

Centro educativo

Código	Centro	Concello	Ano académico
15005397	I.E.S. Fernando Wirtz Suárez	A Coruña	2024-2025

Ciclo formativo

Código da familia profesional	Familia profesional	Código do ciclo formativo	Ciclo formativo	Grao	Réxime
IFC	Informática e comunicacións	SIFC01	Desenvolvemento de Aplicacións Web	Superior	Ordinario

Módulo profesional e unidades formativas de menor duración (*)

Código MP/UF	Nome
MP0374	Proxecto de Desenvolvemento de Aplicacións Web Equivalencia en créditos ECTS: 5.

Profesorado responsable

Tutor	Constantino García Ulla - Pablo Irimia Rega
-------	---

Alumno

Alumno	Silvia González Fernández
--------	---------------------------

Datos do Proxecto

Título	Aplicación web para la solicitud de consultas veterinarias "Consulvet"
--------	--

CONTROL DE VERSIÓNS:

Versión	Data	Observación
V0	25abr'25	
V1	30may'25	
V2	12jun'25	

Índice:

<u>1.Obxectivo</u>	3
<u>2.Descripción</u>	3
<u>3.Alcance</u>	3
<u>4.Planificación</u>	4
<u>5.Medios a emplegar</u>	4
<u>6.Presuposto</u>	4
<u>7.Título</u>	5
<u>8.Execución</u>	5

• Obxectivo

Consulvet es una aplicación web ideada para la solicitud de consultas veterinarias.

La idea de Consulvet nace de aplicaciones ya existentes para este tipo de gestiones, pertenecientes a Servicios de Salud de Humana, concretamente: al Servicio de Salud del Principado de Asturias (SESPA) y al Servicio de Salud de Galicia (SERGAS), que son los que he utilizado a título personal. Ello, aunado a mi experiencia laboral como profesional de la Veterinaria trabajando en los Servicios de Hospitalización y Cuidados Intensivos en varios Hospitales, han impulsado el desarrollo del prototipo de esta aplicación como Proyecto de Fin de Ciclo.

Consulvet persigue un **doble objetivo**:

- Por un lado, pretende ser una herramienta clara, sencilla y fácilmente utilizable a través de la cual toda persona cuidadora de uno o más animales pueda solicitar rápidamente consulta veterinaria para éste o éstos, si la precisan.
- De otra parte, pretende agilizar y aligerar la carga de trabajo en el área de Atención al Cliente de los Centros Veterinarios (sean clínicas u hospitales). Durante mi carrera profesional pude notar como mis compañeras de ese área se veían forzadas a malgastar gran parte de su jornada laboral atendiendo llamadas de cuidadores que se producían solo para agendar visitas al Centro de poca importancia: vacunaciones, revisiones generales, chequeos de salud anuales, desparasitaciones, y otros de similar índole; dejando de lado, de esta forma, tareas y gestiones de mayor peso e importancia, para las cuales tenían, al final de la jornada, mucho menos tiempo disponible del que hubieran necesitado, incurriendo ésto en retrasos o acciones mal gestionadas o ejecutadas.

Si bien es cierto que otro tipo de visitas, como puede ser un caso complicado de Medicina Interna o una intervención quirúrgica minuciosa, no se pueden beneficiar de la existencia y uso de la aplicación, ya que necesitan otro tipo de planificación de la mano de la experiencia del veterinario responsable que proceda; automatizar la gestión de todas aquellas otras de menor relevancia que sean susceptibles de autogestión por parte del cuidador beneficiaría la organización de cualquier Centro y la consecución de todas las tareas y objetivos diarios por parte del equipo veterinario.

• Descripción

Consulvet es lo que se conoce como una “SPA” (Single Page Application); es decir, una **aplicación de una sola página**. Ésto quiere decir que toda su lógica se va a ejecutar en una sola pantalla, de forma que todas las funcionalidades implementadas se recargarán exclusivamente bajo la acción del usuario. Esto es: cada vez que el usuario ejecute una acción (por ejemplo, un click en un botón) sólo se recarga el componente sobre el que el usuario ha hecho click y que, por tanto, ha recibido este evento.

Todo ello en pro de conseguir una aplicación dinámica, con tiempos de espera muy reducidos, creando una buena experiencia de navegación para el usuario.

• Uso de la aplicación

En caso de que sea un usuario registrado, éste sólo necesitará iniciar sesión en su Perfil indicando su nombre de usuario -que indicó al completar su registro-, y su contraseña personal.

The screenshot shows the login interface for the 'CONSULVET' website. At the top, there is a header bar with a logo icon, the text 'CONSULVET' in the center, and 'Login/Register' on the right. Below the header is a large input form titled 'Inicie sesión en su cuenta'. It contains three input fields: 'Nombre de usuario', 'Email', and 'Contraseña', followed by a dark blue 'Iniciar sesión' button. Below the button is a link '¿No tienes una cuenta? ¡Regístrate ahora!'. At the bottom of the form, there is a note: 'Al iniciar sesión, estás aceptando nuestros Términos y condiciones del Servicio y Política de Privacidad.' At the very bottom of the page, there is a footer with links: 'Quienes somos', 'Nuestros servicios', 'Contacta', 'Sobre nosotros', 'Gestión de citas', 'Email', 'Dirección', 'Teléfono', 'Centros asociados', 'Formulario de contacto', 'Preguntas frecuentes', 'Suscríbete a nuestra newsletter' (with a placeholder field and a 'Suscríbeme' button), and copyright information: '© 2025 Consulvet. Todos los derechos reservados.' and 'Política de Privacidad'.

En caso contrario, para poder utilizar la aplicación, el usuario debe completar un formulario de registro sencillo, donde se le solicitan algunos datos personales obligatorios sobre identidad y contacto.

 CONSULVET Login/Register

Formulario de registro

Nombre y apellidos
Fecha de nacimiento
Dirección
Teléfono
Email
Repetir email
Nombre de usuario
Contraseña
Repetir contraseña

Registrarse ahora

Al iniciar sesión, estás aceptando nuestros Términos y condiciones del Servicio y Política de Privacidad.

Quienes somos **Nuestros servicios** **Contacta**

Sobre nosotros	Gestión de citas	Email
Dirección	Teléfono	
Centros asociados	Formulario de contacto	
	Preguntas frecuentes	

Suscríbete a nuestra newsletter

Introduce tu email para estar al tanto de todas las novedades:

Suscríbeme

© 2025 Consulvet. Todos los derechos reservados. [Política de Privacidad](#) [Términos y condiciones del servicio](#)

Una vez completo el registro, y habiendo accedido ya a su Perfil, podrá completar éste añadiendo los datos de su o sus animal/es, incluyendo fotografías tanto de ellos como de sí mismo si lo desea. De la misma manera, puede editar sus datos personales y los de sus animales en cualquier momento, registrar más animales en caso de nueva adopción, y eliminar algún animal en caso, por ejemplo, de deceso.



También podrá solicitar consulta veterinaria si lo necesita, y consultar el historial de sus animales, si lo requiere.

- **Solicitud de consulta veterinaria**

La solicitud de una consulta está dividida en varios pasos, cuyo flujo de acción irá marcado por los botones “Siguiente” y “Anterior”; en dichos pasos, el usuario deberá seleccionar:

→ **Motivo de consulta y paciente**

- Si el cuidador tiene más de un animal, deberá seleccionar a cuál de ellos lleva a consulta en el desplegable que se mostrará a tales efectos.
- El motivo de consulta se seleccionará también en un desplegable que se mostrará con este fin.

The wireframe illustrates the 'Indique motivo y paciente' (Specify reason and patient) step. At the top, there's a header 'CONSULVET' and several navigation links: 'Historial', 'Solicitar consulta' (highlighted in dark grey), 'Agregar nuevo paciente', 'Editar/Eliminar paciente', 'Editar perfil', and 'Eliminar mi cuenta'. Below the header, the main form area has a title 'Indique motivo y paciente'. It contains two dropdown menus: the first is labeled '-- DESPLEGABLE CON LOS POSIBLES MOTIVOS--' and the second is labeled '-- DESPLEGABLE CON LOS ANIMALES ASOCIADOS A ESTA CUENTA --'. At the bottom of the form are two buttons: 'Cancelar' and 'Siguiente'.

Quienes somos **Nuestros servicios** **Contacta**

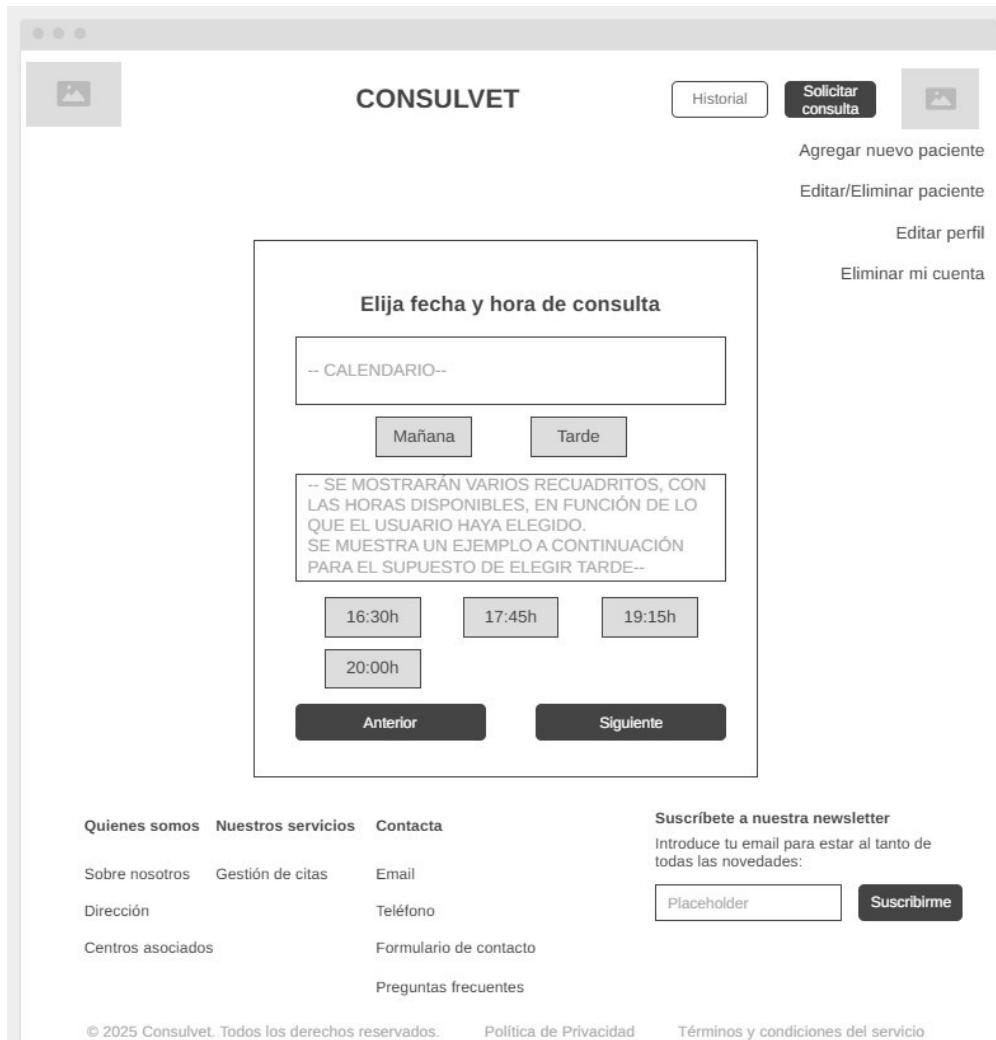
Sobre nosotros Gestión de citas Email
Dirección Teléfono
Centros asociados Formulario de contacto
Preguntas frecuentes

Suscríbete a nuestra newsletter
Introduce tu email para estar al tanto de todas las novedades:
Placeholder **Suscríbeme**

© 2025 Consulvet. Todos los derechos reservados. [Política de Privacidad](#) [Términos y condiciones del servicio](#)

→ Fecha y horario en los que desea acudir

- Para seleccionar la fecha y el horario, se muestra al usuario un calendario y dos botones de radio: “mañana” y “tarde”.



→ Centro al que acude y veterinario que le atenderá

- Para seleccionar el centro veterinario al que acude, se muestra al usuario un desplegable con todos los centros asociados disponibles, **en función del motivo de consulta que haya indicado**.

Por ejemplo, si en motivo de consulta ha indicado “vacunación”, sólo se le mostrarán al usuario aquellos centros que ofrezcan este servicio. De entre los disponibles, el usuario elegirá su centro habitual (si es uno de ellos) o deberá elegir otro centro de su gusto.

- Para seleccionar el veterinario responsable, se le presenta un desplegable que mostrará exclusivamente los veterinarios del centro seleccionado apropiados para el motivo de consulta que se ha indicado (es decir, filtrados por especialidad), y que además estén disponibles para la fecha y el horario concretos a las que se ha agendado la consulta.

Por ejemplo, si el usuario ha indicado que acude con su gato el 8 de abril de 2025 en horario de tarde para vacunación anual a la clínica Veterinaria Mi Clínica, en el desplegable se mostrarán exclusivamente los veterinarios de clínica Mi Clínica cuya especialidad sea Medicina General y que estén disponibles el 8 de abril en horario de tarde.

The screenshot shows a user interface for a veterinary clinic named 'CONSULVET'. At the top right, there are several buttons: 'Historial' (History), 'Solicitar consulta' (Request consultation) in a dark button, 'Agregar nuevo paciente' (Add new patient), 'Editar/Eliminar paciente' (Edit/Delete patient), 'Editar perfil' (Edit profile), and 'Eliminar mi cuenta' (Delete my account). Below these buttons is a large central box titled 'Seleccione su centro y veterinario' (Select your center and veterinarian). Inside this box, there is a note: '- DESPLEGABLE CON LOS CENTROS ASOCIADOS. EN FUNCIÓN DEL CENTRO, DEL MOTIVO DE CONSULTA Y DEL HORARIO QUE PREVIAMENTE HA SELECCIONADO EL USUARIO, SE MOSTRARÁN LOS VETERINARIOS DISPONIBLES-' followed by five buttons labeled 'Veterinario 1', 'Veterinario 2', 'Veterinario 3', 'Veterinario 4', and 'Veterinario 5'. At the bottom of this box are two buttons: 'Anterior' (Previous) and 'Siguiente' (Next). At the very bottom of the page, there is a footer with links: 'Quienes somos', 'Nuestros servicios', 'Contacta', 'Sobre nosotros', 'Gestión de citas', 'Email', 'Dirección', 'Teléfono', 'Centros asociados', 'Formulario de contacto', 'Preguntas frecuentes', '© 2025 Consulvet. Todos los derechos reservados.', 'Política de Privacidad', and 'Términos y condiciones del servicio'. To the right of the footer, there is a newsletter sign-up form with a placeholder for an email address and a 'Suscríbeme' (Subscribe) button.

Cuando haya completado todos los pasos, podrá proceder a confirmar su solicitud. La confirmación de la misma se realiza en dos pasos:

- En primer lugar, se le muestra un resumen de los datos que ha introducido para que pueda revisar si todo es correcto. En caso contrario, el usuario puede volver atrás pulsando el botón “Anterior” y corregir el o los error/es. Si todo es correcto, deberá pulsar el botón “Confirmar” para confirmar su solicitud.

The screenshot shows a web interface for 'CONSULVET'. At the top, there's a navigation bar with icons for user profile, 'CONSULVET', and 'Historial' (History). A prominent black button labeled 'Solicitar consulta' (Request consultation) is centered. To its right are links: 'Agregar nuevo paciente' (Add new patient), 'Editar/Eliminar paciente' (Edit/Delete patient), 'Editar perfil' (Edit profile), and 'Eliminar mi cuenta' (Delete my account). Below this is a large central box titled 'Resumen de su solicitud' (Summary of your request). Inside, it displays patient information: 'Paciente: Smokey', 'Motivo de consulta: Insuficiencia Renal', 'Fecha y hora de consulta: 25/03/2025 17:30h', 'Centro asociado: Clínica Veterinaria Nós', and 'Veterinario responsable: Silvia González'. At the bottom of this box are two buttons: 'Anterior' (Previous) and 'Confirmar' (Confirm). At the very bottom of the page, there's a footer with links: 'Quienes somos', 'Nuestros servicios', 'Contacta', 'Sobre nosotros', 'Gestión de citas', 'Email', 'Dirección', 'Teléfono', 'Centros asociados', 'Formulario de contacto', 'Preguntas frecuentes', 'Suscríbete a nuestra newsletter' (Subscribe to our newsletter), a placeholder for an email address, and a 'Suscríbeme' (Subscribe) button. The footer also includes copyright information: '© 2025 Consulvet. Todos los derechos reservados.', 'Política de Privacidad', and 'Términos y condiciones del servicio'.

- En segundo lugar, si ha pulsado “Confirmar” se le mostrará un cuadro de texto interactivo con un mensaje que le pide que confirme si está seguro de agendar la consulta. Si desea volver un paso atrás, deberá pulsar “Cancelar”; en caso de que todo sea correcto, deberá pulsar “Confirmar” y su cita quedará debidamente agendada. Se le mostrará entonces en pantalla un resumen de los datos de la consulta que ha solicitado, con la posibilidad de descargarlo como documento PDF a modo de volante de cita.



The screenshot shows a mobile application interface for CONSULVET. At the top, there is a header bar with three dots on the left, a logo in the center, and a "Solicitar consulta" button on the right. Below the header, there are several navigation options: "Historial" (white button), "Solicitar consulta" (black button), and three icons for "Añadir nuevo paciente", "Editar/Eliminar paciente", and "Editar perfil". To the right of these are "Añadir nuevo paciente", "Editar/Eliminar paciente", "Editar perfil", and "Eliminar mi cuenta". The main content area contains a message: "Su consulta se ha agendado correctamente". Below this, it says "Resumen de su solicitud:" followed by patient details: Paciente: Smokey, Motivo de consulta: Insuficiencia Renal, Fecha y hora de consulta: 25/03/2025 17:30h, Centro asociado: Clínica Veterinaria Nós, and Veterinario responsable: Silvia González. There are two buttons at the bottom: "Descargar comprobante de cita" and a large black "Finalizar" button.

CONSULVET

[Historial](#) [Solicitar consulta](#)

Añadir nuevo paciente
Editar/Eliminar paciente
Editar perfil
Eliminar mi cuenta

Su consulta se ha agendado correctamente

Resumen de su solicitud:

Paciente: Smokey
Motivo de consulta: Insuficiencia Renal
Fecha y hora de consulta: 25/03/2025 17:30h
Centro asociado: Clínica Veterinaria Nós
Veterinario responsable: Silvia González

[Descargar comprobante de cita](#)

[Finalizar](#)

Quienes somos [Nuestros servicios](#) [Contacta](#)

[Sobre nosotros](#) [Gestión de citas](#) [Email](#)
[Dirección](#) [Teléfono](#)

[Centros asociados](#) [Formulario de contacto](#)

[Preguntas frecuentes](#)

Suscríbete a nuestra newsletter
Introduce tu email para estar al tanto de todas las novedades:

Placeholder [Suscríbeme](#)

© 2025 Consulvet. Todos los derechos reservados. [Política de Privacidad](#) [Términos y condiciones del servicio](#)

- **Consulta de historiales**

El usuario puede consultar el historial de su o sus animal/es en cualquier momento. Pulsando el botón “Consultar Historial” se le mostrará el mismo; en caso de que tenga varios animales, cuando pulse el botón “Consultar Historial” se le mostrará un desplegable donde podrá seleccionar el animal en concreto del que quiere consultar la historia clínica. Esta funcionalidad añadida resulta útil desde el punto de vista del cuidador, no sólo para tener al alcance de su mano toda la información médica de sus animales, sino también a modo de consulta en situaciones en las que, por ejemplo, tenga que administrar medicaciones a su animal y no esté seguro de la pauta; entre otros.

The screenshot shows the CONSULVET mobile application. At the top, there is a navigation bar with icons for profile picture, search, and menu, followed by the text "CONSULVET". To the right of the text are three buttons: "Historial" (highlighted in red), "Solicitar consulta" (in green), and another profile picture icon. Below the navigation bar are several menu options: "Agregar nuevo paciente", "Editar/Eliminar paciente", "Anular cita", "Editar perfil", and "Eliminar mi cuenta". The main content area is titled "Historial de Smokey" and contains the following text:
Ejemplo de vista de un historial clínico, y cómo se especificaría la información del paciente por cada visita al veterinario
Fecha: 05/02/2024
Motivo de consulta: Cistitis
Diagnóstico: Insuficiencia Renal Crónica congénita
Pruebas complementarias: analítica sanguínea completa, analítica de orina completa, ecografía abdominal completa
Tratamiento: Dieta Renal, buprenorfina 1 dosis cada 8 horas durante 5 días, suplemento de potasio oral, suplemento para equilibrar el pH oral
Detalle: Revisión completa en una semana

At the bottom of this section is a "Cerrar historial" button.

[Quienes somos](#) [Nuestros servicios](#) [Contacta](#)

Sobre nosotros Gestión de citas Email

Dirección Teléfono

Centros asociados Formulario de contacto

Preguntas frecuentes

Suscríbete a nuestra newsletter

Introduce tu email para estar al tanto de todas las novedades:

Suscríbeme

© 2025 Consulvet. Todos los derechos reservados.

[Política de Privacidad.](#)

[Términos y condiciones del servicio.](#)

• Alcance

Por una cuestión de limitaciones en cuanto al tiempo disponible para el desarrollo de la aplicación, sólo se va a desarrollar en este proyecto la interfaz y funcionalidades del lado de los Cuidadores.

Esta aplicación es susceptible de ser bidireccional. Esto quiere decir que, en un futuro, sería posible desarrollar también el lado de los Centros Veterinarios para que, con una sola aplicación que simplemente muestra una interfaz y funcionalidades concretas filtrando por el tipo de usuario, pueda funcionar tanto para los Cuidadores, que pueden agendar consultas veterinarias para sus animales de forma rápida y sencilla, como para los veterinarios de los Centros a modo de agenda, pudiendo consultar la programación de la misma desde el inicio de su jornada laboral, lo que facilita la organización y gestión de la misma.

También sería posible implementar una funcionalidad donde, al agendar una consulta, se asigne un código de cita único, a través del cual el paciente sería llamado a consulta en la fecha y hora indicados; así como se gestiona en Medicina Humana, donde el paciente espera en una sala de espera y es llamado de esta forma a través de pantallas instaladas en las mismas.

