ANIMALES SIN ABRIGO

1. Documento Scrum

1.1. Estructura y funcionamiento del equipo Scrum

La organización del equipo se ha tratado de forma fundamental para el inicio del proyecto. Desde el primer momento, hemos intentado organizar los roles de tal forma que al final del proyecto, todos los integrantes del equipo hayamos colaborado en cada una de las áreas del mismo. Es por ello que estos roles quizá se encuentran modificados con respecto al final del proyecto. Los primeros roles designados han sido:

- *Scrum Master*: Antonio Álvarez
- *Software design and architecture*: Javier Peña y Eneko Retolaza
- *GUI design and development*: Fabrizio Zeballos, Guillermo López y Eneko Retolaza
- *Backend development*: Antonio Álvarez y Fabrizio Zeballos
- *Product Owner and Management*: Alonso Salgueiro y Alberto Martínez

A lo largo del desarrollo se buscará que los integrantes de las secciones de desarrollo roten entre sí, para asegurar el completo conocimiento del proyecto.

Las herramientas utilizadas para manejar la información han sido:

- GitHub: Hemos creado un repositorio en remoto para controlar las versiones y el código programado y manejado por Git de forma local en cada uno de los integrantes.
- Google Drive: Hemos creado una unidad compartida como plataforma de redacción de documentos formales simultáneamente.
- Discord: Hemos creado un servidor como plataforma de comunicación y chat de voz. En todo momento en el que uno de los integrantes se encuentre trabajando en el proyecto, debe entrar en la llamada del servidor, para conocer en todo momento quién está trabajando y facilitar la comunicación.
- Miró: Tenemos creado un grupo con pizarras como punto de brainstorming y elaboración de los documentos correspondientes a la metodología scrum, como las historias de usuario o el product backlog.
- Figma: Hay un proyecto en el que tenemos hechos los bocetos de las pantallas que queremos implementar, con las imágenes, botones y funcionamiento.

Como se puede observar, todas estas herramientas tienen en común las facilidades para la edición y el trabajo de forma remota y simultánea. Las reuniones presenciales obligatorias son todas aquellas

organizadas durante horario lectivo, especialmente los martes de 13:00 a 15:00, y también aquellas extraordinarias que organicemos de forma prioritaria. El resto de reuniones y trabajo se realiza de forma remota.

1.2. Historias de Usuario

Las actuales historias de usuario tienen proyecto para ser implementadas en el proyecto a lo largo de su desarrollo. A medida que vayamos avanzando con el desarrollo, encontraremos los impedimentos y/o decisiones que nos harán descartarlas.

<u>Iniciar sesión:</u>

ID: 2

Como usuario quiero poder iniciar sesión.

Al entrar en la página el usuario se encontrará con un botón el cual le llevará a otra página donde el usuario meterá sus datos para poder iniciar sesión.

Criterios de validación:

- Verificar que el usuario y la contraseña sean correctos y dar acceso al sistema.

Valor: 8

Prioridad: 5

<u>Cerrar sesión:</u>

ID: 21

Como usuario quiero poder cerrar mi sesión.

Una vez usado el botón de iniciar sesión, dentro de esa ventana y una vez terminado de usar la página el usuario tendrá el acceso a un botón que le permitirá cerrar sesión y salir de la página.

Criterios de validación:

- Verificar que se ha cerrado la sesión del usuario y que ya no tiene acceso al sistema.

Valor: 2

Prioridad: 5

Eliminar cuenta:

ID: 8

Como usuario quiero poder eliminar mi cuenta.

El usuario tendrá acceso a un botón una vez creada la cuenta y antes de iniciar sesión para poder eliminar su cuenta de la base de datos de la aplicación.

Criterios de validación:

- Verificar que se ha eliminado el usuario de la base de datos y con ello todos los datos del usuario.

Valor: 3

Prioridad: 4

Registrarse:

ID: 1

Como usuario quiero poder registrarme para poder interactuar en la aplicación.

Una vez abierta la aplicación por primera vez el usuario tendrá acceso a un botón que le permitirá entrar en una ventana donde escribirá todos sus datos los cuales se guardarán con el fin de poder volver a usarlos para iniciar sesión más adelante.

Criterios de validación:

- Comprobar que el nombre y contraseña tienen los requerimientos necesarios.
- Comprobar que se ha creado el usuario en la base de datos.

Valor: 5

Prioridad: 3

Blog:

ID:12

Como refugio quiero poder publicar entradas en el blog para dar información y noticias.

Criterios de validación:

- Comprobar que la entrada tiene un autor del refugio.
- Comprobar que la entrada no está vacía.

Valor: 8

Prioridad: 2

Adoptar animal:

ID: 4

Como usuario quiero poder adoptar un animal.

