ANÁLISIS DE PREVALENCIA DE ENFERMEDADES EN MASCOTAS

ANALYSIS OF DISEASE PREVALENCE IN PETS

TRABAJO FIN DE GRADO
CURSO 2024-2025

AUTOR
PABLO JESÚS BLANCO CARRASCO
ALEJANDRO GÓMEZ MARTÍN

DIRECTOR
YOLANDA GARCÍA RUIZ
JOSÉ IGNACIO GÓMEZ PÉREZ

GRADO EN INGENIERÍA INFORMÁTICA
FACULTAD DE INFORMÁTICA
UNIVERSIDAD COMPLUTENSE DE MADRID

ANÁLISIS DE PREVALENCIA DE ENFERMEDADES EN MASCOTAS

ANALYSIS OF DISEASE PREVALENCE IN PETS

TRABAJO DE FIN DE GRADO EN INGENIERÍA INFORMÁTICA

AUTOR
PABLO JESÚS BLANCO CARRASCO
ALEJANDRO GÓMEZ MARTÍN

DIRECTOR
YOLANDA GARCÍA RUIZ
JOSÉ IGNACIO GÓMEZ PÉREZ

CONVOCATORIA: SEPTIEMBRE 2025

GRADO EN INGENIERÍA INFORMÁTICA
FACULTAD DE INFORMÁTICA
UNIVERSIDAD COMPLUTENSE DE MADRID

1 DE SEPTIEMBRE DE 2025

DEDICATORIA

Alejandro Gómez Martín

Quiero dedicar este trabajo a mi familia,
tanto a mis padres como a mi hermano, y
a mis amigos por acompañarme durante
estos años en la carrera.

Pablo Jesús Blanco Carrasco

Dedico este trabajo a mis familiares y amigos.
Especialmente a mis padres, mis hermanos
y mis abuelos que tanto me han
apoyado todos estos años

AGRADECIMIENTOS

Alejandro Gómez Martín

A Nacho, a Yolanda y a Irene por ayudarnos a realizar el trabajo de la mejor forma, especialmente al principio. También quiero agradecer a mi compañero Pablo por el esfuerzo puesto en el trabajo y por todos estos años acompañándome en la carrera.

Pablo Jesús Blanco Carrasco

Agradezco a nuestros tutores Nacho y Yolanda por la paciencia y el empeño que han puesto en el trabajo. También a todos mis seres queridos por apoyarme en todo momento y por último agradecer a mi compañero Alejandro por su esfuerzo en este trabajo, pero sobre todo por todos estos años de carrera.

RESUMEN

Análisis de prevalencia de enfermedades en mascotas

La convivencia entre mascotas y dueños constituye una práctica común en numerosos hogares, aportando múltiples beneficios tanto a nivel emocional y como a nivel social. Sin embargo, esta interacción estrecha entre humanos y animales también puede implicar ciertos riesgos para la salud. Uno de los principales riesgos es la posible transmisión de enfermedades infecciosas, siendo especialmente relevante en personas inmunocomprometidas. En estos casos, las mascotas pueden actuar como reservorios o vectores de diversos patógenos con potencial de afectar la salud humana.

El creciente número de mascotas en los hogares en los últimos años hace cada vez más necesario estudiar la propagación de enfermedades que pueden transmitirse a los humanos, con el fin de prevenirlas y controlarlas de manera efectiva. Este trabajo parte de datos sobre la incidencia de determinadas enfermedades en mascotas en distintas zonas de la geografía española para desarrollar una aplicación web que permite visualizar su distribución espacio-temporal y detectar posibles puntos críticos. La herramienta también permite visualizar de forma espacio-temporal los datos de temperatura, con el objetivo de analizar cómo influye este factor en el aumento de casos de mascotas infectadas. Asimismo, la incorporación de información sobre la incidencia en humanos permite el estudio de si es posible predecir futuros incrementos de casos en personas a partir de las variables ambientales y la evolución de los contagios en animales. Todo este análisis se integra en una plataforma web interactiva y por capas, diseñada para que investigadores puedan explorar visualmente los datos a nivel nacional, identificar zonas representativas y relacionarlas con factores ambientales. La herramienta está pensada para ser intuitiva y útil como apoyo en futuras investigaciones, además de ser diseñada teniendo en cuenta su escalabilidad para que pueda adaptarse y crecer según vayan evolucionando las necesidades del proyecto y/o se amplíe su ámbito de aplicación.

Palabras clave

Mascotas, enfermedades infecciosas, personas inmunocomprometidas, visualización geográfica, puntos calientes, herramienta web, epidemiología, zoonosis

ABSTRACT

Analysis of disease prevalence in pets

The coexistence of pets and their owners is a common practice in many households, providing multiple benefits both emotionally and socially. However, this close interaction between humans and animals can also pose certain health risks. One of the main risks is the possible transmission of infectious diseases, which is particularly relevant in immunocompromised individuals. In these cases, pets can act as reservoirs or vectors for various pathogens with the potential to affect human health.

The growing number of pets in households in recent years makes it increasingly necessary to study the spread of diseases that can be transmitted to humans, to prevent and control them effectively. This work is based on data on the incidence of certain diseases in pets in different areas of Spain to develop a web application that allows their spatial-temporal distribution to be visualised and possible critical points to be detected. The tool also allows temperature data to be visualised spatially and temporally, with the aim of analysing how this factor influences the increase in cases of infected pets. Likewise, the incorporation of information on the incidence in humans allows for the study of whether it is possible to predict future increases in cases in people based on environmental variables and the evolution of infections in animals. All this analysis is integrated into an interactive, layered web platform designed so that researchers can visually explore the data at the national level, identify representative areas and relate them to environmental factors. The tool is designed to be intuitive and useful as a support for future research, as well as being designed with scalability in mind so that it can be adapted and expanded as the needs of the project evolve and/or its scope of application broadens.

Keywords

Pets, infectious diseases, immunocompromised individuals, geographic visualisation, hot spots, web tool, epidemiology, zoonoses

ÍNDICE DE CONTENIDOS

Contenido

Capítulo 1 - Introducción 1

1.1 Motivación 1

1.2 Objetivos 2

1.3 Plan de trabajo 3

Capítulo 2 - Origen y preprocesamiento de los datos 5

2.1 Origen y características de los datos 5

2.1.1 Datos de test a mascotas 5

2.1.2 Datos geográficos de los códigos postales 7

2.1.3 Datos censales de mascotas 8

2.1.4 Datos humanos 9

2.1.5 Datos meteorológicos 11

2.2 Preparación y estructuración para el análisis 14

Capítulo 3 - Análisis espacio-temporal 18

3.1 Métodos estadísticos candidatos 18

3.1.1 Método estadístico Getis-Ord 18

3.1.2 Método estadístico Anselin 20

3.1.3 Justificación e implementación del algoritmo 21

3.2 Aplicación en los datos de mascotas 22

3.2.1 Análisis espacial de los casos 22

3.2.2 Estacionalidad de los datos 25

3.3 Extensión del estudio en humanos 41

3.3.1 Particularidades de los datos de Leishmaniasis en humanos 41

3.3.2 Resultados del análisis espacial en humanos....42
3.3.3 Conclusión y comparativa con el caso de mascotas....46
Capítulo 4 - Introducción a la aplicación....48
4.1 Introducción al funcionamiento de la aplicación....48
4.2 Estructura general....50
4.3 Implementación backend....53
4.3.1 Tecnologías usadas en la implementación....53
4.3.2 Arquitectura modular del ¨backend¨....55
4.3.3 Acceso a la base de datos....57
4.3.4 API: rutas y flujo de peticiones....61
4.3.5 Autenticación y autorización....65
4.4 Implementación frontend....66
4.5 Tecnologías usadas....66
4.5.1 Desarrollo del frontend con React....67
4.5.2 Estructura del "front"....67
4.5.3 Integración con mapas interactivos....75
4.5.4 Uso de la API....76
4.6 Servidor web con Nginx....77
4.7 Justificación de la arquitectura elegida....79
Capítulo 5 - Despliegue de aplicación....80
5.1 Acerca de Docker y Docker Compose....80
5.2 Arquitectura de los contenedores....80
5.2.1 Servicio base de datos....82
5.2.2 Servicio backend....83
5.2.3 Servicio app....84

5.2.4 Servicio Nginx 84
5.3 Despliegue 84
Capítulo 6 - Conclusiones y trabajo futuro 85
6.1 Conclusión 85
6.2 Trabajo futuro 85
Introduction 87
Conclusions and future work 91
Contribuciones Personales 93

ÍNDICE DE FIGURAS

Figura 1-1: Gráfico del censo de mascotas de 2025..... 1

Figura 3-1: Resultados de la aplicación de GetisOrd de forma global a los datos de mascotas 24

Figura 3-2: Gráfico del número de casos positivos de Leishmaniasis semanales por año 28

Figura 3-3: Gráfico representativo de la autocorrelación de los datos de Leishmaniasis 29

Figura 3-4: Gráfico representativo de la autocorrelación de los datos de Leishmaniasis tras hacer una primera diferencia 30

Figura 3-5: Histograma de los residuos tras haber aplicado ARIMA(2,1,3) al número de positivos por semana Fuente: Elaboración propia 32

Figura 3-6: Distribución empírica de los residuos generados por ARIMA(2,1,3 respecto a la normal).....33

Figura 3-7: Gráfico representativo de la predicción realizada por el modelo 34

Figura 3-8: Gráfico representativo de la tasa de positividad por semanas de forma anual 35

Figura 3-9: Gráfico de la autocorrelación de la serie origina de la tasa de positivos 36

Figura 3-10: Gráfico de la autocorrelación de la serie diferenciada de la tasa de positivos 37

Figura 3-11: Función de autocorrelación de los residuos del modelo ARIMA(2,0,3) aplicado sobre la serie tasa de positividad semanal.....38

Figura 3-12: Gráfico de la distribución de los residuos del modelo ARIMA(2,0,3) aplicado a la tasa de positivos semanal 39

Figura 3-13: Predicción de la tasa de positividad para las siguientes 10 semanas con sus respectivos intervalos de confianza.....40

Figura 3-14: Resultados de Getis-Ord Gi* por provincias en casos humanos 43

Figura 3-15: Resultados de Getis-Ord Gi* a nivel de código postal en la provincia de Madrid.....45

Figura 3-16: Resultados de Getis-Ord Gi* a nivel de código postal en la provincia de Barcelona.....46

Figura 4-1: Imagen de la documentación de nuestra API donde se observan los endpoints desarrollados. Fuente: Captura de pantalla de nuestro backend.....49

Figura 4-2: Captura de la representación visual de los puntos calientes y la temperatura sobre el mapa nacional.....50

Figura 4-3: Representación de la estructura de la aplicación.....51

Figura 4-4: Representación gráfica de la estructura del "front-end".....68

Figura 4-5: Imagen de la pantalla de Loading.....71

Figura 4-6: Imagen de la pantalla de inicio de sesión.....72

Figura 4-7: Imagen de la pantalla del mapa interactivo con el panel lateral.....73

Figura 4-8: Imagen de la pantalla de registro.....74

Figura 4-9: Representación gráfica en formato JSON de la estructura de datos que utiliza la aplicación. Fuente: Elaboración propia.....77

Figura 5-1: Representación visual de la estructura desplegada.....81

Figura 5-2: Configuración servicio base de datos.....82

Figura 5-3: Configuración del servicio de backend.....83

Figure 6-1: Graph of the 2025 pet census.....87

ÍNDICE DE TABLAS

Tabla 2.1: Resumen de variables en los datos de entrada de las pruebas 6

Tabla 2.2: Resumen de las variables del "shapefile" de las geometrías..... 8

Tabla 2.3: Resumen de los datos del censo de mascotas..... 9

Tabla 2.4: Resumen de los datos de casos de humanos.....11

Tabla 2.5: Resumen de las variables devueltas por la API de la AEMET 14

Tabla 2.6: Resumen de los tipos de valores de los datos de mascotas por la cantidad de cada uno.....15

Tabla 3.1: Comparativa entre Getis-Ord y Anselin 22

Tabla 3.2: Pseudocódigo de cómo se calcula la tasa de positividad de los casos de animales 22

Tabla 3.3: Pseudocódigo de cómo se calcula la vecindad para aplicar GetisOrd.....23

Tabla 3.4: Pseudocodigo de cálculo de GetisOrd por cada código postal.....23

Tabla 3.5: Pseudocódigo de la clasificación de código postal como punto caliente o frio 25

Tabla 4.1: Tabla descriptiva de las columnas de los datos de pruebas a mascotas.....59

Tabla 4.2: Tabla descriptiva de las columnas de los datos censales de mascotas 59

Tabla 4.3: Tabla descriptiva de las columnas de los datos meteorológicos.....60

Tabla 4.4: Tabla descriptiva de las columnas de los datos precalculados.....61

Tabla 4.5: Tabla descriptiva de las columnas de los usuarios.....61

Tabla 4.6: Comando para crear proyecto en React con estructura básica 67

Tabla 4.7: Descripción de cada archivo contenido en el directorio “components/”.....70

Tabla 5.1: Comando para construir la aplicación 84

Capítulo 1 - Introducción

1.1 Motivación

La convivencia de mascotas en los hogares es una parte integral de nuestra sociedad y conlleva tanto beneficios como ciertos riesgos, que deben ser tenidos en cuenta para poder convivir de forma adecuada. En España, el censo de mascotas ha ascendido en 2025 a más de 20 millones, incluyendo perros, gatos, peces, aves y otros animales como reptiles y roedores, según datos de la Asociación ANFAAC [1]. La Figura 1-1 representa el censo de mascotas en miles en lo que llevamos de 2025.

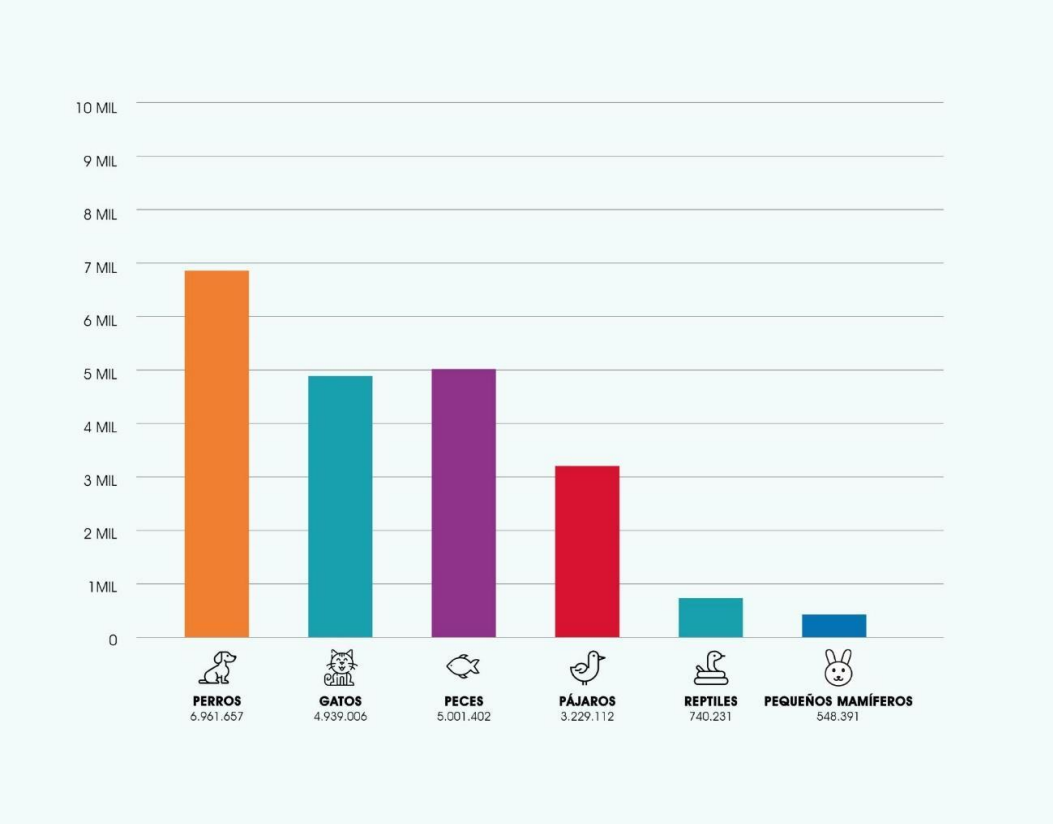

Figura 1-1: Gráfico del censo de mascotas de 2025.

Fuente: Asociación ANFAAC.

Una de las principales preocupaciones causadas por el gran aumento de estos animales de compañía es la posibilidad de que estos animales enfermen y sean transmisores de las mismas enfermedades, lo que generaría oleadas invisibles de animales contagiados. Esta inquietud crece en el caso de que las mascotas puedan propagar enfermedades contagiosas a los seres humanos, especialmente a aquellos con el sistema inmunológico comprometido y generando un problema de salud pública a causa de un seguimiento ineficiente. Por ello, se hace cada vez más necesario algún tipo de herramienta que permita monitorizar los contagios y su evolución a lo largo del tiempo.

1.2 Objetivos

En consecuencia, el objetivo de este TFG es desarrollar una herramienta web intuitiva y que sea útil en el control de la difusión de enfermedades de mascotas, centrada especialmente en Leishmaniasis y Giardia. Para ello, se realizará un análisis espacio-temporal con el fin de identificar patrones y áreas de alta incidencia de enfermedades en mascotas, específicamente perros y gatos que presentan un riesgo significativo para estas personas inmunocomprometidas. Esta herramienta será diseñada teniendo en cuenta su escalabilidad para crecer según vayan evolucionando las necesidades del estudio. Para llevar a cabo el estudio, se contará con distintos conjuntos de datos, entre ellos: datos de diagnóstico clínico proporcionados por Uranovet [2], registros de incidencia de enfermedades en humanos, datos censales de mascotas e información meteorológica suministrada por la AEMET.

Se seguirán esta serie de objetivos:

- Identificar y aplicar algoritmos estadísticos adecuados para realizar el análisis de la incidencia de enfermedades en mascotas, así como su posible estacionalidad.
- Investigar la posible relación entre la aparición de enfermedades en mascotas y su transmisión a humanos, con el objetivo de identificar

patrones comunes, correlaciones y vínculos epidemiológicos que permitan comprender el papel de los animales de compañía en la propagación de ciertas enfermedades a la población humana.

- Desarrollo de un sistema estructurado que permita el almacenamiento y actualización de la información recibida desde distintas fuentes y diseñado para garantizar la escalabilidad ante la incorporación de nueva información relevante para el estudio.
- Desarrollar una interfaz web simple e interactiva que permita visualizar en un mapa de España la incidencia espacio-temporal de diferentes enfermedades en mascotas y humanos, con el fin de facilitar estudios comparativos entre ambas poblaciones.

1.3 Plan de trabajo

Tras la anterior definición de los objetivos de trabajo a continuación se describen los puntos a seguir:

- Estudio de métodos estadísticos adecuados para el análisis espacio-temporal de los datos. Se llevará a cabo un análisis en profundidad para determinar cuál es el enfoque más adecuado para la aplicación. Además, se ampliará con un estudio para determinar si existe estacionalidad en los datos de mascotas.
- Diseño de la estructura de organización de la aplicación, creando una base sobre la que trabajar la parte esencial, permitiendo añadir funcionalidades de forma sencilla.
- Diseño del despliegue con "Docker" y "Docker Compose" para crear contenedores y orquestar la ejecución coordinada de la aplicación.
- Diseño de un modelo de datos relacional y su posterior implementación en un motor de base de datos.
- Limpieza y preparación de los datos recibidos en crudo para su posterior procesamiento con los métodos estadísticos elegidos

- Desarrollar el "backend" de la aplicación, implementando los "endpoints" y funcionalidades necesarias para su funcionamiento.
- Implementar el "frontend" con una interfaz web estructurada en varias capas para representar diferente información.
- Configurar un servidor web encargado de alojar el "frontend" y gestionar la comunicación con el "backend".

Capítulo 2 - Origen y preprocesamiento de los datos

Este capítulo está dedicado a la descripción de los conjuntos de datos utilizados, así como al proceso de limpieza y preprocesamiento aplicado sobre ellos, con el objetivo de prepararlos adecuadamente para la aplicación de los algoritmos requeridos en el análisis.

2.1 Origen y características de los datos

2.1.1 Datos de test a mascotas

Los datos sobre las pruebas realizadas a animales son proporcionados por el laboratorio Uranovet [2], que proporciona un listado, en formato Excel, de los resultados realizados sobre mascotas en clínicas de toda España. Dicho listado, además, incluye toda la información relativa a la prueba y a la mascota.

El conjunto de datos contiene 42.834 pruebas, distribuidas en 1234 códigos postales y registradas entre julio de 2020 y octubre de 2024, lo que proporciona un marco temporal de más de cuatro años para el análisis. Además, en lo que respecta a los resultados predominan los resultados negativos (36.498), seguidos por los positivos (4703) y también se registran categorías menos frecuentes como los resultados inválidos (763 + 175), los dudosos (502), los positivos fuertes (164) y los errores (29).

Cada registro del dataset corresponde con prueba realizada a un animal y cada columna representa una variable. A continuación, en la Tabla 2.1 se presenta un resumen de las columnas más relevantes:

Variable	**Descripción de la variable**
Date	Fecha en la que se realizó la prueba diagnóstica al animal de compañía (perro o gato).

Geo Location	Coordenadas geográficas (latitud y longitud) del lugar exacto donde se llevó a cabo la prueba, generalmente correspondiente a una clínica veterinaria o centro de análisis.
Name Test	Nombre de la enfermedad para la que se realizó la prueba. En este conjunto de datos, las enfermedades registradas son **Leishmania** y **Giardia**.
Value Test	Resultado de la prueba diagnóstica para la enfermedad indicada. Suele indicar si el animal ha dado **positivo, muy positivo** o **negativo** en la detección.
Pet Sex	Sexo del animal analizado, generalmente registrado como **macho** o **hembra**.
Pet Age	Edad del animal en el momento de la prueba, expresada de la siguiente forma: X Año/s. Siendo X el número de años.
Country	País en el que se realizó la prueba al animal.
Postal Code	Código postal que identifica la zona donde se llevó a cabo la prueba.

Tabla 2.1: Resumen de variables en los datos de entrada de las pruebas

Además, se encontraron algunas anomalías que deben tenerse en cuenta antes de la aplicación de los métodos estadísticos. Los años 2022 y 2023 presentan un comportamiento más estable en el número de casos semanales (esto será más relevante cuando se hable de la estacionalidad), lo que los convierte en los periodos más consistentes de la serie. Sin embargo, en diciembre de 2022 se aprecia un repunte anómalo de casos, probablemente asociado a la introducción repentina de un conjunto elevado de registros en un corto intervalo de tiempo. Por otro lado, los años 2020 y 2021 apenas cuentan con casos registrados, lo que limita su valor para el análisis estacional. Por último, los datos de 2024 muestran una volatilidad considerable,

dificultando la identificación de patrones claros. Estas particularidades deben considerarse a la hora de interpretar los resultados de los análisis espacio-temporales y del estudio de la estacionalidad.

2.1.2 Datos geográficos de los códigos postales

Se cuenta con un conjunto de datos geoespaciales en formato "shapefile" que contiene la delimitación territorial de los diferentes códigos postales nacionales y proporciona información geográfica esencial para representarlos. Esto permite vincular cada prueba realizada a un animal con su correspondiente área postal y facilitar el análisis espacial posterior.

Los datos se descargan desde un repositorio en GitHub [3] que aprovecha los datos ofrecidos por el Centro Nacional de Información Geográfica (CNIG) [4] y agrupa todas las geometrías en un solo archivo "shapefile".

Seguidamente, se van a presentar la Tabla 2.2 con una descripción más detallada:

Variable	**Descripción de la variable**
ID_CP	Identificador único de cada registro.
COD_POSTAL	Número del código postal asociado a la geometría.
ALTA_DB	Fecha de incorporación del registro en el shapefile
CODIGO_INE	Código del Instituto Nacional de Estadística (INE) que identifica la unidad a administrativa, como municipios o provincias.
geometry	Campo que contiene la información geoespacial en formato POLYGON. Representa gráficamente los límites de cada código postal sobre el territorio.

Tabla 2.2: Resumen de las variables del "shapefile" de las geometrías

El creador señala diversas consideraciones sobre la fiabilidad del "shapefile" como, por ejemplo, la existencia de un elevado número de códigos postales presentes en el fichero que no aparecen en otras fuentes oficiales como el "ds-códigos-postales-ine", y viceversa, es decir, también hay códigos registrados por el INE que no están representados en este conjunto.

Además, observando los datos se han encontrado códigos postales que abarcan múltiples municipios como el 09640 en Burgos o el 46800 de Xàtiva, que contiene una gran cantidad de polígonos a pesar de corresponder a un único municipio.

Sin embargo, se han revisado estas observaciones y, a pesar de los problemas mencionados, se considera que los datos son una fuente válida y útil para mostrar los códigos postales en el estudio.

2.1.3 Datos censales de mascotas

Se dispone de un conjunto de datos complementario que proporciona información censal sobre la población de mascotas a nivel de código postal. Estos datos fueron facilitados por el CSIC-INIA y se obtuvieron a partir de una estimación basada en la población humana de cada zona junto con el número medio de mascotas por habitante.

