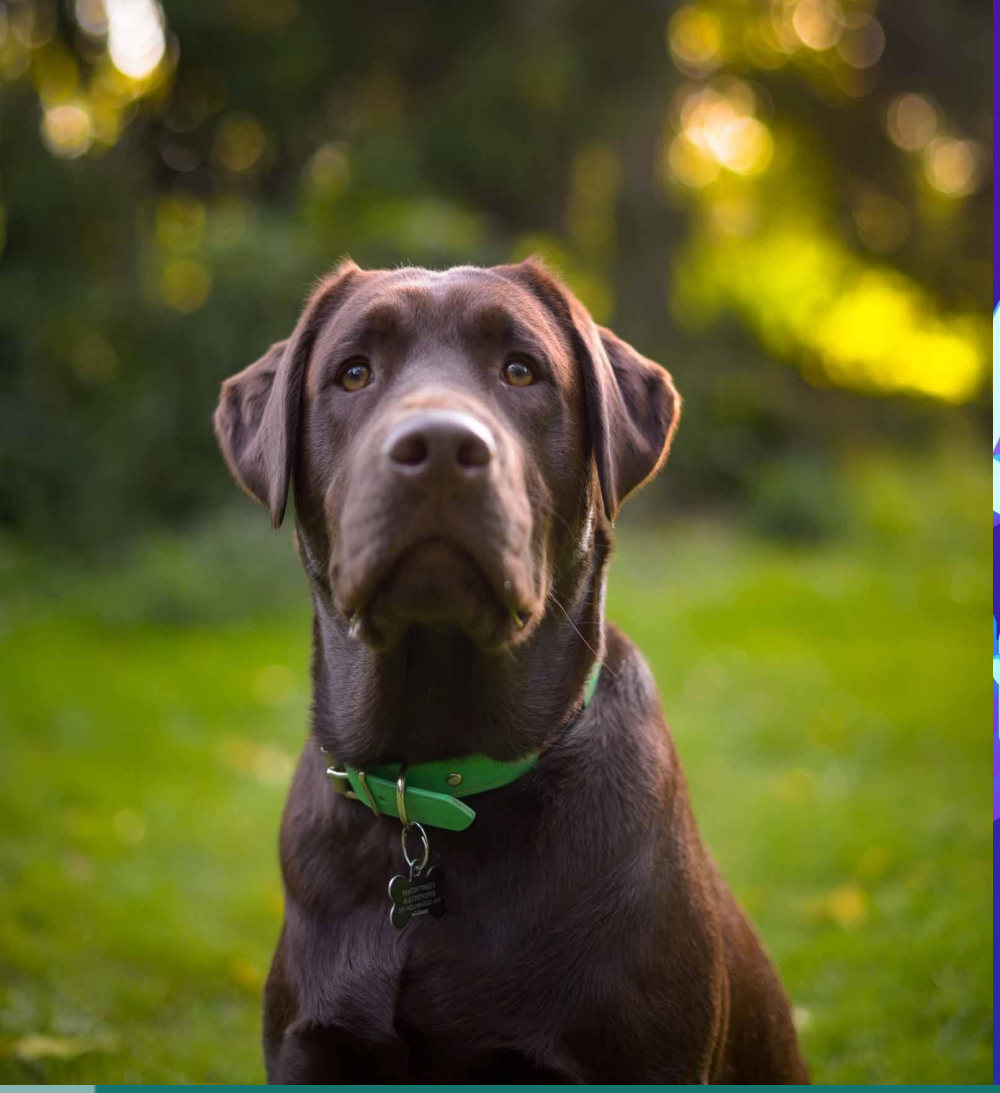

PRÁCTICA 5
PROTOTIPO TECNOLÓGICO

GRUPO SUMMERPET

ALUMNOS:
Alex Guillermo Bonilla Taco
Elena Caridad Zingoni
Marcos Gago Rivas
Jesús González Carrillo
Jaime Jiménez Nieto
Miguel Martínez-Almeida Nistal
Ignacio Redondo Navarro
Luis Vizán Morales
Miguel Marcos Asenjo
Iván Pisonero Díaz

ÍNDICE

1. INTRODUCCIÓN

En esta práctica, se tratará de diseñar un prototipo tecnológico por el equipo de desarrollo, con herramientas sencillas, que ayuden y no compliquen nuestro objetivo.
Además, se incluirán los cambios realizados por el Scrum Master y Product Owner, en el User Story Map, Product Backlog y las Historias de Usuario.
Y por último, se detalla una dinámica grupal, que ayuda a la optimización del tiempo en trabajo de equipo.

2. CAMBIOS USER STORY MAP

La revisión del User Story Map ha sido llevada a cabo por el Scrum Master y el Product Owner, trabajando de manera conjunta de forma online. Fue necesario establecer algunas reuniones entre los mismos, para hablar sobre cada unos de los cambios propuestos y trabajar simultáneamente de forma eficaz ante las posibles dudas o problemas que surgieran durante esta revisión. Ningún cambio provocó debate ya que todos eran una corrección lógica a los resultados obtenidos en la anterior entrega del mapa de historias de usuario. A continuación se explican dichos cambios tanto en actividades y HU como en los diferentes RELEASES y MVP.

2.1. ACTIVIDADES Y HU

Respecto a las actividades, se ha decidido los siguientes cambios:

1. Se ha dividido "Valorar" en dos actividades distintas, "Valorar cuidador", que sólo pueden hacerla los dueños y "Valorar dueño", que sólo pueden hacerla los cuidadores. Se ha tomado esta decisión porque son actividades que tienen lugar en momentos diferentes del costumer journey y separarlas en dos muestra con más claridad cuándo sucede y qué aporta a cada tipo de usuario.

2. Se ha movido "Registrarse" e "Iniciar sesión" a cuando el dueño quiera solicitar servicios, ya que es el momento en el que se le exigirá registrarse. Lo único que puede hacer sin registrarse es ver cuidadores y sus perfiles.

3. Se ha dividido "Modificar perfil" en "Modificar perfil", realizable por cuidadores y dueños al ser la modificación de los datos estándar; y "Modificar perfil de cuidador", sólo realizable por cuidadores. Bajo esta actividad se han colocado las HU "Modificar calendario de disponibilidad", "Cambiar sueldo", "Cambiar fotos de entorno del cuidador" y "Modificar especialidades de cuidador". Es decir, todo lo que anteriormente en "Modificar perfil" pero sólo era realizable por cuidadores. Esta división aclara y diferencia mejor lo que puede hacer cada tipo de usuario en su perfil.

4. "Marcar como favorito" ha pasado a ser "Marcar cuidador/dueño como favorito", para aclarar qué se hace en esta actividad.

5. “Ponerse en contacto” ha pasado a ser “Mensaje en la aplicación”, porque es la actividad que realizará de verdad a través de Summer Pet, ponerse en contacto es ambiguo y puede hacerse de muchas maneras.

Con respecto a las HU. Se ha renombrado “Solicitar básico” a “Solicitar servicios” porque no hay una forma básica y otra compleja de realizarlo, se realiza siempre igual que es indicando los días que quieres que cuiden tus mascotas; “Ver valoraciones anteriores” a “Ver valoraciones”; “Eliminar básico” a “Eliminar perfil”; “Contactar por mensaje/correo” como “Contactar”; “Confirmar” ahora es “Confirmar finalización de servicio”. Todos estos cambios se han llevado a cabo con el objetivo de hacer los nombres de las HU más claros y descriptivos. Además, “Marcar como favorito” ha pasado a ser “Marcar cuidador/dueño como favorito”, para aclarar su funcionalidad. Por otro lado, en la actividad “Darse de baja como cuidador” se ha eliminado las HU “Baja temporal” y “Baja permanente” porque no tenía sentido la diferenciación y se ha dejado solamente y “Baja cuidador” que cumple la funcionalidad de indicar que un usuario quiere dejar de ser cuidador en la aplicación. Por último, se ha eliminado la HU “Ordenar cuidadores Aleatorio” porque no tiene sentido, eso no es ordenar; y se ha eliminado “Buscar cuidador por nombre” porque eso se incluye en buscar por palabra clave, el nombre es una palabra clave por lo que no es necesaria esa HU. También se ha eliminado “Ordenar cuidadores por especialidad” porque la especialidad es un campo de filtrado.