Otra funcionalidad susceptible de ser implementada, es informar al usuario durante el proceso de solicitud de consulta de si su veterinario de familia está disponible en la fecha y horario indicados (para los casos en los que no sea la primera consulta). En caso de no estar disponible, se pedirá al usuario que elija otro veterinario de entre los disponibles si es una consulta que no puede esperar, o que escoga una fecha y/u horario diferentes de entre los que se le muestren disponibles para su veterinario. Los horarios disponibles que se mostrarían para su veterinario de familia, pertenecerían, de forma acotada, a la semana en curso y a la siguiente semana.

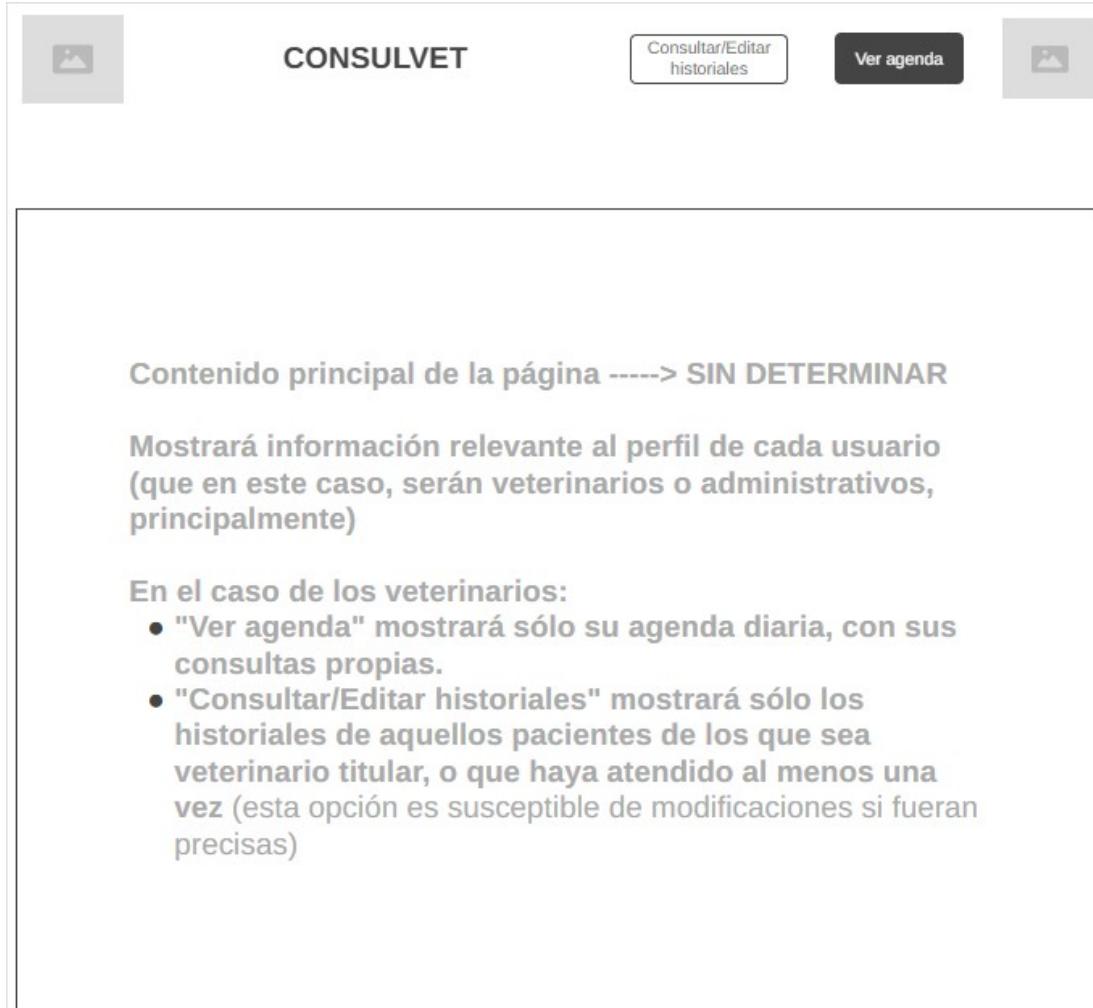
Implementar esta funcionalidad implicaría cambios en la lógica y el funcionamiento de la aplicación, ya que ahora se recogería como filtro el veterinario habitual que atiende al animal de un cuidador, y no la especialidad coherente con el motivo de consulta.

En cuanto a la gestión de la base de datos, dado que sólo se está implementando en este proyecto el lado de los Cuidadores, la tabla Usuarios va a guardar un único tipo de usuario: "Cuidadores". En caso de desarrollar el lado de los Centros Veterinarios en el futuro, esta tabla pasaría a guardar, como mínimo, dos tipos de usuarios más: "Centros" y "Veterinarios" (que sería una entidad débil en identificación, dependiente de Centros). Cada tipo de usuario tendría unos u otros permisos habilitados en relación a sus funciones y el puesto que ocupan dentro del centro. Todo ésto implica incluir en la aplicación la gestión de usuarios y permisos; y la inclusión de la figura del Administrador. Un ejemplo de permisos en función del tipo de usuario podría ser el que sigue:

- Los cuidadores cuentan con permisos para acceder a su perfil, y desde ahí solicitar consulta veterinaria y consultar el historial de su o sus animal/es.
- Los veterinarios, tendrían acceso a su agenda de consultas diarias y a los historiales de sus pacientes para poder rellenarlos, editarlos y/o actualizarlos después de cada visita.
- Los centros, por su parte, cuentan con un abanico más variado de permisos; ya que en esta categoría entrarían los empleados administrativos, servicio de limpieza, y gerencia del Centro. Los

permisos de estos tipos de usuario se aplicarían en base a sus funciones, siendo uno de ellos (un administrador designado, o el/la propio/a gerente) usuario administrador, que decide y designa qué permisos tiene qué empleado/a.

Una vista general muy esquemática y simplificada de lo que podría ser la interfaz de la aplicación web para el lado de los veterinarios y centros, podría ser la que sigue:



The placeholder area represents the main content of the application's user interface, which is currently defined as "SIN DETERMINAR".

CONSULVET

Consultar/Editar historiales

Ver agenda

Contenido principal de la página ----> SIN DETERMINAR

Mostrará información relevante al perfil de cada usuario (que en este caso, serán veterinarios o administrativos, principalmente)

En el caso de los veterinarios:

- "Ver agenda" mostrará sólo su agenda diaria, con sus consultas propias.
- "Consultar/Editar historiales" mostrará sólo los historiales de aquellos pacientes de los que sea **veterinario titular, o que haya atendido al menos una vez** (esta opción es susceptible de modificaciones si fueran precisas)

Pie de página----> SIN DETERMINAR

• Planificación

Para la planificación del proyecto se ha utilizado la plataforma JIRA.

JIRA es una herramienta de planificación de proyectos, que incluye gestión de incidencias. Es ampliamente utilizado en entornos de desarrollo de software, ya que permite a los equipos estar bien coordinados y organizados gracias a la asignación de tareas, gestionar las mismas, indicar errores si se producen y realizar seguimiento de los mismos; así como permitir a golpe de vista que todos los miembros del equipo visualicen el estado de las tareas.

Es una herramienta flexible que se adapta perfectamente al uso de metodologías ágiles, como puede ser la metodología SCRUM; permitiendo crear sprints de más o menos duración en función de la cantidad y tipo de tareas que abarque.

Otra funcionalidad ventajosa es que permite crear subtareas dentro de tareas; ésto es muy útil para agrupar tareas afines dentro de una supratarea general que las engloba, y a la que se le estipula un tiempo de consecución de objetivos. Esta supratarea se denomina "Epic"; al epic se le asigna una duración determinada, y dentro de él se pueden incluir tantas subtareas como sea necesario. Cada subtarea se puede marcar como "por hacer", "en curso", "finalizada" o "bloqueada". La opción "bloqueada" es útil para indicar que esa tarea está parada por tiempo indefinido y no se está trabajando en ella, bien porque hay otras más urgentes que requieren prioridad, bien porque dependen de la finalización de otras tareas para poder ser iniciada, entre otros.

En la planificación de este proyecto, para la organización de los epic y subtareas se ha tenido en cuenta tanto los plazos de entrega propuestos por el centro, como la coherencia en el desarrollo del mismo (por ejemplo, no sería lógico ponerse a crear el código del proyecto sin ni siquiera haber diseñado el proyecto antes, su interfaz, sus estilos, etc).

En las siguientes imágenes, se muestra el detalle de todos los epic, la duración estimada de los mismos, un resumen del objetivo que espera cumplir cada uno de ellos, y las subtareas que abarcan. En cada tarea se especifica a la derecha el estado en el que están (por hacer, en curso, finalizada o bloqueada). Las tareas finalizadas se diferencian del resto a golpe de vista, pues JIRA las presenta tachadas.

The screenshot shows the JIRA Backlog interface for the 'CONSULVET' project. The left sidebar includes links for CONSULVET (Proyecto de software), PLANIFICACIÓN (Resumen, Cronograma, Tablero, Calendario, Lista, Formularios, Metas, Incidencias, Añadir vista), DESARROLLO (Código, Páginas del proyecto, Añadir acceso rápido, Configuración d...), and a note 'Estás en un proyecto gestionado por el anónimo'. The main area displays the 'Backlog' under the 'Proyectos / CONSULVET' section. It shows two sprints: 'Sprint: Planificación 24 mar - 31 mar' (6 incidencias) and 'Sprint: Diseño 31 mar - 7 abr' (4 incidencias). Each sprint has a list of tasks with their respective states (ANÁLISIS Y PLANIFICACIÓN, DISEÑO, BLOQUEADO, EN CURSO, FINALIZADA) and due dates. For example, in the first sprint, tasks like 'SCRUM-4 Requisitos' and 'SCRUM-5 Objetivos' are marked as 'FINALIZADA', while others like 'SCRUM-6 Sprints' and 'SCRUM-7 Presupuesto' are 'EN CURSO'.

Jira Tu trabajo Proyectos Filtros Paneles Equipos Planes Aplicaciones Crear Versión de prueba de Premium Buscar Insights Ver configuración

CONSULVET Proyecto de software

PLANIFICACIÓN

- Resumen
- Cronograma
- Backlog**
- Tablero
- Calendario
- Lista
- Formularios
- Metas
- Incidencias
- Añadir vista

DESARROLLO

- Código
- Páginas del pro...
- Añadir acceso rápido
- Configuración d...

Backlog

Proyectos / CONSULVET Backlog

Buscar SG Epic

Sprint4: BackEnd 28 abr – 19 may (4 incidencias)

Desarrollo de la lógica en el lado Servidor, incluyendo la conexión a la base de datos y la gestión de acciones sobre las tablas

SCRUM-16 MVC	BACKEND	TAREAS POR HACER	SG
SCRUM-17 Conexión a BD	BACKEND	TAREAS POR HACER	SG
SCRUM-18 Código	BACKEND	TAREAS POR HACER	SG
SCRUM-19 Cookies	BACKEND	TAREAS POR HACER	SG

+ Crear incidencia

Sprint3: FrontEnd 7 abr – 28 abr (5 incidencias)

Desarrollo de la lógica del proyecto en el Lado Cliente, incluyendo la vista de la interfaz

SCRUM-21 Angular CLI	FRONTEND	FINALIZADA	SG
SCRUM-22 Componentes	FRONTEND	EN CURSO	SG
SCRUM-23 Código	FRONTEND	EN CURSO	SG
SCRUM-24 Diseño	FRONTEND	EN CURSO	SG
SCRUM-25 Conexión	FRONTEND	BLOQUEADO	SG

+ Crear incidencia

Jira Tu trabajo Proyectos Filtros Paneles Equipos Planes Aplicaciones Crear Versión de prueba de Premium Buscar Insights Ver configuración

CONSULVET Proyecto de software

PLANIFICACIÓN

- Resumen
- Cronograma
- Backlog**
- Tablero
- Calendario
- Lista
- Formularios
- Metas
- Incidencias
- Añadir vista

DESARROLLO

- Código
- Páginas del pro...
- Añadir acceso rápido
- Configuración d...

Backlog

Proyectos / CONSULVET Backlog

Buscar SG Epic

Sprint5: Integración 19 may – 2 jun (2 incidencias)

Integración del Front con el Back, comprobación de que todo funciona correctamente y que todo se comunica correctamente entre sí

SCRUM-27 Pruebas	INTEGRACIÓN Y PRUE...	BLOQUEADO	SG
SCRUM-28 Errores	INTEGRACIÓN Y PRUE...	BLOQUEADO	SG

+ Crear incidencia

Sprint6: Documentación Final 2 jun – 9 jun (3 incidencias)

Documentación final del proyecto, ultimando los últimos detalles del proyecto, incluyendo las diapositivas resumen del mismo que servirán de apoyo durante la defensa, y el README

SCRUM-30 Memoria	DOCUMENTACIÓN	EN CURSO	SG
SCRUM-31 Readme	DOCUMENTACIÓN	TAREAS POR HACER	SG
SCRUM-32 Presentación	DOCUMENTACIÓN	FINALIZADA	SG

+ Crear incidencia

Backlog (0 incidencias)

Tu backlog está vacío.

En el apartado “Tablero” se puede ver esto mismo, pero de un modo más ordenado y visual, organizadas las tareas por columnas en función de su estado, como si fuesen “post-its en una corchera”.

Jira Tu trabajo Proyectos Filtros Paneles Equipos Planes Aplicaciones Crear Quedan 27 días Buscar Insights Ver configuración

CONSULVET Proyecto de software

PLANIFICACIÓN

- Resumen
- Cronograma
- Backlog
- Tablero**
- Calendario
- Lista
- Formularios
- Metas
- Incidencias
- Añadir vista

DESARROLLO

- Código
- Páginas del pro...
- Añadir acceso rápido
- Configuración del ...
- Incidencias archivadas...

Backlog

Proyectos / CONSULVET All sprints

Buscar SG Epic Sprint

AGRUPAR POR Persona asignada Insights Ver configuración

Por hacer 5 EN CURSO 7 BLOQUEADO 4 LISTO 8

+ Silvia González (24 incidencias)

MVC BACKEND SCRUM-16 SG	Sprints ANÁLISIS Y PLANIFICACIÓN SCRUM-6 SG	Estilos DISÑO SCRUM-13 SG	Requisitos ANÁLISIS Y PLANIFICACIÓN SCRUM-4 SG
Conexión a BD BACKEND SCRUM-17 SG	Repository ANÁLISIS Y PLANIFICACIÓN SCRUM-8 SG	Conexión FRONTEND SCRUM-25 SG	Objetivos ANÁLISIS Y PLANIFICACIÓN SCRUM-5 SG
Código BACKEND SCRUM-18 SG	Logotipo DISÑO SCRUM-14 SG	Pruebas INTEGRACIÓN Y PRUEBAS SCRUM-27 SG	Presupuesto ANÁLISIS Y PLANIFICACIÓN SCRUM-7 SG
Cookies BACKEND SCRUM-19 SG	Componentes FRONTEND SCRUM-22 SG	Errores INTEGRACIÓN Y PRUEBAS SCRUM-28 SG	Herramientas ANÁLISIS Y PLANIFICACIÓN SCRUM-9 SG
Readme DOCUMENTACIÓN SCRUM-31 SG	Código FRONTEND SCRUM-23 SG	Diseño FRONTEND SCRUM-24 SG	Base de datos DISÑO SCRUM-11 SG
			Prototipo DISÑO SCRUM-12 SG
			Angular CLI

+ Crear incidencia

Quickstart

• **Medios a emplegar**

Para la realización de este proyecto, se van a emplear diferentes recursos tecnológicos y de desarrollo, los cuales se detallan a continuación.

- Un equipo informático completo (ordenador propiamente dicho, monitor, ratón y teclado).

- Plataforma JIRA para la planificación del proyecto en base a sprints.

- Herramientas de diseño:

- Para el diseño de la base de datos, se va a utilizar Dia.net.

- Para el diseño de la interfaz web se va a utilizar Wireframe.cc

- Para el diseño y vectorización del logotipo se va a utilizar Inkscape

- Editores de código:

- Para el desarrollo del código, se va a utilizar Visual Studio Code.

- Lenguajes de programación:

- Para codificar en el lado Cliente, se va a utilizar Angular, que emplea como lenguaje Typescript y estructura su jerarquía de código en componentes reutilizables que se comunican entre sí en pro de conseguir lo que se denominan “Single Page Application”, es decir: aplicaciones de una sola página. Ofrece componentes y temas predeterminados para facilitar la creación de páginas web dinámicas, sencillas, modernas, accesibles y que fomentan una buena experiencia para el usuario (<https://v14.material.angular.io/components/categories>).

- Para codificar en el lado Servidor, se va a utilizar Laravel, que trabaja en lenguaje PHP.

- Virtualización del proyecto:

- Por determinar -

• Presupuesto

Para la elaboración de este presupuesto, se ha tenido en cuenta todo lo que sigue:

MANO DE OBRA	
Duración estimada	3 meses
Dedicación semanal (en horas)	20
Total de horas estimadas	240
Tarifa estimada para desarrolladores junior	7'5€ / hora
Total (en euros)	1800

SOFTWARE UTILIZADO			
Herramienta	Licencia	Finalidad	Coste (en euros)
Dia.net	Licencia de uso gratuita	Diseño de la base de datos	0
Wireframe.cc	Licencia de uso gratuita	Diseño del prototipo	0
Inkscape	Licencia de uso gratuita	Diseño, creación y vectorización del logotipo	0
Visual Studio Code	Licencia de uso gratuita	Editor de código para el desarrollo del mismo	0
Total (en euros)			0

COSTES FIJOS DERIVADOS DEL DESEMPEÑO DE LA ACTIVIDAD		
Recurso	Consumo estimado por 3 meses de actividad	Coste (en euros)
Electricidad	30kWh	10
Internet (tarifa de 30€ mensuales)	Se estima un 30% del uso diario del mismo dedicado a trabajar en el proyecto	27
Total (en euros)		37

COSTES INDIRECTOS POR MÍNIMA INFRAESTRUCTURA (3% del total por mano de obra)	
Uso de equipo personal, mantenimiento y depreciación del mismo	216€

PLANIFICACIÓN DEL PROYECTO Y ELABORACIÓN DE DOCUMENTACIÓN		
Tareas a desarrollar	Tiempo total estimado (en horas)	Coste total (en euros)
- Planificación de sprints - Planificación y creación de Guía de Estilos - Redacción de Memoria - Creación del repositorio y actualizaciones periódicas de Readme	20	600

Coste total estimado para la realización del proyecto	2653€
---	--------------

- **Título**

Aplicación web para la solicitud de consultas veterinarias "Consulvet"

• Execución

En este apartado se va a profundizar en el funcionamiento y la lógica de la aplicación, apoyando cada explicación con la imagen o imágenes pertinentes en cuanto a código y su ejecución.

Se va a dividir la información en los dos entornos de desarrollo en los que se va a trabajar: el entorno Servidor, y el entorno Cliente.

● DESARROLLO DE LA APLICACIÓN EN EL ENTORNO SERVIDOR

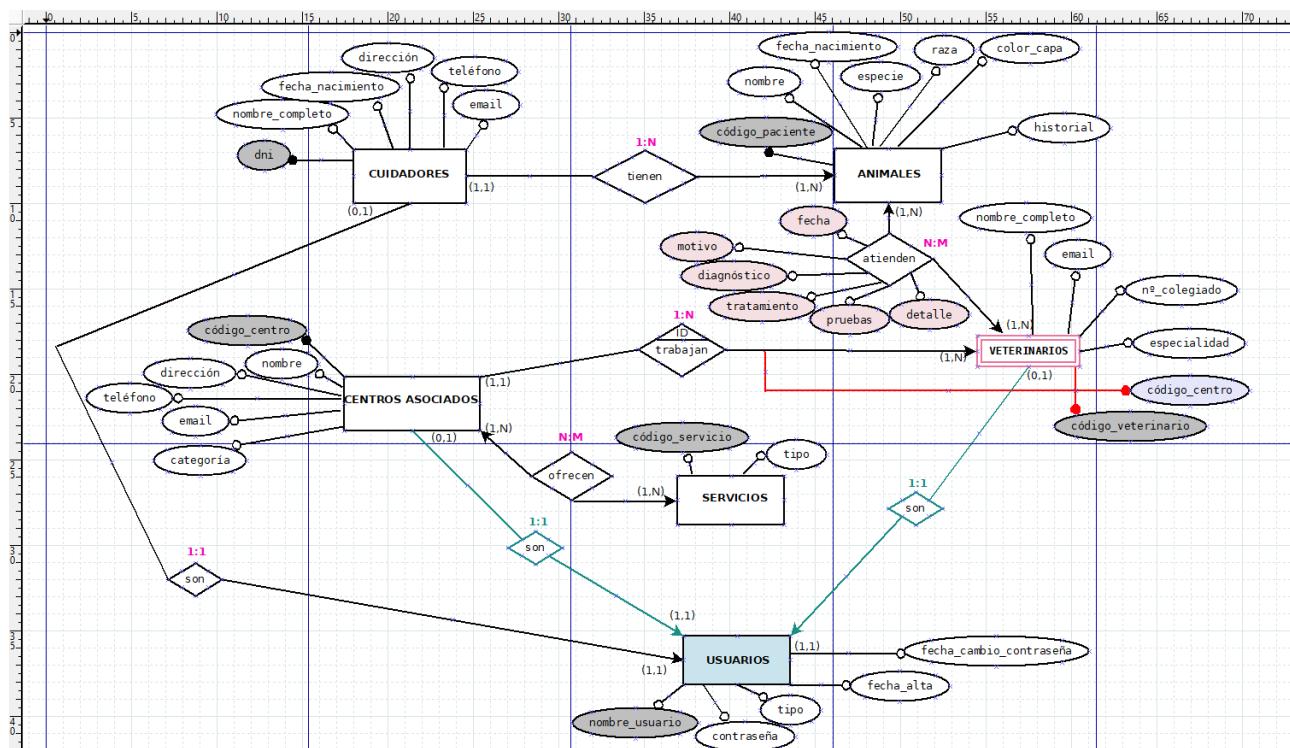
○ Diseño de la base de datos

El primer paso para poder trabajar con nuestra aplicación, es tener una base de datos desde la cual extraer la información pertinente en cada momento (consultas), y en la cual añadir nueva información (registros de nuevos cuidadores y nuevos pacientes, edición del perfil del cuidador, eliminación de animales, etc).