El conjunto de datos contiene 10.874 registros, cada uno correspondiente a un código postal distinto, con valores de censo que van desde 0 hasta 29.021,96, y una media de 901,72.

Estos datos, representados en la Tabla 2.3, incluyen diversas variables geográficas y administrativas:

Variable	**Descripción de la variable**
FID_1	Actúa como identificador interno sin relevancia analítica.
COD_POSTAL	Indica el código postal correspondiente a cada área.

Area_cp	Representa la superficie del área asociada al código postal, medida en metros cuadrados.
Censo_mascota_CP	Recoge una estimación del número de mascotas registradas o calculadas para cada código postal.
Cod_provincia y Provincia	Identifican la provincia tanto por su código numérico como por su nombre.
COD_CCAA y CCAA	Especifican la comunidad autónoma a la que pertenece cada registro, tanto por nombre como por código.

Tabla 2.3: Resumen de los datos del censo de mascotas

Esta información resulta especialmente útil para contextualizar espacialmente los datos de enfermedades y para realizar análisis relativos a la densidad de mascotas por área geográfica.

2.1.4 Datos humanos

De igual modo, se nos proporcionó los datos relativos a los casos positivos de humanos con la enfermedad de Leishmaniasis. Estos datos, provenientes de la Red Nacional de Leishmaniasis "ReNLeish" [5] , están anonimizados para preservar la privacidad de los pacientes, pero cuentan con información relevante a su posición geográfica, por lo que son útiles para este estudio.

El conjunto de datos analizado contiene 354 registros distribuidos en 169 códigos postales, abarcando un período desde el 24 de octubre de 2000 hasta el 26 de marzo de 2025 y refleja un número relativamente bajo de casos con una distribución geográfica dispersa. Se evidencia una predominancia significativa de pacientes masculinos (244) en comparación con los pacientes femeninos (109), con una edad media de 51 años. Sin embargo, se observa la presencia de pacientes de diversas edades, desde recién nacidos hasta 91 años. La mayor parte de los pacientes (297)

nacieron en España y, por lo tanto, residen principalmente en este país (343), con escasos casos de proveniencia o residencia en otros países.

A nivel provincial, la concentración más alta de pacientes se observa en Alicante, Madrid y Murcia tanto en su residencia como en la provincia donde se sospecha el contagio según la historia clínica. En lo que respecta a la enfermedad, la forma más prevalente es la leishmaniasis visceral (205 casos), seguida de la leishmaniasis cutánea (116 casos), mientras que otras formas, como la mucocutánea, la ganglionar o el post kala-azar, son mucho menos frecuentes.

A continuación, se describirán los datos de forma más específica:

Variable	**Descripción de la variable**
Record ID	Identificador entero único del registro de cada paciente dentro del sistema.
Data Access Group	Nombre de la institución o grupo que ha generado dicho registro. Un ejemplo sería: H.U. Ramón y Cajal
Sexo del paciente / Has the consent form been signed?	Indica el sexo biológico del paciente (masculino o femenino), tanto en español como en inglés.
Edad del paciente al momento del diagnóstico de este episodio / Patient age at diagnosis (years)	Edad del paciente, en años, en el momento que fue diagnosticado con la enfermedad.
País de Nacimiento / Country of birth	Nombre del país en inglés donde nació el paciente.

País de residencia del paciente en el último año / Country of residence in the last year	Nombre del país en inglés donde residió el paciente en el último año anterior al diagnóstico.
Provincia de residencia del paciente	Nombre de la provincia donde vive actualmente el paciente.
Código postal de residencia del paciente / Patient's postcode	Código postal correspondiente al domicilio del paciente.
Provincia donde se sospecha el contagio por la historia clínica del paciente	Nombre de la provincia en la que el paciente pudo haber sido contagiado.
Tipo de Episodio de Leishmaniasis de este paciente - Type of Leishmaniasis Episode	Clasificación clínica del episodio de Leishmaniasis diagnosticado (visceral, cutánea, mucocutánea…). Tanto en inglés como en español.
Fecha de Diagnóstico / Date of diagnosis	Fecha en la que se realizó la prueba diagnóstica al paciente en formato DD/MM/YYYY.

Tabla 2.4: Resumen de los datos de casos de humanos

2.1.5 Datos meteorológicos

Por último, se consideró necesario la incorporación de datos de temperaturas al tratarse de un posible factor clave en la propagación de enfermades. Para su obtención, la Agencia Estatal de Meteorología (AEMET) dispone de una API con acceso de forma pública al historial de temperaturas en ventanas temporales de 15 días de todas las estaciones meteorológicas distribuidas en territorio nacional.

La incorporación de datos meteorológicos, especialmente los relacionados con la temperatura, representa un valor añadido significativo para el presente estudio. Este factor puede influir directamente en la aparición, persistencia o propagación de determinadas enfermedades, por lo que es necesario tener en cuenta estos datos para poder realizar un estudio más completo y preciso.

La información proporcionada por la AEMET incluye muchos más datos, además de los de temperatura. Esto permite sentar las bases para incorporar en el futuro nuevos factores ambientales que ayuden a comprender mejor la relación entre el entorno y la propagación.

Seguidamente, se presenta en la Tabla 2.5 todas las variables devueltas junto a su descripción:

Variable	**Descripción de la variable**
Fecha	Fecha del día en el que se hizo la medición, en formato AAAA-MM-DD
Indicativo	Código identificador único asignado a cada estación meteorológica por la AEMET.
Nombre	Denominación oficial de la estación meteorológica, normalmente relacionada con su ubicación geográfica.
provincia	Provincia española donde se encuentra ubicada la estación meteorológica.
altitud	Altura de la estación respecto al nivel medio del mar, expresada en metros.
tmed	Temperatura media diaria registrada en la estación, expresada en grados Celsius (°C). Calculada como el promedio de las temperaturas máximas y mínimas del día.

prec	Precipitación total registrada entre las 07:00 h de un día y las 07:00 h del siguiente, medida en milímetros.
tmin	Temperatura mínima registrada durante el día, expresada en grados Celsius (°C).
horatmin	Hora exacta (en formato UTC) en la que se produjo la temperatura mínima.
tmax	Temperatura máxima diaria, en grados Celsius (°C).
horatmax	Hora (UTC) en la que se alcanzó la temperatura máxima.
dir	Dirección de la racha máxima de viento, expresada en decenas de grado. Los valores 99 y 88 indican dirección variable o datos no disponibles, respectivamente.
velmedia	Velocidad media del viento durante el día, en metros por segundo (m/s).
racha	Velocidad máxima del viento registrada durante el día, en metros por segundo (m/s).
horaracha	Hora (UTC) en la que se registró la racha máxima.
sol	Duración de la insolación diaria, medida en horas. Indica la cantidad de radiación solar directa que ha recibido la estación.
presmax	Valor máximo de presión atmosférica al nivel de la estación durante el día, en hectopascales (hPa).
horapresmax	Hora (UTC) en la que se registró la presión atmosférica máxima.

presmin	Valor mínimo de presión atmosférica al nivel de la estación durante el día, en hectopascales (hPa).
horapresmin	Hora (UTC) en la que se registró la presión atmosférica mínima.
hrmedia	Humedad relativa media diaria, expresada en porcentaje (%).
hrmax	Humedad relativa máxima diaria, expresada en porcentaje (%).
horahrmax	Hora (UTC) en la que se alcanzó la humedad relativa máxima.
hrmin	Humedad relativa mínima diaria, en porcentaje (%).
horahrmin	Hora (UTC) en la que se registró la humedad relativa mínima.

Tabla 2.5: Resumen de las variables devueltas por la API de la AEMET

2.2 Preparación y estructuración para el análisis

Antes de llevar a cabo el análisis espacio-temporal destinado a identificar patrones y áreas con alta incidencia de enfermedades en perros y gatos, se realizó una fase previa de preparación y limpieza de datos.

En primer lugar, se homogeneizó la información de los test realizados a las mascotas, eliminando elementos innecesarios y estandarizando formatos.

A continuación, siguiendo la Tabla 2.1, se detalla el proceso aplicado a cada variable:

- Date: El formato de la fecha proporcionado se encuentra entre comillas dobles y con un símbolo al inicio de esta. Para homogeneizarlo, se eliminó toda esa información y dejando solo la fecha y hora en formato **DATETIME.**
- Geo Location: Durante el proceso de homogeneización de los datos, se tuvo en cuenta esta variable incluida en el conjunto original. Aunque no

se realizaron transformaciones específicas, se mantuvo en esta fase con el objetivo de valorar su posible utilidad en el análisis posterior. Finalmente, se optó por descartarla del estudio principal ya que el análisis espacial se llevó a cabo mediante códigos postales.

- Name Test: Para el nombre de la enfermedad para la que se hizo prueba se verificó que no tuvieran faltas de ortografía ni inconsistencias para mantener la uniformidad.
- Value Test: Aunque en la homogeneización no se definió cómo había que tratar los casos positivos y negativos, ya que dicha decisión se ideó una vez elegido el algoritmo estadístico, si se descartó las pruebas cuyo resultado dio Dudoso, Error, Inválido o Invalido, ya que se consideraron no concluyentes. De este modo el dataset quedó restringido a los valores Negativo, Positivo y Positivo fuerte que aportan información relevante y de valor. A continuación, se presentan los diferentes resultados por la cantidad de ellos en los datos:

Resultado	**Cantidad**
Negativo	36.498
Positivo	4.703
Invalido	763
Dudoso	502
Inválido	175
Positivo fuerte	164
Error	29

Tabla 2.6: Resumen de los tipos de valores de los datos de mascotas por la cantidad de cada uno

- Pet Sex: Para el sexo del animal, igual que para el nombre, se verificó que no tuvieran faltas de ortografía ni inconsistencias entre los dos posibles valores: “macho” o “hembra”. Este campo no se tendrá en cuenta.
- Pet Age: Como se muestra en la Tabla 2.1, los valores de la variable edad siguen el patrón “X Año/s” donde X representa el número de años. Se eliminó el texto y se conservó el valor numérico para facilitar el tratamiento de los datos. En los casos en los que la edad se expresaba en meses, se asignó el valor 0 años para mantener la coherencia y simplificar el análisis. Este campo no se tendrá en cuenta.
- Country: Durante el proceso, se revisaron los valores para asegurar consistencia en la nomenclatura "Spain" en un único valor estándar.
- Postal Code: Se validó que los valores tuvieran un formato coherente de cinco dígitos numéricos y se eliminaron las filas completas donde hubiera valores vacíos. Para los municipios cuyos códigos postales estaban compuestos por menos de cinco dígitos se les añadió un cero a la izquierda con el fin de mantener la longitud estándar.

Tras la fase de preprocesamiento, el conjunto de datos experimentó una ligera reducción, alcanzando un total de 41 201 registros y 1216 códigos postales, manteniendo prácticamente el mismo rango temporal, comprendido entre julio de 2020 y octubre de 2024.

En cuanto a los datos geográficos de los municipios, solo se tuvieron en cuenta las variables del código postal y la geometría, ya que son las más significativas. Se verificó que el código postal siguiera el formato de cinco dígitos numéricos. Para la geometría, se procedió a la representación visual de todos los polígonos presentes con el propósito de verificar que la mayoría de ellos estuvieran incluidos.

Siguiendo con los datos censales distribuidos por código postal, se detectaron tres variables especialmente importantes para el análisis: el propio código postal, el área geográfica correspondiente a cada uno y el censo total de mascotas registrado en dicha zona. Durante la fase de limpieza y preprocesamiento, se confirmó que los valores de estas tres variables estuvieran representados de forma numérica. En particular, se revisó que los códigos postales tuvieran cinco dígitos, que las áreas no tuvieran valores nulos o inconsistentes, y que las cifras del censo de mascotas estuvieran dentro de un rango razonable. Tras el preprocesamiento, el dataset mantuvo 10.874 filas y códigos postales, corrigiendo un valor nulo en el censo y ajustando el mínimo a 0,30, dejando los datos consistentes y listos para análisis.

A continuación, con los datos humanos se hizo énfasis, más que en la información relativa al paciente, al lugar donde se hizo la prueba y a la fecha en la que se realizó. Se validaron los códigos postales para asegurarse de que siguieran el formato estándar de cinco dígitos; los registros que no cumplían esta condición fueron descartados del conjunto de datos. De igual manera, se confirmó que las fechas se ajustaran al formato establecido (YYYY-MM-DD). Las filas que tenían valores ausentes o formatos incorrectos en este campo también se eliminaron para preservar la coherencia y fiabilidad del conjunto de datos. Una vez realizado el preprocesamiento de los datos, el conjunto se redujo de 354 a 277 registros y los códigos postales distintos pasaron de 169 a 163, mientras que el rango temporal se mantuvo igual, desde octubre de 2000 hasta marzo de 2025.

Por último, en el caso de los datos meteorológicos, se prestó especial atención a las variables de fecha, indicativo y temperatura media diaria (Tmed). Primero se comprobó que no existieran valores nulos en estos campos borrándolos en caso de que, si y siguiendo se estandarizó su formato, para lo cual se convirtieron las fechas al patrón YYYY-MM-DD, se limpió el campo indicativo de caracteres no deseados y se aseguraron los valores de tmed como datos numéricos con unidad en grados Celsius.

Capítulo 3 - Análisis espacio-temporal

El objetivo de este capítulo es aplicar métodos estadísticos orientados al análisis espacio-temporal con el fin de identificar patrones y áreas con alta o baja concentración de casos positivos de enfermedad. Para ello, se evaluarán dos enfoques ampliamente utilizados en el análisis geoespacial: el estadístico Getis-Ord [6] (Getis-Ord Gi*), implementado en ArcGIS, y el análisis de clústeres espaciales de Anselin [7] (Anselin Local Moran's I). Ambos métodos permiten detectar concentraciones significativas de valores altos o bajos (conocidas como clústeres calientes y fríos), si bien se diferencian en su lógica estadística y en el tipo de información que proporcionan sobre la autocorrelación espacial.

Su detección es especialmente relevante en epidemiología, ya que señala las áreas donde la incidencia de casos positivos es significativamente mayor que en otras zonas, lo que indica posibles focos de infección. La identificación de estos clústeres permite priorizar la vigilancia, orientar las intervenciones sanitarias y planificar estrategias de prevención de manera más efectiva. De manera complementaria, las zonas frías señalan áreas con baja incidencia, lo que puede aportar información sobre factores protectores o menor exposición a la enfermedad.

En el contexto de las enfermedades de las mascotas y las zoonosis humanas, en nuestro estudio concretamente la Leishmaniasis o la Giardia, el análisis de las zonas calientes proporciona información clave sobre la distribución espacial de los casos, lo que contribuye al estudio estadístico y al desarrollo de herramientas de visualización que faciliten la adopción de medidas para el control y la prevención de futuros brotes.

3.1 Métodos estadísticos candidatos

3.1.1 Método estadístico Getis-Ord

El método estadístico Getis-Ord Gi* [6] es una técnica de análisis espacial que se utiliza para identificar agrupaciones significativas de valores altos o bajos dentro de un

conjunto de datos que tienen referencias geográficas. En este contexto, las agrupaciones de valores altos se denominan puntos calientes (hotspots) y las de valores bajos, se denominan puntos fríos (coldspots). Evalúa, para cada ubicación, si los valores de la variable analizada (tasa de positivos, siendo esta el número de positivos entre el número de casos totales) en esa ubicación y en sus vecinas son más altos o bajos de lo que se esperaría por azar.

Aplicado a nuestro caso de estudio, una zona caliente de casos positivos de enfermedad en mascotas significa que en esa área hay un número de animales infectados notablemente superior al esperado, considerando la distribución global de los casos. No se trata solo de un número alto absoluto, sino de una concentración estadísticamente significativa, es decir, que es improbable que se deba al azar.

Para que un punto caliente sea considerado estadísticamente significativo por este método, no es suficiente con que la ubicación analizada presente un valor alto, sino que además debe estar rodeada de ubicaciones vecinas con valores igualmente altos. De manera similar, un punto frío significativo indica un valor bajo rodeado de otros valores bajos.

El funcionamiento del estadístico Gi* se basa en la comparación entre:

1. La suma local de los valores de una ubicación y sus vecinos, ponderada según una matriz de pesos espaciales.
2. La suma global de los valores de todas las ubicaciones del conjunto.

Si la diferencia entre ambas sumas es mayor de lo que se podría esperar por casualidad, el resultado se considera estadísticamente significativo, es decir, se obtendrá una puntuación Z muy alta para un punto caliente o muy baja para un punto frío.

- Puntuaciones Z positivas y significativas → agrupaciones de valores altos (*hotspots*).
- Puntuaciones Z negativas y significativas → agrupaciones de valores bajos (*coldspots*).

- Valores P bajos → mayor confianza en que el patrón no se debe al azar.

3.1.2 Método estadístico Anselin

El análisis de clúster y valores atípicos, también conocido como el estadístico Anselin Local Moran's I [7], permite identificar puntos calientes, puntos fríos y valores atípicos espaciales que tienen un significado estadístico dentro de un conjunto de datos con referencias geográficas, siendo nuestro caso el de estudio de casos positivos de enfermedades en mascotas.

La herramienta utiliza el índice local de Moran que calcula para cada entidad espacial un valor I local, junto con su puntuación Z y un valor P (pseudo p-value):

- **Puntuación Z**: Muestra la cantidad de desviaciones estándar que presenta el valor observado en comparación con lo esperado, bajo un patrón aleatorio. Una puntuación Z alta y positiva señala que un valor elevado está rodeado de valores también elevados (HH), mientras que una puntuación Z baja y negativa indica que un valor bajo está rodeado de valores igualmente bajos (LL). Los valores extremos, ya sean positivos o negativos, son más probables de ser estadísticamente significativos, es decir, de tener un impacto más notable en los resultados de la investigación.
- **Pseudo P-value**: mide la probabilidad de que el patrón observado sea producto del azar. Si el valor P es bajo (por ejemplo, menor que 0,05) muestra que la agrupación o el valor atípico no es por casualidad.

Estos valores sirven como medidas de la importancia estadística de los patrones observados. La clasificación del tipo de clúster o valor atípico (en inglés, COType) distingue entre:

- **HH** (*High-High*): Indica que un área con un número alto de casos positivos está rodeada de vecinos que también presentan valores altos, señalando un clúster caliente de infección.

- **LL** (*Low-Low*): Corresponde a un lugar con pocos casos rodeado de vecinos con bajas incidencias, identificando un clúster frío.
- **HL** (*High-Low*): Señala un área con un número alto de casos positivos, pero rodeada de vecinos con incidencias bajas, lo que lo convierte en un valor atípico alto.
- **LH** (*Low-High*): Corresponde a un área con un número bajo de casos positivos, pero rodeada de vecinos con valores altos, identificándose como un valor atípico bajo.

Este enfoque no solo permite identificar las concentraciones y anomalías, sino que también ofrece una cuantificación de su relevancia estadística. Esto proporciona un contexto más completo para interpretar los patrones espaciales.

3.1.3 Justificación e implementación del algoritmo

A continuación, se presentan los puntos donde cada algoritmo destaca:

Elemento	**Getis-Ord Gi***	**Anselin Local Moran's**
Tipo de patrón	Agrupaciones significativas en valores altos (hotspots) o valores bajos (coldspots).	Agrupaciones en valores altos (HH) y bajos (LL) y en valores atípicos: alto rodeado de bajos (HL) y bajo rodeado de altos (LH).
Nivel de detalle	Enfocado en dos categorías principales.	Distingue entre clústeres y valores atípicos (outliers).
Facilidad interpretación	Resultados fáciles de explicar y representar.	Interpretación algo más compleja por haber más categorías.

Sensibilidad a valores extremos	Menos influido por outliers aislados.	Identifica outliers incluso sin estar en un patrón fuerte.

Tabla 3.1: Comparativa entre Getis-Ord y Anselin

En este estudio se optó por utilizar el estadístico Getis-Ord Gi* en lugar de Anselin Local Moran's I debido a que ofrece una interpretación más directa y sencilla. Mientras que Moran's I clasifica los resultados en varias categorías (HH, LL, HL, LH), lo que exige un análisis más detallado, Getis-Ord permite identificar de forma inmediata las zonas con concentraciones altas (hotspots) o bajas (coldspots). Además, presenta menor sensibilidad a valores atípicos aislados, destacando únicamente patrones espaciales consistentes. Estas características lo convierten en una herramienta especialmente adecuada cuando el objetivo principal es detectar de manera clara y rápida las áreas críticas de contagio.

3.2 Aplicación en los datos de mascotas

3.2.1 Análisis espacial de los casos

Tras seleccionar el método Getis-Ord Gi*, se procedió a implementarlo utilizando la información geográfica y censal disponible. En primer lugar, se aplicó la estadística Getis-Ord Gi* al conjunto completo de datos, agrupando los casos positivos por código postal y normalizándolos con el censo correspondiente. Para ello, se calcularon los positivos y el total de pruebas en cada área y, a partir de ahí, la tasa de positividad:

```
Para cada código postal:
    contar número de tests positivos
    contar número total de tests
    calcular tasa de positividad = positivos / censo
```

Tabla 3.2: Pseudocódigo de cómo se calcula la tasa de positividad de los casos de animales

Posteriormente, se construyó la matriz de pesos espaciales mediante el criterio de contigüidad tipo Queen [8], donde dos áreas se consideran vecinas si comparten al menos un vértice o un lado:

```
Crear matriz de vecinos (Queen):
    si dos códigos postales comparten frontera o vértice → son vecinos
Aplicar Getis-Ord Gi* usando la tasa de positividad y la matriz de vecinos
```

Tabla 3.3: Pseudocódigo de cómo se calcula la vecindad para aplicar GetisOrd

Finalmente, se almacenaron en el dataset los valores resultantes, que permiten interpretar el patrón espacial:

```
Para cada código postal:
    calcular Z-Score (intensidad del patrón espacial)
    calcular p-value (significancia estadística)
```

Tabla 3.4: Pseudocodigo de cálculo de GetisOrd por cada código postal

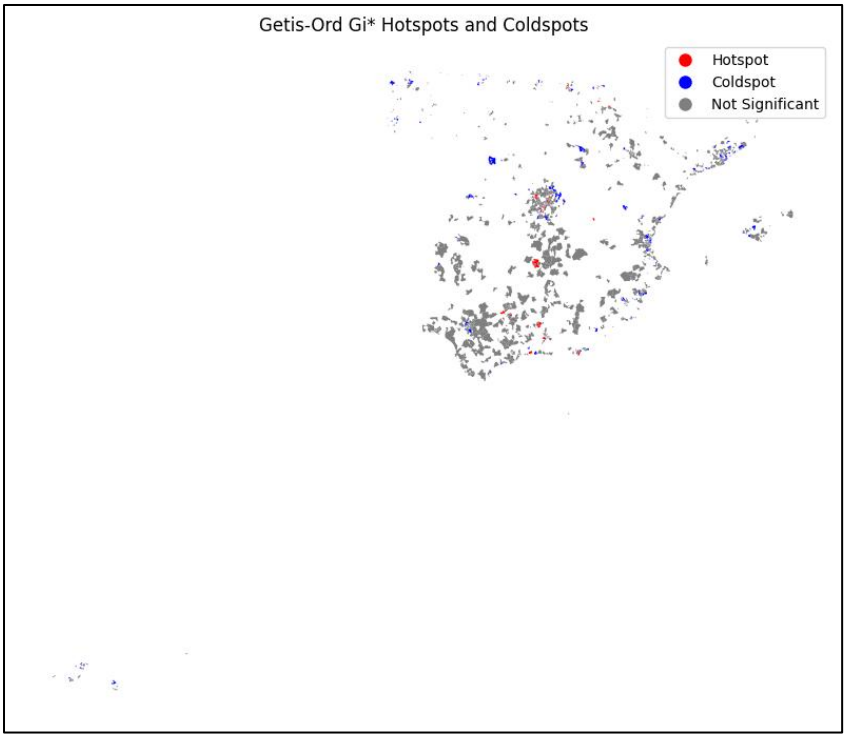

Figura 3-1: Resultados de la aplicación de GetisOrd de forma global a los datos de mascotas

Fuente: Elaboración propia

La Figura 3-1 muestra el resultado del análisis de *Getis-Ord Gi** aplicado al conjunto total de datos de positividad. Se observan zonas puntuales en rojo que corresponden áreas donde se concentran tasas de positividad significativamente altas, y zonas en azul que representan valores significativamente bajos. La gran mayoría de los códigos postales aparecen en gris, lo que indica que no se ha detectado significancia estadística. Este patrón revela que, en general, los casos se encuentran dispersos, pero con algunos núcleos que destacan como focos de interés epidemiológico.

Para obtener esta clasificación, se aplicó la siguiente lógica:

```
Para cada zona:
    Si (Z > 0) y (p < 0.05):
        Clasificar como "Hotspot"
    Sino, si (Z < 0) y (p < 0.05):
        Clasificar como "Coldspot"
    En caso contrario:
        Clasificar como "Not Significant"
```

Tabla 3.5: Pseudocódigo de la clasificación de código postal como punto caliente o frio

Este procedimiento asigna a cada zona una categoría en función de su puntuación *Z* y su *p-value*, donde si p-value supera el nivel de confianza (< 0.05) las áreas se representarán como focos críticos.

