

Universidad ORT Uruguay
Facultad de Ingeniería

Sistema de paseo de mascotas PetVacay

Entregado como requisito para la obtención del título de
Ingeniero en Sistemas

Ximena Moure – 175660
Ismael Puricelli – 187086
Juan Rodríguez – 171983
Tutor: Ing. Luis Barrague

2017

Índice

1. Introducción	1
1.1 Objetivos	3
1.1.1 Objetivos Académicos.....	3
1.1.2 Objetivos del producto	3
2. Alcance general de nuestra aplicación.....	4
2.1 Descripción general del trabajo	4
2.2 Análisis del problema y la situación actual.....	5
2.3 Análisis de grupo de interesados.....	6
3. Metodología de trabajo	7
3.1 Ciclo de vida utilizado.....	7
3.2 Metodología ágil	8
3.3 Scrum.....	10
3.4 Scrum adaptado	12
4. Ingeniería de requerimientos	13
4.1 Técnicas de relevamiento.....	13
4.2 Validación de requerimientos	13
4.3 Especificación de requerimientos	14
4.3.1 Utilización de User Stories	14
4.3.2 Priorización de requerimientos	14
4.3.3 Requerimientos.....	15

4.3.3.1 Requerimientos funcionales	15
4.3.3.2 Requerimientos no-funcionales.....	20
4.3.4 Especificación de historias de usuario	21
5. Bocetos y diseños de la interfaz de usuario de la aplicación.....	37
5.1 Log In	37
5.2 Menú principal	38
5.3 Registro de usuario	39
5.4 Registro de cuidadores y paseadores	40
5.5 Cuenta de usuario.....	41
5.6 Perfil del usuario.....	42
5.7 Menú para agregar mascota.....	43
5.8 Menú de búsqueda y reserva de cuidadores/paseadores.....	44
5.9 Mapa de cuidadores/paseadores cercanos	45
5.10 Mapa de paseador.....	46
5.11 Calificación de cuidadores/paseadores	47
5.12 Feedback con informes	48
5.13 Menú de pago.....	49
5.14 Información de la aplicación	50
6. Gestión de Proyecto	51
6.1 Métricas utilizadas	51
6.1.1 Story Points	51

6.1.1.1 Traducción de Story Points a días	51
6.1.2 Velocidad del equipo.....	52
6.2 Planificación	53
6.2.1 Etapas del proyecto	53
6.2.2 Cronograma	54
6.2.3 Detalle de los entregables.....	56
6.2.4 Planificación de los Sprints	56
6.2.4.1 Cronograma de Sprints.....	57
6.2.5 <i>Sprints Reviews</i>	59
6.2.6 <i>Burnout Chart</i>	60
6.3 Comunicación.....	63
6.4 Herramienta de gestión Trello	63
7. Arquitectura y diseño	64
7.1 Diagramas UML	64
7.1.1 Diagrama de clases del front-end	64
7.1.1.1 com.example.ximenamoure.petvacay.....	64
7.1.1.2 com.example.ximenamoure.petvacay.Adapters	65
7.1.1.3 com.example.ximenamoure.petvacay.BackendConfig.....	66
7.1.1.5 com.example.ximenamoure.petvacay.Models	67
7.1.1.6 com.example.ximenamoure.petvacay.Notifications	68
7.1.2 Diagrama de componentes	69

7.1.2.1 Mecanismos de comunicación.....	70
7.1.3 Diagrama de clases	70
7.1.3.1 PetVacay.Adapters.....	70
7.1.3.2 PetVacay.API	71
7.1.3.3 PetVacay.Data	72
7.1.3.4 PetVacay.Data.DataAccess.....	73
7.1.3.5 PetVacay.Repository	74
7.1.3.6 PetVacay.DependencyResolver	75
7.1.3.7 PetVacay.DTO	76
7.1.3.8 PetVacay.Enumerators.....	77
7.1.3.9 PetVacay.Exceptions.....	77
7.1.3.10 PetVacay.Helpers.....	78
7.1.3.11 PetVacay.JsonMessage	79
7.1.3.12 PetVacay.Services	80
7.1.3.13 PetVacayValidators	81
7.1.4 Diagrama de despliegue	82
7.1.5 Modelo de tablas de base de datos.....	83
2.2 Justificaciones de diseño y arquitectura	84
7.2.1 Arquitectura <i>en capas</i> - <i>Patrón Layered</i>	84
7.2.2 Patrón Unit of Work.....	84
7.2.3 Inyección de dependencias.....	85

7.2.4 Patrón Adapter	85
7.2.5 Patrón Singleton	85
7.2.6 Uso de Data Transfer Objects (DTOs)	86
7.2.7 Arquitectura Restful	86
7.2.8 Patrón Strategy	87
7.2.9 Análisis de principios SOLID	87
7.2.9.1 Single Responsibility	87
7.2.9.2 Open/Closed	87
7.2.9.3 Liskov Sustitution.....	87
7.2.9.4 Interface Segregation	87
7.2.9.5 Dependency Inversion	88
8. Gestión de Riesgos	89
8.1 Identificación de riesgos	89
8.2 Planilla de riegos	90
8.3 Seguimiento	91
8.3.1 Control de riesgos durante el Sprint 1	92
8.3.2 Control de riesgos durante los Sprints 2, Sprint 3 y Sprint 4	92
9. Gestión de configuración	93
9.1 Gestión de código fuente.....	93
9.1.1 Control de versionado	93
9.1.2 Evidencia de versionado	93

9.1.2.1 Repositorio “Ingeniería de software en la práctica”	94
9.1.2.2 Repositorio “Ingeniería de software en la practica - BACKEND”	94
9.2 Gestión de documentación	95
10. Gestión de la calidad	95
10.1 Objetivos de calidad del producto.....	95
10.2 Objetivos de calidad del proceso	95
10.3 Aseguramiento de la calidad.....	96
10.3.1 Calidad del proceso	96
10.3.1.1 Spring retrospective meeting	96
10.3.2 Uso de estándares de codificación.....	96
10.3.3 Uso de estándares de documentación	97
10.3.4 Actividades de calidad	97
10.3.5 Principales actividades.....	99
10.3.6 Pruebas unitarias	101
10.3.7 Pruebas con usuarios	102
11. Conclusiones.....	102
12.Bibliografía.....	103
12. Anexos	104
12.1 Uso de la herramienta Trello.....	104
12.1.1 Evidencia de uso de la herramienta Trello.....	105
12.2 Manual de instalación	105

1. Introducción

Con el paso de los años la tecnología ha ido evolucionando considerablemente desde simples programas para resolver cálculos hasta programas de reconocimiento facial, entre otros.

Es innegable que hoy en día la tecnología se encuentra en todos los ámbitos de la vida humana, simplificando tareas que serían muy difíciles de ser realizadas por una persona, sirviendo medios de entretenimiento y hasta en algunos casos, ser prácticamente imprescindible.

Dentro de estas tecnologías, una de las más novedosas y que ha causado un cambio en como vemos la misma es el avenimiento del mundo Mobile y las aplicaciones móviles.

Estas aplicaciones van más allá de lo que una común puede ser, estas buscan fusionar la prestación de servicios con una experiencia de usuario que sea capaz de aprovechar las diversas ventajas que un dispositivo móvil ofrece, a diferencia de una computadora.

Por ejemplo, una aplicación desktop podría implementar funcionalidad de etiquetado y modificación de fotos, pero una aplicación móvil, aparte de esto, podría hacer uso del GPS para obtener la localización de la foto tomada y etiquetarla, acceso a la cámara del teléfono para aplicar distintos filtros o efectos especiales a fotos tomadas, etc.

Por otro lado, diversos factores que no aplican a software de escritorio se vuelven de importante consideración al momento de desarrollar una aplicación para un dispositivo Mobile, por ejemplo, el contexto en el que se usa la aplicación, el tipo de usuario al que está dirigida, como utiliza el celular, con cuanta frecuencia, etc.

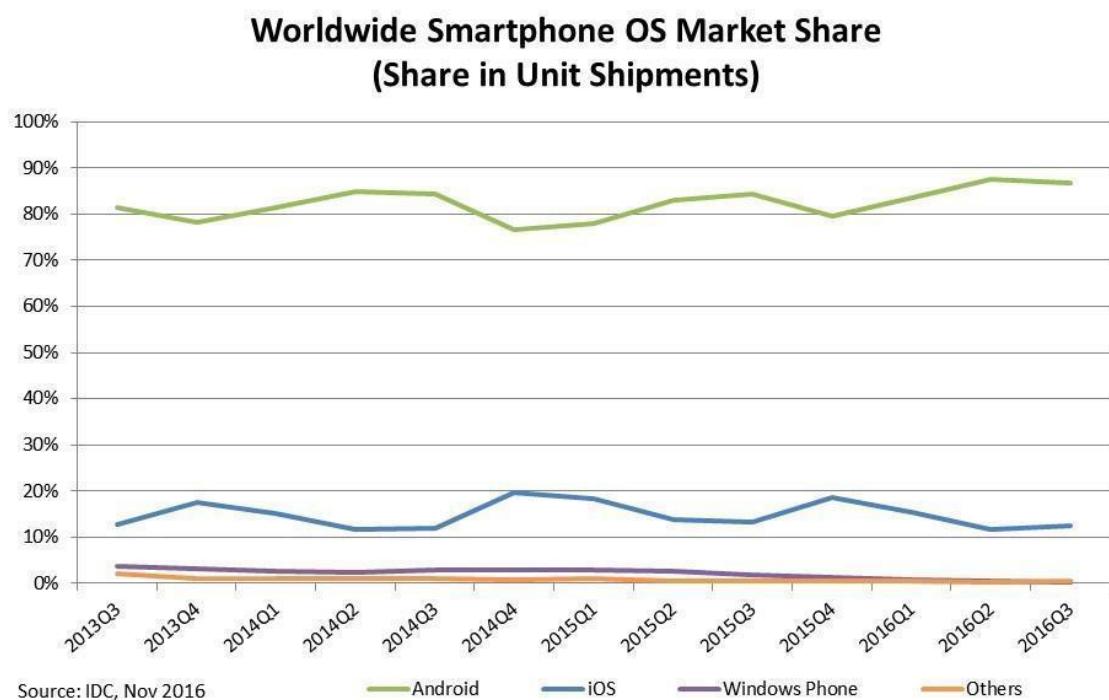
Todos estos son factores que influyen en el momento de desarrollar una aplicación y deben de ser tenidos en cuenta para el éxito de la misma.

