Universidad Nacional de Quilmes

tecnicatura en programación informática

# PET HEROES

Trabajo de Inserción Profesional

Analía Redonda, Pablo Sabaliauskas

Profesores

Fernando Dodino, Susana Rosito


Quilmes, 2019

# Índice general

# Índice de figuras

# Capítulo 1

# Introducción

## 1.1. ¿Qué es Pet Heroes?

Pet Heroes es una aplicación orientada a mejorar el cuidado de las mascotas. A través de esta plataforma, es posible llevar un control más cercano de las atenciones médicas que reciben, las vacunas que se les aplicaron y calendarizar futuras aplicaciones. Por otro lado, la plataforma pone a disposición la posibilidad de seguir en tiempo real a las mascotas, y así poder saber dónde se encuentran en todo momento. Pet Heroes es además una red social, que permite al usuario ponerse en contacto con amigos que estén registrados en la plataforma. Proximamente, a través de esta red social se podrán compartir fotos, actividades y eventos relacionados con las mascotas.

## 1.2. Necesidad de Pet Heroes

Pet Heroes permite acceder desde la Web a la historia clínica de las mascotas siempre que se lo necesite, y en un formato libre de libretas y papeles innecesarios o que tienden a perderse.

## 1.3. Objetivo Principal de Pet Heroes

Que los dueños de mascotas puedan tener acceso inmediato al historial médico de sus mascotas, desde su celular y/o cualquier otro dispositivo con acceso a Internet.

## 1.4. ¿Cómo instalar el entorno?

### 1.4.1. Requisitos

- NodeJS
- MongoDb
- Neo4J

### 1.4.2. Instalación

#### Front-End (Web)

Para clonar el proyecto del repositorio Git:

```
$ git clone https://github.com/RedondaAnalia/TIP-front.git
```

Para instalar las dependencias necesarias:

```
$ cd TIP-front
$ npm install
```

Para levantar el servicio:

```
$ ng serve
```

Abrir el navegador de preferencia en la URL: localhost:4200

#### Back-End

Para clonar el proyecto desde el repositorio Git:

```
$ git clone https://github.com/RedondaAnalia/TIP-core.git
```

Para instalar las dependencias necesarias:

```
$ cd TIP-core
$ npm install
```

Para levantar el servicio:

```
$ npm start
```

 NO OLVIDAR ACTUALIZAR LAS VARIABLES DE ENTORNO

#### Micro-servicios

El Back-End requiere un micro-servicio para la gestión de la red de amistades entre usuarios de la plataforma, el cual se encuentra en otro repositorio. Para descargarlo:

```
$ git clone https://github.com/wisaku/TIP-friendship.git
```

Para instalar las dependencias necesarias:

```
$ cd TIP-friendship
$ npm install
```

Para levantar el servicio:

```
$ npm start
```

# Capítulo 2

# Prueba de concepto

Para la prueba de concepto se levanta la app de forma local. Allí puede ingresarse el código asociado a una mascota (ID), y el sistema responde mostrando cuáles son las vacunas que esa mascota tiene aplicadas y cuáles calendarizadas (pendientes de aplicación). La app permite cambiar el estado de una vacuna de "pendiente de aplicación" a "aplicada".

## 2.1. Arquitectura

### 2.1.1. Tecnologías utilizadas

Ver Figura 2.1

#### Front-End

Se desarrolló una aplicación utilizando Angular 5, Bootstrap y Express para el servidor web.

#### Back-End

Se desarrolló una aplicación utilizando Node.js. La misma otorga servicios via API-Rest en http://localhost:3000/

#### Persistencia

Se utiliza MongoDB para persistencia de datos.

### 2.1.2. Casos de uso

- **Buscar mascota:** el veterinario ingresa el ID de la mascota en el input de la pantalla principal. Al hacer enter es direccionado al perfil de la mascota en cuestión.

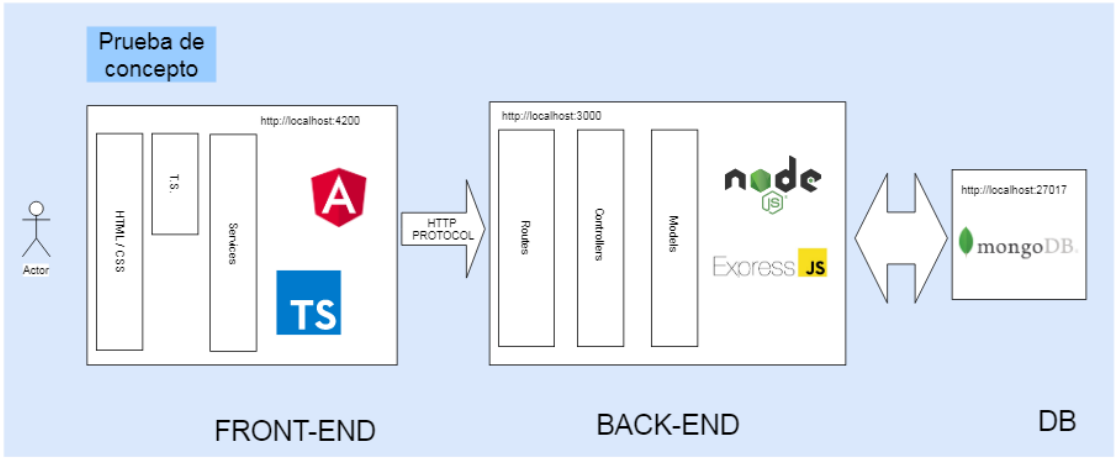



Figura 2.1: Arquitectura de POC

- **Ver Mascota:** el veterinario realiza los pasos del caso de uso "Buscar mascota"donde encuentra los datos de la mascota y sus vacunas (aplicaciones pendientes, por vencer y aplicadas).
- **Aplicar vacuna:** el veterinario realiza los pasos del caso de uso "Ver mascota", y luego se dirige a la solapa "vacunas", selecciona una vacuna pendiente, y toca en el botón .ªplicar", que indica que la vacuna fue aplicada en ese día. Luego puede visualizar que cambia de estado "pendiente de aplicación.ª .ªplicada".

