



[G4-43]

TECHNICAL REPORT A.2.2.4

Periodo de Reporte: – [11/03] - [04/04], [2021]

Fecha de Emisión: 02/04, 2021

Fecha de inicio y fin del Sprint: [11/03, 2021] a [04/04, 2021]

Autores:

- Fernando Andrés Galindo
- Matthew Bwye Lera
- Fernando José González García
- Manuel Guerra Cordon
- Francisco Murillo Prior
- Jesús Aparicio Ortiz



- **SUMARIO**

En este informe trataremos los asuntos técnicos encontrados al realizar el sprint de nuestra versión de Petclinic. Recorreremos también detalles de nuestra forma de trabajar con las herramientas empleadas, mostrando una serie de gráficos y las conclusiones que hemos sacado al trabajar de distintas formas.

- **PRÓLOGO**

Este informe ha sido redactado para el primer sprint de un proyecto para la asignatura de PSG2, en la Universidad de Sevilla. El objetivo de la asignatura, y en general del trabajo que estamos realizando, es aprender a gestionarnos como equipo, utilizando las herramientas y técnicas adecuadas.

Con el objetivo anterior en mente, trabajaremos sobre un proyecto llamado Petclinic, utilizando el framework Spring y el lenguaje de programación Java. Petclinic es una página web para una clínica veterinaria, cuyo objetivo es mejorar la gestión de la clínica en general, almacenando información de mascotas, veterinarios y dueños, entre otros.

• TABLA DE CONTENIDO

TECHNICAL REPORT A.2.2.4	1
• SUMARIO	2
• PRÓLOGO.....	2
• TABLA DE CONTENIDO	3
1. INTRODUCCIÓN.....	4
2. CUERPO.....	4
2.1 Actividades completadas bajo la propuesta del Product Backlog.....	4
2.2.1 Dificultades en desarrollo de tareas.....	4
2.2.2 Dificultades en desarrollo de tareas (Extra).....	5
2.3.1 Uso de “Trunk-based development”	5
2.3.2 Uso de “Gitflow” (extra)	6
2.4 Trabajo en equipo.....	6
2.5 Historial de cambios de A.2.3.3 Y A.2.3.4.....	8
3 CONCLUSIONES	12
4 ANEXOS	13
ANEXO I: Referencias.....	13

1. INTRODUCCIÓN

Este proyecto es una página web diseñada para una clínica veterinaria. La página web almacenará información de dueños, veterinarios, mascotas e incluso reservas de habitaciones para mascotas que tengan que quedarse en la clínica más tiempo.

Con este informe pretendemos informar acerca de nuestra forma de realizar este trabajo, mencionando las distintas estrategias que hemos empleado, el reparto de tareas que hemos diseñado y los aspectos positivos y negativos que nuestras decisiones han acarreado.

Además, nuestro equipo está recién formado. Uno de los objetivos de este sprint era explorar mejores formas de trabajar juntos y conocernos. En este informe explicaremos qué decisiones grupales hemos tomado, observaciones que hemos realizado y qué conclusiones hemos sacado de todo nuestro trabajo en equipo.

2. CUERPO

2.1 Actividades completadas bajo la propuesta del Product Backlog

Listar las tareas de las que se hablarán en este documento.

ID Tarea	Descripción Tarea	Hecha
A.2.2.3.a	Change the project information in pom.xml	X
A.2.2.3.b	Add your information to the developers section in pom.xml	X
A.2.2.3.c	Change the color of the header buttons when selected to light red	X
A.2.2.3.d	Change the background color of the header menu to light green	X
A.2.2.3.e	Change the message of the Welcome page to “Welcome to the PSG2-2021-GX-XY Petclinic”, where GX-XY is your group id	X
A.2.2.3.f	Change the image of the Welcome page with another pet you like	X
A.2.2.3.g	Change the background color of the table header when looking for owners to light grey	X

2.2.1 Dificultades en desarrollo de tareas

Listar y describir cualquier dificultad que haya afectado a la implementación del proyecto y cómo se han solucionado.

El desarrollo de las tareas del A2.2 del Product Backlog ha sido por lo general fluido. Únicamente se encontraron dos problemas:

La primera le surgió a Jesús Aparicio, ya que al completar la tarea 2.2.3.g en el .less donde se establecen los estilos había un estilo para todas las tablas, de manera que si se intentaba directamente añadir un estilo a la cabecera de la tabla de owner no cambiaba nada, ya que el otro estilo tenía prioridad. Finalmente, añadiendo una clase específica a la tabla de owner indicando el color de la cabecera se solucionó.

