UNIVERSITAT POLITÈCNICA DE VALÈNCIA

DISCA
DEPARTAMENTO DE INFORMÁTICA DE SISTEMAS Y COMPUTADORES

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Dpto. de Informática de Sistemas y Computadores

Desarrollo de una aplicación para el tracking de animales domésticos

Trabajo Fin de Máster

Máster Universitario en Ingeniería de Computadores y Redes

AUTOR/A: Wu, Liyu

Tutor/a: Manzoni, Pietro

CURSO ACADÉMICO: 2022/2023

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Universitat Politècnica de València

Desarrollo de una aplicación para el tracking de animales domésticos

Trabajo Fin de Máster

Autor: Liyu Wu

Tutor: Pietro Manzoni

Curso 2022-2023

Agradecimientos

En este proyecto de graduación, me siento profundamente honrado de contar con Pietro Manzoni como mi tutor, quien me ha brindado una inestimable orientación y apoyo. Deseo expresar mi más sincero agradecimiento a mi tutor.

Para este proyecto, decidí utilizar el marco de Django para desarrollar una aplicación de seguimiento de mascotas. A lo largo de todo el proceso de investigación, mi tutor me ha proporcionado valiosa orientación y consejos académicos. Desde la concepción inicial del proyecto hasta su implementación final, la profunda comprensión y dominio de mi tutor en el marco de Django me han inspirado y brindado muchas ideas. Además, su estímulo y motivación en el ámbito de la investigación académica han permitido que adquiera un entendimiento más profundo en ingeniería de redes, despertando en mí un gran interés por esta área.

Resumen

Este proyecto de diseño de graduación utiliza el framework Django como backend, el framework Vue como frontend e integra el módulo de mapas de alta definición (High Amap) de Amap para lograr la funcionalidad de seguimiento de mascotas, proporcionando registros médicos y servicios de alerta para las mascotas. Utilizando una base de datos MySQL, se implementa el almacenamiento, actualización y eliminación de datos en tablas como información del dueño, información de la mascota, registros médicos de mascotas e información de alertas.

El proyecto consta de dos aplicaciones: una para el dueño, que permite localizar en tiempo real tanto al dueño como a la mascota, registrar información de la mascota, registros médicos y funciones de alerta de la distancia entre el dueño y la mascota, así como registros de alertas. La otra aplicación está destinada para uso de las mascotas, donde los dueños de mascotas pueden iniciar sesión con el ID y el nombre de la mascota y ver en tiempo real la ubicación y la información de latitud y longitud en el mapa. Con el diseño e implementación de este proyecto, se ofrece un servicio de gestión de mascotas más completo y conveniente.

Palabras clave: Python, Seguimiento de mascotas, Marco de Django,Vue

Abstract

This graduate design project uses the Django framework as the backend, the Vue framework as the frontend and integrates Amap's High Amap module to achieve pet tracking functionality, providing medical records and alert services for pets. Using a MySQL database, it implements the storage, update and deletion of data in tables such as owner information, pet information, pet medical records and alert information.

The project consists of two applications: one for the owner, which allows real-time location of both the owner and the pet, recording pet information, medical records and alert functions of the distance between the owner and the pet, as well as alert records. The other application is intended for pet use, where pet owners can log in with the pet's ID and name and view real-time location and latitude and longitude information on the map. With the design and implementation of this project, a more comprehensive and convenient pet management service is provided.

Keywords: Python, Pet Tracking, Django Framework, Vue

Prefacio

Antecedentes de la investigación

Las mascotas son compañeros importantes en la vida de las personas, y su número sigue aumentando. Sin embargo, la pérdida y seguridad de las mascotas siempre han sido una preocupación para los dueños. Con el fin de gestionar y cuidar mejor a las mascotas, la tecnología de seguimiento de mascotas se ha convertido en un campo popular en la industria de las mascotas. La aplicación de seguimiento de mascotas, como una solución innovadora, puede ayudar a los dueños a localizar en tiempo real la ubicación de sus mascotas, registrar la información médica de las mismas y proporcionar servicios de alerta para garantizar su seguridad. Este trabajo de investigación tiene como objetivo estudiar y diseñar una aplicación de seguimiento de mascotas que utilice la tecnología de backend de Django y frontend de Vue, e integre el módulo de mapas de alta definición de Amap, para proporcionar a los dueños un servicio de gestión de mascotas más completo y conveniente.

Estado actual de las aplicaciones de seguimiento de mascotas

Actualmente, el mercado mundial de seguimiento de mascotas está experimentando un crecimiento significativo. Con el aumento de la adopción de mascotas, la tecnología de seguimiento de mascotas ha captado la atención y la demanda de un número cada vez mayor de dueños de mascotas. Las aplicaciones de seguimiento de mascotas utilizan tecnologías como GPS y Bluetooth para localizar en tiempo real la posición de las mascotas y ofrecen una amplia gama de funciones y servicios. Algunas aplicaciones de seguimiento de mascotas con características innovadoras han sido bien recibidas por los consumidores y se han destacado en el mercado.

- Tractive: Tractive es una empresa de tecnología de seguimiento de mascotas con sede en Austria, fundada en 2012. Sus dispositivos de seguimiento de mascotas utilizan tecnología GPS para localizar en tiempo real la posición de las mascotas, y también ofrecen funciones de registro de trayectoria histórica y alertas. La interfaz de la aplicación de Tractive es simple y fácil de usar, es compatible con múltiples plataformas y es muy apreciada por los dueños de mascotas.

- Whistle: Whistle es una empresa estadounidense de tecnología para mascotas fundada en 2012. Su dispositivo de seguimiento de mascotas, Go Explore, utiliza la red 4G LTE de AT&T, lo que le proporciona una excelente precisión de ubicación y confiabilidad de la red. La aplicación de Whistle ofrece una amplia gama de funciones, con una interfaz de usuario intuitiva y amigable. Además de la localización en tiempo real de las mascotas, también ofrece monitoreo de datos de salud y función de luz intermitente.

- Fi Smart Dog Collar: Fi es una empresa especializada en productos inteligentes para mascotas. Su Smart Dog Collar utiliza tecnología Bluetooth y GPS para rastrear la ubicación y datos de actividad de las mascotas. La aplicación de Fi ofrece funciones de localización, monitoreo de actividad, cercas electrónicas y compartir en comunidades, lo que la convierte en una aplicación de seguimiento de mascotas completa.

- Pawtrack: Pawtrack es un fabricante de dispositivos de seguimiento de mascotas diseñados especialmente para gatos. Sus dispositivos utilizan tecnología GPS y Bluetooth para rastrear la ubicación de los gatos en tiempo real. La interfaz de su aplicación es simple y fácil de usar, enfocándose en la localización y servicios de gestión para gatos.

- Link AKC: Link AKC es una empresa dedicada a productos inteligentes para mascotas. Su Smart Collar utiliza tecnología GPS para localizar a las mascotas y ofrece funciones de monitoreo de actividad. La aplicación de Link AKC tiene una amplia gama de funciones, incluyendo registros de salud, cercas electrónicas e interacción en comunidades, lo que la convierte en una potente aplicación de seguimiento de mascotas.

Desafíos actuales en las aplicaciones de seguimiento de mascotas

Con el desarrollo de la sociedad y el aumento del nivel de vida de las personas, cada vez más familias eligen tener mascotas. Las mascotas gradualmente se han convertido en parte de la familia, brindando alegría y compañía a las personas. Según el "Informe de Tendencias de Consumo de Mascotas en China de 2021", el tamaño de la industria de cría de mascotas en China ha estado aumentando año tras año, y las mascotas como perros y gatos se han convertido en las especies de mascotas más populares. El aumento en el número de mascotas ha llevado a una mayor demanda de seguridad y gestión de mascotas, lo que ha ampliado la aplicación de las aplicaciones de seguimiento de mascotas.

El aumento en el número de mascotas también ha llevado a un aumento en la probabilidad de que las mascotas se pierdan. Según las estadísticas, más del 80% de las mascotas perdidas no pueden ser encontradas nuevamente. La pérdida de mascotas causa un gran dolor y angustia tanto a los dueños como a las propias mascotas. En entornos urbanos, es bastante común que las mascotas se pierdan o se extravíen. Por lo tanto, el desarrollo de aplicaciones de seguimiento de mascotas tiene una gran importancia para prevenir eficazmente la pérdida y recuperar rápidamente a las mascotas perdidas.

La salud y el cuidado médico de las mascotas son cuestiones de gran preocupación para los dueños de mascotas. Con el aumento de la conciencia sobre la salud de las mascotas, el mercado de atención médica para mascotas continúa expandiéndose. Según el "Informe sobre Consumo de Salud de Mascotas en China de 2023", el mercado de salud de mascotas muestra una tendencia de rápido crecimiento, con un aumento continuo en el consumo de atención médica para mascotas. Las aplicaciones de seguimiento de mascotas pueden proporcionar registros médicos de mascotas a los dueños, ayudándolos a estar al tanto de la salud de sus mascotas y sus necesidades médicas.

A pesar del progreso logrado en las funciones y el rendimiento de las aplicaciones de seguimiento de mascotas, todavía existen algunos desafíos en su aplicación práctica. Por ejemplo, algunos dispositivos de seguimiento pueden tener limitaciones en la precisión y frecuencia de actualización de la ubicación, lo que afecta la precisión de los datos de posición. Además, la vida útil de la batería de algunos dispositivos es corta, lo que requiere una carga frecuente y afecta la experiencia del usuario. Además, algunas aplicaciones de seguimiento de mascotas pueden tener funciones de alerta que no son lo suficientemente rápidas ni sensibles para garantizar eficazmente la seguridad de las mascotas.

Innovación y solución a los problemas clave

El punto innovador de este trabajo de investigación radica en la combinación de la tecnología de backend de Django con la tecnología de frontend de Vue, e integrar el módulo de mapas de alta definición de Amap para diseñar y desarrollar una aplicación de seguimiento de mascotas. Esta aplicación incluirá dos aplicaciones, una para los dueños y otra para las mascotas, que proporcionarán funciones para localizar en tiempo real a las mascotas y a sus dueños. Además, la aplicación registrará la información médica de las mascotas y generará una alerta cuando la distancia entre el dueño y la mascota supere un valor preestablecido, garantizando así la seguridad de las mascotas. Para mejorar la precisión y la experiencia del usuario en la aplicación de seguimiento de mascotas, este trabajo de investigación implementará una serie de medidas técnicas, como optimizar el algoritmo de localización por GPS, prolongar la vida útil de la batería y mejorar la velocidad de respuesta de las alertas.

Tabla de contenidos

1. Introducción 10
1.1 Motivación 10
1.2 Objetivo 11
1.3 Impacto Esperado 11
1.4 Metodología 12
1.5 Estructura 12
2. Estado del arte 13
2.1 Tractive app 13
2.2 Whistle 19
2.3 FI app 22
2.4 Pawtrack app 26
2.5 Findster app 32
3. Herramientas utilizada 36
3.1 MVC y MTV 37
3.1.1 Arquitectura MVC 37
3.1.2 Arquitectura MTV (Modelo-Plantilla-Vista) 39
3.2 Front-end and back-end separation/no separacion 41
3.2.1 Proyecto no separado de front-end y back-end 42
3.2.2 Proyecto de separación de front-end y back-end 43
3.3 Flask o Django 44
3.3.1 Flask (microframework ligero) 45
3.3.2 Django (framework Web completo) 46
3.4 Marco de front-end convencional 47
3.4.1 React 48
3.4.2 Vue.js 49
3.4.3 Angular 51

4. Solución propuesto 54
4.1 Plan de Trabajo 54
4.2 Arquitectura del Sistema 56
4.2.1 Parte front-end (Vue.js) 57
4.2.2 Parte backend (Django): 59
4.2.3 Diseño de la base de datos: 62
4.2.4 Proceso de interacción con el sistema: 64
4.2.5 Resumen 65
4.3 Diseño Detallado 65
4.3.1.Clases y relaciones: 65
4.3.2.Estructura de los módulos 68
4.3.3. Diseño de la base de datos: 71
4.3.4. estructura del catálogo: 71
4.4 Tecnología Utilizada 72
4.4.1. Introducción 72
4.4.2. Selección de tecnología 72
4.4.3. Entorno y sistema operativo 75
4.4.4. Pasos de desarrollo 75
4.4.5. Resultados esperados 77
4.4.6. Conclusión 77
5. Implantación 78
5.1 Instalación de backend de proyecto 78
5.2 Instalación de el frontend de proyecto 87
5.3 Bases de datos 88
5.4 Pruebas 90
5.5 Resultados 99
6. Conclusiones 100
Trabajo futuro 101
7. Referencias 102

1. Introducción

1.1 Motivación

En la sociedad moderna, las mascotas se han convertido en compañeros y miembros indispensables de la familia. Sin embargo, la gestión y seguridad de las mascotas siguen siendo un desafío constante para los dueños de mascotas. Los métodos tradicionales de gestión de mascotas a menudo no satisfacen la necesidad de un monitoreo y gestión en tiempo real de la ubicación de las mascotas, y también presentan algunas deficiencias, como una precisión de ubicación insuficiente y una función de alerta poco sensible. Con el objetivo de abordar estos problemas y proporcionar una solución de gestión de mascotas más conveniente y segura, hemos decidido desarrollar una nueva aplicación de seguimiento de mascotas.

Este proyecto utiliza el framework Django como backend y el framework Vue como frontend, e incorpora un módulo de mapas para crear una aplicación de seguimiento de mascotas con funciones avanzadas. Mediante la integración de tecnologías de geolocalización de vanguardia y características innovadoras, hemos desarrollado dos aplicaciones separadas, una para los dueños de mascotas y otra para las mascotas mismas.

Para los dueños de mascotas, hemos creado una aplicación específicamente diseñada para ellos. Con esta aplicación, los dueños de mascotas pueden rastrear en tiempo real su propia ubicación y la de sus mascotas. Gracias al soporte del módulo de mapas, los dueños pueden visualizar claramente la trayectoria de movimiento de sus mascotas y mantenerse informados sobre sus actividades. Además, los dueños pueden agregar información de sus mascotas, como nombre, edad y raza, así como registros médicos, lo que les permite gestionar la salud y el bienestar de sus mascotas de manera efectiva. Lo más importante es que hemos implementado una función de alerta que avisa a los dueños cuando la distancia entre ellos y sus mascotas excede un valor preestablecido. Si la mascota se aleja de un área segura definida, la aplicación enviará una notificación de alerta al dueño, garantizando así la seguridad de la mascota.

Por otro lado, también hemos desarrollado una aplicación independiente para las mascotas. Las mascotas pueden acceder a esta aplicación utilizando su ID y nombre, y ver en tiempo real su ubicación y coordenadas en el mapa. Esto permite establecer un vínculo más estrecho entre los dueños y las mascotas, permitiendo que los dueños se mantengan informados sobre el estado de sus mascotas en todo momento y brindando un servicio de gestión de mascotas más completo.

1.2 Objetivo

•Desarrollar una aplicación móvil utilizando Django que permita a los dueños de mascotas monitorear la ubicación de sus animales domésticos en tiempo real.

•Utilizar tecnología de geolocalización para realizar el seguimiento de las mascotas, y notificar a los dueños cuando sus animales salgan de zonas seguras predefinidas.

•Implementar un sistema de alertas que notifique a los dueños de mascotas cuando las baterías de los dispositivos de seguimiento se estén agotando.

•Integrar una interfaz para la gestión de perfiles de mascotas, que permita a los dueños de mascotas agregar información como nombre, edad, raza, historial médico y fotografías.

•Utilizar técnicas de aprendizaje automático para analizar los patrones de movimiento de las mascotas y mejorar la precisión de la ubicación.

1.3 Impacto Esperado

•Una aplicación móvil completamente funcional para el seguimiento de mascotas utilizando tecnología de geolocalización.

•Un sistema de alertas que notifique a los dueños de mascotas sobre la ubicación de sus animales y la duración de las baterías de los dispositivos de seguimiento.

•Un estudio de caso para la evaluación de la precisión de la ubicación de la aplicación y la eficacia del sistema de alertas utilizando un conjunto de datos de prueba.

•Un conjunto de recomendaciones para la mejora de la eficiencia y la precisión en la aplicación para el seguimiento de mascotas.

1.4 Metodología

•Investigación del estado del arte de las aplicaciones de seguimiento de mascotas, incluyendo las tecnologías y plataformas más utilizadas en el mercado.

•Desarrollo de la aplicación móvil utilizando Django y tecnología de geolocalización para el seguimiento de las mascotas.

•Implementación de un sistema de alertas para notificar a los dueños de mascotas sobre la ubicación de sus animales y la duración de las baterías de los dispositivos de seguimiento.

•Integración de una interfaz para la gestión de perfiles de mascotas y su información asociada.

• Utilización de técnicas de aprendizaje automático para analizar los patrones de movimiento de las mascotas y mejorar la precisión de la ubicación.

•Evaluación de la precisión de la ubicación de la aplicación y la eficacia del sistema de alertas utilizando un conjunto de datos de prueba.

1.5 Estructura

La estructura general de este artículo es la siguiente:

Capítulo 2, el estado actual de la técnica, es un estudio de varias aplicaciones populares de seguimiento de mascotas, la comprensión en profundidad de sus características implementadas, y la tecnología avanzada.

Capítulo 3, presenta los principales marcos de desarrollo de apps y sus características, las ventajas y desventajas entre ellos, para ayudarnos a encontrar rápidamente el marco adecuado para desarrollar la aplicación.

Capítulo 4, introduce la planificación del desarrollo del proyecto, el plan de trabajo, la estructura del sistema y el diseño de los detalles del proyecto, la aplicación ha tomado forma.

Capítulo 5, que llega a la fase formal de desarrollo y pruebas del proyecto, detalla la composición de cada componente funcional del proyecto.

Capítulo 6, es el resumen de todo el proyecto y la discusión del trabajo futuro.

2. Estado del arte

En este capítulo, se llevará a cabo una exploración de las aplicaciones de seguimiento de mascotas comunes en el mercado, investigando su estado tecnológico y su historia de evolución. Al mismo tiempo, se estudiarán las similitudes en funcionalidades con nuestro proyecto.

2.1 Tractive app

Tractive es una destacada aplicación de seguimiento de mascotas que ha recibido elogios por proporcionar un servicio excepcional a los dueños de mascotas mediante la integración de tecnologías avanzadas y características innovadoras. En este informe técnico más reciente, analizaremos en profundidad las ventajas de Tractive, cómo utiliza la tecnología para el rastreo de mascotas y determinaremos su precisión.

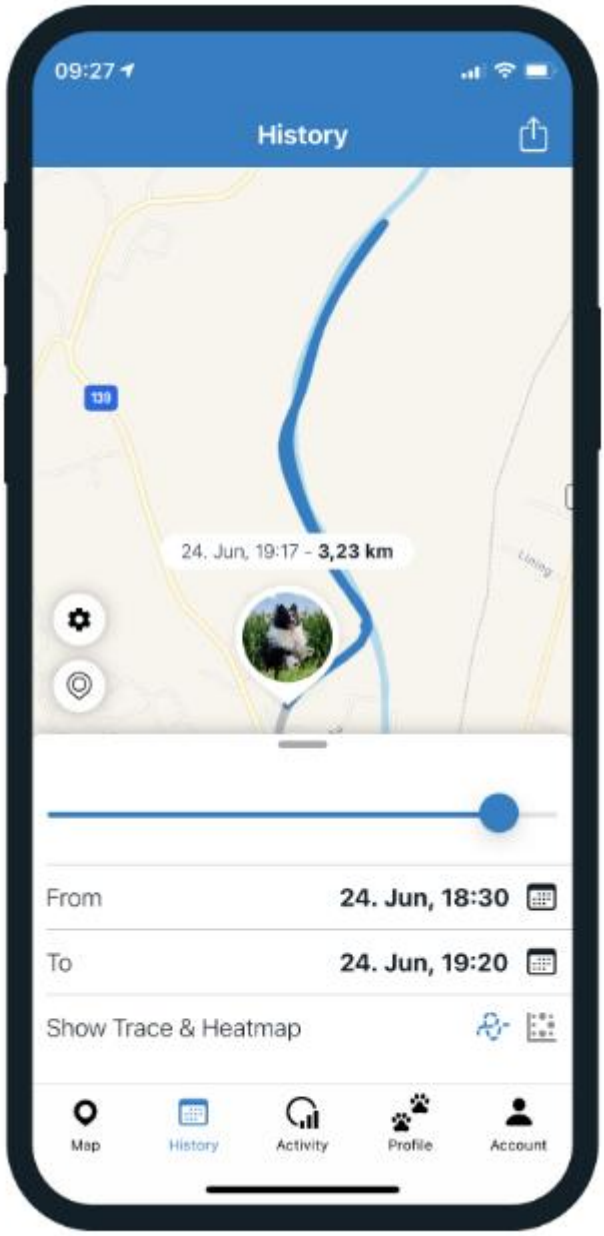
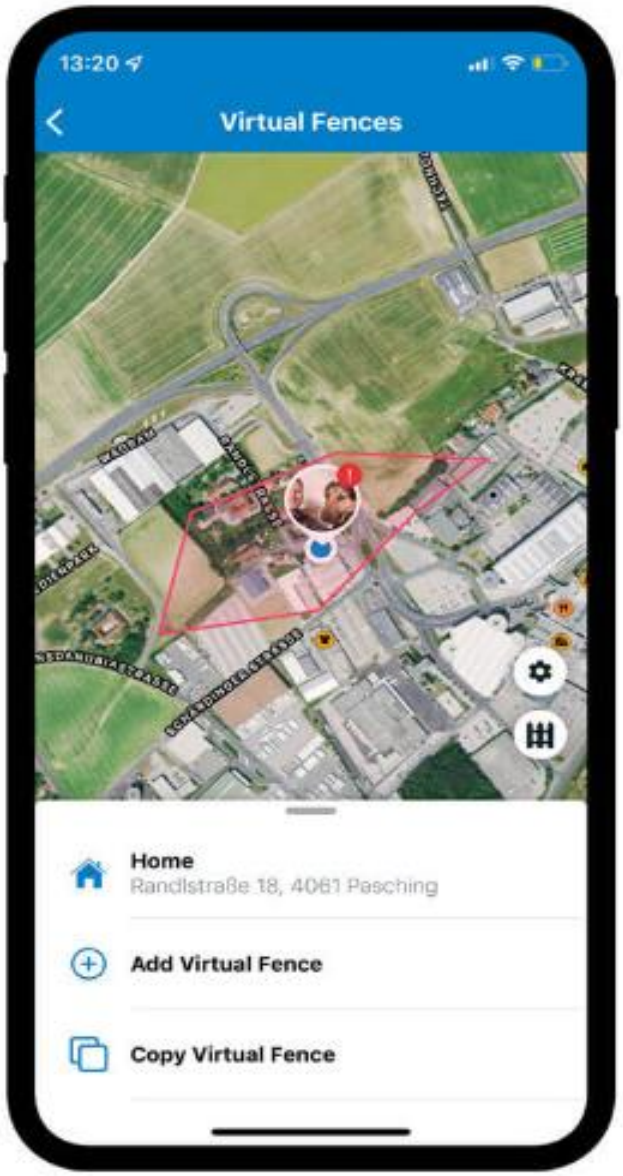


Figura 2.1.1 función de historial

La función de historial de ubicaciones es una de las interesantes y útiles características de Tractive GPS. Con esta función, los usuarios pueden fácilmente revisar y seguir las ubicaciones visitadas por sus mascotas dentro del rango de tiempo establecido. Además, la función de mapa de calor se muestra en la parte superior de la ruta de historial de ubicaciones, lo que permite observar los lugares donde su adorado compañero peludo ha pasado la mayor parte del tiempo.

Esta valiosa función proporciona una visión más detallada del recorrido y actividades de la mascota, lo que puede ser de gran utilidad para entender sus hábitos y comportamiento. Además, la capacidad de visualizar la distribución de las ubicaciones más frecuentadas mediante el mapa de calor facilita una mejor comprensión de los lugares que más le interesan a la mascota.

La función de historial de ubicaciones de Tractive GPS contribuye significativamente a la seguridad y cuidado de las mascotas, ya que permite a los dueños mantener un seguimiento preciso de sus movimientos y asegurarse de que se encuentren en entornos seguros y familiares. Con esta herramienta, los dueños pueden tomar decisiones informadas y brindar a sus adoradas mascotas una vida más feliz y saludable.



Frontera virtual del mapa.

Configure el nombre, la forma y el ícono de la frontera virtual y confirme si es una zona segura o una zona prohibida.

Zona segura, es el área donde normalmente desea que su mascota permanezca, y

Zona prohibida, representa áreas más peligrosas, como zonas con vida silvestre.

La frontera virtual puede ser circular, rectangular o una forma personalizada.

Puede desactivar la frontera virtual yendo a la aplicación, seleccionando el virtual que desea desactivar y haciendo clic en el interruptor al lado de "Activar frontera virtual". Cualquier frontera virtual inactiva se marcará en gris en la aplicación.

Figura 2.1.2 Frontera virtual

El Archivo y Monitoreo de Salud de Mascotas

Permite agregar y actualizar información detallada de la mascota, como raza, género, edad, nombre, peso, longitud corporal, registros médicos, entre otros, abarcando prácticamente toda la información relevante. Además, se puede monitorear en cualquier momento el estado de salud de la mascota, rastrear su sueño y medir su nivel de actividad física.

