por último, se diseñó una sección de Contacto, en donde el usuario puede enviar un mensaje al administrador del sistema para reportar cualquier información requerida. Esto se le mostraría al administrador tan pronto ingrese al aplicativo.

Luego de realizar pruebas y correcciones a lo encontrado, se pasó a realizar la refactorización del código, corrigiendo nombres de variables, eliminado código no utilizado y simplificando funciones. De igual manera, se ajustó el desarrollo a la arquitectura por capas para una mejor presentación y organización del código.

Se utilizó GitHub para el versionamiento del código, cada día que se tenía un desarrollo se iban subiendo los cambios al repositorio.

Para la implementación del aplicativo web y sus funcionalidades se utilizaron las siguientes tecnologías:

Tabla 17 – Tecnologías Utilizadas para el desarrollo del proyecto
Fuente: Propia

Tecnología	Descripción
	Diseño y estructura del sitio web.
	Diseño visual de los documentos web e interfaces de usuario escritas en HTML.
	Conjunto de herramientas de código abierto para diseño de sitios y aplicaciones web.
	Lenguaje de programación utilizado al lado del cliente como en el lado del servidor que permite hacer que las páginas web sean interactivas.
	Lenguaje de programación para la construcción de las funcionalidades y conexión con la base de datos.

	Motor de base de datos para el almacenamiento de la información.
Tecnología	**Descripción**
	Servidor web
	Sistema de gestión de bases de datos MySQL, el servidor web Apache y los intérpretes para lenguajes de script PHP
	Diseño de bases de datos que integra desarrollo de software, administración de bases de datos, diseño de bases de datos, gestión y mantenimiento para el sistema de base de datos MySQL

12. Pruebas

Las pruebas que se ejecutaron a continuación se validaron en un ambiente de pruebas que contaba con las siguientes especificaciones técnicas:

Servidor: Apache versión 2.4
Sistema Operativo: Windows 10
Memoria RAM: 8GB
Procesador: Intel Core 3.0 GHz
Motor Base de Datos: MySQL 10.4.17-MariaDB
Java: 11.0
Jmeter: 5.4.1
Selenium: 3.1

Pruebas Funcionales**:**

Para la ejecución de las pruebas correspondientes al desarrollo del aplicativo LostPet, basados en las historias de usuario se diseñaron los siguientes casos de prueba:

Imagen 10: Gestión y avance de casos de prueba
Fuente: Propia

GESTION Y AVANCE DE CASOS DE PRUEBA								
Funcionalidad	**Descripción**	**Casos Planeados**	**Casos Exitosos**	**Casos Fallidos**	**Casos No aplica**	**Casos Pendientes**	**Casos Dependientes por fallos**	**Total Probados**
Sprint 1	Registro Usuario-IniciarSesion-Recuperar Contraseña	3	3	0	0	0	0	3
Sprint 2	Actualizar Perfil Usuario-Gestionar Usuarios-Visualizar A	3	3	0	0	0	0	3
Sprint 3	Gestionar Razas-Reportar Mascota-Visualizar Mis masc	3	3	0	0	0	0	3
Sprint 4	Generar Carteles - Reportar Información Mascota	2	2	0	0	0	0	2
Sprint 5	Generar Reportes-Aprobar Publicacion-Visualizar Public -Validar Sección Contactenos	4	4	0	0	0	0	4
	Totales	**15**	**15**	**0**	**0**	**0**	**0**	**15**
	% Casos Probados							100,00%
	% Casos Reproceso							0,00%
	% Avance Real							**100,00%**

Imagen 11: Gestión y avance de pasos de prueba
Fuente: Propia

GESTION DE PASOS DE PRUEBA					
Funcionalidad	Pasos Planeados	Pasos Exitosos	Pasos Fallidos	Pasos No Aplica	Pasos Pendientes
Sprint 1	17	17	0	0	0
Sprint 2	16	16	0	0	0
Sprint 3	17	17	0	0	0
Sprint 4	10	10	0	0	0
Sprint 5	17	17	0	0	0
Totales (Pasos)	77	77	0	0	0
% Pasos Probados					100,00%
% Pasos Pendientes					0,00%

En total se diseñaron 15 casos de pruebas, que contienen 77 pasos de prueba los cuales abarcan el requerimiento completo.

El control y seguimiento de las pruebas serán llevadas en un archivo en Excel, el cual mostrará el avance de la ejecución de las pruebas y el estado de cada una de ellas.

Las evidencias correspondientes a la ejecución de las pruebas se anexan a este documento.

No Funcionales**:**

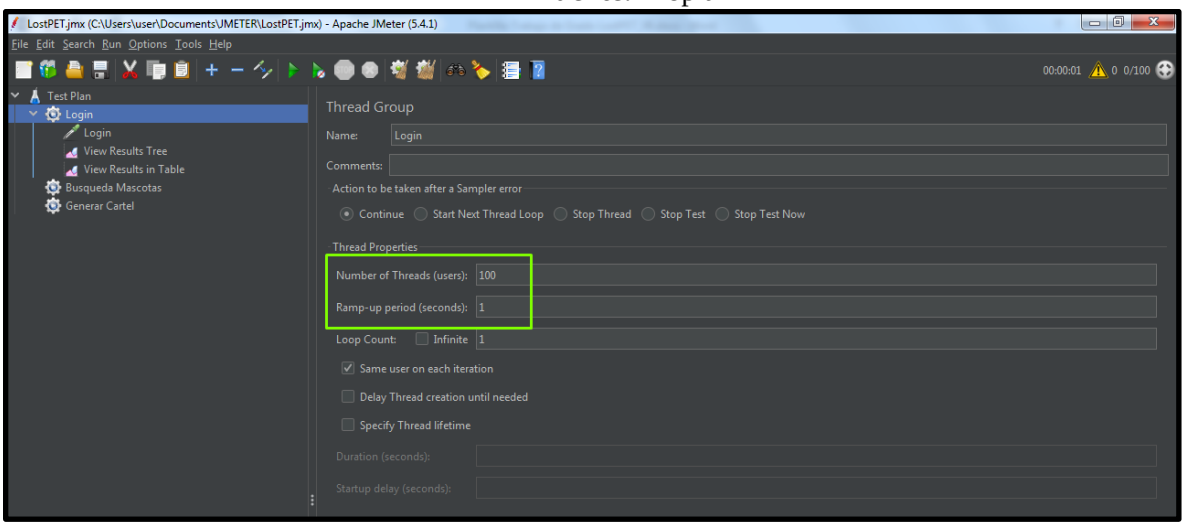
Herramienta Utilizada: JMETER

Se simuló la iteración de 100 usuarios tratando de ejecutar las siguientes acciones al mismo tiempo en un periodo de tiempo de 1 segundo, arrojando los siguientes resultados exitosos:

Inicio de sesión en el aplicativo.

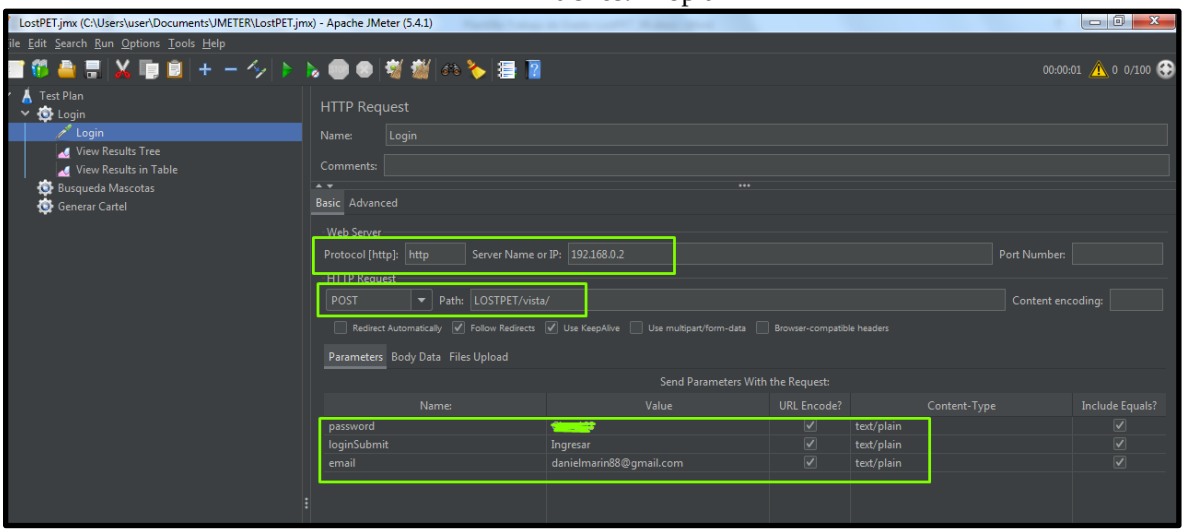
Se realizó configuración de la simulación de 100 usuarios tratando de realizar el inicio de sesión al mismo tiempo en un rango de 1 segundo:

Imagen 12: Configuración usuarios
Fuente: Propia



Se configuró el Request a enviar para procesar las solicitudes:

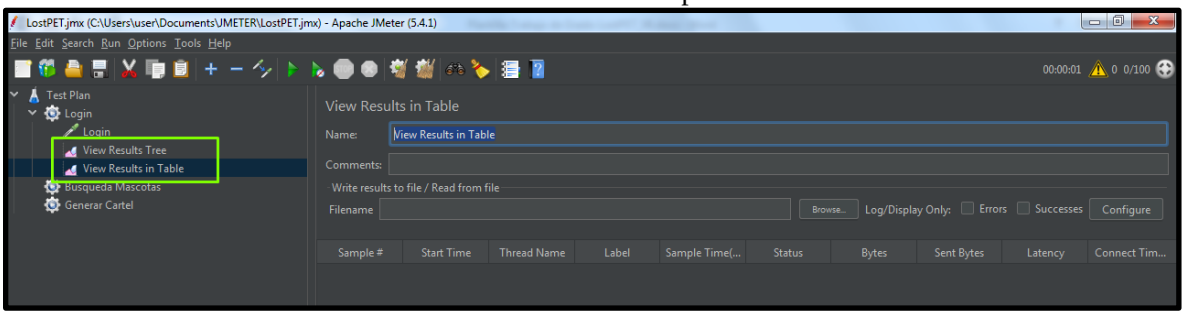
Imagen 13: Configuración datos login
Fuente: Propia



Se configuran los Listener para visualizar el resultado de la ejecución:

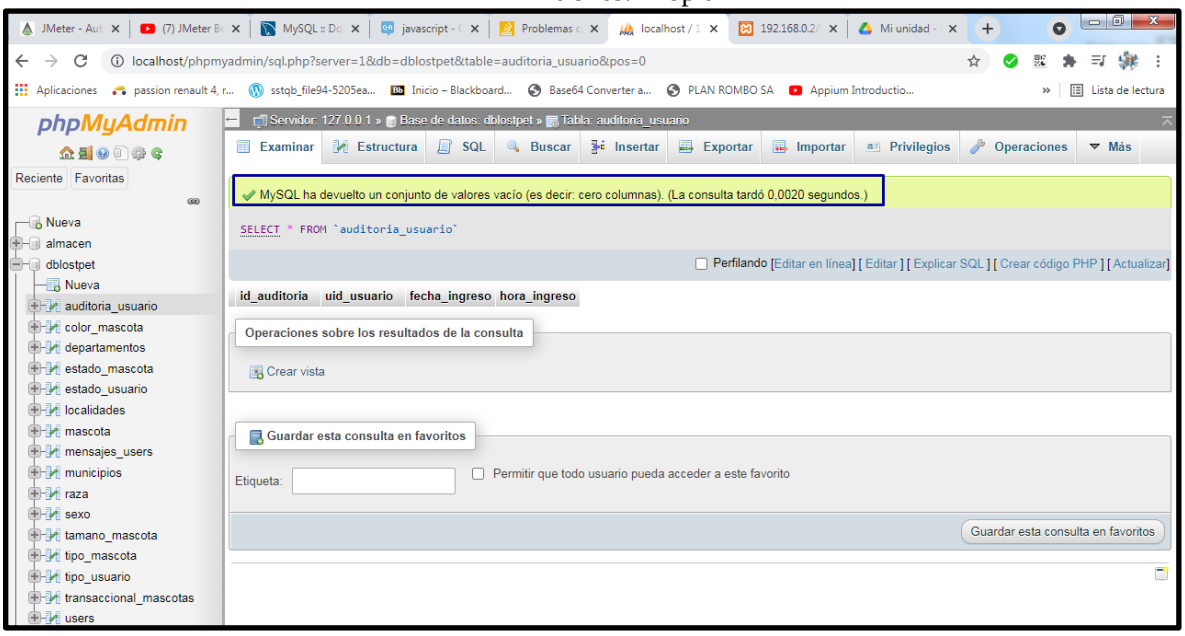
View Results Tree y View Results in Table

Imagen 14: Configuración escuchadores
Fuente: Propia



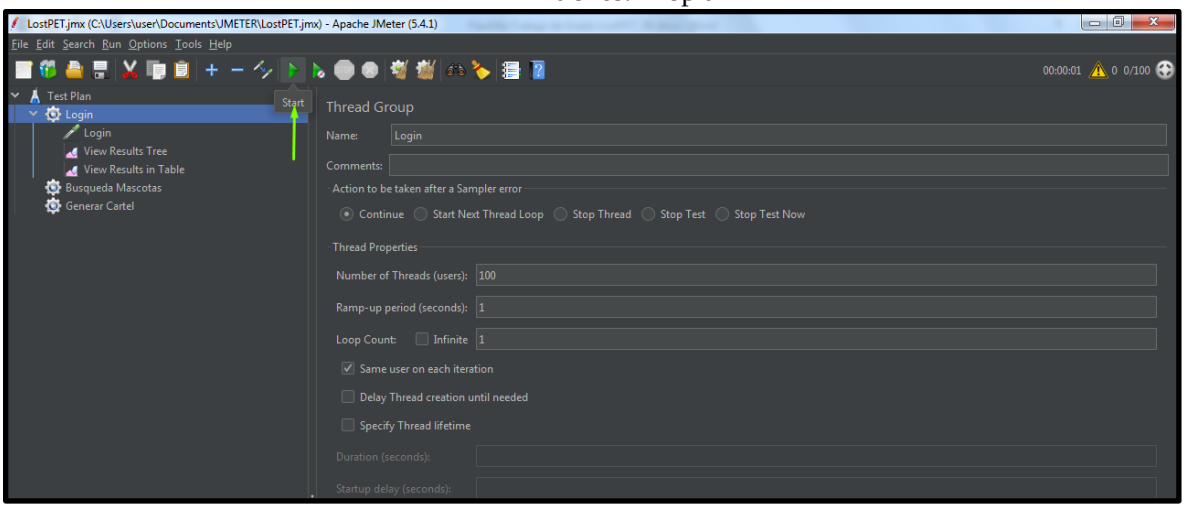
Antes de lanzar la ejecución se valida en la auditoria de la base de datos que esté vacía:

Imagen 15: Auditorio inicio de sesión usuarios
Fuente: Propia



Se ejecuta la prueba de rendimiento:

Imagen 16: Ejecución Prueba de desempeño
Fuente: Propia



Se visualizan los resultados:

Imagen 17: Resultados Ejecución Prueba
Fuente: Propia

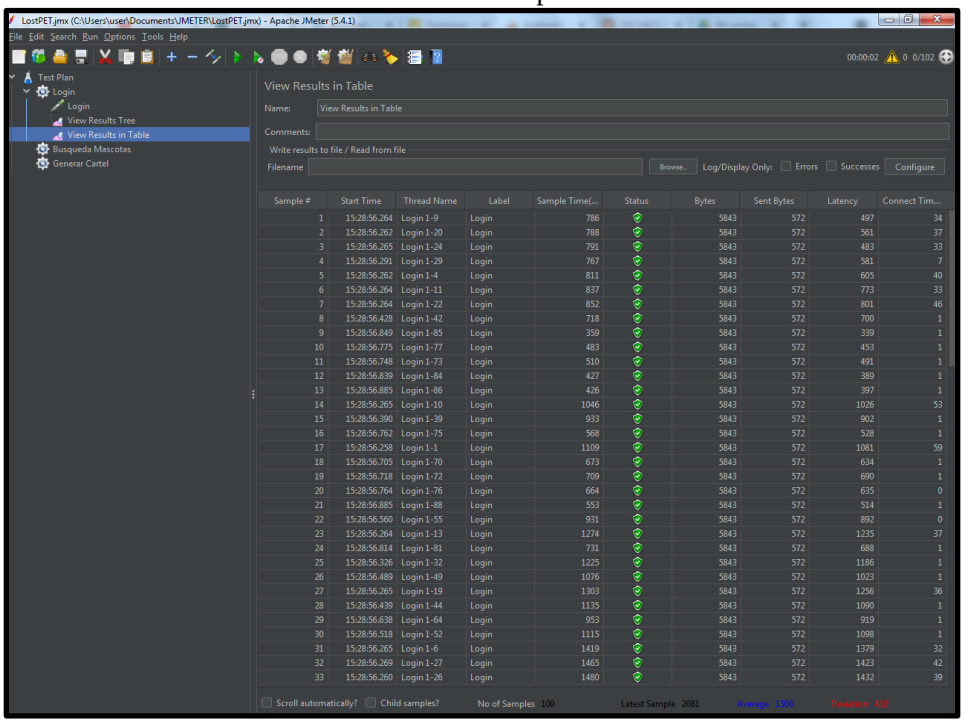


Imagen 18: Resultados Ejecución Prueba
Fuente: Propia

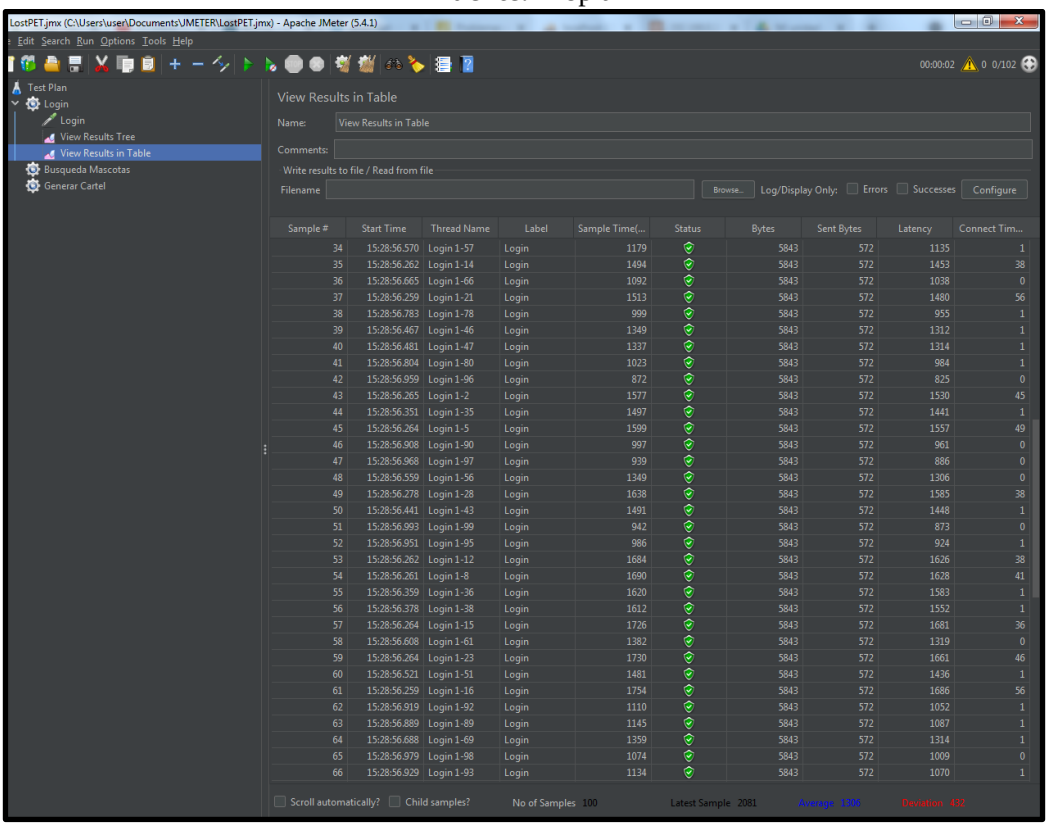


Imagen 19: Resultados Ejecución Prueba
Fuente: Propia

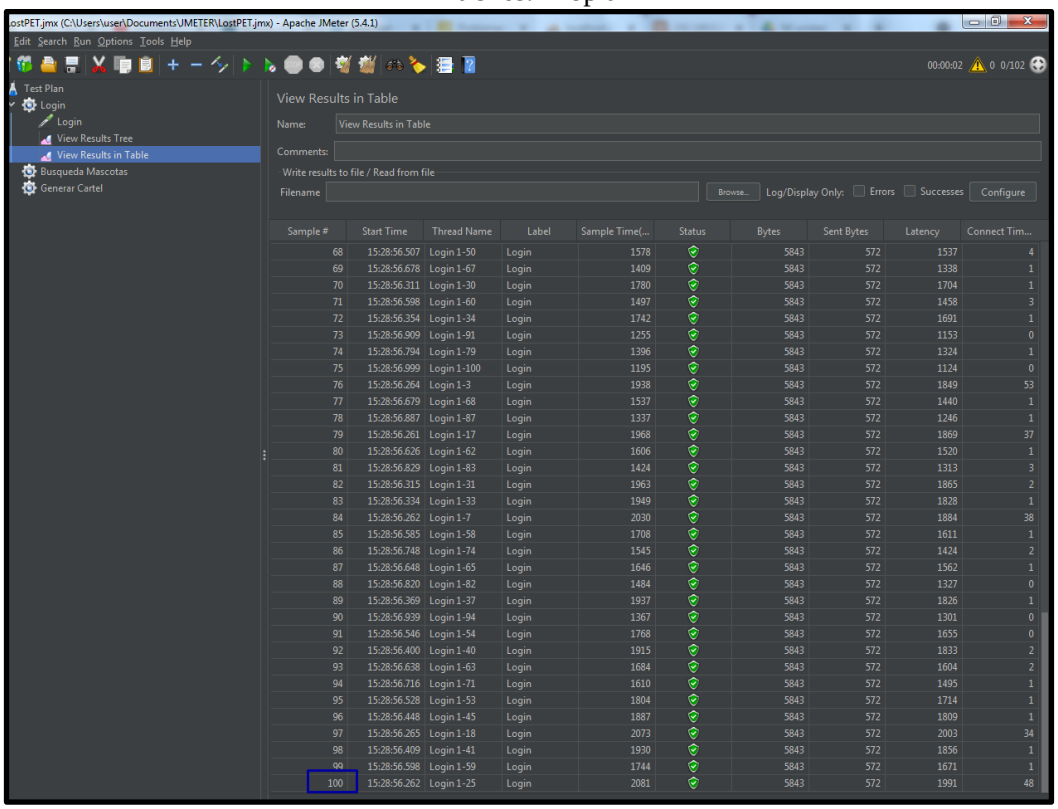
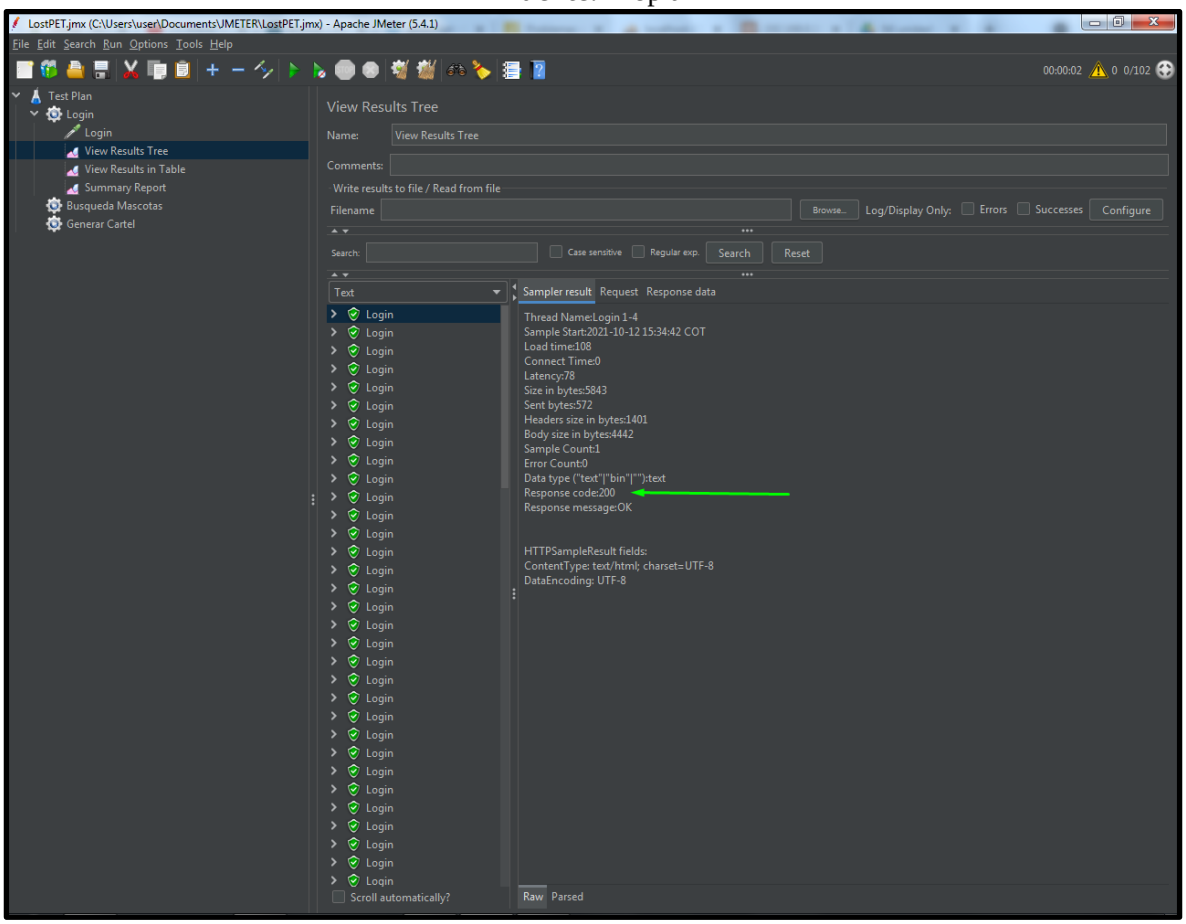
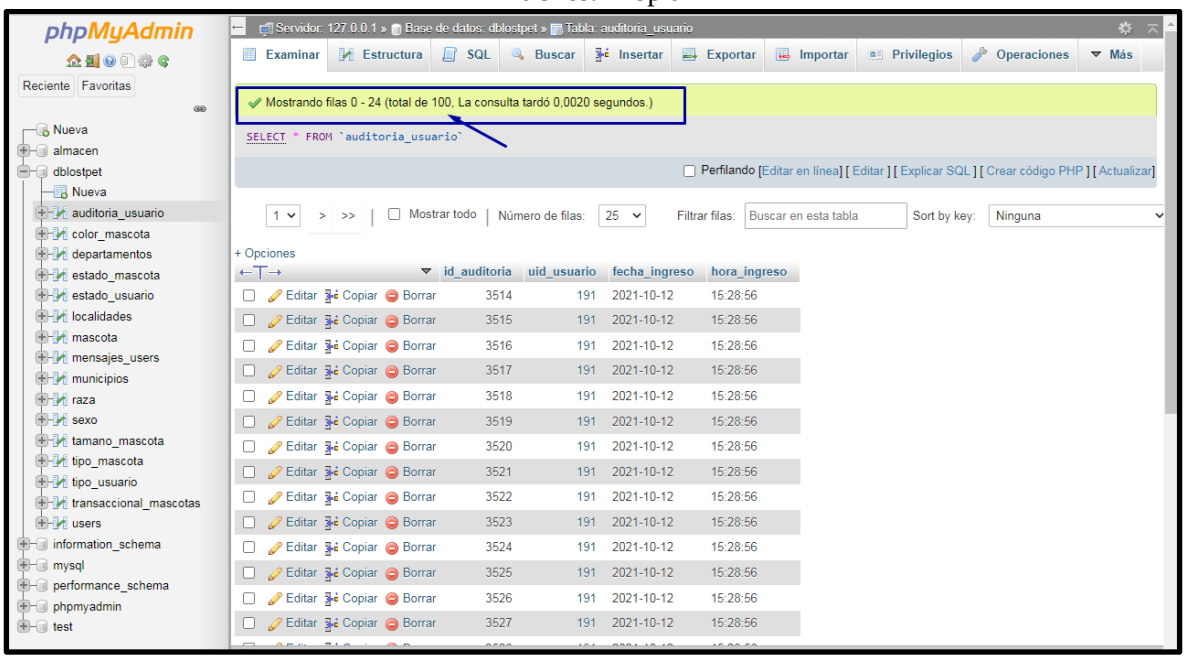


Imagen 20: Resultados Ejecución Prueba
Fuente: Propia



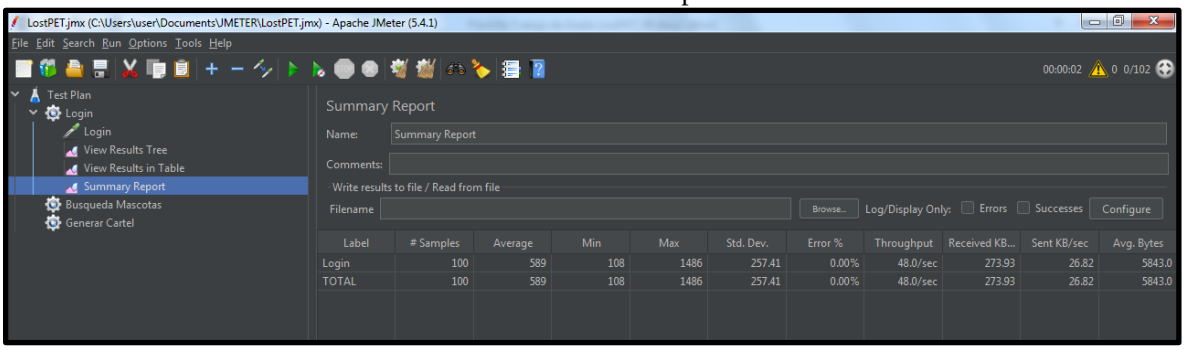
Se consultó nuevamente en la base de datos para validar que efectivamente se simuló el ingreso de los 100 usuarios:

Imagen 21: Resultados Auditoría BD
Fuente: Propia



Finalmente se puede visualizar el resumen de la prueba:

Imagen 22: Resultados Resumen Prueba
Fuente: Propia

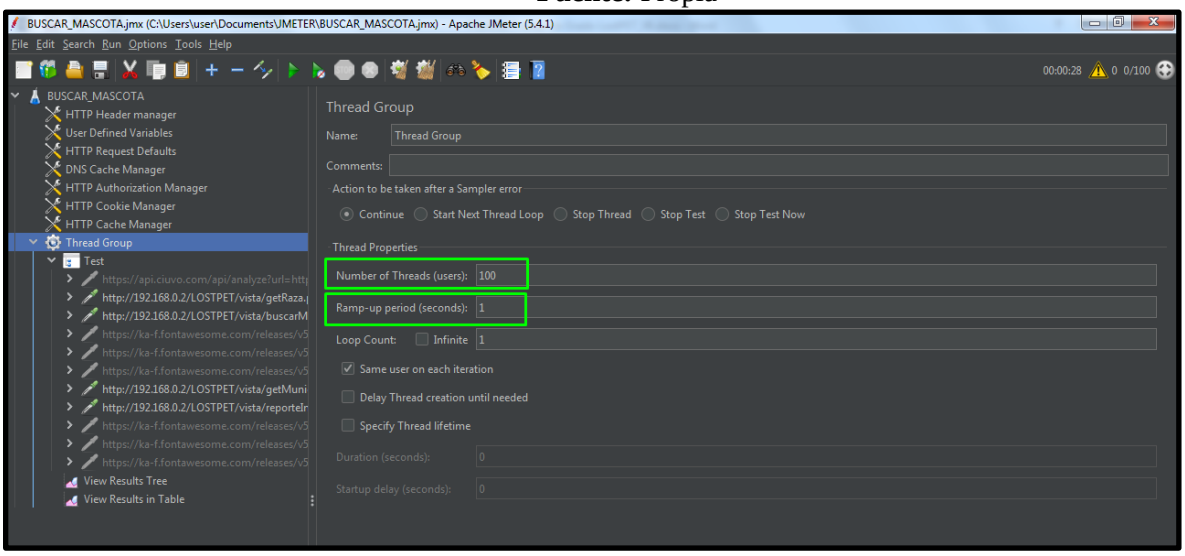


Teniendo en cuenta los resultados arrojados, se pudo evidenciar que los 100 usuarios pudieron iniciar sesión exitosamente en un promedio de 589 Milisegundos, lo cual se concluye que el aplicativo permite trabajar concurrentemente con esta cantidad de usuarios sin presentar lentitud.

Búsqueda de Mascotas (Desde el Front)

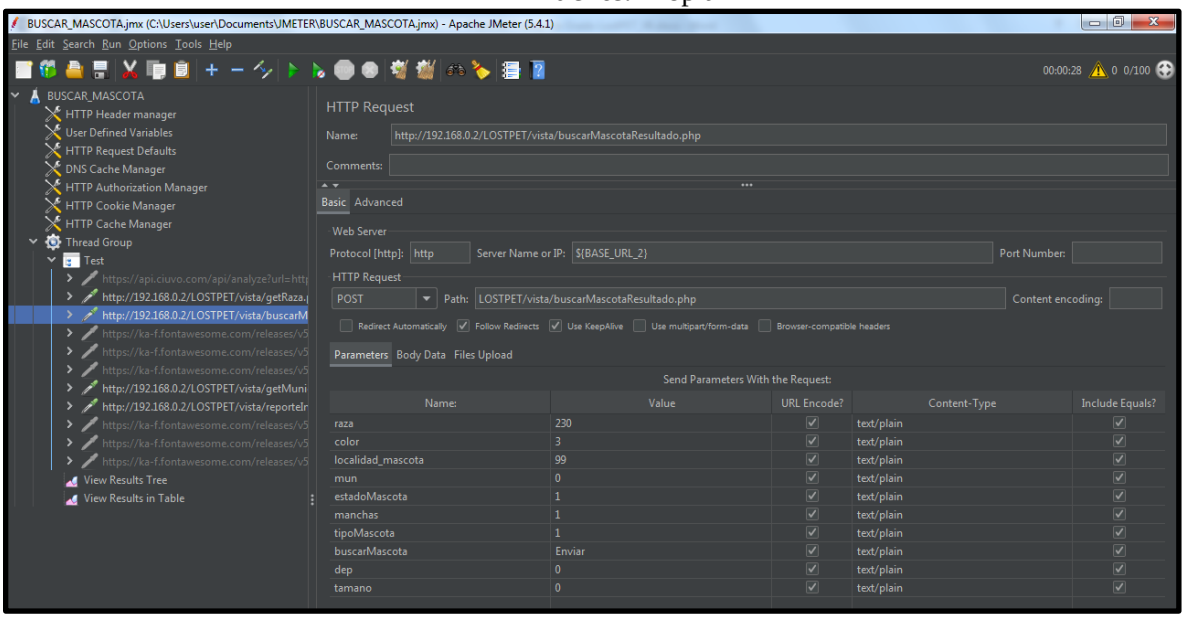
Se realizó configuración de la simulación de 100 usuarios tratando de realizar la búsqueda de mascotas al mismo tiempo en un rango de 1 segundo:

Imagen 23: Configuración Usuarios
Fuente: Propia



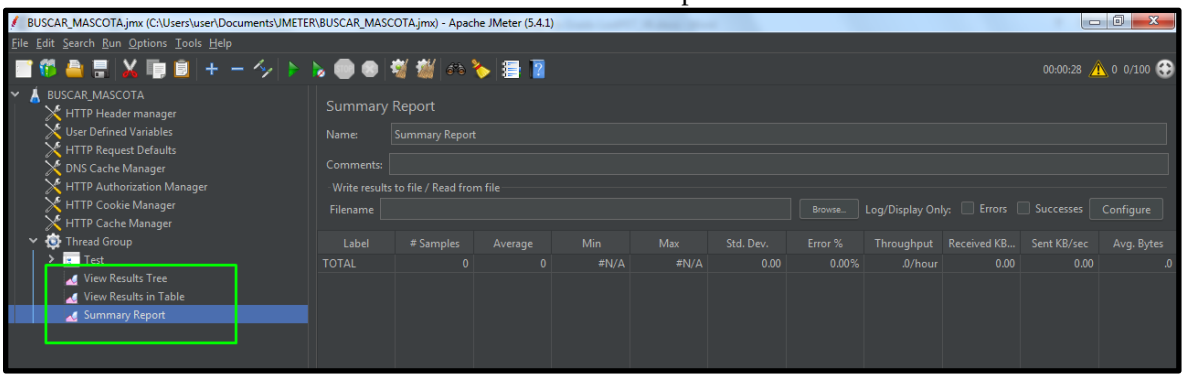
Se configuró el request a enviar para procesar las solicitudes:

Imagen 24: Configuración Request
Fuente: Propia



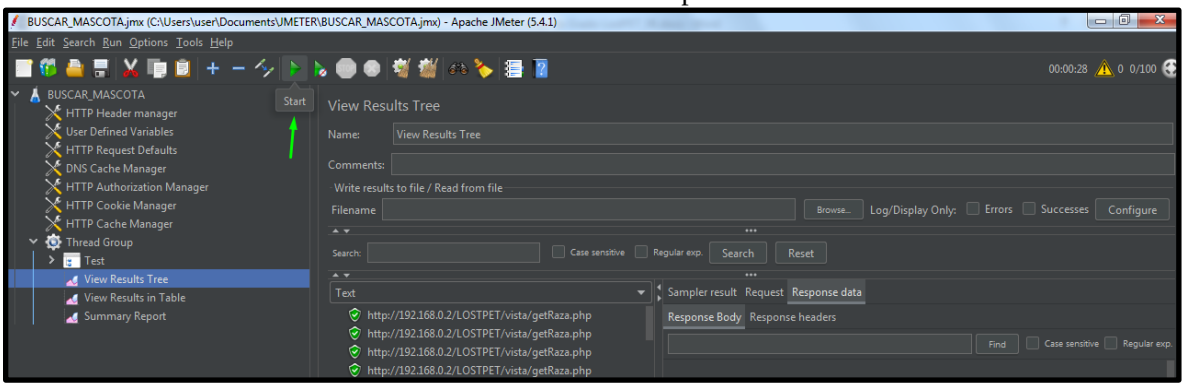
Se configuran los Listener para visualizar el resultado de la ejecución:

Imagen 25: Resultados
Fuente: Propia



Se ejecuta la prueba de rendimiento:

Imagen 26: Resultados
Fuente: Propia



Se visualizan los resultados:

Imagen 27: Resultados
Fuente: Propia

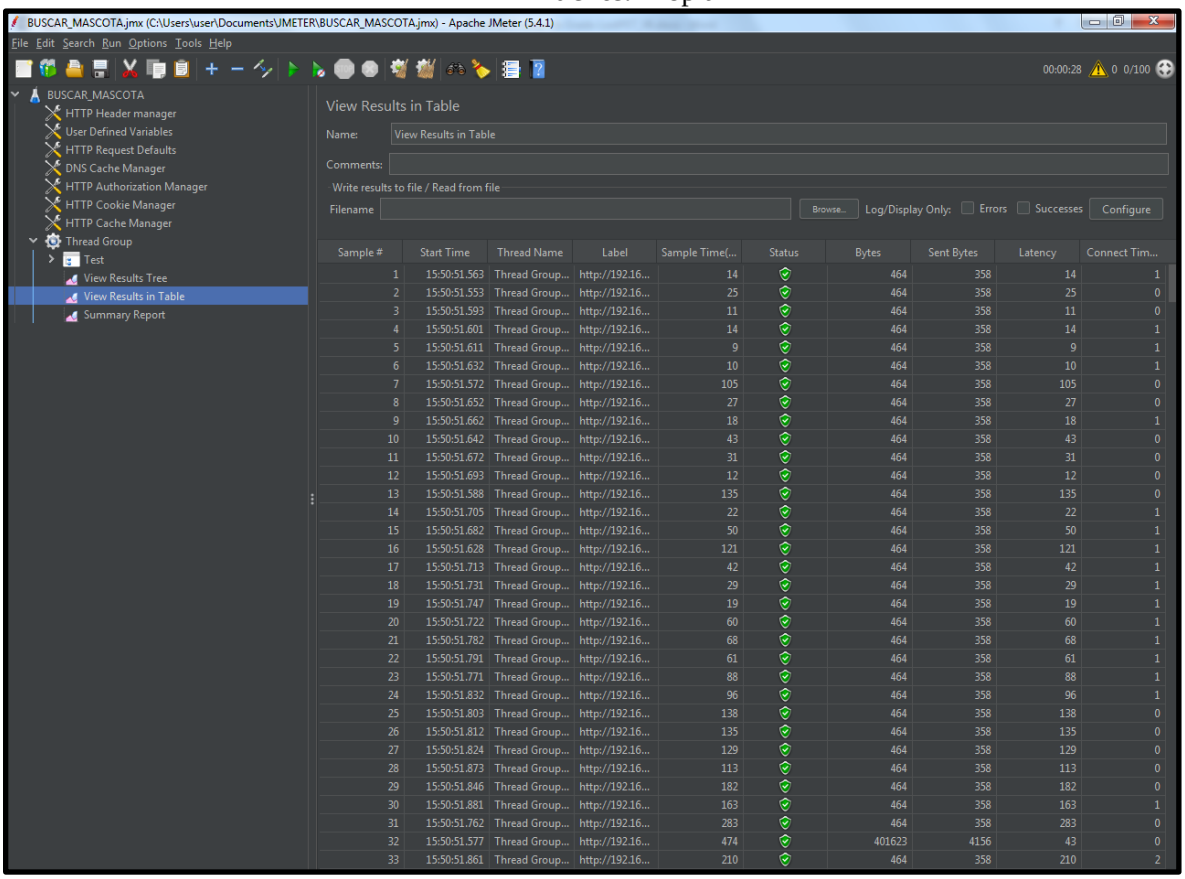
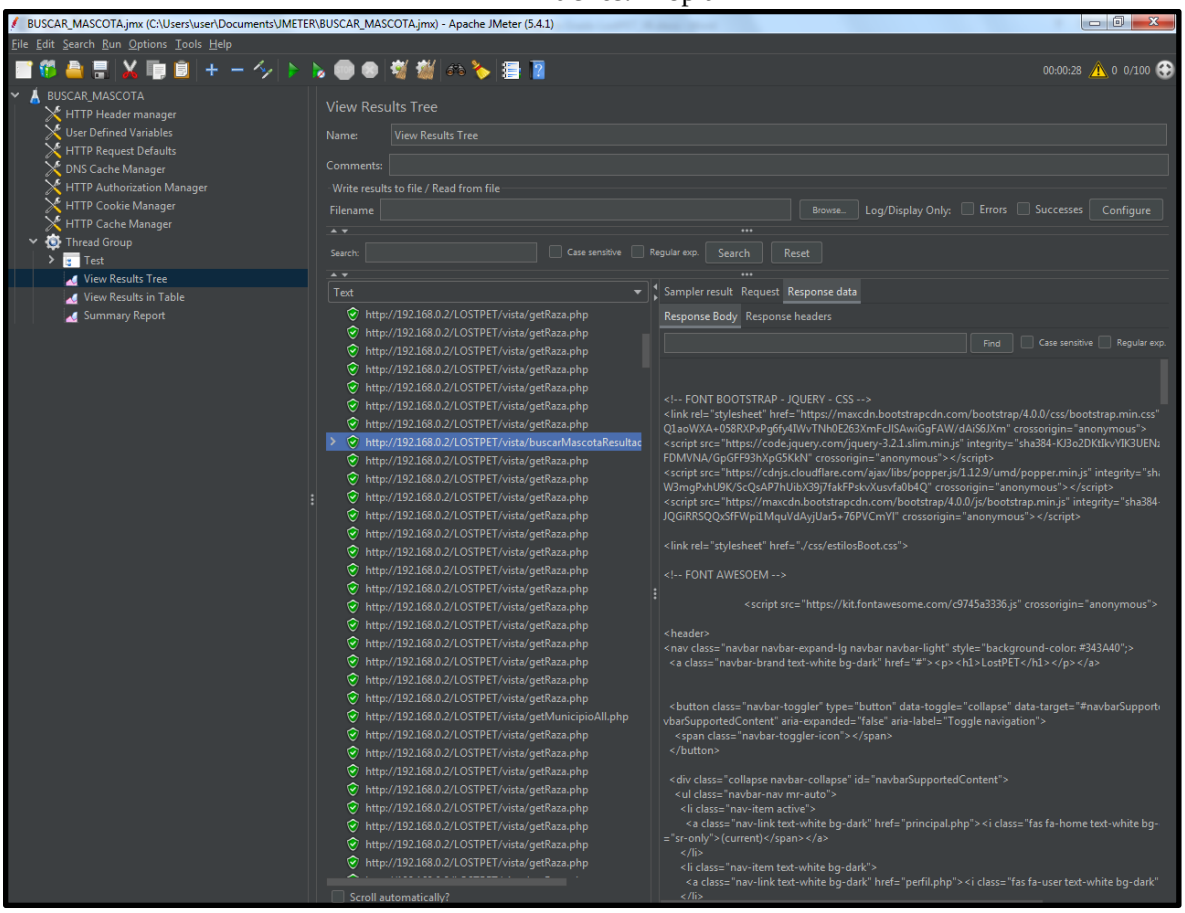
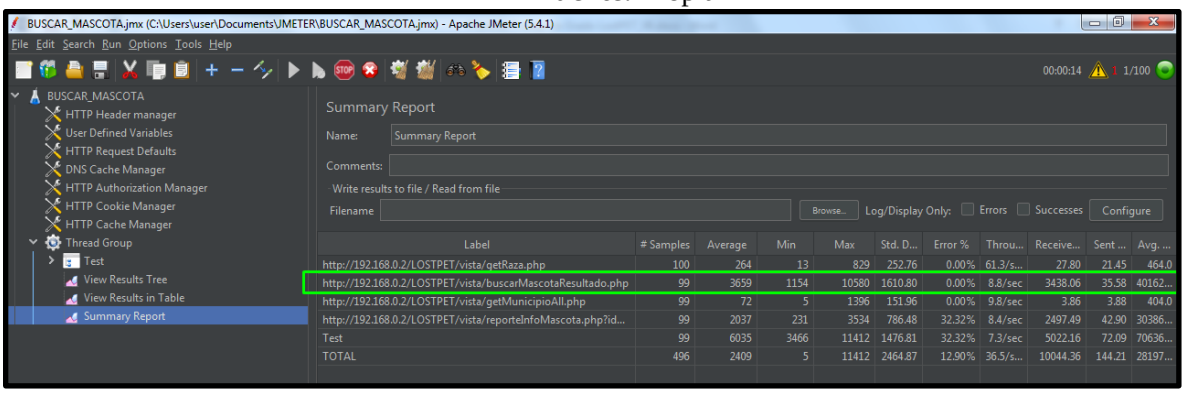


Imagen 28: Resultados
Fuente: Propia



Finalmente se puede visualizar el resumen de la prueba:

Imagen 29: Resuman Resultados
Fuente: Propia



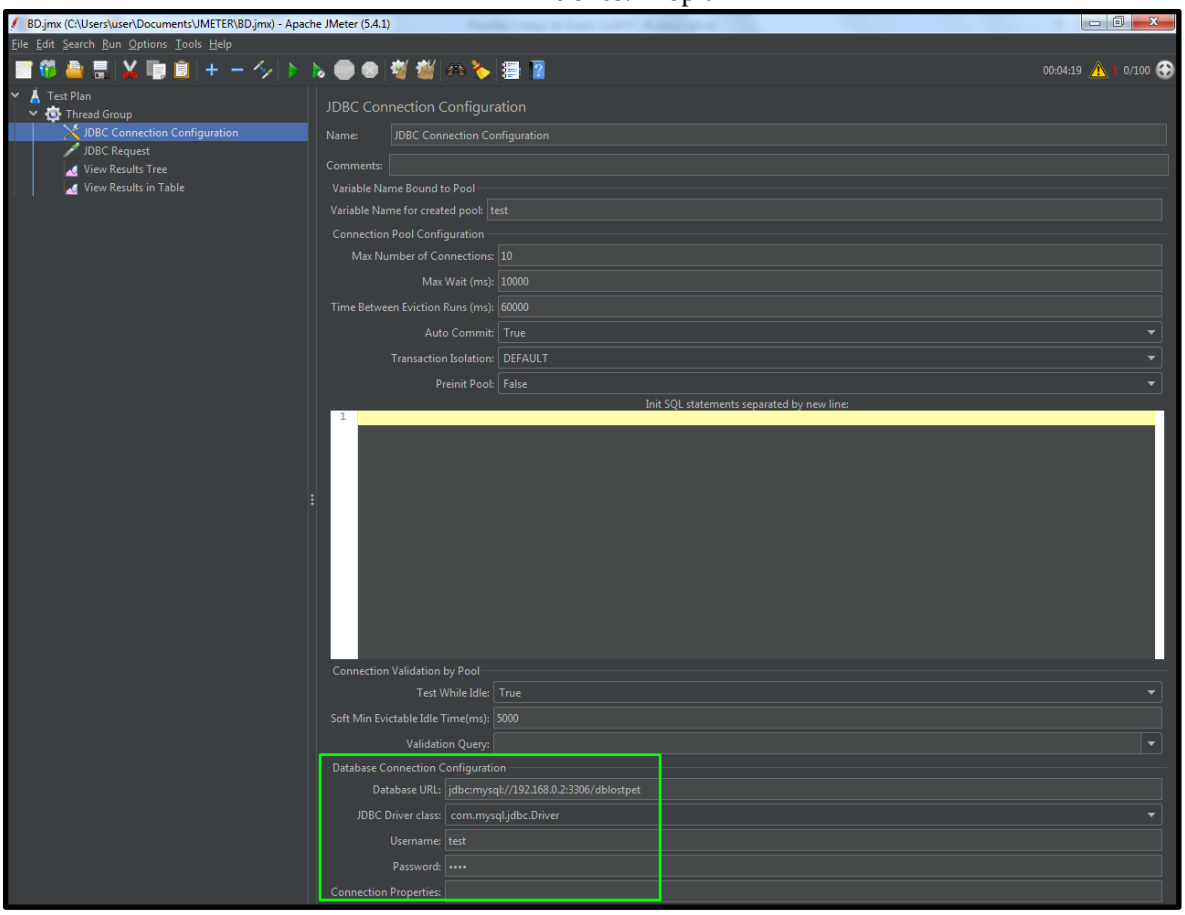
Teniendo en cuenta los resultados arrojados, se pudo evidenciar que los 100 usuarios pudieron consultar las mascotas registradas en un promedio de 3.6 Segundos, lo cual se concluye que el aplicativo permite trabajar concurrentemente con esta cantidad de usuarios sin presentar lentitud.

Búsqueda de Mascotas (Desde la BD)

De igual manera se evaluó la rapidez con lo cual la base de datos tarda en mostrar los resultados y a continuación se muestra la respuesta:

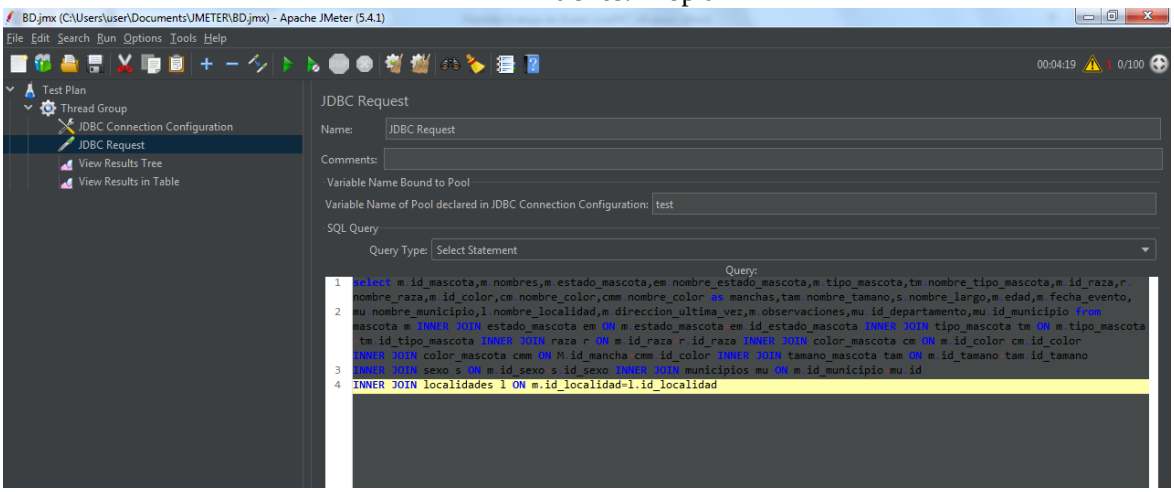
Se realiza la configuración para la conexión con la base de datos desde JMETER:

Imagen 30: Configuración BD
Fuente: Propia



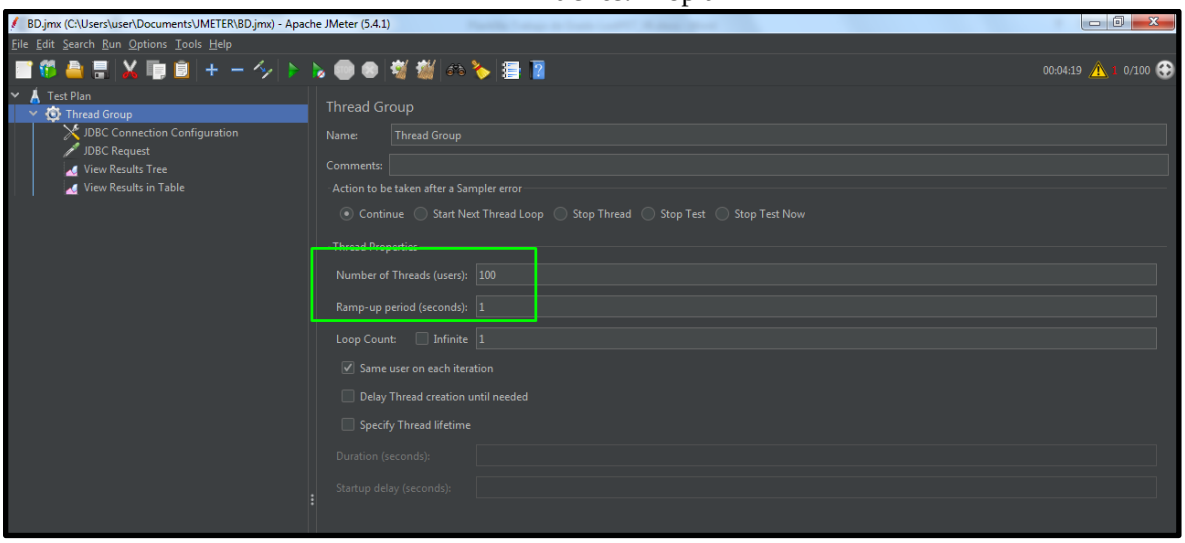
Se arma el request que será enviado a ejecutar:

Imagen 31: Consulta para la prueba
Fuente: Propia



Se configura el número de usuarios (100) y el periodo de tiempo en el cual se ejecutará la consulta (1 Segundo):

Imagen 32: Configuración Usuarios
Fuente: Propia



Se ejecuta la consulta y se visualizan los resultados:

Imagen 33: Visualización Resultados
Fuente: Propia

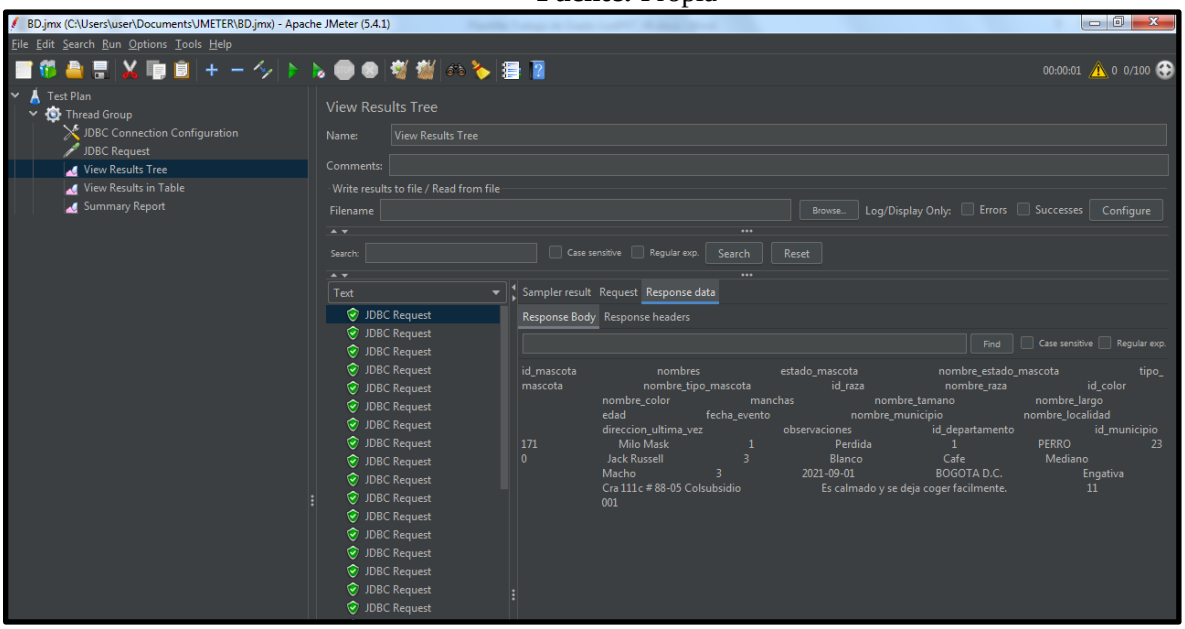
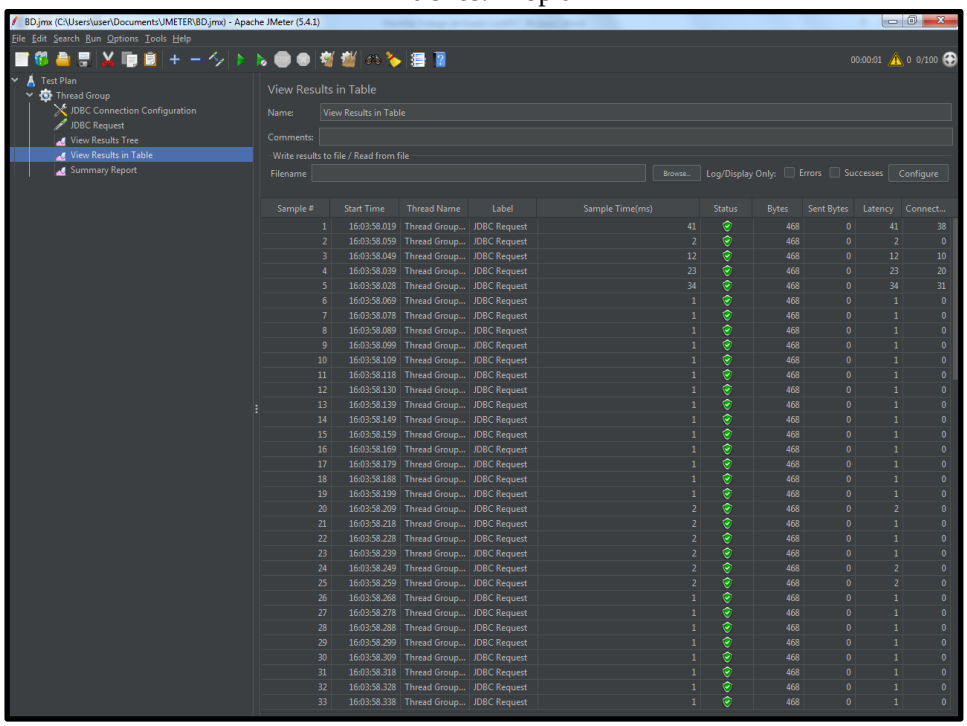
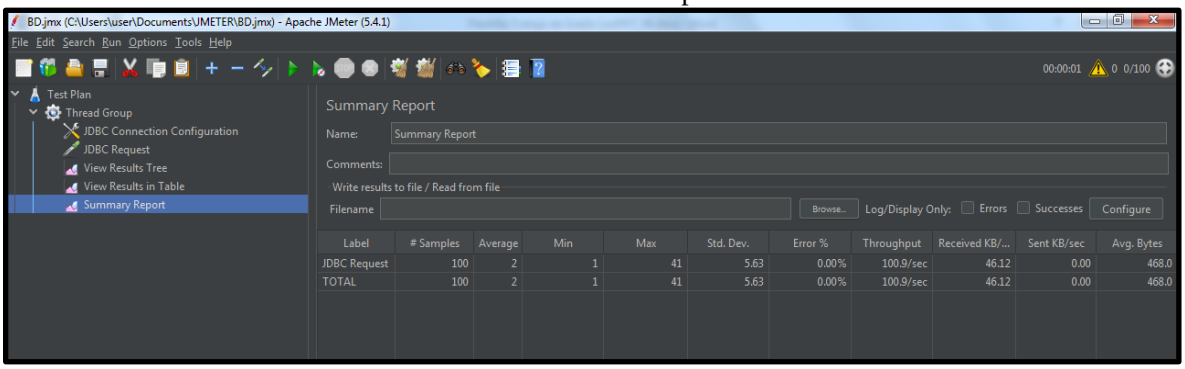


Imagen 34: Visualización Resultados
Fuente: Propia



Finalmente se puede visualizar el resumen de la prueba:

Imagen 35: Resumen de la prueba
Fuente: Propia



Teniendo en cuenta los resultados arrojados, se pudo simular el hecho que 100 usuarios pudieron ejecutar el SELECT SQL correspondiente a la consulta de las mascotas registradas en un promedio de 2 Milisegundos, lo cual se concluye que la base de datos permite trabajar concurrentemente con esta cantidad de usuarios sin presentar lentitud.

Automatizadas**:**

Se tomó un flujo exitoso del aplicativo que abarca el 70% de las funcionalidades, las cuales serán ejecutadas de forma automática utilizando las siguientes tecnologías:

Tabla 18 – Tecnologías Automatización de Pruebas
Fuente: Propia

Tecnología	Descripción
 **Java 11**	Lenguaje de Programación
 **Selenium Web Driver**	Controlar y programar las acciones con el navegador
	Permite que los comportamientos de software esperados se especifiquen en un lenguaje lógico que los clientes puedan entender.
	Gestión, construcción y ejecución del proyecto.

Las funcionalidades que fueron automatizadas se detallan a continuación:

- Crear una Raza
- Reportar una mascota
- Visualizar una mascota
- Visualizar cartel pdf creado
- Aprobar Publicación
- Visualizar Publicación
- Reportar Información de la mascota
- Visualizar y actualizar Notificaciones

Utilizando la metodología BDD y la herramienta Cucumber se crearon los Features de cada una de las funcionalidades a automatizar:

Imagen 36: Feature Crear Raza
Fuente: Propia

1-CrearRaza.feature

```
1  Feature: Crear Raza
2
3    Scenario: Creacion de raza exitosa
4      Given Se inicia sesion como administrador
5      And Se ingresa al modulo de crear raza
6      When Se diligencian los campos del formulario CR
7      And Se pulsa el boton Crear Raza
8      Then El sistema crea la raza exitosamente
9
```

Imagen 37: Feature Reportar Mascota
Fuente: Propia

2-ReportarMascota.feature

```
1  Feature: Reportar Mascota
2
3    Scenario: Creacion Exitosa de Mascota
4      Given Se inicia sesion como usuario
5      And Se ingresa al modulo de Reportar Mascota
6      When Diligencia los campos del formulario RM
7      And Se Pulsa el boton Reportar Mascota
8      Then Se crea la mascota exitosamente
9
10     Scenario: Visualizar Imagen
11     Given Se ingresa al modulo Mis Mascotas
12     When se pulsa el icono para visualizar imagen
13     Then Se carga una pestana nueva y se visualiza la imagen
14
```

Imagen 38: Feature Visualizar Cartel Mascota
Fuente: Propia

3-VisualizarCartelMascota.feature

```
Feature: Visualizar Cartel Mascota

  Scenario: Visualizacion Exitosa Cartel Mascota
    Given Se inicia sesion como usuario VCM
    And Se ingresa al modulo de Mis Mascotas
    When Se pulsa el icono Generar Cartel
    Then Se muestra el cartel correctamente.
```

Imagen 39: Feature Modificar Mascota
Fuente: Propia

4-ModificarMascota.feature

```
Feature: Modificar Mascota

  Scenario: Modificar Mascota Exitosamente
    Given Se inicia sesion en el aplicativo como usuario MM
    When Se ingresa al modulo de Mis Mascotas MM
    And Se pulsa sobre el icono para editar la mascota
    And Se modifican los campos
    And Se pulsa el boton Modificar Mascota
    Then Se modifica la mascota exitosamente

```

Imagen 40: Feature Reportar Información Mascota
Fuente: Propia

5-ReportarInfoMascota.feature

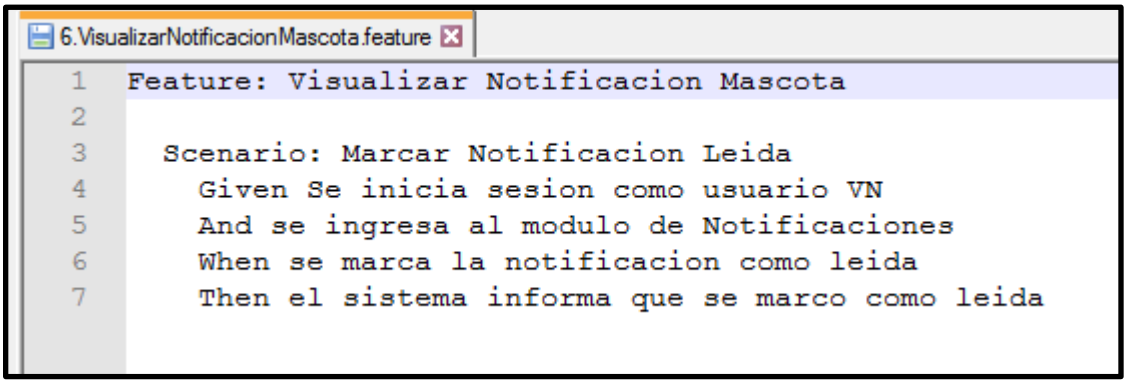
```
Feature: Reportar Informacion mascota

  Scenario: Buscar Mascota exitosamente
    Given Se inicia sesion como usuario RIM
    And se ingresa al modulo de Buscar Mascota
    When se eligen los filtros de busqueda
    And Se pulsa el boton Buscar Mascota
    Then el sistema muestra el resultado exitosamente

  Scenario: Reportar Informacion Mascota
    Given Se pulsa sobre el icono de Registrar Novedad
    And se diligencian los campos del formulario
    When se pulsa el boton Enviar informacion
    Then El sistema envia la notificacion exitosamente

```

Imagen 41: Feature Visualizar Notificación Mascota
Fuente: Propia



Luego de esto, se generaron las clases en java en donde se integraría con selenium para automatizar las acciones a ejecutar en el navegador:

Imagen 42: Clase TestLoginAdministrador
Fuente: Propia

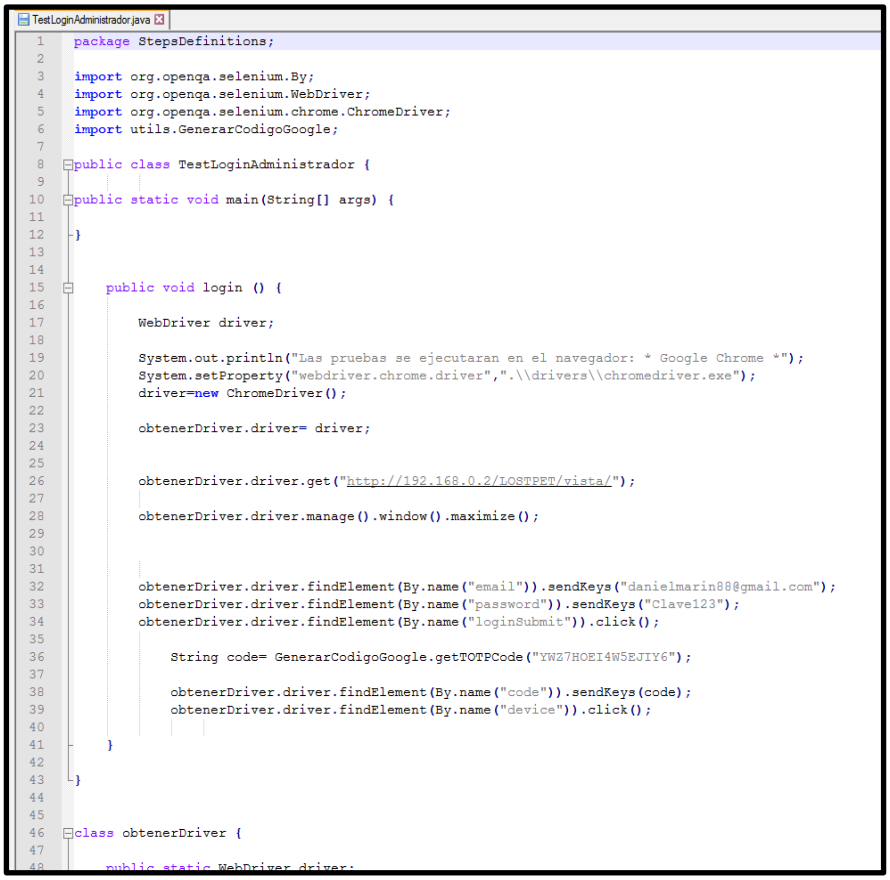


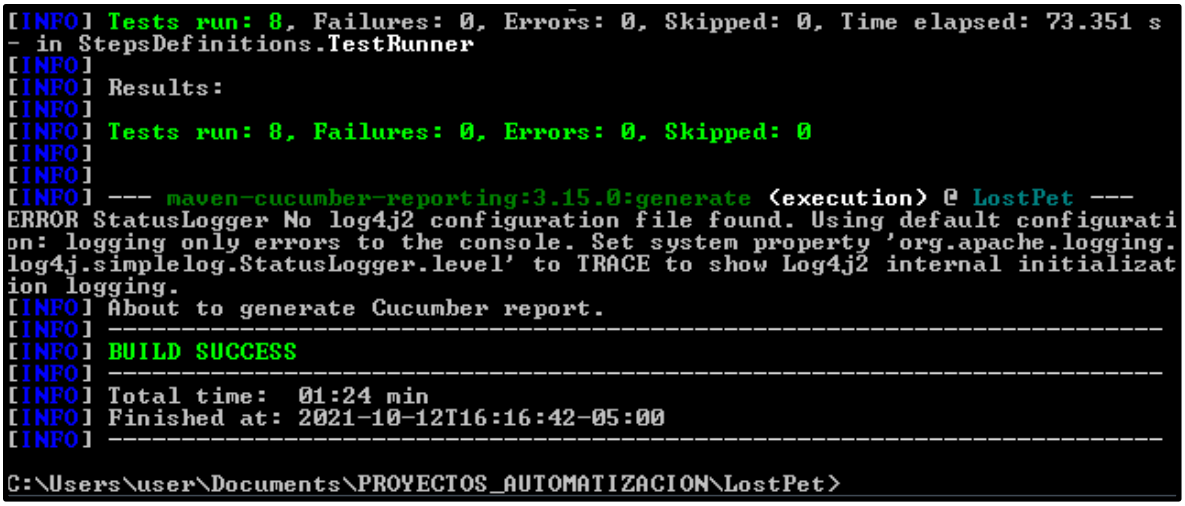
Imagen 43: Clase CrearRaza
Fuente: Propia

CrearRaza.java

```
3  import static org.junit.Assert.assertEquals;
4  import org.openqa.selenium.By;
5  import io.cucumber.java.en.And;
6  import io.cucumber.java.en.Given;
7  import io.cucumber.java.en.Then;
8  import io.cucumber.java.en.When;
9  import utils.CaptureScreen;
10 import utils.EliminarRegistros;
11 import utils.borrarArchivoDirectorio;
12
13
14 public class CrearRaza {
15
16     String escenario="1-CrearRaza";
17
18     @Given("Se inicia sesion como administrador")
19     public void se_inicia_sesion_como_administrador() {
20
21         TestLoginAdministrador miloginAdmin = new TestLoginAdministrador();
22         miloginAdmin.login();
23
24         EliminarRegistros elimina = new EliminarRegistros();
25         elimina.eliminar();
26
27         borrarArchivoDirectorio miborrarArchivoDirectorio= new borrarArchivoDirectorio();
28          miborrarArchivoDirectorio.borrar(".\\evidencias\\");
29     }
30
31     @And("Se ingresa al modulo de crear raza")
32     public void se_ingresa_al_modulo_de_crear_raza() {
33
34         System.out.println("====Inicia Creacion de Raza====");
35
36         obtenerDriver.driver.findElement(By.linkText("Administracion")).click();
37         obtenerDriver.driver.findElement(By.linkText("Administracion")).click();
38         obtenerDriver.driver.findElement(By.linkText("Razas")).click();
39         obtenerDriver.driver.findElement(By.linkText("Crear Raza")).click();
40     }
41
42     @When("Se diligencian los campos del formulario CR")
43     public void se_diligencian_los_campos_del_formulario_cr() {
44
45         obtenerDriver.driver.findElement(By.id("raza")).sendKeys("Jack Russell");
46
47     }
```