La segunda le surgió a Francisco Murillo, ya que dentro de los archivos .less era necesario cambiar el color del botón correspondiente así como, cambiar a su vez el path al que éste (el color) estaba vinculado.

2.2.2 Dificultades en desarrollo de tareas (Extra)

Listar y describir cualquier dificultad que haya afectado a la segunda parte de la implementación del proyecto.

En cuanto al desarrollo de las tareas correspondientes al A2.3.3, se encontraron los siguientes problemas:

Jesús Aparicio dudó en implementar una clase diferenciada para las habitaciones, ya que al solo necesitar registrar la ID se podrían implementar de manera más sencilla como un campo dentro de la clase Book, en vez de como una relación. Finalmente, sí se implementó mediante la clase Room, para evitar una posible deuda técnica en caso de que nuevos requisitos creasen la necesidad de campos para esta entidad.

Fernando González pensó implementar las validaciones de las reservas mediante excepciones, al pensar que solo habría que realizar una única validación (Comprobar si existen habitaciones libres). Sin embargo, a la hora de desarrollarlo surgió la necesidad de una nueva validación para comprobar que la fecha de llegada sea anterior a la de salida, lo que imposibilitó la idea principal de implementarlas únicamente mediante excepciones, ya que este método tiene el inconveniente de que al solo poder saltar una excepción a la vez, no es adecuado para la comprobación de múltiples validaciones. Al final se optó por implementar la validación para comprobar habitaciones libres de la forma originalmente planeada, mientras que el resto se implementaron mediante un validador custom, en la clase BookValidator.

Manuel Guerra, Fernando Andrés y Francisco Murillo se encontraron con un fallo a la hora de realizar el formulario para crear y editar veterinarios, ya que no se podía seleccionar más de una especialidad. La solución fue implementar un nuevo método de selección distinto al que incluye petclinic de base.

2.3.1 Uso de “Trunk-based development”

Listar y describir cualquier dificultad surgida a la hora de trabajar con esta estrategia.

Ya que todos los integrantes del grupo han actualizado su rama local antes de realizar cambios en el proyecto y no ha coincidido que dos personas hagan un push a la rama master donde hubiese conflictos, entonces no se ha dado ningún conflicto a la hora de trabajar todos con el master. Sin embargo, el grupo es consciente de que al trabajar directamente con la rama master lo más común es encontrarse con conflictos.

El hecho de trabajar directamente con el master conlleva de manera directa a un tema de ineficiencia. Aun así, consideramos que es una mejora respecto a herramientas con las que los miembros del grupo trabajaron anteriormente, como puede ser el caso de Tortoise SVN. Esta herramienta, a pesar de ser preferible a no usar ningún tipo de herramienta de gestión, tiene la gran desventaja de utilizar un mecanismo bloqueante, ya que solo una persona puede estar modificando un fichero a la vez. En caso de que una persona bloquee un fichero en el que está trabajando y seguidamente otra intente realizar un merge habiendo modificado dicho archivo, resultará en la prohibición de realizar el merge por parte de la herramienta. Por otro lado, con el uso de GitHub no se tiene en ningún momento este mecanismo bloqueante, por lo que todo el mundo puede trabajar libremente en cualquier archivo sin miedo a tener represalias en un futuro, siempre y cuando si hubiera un conflicto este se solucionara a la hora de hacer merge.

2.3.2 Uso de “Gitflow” (extra)

Ventajas y desventajas del uso de Gitflow.

A la hora de hacer uso de una estrategia de branching como lo es Gitflow se dan una serie de ventajas e inconvenientes respecto de hacer uso de una única rama, como lo es el “Trunk-based development”.