Mediante un botón, el usuario será capaz de acceder a la lista de animales y ahí elegir cuál es el animal que quiere adoptar.

Criterios de validación:

- Poder dar al usuario una posibilidad de adoptar animales en los centros de acogida.

Valor: 7

Prioridad: 5

Solicitar curso de adopción:

ID:19

Como usuario quiero poder hacer un curso de adopción de animales.

Dentro de la ventana del adoptante aparecerá un botón en el cual puedes pinchar y apuntarte para solicitar un curso de adopción.

Criterios de validación:

- Enseñar al usuario las habilidades necesarias para poder adoptar a un animal.
- Comprobar el buen estado psicológico del adoptante para poder adoptar un animal.

Valor: 5

Prioridad: 1

Apadrinar animal:

ID: 10

Como usuario quiero poder apadrinar a un animal.

Una vez dentro de la aplicación el usuario contará con la posibilidad de elegir un botón el cual le permitirá poder apadrinar un animal.

Criterios de validación:

- Verificar que el usuario haya realizado el pago.
- Verificar que el animal haya sido apadrinado.

Valor: 7

Prioridad: 3

Acogida animal:

ID: 9

Como usuario quiero poder acoger un animal.

Al iniciar sesión como usuario, este será capaz al igual que adoptar y apadrinar un animal, de tener acceso a un botón el cual le permitirá acoger un animal.

Criterios de validación:

- Verificar que el usuario está en plenas capacidades para acoger un animal.
- Hacer la transacción de acoger un animal.

Valor: 7

Prioridad: 3

Curso adopción:

ID: 11

Se pone disponible el curso con la información necesaria sobre el curso de adopción para el usuario.

En la ventana de los refugios habrá una opción que será la de ofrecer un curso de adopción al cual podrán solicitar acceso los usuarios.

Criterios de validación:

- Poner disponible la información que necesita el usuario sobre el curso.

Valor: 6

Prioridad: 2

Añadir animal:

ID: 5

Se añade la opción al sistema gestor de poder añadir un animal a un refugio.

El refugio tiene la capacidad de añadir animales a su lista de animales dentro de la aplicación con un botón que aparecerá en la ventana una vez iniciada la sesión.

Criterios de validación:

- Comprobar que los datos del animal son coherentes y añadirlo a la base de datos.

Valor: 2

Prioridad: 4

Eliminar animal:

ID: 16

Se añade la opción al sistema gestor de poder eliminar un animal del refugio.

Dentro de la ventana de la lista de animales, el refugio tendrá el acceso a un botón que le dará la posibilidad de poder eliminar animales de la lista una vez que son adoptados por los usuarios.

Criterios de validación:

- Eliminar el animal de la base de datos.

Valor: 3

Prioridad: 4

Actualizar animal:

ID: 17

Se añade la opción de actualizar los datos de un animal ya existente en el sistema.

El refugio podrá a través de un botón poder actualizar la información de un animal ya existente en la lista de animales, ya sea porque ha sido adoptado o porque ha sido devuelto.

Criterios de validación:

- Comprobar que los nuevos datos del animal son coherentes.

Valor: 2

Prioridad: 3

Ver lista animales (adopter):

ID: 3

Como adopter quiero poder visualizar la lista de animales disponibles.

Al entrar en la página como adoptante habrá un botón el cual si lo clickeamos nos mostrará la lista de animales disponibles.

Criterios de validación:

- Comprobar los animales que hay disponibles en el centro de acogida y mostrarlos.

Valor: 6

Prioridad: 2

Ver lista animales (refuge):

ID: 6

Como administradores del refugio quiero poder ver la lista de animales que hay disponibles.

Al entrar en la página como refugio habrá un botón el cual si lo clickeamos nos mostrará la lista de animales disponibles.

Criterios de validación:

- Comprobar la lista de animales disponibles y mostrarlos.

Valor: 2

Prioridad: 2

Solicitar voluntarios:

ID:7

Como refugio quiero poder solicitar voluntarios para mejorar la plantilla y el funcionamiento del refugio.

Criterios de validación:

- Poder dar al refugio una posibilidad de mejorar la plantilla.
- Comprobar si hay espacio para nuevos voluntarios y solicitarlos.

Valor: 1

Prioridad: 1

Solicitar trabajadores:

ID:22

Como centro de acogida, quiero poder solicitar nuevos trabajadores

Criterios de validación:

- Comprobar si hay espacio para nuevos trabajadores y solicitarlos

Valor: 1

Prioridad:1

Enviar CV:

ID:27

Como candidato, quiero poder enviar el CV al centro de acogida

Criterios de validación:

- Comprobar que el CV sea válido y meterlo al centro de solicitudes

Valor: 2

Prioridad:1

Donaciones:

ID:14

Como centro de acogida, quiero poder recibir donaciones para los centros de acogida

Criterios de validación:

- Comprobar que la transacción se ha realizado y codificado correctamente

Valor: 4

Prioridad:1

Contactar refugio:

ID:23

Como usuario quiero poder contactar con el refugio.