Ver Figura 2.2

### 2.1.3. Endpoints

Ver Tabla 2.1

| | Endpoints POC | | |
|---|---|---|---|
| | Method | URI | Body |
| Get pets by ID | GET | pets/id | – |
| Mark application as Done | POST | applications | { "application_id": String, "application_date": Date } |

Tabla 2.1: Endpoints POC

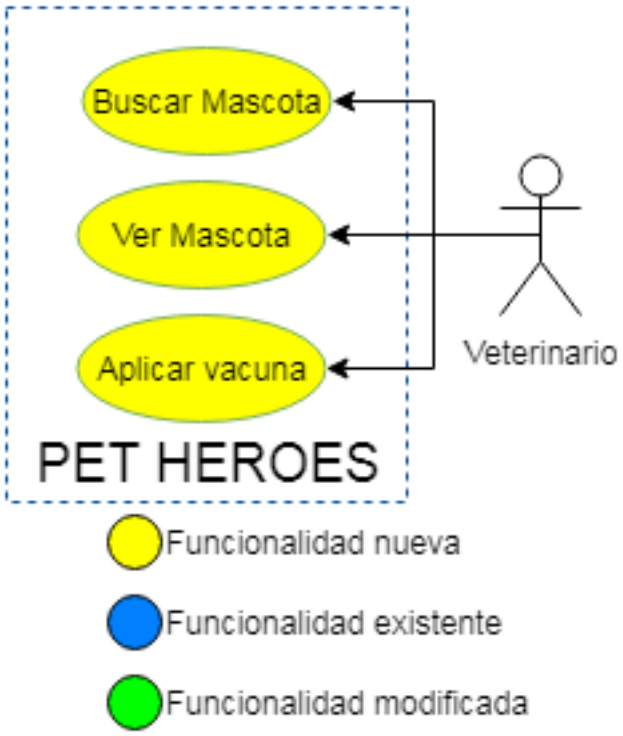



Figura 2.2: Casos de uso POC

# Capítulo 3

# Entregas

## 3.1. Entrega 1

En esta entrega se suman funcionalidades tanto en Front-End como en Back-End. Para empezar, se diferencian dos roles en la app: VETERINARIO y USUARIO. Se incorpora un login para determinar el rol y los permisos del usuario que va a hacer uso de la app.

- Para el rol VETERINARIO se incorpora la capacidad de dar de alta a nuevas atenciones médicas y calendarizar nuevas aplicaciones de vacunas en el historial clínico de una mascota.
- Para el rol USUARIO se crea un layout desde cero, con funcionalidades acordes a los permisos de usuario. El USUARIO puede ver el historial clínico de su mascota, las vacunas aplicadas, las que faltan aplicar y la fecha óptima de aplicación.

### 3.1.1. Arquitectura

#### Front-End

Se agrega Angular Material para el diseño de la Aplicación Web. Se agrega CI (Travis) y se sube la Aplicación a la nube (Heroku).

#### Back-End

En Back-End se agrega CI (Travis) y se sube a la nube (Heroku).

#### Persistencia

Se sube a la nube la base de datos MongoDB
Ver Figura 3.1

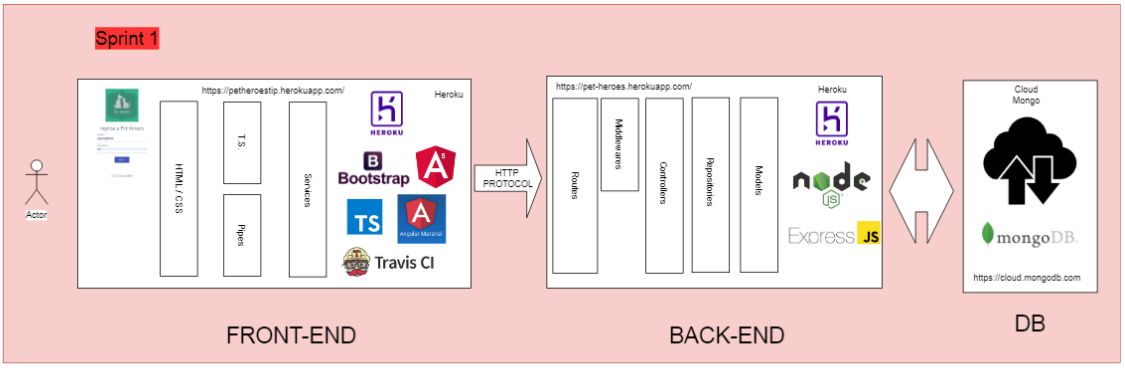



Figura 3.1: Arquitectura Sprint 1

### 3.1.2. Casos de uso

- **Loguearse:** el USUARIO o el VETERINARIO, para ingresar al sistema, deben loguearse. Para ello, en la primera pantalla aparecen dos input: en el primero ingresa su nombre de usuario (email) y en el segundo ingresa su contraseña (con un mínimo de 5 caracteres). Luego hace click en el botón Ïngresar"; en caso de ingresar un usuario y contraseña correctos, se ingresará al sistema, y en caso contrario, figurará el error correspondiente.

- **Ver Mascotas:** el USUARIO, luego de loguearse, ve la lista de sus mascotas.

- **Ver historia clínica:** el USUARIO realiza el caso de uso "Ver Mascotas", hace click sobre una de ellas, elige la solapa "historia clinicaz es dirigido a la lista de historias clínicas de esa mascota.

- **Ver Vacunas:** El USUARIO realiza el caso de uso "Ver Mascotas", hace click sobre una de ellas, elige la solapa "vacunasz allí puede visualizar la lista de vacunas aplicadas, pendientes, por vencer y vencidas de la mascota.

- **Buscar Mascota (mejora):** el VETERINARIO, luego de loguearse (Ver caso de uso "Loguearse") ingresa el mail del USUARIO de interés, hace enter, y es dirigido a una pantalla donde elige a una mascota de la lista de mascotas asociadas al USUARIO ingresado.

- **Crear aplicaciones:** el veterinario realiza el caso de uso "Buscar Mascotaz luego elige una de ellas haciendo click. Se dirige a la sopala "vacunas", y en la parte superior izquierda de la pantalla toca el botón (+) que lo dirige a un formulario en el cual puede elegir el tipo de vacuna y la fecha de aplicación. Luego de hacer click en çrear aplicación", la aplicación es agregada a la lista de aplicaciones como "pendiente.º "próxima a vencer".