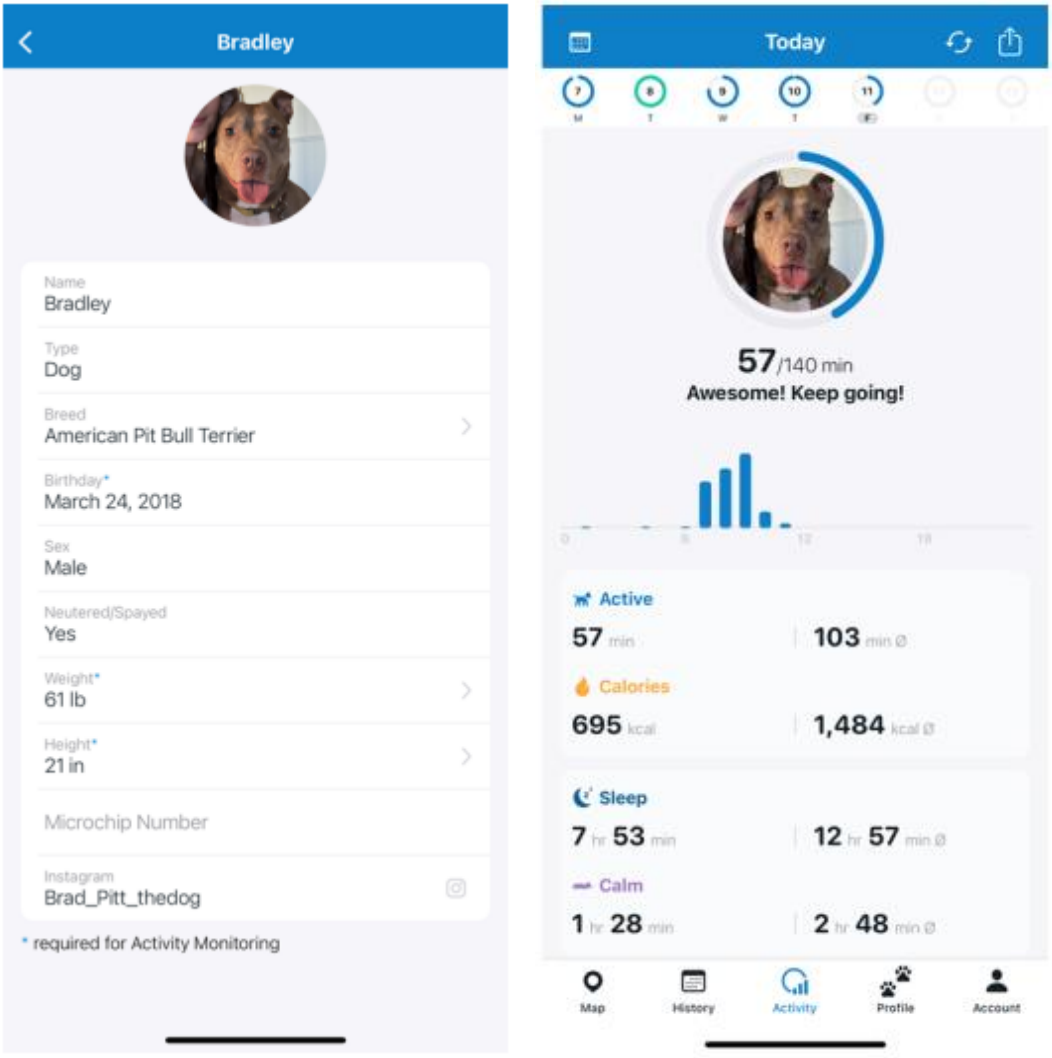


Figura 2.1.3 Monitoreo de Salud

Configurar una zona de ahorro

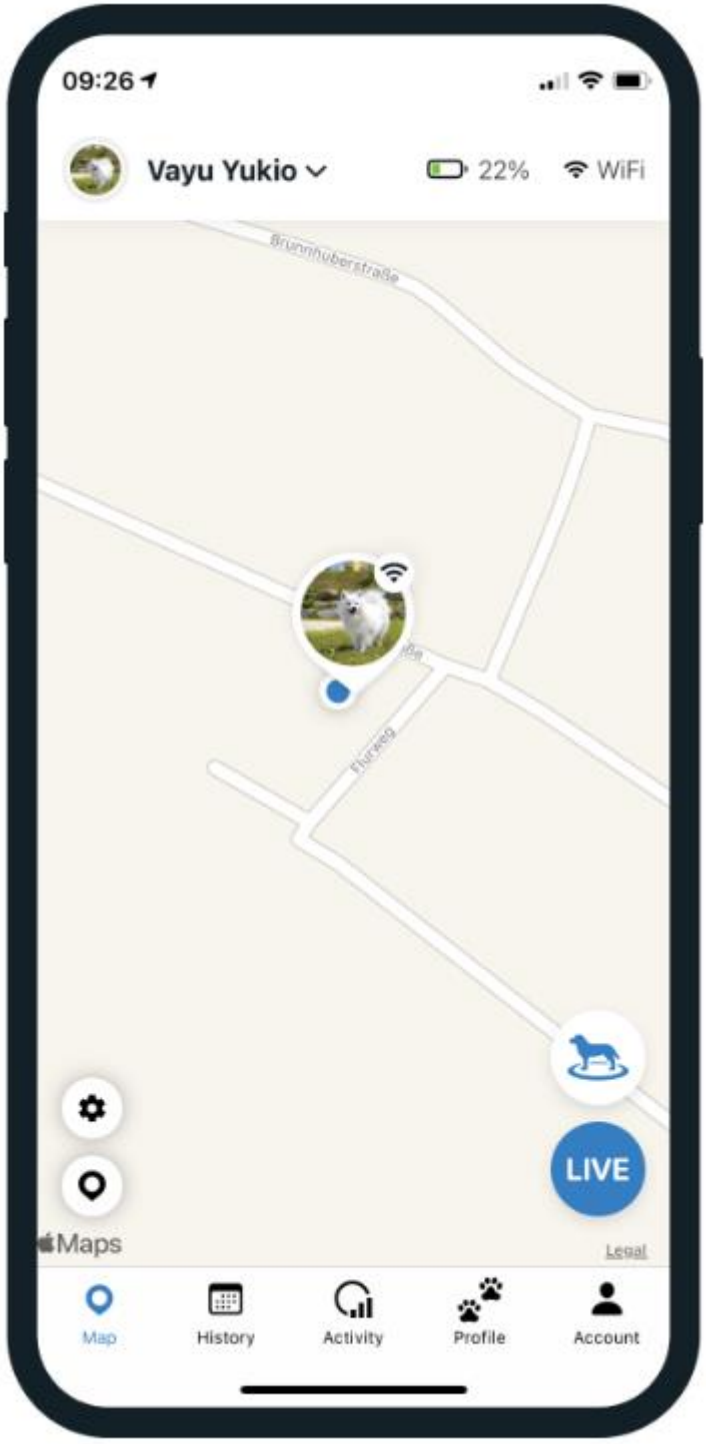


Figura 2.1.4 zona de ahorro

En la aplicación móvil de Tractive GPS, se puede configurar una zona de ahorro de energía de la siguiente manera: primero, escaneando las redes WiFi cercanas y etiquetándolas como zona de ahorro de energía. Si la ubicación no es lo suficientemente precisa, también es posible ajustarla manualmente.

Una vez que haya confirmado que la ubicación es correcta, puede asignar un nombre a esta nueva zona de ahorro de energía, como "WiFi del trabajo" o "WiFi de casa", entre otros. Asegúrese de que el nombre sea claro y conciso para facilitar su identificación.

Physical Tracking Device

Tractive Pet Physical Tracking Device es un excelente rastreador de mascotas que goza del favor y la confianza de los dueños de mascotas por su función de rastreo de alta precisión, servicio de posicionamiento de cobertura global, resistencia duradera y una gran variedad de funciones útiles. Estas ventajas hacen que destaque en el sector de los animales de compañía y ofrece a los propietarios una solución eficaz de seguimiento de mascotas.

Figura 2.1.5 Physical Tracking Device

- Función de posicionamiento GPS: El dispositivo de rastreo de mascotas Tractive está equipado con tecnología de posicionamiento GPS de alta precisión, que puede rastrear con precisión la ubicación de la mascota, lo que permite al propietario conocer el paradero de la mascota en cualquier momento.

- Seguimiento en tiempo real: el dispositivo se sincroniza con la APP del teléfono móvil para conseguir una sincronización de datos en tiempo real, de modo que el propietario puede comprobar la ubicación en tiempo real de la mascota a través del teléfono móvil en cualquier momento para garantizar la seguridad de la mascota.

- Monitorización de la actividad: El dispositivo de seguimiento de mascotas Tractive está equipado con la función de monitorización de la actividad, registrando el ejercicio de la mascota, el tiempo de descanso y las calorías quemadas y otra información, para ayudar a los propietarios a comprender el estado de salud de sus mascotas.
- Configuración de área segura: el dispositivo admite la creación de vallas virtuales, y los propietarios pueden configurar un área segura en la aplicación móvil, una vez

que la mascota salga del área segura, recibirá notificaciones instantáneas para ayudar a los propietarios a realizar un seguimiento de sus mascotas.

- Diseño impermeable: El dispositivo de rastreo de mascotas Tractive adopta un diseño impermeable, incluso en días lluviosos o ambientes húmedos, todavía puede funcionar normalmente, para garantizar la estabilidad y durabilidad del dispositivo.

- Ligero y cómodo: El dispositivo es compacto y ligero, por lo que es cómodo de llevar y no es una carga para su mascota, al tiempo que incorpora una variedad de métodos de fijación para garantizar que el dispositivo está fijado de forma segura y firme al cuello de su mascota.

- Compatible con múltiples plataformas: El dispositivo de seguimiento de mascotas Tractive es compatible con los sistemas iOS y Android, cubriendo la mayoría de los usuarios de teléfonos inteligentes, cómodo y práctico.

2.2 Whistle

Whistle es un avanzado software de seguimiento de mascotas diseñado para los amantes de los perros. Este software integra varias funciones y servicios para ayudar a los usuarios a tener una visión completa del comportamiento y la salud de su mascota.

En primer lugar, el software Whistle cuenta con seguimiento de localización en tiempo real. No importa dónde se encuentre su mascota, siempre que haya conexión a Internet, podrá saber exactamente dónde está utilizando la función de localización GPS del software. Se trata de una función muy útil, especialmente cuando tu mascota se pierde o se escapa de casa. Puedes establecer una zona segura y recibirás una notificación en cuanto tu mascota salga de ella.

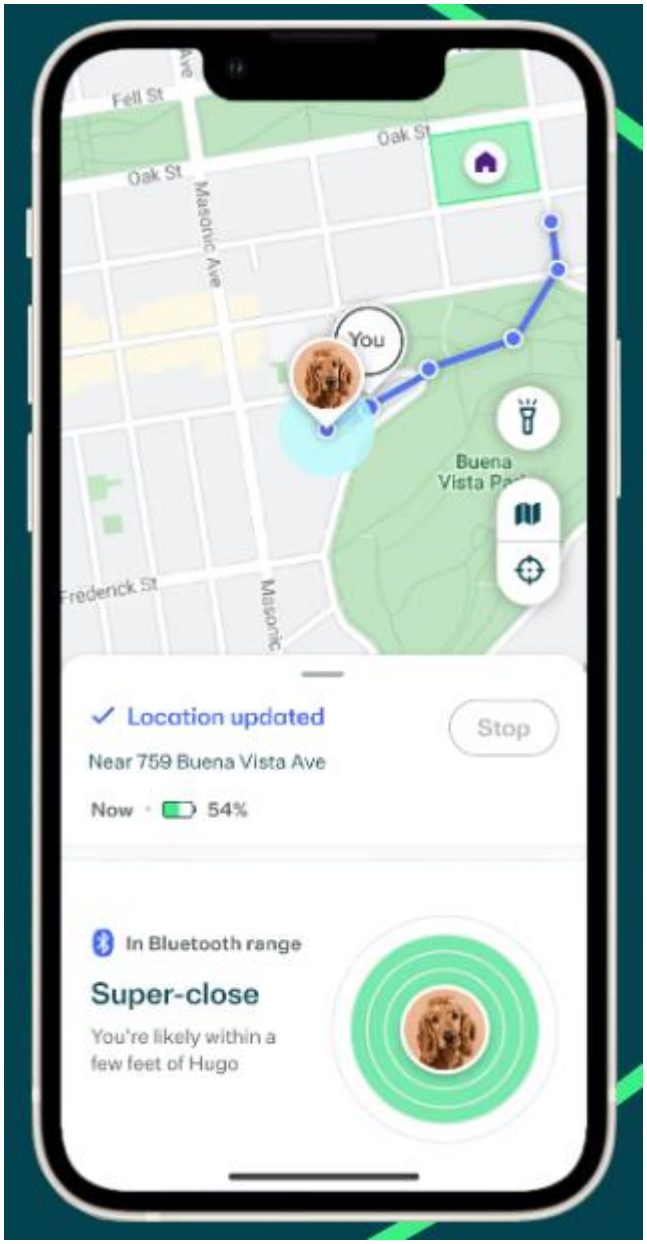


Figura 2.2.1 localización en tiempo Whistle

En segundo lugar, Whistle también tiene una función de control de la salud. El software es capaz de seguir y analizar las actividades diarias de su mascota, incluido cuánto ejercicio hace, cómo duerme e incluso sus hábitos alimenticios. Si el programa detecta algún problema de salud o patrón de comportamiento inusual, se lo notificará inmediatamente. Por ejemplo, puede utilizar el software para concertar citas con el veterinario, acceder a recursos de adiestramiento e incluso comunicarse con otros propietarios de mascotas.

Figura 2.2.2 El dispositivo Whistle

El dispositivo de seguimiento físico Whistle es un dispositivo de seguimiento y control de la salud de las mascotas que incorpora una serie de tecnologías de vanguardia con el fin de proporcionar una gama completa de servicios de seguimiento de la salud y la ubicación de las mascotas.

Una de las características más notables del dispositivo Whistle es el seguimiento GPS en tiempo real. El dispositivo utiliza el Sistema de Posicionamiento Global (GPS) para garantizar que siempre pueda localizar a su mascota en un mapa, esté donde esté. El dispositivo también le permite establecer "zonas seguras" y envía notificaciones en cuanto su mascota sale de estas zonas.

El dispositivo Whistle supervisa la actividad y el comportamiento diarios de su mascota, incluidos los niveles de ejercicio, los patrones de sueño y los hábitos alimentarios. Si el dispositivo detecta alguna anomalía en la salud o el comportamiento, envía una alerta inmediata.

Por último, el dispositivo Whistle tiene una función única: el seguimiento de las tendencias de salud. Con esta función, el dispositivo supervisa los datos de salud de su mascota a lo largo del tiempo y genera informes de tendencias para que los usuarios puedan ver si la salud de su mascota está mejorando o empeorando.

Tecnología avanzada en el software de seguimiento de mascotas Whistle

La extraordinaria funcionalidad del software Whistle es posible gracias a la gama de tecnologías avanzadas que lo respaldan. Las capacidades de localización del software se basan en el Sistema de Posicionamiento Global (GPS alimentado por la red 4G LTE-M de AT&T). A través de esta tecnología, Whistle es capaz de rastrear con precisión la ubicación de su mascota en todo el mundo.

En segundo lugar, la función de supervisión de la salud del software utiliza el aprendizaje automático y el análisis de datos. El software Whistle aprende y comprende los patrones de comportamiento de su mascota y, a continuación, utiliza el análisis de big data para predecir su estado de salud. Por ejemplo, si el software detecta un descenso repentino en el ejercicio de su mascota, puede predecir que su mascota puede estar enferma y notificárselo.

Por último, el diseño de la interfaz de usuario (UI) y la experiencia de usuario (UX) del software Whistle es también un factor clave de su éxito. El software presenta un diseño limpio y nítido que permite a los usuarios navegar fácilmente y utilizar diversas funciones. Además, el programa ofrece diversas opciones de personalización que permiten a los usuarios adaptarlo a sus necesidades y preferencias.

2.3 FI app

Fi pet tracking app es un potente collar inteligente para perros con muchas tecnologías y funciones avanzadas

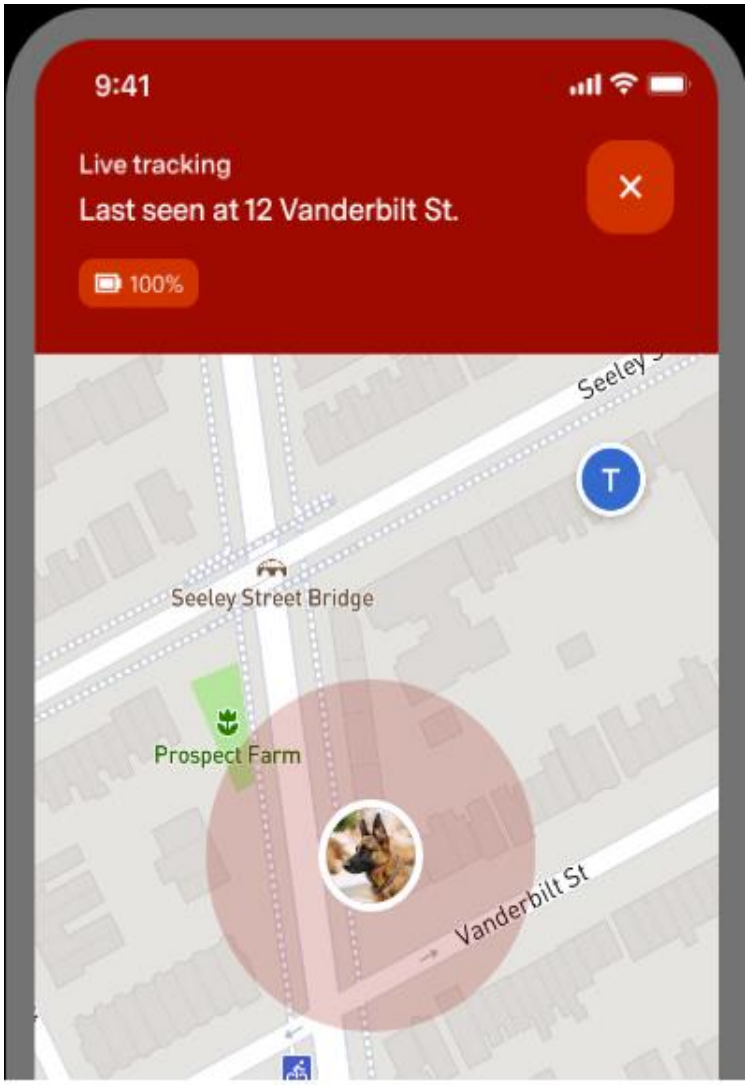


Figura 2.3.1 rastrear la ubicación

Función de posicionamiento GPS: la aplicación de seguimiento de mascotas Fi adopta tecnología de posicionamiento GPS de alta precisión, que puede rastrear la ubicación de su mascota en tiempo real, para que el propietario pueda saber en todo momento el paradero de la mascota.

Monitorización de la actividad: la aplicación monitoriza las actividades diarias de tu mascota, incluyendo el recuento de pasos y la monitorización del sueño, ayudando a los propietarios a entender la salud y el nivel de actividad de su mascota.

Configuración de zona segura: la aplicación de seguimiento de mascotas Fi permite configurar una zona segura en casa; una vez que la mascota abandona la zona segura establecida, el propietario recibirá alertas de fuga en tiempo real y tomará las medidas oportunas.

Figura2.3.2 Monitor dog's sleep

Modo Mascota Perdida: Fi Pet Tracker proporciona un "Modo Mascota Perdida" para ayudar a los propietarios a encontrar a sus mascotas perdidas más rápido cuando se pierden.

Batería de larga duración: Fi Pet Tracker está equipado con una batería de alto rendimiento que puede funcionar continuamente durante un período de tiempo más largo con una sola carga, proporcionando una vida útil más larga y reduciendo la molestia de la carga frecuente.

Algoritmo inteligente de ahorro de energía: la aplicación tiene incorporado un algoritmo inteligente de ahorro de energía que optimiza el consumo de energía y prolonga la duración de la batería del dispositivo, lo que permite a los propietarios vigilar a sus mascotas durante períodos más largos.

Funciones interactivas: la aplicación de seguimiento de mascotas de Fi permite que otros usuarios, como familiares, amigos o paseadores de perros, compartan la información de seguimiento de las mascotas, para que varias personas puedan preocuparse por la seguridad de las mascotas al mismo tiempo.

Características/ventajas del dispositivo de rastreo físico de mascotas Fi

FUNCIÓN DE LOCALIZACIÓN FIABLE: El dispositivo de seguimiento físico de mascotas Fi utiliza señales GPS y LTE-M para un posicionamiento preciso y fiable, lo que permite a los propietarios saber dónde están sus mascotas en todo momento, ayudando a evitar que las mascotas se pierdan.

Figura 2.3.3 dispositivo de físico Fi

BATERÍA DE LARGA DURACIÓN: El dispositivo Fi tiene una excelente duración de la batería que dura meses con una sola carga, proporcionando a los propietarios servicios de vigilancia de larga duración y reduciendo la molestia de las cargas frecuentes.

Recordatorio de zona segura: el dispositivo Fi admite la configuración de zonas seguras, una vez que la mascota sale de la zona designada, el propietario recibirá una notificación oportuna para proteger la seguridad de la mascota.

Aplicación fácil de usar: la aplicación de seguimiento de mascotas Fi tiene una interfaz fácil de usar y un funcionamiento sencillo, lo que permite a los propietarios realizar un seguimiento de la ubicación y las actividades de su mascota a través de la aplicación móvil.

Algoritmo inteligente de ahorro de energía: el dispositivo Fi ahorra energía de forma inteligente, optimiza el consumo de energía y prolonga la duración de la batería, lo que permite a los dueños cuidar de sus mascotas durante más tiempo.

Cobertura de posicionamiento global: la tecnología GPS y LTE-M del dispositivo Fi admite el posicionamiento en todo el mundo, lo que permite a los dueños seguir la ubicación de su mascota en tiempo real sin importar dónde se encuentre.

Tecnología avanzada

1. TECNOLOGÍA GPS Y LTE: La aplicación de seguimiento de mascotas Fi utiliza tecnología GPS y LTE avanzada para garantizar una localización precisa y en tiempo real de la mascota. Con la red LTE, la información de ubicación de la mascota se puede transmitir globalmente en tiempo real, y el propietario puede comprobar la ubicación de la mascota en cualquier momento a través de la aplicación móvil.

2. Compatibilidad con Wi-Fi: El collar inteligente para perros Fi Serie 2 añade compatibilidad con Wi-Fi para mejorar la precisión y la eficacia del posicionamiento. Cuando la mascota está en el área de cobertura de la red Wi-Fi, puede obtener la información de ubicación de la mascota más rápidamente y mejorar la estabilidad del posicionamiento.

3. Tecnología de monitorización de actividad: La tecnología de monitorización de actividad de la aplicación de seguimiento de mascotas Fi puede contar con precisión los pasos y el sueño de la mascota y, mediante el análisis de datos, puede ayudar al propietario a comprender el estado de salud de la mascota y encontrar anomalías a tiempo.

4. Alerta de fuga en tiempo real: La función de alerta de fuga en tiempo real de la aplicación Fi Pet Tracker puede enviar notificaciones al propietario inmediatamente cuando la mascota sale de la zona segura, ayudando al propietario a responder rápidamente para evitar la pérdida de la mascota.

5. Tecnología de batería de larga duración: el dispositivo de rastreo de mascotas Fi adopta una tecnología de batería avanzada y algoritmos inteligentes de ahorro de energía, lo que hace que la vida útil de la batería del dispositivo sea mucho más larga, reduciendo la necesidad de carga frecuente y proporcionando a los propietarios una experiencia de uso más prolongada.

6. Tecnología de Internet de las Cosas: la aplicación de seguimiento de mascotas Fi adopta la tecnología de Internet de las Cosas, que realiza la interconexión entre el dispositivo y la APP del teléfono móvil, de modo que el propietario puede captar la ubicación y las actividades de la mascota en tiempo real a través del teléfono móvil en cualquier momento.

La aplicación de seguimiento de mascotas Fi destaca en el mercado de los collares inteligentes para perros por su avanzada tecnología y sus versátiles funciones. Al proporcionar un seguimiento en tiempo real de la ubicación y la actividad de tu mascota, además de ofrecer una gran cantidad de funciones de seguridad, Fi ayuda a los propietarios a cuidar y proteger mejor a sus mascotas.

2.4 Pawtrack app

La aplicación Pawtrack Pet Cat Tracking es una aplicación avanzada de seguimiento para gatos que ofrece muchas funciones y tecnologías avanzadas diseñadas para ayudar a los propietarios de mascotas a vigilar y proteger a sus gatos en tiempo real. Las características y la tecnología avanzada de la aplicación Pawtrack Pet Cat Tracking se analizarán en detalle a continuación:

Figura 2.4.1 Pawtrack Pet Cat Tracking

Pawtrack app Características

Figura 2.4.2 Localización Pawtrack

1. Localización GPS: la app Pawtrack utiliza tecnología GPS y Glonass para rastrear con precisión la ubicación de su gato. Los datos de localización se actualizan cada 6 segundos para garantizar que el propietario conozca el paradero del gato en tiempo real.

2. Notificación de hogar Wifi: la aplicación está habilitada para notificaciones de hogar Wifi, de modo que cuando el gato regresa a una zona Wifi dentro de los confines del hogar, el propietario recibe una notificación que le permite saber que el gato ha regresado sano y salvo.

3. Aplicación de configuración del collar: La aplicación Pawtrack permite al propietario configurar y gestionar el collar a través de la aplicación, lo que facilita y flexibiliza la configuración de diversos parámetros y funciones.

4. Seguimiento de varios gatos: La aplicación admite el seguimiento de varios gatos, lo que permite a los propietarios rastrear la ubicación de varios gatos simultáneamente en Google Maps.

5. Alertas personalizadas: La aplicación Pawtrack permite a los propietarios configurar varias alertas, como alertas de hogar y alertas de batería baja, para asegurarse de que los propietarios pueden vigilar el estado de su gato.

6. Localización por baliza: La función de baliza de la aplicación ayuda a los propietarios a localizar a sus gatos con mayor precisión, ya que proporciona un indicador de señal que facilita a los propietarios encontrar rápidamente la ubicación de su gato.

7. Descarga de datos GPS y ajustes personalizados: la app Pawtrack admite la descarga de datos GPS, lo que permite al propietario conocer mejor las actividades del gato. Al mismo tiempo, se pueden realizar ajustes personalizados según las necesidades del propietario para satisfacer requisitos de seguimiento personalizados.

Pawtrack Dispositivo Físico de Seguimiento de Gatos Mascota

Pawtrack es un avanzado dispositivo de localización GPS diseñado para gatos con las siguientes potentes características

Figura 2.4.3 Pawtrack Dispositivo Físico

Tecnología de localización GPS

Pawtrack utiliza tecnología avanzada de localización GPS para rastrear con precisión la ubicación de su gato. Cada 6 segundos, Pawtrack registra la ubicación del gato y carga los datos en una cuenta en línea o en una aplicación móvil. Esta tecnología permite a los propietarios de gatos conocer el paradero de su mascota en tiempo real, evitando que se pierda o se desoriente. Además, Pawtrack está equipado con varias tecnologías GPS, como GPS, GLONASS, WIFI y 4G, para mejorar la precisión y estabilidad del posicionamiento. Además, la nueva antena GPS-4G flexible de Pawtrack mejora aún más el rendimiento del seguimiento mediante la observación del cielo, lo que le hace destacar entre sus competidores.

Notificación de Wi-Fi en casa

Pawtrack también cuenta con Home Wi-Fi Notification, que reconoce cuándo el gato está en casa a través de Wi-Fi y ahorra energía cuando está en modo de suspensión. Cuando el gato sale de casa, reanuda automáticamente su funcionamiento, garantizando el seguimiento en tiempo real del paradero de su mascota. Esta función proporciona a los dueños de gatos la comodidad añadida de estar mejor informados sobre las actividades diarias de su mascota.

Múltiples configuraciones de alerta

Pawtrack también ofrece múltiples configuraciones de alerta para que los dueños de gatos puedan estar informados al instante de los movimientos de su mascota. Cuando el gato sale de la zona de seguridad establecida, el sistema hace sonar una alarma para avisar al dueño de que la mascota puede estar perdida o fuera de custodia. Pawtrack también notifica a los propietarios cuando el dispositivo está en modo "gato perdido" y la mascota se recupera con éxito. Estas alertas contribuyen a mejorar la seguridad de las mascotas y dan mayor tranquilidad a los propietarios de gatos.

Descarga y almacenamiento de datos

Pawtrack no sólo rastrea la ubicación de la mascota en tiempo real, sino que también descarga y almacena datos históricos de rastreo. Los usuarios pueden observar el historial de rastreo de tres formas distintas, Reproducción, Trayectoria y Mapa de calor, para comprender la trayectoria de actividad y los hábitos de comportamiento de la mascota. Esta función no sólo facilita a los propietarios de gatos la comprensión de las actividades de sus mascotas, sino que también ayuda a compartir las características de comportamiento de la mascota con los veterinarios para proporcionar una atención más específica para la salud de la mascota.

Batería de larga duración y carga rápida

El dispositivo físico Pawtrack de seguimiento de mascotas y gatos tiene una batería de larga duración y puede utilizarse hasta seis días en sólo cuatro horas por carga. Además, cuenta con un módulo de alimentación de liberación rápida curvado que permite a los usuarios cambiar la batería en caliente en un estado de rastreo sin interrupciones para garantizar un rastreo ininterrumpido. Esta característica es muy cómoda para los propietarios de gatos y reduce las molestias de las cargas frecuentes.

Seguimiento de varios gatos y cobertura global

El dispositivo físico de rastreo de gatos Pawtrack es compatible con el rastreo multi-gato, lo que le permite rastrear la ubicación de varios gatos en la misma cuenta. Además, Pawtrack tiene cobertura global y se puede utilizar en todo el mundo, proporcionando a los propietarios de gatos escenarios de aplicación más amplios y una experiencia de uso más cómoda. No importa dónde estén los gatos, los propietarios pueden estar al tanto de su dinámica de localización, lo que aumenta la sensación de seguridad de la mascota y la tranquilidad del propietario.

En resumen, la avanzada tecnología del dispositivo físico de rastreo de gatos Pawtrack, que incluye tecnología de posicionamiento GPS, notificaciones Wi-Fi domésticas, múltiples configuraciones de alerta y funciones de descarga y almacenamiento de datos, proporciona a los propietarios de gatos una experiencia de rastreo de mascotas completa y práctica. Sus potentes funciones y su cobertura mundial proporcionan a los propietarios de gatos más opciones para garantizar la seguridad y la salud de sus gatos.

Tecnología avanzada

Tecnología GPS y Glonass avanzada: la aplicación de seguimiento de gatos Pawtrack utiliza tecnología GPS y Glonass avanzada para garantizar un posicionamiento preciso de la ubicación del gato, al tiempo que proporciona una transmisión de señal más rápida y estable.

Tecnología de notificación Wifi a domicilio: la tecnología de notificación Wifi a domicilio de la aplicación ayuda a los propietarios a saber si sus gatos han vuelto a casa sanos y salvos detectando si el gato ha entrado en la zona Wifi designada, lo que aumenta la seguridad de la vigilancia del gato.

Tecnología de rastreo de varios gatos: La aplicación Pawtrack permite rastrear la ubicación de varios gatos al mismo tiempo, utilizando eficientes algoritmos de

posicionamiento para garantizar que el paradero de varios gatos se muestre con precisión en el mapa.

Tecnología de localización Beacon: La función Beacon de la aplicación utiliza tecnología de localización avanzada para ayudar a los propietarios a encontrar la ubicación de sus gatos de forma más rápida y precisa, proporcionando una guía de localización fiable.

Tecnología de descarga de datos y ajustes personalizados: La aplicación Pawtrack ofrece una función de descarga de datos que permite a los propietarios conocer mejor las actividades de sus gatos, al tiempo que admite ajustes personalizados para satisfacer las necesidades individuales de los distintos propietarios.