2.2. MVP Y RELEASES

Se han realizado cambios importantes en el MVP para hacerlo más Lean. Lo básico que debe poder hacer SummerPet es facilitar a los dueños la búsqueda de cuidadores, ser un portal al que acceder para encontrar a alguien que cuide de sus mascotas. Nos dimos cuenta que para realizar eso no eran necesarias las funcionalidades de contratar y valorar en el MVP. Lo único necesario en esta fase de desarrollo es que se pudiesen buscar y ver cuidadores y se pudiese ver su perfil para decidir si quieres solicitar los servicios. La funcionalidad de solicitar servicios es necesaria por razones de privacidad, para evitar mostrar los datos de contacto del cuidador a todos los usuarios. Sólo se muestran a aquellos dueños a los que el cuidador les acepte una solicitud de servicios, para mostrarles sus datos de contacto y que puedan comunicarse.

Tras tener claro qué debía hacer y qué HUs habría en el MVP se pasó a repensar el objetivo de la RELEASE 2. Se llegó a la conclusión de que lo siguiente más prioritario es aumentar la confianza de los dueños al contratar para que hubiese más dueños de mascotas usando SummerPet tranquilamente. Por eso, el objetivo de la RELEASE 2 es añadir más campos al perfil del cuidador y la posibilidad de contratar y valorar para que los dueños dispongan de más información para decidir con más confianza. Además, se añadirán opciones de ordenación y filtrado en base a los nuevos datos de perfil para hacer la experiencia de búsqueda más eficiente. Esta RELEASE se centra en el dueño, pero también hemos querido añadir la HU de valorar dueño para que los cuidadores no se sientan abandonados, que también tengan voz dentro de la aplicación si un dueño no les tratase adecuadamente.

Una vez aumentada la confianza de los dueños e implementadas suficientes opciones de búsqueda, lo siguiente que se implementará es la interacción entre usuarios dentro de SummerPet y este será el objetivo de la RELEASE 3. Para alcanzarlo, se implementan las HU de “Contactar por mensaje en la aplicación”, “Reportar”, “Bloquear” y “Añadir a favoritos”. Además, en esta RELEASE se añaden

funcionalidades básicas como el cerrar sesión y los registros por Google y Facebook para dar más facilidades a los usuarios, son funcionalidades básicas que aunque no están relacionadas con el objetivo de la RELEASE 3 no tendría sentido retrasar más porque son muy básicas.

Por último, en la RELEASE 4 se implementarán los pagos a través de la aplicación, lo cual da más seguridad tanto a dueños y cuidadores de que no se sufrirán estafas o engaños. Es una funcionalidad importante, pero menos prioritaria que las de las RELEASES anteriores ya que SummerPet puede ser completamente operativa y funcional sin ella, pero aporta valor a los usuarios.

3. PRODUCT BACKLOG

Se han realizado numerosos cambios en el Product Backlog de SummerPet durante el desarrollo de esta práctica. Todos ellos han sido llevados a cabo por el Product Owner y el Scrum Master, quienes trabajaron de manera conjunta. Para modificar el Product Owner se encargó de realizar las correcciones sobre las HU pertenecientes al MVP y el Scrum Master fue el encargado de generar las HU de la RELEASE 2, de esta manera el trabajo se realizó de una manera más eficiente al trabajar en dos partes independientes de manera simultánea sin ningún tipo de conflicto ya que cada uno de los cambios realizados además, era informado inmediatamente. En los siguientes apartados se explica en detalle cómo y por qué se realizaron los cambios, cómo se realizaron las priorizaciones y cómo se validó el PB.

3.1. CAMBIOS

Los cambios se realizaron sobre las HU que pertenecen al MVP, por lo tanto fueron responsabilidad del Product Owner. Lo primero que se hizo fue separar las HU habían dejado de ser parte del MVP. Debido a los cambios realizados en el User Story Map, todas las HU relacionadas con las actividades con Contratar, valorar y contactar así como el ordenar solicitudes han pasado a formar parte del RELEASE 2.

Además se decidió desglosar las HU de “Ver perfil” y “Ver valoraciones” para diferenciar cuando un dueño o cuidador ve el de otro dueño o cuidador de cuando ve el suyo propio ya que los beneficios que busca son diferentes. En el caso de ver el perfil propio hemos considerado que podrían unirse ambos tipos de usuario en una HU porque buscan el mismo beneficio.

Una vez se tuvieron claramente separadas las HU que, por ahora, forman parte del MVP de las que forman parte del RELEASE 2 se pasó a realizar la corrección de los títulos. Tuvieron que cambiarse todos, ya que todos eran incorrectos. Con los nuevos títulos se ha intentado conseguir que sean más cortos y claros que los anteriores. Para conseguir esto, se han hecho lo más similares a su respectiva HU del User Story Map. Un ejemplo representativo es el cambio de la HU para darse de alta como cuidador. Previamente su título era “Añadir botón para darse de alta como cuidador en el perfil del cuidador”, con excesivo enfoque en el “cómo” en lugar del qué y siendo demasiado largo. Ha pasado a ser “Darse de alta básico como cuidador”, centrado en el “qué” y más corto y conciso.

El PO y el Scrum Master tuvieron que ponerse de acuerdo para determinar cómo sería el título de las HU que son realizables por usuarios y cuidadores. Decidieron que una forma clara y simple de

diferenciarlas sería especificar en el propio título para quién es esa HU. Por ejemplo teniendo las HU “Modificar foto de perfil para cuidador” y “Modificar foto de perfil para dueño”.

Con respecto a las descripciones, se ha mantenido el mismo formato “Como <usuario> quiero <funcionalidad> para <beneficio>” pero se han tenido que corregir la mayoría de las existentes. El principal problema de las antiguas descripciones estaba en los beneficios. Algunas no tenían y en otras era incorrecto. Esto se debió a una mala organización en la creación del PB. En la nueva versión se ha intentado mantener las descripciones cortas y tanto el Product Owner como el Scrum Master se han dado feedback mutuo para pensar el beneficio real que obtiene cada usuario. Otro problema que se ha solucionado gracias al feedback de la anterior práctica y los ejemplos de clase es la diferenciación entre los beneficios que reciben dueños por un lado y cuidadores por otro. Todas estas correcciones y mejoras en las descripciones y títulos se han aplicado en la creación de las HU para la RELEASE 2, asegurando que no se repitieran en estas los fallos cometidos en la primera versión del PB.

El siguiente apartado que tuvo que ser corregido son los criterios de aceptación. Al igual que con los títulos, prácticamente todos estaban mal. De nuevo tenían elementos de diseño en lugar de servir de una guía clara para que el desarrollador implementara como viera oportuno. Para lograr esta mejora se han creado CA más específicos, que detallen cómo debe ser el funcionamiento y en los casos que haya formularios a rellenar indique los requisitos de los campos a rellenar. También se han eliminado los conceptos ambiguos como “correcto”, por ejemplo en la HU de “Registrarse” se especifica cómo se quiere el correo, en lugar de poner que se debe recibir un correo “correcto”.

Por último, se han modificado los identificadores. Como los de la antigua versión no estaban correctamente adjudicados a las HU diferentes y no diferenciaban claramente unas HU de otras se han borrado todos y se ha pensado un formato nuevo. El nuevo formato es SMPT-#. “SMPT” es el código del proyecto SummerPet. Después, # será el número que la identifica, que en el caso de las HU del MVP se ha adjudicado por orden según se iban corrigiendo y en las de la RELEASE 2 por orden según iban siendo creadas.