- **Alta historia clínica:** el veterinario realiza el caso de uso "Buscar Mascotaz luego elige una de ellas haciendo click. Se dirige a la solapa "Historias clínicas",

y en la parte superior izquierda de la pantalla toca el botón (+) que lo dirige a un formulario en el cual puede ingresar el título de la historia clínica, y el diagnóstico de la mascota. Luego, haciendo click en çrear historia clínica", la historia clínica es agregada a la lista de historias clínicas de la mascota. Allí figuran los datos ingresados (diagnótico, etc.) el nombre del veterinario y la fecha de la historia clínica realizada.

Ver figura 3.2

### 3.1.3. Endpoints

| | Endpoints Sprint 1 | | |
|---|---|---|---|
| | Method | URI | Body |
| Get vaccines | GET | /vaccine | – |
| Add new application | POST | /applications?token=<TOKEN> | { 'pet_id' : String, 'vaccine_id': String, 'estimated_date': Date, 'email': Date } |
| Mark application as Done | PUT | /applications?token=<TOKEN> | { application_id: String, application_date: Date, email: String }; |
| Add new medical attention | POST | /pets/medicalCard?token=<TOKEN> | { pet_id: String, email: String, medicalCard: { title: String, diagnostic: String, veterinary: String } } |
| Find User's Pets | GET | /users/<mail> | — |
| User Login | POST | /login | { email : String, password : String } |