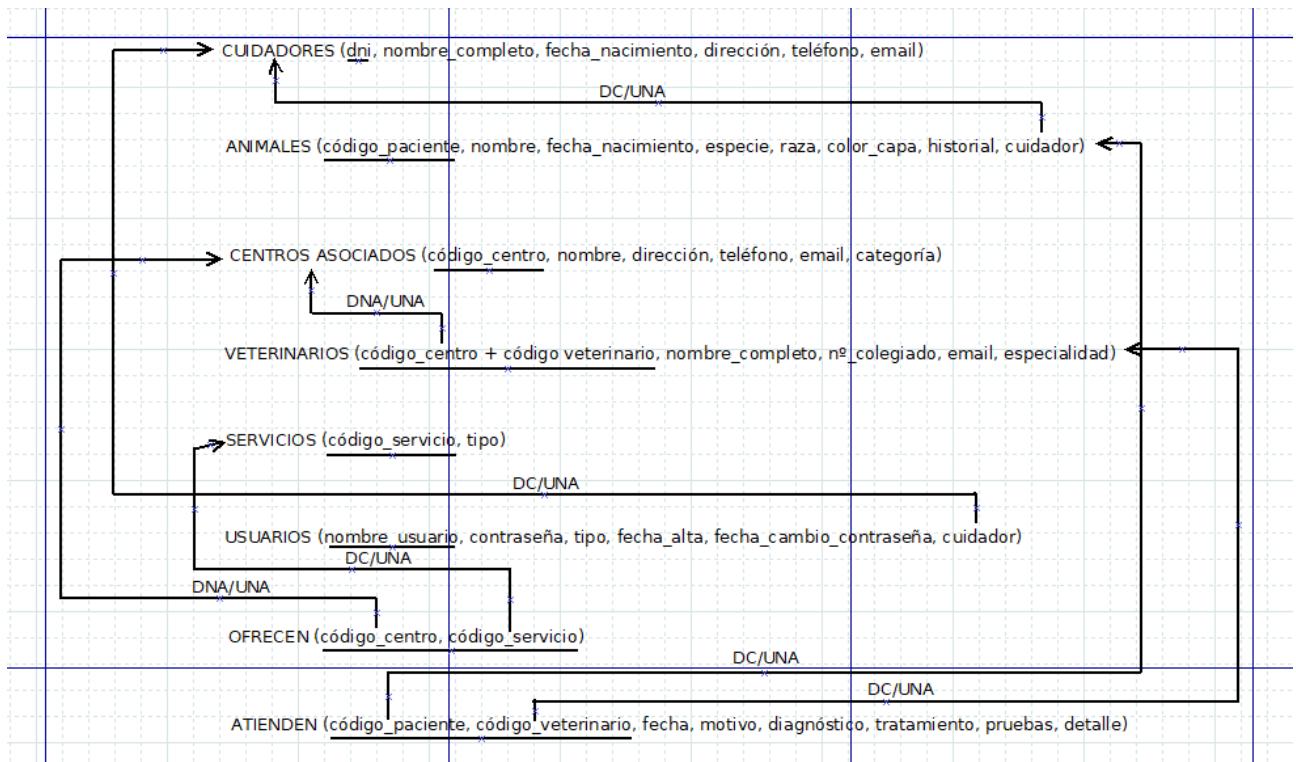
Antes de crear la base de datos, es fundamental tener un correcto diseño conceptual y lógico de la misma, para poder ver y comprender toda su estructura y relaciones a golpe de vista, y así saber cómo están estructurados los datos que después se van a manejar.

Las imágenes que siguen ilustran la estructura de la base de datos de Consulvet, tanto a nivel conceptual como a nivel lógico:

ESQUEMA CONCEPTUAL:



ESQUEMA LÓGICO:



Detalle de las restricciones en cuanto a borrado y actualización de datos en la base de datos:

*NOTA: algunas de estas restricciones están pensadas para la parte que compete al proyecto, que es el desarrollo de la parte Cuidadores. En caso de ampliación del proyecto (lado de los centros veterinarios, gestión de usuarios y permisos en función del tipo de usuario, etc.) algunas de éstas podrían variar para ser más coherentes con el mundo real.

- Si el cuidador se elimina, se eliminan completamente los animales que tiene a cargo; el DNI no se puede modificar ya que es un identificador único de persona.
- Si se elimina un animal, todo su historial y sus datos quedan eliminados; el código de paciente asignado es único y no se puede actualizar.
- Un veterinario no se puede eliminar si tiene consultas asociadas; el código de veterinario es único y por tanto no se puede actualizar.
- Si un centro se borra, sólo se eliminan todos sus datos en caso de que no tengan datos sensibles asociados (como el historial de los pacientes); el código del centro es único y por tanto, inmutable.
- Si un centro es eliminado, se eliminan también los servicios que ofrece; los códigos únicos asociados a cada centro no se pueden modificar.
- Si se elimina un usuario, todos sus datos son eliminados; su identificador es único y no se puede modificar.

○ Creación de la base de datos y desarrollo del backend

Una vez la base de datos está diseñada, el siguiente paso es crearla para poder empezar a trabajar con ella y a desarrollar el backend de la aplicación.

Para el desarrollo del lado Servidor - como se ha indicado en el apartado *Medios a Emplear* – se va a utilizar *Laravel*.

► Un poco de contexto

Laravel es un framework de código abierto para el desarrollo de aplicaciones y servicios web con lenguaje PHP. Trabaja por sistema de paquetes, y es de tipo MVC (modelo vista-controlador), de manera que implementa de forma predefinida gran parte del código necesario para hacer funcionar la aplicación, facilitando la vida del programador y agilizando considerablemente el proceso de desarrollo. Un ejemplo de ello, es la instanciación de clases y métodos reutilizables a lo largo de la aplicación, que Laravel implementa por nosotros.

Laravel trabaja con el ORM (Object-Relational Mapping) *Eloquent*, el cual es muy intuitivo para escribir consultas sobre objetos en PHP.

Su sistema de comandos *Artisan* ofrece grandes facilidades a los programadores para crear de forma rápida y cómoda controladores, entidades, o incluso actualizar la base de datos, entre otros.

Además de sus librerías propias, Laravel se apoya en Symfony, que evoluciona y progresá favorablemente gracias a la gran comunidad que tiene detrás.

Permite actualizar y migrar la base de datos cuando se producen cambios, sin necesidad de tener que borrarla y volverla a crear una y otra vez. Ésto es realmente potente, porque minimiza drásticamente la posibilidad de perder datos. Además, gracias a su “Schema Builder”, no es necesario utilizar SQL, sino que cuenta directamente con un sistema propio e intuitivo en PHP que facilita considerablemente el proceso.

► Preparando Laravel

1) Instalación de Laravel:

La instalación de Laravel se realiza por terminal utilizando *Composer*, que es el gestor de dependencias de PHP.

```
composer create-project laravel/laravel Backend
```

Al finalizar la instalación, se genera toda la estructura básica del proyecto. En la siguiente imagen se puede visualizar esa estructura:

Nombre	Fecha de modificación	Tipo	Tamaño
app	16/04/2025 16:19	Carpeta de archivos	
bootstrap	16/04/2025 16:19	Carpeta de archivos	
config	16/04/2025 16:19	Carpeta de archivos	
database	16/04/2025 16:26	Carpeta de archivos	
public	16/04/2025 16:19	Carpeta de archivos	
resources	16/04/2025 16:19	Carpeta de archivos	
routes	19/04/2025 17:33	Carpeta de archivos	
storage	16/04/2025 16:19	Carpeta de archivos	
tests	16/04/2025 16:19	Carpeta de archivos	
vendor	16/04/2025 16:26	Carpeta de archivos	
.editorconfig	15/04/2025 8:24	Archivo de origen ...	1 KB
.env	19/04/2025 19:00	Archivo ENV	2 KB
.gitattributes	15/04/2025 8:24	Archivo de origen ...	1 KB
.gitignore	15/04/2025 8:24	Archivo de origen ...	1 KB
artisan	15/04/2025 8:24	Archivo	1 KB
composer.json	15/04/2025 8:24	Archivo de origen ...	3 KB
composer.lock	16/04/2025 16:21	Archivo LOCK	291 KB
package.json	15/04/2025 8:24	Archivo de origen ...	1 KB
phpunit.xml	15/04/2025 8:24	Archivo XML	2 KB
README.md	15/04/2025 8:24	Archivo de origen ...	4 KB
vite.config.js	15/04/2025 8:24	Archivo de origen ...	1 KB

2) Configuración del entorno:

Para configurar correctamente la conexión con la base de datos local del proyecto, en el archivo `.env` del mismo se debe indicar la base de datos que se va a utilizar, así como el tipo de conexión a emplear y la contraseña (si la hubiere).

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=consulvet
DB_USERNAME=root
DB_PASSWORD=
```

3) Verificación del entorno:

Una vez configurado, se inicia el servidor de desarrollo de Laravel, que arranca en <http://localhost:8000>.

```
php artisan serve
```

Confirmamos que todo ha funcionando correctamente accediendo a través del navegador; si ha sido así, Laravel mostrará en pantalla su página de bienvenida.

► Desarrollo del código

➤ Creación de la base de datos:

Para crear la base de datos con Laravel, lo primero es crearla **vacía** por entorno gráfico desde, en mi caso, phpMyAdmin.

Para todo lo demás, Laravel cuenta con una potente herramienta de migraciones que permite definir la estructura de la base de datos directamente desde la terminal, utilizando comandos de migración. Con esas migraciones, permite crear todas las tablas de la base de datos e insertar todos sus datos.

- CREACIÓN DE LAS TABLAS:

```
php artisan make:migration create_nombreTabla_table
```

Dentro de cada archivo de migración (almacenados en *database/migrations*) se define la estructura de la tabla, incluyendo el tipo de cada dato, las restricciones de eliminación y actualización si las hubiere, la clave primaria y la clave ajena (si recibe).

En la siguiente imagen se incluye, a modo de ejemplo, el código de creación de la tabla Usuarios:

```
> Users > sgf94 > Documents > DAW_2ºCURSO > TFC > CONSULVET > Backend > database > migrations > 2025_04_17_112416_create_usuarios_table.php
1  <?php
2
3  use Illuminate\Database\Migrations\Migration;
4  use Illuminate\Database\Schema\Blueprint;
5  use Illuminate\Support\Facades\Schema;
6
7  return new class extends Migration
8  {
9      /**
10      * Run the migrations.
11      */
12      public function up(): void
13      {
14          Schema::create('usuarios', function (Blueprint $table) {
15              $table->string('nombre_usuario', 30)->primary();
16              $table->string('contraseña', 60);
17              $table->enum('tipo', ['cuidador', 'veterinario', 'centro']);
18              $table->date('fecha_alta');
19              $table->date('fecha_cambio_contraseña');
20              $table->char('cuidador', 9);
21              $table->timestamps();
22
23              $table->foreign('cuidador')->references('dni')->on('cuidadores')->onDelete('cascade')->onUpdate('no action');
24          });
25      }
26
27      /**
28      * Reverse the migrations.
29      */
30      public function down(): void
31      {
32          Schema::dropIfExists('usuarios');
33      }
34  };
35
```

Después de crear todas las migraciones, es necesario confirmarlas para que se creen automáticamente todas las tablas en la base de datos:

```
php artisan migrate
```

- INSERCIÓN DE DATOS EN LAS TABLAS:

Laravel ofrece un sistema de inserción de datos en tablas basado en lo que se denominan “seeders” (“sembradoras”). Las seeders permiten insertar múltiples registros en cada tabla sin que el programador lo tenga que hacer manualmente desde phpMyAdmin, o aquel con el que esté trabajado.

El uso de seeders hacen que el desarrollo del código sea más rápido y automatizado, mejora el control sobre la estructura y coherencia de los datos, y permite que éstos sean reproducibles entre los diferentes entornos de desarrollo (pruebas, producción...)

Se necesita crear un seeder por cada entidad de la base de datos:

```
php artisan make:seeder nombreTablaSeeder
```

Dentro de cada seeder (almacenados en *database/seeders*) se insertan todos los registros que competan, en función de los datos que almacena la tabla para la que haya sido creado.

En la siguiente imagen se incluye, a modo de ejemplo, el código de inserción de datos de la tabla Usuarios:

```
C:\> Users > sgf94 > Documents > DAW_2ºCURSO > TFC > CONSULVET > Backend > database > seeders > usuariosSeeder.php
1  <?php
2
3  namespace Database\Seeders;
4
5  use Illuminate\Database\Console\Seeds\WithoutModelEvents;
6  use Illuminate\Database\Seeder;
7  use App\Models\Usuarios;
8
9  class usuariosSeeder extends Seeder
10 {
11
12     /**
13      * Run the database seeds.
14     */
15     public function run(): void
16     {
17         Usuarios::firstOrCreate([
18             'nombre_usuario' => 'Silvia123',
19             'contraseña' => 'Smokey21',
20             'tipo' => 'cuidador',
21             'fecha_alta' => '2025-04-18',
22             'fecha_cambio_contraseña' => '2025-04-18',
23             'cuidador' => '55643311A'
24         ]);
25         Usuarios::firstOrCreate([
26             'nombre_usuario' => 'Victor456',
27             'contraseña' => 'Niobe21',
28             'tipo' => 'cuidador',
29             'fecha_alta' => '2025-04-18',
30             'fecha_cambio_contraseña' => '2025-04-18',
31             'cuidador' => '55717994C'
32         ]);
33     }
34 }
```

Después de crear todas las seeders, es necesario confirmarlas para que los datos se inserten automáticamente en sus respectivas tablas en la base de datos:

```
php artisan db:seed --class=nombreTablaSeeder
```

A modo de ejemplo, se incluye la siguiente imagen de la tabla Usuarios de la base de datos, en phpMyAdmin, donde se puede ver como ha sido creada y rellenada con sus datos:

	nombre_usuario	contraseña	tipo	fecha_alta	fecha_cambio_contraseña	cuidador	created_at	updated_at
<input type="checkbox"/> Editar Copiar Borrar	Silvia123	Smokey21	cuidador	2025-04-18	2025-04-18	55643311A	2025-04-18 09:51:10	2025-04-18 09:51:10
<input type="checkbox"/> Editar Copiar Borrar	Victor456	Niobe21	cuidador	2025-04-18	2025-04-18	55717994C	2025-04-18 09:51:10	2025-04-18 09:51:10

➤ Modelo Vista Controlador:

- CREACIÓN DE LOS MODELOS:

Una vez creadas las tablas e insertado los datos pertinentes en las mismas, se crean los modelos (almacenados en *app/Models*). Cada entidad de la base de datos necesita su propio modelo personalizado, que define la tabla para la que ha sido creado, su clave primaria, los campos rellenables y las relaciones con otros modelos (si procede).

Los modelos se crean en Laravel mediante la instrucción:

```
php artisan make:model nombreModelo
```

En la siguiente imagen se incluye, a modo de ejemplo, el modelo de la tabla Usuarios:

```
C: > Users > sgf94 > Documents > DAW_2ºCURSO > TFC > CONSULVET > Backend > app > Models > Usuarios.php
1  <?php
2
3  namespace App\Models;
4
5  use Illuminate\Database\Eloquent\Model;
6  use Illuminate\Database\Eloquent\Factories\HasFactory;
7
8  class Usuarios extends Model
9  {
10     protected $table = 'usuarios';
11     protected $primaryKey = 'nombre_usuario';
12
13     protected $fillable = [
14         'nombre_usuario',
15         'contraseña',
16         'tipo',
17         'fecha_alta',
18         'fecha_cambio_contraseña',
19         'cuidador'
20     ];
21 }
22
```

- CREACIÓN DE LOS CONTROLADORES:

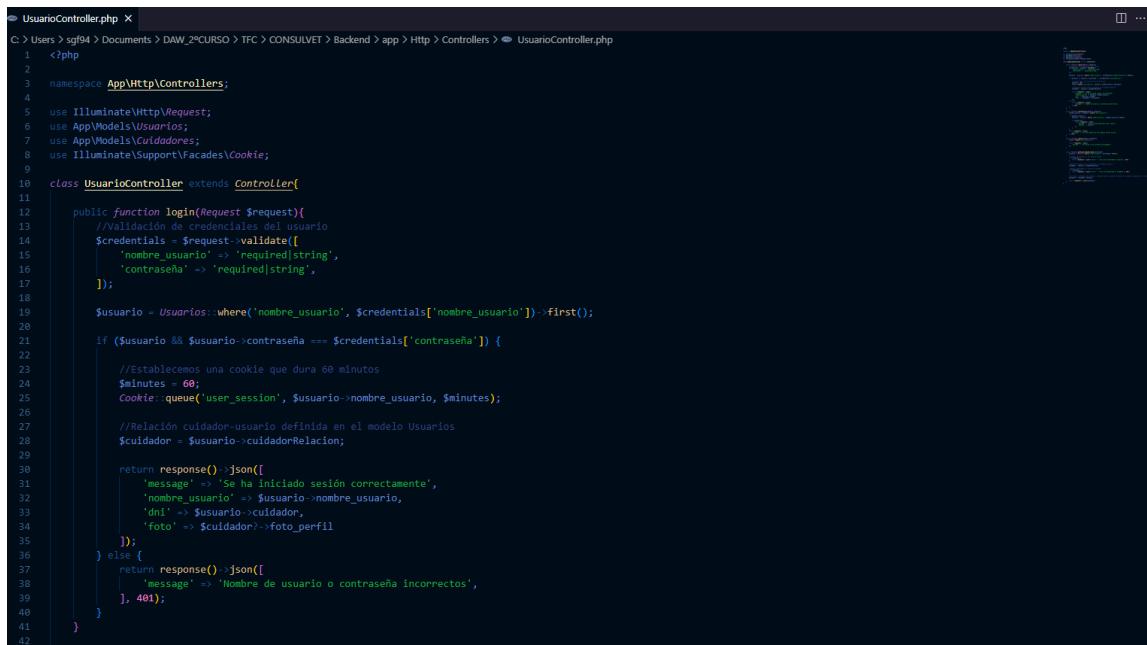
Los controladores (ubicados en *app/Http/Controllers*) se encargan de manejar lo que se llama “lógica de negocio” de la aplicación, actuando como intermediarios entre los modelos y las rutas. Cada entidad de la base de datos necesita, al igual que sucede con los modelos, su propio controlador personalizado, el cual define qué hacer cuando recibe una solicitud desde el frontend. Los controladores implementan el patrón CRUD (crear, listar, actualizar y eliminar) y otras funciones necesarias para satisfacer las posibles peticiones del cliente, siguiendo el patrón REST que usa distintos métodos HTTP (GET para obtener, POST para crear, PUT para actualizar y DELETE para eliminar) excepto en las funciones personalizadas.

Los controladores se crean en Laravel mediante la instrucción:

```
php artisan make:controller nombreTablaController
```

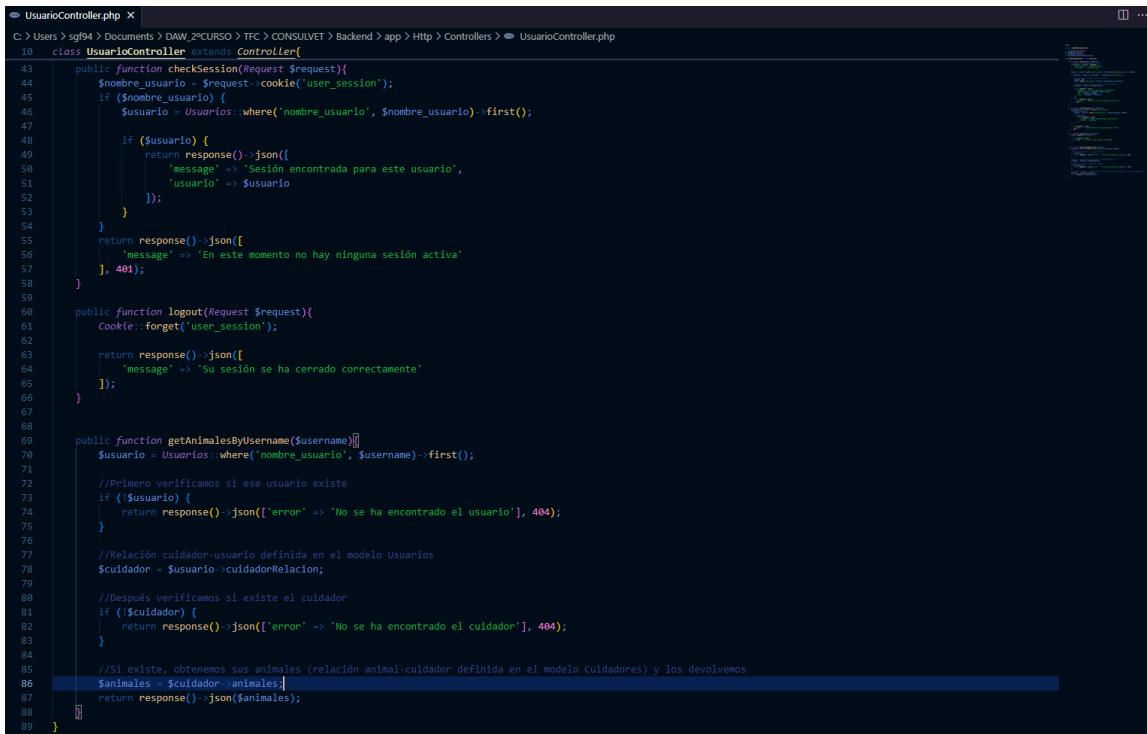
Para que los controladores funcionen correctamente, es necesario que puedan acceder a su modelo correspondiente. Por tanto, hay que importar el modelo en su controlador incluyendo en este último la ruta: `use App\Models\nombreModelo`

En las siguientes imágenes se incluye, a modo de ejemplo, el controlador de la tabla Usuarios:



```
C:\> Users > sgf94 > Documents > DAW_2ºCURSO > TFC > CONSULVET > Backend > app > Http > Controllers > UsUARIOController.php