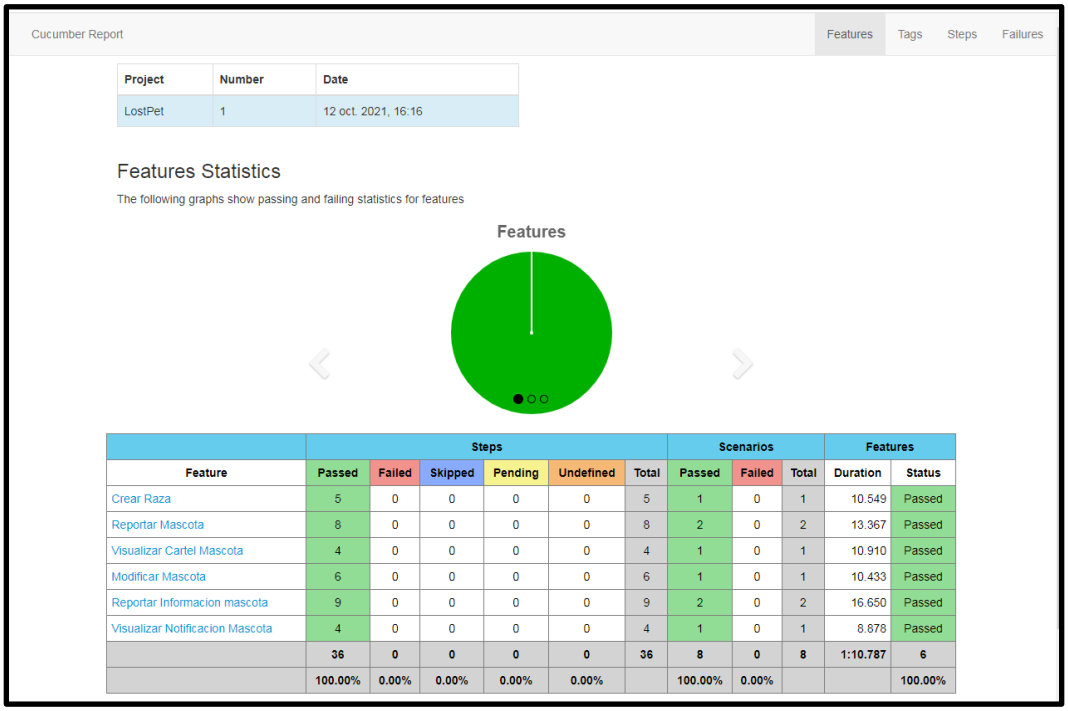
Luego de programar cada uno de las clases, se realizó la ejecución de las pruebas utilizando *Maven* desde la consola de Windows y los resultados fueron los siguientes:

Imagen 44: Resultados Ejecución Maven
Fuente: Propia



Para comprobar que todo se ejecutó exitosamente, se configuraron los reportes que ofrece la herramienta Cucumber, esto con el objetivo de visualizar el estado de la ejecución de cada Feature:

Imagen 45: Informe Pruebas Automatizadas
Fuente: Propia



Cucumber Report — Features | Tags | Steps | Failures

Project	Number	Date
LostPet	1	12 oct. 2021, 16:16

Features Statistics

The following graphs show passing and failing statistics for features

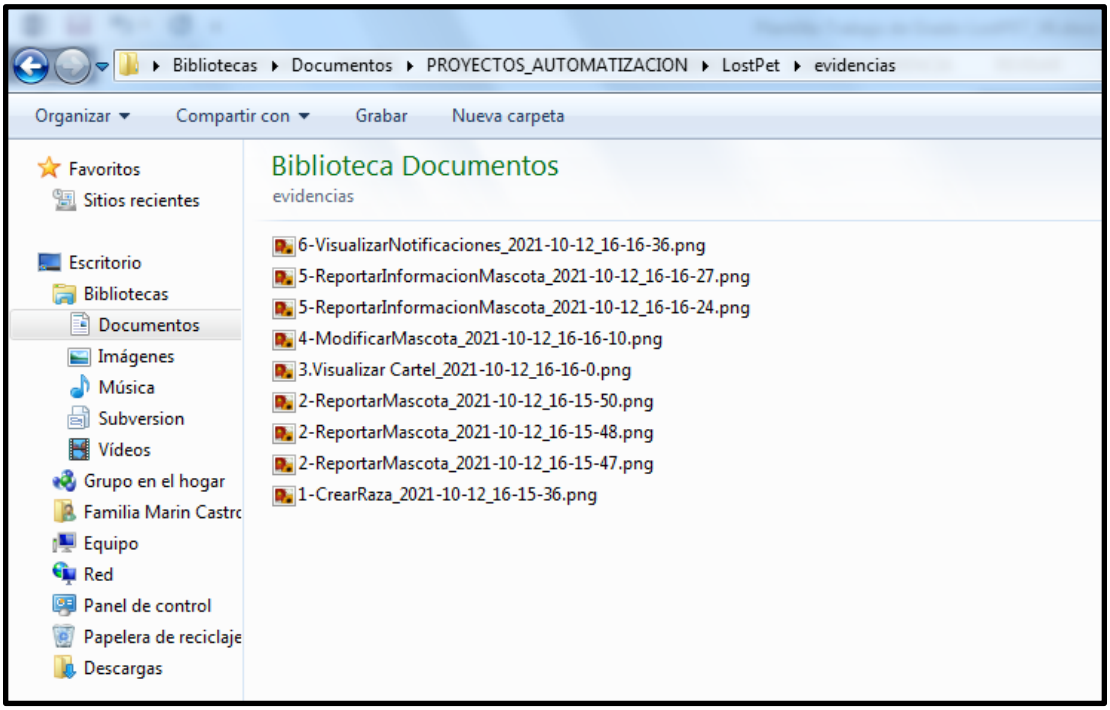
Features

	Steps						Scenarios			Features	
Feature	Passed	Failed	Skipped	Pending	Undefined	Total	Passed	Failed	Total	Duration	Status
Crear Raza	5	0	0	0	0	5	1	0	1	10.549	Passed
Reportar Mascota	8	0	0	0	0	8	2	0	2	13.367	Passed
Visualizar Cartel Mascota	4	0	0	0	0	4	1	0	1	10.910	Passed
Modificar Mascota	6	0	0	0	0	6	1	0	1	10.433	Passed
Reportar Informacion mascota	9	0	0	0	0	9	2	0	2	16.650	Passed
Visualizar Notificacion Mascota	4	0	0	0	0	4	1	0	1	8.878	Passed
	36	0	0	0	0	36	8	0	8	1:10.787	6
	100.00%	0.00%	0.00%	0.00%	0.00%		100.00%	0.00%			100.00%

En el anterior gráfico se pudo visualizar que las pruebas de regresión se ejecutaron exitosamente en un tiempo de 1:10 (1 minuto y 10 segundos).

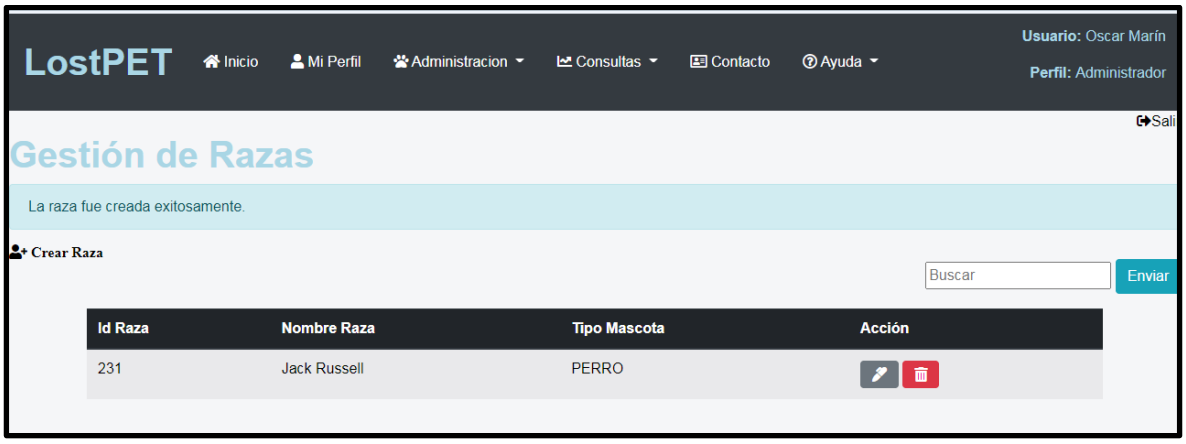
También se programó que los scripts por cada resultado (THEN) tomará una captura de pantalla y así evidenciar que todo se ejecutó sin ningún problema:

Imagen 46: Evidencias Pruebas Automatizadas
Fuente: Propia



CREACIÓN DE LA RAZA

Imagen 47: Evidencias Pruebas Automatizadas
Fuente: Propia



REPORTAR MASCOTA

Imagen 48: Evidencias Pruebas Automatizadas
Fuente: Propia

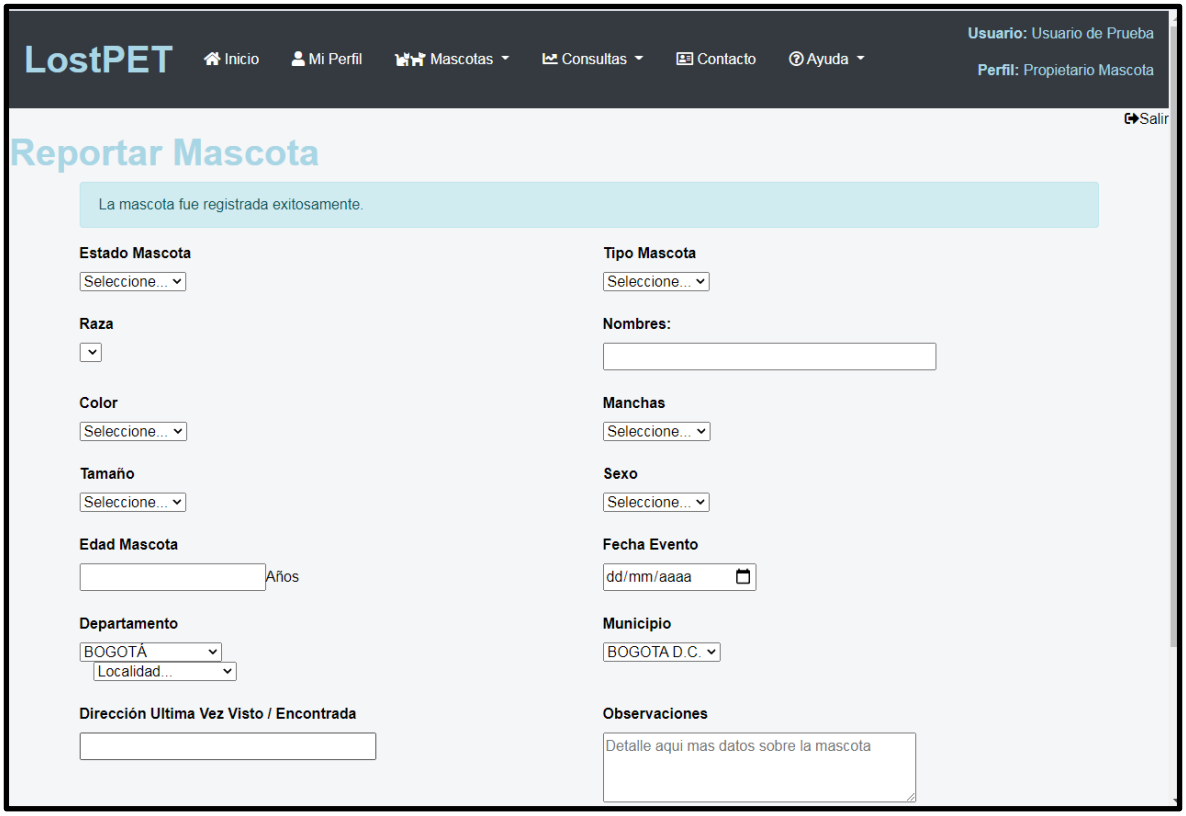
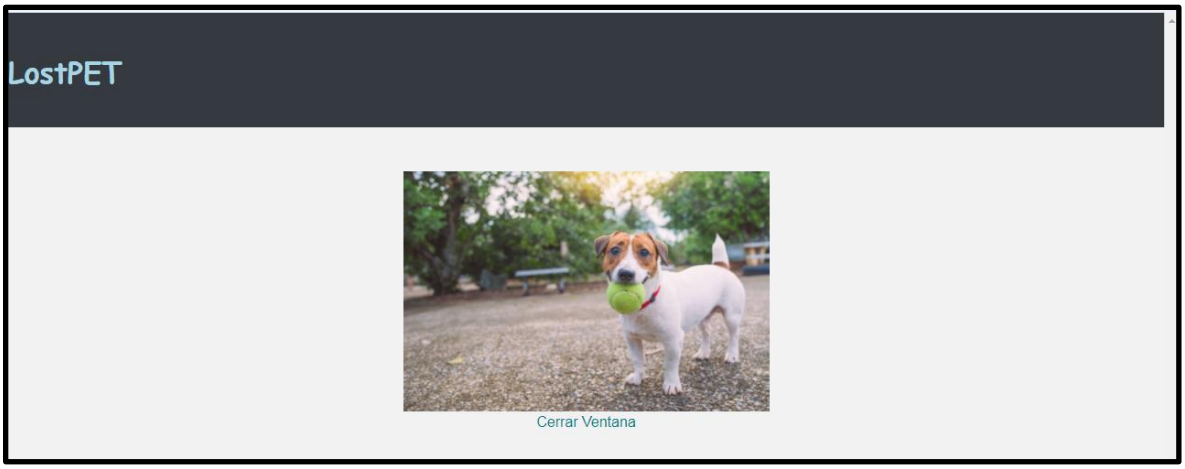


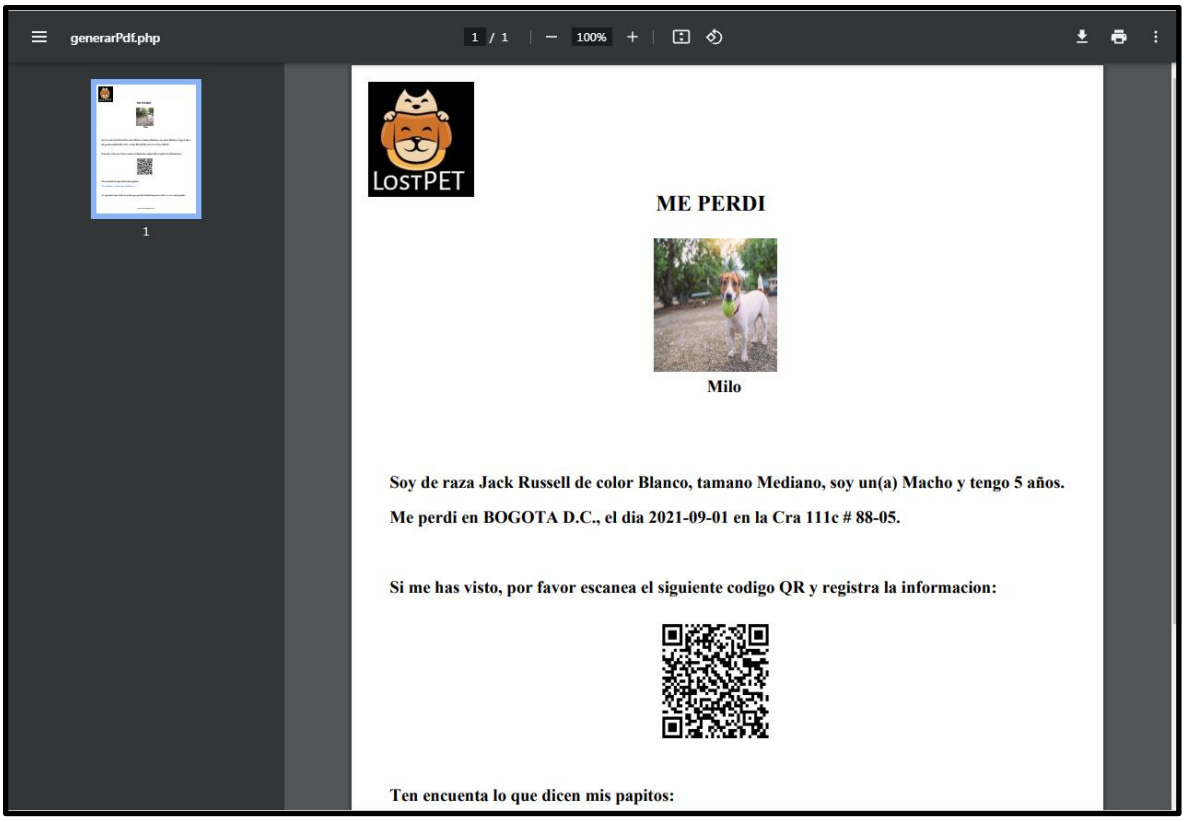
Imagen 46: Evidencias Pruebas Automatizadas
Fuente: Propia