En resumen, la aplicación de rastreo de gatos Pawtrack ofrece a los propietarios un potente servicio de vigilancia de mascotas con sus múltiples funciones prácticas y su avanzada tecnología, que permite a los propietarios conocer la ubicación y el paradero de sus gatos en cualquier momento y en cualquier lugar, garantizando la seguridad y la salud de sus mascotas. Estas funciones y tecnologías hacen de Pawtrack una aplicación muy apreciada y de confianza en el campo del rastreo de mascotas.

2.5 Findster app

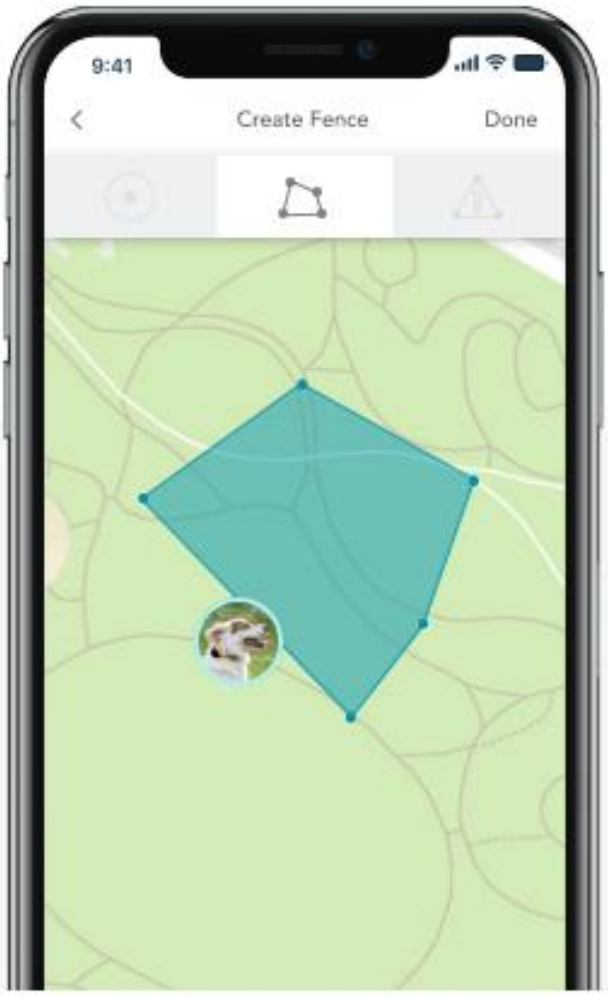
Findster es una aplicación de rastreo de mascotas diseñada para dueños de mascotas con características y tecnología avanzadas diseñadas para ayudar a los dueños a rastrear y monitorear a sus mascotas en tiempo real.

Figura 2.5.1 Findster app

Localización en tiempo real y seguimiento de la actividad: La aplicación Findster Pet Tracking proporciona localización en tiempo real y seguimiento de la actividad de las mascotas mediante la conexión a un módulo en el collar de la mascota. Los usuarios sólo tienen que abrir la aplicación y activar el módulo para seguir con precisión la ubicación de su mascota en un mapa y conocer su paradero.

Función de seguimiento y radar en exteriores: La función de seguimiento en exteriores de la aplicación Findster Pet Tracker es posible gracias a la tecnología GPS, que proporciona los resultados de seguimiento más precisos en entornos abiertos al aire libre. La función de radar, por otro lado, guía a los usuarios hasta la ubicación exacta de su mascota, y su precisión puede mejorarse actualizando el firmware de la aplicación y del módulo, así como evitando zonas densas, interiores o cerca de grandes objetos metálicos.

Tecnología MAZE: La aplicación de seguimiento de mascotas Findster utiliza la tecnología MAZE, que permite a los módulos comunicarse de forma inalámbrica entre sí hasta un alcance máximo de 4,8 kilómetros (3 millas). Esta tecnología permite a los propietarios rastrear la ubicación de su mascota en una amplia gama de entornos al aire libre sin tener que depender de la cobertura celular, eliminando así las limitaciones geográficas y garantizando que los propietarios siempre puedan realizar un seguimiento de sus mascotas.



Valla virtual: La aplicación de seguimiento de mascotas Findster admite la configuración de vallas virtuales, lo que permite a los propietarios establecer una zona segura en el mapa y, cuando la mascota salga de esta zona, la aplicación enviará inmediatamente una notificación para recordar al propietario el paradero de la mascota. Esta función es especialmente importante para garantizar la seguridad de las mascotas, de modo que los dueños puedan reaccionar a la primera para evitar que sus mascotas se pierdan.

Adecuado para múltiples mascotas: Findster Pet Tracker App soporta el seguimiento de múltiples mascotas, una cuenta de App se puede conectar a múltiples módulos, el propietario puede realizar un seguimiento de la ubicación y las actividades de múltiples mascotas al

Figura 2.5.2 Valla virtual

mismo tiempo, la realización de la gestión de mascotas de ventanilla única.

Modo de ahorro de energía: Con el fin de prolongar la vida de la batería del módulo, Findster Pet Tracking App también tiene un modo de ahorro de energía. Durante las actividades al aire libre, el módulo realizará un seguimiento GPS en tiempo real, pero al final de la actividad, el módulo cambiará a un modo de ahorro de energía que sólo realiza un seguimiento de la actividad sin posicionamiento GPS, ahorrando energía y preparándose para el siguiente uso.

Dispositivo Físico

El rastreador de mascotas Findster es un dispositivo que proporciona seguimiento GPS en tiempo real y monitorización de la actividad de la mascota sin la carga de las cuotas mensuales. El rastreador puede montarse fácilmente en el collar de la mascota, lo que permite a los propietarios seguir la ubicación y la actividad de su mascota a través de su teléfono móvil. A continuación se ofrece un análisis detallado de las características y la tecnología del rastreador Findster para mascotas

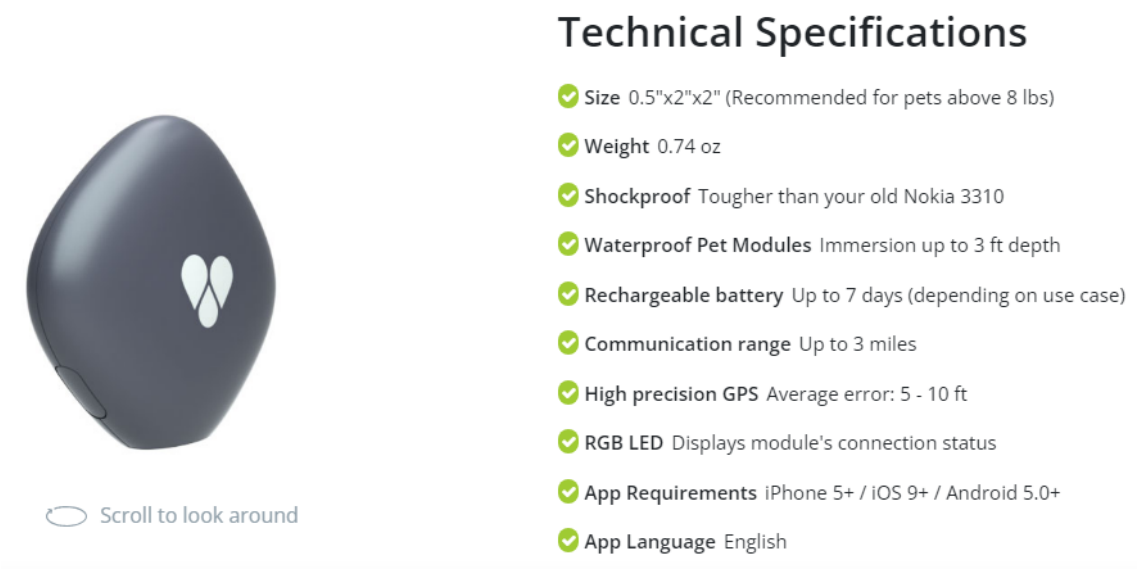


Figura 2.5.3 Dispositivo Físico

- TECNOLOGÍA MAZE: El rastreador de mascotas Findster utiliza la tecnología MAZE (tecnología de comunicación inalámbrica), que permite que el módulo para mascotas y el módulo guardián se comuniquen de forma inalámbrica dentro de un rango de aproximadamente 3 millas. A través de la tecnología MAZE, el módulo para mascotas rastrea la ubicación de la mascota a través del posicionamiento GPS y transmite los datos al módulo guardián, que a su vez se conecta a la aplicación del teléfono móvil a través de Bluetooth para lograr el seguimiento en tiempo real y la visualización de la ubicación de la mascota.
- Seguimiento GPS en tiempo real: el rastreador de mascotas Findster es ideal para proporcionar a los dueños de mascotas un seguimiento GPS en tiempo real de sus mascotas, que realiza el seguimiento de la ubicación de las mascotas en tiempo real a través de la tecnología GPS. Los dueños sólo tienen que llevar el módulo guardián y permanecer cerca del módulo para mascotas para ver la ubicación exacta de su mascota en la aplicación móvil sin tener que pagar una cuota mensual.
- Función de valla virtual: El rastreador de mascotas Findster también proporciona una función de valla virtual, en la que el propietario de la mascota puede establecer una zona segura en la aplicación móvil, y la aplicación enviará inmediatamente una notificación para recordar al propietario cuando la mascota salga de la zona. La

función de valla virtual puede garantizar la seguridad de las mascotas y evitar que se pierdan.

- Admite el seguimiento de varias mascotas: Findster Pet Tracker admite la conexión de un teléfono móvil a varios módulos para mascotas, lo que permite al propietario realizar un seguimiento de la ubicación y las actividades de varias mascotas al mismo tiempo. Esta función proporciona una cómoda solución de gestión para hogares con varias mascotas sin necesidad de adquirir varios dispositivos móviles adicionales.
- Sin cuota mensual: Findster Pet Tracker renuncia a la cuota mensual y no requiere que los usuarios paguen ninguna cuota de servicio adicional. Los dueños de mascotas pueden disfrutar de seguimiento GPS en tiempo real gratuito y servicio de monitoreo de actividad de mascotas después de comprar el rastreador de mascotas una vez, ahorrando el costo de uso a largo plazo.
- Diseño resistente al agua: El módulo para mascotas del rastreador Findster tiene un diseño resistente al agua que garantiza que el dispositivo funcionará correctamente y no se verá afectado por el agua incluso en entornos lluviosos o húmedos. Esta característica proporciona una capa adicional de seguridad para que los propietarios puedan sentirse seguros al dejar que sus mascotas estén al aire libre.
- Limitaciones del rastreo remoto: cabe señalar que el alcance del rastreador de mascotas Findster se ve afectado por el entorno. Al no depender de la cobertura de la red móvil, es más adecuado para el rastreo en exteriores, mientras que el rastreo a distancia puede no ser posible en interiores y zonas alejadas. Por lo tanto, los propietarios pueden considerar otro tipo de dispositivos o soluciones si necesitan rastrear a su mascota en tiempo real en interiores o en zonas lejanas.
- Con su avanzada tecnología MAZE, seguimiento GPS en tiempo real y funciones de valla virtual, el rastreador de mascotas Findster proporciona a los propietarios una solución de seguimiento de mascotas cómoda, segura y asequible. Sin embargo, a la hora de elegir un rastreador de mascotas, los propietarios deben tener en cuenta sus propias necesidades y escenarios de uso para elegir el dispositivo adecuado que garantice la seguridad y la salud de sus mascotas.

3. Herramientas utilizada

En este capítulo, vamos a echar un vistazo en profundidad a las diversas herramientas y técnicas utilizadas en la construcción de nuestra aplicación de seguimiento de mascotas. Comenzaremos explorando dos populares patrones de diseño de software: el MVC (Modelo-Vista-Controlador) y el MTV (Modelo-Templa-Vista). Cada uno de estos patrones proporciona una separación clara entre los datos, la interfaz de usuario y las partes lógicas que procesan los datos. Contrastaremos estos dos patrones y discutiremos en detalle cómo implementamos y aprovechamos esta separación en este proyecto.

A continuación, hablaremos de la separación entre el front y el backend. Se trata de una decisión importante que afecta a la arquitectura y al flujo de trabajo de todo el proyecto. Discutiremos los pros y los contras de separar el front-end y el back-end, y analizaremos las circunstancias en las que decidimos separar y aquellas en las que decidimos no hacerlo.

Para elegir el marco web adecuado para nuestro proyecto, compararemos dos marcos web populares de Python, Flask y Django. Flask es un marco ligero que ofrece mucha flexibilidad, mientras que Django es un marco web completo con muchas funcionalidades incorporadas, como un backend de administrador, un sistema de autenticación, etcétera. Exploraremos en detalle los pros y los contras de ambos frameworks y en qué caso cuál elegir.

Por último, hablaremos del desarrollo front-end. Elegir el marco front-end adecuado es fundamental a la hora de construir una interfaz de usuario. Compararemos tres populares frameworks front-end, React, Vue.js y Angular, y exploraremos sus pros y sus contras, así como qué framework elegir en cada situación.

El objetivo de este capítulo es proporcionar al lector una visión completa de las distintas herramientas y técnicas a tener en cuenta a la hora de crear una aplicación web compleja. Explicaremos en detalle qué hace cada herramienta y técnica y discutiremos por qué las elegimos. Esperamos que este capítulo ayude a los lectores a entender la complejidad de construir aplicaciones web y les proporcione algunas referencias para tomar decisiones informadas en sus propios proyectos.

3.1 MVC y MTV

MVC (Modelo-Vista-Controlador) y MTV (Modelo-Template-Vista) son dos patrones de diseño de software populares que separan las responsabilidades de los datos, la interfaz de usuario y la lógica de procesamiento de datos. Al construir una aplicación de seguimiento de mascotas, has elegido Django, que sigue el patrón MTV. Esto significa que nuestro Modelo (Model) define la estructura de los datos, la Plantilla (Template) se encarga de cómo se muestran los datos, y la Vista (View) se encarga de procesar las solicitudes del usuario y devolver respuestas. Este patrón de diseño ayuda a mantener el código claro y facilita su mantenimiento.

3.1.1 Arquitectura MVC

La arquitectura MVC (Modelo-Vista-Controlador) es un patrón de arquitectura de software diseñado para dividir una aplicación en tres componentes principales con el fin de mejorar el mantenimiento y la escalabilidad del código.

MVC apareció por primera vez en el lenguaje de programación Smalltalk en los años 70 para construir interfaces gráficas de usuario (GUI). Con el auge de las aplicaciones web, el patrón MVC se introdujo en el desarrollo web, como SpringMVC. su idea central es dividir la aplicación en las tres partes siguientes:

Modelo

Se encarga de manejar los datos y la lógica de negocio de la aplicación. Interactúa con la base de datos y es responsable de añadir, borrar, modificar y comprobar los datos.

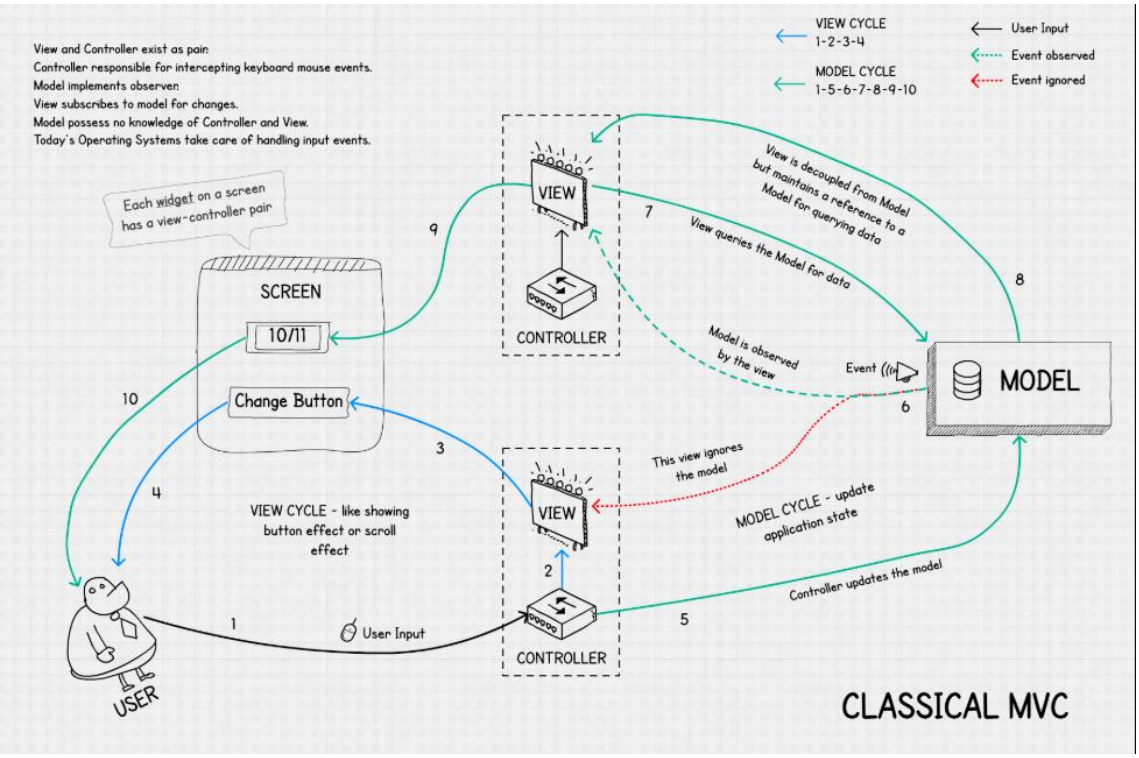
Vista

Responsable de la visualización de la interfaz de usuario y de la interacción con el usuario. Muestra datos al usuario y recibe entradas de éste.

Controlador (Controller)

responsable de procesar la entrada del usuario, de acuerdo con la entrada para llamar a la lógica del Modelo correspondiente, y decidir qué Vista actualizar. como el núcleo de MVC, el Modelo y la Vista desacoplados, de modo que puedan cambiar de forma independiente.

"La siguiente ilustración muestra un ejemplo de flujo de datos en el patrón MVC. En este caso concreto, hemos diseñado un contador sencillo con botones de incremento y decremento, y el estado del contador es mantenido por el modelo. Para mayor claridad, hemos simplificado estos dos botones en un único botón."



Harshal Patil - Contemporary Front-end Architectures

El patrón MVC evolucionó en diversas variantes como MVP y MVVM para adaptarse a diferentes necesidades de desarrollo. Los frameworks de front-end como React y Vue.js adoptaron la idea MVC, pero también introdujeron nuevos conceptos como el DOM virtual y la gestión de estados.

El patrón de arquitectura MVC sigue ocupando un lugar importante en el desarrollo de software, pero a medida que la tecnología sigue evolucionando, los desarrolladores necesitan tomar decisiones flexibles basadas en necesidades y tendencias específicas. Nuevos conceptos como la separación entre front-end y back-end, la gestión de estados y el DOM virtual seguirán influyendo en la evolución de la arquitectura de software.

Es probable que en el futuro el desarrollo de software incluya más arquitecturas de separación de front-end y back-end con patrones de diseño similares a MVC. Nuevos conceptos como la separación entre front-end y back-end, la gestión de estados y el DOM virtual seguirán influyendo en la evolución de la arquitectura de software.En conclusión.

3.1.2 Arquitectura MTV (Modelo-Plantilla-Vista)

La arquitectura MTV (Model-Template-View) es un patrón de diseño utilizado para construir aplicaciones web, que es una variante del patrón MVC (Model-View-Controller) y se utiliza normalmente para manejar la interfaz de usuario (UI) y el desarrollo front-end.

La arquitectura MTV fue introducida por primera vez por Django Web Framework para simplificar el desarrollo de aplicaciones web. Divide la aplicación en los tres componentes siguientes:

Modelo

Similar al modelo en MVC, se encarga de manejar los datos y la lógica de negocio de la aplicación.

```
# Create your models here.
class Pet(models.Model):
    id = models.AutoField("id", primary_key=True)
    name = models.CharField("name", max_length=255)
    age = models.IntegerField("age")
    breed = models.CharField("breed", max_length=255)
    user_id = models.IntegerField("user id")
    image = models.CharField("image", max_length=255)
    create_time = models.DateTimeField("create time", auto_now=True)
```

Con esta clase Pet model, podemos crear objetos mascota en nuestra aplicación Django y almacenar información sobre ellos. Por ejemplo, podemos guardar el nombre de la mascota, edad, raza, usuario al que pertenece, foto y otros datos en la base de datos a través de este modelo. Al mismo tiempo, como Django genera automáticamente los campos id y create_time, también podemos gestionar fácilmente los objetos mascota con identificadores únicos y marcas de tiempo

Plantilla (template)

Similar a la Vista en MVC , se encarga de definir la forma de visualización de la interfaz de usuario .

Vista

Similar al Controlador en MVC, es responsable de manejar las peticiones del usuario, obtener los datos del modelo y presentarlos en la plantilla.

```
class LocationViewSet(CustomModelViewSet):
    queryset = Location.objects.all()
    serializer_class = LocationSerializer
    permission_classes = []

    @action(methods=["GET"], detail=False,
permission_classes=[IsAuthenticated])
    def get_last(self, request):
        location = Location.objects.last()
        location = LocationSerializer(location)
        return DetailResponse(data=location.data)

    @action(methods=["POST"], detail=False,
permission_classes=[AllowAny])
    def add(self, request):
        return self.create(request)
```

Este código es un ViewSet basado en el framework REST de Django para gestionar peticiones HTTP relacionadas con el modelo Location

Este conjunto de vistas LocationViewSet proporciona las siguientes características:

- Permite a los usuarios autenticados obtener el último objeto Location.
- Permitir a cualquier usuario añadir un nuevo objeto Location al conjunto de objetos Location.

En Resumen

MVC (Model-View-Controller) y MTV (Model-Template-View) son dos patrones de diseño de software comunes utilizados para la interfaz de usuario y el desarrollo web. Tienen algunos puntos en común y diferencias significativas en su diseño.

Entre los puntos en común se encuentra el uso de una arquitectura en capas, que ayuda a separar las distintas preocupaciones y mejora el mantenimiento y la extensibilidad del código. En ambos patrones, existe el concepto de Modelo, que es responsable de la lógica que maneja los datos, incluyendo su adquisición,

almacenamiento y manipulación. Además, ambos implican una Vista, que es responsable de visualizar los datos al usuario.

Sin embargo, existen diferencias significativas. En MVC, el Controlador asume el papel de manejar la entrada del usuario y la lógica de control de la aplicación, lo que normalmente requiere que el desarrollador escriba manualmente el código del controlador. Por el contrario, en MVC, la Plantilla maneja el renderizado y la representación del HTML, mientras que el papel del Controlador es manejado automáticamente por un framework, como Django, lo que hace que MVC tenga un acoplamiento menor porque sus plantillas y frameworks manejan la lógica de control automáticamente, mientras que MVC suele tener un acoplamiento mayor.

Además, MVC suele ser adecuado para aplicaciones más grandes, ya que proporciona más control y organización, pero puede ser más complejo. En cambio, MVC es adecuado para una amplia gama de tamaños de aplicaciones, tanto pequeñas como grandes, porque proporciona un control automatizado y es más fácil de usar.

En resumen, ambos patrones pretenden desacoplar las distintas partes de una aplicación, lo que facilita el mantenimiento y la ampliación del código. MVC es un patrón de diseño más antiguo que se introdujo y resulta útil para aplicaciones grandes y lógica de control compleja, pero requiere una codificación más manual. Por el contrario, Django utiliza la arquitectura MTV para el diseño de aplicaciones.MTV es más adecuado para el desarrollo rápido, especialmente para aplicaciones web, y proporciona controles automatizados que hacen que el desarrollo sea más eficiente.

3.2 Front-end and back-end separation/no separacion

En esta subsección se discutirán las dos formas diferentes de estructurar un proyecto, separando el front-end del back-end y sin separar el front-end del back-end, así como sus respectivas características, estructura del proyecto, diferencias y ventajas.

En un proyecto de front-end y back-end no separados, el back-end es responsable de gestionar la lógica de negocio, el almacenamiento de datos y la representación de la página HTML, mientras que el front-end se encarga principalmente de recibir los datos proporcionados por el back-end y mostrarlos en la página HTML. Este enfoque es más rápido de desarrollar, resulta adecuado para aplicaciones sencillas y favorece la optimización para motores de búsqueda (SEO), pero puede parecer poco flexible en aplicaciones complejas.

En cambio, separar los proyectos de front-end y back-end permite a los equipos de front-end y back-end desarrollar en paralelo, comunicándose a través de API. El back-end proporciona la interfaz de datos y el front-end se encarga de la interfaz de usuario y las interacciones con el usuario, lo que permite utilizar diferentes pilas

tecnológicas y marcos de trabajo, proporcionando una mejor experiencia de usuario con una mejor mantenibilidad y flexibilidad.

Además, en esta sección se mencionan las tendencias modernas de arquitectura de aplicaciones, como la arquitectura de microservicios y la arquitectura nativa en la nube, que se adaptan a los requisitos multiplataforma, mejoran la eficiencia del desarrollo, optimizan la experiencia del usuario y logran el desacoplamiento, entre otras ventajas, lo que las convierte en la opción arquitectónica dominante. Asimismo, esta destaca las tendencias modernas de arquitectura de aplicaciones para adaptarse a la evolución de la tecnología y los requisitos de las aplicaciones.

3.2.1 Proyecto no separado de front-end y back-end

Funcionalidad y estructura del proyecto

En un proyecto no separado de front-end y back-end, el back-end es responsable de manejar la lógica de negocio, el almacenamiento de datos y la renderización de páginas HTML. El front-end se encarga principalmente de recibir los datos del back-end y mostrarlos en la página HTML. Los proyectos de front-end y back-end no separados suelen utilizar un motor de plantillas para renderizar páginas HTML e incrustar datos dinámicos en las plantillas HTML.

Diferencias y ventajas

Los proyectos front-end y back-end tradicionales no separados se desarrollan con relativa rapidez porque el desarrollo front-end y back-end están estrechamente integrados y los desarrolladores pueden compartir los mismos lenguajes de programación y marcos de trabajo.

Es compatible con la optimización para motores de búsqueda (SEO) y puede ser indexado más fácilmente por los motores de búsqueda, ya que utiliza un motor de plantillas para la representación de las páginas.

En el caso de aplicaciones sencillas, puede resultar más fácil implementarlas sin separar el front-end del back-end, ya que no existe la complejidad adicional de interactuar con las interfaces del front-end y del back-end.

3.2.2 Proyecto de separación de front-end y back-end

Funcionalidad y estructura del proyecto

En un proyecto de separación de front-end y back-end, el front-end y el back-end se desarrollan de forma independiente y se comunican a través de API. El backend es responsable de proporcionar la interfaz de datos, normalmente utilizando una API RESTful o GraphQL para proporcionar los datos. El front-end es responsable de manejar la interfaz de usuario y las interacciones con el usuario y de llamar a los datos proporcionados por el back-end a través de la API.

Diferencias y ventajas

Los proyectos front-end y back-end separados permiten a los equipos front-end y back-end trabajar en el desarrollo al mismo tiempo, lo que aumenta la eficiencia del desarrollo.

Dado que el front-end y el back-end se desarrollan de forma independiente, se pueden utilizar diferentes pilas tecnológicas y marcos de trabajo para aprovechar mejor sus respectivos puntos fuertes.

Se puede desacoplar el front-end y el back-end, lo que permite al back-end reutilizar servicios API para proporcionar datos a múltiples aplicaciones front-end, y al front-end refactorizar y optimizar de forma independiente.

Separar los proyectos front-end y back-end suele proporcionar una mejor experiencia de usuario, ya que el front-end puede utilizar marcos y tecnologías JavaScript modernos para crear interfaces de usuario más rápidas y fluidas.

En resumen

En la actualidad, existen principalmente las siguientes arquitecturas principales de proyectos:

Arquitectura de separación de front-end y back-end: como se ha mencionado anteriormente, la arquitectura de separación de front-end y back-end se comunica a través de API, y el front-end y el back-end son relativamente independientes, con mejor escalabilidad y eficiencia de desarrollo.

Arquitectura de microservicios: la arquitectura de microservicios divide una aplicación grande en varios servicios pequeños, cada uno de los cuales se ejecuta de forma independiente y se comunica a través de API. Esta arquitectura proporciona un mejor desacoplamiento y escalabilidad y es adecuada para sistemas distribuidos grandes y altamente concurrentes.

Arquitectura nativa de la nube: La arquitectura nativa de la nube es una forma de crear y desplegar aplicaciones en un entorno de nube que hace hincapié en el uso de tecnologías en contenedores (como Docker), microservicios y operaciones

automatizadas. Esta arquitectura es adecuada para aplicaciones muy elásticas y de alta disponibilidad.

Razones por las que la separación entre front-end y back-end, la arquitectura de microservicios y la arquitectura nativa en la nube se han generalizado

- Adaptarse a las necesidades de múltiples plataformas: con el desarrollo de móviles y Web, existe la necesidad de adaptarse a la aplicación de diferentes plataformas, y la separación de front-end y back-end y la arquitectura de microservicios pueden satisfacer mejor estas necesidades.