Tabla 3.1: Endpoints Sprint 1

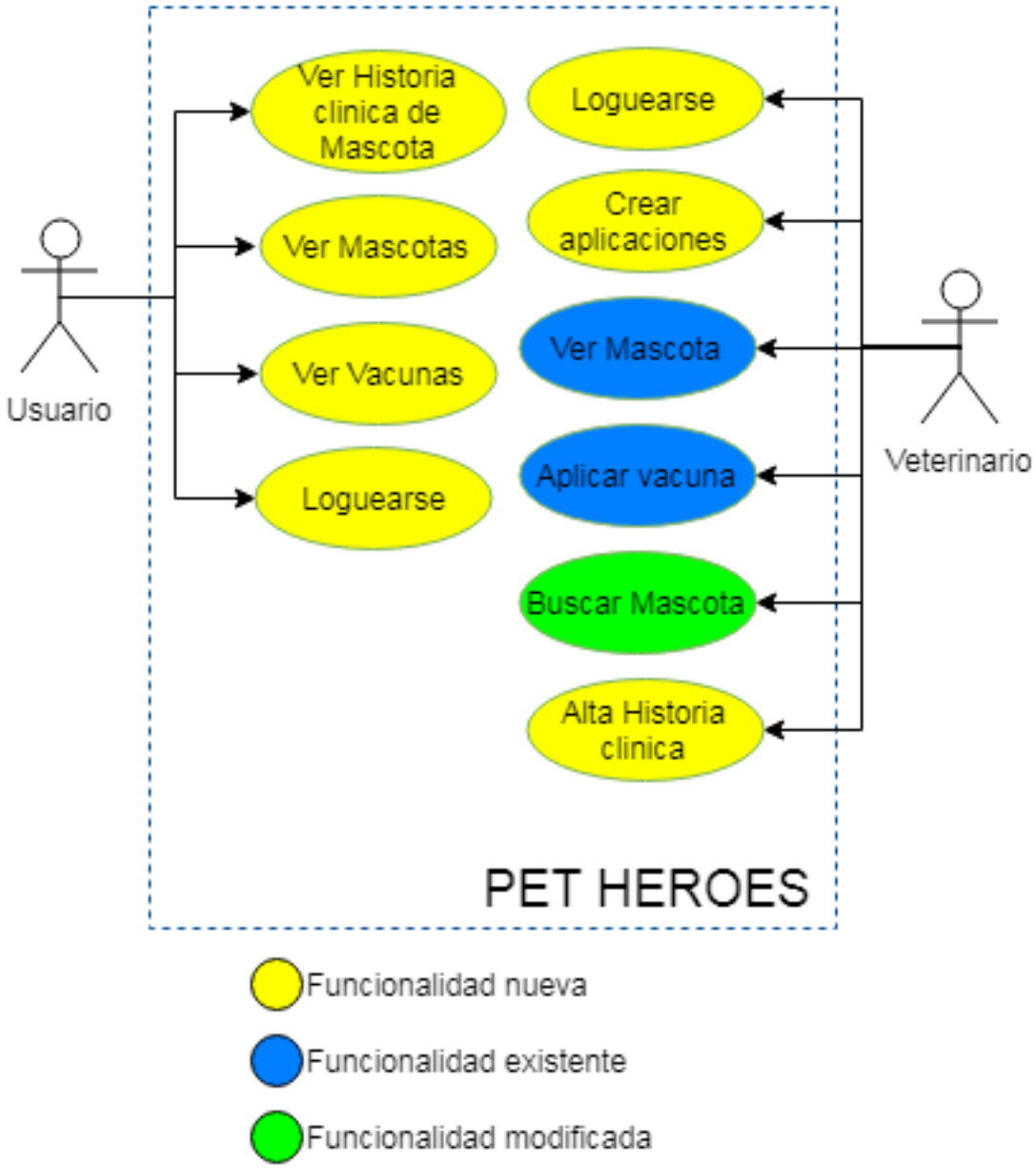



Figura 3.2: Casos de uso Sprint 1

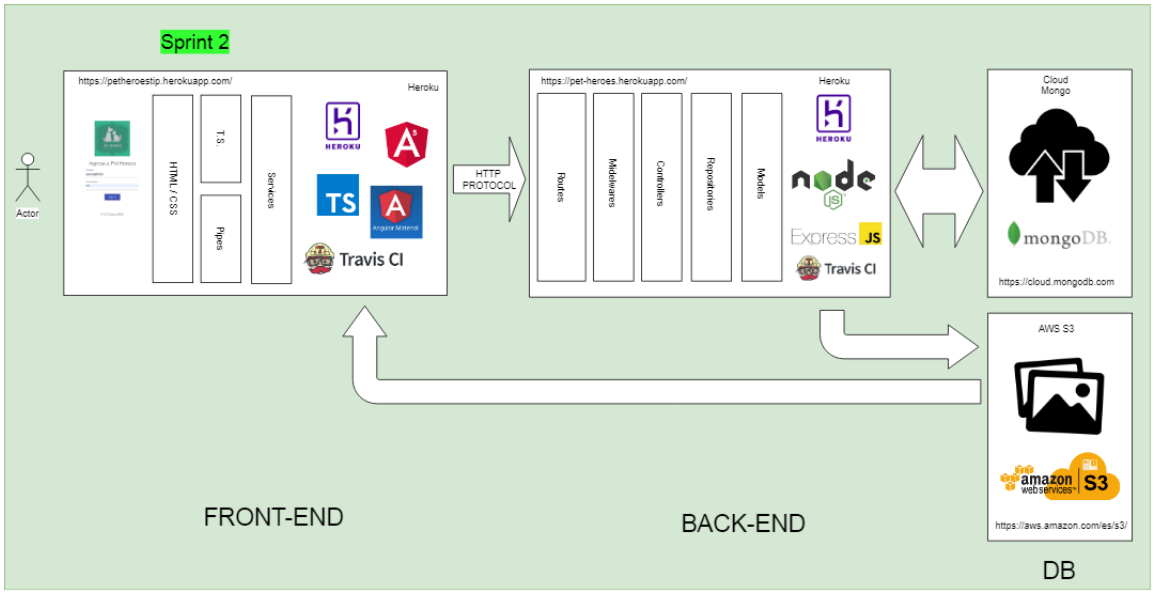



Figura 3.3: Arquitectura Sprint 2

## 3.2. Entrega 2

En esta entrega trabajan a la par el Back-End y el Front-End.

- El USUARIO puede cargar nuevas mascotas.
- El USUARIO obtiene logros y puntos por vacunar a sus mascotas en tiempo y forma.
- El USUARIO puede cambiar su foto de perfil y la foto de sus mascotas registradas.

### 3.2.1. Arquitectura

#### Front-End

Se migró a una única herramienta de diseño de aplicaciones: se pasó de utilizar Bootstrap y Angular Material a solo utilizar Angular Material.

#### Back-End

Se agregó Amazon S3 como servicio de almacenamiento para guardar las imágenes de usuario y mascotas.

#### Persistencia

Se siguen persistiendo datos en la nube de MongoDB.
Ver Figura 3.3

### 3.2.2. Casos de uso

- **Alta mascota:** el USUARIO, luego de loguearse, ingresa al sidebar y hace click sobre "Mascotas". Allí aparecen dos opciones, hace click sobre .ªgregar una mascotaz se abre un formulario en el que ingresa: el nombre de la mascota, la fecha de nacimiento, si está castrado y el género. El alta de una nueva mascota se hace efectiva al hacer click en el botón .ªgregar nueva mascota!"
- **Ver logros:** el USUARIO, luego de loguearse, ingresa al sidebar y hace click sobre "Logros". Allí hace click sobre "Ver mis logros", lo que muestra todos los logros obtenidos por el USUARIO, como por ejemplo aplicar una vacuna en tiempo y forma.
- **Cambiar imagen de perfil:** el USUARIO, luego de loguearse, ingresa al sidebar y chace lick sobre su imagen de perfil. Se abre una ventana que permite seleccionar una imagen en formato png o jpg desde el ordenador, haciendo click en "seleccionar archivo". Para otros formatos de imagen indicará .error de formato". Haciendo click en Çambiar foto!"la foto de perfil actual será reemplazada por la elegida.
- **Cambiar imagen de mascota:** El USUARIO, luego de loguearse, ingresa al sidebar y hace click sobre "Mascotas". Luego en "Ver mis mascotas". Dentro del perfil de una mascota ve su foto, y haciendo click sobre ella se abre una ventana que permite seleccionar una imagen desde el ordenador haciendo click en "seleccionar archivo". Las especificaciones del formato de la imagen son las mismas que para la foto de perfil del USUARIO. Haciendo click en Çambiar foto!"la foto de la mascota es reemplazada por la elegida.

Ver figura 3.4

### 3.2.3. Endpoints

## 3.3. Entrega 3

La aplicación empieza a ser un poco más interactiva, a ir más allá de datos e interactúa con las acciones que se registran en ella.

- El USUARIO sube de nivel a medida que logra puntos de experiencia
- El USUARIO puede encontrar las veterinarias cercanas
- El VETERINARIO puede obtener un PDF con todas las historias clínicas de una mascota