Imagen 48: Evidencias Pruebas Automatizadas
Fuente: Propia



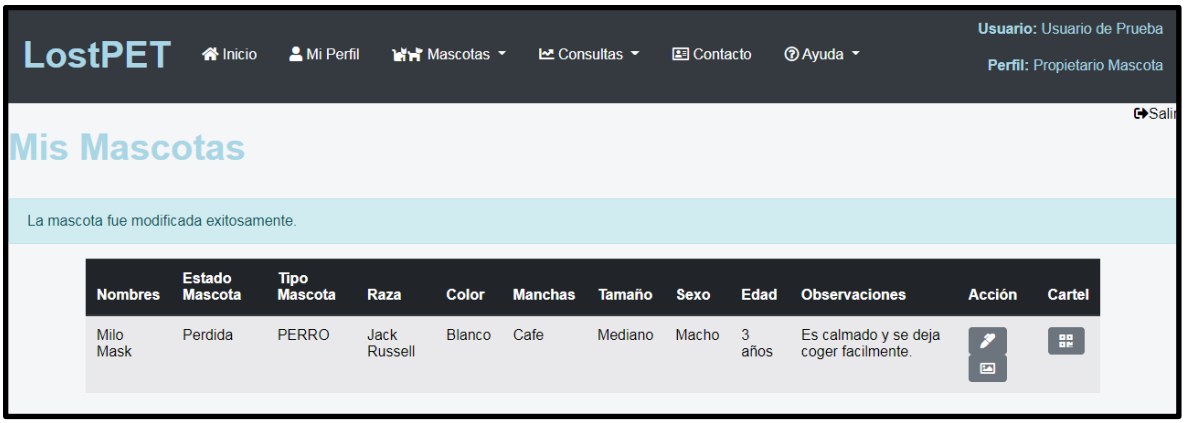
VISUALIZAR CARTEL

Imagen 49: Evidencias Pruebas Automatizadas
Fuente: Propia



MODIFICAR MASCOTA

Imagen 50: Evidencias Pruebas Automatizadas
Fuente: Propia



REPORTAR INFORMACIÓN MASCOTA

Imagen 51: Evidencias Pruebas Automatizadas
Fuente: Propia

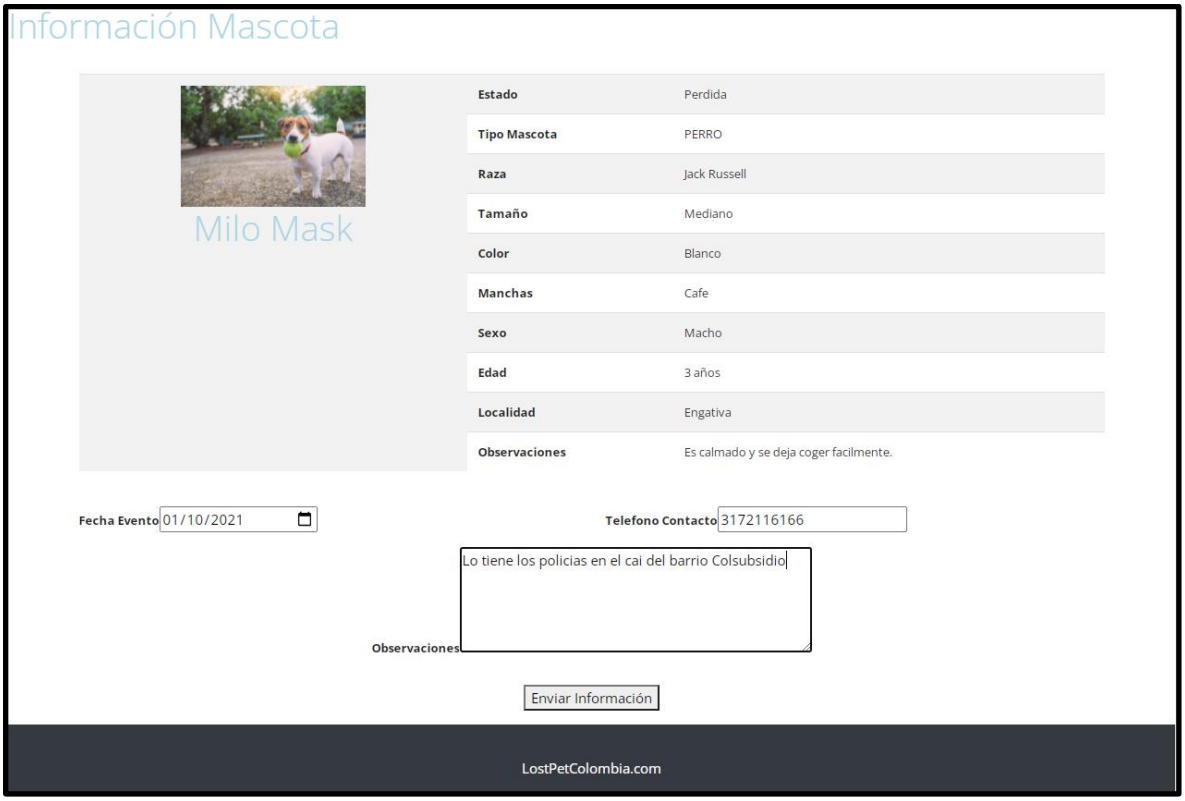
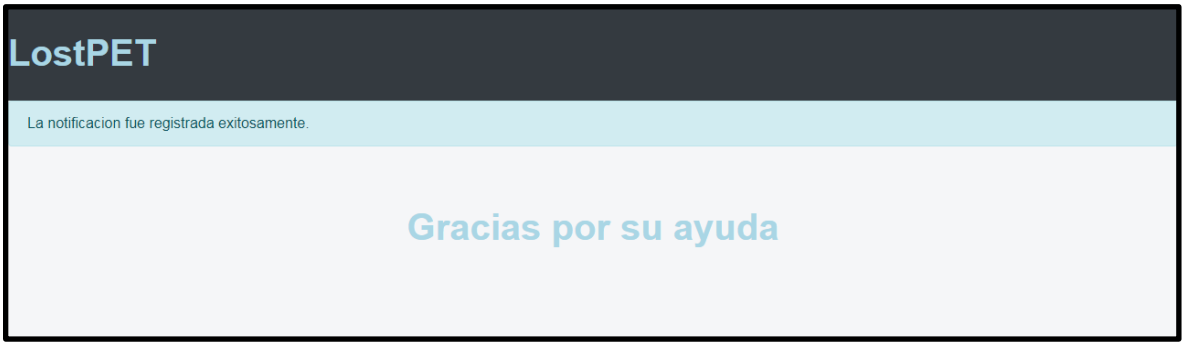


Imagen 52: Evidencias Pruebas Automatizadas
Fuente: Propia



VISUALIZAR NOTIFICACIONES

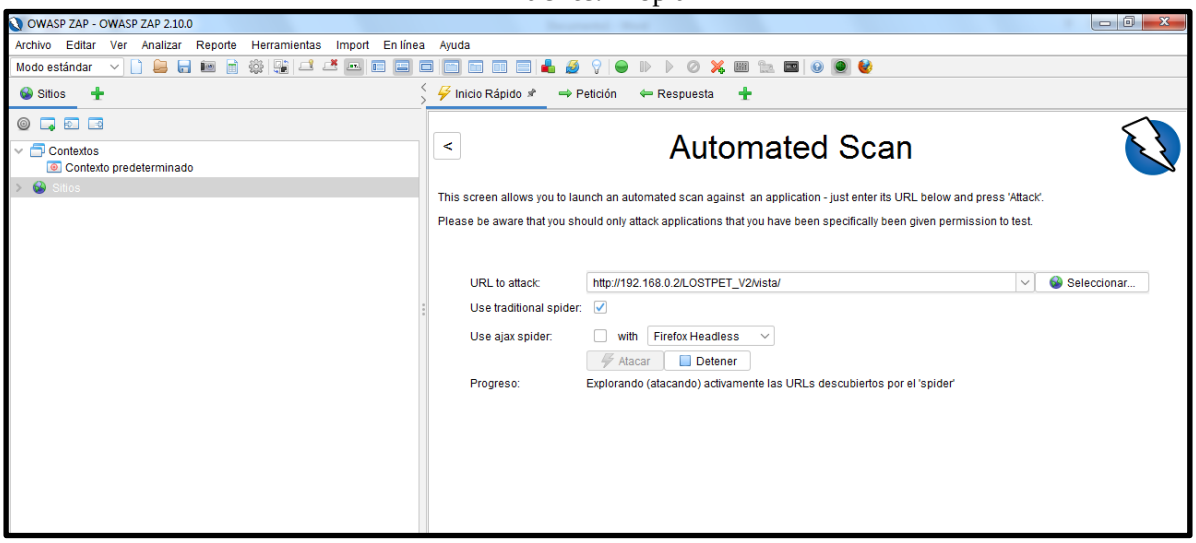
Imagen 53: Evidencias Pruebas Automatizadas
Fuente: Propia

Pruebas de Seguridad**:**

Se utilizó la herramienta Owasp ZAP para la búsqueda de vulnerabilidades en el proyecto. Las pruebas se realizaron teniendo presente los siguientes pasos:

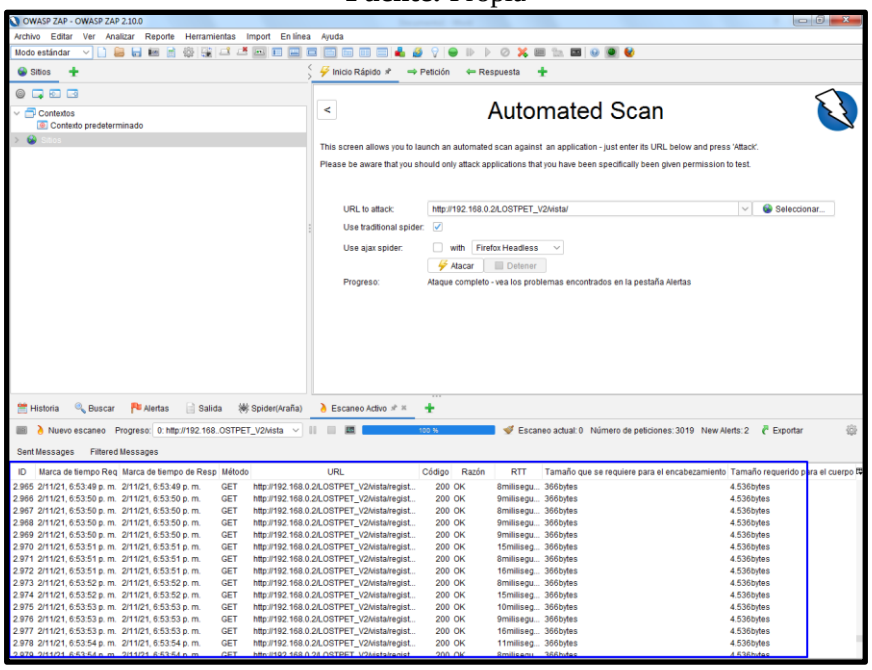
Configurar Página de Escaneo

Imagen 54: Configuración OWAS ZAP
Fuente: Propia



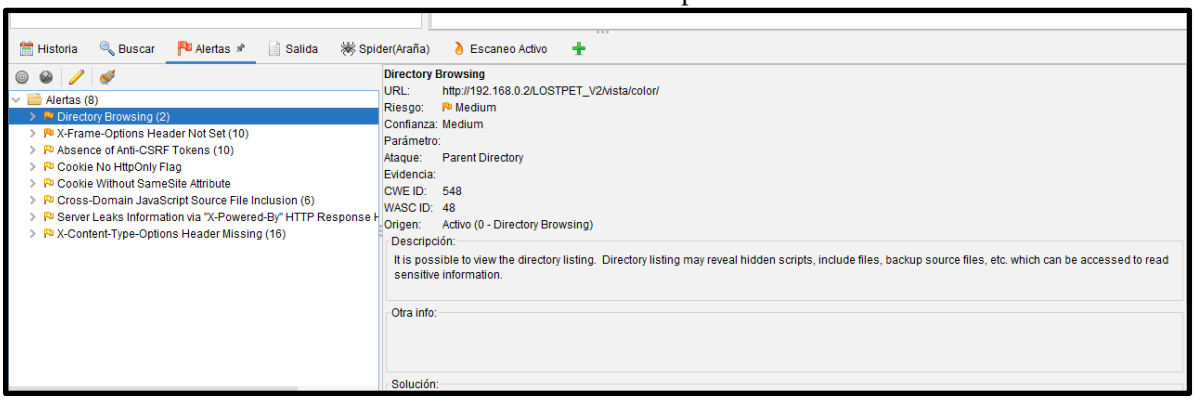
Inicia el escaneo del aplicativo

Imagen 55: Escaneo OWAS ZAP sobre aplicativo LostPet
Fuente: Propia



Se encontraron 8 alertas después del escaneo

Imagen 56: Resultados escaneo OWAS ZAP
Fuente: Propia



2 Riesgo Medio

6 Riesgo Bajo

Se ajustaron las alertas de Riesgo Medio **(Directory Browsing)** y solo quedaron las 6 de riesgo bajo.

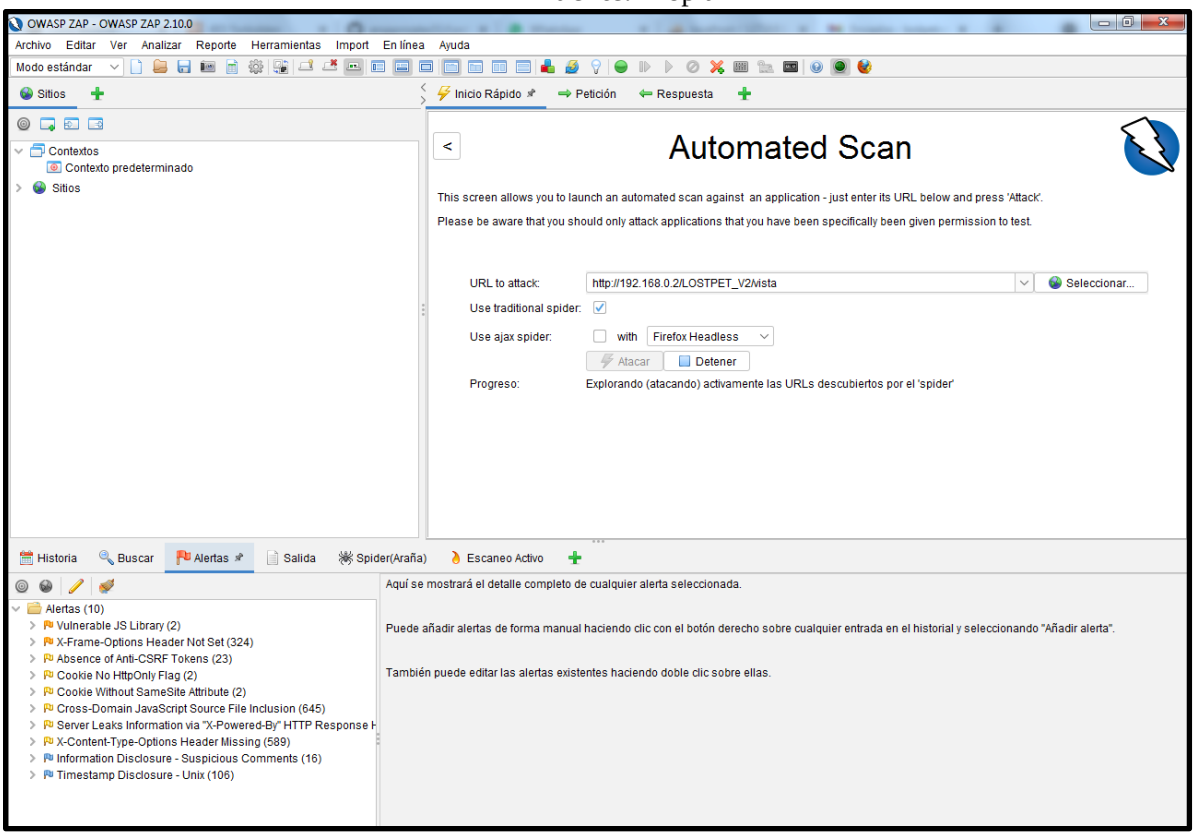
La alerta de riesgo medio se corrige ajustando en el servidor apache para que no se listen los directorios:

Imagen 57: No se listan directorios de carpetas
Fuente: Propia

Imagen 58: No se listan directorios de carpetas
Fuente: Propia

Se hace un nuevo escaneo y ya no aparece el riesgo asociado a **Directory Browsing**

Imagen 59: Resultado Nuevo Escaneo
Fuente: Propia



Controles de Seguridad Implementados

- Doble Factor de Autenticación Google Authenticator (Solicita un segundo paso para poder autenticarse, la App genera un código en la App del teléfono.)

Imagen 60: Inicio de Sesión LostPet
Fuente: Propia

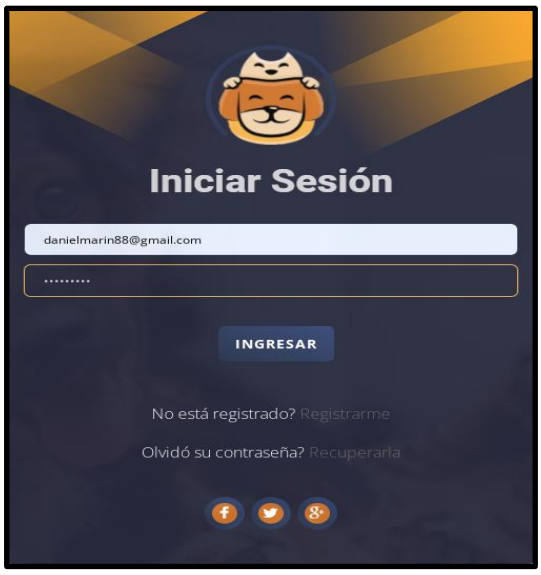
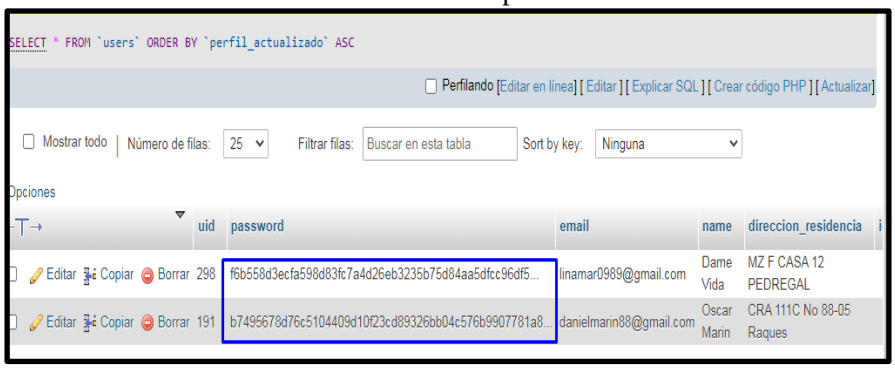


Imagen 61: Factor Doble Autenticación
Fuente: Propia

- Contraseña Segura (Mínimo 6 Caracteres, una letra mayúscula, una letra minúscula y un carácter numérico)

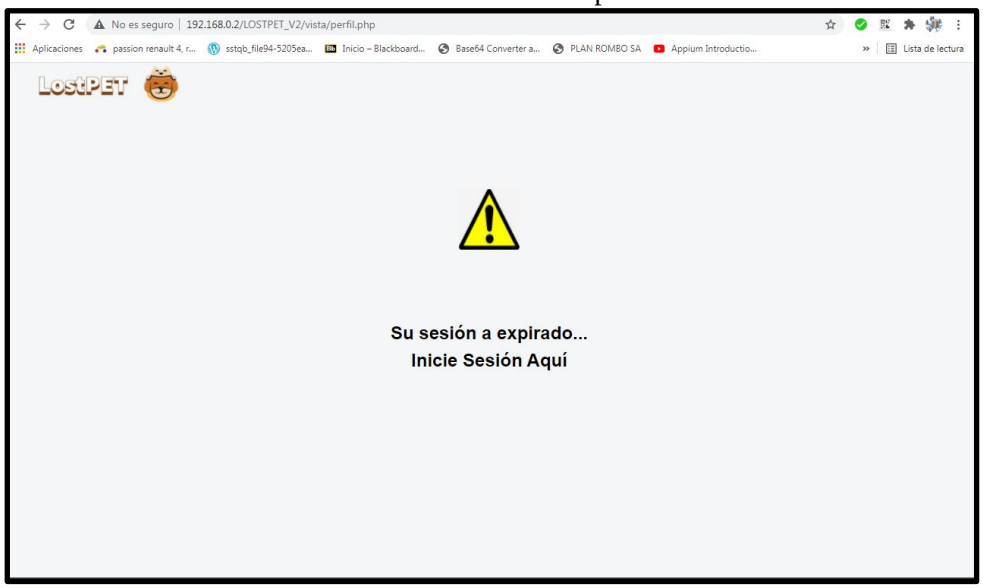
- Contraseña Encriptada (Utilizando algoritmo Sha56)