Mediante un botón el usuario será capaz de contactar con el refugio del cual quiere adoptar un animal.

Criterios de validación:

- Comprobar que el refugio está disponible para poder ser contactado

Valor: 5

Prioridad:1

FAQ:

ID:15

Como usuario quiero poder consultar las dudas más frecuentes.

Criterios de validación:

- Comprobar el usuario que hace la consulta

Valor: 5

Prioridad:1

Sobre nosotros:

ID:24

Como refugio quiero poder explicar los motivos por los que tenemos un refugio y darme a conocer.

Criterios de validación:

- Comprobar que el usuario coincide con el refugio
- Comprobar que se ha creado bien la descripción

Valor: 5

Prioridad:1

Reviews de los refugios:

ID:18

Como adoptante quiero poder dar y consultar reviews para valorar a los mejores refugios.

Criterios de validación:

- Comprobar la lista de las críticas de antiguos adoptantes

Valor: 6

Prioridad:2

Solicitar examen:

ID:13

Como usuario quiero poder solicitar un examen

Criterios de validación:

- Comprobar que el usuario es compatible con el examen

Valor: 6

Prioridad:1

Ofrecer plazas guardería animales:

ID:20

Como usuario quiero poder dejar a mi animal al cuidado del refugio de forma temporal.

Criterios de validación:

- Comprobar el usuario que va a dejar a su animal
- Comprobar que los datos del animal son correctos
- Comprobar los datos del servicio: fechas de inicio y final, datos del usuario y del animal, precio, etc.

Valor: 5

Prioridad:5

Buscar plazas guardería animales:

ID:25

Como usuario quiero poder buscar plazas para guardería de animales

Criterios de validación:

- Comprobar que hay plazas de guardería en el centro solicitado para poder ofrecerle asistencia al usuario solicitante

Valor: 6

Prioridad:1

Eventos promocionales:

ID:26

Como centro de acogida, quiero poder promocionar eventos sobre el centro

Criterios de validación:

- Comprobar que la información haya llegado bien a los potenciales clientes

Valor: 3

Prioridad:1

1.3. Sprint Reviews

Review Sprint 1 - 15/03/2022

Organizamos una reunión informal para analizar especialmente el grado de éxito del sprint y los fallos que hemos tenido durante la organización.

Los principales fallos que hemos encontrado han sido:

- La falta de orden a la hora de empezar a trabajar. Esto es, querer diseñar partes que necesitaban previo diseño de otras que no estaban diseñadas, entre otros casos.
- La falta de asistencia en algunas ocasiones para reuniones, que nos ha dificultado un poco la comunicación.
- Hemos necesitado mucho tiempo para integrar GitHub con nuestro código y aprender a usarlo.
- Hemos encontrado problemas con JavaFX, que es la librería que hemos decidido usar para nuestras interfaces de usuario.

Sin embargo, el sprint ha sido exitoso porque hemos encontrado muchísimas ideas de implementación y mejora de los servicios de la aplicación, y hemos conseguido transmitir de forma satisfactoria los objetivos e ideas del proyecto, por lo que todos tenemos una idea general de cómo hacer las cosas.

Review Sprint 2 - 29/03/2022

Organizamos una reunión on line para ver el grado de éxito del sprint, los fallos y las mejoras posibles.

Principales fallos:

- Falta de organización y coordinación del equipo

- Incompatibilidad de horarios entre los diferentes integrantes
- Fallos notables en los documentos del primer sprint: clases y diagrama uml
- Grandes cambios en la estructura del programa, tipo de interfaz gráfica (javaFX a swing), lo que ha provocado que desechamos el proyecto inicial y empezáramos de nuevo

Aun con todos estos problemas, hemos conseguido que quede claro la estructura del programa, y hemos hecho los diagramas necesarios para aclarar la estructura. Además, hemos elegido los estándares de diseño de programa necesarios para llevarlo a cabo.

Review Sprint 3 - 19/04/2022

Durante este sprint, nuestra frecuencia de trabajo ha aumentado considerablemente, sumado a la ayuda de las vacaciones de Semana Santa. Hemos establecido unos horarios de trabajo fijos, y durante las vacaciones, hemos trabajado todas las mañanas, consiguiendo mejorar nuestro producto.

La organización de este sprint ha sido excelente, y hemos sabido dividir el trabajo y diversificar nuestras funciones, gracias a un diseño del programa mucho más claro, que nos ha permitido una comunicación fluida.