Cabe destacar que los números de las HU de la RELEASE 2 se adjudicaron a partir del último que ocuparía el MVP, pero eso no tiene nada que ver con la prioridad de las HU ya que esta puede cambiar.

3.2. PRIORIZACIÓN

A la hora de priorizar el PB se ha usado como guía principal los objetivos que tendrían respectivamente el MVP y la RELEASE 2. Teniendo en cuenta que el objetivo del MVP ,explicado en el punto 2.2, es facilitar la búsqueda y el contacto con cuidadores a los dueños, se ha decidido que las principales HU serán buscar para dueño y enviar solicitud, seguida de aceptar solicitud. Las solicitudes son importantes por cuestiones de privacidad, son las que permitirán ver los datos de contacto.

Una vez implementadas las funcionalidades para contactar lo siguiente más importante es aumentar la confianza del dueño al querer contactar y por eso se priorizan las HU relacionadas con visualizar el perfil del cuidador. Esto es crucial porque si los dueños no confían en la aplicación no habrá cuidadores que se muestren en ella. Teniendo a los dueños más cómodos en la aplicación, el

siguiente paso es atraer a cuidadores y para ello se implementa que puedan ver el perfil de los dueños y que puedan darse de alta. Como al darse de alta puede introducir datos erróneos es importante que pueda modificar su calendario de disponibilidad y ver su propio perfil para comprobar que sus datos de contacto y localización son correctos. Implementado todo esto, se vuelven prioritarios que tanto dueños como cuidadores puedan registrarse e iniciar sesión para que puedan acceder a la aplicación con sus propios perfiles, sin estas funcionalidades es imposible mantener a los usuarios. Finalmente, las funcionalidades menos prioritarias del MVP son las relacionadas con búsquedas y visualizaciones de perfil de cuidadores realizadas por los propios cuidadores ya que su principal beneficio será mejorar su propio perfil pero por el momento no dispondrán de la funcionalidad para modificarlo.

Por otro lado, el objetivo principal de la RELEASE 2 es aumentar la confianza del dueño, por eso lo primero que se implementan son las valoraciones de dueño a cuidador y el poder buscar por valoraciones. Consideramos que las valoraciones son lo que más confianza dará a los dueños. También se expanden los campos del perfil de cuidador y se añaden las funcionalidades para modificar esos campos para que los dueños puedan disponer de más información sobre los cuidadores lo cual promueve la confianza. De las HU restantes, las que más confianza aportarán al dueño son las relacionadas con los Contratos, ya que establece el compromiso más allá de un acuerdo verbal entre dueño y cuidador. Con estas funcionalidades se aumentan las fuentes de información del dueño lo cual debe darle más confianza. Ahora el foco de la RELEASE pasa a mejorar la experiencia del cuidador y para eso se implementan las valoraciones a dueños así como nuevas funcionalidades para darse de alta y modificar otros aspectos de su perfil. Los darse de alta son menos importantes que los modificar porque una vez realizado el alta básico como cuidador podría modificar todos esos campos en su perfil. El darse de alta con ellos le aporta comodidad. Lo siguiente a implementar será el cancelar contrato con explicación, muy alejado del resto de funcionalidades de contratar porque la explicación da valor pero no es imprescindible.

Finalmente, lo menos prioritario de la RELEASE 2 son funcionalidades que aportan poca confianza a dueños y cuidadores como el ordenar sus solicitudes o que el cuidador pueda disfrutar de las nuevas funcionalidades de búsqueda de cuidadores.

3.3. VALIDACIÓN

Para validar se ha intentado asegurar que todas las historias de usuario tengan las características INVEST. El proceso de validación ha sido llevado a cabo por el Product Owner y revisado por el Scrum Master. El Product Owner se ha ayudado del equipo de desarrollo para comprobar si cumplían las características de estimables y testeables. La primera característica que se comprobó que cumpliesen es la de “Small”. Se asumió que lo cumplían si para esa HU podía crearse un título corto y concreto y hacerse unos CA que explicaran qué debía hacer la HU. Esto se cumplía para todas las HU del PB del MVP ya que en el proceso de creación del PB se había puesto gran foco en poder definir las HU con títulos concretos y en generar CA lo más claros y explicativos posibles, por lo tanto se consideró que las del MVP sí lo cumplen.

Para las de la RELEASE 2 todavía no se han determinado los CA por lo tanto no se comprobó que cumpliesen esta característica. Después se pasó a comprobar que eran valiosas. Se asumió que lo eran si se podía crear un beneficio para el usuario al realizar la descripción. La única HU que presentó

problemas fue la HU “Contratar” que no tenía un beneficio claro de primeras, hasta que el PO pensó que el beneficio que aporta al dueño el poder contratar es establecer un compromiso entre él y el cuidador lo cual le da más seguridad.

También causó dudas las funcionalidades de buscar y filtrar cuidadores para cuidadores ya que el valor que aportan es escaso. Sin embargo, sí que lo aportan y los cuidadores las utilizarán por lo tanto lo que se hizo darlas baja prioridad. Para comprobar que fuesen testeables lo que se hizo fue pensar un test que podría pasar la HU. Por ejemplo, para testear si se cumple la HU de filtrar por valoración el test sería precisamente comprobar que la aplicación filtra correctamente según la valoración que pida el usuario por lo que sería fácil de comprobar.

Para todas las HU se pudo comprobar que existía un test posible y por lo tanto se asumió que son testables. Comprobar que son estimables resultó más problemático debido a que todavía no hemos adquirido conocimientos en clase sobre cómo estimar. Para realizarlo se aprovecharon los conocimientos de Jaime quién al ser el que más sabe de construir aplicaciones del equipo es el que mejor puede hacerse una idea de si una funcionalidad es estimable o no. Teniendo en cuenta lo anterior, el Product Owner se reunió con Jaime y revisaron que todas las HU fuesen estimables y se llegó a la conclusión de que las del MVP sí lo son ya que todas son funcionalidades concretas y pequeñas con criterios de aceptación.

Sobre las de la RELEASE 2 al no haber criterios de aceptación la afirmación de que son estimables no es tan fiable, pero de momento lo más prioritario era validar que las HU del MVP ya que serán las primeras en ser implementadas y las primeras que habrá que estimar. Finalmente, la característica que sí provocó que hubiese que hacer cambios en algunas HU fue la de “Independencia”. Muchas HU estaban expresadas de tal forma que necesitaban de otra para funcionar. Por ejemplo, la HU de “Darse de alta como cuidador” especificaba que lo realizaría desde la vista de perfil, lo cual la hacía dependiente de “Ver perfil”. Este tipo de errores era común en numerosos CA y se corrigieron, asegurando que las HU fuesen realizables por sí mismas, sin nombrar ninguna otra HU en sus CA.

En general, el proceso de validación ha servido para mejorar la calidad del PB ya que ha permitido revisar fallos de la versión anterior y evitar que se repitieran. Se ha puesto especial esfuerzo en las HU del MVP, por lo que se espera que a la hora de redactar los CA de las HU de la RELEASE 2 haya cambios y se solucionen errores que surjan en las HU.

4. PROTOTIPO TECNOLÓGICO

A continuación se explica todo lo referente al prototipo tecnológico el cual ha implementado la primera historia de usuario: ordenar cuidadores.

Se empezará explicando el modelo de arquitectura utilizado seguido del proceso que se ha llevado a cabo para desarrollarla. La siguiente subsección explica la implementación de esta historia de usuario comentando los problemas y soluciones que se han encontrado a la hora de desarrollarla. Por último se aporta un manual de usuario para explicar la manera de ejecutar este prototipo así como el funcionamiento del mismo.