Imagen 62: Tabla User - LostPet
Fuente: Propia



- Expiración de Sesión (Se parametriza que a los 10 minutos el sistema termina la sesión del usuario y lo obligue a volver a loguearse)

Imagen 63: Validación Sesión Expirada
Fuente: Propia



- Control de módulos por perfilamiento (Un usuario diferente a administrador no puede ingresar a módulos mediante la url al cual no tiene permisos)

Imagen 64: Validación permisos Usuarios
Fuente: Propia

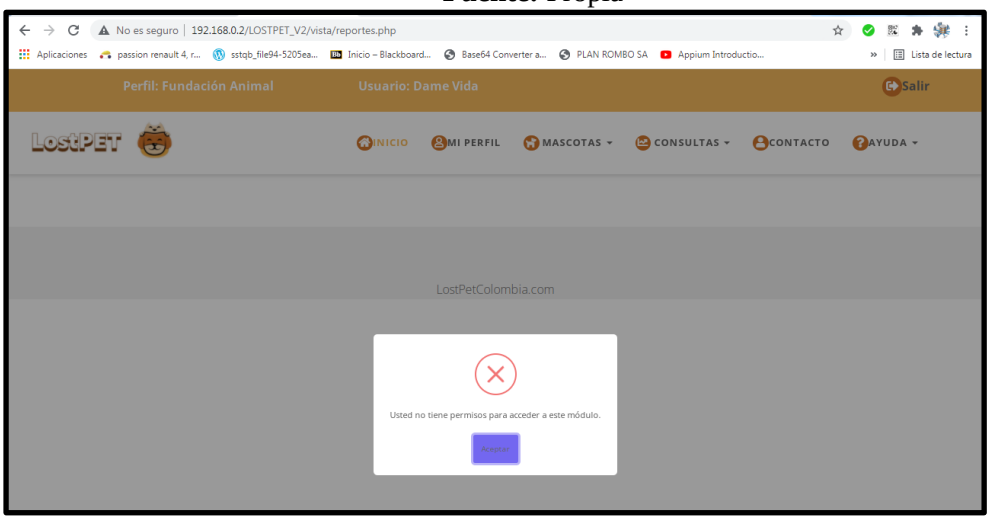
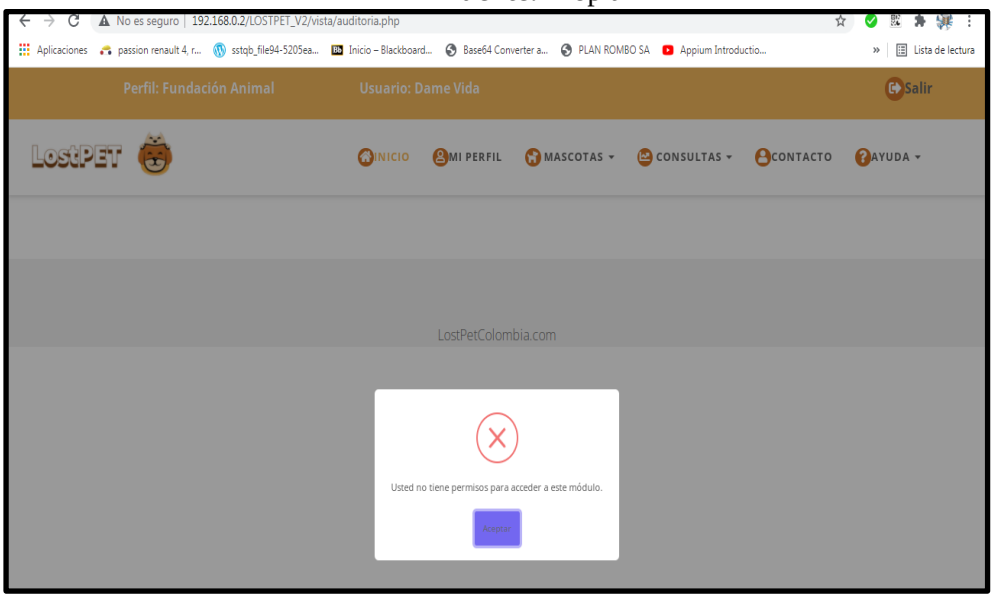


Imagen 65: Validación permisos Usuarios
Fuente: Propia

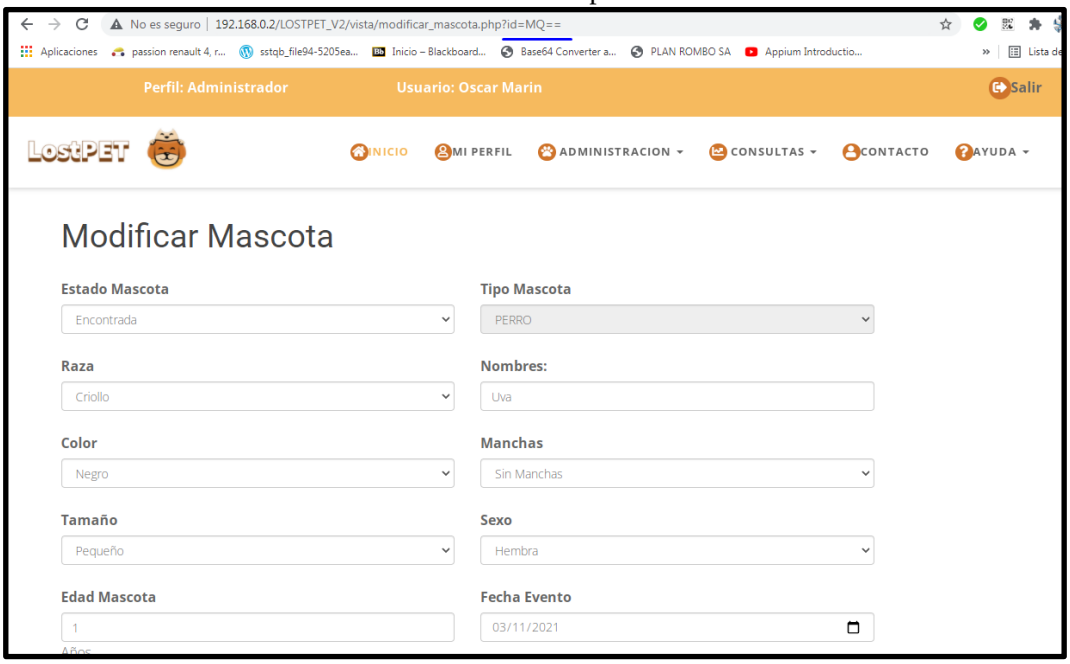


- Validación de Sesión (El sistema valida siempre que el usuario se haya logueado para poder navegar dentro del aplicativo. Si no encuentra una sesión activa, lo saca y lo envía al login para iniciar sesión.

Imagen 66: Validación permisos módulos
Fuente: Propia

- Enmascaramiento de variables pasadas por método GET

Imagen 67: Variables Enmascaradas
Fuente: Propia



13. Instalación y Configuración

En este apartado se mencionará el paso a paso para instalar la distribución de Apache en un computador para así convertirlo en el servidor.

1. Ingresar a la página web del fabricante (https://www.apachefriends.org/es/index.html) y dar clic en descargar de acuerdo al Sistema Operativo de su computador.

Imagen 68: Página de descarga Xampp
Fuente: https://www.apachefriends.org/es/index.html

2. En la parte inferior se comenzará a descargar un ejecutable; dar clic para iniciar la instalación.

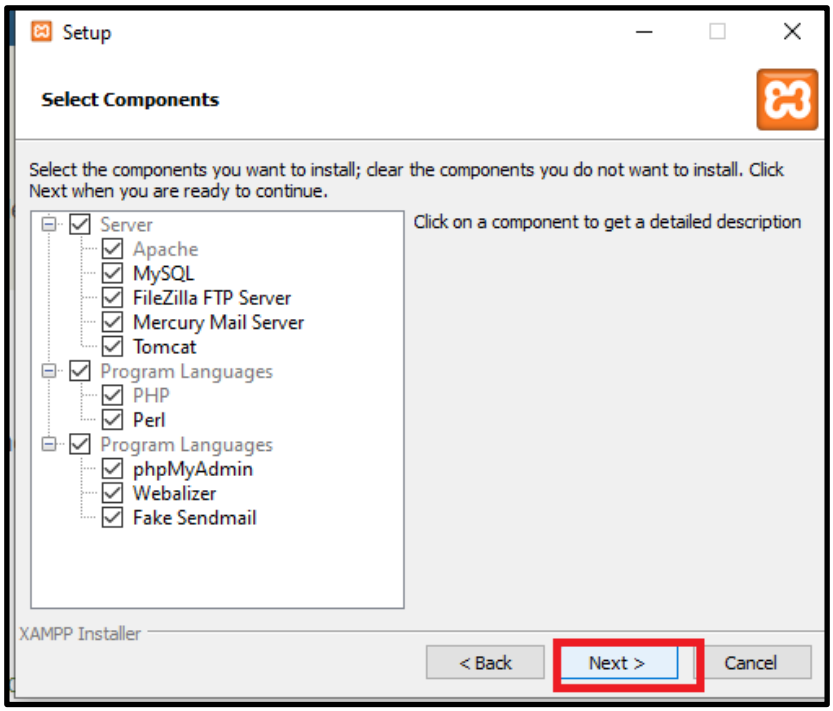
Imagen 69: Página de descarga Xampp
Fuente: https://www.apachefriends.org/es/index.html

3. Ejecutar el archivo previamente descargado y dar clic en “Next” o “Siguiente” En caso de solicitar permisos elevados otorgarlos.

Imagen 70: Instalación Xampp
Fuente: https://www.apachefriends.org/es/index.html

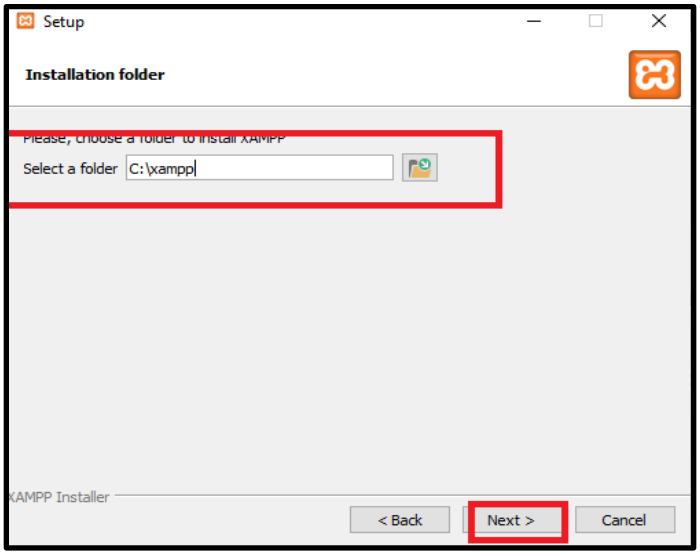
4. Dar clic en “Next” o “Siguiente”.

Imagen 71: Instalación Xampp
Fuente: https://www.apachefriends.org/es/index.html



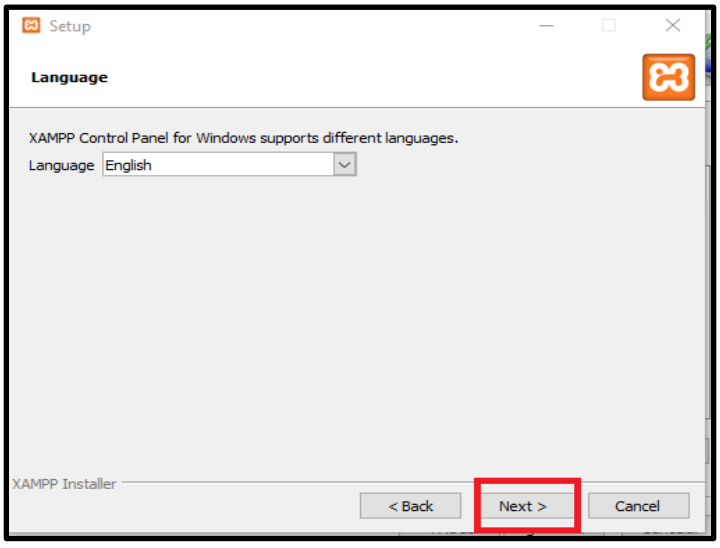
5. Si se deja la instalación en una ruta diferente a la establecida por defecto, cambiarla, en caso contrario dar en next.

Imagen 72: Instalación Xampp
Fuente: https://www.apachefriends.org/es/index.html



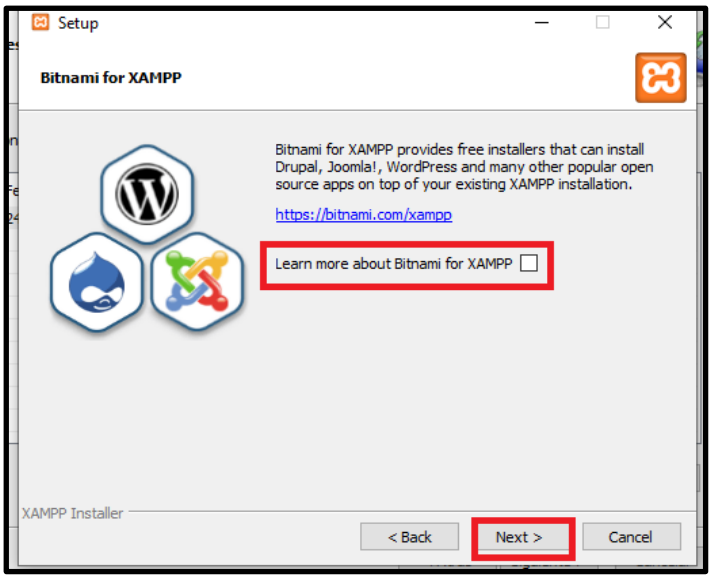
6. Seleccionar el idioma de instalación; el idioma por default es Inglés.

Imagen 73: Instalación Xampp
Fuente: https://www.apachefriends.org/es/index.html



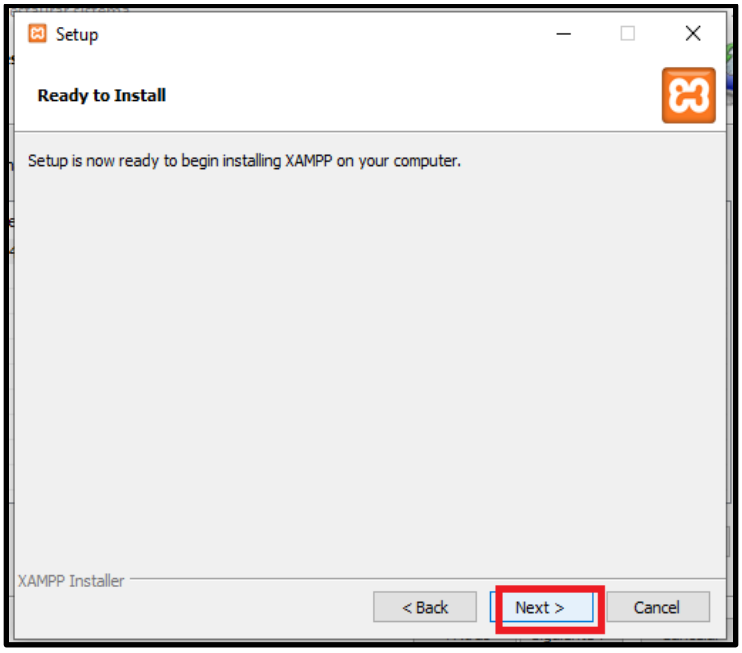
7. Se procede a desmarcar el Check y dar clic en “Next” o “Siguiente”.

Imagen 74: Instalación Xampp
Fuente: https://www.apachefriends.org/es/index.html



8. Dar clic en “Next” o “Siguiente”.

Imagen 75: Instalación Xampp
Fuente: https://www.apachefriends.org/es/index.html



9. Comenzará la instalación de Xampp por lo cual debemos esperar hasta que finalice.

Imagen 76: Instalación Xampp
Fuente: https://www.apachefriends.org/es/index.html

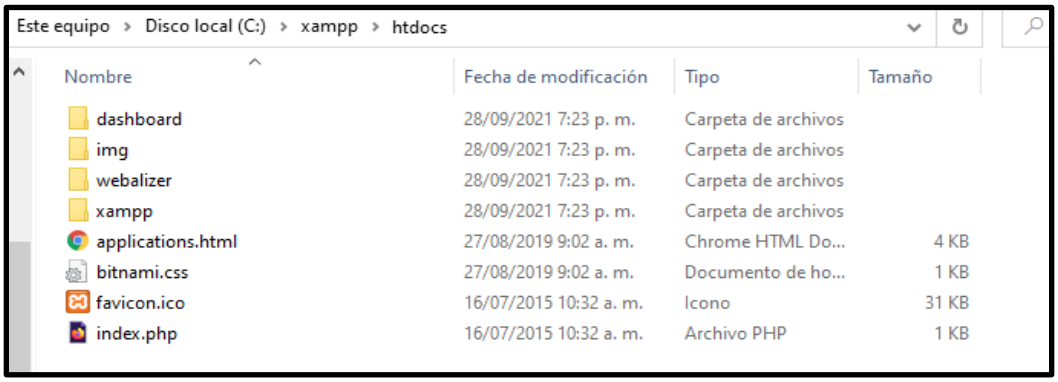
10. Por último dar clic en “Finish” o “finalizar

Imagen 77: Instalación Xampp
Fuente: https://www.apachefriends.org/es/index.html

6

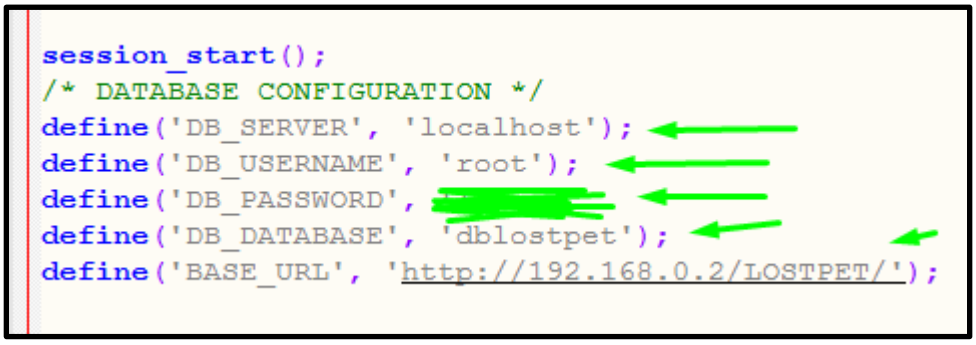
11. Nos situamos en la carpeta htdocs y copiamos allí los binarios de la aplicación que se encuentra compartida dentro de los entregables (Capa Media).