Las principales ventajas de hacer uso de varias ramas son bastantes. Por un lado la capacidad de tener varias ramas en función del estado del proyecto (desarrollo, producción...), ya que con esto tenemos la capacidad de realizar el desarrollo del proyecto a la vez que tenemos otra rama con una versión estable del mismo. Además, el uso de varias ramas nos permite realizar una rama por cada tarea del proyecto, de modo que en caso de querer realizar alguna modificación al proyecto en estado de desarrollo, se podría directamente hacer uso de la rama de la tarea y modificarla antes de pasar los cambios a la rama de desarrollo y de esta manera evitar que en la rama global de desarrollo se generen errores. De la misma manera si en la versión estable del proyecto existe un fallo, se podría crear una rama auxiliar (“Hot Fix”) para solucionar dicho error sin tener que arriesgarse a realizar una modificación directamente sobre la versión estable del proyecto, lo cual es una ventaja enorme.

Las principales desventajas de hacer uso de varias ramas a pesar de ser menores es importante tenerlas en cuenta. Por un lado la complejidad de realizar una gestión de un número elevado de ramas frente a una única rama, además del tiempo requerido para crear todas las ramas y hacer un uso correcto de todas ellas.

2.4 Trabajo en equipo

Describir los datos relevantes respecto al trabajo en equipo

En nuestro grupo tenemos a dos miembros (Fernando Andrés y Francisco Murillo) que no cursaron DP1, por lo que no sabían utilizar el framework o github y, por otro lado, el sprint presentaba en general una serie de tareas poco exigentes. Esto hizo que tomáramos la decisión de

hacer que Francisco, Fernando Andrés y Manuel trabajasen siempre juntos en las tareas relacionadas con el uso del framework y github, y que trabajasen los tres juntos aproximadamente lo mismo que lo que trabajaría una única persona, reduciendo así la carga de trabajo, pudiéndose así dejar más tiempo para explorar y comprender el framework.

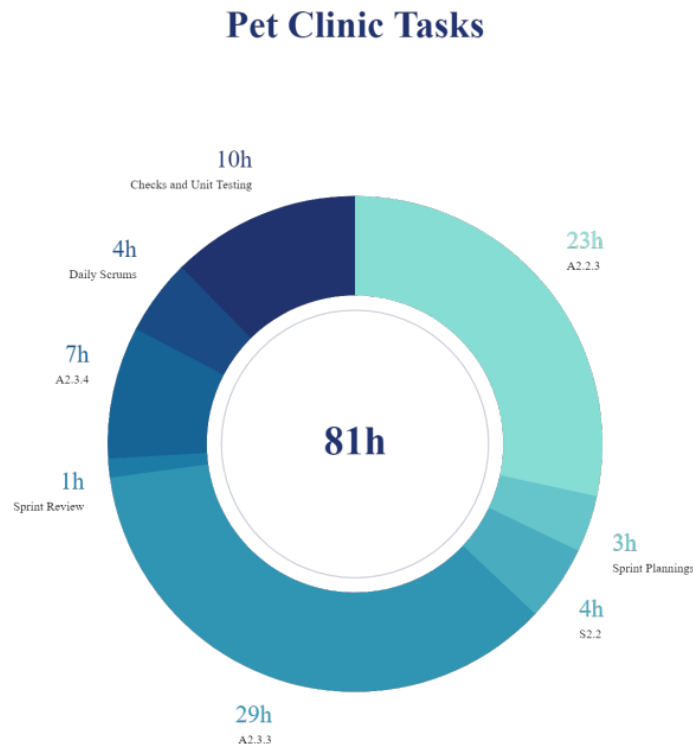


Figura X: Horas invertidas en las distintas áreas del proyecto

Por otro lado, el grupo ha mostrado ser muy capaz de trabajar de forma muy autoorganizada. De esta forma, hemos podido cubrir de muy buena manera los roles de un equipo scrum, al contrario de lo que han hecho otros equipos, donde el Scrum master cubría la función de organizador o jefe de proyecto.

Algunos buenos ejemplos de estas prácticas han sido: en momentos de emergencia cualquier miembro del grupo decidía un reparto de tareas ágil sin importar su rol dentro de el grupo; si alguna vez un miembro ha faltado a una reunión, ha cubierto su parte de manera asíncrona, grabándose o redactando lo que debía decir, antes o después de la reunión; planificamos desde el principio fechas de reuniones, conceptos de acabado, metodologías de trabajo, etc., y seguimos todo lo decidido sin problemas. En el diagrama adjunto se lista el tiempo invertido en las distintas áreas del proyecto. Entre ellas, se aprecia el tiempo invertido en reuniones.