- Mejorar la eficiencia del desarrollo: la separación de front-end y back-end y la arquitectura de microservicios pueden desarrollarse al mismo tiempo, lo que mejora la eficiencia del desarrollo.

- Optimizar la experiencia de usuario: la separación de front-end y back-end y la arquitectura de microservicios pueden utilizar tecnologías front-end modernas para ofrecer una mejor experiencia de usuario.

- Desacoplamiento: la separación de front-end y back-end y la arquitectura de microservicios realizan el desacoplamiento entre front-end y back-end, y los servicios, de modo que cada parte puede desarrollarse y desplegarse de forma independiente, lo que mejora la flexibilidad y la capacidad de mantenimiento del sistema.

3.3 Flask o Django

Ahora vamos a discutir dos diferentes frameworks web de Python, Flask y Django, sus respectivas características, escenarios y ventajas.

En primer lugar, introducimos Flask, que es un micro-framework ligero.Flask proporciona funcionalidad básica de framework web, pero no tiene la potencia del tipo de funcionalidad incorporada de Django. Debido a su flexibilidad y simplicidad, Flask se utiliza normalmente para proyectos pequeños y medianos, prototipos y servicios API simples. Los desarrolladores tienen más libertad para elegir y combinar bibliotecas y herramientas para personalizar y ampliar sus proyectos.Flask es adecuado para proyectos que tienen necesidades específicas, pero no requieren un conjunto completo de características. Enumero las principales ventajas de Flask, como la flexibilidad, la ligereza y la posibilidad de elegir libremente las tecnologías de bases de datos.

A continuación, una introducción a Django, que es un framework web completo. Hay muchas características incorporadas, incluyendo ORM de base de datos, autenticación de usuarios, gestión de back-end, etc. Django es adecuado para grandes proyectos de desarrollo, aplicaciones empresariales y escenarios donde las aplicaciones web ricas en características deben ser construidas rápidamente.

Por último, se comparan las principales diferencias entre Flask y Django. Entre ellas se incluyen la complejidad funcional, ya que Django ofrece más funcionalidades integradas mientras que Flask es más conciso; la flexibilidad, ya que Flask es más flexible mientras que Django ofrece soluciones empaquetadas; y el tamaño del proyecto, ya que Flask es adecuado para proyectos pequeños y medianos mientras que Django es adecuado para proyectos grandes y complejos.

3.3.1 Flask (microframework ligero)

Flask es un framework web ligero que proporciona funcionalidades básicas sin las potentes características integradas de Django.

Debido a su flexibilidad y sencillez, Flask se utiliza habitualmente para proyectos pequeños o medianos, creación de prototipos y servicios API sencillos.

Flask permite a los desarrolladores utilizar sus bibliotecas y herramientas favoritas para personalizar y ampliar el proyecto con mayor libertad.

Es adecuado para proyectos con necesidades específicas que no requieren un conjunto completo de características.

Ventajas clave

Flexible : Flask proporciona pocas restricciones , permitiendo a los desarrolladores elegir y combinar herramientas libremente .

Ligero : Debido al diseño aerodinámico, las aplicaciones Flask tienen una pequeña huella de memoria y son adecuadas para entornos con recursos limitados.

Libre : Flask no obliga al uso de bases de datos y ORM específicos , los desarrolladores pueden elegir la tecnología de base de datos adecuada según la demanda .

3.3.2 Django (framework Web completo)

Django es un framework Web completo, tiene un gran número de características incorporadas, incluyendo bases de datos ORM, autenticación de usuarios, gestión de fondo y así sucesivamente.

Adecuado para el desarrollo de proyectos a gran escala , aplicaciones empresariales , y la necesidad de construir rápidamente aplicaciones Web ricas en características .

Las potentes características de Django y su rico ecosistema de plug-ins , permiten a los desarrolladores construir eficientemente aplicaciones Web complejas .

Principales ventajas

- Con todas las funciones : Django incorporado un gran número de características de uso común , los desarrolladores no necesitan construir repetidamente la rueda , para construir rápidamente aplicaciones Web complejas .
- Eficiencia en el desarrollo: Django proporciona potentes herramientas de generación automática de código y gestión de fondo, reduciendo la carga de trabajo del desarrollador.
- Comunidad y ecología : Django tiene una gran comunidad activa y un rico ecosistema de plug-in , por lo que es fácil para los desarrolladores a encontrar soluciones .

Diferencias clave entre ambos

- Complejidad funcional: Django es un framework completo con muchas características incorporadas, mientras que Flask es más conciso y proporciona funcionalidad básica.

- Flexibilidad : Flask es más flexible , permitiendo a los desarrolladores elegir libremente los componentes , mientras que Django es más restringido , proporcionando una solución de paquete .

- Tamaño del proyecto: Flask es adecuado para proyectos pequeños y medianos , mientras que Django es adecuado para proyectos grandes y complejos.

Característica/Función	Flask	Django
Tipo	Micro-framework	Marco de trabajo completo
Año de Lanzamiento	2011	2005
Lenguaje	Python	Python
Tamaño del Ecosistema	Pequeño	Grande
Riqueza de Funciones	Flexible y extensible	Características incorporadas abundantes
Soporte para API	Soportado	No soportado
Soporte para páginas HTML dinámicas	No soportado	Soportado
Escenarios adecuados	Desarrollo rápido, proyectos pequeños	Proyectos grandes, aplicaciones complejas
Soporte para base de datos	Basado en extensiones	Soporte ORM incorporado
Curva de aprendizaje	Moderada	Moderada
Soporte de la comunidad	Pequeño	Grande

Tabla3.3.2

3.4 Marco de front-end convencional

En el mundo del desarrollo front-end, existen muchos frameworks y librerías diferentes a disposición de los desarrolladores para ayudarles a crear aplicaciones web modernas y de alto rendimiento. Estos frameworks y librerías proporcionan una gran cantidad de características y herramientas que permiten a los desarrolladores gestionar la parte front-end de sus aplicaciones con mayor facilidad. En este artículo, nos centraremos en tres frameworks de front-end comunes, React, Vue.js y Angular, y profundizaremos en sus características, ventajas y escenarios que se les aplican para ayudar a los desarrolladores a comprender mejor y elegir el framework de front-end adecuado para su proyecto.

3.4.1 React

React es un popular framework de JavaScript desarrollado y mantenido por Facebook. Permite construir componentes de interfaz de usuario reutilizables, haciendo que el desarrollo front-end sea más eficiente y flexible. Estas son las características, ventajas y desventajas del framework front-end React

Características:

- Desarrollo por componentes : React divide la página en componentes, cada uno de los cuales es responsable de su propia lógica de negocio y presentación de la vista. Este desarrollo basado en componentes hace que el código sea modular y fácil de reutilizar y mantener.
- DOM virtual: React utiliza la tecnología DOM virtual para rastrear y comparar los cambios en el DOM real manteniendo un árbol DOM virtual en memoria. De este modo, React puede actualizar eficientemente la vista cuando cambian los datos, mejorando el rendimiento.
- Flujo de datos unidireccional: React utiliza un flujo de datos unidireccional para la vinculación de datos, donde el flujo de datos es unidireccional, pasando del componente padre al componente hijo. Esto asegura datos controlables y predecibles y evita el problema de datos desordenados y difíciles de rastrear.
- Sintaxis JSX: React introduce la sintaxis JSX, que permite escribir estructuras similares a HTML en código JavaScript. Esto hace que el código sea más intuitivo y legible, y también facilita la escritura y renderización de componentes.
- Ecosistema rico: React tiene un gran ecosistema con un gran número de bibliotecas de terceros y plug-ins para elegir, como React Router para el manejo de enrutamiento, Redux para la gestión de estados, etc., lo que puede mejorar en gran medida la eficiencia del desarrollo.

Ventajas

- Alto rendimiento: a través del DOM virtual y los algoritmos de comparación de diferencias, React puede lograr actualizaciones eficientes del DOM y mejorar el rendimiento de la aplicación.
- Desarrollo por componentes: el desarrollo por componentes hace que el código sea modular, fácil de reutilizar y mantener, y ayuda a la colaboración en equipo.
- Comunidad activa: React tiene una gran comunidad y soporte, los desarrolladores pueden encontrar fácilmente soluciones y recursos.
- Ecosistema rico: el ecosistema React es rico y extensible con muchos plugins y herramientas para elegir.

Desventajas:

- Curva de aprendizaje: para los principiantes, empezar con React puede tomar algún tiempo para aprender y acostumbrarse.
- Sintaxis JSX: la sintaxis JSX puede no ser aceptada por todos los desarrolladores, lo que a veces lleva a un código confuso o difícil de mantener.
- Enfoque sólo en la capa de vista: React sólo se centra en la capa de vista, para aplicaciones de pila completa se necesitan herramientas y bibliotecas adicionales para manejar otros aspectos de la funcionalidad.

3.4.2 Vue.js

Vue.js es un popular framework front-end progresivo de JavaScript que está ganando terreno gradualmente en el desarrollo de aplicaciones web y móviles. Ofrece las ventajas de la simplicidad, la flexibilidad, la arquitectura por componentes y una integración perfecta. A pesar del dominio de React y Angular, Vue.js está creciendo rápidamente y se está convirtiendo en una alternativa muy popular, como lo demuestran las tendencias y estadísticas del mercado.

Características de Vue.js:

- Framework incremental : Vue.js es un framework incremental que permite a los desarrolladores introducir sus características gradualmente según las necesidades del proyecto sin cargarlas todas a la vez. Esto hace que Vue.js sea fácil de integrar en proyectos existentes y también aligera la carga inicial.
- DOM virtual: Vue.js utiliza la tecnología DOM virtual para optimizar el rendimiento de la renderización de páginas. Al crear un árbol DOM virtual en memoria, Vue.js puede calcular rápidamente las actualizaciones mínimas del DOM cuando cambian los datos, reduciendo así el número de operaciones en el DOM real y mejorando el rendimiento.
- Desarrollo basado en componentes: Vue.js fomenta el uso del desarrollo basado en componentes , la interfaz se divide en componentes independientes , cada componente es responsable de su propia lógica de negocio y visualización de la vista . Este desarrollo por componentes hace que el código sea más modular, fácil de mantener y reutilizar.
- Vue.js implementa la vinculación de datos bidireccional, de modo que los cambios en los datos se pueden reflejar automáticamente en la vista, pero también soporta el flujo de datos unidireccional. Este enlace de datos responsivo mantiene los datos sincronizados con la vista y mejora la eficiencia del desarrollo.

- Plug-ins y extensiones : Vue.js tiene una rica biblioteca de plug-ins y extensiones , los desarrolladores pueden introducir fácilmente las bibliotecas de terceros de acuerdo con los requisitos del proyecto para ampliar la funcionalidad de Vue.js para mejorar la eficiencia del desarrollo .

Ventajas de Vue.js:

- Fácil de aprender : Vue.js API y la sintaxis es simple y fácil de entender , la curva de aprendizaje es relativamente plana , incluso los principiantes pueden empezar rápidamente .
- Marco incremental: La naturaleza incremental de Vue.js permite a los desarrolladores introducir sus características gradualmente según las necesidades y también facilita su integración en proyectos existentes.
- Responsive Data Binding: Vue.js implementa un eficiente responsive data binding que mantiene los datos sincronizados con la vista y elimina la necesidad de que los desarrolladores manipulen manualmente el DOM.
- Desarrollo basado en componentes: Vue.js fomenta el desarrollo basado en componentes, lo que hace que el código sea más modular y reutilizable, y facilita la colaboración en equipo.
- Ecosistema rico: Vue.js tiene un gran ecosistema con muchos plugins, extensiones y herramientas para elegir, por lo que es altamente escalable.

Desventajas de Vue.js

- Ecosistema relativamente pequeño: en comparación con React y Angular, el ecosistema de Vue.js es todavía relativamente pequeño y puede carecer de soluciones maduras en ciertas áreas.
- Comunidad relativamente pequeña: aunque la comunidad de Vue.js está creciendo, sigue siendo relativamente pequeña en comparación con React y Angular, lo que puede significar que la documentación y el soporte correspondientes sean relativamente limitados.
- Menos consistente que Angular: Vue.js puede no ser tan consistente como Angular, ya que la naturaleza incremental de Vue.js permite a los desarrolladores ser flexibles en su elección de funcionalidad, pero también puede conducir a implementaciones inconsistentes en los proyectos.
- En términos de extensiones apropiadas, Vue.js también tiene algunas ventajas en el desarrollo de aplicaciones móviles y el desarrollo de aplicaciones multiplataforma. Debido a su naturaleza ligera y progresiva, Vue.js puede utilizarse para crear aplicaciones móviles y PWA (aplicaciones web progresivas) de alto rendimiento. Además, Vue.js tiene una buena integración con algunos frameworks y librerías (como Ionic y Quasar), lo que facilita a los desarrolladores la creación de aplicaciones multiplataforma.

3.4.3 Angular

Angular es un popular framework JavaScript front-end mantenido y desarrollado por Google. Tiene un rico conjunto de características y un ecosistema robusto para el desarrollo de aplicaciones de una sola página y aplicaciones web complejas. Las características, ventajas y desventajas del framework Angular se describen en detalle a continuación.

Características de Angular Framework:

- Arquitectura Componentised : Angular utiliza la arquitectura componentised para dividir la aplicación en componentes separados. Cada componente tiene su propia plantilla, estilo y lógica de negocio , haciendo el código más modular y fácil de mantener .
- Potente sintaxis de plantillas: la sintaxis de plantillas de Angular admite directivas enriquecidas y vinculación de datos, lo que permite a los desarrolladores una manipulación más flexible del DOM y la visualización de datos.
- Vinculación bidireccional de datos: Angular implementa la vinculación bidireccional de datos, de modo que los cambios en los datos pueden reflejarse automáticamente en la vista, mientras que los cambios en la vista también pueden actualizarse de forma sincrónica en el modelo de datos.
- Inyección de dependencias: Angular tiene incorporado un mecanismo de inyección de dependencias que facilita la gestión de dependencias entre componentes, lo que facilita la reutilización y la prueba de componentes.
- Enrutamiento y navegación: Angular proporciona potentes características de enrutamiento y navegación que permiten a los desarrolladores cargar diferentes componentes basados en URLs, permitiendo la navegación en aplicaciones de una sola página.
- Potentes herramientas CLI: Angular proporciona herramientas de interfaz de línea de comandos (CLI) para crear, compilar y probar rápidamente aplicaciones Angular, mejorando enormemente la eficiencia del desarrollo.
- Desarrollo multiplataforma: Angular admite el desarrollo de aplicaciones móviles mediante el framework Ionic, lo que permite a los desarrolladores utilizar la misma pila tecnológica para aplicaciones web y móviles.
- Comunidad grande y activa: Angular tiene una comunidad grande y activa, lo que significa que hay una gran cantidad de documentación, tutoriales y plugins disponibles, así como soporte técnico oportuno.

Pros del framework Angular:

- Potente: Angular proporciona un rico conjunto de características y herramientas que permiten a los desarrolladores construir aplicaciones web complejas de manera más eficiente.
- Vinculación bidireccional de datos: La característica de vinculación bidireccional de datos de Angular permite que los datos y la vista permanezcan sincronizados, los desarrolladores no necesitan manipular manualmente el DOM.
- Arquitectura basada en componentes: la arquitectura basada en componentes de Angular hace que el código sea más modular y fácil de mantener, a la vez que facilita la reutilización y las pruebas.
- Enrutamiento y navegación: Angular proporciona potentes funciones de enrutamiento y navegación que facilitan y hacen más eficiente el desarrollo de aplicaciones de una sola página.
- Desarrollo multiplataforma : Angular soporta el desarrollo de aplicaciones móviles utilizando el framework Ionic , permitiendo a los desarrolladores construir aplicaciones web y móviles bajo la misma pila tecnológica .

Desventajas:

- Curva de aprendizaje pronunciada: en comparación con otros frameworks de front-end como React y Vue.js, Angular tiene una curva de aprendizaje pronunciada y lleva algún tiempo dominar sus conceptos y funcionamientos complejos.
- Grande y complejo: Angular es grande debido al rico conjunto de características y herramientas que proporciona, lo que puede aumentar los tiempos de carga de la aplicación y los gastos generales de rendimiento.
- Problemas de rendimiento: cuando se trata de datos a gran escala, Angular puede no funcionar tan bien como frameworks como React y debe optimizarse razonablemente para garantizar una mejor experiencia de usuario.
- No es muy adecuado para proyectos pequeños: Angular puede ser un poco demasiado potente para proyectos pequeños y no es muy adecuado para aplicaciones de páginas simples.
- Cuando se trata de escalar adecuadamente, Angular sobresale en grandes proyectos a nivel empresarial y en el desarrollo de aplicaciones complejas. Debido a su potencia y extensibilidad, Angular es ampliamente utilizado para desarrollar aplicaciones de nivel empresarial, sistemas de gestión, plataformas de comercio electrónico y otras aplicaciones web complejas. Además, Angular también se utiliza ampliamente en grandes equipos y organizaciones que a menudo necesitan un marco front-end maduro y potente para hacer frente a requisitos empresariales complejos y grandes bases de código.

Esta tabla proporciona información sobre las características clave de tres populares frameworks de front-end (Angular, React y Vue), destacando Angular en proyectos de mayor envergadura, y React y Vue por ser más adecuados para aplicaciones de distintos tamaños y relativamente fáciles de aprender.

Característica	**Angular**	**React**	**Vue**
Tipo	Marco Frontal	Biblioteca Frontal	Marco Frontal
Año de Lanzamiento	2016	2013	2014
Lenguaje	TypeScript	JavaScript	JavaScript
Tamaño del Ecosistema	Grande	Grande	Mediano
Enlace de Datos Bidireccional	Sí	Sí	Sí
Arquitectura de Componentes	Sí	Sí	Sí
Virtual DOM	No	Sí	Sí
Curva de Aprendizaje	Relativamente pronunciada	Moderada	Moderada
Rendimiento	Bueno	Muy bueno	Bueno
Soporte Comunitario	Fuerte	Fuerte	Fuerte
Casos de Uso	Aplicaciones de gran escala	Varias escalas	Varias escalas

Tabla 3.4.3

4. Solución propuesto

En este esbozo de solución de proyecto, proporcionaremos información sobre cómo planificar y crear una aplicación web compleja. El proyecto abarca todos los aspectos, incluido el plan de trabajo, la arquitectura del sistema, el diseño detallado, las tecnologías utilizadas y mucho más. Recorreremos las siguientes secciones para comprender la solución en detalle.

4.1 Plan de Trabajo

La planificación es clave para el éxito de un proyecto. En esta sección analizaremos cómo desarrollar un plan eficaz para garantizar que el proyecto se complete a tiempo y dentro del presupuesto. Esto incluye la elaboración de un calendario, la asignación de recursos, la gestión de riesgos y un plan de comunicación para el proyecto. Nos centraremos en las distintas fases del proyecto y nos aseguraremos de que se ha prestado la debida atención y planificación a cada una de ellas.

Fase 1: Planificación del proyecto y configuración del entorno (2 semanas)

- Determinar los requisitos del proyecto y la lista de características.
- Diseñar el modelo de base de datos, incluyendo información del usuario, información de la mascota, ubicación geográfica y otras tablas.
- Construir el entorno del proyecto Django, configurar la conexión a la base de datos y los ajustes básicos.
- Crear proyecto Vue, configurar enrutamiento y diseño de página.

Fase 2: Autenticación de usuarios y gestión de la información (2 semanas)

- Implementar la función de registro de usuarios, guardar la información del usuario en la base de datos.
- Desarrollar la función de inicio de sesión de usuario, utilizar JWT para la autenticación y autorización de usuarios.
- Permitir a los usuarios modificar su información personal e implementar funciones básicas de gestión de la información del usuario.

Fase 3: Gestión de la información sobre mascotas (2 semanas)

- Añadir/modificar las funciones de información sobre mascotas, incluyendo subir y guardar fotos de mascotas.
- Asociar la información de las mascotas a los usuarios para garantizar que cada usuario sólo pueda gestionar la información de su propia mascota.
- Desarrollar la función de actualización de la ubicación de la mascota para guardar su ubicación geográfica en la base de datos.

Fase 4: Seguimiento de mapas y ubicaciones (2 semanas)

- Integrar Gaode map para mostrar la ubicación de la mascota en tiempo real.
- Configurar una geo-valla para activar una alerta de alarma cuando la ubicación de la mascota esté fuera de la valla.
- Implementar la función de comprobar la ubicación de la mascota cada dos segundos y registrar la ubicación en la base de datos.

Fase 5: Alarma y registros médicos (2 semanas)

- Registra la información de la alarma, incluida la hora y la ubicación que la activaron.
- Permite a los usuarios añadir y ver los registros médicos de sus mascotas, incluida información como vacunas e historial médico.

Fase 6: Optimización y pruebas del front-end (1 semana)

- Optimizar el diseño y la experiencia de usuario de la página front-end.
- Realizar pruebas del sistema para garantizar la estabilidad y el rendimiento de las funciones.
- Corregir posibles errores y problemas.

Etapa 7: Despliegue y puesta en marcha (1 semana)

- Desplegar la aplicación en el servidor o la plataforma en la nube.
- Realizar las pruebas finales y la validación.
- Poner en marcha la aplicación y dejar que los usuarios empiecen a utilizar la aplicación de seguimiento de mascotas.

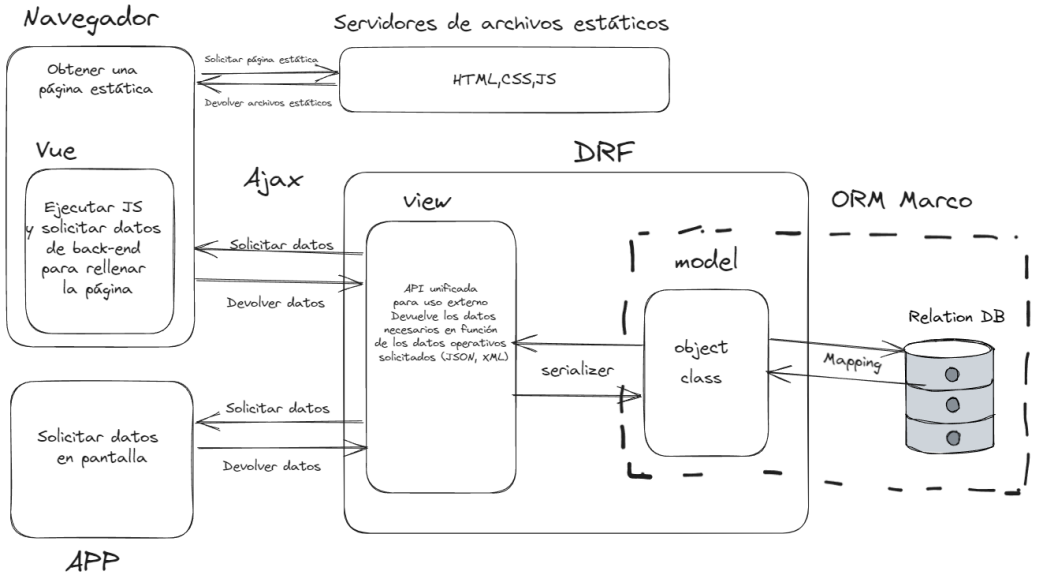
Resumir

Este plan de trabajo completará el desarrollo y despliegue de la aplicación de seguimiento de mascotas en 12 semanas. Cada fase tiene objetivos y tareas claros y se ha programado adecuadamente en función de los requisitos del proyecto. Durante el

proceso de desarrollo, utilizaremos el framework Django con el front-end Vue para la autenticación de usuarios, la gestión de la información de las mascotas, la visualización de mapas y el seguimiento de la ubicación. En la fase final, optimizaremos y probaremos la aplicación y, por último, la pondremos en línea para que los usuarios experimenten esta útil aplicación de seguimiento de mascotas.

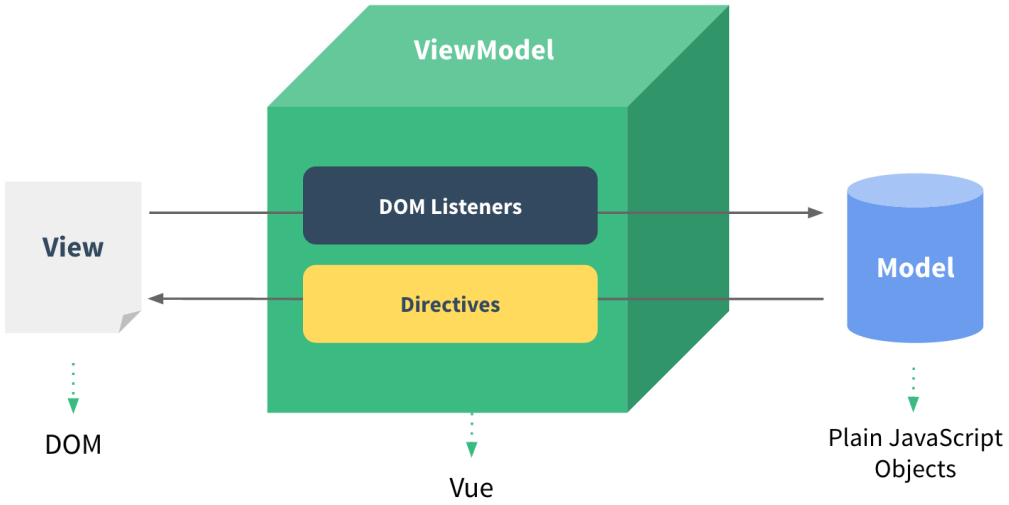
4.2 Arquitectura del Sistema

La arquitectura del sistema es el esqueleto del proyecto, que define la estructura y los componentes de todo el sistema. En esta solución, utilizamos una arquitectura separada de front-end y back-end, donde usamos Vue.js para el front-end y Django para el back-end, junto con una base de datos bien diseñada. Vamos a discutir en detalle lo que hace cada componente, cómo interactúa y cómo asegurarse de que trabajan juntos.



4.2.1 Parte front-end (Vue.js)

En Vue, MVVM significa Model-View-ViewModel, un patrón de arquitectura de software para crear aplicaciones web. Ayuda a implementar una vinculación de datos bidireccional efectiva entre la interfaz de usuario del front-end (Vista) y el modelo de datos del back-end (Modelo), con la capa ViewModel actuando como puente. A continuación se ofrece una descripción detallada de cada parte de MVVM:



Vista: Una vista es un elemento DOM de la interfaz de usuario que es responsable de mostrar el contenido en la página. En Vue, las vistas suelen consistir en plantillas HTML y se muestran en el navegador del usuario.

Modelo: El modelo representa la capa de datos y normalmente consiste en datos extraídos de una base de datos u otra fuente de datos. En Vue, los datos del modelo se almacenan en el atributo data y representan el estado y el contenido de la aplicación.

ViewModel: ViewModel es el núcleo del framework Vue.js, que se implementa en el código fuente de Vue y es responsable de gestionar los datos y la lógica.La principal tarea del ViewModel es conectar la vista (View) con el modelo (Model) para la vinculación bidireccional de datos. Obtiene los datos del modelo back-end a través de una interfaz y se asegura de que la representación de estos datos en la vista front-end esté sincronizada.Una instancia de Vue es una instancia de ViewModel.

Un concepto importante del patrón MVVM es la vinculación bidireccional de datos, lo que significa que cuando los datos del modelo cambian, la vista se actualiza automáticamente para reflejar esos cambios, y las acciones del usuario en la vista pueden retroalimentarse al modelo. Esta actualización y sincronización dinámica de los datos simplifica enormemente el desarrollo del front-end.