4.1. MODELO DE ARQUITECTURA

La arquitectura que hemos diseñado (Imagen 4.1.1) para nuestra aplicación se basa en el modelo cliente-servidor. Así, los clientes interactúan con la aplicación por medio de la realización de peticiones, que serán recogidas por el servidor, el encargado de realizar un procesamiento derivado de esas peticiones y elaborar respuestas. Como la aplicación web se encuentra desarrollada de manera local, no es necesario la conexión a una red que permita tener acceso a Internet. Para gestionar este modelo de comunicación cliente-servidor usaremos Apache, un software que facilita la implementación de un servidor web, capaz de recibir, manejar y responder peticiones usando el protocolo HTTP.

El cliente necesitará la tecnología que ofrece Node.js así como poder acceder y abrir los puertos correspondientes y crear la base de datos con el script proporcionado. También necesitará un navegador web para poder usar la aplicación (aparte del acceso a Internet), es decir, una herramienta que le permita solicitar recursos web al servidor (peticiones), e interpretar las respuestas, que generalmente serán documentos HTML (Imagen 4.1.2).

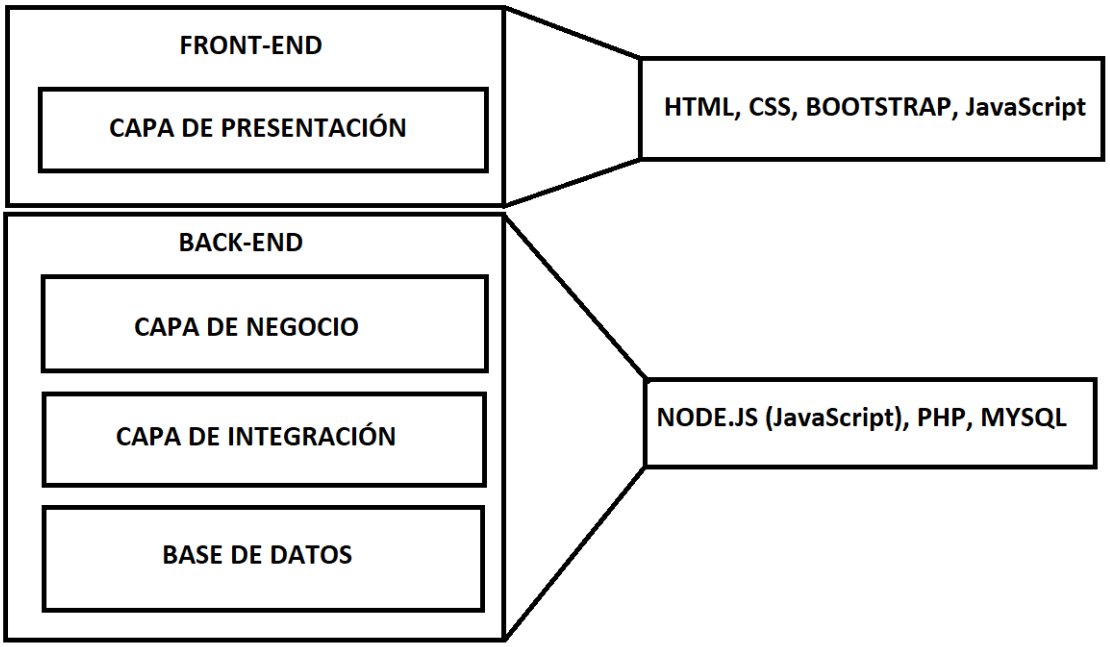

Imagen 4.1.2. Implementación del Frontend y Backend

Por otra parte, es en el servidor dónde se necesitará una mayor complejidad arquitectónica para garantizar el correcto funcionamiento de nuestra aplicación. La utilización de la arquitectura multicapa, que divide nuestra aplicación en datos, integración, negocio y presentación, nos proporcionará un modelo sólido, que facilitará enormemente la ampliación de las funcionalidades de nuestra aplicación y el mantenimiento de la misma, dos actividades que consideramos de vital importancia para nuestro proyecto. En primer lugar, será imprescindible disponer de una base de datos, puesto que es central para la aplicación poder almacenar todos los datos derivados de la interacción con los clientes (nombres de usuario, contraseñas, solicitudes, mensajes, contratos, valoraciones…), así cómo poder acceder a ellos. Para la creación y gestión de la base de datos,

usaremos MySQL, un sistema de gestión de bases de datos relacionales, que usaremos junto con PHP, un lenguaje de scripting que nos permite ejecutar scripts en el servidor. PHP y MySQL son dos herramientas que nos permitirán acceder a los datos, gestionarlos, y modificarlos, funcionalidades propias de la capa de datos / integración, y que forman parte del back-end junto con la capa de negocio. En cuanto a las capas de integración y negocio, usaremos especialmente el lenguaje JavaScript, un lenguaje de programación que nos permite usar programación orientada a objetos, y es de enorme utilidad para aplicaciones web. Con objeto de proporcionar solidez a nuestra aplicación, así como facilitar el proceso de desarrollo y la ampliación de funcionalidades, usaremos patrones de desarrollo propios de la arquitectura multicapa. Usamos, por ejemplo, el Data Access Object, o DAO, para encapsular el acceso a la capa de datos, de forma que la capa de negocio queda totalmente desacoplada de la capa de datos, lo que nos hará más ágiles a la hora de adaptarnos a cambios en la gestión de los datos almacenados o a la hora de ampliar funcionalidades. Para poder ejecutar JavaScript en nuestro servidor instalaremos Node.js, un entorno que precisamente lleva a cabo esta función.

Profundizando ahora en el front-end, usaremos tanto HTML como EJS, ambos lenguajes se utilizan para estructurar y desplegar los contenidos de una página web, junto con CSS, un lenguaje que permite especificar cómo se dispondrán los contenidos, descritos en HTML y EJS (dependiendo de la circunstancia), en la propia página web. Para agilizar el desarrollo de esta parte de la aplicación usaremos Bootstrap, un framework de HTML, CSS y JavaScript.

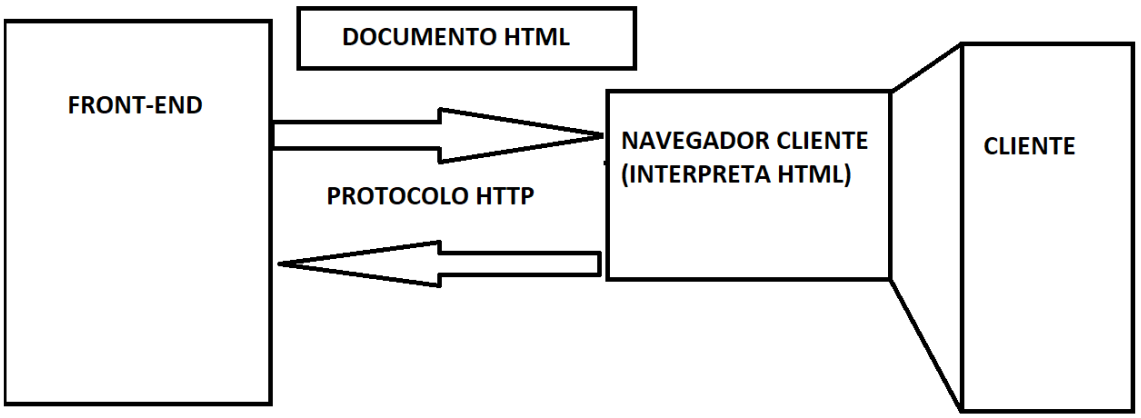

Imagen 4.1.2 Interacción del cliente con la aplicación

4.2. PROCESO

El proceso de creación del prototipo tecnológico se llevó a cabo en aproximadamente dos horas en las cuales participaron todos los integrantes del equipo de desarrolladores de manera online. Debido a que los desarrolladores Jaime Jiménez y Luis Vizán tienen más conocimientos de aplicaciones web fueron los encargados de guiar a los demás integrantes a la hora de implementar el prototipo tecnológico.