En las primeras reuniones ya empezamos a realizar labores de gestión sobre las que nos hemos ido basando sin problemas en todo momento. Decidimos los días para realizar los daily

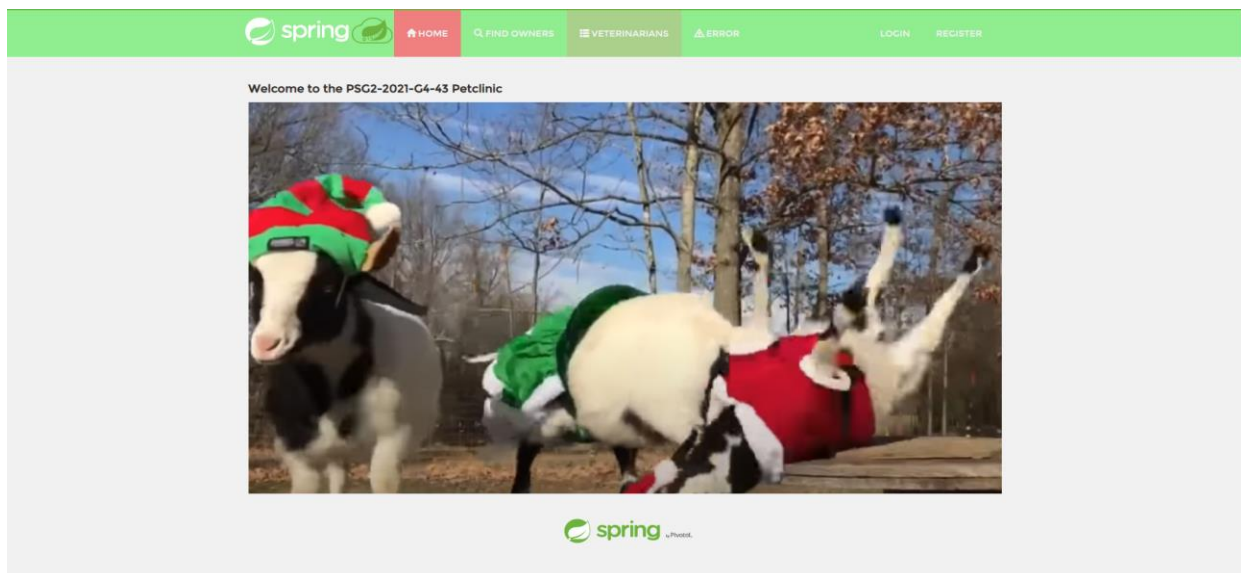
scrums, además de que haríamos varios replannings para las distintas subdivisiones del sprint. También nos pusimos objetivos dentro del mismo sprint, como si fuesen subsprints, en los que debíamos hacer ciertas tareas para determinadas fechas. Otras iniciativas que tuvimos fueron un sistema rotatorio para recoger actas de las reuniones, decidir deadlines anteriores al deadline real del sprint y decidir qué estrategia de branching aplicar (aunque finalmente se nos pidió usar gitflow).

Nos propusimos además una serie de objetivos más allá de lo que se nos pedía en el Product Backlog. Los principales objetivos que nos propusimos fueron conocernos como equipo, conocer mejor las herramientas, probar distintas técnicas y planificaciones, en general, experimentar un poco como grupo para poder mejorar nuestra capacidad de trabajo en los próximos sprints, que seguramente serán más duros.