1 <?php
2
3 namespace App\Http\Controllers;
4
5 use Illuminate\Http\Request;
6 use App\Models\USUARIOS;
7 use App\Models\CUIDADORES;
8 use Illuminate\Support\Facades\Cookie;
9
10 class USUARIOController extends Controller{
11
12     public function login(Request $request){
13         //Validación de credenciales del usuario
14         $credentials = $request->validate([
15             'nombre_usuario' => 'required|string',
16             'contraseña' => 'required|string',
17         ]);
18
19         $usuario = USUARIOS::where('nombre_usuario', $credentials['nombre_usuario'])->first();
20
21         if ($usuario && $usuario->contraseña === $credentials['contraseña']) {
22
23             //Establecemos una cookie que dura 60 minutos
24             $minutes = 60;
25             Cookie::queue('user_session', $usuario->nombre_usuario, $minutes);
26
27             //Relación cuidador-usuario definida en el modelo USUARIOS
28             $cuidador = $usuario->cuidadorRelacion;
29
30             return response()->json([
31                 'message' => 'Se ha iniciado sesión correctamente',
32                 'nombre_usuario' => $usuario->nombre_usuario,
33                 'dni' => $usuario->cuidador,
34                 'foto' => $cuidador->foto_perfil
35             ]);
36         } else {
37             return response()->json([
38                 'message' => 'Nombre de usuario o contraseña incorrectos',
39             ], 401);
40     }
41 }
42
```



```
C:\> Users > sgf94 > Documents > DAW_2ºCURSO > TFC > CONSULVET > Backend > app > Http > Controllers > UsUARIOController.php

10 class USUARIOController extends Controller{
11
12     public function checkSession(Request $request){
13         $nombre_usuario = $request->cookie('user_session');
14         if ($nombre_usuario) {
15             $usuario = USUARIOS::where('nombre_usuario', $nombre_usuario)->first();
16
17             if ($usuario) {
18                 return response()->json([
19                     'message' => 'Sesión encontrada para este usuario',
20                     'usuario' => $usuario
21                 ]);
22             }
23         }
24         return response()->json([
25             'message' => 'En este momento no hay ninguna sesión activa'
26         ], 401);
27     }
28
29     public function logout(Request $request){
30         Cookie::forget('user_session');
31
32         return response()->json([
33             'message' => 'Su sesión se ha cerrado correctamente'
34         ]);
35     }
36
37     public function getAnimalesByUsername($username){
38         $usuario = USUARIOS::where('nombre_usuario', $username)->first();
39
40         //Primero verificamos si ese usuario existe
41         if (! $usuario) {
42             return response()->json(['error' => 'No se ha encontrado el usuario'], 404);
43         }
44
45         //Relación cuidador-usuario definida en el modelo USUARIOS
46         $cuidador = $usuario->cuidadorRelacion;
47
48         //Después verificamos si existe el cuidador
49         if (! $cuidador) {
50             return response()->json(['error' => 'No se ha encontrado el cuidador'], 404);
51         }
52
53         //Si existe, obtenemos sus animales (relación animal-cuidador definida en el modelo CUIDADORES) y los devolvemos
54         $animales = $cuidador->animales;
55         return response()->json($animales);
56     }
57 }
```

➤ Cookies

Se han implementando cookies para permitir que el usuario permanezca con la sesión iniciada aunque refresque la página.

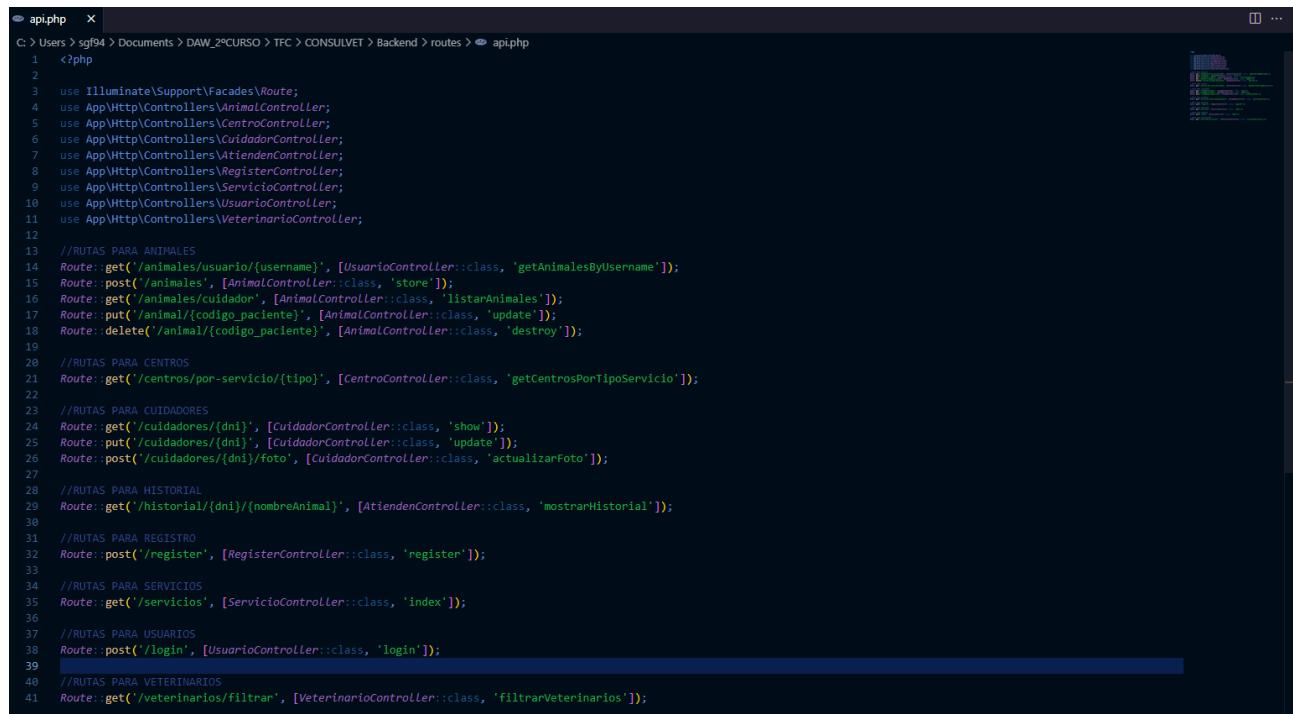
Para ello, cuando el usuario inicia sesión se valida su nombre de usuario y su contraseña; si sus credenciales son correctas, se crea la cookie y se almacenará durante 60 minutos.

En las imágenes del controlador de Usuarios incluidas en el apartado anterior, se puede visualizar también la implementación de la cookie.

➤ Enrutamiento

Las rutas conectan las peticiones que realiza el frontend con los métodos correspondientes definidos en los controladores. Laravel separa las rutas según el tipo de petición. Estas rutas deben declararse en el archivo *api.php*, ubicado dentro de la carpeta *routes*. Cada ruta indica el tipo de petición HTTP (get, post, put, delete...), el *endpoint* (ruta concreta) que se usará en la URL, y el controlador y método que se va a ejecutar para satisfacer la petición.

En la siguiente imagen se incluye el conjunto de rutas definidas:



```
C:\> Users > sgf94 > Documents > DAW_2ºCURSO > TFC > CONSULVET > Backend > routes > api.php
1 <?php
2
3 use Illuminate\Support\Facades\Route;
4 use App\Http\Controllers\AnimalController;
5 use App\Http\Controllers\CentroController;
6 use App\Http\Controllers\CuidadorController;
7 use App\Http\Controllers\AtiendenController;
8 use App\Http\Controllers\RegisterController;
9 use App\Http\Controllers\ServicioController;
10 use App\Http\Controllers\UsuarioController;
11 use App\Http\Controllers\VeterinarioController;
12
13 //RUTAS PARA ANIMALES
14 Route::get('/animales/usuario/{username}', [UsuarioController::class, 'getAnimalesByUsername']);
15 Route::post('/animales', [AnimalController::class, 'store']);
16 Route::get('/animales/cuidador', [AnimalController::class, 'listarAnimales']);
17 Route::put('/animal/{codigo_paciente}', [AnimalController::class, 'update']);
18 Route::delete('/animal/{codigo_paciente}', [AnimalController::class, 'destroy']);
19
20 //RUTAS PARA CENTROS
21 Route::get('/centros/por-servicio/{tipo}', [CentroController::class, 'getCentrosPorTipoServicio']);
22
23 //RUTAS PARA CUIDADORES
24 Route::get('/cuidadores/{dni}', [CuidadorController::class, 'show']);
25 Route::put('/cuidadores/{dni}', [CuidadorController::class, 'update']);
26 Route::post('/cuidadores/{dni}/foto', [CuidadorController::class, 'actualizarFoto']);
27
28 //RUTAS PARA HISTORIAL
29 Route::get('/historial/{dni}/{nombreAnimal}', [AtiendenController::class, 'mostrarHistorial']);
30
31 //RUTAS PARA REGISTRO
32 Route::post('/register', [RegisterController::class, 'register']);
33
34 //RUTAS PARA SERVICIOS
35 Route::get('/servicios', [ServicioController::class, 'index']);
36
37 //RUTAS PARA USUARIOS
38 Route::post('/login', [UsuarioController::class, 'login']);
39
40 //RUTAS PARA VETERINARIOS
41 Route::get('/veterinarios/filtrar', [VeterinarioController::class, 'filtrarVeterinarios']);
```

Para verificar que todas las rutas se han registrado correctamente, se pueden listar mediante el comando:

```
php artisan route:list
```

➤ Pruebas

Para validar el correcto funcionamiento de rutas y controladores, previo a la creación del lado frontend, se ha utilizado [Postman](#).

Postman es una herramienta que sirve para lanzar peticiones HTTP de forma visual. Por tanto, es muy útil para corroborar las distintas funcionalidades de la aplicación: login, consultas de inserción, de edición, de borrado... y todas las que hayan sido programadas.

Si todo está correctamente configurado y funcionando, tras enviar la petición, Postman muestra la respuesta JSON que ha devuelto Laravel.

En la siguiente imagen se incluye, a modo de ejemplo, la petición para el login y cómo Postman ha mostrado la respuesta devuelta por Laravel:

The screenshot shows the Postman interface. At the top, it says "POST http://localhost:8000/api/login". The "Body" tab is selected, showing a JSON payload with two fields: "nombre_usuario": "Silvia123" and "contraseña": "Smokey21". Below the body, the "Test Results" section shows a successful response with status 200 OK, time 248 ms, and size 293 B. The response body is also JSON, containing a message "Inicio de sesión correcto" and the user's name "Silvia123".

● DESARROLLO DE LA APLICACIÓN EN EL ENTORNO CLIENTE

○ Creación del proyecto en Angular y desarrollo del frontend

Una vez tenemos una base de datos sobre la que trabajar y un backend robusto bien definido, el siguiente paso es crear el proyecto en Angular para poder empezar a desarrollar el código del lado frontend e iniciar la comunicación con el backend, para que la aplicación funcione correctamente como un todo dinámico.

Para el desarrollo del lado Cliente - como se ha indicado en el apartado *Medios a Emplear* – se va a utilizar *Angular*.

► Un poco de contexto

Angular es una plataforma de desarrollo creada por Google que permite la creación de lo que denominamos aplicaciones web dinámicas “SPA”, es decir: aplicaciones de una sola página (“Single Page Application”). Está basado en TypeScript, un lenguaje JavaScript tipado; e implementa el patrón MVC (Modelo-Vista-Controlador) para separar la lógica de negocio de la interfaz de usuario. Su enfoque se basa en la componentización de la aplicación, es decir: la aplicación se divide en componentes independientes y reutilizables. Éstos componentes se comunican entre sí para que toda la aplicación funcione correctamente de forma rápida, sencilla y fluida, sin tener que recargar toda la página cada vez que el usuario realiza una acción (como un click en un botón). Cada uno de estos componentes cuenta con:

- su propia interfaz (plantilla HTML, la Vista)
- su propio estilo dentro del estilo general de la aplicación (archivo CSS)
- una clase que define el comportamiento del componente, es decir, su lógica (archivo TypeScript).

Ésto facilita considerablemente el mantenimiento de la aplicación y la escalabilidad del proyecto. La comunicación entre dichos componentes se realiza, en esencia, de dos formas:

- Desde el componente padre hacia el componente hijo: el componente padre puede pasarle datos a su/s componente/s hijo/s mediante propiedades. La información que recibe el hijo se identifica precedida del decorador “@Input()”.
- Desde el componente hijo hacia el componente padre: los hijos pueden pasar datos a su padre mediante la emisión de eventos, que éste escuchará. Esa información que envían al padre se identifica mediante el decorador “@Output()”, y se envía haciendo uso de la clase *EventEmitter*.

Para que esta comunicación sea efectiva, es necesario importar y exportar las funcionalidades, servicios, incluso componentes, que cada componente va a utilizar. Por ejemplo, podemos importar elementos de librerías (como puede ser *EventEmitter*, o los propios *Input* y *Output*, recogidos en *@angular/core*). También se pueden importar módulos completos de Angular, con sus directivas, funcionalidades etc; por ejemplo, *FormsModule* para trabajar con formularios, *HttpClientModule* para trabajar bajo el protocolo seguro http, entre otras. En cuanto a las exportaciones, éstas permiten que los componentes, directivas, etc. estén disponibles para aquellos módulos o componentes que los necesiten importar para utilizarlos.

La renderización de toda la lógica con sus vistas y estilos corre a cargo del componente principal *app*; este componente es el que se carga en el navegador cuando se ejecuta la aplicación, y llama y ejecuta todos los componentes que la forman, como un todo bien engranado.

► Preparando Angular

*Nota: la empresa trabaja en la versión 14 de Angular, por lo que mi entorno de trabajo para las prácticas se preparó para funcionar con esa versión. Debido a ésto, y por motivos de compatibilidades, el presente proyecto ha sido desarrollado en la misma versión de Angular que la empleada para las prácticas en la empresa.

1) Instalación de Angular:

Para poder trabajar con Angular, el primer paso es instalar correctamente todo el entorno; ésto incluye: CLI de Angular, Angular core, Node.js y Package Manager. Es importante que las versiones de cada uno de ellos sean compatibles entre sí para que todo pueda funcionar.

Angular es dependiente de Node.js para funcionar, por lo cual el primer paso es instalar Node en una versión compatible con la versión de Angular utilizada (en este caso, la versión 16.13.0):

[Descargar Node](#)

Una vez descargado, sólo hay que seguir los pasos del instalador. Junto con Node se instala también Package Manager (“npm”); es necesario que su versión sea compatible también (en este caso, la versión 9.3.1).

Ambas versiones se pueden comprobar mediante los comandos:

```
node -v
```

```
npm -v
```

Si la versión de npm no coincide, la instalamos especificando la versión con la instrucción:

```
npm install -g npm@9.3.1
```

El siguiente paso será instalar el CLI de Angular, en este caso en la versión 14.2.13:

```
npm install -g @angular/cli@14.2.13
```

Por último, instalamos Angular Core, en este caso en la versión 14.3.0, y verificamos que la instalación se ha realizado correctamente:

```
npm install @angular/core@14.3.0
```

```
ng version
```

```
Angular CLI: 14.2.13
Node: 16.13.0
Package Manager: npm 9.3.1
OS: win32 x64

Angular: 14.3.0
... animations, common, compiler, compiler-cli, core, forms
... platform-browser, platform-browser-dynamic, router

Package                      Version
-----
@angular-devkit/architect      0.1402.13
@angular-devkit/build-angular   14.2.13
@angular-devkit/core            14.2.13
@angular-devkit/schematics      14.2.13
@angular/cli                   14.2.13
@schematics/angular             14.2.13
rxjs                           7.5.7
typescript                     4.7.4
```

2) Creación de un proyecto:

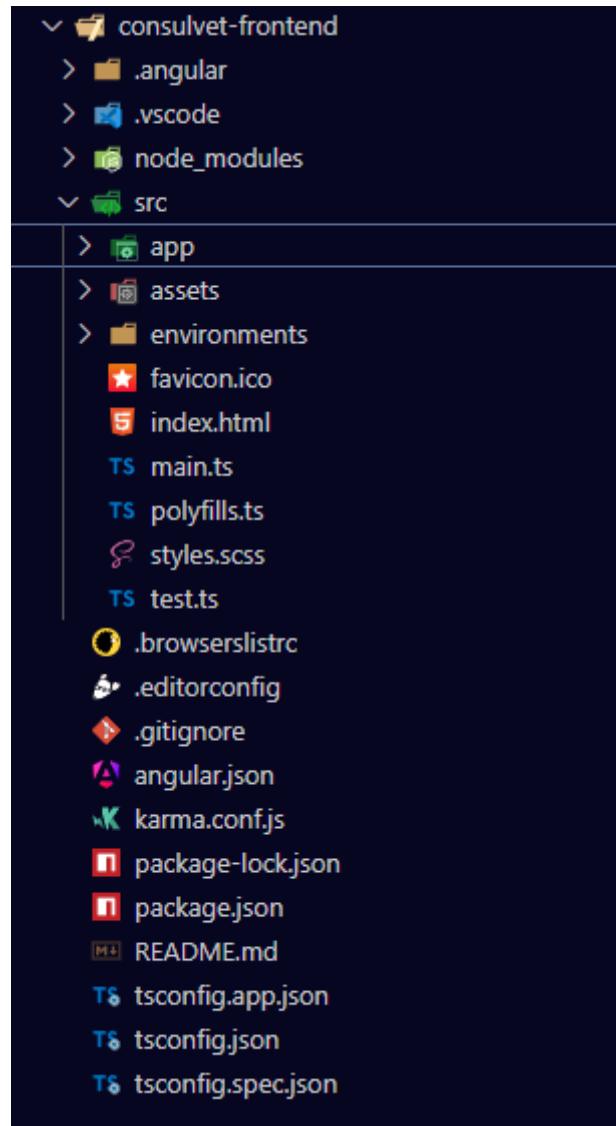
Desde la misma terminal, o en una nueva terminal desde el propio VSCode, el proyecto se crea mediante la instrucción:

```
ng new nombreProyecto
```

Accedemos a él para empezar a trabajar:

```
cd nombreProyecto
```

En la siguiente imagen se puede visualizar la estructura base del proyecto:



Si todo se ha generado correctamente, podemos lanzar el proyecto mediante la instrucción:

```
ng serve
```

► Desarrollo del código

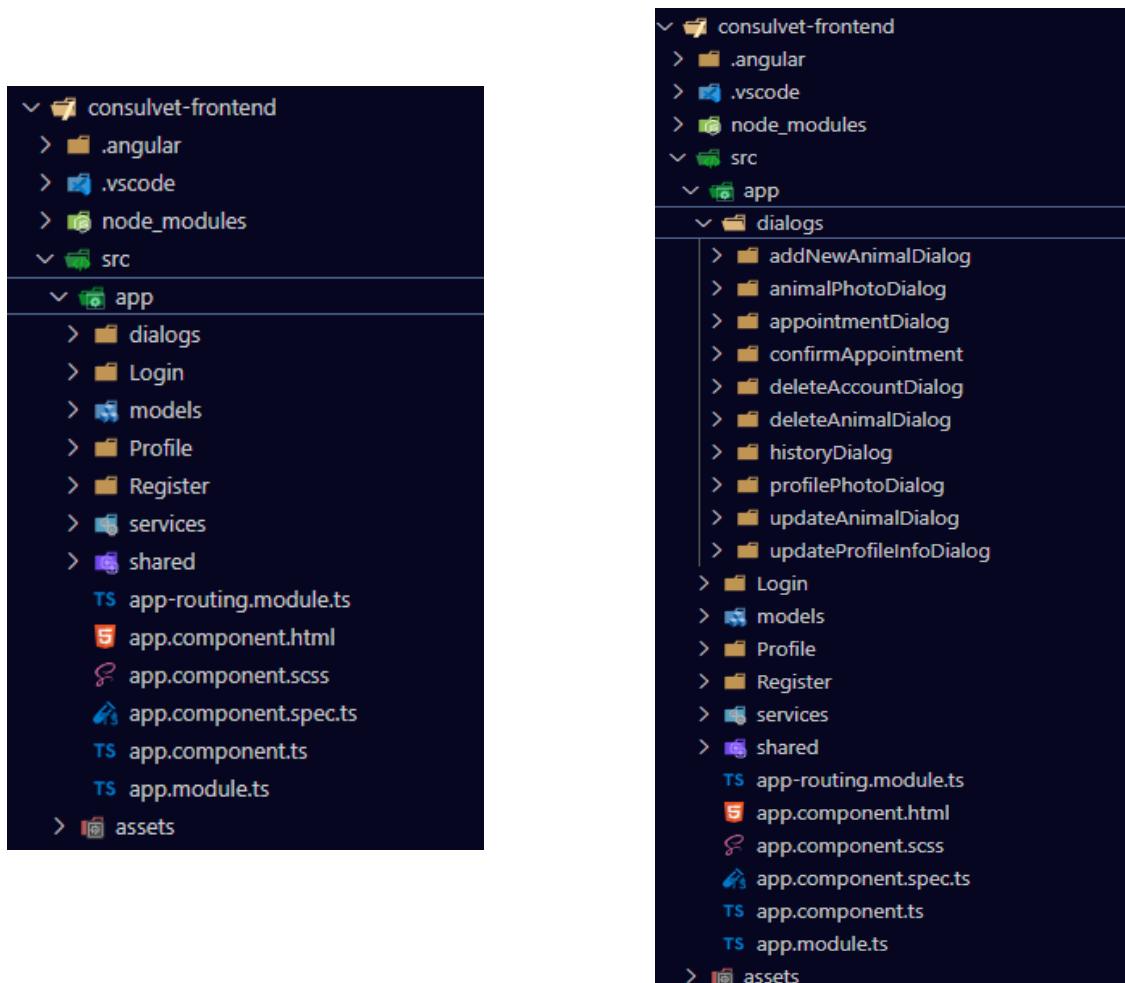
➤ Creación de los componentes:

Con el proyecto creado, el siguiente paso es crear los componentes de la aplicación. El árbol de componentes de Consulvet consta, en esencia, de tres componentes hermanos que manejan la lógica general de la aplicación: Login, Register y Profile; junto con múltiples componentes de diálogo, que son los encargados de presentar y manejar el grueso de las funcionalidades y comunicaciones con la base de datos de la misma.

La elección de cuadros de diálogo en lugar de componentes al uso, ha sido tomada en base a criterios de interfaz sencilla, intuitiva y fácilmente usable, presentándose ésta al usuario limpia de contenido amontonado y ruido visual.

En las siguientes imágenes se puede ver el árbol de componentes y los cuadros de diálogo:

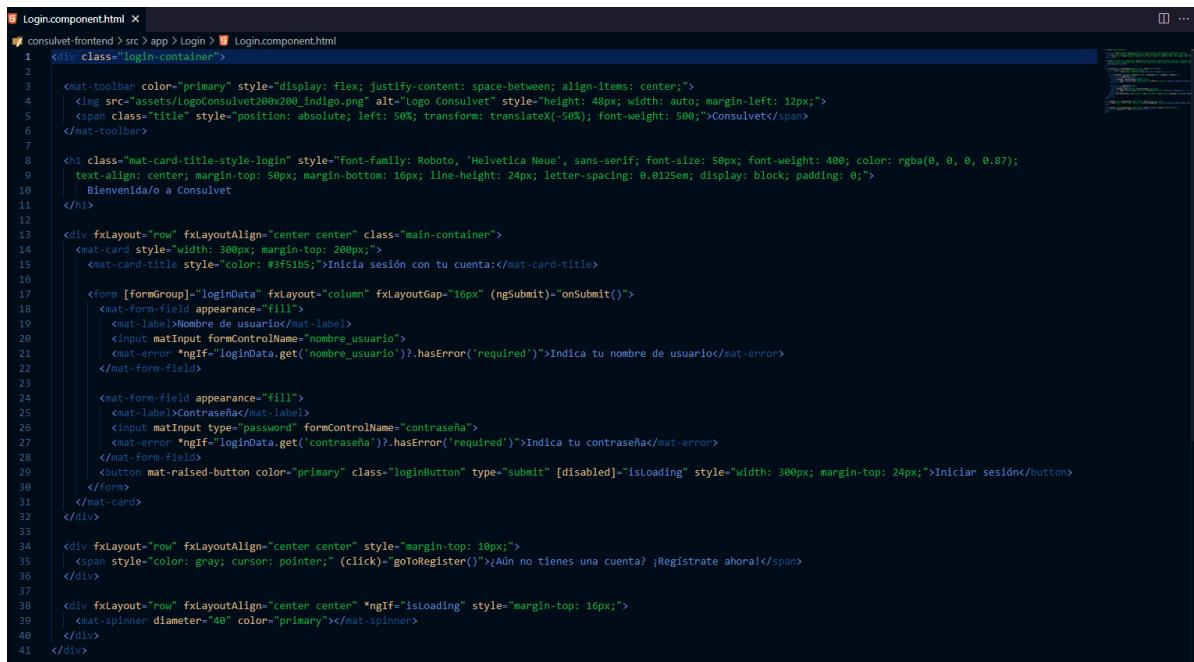
*Nota: las carpetas para los modelos y los servicios también están visibles en estas imágenes



Un componente se puede crear de dos formas diferentes, bien por consola de comandos con la instrucción: `ng generate module nombreModulo` (versión abreviada: `ng g m nombreModulo`); o bien por entorno gráfico, haciendo click derecho sobre la carpeta app → generate module. En cualquiera de las dos, el resultado es un componente completo que consta de cuatro archivos: archivo .html, archivo .css, archivo typescript (.ts) y módulo (archivo .module.ts).

El módulo exporta al componente para que pueda ser utilizado por otros componentes que necesiten llamarlo, e importa todos los módulos de otros componentes, servicios, módulos de Angular Material, etc. que necesita utilizar para que la lógica del componente y su constructo html funcionen como deben.

En las siguientes imágenes se muestra, a modo de ejemplo, el componente Login y su vista en la interfaz web:



```
<div class="login-container">
  <mat-toolbar color="primary" style="display: flex; justify-content: space-between; align-items: center;">
    
    <span class="title" style="position: absolute; left: 50%; transform: translateX(-50%); font-weight: 500; display: block; padding: 0;">Consulvet</span>
  </mat-toolbar>

  <h1 class="mat-card-title-style-login" style="font-family: Roboto, 'Helvetica Neue', sans-serif; font-size: 50px; font-weight: 400; color: rgba(0, 0, 0, 0.87); text-align: center; margin-top: 50px; margin-bottom: 16px; line-height: 24px; letter-spacing: 0.0125em; display: block; padding: 0;">Bienvenida/o a Consulvet</h1>

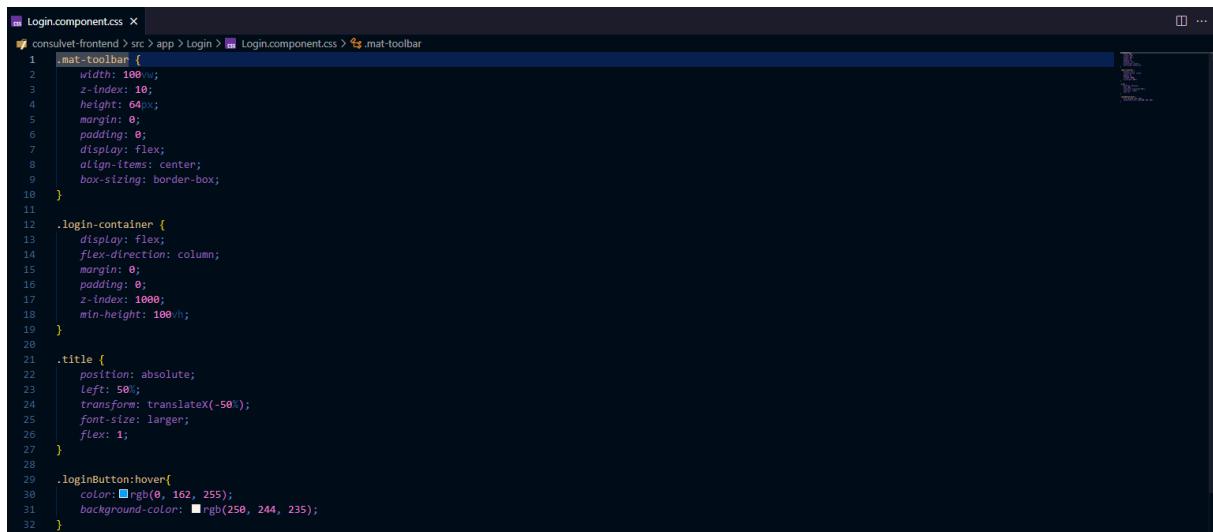
  <div fxLayout="row" fxLayoutAlign="center center" class="main-container">
    <mat-card style="width: 300px; margin-top: 200px;">
      <mat-card-title style="color: #3f51b5;">Inicia sesión con tu cuenta:</mat-card-title>

      <form [FormGroup] "loginData" fxLayout="column" fxLayoutGap="16px" (ngSubmit)="onSubmit()">
        <mat-form-field appearance="fill">
          <mat-label>Nombre de usuario</mat-label>
          <input matInput formControlName="nombre_usuario">
          <mat-error *ngIf="loginData.get('nombre_usuario').hasError('required')">Indica tu nombre de usuario</mat-error>
        </mat-form-field>

        <mat-form-field appearance="fill">
          <mat-label>Contraseña</mat-label>
          <input matInput type="password" formControlName="contraseña">
          <mat-error *ngIf="loginData.get('contraseña').hasError('required')">Indica tu contraseña</mat-error>
        </mat-form-field>
        <button mat-raised-button color="primary" class="loginButton" type="submit" [disabled]="isLoading" style="width: 300px; margin-top: 24px;">Iniciar sesión</button>
      </form>
    </mat-card>
  </div>

  <div fxLayout="row" fxLayoutAlign="center center" style="margin-top: 10px;">
    <span style="color: gray; cursor: pointer;" (click)="goToRegister()">Aún no tienes una cuenta? ¡Regístrate ahora!</span>
  </div>

  <div fxLayout="row" fxLayoutAlign="center center" *ngIf="isLoading" style="margin-top: 16px;">
    <mat-spinner diameter="40" color="primary"></mat-spinner>
  </div>
</div>
```

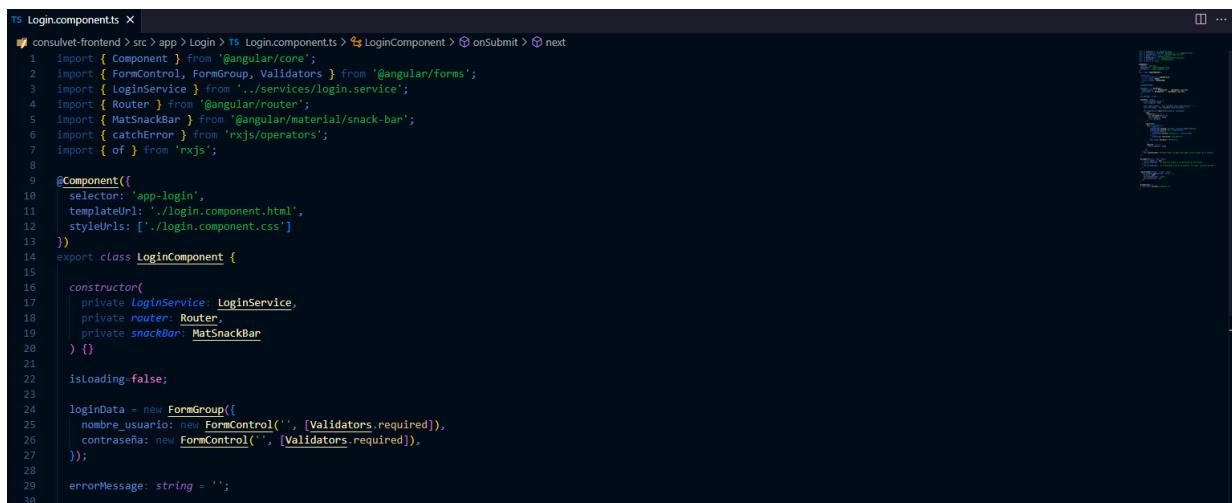


```
.mat-toolbar {
  width: 100vw;
  z-index: 10;
  height: 64px;
  margin: 0;
  padding: 0;
  display: flex;
  align-items: center;
  box-sizing: border-box;
}

.login-container {
  display: flex;
  flex-direction: column;
  margin: 0;
  padding: 0;
  z-index: 1000;
  min-height: 100vh;
}

.title {
  position: absolute;
  left: 50%;
  transform: translateX(-50%);
  font-size: larger;
  flex: 1;
}

.loginButton:hover {
  color: #rgb(0, 162, 255);
  background-color: #rgb(250, 244, 235);
}
```



```
import { Component } from '@angular/core';
import { FormControl, FormGroup, Validators } from '@angular/forms';
import { LoginService } from './services/login.service';
import { Router } from '@angular/router';
import { MatSnackBar } from '@angular/material/snack-bar';
import { catchError } from 'rxjs/operators';
import { of } from 'rxjs';

@Component({
  selector: 'app-login',
  templateUrl: './login.component.html',
  styleUrls: ['./login.component.css']
})
export class LoginComponent {

  constructor(
    private loginService: LoginService,
    private router: Router,
    private snackBar: MatSnackBar
  ) {}

  isLoading = false;

  loginData = new FormGroup({
    nombre_usuario: new FormControl('', [Validators.required]),
    contraseña: new FormControl('', [Validators.required]),
  });

  errorMessage: string = '';
}
```

```
30  onSubmit(): void {
31    if (this.loginData.valid) {
32      this.isLoading = true;
33
34      const nombre_usuario = this.loginData.value.nombre_usuario ?? '';
35      const contraseña = this.loginData.value.contraseña ?? '';
36
37      this.loginService.login(nombre_usuario, contraseña)
38        .pipe(
39          catchError(error => {
40            this.messageError(error);
41            this.isLoading = false;
42            return of(null);
43          })
44        )
45      .subscribe({
46        next: (response) => {
47          if (response) {
48            localStorage.setItem('username', response.nombre_usuario);
49            localStorage.setItem('dni', response.dni);
50            if (response.foto) []
51              | localStorage.setItem('foto_perfil', response.foto);
52            | ] else {
53            | | localStorage.removeItem('foto_perfil');
54            | }
55            | this.router.navigate(['/Profile']);
56          }
57        },
58        complete: () => {
59          this.isLoading = false;
60        }
61      });
62    } else {
63      this.loginSnackBar('Introduce todos tus datos para poder iniciar sesión con tu cuenta');
64    }
65  }
66}
```

```
57
58  messageError(error: any): void {
59    if (error.status === 401) {
60      this.errorMessage = 'El nombre de usuario o la contraseña son incorrectos';
61    } else {
62      this.errorMessage = 'Se ha producido un error de conexión. Por favor, inténtalo de nuevo';
63    }
64  }
65
66  loginSnackBar(message: string): void {
67    this.snackBar.open(message, 'Cerrar', {
68      duration: 2000,
69      horizontalPosition: 'center',
70      verticalPosition: 'top'
71    });
72  }
73
74  goToRegister() {
75    this.router.navigate(['/Register']);
76  }
77}
```

```
TS Login.module.ts ✘
consulvet-frontend > src > app > Login > TS Login.module.ts > LoginModule
1  import { NgModule } from '@angular/core';
2  import { CommonModule } from '@angular/common';
3  import { LoginComponent } from './Login.component';
4  import { FormsModule, ReactiveFormsModule } from '@angular/forms';
5  import { MatButtonModule } from '@angular/material/button';
6  import { MatInputModule } from '@angular/material/input';
7  import { MatFormFieldModule } from '@angular/material/form-field';
8  import { MatCardModule } from '@angular/material/card';
9  import { FlexLayoutModule } from '@angular/flex-layout';
10 import { MatProgressSpinnerModule } from '@angular/material/progress-spinner';
11 import { MatToolbarModule } from '@angular/material/toolbar';
12 import { MatIconModule } from '@angular/material/icon';
13
14 @NgModule({
15   imports: [
16     CommonModule,
17     FormsModule,
18     ReactiveFormsModule,
19     MatButtonModule,
20     MatInputModule,
21     MatFormFieldModule,
22     MatCardModule,
23     FlexLayoutModule,
24     MatProgressSpinnerModule,
25     MatToolbarModule,
26     MatIconModule
27   ],
28   declarations: [LoginComponent]
29 })
30 export class LoginModule {}
```



Bienvenida/o a Consulvet

Inicia sesión con tu cuenta:

Nombre de usuario *

Contraseña *

Iniciar sesión

¿Aún no tienes una cuenta? ¡Regístrate ahora!

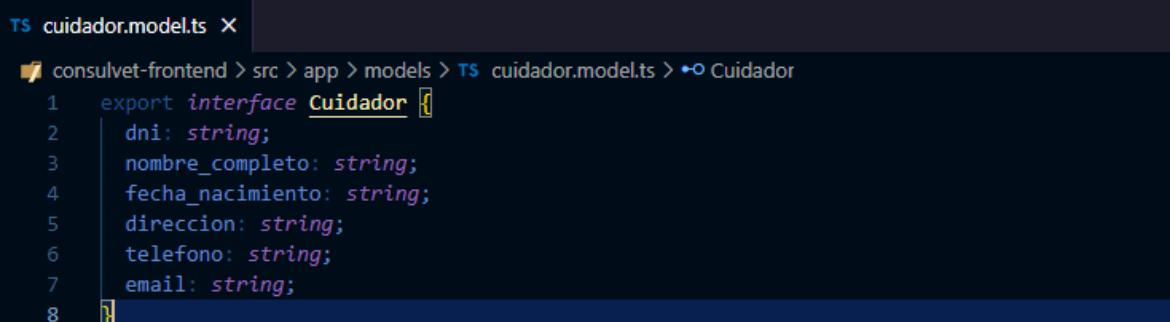
La “magia” de los componentes no reside sólo en que se pueden comunicar entre sí; sino también en que un mismo componente se puede reutilizar en otras partes de la aplicación, incluyendo su estructura, su vista y sus estilos, simplemente modificando mínimamente su lógica y los datos presentados al usuario, de forma coherente con la funcionalidad que se pretende conseguir y la consulta que se está aplicando a la base de datos.

➤ Creación de los modelos:

Replicar en el frontend los modelos para cada entidad de la base de datos, tal y como están definidos en el backend, es una práctica muy recomendable porque:

- Hace que los datos entre ambos entornos sean consistentes, ya que se están manejando los mismos campos, estructuras y tipos de datos. Ello reduce las posibilidades de aparición de errores por discrepancias entre los mismos y los datos fluyen correctamente entre el front y el back.
- Mejoran el mantenimiento del código, ya que si algo cambia en el backend, estos cambios se pueden replicar fácilmente en el frontend. Además, facilita la comprensión del código por cualquier desarrollador/ora, quien sabrá qué esperar de cada entidad, y también qué espera recibir la base de datos.
- Facilita considerablemente las validaciones en campos y formularios, ya que al ser una réplica del modelo en el backend, asegura que los datos que van a ir en las querys de las consultas son exactamente los correctos, los que la base de datos espera recibir.
- Mejora la escalabilidad de un proyecto, ya que si éste crece, los modelos ya creados – iguales a los de la base de datos- se pueden compartir entre diferentes componentes y servicios.

En la siguiente imagen, a modo de ejemplo, se puede ver el modelo para la entidad Cuidadores:



```
TS cuidador.model.ts X
consulvet-frontend > src > app > models > TS cuidador.model.ts > Cuidador
1 export interface Cuidador {
2   dni: string;
3   nombre_completo: string;
4   fecha_nacimiento: string;
5   direccion: string;
6   telefono: string;
7   email: string;
8 }
```

➤ Creación de los diálogos:

Los cuadros de diálogo son componentes de Angular Material que se abren y muestran en pantalla al evento de click. Es decir, cuando el usuario pulsa un botón en la aplicación, se abre el cuadro de diálogo correspondiente.

El potencial de los cuadros de diálogo viene dado por dos razones: por un lado, permiten mantener despejada de un exceso de contenido la aplicación; y por otro, pueden contener y presentar al usuario cualquier tipo de estructura de información (formularios, tablas, paso a pasos, otros cuadros de diálogo, etc.).

Como los cuadros de diálogo reaccionan al evento de click, es posible anidar tantos cuadros de diálogo como sean necesarios (siempre de forma coherente y no abusiva); ello es muy útil en procesos donde se requieren confirmaciones del usuario, de forma que se presentan de forma clara al usuario pero mínimamente invasiva, y con las mismas desaparecen de la interfaz, fomentando una experiencia de usuario limpia y agradable.

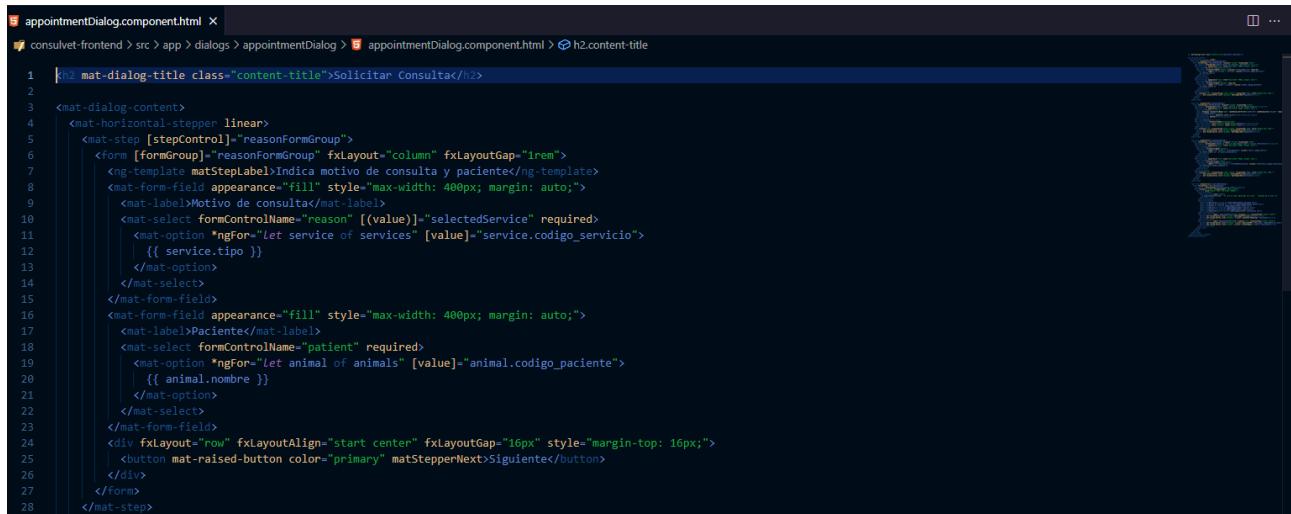
Por todas esas razones, la funcionalidad principal de Consulvet – solicitud de consulta veterinaria, la cual requiere dar varios pasos e implica confirmaciones por parte del usuario – ha sido presentada dentro de un cuadro de diálogo, en forma de “stepper”. Un stepper es un componente prefabricado de Angular Material, que permite al usuario realizar una acción en varios pasos. En cada paso, se pueden realizar las subacciones que sean necesarias.

En el caso del proceso de solicitud de consulta de Consulvet, cada subacción es un paso lógico para solicitar la consulta (por ejemplo, indicar el nombre del animal que acude) y está asociado a una consulta diferente sobre la base de datos, de forma que la información dada por el usuario se vaya almacenando y sirva de filtro para la información que se va a recoger y mostrar desde la base de datos en pasos posteriores.

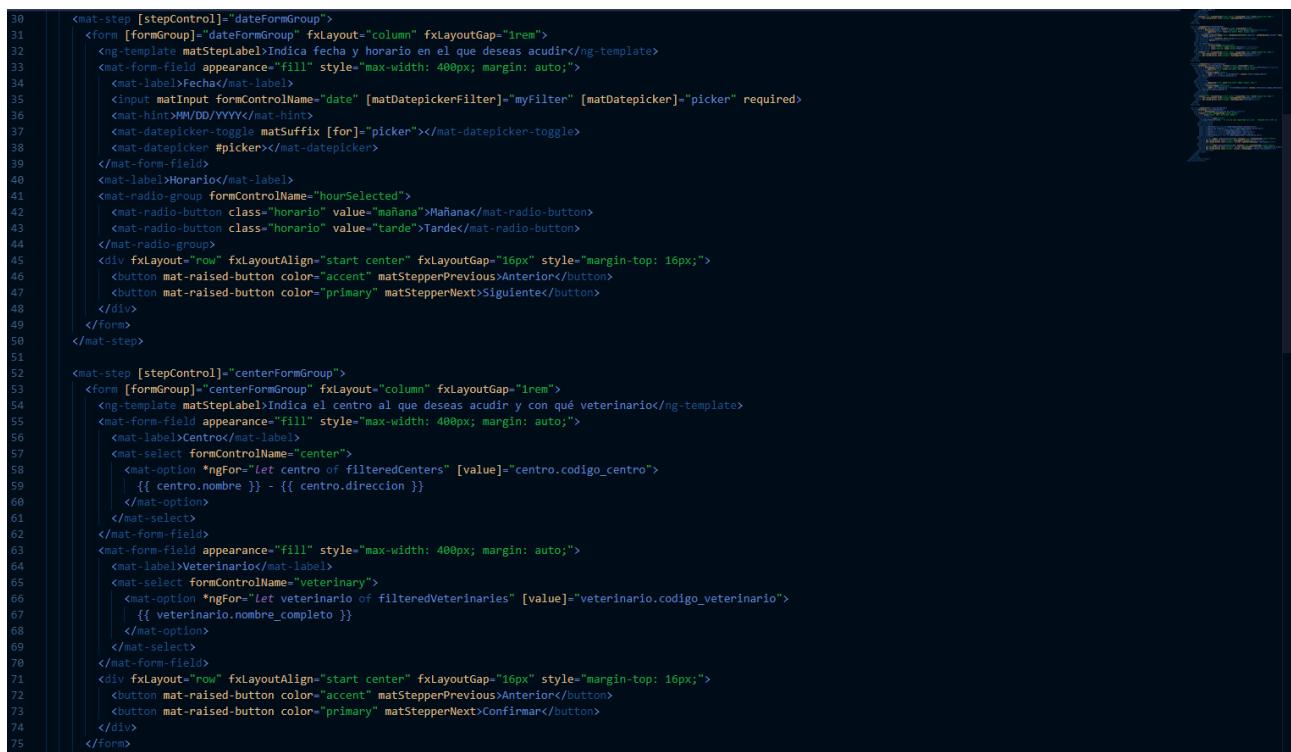
Un punto fuerte que tiene Angular es que permite crear plantillas html (“ngTemplates”), de forma que podemos presentar dos opciones en la misma vista, en función de una acción del usuario. De esa forma, no es necesario crear un componente más, sino que con una condición (“ngIf”) se muestra la vista de una forma o de otra, cambiando ésta en tiempo real, sin recargas de página y sin ni siquiera recarga del propio componente.

En este caso, en el último paso de la solicitud se le muestra al usuario un resumen de los datos que ha introducido y se le pide que confirme. Al momento de confirmar, se le muestra un mensaje de confirmación de cita solicitada y un resumen de los datos. Este cambio implica sólo un cambio de plantilla html, lo cual, además de todos los beneficios ya descritos, optimiza el trabajo del desarrollador.

En las siguientes imágenes se muestra el código del diálogo de solicitud de consulta veterinaria, y su vista en la interfaz web:



```
appointmentDialog.component.html
1 <h2 mat-dialog-title class="content-title">Solicitar Consulta</h2>
2
3 <mat-dialog-content>
4   <mat-horizontal-stepper linear>
5     <mat-step [stepControl]="reasonFormGroup">
6       <form [formGroup]="reasonFormGroup" fxLayout="column" fxLayoutGap="1rem">
7         <ng-template matStepLabel>Indica motivo de consulta y paciente</ng-template>
8         <mat-form-field appearance="fill" style="max-width: 400px; margin: auto;">
9           <mat-label>Motivo de consulta</mat-label>
10          <mat-select formControlName="reason" [(value)]="selectedService" required>
11            <ngFor="let service of services" [value]="">
12              {{ service.tipo }}</ngFor>
13            </mat-option>
14          </mat-select>
15        </mat-form-field>
16        <mat-form-field appearance="fill" style="max-width: 400px; margin: auto;">
17          <mat-label>Paciente</mat-label>
18          <mat-select formControlName="patient" required>
19            <ngFor="let animal of animals" [value]="">
20              {{ animal.nombre }}</ngFor>
21            </mat-option>
22          </mat-select>
23        </mat-form-field>
24        <div fxLayout="row" fxLayoutAlign="start center" fxLayoutGap="16px" style="margin-top: 16px;">
25          <button mat-raised-button color="primary" matStepperNext>Siguiente</button>
26        </div>
27      </form>
28    </mat-step>
```



```
30   <mat-step [stepControl]="dateFormGroup">
31     <form [formGroup]="dateFormGroup" fxLayout="column" fxLayoutGap="1rem">
32       <ng-template matStepLabel>Indica fecha y horario en el que deseas acudir</ng-template>
33       <mat-form-field appearance="fill" style="max-width: 400px; margin: auto;">
34         <mat-label>Fecha</mat-label>
35         <input matInput formControlName="date" [matDatepickerFilter]="" [matDatePicker]="myFilter" [matDatePicker]="picker" required>
36         <mat-hint>MM/DD/YYYY</mat-hint>
37         <mat-datepicker-toggle matSuffix [for]=""></mat-datepicker-toggle>
38         <mat-datepicker #picker></mat-datepicker>
39       </mat-form-field>
40       <mat-label>Horario</mat-label>
41       <mat-radio-group formControlName="hourSelected">
42         <mat-radio-button class="horario" value="mañana">Mañana</mat-radio-button>
43         <mat-radio-button class="horario" value="tarde">Tarde</mat-radio-button>
44       </mat-radio-group>
45       <div fxLayout="row" fxLayoutAlign="start center" fxLayoutGap="16px" style="margin-top: 16px;">
46         <button mat-raised-button color="accent" matStepperPrevious>Anterior</button>
47         <button mat-raised-button color="primary" matStepperNext>Siguiente</button>
48       </div>
49     </form>
50   </mat-step>
51
52   <mat-step [stepControl]="centerFormGroup">
53     <form [formGroup]="centerFormGroup" fxLayout="column" fxLayoutGap="1rem">
54       <ng-template matStepLabel>Indica el centro al que deseas acudir y con qué veterinario</ng-template>
55       <mat-form-field appearance="fill" style="max-width: 400px; margin: auto;">
56         <mat-label>Centro</mat-label>
57         <mat-select formControlName="center">
58           <ngFor="let centro of filteredCenters" [value]="">
59             {{ centro.nombre }} - {{ centro.direccion }}</ngFor>
60           </mat-option>
61         </mat-select>
62       </mat-form-field>
63       <mat-form-field appearance="fill" style="max-width: 400px; margin: auto;">
64         <mat-label>Veterinario</mat-label>
65         <mat-select formControlName="veterinary">
66           <ngFor="let veterinarian of filteredVeterinaries" [value]="">
67             {{ veterinarian.nombre_completo }}</ngFor>
68           </mat-option>
69         </mat-select>
70       </mat-form-field>
71       <div fxLayout="row" fxLayoutAlign="start center" fxLayoutGap="16px" style="margin-top: 16px;">
72         <button mat-raised-button color="accent" matStepperPrevious>Anterior</button>
73         <button mat-raised-button color="primary" matStepperNext>Confirmar</button>
74       </div>
75     </form>
```

```

76   </mat-step>
77
78   <mat-step [stepControl]="confirmFormGroup">
79     <form [formGroup]="confirmFormGroup">
80       <ng-template matStepLabel>Confirmación de cita</ng-template>
81       <div fxLayout="column" fxLayoutAlign="center center">
82         <mat-card style="width: 100%; max-width: 600px;">
83           <mat-card-header>
84             <mat-card-title class="title">
85               {{ appointmentConfirmed ? 'Tu cita ha sido registrada con éxito' : 'Resumen de la Cita' }}</mat-card-title>
86           </mat-card-header>
87           <mat-card-content>
88             <p><strong>Paciente:</strong> {{ summaryAppointment?.paciente }}</p>
89             <p><strong>Motivo de consulta:</strong> {{ summaryAppointment?.motivo }}</p>
90             <p><strong>Fecha:</strong> {{ summaryAppointment?.fecha }}</p>
91             <p><strong>Horario:</strong> {{ summaryAppointment?.horario }}</p>
92             <p><strong>Centro:</strong> {{ summaryAppointment?.centro }}</p>
93             <p><strong>Veterinario:</strong> {{ summaryAppointment?.veterinario }}</p>
94           </mat-card-content>
95           <mat-card-actions *ngIf="!appointmentConfirmed" fxLayout="row" fxLayoutAlign="center center">
96             <button mat-raised-button color="accent" matStepperPrevious="Anterior"></button>
97             <button mat-raised-button color="primary" (click)="openConfirmDialog()">Continuar</button>
98           </mat-card-actions>
99           <mat-card-actions *ngIf="appointmentConfirmed" fxLayout="row" fxLayoutAlign="center center">
100             <button mat-raised-button color="primary" (click)="generatePDF()">Descargar comprobante de cita</button>
101             <button mat-raised-button color="accent" (click)="closeStepper()">Cerrar solicitud</button>
102           </mat-card-actions>
103         </mat-card>
104       </div>
105     </form>
106   </mat-step>
107 </mat-horizontal-stepper>
108 </mat-dialog-content>

```

```

ts appointmentDialog.component.ts ✘
ts consulvet-frontend > src > app > dialogs > appointmentDialog > TS appointmentDialog.component.ts > AppointmentDialogComponent > filterVeterinaries
1 import { Component, OnInit } from '@angular/core';
2 import { FormBuilder, FormGroup, Validators } from '@angular/forms';
3 import { ServiciosService } from 'src/app/services/servicios.service';
4 import { Servicio } from 'src/app/models/servicio.model';
5 import { Animal } from 'src/app/models/animal.model';
6 import { AnimalesService } from 'src/app/services/animales.service';
7 import { VeterinariosService } from 'src/app/services/veterinarios.service';
8 import { Veterinario } from 'src/app/models/veterinario.model';
9 import { Centro } from 'src/app/models/centro.model';
10 import { CentrosService } from 'src/app/services/centros.service';
11 import { ConfirmAppointmentComponent } from '../confirmAppointment/confirmAppointment.component';
12 import { MatDialog, MatDialogRef } from '@angular/material/dialog';
13 import { jsPDF } from 'jspdf';
14
15 @Component({
16   selector: 'app-appointmentDialog',
17   templateUrl: './appointmentDialog.component.html',
18   styleUrls: ['./appointmentDialog.component.css']
19 })
20 export class AppointmentDialogComponent implements OnInit {
21   //VARIABLES:
22   //Grupos del formulario para presentar en el stepper
23   reasonFormGroup: FormGroup;
24   dateFormGroup: FormGroup;
25   centerFormGroup: FormGroup;
26   confirmFormGroup: FormGroup;
27
28   //Motivo de consulta (= servicios en la base de datos)
29   services: Servicio[] = [];
30   selectedService: number | null = null;
31
32   //Paciente
33   animals: Animal[] = [];
34
35   //Centro al que acude (filtrado por el motivo de consulta)
36   filteredCenters: Centro[] = [];
37
38   //Veterinario que le va a atender (filtrado por el motivo de consulta y el centro al que acude)
39   filteredVeterinaries: Veterinario[] = [];
40   selectedVeterinaryName: string = '';
41
42   //Confirmación de cita
43   appointmentConfirmed: Boolean = false;
44

```

```

45  constructor(
46    private formBuilder: FormBuilder,
47    private servicioService: ServiciosService,
48  ) {
49    this.dateFormGroup.get('hourSelected').valueChanges.subscribe(() => {
50      const selectedServiceCode = this.reasonFormGroup.value.reason;
51      const selectedCentroid = this.centerFormGroup.value.center;
52      const selectedHora = this.dateFormGroup.value.hourSelected;
53      if (selectedServiceCode && selectedCentroid && selectedHora) {
54        const servicioSeleccionado = this.services.find(s => s.codigo_servicio === selectedServiceCode);
55        if (servicioSeleccionado) {
56          this.filterVeterinaries(servicioSeleccionado.tipo, selectedCentroid, selectedHora);
57        }
58      }
59    });
60
61    this.centerFormGroup.get('center').valueChanges.subscribe((selectedCentroid) => {
62      const selectedServiceCode = this.reasonFormGroup.value.reason;
63      const selectedHora = this.dateFormGroup.value.hourSelected;
64      if (selectedServiceCode && selectedCentroid && selectedHora) {
65        const servicioSeleccionado = this.services.find(s => s.codigo_servicio === selectedServiceCode);
66        if (servicioSeleccionado) {
67          this.filterVeterinaries(servicioSeleccionado.tipo, selectedCentroid, selectedHora);
68        }
69      }
70    });
71
72    /*Función propia del Datepicker de Material para manejar los días hábiles que queremos que se muestren
73     - He excluido el domingo por ser dia festivo (es dia de urgencias, pero mi app solo contempla consultas ordinarias)*/
74    myFilter = (d: Date | null): boolean => {
75      const day = (d || new Date()).getDay();
76      return day !== 0;
77    };
78
79    filterVeterinaries(servicio: string, centroid: number, horario: string): void {
80      if (servicio && centroid && horario) {
81        this.serviciosService.getFilteredVeterinaries(servicio, centroid, horario)
82          .subscribe({
83            next: (vets) => {
84              this.filteredVeterinaries = vets;
85            },
86            error: (error) => console.error('Lo sentimos, no hemos podido cargar los veterinarios:', error)
87          });
88      }
89    };
90  }
91

```

```

136  loadCenters(tipoSeleccionado: string): void {
137    if (!tipoSeleccionado) return;
138    this.centroService.getCentersByType(tipoSeleccionado).subscribe({
139      next: (centros) => {
140        this.filteredCenters = centros;
141      },
142      error: (error) => console.error('Lo sentimos, no hemos podido cargar los centros:', error)
143    });
144  }
145
146  get summaryAppointment(): {
147    const { reason } = this.reasonFormGroup.value;
148    const { patient } = this.reasonFormGroup.value;
149    const { date, hourSelected } = this.dateFormGroup.value;
150    const { center, veterinary } = this.centerFormGroup.value;
151
152    //El resumen de cita no se muestra hasta que todos los campos estén rellenos
153    if (!reason || !patient || !date || !hourSelected || !center || !veterinary) {
154      return null;
155    }
156
157    const paciente = this.animals.find(a => a.codigo_paciente === patient);
158    const motivo = this.services.find(s => s.codigo_servicio === reason);
159    const centro = this.filteredCenters.find(c => c.codigo_centro === center);
160    const veterinario = this.filteredVeterinaries.find(v => v.codigo_veterinario === veterinary);
161
162    if (!paciente || !motivo || !centro || !veterinario) {
163      return null;
164    }
165
166    return {
167      paciente: paciente.nombre,
168      motivo: motivo.tipo,
169      fecha: new Date(date).toLocaleDateString(),
170      horario: hourSelected,
171      centro: `${centro.nombre} - ${centro.direccion}`,
172      veterinario: veterinario.nombre_completo
173    };
174  }
175

```

```

175 //Llamamos al componente que maneja la lógica para la confirmación de una cita
176 openConfirmDialog() {
177   const dialogRef = this.confirmDialog.open(ConfirmAppointmentComponent);
178
179   dialogRef.afterClosed().subscribe(result => {
180     if (result === true) {
181       | this.appointmentConfirmed = true;
182     }
183   });
184 }
185
186 //Generamos un pdf con los datos resumen de la cita, para que el cuidador lo descargue si quiere y lo tenga a modo de volante de cita
187 generatePDF() {
188   const doc = new jsPDF();
189
190   const logo = new Image();
191   logo.src = 'assets/LogoConsulvet200x200.png';
192   logo.onload = () => {
193     doc.addImage(logo, 'PNG', 10, 10, 30, 30);
194
195     doc.setFont('helvetica', 'bold');
196     doc.setFontSize(18);
197     doc.setTextColor(63, 81, 181);
198     doc.text('Volante de Cita', 105, 25, { align: 'center' });
199
200     doc.setDrawColor(200, 200, 200);
201     doc.line(10, 40, 200, 40);
202
203     const datos = [
204       { label: 'Paciente:', value: this.summaryAppointment?.paciente },
205       { label: 'Motivo de consulta:', value: this.summaryAppointment?.motivo },
206       { label: 'Fecha:', value: this.summaryAppointment?.fecha },
207       { label: 'Horario:', value: this.summaryAppointment?.horario },
208       { label: 'Centro:', value: this.summaryAppointment?.centro },
209       { label: 'Veterinario:', value: this.summaryAppointment?.veterinario },
210     ];
211
212
213   let y = 55;
214   for (const item of datos) {
215     doc.setFontSize(12);
216     doc.setTextColor(0, 0, 0);
217
218     doc.setFont('helvetica', 'bold');
219     const labelWidth = doc.getTextWidth(item.label);
220     doc.text(item.label, 20, y);

```

```

221
222     doc.setFont('helvetica', 'normal');
223     doc.text(item.value || '', 20 + labelWidth + 1, y);
224
225     y += 10;
226   }
227
228   doc.setDrawColor(63, 81, 181);
229   doc.rect(5, 5, 200, 287);
230
231   doc.setFontSize(18);
232   doc.setTextColor(100);
233   doc.text(
234     'Gracias por confiar en Consulvet - www.consulvet.com - Tel: (+34) 123-456-789',
235     105,
236     285,
237     { align: 'center' }
238   );
239
240   doc.save('volante-cita.pdf');
241 }
242
243 closeStepper(){
244   this.dialogRef.close();
245 }
246
247

```