Con respecto al anterior sprint, los problemas de organización no nos permitieron entregar un producto funcional, además de los fallos en diseño. En este sprint hemos conseguido solucionarlo, los diseños están bien diseñados, hemos implementado mucho código y hay muchas funcionalidades bien implementadas.

En cuanto al diseño, hemos creado los primeros diagramas de clase fieles al código y a la idea del proyecto, en especial los relacionados con los patrones, como el de la DAO o del MVC.

En cuanto a código, hemos creado la mayoría de estructura de los patrones implementados. Así, han sido implementadas la DAO (especialmente de usuarios), el patrón Command para la entrada de datos por consola, el patrón Factorías para la creación de animales y el patrón State para los estados de los animales.

Como resultado, la aplicación ya es capaz de reconocer usuarios y registrarlos, aunque todavía queda por conectarlos correctamente con sus respectivos animales y con las pantallas, pero hemos conseguido establecer una base sólida para el posterior desarrollo.

Review Sprint 4 - 02/05/2022

Este sprint ha sido el sprint más productivo sin duda, gracias al aumento de la frecuencia de trabajo. Con la entrega final como objetivo, hemos conseguido implementar todas las funciones que nos permiten manejar los datos (tal y como nos establecimos de objetivo). No hemos conseguido implementar funcionalidades extra que teníamos contempladas como el contacto entre refugios y adoptantes o reviews.

En cuanto al diseño, hemos creado todos los diagramas de clase para cada uno de los patrones y para cada una de las historias de usuario. Además, para estas últimas tenemos propios diagramas de secuencia.

En cuanto al código, hemos implementado el patrón MVC (Modelo vista controlador) haciendo dos controllers, uno para los refugios y otro para los adoptantes. La interfaz la hemos realizado por

builders. En cuanto a la vista, la creación de las ventanas la hemos implementado con el patrón factorías complementado con el patrón builder.

1.4. Sprint Retrospectives

Sprint 1 Retrospective

En el sprint 1, el objetivo inicial ha sido estructurar el proyecto, decidiendo qué software hay que utilizar para estructurarlo correctamente y qué estructura de clases vamos a usar. Fue un error elegir JavaFX al principio, ya que no teníamos mucho conocimiento sobre cómo funcionaba y tampoco con que programa era apropiado desarrollarlo. Además, la organización de las reuniones no era correcta, ya que en la mayoría de estas no estaba el equipo al completo y se generan duplicidades a la hora de explicar las decisiones más importantes que habían que tomar al principio del proyecto. Más tarde solucionamos estos problemas.

Sprint 2 Retrospective

En temas de organización, hemos pecado mucho de realizar reuniones vacías y escasas, por lo que no quedaban los temas claros y en consecuencia el desarrollo se ha visto ralentizado. Por la parte de desarrollo, diseñamos modelos de dominio que nos ha ayudado a crear consenso en el desarrollo de código y arquitectura. También ha sido un error no diseñar la estructura de clases y los patrones de diseño a usar desde el principio, tuvimos que hacer muchas versiones de la estructura de clases, ya que cuando avanzamos en el proyecto, nos dábamos cuenta de los errores que tenía este.

Sprint 3 Retrospective

Hemos realizado cambios en la organización, creando horarios fijos y reuniones más cortas con cambios pequeños pero frecuentes. Por otro lado en temas de desarrollo de código y diseño hemos avanzado bastante quitando funcionalidades “extras” y reduciendo el proyecto a algo más sencillo que nos permita avanzar poco a poco y escalar a largo plazo. Hemos tenido problemas en cuanto al diseño en ocasiones pero se solucionaron la mayoría con pequeños cambios.

Sprint 4 Retrospective

En el sprint 4 teníamos el objetivo de completar la mayor cantidad de funcionalidades básicas del proyecto. Nos organizamos por subgrupos en función de las necesidades del proyecto (programación, interfaces, diagramas, documentos, etc.). También realizamos diferentes reuniones para actualizar al resto del equipo de los progresos de cada subgrupo y para poner ideas en común. Ha habido problemas con las continuas actualizaciones del repositorio GitHub del proyecto pero pudieron ser resueltas.

1.5. Sprint Plannings

Sprint 1 Planning

En cuanto a documentación, hemos planeado preparar la base de los documentos UML y la organización del proyecto. Pensamos que establecer unos objetivos, historias de usuario y diseño fijos y claros es bastante importante para que todos los integrantes entiendan el objetivo del proyecto.

En relación a lo implementado para el primer sprint, el objetivo es conseguir el manejo de los usuarios de forma correcta. Para su almacenamiento, queremos usar "*JSON Objects",* que nos facilitará cargar los documentos sin tener que aplicar bases de datos en nuestro proyecto.