Debido a esto, el paradigma de la usabilidad y experiencia de usuario al momento de utilizar una aplicación cambia con respecto al que ya se tenía, por eso se deben tomar decisiones que intenten innovar sobre estos últimos temas, aprovechando la capacidad y las oportunidades que tiene un dispositivo móvil que no las tiene uno fijo, como por ejemplo una computadora.

Antes de adentrarse en este mundo es menester mencionar las distintas plataformas disponibles que se encuentran para todas estas aplicaciones.

Actualmente existen tres grandes plataformas móviles: Android, iOS y Windows Mobile, siendo los dos primeros los mayores competidores del mercado.

Según estadísticas de la IDC (International Data Corporation) la plataforma Android es la más popular a nivel mundial con un 86.6% de usuarios, seguida de iOS con un 12.5% de usuarios alrededor del mundo^[1].



Como se observa, dado el gran crecimiento de la plataforma Android y el gran número de usuarios a nivel mundial, para el desarrollo de este trabajo se desarrollará una aplicación para dicha plataforma.

1.1 Objetivos

A continuación, se realiza una descripción de los objetivos que se desean lograr. Los mismos abarcan objetivos del producto y académicos.

1.1.1 Objetivos Académicos

Todos los integrantes del equipo buscan poner en práctica todo lo aprendido hasta ahora a lo largo de la carrera. También se busca que este trabajo obligatorio sirva como una manera de prepararnos para lo que va a ser la Tesis final.

Aprender sobre nuevas tecnologías. Como ninguno de los miembros del equipo tiene experiencia desarrollando en Android se busca que este trabajo sirva para ganar experiencia y para aprender sobre Android.

1.1.2 Objetivos del producto

Se busca crear un producto atractivo para el usuario final. Así como también un producto que sea fácil de usar.

Creemos que las mascotas son una parte importante en la vida de las personas por lo que pensamos que la aplicación puede ser de mucha ayuda para personas que necesitan que alguien de confianza cuide de ellas. A veces es demasiado complicado conseguir a alguien dispuesto a cuidar a tu mascota, por eso creemos que la aplicación va a ser de mucha ayuda.

2. Alcance general de nuestra aplicación

2.1 Descripción general del trabajo

Dado lo anterior, hemos decidido realizar una aplicación Mobile que intente innovar sobre las ya existentes, que brinde una buena experiencia de usuario y que brinde servicios y provea funcionalidades que no existen en el mercado.

La aplicación a desarrollar consiste en una que permite a los usuarios de la misma contratar paseadores o cuidadores de mascotas los cuales se encuentran registrados en la misma.

Se podrá buscar paseadores o cuidadores ya sea aplicando filtros de ubicación o por tipo de mascota o vía ubicación GPS, eligiendo el tiempo que se necesita (horas, días, semanas, etc.) e ingresar ciertos requisitos como por ejemplo si la mascota duerme dentro de la casa o no.

A su vez, los usuarios de la misma que hayan contratado el servicio de algún paseador o cuidador podrán posteriormente evaluar y calificarlos, así como también realizar comentarios acerca del mismo.

La idea de la aplicación es ayudar a las personas a poder encontrar con facilidad cuidadores o paseadores para su mascota en caso que ellos no puedan hacerlo ya sea por motivos de trabajo, viaje, enfermedad, etc.

La aplicación se encargará de mantener al usuario informado mientras que el servicio es provisto mediante reportes diarios por los paseadores o cuidadores, ubicación GPS en tiempo real para los paseadores y forma de contacto personal con el proveedor del servicio.

2.2 Análisis del problema y la situación actual

En la actualidad una gran parte de la población tiene al menos una mascota o animal de compañía en sus hogares.

En Uruguay, por ejemplo, se estima que un 69% de ciudadanos opta por tener un perro en sus hogares, ya sea por compañía, seguridad, etc. [2]. Porcentaje que aumenta si tenemos en cuenta otras mascotas.

Tener una mascota ofrece muchas ventajas y muchos momentos de felicidad, pero también es cierto que en determinadas situaciones se puede convertir en una complicación.

Estas complicaciones se presentan en diversas situaciones, si una familia trabaja y tiene mascota, es complicado atender o cuidar de la misma estando ausentes, por lo que generalmente se requiere a terceros ya sean vecinos o familiares para hacerse cargo de la misma.

Por otro lado, en caso de viajes al exterior, generalmente termina ocurriendo lo mismo, la mascota se deja en cuidado de terceros. Lo mismo ocurre en casos de enfermedades, etc.

Otro problema encontrado es la dificultad o trabajo a la hora de encontrar alguien que brinde estos servicios, por lo general como se dijo anteriormente, se trata de personas conocidas, pero en caso de que estos no se encuentren disponibles es necesario buscar en diversos lugares, ya sea el diario, internet, etc., y, aun así, es difícil encontrar una persona confiable que sea capaz de realizar el trabajo.

Es por esto que surge la necesidad de ofrecer una solución a este problema, brindando una plataforma en la cual personas interesadas puedan postularse para ser cuidadores o paseadores de mascotas, teniendo la posibilidad de brindar sus servicios y hacerse conocidos, mientras que por otro lado, para los usuarios de la misma, la posibilidad de afrontar todas estos problemas anteriormente mencionados, permitiéndoles también encontrar con facilidad contactos que de otra forma sería complicado buscarlos por internet o por distintos lugares.

2.3 Análisis de grupo de interesados

Con el objetivo de guiar las funcionalidades de la aplicación se analizaron los principales grupos de interesados, así como también sus necesidades y cómo impacta el desarrollo de la aplicación en los mismos.

Se encontraron dos grupos de principales interesados los cuales se listan a continuación:

Cuidadores y paseadores de mascotas

Estas personas están interesadas en formar parte de una plataforma en donde puedan ofrecer abiertamente sus servicios y hacerse conocidos a través de la misma, garantizándoles de esta manera una mayor oportunidad de trabajo.

Para este grupo de personas, tanto el interés por el proyecto como su poder de influencia sobre el mismo es alto.

Gente con mascotas en sus hogares

Los dueños de las mascotas requieren de terceros que brinden servicios para cuidar de las mismas, y a su vez necesitan un medio eficaz por el cual sea sencillo encontrar dichas personas, ofreciendo garantía y confianza.

Para este grupo de personas, tanto el interés por el proyecto como su poder de influencia sobre el mismo es alto.



Como ambos grupos de interés se ubican en el cuadrante I de la matriz de interesados [3], es de vital importancia relevar sus requerimientos y mantenerlos satisfechos.

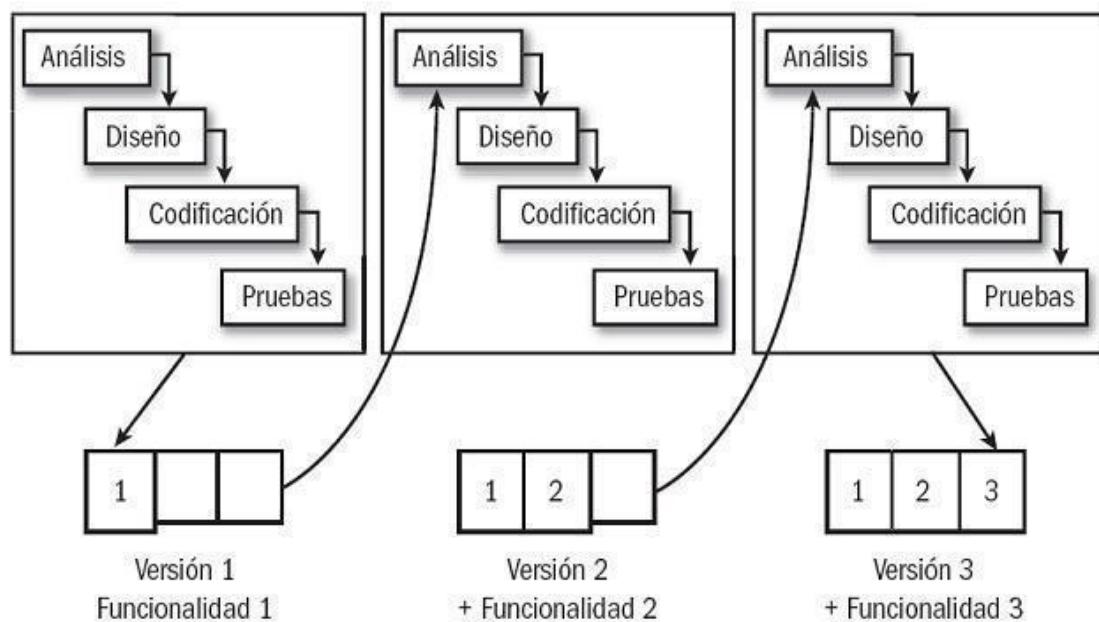
3. Metodología de trabajo

3.1 Ciclo de vida utilizado

Elegimos el ciclo de vida iterativo e incremental, ya que estos se caracterizan por ir repitiendo una o más actividades dentro de las etapas del proyecto e ir integrando los avances para generar un producto con más grande.

Al final de cada iteración se consigue una versión más estable del software, de más calidad, y con nuevas funcionalidades respecto a versiones anteriores del mismo.

Dada la dinámica y contexto del trabajo a realizar, si bien los requerimientos son medianamente estables hasta la entrega, es conveniente ir iterando sobre todas las etapas del desarrollo e ir aplicando *refactoring* en el código de manera de no solo ir mejorando lo hecho, sino de ir agregándole nueva funcionalidad a la aplicación al final de cada iteración [4].



Como se observa en la imagen superior, se va iterando en todas las fases del desarrollo, agregándole más funcionalidad al producto. [4]

En base a esto, hemos optado por seguir una metodología que aproveche de este ciclo de vida al máximo, por ende, decidimos utilizar una metodología ágil como se detalla a continuación en el siguiente punto.