Vamos a repasar los componentes de página front-end en Vue.js y lo que hacen en una aplicación front-end típica. A continuación se muestra una breve descripción de cada componente de página:

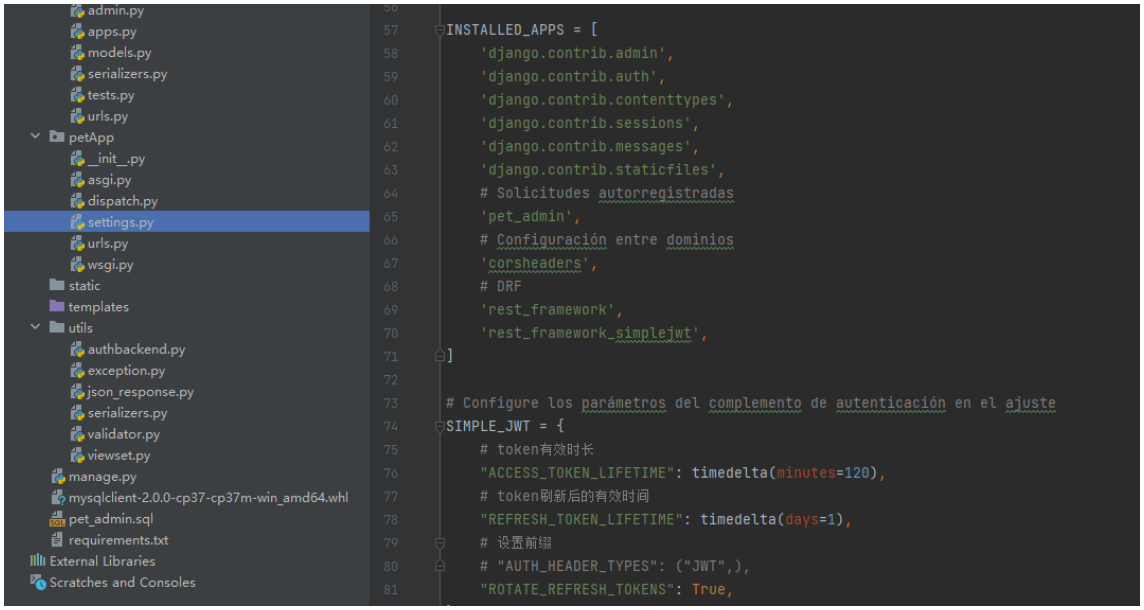
Página de registro: los usuarios pueden introducir información personal y enviarla al backend para registrarse.
Página de inicio de sesión: los usuarios pueden introducir su cuenta registrada y contraseña para iniciar sesión y obtener el JWT (JSON Web Token) devuelto por el backend para la autenticación de solicitudes posteriores.
Página de modificación de la información del usuario: permite al usuario modificar su información básica, incluido el nombre de usuario, la contraseña, la información de contacto, etc.
Página de adición/modificación de la información de la mascota: permite a los usuarios añadir o modificar la información de la mascota, incluido su nombre, especie, foto, etc.
Página de mapa: muestra la ubicación en tiempo real de la mascota a través de la API de mapas de AutoNavi y permite establecer geocercas.
Página de ventana emergente: cuando la ubicación de la mascota está fuera del alcance de la geo-valla, una ventana emergente avisa al usuario.
Página de información de alarma: muestra una lista de información de alarma, los usuarios pueden ver la información de alarma relacionada.
Página de registros médicos de mascotas: muestra una lista de los registros médicos de las mascotas, los usuarios pueden añadir nuevos registros.

4.2.2 Parte backend (Django):

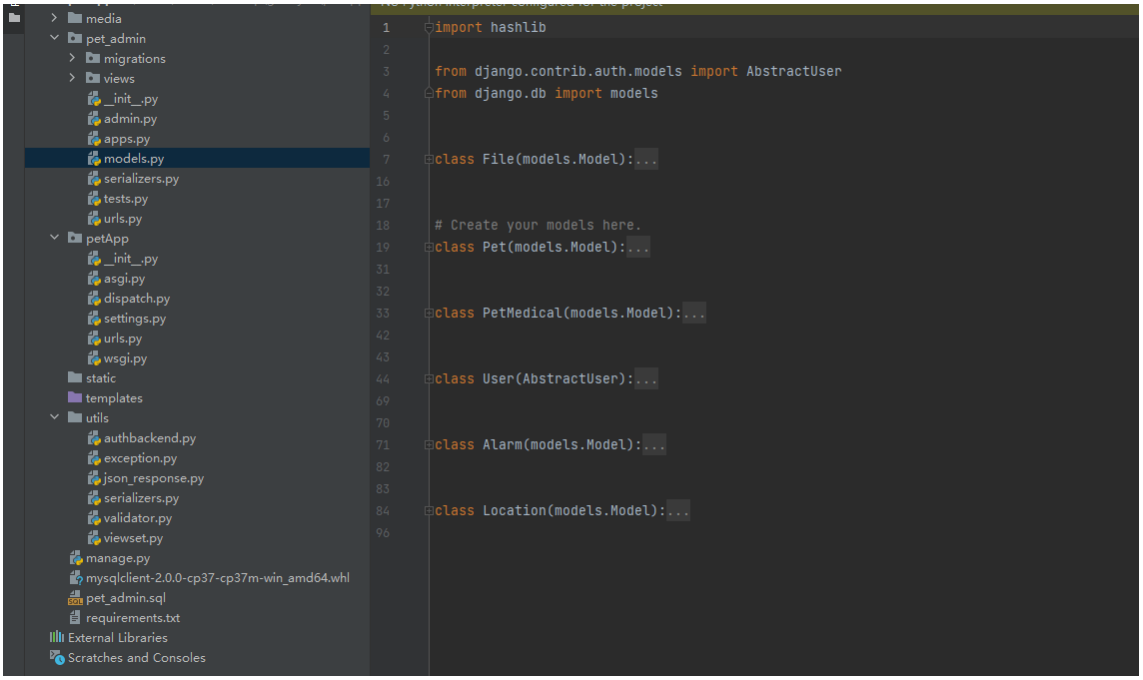
Django proporciona un rico conjunto de características como un marco de backend, incluyendo enrutamiento, procesamiento de formularios, acceso a bases de datos, y mucho más. Exploraremos en detalle cómo utilizar Django para construir servicios backend robustos y cómo manejar las solicitudes de datos y la lógica de negocio.

Django es un potente framework backend de Python para construir robustos servicios backend. Estos son los pasos clave para construir un servicio backend usando Django:

- Creación y configuración del proyecto: crear un nuevo proyecto utilizando las herramientas de línea de comandos de Django y luego configurar los ajustes del proyecto, incluyendo la configuración de la base de datos, configuración de archivos estáticos, configuración de seguridad, etc.



- Definición de Modelos: Define modelos de datos que serán mapeados a tablas de base de datos.El ORM (Object Relational Mapping) de Django hace que la definición de modelos sea muy fácil y puedes usar clases de Python para definir estructuras de datos.



- Procesamiento de vistas: Crea funciones de vista que manejarán las peticiones HTTP desde el front-end. Estas funciones de vista serán responsables de manejar las solicitudes de datos y la lógica de negocio. Puede consultar la base de datos, validar datos, ejecutar lógica de negocio, etc. en la vista.

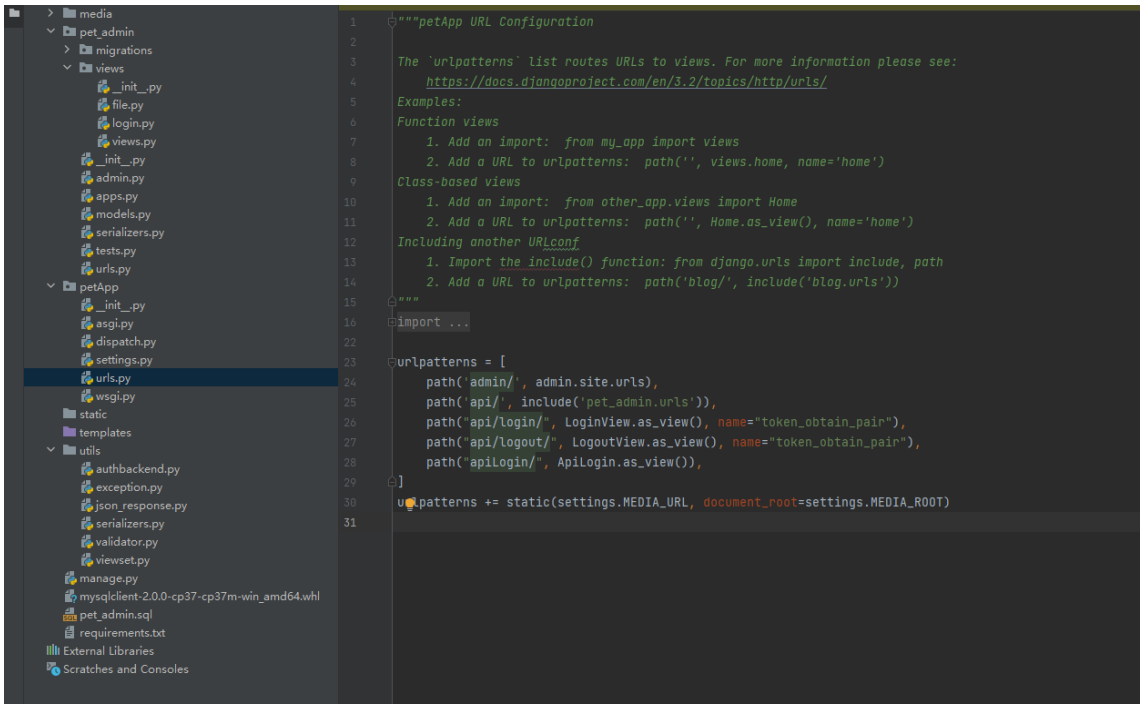
```python
# Create your views here.
from rest_framework import viewsets
from rest_framework.decorators import action
from rest_framework.permissions import IsAuthenticated, AllowAny

from utils.json_response import DetailResponse, ErrorResponse
from utils.viewset import CustomModelViewSet
from pet_admin.models import Pet, Location, Alarm, User, PetMedical
from pet_admin.serializers import PetSerializer, AlarmSerializer, UserSerializer, LocationSerializer, \
    UserCreateSerializer, UserInfoUpdateSerializer, PetMedicalSerializer


class PetViewSet(CustomModelViewSet):
    """
        interfaz de usuario
        list:
        create:
        update:
        retrieve:
        destroy:
        """
    queryset = Pet.objects.all()
    serializer_class = PetSerializer

    @action(methods=["GET"], detail=False, permission_classes=[IsAuthenticated])
    def get_pet_by_user(self, request):
        user = request.user
        user_id = user.id
        try:
            pet = Pet.objects.get(user_id=user_id)
        except:
            return ErrorResponse(msg="Please add your pet")
        else:
            pet = PetSerializer(pet)
            return DetailResponse(data=pet.data)
```

- Enrutamiento: Configura el enrutamiento de URL para asignar diferentes URLs a las funciones de vista apropiadas.Django utiliza el enrutamiento para determinar qué vista maneja una solicitud de URL en particular.



- Procesamiento de formularios: Si necesitas recoger datos de usuario en el front-end, Django proporciona potentes capacidades de procesamiento de formularios. Puedes crear clases de formularios para validar y procesar los datos enviados por los usuarios.
- API RESTful (opcional): Si necesitas construir APIs RESTful, Django Rest Framework (DRF) es una poderosa herramienta que te ayudará a crear APIs rápidamente y manejar serialización, autenticación, permisos y más.
- Middleware y seguridad: Django proporciona middleware para gestionar peticiones y respuestas. También tiene características de seguridad integradas, como la protección contra la falsificación de solicitudes en sitios cruzados (CSRF), la autenticación, etc.
- Despliegue: Por último, despliega tu aplicación Django en un servidor, con la opción de utilizar una variedad de servidores web y bases de datos.

A continuación se enumeran los diferentes módulos de un proyecto Django completo que desempeñan diferentes funciones en la construcción de una aplicación web completa.

Juntos, estos módulos construyen un potente proyecto Django que permite a los usuarios registrarse, gestionar información personal, realizar un seguimiento de la información y geolocalización de la mascota, recibir mensajes de alerta y registrar el historial médico. Esta completa aplicación puede ser utilizada por los dueños de mascotas para garantizar la seguridad y el bienestar de sus animales de compañía y proporciona una cómoda herramienta de gestión.

Sistema de autenticación de usuarios: gestiona el registro de usuarios, el inicio de sesión y la verificación de generación de JWT.
Gestión de la información del usuario: gestiona la información personal del usuario, incluyendo las operaciones de añadir, eliminar, cambiar y comprobar.
Gestión de información de mascotas: gestiona la información básica de las mascotas, incluyendo operaciones de añadir, eliminar, cambiar y comprobar, así como subir y guardar fotos de mascotas.
Gestión de la información de ubicación geográfica: Registre la información de ubicación geográfica de la mascota y asóciela al usuario.
Gestión de la información de alarma: Gestione la información de alarma de las mascotas que superen el alcance de la geo-valla y regístrela en la base de datos.
Gestión de historiales médicos: Gestionar los historiales médicos de las mascotas, incluyendo las operaciones de añadir, eliminar, cambiar y comprobar.

4.2.3 Diseño de la base de datos:

Una base de datos es el corazón de cualquier aplicación, ya que almacena y gestiona datos. Diseñaremos una arquitectura de base de datos adecuada, analizaremos las estructuras de las tablas, las relaciones y cómo garantizar la integridad y coherencia de los datos.

Hablamos del diseño de una base de datos que consta de las siguientes tablas:

Tabla de información del usuario: almacena la información de registro del usuario, incluido el nombre de usuario, la contraseña, la información de contacto, etc.
Tabla de información de mascotas: almacena la información básica de las mascotas, incluido su nombre, especie, etc.
Tabla de información de ubicación geográfica: registra la información de ubicación geográfica de la mascota, incluyendo latitud y longitud, marca de tiempo, etc.
Tabla de información de alarma: almacena la información de alarma de la mascota fuera del alcance de la geo-valla, incluyendo hora de alarma, ubicación, etc.
Tabla de historial médico: Almacena el historial médico de la mascota, incluida la hora de consulta, la información de diagnóstico, etc.

Entre estas tablas, creamos una serie de relaciones para organizar mejor los datos:

- Existe una relación de uno a muchos entre la tabla de información sobre usuarios y la tabla de información sobre mascotas, lo que significa que un usuario puede tener varias mascotas, pero sólo un propietario por mascota.
- También existe una relación de uno a muchos entre la tabla de información de mascotas y la tabla de información de geolocalización, lo que indica que una mascota puede tener varios registros de información de localización, lo que ayuda a realizar un seguimiento de los movimientos de la mascota.
- También se establece una relación de uno a muchos entre la tabla de información sobre mascotas y la tabla de información sobre alarmas, lo que permite que una mascota tenga múltiples mensajes de alarma para garantizar su seguridad.
- La misma relación de uno a muchos se establece entre la tabla de Información de la mascota y la tabla de Historiales médicos, lo que permite que una mascota tenga múltiples historiales médicos, ayudando a mantener el perfil de salud de la mascota.

Para garantizar la integridad y coherencia de los datos, se adoptaron las siguientes medidas:

- Para garantizar la unicidad de los datos y la corrección de las relaciones, se utilizaron restricciones de la base de datos como la clave primaria, la clave externa y las restricciones de unicidad.
- Hemos insistido en la importancia de la validación de entradas y datos para evitar que se introduzcan en la base de datos datos datos no válidos o ilegítimos y garantizar así la calidad de los datos.
- Utilizamos transacciones para gestionar operaciones de datos complejas, especialmente en lo que respecta a la información sobre la ubicación de las mascotas y la generación de alertas, para garantizar la coherencia de los datos, que es fundamental para mantener la seguridad de las mascotas.
- Se realizaron copias de seguridad periódicas de la base de datos para evitar su pérdida, y se estudiaron planes de recuperación ante desastres para garantizar que los datos pudieran recuperarse rápidamente en caso de circunstancias impredecibles.
- También se hizo hincapié en la necesidad de utilizar índices de base de datos para mejorar el rendimiento de las consultas, sobre todo en el caso de los datos consultados con frecuencia, lo que puede acelerar enormemente la recuperación de datos.

Teniendo en cuenta la estructura de estas tablas, las relaciones y la validación de los datos, esta base de datos está diseñada para ayudar a crear aplicaciones robustas y garantizar la integridad y coherencia de los datos para ofrecer servicios fiables a los usuarios.

4.2.4 Proceso de interacción con el sistema:

La interacción con el sistema es la clave de la interacción de los usuarios con una aplicación. Exploraremos el diseño detallado de los procesos de inicio de sesión, solicitud de datos y envío de formularios para garantizar una experiencia de usuario fluida y sin fallos.

En la siguiente tabla se describen los distintos flujos de usuarios de una aplicación en función de las interacciones front-end y back-end

Proceso de registro del usuario: los usuarios introducen la información de registro en la página del front-end, envían solicitudes de registro al back-end, y el back-end guarda la información en la tabla de información del usuario.
Proceso de inicio de sesión del usuario: los usuarios introducen la información de inicio de sesión en la página del front-end, envían la solicitud de inicio de sesión al back-end, el back-end verifica la información y devuelve el JWT, y el front-end guarda el JWT para posteriores solicitudes de autenticación.
Proceso de modificación de la información del usuario: el usuario modifica la información personal en la página del front-end, envía la solicitud de modificación al back-end, y el back-end actualiza los datos correspondientes en la tabla de información del usuario.
Proceso de adición/modificación de información sobre mascotas: el usuario añade o modifica información sobre mascotas en la página del front-end, envía una solicitud al back-end, y el back-end actualiza la tabla de información sobre mascotas.
Proceso de la página de mapas: el front-end muestra la ubicación en tiempo real de la mascota a través de la API de mapas de AutoNavi y envía una solicitud al back-end cada dos segundos para obtener la información de geolocalización más reciente.
Proceso de configuración de la geo-valla: los usuarios configuran el alcance de la geo-valla en la página del front-end, envían una solicitud al back-end y éste guarda la configuración en la base de datos.
Proceso de información de alarma: el back-end supervisa la ubicación geográfica de la mascota y, si supera el intervalo de la geo-valla, guarda la información de alarma en la tabla de información de alarma y abre una ventana emergente en el front-end.
Proceso de historial médico de la mascota: los usuarios añaden o consultan el historial médico de la mascota en la página del front-end, y el back-end guarda el historial en la tabla de historial médico.

4.2.5 Resumen

Al final de la sección Arquitectura del sistema, resumiremos la arquitectura general, destacando las relaciones y la colaboración entre los distintos componentes y la forma de garantizar el rendimiento, la seguridad y la mantenibilidad del sistema.

La estructura del sistema de este proyecto incluye la parte de front-end (Vue.js) y la parte de back-end (Django), que realiza las funciones de registro de usuarios, inicio de sesión, modificación de información, adición y modificación de información de mascotas, visualización de mapas, configuración de geo-vallas, registro de información de alarmas y gestión de historiales médicos de mascotas mediante la interacción de front-end y back-end. El front-end realiza la visualización y el seguimiento de la ubicación de la mascota a través de la API Gaode Map, y el back-end se encarga de gestionar la información del usuario y de la mascota, y de interactuar con la base de datos. Todo el sistema tiene una estructura clara y funciones perfectas, que pueden satisfacer los requisitos del proyecto de aplicación de seguimiento de mascotas.

4.3 Diseño Detallado

El diseño detallado es un paso fundamental para traducir la arquitectura del sistema en código real. En esta sección hablaremos en detalle de las clases y las relaciones, definiremos la estructura de los módulos, seguiremos optimizando el diseño de la base de datos y diseñaremos la estructura de directorios de la aplicación.

4.3.1.Clases y relaciones:

En el diseño detallado, definiremos las clases del sistema y las relaciones entre ellas. Estas clases se asignarán directamente a la implementación real del código y definirán el modelo de datos y la lógica empresarial

4.3.1.1 Clase Usuario (User):

- Atributos: ID de usuario (user_id), nombre de usuario (username), contraseña (password), información de contacto (contact_info), etc.
- Relación: la información de usuario y mascota (Pet) es una relación de uno a muchos, un usuario puede tener más de una mascota.

```python
class User(AbstractUser):
    id = models.AutoField("id", primary_key=True)
    username = models.CharField("ID de usuario", max_length=255, unique=True)
    password = models.CharField("clave", max_length=255)
    nickname = models.CharField("término cariñoso", max_length=255)
    birthday = models.DateField("cumpleaños")
    phone = models.CharField("número de teléfono móvil", max_length=255)
    create_time = models.DateTimeField("Tiempo de creación", auto_now=True)
    REQUIRED_FIELDS = []
    first_name = None
    groups = None
    user_permissions = None
    last_name = None
    email = None
    is_staff = None
    is_active = True
    date_joined = None
    last_login = None
    is_superuser = None

    def set_password(self, raw_password):
        super().set_password(hashlib.md5(raw_password.encode(encoding="UTF-8")).hexdigest())

    class Mate:
        db_tabel = 'user'
```

4.3.1.2 Clase de información sobre mascotas (Pet):

- Atributos: ID de la mascota (pet_id), nombre de la mascota (pet_name), especie (species), foto (photo), etc.
- Relaciones: las mascotas y los usuarios tienen una relación de muchos a uno, y cada mascota pertenece a un usuario.

```python
class Pet(models.Model):
    id = models.AutoField("id", primary_key=True)
    name = models.CharField("nombre", max_length=255)
    age = models.IntegerField("ano")
    breed = models.CharField("tipo", max_length=255)
    user_id = models.IntegerField("user id")
    # medical_records = models.CharField("medical records", max_length=255)
    image = models.CharField("picture", max_length=255)
    create_time = models.DateTimeField("tiempo crear", auto_now=True)

    class Mate:
        db_tabel = 'pet_info'
```

4.3.1.3 Clase de información de ubicación (Location):

- Atributos: ID de ubicación (location_id), ID de mascota (pet_id), longitud, latitud, fecha y hora, etc.
- Relación: la información de localización geográfica y la información de mascotas (Pet) es una relación de uno a muchos, cada mascota tiene múltiples registros de información de localización.

```
84  class Location(models.Model):
85      id = models.AutoField("id", primary_key=True)
86      pet_id = models.IntegerField("pet id")
87      pet_name = models.CharField("pet name", max_length=255)
88      status = models.IntegerField("Alarma o no (1.si 0.no)", default=0, null=True)
89      location = models.CharField("location", max_length=255, null=True)
90      longitude = models.CharField("longitude", max_length=255)
91      latitude = models.CharField("latitude", max_length=255)
92      create_time = models.DateTimeField("time crear", auto_now=True)
93
94      class Mate:
95          db_tabel = 'location'
```

4.3.1.4 Clase de información sobre alarmas (Alarm):

- Atributos: ID de alarma (alarm_id), ID de mascota (pet_id), hora de alarma (alarm_time), ubicación de alarma (alarm_location), etc.
- Relación: La información sobre alarmas y mascotas (Pet) es una relación de uno a muchos, cada mascota puede tener varios registros de información sobre alarmas.

```
71  class Alarm(models.Model):
72      id = models.AutoField("id", primary_key=True)
73      alarm_type = models.CharField("报警类型", max_length=255, null=True)
74      alarm_content = models.CharField("报警内容", max_length=255, null=True)
75      location = models.CharField("地点", max_length=255, null=True)
76      longitude = models.CharField("经度", max_length=255)
77      latitude = models.CharField("纬度", max_length=255)
78      create_time = models.DateTimeField("创建时间", auto_now=True)
79
80      class Mate:
81          db_tabel = 'alarm_info'
```

4.3.1.5 Clase de historia clínica (MedicalRecord):

- Atributos: ID del registro (record_id), ID de la mascota (pet_id), hora de la visita (visit_time), información sobre el diagnóstico (diagnosis_info), etc.
- Relación: el registro médico y la información de la mascota (Pet) es una relación de uno a muchos, cada mascota puede tener varios registros médicos.

```
33  class PetMedical(models.Model):
34      id = models.AutoField("id", primary_key=True)
35      name = models.CharField("姓名", max_length=255)
36      pet_id = models.IntegerField("宠物id")
37      medical_date = models.DateField('诊断时间')
38      create_time = models.DateTimeField("创建时间", auto_now=True)
39
40      class Mate:
41          db_tabel = 'pet_info'
```

4.3.2.Estructura de los módulos

La estructura modular define cómo se organiza el código de una aplicación. Analizaremos cómo puede dividirse el código en módulos para garantizar la mantenibilidad y la extensibilidad.

4.3.2.1 Módulo backend:

La siguiente tabla describe los diferentes módulos funcionales del módulo backend que trabajan conjuntamente para dar soporte a la funcionalidad principal de la aplicación. Un breve resumen es el siguiente:

- Módulo de autenticación de usuarios: gestiona el registro de usuarios, el inicio de sesión, genera y valida JWT.

- Módulo de información del usuario: responsable de gestionar la adición, eliminación, modificación y comprobación de la información del usuario.

- Módulo de información sobre mascotas: se encarga de añadir, eliminar, modificar y comprobar la información sobre mascotas, así como de cargar y guardar fotos de mascotas.

- Módulo de información sobre la ubicación geográfica: registra la información sobre la ubicación geográfica de la mascota y la asocia al usuario.

- Módulo de información de alarma: gestiona la información de alarma cuando la mascota sobrepasa el radio de acción de la geo-valla y la guarda en la base de datos.

- Módulo de historiales médicos: gestiona los historiales médicos de la mascota, incluidas las operaciones de añadir, eliminar, modificar y comprobar.

El módulo backend incluye los módulos de autenticación de usuario, información de usuario, información de mascotas, información de geolocalización, información de alarmas y registros médicos. El módulo de autenticación de usuarios gestiona el registro de usuarios, el inicio de sesión y la generación y validación de JWT para garantizar la autenticación segura de los usuarios. El módulo de información del usuario se encarga de gestionar la información del usuario, incluyendo la adición, eliminación, modificación y verificación, lo que está estrechamente relacionado con la autenticación del usuario. El módulo de información de la mascota se utiliza para gestionar la información de la mascota, incluyendo la adición, eliminación, modificación y verificación de la información básica, así como la carga y almacenamiento de fotos de la mascota. El módulo de información sobre la ubicación geográfica registra la información sobre la ubicación de la mascota y se asocia con el usuario para el seguimiento cartográfico. El módulo de información de alarma gestiona la información de alarma cuando la mascota supera el alcance de la geo-valla y la almacena en la base de datos. El módulo de historial médico gestiona el historial médico de la mascota, incluidas las operaciones de adición, eliminación, modificación y comprobación para mantener el historial médico de la mascota.

Estos módulos están interrelacionados, por ejemplo, los módulos de información del usuario y de información de la mascota pueden utilizarse para asociar una mascota a un usuario, el módulo de información de geolocalización puede utilizarse para la generación de información de alarma, y el módulo de historial médico está asociado a la información de la mascota. Juntos, estos módulos forman un sistema backend completo que soporta la gestión de usuarios, la gestión de mascotas, la monitorización de la geolocalización y el mantenimiento del historial médico.

4.3.2.2 Módulo Frontend:

Estos módulos frontales constituyen la interfaz y el enlace entre el usuario y la funcionalidad back-end.

Página de registro: el usuario introduce la información de registro y envía la solicitud de registro al backend para su procesamiento.
Página de inicio de sesión: el usuario introduce la información de inicio de sesión y envía la solicitud de inicio de sesión al backend para su verificación.
Página de modificación de la información del usuario: el usuario modifica la información personal y envía una solicitud al backend para actualizar la información del usuario.
Página de adición/modificación de información de la mascota: el usuario añade o modifica la información de la mascota, carga la foto de la mascota y envía la solicitud al backend para guardarla.
Página de mapa: utiliza la API de AutoNavi Maps para mostrar la ubicación en tiempo real de la mascota y envía solicitudes al backend cada dos segundos para obtener la información de ubicación más reciente.
Página de configuración de geo-valla: los usuarios configuran el alcance de la geo-valla y envían solicitudes al backend para guardar la configuración.
Página de información de alarma: muestra una lista de la información de alarma de la mascota, y los usuarios pueden ver la información de alarma relacionada.
Página de historiales médicos de mascotas: muestra el historial médico de la mascota, y los usuarios pueden añadir nuevos historiales médicos.

Los módulos front-end incluyen una página de registro, una página de inicio de sesión, una página de modificación de la información del usuario, una página de adición/modificación de la información de la mascota, una página de mapas, una página de configuración de geocercas, una página de información de alertas y una página de historial médico de la mascota. La página de registro permite al usuario introducir información de registro y envía la información al backend para su procesamiento con el fin de crear una nueva cuenta de usuario. La página de inicio de sesión se utiliza para que el usuario introduzca la información de inicio de sesión y valide el JWT devuelto por el backend para la autenticación. La página Modificar Información de Usuario permite a los usuarios actualizar su información personal enviando una solicitud al backend. La página Añadir/Modificar información de la mascota permite a los usuarios gestionar la información y las fotos de la mascota, y guardar los cambios en el backend. La página Mapa muestra la ubicación de la mascota en tiempo real utilizando la API de mapas de Goldmind y obtiene la información de ubicación más reciente enviando solicitudes periódicas al backend. La página Configuración de geo-valla permite a los usuarios definir rangos de geo-valla y enviar la configuración al backend para guardarla. La página de Mensajes de Alerta muestra una lista de mensajes de alerta para la mascota y se utiliza para ver los eventos que están fuera del rango de la geo-valla. La página Historial

médico de la mascota se utiliza para gestionar los historiales médicos de la mascota, incluida la adición de nuevos historiales médicos.

Estos módulos front-end trabajan conjuntamente con los módulos back-end para proporcionar registro de usuario, inicio de sesión, gestión de la información, gestión de mascotas, monitorización de la geolocalización, visualización de mensajes de alerta y registro del historial médico a través de la interfaz de usuario para lograr un sistema integral de gestión de mascotas.