Sprint 2 Planning

Después de los problemas del sprint 1, hemos replanteado el proyecto entero y hemos ido cambiando el enfoque de cada parte del proyecto, y ver qué patrones podríamos usar para hacer que todo funcione de forma correcta.

Para la interfaz gráfica, hemos decidido dejar de lado la idea inicial de usar javaFX y empezar a implementar la interfaz con swing, ya que tenemos conocimientos de cómo utilizarlo y tenemos más conocimientos. En cuanto a la estructura de objetos, hemos reorganizando los objetos, haciendo un diagrama de cómo se relacionan entre sí.

Sprint 3 Planning

El objetivo es comenzar a implementar la idea que fue comentada en el sprint 2, y cuya viabilidad estuvimos estudiando. Ya tenemos claros los objetivos del proyecto y las tecnologías (modelos, interfaces y IDE) que vamos a utilizar, por lo que comenzar a trabajar será mucho más sencillo.

En cuanto a historias de usuario a implementar, el objetivo es comenzar con un programa funcional, implementando los inicios de sesión. Para el manejo de los dos tipos de usuarios, hemos decidido crear dos ramas en el mismo programa, con sus respectivos controladores y vistas. Por lo tanto, el objetivo principal de este sprint es que los usuarios puedan iniciar sesión, y el programa sea capaz de identificar qué usuario ha iniciado sesión y cómo debe manejar esa información.

En cuanto a interfaces, para comenzar a tener feedback gráfico, el objetivo de este sprint es preparar bocetos de cómo queremos que sean las interfaces, para cuando comencemos a escribir código, todos tengamos la misma idea de interfaces.

Sprint 4 Planning

El Sprint 3 fue exitoso, por lo que el objetivo de este sprint es el de continuar implementando funcionalidades hasta que el programa esté terminado. La principal prioridad son las tareas relacionadas con las bases de datos, y más en específico las denominadas "CRUD" (*Create, Read, Update, Delete*), tanto de usuarios como de animales.

Para ello, tenemos que conseguir tener una interfaz funcional que nos permita realizar este tipo de operaciones de forma rápida e interactiva. Esta es una de las prioridades del sprint, mientras que la otra prioridad es establecer las funcionalidades del controller, es decir, desarrollar de forma back-end los supuestos establecidos por las pantallas.

En la parte de back-end mencionada, es necesario el control y manejo de datos, y la creación de las funciones correspondientes para crear, actualizar y eliminar. En especial esta última tiene complejidad, ya que debido a la estructura de nuestras DAO, los identificadores de los objetos pueden llegar a solaparse.

La meta del sprint es establecer todas estas funcionalidades y resolver los problemas correspondientes, para entregar un producto que cumpla, al menos, los requisitos mínimos. En caso sobrante de tiempo, nos gustaría implementar historias de usuario como la de las reviews o de contacto, pero las establecemos de prioridad baja.