3.2 Metodología ágil

Dada nuestra forma de trabajo en proyectos anteriores y la naturaleza del actual, creemos conveniente optar por guiar nuestro desarrollo en base a una metodología ágil.

El enfoque de desarrollo ágil ofrece diversas ventajas con respecto al enfoque tradicional guiado por los planes, permitiendo planes adaptativos y el desarrollo evolutivo del software.

A continuación, se muestra una imagen que compara las metodologías ágiles con las tradicionales^[5], permitiendo observar lo mencionado anteriormente.



De esta manera detallaremos a continuación las características de nuestro proyecto que se ajustan con lo propuesto por las dichas metodologías.

Ciclo de vida elegido

Las metodologías ágiles se adaptan perfectamente al ciclo de vida elegido, ya que estas, a diferencia de las guiadas por los planes, plantean enfocarse más en el cliente y en los cambios, por lo que permiten ir constantemente iterando sobre las distintas etapas de desarrollo agregándole nuevo valor al producto al final de cada iteración.

Tecnología nueva

Si bien el *back-end* se desarrollará utilizando tecnología conocida por nosotros, no es así con el *front-end* ya que no tenemos experiencia en desarrollo de aplicaciones para la plataforma Android.

Debido a esto surge la necesidad de utilizar una metodología que permita mitigar cualquier tipo de inconveniente que pueda surgir en base a la dificultad mencionada con anterioridad.

Adaptabilidad a los cambios

A lo largo del proyecto es muy probable que los requerimientos cambien, así como también pueden cambiar distintas implementaciones de servicios, refactoreo de código, principalmente sobre el *front-end*, en el cual como se mencionó anteriormente, no contamos con experiencia de desarrollo.

Trabajo en equipo

Consideramos que el trabajo en equipo es sumamente importante en el desarrollo de software, así como también la interacción y comunicación con los integrantes del mismo, permitiendo una mayor comprensión del trabajo que se está realizando y la posibilidad de incluir las ideas de los distintos integrantes y poder aplicarlas al producto.

3.3 Scrum

Posteriormente a la decisión de implementar una metodología ágil como nuestra forma de trabajo y tomando en cuenta las características de nuestro proyecto, de las diversas metodologías existentes se optó por Scrum.

Scrum es una metodología de trabajo ágil orientada al trabajo en equipo durante una serie de iteraciones llamadas *Sprints*.

Al inicio de la misma se cuenta con un *Product Backlog* el cual representa todas las historias de usuario a realizarse durante todo el proceso.

Dicha metodología define tres roles básicos los cuales son:

- *Product Owner*: Representa la visión del cliente y es quien prioriza las historias de usuario del *Product Backlog* al comienzo de cada *Sprint*.
- *Scrum Master*: Es quien lidera el grupo, guiándolo y encargándose de que se cumpla con la metodología y no surjan inconvenientes.
- *Development Team*: Son los encargados de desarrollar el producto.

La idea detrás de esta metodología es ir realizando un producto de forma incremental a través de pequeñas iteraciones de entre dos a cuatro semanas.

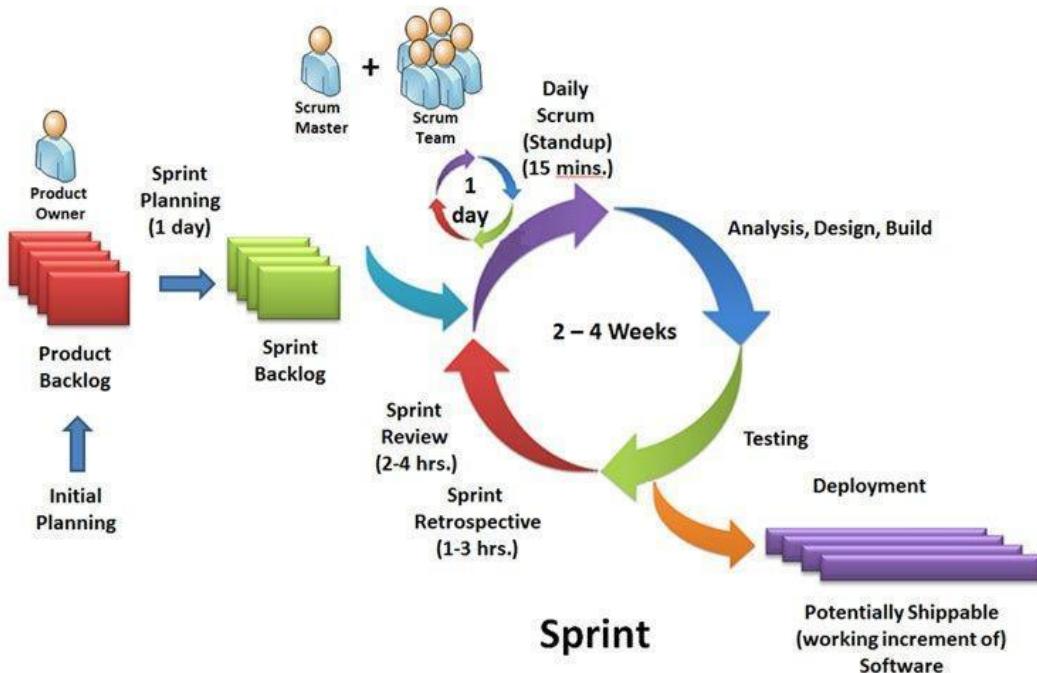
Durante estas iteraciones el equipo de desarrollo se encarga de construir un primer entregable de calidad para el usuario y a su vez el equipo entero se reúne diariamente para obtener *feedback*.

Al final del Sprint se realiza la *Sprint Retrospective* que consiste en evaluar el Sprint realizado en base a tres interrogantes:

- ¿Que se hizo mal durante el mismo?
- ¿Que se hizo bien durante el mismo?
- ¿Qué inconvenientes se encontraron durante el mismo?

En base a esto se sacan conclusiones y se planifica el próximo Sprint.

La siguiente imagen ejemplifica gráficamente la metodología Scrum:



Copyright © 2011, William B. Heys

Las características de esta metodología, como quedó dicho implícitamente en el párrafo anterior, se ajustan a las de nuestro proyecto y nuestra forma de trabajo, aunque con ciertas modificaciones sobre la original. A continuación, se listan algunas de las principales las cuales nos ayudaron a elegir esta metodología.

Iteraciones cortas

Con un ciclo de vida de desarrollo iterativo e incremental, contar con iteraciones cortas de desarrollo se adhiere a este principio perfectamente, en cada iteración se desarrolla parte de la aplicación, mejorándose y agregándole nuevas funcionalidades en las iteraciones siguientes, logrando de esta manera un producto final de buena calidad.

Retroalimentación constante

Al final de cada sprint se realiza la *Sprint Retrospective* donde se puede obtener *feedback* y sacar conclusiones acerca del trabajo realizado en dicho sprint, permitiendo al equipo una mayor respuesta al cambio y una mayor flexibilidad sobre el desarrollo del producto. Esto nos permite como equipo evaluar nuestra forma de trabajo hasta ese momento y dar una respuesta temprana a problemas que puedan surgir, como por ejemplo atrasos, etc.

Equipo auto gestionado

Como en esta metodología no hay roles definidos para cada miembro del equipo de desarrollo, cada uno puede realizar parte de una funcionalidad con la cual se sienta más cómodo trabajando o posea más experiencia, esto nos brinda una mayor libertad al momento del desarrollo del producto.

[3.4 Scrum adaptado](#)

Nos parece importante que al ser un equipo pequeño (tres personas) y además teniendo en consideración que ninguno de los integrantes tiene experiencia aplicando la metodología Scrum que no haya un Scrum Master ni un Product Owner.

Es por esto que se aplicara Scrum pero sin los roles anteriormente mencionados.

4. Ingeniería de requerimientos

Al comienzo del proyecto el equipo buscó definir cuál era el producto (la aplicación móvil) que se deseaba construir, así como también cuales eran las funcionalidades de más alto nivel que el mismo debía proporcionar. Para poder a partir de esto generar requerimientos funcionales y no funcionales.

4.1 Técnicas de relevamiento

Para poder definir la aplicación a realizar y cuales iban a ser las funcionalidades de la misma, se realizó una tormenta de ideas.

En la misma cada integrante del grupo planteó ideas sobre aplicaciones que le gustaría realizar. Luego se analizó la factibilidad de la misma y se buscó responder a las siguientes preguntas:

- ¿Qué problema resuelve la aplicación?
- ¿Quiénes son los usuarios objetivos?
- ¿Es posible realizarla en el plazo establecido para este trabajo obligatorio?

Por último, se realizó una lista de alto nivel de los requerimientos de cada aplicación planteada.

Al concluir esta etapa se obtuvo una lista con las posibles ideas a desarrollar y una lista con los requerimientos iniciales.

4.2 Validación de requerimientos

Al ser un trabajo obligatorio se decidió validar los requerimientos y cual idea era la más conveniente o cual idea aportaba más para este trabajo con el profesor del curso.

De las ideas planteadas el profesor nos informó que le parecía que todas eran correctas que decidiera el grupo cual realizar.

En base a esto el grupo realiza una votación y se decide por una de las ideas.

Luego de tomada esta decisión se validan los requerimientos con el profesor del curso.

4.3 Especificación de requerimientos

En esta etapa de ingeniería de requerimientos se escribieron los requerimientos obtenidos en la etapa de relevamiento como *User Stories*.

4.3.1 Utilización de User Stories

Como se mencionó anteriormente la aplicación va a ser desarrollada utilizando SCRUM por lo que la especificación de los requerimientos decidimos realizarla utilizando *User Stories*.

Una de las ventajas de utilizar *User Stories* es que las mismas facilitan el dividir el proyecto en entregas (o *releases*).

Otra de las ventajas es que las *User Stories* son escritas desde el punto de vista del usuario en un lenguaje natural.

A cada *User Story* se le asocia una descripción de la funcionalidad, así como los criterios de aceptación para darla como finalizada.