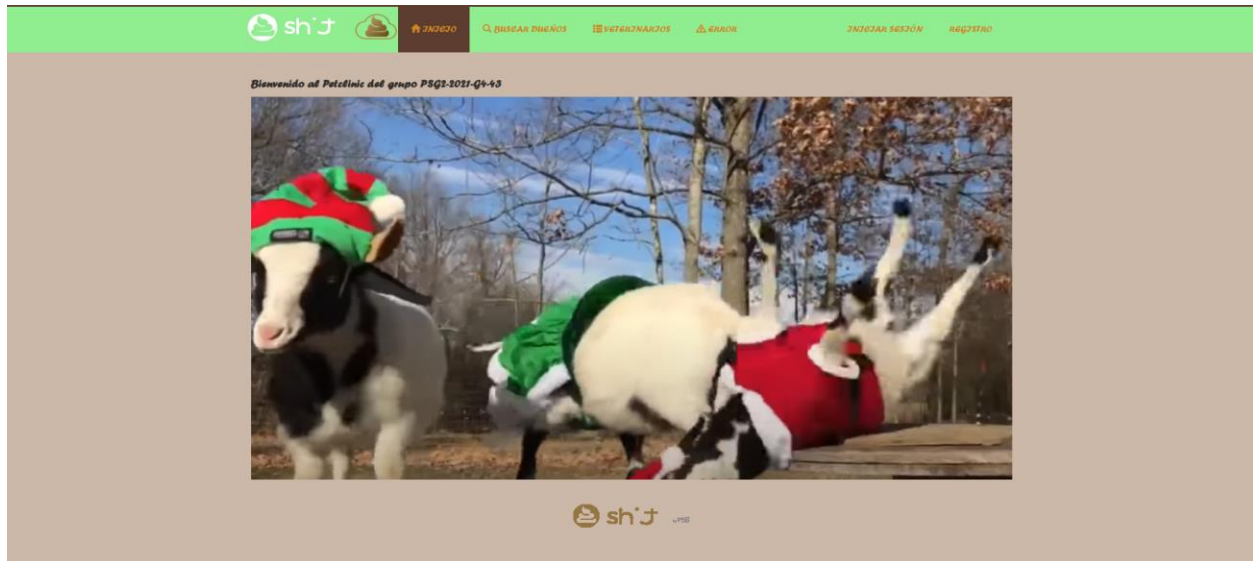
2.5 Historial de cambios de A.2.3.3 Y A.2.3.4

Describir los distintos cambios realizados entre la rama develop y la rama release

La rama develop en un principio tenía únicamente el contenido de la rama master, de modo que se veía tal que así:



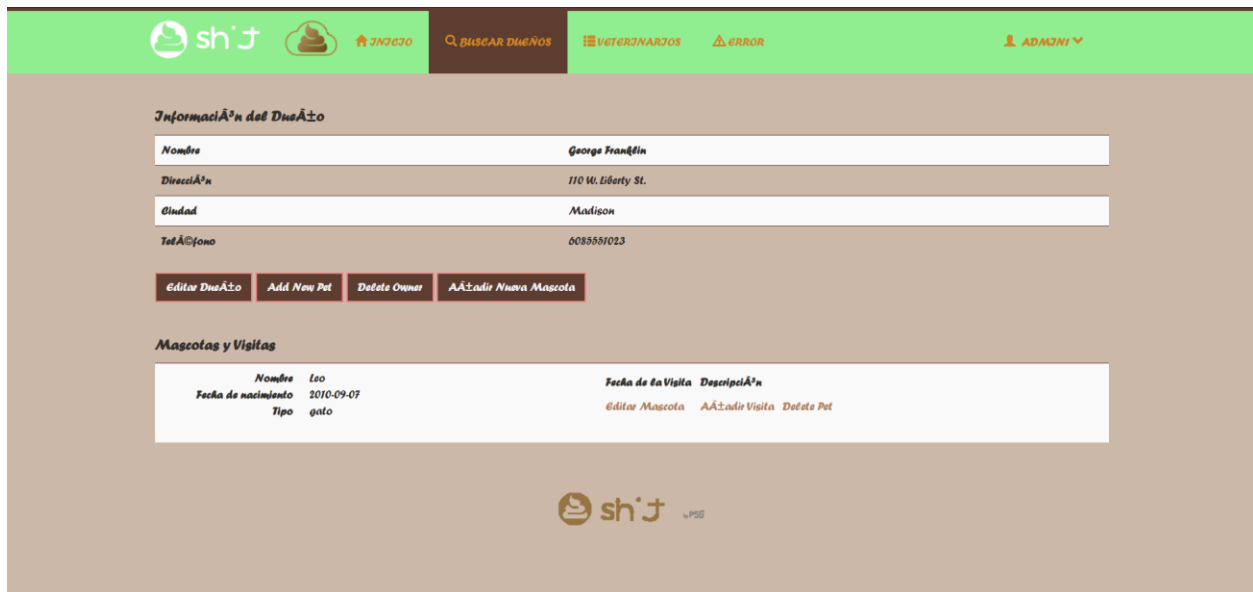
En este punto únicamente se habían implementado las primeras tareas requeridas en el Product Backlog. Tras la implementación de las distintas tareas que se requerían en la segunda parte del product backlog, la rama develop quedó de la siguiente manera:



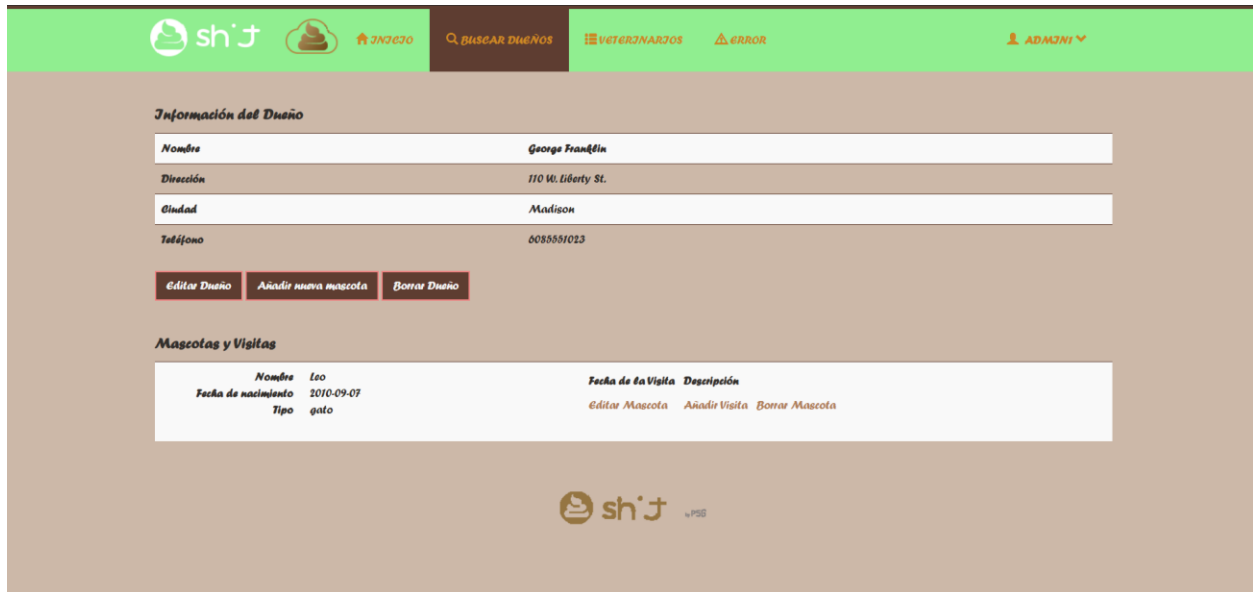
Tras terminar las tareas del Product Backlog y crear la rama Release a partir de la develop, se hicieron una serie de cambios en la release, ya que se encontraron una serie de fallos en la aplicación. De este modo, se va a listar una serie de errores y sus respectivas capturas del cambio.

-Codificación utf-8 no aplicada en la página. Además de eliminar un botón de añadir mascota que estaba repetido y en inglés.

Develop



Release



-Visualización de veterinarios en formato xml, ya que en la rama develop no estaba implementado y se implementó en la release.

Develop

Whitelabel Error Page

This application has no explicit mapping for /error, so you are seeing this as a fallback.

Sun Apr 04 11:59:03 CEST 2021

There was an unexpected error (type=Forbidden, status=403).

Forbidden

Release

This XML file does not appear to have any style information associated with it. The document tree is shown below.