Imagen 78: Carpeta Htdocs
Fuente: Propia



Al tener los binarios alojados en el server que se acabó de instalar, se debe realizar la configuración del archivo config.php ubicado en la carpeta modelo:

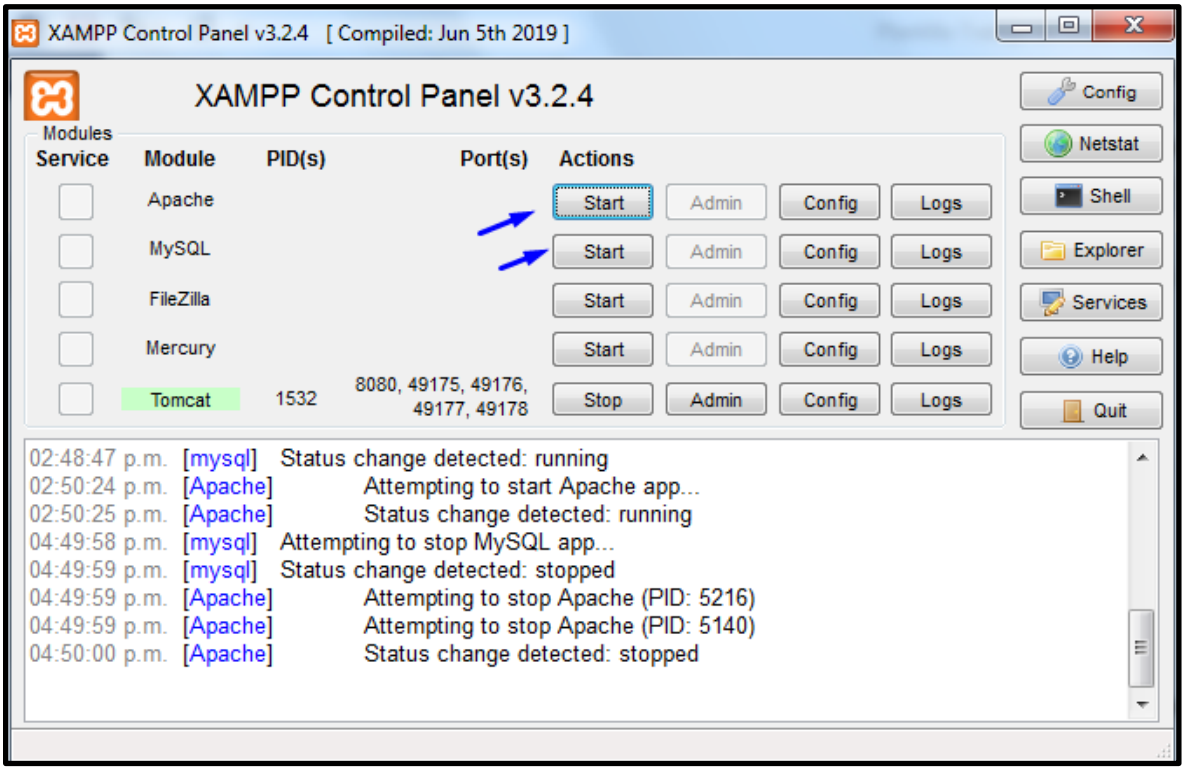
Imagen 79: Archivo configuración conexión BD
Fuente: Propia



Aquí se debe configurar los datos del servidor de aplicación y base de datos en donde quedará alojada la aplicación.

Luego de haber realizado la instalación del Xampp y configurado las variables de conexión con la base de datos, procedemos a iniciar los servicios de MySql y Apache abriendo la consola del programa y pulsando los botones de Start:

Imagen 80: Panel de control XAMPP
Fuente: Propia



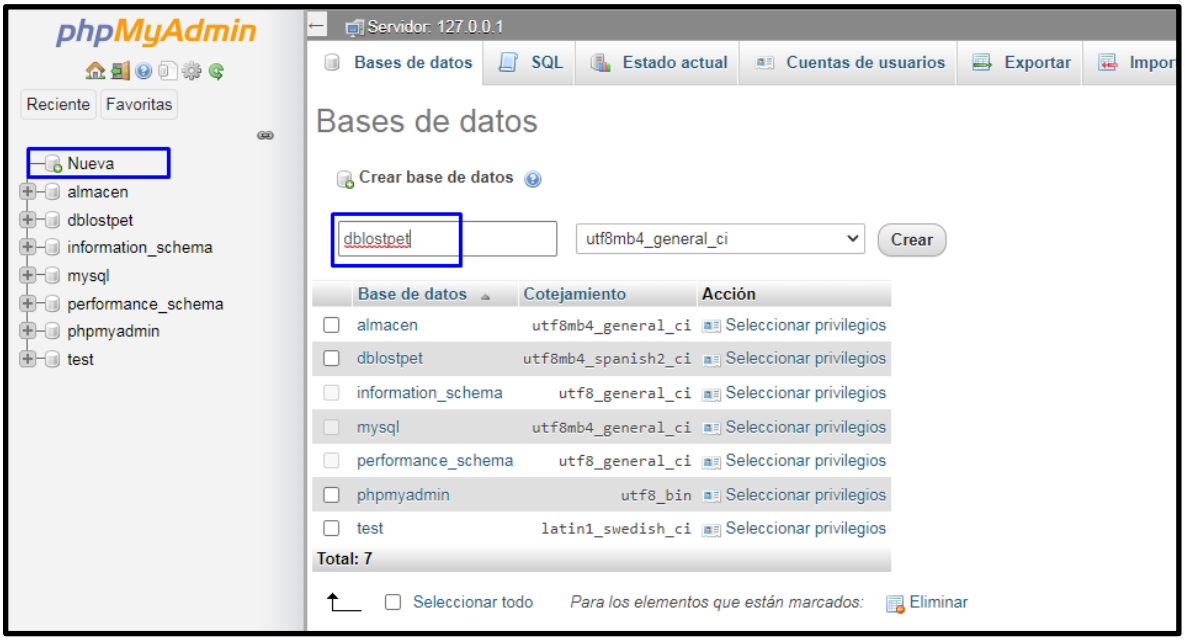
En la parte inferior del gráfico se muestra que los servicios han iniciado.

Luego de iniciar los servicios, en el equipo en donde se ha hecho la instalación del server, ingresamos a la siguiente url para realizar la creación de la base de datos:

http://localhost/phpmyadmin/

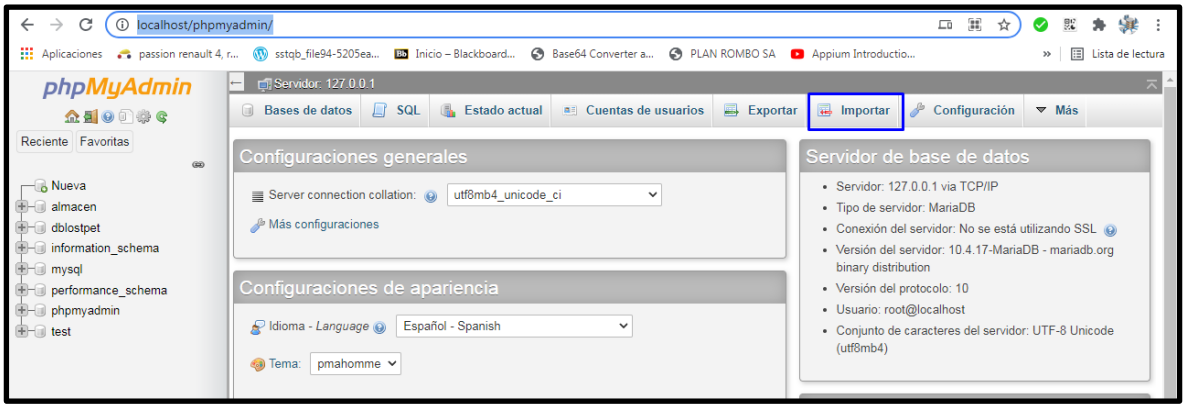
Se debe crear la base de datos con el nombre dblostpet por la siguiente opción:

Imagen 81: Creación BD
Fuente: Propia



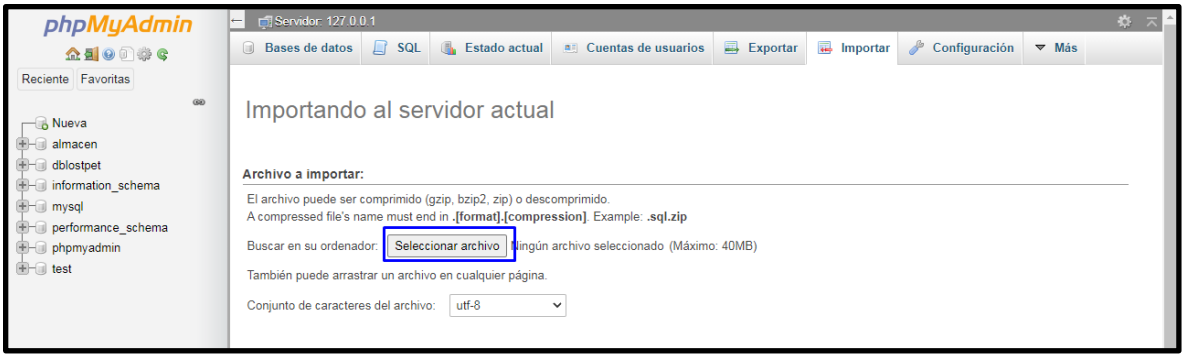
Luego de la creación de la base de datos y estando dentro de ellas, se pulsa la opción Importar:

Imagen 82: Creación BD
Fuente: Propia



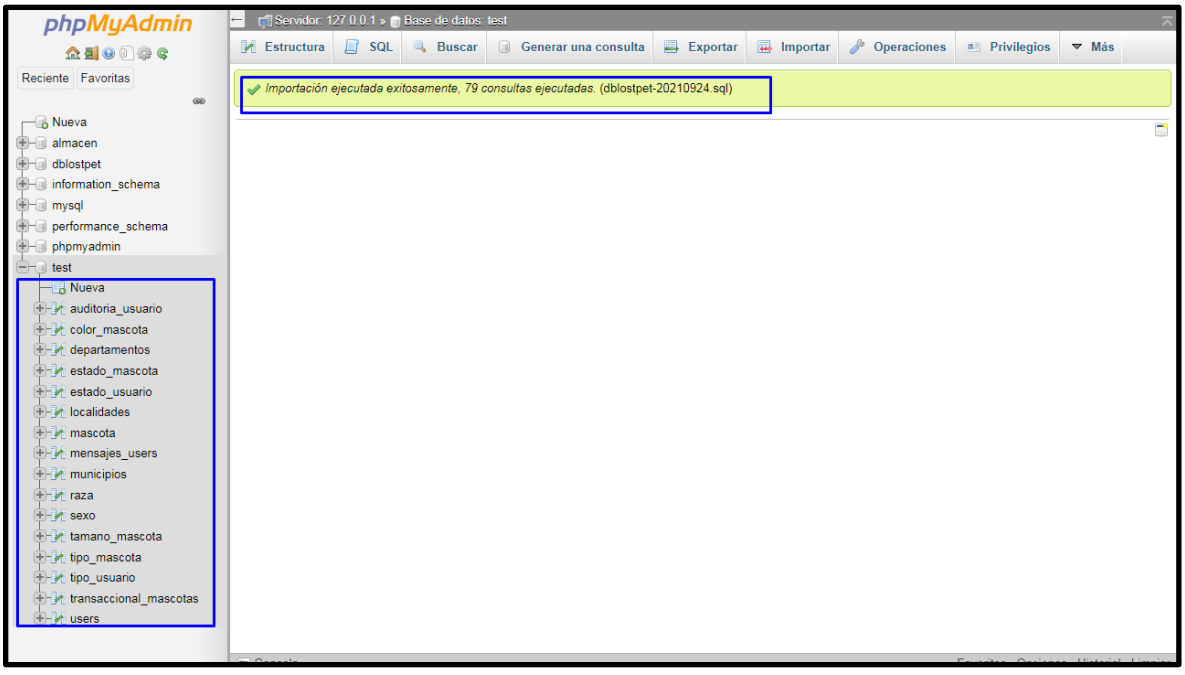
Aquí se selecciona el archivo que se encuentra compartido dentro de los entregables (Base de Datos-dblostpet) y pulsa el botón continuar.

Imagen 83: Creación BD
Fuente: Propia



Se procesa el archivo y al final se muestra que se ejecutó exitosamente la consulta:

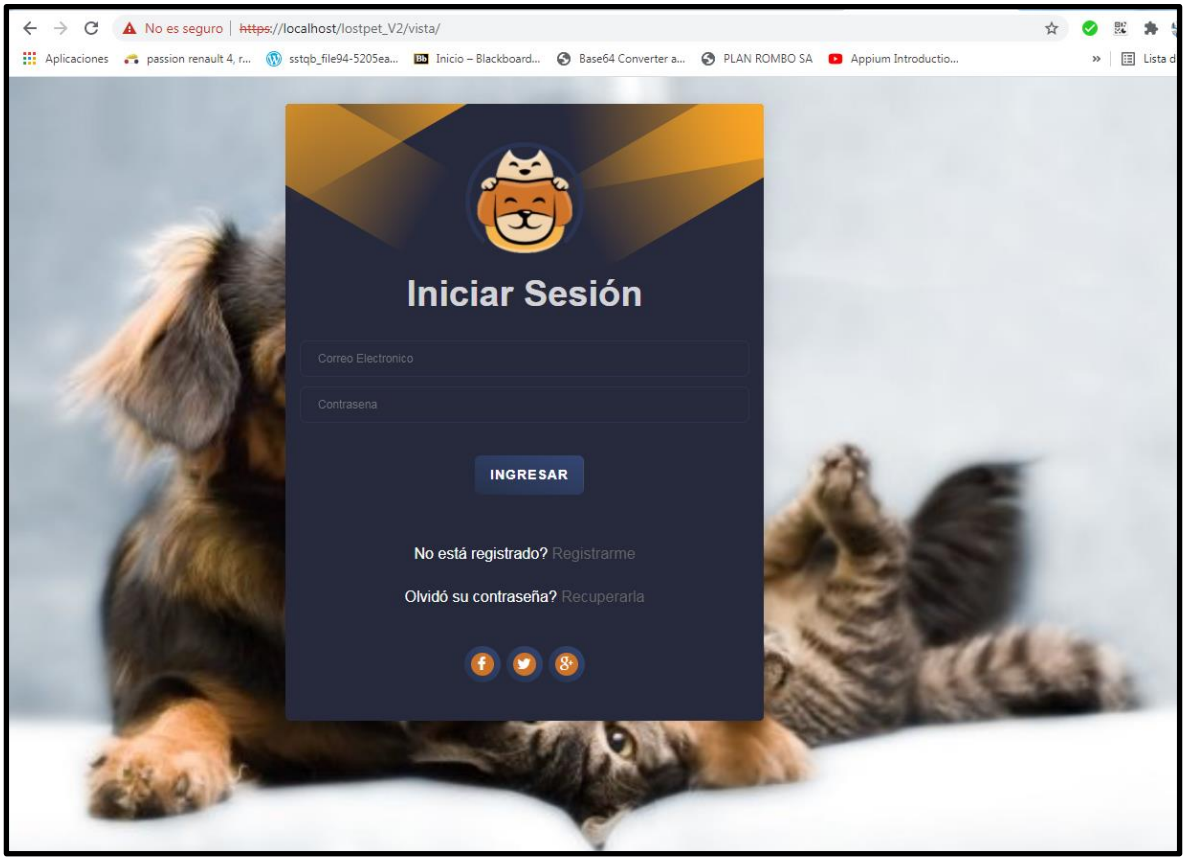
Imagen 84: Creación BD
Fuente: Propia



Posteriormente podrá ingresar a la siguiente url (esta url depende en donde esté instalado el servidor web) y empezar a interactuar con el aplicativo:

http://192.168.0.2/LOSTPET/vista/

Imagen 85: Login Aplicativo LostPET
Fuente: Propia



Nota: *Para poder crear un usuario Administrador, deberá registrarse inicialmente y luego por base de datos, en la tabla users actualizar el campo tipo_usuario a 1.*

14. Conclusiones

- Se logró implementar en el aplicativo la centralización de la información de animales perdidos o encontrados con una interfaz gráfica que permitió registrar, modificar y consultar la información.

- Se diseñó un cartel en Pdf que contenía un código QR que al ser escaneado permitió reportar información sobre alguna mascota perdida o encontrada.

- Las notificaciones sobre la información reportada sobre una mascota se implementaron vía correo electrónico.

- La protección de los datos se salvaguardan al no brindar información personal a los usuarios de quien perdió una mascota para evitar llamadas fraudulentas.

- Se utilizó el correo electrónico como medio de contacto entre los usuarios y los dueños de mascotas para evitar la divulgación de datos personales.

- Se automatizan las pruebas funcionales utilizando la metodología (Behavior Driven Development), desarrollo guiado por comportamiento, la cual ejecutaba automáticamente las pruebas de regresión y dejaba un informe con el resultado y las evidencias de la ejecución

15. Anexos

- Historias de Usuario
- Formato Guion de Pruebas
- Evidencias de Pruebas Funcionales
- Formato de Reporte de Incidencias

16. Referencias

Andrews, S. Fastqc, (2010). A quality control tool for high throughput sequence data.

Augen, J. (2004). Bioinformatics in the post-genomic era: Genome, transcriptome, proteome, and information-based medicine. Addison-Wesley Professional.

Blankenberg, D., Kuster, G. V., Coraor, N., Ananda, G., Lazarus, R., Mangan, M., ... & Taylor, J. (2010). Galaxy: a web-based genome analysis tool for experimentalists. Current protocols in molecular biology, 19-10.

Bolger, A., & Giorgi, F. Trimmomatic: A Flexible Read Trimming Tool for Illumina NGS Data. URL http://www.usadellab.org/cms/index.php.

Giardine, B., Riemer, C., Hardison, R. C., Burhans, R., Elnitski, L., Shah, P., ... & Nekrutenko, A. (2005). Galaxy: a platform for interactive large-scale genome analysis. Genome research, 15(10), 1451-1455.

Periódico el Tiempo, Mascotas: cielos en calma y sin pólvora en festividades, Edicion Digital , 04 de Diciembre del 2020 https://www.eltiempo.com/vida/mascotas/navidad-y-ano-nuevo-efectos-de-la-polvora-en-mascotas-552275

Reporte de vacunación antirrábica de perros y gatos Colombia año 2017 https://www.minsalud.gov.co/sites/rid/Lists/BibliotecaDigital/RIDE/VS/PP/SA/nacional-municipio-2017.pdf