Es importante señalar que este resultado corresponde al análisis realizado sobre la totalidad de los datos disponibles. Sin embargo, el algoritmo de Getis-Ord se implementa de forma flexible en ventanas temporales de *X* días, donde *X* es definido por el usuario, permitiendo así explorar la evolución de los patrones espaciales desde la fecha actual hasta el inicio de la serie temporal.

3.2.2 Estacionalidad de los datos

Con el fin de evaluar el comportamiento temporal de los datos y explorar su potencial para predecir futuros brotes, se realizó un análisis de estacionalidad en la enfermedad Leishmaniosis, ya que posee más datos que de Giardia. El objetivo principal del estudio es identificar patrones recurrentes en los datos proporcionados para poder predecirlos. Así, se pretendía determinar si dichos patrones podían constituir una base sólida para generar modelos predictivos que ampliaran las funcionalidades de la herramienta.

Para la realización del estudio de estacionalidad se optó por utilizar dos modelos estadísticos dependiendo de si la serie temporal presenta estacionalidad o no.

En el caso de series sin estacionalidad significativa, se utilizará el método ARIMA(p, d, q) [9] [10] [11](AutoRegressive Integrated Moving Average), cuyo objetivo busca capturar patrones en los valores pasados para hacer predicciones sobre valores futuros. ARIMA se compone de tres partes principales:

- p (AR, Autorregresivo): indica el numero valores pasados de la serie para predecir los futuros.
- d (I, Integrado): indica el número de veces que se hace la diferencia la serie para hacerla estacionaria.

- q (MA, Media Móvil): indica el número de errores de predicciones pasadas para mejorar la predicción.

Asimismo, es posible automatizar la selección de los parámetros p, d y q mediante herramientas como autoARIMA, que realizan un ajuste sistemático del modelo en busca de la combinación que minimice criterios de información estadísticos, como el AIC (Akaike Information Criterion).

Cabe señalar que, para aplicar correctamente el modelo ARIMA, la serie no tiene por qué ser estacionaria inicialmente pero el modelo requiere que los datos se vuelvan aproximadamente estacionarios durante el proceso de modelado. Esto se logra mediante la diferenciación (restando cada valor de su anterior), que es precisamente la componente "I" (Integrado) del modelo ARIMA. Para determinar si la serie necesita diferenciación, suele utilizarse la prueba de Dickey-Fuller aumentada (ADF), que proporciona un valor p. Si este valor es superior a 0,05, la serie no es estacionaria y hay que aplicar las transformaciones necesarias para que alcance la estabilidad y poder modelarla adecuadamente.

Por otro lado, cuando la serie presenta una estacionalidad muy marcada, es decir, patrones repetitivos en intervalos de tiempo fijos, es más adecuado utilizar el modelo SARIMA [9] [10] [11](*Seasonal ARIMA*). Este modelo es una extensión del ARIMA que incorpora parámetros diseñados específicamente para capturar la dinámica estacional de la serie y se expresa mediante la siguiente notación: SARIMA(p, d, q)(P, D, Q, s), donde los parámetros (p, d, q) corresponden a la parte no estacional del modelo, igual que en ARIMA. Mientras que (P, D, Q) corresponden a los términos estacionales de autorregresión, diferenciación y media móvil, y "s" indica la longitud del período estacional, es decir, el número de pasos de tiempo después del cual el patrón se repite.

En definitiva, el modelo ARIMA es apropiado para series temporales que no presentan estacionalidad o en las que esta no es significativa. Por otro lado, el modelo SARIMA es más apropiado cuando la serie contiene patrones estacionales claramente definidos.

- **Análisis del número de positivos por semana del año**

Para estudiar la posible estacionalidad de los casos de leishmaniasis y evaluar su potencial predictivo mediante modelos de series temporales, fue necesario transformar los datos brutos de las pruebas en una estructura adecuada para el análisis. El estudio se llevó a cabo utilizando los datos de la Tabla 2.1 y aplicando una transformación previa sin afectar a la validez de estos. Las transformaciones fueron las siguientes:

- Eliminación de las columnas pet sex y pet age.
- Separación de la columna geolocation en dos nuevas columnas: latitud y longitud.
- Creación de la columna cod_prov, correspondiente al código numérico de la provincia asociada al código postal.
- Definición de la columna date como índice temporal del dataset.
- Reestructuración de la columna Valor (antes Value Test):
    - Los valores Inválido, Dudoso, Inválido y Error se unificaron bajo la categoría Resto.
    - El valor Positivo fuerte se integró en la categoría Positivo.

Sabiendo esto, primero se creó una serie temporal del número de casos por cada mes, pero se descartó continuar con ella ya que la carencia y diferencia de los datos los primeros años de 2020 y 2021 hacían complicado la aplicación del método.

Por tanto, se creó una serie temporal semanal, ya que esta frecuencia ofrecía un equilibrio entre la granularidad temporal y la suavidad de la serie, lo que facilitaba la detección de patrones recurrentes a lo largo del año.

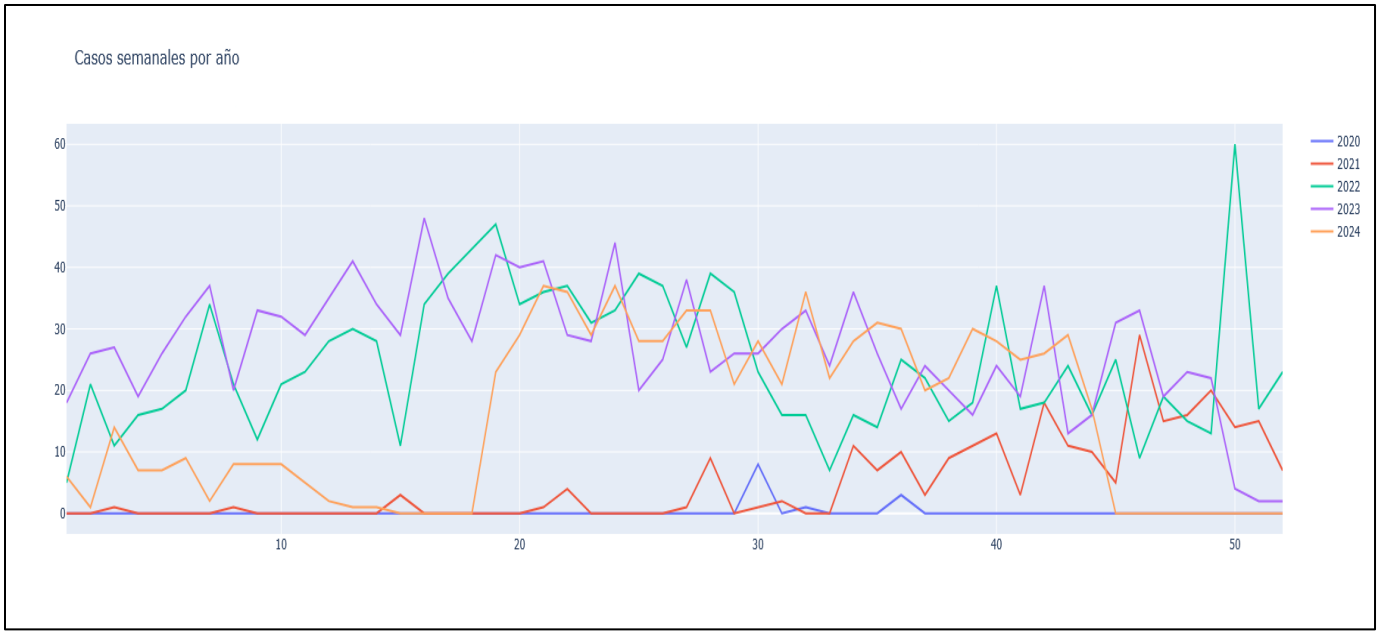

Figura 3-2: Gráfico del número de casos positivos de Leishmaniasis semanales por año

Fuente: Elaboración propia

La Figura 3-2, que representa la serie temporal, refleja la evolución semanal de los casos positivos desde julio de 2020 hasta octubre de 2024. A partir de esta serie se aplicaron pruebas de estacionalidad y modelado con ARIMA.

Estudio de estacionalidad y autocorrelación

El primer paso fue evaluar si la serie temporal mostraba patrones estacionales o dependencias significativas en el tiempo, para ello se realizaron las siguientes pruebas.

Primero, el test ADF (Dickey-Fuller Aumentada) se aplicó a la serie "week" arrojando un resultado estadístico ADF de -2.414806 y un p-value de 0.137601. Al ser p-value mayor que 0.05, indica que la serie no es estacionaria, es decir, la media y la varianza cambian con el tiempo.

Seguido se aplicó la Autocorrelación (ACF) y la Autocorrelación Parcial (PACF). La autocorrelación mide la correlación entre una secuencia y su versión desplazada en el tiempo, se hará un análisis de autocorrelación para ver si los datos se correlacionan con la misma serie desplazada una semana, dos semanas, tres semanas, etc. Quizá podríamos observar que, si en un año X el calor llega más tarde, los casos de

leishmaniasis también aparecen más tarde en el año Y. Se observaría estacionalidad anual si en la semana 52 (más o menos) se produjera un pico alto. A continuación, se presentan los dos gráficos (ACF y PACF):

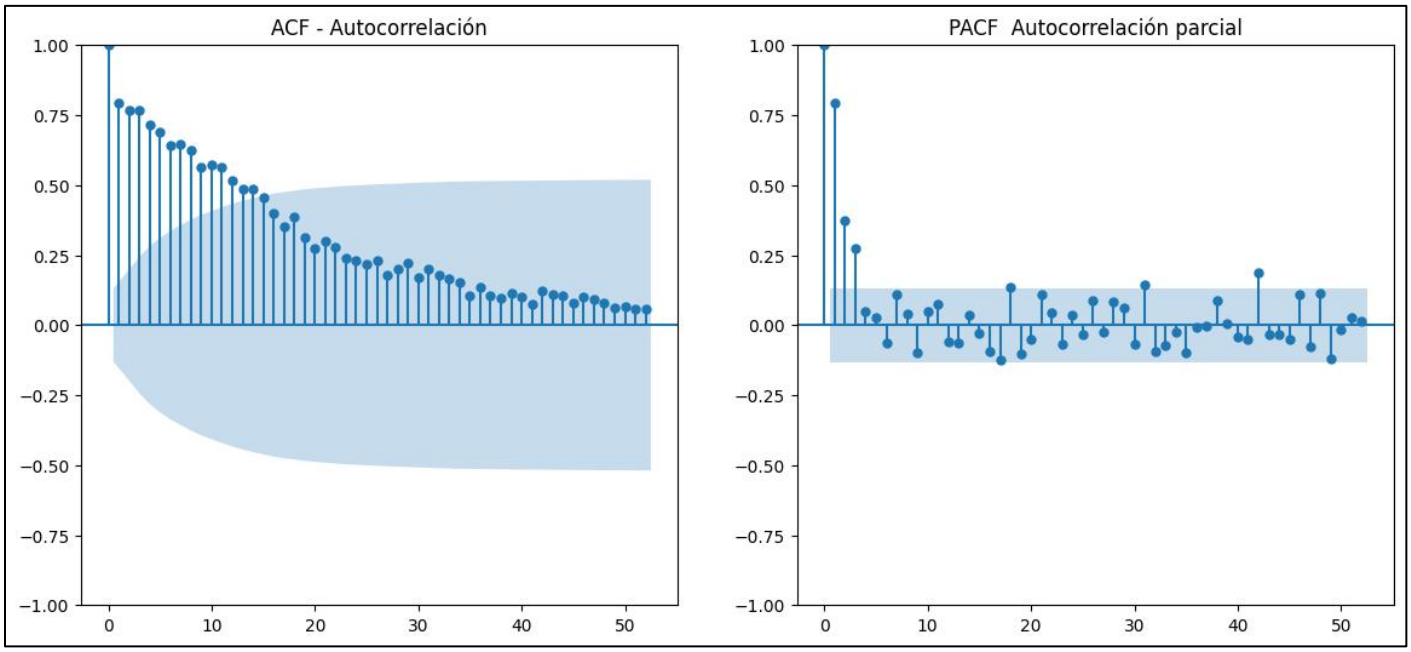

Figura 3-3: Gráfico representativo de la autocorrelación de los datos de Leishmaniasis

Fuente: Elaboración propia

En el gráfico de autocorrelación, los valores del eje X representan retrasos en semanas y el eje Y indica el coeficiente de autocorrelación, que está limitado entre -1 y 1.

La primera línea refleja la correlación de la serie consigo misma, que siempre será 1 porque la correlación entre un valor y si mismo siempre lo es, mientras que las siguientes representan la relación con versiones pasadas de la serie, con la serie con una semana de retraso y así sucesivamente. La zona azul alrededor del eje X permite identificar si las correlaciones observadas son significativas desde el punto de vista estadístico. En este análisis se observó que la correlación solo era fuerte con los valores más cercanos en el tiempo, lo que significa que el número de positivos de una semana depende principalmente del de la semana anterior y, en muy menor medida, del de dos semanas

atrás. Esta conclusión se refuerza al observar el gráfico de autocorrelación parcial, en el que la dependencia se concentra claramente en el primer rezago.

Por último, al verse que la serie no muestra estacionalidad visible, se aplicó una diferenciación a la serie permitiendo hacerla estacionaria. Este hecho fue confirmado por la prueba ADF sobre la serie diferenciada, que arrojó un valor estadístico de -17.445696 y un p-value menor a 0.001. Sin embargo, al analizar nuevamente los gráficos (Figura 3-4) de autocorrelación y autocorrelación parcial, los valores se situaban cercanos a cero y dentro de los intervalos de confianza, evidenciando un comportamiento mucho más aleatorio y sin indicios claros de patrones estacionales de medio o largo plazo.

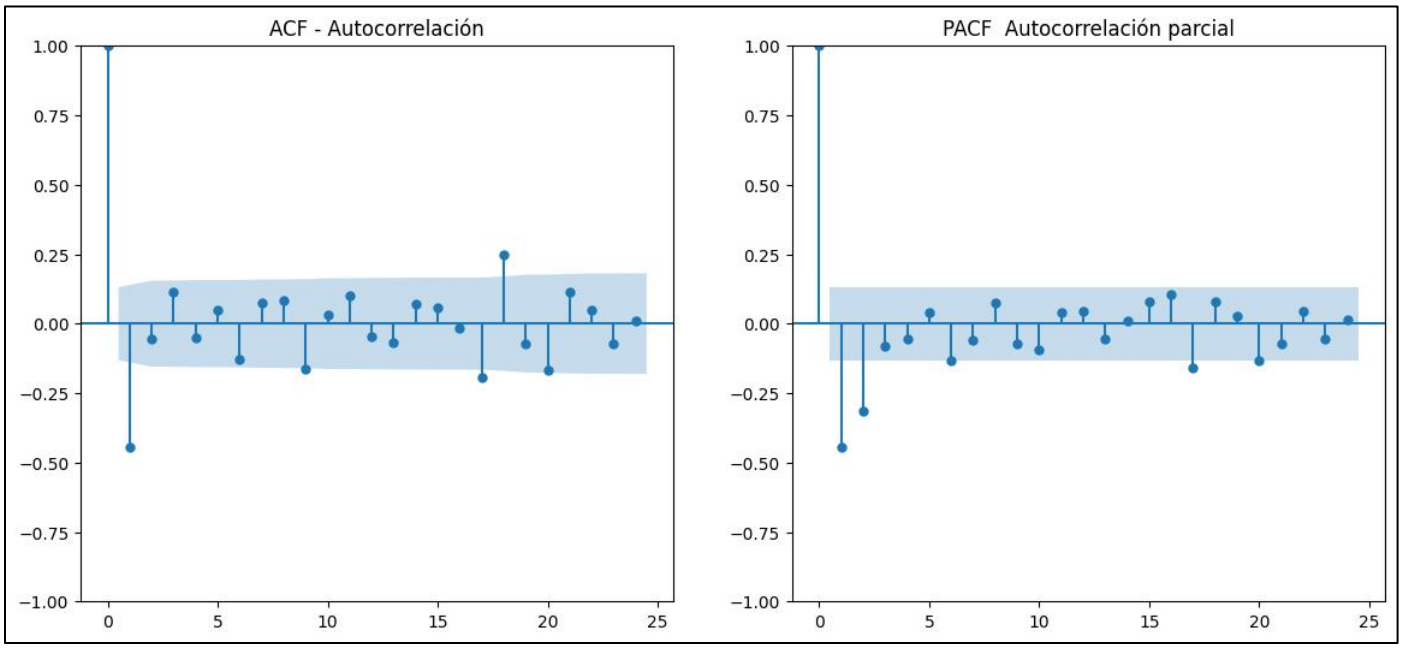

Figura 3-4: Gráfico representativo de la autocorrelación de los datos de Leishmaniasis tras hacer una primera diferencia

Fuente: Elaboración propia

Como conclusión de esta parte, los resultados obtenidos afirman que la serie temporal de positivos semanales carece de una estacionalidad clara. El comportamiento observado refleja una dependencia al corto plazo, limitada a la semana anterior o como mucho a la segunda semana, y no se observan periodicidad de picos, lo que dificulta la construcción de modelos predictivos fiables. De hecho, la

mejor aproximación a una predicción en este caso sería considerar que el valor más probable de una semana es similar al de la semana inmediatamente anterior.

Modelado con ARIMA

Siguiendo con el estudio, a continuación, se va a explorar la capacidad predictiva de la serie, y para ello se aplicó el modelo ARIMA apropiado para series no estacionarias tras diferenciación. Para ello se siguieron los siguientes pasos:

Selección del modelo: se realizó una búsqueda para encontrar los mejores hiperparámetros (p, d, q) para ARIMA. El mejor modelo, según el Criterio de Información de Akaike (AIC), fue ARIMA(2, 1, 3) con un valor AIC de1537.07.

- p = 2: número de retardos autorregresivos (AR). El modelo usa las dos observaciones previas de la serie diferenciada.
- d = 1: número de diferenciaciones necesarias para hacer la serie estacionaria (I). Se aplicó una diferenciación de 1 para estacionarizar la serie.
- q = 3: número de términos del promedio móvil (MA). Incorporación de los errores (residuos) de las tres últimas predicciones.

Una vez obtenido el modelo, se procedió a analizar exhaustivamente sus residuos con el objetivo de evaluar su validez y comprobar si cumplía los supuestos principales de un modelo de series temporales.

Para empezar, se examinó la normalidad de los residuos mediante la construcción de un histograma acompañado de una estimación de densidad de Kernel (KDE) y la aplicación de la prueba estadística de Shapiro-Wilk. El resultado de la prueba arrojó un p-value de 0.0000, muy inferior al umbral de significancia de 0.05, por lo que se puede rechazar la hipótesis nula de que los datos siguen una distribución normal. Este hecho se confirmó mediante la inspección visual del histograma (Figura 3-5), en el que se observó una clara desviación respecto a la forma acampanada característica de una distribución normal, con picos pronunciados y cierta asimetría.

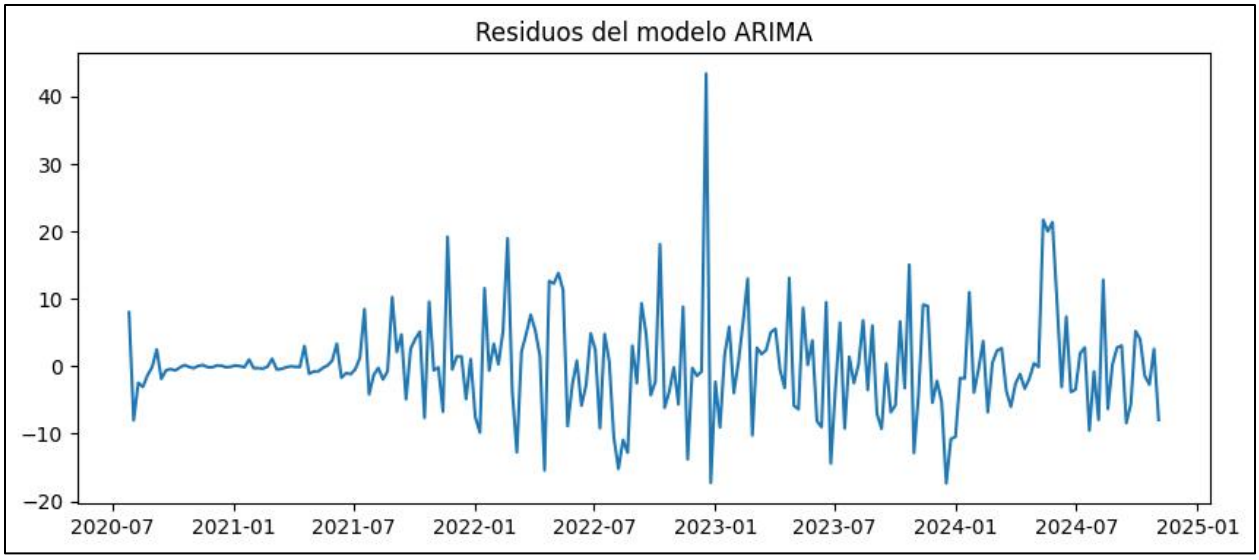

Figura 3-5: Histograma de los residuos tras haber aplicado ARIMA(2,1,3) al número de positivos por semana
Fuente: Elaboración propia

Para reforzar este análisis, se comparó la distribución teórica normal con la distribución empírica de los residuos mediante un gráfico de densidad. En la Figura 3-6, la distribución de los residuos muestra una forma general que se aproxima a la normalidad, con una concentración central alrededor de cero y una dispersión relativamente simétrica hacia ambos lados. Sin embargo, se aprecian ciertas anomalías o desviaciones, como colas más alargadas de lo esperado y la presencia de valores atípicos en los extremos. Estos elementos indican que, aunque los residuos siguen en gran medida un comportamiento cercano a la distribución normal, no cumplen de manera estricta con esta condición y puede afectar parcialmente la validez de los supuestos del modelo ARIMA.

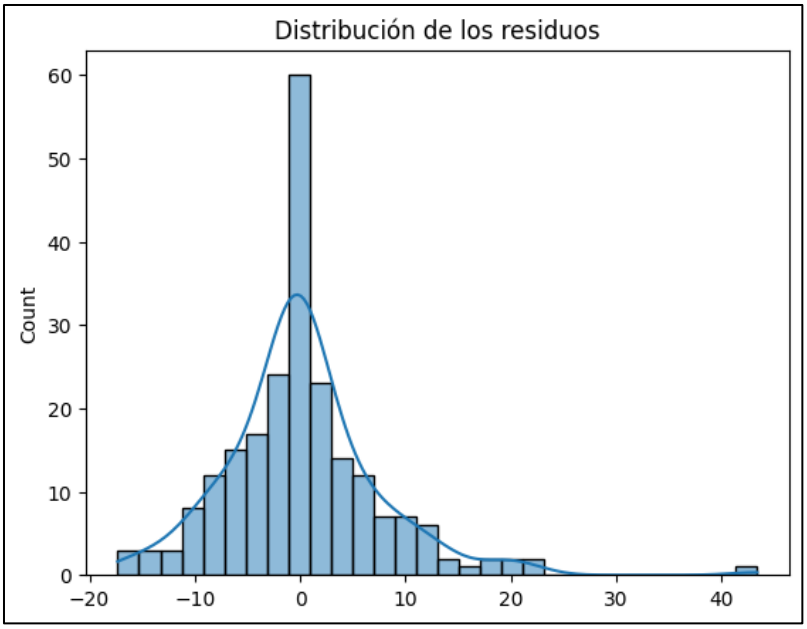

Figura 3-6: Distribución empírica de los residuos generados por ARIMA(2,1,3 respecto a la normal)

Fuente: Elaboración propia

En segundo lugar, se evaluó la autocorrelación de los residuos mediante la prueba de Ljung-Box, aplicada con un retraso de 52 semanas. El p-value obtenido fue 0.1004, superior a 0.05, lo que indica que no hay evidencia estadísticamente significativa de autocorrelación en los residuos. Este resultado es positivo, ya que indica que el modelo ha capturado la mayor parte de la dependencia lineal presente en la serie original.

Siguiendo con el proceso, se investigó la posible presencia de heterocedasticidad en los residuos mediante la prueba ARCH (Autoregressive Conditional Heteroskedasticity). El valor p obtenido fue de 0.4022, superior al nivel de significancia del 5 % y sugiere que no hay pruebas suficientes para afirmar que la varianza condicional de los residuos cambie. En términos prácticos, este resultado indica que la varianza de los errores se mantiene estable con el tiempo. Sin embargo, esta conclusión contrasta parcialmente con la observación del gráfico de residuos, en el que se aprecian episodios aislados de mayor volatilidad o "explosiones" de varianza. Este hecho podría indicar la presencia de dinámicas más complejas en la estructura de los

errores, lo que conllevaría la necesidad de considerar modelos alternativos o más sofisticados para una descripción más precisa de la variabilidad.