Se utiliza el siguiente formato para las *User Stories*:

Como un [tipo de usuario]

Quiero [realizar alguna tarea]

Para [alcanzar una meta/beneficio/valor]

4.3.2 Priorización de requerimientos

Una vez definidos los requerimientos, los mismos deben ser priorizados. Se debe asignar a cada requerimiento funcional un valor de prioridad. Esto se realizó utilizando el método MoSCow:

- Alto (*Must*)
- Medio (*Should*)
- Bajo (*Could*)

4.3.3 Requerimientos

A continuación, se listan los requerimientos funcionales y no funcionales que deberá cumplir la aplicación a desarrollar, estos últimos han sido especificados teniendo en cuenta las necesidades de los distintos interesados analizados anteriormente.

Los requerimientos aquí descritos son los que se obtuvieron en una primera instancia luego de la tormenta de idea y la validación de los mismos con el profesor del curso.

En el Anexo 11.1 se pueden ver los requerimientos descritos como historias de usuario.

4.3.3.1 Requerimientos funcionales

RF1 Log In

Descripción: El sistema deberá permitir al usuario iniciar sesión con una cuenta previamente registrada para poder utilizar las funcionalidades de la aplicación.

Prioridad: Alta

RF2 Log Out

Descripción: El usuario podrá cerrar sesión en la aplicación y poder ingresar con otro usuario si este lo desea.

Prioridad: Alta

RF3 Registro de usuarios

Descripción: El usuario podrá registrarse en la aplicación para así poder acceder a las funcionalidades que ofrece el mismo

Prioridad: Alta

RF4 Registro de cuidadores y paseadores

Descripción: El usuario podrá registrarse como cuidador o paseador de mascotas en la aplicación.

Prioridad: Alta

RF5 Crear perfil de usuarios

Descripción: Los usuarios podrán crear su perfil personal brindando su información personal y de sus mascotas.

Prioridad: Alta

RF6 Crear perfil de cuidadores y de paseadores

Descripción: Los cuidadores o paseadores podrán crear su perfil personal brindando información relativa a sus servicios.

Prioridad: Alta

RF7 Búsqueda de cuidadores/paseadores aplicando distintos filtros

Descripción: El usuario de la aplicación podrá buscar paseadores o cuidadores de mascotas aplicando distintos filtros de búsqueda como el tipo de mascota, fecha para el servicio o barrio de preferencia, mostrándose posteriormente en la pantalla.

Prioridad: Alta

RF8 Calificar a paseadores/cuidadores

Descripción: Los usuarios de la aplicación podrán evaluar a los paseadores o cuidadores contratados a través de un sistema de calificación de cinco estrellas.

Prioridad: Alta

RF9 Realizar comentarios sobre paseadores/cuidadores

Descripción: Los usuarios de la aplicación podrán evaluar a los paseadores o cuidadores contratados dejando comentarios acerca de su experiencia.

Prioridad: Media

RF10 Obtener localización por GPS cuando la mascota es paseada

Descripción: Los usuarios de la aplicación podrán ver en tiempo real la ubicación por GPS de su mascota mientras es paseada por el paseador contratado.

Prioridad: Alta

RF11 Realizar informe al terminar un paseo por parte de los paseadores

Descripción: Los paseadores tendrán la opción de realizar un informe general sobre el paseo luego de haber culminado con el mismo, este informe será visible para el usuario que contrató a dicho paseador.

Prioridad: Alta

RF12 Realizar informe diariamente por parte de los cuidadores

Descripción: Los cuidadores tendrán la opción de realizar informes diarios, durante el período de contratación de un cuidador, este enviará diariamente un informe sobre el resumen de su actividad y el comportamiento de la mascota, los cuales serán visibles para el usuario que lo contrató.

Prioridad: Alta

RF13 Baja usuario

Descripción: El usuario de la aplicación podrá darse de baja cuando lo desee siempre que no tenga algún cuidador o paseador reservado.

Prioridad: Media

RF14 Modificar usuario

Descripción: El usuario de la aplicación podrá actualizar sus datos personales cuando lo desee.

Prioridad: Media

RF15 Reservar cuidador

Descripción: El usuario de la aplicación podrá reservar un cuidador de los disponibles para su mascota estableciendo el tiempo que desea contratar el servicio.

Prioridad: Alta

RF16 Reservar paseador

Descripción: El usuario de la aplicación podrá reservar un paseador de los disponibles para su mascota estableciendo el tiempo que desea contratar el servicio.

Prioridad: Alta

RF17 Pagar cuidador/paseador

Descripción: El usuario de la aplicación tendrá la opción de pagar por el servicio brindado por el cuidador/paseador contratado a través de su tarjeta de crédito.

Prioridad: Alta

RF18 Contactar cuidador/paseador

Descripción: El usuario podrá contactarse en cualquier momento con el cuidador o paseador contratado.

Prioridad: Alta

RF19 Localizar cuidadores/paseadores cercanos a la ubicación del usuario vía GPS

Descripción: El usuario podrá visualizar vía GPS todos los cuidadores y paseadores cercanos a su ubicación.

Prioridad: Alta

RF20 Preguntas frecuentes

Descripción: La aplicación contará con un apartado de “Preguntas frecuentes” donde se brindarán respuestas a las dudas más comunes con respecto al manejo de la aplicación y la funcionalidad del sistema.

Prioridad: Baja

RF21 Información sobre la aplicación

Descripción: La aplicación contará con un apartado de información el cuál contendrá datos relevantes sobre la aplicación como el nombre de la misma, la versión actual y los creadores.

Prioridad: Baja

RF22 Cancelar reserva de paseador/cuidador

Descripción: Un usuario de la aplicación que previamente haya reservado un cuidador/paseador para su mascota podrá dar de baja la reserva. La misma no tendrá costo si la reserva se da de baja con 24hs. de anticipación, en caso contrario se deberá abonar el 50% de la tarifa pactada.

Prioridad: Media

4.3.3.2 Requerimientos no-funcionales

RNF1 La aplicación debe funcionar en dispositivos Android con versión 4.4 en adelante

Descripción: La aplicación deberá funcionar correctamente en dispositivos con plataforma Android cuya versión sea 4.4 o mayor.

Prioridad: Alta

RNF2 La aplicación debe ser intuitiva y fácil de usar para el usuario

Descripción: La aplicación deberá proveer una buena experiencia de usuario y una buena usabilidad.

Prioridad: Alta

RNF3 El *back-end* de la aplicación debe ser desarrollado en ASP.NET Web API con Visual Studio y la base de datos con SQL Server

Descripción: El *back-end* de la aplicación deberá ser desarrollado en Visual Studio con ASP.NET Web API y se deberá utilizar una base de datos con SQL Server.

Prioridad: Alta

4.3.4 Especificación de historias de usuario

En esta sección se presentan las historias de usuario definidas inicialmente como parte del análisis del problema, así como algunos bocetos de interfaz de usuario asociados a dichas historias para lograr una mejor idea de las mismas.

RF1 Log In

Historia de Usuario 01: Log In	
Como:	Usuario del sistema
Quiero:	Iniciar sesión en el sistema
Para:	Acceder a las funcionalidades que ofrece el mismo
Estimación:	5 SP
Criterios de aceptación	
Escenario 1: El usuario se encuentra registrado en el sistema Dado un usuario registrado Cuando ingresa su usuario y contraseña correctamente Entonces se abre la aplicación	
Escenario 2: El usuario no se encuentra registrado en el sistema o los datos son incorrectos Dado un usuario Cuando intenta ingresar al mismo con datos que no están registrados Entonces se muestra un mensaje diciendo que los datos son incorrectos	

RF2 Log Out

Historia de Usuario 02: Log Out	
Como:	Usuario del sistema
Quiero:	Cerrar sesión en el sistema
Para:	Terminar de manera segura con lo que estuve haciendo y también en un futuro acceder con otras credenciales al mismo
Estimación:	3 SP
Criterios de aceptación	
<p>Escenario: El usuario cierra sesión en la aplicación</p> <p>Dado un usuario logueado previamente</p> <p>Cuando cierra sesión en la aplicación</p> <p>Entonces dicho usuario deja de acceder a las distintas funcionalidades de la aplicación y se lo redirige a la página de inicio de sesión de la misma</p>	

RF3 Registro de usuarios

Historia de Usuario 03: Registro de usuarios	
Como:	Usuario sin identificar en el sistema
Quiero:	Registrarme como un usuario del mismo
Para:	Acceder a las funcionalidades que ofrece el mismo
Estimación:	5 SP
Criterios de aceptación	
Escenario 1: El usuario se registra con datos válidos Dado un usuario Cuando intenta registrarse en el sistema, los datos son válidos y el usuario no se encuentra previamente registrado en el mismo Entonces se da de alta al usuario en el sistema	
Escenario 2: El usuario ya se encuentra registrado Dado un usuario registrado Cuando intenta registrarse ingresando los datos correspondientes Entonces se muestra un mensaje diciendo que dicho usuario ya existe	

RF4 Registro de cuidadores y paseadores

Historia de Usuario 04: Registro de cuidadores	
Como:	Usuario del sistema
Quiero:	Registrarme como “cuidador” o “paseador” de mascotas
Para:	Empezar a realizar dichas funciones
Estimación:	5 SP
Criterios de aceptación	
Escenario 1: El cuidador/paseador se registra con datos válidos Dado un usuario Cuando intenta registrarse en el sistema, los datos son válidos y el cuidador/paseador no se encuentra previamente registrado en el mismo Entonces se da de alta al cuidador/paseador en el sistema	
Escenario 2: El cuidador/paseador ya se encuentra registrado Dado un cuidador/paseador registrado Cuando intenta registrarse ingresando los datos correspondientes Entonces se muestra un mensaje diciendo que dicho cuidador/paseador ya existe	

RF5 Crear perfil de usuarios

Historia de Usuario 05: Crear perfil de usuarios	
Como:	Usuario
Quiero:	Crear un perfil personal
Para:	Proporcionar información a los cuidadores y paseadores
Estimación:	3 SP
Criterios de aceptación	
Escenario 1: El usuario se crea un perfil Dado un usuario Cuando intenta crearse un perfil para exponerse en el sistema y poder así utilizar el servicio Entonces se guarda dicho perfil	