4.3.3. Diseño de la base de datos:

En el diseño detallado, también optimizaremos aún más el diseño de la base de datos teniendo en cuenta la indexación de los datos, el ajuste del rendimiento y las estrategias de copia de seguridad para garantizar un funcionamiento eficaz de la base de datos en la aplicación.

Tabla de información de usuario (User): ID de usuario, nombre de usuario, contraseña, información de contacto, etc.
Tabla de información de la mascota (Pet): ID de la mascota, nombre de la mascota, especie, foto, etc.
Tabla de información de ubicación (Ubicación): ID de ubicación, ID de mascota, longitud, latitud, fecha y hora, etc.
Tabla de información de alarma (Alarma): ID de la alarma, ID de la mascota, hora de la alarma, ubicación de la alarma, etc.
Tabla MedicalRecord (MedicalRecord): ID de registro, ID de mascota, hora de visita, información de diagnóstico, etc.

4.3.4. estructura del catálogo:

La estructura de directorios de una aplicación es importante para la organización y el mantenimiento del código. Definiremos una estructura de directorios clara para garantizar que cada componente y módulo tenga una ubicación clara.

backend: directorio backend del proyecto Django, contiene autenticación de usuarios, gestión de información, registros de localización geográfica y otros módulos.
frontend: directorio del proyecto Vue del front-end, contiene módulos como registro, inicio de sesión, modificación de información, visualización de mapas, información de alarmas, registros médicos, etc.
static: almacena los archivos estáticos del front-end, incluyendo CSS, JavaScript, imágenes y otros recursos.
templates: los archivos de plantilla HTML del proyecto front-end Vue.

4.4 Tecnología Utilizada

En este proyecto, utilizamos una variedad de tecnologías para construir las partes front-end y back-end, incluyendo Vue.js, Django, tecnologías de bases de datos, y más. En esta sección, discutiremos en detalle por qué se eligió cada tecnología, su configuración y cómo se integraron en el sistema.

4.4.1. Introducción

El propósito de este informe técnico es presentar el desarrollo de un proyecto de aplicación de seguimiento de mascotas utilizando el framework Django con el front-end Vue. El proyecto tiene como objetivo proporcionar una aplicación de seguimiento de mascotas práctica y rica en características que permita a los usuarios registrar una cuenta y añadir la información de su mascota, realizar un seguimiento de la ubicación en tiempo real de su mascota a través de un mapa, y proporcionar otras características auxiliares como el registro de alarmas, historial médico, etc.

4.4.2. Selección de tecnología

Para desarrollar este proyecto, utilizaremos las siguientes tecnologías y frameworks:

4.4.2.1 Tecnología front-end:

En esta sección se describen las tecnologías y herramientas front-end para sistemas de gestión de mascotas:

- Vue.js: como marco de front-end, Vue.js proporciona un marco de capa de vista ligero y eficiente. Permite a los desarrolladores crear interfaces frontales interactivas y con capacidad de respuesta que mejoran la experiencia del usuario.
- Axios: es una biblioteca para enviar peticiones HTTP, principalmente para la interacción de datos con el backend.Axios permite al front-end enviar peticiones al back-end para obtener datos y procesar la respuesta del back-end.
- Vue Router: Vue Router es una herramienta para la gestión de enrutamiento front-end que soporta la navegación de páginas en aplicaciones de una sola página. Esto permite a los usuarios cambiar sin problemas entre las páginas, proporcionando una mejor experiencia de navegación del usuario.
- Element UI: Como framework de interfaz de usuario para Vue, Element UI proporciona un conjunto de componentes front-end atractivos y fáciles de usar. Estos componentes pueden utilizarse para crear interfaces fáciles de usar que mejoran la calidad de la visualización del sistema y el diseño de la interfaz de usuario.

Estas tecnologías frontales trabajan en tándem para dotar al sistema de gestión de mascotas de potentes funciones frontales, como la interactividad, la obtención de datos y la navegación por rutas, al tiempo que garantizan que la interfaz sea estéticamente agradable y fácil de usar, proporcionando a los usuarios una experiencia de aplicación excepcional.

4.4.2.2 Tecnologías de backend:

Voy a cubrir la pila de tecnología back-end de un sistema de gestión de mascotas que Esta sección cubre las tecnologías y herramientas de back-end para el Sistema de Gestión de Mascotas:

Django: Django es un potente marco de back-end que proporciona una amplia gama de características de desarrollo web, incluyendo el mapeo objeto-relacional (ORM), autenticación de usuarios, enrutamiento, y más. Proporciona una infraestructura de back-end fiable para el sistema.

Django REST Framework : Este marco se utiliza para construir API RESTful para que la interacción de datos entre el front-end y el back-end sea más fácil y eficiente. Ayuda en la transferencia de datos y la comunicación entre el front-end y el back-end.

JWT (JSON Web Token): JWT es un mecanismo de seguridad utilizado para la autenticación de usuarios y el inicio de sesión. Asegura la identidad del usuario y se utiliza para validar la solicitud del usuario y proteger la confidencialidad de los datos.

Django proporciona potentes capacidades de desarrollo, mientras que el marco Django REST y JWT mejoran la transferencia de datos y la autenticación de usuarios entre el sistema y el front-end, proporcionando una base sólida para el correcto funcionamiento y la seguridad del sistema.

4.4.2.3 API de mapas:

A medida que las aplicaciones web siguen evolucionando, la integración de la funcionalidad de mapas en los proyectos front-end de Vue 3 se ha convertido en un requisito común. La API de AutoNavi Maps proporciona a los desarrolladores potentes herramientas para implementar diversas funciones relacionadas con los mapas. En este artículo, detallaremos cómo llamar a la API de AutoNavi Maps en un proyecto Vue 3 para que pueda integrar fácilmente la funcionalidad de mapas en su aplicación.

Paso 1: Regístrese para obtener una cuenta de desarrollador de Gaode

Antes de comenzar, deberá registrarse e iniciar sesión en la plataforma abierta Gaode para obtener una clave de API (Key). Esta Clave le permitirá acceder a las características de la API de AutoNavi Maps. Asegúrese de que la información de su cuenta se almacena correctamente para su uso futuro.

Paso 2: Instalar dependencias relevantes

En el proyecto Vue 3, debe instalar algunas bibliotecas de dependencias necesarias para poder cargar y utilizar la API de AutoNavi Maps. A continuación se indican algunas de las opciones de bibliotecas de dependencias más utilizadas:

Uso de vue-amap

vue-amap es un complemento diseñado específicamente para Vue.js, que se integra perfectamente con proyectos Vue 3, proporcionando componentes y directivas con estilo Vue para facilitar la integración de mapas de manera más sencilla e intuitiva. Además, vue-amap ofrece una serie de componentes fáciles de usar, como <a-map>, <a-marker>, <a-info-window>, entre otros, lo que facilita el desarrollo y la gestión de funciones de mapas. Estos componentes se pueden utilizar directamente en las plantillas de Vue, reduciendo la necesidad de manipulación del DOM. Por lo tanto, en este proyecto hemos elegido utilizar el complemento vue-amap.

Uso de @amap/amap-jsapi-loader

@amap/amap-jsapi-loader es un cargador oficial de API de JavaScript proporcionado por la plataforma de mapas de alta calidad (Amap), que permite un control más preciso sobre la carga y el uso de las API. Después de instalarlo, puedes importarlo en componentes Vue y utilizarlo para cargar la API de mapas de alta calidad (Amap).

Paso tres: Crear un contenedor de mapa

En un componente Vue, se necesita un contenedor HTML para alojar el mapa. Por lo general, usamos un elemento para crear este contenedor. Asegúrate de que el tamaño y el estilo del contenedor se ajusten a los requisitos del proyecto.

Paso cuatro: Inicializar el mapa

En un componente Vue, utiliza la clave de API y el contenedor de mapa creado anteriormente para inicializar el mapa. Por lo general, este paso se realiza en el archivo de entrada del proyecto (como main.js).

Paso cinco: Agregar funcionalidad al mapa

Una vez que se ha inicializado el mapa, puedes comenzar a utilizar las diversas funciones de la API de mapas de alta calidad (Amap), que incluyen, entre otras:

- Agregar marcadores: para mostrar puntos de interés personalizados en el mapa y mostrar información de ubicación.

- Dibujar rutas: para planificar rutas, navegación y mostrar rutas en el mapa.
- Geocodificación: para convertir información de direcciones en coordenadas de latitud y longitud, o viceversa.

4.4.3. Entorno y sistema operativo

Configurar el entorno de desarrollo y las operaciones del sistema es una parte fundamental del proyecto. Hablaremos de cómo configurar el entorno de desarrollo para asegurarnos de que las distintas tecnologías funcionan correctamente.

Las recomendaciones para el entorno de desarrollo son las siguientes:

- Sistema operativo: cualquier sistema operativo mainstream es aceptable, incluidos Windows, macOS y Linux.
- Entorno Python: Utiliza la versión Python 3.x, se recomienda Python 3.7 y superiores.
- Node.js: Asegúrate de tener instalado Node.js y npm para la gestión de dependencias del proyecto front-end.

4.4.4. Pasos de desarrollo

El desarrollo de un proyecto es gradual, y cada fase tiene tareas y objetivos específicos. Analizaremos los pasos y puntos clave de cada fase de desarrollo.

Paso 1: Inicialización del proyecto

Crear proyecto Django: Utiliza la herramienta de línea de comandos de Django para crear un proyecto Django y establecer la base de datos y otras configuraciones.

Crear proyecto Vue: Utilizar Vue CLI para crear un proyecto Vue y configurar Vue Router y otras configuraciones relacionadas.

Paso 2: Autenticación de usuario e implementación de JWT

Integrar Django REST framework en Django: Configurar Django REST framework, incluyendo enrutamiento API, serializador, etc.

Implementar funciones de registro de usuario e inicio de sesión: escribir vistas de registro de usuario e inicio de sesión en Django, utilizando el sistema de autenticación de usuario del framework Django REST.

Implementación de autenticación JWT: utilizar JWT para implementar la autenticación de usuarios para garantizar la seguridad de los extremos frontal y posterior.

Paso 3: Gestión de la información de usuario

Diseño de la tabla de información del usuario: crear la tabla de información del usuario en Django, incluyendo nombre de usuario, contraseña, información de contacto, etc.

API de información del usuario: implementar la función de añadir, eliminar, cambiar y comprobar la información del usuario, los usuarios pueden modificar su propia información.

Paso 4: Gestión de información de mascotas

Diseño de la tabla de información de mascotas: crear una tabla de información de mascotas en Django, incluyendo nombre de la mascota, especie, fotos, etc.

API de información de mascotas: implementar la función de añadir, eliminar, cambiar y comprobar la información de mascotas, los usuarios pueden añadir, modificar y eliminar su propia información de mascotas.

Paso 5: Registro de geolocalización y visualización de mapas

Diseño de la tabla de información de geolocalización: crear una tabla de información de geolocalización en Django, incluyendo el ID de la mascota, longitud, latitud, fecha y hora, etc.

API de información de geolocalización: implementar la función de registro y consulta de información de geolocalización, y registrar la información de ubicación de la mascota una vez cada dos segundos.

Visualización de mapas: en el proyecto front-end Vue, utilice la API de mapas Goldmind para mostrar la ubicación en tiempo real de la mascota y monitorizarla de acuerdo con el rango de geo-cerca establecido, con una ventana emergente que avise al usuario cuando supere el rango.

Paso 6: Información de alarmas e historial médico

Diseño de la tabla de información de alarma: crear la tabla de información de alarma en Django, incluyendo el ID de la mascota, la hora de la alarma, la ubicación de la alarma, etc.

API de información de alarma: implementar la función de registro y consulta de información de alarma, y registrar la información de alarma cuando la mascota supera el rango de la geo-valla.

Historial médico de la mascota: Crear una tabla de historial médico en Django, que incluya el ID de la mascota, la hora de consulta, la información de diagnóstico, etc. Los usuarios pueden añadir y consultar el historial médico de la mascota.

4.4.5. Resultados esperados

A través de los pasos anteriores, esperamos obtener un proyecto completo de aplicación de seguimiento de mascotas con las siguientes características:

- Registro de usuarios e inicio de sesión: implementar el registro de usuarios y el inicio de sesión para garantizar la seguridad y la privacidad de la información del usuario.
- Gestión de la información del usuario: los usuarios pueden modificar su información básica.
- Gestión de la información de la mascota: los usuarios pueden añadir, modificar y eliminar la información de su mascota, incluida la carga de fotos.
- Registro de geolocalización y visualización en mapa: seguimiento en tiempo real de la ubicación de la mascota y visualización en mapa, admite la configuración del alcance de la geocerca y la supervisión en tiempo real.
- Información de alarma e historial médico: Registra la información de alarma de la mascota y el historial médico para que los usuarios puedan consultarlos y verlos.

4.4.6. Conclusión

Utilizando el framework Django con el front-end Vue, podemos desarrollar una aplicación de seguimiento de mascotas potente y fácil de usar. El proyecto proporcionará una interfaz API a través del framework Django REST, que permite al front-end Vue interactuar con el back-end para obtener datos y comunicación en tiempo real. Al mismo tiempo, combinado con la API de mapas Gaode, para lograr el seguimiento en tiempo real de la ubicación de la mascota y la función de vigilancia geo-cerca. La autenticación de usuario se logra a través de JWT para garantizar la seguridad de los datos de usuario. Se espera que el desarrollo de este proyecto mejore en gran medida la experiencia del usuario y proporcione a los usuarios servicios de gestión y seguimiento de mascotas más convenientes.

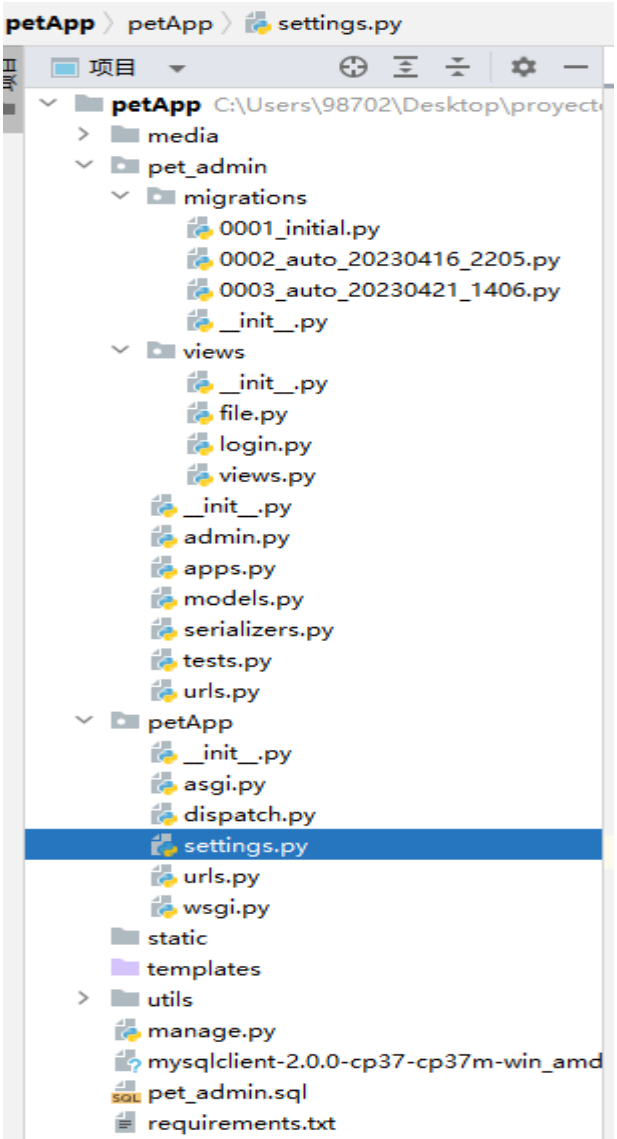
5. Implantación

Antes de la instalación, se debe garantizar que el entorno de producción cumpla con los requisitos del sistema, como la versión adecuada de Django, las bibliotecas necesarias y la configuración de la base de datos.

5.1 Instalación de backend de proyecto

Configuramos e instalamos la parte backend del sistema de gestión de mascotas. Utilizando Django como marco de backend, configuramos los entornos, bibliotecas y dependencias necesarios para garantizar que el backend estuviera listo para funcionar. En este paso se prepararon y configuraron características clave del backend, como la autenticación de usuarios, la gestión de API y el modelo de datos.

Se procederá a la instalación del backend desarrollado con Django en el servidor de producción. Se configurarán los archivos de configuración, como las variables de entorno, para asegurar un funcionamiento óptimo del sistema.



Establecer la conexión entre la app y el proyecto requiere registrar la app. Modifica la entrada de configuración INSTALLED_APPS en settings.py

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    # Solicitudes autorregistradas
    'pet_admin',
    # Configuración entre dominios
    'corsheaders',
    # DRF
    'rest_framework',
    'rest_framework_simplejwt',
]
```

Figura 5.1.1 settings.py

Registrar el nombre de la aplicación en el archivo de configuración (settings.py) de este proyecto Django, importar el rest_framework para construir la API, y también importar la funcionalidad de autenticación JWT en él.

5.1.1 URL configuration

La configuración URL (URLconf) desempeña un papel muy importante en el proyecto y se utiliza para definir la relación de mapeo entre las rutas URL y las funciones o clases de vista correspondientes.

```
urlpatterns = [
    path('admin/', admin.site.urls),
    path('api/', include('pet_admin.urls')),
    path("api/login/", LoginView.as_view(),
name="token_obtain_pair"),
    path("api/logout/", LogoutView.as_view(),
name="token_obtain_pair"),
    path("apiLogin/", ApiLogin.as_view()),
]
urlpatterns += static(settings.MEDIA_URL,
document_root=settings.MEDIA_ROOT)
```

URLconf Incluye dos partes: reglas URL y vistas

La asignación de diferentes rutas URL a las correspondientes funciones de vista o clases permite la gestión del backend, el inicio de sesión en la API, el cierre de sesión, etc., y proporciona servicios de archivos estáticos para archivos multimedia en el entorno de desarrollo. Estas normas y vistas desempeñan las siguientes funciones en el proyecto:

- Orientación de rutas: Las reglas de URL asocian diferentes rutas de URL con diferentes funciones o clases de vistas en el proyecto. Esto permite al usuario acceder a una URL específica en el navegador y luego hacer que el sistema dirija la petición a la vista apropiada para realizar la función correspondiente. Por ejemplo, la ruta "/admin" puede dirigirse a la vista de administración del backend y la ruta "/api/login" a la vista de inicio de sesión de la API.
- Activación de funciones: El propósito de una regla de URL es activar una función específica. Cuando un usuario accede a una URL específica, la función o clase de la vista asociada a ella será llamada para realizar la función específica asociada a esa ruta URL. Esto podría ser que el usuario inicie sesión, se registre, cierre sesión o realice otras acciones como acceder a recursos específicos.
- Servir archivos estáticos: la configuración de URL también puede utilizarse para servir archivos estáticos, como archivos multimedia (por ejemplo, imágenes, audio, vídeo, etc.). Cuando un navegador solicita estos archivos estáticos, las reglas de URL pueden asignar la solicitud a la ruta de archivo estático adecuada para que el navegador pueda cargar los archivos.

- Gestión de rutas: La configuración de URL también ayuda en la gestión de rutas del proyecto. Definiendo reglas URL claras, es más fácil entender y mantener la estructura de enrutamiento de un proyecto, haciendo el código más modular y fácilmente extensible.

5.1.2 Diseño de modelos

Algunos pasos necesarios para crear un modelo de datos en Django:

Importa los módulos necesarios:

En la parte superior del archivo models.py, primero importa el módulo models de Django, que es el módulo clave para crear el modelo de datos.

```
1   import hashlib
2   from django.contrib.auth.models import AbstractUser
3   from django.db import models
```

Crea la clase del modelo de datos:

Crea las clases del modelo de datos utilizando la clase models.Model como clase base. Cada atributo de la clase será un campo en la tabla de la base de datos.Django proporciona varios tipos de campo como CharField para texto corto, DateField para fechas, ForeignKey para asociaciones de clave foránea, etc. Puedes elegir el campo apropiado según tus necesidades. Puede elegir el tipo de campo apropiado según su necesidad.

```
1   import ...
5
6
7   class File(models.Model):...
16
17
18  # Create your models here.
19  class Pet(models.Model):...
31
32
33  class PetMedical(models.Model):...
42
43
44  class User(AbstractUser):...
69
70
71  class Alarm(models.Model):...
82
83
84  class Location(models.Model):...
```

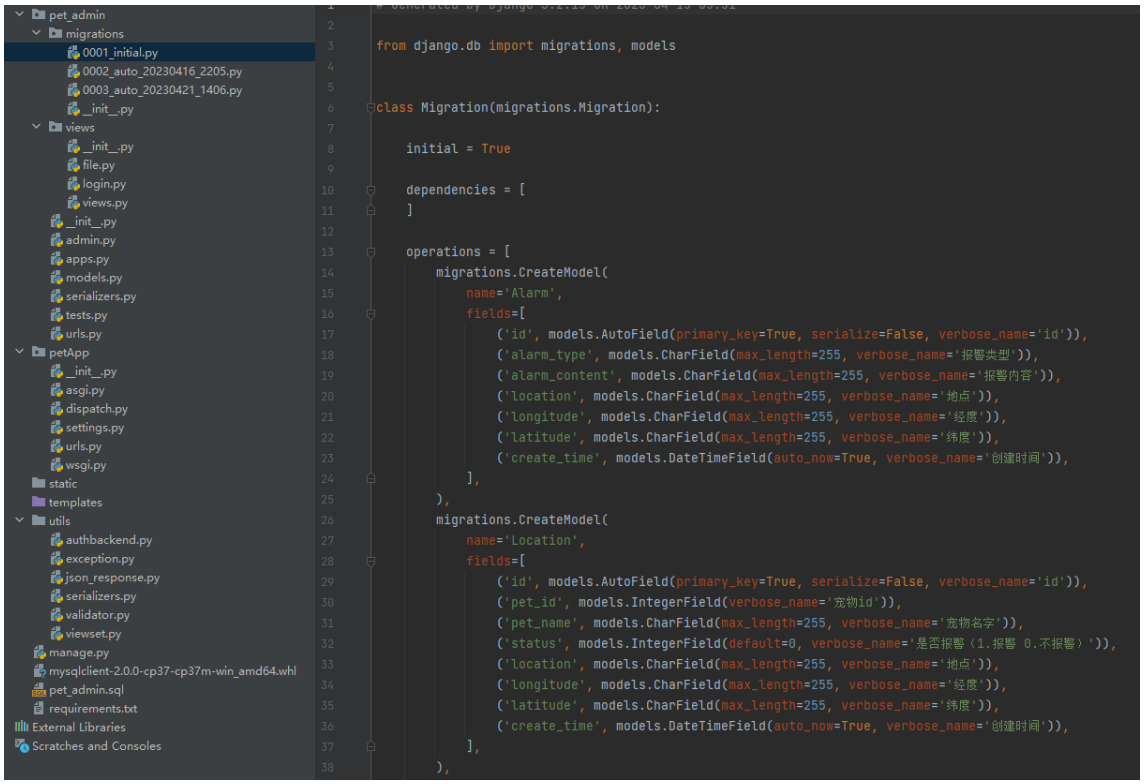
Figura 5.1.2 modelos

Relaciones de modelo:

En el modelo de datos, puede definir relaciones entre modelos. Por ejemplo, en el ejemplo anterior, se establece una relación de uno a muchos entre el modelo Mascota y el modelo Usuario, indicando que un usuario puede tener varias mascotas.

Migración de datos:

Una vez construidos los modelos, se migra la base de datos y se genera el fichero de migración.:



La migración de datos juega un papel muy importante en Django. Es el proceso de transformar la definición de un modelo de datos en una estructura de tablas de base de datos. Específicamente, el papel de la migración de datos incluye lo siguiente:

Creación de la estructura de tablas de la base de datos : Cuando defines una clase de modelo de datos, las propiedades y tipos de campo de esta clase describen la estructura de la tabla de datos, pero las tablas reales de la base de datos no se crean. Migración de datos crea automáticamente tablas de base de datos, incluidos campos, índices y restricciones, basándose en la definición del modelo.

Actualización de la estructura de las tablas de la base de datos: si realiza cambios posteriores en el modelo de datos, como añadir, eliminar o modificar campos, Migración de datos también captura estos cambios y actualiza automáticamente la estructura de las tablas de la base de datos para mantener la coherencia del modelo y de la base de datos.

Conservación de datos: la migración de datos no sólo crea o actualiza la estructura de las tablas, sino que también conserva los datos que ya existen en la medida de lo posible. Esto significa que no necesitas migrar manualmente los datos cuando el modelo cambia, Django intenta migrarlos automáticamente y hace todo lo posible para mantener la integridad de los datos.

Control de versiones: Las migraciones de datos registran el historial de cambios en el modelo de base de datos, y cada migración tiene un número de versión único. Esto permite pasar de una versión a otra y volver a un estado anterior de la estructura de la base de datos.

Colaboración entre varios desarrolladores: cuando se desarrollan proyectos con varios desarrolladores, cada uno de ellos puede crear sus propias migraciones de datos según sea necesario. Estos archivos de migración se aplican por orden de versión, lo que garantiza la coherencia de la estructura de la base de datos en el desarrollo colaborativo.

La migración de datos es una herramienta clave para gestionar la estructura de la base de datos y los cambios de datos en Django. Garantiza la sincronización entre el modelo de datos y la base de datos y permite a los desarrolladores realizar cambios en el modelo sin destruir los datos existentes. Este enfoque automatizado de la gestión de bases de datos simplifica enormemente el proceso de desarrollo y aumenta la eficiencia del desarrollo.

NOTA:

- Debe heredar la clase models.Model. Manipulación de la base de datos mediante models.py
- La base de datos es creada por nosotros; las tablas dentro y los datos en las tablas pueden ser manipulados por ORM;El marco ORM permite establecer correspondencias entre clases y tablas de datos, de modo que las tablas de datos sólo pueden manipularse a través de clases y objetos.
- En Django, crear tablas de base de datos y definir modelos de datos es una parte central del uso de ORM (Object Relational Mapping). Con ORM, puedes definir un modelo de datos usando clases de Python, y luego Django mapea automáticamente esas clases a tablas de bases de datos, permitiéndote manipular la base de datos a través de clases y objetos.

5.1.4 Serializador

Define serializadores para diferentes modelos de Django que son esenciales para convertir datos complejos en un formato adecuado para las respuestas y solicitudes de la API. Estos serializadores gestionan la validación de datos, la selección de campos y la lógica personalizada para la serialización y deserialización. Son una parte clave de la construcción de API RESTful en Django, ayudando a mantener la consistencia de los datos y el control de los datos expuestos a través de la API.