Por último, se ajustó el modelo a una parte de la serie temporal (excluyendo las últimas diez semanas) para realizar predicciones y compararlas con los datos reales. Las predicciones mostraron una tendencia a aplanarse con intervalos de confianza relativamente amplios, sobre todo a medida que se extendía el horizonte de pronóstico. Esto refleja la alta variabilidad intrínseca de la serie y las limitaciones del modelo ARIMA para capturar esta aleatoriedad. Esto se ve reflejado en la Figura 3-7.

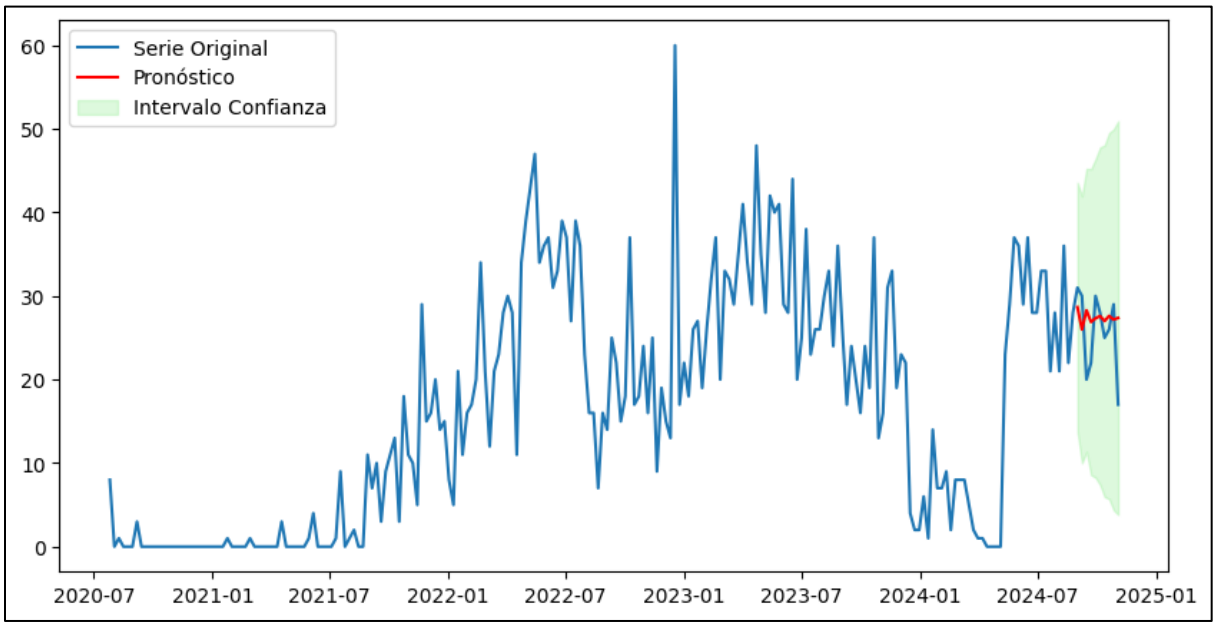

Figura 3-7: Gráfico representativo de la predicción realizada por el modelo

Fuente: Elaboración propia

En conclusión, aunque el modelo ARIMA(2,1,3) fue el que obtuvo el mejor resultado según el AIC, el análisis de los residuos revela ciertas limitaciones. La ausencia de autocorrelación es positiva, pero la falta de normalidad y la presencia de periodos con varianza cambiante indican que el modelo no refleja completamente la dinámica de la serie. Además, las anomalías presentes en los datos, comentadas en el punto **Datos de test a mascotas**, dificultan la obtención de conclusiones sólidas. Esto sugiere que, si bien el modelo ARIMA permite describir algunas dependencias temporales y apuntar cierta tendencia, la capacidad predictiva es limitada y se podría requerir una

depuración más cuidadosa de los datos para abordar la aleatoriedad y variabilidad observadas.

- **Análisis de la tasa de positividad semanal por año**

De forma complementaria, se estudió la serie temporal de la tasa de positivos semanales. Los datos fueron los mismos que los que se usaron en el estudio de casos totales positivos, pero calculando la tasa de positividad. La tasa es la proporción de resultados positivos respecto al total de pruebas realizadas. Posteriormente, se construyó la serie semanal, que constituye la base de este nuevo análisis. La Figura 3-8 refleja cómo se ve representada en un gráfico la serie.

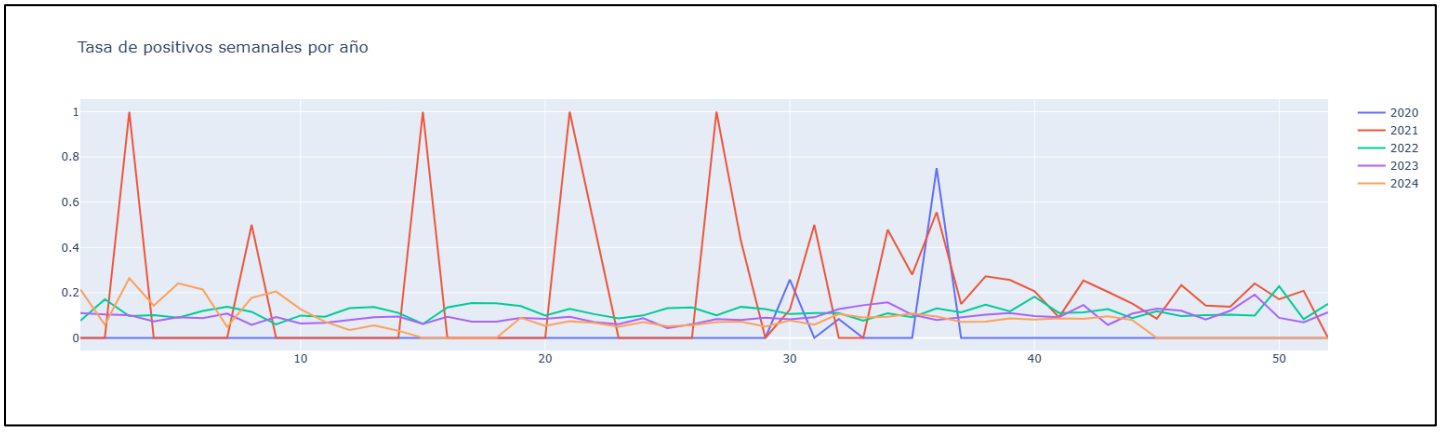

Figura 3-8: Gráfico representativo de la tasa de positividad por semanas de forma anual

Fuente: Elaboración propia

Estudio de estacionalidad y autocorrelación

Para evaluar la estacionariedad de la serie, primero se aplicó el test de Dickey-Fuller Aumentado (ADF). El resultado, ADF = -2.3198 y p-value = 0.1657, indica que la serie no es estacionaria, pero sugiere la presencia de tendencia y/o estacionalidad. Esto implica que es necesario transformar la serie usando la diferenciación para hacerla estacionaria.

A continuación, se pasó al análisis de las funciones de autocorrelación (ACF) y autocorrelación parcial (PACF). Estas reflejaron un comportamiento similar al de un "random walk", con correlaciones débiles (0.37 aproximadamente) pero significativas en ciertos rezagos (lag 6 y 12 en la ACF, y lag 6 en la PACF). Esto confirma la no estacionariedad inicial de la serie, ya que no se observa un patrón claro de estacionalidad o tendencia en los gráficos, como muestra la Figura 3-9.

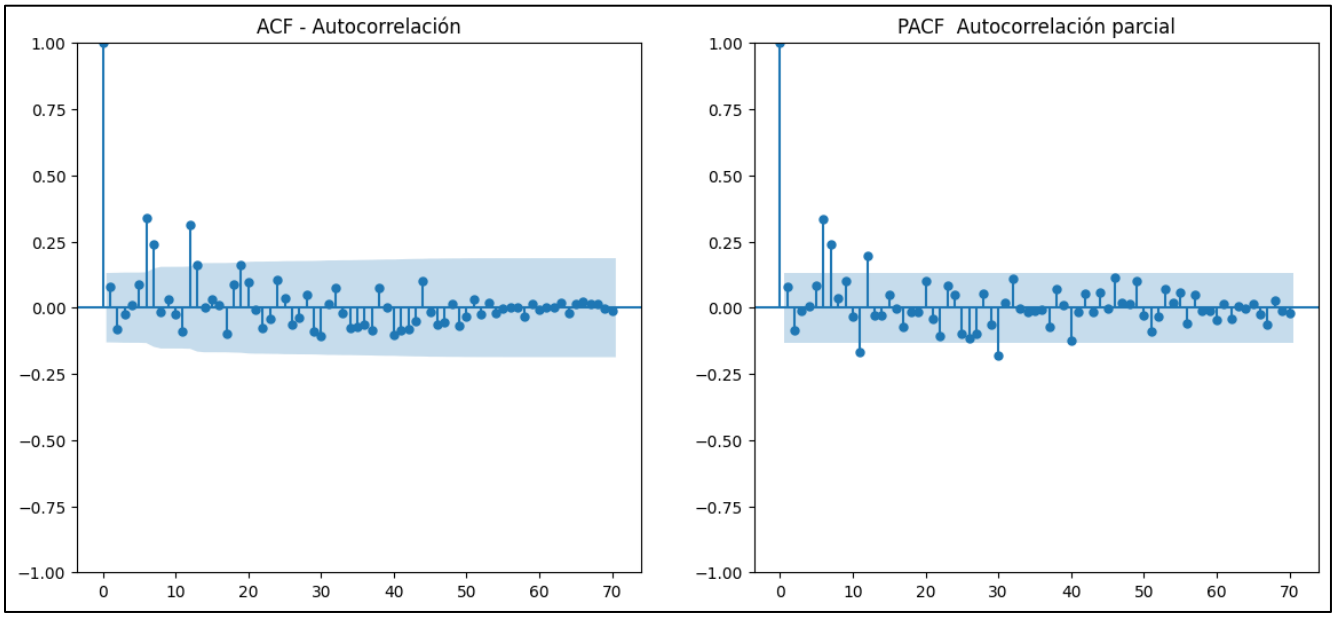

Figura 3-9: Gráfico de la autocorrelación de la serie origina de la tasa de positivos

Fuente: Elaboración propia

Dado que la serie no era estacionaria, se aplicó una diferenciación de primer orden. El nuevo test ADF (ADF = -7.0526, p-value = 0.0000) confirmó que la serie diferenciada es estacionaria. Además, como muestran los gráficos ACF y PACF tras la diferenciación (Figura 3-10), las autocorrelaciones se redujeron dentro de los intervalos de confianza e indica que la diferenciación eliminó las dependencias previas y estabilizó la serie.

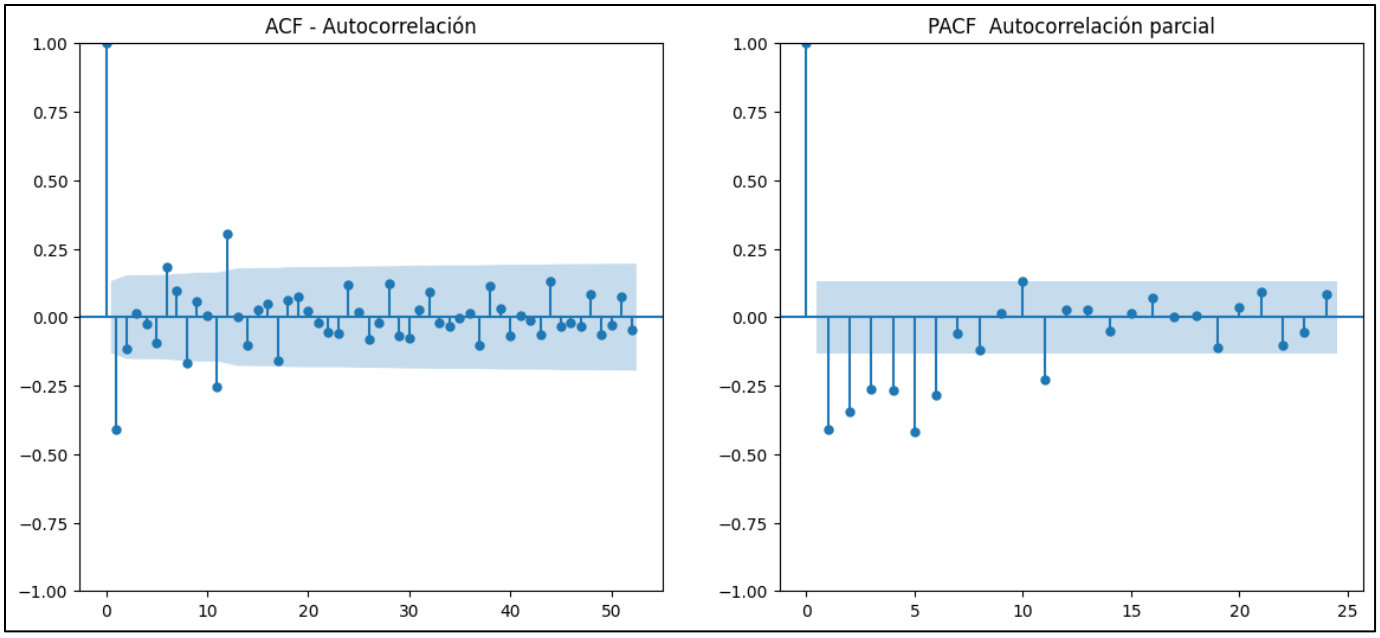

Figura 3-10: Gráfico de la autocorrelación de la serie diferenciada de la tasa de positivos

Fuente: Elaboración propia

Modelado con ARIMA

El siguiente paso fue seleccionar el modelo ARIMA, para lo cual se utilizó el Criterio de Información de Akaike (AIC). El modelo identificado como óptimo fue el ARIMA(2, 0, 3), con un valor de AIC de -205,73. En cuanto a sus parámetros, el modelo se caracteriza por:

- p = 2: dos términos autorregresivos. El modelo establece que el valor actual se explica en función de las dos observaciones anteriores.
- d = 0: número de diferenciaciones necesarias para que la serie sea estacionaria. Esto es raro. Sale 0, como si la serie inicial fuera estacionaria, pero no lo es. El caso es que, al devolver 0, es como si no hiciéramos diferenciación.
- q = 3: tres términos de media móvil, es decir, incorpora los tres últimos errores del modelo para mejorar la predicción.

Al igual que con el estudio sobre el número de positivos semanal, se procedió a realizar el estudio de los residuos para comprobar su validez.

El modelo identificado como óptimo fue el ARIMA(2, 0, 3). Este modelo, además, presentó un "Log-Likelihood" negativo y un valor estimado de sigma² bajo, lo que indicaría una varianza reducida de los errores. Sin embargo, el análisis detallado de los residuos reveló deficiencias importantes.

En primer lugar, se observó una inconsistencia respecto a la estacionariedad: el modelo seleccionó un valor de d=0, lo que supone que la serie es estacionaria, contradiciendo los resultados previos del test ADF.

Posteriormente, se aplicaron pruebas estadísticas a los residuos:

- Autocorrelación en los residuos: El test de Ljung-Box con 52 rezagos arrojó un estadístico de x^2=78.25 y un p-value de 0.0107 (siendo mucho menor a 0.05), lo que confirma la existencia de autocorrelación. Este resultado también se observa en el ACF de los residuos (Figura 3-11).

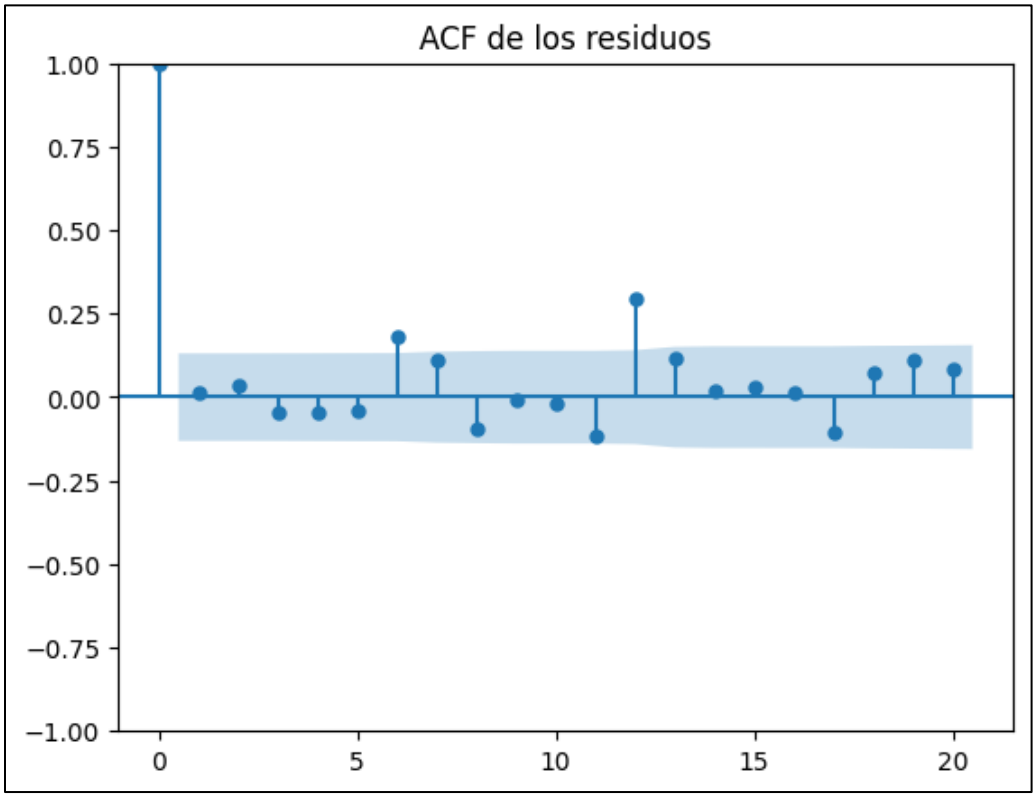

Figura 3-11: Función de autocorrelación de los residuos del modelo ARIMA(2,0,3) aplicado sobre la serie tasa de positividad semanal

Fuente: Elaboración propia

- Normalidad de los residuos: El test de Shapiro-Wilk obtuvo un p-value de 0.0000 y el test de Jarque-Bera también rechazó la hipótesis de normalidad. Esto indica que los residuos no siguen una distribución normal, como se aprecia en su distribución (Figura 3-12), en la que se observan asimetrías y picos.

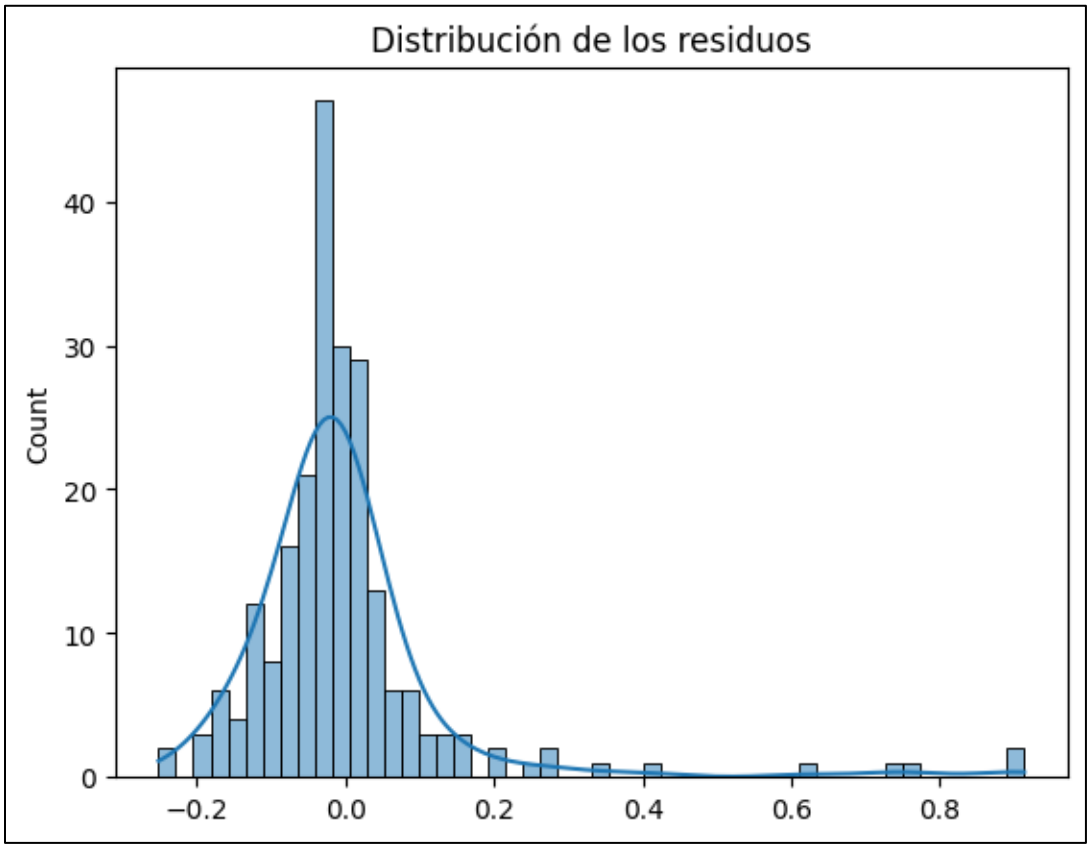

Figura 3-12: Gráfico de la distribución de los residuos del modelo ARIMA(2,0,3) aplicado a la tasa de positivos semanal

Fuente: Elaboración propia

- Heterocedasticidad: El test ARCH reportó un p-valor de 0.0000, lo que demuestra que la varianza de los residuos no es constante, es decir, que existe heterocedasticidad.

De esta forma y junto con la observación visual de residuos con picos de varianza en determinados periodos, muestran que el ARIMA no es capaz de capturar de forma adecuada la estructura de la serie temporal.

A pesar de estas limitaciones, se procedió a realizar predicciones con el modelo ARIMA(2, 0, 3) y se generaron proyecciones para las 10 semanas posteriores.

Las predicciones obtenidas mostraron una tasa de positivos con fluctuaciones irregulares, acompañada de intervalos de confianza amplios (Figura 3-13) reforzando la conclusión de que el modelo no es el más adecuado para esta serie temporal.

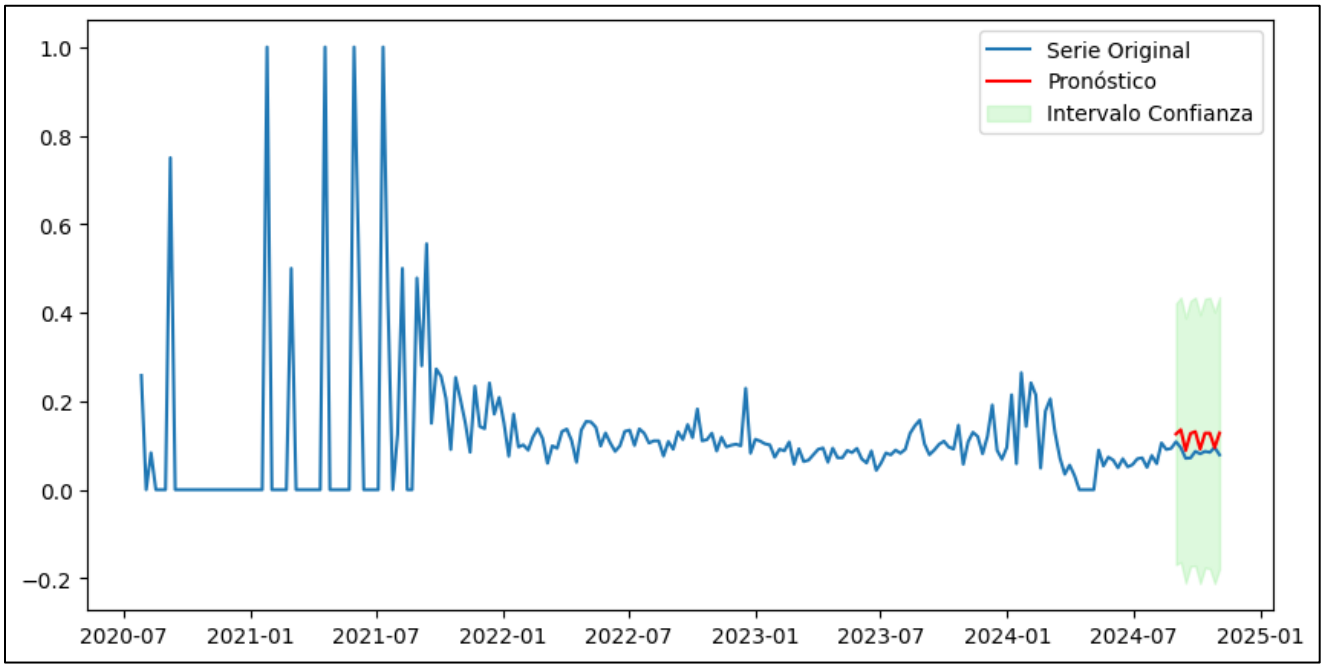

Figura 3-13: Predicción de la tasa de positividad para las siguientes 10 semanas con sus respectivos intervalos de confianza.