RF6 Crear perfil de cuidadores y de paseadores

Historia de Usuario 06: Crear perfil de cuidadores/paseadores	
Como:	Usuario del sistema del tipo “Cuidador” o “Paseador”
Quiero:	Crear un perfil personal
Para:	Que los otros usuarios puedan conocer más acerca del cuidador o paseador
Estimación:	5 SP
Criterios de aceptación	
Escenario 1: El cuidador/paseador se crea un perfil Dado un cuidador/paseador Cuando intenta crearse un perfil para exponerse en el sistema y poder así ser contratado Entonces se guarda dicho perfil	

RF7 Búsqueda de cuidadores y paseadores aplicando distintos filtros

Historia de Usuario 07: Búsqueda de cuidadores y paseadores aplicando filtros	
Como:	Usuario del sistema
Quiero:	Aplicar distintos filtros en la búsqueda
Para:	Poder contratar un servicio detallado y a gusto del cliente
Estimación:	13 SP
Criterios de aceptación	
Escenario 1: Los filtros aplicados muestran al menos un resultado Dado un usuario del sistema Cuando intenta buscar a un cuidador o paseador con determinadas características Entonces se muestra el resultado acorde al filtro ingresado	
Escenario 2: Los filtros aplicados no muestran ningún resultado Dado un usuario del sistema Cuando intenta buscar a un cuidador o paseador con determinadas características Entonces se muestra un mensaje indicando que no se encontraron resultados	

RF8 Calificar a paseadores/cuidadores

Historia de Usuario 08: Calificar a cuidadores y paseadores	
Como:	Usuario del sistema
Quiero:	Calificar a los cuidadores y paseadores
Para:	Dejar mi opinión acerca del servicio ofrecido por los mismos y ayudar a los otros clientes a la hora de contratar el servicio deseado
Estimación:	5 SP
Criterios de aceptación	
Escenario 1: Calificación de un cuidador o paseador Dado un usuario del sistema Cuando desea calificar a un cuidador o paseador luego de haber contratado el servicio Entonces se registra dicha calificación en el sistema y se ingresa en el perfil del cuidador o paseador	

RF9 Realizar comentarios sobre paseadores/cuidadores

Historia de Usuario 09: Realizar comentarios sobre cuidadores y paseadores	
Como:	Usuario del sistema
Quiero:	Dejar comentarios acerca de los cuidadores y paseadores
Para:	Dejar registrado en el sistema la experiencia vivida a la hora de usar el mismo
Estimación:	5 SP
Criterios de aceptación	
Escenario 1: Se ingresa un comentario a un cuidador o paseador Dado un usuario del sistema Cuando desea registrar un comentario a un cuidador o paseador luego de haber contratado el servicio Entonces se registra dicho comentario en el sistema y se ingresa en el perfil del cuidador o paseador	

RF10 Obtener localización por GPS cuando la mascota es paseada

Historia de Usuario 10: Obtener localización por GPS de la mascota	
Como:	Usuario del sistema
Quiero:	Ver en tiempo real la ubicación de mi mascota
Para:	Tener un control acerca de donde la misma se encuentra ya sea a cargo de un cuidador o un paseador
Estimación:	13 SP
Criterios de aceptación	
<p>Escenario 1: Visualizo la ubicación del cuidador o paseador que contraté Dado un usuario del sistema Cuando desea tener conocimiento acerca de la ubicación de su mascota Entonces el sistema le muestra en el mapa la ubicación de la misma</p>	

RF11 Realizar informe al terminar un paseo por parte de los paseadores

Historia de Usuario 11: Realizar un informe al terminar un paseo por parte de los paseadores	
Como:	Paseador del sistema
Quiero:	Realizar un informe sobre cómo resultó el paseo para la mascota y si pasó algo durante el paseo
Para:	Que el dueño de la mascota lo pueda visualizar
Estimación:	5 SP
Criterios de aceptación	
Escenario 1: Registro el informe con todos los datos correspondientes Dado un paseador del sistema Cuando desea un realizar un informe ingresando nombre de la mascota, lugar de paseo, descripción de cómo resultó dicho paseo y si hubo algún inconveniente o imprevisto en el mismo Entonces el sistema registra el informe y se lo envía al dueño de la mascota	
Escenario 2: Registro el informe con datos faltantes Dado un paseador del sistema Cuando desea un realizar un informe, pero no ingresando todos los datos requeridos Entonces el sistema informa al paseador mediante un mensaje de la falta de campos sin llenar	

RF12 Realizar informe diariamente por parte de los cuidadores

Historia de Usuario 12: Realizar informe diariamente por parte de los cuidadores	
Como:	Cuidador del sistema
Quiero:	Realizar un informe diariamente sobre cómo se encuentra la mascota y cómo se está adaptando a su “hogar temporal”
Para:	Que el dueño de la mascota lo pueda visualizar y quedarse tranquilo que su mascota se encuentra bien
Estimación:	5 SP
Criterios de aceptación	
Escenario 1: Registro del informe con todos los datos correspondientes Dado un cuidador del sistema Cuando desea un realizar un informe ingresando nombre de la mascota, descripción de cómo se está adaptando la mascota y cualquier otro comentario que crea pertinente Entonces el sistema registra el informe y se lo envía al dueño de la mascota	
Escenario 2: Registro del informe con datos faltantes Dado un cuidador del sistema Cuando desea un realizar un informe Entonces el sistema informará al cuidador mediante un mensaje de la falta de campos sin rellenar	

RF13 Baja usuario

Historia de Usuario 13: Baja de usuario	
Como:	Usuario del sistema
Quiero:	Poder darme de baja de la aplicación
Para:	No acceder más a la aplicación
Estimación:	3 SP
Criterios de aceptación	
Escenario 1: No disponiendo de ninguna contratación de servicio previa Dado un usuario del sistema Cuando desea darse de baja del mismo Entonces el sistema lo borra correctamente de su base de datos de usuarios	
Escenario 2: Disponiendo de al menos una contratación de servicio previa Dado un usuario del sistema Cuando desea darse de baja del mismo Entonces el sistema notificará al usuario indicándole que cuenta con una reserva realizada y que para proceder deberá primero cancelar dicha reserva	

RF14 Modificar usuario

Historia de Usuario 14: Modificar usuario	
Como:	Usuario del sistema
Quiero:	Poder modificar mis datos personales
Para:	Así poder mantener actualizado mi perfil y mi número de contacto, así como también mi dirección
Estimación:	3 SP
Criterios de aceptación	
Escenario 1: Modificación de los datos Dado un usuario del sistema Cuando desea actualizar sus datos personales Entonces el sistema actualiza dichos datos y los guarda en la base de datos de usuarios	
Escenario 2: Modificación de los datos dejando al menos un campo sin completar Dado un usuario del sistema Cuando desea actualizar sus datos personales Entonces el sistema informará al usuario mediante un mensaje de la falta de campos sin llenar	

RF15 Reservar cuidador

Historia de Usuario 15: Reservar cuidador	
Como:	Usuario del sistema
Quiero:	Poder reservar un cuidador para mi mascota seleccionando el cuidador que desee, en el periodo de tiempo que desee e ingresando los comentarios que me parezcan pertinentes
Para:	Así poder dejar a mi mascota al cuidado del mismo
Estimación:	3 SP
Criterios de aceptación	
Escenario 1: Registro de la reserva correctamente Dado un usuario del sistema Cuando desea reservar a un cuidador Entonces el sistema registrará dicha reserva y notificará al usuario indicándole que se completó la reserva correctamente	
Escenario 2: Registro de la reserva con la falta de algunos campos Dado un usuario del sistema Cuando desea reservar a un cuidador Entonces el sistema notificará al usuario acerca de la falta de campos necesarios sin llenar para completar el registro de la reserva	

RF16 Reservar paseador

Historia de Usuario 16: Reservar paseador	
Como:	Usuario del sistema
Quiero:	Poder reservar un paseador para mi mascota seleccionando el paseador que desee, por la horas que desee e ingresando los comentarios que me parezcan necesarios para el paseador
Para:	Así poder agendar cuando mi mascota será paseada
Estimación:	3 SP
Criterios de aceptación	
<p>Escenario 1: Registro de la reserva correctamente</p> <p>Dado un usuario del sistema</p> <p>Cuando desea reservar a un paseador</p> <p>Entonces el sistema registrará dicha reserva y notificará al usuario indicándole que se completó la reserva correctamente</p>	
<p>Escenario 1: Registro de la reserva con la falta de algunos campos</p> <p>Dado un usuario del sistema</p> <p>Cuando desea reservar a un paseador</p> <p>Entonces el sistema notificará al usuario acerca de la falta de campos necesarios sin llenar para completar el registro de la reserva</p>	

RF17 Pagar cuidador/paseador

Historia de Usuario 17: Pagar cuidador/paseador	
Como:	Usuario del sistema
Quiero:	Poder pagar al cuidador/paseador de mi mascota a través de la aplicación mediante el uso de tarjeta de crédito
Para:	Así no tener que manejar efectivo
Estimación:	5 SP
Criterios de aceptación	
Escenario 1: Realización del pago Dado un usuario del sistema Cuando desea pagar por un servicio utilizado Entonces el sistema notificará al usuario acerca de la realización del pago	

RF18 Contactar cuidador/paseador

Historia de Usuario 18: Contactar cuidador/paseador	
Como:	Usuario del sistema
Quiero:	Poder contactar directamente al cuidador o paseador de mi mascota para así poder comentarle acerca de alguna cuestión que crea pertinente y poder clarificar cualquier situación que pueda surgir
Para:	Así poder quedarme tranquilo que tengo la posibilidad de contactarlo en caso de necesitarlo.
Estimación:	5 SP
Criterios de aceptación	
Escenario 1: Luego de haber registrado una reserva de servicio Dado un usuario del sistema Cuando quiero contactar a un cuidador o paseador para comunicarle cualquier particularidad Entonces el sistema enviará el mensaje al cuidador o paseador y le indicará al usuario que le mensaje fue enviado correctamente	