```

T appointmentDialog.module.ts ✘
 consulvet-frontend > src > app > dialogs > appointmentDialog > T appointmentDialog.module.ts > ...
1 import { NgModule } from '@angular/core';
2 import { CommonModule } from '@angular/common';
3 import { AppointmentDialogComponent } from './appointmentDialog.component';
4 import { MatDialogModule } from '@angular/material/dialog';
5 import { MatStepperModule } from '@angular/material/stepper';
6 import { MatInputModule } from '@angular/material/input';
7 import { MatButtonModule } from '@angular/material/button';
8 import { ReactiveFormsModuleModule } from '@angular/forms';
9 import { FlexLayoutModule } from '@angular/flex-layout';
10 import { MatSelectModule } from '@angular/material/select';
11 import { MatOptionModule } from '@angular/material/core';
12 import { MatDatepickerModule } from '@angular/material/datepicker';
13 import { MatNativeDateModule } from '@angular/material/core';
14 import { MatRadioModule } from '@angular/material/radio';
15 import { FormsModule } from '@angular/forms';
16 import { MatCardModule } from '@angular/material/card';
17
18 @NgModule({
19   imports: [
20     CommonModule,
21     MatDialogModule,
22     MatStepperModule,
23     MatInputModule,
24     MatButtonModule,
25     ReactiveFormsModuleModule,
26     FlexLayoutModule,
27     MatSelectModule,
28     MatOptionModule,
29     MatDatepickerModule,
30     MatNativeDateModule,
31     MatRadioModule,
32     FormsModule,
33     MatCardModule
34   ],
35   declarations: [AppointmentDialogComponent]
36 })
37 export class AppointmentDialogModule { }

```

 Consulvet

Smokey
Felina - Europeo de pelo corto

Iris
Felina - Europea de pelo corto

Casper
Felina - Mestizo de siamés

Solicitar Consulta

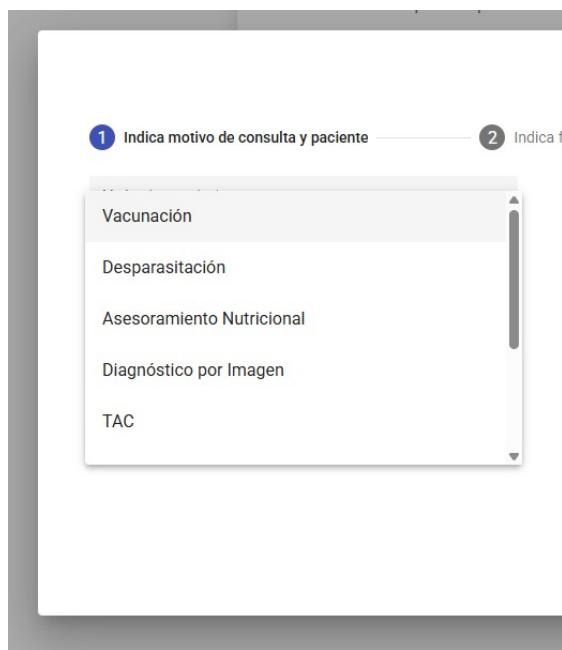
1 Indica motivo de consulta y paciente 2 Indica fecha y horario en el que deseas acudir 3 Indica el centro al que deseas acudir y con qué veterinario 4 Confirmación de cita

Motivo de consulta *

Paciente *

Siguiente

Sobre nosotros Contacto Política de privacidad
[f](#) [x](#) [@](#) [in](#)
© 2025 Consulvet. Todos los derechos reservados.



Smokey
Felina - Europeo de pelo corto

Iris
Felina - Europea de pelo corto

Casper
Felina - Mestizo de siamés

Solicitar Consulta

- 1 Indica motivo de consulta y paciente
- 2 Indica fecha y horario en el que deseas acudir
- 3 Indica el centro al que deseas acudir y con qué veterinario
- 4 Confirmación de cita

Motivo de consulta *

Vacunación

Smokey
Iris
Casper

Felina - Europea de pelo corto

Felina - Europea de pelo corto

Felina - Mestizo de siamés

Solicitar Consulta

- 1 Indica motivo de consulta y paciente
- 2 Indica fecha y horario en el que deseas acudir
- 3 Indica el centro al que deseas acudir y con qué veterinario
- 4 Confirmación de cita

Fecha *

MM/DD/YYYY

Horario

Mañana Tarde

[Anterior](#) [Siguiente](#)

Felina - Europeo de pelo corto

Solicitar Consulta

- 1 Indica motivo de consulta y paciente
- 2 Indica fecha y horario en el que deseas acudir
- 3 Indica el centro al que deseas acudir y con qué veterinario
- 4 Confirmación de cita

Fecha *

[Calendario](#)

MAY 2025						
S	M	T	W	T	F	S
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

Felina - Europeo de pelo corto

Solicitar Consulta

- 1 Indica motivo de consulta y paciente
- 2 Indica fecha y horario en el que deseas acudir
- 3 Indica el centro al que deseas acudir y con qué veterinario
- 4 Confirmación de cita

Fecha *

5/21/2025

MM/DD/YYYY

Horario

Mañana Tarde

[Anterior](#) [Siguiente](#)

Página | Europea de perros corse

Página | Europea de perros corse

Página | Europea de perros corse

Solicitar Consulta

1 Indica motivo de consulta y paciente 2 Indica fecha y horario en el que deseas acudir 3 Indica el centro al que deseas acudir y con qué veterinario 4 Confirmación de cita

Centro *

Veterinario *

Anterior Confirmar

Indica motivo de consulta y paciente Indic

Clínica Veterinaria Coruña Centro - Calle Coruña Ce...

Clinica Veterinaria Mondoñedo - Calle de la Higuera...

Veterinario *

Anterior Confirmar

Indica motivo de consulta y paciente Indic

Centro *
Clínica Veterinaria Coruña Centro - Calle Coruña ...

Ana López García

Anterior Confirmar

Solicitar Consulta

Indica motivo de consulta y paciente Indica fecha y horario en el que deseas acudir Indica el centro al que deseas acudir y con qué veterinario 4 Confirmación de cita

Resumen de la Cita

Paciente: Smokey
Motivo de consulta: Vacunación
Fecha: 21/5/2025
Horario: mañana
Centro: Clínica Veterinaria Coruña Centro - Calle Coruña Centro, nº 5, bajo, Coruña
Veterinario: Ana López García

Anterior Continuar

Solicitar Consulta

Indica motivo de consulta y paciente Indica fecha y horario en el que deseas acudir Indica el centro al que deseas acudir y con qué veterinario 4 Confirmación de cita

Resumen de la Cita

Paciente: Smokey
Motivo de consulta: Vacunación
Fecha: 21/5/2025
Horario: mañana
Centro: Clínica Veterinaria Coruña Centro - Calle Coruña Centro, nº 5, bajo, Coruña
Veterinario: Ana López García

Confirmar cita
¿Estás seguro de que deseas confirmar esta cita?

Cancelar Confirmar

Anterior Continuar

*Nota: El código del diálogo para confirmar la cita no está incluido en este ejemplo

Solicitar Consulta

Indica motivo de consulta y paciente Indica fecha y horario en el que deseas acudir Indica el centro al que deseas acudir y con qué veterinario 4 Confirmación de cita

Tu cita ha sido registrada con éxito

Paciente: Smokey
Motivo de consulta: Vacunación
Fecha: 21/5/2025
Horario: mañana
Centro: Clínica Veterinaria Coruña Centro - Calle Coruña Centro, nº 5, bajo, Coruña
Veterinario: Ana López García

Descargar comprobante de cita Cerrar solicitud

*Nota: obsérvese como en esta captura y la anterior, el componente presentado es el mismo y sólo cambia la plantilla html en función de que sea resumen de cita, o cita confirmada

➤ Creación de los servicios:

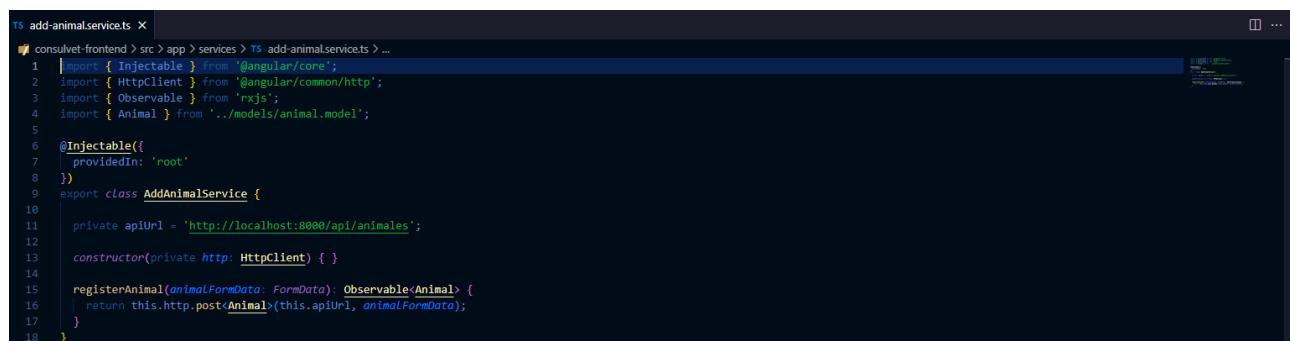
Los servicios en Angular son los componentes que se encargan de conectar el lado cliente con el lado servidor. Cada servicio recoge la ruta pertinente del archivo api.php que conlleva la ejecución de una consulta concreta. Después, cada uno de estos servicios es llamado por el o los componentes que van a manejar la lógica en el lado cliente para mostrar el resultado de dicha consulta.

Pongamos como ejemplo la funcionalidad: añadir un nuevo animal.

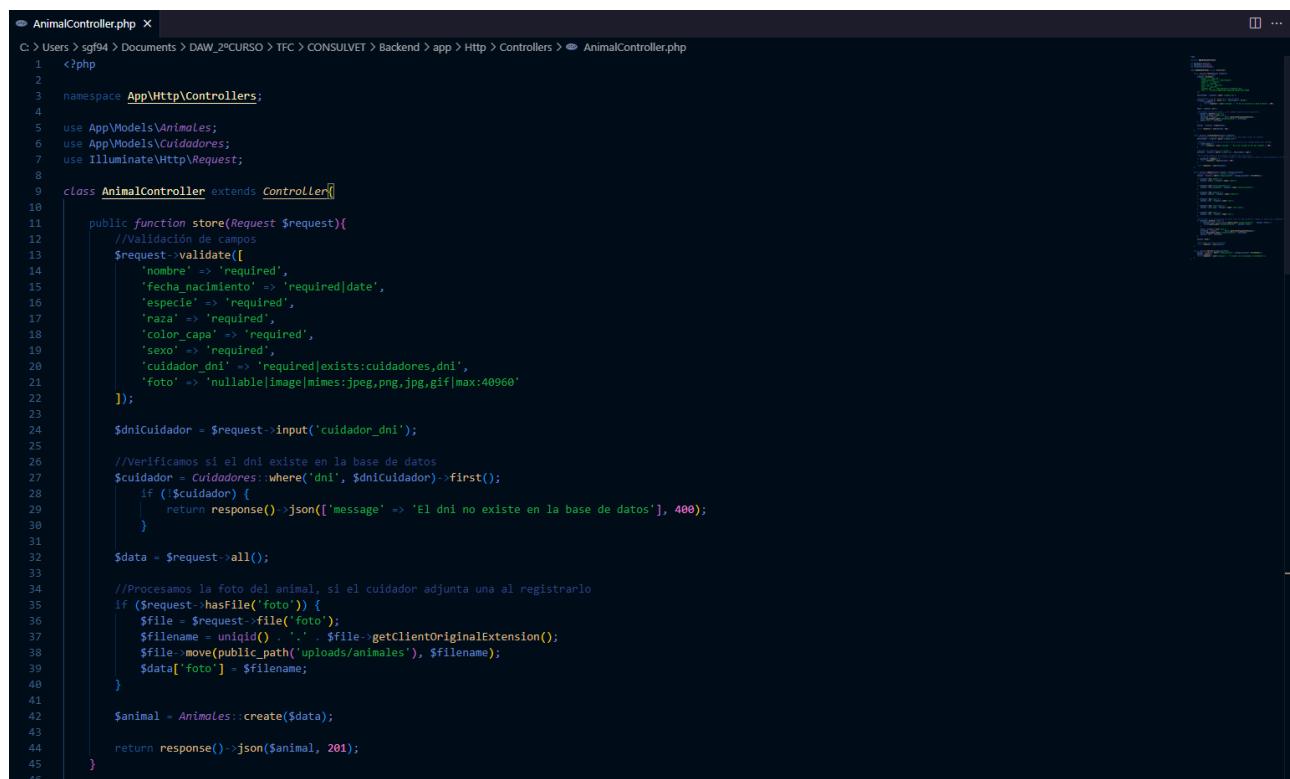
Esta funcionalidad que estamos implementando desde el frontend no deja de ser una consulta insert en la tabla animales, donde los animales se registran para el cuidador que tiene iniciada la sesión en base a su dni, que se almacena en local storage cuando inicia sesión para tenerlo disponible para todas las consultas donde se necesite.

En el servicio “*add-animal*” se recoge la ruta post al controlador *AnimalController* de Laravel, que contiene la función con la lógica necesaria para ejecutar esta consulta.

Las siguientes imágenes muestran el servicio add-animal.service.ts y la función responsable de la inserción de animales en el controlador AnimalController.php:



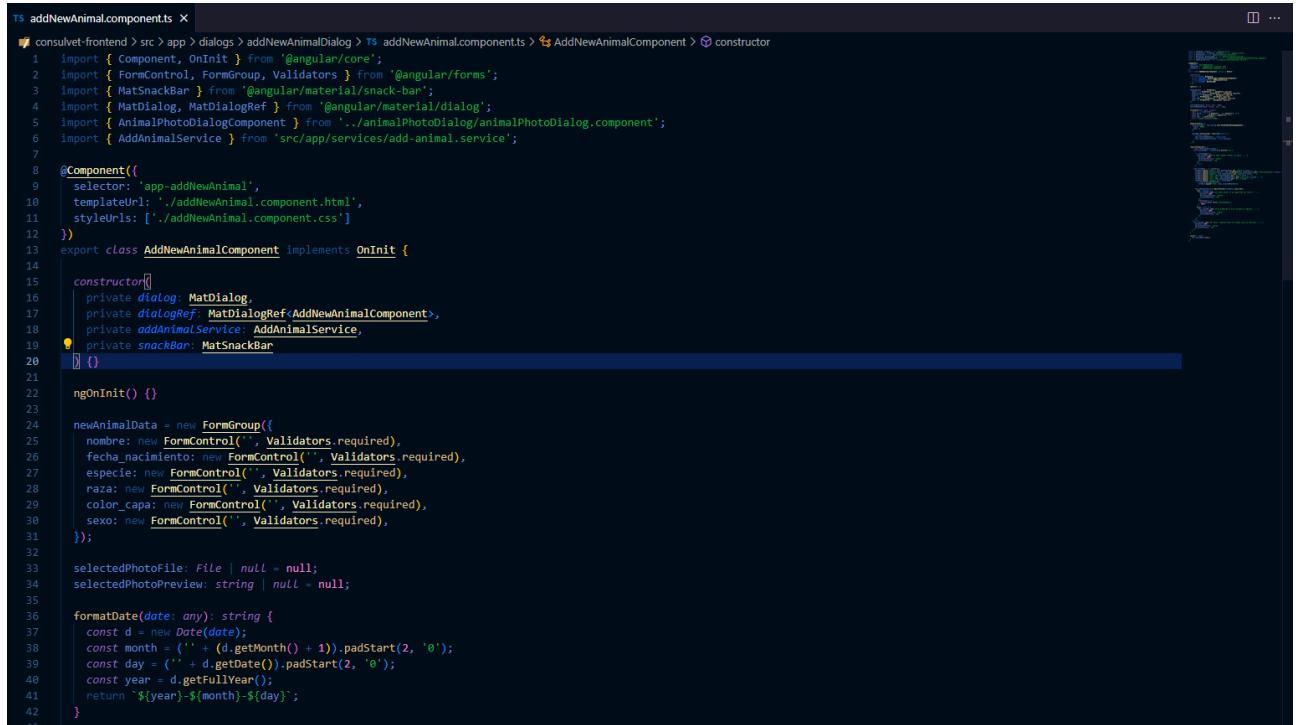
```
ts add-animal.service.ts
1 import { Injectable } from '@angular/core';
2 import { HttpClient } from '@angular/common/http';
3 import { Observable } from 'rxjs';
4 import { Animal } from '../models/animal.model';
5
6 @Injectable({
7   providedIn: 'root'
8 })
9 export class AddAnimalService {
10
11   private apiUrl = 'http://localhost:8000/api/animales';
12
13   constructor(private http: HttpClient) { }
14
15   registerAnimal(animalFormData: FormData): Observable<Animal> {
16     return this.http.post<Animal>(this.apiUrl, animalFormData);
17   }
18 }
```



```
AnimalController.php
1 <?php
2
3 namespace App\Http\Controllers;
4
5 use App\Models\Animales;
6 use App\Models\Cuidadores;
7 use Illuminate\Http\Request;
8
9 class AnimalController extends Controller
10
11   public function store(Request $request){
12     //Validación de campos
13     $request->validate([
14       'nombre' => 'required',
15       'fecha_nacimiento' => 'required|date',
16       'especie' => 'required',
17       'raza' => 'required',
18       'color_cara' => 'required',
19       'sexo' => 'required',
20       'cuidador_dni' => 'required|exists:cuidadores,dni',
21       'foto' => 'nullable|image|mimes:jpeg,png,jpg,gif|max:40960'
22     ]);
23
24     $dniCuidador = $request->input('cuidador_dni');
25
26     //Verificamos si el dni existe en la base de datos
27     $cuidador = Cuidadores::where('dni', $dniCuidador)->first();
28     if (!$cuidador) {
29       return response()->json(['message' => 'El dni no existe en la base de datos'], 400);
30     }
31
32     $data = $request ->all();
33
34     //Procesamos la foto del animal, si el cuidador adjunta una al registrarla
35     if ($request->hasFile('foto')) {
36       $file = $request->file('foto');
37       $filename = uniqid() . '.' . $file->getClientOriginalExtension();
38       $file->move(public_path('uploads/animales'), $filename);
39       $data['foto'] = $filename;
40     }
41
42     $animal = Animales::create($data);
43
44     return response()->json($animal, 201);
45   }
46 }
```

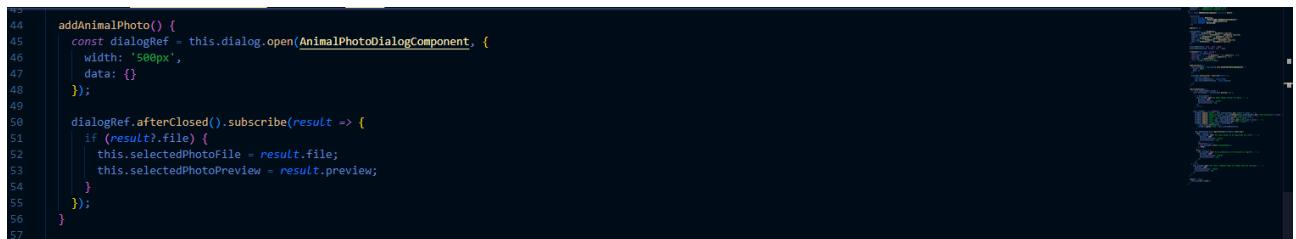
En el componente “*addNewAnimalDialog*”, este servicio es llamado para que la consulta sobre la base de datos pueda ejecutarse al evento de click, cuando el usuario rellena todos los campos y pulsa el botón “Registrar animal”.

En las siguientes imágenes se muestra el código typescript del componente *addNewAnimalDialog*, donde se realiza la llamada al servicio:



```
TS addNewAnimal.component.ts
TS consulvet-frontend > src > app > dialogs > addNewAnimalDialog > TS addNewAnimal.component.ts > AddNewAnimalComponent > constructor

1 import { Component, OnInit } from '@angular/core';
2 import { FormControl, FormGroup, Validators } from '@angular/forms';
3 import { MatSnackBar } from '@angular/material/snack-bar';
4 import { MatDialog, MatDialogRef } from '@angular/material/dialog';
5 import { AnimalPhotoDialogComponent } from '../animalPhotoDialog/animalPhotoDialog.component';
6 import { AddAnimalService } from 'src/app/services/add-animal.service';
7
8 @Component({
9   selector: 'app-addNewAnimal',
10  templateUrl: './addNewAnimal.component.html',
11  styleUrls: ['./addNewAnimal.component.css']
12 })
13 export class AddNewAnimalComponent implements OnInit {
14
15   constructor(
16     private dialog: MatDialog,
17     private dialogRef: MatDialogRef<AddNewAnimalComponent>,
18     private addAnimalService: AddAnimalService,
19     private snackBar: MatSnackBar
20   ) {}
21
22   ngOnInit() {}
23
24   newAnimalData = new FormGroup({
25     nombre: new FormControl('', Validators.required),
26     fecha_nacimiento: new FormControl('', Validators.required),
27     especie: new FormControl('', Validators.required),
28     raza: new FormControl('', Validators.required),
29     color_capa: new FormControl('', Validators.required),
30     sexo: new FormControl('', Validators.required),
31   });
32
33   selectedPhotoFile: File | null = null;
34   selectedPhotoPreview: string | null = null;
35
36   formatDate(date: any): string {
37     const d = new Date(date);
38     const month = ('' + (d.getMonth() + 1)).padStart(2, '0');
39     const day = ('' + d.getDate()).padStart(2, '0');
40     const year = d.getFullYear();
41     return `${year}-${month}-${day}`;
42   }
43
44
45   addAnimalPhoto() {
46     const dialogRef = this.dialog.open(AnimalPhotoDialogComponent, {
47       width: '500px',
48       data: {}
49     });
50
51     dialogRef.afterClosed().subscribe(result => {
52       if (result?.file) {
53         this.selectedPhotoFile = result.file;
54         this.selectedPhotoPreview = result.preview;
55       }
56     });
57   }
58 }
```



```
44
45   addAnimalPhoto() {
46     const dialogRef = this.dialog.open(AnimalPhotoDialogComponent, {
47       width: '500px',
48       data: {}
49     });
50
51     dialogRef.afterClosed().subscribe(result => {
52       if (result?.file) {
53         this.selectedPhotoFile = result.file;
54         this.selectedPhotoPreview = result.preview;
55       }
56     });
57   }
58 }
```

```

58     registerNewAnimal() {
59       if (this.newAnimalData.valid) {
60         const dniCuidador = localStorage.getItem('dni');
61
62         if (!dniCuidador) {
63           this.snackBar.open('No hemos podido cotejar tus datos', '', {
64             duration: 2000,
65             horizontalPosition: 'center',
66             verticalPosition: 'top'
67           });
68           return;
69         }
70
71         const formData = new FormData();
72         formData.append('nombre', this.newAnimalData.get('nombre')?.value || '');
73         formData.append('fecha_nacimiento', this.formatDate(this.newAnimalData.get('fecha_nacimiento')?.value || ''));
74         formData.append('especie', this.newAnimalData.get('especie')?.value || '');
75         formData.append('raza', this.newAnimalData.get('raza')?.value || '');
76         formData.append('color_capa', this.newAnimalData.get('color_capa')?.value || '');
77         formData.append('sexo', this.newAnimalData.get('sexo')?.value || '');
78         formData.append('cuidador_dni', dniCuidador);
79         if (this.selectedPhotoFile) {
80           formData.append('foto', this.selectedPhotoFile);
81         }
82
83         this.addAnimalService.registerAnimal(formData).subscribe({
84           next: response => {
85             this.snackBar.open('Tu nuevo animal se ha registrado con éxito!', '', {
86               duration: 2000,
87               horizontalPosition: 'center',
88               verticalPosition: 'top'
89             });
90             setTimeout(() => {
91               this.dialogRef.close('animalAñadido');
92             }, 2000);
93           },
94           error: error => {
95             this.snackBar.open('Se ha producido un error durante el registro', '', {
96               duration: 2000,
97               horizontalPosition: 'center',
98               verticalPosition: 'top'
99             });
100           }
101         });
102       }
103     }
104   }
105
106   cancel(): void {
107     this.dialogRef.close();
108   }
109 }
110
111 }
112
113 }
114 }
```