Fuente: Elaboración propia

- **Conclusión del estudio de estacionalidad**

Tanto en el análisis de la tasa de positividad semanal como en el del número absoluto de positivos, no se identificó una estacionalidad clara en los casos de Leishmaniasis. Si bien se aprecian ciertas tendencias, la presencia de anomalías en los datos limita la solidez de las conclusiones. En este sentido, sería necesario estudiar con mayor detalle la evolución semanal de los positivos para confirmar si dichas tendencias responden a un patrón real o a irregularidades en la recogida de datos. En general, las fluctuaciones parecen obedecer más a procesos aleatorios con dependencia de corto

plazo que a ciclos estacionales estables, por lo que este estudio debe entenderse como una base preliminar para futuras investigaciones con modelos más robustos y datos depurados.

3.3 Extensión del estudio en humanos

Otro aspecto del estudio ha sido el análisis de enfermedades que pueden afectar directamente a los seres humanos inmunodeprimidos, siendo la leishmaniasis el caso en el que se ha centrado. La elección de esta enfermedad se justifica por dos motivos. En primer lugar, su relevancia clínica en pacientes inmunocomprometidos y, en segundo lugar, la disponibilidad de datos que permiten su análisis espacio-temporal. El propósito de este análisis fue identificar patrones espaciales que ayudaran a encontrar áreas con una cantidad considerable de casos. Para ello, se aplicó la estadística Getis-Ord Gi* siguiendo la misma metodología empleada previamente en el análisis de enfermedades de mascotas, con el fin de garantizar la coherencia en la comparación de resultados.

3.3.1 Particularidades de los datos de Leishmaniasis en humanos

Los datos utilizados se cargaron desde un archivo CSV que contenía información relativa a los casos, siendo los mismos que en los representados por la Tabla 2.4. Tras el proceso de depuración de los datos se obtuvieron un total de 277 registros válidos, distribuidos en 163 códigos postales distintos y abarcando un amplio espectro temporal que comprende desde el año 2000 hasta el año 2025. Esta distribución temporal, en conjunto con la cantidad relativamente limitada de casos registrados, dificulta la identificación clara de patrones epidemiológicos consistentes.

Como resultado, a pesar de la aplicación del análisis de Getis-Ord, la calidad y cantidad de los datos disponibles no permiten extraer conclusiones definitivas sobre la distribución de la Leishmaniasis en España. En este sentido, los resultados deben ser interpretados con cautela, y más que ofrecer respuestas definitivas, estos hallazgos sirven como una primera aproximación que subraya la necesidad de contar con bases de datos más completas y homogéneas en futuros estudios.

De la misma forma que ocurrió con el análisis de estacionalidad mediante ARIMA, el estudio puede entenderse como una extensión adicional que, si bien no constituye el núcleo principal de la herramienta, ofrece información adicional de gran valor para el control y la prevención.

3.3.2 Resultados del análisis espacial en humanos

Por tanto, el análisis se organizó en dos niveles: provincial y por códigos postales dentro de cada provincia, de manera que se pudieran captar patrones tanto generales como locales.

- **Análisis por provincias**

En primer lugar, se agruparon los casos por provincia y se calculó el número total de diagnósticos en cada una de ellas. A continuación, los resultados se representaron en un mapa con datos geoespaciales de las provincias en formato GeoJSON. A partir de estos datos, se construyó una matriz de pesos espaciales basada en la contigüidad de tipo Queen (de igual forma que con los casos de mascotas). Con esta matriz, se aplicó el cálculo de la estadística Getis-Ord Gi*, lo que permitió obtener valores Z e identificar las provincias con mayor o menor concentración de casos. A continuación, se muestra un ejemplo del resultado obtenido tras aplicar el análisis:

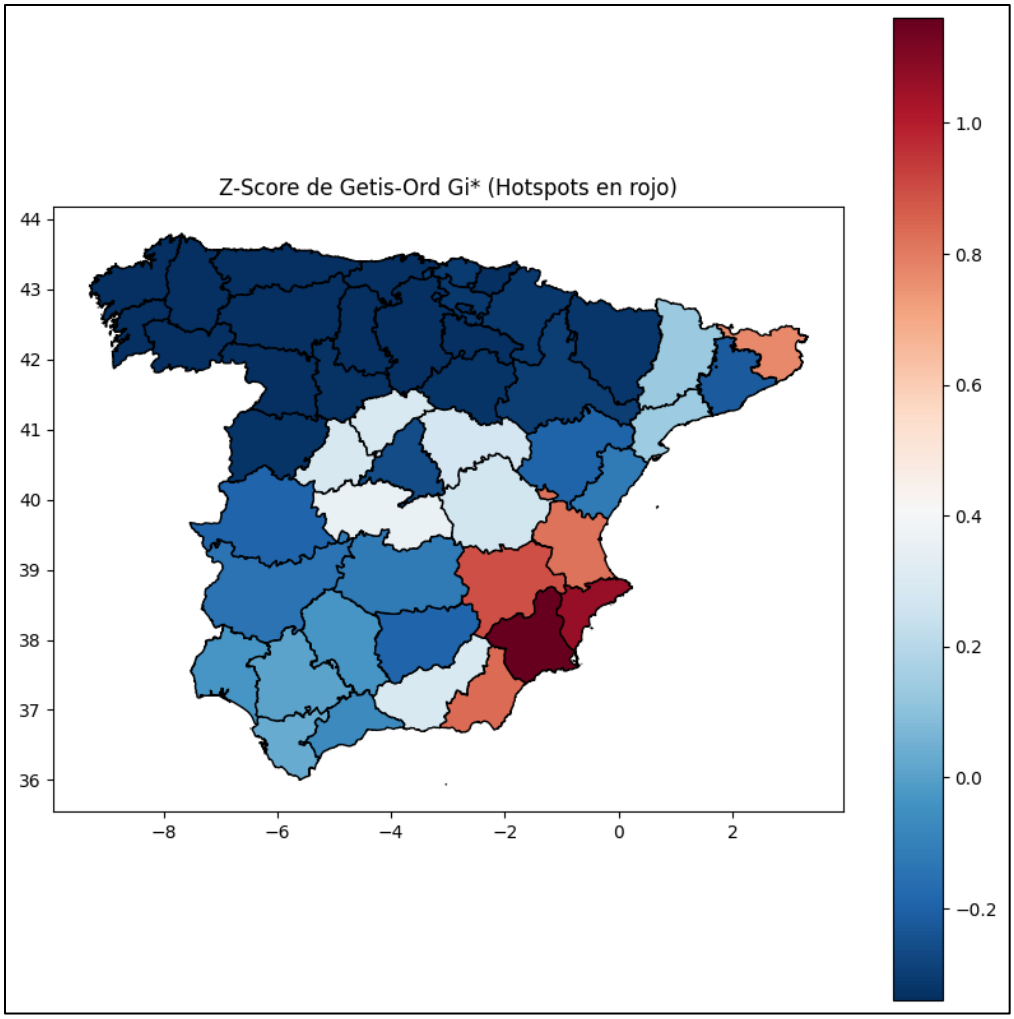

Figura 3-14: Resultados de Getis-Ord Gi por provincias en casos humanos*

Fuente: Elaboración propia

En conclusión, La Figura 3-14 muestra los resultados del análisis estadístico Getis-Ord Gi* aplicado a los casos humanos por provincias en España. El mapa muestra con claridad un clúster caliente en el sureste de la península, con la provincia de Murcia destacando como el área de mayor intensidad, acompañada por valores positivos intermedios en Alicante y Albacete. En contraste, la mayor parte del norte y el oeste del país presentan tonos azules oscuros, lo que sugiere la presencia de zonas frías o con baja incidencia de casos. Un aspecto llamativo se encuentra en las provincias circundantes a Madrid, donde los valores aparecen cercanos a cero. Este resultado sugiere la presencia de una situación ambigua o inestable en la distribución de casos en la zona

central, que podría estar influenciada por limitaciones en los datos disponibles o por la falta de una tendencia espacial clara.

- **Análisis por códigos postales**

Para refinar el análisis, se estudió el patrón espacial a nivel de código postal. En este caso, se utilizaron datos geoespaciales en formato SHP, lo que permitió representar cada código postal dentro de su provincia. Los casos se agruparon por código postal y se seleccionaron aquellos que contaran con al menos un diagnóstico registrado. La matriz de pesos espaciales se construyó mediante el método de los K vecinos más cercanos (KNN), más adecuado para este nivel de detalle. Los resultados obtenidos se guardan en una carpeta donde están todas las imágenes de todas las provincias analizadas.

En las siguientes figuras se presentan los resultados del análisis para las provincias de Barcelona y Madrid, que destacan por la mayor disponibilidad de datos y donde la aplicación de Getis-Ord permite observar con más claridad la concentración espacial de los casos:

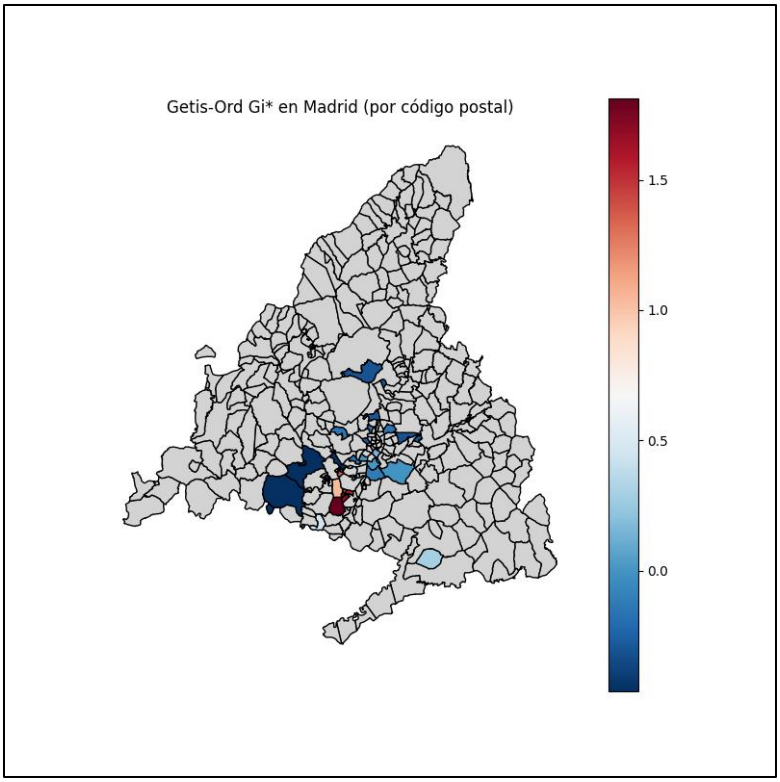

Figura 3-15: Resultados de Getis-Ord Gi a nivel de código postal en la provincia de Madrid*

Fuente: Elaboración propia

La Figura 3-15 muestra los resultados del estadístico Getis-Ord Gi* aplicado a los casos de Leishmaniasis por código postal en la Comunidad de Madrid. Se identifican clústeres calientes en el suroeste de la región, indicando zonas con mayor densidad de casos, y clústeres fríos en azul oscuro, correspondientes a áreas de baja incidencia. Estos patrones evidencian que la enfermedad no se distribuye de manera homogénea, sino que se concentra en determinados municipios.

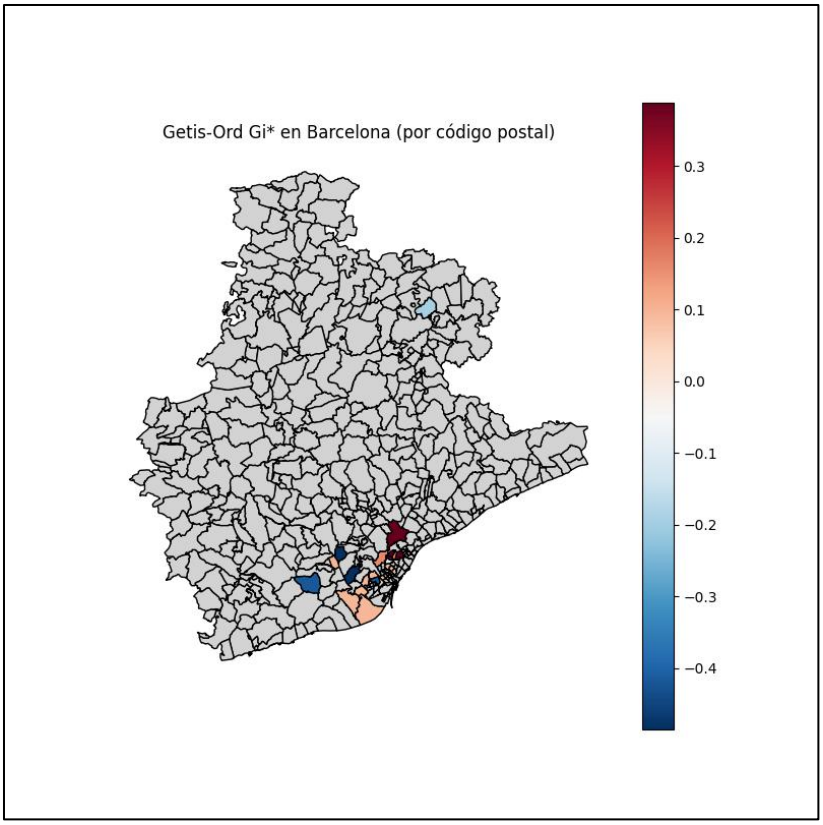

Figura 3-16: Resultados de Getis-Ord Gi a nivel de código postal en la provincia de Barcelona*

Fuente: Elaboración propia

La Figura 3-16 muestra los resultados del análisis de casos en la provincia de Barcelona por código postal. Se observan clústeres calientes en la zona norte, indicando mayor concentración de casos, y clústeres fríos en el centro y noreste, reflejando menor incidencia. Estos patrones muestran que la distribución de la enfermedad no es uniforme y pueden orientar la vigilancia en áreas de mayor riesgo.

3.3.3 Conclusión y comparativa con el caso de mascotas

El análisis espacial realizado tiene limitaciones significativas derivadas de la escasez de datos y de la dispersión temporal de los casos. Estas restricciones afectan a la solidez de los resultados y dificultan la identificación de patrones espaciales consistentes. Además, se han detectado comportamientos anómalos, como la

reducción inesperada del valor Z en provincias adyacentes a Madrid, lo que pone en evidencia posibles distorsiones relacionadas con la baja densidad de observaciones. No obstante, los resultados pueden considerarse una primera aproximación que ofrece una base para que futuras investigaciones, apoyadas en un mayor volumen de datos y series temporales más completas, desarrollen análisis más precisos y concluyentes.

Capítulo 4 - Introducción a la aplicación

La herramienta web desarrollada se llama **Desease Pet Scan** (DPS). Esta aplicación tendrá como principal función ofrecer de forma visual, mediante un mapa interactivo del territorio nacional, brotes inusualmente altos de las enfermedades Leishmaniasis y Giardia.

Además, se incorporará una visualización en el mapa que muestre la evolución histórica de la temperatura en distintos puntos distribuidos por España, utilizando datos proporcionados por la Agencia Estatal de Meteorología (AEMET).

Para acceder a la aplicación se requiere estar conectado a la VPN de la Universidad Complutense usando el siguiente enlace: **galeria.ucm.es**

4.1 Introducción al funcionamiento de la aplicación

La aplicación está pensada para el análisis geoespacial de resultados de test realizados en mascotas de forma que permite a los usuarios realizar las siguientes funciones:

- Registrar nuevos datos en el sistema a través de los endpoints que me suestran en la Figura 4-1, en concreto "/api/test/upload_csv" para añadir nuevos test realizados en mascotas, "/api/aemet/fill_db" para rellenar la tabla de la AEMET con datos climáticos y "/api/human/add_human_data" para añadir incidencias en humanos. Podemos acceder a dicha documentación a través de este enlace https://147.96.80.52/MapPage/api/docs.

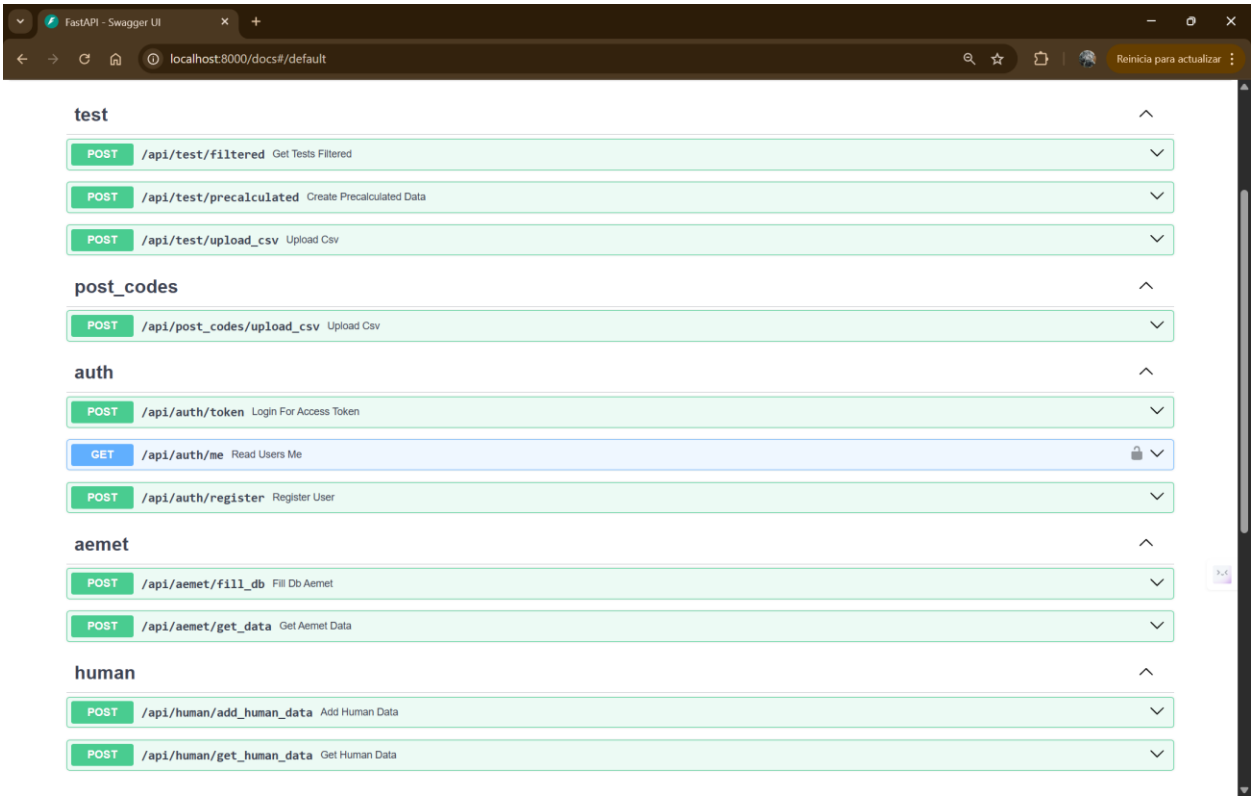

Figura 4-1: Imagen de la documentación de nuestra API donde se observan los endpoints desarrollados. Fuente: Captura de pantalla de nuestro backend.

- Visualizar sobre un mapa interactivo los resultados obtenidos al aplicar Getis Ord sobre nuestros datos de test realizados en mascotas, también se podrá ver la información climática y los datos en crudo sobre los test realizados en humanos. En la Figura 4-2 podemos observar cómo se vería representada la información sobre el mapa, en este caso se ha ajustado con un intervalo de tiempo de un año por tanto lo que estamos observando es el resultado de aplicar el algoritmo Getis Ord sobre todos los datos comprendidos entre el 01-01-2022 y 01-01-2023. Las superficies de color rojo y azul representan los puntos calientes y fríos que detecta el algoritmo sobre los datos de los test en mascotas y la capa que cubre el mapa de España representa los datos climáticos.

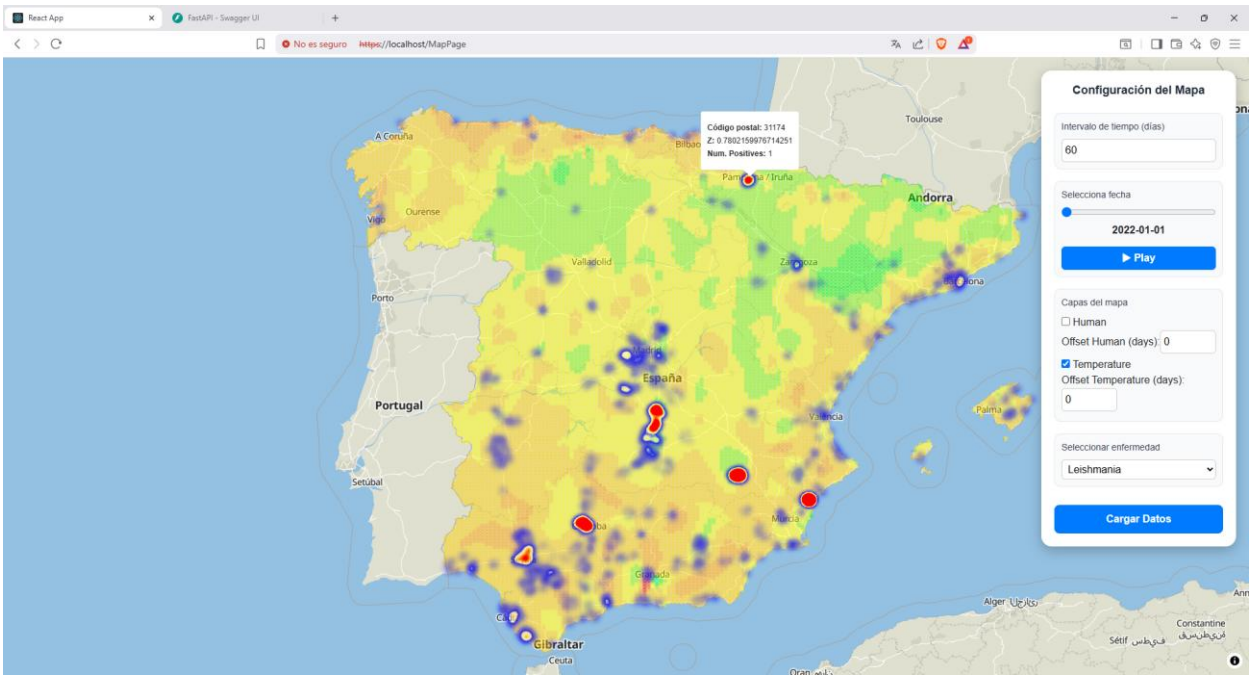

Figura 4-2: Captura de la representación visual de los puntos calientes y la temperatura sobre el mapa nacional

Fuente: Captura de pantalla de la interfaz

4.2 Estructura general

A continuación, se va a describir la estructura sobre la que está montada la herramienta, en particular la Figura 4-3 representa el mapa de flujo que sigue la aplicación de forma interna.

La arquitectura está basada en contenedores Docker, donde cada servicio se ejecuta de forma independiente y se comunica con el resto a través de redes internas. Esto permite aislar los componentes, mejorar la seguridad y facilitar la escalabilidad de la aplicación.

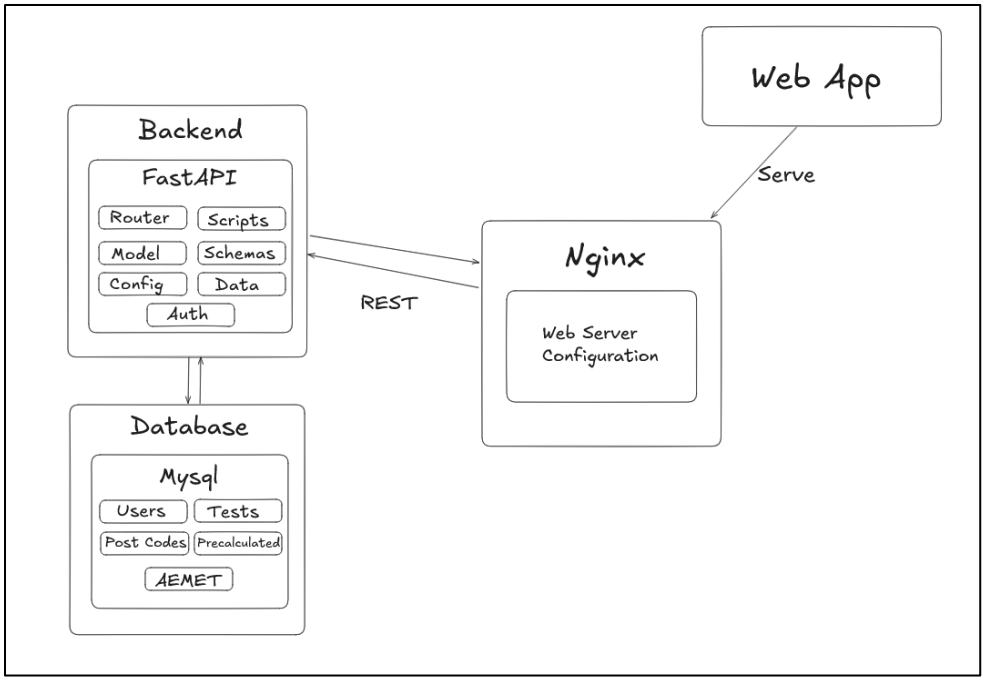

Figura 4-3: Representación de la estructura de la aplicación

Fuente: Elaboración propia

- "Backend": centro lógico de **DPS** que se encarga de manejar, transformar e interpretar todos los datos de entrada y enviárselos al "frontend". Además de responder ante solicitudes hechas en el mismo.
- "Nginx": Servidor web que actúa alojando la página web de forma estática y redirigiendo las peticiones HTTP de los clientes al backend. Además, permite la gestión de certificados SSL haciendo más segura las conexiones y ampliar la escalabilidad en caso de ser necesario.
- "Frontend": Es la parte de la aplicación encargada de mostrar los resultados. Está formado únicamente por una aplicación web que hace las peticiones al "backend" usando los endpoints predefinidos para obtener los resultados y otros datos de interés, mostrándolos en un mapa interactivo a continuación.
- "Database": Sistema de gestión de bases de datos encargado de almacenar y proveer los datos utilizados por el backend.