1.6. Product Backlog

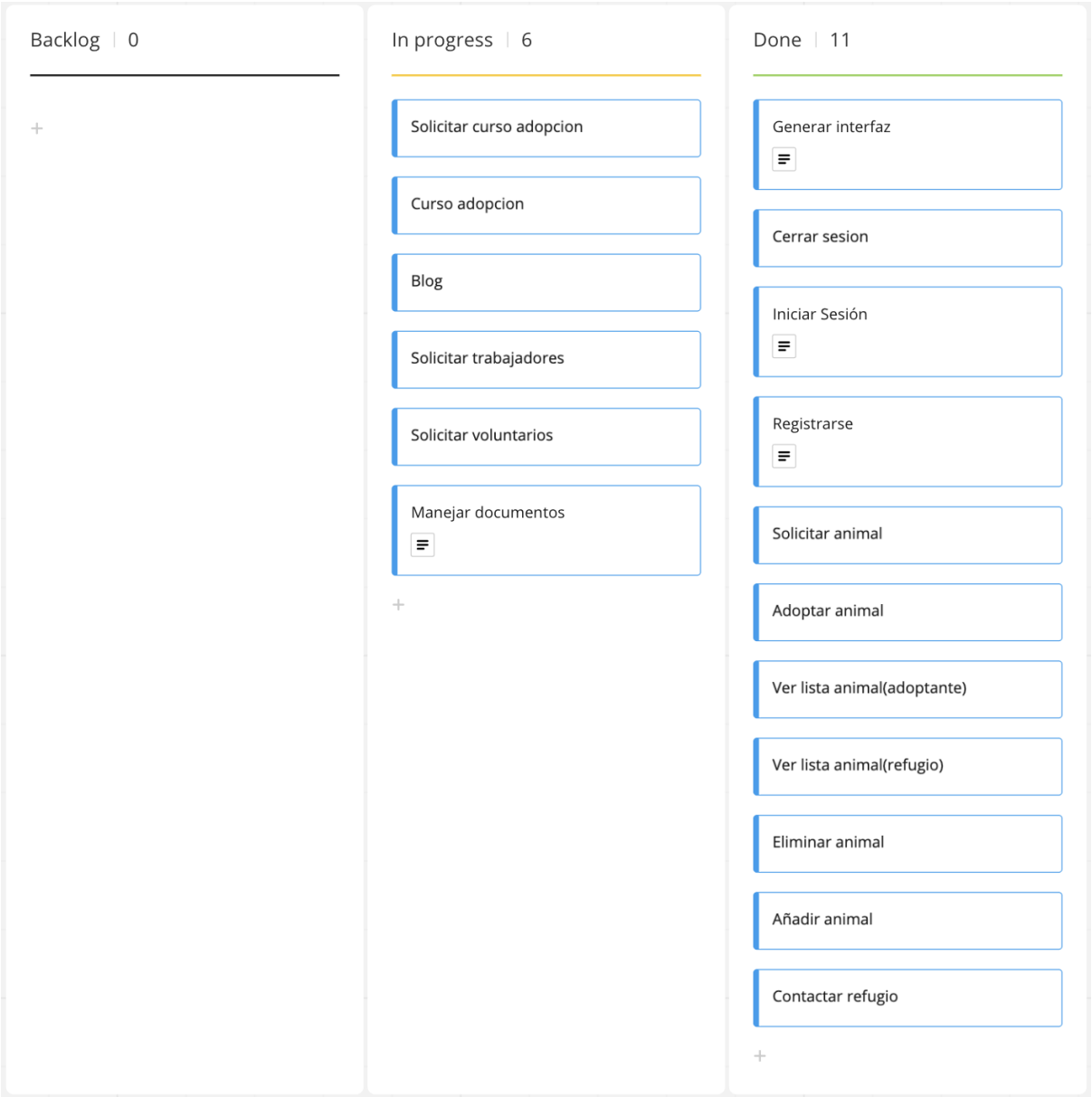
ID	HISTORIA DE USUARIO	PRIORIDAD	VALOR
1	Como usuario quiero poder registrarme para mantener mi cuenta y mi información	3	5
2	Como usuario quiero poder iniciar sesión para acceder a mi cuenta y a las funcionalidades de la aplicación	5	8
3	Como adoptante quiero poder comprobar los animales disponibles para decidir correctamente	2	6
4	Como adoptante quiero poder adoptar animales para contactar con el refugio y recibir el animal	5	7
5	Como refugio quiero poder añadir animales para aumentar mi refugio	4	2
6	Como administradores del refugio quiero poder ver la lista de animales que hay disponibles.	2	2
7	Como refugio, quiero poder solicitar voluntarios para mejorar la plantilla y el funcionamiento del refugio	1	1
8	Como usuario, quiero poder eliminar mi cuenta para preservar mis datos y finalizar mis adopciones.	4	3
9	Como adoptante, quiero poder acoger a un animal para cuidarlo y liberar al refugio hasta que alguien lo adopte	3	7

ID	HISTORIA DE USUARIO	PRIORIDAD	VALOR
10	Como adoptante, quiero poder apadrinar a un animal para dar dinero al refugio y apoyarlo	3	7
11	Como refugio quiero poder ofertar cursos de adopción para poder	2	6
12	Como refugio quiero poder publicar entradas en el blog para dar información y noticias	2	8
13	Como adoptante quiero poder solicitar realizar el examen para poder adoptar animales	1	6
14	Como adoptante quiero poder realizar donaciones libres para	1	4
15	Como adoptante quiero poder acceder a la FAQ para resolver las dudas	1	5
16	Como refugio quiero poder eliminar animales del refugio para no dar información falsa	4	3
17	Como refugio quiero poder editar la información de los animales para que estén actualizados	3	2
18	Como adoptante quiero poder dar y consultar reviews para valorar a los mejores refugios	2	6

ID	HISTORIA DE USUARIO	PRIORIDAD	VALOR
19	Como adoptante quiero poder hacer el curso de adopcion	1	5
20	Como adoptante quiero poder utilizar el servicio de guardería para ocasiones en las que no me pueda hacer cargo del animal	5	5
21	Como usuario quiero poder cerrar mi sesión.	5	2
22	Como centro de acogida, quiero poder solicitar nuevos trabajadores	1	1
23	Como usuario quiero poder contactar con el refugio	1	5
24	Como refugio quiero poder explicar los motivos por los que tenemos un refugio y darme a conocer.	1	5
25	Como usuario quiero poder buscar plazas para guardería de animales.	1	6
26	Como centro de acogida, quiero poder promocionar eventos sobre el centro.	1	3
27	Como candidato, quiero poder enviar el CV al centro de acogida	1	2

1.7. Sprint Backlog

El sprint backlog lo hemos elaborado a partir de un Tablero Kanban gracias a la herramienta de Miró. La tarea más fundamental terminada ha sido la interfaz de login, pues hemos tenido algunas dificultades para terminar de forma completa el resto de tareas.