### 3.3.1. Arquitectura

#### Front-End

Se utilizan servicios de Google Maps

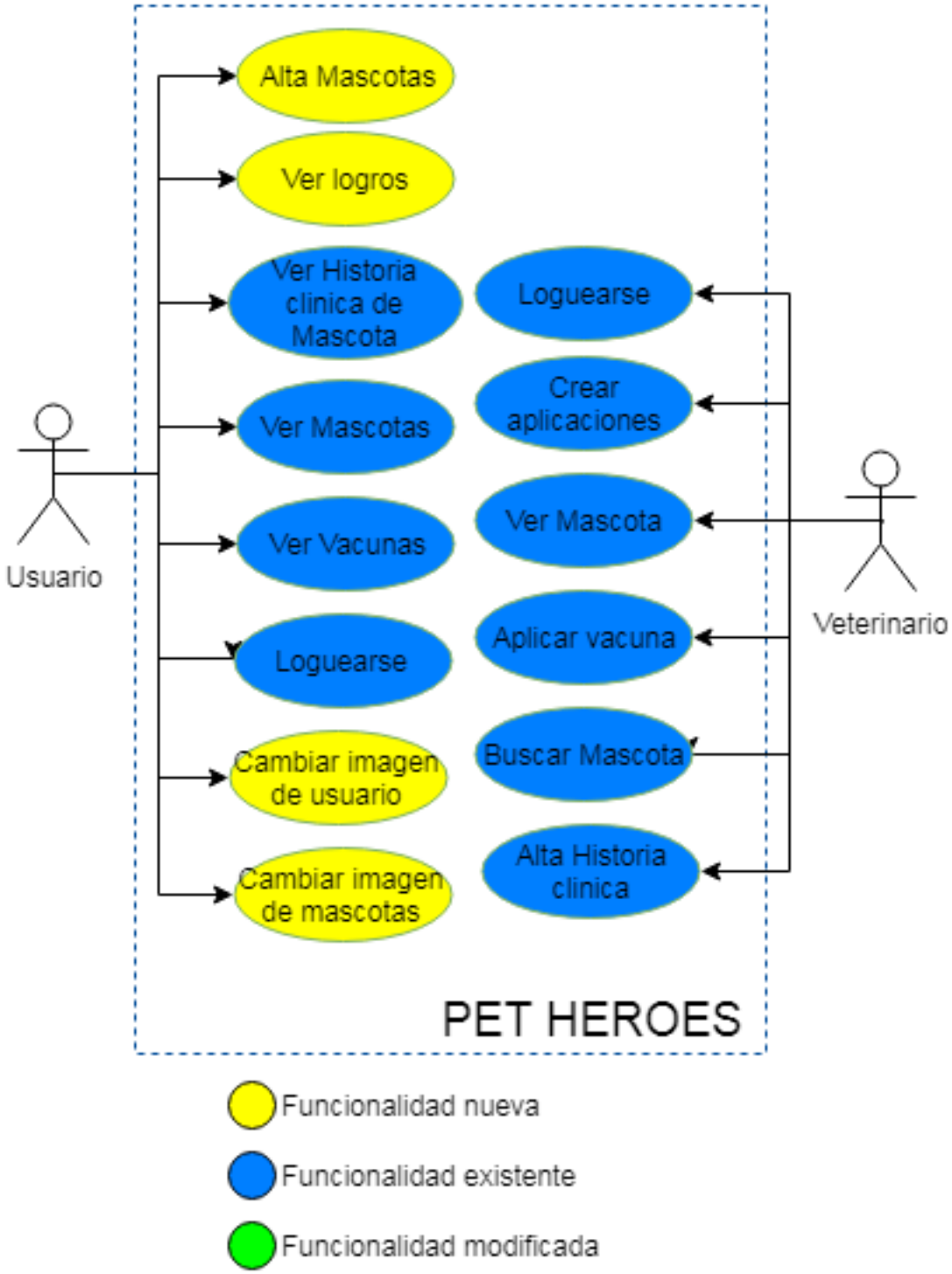



Figura 3.4: Casos de uso Sprint 2

| | Endpoints Sprint 1 | | |
|---|---|---|---|
| | Method | URI | Body |
| Change Pet Image Profile | PUT | pets/image | `{ id: String, image: File };` |
| Add Pet | POST | users/pet?token=<TOKEN> | `{ user_id: String, pet : { name : String, date_of_birth : Date, castrate: Boolean, gender: String } };` |
| Change User Image Profile | PUT | users/image | `{ id: String, image: File };` |

Tabla 3.2: Endpoints Sprint 2

**Back-End**

En Back-End se estableció un micro-servicio para el manejo de relaciones entre amigos (Pet Heroes - FriendShip),

**Persistencia**

Se continúa persistiendo datos en la nube de MongoDB para el core. Se agrega una base de datos Neo4J para el micro-servicio de relaciones entre amigos.

Ver Figura 3.5

### 3.3.2. Casos de uso

- **Ver experiencia y nivel:** el USUARIO se loguea y en el sidebar puede visualizar su experiencia y nivel, cerca de su foto de perfil.
- **Ver veterinarias cercanas:** el USUARIO se loguea y en el sidebar hace click sobre "Veterinarias cercanas", y luego sobre "Ver veterinarias cercanas". Allí visualizará su ubicación en el mapa y los markers de las veterinarias cercanas.
- **Descargar PDF de historial clínico:** El VETERINARIO busca a una mascota, hace click en la solapa "historia clínica", y luego hace click en el boton "Descargar PDF"

Ver figura 3.6

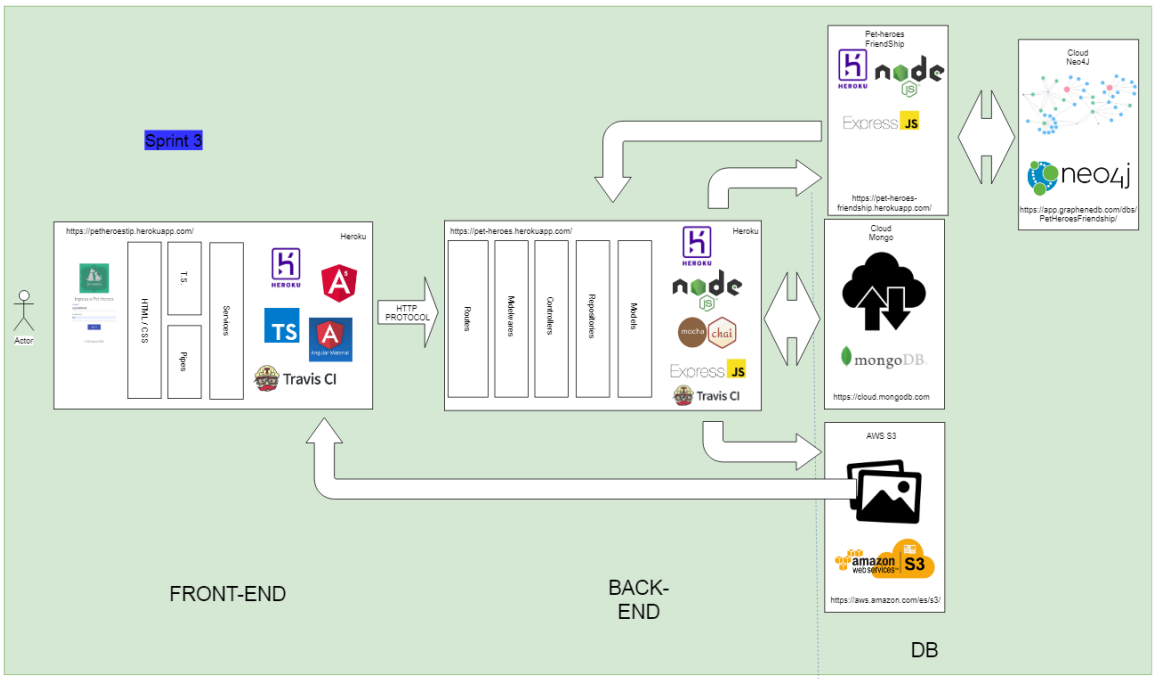



Figura 3.5: Arquitectura Sprint 3

### 3.3.3. Endpoints

No hay endpoints nuevos en esta entrega.

## 3.4. Entrega 4

Se inicia el rumbo hacia convertir la app en una Red Social para incentivar su uso.

- El USUARIO puede crear un perfil.
- El USUARIO puede agregar amigos.
- El USUARIO puede ver sus amigos.
- El USUARIO puede buscar a otros usuarios para agregarlos como amigos.
- Mejoras en la búsqueda de usuarios para el VETERINARIO.

### 3.4.1. Arquitectura

#### Front-End

Se agrega conexión a BD Firebase para mostrar ubicación actual de una mascota.

#### Back-End

Sin cambios a nivel arquitectura.