4.3 Implementación backend

El backend del sistema ha sido desarrollado en Python utilizando el framework FastAPI [12] que proporciona una API rápida, segura y escalable, permitiendo gestionar la lógica del método estadístico y el acceso a los datos. Este componente actúa como intermediario entre el frontend y la base de datos, donde se asegura que la información fluya de manera consistente y protegida. Además, su diseño modular y el uso de estándares abiertos permiten una fácil mantenibilidad, así como la incorporación de nuevas funcionalidades en el futuro.

4.3.1 Tecnologías usadas en la implementación

El backend de la aplicación se ha desarrollado completamente en Python, utilizando un conjunto de librerías que se especializan y permiten cubrir los requerimientos del sistema, tanto la exposición de servicios web como el análisis espacial avanzado con Getis-Ord. A continuación, se describen las librerías usadas:

Librerías del Framework principal

- **FastAPI**: Framework ligero y de alto rendimiento para la creación de APIs REST, con validación de datos y documentación automática (Swagger).
- **Uvicorn**: Servidor ASGI para ejecutar las aplicaciones web desarrolladas en Python.
- **Starlette**: Requisito base de FastAPI, responsable de la gestión de peticiones, middleware y respuestas.

Librerías para validación de datos y configuración de la aplicación

- **Pydantic** [13]: Útil para la validación de datos y definición modelos para garantizar integridad de la información.
- **Pydantic-settings**: Librería basada en Pydantic que gestiona de forma centralizada la configuración mediante variables de entorno.
- **Python-dotenv**: Librería que carga las variables de entorno desde los archivos *.env*.

- **Typing-expressions**: Ofrece soporte para tipos avanzados en Python.

Librerías para las bases de datos y seguridad

- **SQLAlchemy**: Usa el Object-Relational Mapping (ORM) que se utiliza para definir modelos, crear tablas y ejecutar consultas.
- **PyMySQL**: Conector del backend en Python con bases de datos de MySQL.
- **Greenlet**: Requisito interno de SQLAlchemy para la ejecución eficiente de operaciones concurrentes.
- **passlib[bcrypt], bcrypt**: Realiza el hashing seguro de contraseñas.
- **Cryptography**: Soporte criptográfico necesario para la seguridad de JWT y hashing.
- **PyJWT, python-jose[cryptography]:** Gestión de JWT (JSON Web Tokens) para autenticación y protección de rutas y operaciones criptográficas.

Librerías para el análisis espacial y estadístico

- **GeoPandas**: Extensión de pandas que soporta datos espaciales (geoJSON, shapefiles).
- **Shapely**: Especializada en la creación, análisis y manipulación de geometrías (puntos, polígonos, intersecciones).
- **PyProj**: Enfocada en la transformación de coordenadas entre distintos sistemas de referencia.
- **PyOgrio**: Herramienta para la lectura y escritura de datos geoespaciales.
- **libpysal**: Utilidades de econometría espacial. Incluye funciones como Queen usada para la preparación de datos del Getis-Ord.
- **esda**: Librería Exploratory Spatial Data Analysis(ESDA), donde se incluye la implementación de Getis-Ord G* para identificar focos de enfermedades en mascotas.

Librerías para la gestión y manipulación de datos

- **Pandas**: Permite la manipulación de datos en formato tabla, útil para poder gestionarlos.

- **Python-dateutil, pytz, tzdata**: Librería útil para el manejo de fechas, intervalos temporales y zonas horarias
- **NumPy**: Fundamental para el cálculo numérico. Permite trabajar con arrays multidimensionales y operaciones con vectores, acelerando los cálculos necesarios para el procesamiento de los datos.
- **SciPy**: Conjunto de herramientas científicas avanzadas que se emplea para realizar interpolaciones, optimización, análisis estadístico y complementan las funcionalidades de NumPy.
- **Scikit-learn**: Biblioteca de machine learning enfocada en análisis y modelado de datos. Se utiliza principalmente en el preprocesamiento, escalado y validación de datos.
- **Joblib**: Herramienta para la serialización y ejecución paralela de tareas. Facilita el manejo eficiente de cálculos costosos y distribuidos.
- **Threadpoolctl**: Controla el uso de hilos en operaciones de bajo nivel (BLAS, OpenMP) usadas por NumPy, SciPy y Scikit-learn. Su función es optimizar el rendimiento evitando sobrecarga en los cálculos paralelos.

Librerías extras

- **APScheduler**: Permite la ejecución de tareas programadas dentro de la aplicación.
- **Request**: Útil para la realización de solicitudes HTTP a servicios externos, en especial se usa para conseguir los datos de la API de la AEMET.
- **python-multipart**: Procesa formularios con archivos subidos (multipart/form-data), necesarios para el frontend.

4.3.2 Arquitectura modular del ¨backend¨

Una vez expuestos las librerías externas que se han usado para la implementación, se va a explicar la estructura que sigue el backend. El backend fue diseñado siguiendo una arquitectura modular y desacoplada, es decir, se hizo foco en que cada componente del sistema estuviera organizado en módulos independientes donde cada uno tuviera una responsabilidad definida.

En concreto, esta organización facilita que las funcionalidades como la gestión de usuarios, la autenticación, la conexión con la base de datos o el análisis de los datos estén claramente separadas en módulos.

Gracias a este enfoque, el código se puede mantener y escalar con facilidad ya que los cambios a un módulo no afectan al resto. Además, este enfoque ofrece la posibilidad de integrar nuevas funcionalidades como, por ejemplo, nuevas enfermedades de mascotas, sin modificar en exceso el código ya existente.

A continuación, se va a detallar los módulos que componen la arquitectura del backend, junto con una descripción de su función:

- "Router": Modulo encargado de generar los endpoints a los que accede el servidor web aplicando diferentes funciones entre las que se pueden encontrar. Para ello se usa la propia librería de FastAPI, que permite definir rutas.
- "Scripts": Son los conjuntos de implementaciones de las transformaciones explicadas en el punto 2.2 y devuelven un resultado con una serie de restricciones. Se pueden encontrar por ejemplo la aplicación del algoritmo de GetisOrd o la transformación de los datos meteorológicos para su posterior uso.
- "Model": Módulo donde están almacenados los modelos que se usarán para definir las tablas de la base de datos, utilizando la librería SQLAlchemy.
- "Schemas": Módulo encargado de definir los modelos de datos que se usaran en la API, permitiendo controlar que campos son necesarios al crear o actualizar. Además, permite definir modelos para devolver al cliente sencillamente. Este módulo usa la librería de Pydantic.
- "Config": Esta parte se encarga principalmente de definir las variables de configuración donde trabajará el backend definidas en el archivo ***.env***. Se puede acceder al valor de estas variables a través de una clase Settings de la librería de Pydantic. También se encarga de establecer la conexión con el contenedor de la base de datos.

- "Data": Repositorio donde se guardan los shapefiles de los códigos postales donde está su geometría, ya que guardar la información en la base de datos es inviable.
- "Auth": Módulo encargado de gestionar la autenticación de los usuarios de la aplicación y de verificar de inicio de sesión usando JWT.

Gestión de las variables de entorno

Un aspecto clave en la arquitectura del backend es la gestión de las variables de entorno y permiten separar la configuración sensible del código.

Toda la información crítica, como las credenciales de acceso a la base de datos o las claves utilizadas para el hasheo de contraseñas y la generación de JWT, se encuentra centralizada en un archivo .env. Este archivo es cargado automáticamente por el sistema al inicio de ejecución gracias al uso de *Pydantic* y *python-dotenv*, garantizando que la aplicación pueda adaptarse a distintos entornos sin necesidad de modificar el código.

Además, se ha incluido un archivo .env.example que actúa como plantilla de referencia, facilitando la creación de su propio archivo .env con las configuraciones adecuadas para ejecutar la aplicación correctamente.

4.3.3 Acceso a la base de datos

El acceso a la base de datos en el backend se gestiona a través de la librería SQLAlchemy, que actúa como *ORM* (Object Relational Mapper) y permite manejar tablas de manera declarativa y estructurada, evitando la necesidad de escribir consultas SQL puras. Esta implementación se encuentra organizada en dos partes principales: la configuración de la conexión (en la carpeta "config") y la definición de modelos (en la carpeta "model").

Toda esta implementación se ve reflejada en un archivo de conexión, desde donde se configura. En este archivo se establecen los parámetros necesarios (dialecto de la conexión, usuario que va a conectarse, contraseña del usuario, nombre del host

y nombre de la base de datos) definidas en las variables de entorno. Esto garantiza mucha seguridad y flexibilidad ya que no expone las credenciales en el código directamente.

Una vez configurado el acceso, cuando la aplicación necesita interactuar con la base de datos se abre una conexión temporal que se cierra automáticamente al finalizar la operación, véase por ejemplo una inserción en una tabla. Este enfoque permite un control más seguro de los recursos, evita fugas de conexiones y garantiza un manejo eficiente de la base de datos.

Por otro lado, los modelos se definen dentro del módulo model y describen la estructura de las tablas que compondrán la base de datos. A continuación, se exponen las tablas que existen en la base de datos:

- **Tests:** tabla que representa los registros de pruebas realizadas a cada mascota.

Columna	**Tipo de dato**	**Descripción**
id_test	Integer	Clave primaria de la tabla. Identificador único para cada prueba.
post_code	String(5)	Código postal asociado a la prueba. Campo no nulo.
date_done	Date	Fecha en la que se realizó la prueba. Campo no nulo.
desease	String(50)	Nombre de la enfermedad analizada en la prueba. Campo no nulo.
result	Integer	Resultado numérico de la prueba donde 1 es positivo y 0 negativo. Campo no nulo.
city	String(50)	Ciudad en la que se realizó la prueba. Campo opcional.

age	Integer	Edad de la mascota analizada en la prueba. Campo opcional.
sex	String(50)	Sexo de la mascota analizada en la prueba. Campo opcional.

Tabla 4.1: Tabla descriptiva de las columnas de los datos de pruebas a mascotas

- **Post Code:** Esta tabla guarda el censo de mascotas por código postal.

Columna	**Tipo de dato**	**Descripción**
post_code	String(5)	Clave primaria de la tabla. Representa el código postal en el código de 5 dígitos.
census	Float	Cálculo aproximado de la cantidad de mascotas registradas en el área postal. Campo no nulo.

Tabla 4.2: Tabla descriptiva de las columnas de los datos censales de mascotas

- **AEMET:** Tabla que almacena la información meteorológica obtenida de la Agencia Estatal de Meteorología (AEMET). Su finalidad es proporcionar información climática relativa a la temperatura, con su referencia geográfica.

Columna	**Tipo de dato**	**Descripción**
id	String(10)	Clave primaria de la tabla. Identificador único de cada registro meteorológico.
lon	Float	Longitud geográfica medida en grados de la estación de medición. Campo no nulo.

lat	Float	Latitud geográfica medida en grados de la estación de medición. Campo no nulo.
date	String(10)	Fecha del registro meteorológico en formato de cadena (ejemplo: "YYYY-MM-DD").
temp	Float	Temperatura medida en grados centígrados registrada en el momento de la medición.
location	String(50)	Nombre de la provincia donde se encuentra la estación.

Tabla 4.3: Tabla descriptiva de las columnas de los datos meteorológicos

- **Precalculated:** Tabla que almacena los resultados significativos previamente procesados y calculados a partir de los datos de los tests realizados a las mascotas en ventanas de tiempo que suelen ser comunes con el fin de ahorrar ejecuciones y mejorar el rendimiento de la aplicación.

Columna	**Tipo de dato**	**Descripción**
id	String(10)	Clave primaria de la tabla. Identificador único de cada resultado precalculado.
desease	String(15)	Nombre de la enfermedad (Leishmaniosis o Giardia)a la que corresponde el resultado.
days_interval	String	Intervalo de días considerado para el cálculo (ejemplo: 15D o 30D). Campo no nulo.
end_date	DateTime	Fecha final en la que se realizó el cálculo o hasta la última fecha disponible en los datos.

result_data	MEDIUMTEXT	Campo de gran tamaño que almacena el resultado de la aplicación de GetisOrd y el polígono del código postal para poder representarlo. En formato JSON.

Tabla 4.4: Tabla descriptiva de las columnas de los datos precalculados

- **Users:** Esta tabla gestiona la información de los usuarios de la aplicación, donde almacena los datos necesarios para la autenticación y autorización.

Columna	**Tipo de dato**	**Descripción**
username	String(50)	Clave primaria de la tabla. Nombre único que identifica a cada usuario en el sistema.
email	String(100)	Dirección de correo electrónico. Debe ser única y no puede ser nula.
full_namel	Integer	Nombre completo del usuario (opcional).
hashed_password	String(255)	Contraseña almacenada de forma segura mediante un algoritmo de hash. Campo no nulo.
disabled	Boolean	Indica si el usuario está activo (False) o inhabilitado (True). Campo no nulo.

Tabla 4.5: Tabla descriptiva de las columnas de los usuarios

4.3.4 API: rutas y flujo de peticiones

El backend se organiza en distintos módulos de rutas, cada uno de ellos se encarga de gestionar un conjunto específico de recursos y operaciones. Estos archivos

agrupan los endpoints disponibles, así como las reglas de validación y procesamiento necesarias para garantizar la coherencia y seguridad en la manipulación de los datos. A continuación, se describen las principales rutas implementadas:

▪ Rutas de las pruebas

Este módulo define los endpoints relacionados con la gestión de pruebas (*tests*) y sus datos asociados.

Endpoints principales

- /filtered: recibe como parámetros la enfermedad y el intervalo de tiempo que se desea estudiar. Si dicho intervalo es uno de los ya previamente calculados, obtendrá la información de la tabla "precalculated" y la devolverá en formato JSON. En caso contrario se aplicará el algoritmo Getis-Ord a los datos de la tabla "test" y los devuelve en formato JSON también.
- /precalculated: genera datos precalculados aplicando el algoritmo de Getis-Ord que se encuentra definido.
- /upload_csv: permite subir el archivo de datos de test tipo CSV para su almacenamiento en la base de datos.

Validación de datos

La validación se realiza manualmente, comprobando que el cuerpo de la solicitud sea un JSON válido y que contenga los parámetros obligatorios como start_date, interval y disease.

Manejo de errores

- 400 Bad Request: errores de validación.
- 500 Internal Server Error: fallos durante el procesamiento.

▪ Rutas de autenticación de usuarios

Este módulo gestiona la autenticación y el registro de usuarios.

Endpoints principales

- /token: autentica a los usuarios y genera un token de acceso JWT.
- /me: devuelve información del usuario autenticado.
- /register: registra nuevos usuarios en la base de datos.

Validación de datos

Se utiliza Pydantic para validar entradas como credenciales y datos de registro, y para definir los modelos de salida (Token, UserOut).

Manejo de errores

- 401 Unauthorized: credenciales inválidas.
- 400 Bad Request: usuario o correo duplicado.
- 500 Internal Server Error: errores inesperados.

▪ Rutas de la AEMET

Este módulo gestiona la integración con la API de la Agencia Estatal de Meteorología (AEMET).

Endpoints principales

- /fill_db: consulta la API y almacena datos meteorológicos en la base de datos.
- /get_data: recupera y procesa datos meteorológicos que después se envían al frontend.

Validación de datos

Se realiza manualmente, verificando que el JSON recibido contenga los parámetros requeridos.

Manejo de errores

- 400 Bad Request: cuerpo de solicitud inválido.
- 500 Internal Server Error: fallos de la API o del procesamiento interno.

- Rutas de casos de humanos

Este módulo gestiona los datos relacionados con casos humanos.

Endpoints principales

- /add_human_data: permite subir el archivo CSV con datos de casos humanos.
- /get_human_data: recupera datos filtrados y genera GeoJSONs para visualización espacial.

Validación de datos

Se realiza de manera manual, asegurando que los datos enviados tengan el formato y parámetros correctos.

Manejo de errores

- 400 Bad Request: solicitud inválida.
- 500 Internal Server Error: fallos internos durante el procesamiento.

- Rutas de censo de mascotas por código postal

Este módulo gestiona los datos relacionados con el censo de mascotas por cada código postal.

Endpoint principal

- /upload_csv: recibe el archivo CSV que contiene la información relativa al censo de mascotas por código postal y los almacena en la base de datos.

Validación de datos

Se verifica manualmente que los archivos no estén vacíos y tengan el formato correcto.

Manejo de errores

- 400 Bad Request: archivo vacío o formato inválido.

- 500 Internal Server Error: errores durante el procesamiento.

4.3.5 Autenticación y autorización

El módulo de autenticación es una parte esencial del sistema, ya que gestiona el acceso de los usuarios a la aplicación. Su función principal es garantizar que solo las personas registradas y autorizadas puedan utilizar las herramientas disponibles.

Para ello, este módulo se organiza en diferentes archivos que trabajan de manera conjunta: auth_service.py, dependencies.py y security.py.

Registro e inicio de sesión

Este archivo auth_service.py gestiona las operaciones básicas de los usuarios: el registro y el inicio de sesión.

- En el registro, se comprueba que el nombre de usuario no exista previamente y, si es válido, se guarda junto con una contraseña encriptada.
- En el inicio de sesión, se comprueban los datos introducidos y, si son correctos, se genera un token de acceso JWT que permite al usuario interactuar con el sistema de manera segura durante un tiempo determinado.

De esta forma, se asegura que cada usuario disponga de una cuenta propia y que sus credenciales se manejen de manera protegida.

Protección de rutas

El archivo dependencies.py no gestiona directamente el registro o inicio de sesión, aunque sí actúa como un filtro de seguridad para las partes de la aplicación que requieren autenticación.

Su función es verificar que el token de acceso enviado por el usuario es válido y que la cuenta está activa. En caso contrario, el sistema bloquea la petición. Gracias a este control, se garantiza que solo los usuarios autorizados puedan acceder a las funcionalidades más sensibles.

Gestión de contraseñas

El archivo security.py se centra en la forma en que se almacenan y verifican las contraseñas. En lugar de guardarlas directamente, el sistema crea un hash que no puede descifrarse. De esta forma, aunque la base de datos fuera comprometida, las contraseñas reales no quedarían expuestas.

Cuando un usuario inicia sesión, la contraseña que introduce se compara con este código, lo que garantiza que solo el propietario de la cuenta puede acceder a ella.

4.4 Implementación frontend

El apartado del front-end se encarga de facilitar a los usuarios una interfaz visual e interactiva de los datos que previamente han sido procesados y almacenados en nuestra base de datos.

4.4.1 Tecnologías usadas

Este apartado del proyecto ha sido desarrollado en su totalidad con JavaScript, en concreto hemos utilizado una librería llamada React [13] que permite la creación de interfaces de usuarios de forma eficiente y escalable. Para el correcto funcionamiento de la aplicación se han implementado las siguientes herramientas adicionales:

- **React Router DOM:** Permite la navegación fluida sin la necesidad de recargas en aplicaciones de una sola página.
- **Axios:** Establece comunicaciones con el backend a través de solicitudes http.
- **Bootstrap:** Framework que permite la personalización de estilos para crear interfaces de usuario atractivas y responsivas.
- **MapLibre:** Contiene un amplio catálogo de librerías que permiten la creación de mapas interactivos en React.

4.4.2 Desarrollo del frontend con React

El frontend se ha estructurado en distintos componentes de forma que se facilite tanto el desarrollo como la organización del proyecto. Además, se ha implementado un mapa interactivo que muestra los datos procesados y proporcionados por el backend, con el que se establece una comunicación.

4.4.3 Estructura del "front"

El origen del frontend nace del comando que se muestra en la Tabla 4.6.

npx create-react-app "nombre-del-proyecto"

Tabla 4.6: Comando para crear proyecto en React con estructura básica

Dicho comando genera la estructura básica de archivos junto con las configuraciones básicas que permiten la inicialización de un proyecto en React.

En el directorio principal encontramos diversos ficheros proporcionados por la plantilla, pero donde se encuentra la mayor parte del desarrollo es en el directorio "src/".

Figura 4-4: Representación gráfica de la estructura del "front-end"

Fuente: Elaboración propia

Como podemos ver en la Figura 4-3 dentro del directorio principal encontramos diversos archivos generados por la plantilla y el componente principal de la aplicación llamado App.jsx. Dicho componente centraliza las tres tareas principales del frontend:

- Define las rutas principales como el login, register y el mapa.
- Gestiona la autenticación del usuario mediante un contexto global.

- Protege las rutas privadas, en este caso el mapa interactivo.

El directorio "components/" contiene todos aquellos componentes que son reutilizables. Estos componentes están diseñados para ser independientes y modulares de forma que el código sea más mantenible. En la Figura 4-7 se aprecia como se unen todos estos componentes para formar la interfaz de usuario.

Componente	**Descripción**
ExecutionButton.jsx	Botón de ejecución, cuya función es solicitar los datos al backend con los parámetros que haya seleccionados en ese momento.
LayerSelector.jsx	Permite activar o desactivar las capas del mapa interactivo que el usuario desee mediante casillas de verificación.
Map.jsx	En ella se encuentra todo el desarrollo del mapa interactivo. Cuenta con tres capas principales sobre las que se muestra toda la información visual, estas capas son para los test de mascotas, temperatura y test en humanos. También cuenta con otra serie de capas secundarias encargadas de mostrar información adicional a la hora de arrastrar el cursor sobre el mapa.

PrivateRoute.jsx	Componente encargada de comprobar la autenticación del usuario con el fin de proteger las rutas privadas.
SelectDisease.jsx	Desplegable que muestra las enfermedades disponibles para que el usuario pueda seleccionarlas.
SidePanel.jsx	Se trata de una ventana flotante que integra el resto de los componentes para ofrecer al usuario una interfaz de configuración del mapa. En la parte superior derecha de la **¡Error! No se e ncuentra el origen de la referencia.** podemos observar este panel.
TimeIntervalSelector.jsx	En este cuadro de texto podemos indicar el intervalo de tiempo en días con el que queremos solicitar los datos.
TimeSlider.jsx	Barra lateral deslizable que nos permite navegar por una línea temporal que va desde 2022 (inicio de los datos) hasta el día de hoy, a la vez que vemos en el mapa como la información que vemos representada se va modificando. También cuenta con un botón que avanza el tiempo de forma automática cada 500 milisegundos.

Tabla 4.7: Descripción de cada archivo contenido en el directorio "components/".

En el siguiente directorio “pages/”, encontramos todas las páginas principales por las que podremos navegar dentro de nuestra aplicación.

Loading

El funcionamiento de esta página es únicamente decorativo. A la hora de pulsar el botón de ejecución, veremos una ruleta girando mientras que recibimos los datos que el backend está procesando.

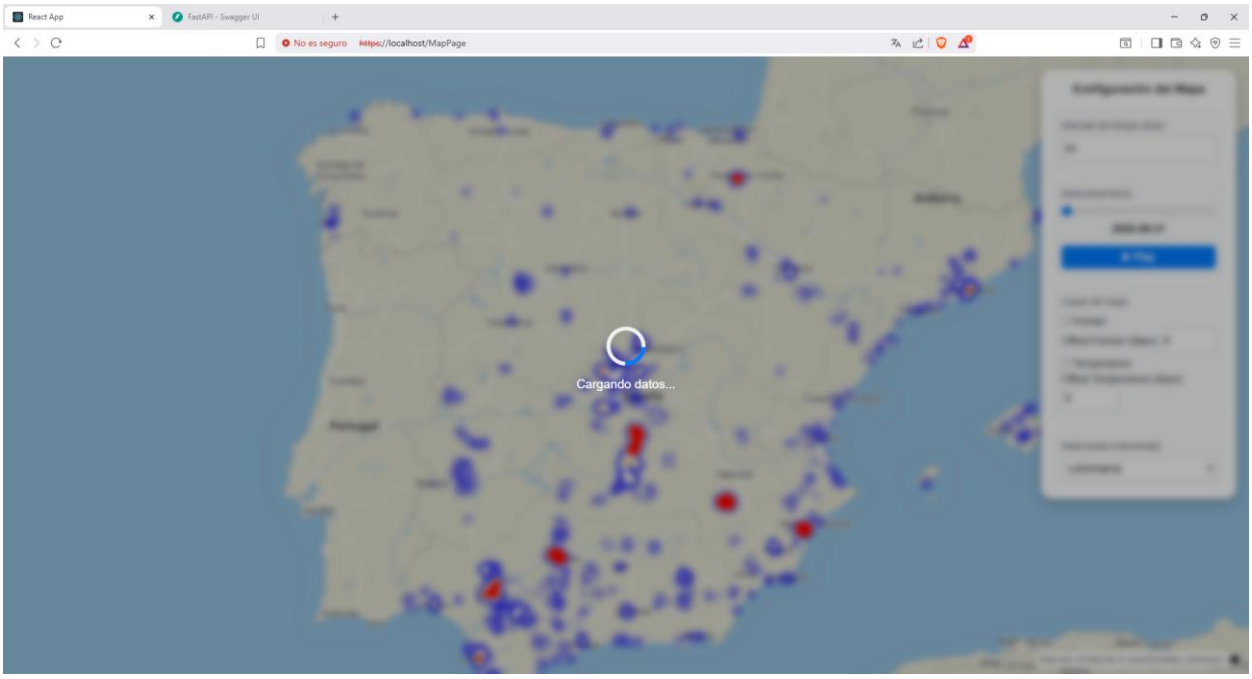

Figura 4-5: Imagen de la pantalla de Loading

Fuente: Captura de pantalla de la aplicación.