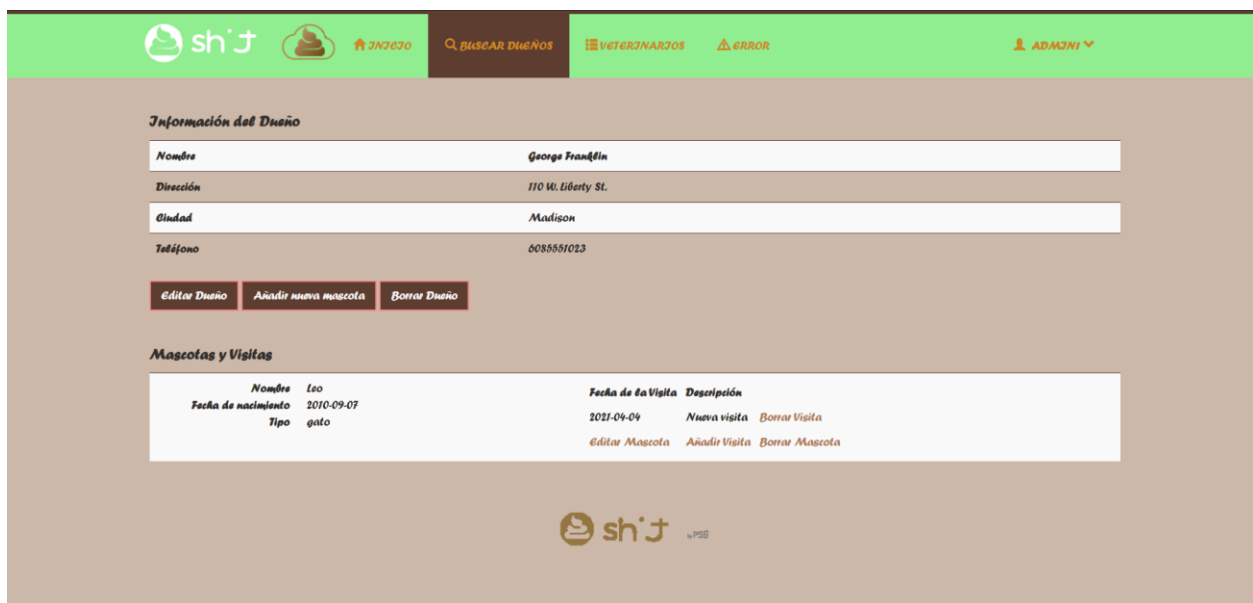
```
▼<vets>
  ▼<vetList>
    <id>1</id>
    <firstName>James</firstName>
    <lastName>Carter</lastName>
  </vetList>
  ▼<vetList>
    <id>2</id>
    <firstName>Helen</firstName>
    <lastName>Leary</lastName>
    ▼<specialties>
      <id>1</id>
      <name>radiología</name>
    </specialties>
    ▼<specialtiesXml>
      <id>1</id>
      <name>radiología</name>
    </specialtiesXml>
  </vetList>
  ▼<vetList>
    <id>3</id>
    <firstName>Linda</firstName>
    <lastName>Douglas</lastName>
    ▼<specialties>
      <id>3</id>
      <name>odontología</name>
    </specialties>
    ▼<specialties>
      <id>2</id>
      <name>cirugía</name>
    </specialties>
    ▼<specialtiesXml>
      <id>2</id>
      <name>cirugía</name>
    </specialtiesXml>
    ▼<specialtiesXml>
      <id>3</id>
      <name>odontología</name>
    </specialtiesXml>
  </vetList>
  ▼<vetList>
    <id>4</id>
    <firstName>Rafael</firstName>
    <lastName>Ortega</lastName>
    ▼<specialties>
      <id>2</id>
      <name>cirugía</name>
    </specialties>
    ▼<specialtiesXml>
      <id>2</id>
      <name>cirugía</name>
    </specialtiesXml>
  </vetList>
  ▼<vetList>
    <id>5</id>
    <firstName>Henry</firstName>
    <lastName>Stevens</lastName>
    ▼<specialties>
      <id>1</id>
```

-La funcionalidad de borrar una visita de una mascota no estaba implementada.

Develop



Release



3 CONCLUSIONES

En primer lugar, se han realizado las tareas representadas en el Product Backlog, con dificultades en algunas de las respectivas tareas, solucionados más adelante al indagar más en dichos errores. En segundo lugar, destacar el uso de Trunk-based development, ya que a pesar de ser menos eficiente que Gitflow, hay una diferencia enorme respecto a otros mecanismo

Technical Report

bloqueantes con los que antiguamente algunos miembros del equipo habían trabajado, como Tortoise SVN. Por último, cabe destacar el trabajo en equipo que se ha seguido, ya que ha sido un poco ineficiente por la razón de que varios miembros del equipo no habían interactuado con el entorno de github anteriormente y se les ha prestado ayuda.

4 ANEXOS

ANEXO I: Referencias

[Technical Report: What is it & How to Write it? \(Steps & Structure Included\) \(bit.ai\)](#)

https://www.itu.dk/people/sestoft/itu/writing_reports.pdf

https://templatelab.com/technical-reports/#google_vignette

<https://www.sample.net/business/analytics/technical-reports/>