```

102   } else {
103     this.snackBar.open('Por favor, completa todos los campos antes de continuar', '', {
104       duration: 2000,
105       horizontalPosition: 'center',
106       verticalPosition: 'top'
107     });
108   }
109 }
110
111 cancel(): void {
112   this.dialogRef.close();
113 }
114 }
```

Para comprobar la funcionalidad, se va a presentar el estado actual de los animales registrados en la base de datos y el estado posterior a la inserción de uno nuevo. Para la inserción de un nuevo animal, se va a utilizar al usuario **Silvia123** (para comprobar la inserción, dni ficticio: 55643311A)

Estado actual de los registros de animales en la base de datos:

	codigo_paciente	nombre	fecha_nacimiento	especie	raza	color_capa	sexo	cuidador_dni	created_at	updated_at	foto
1	1	Smokey	2021-03-12	Felina	Europeo de pelo corto	Aligrado gris	M	55643311A	2025-05-19 11:10:14	2025-05-19 13:01:47	682b2bbb03935.jpg
2	2	Nioleta	2021-09-15	Felina	Mestizo de Blue British	Negra	F	55717994C	2025-05-19 11:10:14	2025-05-19 13:20:26	682b301aa0dea.jpg
3	3	Iris	2015-05-15	Felina	Europea de pelo corto	Aligrado gris	F	55643311A	2025-05-19 11:10:14	2025-05-19 13:02:27	682b2be339c73.jpg
4	4	Sally	2015-05-09	Canina	Beagle	Propia	F	55717994C	2025-05-19 11:10:14	2025-05-19 13:20:55	682b30794f5.jpg
5	5	Casper	2017-05-15	Felina	Mestizo de siamés	Aligrado crema	M	55643311A	2025-05-19 13:00:46	2025-05-19 13:00:46	682b2b7e81285.jpg
6	6	Ave	2020-11-22	Ave	Agaporni	Propia	F	55717994C	2025-05-19 13:21:49	2025-05-19 13:21:49	682b306d7422.jpg
7	7	Carlito	2025-04-30	Felina	Abisinio	Canela	M	55400871F	2025-05-19 13:28:29	2025-05-19 13:28:29	682b31fe0b08b.jpg
8	9	Lolo	2024-05-06	Felina	Maine Coon	Gris y blanca	M	55400871F	2025-05-19 13:29:47	2025-05-19 13:29:47	682b324b276aa.jpg
9	10	Boss	2015-03-24	Canina	Pastor Alemán	Negro fuego	F	66331155P	2025-05-19 13:33:18	2025-05-19 13:33:18	682b331ebcc7a.jpg
10	11	Lila	2018-08-15	Canina	Boyero de Berna	Propia	M	66331155P	2025-05-19 13:34:03	2025-05-19 13:34:03	682b334b1a1a6.jpg
11	12	Rey	2018-08-15	Canina							

Inserción de un nuevo animal para esta cuidadora:

Consulvet

- Smokey**
Felina - Europeo de pelo corto

Fecha de nacimiento: Mar 12, 2021
Color de capa: Atigrado gris
Sexo: M
- Iris**
Felina - Europea de pelo corto

Fecha de nacimiento: May 15, 2015
Color de capa: Atigrada gris
Sexo: F
- Casper**
Felina - Mestizo de siamés

Fecha de nacimiento: May 15, 2017
Color de capa: Atigrado crema
Sexo: M

Editar foto de perfil
Editar información de perfil
Añadir nuevo animal
Editar animal
Eliminar animal
Cerrar Sesión

Sobre nosotros Contacto Política de privacidad
f X in
© 2025 Consulvet. Todos los derechos reservados.

Consulvet

Añade un nuevo animal

Nombre: *	Navia
Fecha de nacimiento: *	9/24/2020
Especie: *	Felina
Raza: *	Mestiza de Maine Coon
Color de la capa: *	Carey
Sexo: *	F

Subir una foto
Registrar animal Cancelar

© 2025 Consulvet. Todos los derechos reservados.

Añade una foto de tu animal

Seleccionar foto
Puedes seleccionar cualquier foto - se redimensionará automáticamente
Cancelar Guardar foto

Añade una foto de tu animal

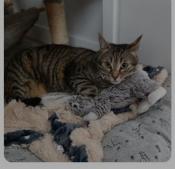
Seleccionar foto
Puedes seleccionar cualquier foto - se redimensionará automáticamente
Cancelar Guardar foto

Vista previa

Cancelar Guardar foto

¡Tu nuevo animal se ha registrado con éxito!

Smokey
Felina - Europeo de pelo corto



Fecha de nacimiento: Mar 12, 2021
Color de capa: Atigrado gris
Sexo: M

Añade un nuevo animal

Nombre: *

Fecha de nacimiento: *

Especie: *

Raza: *

Color de la capa: *

Sexo: *

Casper
Felina - Mestizo de siamés



Fecha de nacimiento: May 15, 2017
Color de capa: Atigrado crema
Sexo: M

© 2025 Consulvet. Todos los derechos reservados.

En la siguiente imagen, se muestra cómo se ha actualizado automáticamente la lista de animales para esta cuidadora:

Consulvet



Fecha de nacimiento: Mar 12, 2021
Color de capa: Atigrado gris
Sexo: M



Fecha de nacimiento: May 15, 2015
Color de capa: Atigrada gris
Sexo: F



Fecha de nacimiento: May 15, 2017
Color de capa: Atigrado crema
Sexo: M

Navia
Felina - Mestiza de Maine Coon



Fecha de nacimiento: Sep 24, 2020
Color de capa: Carey

Editar foto de perfil
Editar información de perfil
Añadir nuevo animal
Editar animal
Eliminar animal
Cerrar Sesión

Y en la siguiente imagen, se muestra el estado actualizado, después de la inserción, de los animales registrados en la tabla Animales de la base de datos:

	codigo_paciente	nombre	fecha_nacimiento	especie	raza	color_capa	sexo	cuidador_dni	created_at	updated_at	foto
1	Smokey	2021-03-12	Felina	Europeo de pelo corto	Aligrado gris	M	55643311A	2025-05-19 11:10:14	2025-05-19 13:01:47	662b2bbb03935.jpg	
2	Niobe	2021-09-15	Felina	Mestiza de Blue British	Negra	F	557179940	2025-05-19 11:10:14	2025-05-19 13:20:26	682b301aa0dea.jpg	
3	Iris	2015-05-15	Felina	Europea de pelo corto	Aligrada gris	F	55643311A	2025-05-19 11:10:14	2025-05-19 13:02:27	682b2b03b39e73.jpg	
4	Sally	2015-05-09	Canina	Beagle	Propia	F	557179940	2025-05-19 11:10:14	2025-05-19 13:20:55	682b30379455.jpg	
5	Casper	2017-05-15	Felina	Mestizo de siamés	Aligrado crema	M	55643311A	2025-05-19 13:00:04	2025-05-19 13:00:46	682b2b7e81285.jpg	
6	Carlota	2020-11-22	Ave	Agaporni	Propia	F	557179940	2025-05-19 13:21:49	2025-05-19 13:21:49	682b316de0b2b.jpg	
7	Lolo	2025-04-30	Felina	Abisinio	Canela	M	55400871F	2025-05-19 13:28:29	2025-05-19 13:28:29	682b31fde0b2b.jpg	
8	Boss	2024-05-06	Felina	Maline Coon	Gris y blanca	M	55400871F	2025-05-19 13:29:47	2025-05-19 13:29:47	682b324b276aa.jpg	
9	Lía	2015-03-24	Canina	Pastor Alemán	Negro fuego	F	68331155P	2025-05-19 13:33:18	2025-05-19 13:33:18	682b3131ebca7.jpg	
10	Roy	2018-08-15	Canina	Boyero de Berna	Propia	M	68331155P	2025-05-19 13:34:03	2025-05-19 13:34:03	682b334b1a1a6.jpg	
11	Navia	2020-09-24	Felina	Mesíza de Maine Coon	Carey	F	55643311A	2025-05-20 18:50:42	2025-05-20 18:50:42	55643311A.jpg	

➤ Creación de un modal:

Los “modales” son ventanas emergentes que se abren sobre el contenido principal de la aplicación web. Se suelen implementar como componentes independientes que se muestran de forma condicional (*ngIf), y generalmente se aplican para mostrar información, presentar formularios al usuario o incluso pedir confirmaciones al mismo, sin necesidad de navegar a otra página (similar a los cuadros de diálogo).

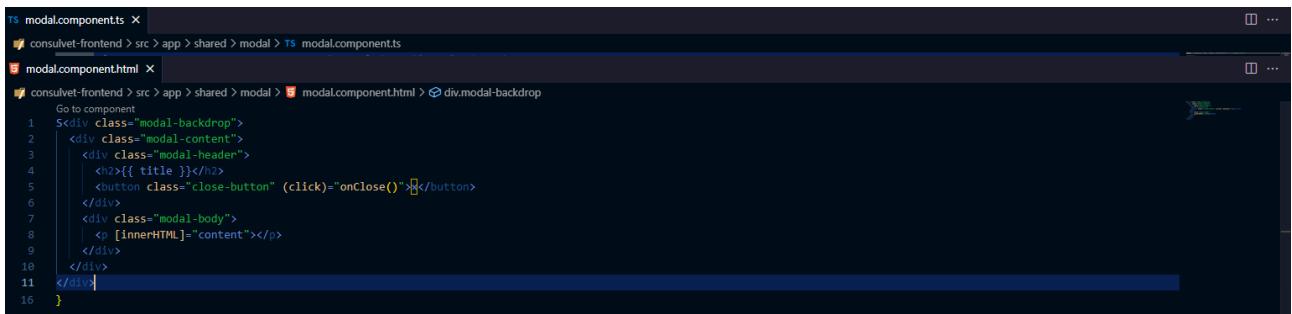
Cuando un modal se abre, bloquea la interacción con el resto del contenido de la aplicación, el cual “queda al fondo”.

En el caso de Consulvet, se ha implementado un modal para mostrar la información del pie de página; concretamente, la información: “Sobre nosotros”, “Contacto” y “Política de privacidad”.

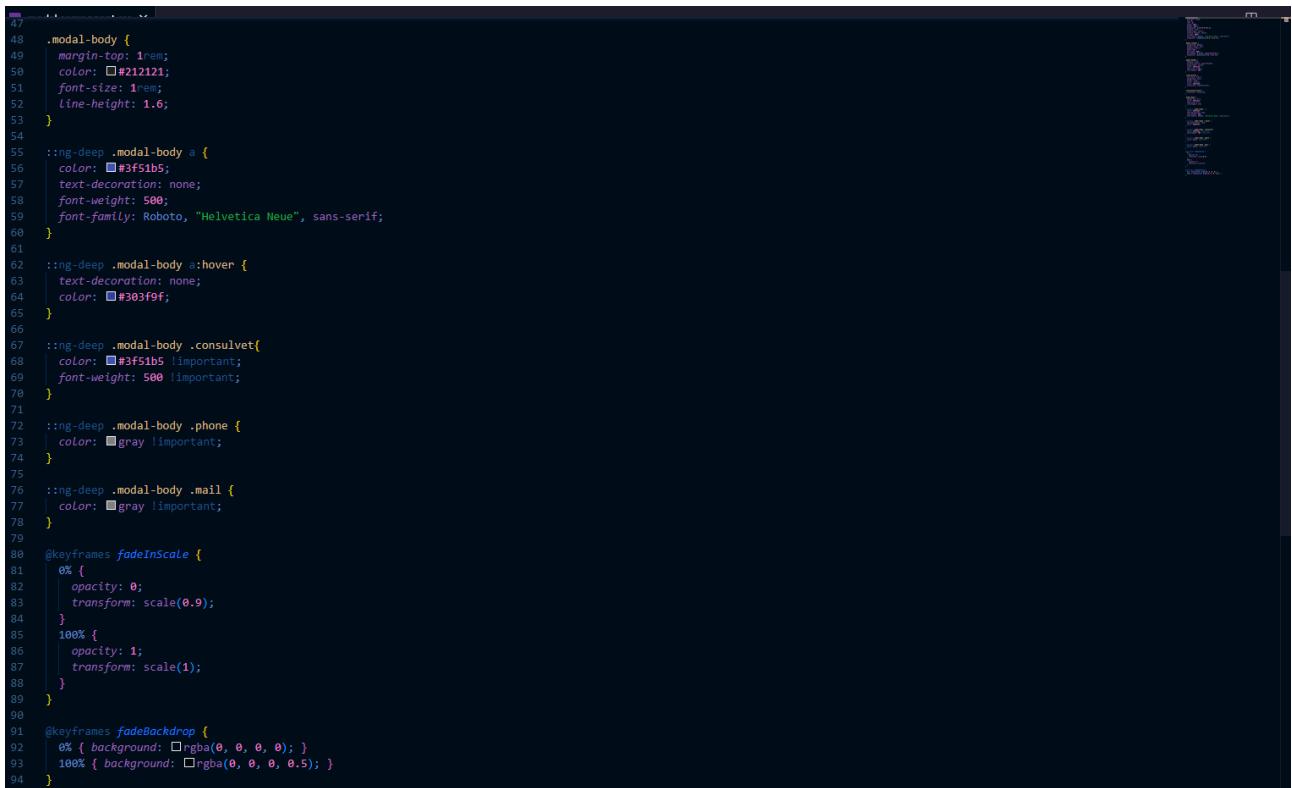
El componente modal, como hemos dicho, se crea como un componente aparte y es llamado desde el componente donde se va a ejecutar la lógica para abrirlo al evento de click. Dado que responde a condiciones, el mismo modal se presenta tres veces en Consulvet: una para cada información mencionada. Desde el componente que llama al modal, se declaran tres variables: una por información; y se crea asociado a cada una de ellas la plantilla html que se va a mostrar, en función de la información pertinente a cada caso.

En las siguientes imágenes se muestra el modal implementado, las plantillas html declaradas en el archivo typescript del componente Profile, y la llamada al modal con condiciones desde el archivo html del componente Profile.

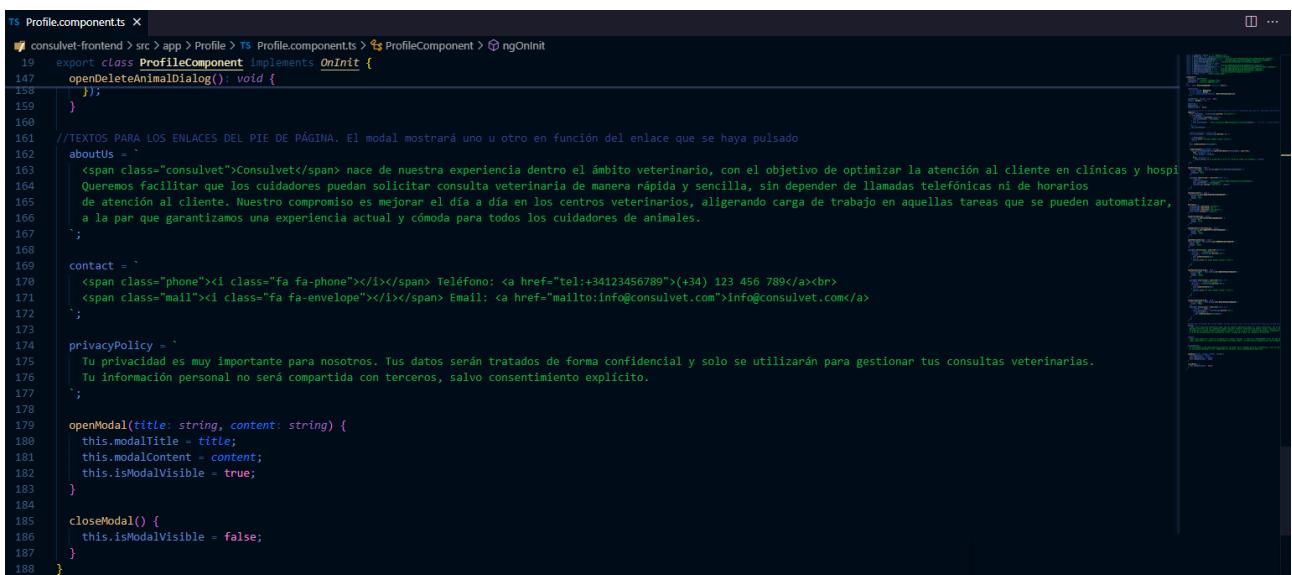
*Nota: los estilos para el modal se aplican directamente en el archivo css del componente modal. Aunque las plantillas html a mostrar estén declaradas en el componente Profile, sus estilos deben aplicarse en el modal. Dado que Angular Material tiene unos estilos predeterminados fuertes, en muchas ocasiones no permite ciertos estilos aplicados salvo forzándolos. Por ejemplo, en el caso de los iconos de contacto, se ha dado una clase a cada ícono, y esta clase ha sido utilizada en el archivo de estilos css del modal, forzando los mismos aplicando “!important”. Esta práctica, junto con el uso de “::ng-deep”, es común cuando estamos trabajando en Angular y queremos modificar alguno de los estilos predeterminados de Material.



```
ts modal.component.ts
consulvet-frontend > src > app > shared > modal > ts modal.component.ts
modal.component.html
consulvet-frontend > src > app > shared > modal > modal.component.html > div.modal-backdrop
Go to component
1 5<div class="modal-backdrop">
2 | <div class="modal-content">
3 |   <div class="modal-header">
4 |     <h2>{{ title }}</h2>
5 |     <button class="close-button" (click)="onClose()">&times;</button>
6 |   </div>
7 |   <div class="modal-body">
8 |     <p [innerHTML]="content"></p>
9 |   </div>
10 | </div>
11 </div>
12 }
```



```
47 .modal-body {
48   margin-top: 1rem;
49   color: #212121;
50   font-size: 1rem;
51   line-height: 1.6;
52 }
53 }
54 ::ng-deep .modal-body a {
55   color: #3f51b5;
56   text-decoration: none;
57   font-weight: 500;
58   font-family: Roboto, "Helvetica Neue", sans-serif;
59 }
60 }
61 ::ng-deep .modal-body a:hover {
62   text-decoration: none;
63   color: #3039ff;
64 }
65 }
66 ::ng-deep .modal-body .consulvet {
67   color: #3f51b5 !important;
68   font-weight: 500 !important;
69 }
70 }
71 ::ng-deep .modal-body .phone {
72   color: #gray !important;
73 }
74 }
75 ::ng-deep .modal-body .mail {
76   color: #gray !important;
77 }
78 }
79 @keyframes fadeInScale {
80   0% {
81     opacity: 0;
82     transform: scale(0.9);
83   }
84   100% {
85     opacity: 1;
86     transform: scale(1);
87   }
88 }
89 }
90 @keyframes fadeBackdrop {
91   0% { background: rgba(0, 0, 0, 0); }
92   100% { background: rgba(0, 0, 0, 0.5); }
93 }
94 }
```

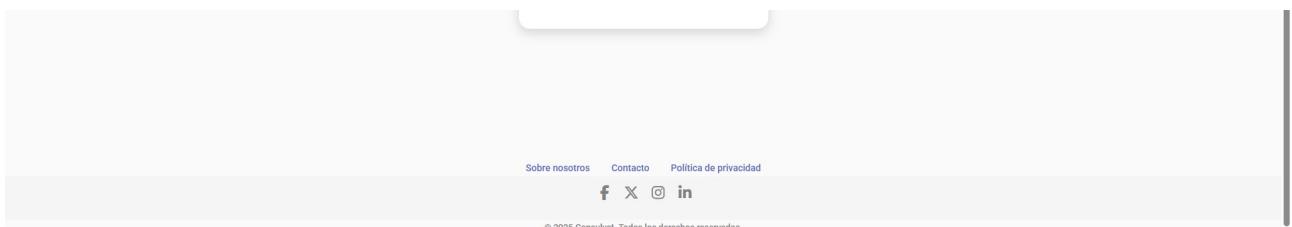


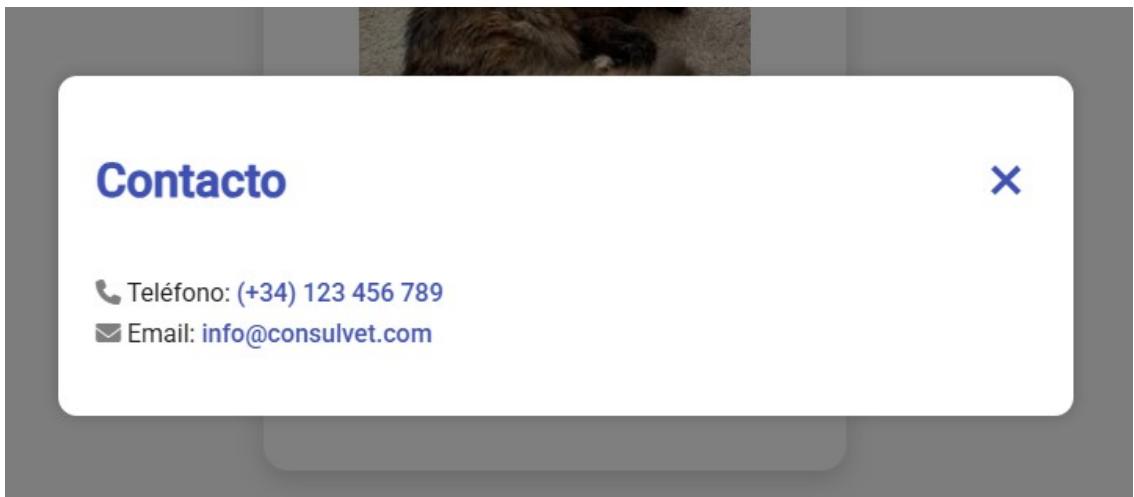
```
ts Profile.component.ts
consulvet-frontend > src > app > Profile > ts Profile.component.ts > ProfileComponent > ngOnInit
19 export class ProfileComponent implements OnInit {
147   openDeleteAnimalDialog(): void {
158   }
159 }
160
161 //TEXTOS PARA LOS ENLACES DEL PIE DE PÁGINA. El modal mostrará uno u otro en función del enlace que se haya pulsado
162 aboutUs =
163   <span class="consulvet">Consulvet</span> nace de nuestra experiencia dentro el ámbito veterinario, con el objetivo de optimizar la atención al cliente en clínicas y hospitales. Queremos facilitar que los cuidadores puedan solicitar consulta veterinaria de manera rápida y sencilla, sin depender de llamadas telefónicas ni de horarios de atención al cliente. Nuestro compromiso es mejorar el día a día en los centros veterinarios, aligerar carga de trabajo en aquellas tareas que se pueden automatizar, a la par que garantizamos una experiencia actual y cómoda para todos los cuidadores de animales.
164
165 contact =
166   <span class="phone"><i class="fa fa-phone"></i></span> Teléfono: <a href="tel:+34123456789">(+34) 123 456 789</a><br>
167   <span class="mail"><i class="fa fa-envelope"></i></span> Email: <a href="mailto:info@consulvet.com">info@consulvet.com</a>
168
169 privacyPolicy =
170   Tu privacidad es muy importante para nosotros. Tus datos serán tratados de forma confidencial y solo se utilizarán para gestionar tus consultas veterinarias.
171   Tu información personal no será compartida con terceros, salvo consentimiento explícito.
172
173
174 openModal(title: string, content: string) {
175   this.modalTitle = title;
176   this.modalContent = content;
177   this.isModalVisible = true;
178 }
179
180 closeModal() {
181   this.isModalVisible = false;
182 }
183 }
```

```

Profile.component.html ✘
consulvet-frontend > src > app > Profile > Profile.component.html > mat-sidenav-container > mat-drawer-container.example-container > mat-drawer-content > mat-toolbar.footer > div.social-icons > a
  1  <div class="sidenav-container">
  2    <mat-drawer-container class="example-container">
  3      <mat-drawer-content>
  4
  5        <mat-toolbar color="primary" class="footer" fxLayout="column" fxLayoutAlign="center center">
  6          <div fxLayout="row" fxLayoutAlign="center center" class="footer-links" fxLayoutGap="32px">
  7            <a class="footer-link" (click)="openModal('Sobre nosotros', aboutUs)">>Sobre nosotros</a>
  8            <a class="footer-link" (click)="openModal('Contacto', contact)">>Contacto</a>
  9            <a class="footer-link" (click)="openModal('Política de privacidad', privacyPolicy)">>Política de privacidad</a>
 10        </div>
 11        <div fxLayout="row" fxLayoutAlign="center center" class="social-icons">
 12          <a mat-icon-button href="https://www.facebook.com" target="_blank" rel="noopener" aria-label="Facebook">
 13            <mat-icon fontSet="fa-brands" fontIcon="fa-facebook-f" class="footer-icon"></mat-icon>
 14          </a>
 15          <a mat-icon-button href="https://www.twitter.com" target="_blank" rel="noopener" aria-label="Twitter">
 16            <mat-icon fontSet="fa-brands" fontIcon="fa-x-twitter" class="footer-icon"></mat-icon>
 17          </a>
 18          <a mat-icon-button href="https://www.instagram.com" target="_blank" rel="noopener" aria-label="Instagram">
 19            <mat-icon fontSet="fa-brands" fontIcon="fa-instagram" class="footer-icon"></mat-icon>
 20          </a>
 21          <a mat-icon-button href="https://www.linkedin.com" target="_blank" rel="noopener" aria-label="LinkedIn">
 22            <mat-icon fontSet="fa-brands" fontIcon="fa-linkedin-in" class="footer-icon"></mat-icon>
 23          </a>
 24        </div>
 25        <span class="footer-text">© 2025 Consulvet. Todos los derechos reservados.</span>
 26      </mat-toolbar>
 27
 28      <app-modal
 29        *ngIf="isModalVisible"
 30        [title]="modalTitle"
 31        [content]="modalContent"
 32        (close)="closeModal()">
 33      </app-modal>
 34    </mat-drawer-content>
 35  </mat-drawer-container>
 36
```

En las siguientes imágenes, a modo de ejemplo, se muestra el modal abierto para el apartado del pie de página “Sobre nosotros” y para el apartado “Contacto”.





*Nota: obsérvese como el componente es el mismo, y sólo cambia el contenido que muestra en función de la plantilla que se está utilizando para cada caso