El primer tema que se trató en la reunión fue la base de datos con sus respectivas tablas. El equipo decidió de manera unánime que, para que el prototipo funcione de manera correcta era necesario la creación de una única tabla denominada Usuario. Esta tabla contendrá los valores del nombre,

dirección, correo electrónico y teléfono los cuales son los únicos que permitirán al resto de usuarios obtener información acerca de los cuidadores disponibles. Por ello, se creó la base de datos en PHPMyAdmin con la tabla mencionada y con algunos datos de prueba.

A continuación, se creó el repositorio Github para almacenar tanto la documentación como el código de la aplicación y se añadieron a los integrantes del grupo como contribuidores. Dicho repositorio se localiza en: https://github.com/JaimeJimeenez/SummerPet

Una vez creado el repositorio, los miembros del equipo ya disponían del gestor de versiones git en sus equipos así que se clonó dicho repositorio de manera local para poder empezar a desarrollar la primera historia de usuario. Antes de empezar con la parte de aplicaciones web, el equipo decidió crear en el repositorio local las dos ramas restantes necesarias para el correcto desarrollo del prototipo (release y devs) ya que la rama main se creó de manera automática al crear el repositorio y cuyos nombres ya se especificaron en prácticas anteriores. También se creó de manera local la rama cuyo nombre sigue la nomenclatura que el equipo declaró en la segunda práctica (DEV-BuscarPalabraClave). Esta rama, creada sobre devs, se encarga de contener todas las implementaciones y cambios que se realicen en esta práctica. Todas estas implementaciones se realizaron a través de un terminal y usando los comandos que git proporciona.

Una vez terminada la creación de las ramas los dos miembros del equipo que poseían conocimientos de aplicaciones web explicaron a los demás integrantes de una manera rápida y resumida los conceptos básicos que iban a ser necesarios para desarrollar esta primera tarea para que, en prácticas posteriores, sean capaces de implementar historias de usuario por su cuenta. Se trataron los temas de servidores, herramientas a utilizar, páginas estáticas, dinámicas, conexiones a la base de datos, lenguajes de programación a usar e implementación del gestor de paquetes npm. Cuando el equipo había comprendido estos conceptos y no quedaba ninguna duda se pasó a la implementación de esta.

Antes de empezar, se crearon las diferentes carpetas que almacenan tanto la documentación como el código. Estas carpetas siguen la estructura especificada en el Plan de Configuración de Software. Se modificó el nombre de DOCS a Documentos de Configuración de Software ya que el equipo estuvo de acuerdo en que ese nombre era más apropiado.

Ya terminado de estructurar todo el espacio de trabajo, Jaime y Luis indicaron al resto que debían descargarse las herramientas y paquetes necesarios para poder empezar a desarrollar la historia de usuario: node.js, mysql, ejs, nodemon y express. Los paquetes mysql, ejs, nodemon y express se descargaron por el gestor de paquetes npm. Node.js se descargó accediendo a la página oficial del mismo.

Ya descargados todos los paquetes se empezó con la parte backend, en concreto con la configuración del acceso a la base de datos y el puerto que iba a utilizar el prototipo para crear el servidor y con el importe de los paquetes mencionados anteriormente en el fichero app.js. Una vez terminado esto, se decidió subir los cambios realizados (a la rama destinada a almacenar estos cambios) al repositorio remoto a través de un terminal. Dado que las ramas anteriormente existían en el repositorio local, al realizar este commit se crearon todas las mencionadas anteriormente.

Los siguientes commits estuvieron enfocados a disponer de un script de la base de datos y desarrollar la parte frontend así como acabar la backend. Una vez se consideró que la historia de

usuario se encontraba acabada se realizó un merge a la rama main. No se realizó este merge en la rama release ya que la historia de usuario se considera un prototipo y no una versión obtenida de un sprint.

Una vez terminada la primera historia de usuario, el equipo realizó un video enseñando el prototipo al Product Owner y al Scrum Master con el fin de que estos analizasen la funcionalidad y el trabajo realizado por los desarrolladores.

Además, para comprobar que el manual de usuario estuviese bien redactado y diese una información clara y concisa al usuario, el Scrum Master junto con el Product Owner decidieron seguir el manual descrito por los desarrolladores para ver si podían llegar hasta el objetivo que el equipo de desarrollo pretendía con este manual sin ningún tipo de confusión o problema durante el proceso. Después de realizar este proceso el Scrum Master y el Product Owner establecieron una reunión con el equipo de desarrollo para comentar aquellos posibles problemas que surgieron al seguir el manual de usuario propuesto y mejorarlo para que fuese lo más cómodo y claro posible hacia los usuarios.

4.3. IMPLEMENTACIÓN

El equipo de desarrolladores empezó a implementar el prototipo tecnológico una vez realizado los preparativos explicados en el subapartado anterior.
Esta implementación no tuvo ningún problema excepto dos: el desconocimiento de la mayor parte de los integrantes del equipo en el campo de aplicaciones web y la búsqueda de un servidor remoto capaz de almacenar el prototipo así como la base de datos. Para solucionar el primer problema, el desarrollador Jaime Jiménez fue quien escribió el código necesario para la implementación al mismo tiempo que iba explicando qué hacía en cada momento. De esta manera, el equipo ganó cierta confianza para que en un futuro desarrollasen e implementasen historias de usuario por su cuenta y realizaron apuntes que se compartieron con todo el grupo a modo de recordatorios de ciertos conceptos que pueden resultar complicados de entender en un principio. Para resolver el segundo contratiempo, el equipo decidió buscar páginas web que ofrecieran servidores de hosting con el fin de almacenar el prototipo pero no encontró ninguna que pudiese soportar las tecnologías que el equipo está utilizando para implementar la aplicación. Por ello, el equipo decidió de manera unánime trasladar toda la aplicación de forma local, proporcionando en el manual de usuario toda la información necesaria para poder instalar el prototipo en cualquier dispositivo.
Como ya se ha mencionado, el equipo empezó con la parte del backend, para ello creó tanto el fichero app como el config, ambos ficheros en lenguaje javascript. Con ello se configuró todo lo relevante a la conexión de la base de datos y la elección del puerto que se iba a utilizar para crear el servidor.
A continuación, se utilizó el gestor de paquetes npm para instalar los paquetes mencionados anteriormente, lo cual generó el fichero package-lock.json y la carpeta node_modules. El fichero tiene como finalidad almacenar un historial de los paquetes que se han instalado y que permanecen en la carpeta node_modules mientras que dicho directorio contiene todo el código necesario para el correcto funcionamiento de los paquetes instalados. Gracias a package-lock.json y dado que node_modules puede llegar a ocupar mucho espacio, el equipo decidió subir solo el package-lock.json al repositorio para que, en el caso de que se quieran instalar los paquetes

necesario para el funcionamiento del prototipo, el usuario sólo debería utilizar el comando npm install en el mismo directorio en el cual se almacena el fichero mencionado anteriormente.
Una vez terminado de instalar y subir estos cambios del repositorio local al remoto, el equipo decidió centrarse en el Data Access Object (DAO). Se creó la carpeta necesaria para almacenar los DAOs que el equipo necesitaría para desarrollar la aplicación y que, en este momento solo contiene el fichero DAOUsuario.js. Este fichero almacena una clase que tiene como única función obtener una lista de usuarios que coinciden con un patrón establecido por usuario el cual se pasa por parámetro. Se probó esta funcionalidad utilizando datos almacenados en la base de datos e imprimiendolos por consola y una vez se consideró como correcto se decidió subir estas nuevas implementaciones al repositorio.
Ya terminado con la carpeta DAOs y su único fichero, el equipo se centró en ampliar el fichero javascript principal: app.js. Se declararon las rutas que el servidor iba a seguir para poder acceder a los ficheros HTML, EJS y CSS. Se declararon el recibimiento de peticiones que el servidor iba a recibir en este prototipo y se declaró la función que pondría en marcha dicho servidor. Una vez comprobado que el servidor se encontraba en marcha y no se indicaba ningún tipo de error, se decidió subir estos cambios al repositorio.
Por último, se trabajó con el apartado frontend. En esta historia de usuario no fue necesaria la implementación de ningún fichero HTML pero sí de CSS y EJS, los cuales se crearon uno y dos respectivamente. El equipo se apoyó en el framework Bootstrap 5 que permite agilizar el proceso de adornar la página web. Se creó el fichero CSS para ultimar los detalles según las necesidades del equipo. Cuando el equipo decidió que esta parte del prototipo se encontraba acabada se realizó un commit al repositorio remoto con los cambios e implementaciones realizados.
Dado que se trata de un prototipo y la implementación se basa en la búsqueda de cuidadores a partir de una palabra clave, el equipo acordó con el Product Owner que se mostrasen todos los datos relacionados con los usuarios que coincidían con la petición del usuario. Más adelante se adaptará esta historia de usuario para que solo puedan verse los campos Nombre y Dirección entre otros no añadidos en esta historia y solo se verán todos los datos y la solicitud enviada al usuario en cuestión es aceptada.
No se implementó ningún fichero de pruebas ya que el equipo no lo consideraba necesario, es por ello que el repositorio no incluye ninguna carpeta con dicho nombre.