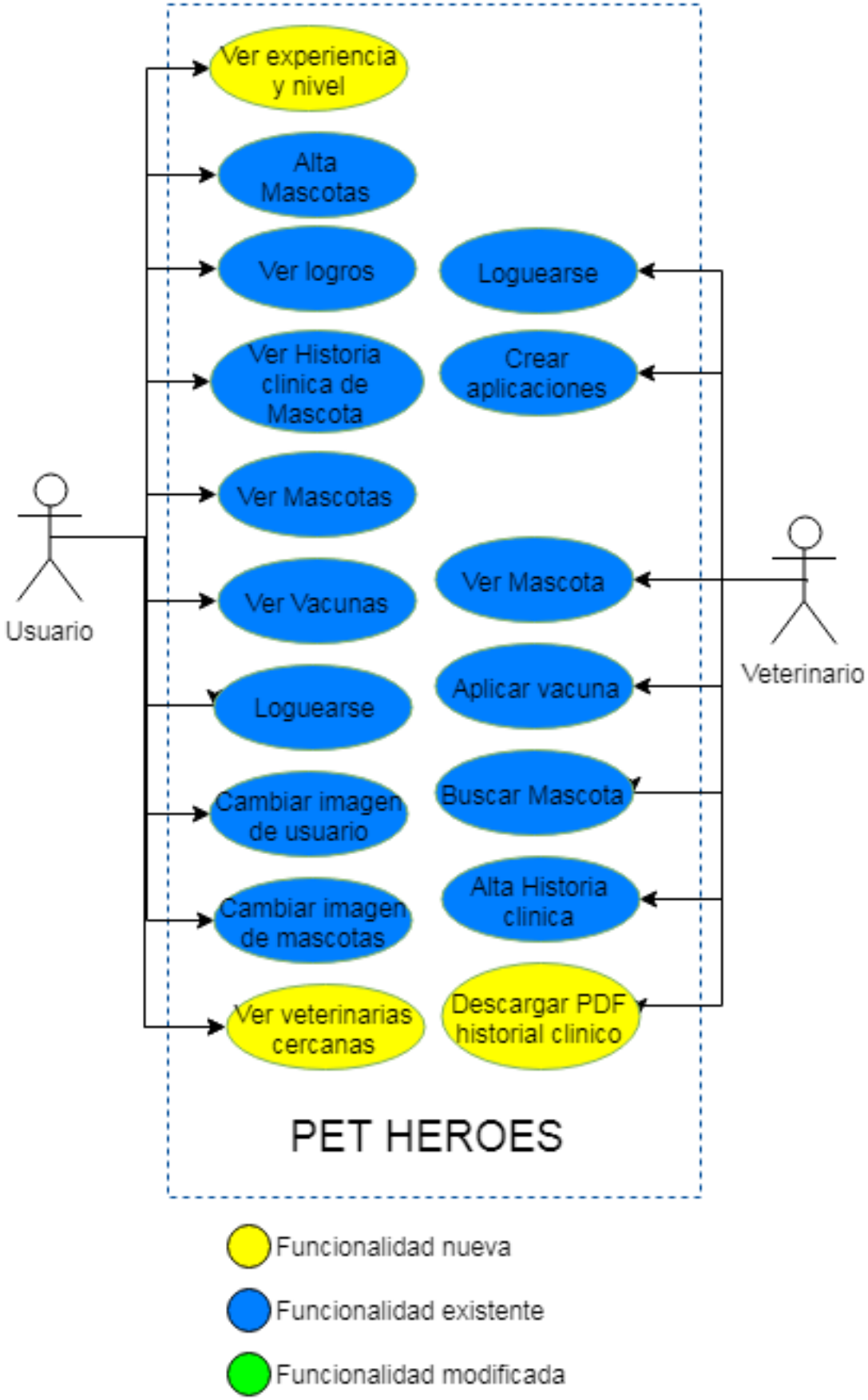



Figura 3.6: Casos de uso Sprint 3

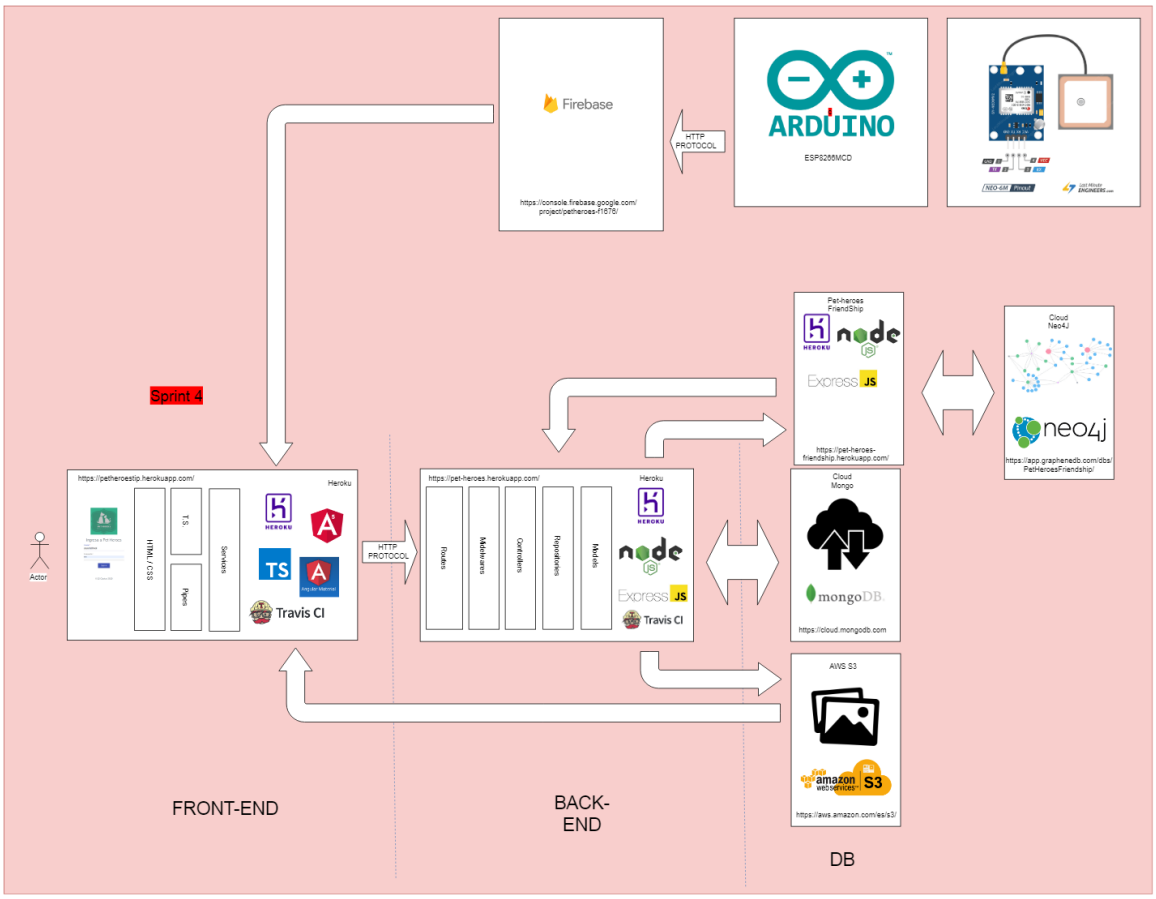



Figura 3.7: Arquitectura Sprint 4

### Persistencia

Se agrega una BD Firebase que se conecta a una funcionalidad experimental de ubicación de mascotas.

Ver Figura 3.7

## 3.4.2. Casos de uso

- **Buscar usuario:** el USUARIO abre el sidebar y hace click sobre .ªmigos", donde se muestran 2 nuevas opciones. Hace click en la opción .ªgregar amigos", y en el input que se visualiza ingresa un criterio de búsqueda (texto) que debe contener el email, el nombre o el teléfono de algún usuario. Por ej: si se ingresa .ªnita", traerá al usuario .ªnita@mail.com". Suponiendo que existe este usuario, con su foto, su nombre y mail, también va a encontrar un botón para agregarlo como amigo (en caso de no serlo) o el botón "desabilitado"que informa que ese usuario ya es su amigo.

- **Agregar amigo:** el USUARIO sigue los pasos del caso de uso "Buscar usuario", y luego, si el usuario no es aún un amigo, puede agregarlo haciendo click en el

boton .Agregar amigo".

- **Ver amigo:** el USUARIO abre el sidebar y hace click sobre .Amigos", donde se muestran 2 nuevas opciones. Hace click en la opción "Ver amigos", que lo direcciona a una pantalla en la cual se visualiza una lista con todos sus amigos; en caso de no tenerlos, muestra un botón "buscar amigos"