1.8. Descripción del trabajo realizado por cada miembro del grupo

Sprint 1

El trabajo realizado por cada miembro del grupo está muy relacionado en este primer sprint con los roles establecidos en el punto 1.1. anterior.

Fabrizio Zeballos se ha encargado del diseño de la GUI junto a **Guillermo López**, y se han encargado de estudiar en profundidad el funcionamiento de JavaFX y han estudiado cómo implementarlo en nuestro proyecto. Además, junto a la ayuda de **Eneko Retolaza** han programado el

código correspondiente a la pantalla de inicio. Además también han preparado los objetos JSON y su estructura.

Javier Peña se ha encargado del diseño de los diagramas UML de clases y el diseño de los objetos básicos, como los animales o los usuarios.

Alonso Salgueiro y **Alberto Martínez** se han encargado de elaborar las historias de usuario y preparar los documentos de backlog e historias de usuarios, después del brainstorming y las ideas principales del proyecto. **Antonio Álvarez** les ha ayudado en la revisión de estos documentos, así como la planificación y asignación de tareas del proyecto. También ha trabajado en el ejecutable de GitHub junto a **Fabrizio Zeballos.**

Sprint 2

Hemos creado consenso en la arquitectura y diseño de la aplicación. **Eneko Retolaza, Antonio Álvarez** y **Guillermo López** se encargaron de desarrollar el modelo de dominio con la intención de satisfacer las necesidades del cliente (ficticio). **Javier Peña** y **Fabrizio Zeballos** se han encargado del diseño de los patrones creacionales de los apartados importantes del proyecto, enfocándose en alto nivel (diseño UML). **Alonso Salgueiro** y **Alberto Martínez** han seguido la misma metodología y han estado trabajando en la documentación sprint backlog, entre otros.

Eneko Retolaza y Fabrizio Zeballos han creado los JSON que se utilizarán como recursos para los datos de los usuarios y las entidades correspondientes.

Sprint 3

Durante este sprint, la diversificación de trabajo ha sido fundamental, y la mayoría de miembros ha trabajado en muchas partes del proyecto.

La arquitectura y diseño final ha sido consensuada, mientras que la implementación de los patrones ha sido diversa: **Antonio Álvarez** en el patrón command y la UserDao, **Eneko Retolaza** en la AnimalDao, **Fabrizio Zeballos** en el patrón state y el patrón factorías. En cuanto a las interfaces, la labor principal ha sido de **Guillermo López**, que se ha encargado de la implementación de las mismas y ha contado con la ayuda de **Alberto Martínez** y **Javi Peña** para su diseño y boceto. Mientras que Guillermo se encargó de la parte de la interfaz del adoptante, **Alonso Salguiero** se encargó de la parte del refugio, con su respectivo diseño.

En cuanto a los documentos, **Alberto** se encargó especialmente de cohesionar las historias de usuario con nuestro código, mientras que **Javi** se encargó de la misma tarea en relación a los diagramas de clases.

Sprint 4

De nuevo, una característica principal de la organización del trabajo ha sido la diversidad de ámbitos en los que cada miembro ha trabajado, haciendo que todos conozcan todas las partes del proyecto y formen parte del mismo. Aún así, han existido labores diferenciadas:

- **Antonio Álvarez:** Se ha encargado de la carga y guardado de archivos, la adaptación de la estructura de ventanas al refugio, elaborando parte de la implementación de las mismas. Además, se ha encargado de la estructura interna de los objetos como Animal y Users, y su adaptación hacia los controladores y carga de archivos.