RF19 Localizar cuidadores/paseadores cercanos a la ubicación del usuario vía GPS

Historia de Usuario 19: Localizar cuidadores/paseadores cercanos a la ubicación del usuario vía GPS	
Como:	Usuario del sistema
Quiero:	Poder localizar mediante GPS cuidadores o paseadores cercanos a mi ubicación
Para:	Así poder contratarlos
Estimación:	13 SP
Criterios de aceptación	
<p>Escenario 1: Al menos un cuidador o paseador visible en el mapa</p> <p>Dado un usuario del sistema</p> <p>Cuando quiero contratar el servicio de un cuidador o paseador presente en la aplicación</p> <p>Entonces el sistema me permitirá contratar a dicho cuidador o paseador</p>	

RF20 Preguntas frecuentes

Historia de Usuario 20: Preguntas frecuentes	
Como:	Usuario del sistema
Quiero:	Poder visualizar preguntas frecuentes sobre el funcionamiento de la aplicación y sobre cómo contratar a cuidadores/paseadores
Para:	Así poder clarificar mis dudas
Estimación:	1 SP
Criterios de aceptación	
<p>Escenario 1: Visualizo las preguntas frecuentes del sistema</p> <p>Dado un usuario del sistema</p> <p>Cuando quiero tener conocimiento acerca de determinadas cuestiones de la aplicación</p> <p>Entonces el sistema me permitirá visualizar dichas inquietudes para tener un mayor conocimiento de la misma</p>	

RF21 Información sobre la aplicación

Historia de Usuario 21: Información sobre la aplicación	
Como:	Usuario del sistema
Quiero:	Poder ver la información sobre la aplicación
Para:	Consultarla cuando lo crea necesario
Estimación:	1 SP
Criterios de aceptación	
Escenario 1: Visualizo la información sobre la aplicación Dado un usuario del sistema Cuando quiero tener conocimiento acerca de la información de la aplicación Entonces el sistema me permitirá acceder a la misma	

RF22 Cancelar reserva de paseador/cuidador

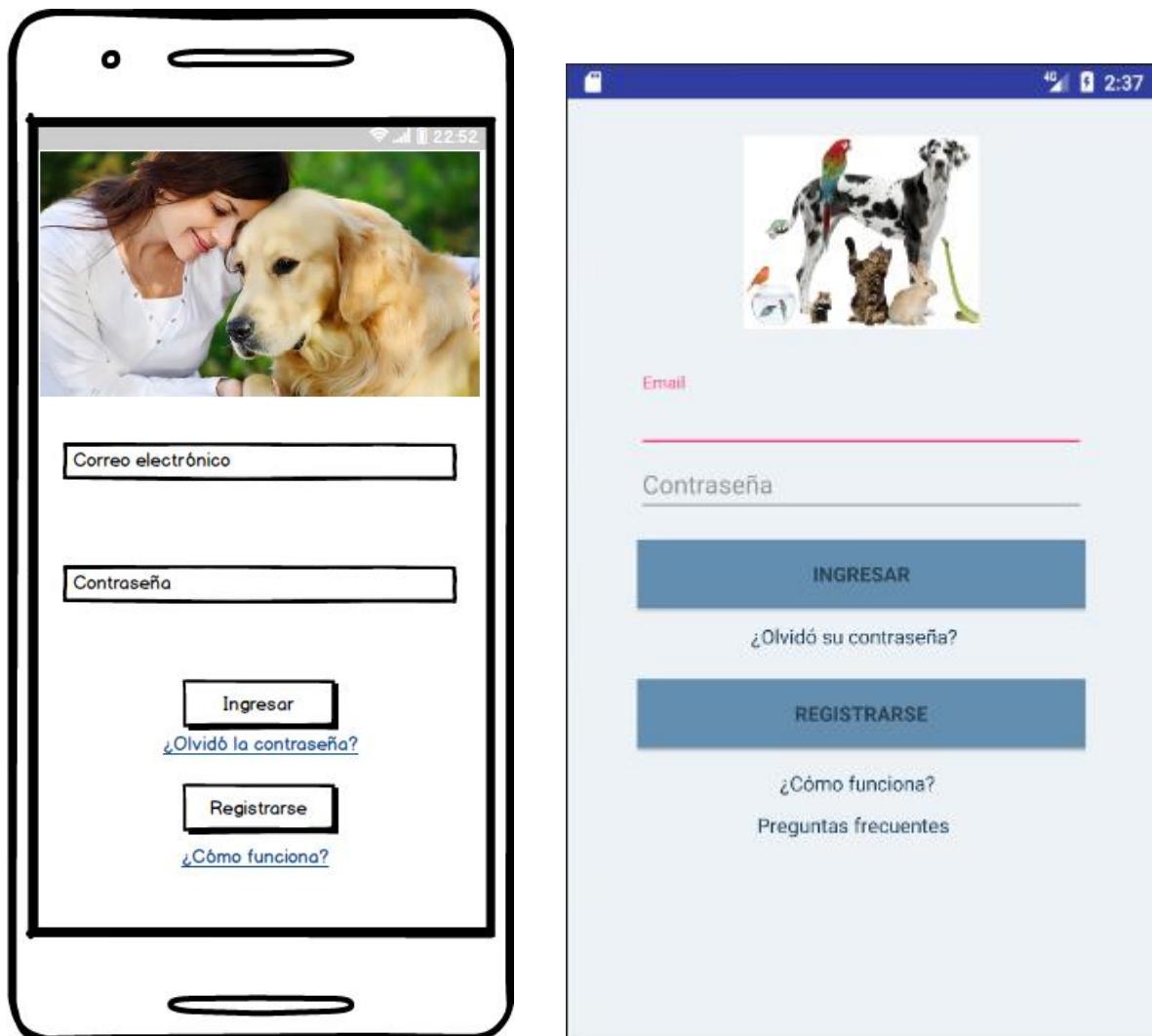
Historia de Usuario 22: Cancelar reserva de cuidador/paseador	
Como:	Usuario del sistema
Quiero:	Poder cancelar una reserva de cuidador o paseador
Para:	Así no tener que pagarla en caso que ya no la necesite
Estimación:	3 SP
Criterios de aceptación	
Escenario 1: Cancelar una reserva previamente realizada Dado un usuario del sistema Cuando quiero cancelar la contratación de un servicio Entonces el sistema le indicará al usuario que la reserva fue cancelada correctamente	
Escenario 2: No se encuentran reservas registradas Dado un usuario del sistema Cuando quiero cancelar la contratación de un servicio Entonces el sistema no mostrará ninguna reserva ya que no se contrató ningún servicio previamente	

5. Bocetos y diseños de la interfaz de usuario de la aplicación

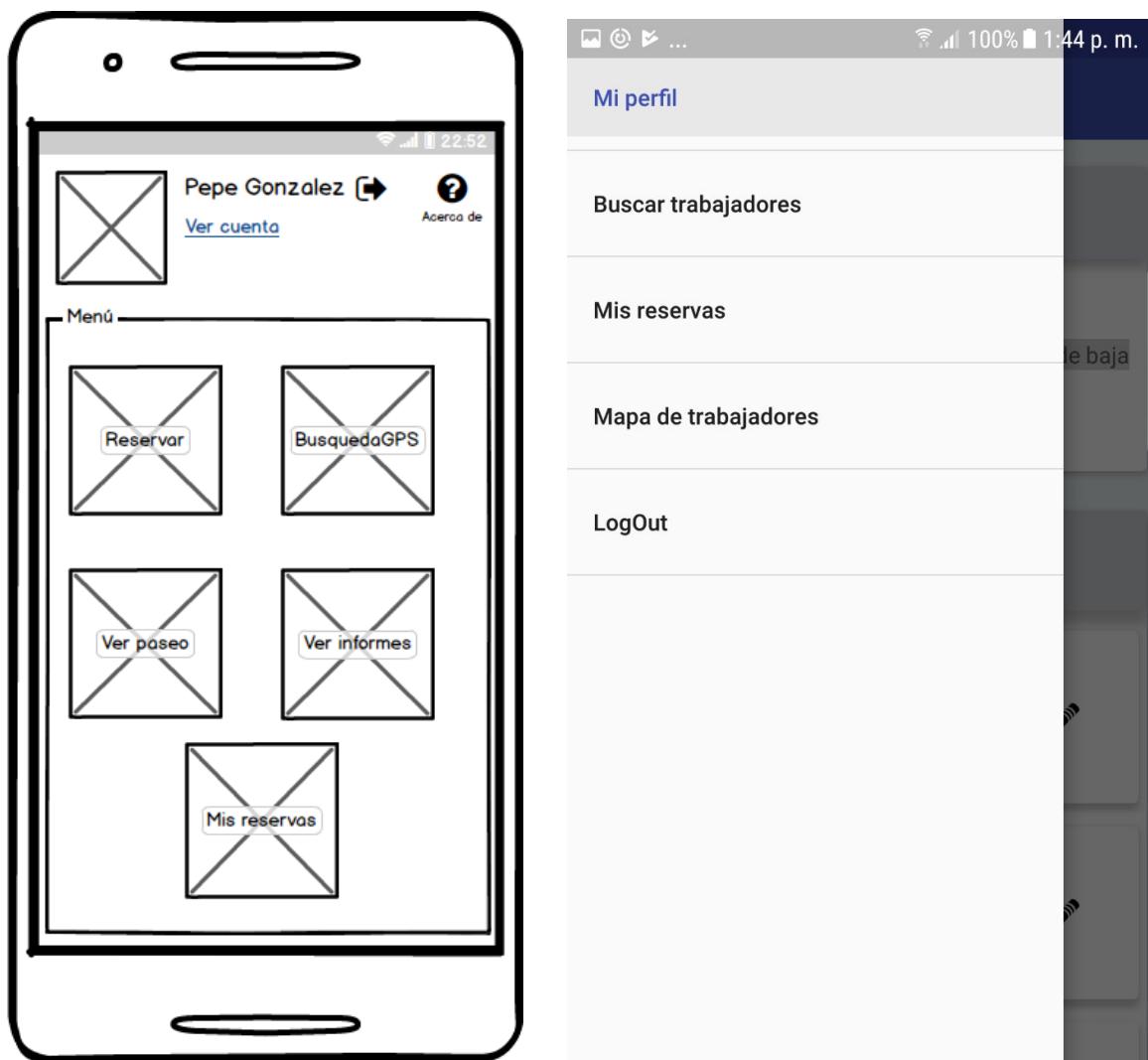
Con el objetivo de lograr una mayor comprensión sobre la aplicación a desarrollar decidimos hacer pequeños bocetos o *mockups* de la interfaz de usuario a modo de imaginarnos como sería la misma y de esta manera facilitar su desarrollo.