- **Ver ubicación mascota:** en el perfil de una mascota aparece una cuarta solapa que, al ingresar, direcciona a un mapa con foco en la ubicación del USUARIO, y un marker que indica dónde se encuentra ubicada su mascota.
- **Alta usuario:** el USUARIO ingresa a la aplicación por primera vez y hace click en el botón registrarse". Se le solicitará que complete un formulario con un nombre de usuario, email, contraseña y género. Si los datos ingresados son válidos, retorna a la página de login con el username del usuario ya cargado. Allí puede completar con su contraseña; y si los datos son correctos, es direccionado a la página principal estando logueado.

Ver figura
3.8

### 3.4.3. Endpoints

| | Endpoints Sprint 3 | | |
|---|---|---|---|
| | Method | URI | Body |
| Get User's friends | GET | /users/friends?mail='<email>' | — |
| Find Users | GET | /users/search?query=<query> | — |
| Add New User | POST | /users | {<br>name : String ,<br>email : String ,<br>gender: String ,<br>phone: Number ,<br>password : String<br>}; |
| Add Friend | POST | /friends/relationship | {<br>aMail: String ,<br>bMail: String<br>}; |

Tabla 3.3: Endpoints Sprint 3

## 3.5. Conclusión de las entregas

Se llegó a implementar de manera satisfactoria y según lo estimado una gran parte del proyecto, tal como se lo presentó al inicio. Todas las funcionalidades antes

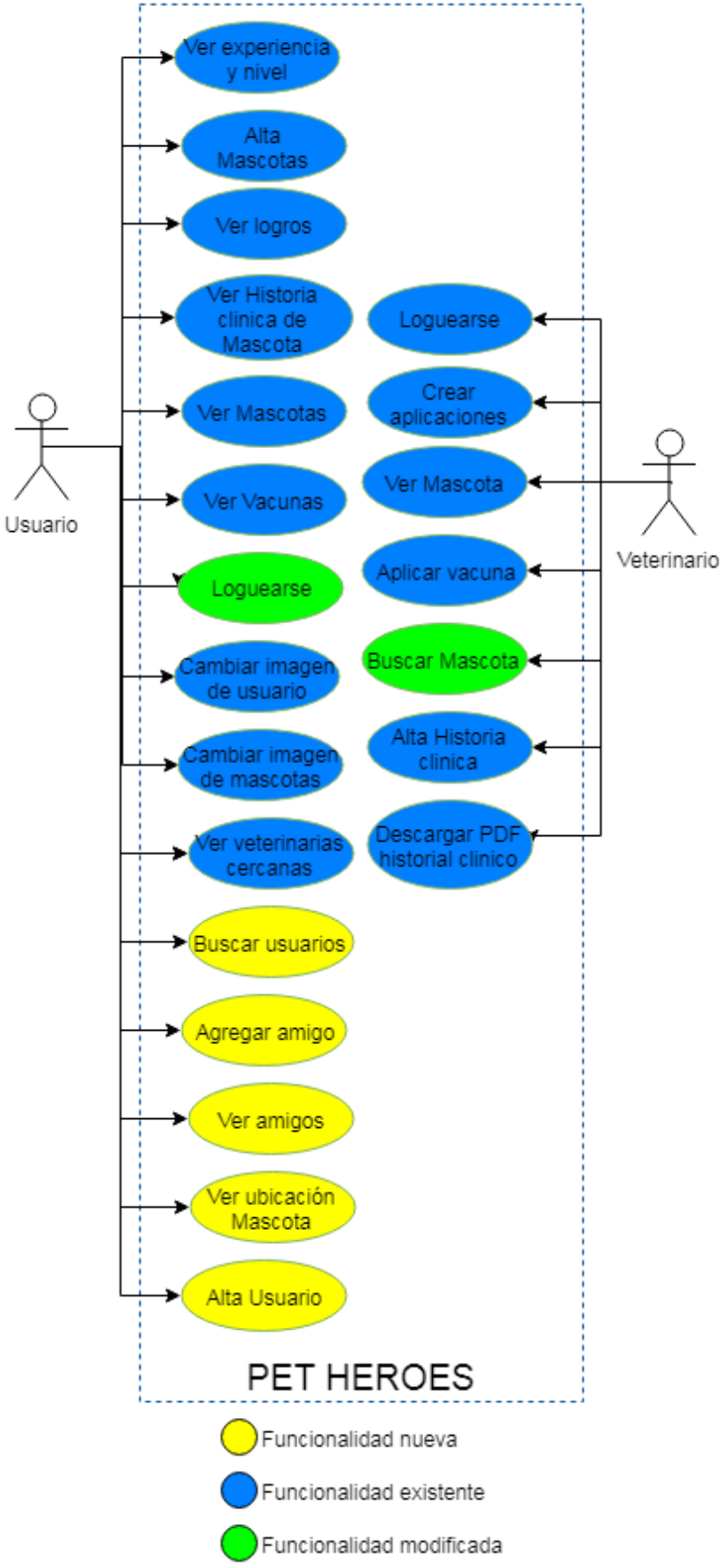
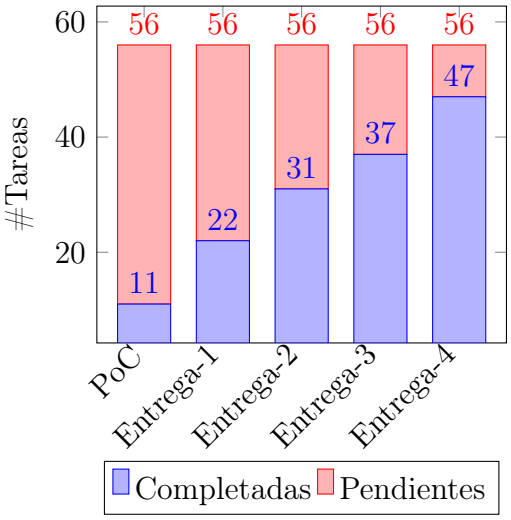