4.4. MANUAL DE USUARIO

Para que el usuario conozca la historia de usuario implementada así como su instalación y como se ha de usar se ha desarrollado este manual de usuario para explicar de una manera simple los pasos que ha de seguir.

El usuario ha de tener instalado el entorno de ejecución Node.js para poder ejecutar el prototipo. Gracias a este entorno y una terminal se puede ejecutar sin necesidad de utilizar ningún IDE.

Si no se tiene este entorno instalado se proporciona un enlace a la página web oficial de Node.js tal y como se muestra en la imagen 4.4.1. https://nodejs.org/en/.

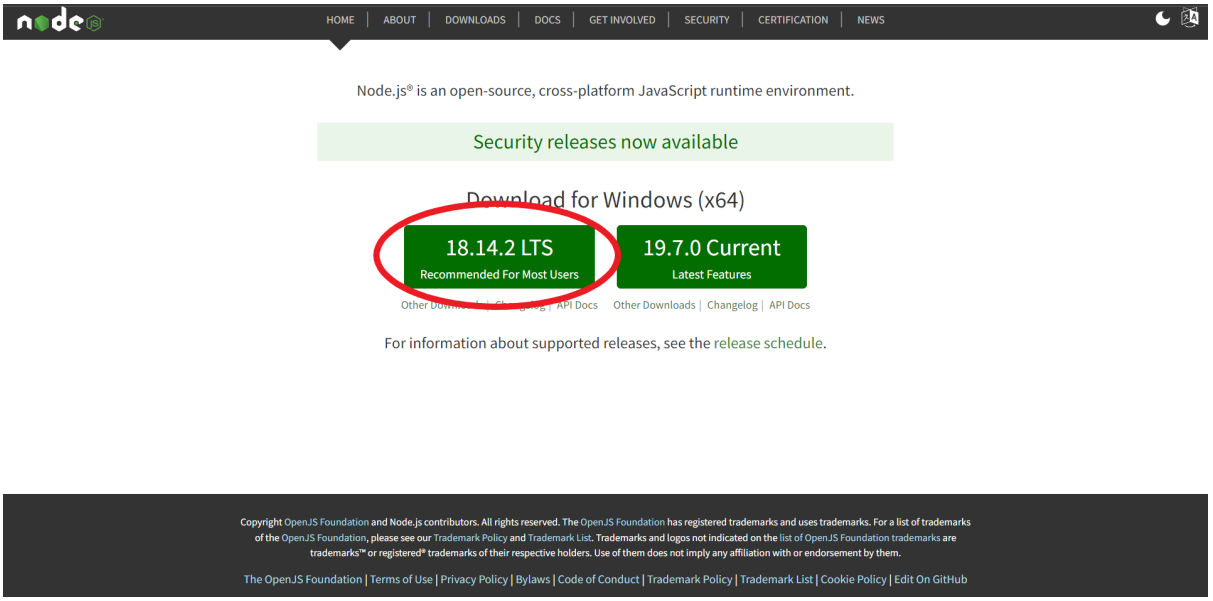

Imagen 4.4.1. página principal Node

Dado que los paquetes utilizados ocupan mucho espacio, no se han proporcionado con el código. Para solucionar esto se accede a la carpeta Src donde se dispone de todo el código del prototipo y con una terminal activa en dicho directorio se ejecuta el comando npm install para descargar los paquetes de manera automática gracias al fichero package-lock.json y package.json proporcionados.

Una vez la instalación esté terminada, es necesario tener abiertos los puertos de MySQL y Apache que podemos encontrar en la imagen 4.4.3., se puede utilizar la aplicación XAMPP para esto. Si no se dispone de esta aplicación se puede descargar en el enlace que se muestra en la imagen 4.4.2. https://sourceforge.net/projects/xampp/