```
1   from django.contrib.auth.hashers import make_password
2   from rest_framework import serializers
3
4   from pet_admin.models import Pet, User, Alarm, Location, PetMedical
5   from utils.serializers import CustomModelSerializer
6   from utils.validator import CustomUniqueValidator
7
8
9   class PetSerializer(CustomModelSerializer):...
13
14
15  class UserSerializer(CustomModelSerializer):...
19
20
21  class PetMedicalSerializer(CustomModelSerializer):...
25
26
27  class UserCreateSerializer(CustomModelSerializer):...
64
65
66  class UserInfoUpdateSerializer(CustomModelSerializer):...
87
88
89  class AlarmSerializer(CustomModelSerializer):...
93
94
95  class LocationSerializer(CustomModelSerializer):...
99
```

Figura 5.1.3 serializer.py

- Serialiser Classes: Este código define varias clases serializadoras, cada una asociada a un modelo específico de Django.Los serializadores en el framework REST de Django se utilizan para convertir tipos de datos complejos (como instancias de modelos de Django) en tipos de datos nativos de Python (como

diccionarios) que pueden ser fácilmente renderizados en contenido JSON/XML para su uso en la API.

- PetSerializer: Serializa instancias del modelo Pet en un formato que puede ser fácilmente convertido a JSON. Especifica qué campos del modelo Pet deben incluirse en la representación serializada.
- UserSerializer: Realiza la serialización para el modelo de usuario, especificando qué campos incluir.
- PetMedicalSerializer: Serializa una instancia del modelo PetMedical, incluyendo todos los campos.
- UserCreateSerializer: Serializador para la creación de nuevas instancias de usuario. Incluye validación de nombre de usuario y opcionalmente hash de la contraseña antes de guardar el usuario.
- UserInfoUpdateSerializer: serializador para actualizar la información del usuario. Incluye validación de campos móviles y especifica qué campos se pueden actualizar.
- AlarmSerializer: serializa una instancia del modelo de alarma.
- LocationSerializer: serializa una instancia del modelo de localización.
- Meta Class: Cada clase de serializador tiene una Meta class anidada que proporciona metadatos para el serializador. Estos metadatos suelen incluir el modelo de asociación y los campos que se van a serializar.
- Validación Personalizada: Algunas clases serializadoras contienen lógica de validación personalizada, como CustomUniqueValidator para comprobar las restricciones de unicidad en campos como nombre de usuario y teléfono móvil, y lógica para hashing de contraseñas utilizando make_password al crear un nuevo usuario.

Conversión de datos de bases de datos (diccionarios, listas) en datos json, XML para el front-end.Crea el archivo serializer.py en la carpeta de la aplicación y escribe el serializador correspondiente según la clase de datos.Herencia de la clase CustomModelSerializer，Reescribir algunos de estos métodos。

5.1.5 Views

Estos conjuntos de vistas definen la lógica de solicitud y respuesta de la API para diferentes modelos de datos y se importan en las rutas API del proyecto a través de funciones relevantes en la configuración de URL. En el marco Django REST, los conjuntos de vistas proporcionan a los desarrolladores un enfoque basado en clases para las solicitudes de API, y las operaciones CRUD comunes y la funcionalidad personalizada se pueden implementar fácilmente heredando de un conjunto de vistas genérico. El papel de estos conjuntos de vistas es tomar los datos solicitados y procesarlos, consultando los datos de la base de datos o guardándolos, y utilizando un serializador para convertir los datos en formato JSON para devolverlos al cliente.

Figura 5.1.4 views

Este código define varios conjuntos de vistas del framework REST de Django, que son clases que proporcionan una forma de interactuar con los modelos de Django a través de los puntos finales de la API.

- Conjunto de vistas PetMedical:

  Este conjunto de vistas está asociado al modelo PetMedical.Define un conjunto de consulta para recuperar todas las instancias de PetMedical.PetMedicalSerializer se utiliza para serializar y deserializar las instancias de PetMedical.

- Conjunto de vistas de usuario:

  Este conjunto de vistas está asociado al modelo User.Define un conjunto de consultas para recuperar todas las instancias de User.UserSerializer se utiliza para serializar y deserializar instancias de User.Incluye dos operaciones personalizadas: user_info: una solicitud GET para recuperar información sobre el usuario autenticado actualmente.update_user_info: una petición PUT para permitir a los usuarios autenticados actualizar su información.

- register_view_set:

  Este conjunto de vistas está asociado al modelo de usuario.
  Se utiliza para el registro de usuarios y la creación de cuentas.

Se serializa utilizando UserSerializer y se deserializa utilizando UserCreateSerializer.No define una clase de permiso específica.

- AlarmViewSet:

  Este conjunto de vistas está asociado al modelo Alarm.Define un conjunto de consulta para recuperar todas las instancias de Alarm.AlarmSerializer se utiliza para la serialización y deserialización de las instancias de alarma.Incluye una operación personalizada:get_last: Una petición GET para recuperar la última instancia de Alarm.

- LocationViewSet:

  Este conjunto de vistas está asociado al modelo de localización.Define un conjunto de consultas para recuperar todas las instancias de Location.LocationSerializer se utiliza para serializar y deserializar instancias de Location.Incluye operaciones personalizadas:get_last: petición GET para recuperar la última instancia de Location.add: add: petición POST que permite crear nuevas instancias de Location con permisos abiertos (AllowAny).

Se define un conjunto de conjuntos de vistas para varios modelos, lo que permite realizar operaciones CRUD (Crear, Leer, Actualizar, Eliminar) y operaciones personalizadas en estos modelos a través de puntos finales de API RESTful. Estos conjuntos de vistas se configuran con serializadores, conjuntos de consultas y permisos para manejar interacciones de datos en aplicaciones Django, proporcionando una forma estructurada y eficiente de exponer modelos como recursos API.

5.2 Instalación de el frontend de proyecto

Para el frontend desarrollado con Vue.js, se generará el código optimizado que se servirá a través del servidor web.

Http.api.js

Todas las interfaces están definidas

```
const install = (Vue, vm) => {
    let register = (params = {}) => vm.$u.post('/api/register/', params);
    // Aquí se utiliza el parámetro incoming params, ¡sólo hay que personalizarlo todo!
    let login = (params = {}) => vm.$u.post('api/login/', params);
    // Obtención de información personal
    let getInfo = (params = {}) => vm.$u.get(`api/user/user_info/`);
    let logout = (params = {}) => vm.$u.post(`api/logout/`);
    let getPet = (params = {}) => vm.$u.get(`api/pet/get_pet_by_user/`);
    let addPet = (params = {}) => vm.$u.post(`api/pet/`, params);
    let updatePet = (params = {}) => vm.$u.put(`api/pet/${params.id}/`, params);

    let addMedicalRecords = (params = {}) => vm.$u.post(`api/medical/`, params);
    let getMedicalList = (params = {}) => vm.$u.get(`api/medical/`, params);

    let getAlarmRecords = (params = {}) => vm.$u.get(`api/alarm/`, params);
    let addAlarm = (params = {}) => vm.$u.post(`api/alarm/`, params);

    let getPetLocationNow = (params = {}) => vm.$u.get(`api/location/get_last/`, params);

    // 将各个定义的接口名称，统一放进对象挂载到vm.$u.api(因为vm就是this，也即this.$u.api)下
    //Poner el nombre de cada interfaz definida en un objeto montado bajo vm.$u.api
    //(porque vm es this, es decir, this.$u.api)
    vm.$u.api = {
        register,
        login,
        getInfo,
        getPet,
        updatePet,
        logout,
        addPet,
        addMedicalRecords,
        getMedicalList,
        getAlarmRecords,
        addAlarm,
        getPetLocationNow
    };
}
```

Creadas dos aplicaciones, una para el usuario y otra para la mascota. Estas son algunas de las páginas creadas:

- Registro e inicio de sesión
- Historial médico de mascotas
- Información del usuario
- Información sobre mascotas
- Sistema de alertas
- Se ha añadido un módulo de mapas a la aplicación.

5.3 Bases de datos

Se configurarán las bases de datos y las tablas necesarias para almacenar la información de los usuarios, mascotas, registros de alarmas, entre otros.

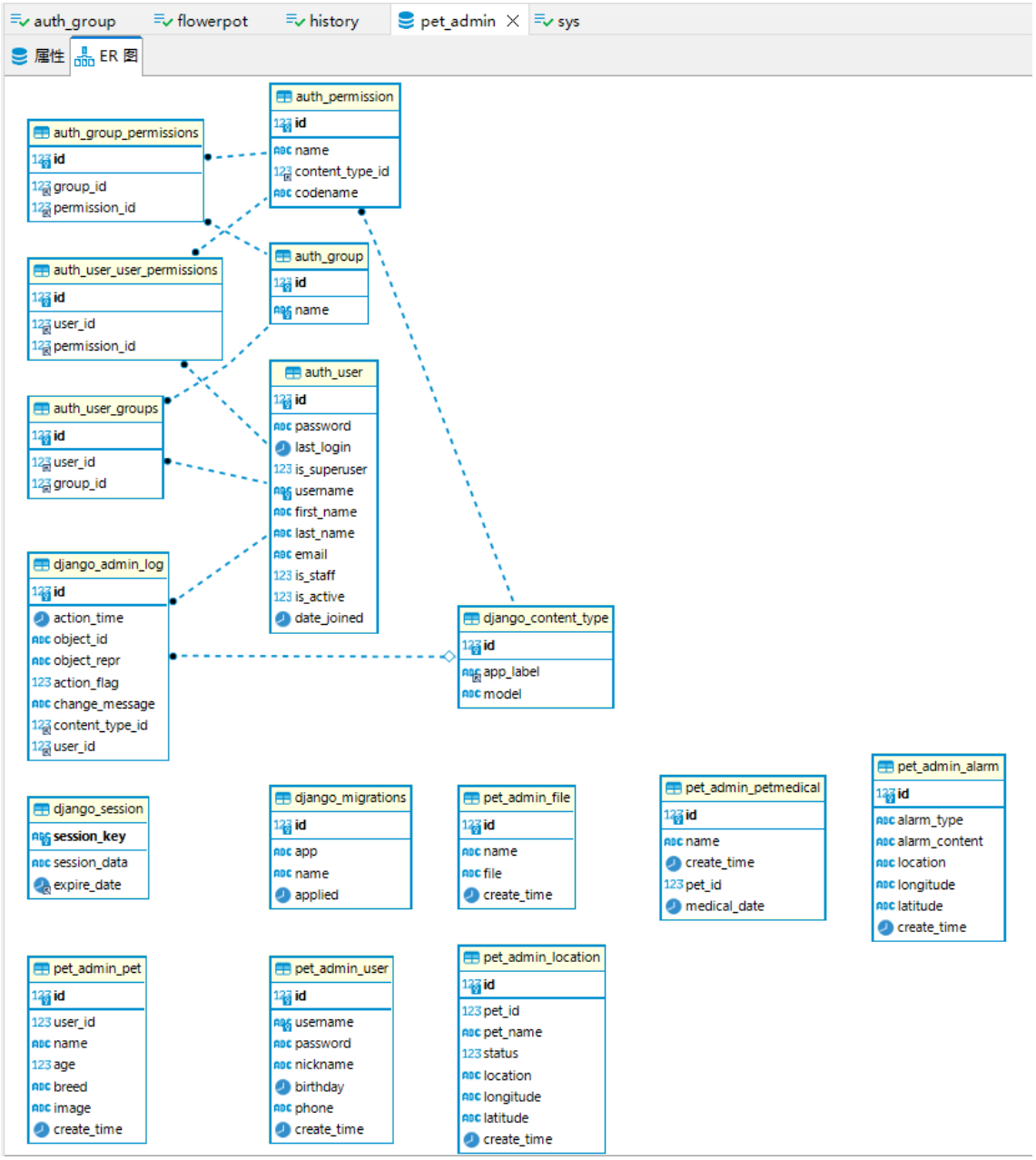


Figura 5.3.1 base de dato

Importación de una base de datos local en la configuración del proyecto back-end

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'pet_admin',
        "USER": 'root',
        "PASSWORD": '',
        "HOST": 'localhost',
        "PORT": 3306,
    }
}
```

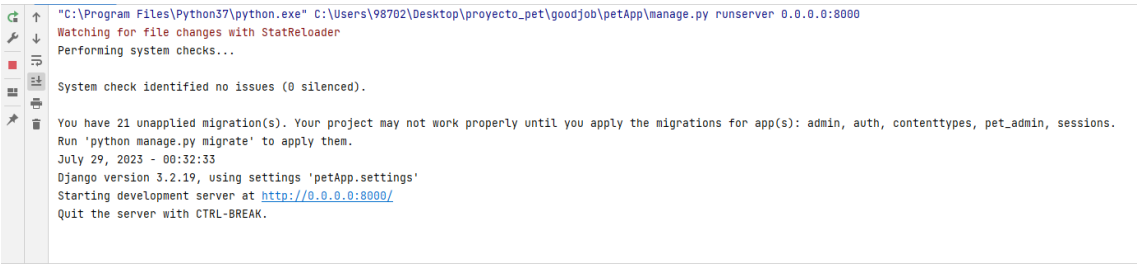
Configure los detalles de conexión de base de datos para el proyecto Django para permitir que el proyecto almacene y recupere datos en la base de datos MySQL especificada llamada "pet_admin" utilizando las credenciales de usuario y la configuración de conexión proporcionadas. Realizamos la siguiente configuración de la base de datos para el proyecto Django.

- MOTOR: Especifica el sistema de gestión de base de datos que Django debe utilizar. En este ejemplo, se establece en "django.db.backends.mysql", indicando el motor de base de datos MySQL a utilizar.
- NOMBRE: Establece el nombre de la base de datos a la que se conectará el proyecto Django. Aquí se llama "pet_admin".
- USUARIO: Especifica el usuario de la base de datos que el proyecto Django utilizará para autenticarse e interactuar con la base de datos. En este ejemplo, el usuario es "root".
- CONTRASEÑA: Establece la contraseña asociada al usuario de la base de datos. En este ejemplo, la contraseña está vacía y no se recomienda para uso en producción.
- HOST: Define el nombre de host o la dirección IP donde reside el servidor de base de datos. En este ejemplo, se establece en "localhost", indicando que el servidor de base de datos se encuentra en la misma máquina que la aplicación Django.
- PUERTO: Especifica el número de puerto en el que el servidor de base de datos escucha las conexiones entrantes. Por defecto, se utiliza el puerto 3306 de MySQL.

5.4 Pruebas

Antes del despliegue, realizamos pruebas exhaustivas. Las pruebas son un paso fundamental para garantizar la calidad y el rendimiento del sistema. Probamos la autenticación de usuarios, las interacciones de datos, el registro de geolocalización, los mensajes de alerta y los historiales médicos para garantizar que el sistema funcionara con fiabilidad en un entorno de producción.

5.4.1 En primer lugar, necesitamos iniciar el servidor de desarrollo de Django, que está localizado en http://0.0.0.0:8000/. Este servidor se utilizará para ejecutar la aplicación localmente, de forma que podamos probarla y depurarla.



5.4.2 Después de iniciar el servicio front-end, podemos ejecutar nuestra primera aplicación (para usuarios). Accediendo a la aplicación a través de un navegador nos llevará a la página de login. Aquí podemos registrarnos para obtener una cuenta e iniciar sesión.

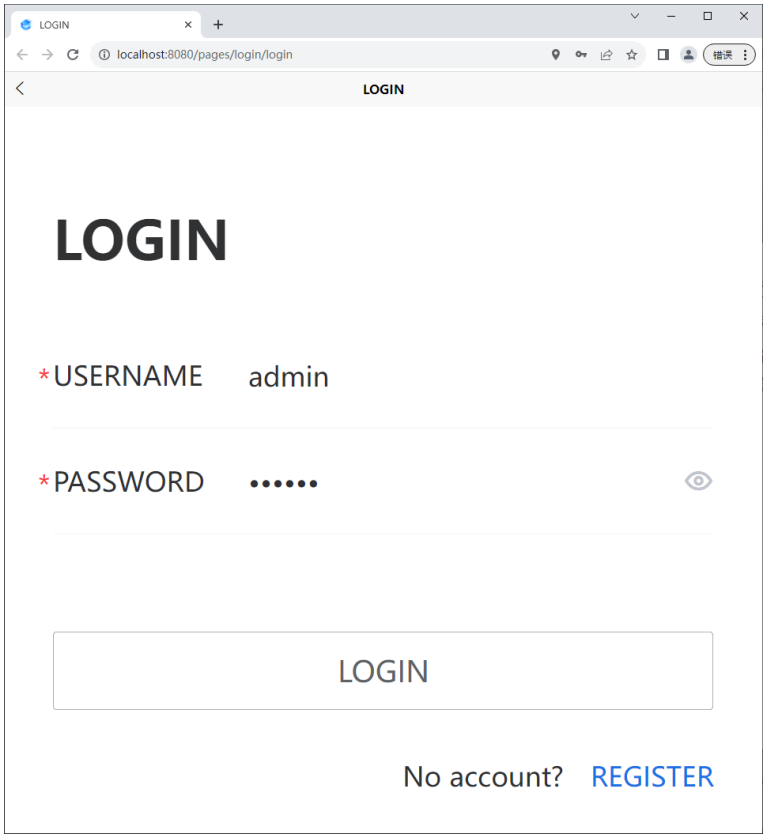


Figura 5.4.1 pantalla de acceso

5.4.3 Pulsando F12 en el navegador para utilizar la herramienta, puede ver que se envía una solicitud de inicio de sesión aquí, El front end y el back end interactúan con los datos a través de Ajax.

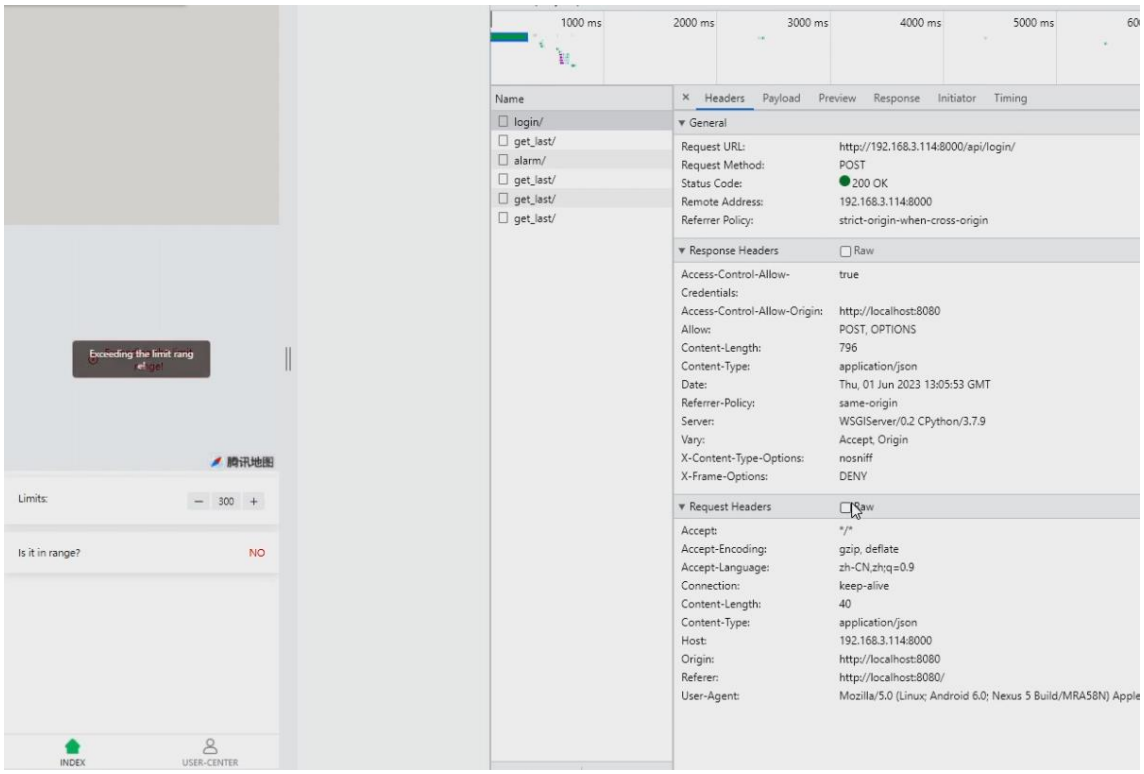


Figura 5.4.2 solicitar login

5.4.4 También podemos ver los datos en el cuerpo de la solicitud

In web development or application interface integration, a request payload is typically used to transfer structured data between a client and a server. This payload format is usually associated with a POST request in which data is sent to the server for processing.

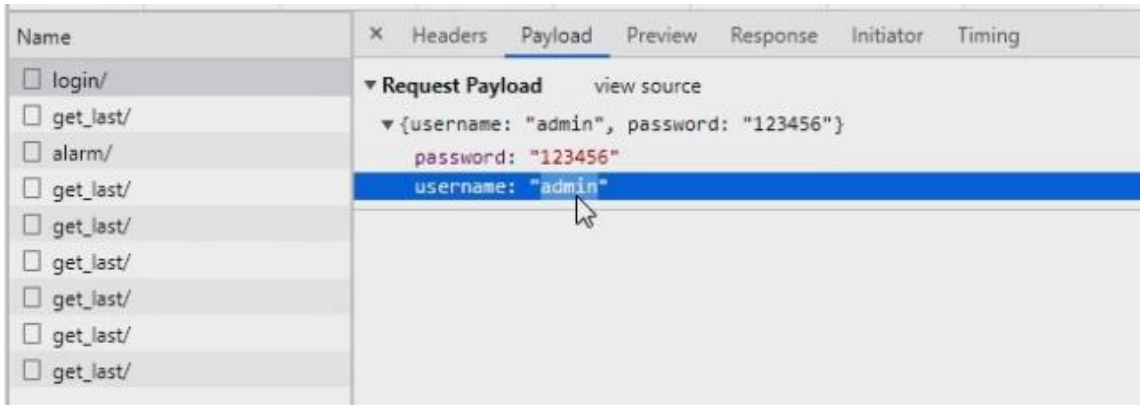


Figura 5.4.3 Datos transportados

5.4.5 Después de lanzar con éxito la aplicación de usuario, veremos que la mitad superior de la página es el módulo de mapa, que se utiliza para mostrar información sobre la ubicación geográfica del propietario y de la mascota. Este módulo de mapa es una de las características principales del sistema de seguimiento de mascotas. A continuación, podemos establecer la distancia entre la valla y el mapa. Por ejemplo, podemos establecer la distancia en 300 metros. Cuando la distancia entre la mascota y el dueño supere este umbral establecido, el sistema activará una alerta. Al mismo tiempo, la información de la alerta (latitud y longitud de la ubicación geográfica de la mascota) se almacenará para que el usuario pueda localizarla rápidamente.

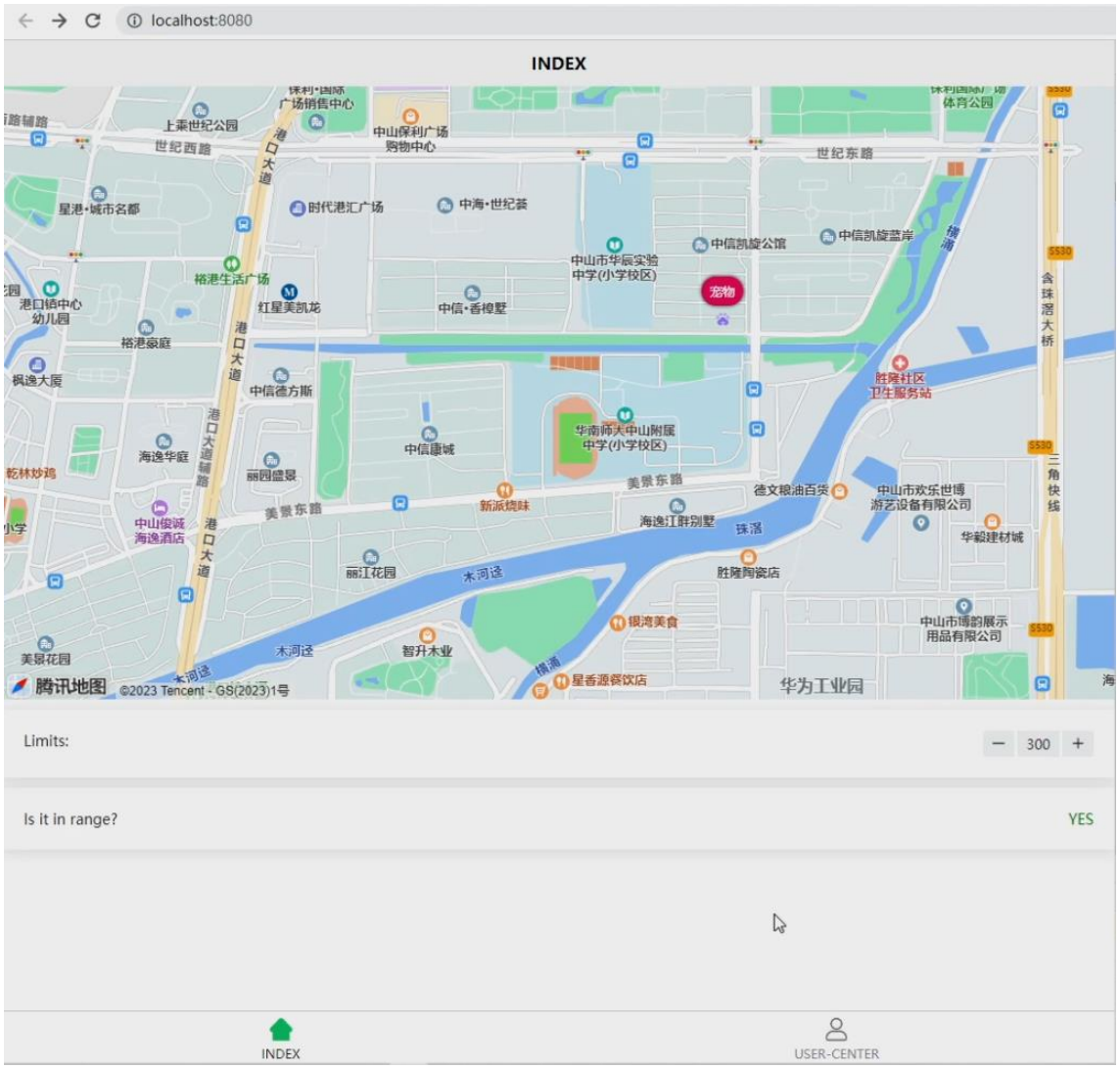


Figura 5.4.4 interfaz principal

5.4.6 A continuación, podemos entrar en el centro de usuarios. En el centro de usuarios, podemos registrarnos y modificar la información del usuario.

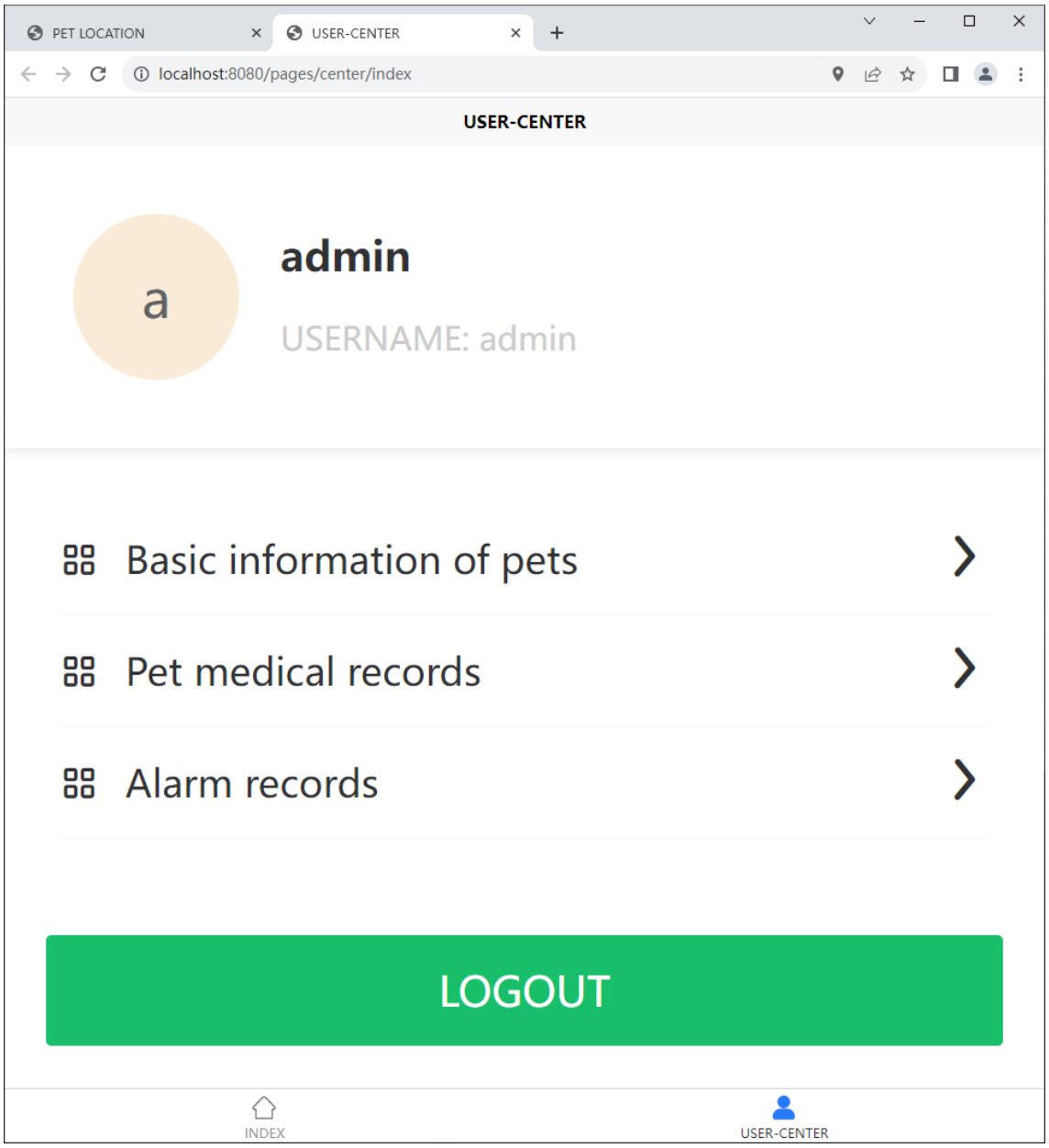


Figura 5.4.5 centro de usuarios

5.4.7 Esta página proporciona una interfaz cómoda para que los usuarios gestionen su información personal fácilmente.

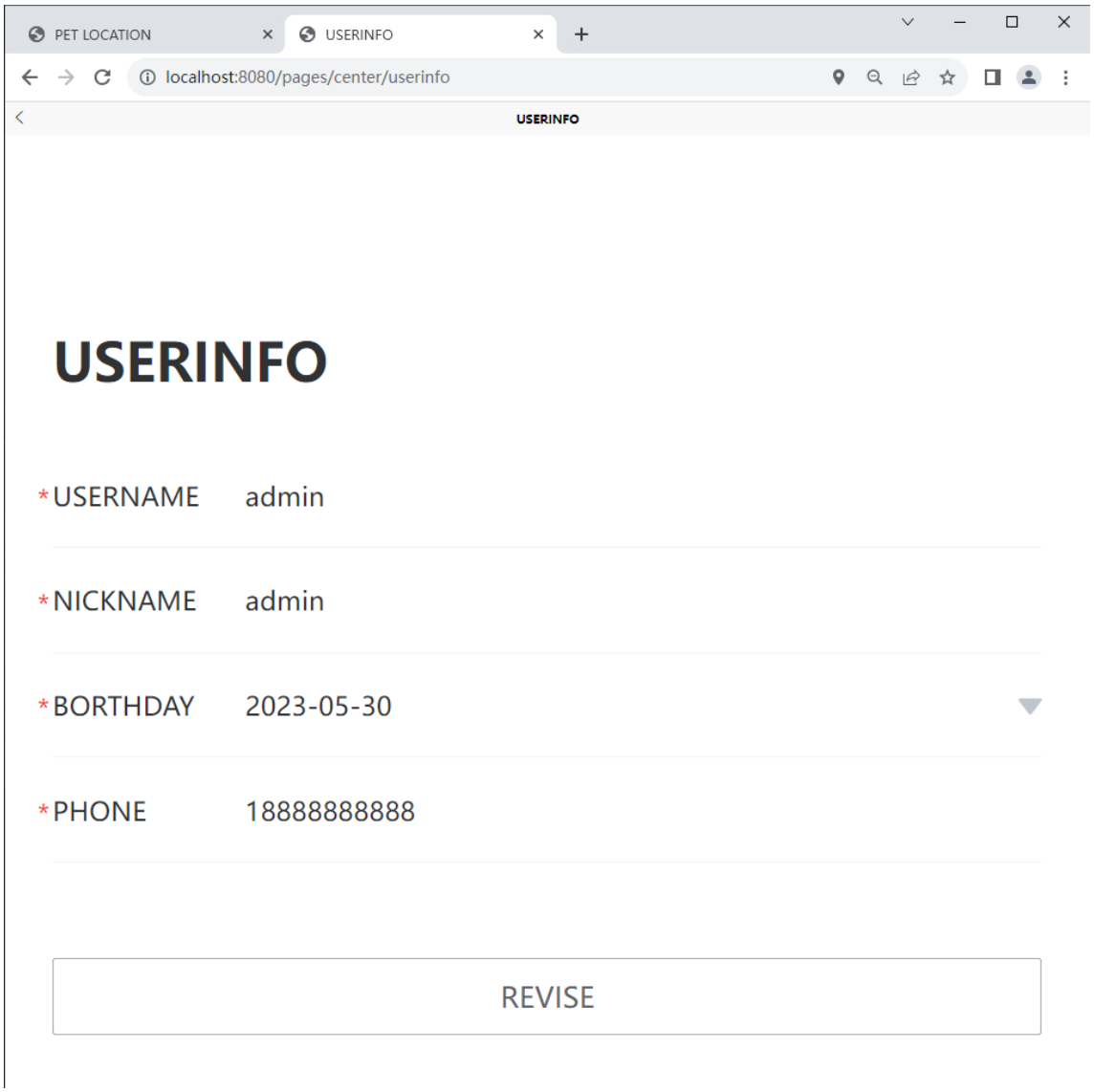


Figura 5.4.6 Interfaz de información al usuario

5.4.8 La página Información sobre mascotas es otra página funcional importante. En ella, los usuarios pueden introducir información como la foto, el nombre, la edad y la raza de la mascota. Esta información se almacenará en la base de datos y se asociará a la cuenta del usuario.