Login

Esta página es la primera con la que se encontrarán los usuarios. En ella vemos un formulario para rellenar nuestras credenciales (usuario y contraseña). También podremos acceder al registro.

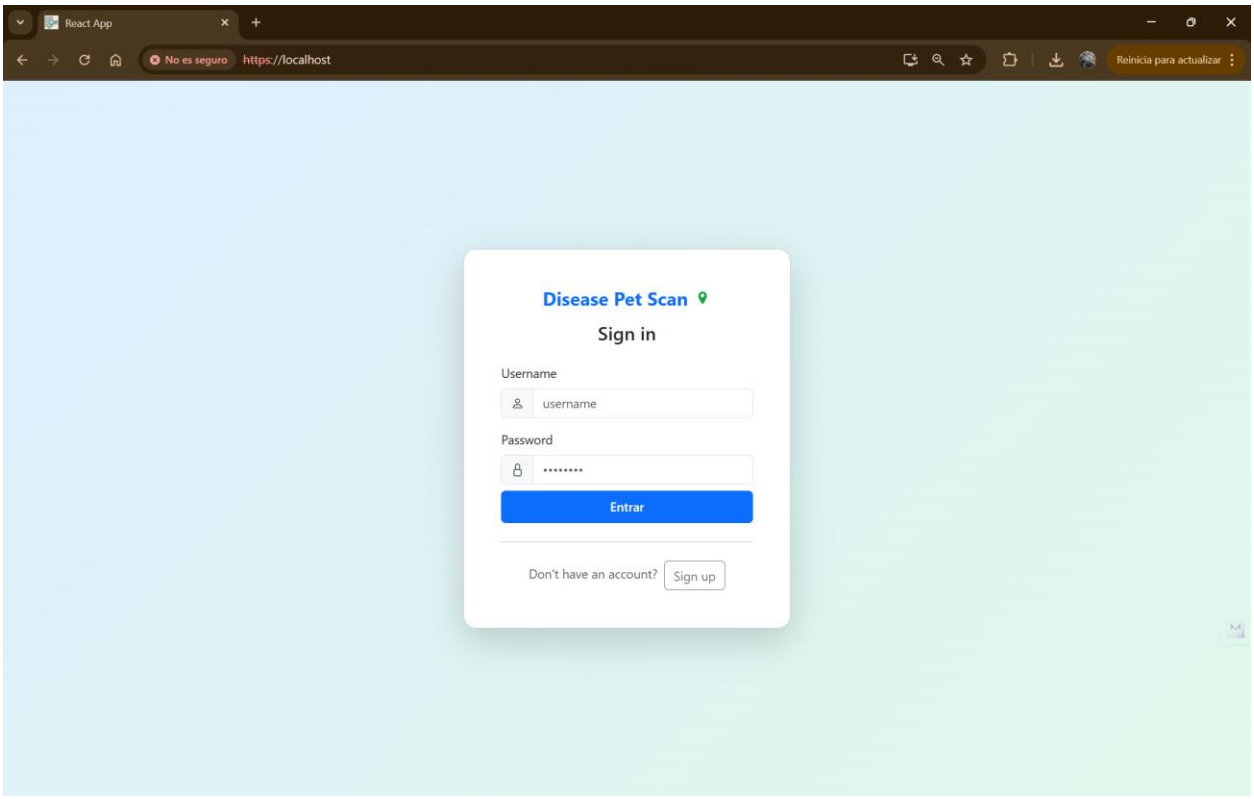

Figura 4-6: Imagen de la pantalla de inicio de sesión.

Fuente: Captura de pantalla de la aplicación.

MapPage

Página principal de la aplicación. Se encarga de renderizar el mapa interactivo junto con el panel lateral, permitiendo la visualización y configuración de los datos. También gestiona las peticiones al backend para obtener la información pedida por el usuario de forma simultánea y almacena tanto los datos recibidos como información adicional que requiere la aplicación, como los parámetros de búsqueda seleccionados (intervalo de días o la enfermedad seleccionada), el estado de carga actual (si se están solicitando datos al backend) y otras variables que permiten al usuario la interacción con el mapa, como las capas visibles y la fecha seleccionada. En la Figura 4-7 se muestra el mapa junto con el panel para ajustar los parámetros.

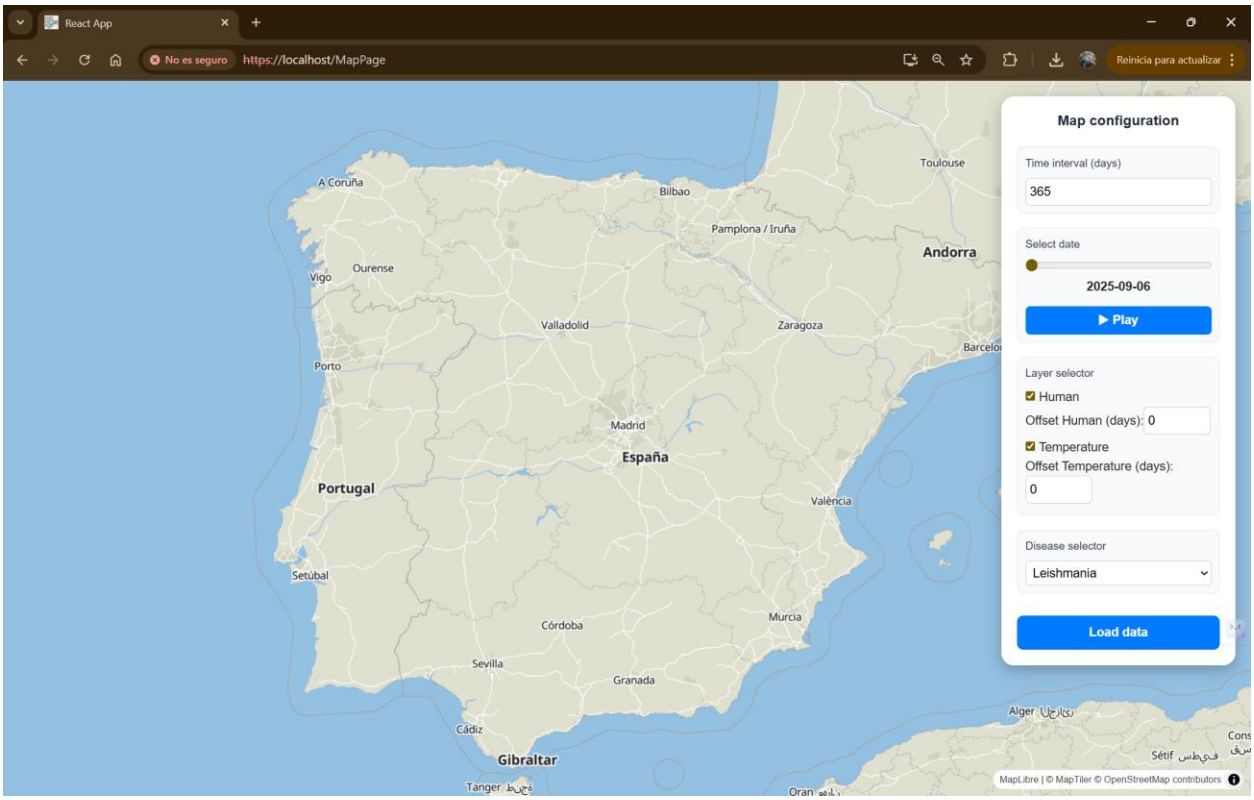

Figura 4-7: Imagen de la pantalla del mapa interactivo con el panel lateral.

Fuente: Captura de pantalla de la aplicación.

Register

En esta página tendremos un formulario para introducir nuestra información (Nombre de usuario, correo, nombre completo y contraseña) y quedar registrados como usuarios en la base de datos en caso de que la información introducida se válida.

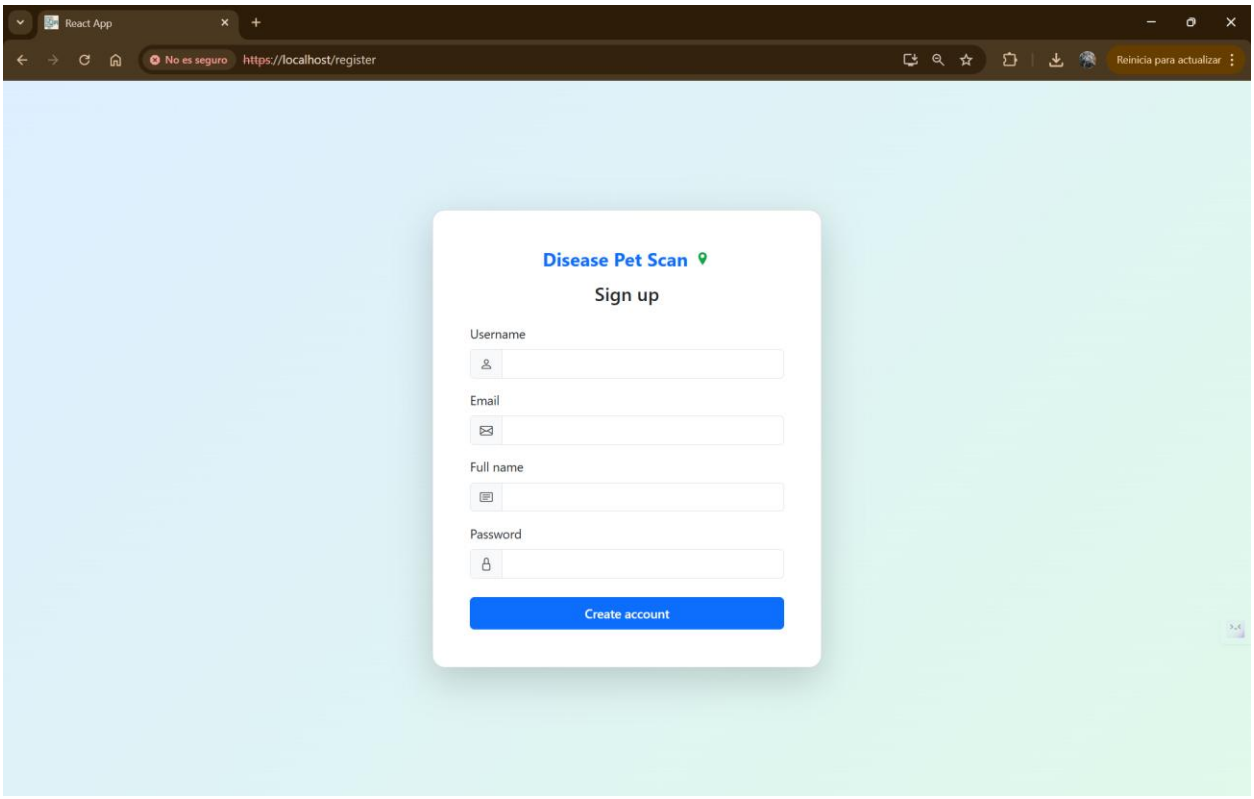

Figura 4-8: Imagen de la pantalla de registro.

Fuente: Captura de pantalla de la aplicación.

En el directorio "context/" encontramos el archivo AuthContext.jsx el cual gestiona la autenticación de los usuarios en la aplicación usando la función createContext() de React.

En este archivo se crea un contexto global de forma que todos los componentes puedan acceder a la información del usuario y saber si este está autenticado. Además, se solicita al backend un token de autenticación el cual se almacena y se utiliza posteriormente en cada consulta que realice el usuario para demostrar que dicho usuario está autorizado. En caso de que el token no exista o sea inválido, el backend bloqueará el acceso.

Por último, tenemos la carpeta de "styles/" donde veremos el fichero LoadingScreen.css, Login.css y Register.css los cuales son responsables de darle un estilo visual atractivo a cada una de las pantallas como podemos observar en la Figura 4-5, la Figura 4-6 y la Figura 4-8.

4.4.4 Integración con mapas interactivos

El desarrollo del mapa se ha realizado a través de la herramienta MapLibre utilizando su herramienta MapLibre GL JS [14] la cual utiliza WebGL que permite a los navegadores web renderizar figuras geométricas con un gran rendimiento.

Primero creamos la componente de React llamada Heatmap, la cual recibe como parámetros los datos de tests, AEMET, humanos y capas seleccionadas. Estas variables estarán siendo observadas en todo momento para poder actualizar la información en tiempo real cuando se detecten cambios en cualquiera de ellas.

El mapa se inicializa con una herramienta que ofrece MapLibre y se centra en el medio de la península Ibérica, el estilo del mapa viene determinado por una herramienta que ofrece plantillas para mapas a través de su API llamada Maptiler [15].

Después de inicializarlo, se generan fuentes de datos vacías con formato de geojson las cuales se actualizan en el momento en el que se detecten cambios en las variables que almacenan la información recibida desde el backend. Estas fuentes cuentan con los siguientes nombres:

- **disease-source**: resultados de test en mascotas.
- **aemet-source**: datos de temperatura ofrecidos por la AEMET.
- **human-source**: resultados de test en humanos.

A continuación, creamos las tres capas principales que van a representar estas tres fuentes de datos, una cada una.

- **disease-layer**: Esta capa representa en formato de heatmap los puntos calientes y fríos en función del resultado normalizado obtenido por el Getis Ord.
- **aemet-layer**: Obtiene una malla de puntos que cubre toda la superficie de España y los representa en un gradiente de colores dependiendo de la temperatura de cada punto.

- **human-layer**: Representa círculos indicando la incidencia de casos en humanos.

Para cada una de estas capas se han añadido capas invisibles (hover-layers) cuya función es detectar el movimiento del ratón con el fin de mostrar información adicional en el punto en el que el usuario esté interesado.

En el caso de humanos, al desplazar el cursor hacia uno de los puntos de incidencia se mostrará un "popup" con el código postal y el número de casos.

Para la capa de la AEMET veremos en cada puto su temperatura correspondiente.

Por último, en el caso de las enfermedades en mascotas si movemos el cursor a un punto de incidencia, se mostrará el código postal, el valor Z obtenido por el Getis Ord y número de positivos dentro de dicho código postal.

4.4.5 Uso de la API

El frontend establece una comunicación con el backend realizando solicitudes HTTP POST a través de la librería Axios [16]. Estas solicitudes envían parámetros para configurar las consultas que realice el backend a la base de datos y este responde con un conjunto de datos procesados y estructurados en formato GeoJSON.

El flujo de comunicación es el siguiente:

- Solicitud desde el frontend: El usuario selecciona los parámetros de búsqueda deseados a través de la UI mostrada en la Figura 4-7 y selecciona el botón de ejecución. El frontend envía tres solicitudes al backend:
    - "/api/test/filtered": datos filtrados de enfermedades en mascotas.
    - "/api/aemet/get_data": datos climáticos de la AEMET.
    - "/api/human/get_human_data": datos de incidencia en humanos.

- Respuesta del backend: Una vez procesados y estructurados todos los datos, devuelve un JSON con la lista de elementos GeoJSON procesados.

```
[
  {
    "date": "2022-01-01",                // Fecha de inicio del intervalo
    "geojson": {
      "type": "FeatureCollection",
      "features": [
        {
          "type": "Feature",
          "geometry": {
            "type": "Point",             // Tipo de geometría (puede ser Point o Polygon)
            "coordinates": [0.0, 0.0]  // Coordenadas geográficas
          },
          "properties": {
            "post_code": "28001",        // Código postal
            // ... más properties, dependiendo de la fuente de datos que sea

          }
        }
        // ... más features para ese intervalo
      ]
    }
  },
  {
    "date": "2022-01-15",
    "geojson": {
      "type": "FeatureCollection",
      "features": [
        // Features del siguiente intervalo
      ]
    }
  }
  // ... más intervalos
]
```

Figura 4-9: Representación gráfica en formato JSON de la estructura de datos que utiliza la aplicación. Fuente: Elaboración propia.

- Actualización de los datos en el frontend: Se actualiza el estado de "geojsonList" el cual contenía la información anterior, esto es detectado por el componente Map.jsx y renderiza la nueva información obtenida.

4.5 Servidor web con Nginx

Para implementar el frontend se ha utilizado Nginx [17], un servidor web ligero y de alto rendimiento ampliamente utilizado. Se ha elegido por su capacidad para gestionar múltiples conexiones simultáneas de forma eficiente, lo que lo convierte en

una solución adecuada para servir aplicaciones web modernas y sirve como punto de partida para futuros despliegues en servidores.

En este proyecto, la función principal de Nginx es servir los archivos estáticos que se generan tras la compilación de la aplicación en React de manera que la interfaz web pueda mostrarse correctamente al usuario a través del navegador. Además, Nginx facilita la comunicación con el backend, cuando el usuario interactúa con la herramienta y requiere información de la base de datos o de los análisis estadísticos, las peticiones se redirigen a la API del backend y garantiza la coherencia del flujo de datos.

Nginx se configuró específicamente para este proyecto mediante un archivo *default.conf*. En dicho archivo se definen dos aspectos esenciales:

- Distribución del frontend: en la ruta raíz (/), Nginx sirve la aplicación web ya compilada. Gracias a la directiva "try_files", se garantiza que todas las rutas de React se resuelven correctamente en el cliente, lo que evita errores de navegación.
- Comunicación con el backend: en la ruta /api/, las peticiones se redirigen al servicio de backend en el puerto 8000 mediante la directiva "proxy_pass". Este mecanismo permite que la API se integre de manera natural con la interfaz de usuario y mantiene la trazabilidad de las peticiones a través de las cabeceras "HTTP Host", "X-Real-IP" y "X-Forwarded-For".

En cuanto a la seguridad, todas las comunicaciones se canalizan a través de HTTPS, utilizando un certificado auto firmado generado con Certbot [18]. Para ello, se configuró Nginx de manera que cualquier intento de conexión a través de HTTP (puerto 80) se redirija automáticamente a HTTPS (puerto 443), lo que refuerza la protección de los datos transmitidos.

En resumen, Nginx cumple las siguientes funciones en la infraestructura del sistema:

1. Servir los archivos del frontend de manera eficiente.
2. Canalizar las peticiones del usuario al backend a través de la API.

3. Asegurar la comunicación mediante HTTPS.

4.6 Justificación de la arquitectura elegida

En conclusión, la estructura del frontend ha sido diseñada de forma modular y separando claramente las distintas funcionalidades como la interfaz de control, la visión de los datos, la autenticación. Este facilita enormemente el mantenimiento, la escalabilidad y la limpieza del código.

El uso de un contexto global de autenticación que centraliza el uso del token del usuario y protege las rutas privadas impide que usuarios no autorizados accedan a funcionalidades protegidas de la aplicación.

En cuanto al uso de datos, todos ellos comparten un mismo formato, lo que facilita mucho su representación y escalabilidad. A su vez es un formato compatible con la herramienta de visualización utilizada, Maplibre, la cual es una de las que más funciones ofrecen de forma gratuita en cuanto a la representación geoespacial web se refiere.

Por último, en cuanto a la comunicación entre el frontend y el backend mediante solicitudes HTTP gestionadas por Axios junto con la configuración de Nginx permite servir el frontend mediante HTTPS, lo cual aporta una capa de seguridad adicional estableciendo autenticación básica y evitando accesos no autorizados.

Capítulo 5 - Despliegue de aplicación

En este capítulo se explicará la disposición de los contenedores, redes y volúmenes de Docker y como se monta la aplicación usando Docker Compose.

5.1 Acerca de Docker y Docker Compose

En el despliegue de la aplicación se ha utilizado Docker [19] [20], que es una plataforma para empaquetar y ejecutar una aplicación en un entorno aislado llamado contenedor. Estos contenedores son ligeros y contienen lo necesario para ejecutar la aplicación sin depender del host donde se aloja: sistema de ficheros, librerías, programas.... Además, Docker trabaja con otra serie de elementos:

- Docker Imágenes: es una plantilla de solo lectura con las instrucciones y dependencias necesarias para crear un contenedor.
- Dockerfile: es el fichero necesario que define como crear un contenedor Docker a través de su propio lenguaje.

Este Dockerfile define contenedores de forma individual pero cuando se quiere que dos o más contenedores se comuniquen entre sí ya sea mediante peticiones o compartiendo un espacio de almacenamiento común, es necesario pasar a usar Docker Compose. Docker Compose [21]es una herramienta para ejecutar varias instancias de contenedores de una aplicación, simplifica el proceso y hace sencillo manejar servicios, redes y volúmenes. Para ello utiliza un único fichero de configuración YAML.

5.2 Arquitectura de los contenedores

En el despliegue de la aplicación se generan 4 servicios correspondientes a los descritos en la Figura 4-3, además de 2 redes que comunican los contenedores entre sí y 2 volúmenes que guardan datos en el sistema de forma permanente: uno para las

entradas de la base de datos y el otro para guardar los archivos compilados del "front" que luego Nginx copiará y servirá.

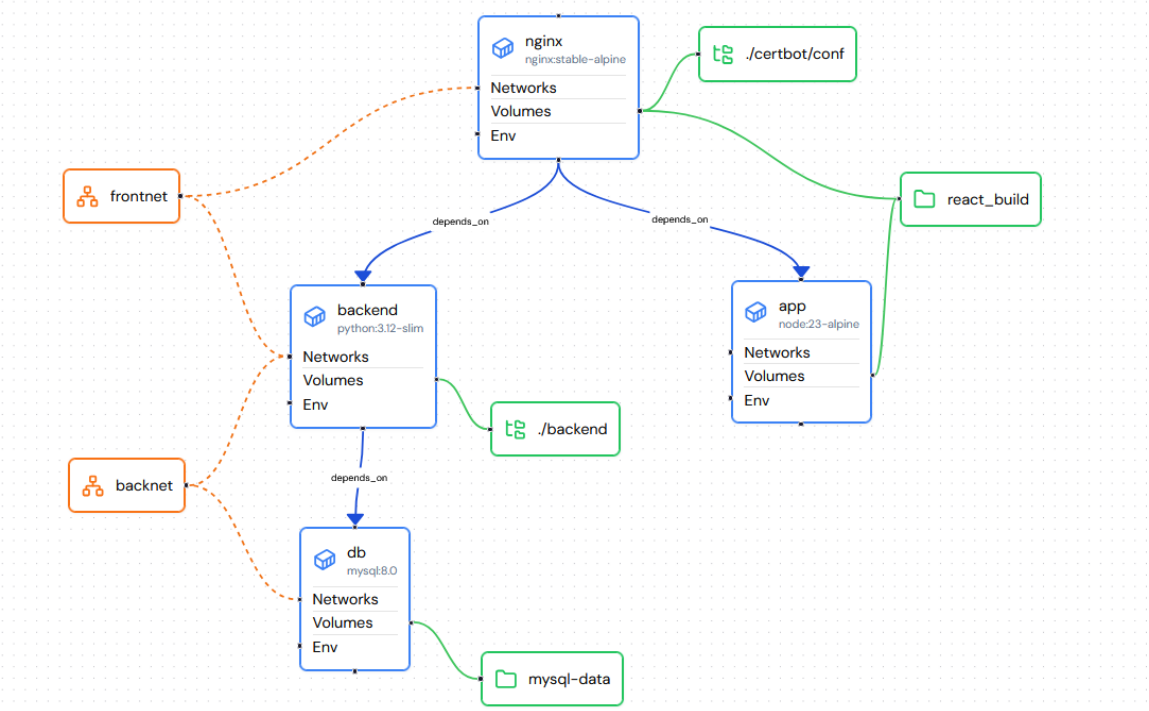

Figura 5-1: Representación visual de la estructura desplegada

En la Figura 5-1 se puede ver la disposición de la estructura, siendo los elementos naranjas las redes, los elementos verdes los volúmenes y los azules los contenedores.[1]

Cada servicio de la aplicación que se necesitaba personalizar tiene un Dockerfile dentro de su carpeta correspondiente que define como configura Docker cada contenedor, siendo aquellos los servicios del backend, frontend y Nginx. La base de datos es una excepción ya que se usa el contenedor original, por lo que se descarga en el archivo YAML de Docker Compose.