Para realizar los bocetos se utilizó la herramienta Blasmiq Mockups [6], la misma se trata de una aplicación de escritorio que permite diseñar de forma sencilla bocetos de interfaz de usuario.

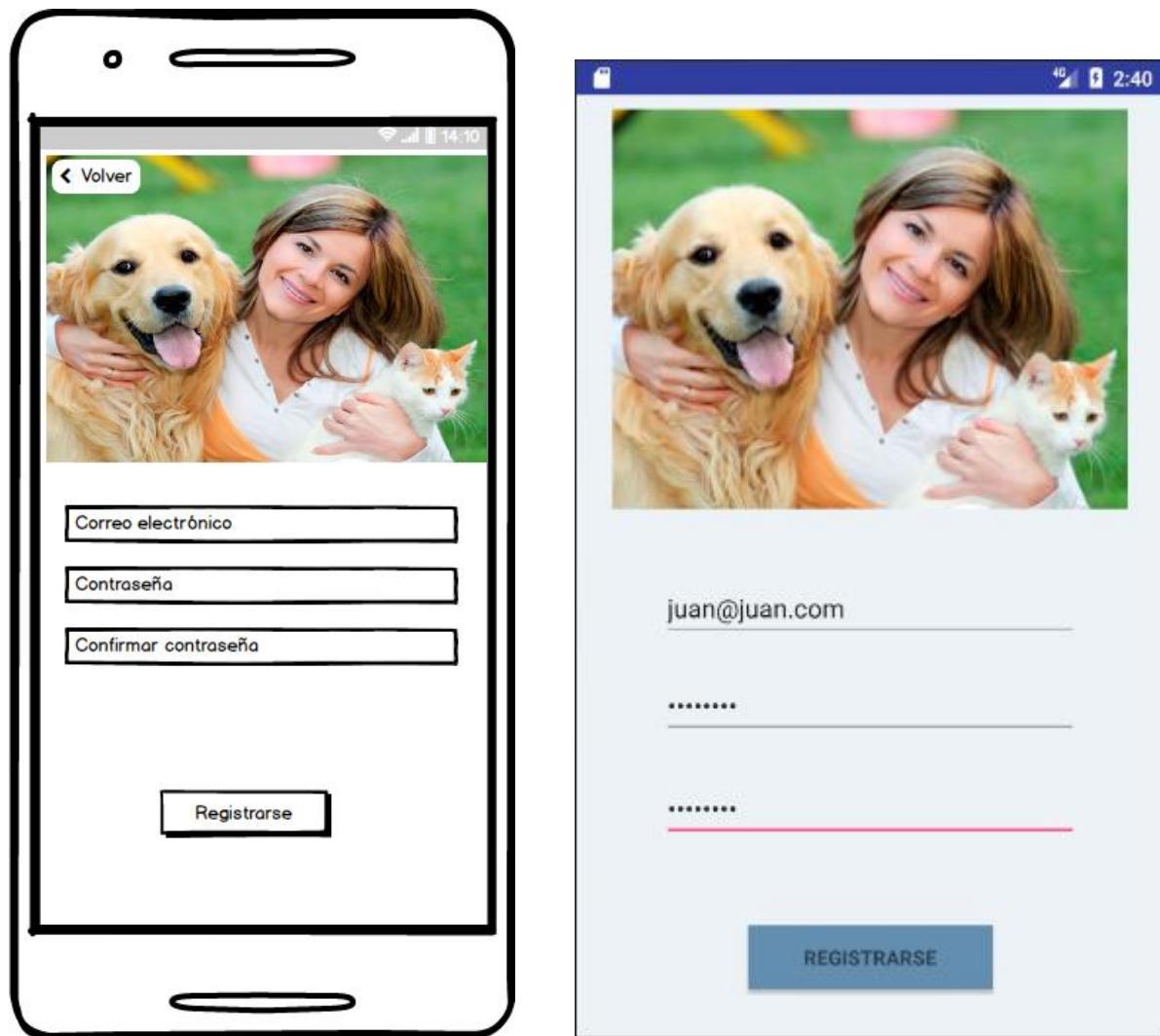
5.1 Log In



5.2 Menú principal



5.3 Registro de usuario



5.4 Registro de cuidadores y paseadores

The image displays two screenshots of a mobile application interface for user registration.

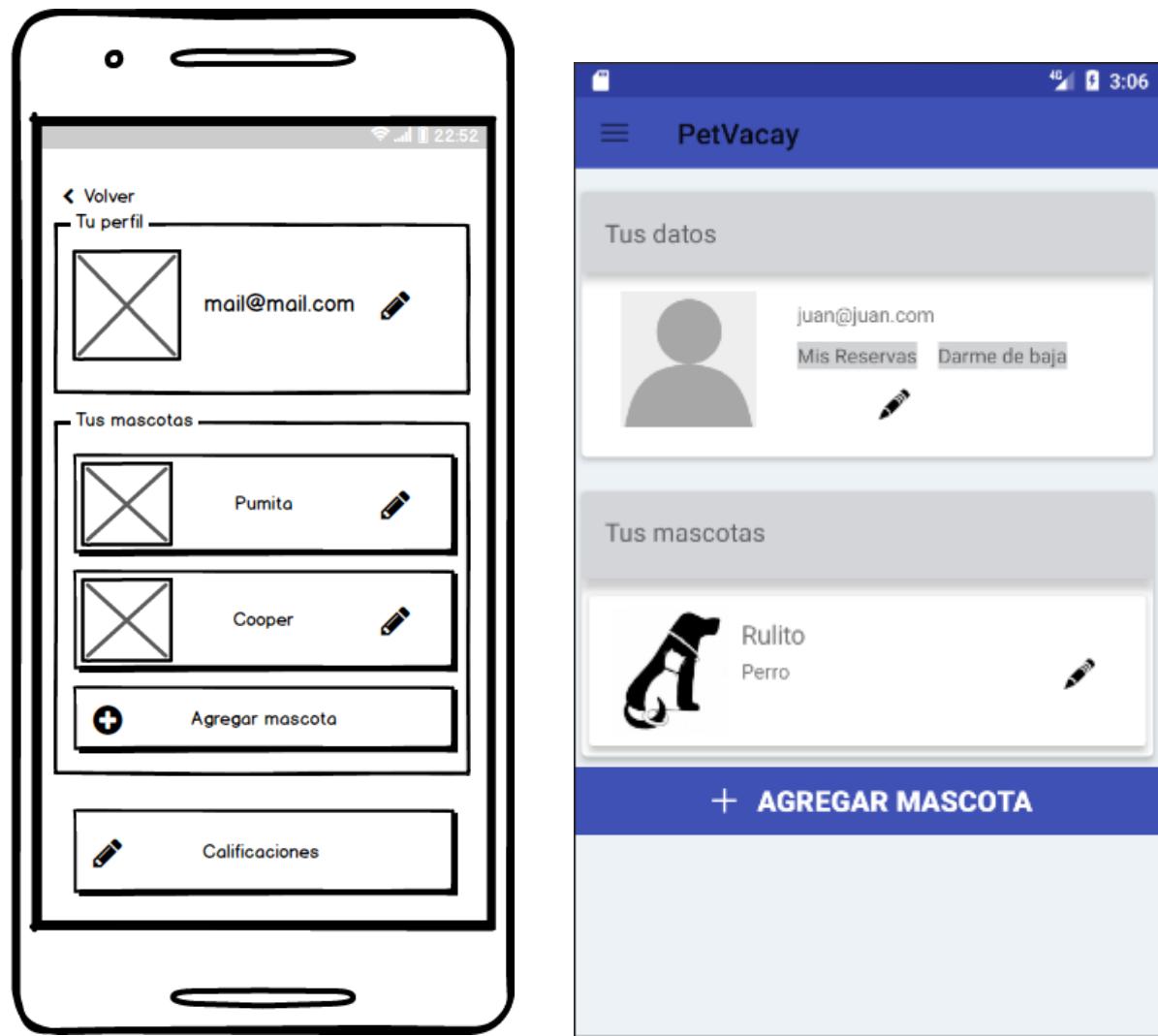
Screenshot 1 (Left): Registration Form

- Header: "Correo electrónico" (Email) and "Contraseña" (Password).
- Section: "Mascotas capacitadas" (Trained pets) with buttons for adding (+) or removing (-) "Perro" (Dog) and "Gato" (Cat).
- Section: "Disponibilidad" (Availability) with checkboxes for days of the week: Lunes, Martes, Miércoles, Jueves, Viernes, Domingo, and Sábado.
- Section: "Seleccione el barrio" (Select neighborhood) with a dropdown menu.
- Buttons: "Cuidador" (Caregiver), "Paseador" (Walker), and "Registrarse" (Register).

Screenshot 2 (Right): Profile and Availability Selection

- Header: "2:43".
- Image: A woman holding a golden retriever dog and an orange and white cat.
- Email input field: "pedro@pedro.com".
- Text input fields: "....." and ".....".
- Checkboxes:
 - "Puedo pasear mascota" (Can walk pets) is checked.
- Section: "Días disponibles" (Available days) with checkboxes for "Domingo", "Lunes" (which is checked), and "Martes".

5.5 Cuenta de usuario



5.6 Perfil del usuario



5.7 Menú para agregar mascota

The image displays two side-by-side screenshots of a mobile application interface for adding a pet. The left screenshot shows a smartphone screen with a white background and black borders. The right screenshot shows a desktop or tablet screen with a blue header bar.