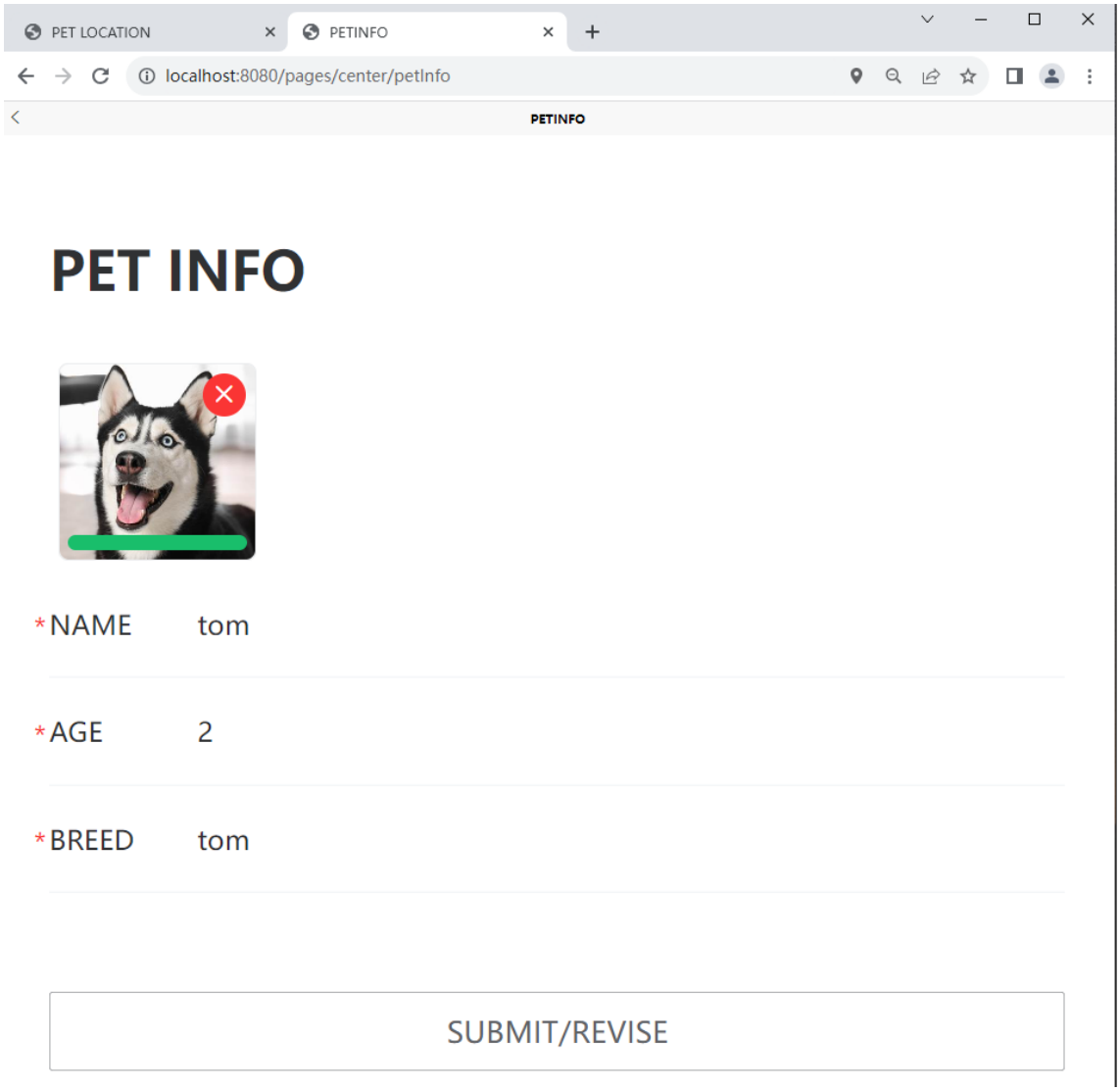


Figura 5.4.7 Página de información sobre mascotas

5.4.9 El sistema activará una alerta cuando la distancia entre la mascota y el propietario supere un umbral establecido. Esta información de alerta se almacenará en la tabla de información de alertas para que el usuario pueda ver la información de localización de la mascota en cualquier momento y actuar en consecuencia.

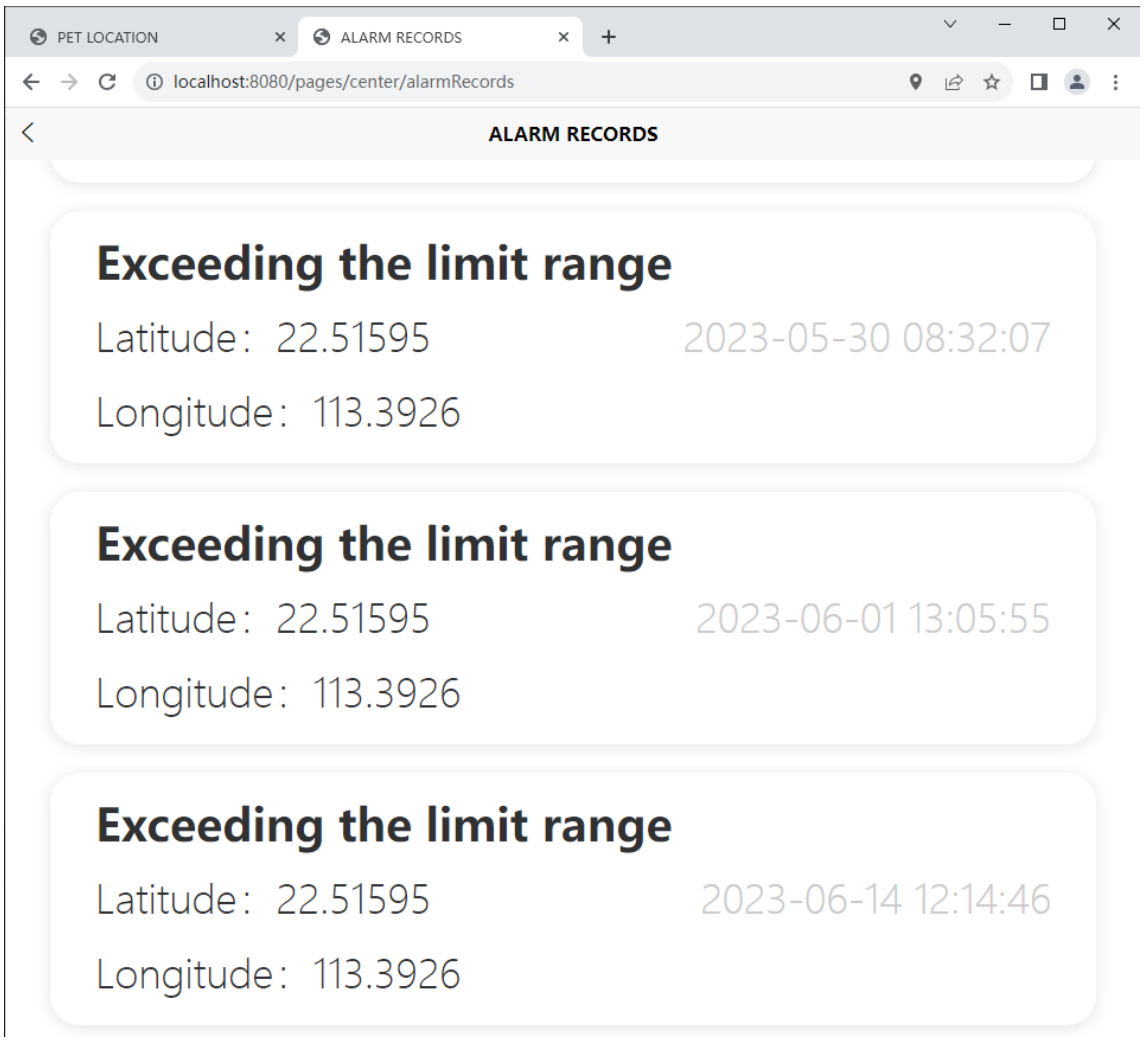


Figura 5.4.8 Página de registro de alarmas

5.4.10 La página de historiales médicos de la mascota proporciona una cómoda interfaz para que el usuario gestione los historiales médicos de la mascota. En esta página, el usuario puede añadir rutinas, por ejemplo, desparasitación u otras operaciones médicas. Estos registros médicos se almacenarán en un formulario de historial médico para que el usuario pueda ver y actualizar la información sanitaria de su mascota en cualquier momento.

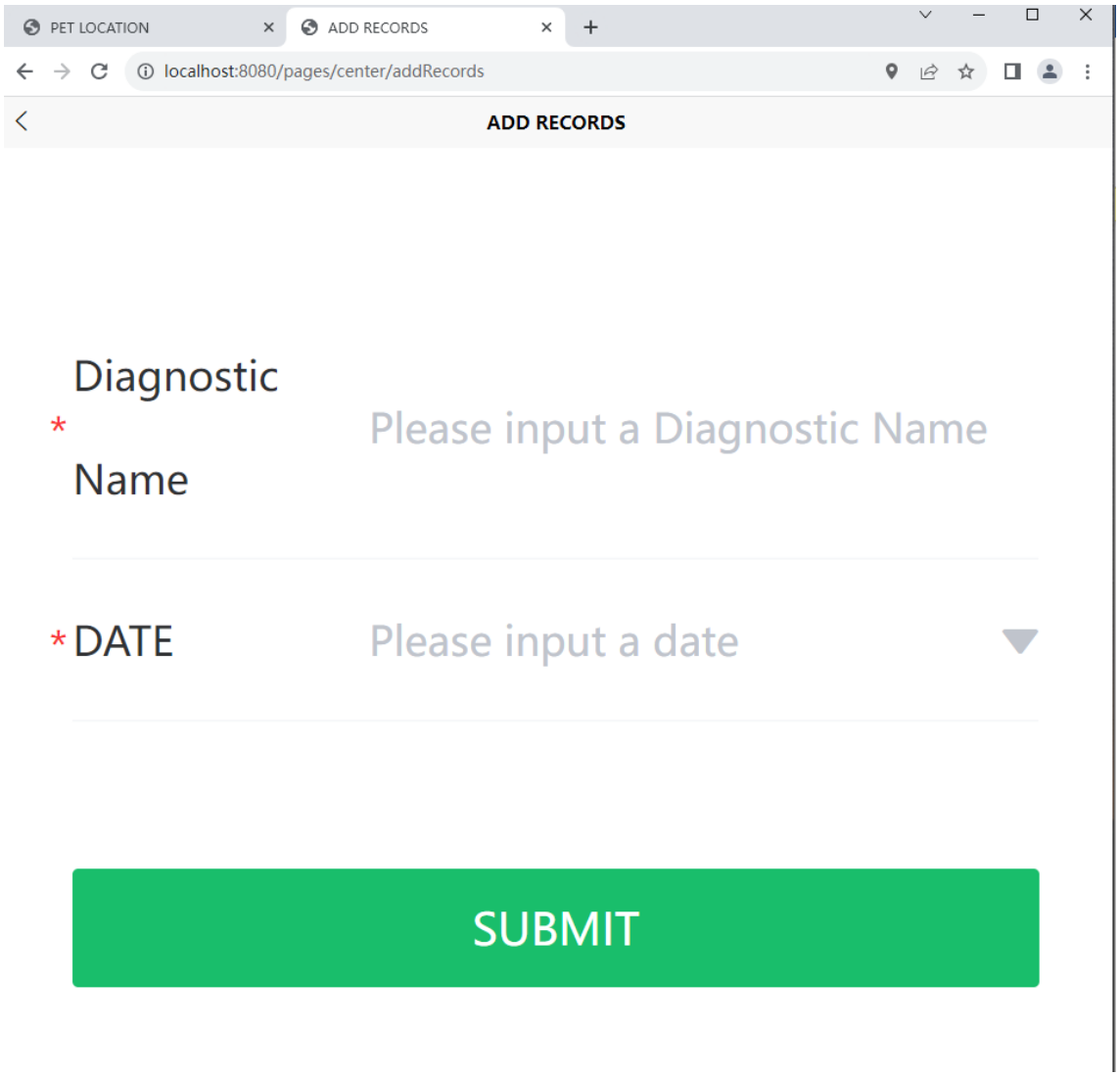


Figura 5.4.9 Información médica sobre mascotas

5.4.11 La parte superior de la interfaz principal sigue siendo la misma que la de la aplicación utilizada por los propietarios. Pero hay un cuadro de entrada adicional para el ID único de la mascota y el nombre de la mascota, esto se utiliza para vincular la mascota al propietario, aquí es una relación de muchos a uno, un propietario puede tener varios perros, y el perro sólo tiene un propietario.

A continuación, también he añadido dos cuadros de entrada para modificar manualmente la información de la ubicación geográfica de la mascota en cuanto a latitud y longitud. Es fácil ampliar la investigación más adelante. Por ejemplo, grabar la trayectoria de movimiento de la mascota y luego realizar análisis de movimiento.

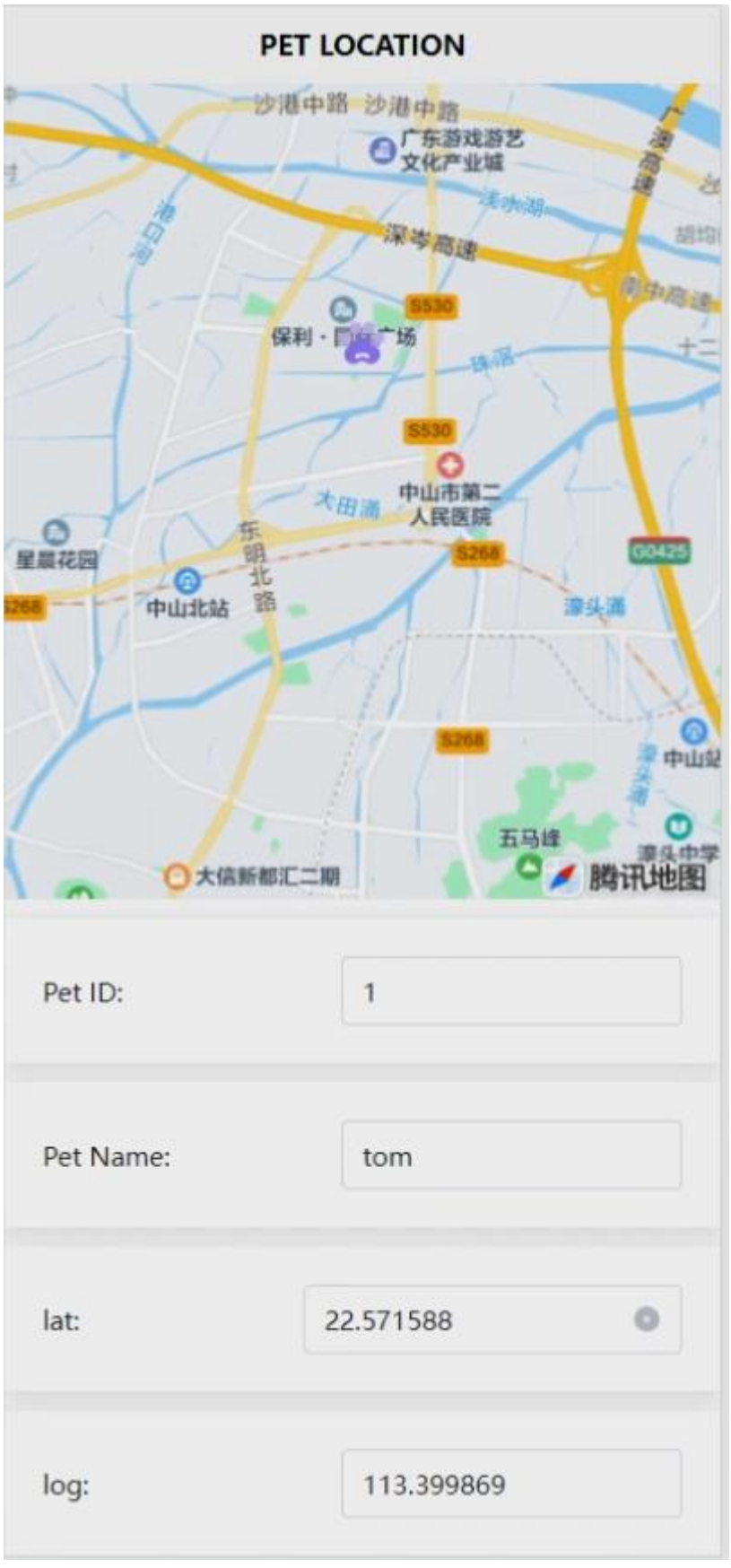


Figura 5.4.10 Interfaz de la aplicación para mascotas

5.5 Resultados

Registro de usuarios e inicio de sesión: En primer lugar, hemos probado la funcionalidad de registro de usuarios e inicio de sesión. Los usuarios pueden introducir con éxito su información en la base de datos y pueden iniciar sesión utilizando la información registrada. La aplicación utiliza el método de autenticación JWT (Json Web Token) para hacer más segura la transmisión de los datos de los usuarios. Durante las pruebas, comprobamos que las solicitudes enviadas por el front-end contenían cabeceras de solicitud de autorización, y resultó que así era.
Modificación de la información del usuario: La aplicación permite a los usuarios modificar su información básica. Hemos comprobado a fondo que todas las solicitudes de modificación se procesaban correctamente y se actualizaban en la base de datos.
Añadir/modificar información sobre mascotas: Pasamos a probar la posibilidad de añadir y modificar información sobre mascotas. Esta función permite a los usuarios cargar información fotográfica sobre sus mascotas. Probamos distintos formatos y tamaños de imagen y comprobamos que la función funcionaba correctamente.
Visualización de mapas y alertas: La función de mapas utiliza los mapas de Goldmind y permite establecer rangos de vallado. Para probar la función de alerta, establecimos varios rangos de vallado diferentes y fijamos la ubicación de la mascota tanto dentro como fuera del vallado. Cuando la ubicación de la mascota estaba fuera del alcance de la valla, aparecía una ventana emergente de alerta, como era de esperar. El sistema comprobaba la ubicación de la mascota cada dos segundos, y observamos que esta frecuencia era muy eficaz para mantener las actualizaciones en tiempo real.
Registro de alertas: También probamos la función de registro de mensajes de alerta. Todos los mensajes de alerta se añadieron correctamente a la base de datos y los usuarios pueden ver fácilmente los mensajes de alerta.
Historial médico de mascotas: los usuarios pueden añadir y consultar el historial médico de sus mascotas. Simulamos una variedad de entradas posibles, incluidos varios tipos de tiempos de tratamiento médico y tipos de medicación, y los resultados fueron todos los esperados.
Conversión de latitud, longitud y distancia: la aplicación es capaz de convertir la latitud y la longitud de dos puntos de la Tierra en distancia. Confirmamos la precisión de esta función probándola con ubicaciones y distancias conocidas de antemano.
Actualización de la ubicación de la mascota: el sistema envía información sobre la ubicación de la mascota cada dos segundos y la registra en la base de datos. Lo controlamos continuamente y comprobamos que las actualizaciones de la información eran siempre precisas.
Diseño de la base de datos: En cuanto al diseño de la base de datos, realizamos pruebas exhaustivas de todas las tablas implicadas, incluida la tabla de información del usuario, la tabla de información de la mascota, la tabla de carga de archivos, la tabla de registro de alertas, la tabla de información sobre la ubicación de la mascota y la tabla de información médica de la mascota. Todas las tablas funcionaron como se esperaba, sin pérdidas de datos ni errores.

En general, los resultados de nuestras pruebas muestran que esta aplicación de seguimiento de mascotas funciona bien en todas las funciones sin ningún problema o error importante.

6. Conclusiones

En este proyecto, hemos desarrollado con éxito una aplicación de seguimiento de mascotas que ayuda a los usuarios a gestionar y controlar sus mascotas de manera más eficaz. Esta aplicación utiliza las últimas tecnologías front-end y back-end, incluyendo el framework Django para Python, el framework Django REST, la base de datos MySQL y el framework Vue para el front-end.

Utilizamos Django y el marco Django REST para nuestro backend, lo que nos permitió desarrollar rápidamente una aplicación del lado del servidor eficiente y segura que gestiona bien el almacenamiento de datos y la lógica empresarial. Para mantener la coherencia e integridad de los datos, elegimos MySQL como sistema de base de datos.

En el front-end, utilizamos el framework Vue, que proporciona una vinculación de datos bidireccional fácil de entender y eficiente, lo que permite que la interfaz de usuario responda a las acciones del usuario con mayor rapidez. Además, utilizamos la API Goldmap, que nos proporciona información de localización precisa y en tiempo real, lo que permite a los usuarios controlar a sus mascotas en tiempo real.

En términos de funcionalidad, esta aplicación ofrece funciones básicas como el registro del usuario, el inicio de sesión y la modificación de la información del usuario, así como funciones avanzadas como la adición y modificación de la información de la mascota, la visualización del mapa, el registro de alarmas, los registros médicos de la mascota y las actualizaciones de la ubicación de la mascota. Todas estas funciones proporcionan a los usuarios una plataforma de gestión de mascotas completa y cómoda.

Sin embargo, nuestro proyecto no es perfecto. Por ejemplo, es posible que tengamos que añadir más funciones de gestión de mascotas, mejorar el diseño de nuestra interfaz de usuario y aumentar nuestra capacidad de procesamiento de datos.

Trabajo futuro

- Integración de notificaciones push: considere la posibilidad de integrar notificaciones push para avisar a los usuarios de eventos importantes relacionados con las mascotas, como alertas de vallas, recordatorios de citas médicas, etc. Esto ayudará a los usuarios a mantenerse informados incluso cuando no estén utilizando la aplicación.
- Compartir la ubicación con amigos y familiares: permite a los usuarios compartir la ubicación de su mascota con amigos y familiares. Esto es muy útil cuando el usuario necesita que alguien cuide de la mascota durante un periodo de tiempo.
- Integración en redes sociales: permite a los usuarios compartir fotos y actualizaciones de sus mascotas en las redes sociales directamente a través de la aplicación. Esto puede aumentar la interacción de los usuarios y promocionar la aplicación entre otros amantes de las mascotas.
- GeoFence personalizada: amplía la funcionalidad de GeoFence permitiendo a los usuarios definir zonas valladas personalizadas para sus mascotas. Esto proporcionará mayor flexibilidad y control a la hora de proteger a las mascotas.
- Análisis de datos y estadísticas: Añade paneles de análisis y estadísticas para permitir a los usuarios ver datos relevantes como la frecuencia de actividad de su mascota, patrones de sueño y otras métricas interesantes.

7. Referencias

[1] Plantillas Django. https://docs.djangoproject.com/zh-hans/4.2/topics/templates/

[2] Sistema de alojamiento de adopción de mascotas basado en python para django framework. https://zhuanlan.zhihu.com/p/579528890

[3] Cómo escribir una API Web en Python con la ayuda de Django. https://zhuanlan.zhihu.com/p/102269279

[4] The Best GPS Dog Collars and Pet Trackers. https://www.nytimes.com/wirecutter/reviews/best-gps-pet-trackers/

[5] Pet Tech Market Size By Product (Pet Wearables, Smart Pet Crates & Beds, Smart Pet Doors, Smart Pet Feeders & Bowls, Smart Water Dispenser, Smart Pet Fence, Smart Pet Toys), Application, End Use, Distribution Channel & Forecast, 2023 - 2032. https://www.gminsights.com/industry-analysis/pet-tech-market

[6] IoT-Enabled Pet Location Tracking. https://www.iotforall.com/use-case/pet-location-tracking-2

[7] Best GPS Pet Trackers. https://www.consumerreports.org/electronics-computers/gps-pet-trackers/best-gps-pet-trackers-a1117406199/

[8] Pet Trackers Market Size, Trends, Growth Analysis Forecast 2027. https://www.advancemarketanalytics.com/reports/6146-global-and-united-states-pet-trackers-market

[9] Pet Industry Trends, Growth & Statistics in 2022 and Beyond: Unleashing Your Ecommerce Pet Marketing Strategies. https://commonthreadco.com/blogs/coachs-corner/pet-industry-trends-growth-ecommerce-marketing

[10] Whistle GPS Pet Tracker and Activity Monitor for Pets | Whistle Store. https://www.whistle.com/

[11] "Django for Beginners" de William S. Vincent.P20-P25

[12] "Django 3 By Example" de Antonio Melé.P30-P33

[13] Documentación oficial de Django: https://docs.djangoproject.com/

[14] Real Python Django Tutorials: https://realpython.com/tutorials/django/

[15] Django Girls Tutorial: https://tutorial.djangogirls.org/

[16] "Programming GPS and OpenStreetMap Applications with Python" de Joel Lawhead

[17] "Practical Mobile Forensics" de Heather Mahalik y Rohit Tamma

[18] "Python Machine Learning" de Sebastian Raschka y Vahid Mirjalili

[19] scikit-learn https://scikit-learn.org/

[20] TensorFlow (https://www.tensorflow.org/

[21] https://www.whistle.com/blogs/all-posts

[22] https://tryfi.com/theapp

[23] https://flynn-product-design.com/portfolio/pawtrack-2018/

[24] https://getfindster.com/

[25] https://help.tractive.com/hc/en-us/categories/4414893284370-App-features