Figura 3.8: Casos de uso Sprint 4

descriptas están en funcionamiento en la nube, con excepción de la ubicación en tiempo real de la mascota, que está en etapa experimental.



# Capítulo 4

# Casos de uso

## 4.1. CU-01 Autenticar usuario en app

### 4.1.1. Descripción del escenario

Un usuario ingresado previamente en el sistema desea iniciar sesión en la aplicación.

### 4.1.2. Actores

Usuario autenticado (ACT-02).

### 4.1.3. Secuencia de interacciones entre los actores y el sistema

El usuario ingresa al home y otorga usuario y contraseña. El sistema comprueba que los datos ingresados coincidan con algún registro ingresado previamente. El sistema comprueba los permisos del usuario. El usuario obtiene la página que corresponde a sus permisos.

### 4.1.4. Extensiones / Flujos secundarios

Si el usuario otorga credenciales inválidas, se le prohibe avanzar.

### 4.1.5. Resumen

- ID: CU-01
- Nombre: Autenticar usuario en app.
- Objetivo: Autenticar un usuario en la app a partir de sus credenciales.
- Actores: Usuario a autenticar.
- Condiciones iniciales: El usuario debe estar registrado previamente en el sistema.

- Condiciones finales: El usuario consigue autenticarse para hacer uso de los servicios.

## 4.2. Caso de uso: CU-02 Dar de alta una mascota

### 4.2.1. Descripción del escenario

Un usuario del sistema con una sesión abierta en el mismo desea dar de alta una nueva mascota.

### 4.2.2. Actores

Usuario autenticado (CU-01).

### 4.2.3. Secuencia de interacciones entre los actores y el sistema

El usuario solicita agregar una mascota al sistema. El usuario ingresa el nombre de la mascota, la fecha de nacimiento, si está castrado (o no) y el género. El sistema le muestra los datos que ingresó. El usuario confirma los datos. El sistema registra en su base a la mascota y la vincula al usuario que le dio de alta. El usuario ve en su lista de mascotas la nueva mascota que añadió.

### 4.2.4. Extensiones / Flujos secundarios

Si el usuario no otorga todos los datos que el sistema solicita, se le prohibe avanzar.

### 4.2.5. Resumen

- ID: CU-02
- Nombre: Dar de alta una mascota.
- Objetivo: Registrar una nueva mascota en el sistema.
- Actores: Usuario autenticado (CU-01).
- Condiciones iniciales: El usuario debe estar autenticado y con una sesión abierta en el sistema.
- Condiciones finales: El usuario consigue registrar una mascota en el sistema.

## 4.3. Caso de uso: CU-03 Cambiar foto de mascota

### 4.3.1. Descripción del escenario

Un usuario del sistema con una sesión abierta en el mismo desea cambiar la foto de su mascota.

### 4.3.2. Actores

Usuario autenticado (CU-01).

### 4.3.3. Secuencia de interacciones entre los actores y el sistema

El usuario selecciona la mascota a la cual quiere cambiarle la foto. El usuario clickea sobre la foto actual de esa mascota, selecciona la foto por la cual desea cambiarla. El usuario confirma la foto por la que va a reemplazar la actual. El sistema carga la nueva foto y refresca la foto de la mascota por la nueva. El usuario ve en el perfil de su mascota la foto nueva que seleccionó.

### 4.3.4. Extensiones / Flujos secundarios

* Si el usuario intenta subir una foto no válida, el sistema no le permite avanzar.

### 4.3.5. Resumen

- ID: CU-03
- Nombre: Cambiar foto de una mascota.
- Objetivo: Cambiar la foto de una mascota en el sistema.
- Actores: Usuario autenticado (CU-01).
- Condiciones iniciales: El usuario debe estar autenticado y con una sesión abierta en el sistema.
- Condiciones finales: El usuario consigue cambiar la foto de su mascota en el sistema.

# Capítulo 5

# Conclusiones

El proyecto ha sido muy amplio desde su planteo inicial. Fue abordado de manera iterativa e incremental, como puede leerse a lo largo de los capítulos anteriores. La arquitectura del proyecto se ha ido complejizando de forma secuencial, y cada módulo se desarrolló de forma desacoplada de los otros, para poder desarrollar las funcionalidades de forma independiente entre sí. Esta forma de crecimiento del proyecto permitió también que podamos utilizar varias tecnologías, novedosas para nosotros; y su integración al proyecto representó un verdadero desafío. La ubicación de la mascota en tiempo real permanece a la fecha en etapa experimental.