[1] La imagen se generó usando una herramienta creada por el usuario LucasSovre llamada Compose Craft [23]

En el Dockerfile se descarga una imagen ya existente del repositorio Docker Hub [22] ya preconstruida con la configuración de la tecnología ya dentro. Las imágenes descargadas son de tipo Alpine para los servicios de Frontend y Nginx, y Slim para Python. Esto reduce el peso de los contenedores y permite agregar solo las dependencias necesarias para el desarrollo, haciendo una aplicación más ligera y portátil.

Tras la descarga de la imagen, se pasa a la configuración del contenedor que establece su directorio interno de trabajo y expone los puertos en caso de ser necesario. Por último, el servicio finaliza con la ejecución de las instrucciones del archivo YAML de Docker Compose.

5.2.1 Servicio base de datos

Este servicio es el único sin archivo de configuración Dockerfile propio y el primero que se inicia.

```yaml
db:
  environment:
    - MYSQL_ROOT_PASSWORD=${MYSQL_ROOT_PASSWORD}
    - MYSQL_DATABASE=${MYSQL_DATABASE}
  ...
  restart: always
  healthcheck:
    test: ["CMD", "mysqladmin", "ping", "-h", "localhost"]
      interval: 10s
      retries: 5
      start_period: 10s
```

Figura 5-2: Configuración servicio base de datos

En la Figura 5-2 se observa parte de la configuración, omitiendo la declaración de la red y el volumen, que establece las variables de entorno que va a usar el contenedor y un chequeo que se realiza al terminar para comprobar que la base está lista para su uso. Esto se lleva a cabo antes de que se inicialicen el resto de los servicios,

con el fin de evitar errores de ejecución que podrían surgir si otro servicio realizara una consulta antes de que todo estuviera listo.

5.2.2 Servicio backend

En el Dockerfile, el backend copia el archivo de dependencias al volumen propio donde las instala y una vez terminado, se ejecuta utilizando la configuración ya establecida en el YAML.

```
backend:
    ...
    depends_on:
      db:
        condition: service_healthy
    ports:
      - "8000:8000"
    environment:
      - DB_DIALECT=${DB_DIALECT}
      - DB_USER=${DB_USER}
      - DB_PASSWORD=${DB_PASSWORD}
      - DB_HOST=${DB_HOST}
      - DB_NAME=${DB_NAME}
```

Figura 5-3: Configuración del servicio de backend

El servicio del backend solo puede iniciarse una vez se haya verificado que la base de datos se haya iniciado de forma exitosa, una vez haya acabado el chequeo. Además, se expone el puerto 8000 para poder acceder desde el exterior y se fijan las variables de configuración necesarias para acceder a la base de datos.

5.2.3 Servicio app

El servicio con nombre app es el servicio del frontend que, tras acabar de compilar la web, copia los archivos en el volumen compartido con Nginx definido en el YAML. Una vez finalizada la compilación el contenedor se cierra.

5.2.4 Servicio Nginx

El servicio Nginx del YAML define, además de los volúmenes suyos y compartidos con el frontend y las redes que va a usar para comunicarse, como se expone el puerto 443 para recibir y enviar solicitudes HTTP. Por último, copia los archivos resultados de la compilación del frontend del volumen y los envía a su directorio interno donde desde ahí los sirve.

5.3 Despliegue

Para el despliegue de la aplicación se tiene que seguir una serie de pasos:

- Clonar el repositorio y entrar en él.
- Crear un archivo .env, siguiendo como referencia el de ejemplo, en la carpeta raíz.
- Ejecutar el siguiente comando:

```
docker compose up --build
```

Tabla 5.1: Comando para construir la aplicación

- Una vez todos los contenedores estén ejecutándose hay que entrar en https://localhost para acceder a la página principal.
- Si se quiere entrar al backend para poder subir los Excel correspondientes hay que entrar en http://localhost:8000/docs.

Capítulo 6 - Conclusiones y trabajo futuro

6.1 Conclusión

Este Trabajo de Fin de Grado se ha centrado en la realización de un estudio espacio temporal de los datos de mascotas usando Getis Ord, acompañándolo con el desarrollo de una herramienta web interactiva de visualización de esos datos. Ambos objetivos, combinados con el estudio de estacionalidad con ARIMA y el estudio espacio temporal con casos de humanos, han arrojado una primera aproximación a un sistema que facilite la interpretación y la monitorización de los resultados.

Aunque la cantidad de los datos humanos ha resultado insuficiente para extraer conclusiones firmes al respecto y el estudio de estacionalidad relacionada a los casos de mascotas tampoco ha sido concluyente, aun mostrando cierta tendencia. Estos hallazgos, junto con la herramienta desarrollada, pueden servir de apoyo para la detección temprana de focos de infección y para el diseño de medidas de prevención tanto en el ámbito veterinario como en el humano.

Por tanto, este trabajo constituye una primera aproximación en el análisis de enfermedades de mascotas con datos espacio-temporales, sirviendo como base sobre las que futuras investigaciones se podrán apoyar para ampliarse y enriquecerse con nuevos datos, de otras enfermedades, y/o metodologías más avanzadas.

6.2 Trabajo futuro

Se proponen varias líneas de trabajo futuro basándose en los resultados conseguidos:

- Añadir funcionalidades adicionales a la herramienta web como filtros avanzados de búsqueda que permitan filtrar los códigos postales más afectados.

- Mejorar las limitaciones observadas en el modelo ARIMA mediante la aplicación de técnicas de aprendizaje automático, como las redes neuronales recurrentes (LSTM), con el fin de detectar posibles patrones de estacionalidad y generar predicciones más precisas.
- Replicar la metodología en otras zonas geográficas o con otras enfermedades que puedan afectar tanto a animales como a humanos.
- Mejorar la recopilación de casos de mascotas, humanos y censos, ya sea mediante la incorporación de una sección específica en la aplicación web para cargar archivos CSV o a través de conexiones directas con APIs oficiales que centralicen estos datos.
- Incluir datos históricos adicionales sobre mascotas y humanos, lo que permitiría obtener un análisis más completo y aumentar la fiabilidad de los resultados del estudio.
- Garantizar que los endpoints de la API estén protegidos, de manera que solo usuarios autenticados puedan acceder a las operaciones más sensibles como las de acceso a los datos.
- Reestructurar el despliegue del frontend para usar una construcción multietapa de Docker, de modo que Nginx sirva directamente la construcción completa de React con los archivos compilados y los certificados SSL, ya que no tiene sentido mantener un contenedor de aplicaciones que compile React, copie la construcción a un volumen y luego la recoja Nginx. Esto simplifica la arquitectura.
- Ser capaces de poder migrar el contenido de la base de datos si se desea desplegar la aplicación en otro lugar o alojar dicha base de datos en un servidor de forma estática.

Introduction

Motivation

The coexistence of pets in homes is an integral part of our society and brings both benefits and certain risks, which must be considered to live together harmoniously. In Spain, the number of pets has risen to more than 20 million in 2025, including dogs, cats, fish, birds and other animals such as reptiles and rodents, according to data from the ANFAAC Association [1].

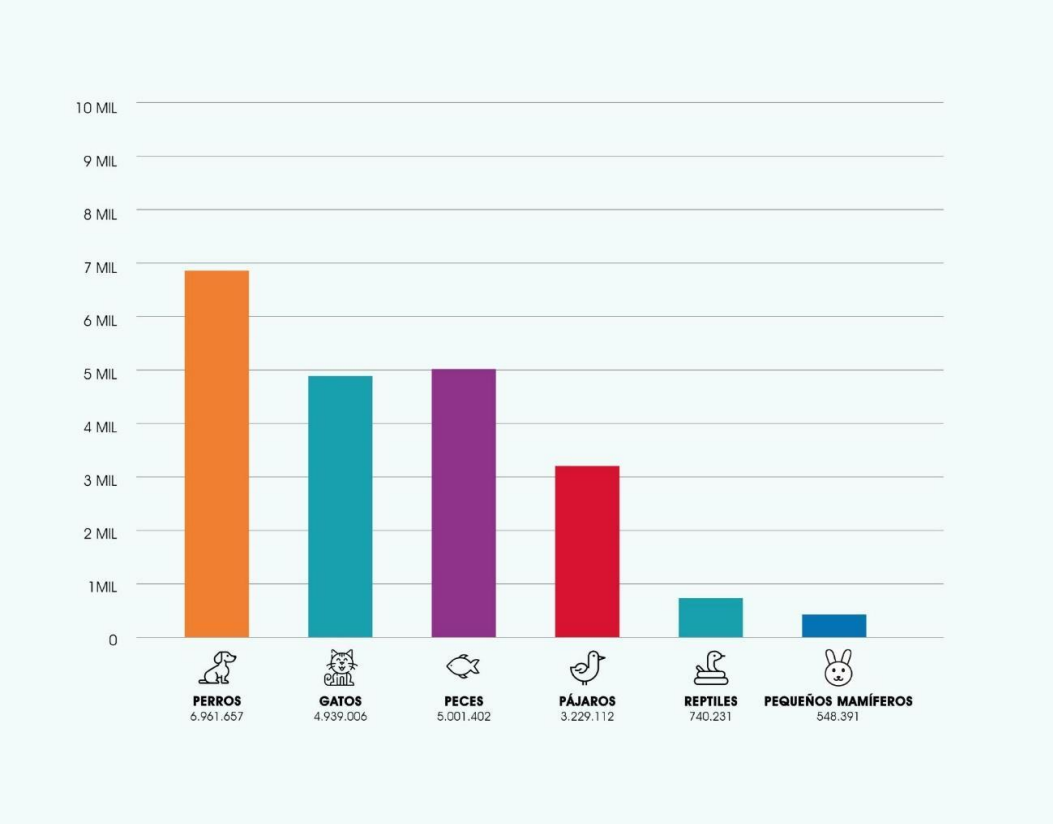

Figure 6-1: Graph of the 2025 pet census.

Source: ANFAAC Association

One of the major concerns caused by the large increase in these companion animals is the possibility that they may become sick and transmit diseases, which would generate invisible waves of infected animals. This concern is increasing in cases where pets can spread contagious diseases to humans, especially those with compromised immune systems, creating a public health problem due to inefficient monitoring. Therefore, there is a growing need for some kind of tool to monitor infections and their evolution over time.

Goals

The aim of this Bachelor's degree is to develop a user-friendly web tool to help control the spread of pet diseases, focusing specifically on Leishmaniasis and Giardia. A space-time analysis will be performed to identify areas of high disease incidence in pets, particularly dogs and cats, which pose a significant risk to immunocompromised individuals. This tool will be designed with scalability in mind so that it can grow as the study's needs evolve. Various data sets will be used to carry out the study, including: clinical diagnostic data provided by Uranovet [2] and CISA (INIA-CSIC), records of disease incidence in humans, pet census data, and meteorological information provided by AEMET.

The aims of the project are as follows:

- To identify and apply the appropriate statistical algorithms to analyse the incidence of diseases in pets and their possible seasonal variation.
- To investigate the possible relationship between the onset of diseases in pets and their transmission to humans. The aim is to identify common patterns, correlations and epidemiological links that will help us to understand the role of pet animals in spreading certain diseases to the human population.
- To develop a structured system for storing and updating information received from various sources, which is designed to ensure scalability when new, relevant information is added to the study.

- To develop a simple, interactive web interface that visualises the spatial and temporal incidence of different diseases in pets and humans on a map of Spain, facilitating comparative studies between the two populations.

Work plan

The points to be followed, as defined by the work objectives, are described below:

- A study of the most fitting statistical methods for spatial and temporal data analysis. An in-depth analysis will be conducted to determine the most suitable approach for application. Additionally, a study will be conducted to determine whether the pet data exhibits seasonality.
- Design of the organisation's application structure, providing a foundation on which to build the essential elements and enabling the addition of functionalities with ease.
- Design of deployment using Docker and Docker Compose to create containers and orchestrate the coordinated execution of the application.
- Designing a relational data model and implementing it in a database engine.
- Cleaning and preparation of raw data for subsequent statistical processing.
- Development of the application's backend, implementing the endpoints and functionalities necessary for its operation.
- Implement the frontend with a web interface structured in several layers to represent different information.
- Configure a web server responsible for hosting the frontend and managing communication with the backend.

Conclusions and future work

Conclusions

The focus of this Bachelor's degree project has been on conducting a spatial and temporal study of pet data using Getis Ord, as well as developing an interactive web tool for visualising this data. Combining these objectives with a study of seasonality using ARIMA and a spatiotemporal study of human cases has provided an initial approximation of a system that facilitates the understanding and monitoring of the results.

Despite the limited amount of human data available, which has prevented the formulation of definitive conclusions, the study has revealed a clear trend. The findings, along with the newly developed tool, can provide a foundation for the early identification of infection outbreaks and the creation of prevention strategies in both veterinary and human health contexts.

This work is a first step in analysing pet diseases using spatiotemporal data. It provides a foundation for future research, which can be built upon and enhanced with new data, additional diseases and/or more advanced methodologies.

Future Work

Based on the results achieved, several lines of future work are proposed:

- Add extra features to the web tool, such as advanced search filters that enable users to identify the postcodes most affected.
- Improve the limitations of the ARIMA model by applying machine learning techniques, such as long short-term memory (LSTM) recurrent neural networks, to detect possible seasonal patterns and generate more precise predictions.
- Apply the same methodology to other geographical areas or diseases that affect both animals and humans.

- Improve the collecting of pet and human cases and census data, either by adding a specific section to the web application for uploading CSV files or through direct connections to official APIs that centralise this data.
- Include additional historical data on pets and humans, which would allow for a more complete analysis and increase the reliability of the study results.
- Ensure that API endpoints are protected so that only authenticated users can access them.
- Restructure the front-end deployment to use a multi-stage Docker build. This will enable Nginx to serve the complete React build, including the compiled files and SSL certificates. There is no point in maintaining an application container that compiles React, copies the build to a volume and then has Nginx pick it up. This simplifies the architecture.
- Be able to migrate the database content if you want to deploy the application elsewhere or host that database on a server statically.

CONTRIBUCIONES PERSONALES

Estudiante Alejandro Gómez Martín

Implementación del backend con FastAPI

Desarrollé la lógica del servidor web mediante el framework FastAPI. En este apartado definí los endpoints principales de la API, organizados en distintos módulos, y diseñé el esquema de comunicación entre el cliente y el servidor. Además, incorporé la validación de datos con modelos definidos en Pydantic, garantizando consistencia y robustez en las peticiones y respuestas.

Gestión de usuarios y autenticación

Implementé todo el sistema de gestión de usuarios en el backend. Esto incluyó el registro de cuentas en la base de datos, el cifrado y almacenamiento seguro de las contraseñas mediante algoritmos de hash, así como la creación de rutas protegidas a las que solo pueden acceder usuarios autenticados. Para ello, configuré un sistema de emisión y validación de tokens JWT que garantiza la integridad de las sesiones y el control de acceso a recursos sensibles.

Diseño y desarrollo de la base de datos SQL

Diseñé la estructura de la base de datos teniendo en cuenta las necesidades del proyecto. Para ello, diseñé un modelo relacional que organiza las tablas en función de los casos registrados, el censo de mascotas por código postal, los datos de usuario y los datos de casos humanos.

Estudio inicial con GetisOrd y Anselin's Local Moran I

Me encargué de realizar los primeros análisis estadísticos con los datos de mascotas, empezando por el estadístico Getis-Ord, con el que se pueden identificar posibles patrones espaciales y puntos de concentración anómala. Además, realicé pruebas comparativas con el estadístico Local Moran I de Anselin, lo que me permitió contrastar la solidez de los resultados y obtener una visión más completa de la distribución espacial de los casos.

Análisis de la estacionalidad con ARIMA

Evalué la capacidad del modelo ARIMA para modelar la serie temporal de casos semanales mediante un análisis detallado. Esto no solo implicó aplicar el modelo, sino también estudiar sus limitaciones ante la escasez de datos y la presencia de valores atípicos. Este trabajo me permitió identificar las debilidades del modelo ARIMA en este contexto y abrir la puerta a la exploración de otros modelos predictivos más avanzados en el futuro.

Análisis de los datos de humanos

Amplíe el estudio para incluir también la serie temporal semanal de casos humanos. Este análisis paralelo permitió explorar posibles conexiones entre la evolución de la enfermedad en ambos grupos y reflexionar sobre la importancia de adoptar un enfoque conjunto de salud pública y veterinaria. La comparación entre ambas series de datos aportó información complementaria que enriquece la interpretación global del problema.

Configuración del servidor web Nginx

Implementé y configuré el servidor web Nginx como pieza central del despliegue de la aplicación. Su función principal fue servir los archivos estáticos del frontend ya compilados y redirigir las peticiones de la API al backend. Con el fin de reforzar la seguridad y dar coherencia al proyecto, configuré certificados autofirmados que habilitan la navegación bajo el protocolo HTTPS de modo que se ofrece un entorno más realista.

Diseño de la arquitectura con Docker

Organicé el despliegue del sistema completo mediante contenedores Docker, lo que permitió separar de manera modular los distintos servicios (frontend, backend, base de datos y servidor Nginx). Definí los volúmenes necesarios para compartir recursos entre contenedores, así como las redes de comunicación internas que permiten la correcta interacción de los módulos. El objetivo era garantizar la portabilidad, escalabilidad y facilidad de replicación del proyecto en diferentes entornos.

Estudiante Pablo Blanco Carrasco

Preprocesado de datos

Desarrollé scripts capaces de leer ficheros, normalizar datos , eliminar inconsistencias y aplicar transformaciones a los datos para su posterior análisis garantizando que cualquier CSV con el formato adecuado pudiera añadirse a nuestra base de datos de manera automática y evitando cualquier riesgo.

Diseño del backend y comunicación con la base de datos

Diseñé una versión inicial del backend utilizando FastAPI mediante la cual ha ido evolucionando el proyecto. Para ello modularicé la aplicación en varias carpetas para donde definí distintos apartados como la conexión a la base de datos, así como los primeros modelos de datos como "tests" o "post_codes" ya que esta versión únicamente mostraba datos en mascotas. Por último, diseñé el modelo de datos utilizado en el proyecto para la representación visual. Este consiste en un JSON el cual define una lista de geoJSONS separados por intervalos de tiempo. Se puede ver la estructura en Figura 4-9.

Datos meteorológicos y humanos

Diseñé y desarrollé los endpoints necesarios para la integración de datos provenientes de la API de la AEMET y de incidencias en humanos, cada uno con su respectiva tabla en la base de datos. En el caso de los datos meteorológicos implementé un método de interpolación de temperaturas sobre una malla de putos que cubriese toda la superficie de España con el objetivo de lograr un mapa de temperaturas consistente. Por último, gracias a la librería apscheduler implementé proceso que se encarga de mantener la tabla de datos meteorológicos de la AEMET siempre actualizada. Esto automatiza aún más la aplicación y mejora la experiencia de usuario.

Diseño del frontend

Diseñé y desarrollé el frontend , gestionando tanto el apartado gráfico, como la comunicación con el backend a través de solicitudes gestionadas por la herramienta Axios. Utilicé React para crear una estructura modular y escalable incluyendo componentes especializados para la interacción con el mapa implementado mediante LibreMap GL. También implementé un sistema de autenticación y autorización basado en tokens y contextos globales , de forma que solo usuarios registrados puedan acceder a la aplicación.

Optimización de la aplicación

Con el objetivo de mejorar la experiencia del usuario y reducir los tiempos de espera al visualizar los datos solicitados a nuestra API, diseñé y desarrollé un método para precalcular los resultados. En lugar de calcular los valores de Getis Ord de forma dinámica cada vez que se solicita la visualización, los resultados se generan previamente y se almacenan en la base de datos. Para hacerlo, definí los siguientes intervalos de tiempo: 15, 30, 60, 90 y 120 días. Cada intervalo corresponde a un conjunto de datos calculados previamente, de manera que cuando el usuario selecciona uno de estos rangos de tiempo, los datos ya están disponibles para su consulta, sin tener que procesarlos en tiempo real. La implementación está pensada para ser escalable, de forma que en el futuro se puedan añadir nuevos intervalos simplemente añadiendo el valor deseado en el array intervalos dentro de test_router.py.

Estudio de estacionalidad con ARIMA

De manera paralela al estudio que realizó Alex, analicé los resultados de aplicar este modelo estadístico sobre nuestros datos de test realizados en mascotas. Nuestro objetivo era analizar el comportamiento de los datos en función del tiempo y barajar la posibilidad de aplicar modelos de predicción para posteriormente ser capaces de implementar esta funcionalidad en nuestra aplicación web. Finalmente debido a la escasez de datos y valores anormales no fue posible generar predicciones fiables. A pesar de ello, este estudio ha sentado las bases para en un futuro cuando se disponga de una mayor calidad de datos poder ampliar las funcionalidades de nuestra aplicación.

BIBLIOGRAFÍA

[1] ANFAAC, «ANFAAC Datos Sectoriales,» ANFAAC, [En línea]. Available: https://www.anfaac.org/datos-sectoriales/. [Último acceso: 24 julio 2025].

[2] «Uranovet,» Uranovet SL, [En línea]. Available: https://www.uranodiagnostics.com/es-es. [Último acceso: 25 julio 2025].

[3] inigoflores, «Geometrias Codigos Postales España,» 12 08 2017. [En línea]. Available: https://github.com/inigoflores/ds-codigos-postales/tree/master.

[4] C. N. d. I. Geográfica, «Centro Nacional de Información Geográfica (CNIG),» Centro Nacional de Información Geográfica (CNIG), [En línea]. Available: http://centrodedescargas.cnig.es/CentroDescargas/buscadorCatalogo.do?codFamilia=02122. [Último acceso: 3 agosto 2025].

[5] I. d. S. CarlosIII, «Instituto de Salud CarlosIII,» [En línea]. Available: https://www.isciii.es/en/. [Último acceso: 2025].

[6] «GetisOrd,» ArcgisPro, [En línea]. Available: https://pro.arcgis.com/es/pro-app/latest/tool-reference/spatial-statistics/h-how-hot-spot-analysis-getis-ord-gi-spatial-stati.htm. [Último acceso: 29 julio 2025].

[7] Arcgis, «Anselin Local Moral,» Anselin, [En línea]. Available: https://pro.arcgis.com/es/pro-app/latest/tool-reference/spatial-statistics/h-how-cluster-and-outlier-analysis-anselin-local-m.htm. [Último acceso: 14` agosto 2025].

[8] «Queen Weights,» [En línea]. Available: https://docs.momepy.org/en/v0.7.2/user_guide/weights/weights_nb.html. [Último acceso: 29 julio 2025].

[9] Neptune, «Arima,» [En línea]. Available: https://neptune.ai/blog/arima-sarima-real-world-time-series-forecasting-guide. [Último acceso: 22 agosto 2025].

[10] S. Msac, «Arima vs Sarima,» Medium, [En línea]. Available: https://medium.com/@sophiamsac/arima-vs-sarima-vs-sarimax-03dd04fc7c66. [Último acceso: 27 agosto 2025].

[11] Mubarakdaha, «Arima,» Medium, [En línea]. Available: https://medium.com/@mubarakdaha/understanding-time-series-forecasting-with-arima-59cd7140d6c3. [Último acceso: 27 agosto 2025].

[12] FastAPI, «FastAPI,» [En línea]. Available: https://fastapi.tiangolo.com/. [Último acceso: 21 agosto 2025].

[13] Facebook, «React,» [En línea]. Available: https://es.react.dev/.

[14] MapLibre, «MapLibre GL JS,» [En línea]. Available: https://maplibre.org/maplibre-gl-js/docs/.

[15] MapTiler, «MapTiler: Maps for developers,» [En línea]. Available: https://www.maptiler.com/.

[16] Axios, «Axios HTTP,» [En línea]. Available: https://axios-http.com/es/docs/intro.

[17] Nginx, «Nginx,» [En línea]. Available: https://nginx.org/en/docs/index.html. [Último acceso: 1 agosto 2025].

[18] Certbot, «Certbot,» [En línea]. Available: https://certbot.eff.org/pages/about. [Último acceso: 29 agosto 2025].

[19] S. Hykes, «Docker,» Docker Inc, 13 marzo 2013. [En línea]. Available: https://docs.docker.com/get-started/docker-overview/. [Último acceso: 22 julio 2025].

[20] Docker, «Docker,» Docker, [En línea]. Available: https://www.docker.com/. [Último acceso: 1 agosto 2025].

[21] «Docker Compose,» Docker Inc, 12 marzo 2013. [En línea]. Available: https://docs.docker.com/compose/.

[22] «Docker Hub,» [En línea]. Available: https://hub.docker.com/.

[23] LucasSovre, «ComposeCraft,» 22 junio 2025. [En línea]. Available: https://composecraft.com. [Último acceso: 25 julio 2025].

[24] Folium, «Folium 0.20.0 documentation,» [En línea]. Available: https://python-visualization.github.io/folium/latest/.

[25] Pydantic, «Pydantic,» [En línea]. Available: https://docs.pydantic.dev/latest/. [Último acceso: 21 agosto 2025].

[26 MapLibre, «MapLibre,» [En línea]. Available: https://maplibre.org/.
]