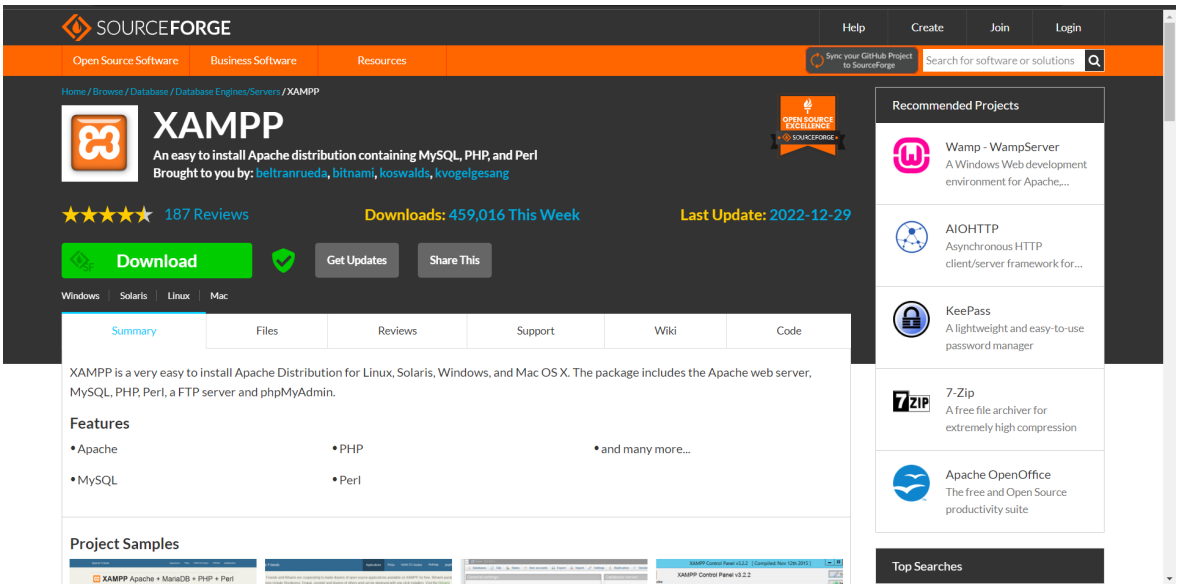

Imagen 4.4.2. página principal Xampp

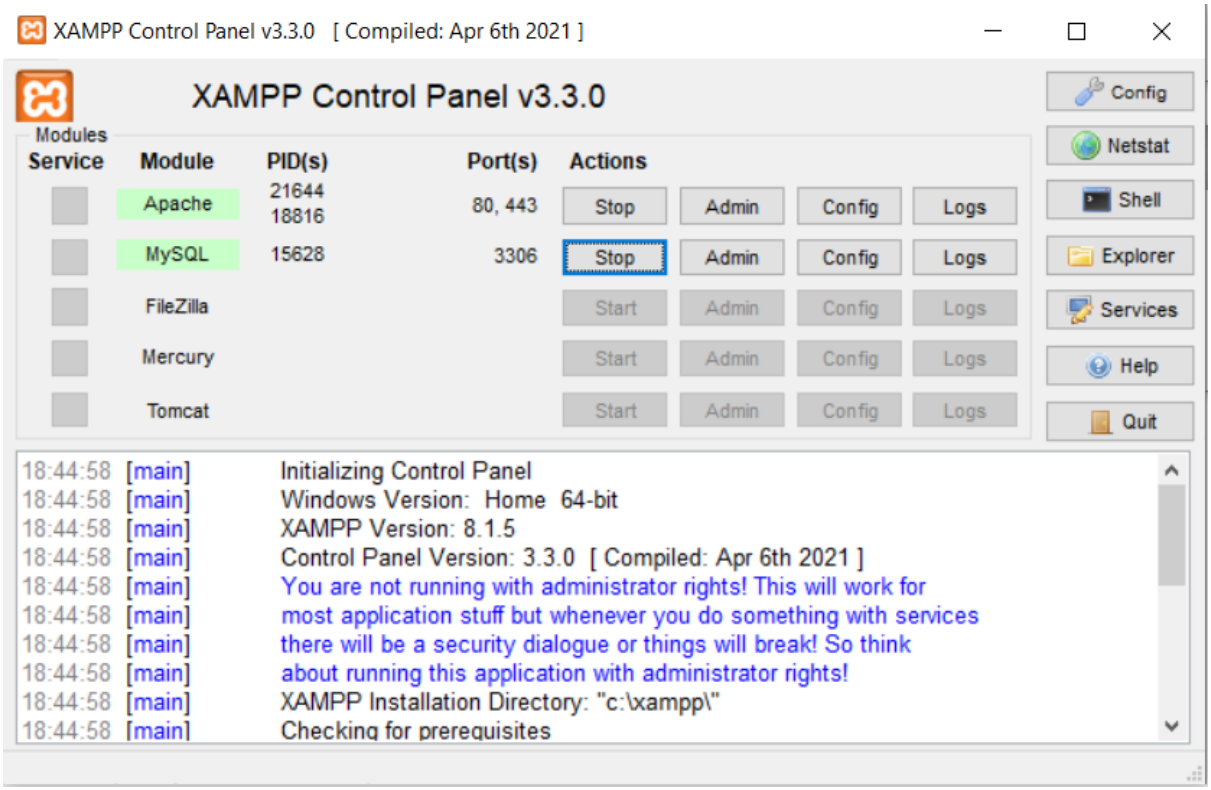

Imagen 4.4.3. Menu de activacion de opciones de Xampp

Ya descargado, se accede a la página web PHPMyAdmin a través del siguiente enlace: http://localhost/phpmyadmin/index.php?route=/ creando en esta página una nueva base de datos llamada SummerPet e importando el script que viene incorporado en la carpeta Base de Datos dentro del proyecto.

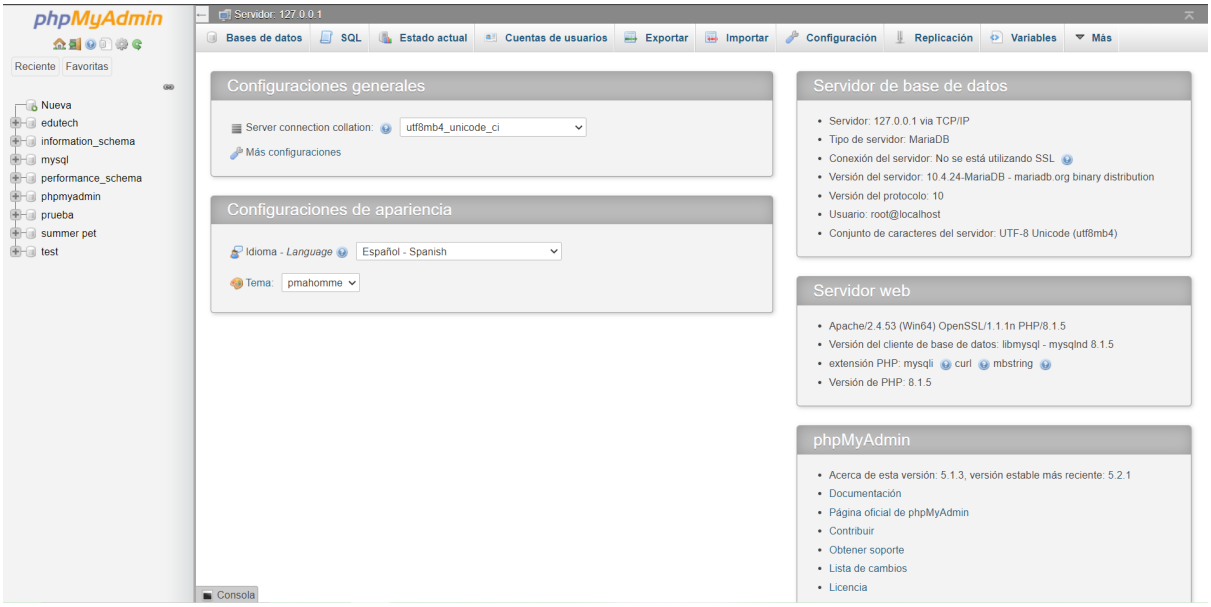

Imagen 4.4.4. Visión de la página principal phpMyadmin

Ya terminada la instalación se accede mediante una terminal a la carpeta Src del proyecto donde el usuario ha de ejecutar el comando nodemon app.js o node app.js. Si se ha implementado correctamente se deberá de ver en la misma terminal un mensaje de confirmación.

Se accede mediante un entorno web (Google Chrome, Safari, Mozilla, …) a la dirección: http://localhost:3000/

Una vez allí, el usuario podrá observar que dispone en la esquina superior derecha una barra de búsqueda donde, si introduce una palabra o carácter se proporcionarán los cuidadores resultantes de la búsqueda a partir de la palabra clave introducida. Si no se ha encontrado ninguno, se enviará un mensaje indicando que no existe ningún usuario que concuerde con los campos introducidos.

5. TEAM BUILDING

A continuación se explica todo lo referente a la dinámica de Team Building realizada por nuestro equipo. Primeramente, se explica el objetivo que nos propusimos conseguir con la dinámica. Después se explica en qué consistió la actividad y por qué podría ayudar a conseguir el objetivo, seguido del desarrollo de la misma en el que se cuenta en detalle cómo se llevó a cabo. Por último, se realizará un análisis sobre la dinámica en el que se observa si se ha conseguido el objetivo.

5.1. OBJETIVO

Para decidir qué objetivo queríamos ponerle a la dinámica de Team Building el equipo de SummerPet realizó una reunión el día de laboratorio. Para realizar la reunión no se llevó a cabo ninguna actividad, nos sentamos juntos en clase y se compartieron diferentes ideas. Debido a los malos resultados obtenidos en la práctica del Product Backlog, quisimos que la dinámica estuviese dirigida a solucionar problemas que hubiesen colaborado a obtener esos malos resultados. Tras deliberar nos dimos cuenta de que uno de los principales problemas había sido que nuestras reuniones eran poco efectivas. Por ejemplo, para realizar el Product Backlog estuvimos reunidos más de 3 horas y no conseguimos un buen resultado. Por ello, nos centramos en pensar qué aspectos de las reuniones fallaban. Llegamos a la conclusión de que los dos principales problemas eran la lentitud a la hora de llegar a un consenso con las decisiones y la falta de claridad a la hora de expresar ideas. Estos problemas se traducen en reuniones excesivamente largas, en las que el equipo tarda demasiado en tomar decisiones y no todos los miembros entienden bien lo que otro compañero está proponiendo con sus ideas. Esto provoca errores de coordinación y de ejecución cuya corrección consume más tiempo que podría ser invertido en otras actividades más útiles, como por ejemplo crear un mejor Product Backlog en el caso de la práctica 4. Por todo esto, decidimos que esta dinámica debía entrenar nuestra capacidad de expresar claramente nuestras ideas y tomar decisiones más rápidas en grupo para poder realizar reuniones más efectivas en el futuro.