- **Guillermo López:** Se especializó en establecer un sistema de ventanas sostenible, que permitiera un cambio constante estable y actualizado a los eventos del programa. Este sistema fue implementado en primer lugar en el Adoptante, y después migrado al Refugio. Utilizó factorías y builder para las ventanas, y el patrón MVC para sus funcionalidades.
- **Alberto Martínez:** Se ha encargado de la gestión de historias de usuario, descripción de documentos y colaboración al diseño.
- **Javier Peña:** Especialmente, se ha centrado en el diseño que nos precedió a la implementación, a la elaboración UML y se ha asegurado durante el desarrollo, que tanto los diagramas de secuencia como de clases se mantenían fieles al código.
- **Eneko Retolaza:** Se encargó de la elaboración de pruebas con JUnit y la revisión de código, junto a algunas implementaciones en los controladores.
- **Alonso Salgueiro:** Se ha encargado de la implementación de los refugios y la mayoría de funcionalidades, colaborando en la implementación de la factoría de Guillermo.
- **Fabrizio Zeballos:** Al igual que Eneko, se encargó también del JUnit y revisión de código. También se ha encargado de procesos en el controlador, como la eliminación de usuarios y su creación, también ha solucionado conflictos con la creación de IDs, y ha cambiado la interfaz para que sea mucho más agradable a la vista.

2. Documento de Diseño UML

IMPORTANTE: Por facilidad y apariencia del Documento Scrum, todos los diagramas UML de arquitectura y diseño se encuentran en la carpeta “Diagramas”, dentro del GitHub. Por favor, compruebe estos diagramas durante la lectura de este documento.

2.1. Arquitectura

La arquitectura está relacionada con el diseño en tanto que los documentos que describen los patrones, y en especial el MVC, se encuentran disponibles en la misma carpeta.

En especial, para especificar la arquitectura a alto nivel, hemos creado un modelo de dominio en el que expresamos las funcionalidades a la hora de crear la aplicación, pero sin entrar en detalle. En el mismo, expresamos que los adoptantes son capaces de adoptar animales, ya sean gatos o perros, y los refugios cuentan con animales propios y aceptan las solicitudes de adopción, Se encuentra toda la documentación en la carpeta diagramas.

2.2. Diseño

Los patrones de diseño utilizados han sido:

- **MVC**: Modelo Vista Controlador, ha sido el patrón de diseño principal utilizado de cara a la interacción entre los datos y las pantallas. Las clases principales del mismo han sido dos controladores, RefugeController y AdopterController, a las que se accede según el tipo de usuario que haya iniciado sesión. Además, para el correspondiente usuario, tenemos una serie de pantallas que son las que se relacionan con el controlador en cuestión. Los controladores cuentan con la información de los objetos (Usuarios y Animales) para respetar el modelo.
- **Patrón DAO**: Data Access Object, lo hemos implementado para el almacenamiento y control de Bases de Datos. Para ello, tenemos una interfaz que nos asegura que ambas DAO están bien implementadas y tienen las funcionalidades requeridas. De cara a los objetos, tenemos

dos DAO, la UserDao para el control de usuarios y la AnimalDao, para la creación de animales.

- **Patrón State:** Está implementado de cara a los animales, puesto que antes de ser adoptados, pasan por un proceso de solicitud. Dentro de este patrón, están los estados de Adoptado, No Adoptado y Adopción solicitada.
- **Patrón Command:** El patrón Command lo hemos aplicado para la entrada de datos por consola, ya que tenemos que reconocer muchos caracteres diferentes. Así, tenemos diversos comandos para los registros, el login, ayuda y exit. Para controlarlos, tenemos un generador que se encarga de parsearlos y comprobar que el comando introducido es fiel a lo implementado.
- **Patrón Factorías (2):** Es un patrón que hemos utilizado en dos ocasiones. La primera ha sido utilizado para la creación de animales, que a pesar de tener Perros y Gatos, el trato y la pertenencia era a la clase Animal, por lo que era necesario una factoría que interpretase los json y crease el Objeto correspondiente.

  El otro uso que le hemos dado a las factorías ha sido mediante las ventanas. El patrón Factorías en este caso se ha visto complementado con el patrón Builder. Las factorías se han utilizado mediante códigos que indicaban la ventana que se encontraba activa, y mediante un frame común en el controller, se iba cargando y eliminando la pantalla que correspondía.

- **Patrón Builder:** Lo hemos utilizado como hemos mencionado anteriormente para apoyar a las factorías en el caso de las ventanas. Llamábamos a un constructor común, y cada una de las ventanas tenía su propio Builder que implementan su constructor correspondiente. El modelo era homogéneo para adoptantes y refugios al mismo tiempo, por lo que conseguimos unificar código de forma eficaz.

Todos los diagramas de clase de los respectivos patrones se encuentran en la carpeta de diagramas mencionada anteriormente.

3. Código

El código se encuentra en el mismo repositorio de GitHub que este documento, y está accesible para todos los colaboradores, pues se trata de un repositorio privado.

4. Referencias

A continuación se encuentran enlaces e información extra acerca de las herramientas utilizadas:

- GitHub: https://github.com
- Google Drive: https://www.google.com/intl/es_es/drive/
- Discord: https://discord.com
- Miró: https://miro.com
- Figma: https://www.figma.com