5.2. DINÁMICA (Cruce de debates)

La actividad consiste en una serie de debates que los miembros del equipo tendrán entre sí. El equipo se divide en dos grupos ,que debatirán entre sí, y un miembro del equipo queda sin equipo y será el juez de ese debate. Los grupos y el juez deben rotar y cambiar en cada ronda. Se preparan una serie de preguntas en las que hay dos posturas bien diferenciadas. Por ejemplo, ¿Qué fue primero, la gallina o el huevo? El juez de la ronda le asigna a cada grupo la postura que van a defender. Tras esto, los grupos disponen de 2 minutos para pensar juntos argumentos con los que defender su postura. Pasado ese tiempo, cada grupo dispone de 1 minuto para exponer sus argumentos. Tras esto el juez decide quién gana esa ronda y se rota, cambiando los integrantes de los dos grupos y habiendo un nuevo juez. Las rotaciones deben asegurar que todos los miembros coincidan con todos los compañeros, para que todo el mundo aprenda a colaborar entre sí. Respecto a las preguntas, es importante que no traten temas sensibles para evitar discusiones reales durante la actividad. Esta dinámica es capaz de entrenar los objetivos que nos propusimos: expresar las ideas con claridad y llegar a un consenso de manera rápida. Por un lado, la habilidad de expresar ideas con claridad se entrena tanto en la reunión con tu grupo, cuando cada uno debe dar ideas de argumentos lo más rápidamente y claramente posible para aprovechar los 2 minutos de los que dispone el grupo; y también a la hora de exponer estos argumentos ya que si no se expresan de manera clara no serán capaces de convencer al juez. Y por otro lado, la habilidad de ponerse de acuerdo rápidamente se entrena en la reunión en grupo, que es la función principal de esa parte de la dinámica. Al forzar a cada grupo a decidir sus argumentos en tan sólo 2 minutos se promueve llegar a consenso rápido sobre qué argumentos son mejores y cuáles peores y qué ideas deben exponerse primero ya que sólo se dispondrá de 1 minuto para defender la postura. En resumen, esta dinámica puede servir para entrenar la capacidad de los equipos para ponerse de acuerdo y decidir qué ideas perseguir, así como ayudar a los individuos a ser más claros y directos al expresar sus ideas.

5.3 DESARROLLO

En el desarrollo de la dinámica pudieron estar presentes todos los integrantes excepto Jesús González, Elena Caridad y Álex Bonilla. Jesús y Elena tuvieron un compromiso deportivo y Álex sufrió un contratiempo y no pudo atender. Sin embargo, no había otro día en la semana que fuésemos a coincidir más de 7 compañeros y preferimos hacer la dinámica presencial que hacerla online.

En el siguiente enlace puede verse el desarrollo de la dinámica: https://youtu.be/hQ1tUUiW3SI

Para desarrollar la actividad utilizamos una de las salas de la biblioteca, y nos sentamos alrededor de una mesa de tal forma que en un lado había 3 compañeros y en el otro lado otros 3, los dos grupos que debaten, y la mesa la presidía el juez de la ronda. Tras cada ronda se cambiaba de juez y se mezclaban los grupos rotando hacia la derecha, lo cual hacía que en cada pregunta el juez y los equipos fuesen distintos que es lo que se pretende. Lo primero que hicimos fue pensar 7 preguntas sobre las que debatir, una para que cada miembro hiciera de juez. Terminado esto pudimos comenzar la actividad. En las dos primeras rondas hubo el problema de que las reuniones de grupo eran poco fructíferas porque sólo daban ideas los miembros del grupo con personalidad más extrovertida, compañeros como Luis, mientras que el resto no aportaban muchas ideas lo cual era un problema porque no se entrenaba la capacidad de ponerse acuerdo en grupo. Además, después en

el minuto de debate también eran los mismos compañeros más extrovertidos los que hablaban. Como esto conduciría a que la dinámica fuese un fracaso se paró un momento la dinámica y se estipuló que estos compañeros más extrovertidos no podían ser los que expusiesen sus ideas al juez puesto que ellos ya eran capaces de expresarse con claridad. Además, al hacerse conscientes de la situación los propios integrantes comenzaron a animar a los miembros más introvertidos a compartir sus ideas. A esto se sumó que algunos de los siguientes temas de debate ayudaron a animar mucho la dinámica, por ejemplo la pregunta ¿la tortilla de patata con o sin cebolla? Todo esto produjo que en las rondas siguientes la actividad empezase a cumplir la función con la que había sido pensada. En las reuniones de grupo había múltiples ideas y se llegaba a un consenso antes de finalizar los dos minutos y a la hora de expresar en alto sus ideas se le encargó la tarea a compañeros que no dominaban la habilidad de expresarse. También se le encargaba a compañeros cuyo problema no era la introversión si no que precisamente hablaban demasiado y les ocurría que agotaban el minuto de exposición sin haber terminado sus argumentos, lo cual provocaba que su equipo perdiese esa ronda. En general, a partir de la tercera ronda todos los turnos fueron parecidos en cuanto a equipos se refiere, siempre se llegaba a un consenso antes de los dos minutos y en general los debates solían estar bastante empatados en cuanto a calidad de los argumentos. En lo que se notó mejoría fue en la capacidad para expresarse de algunos compañeros como Iván o Miguel Marcos que consiguieron sentirse mucho más cómodos y argumentar de manera más convincente a medida que avanzaba la dinámica. Para esto también fue de ayuda que compañeros hábiles en este aspecto como Luis o Marcos ayudarán a sus compañeros y sirvieran de ejemplo. Cabe destacar que no sólo exponían los dos compañeros comentados anteriormente ya que todos debíamos aprovechar la dinámica para mejorar. Cada grupo podía decidir quién hablaría durante el minuto de exposición y en general se optó por que un compañero hablase 30 segundos y otro los otros 30, para aumentar el número de miembros que entrenaban sus habilidades orales.

5.4. RESULTADOS

Consideramos que la actividad de Team Building sí ha ayudado a mejorar la habilidad de los integrantes del equipo a comunicarse de manera más clara y efectiva pero no ha cumplido el objetivo de hacernos ponernos de acuerdo como grupo más rápidamente. Lo que nos hace afirmar que los miembros del equipo han desarrollado, son las diferentes mejoras observadas durante el desarrollo de la actividad. El ejemplo más claro fueron Iván y Miguel Marcos que pasaron de mostrarse reacios a ser los que exponían en las primeras rondas a argumentar con relativa confianza en las últimas. A parte de estos ejemplos particulares, el resto del grupo en general de lo que pecaba era de pasarse del minuto y seguir hablando cuando ya se había agotado su turno. Esto tampoco es deseable pues no demuestra la habilidad de expresarse clara y concisamente. Pero como al acabarse el minuto no se permitía argumentar más esto forzó a que todos nos tuviésemos que forzar a ser más rápidos y escuetos pero seguir siendo convincentes y expresar sus argumentos. Para el final de la actividad ningún compañero llegaba al final del minuto sin haber expresado todas sus ideas lo cual es un excelente resultado y muestra mejoría. Sin embargo, consideramos que la capacidad para llegar a consenso como grupo no se ha entrenado de la manera que se esperaba. Los dos principales problemas que han provocado que no se alcance el objetivo han sido que, por un lado, la situación no era una de equipo real. Normalmente tenemos que ponernos de acuerdo entre 10 personas y en la dinámica era sólo entre tres los cuál era mucho más fácil. Por otro lado, no era complicado ponerse de acuerdo ya que en general surgían ideas de argumentos parecidas, muy diferente de lo

que sucede en un proyecto cuando a veces hay que elegir entre ideas totalmente diferentes. Por todo esto, consideramos que la dinámica ha servido para expresar mejor nuestras ideas pero no para mejorar nuestra capacidad de ponernos de acuerdo rápido.