Smartphone Screenshot:

- Header:** Shows signal strength, battery level, and time (22:52).
- Buttons:** "Cancelar" (Cancel) and "Guardar" (Save).
- Section: Info**
 - Text input fields for "Nombre" (Name), "Edad" (Age), "Sexo" (Gender), "Raza" (Breed), and "Peso" (Weight).
 - A button labeled "Agregar foto" (Add photo) with a camera icon.
- Section: Mas info**
 - Checkboxes for "Amigable con otras mascotas" (Friendly with other pets): "Si" (Yes), "No" (No), "No estoy seguro" (Not sure).
 - Checkboxes for "Vacunas al dia" (Up-to-date vaccines): "Si" (Yes), "No" (No), "No estoy seguro" (Not sure).
 - Checkboxes for "Nivel de actividad" (Activity level): "Muy activo" (Very active), "Tranquilo" (Relaxed), "Duerme mucho" (Sleeps a lot).
 - A text input field for "Instrucciones o cualquier informacion que crea importante sobre su mascota" (Instructions or any information you consider important about your pet).

Desktop Screenshot:

- Header:** Shows signal strength, battery level, and time (3:05).
- Title:** "Datos de tu mascota" (Pet data).
- Text Input:** "Nombre" (Name) with a placeholder icon of a dog's head.
- Image:** A small icon of a dog's head with the text "Cambiar foto" (Change photo) below it.
- Text Input:** "Edad" (Age).
- Text Input:** "Peso" (Weight).
- Section: Tipo mascota**
 - Radio buttons for "Gato" (Cat) and "Perro" (Dog).
- Section: Sexo**
 - Radio buttons for "Hembra" (Female) and "Macho" (Male).
- Section: Mas datos**
 - Checkboxes for "Amigable" (Friendly) and "Tiene vacunas" (Has vaccines).

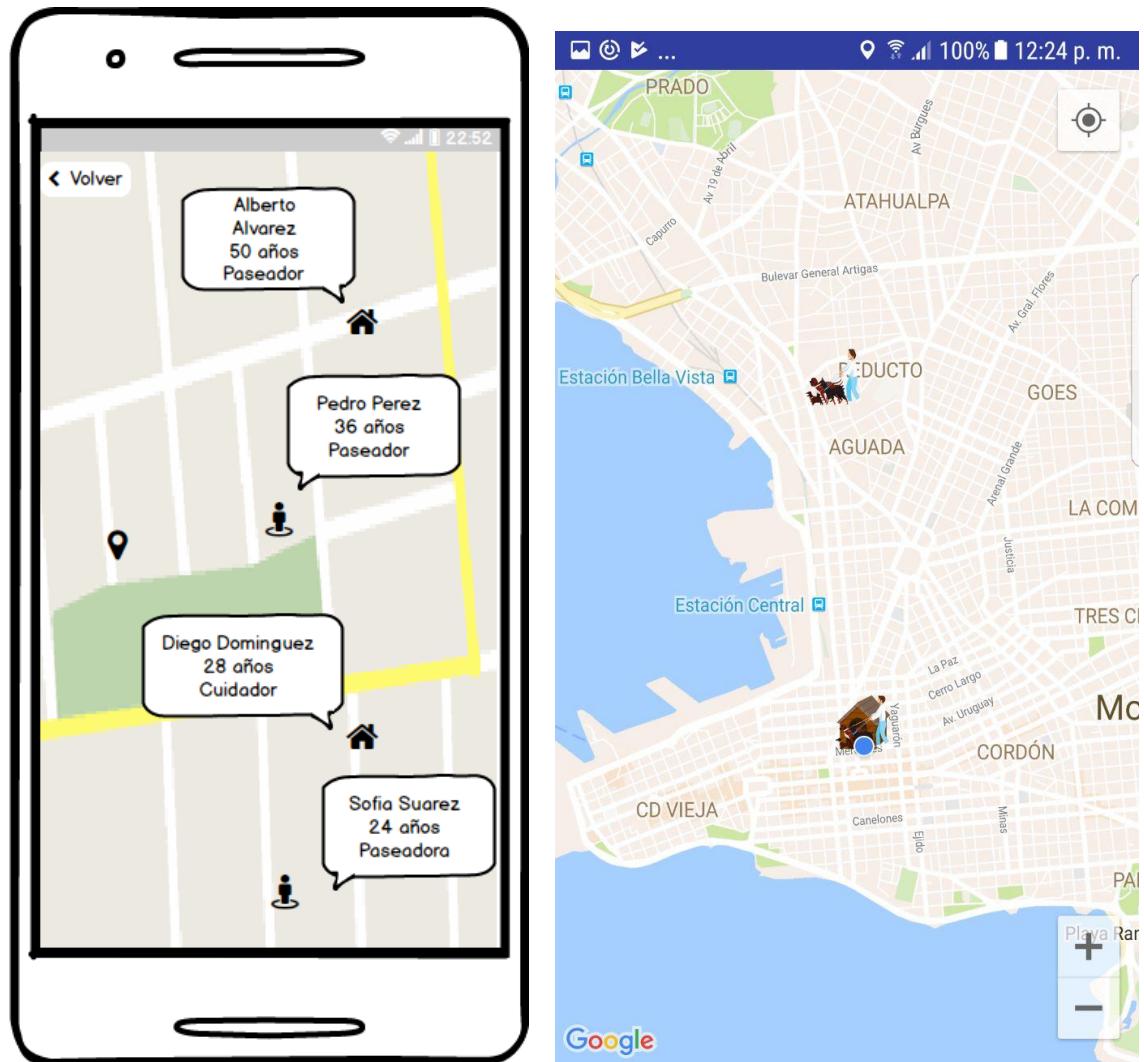
5.8 Menú de búsqueda y reserva de cuidadores/paseadores

The image displays two side-by-side screenshots of a mobile application interface.

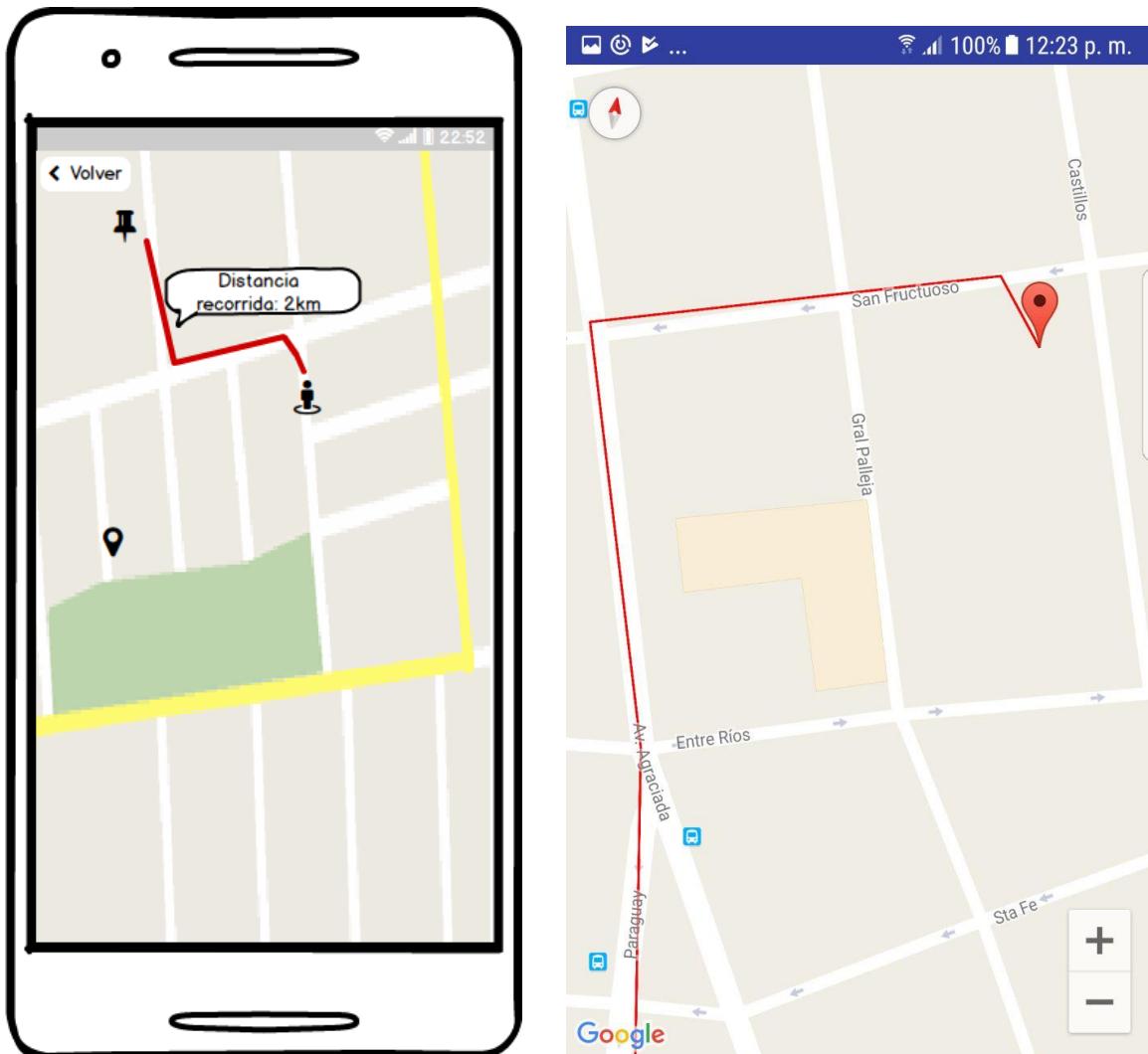
Left Screenshot: A smartphone screen showing a search interface. At the top are tabs for "Cuidador" and "Paseador". Below them are fields for "Localidad" (set to "Cidade"), "Inicio" (set to "27/04/2017"), and "Fin" (set to "27/04/2017"). There is also a search bar with placeholder text "Ingrese un nombre..." and a magnifying glass icon. The results section shows three entries: "Pedro Perez" (Info Reservar), "Sofia Suarez" (Info Reservar), and "Martin Rodriguez" (Info Reservar). Each entry has a small profile picture placeholder.

Right Screenshot: A search results page for "Cuaireim". The search bar contains "Cuaireim" and has a checked checkbox for "Buscar paseadores". A large "BUSCAR" button is below it. The results area is currently empty, showing a large gray placeholder. Below this is a card for "Paseos Company". It features a silhouette of a person walking a dog, the name "Paseos Company", the title "paseador y cuidador", a rating of five stars, and service fees: "Estadia: \$ 450" and "Paseo: \$110".

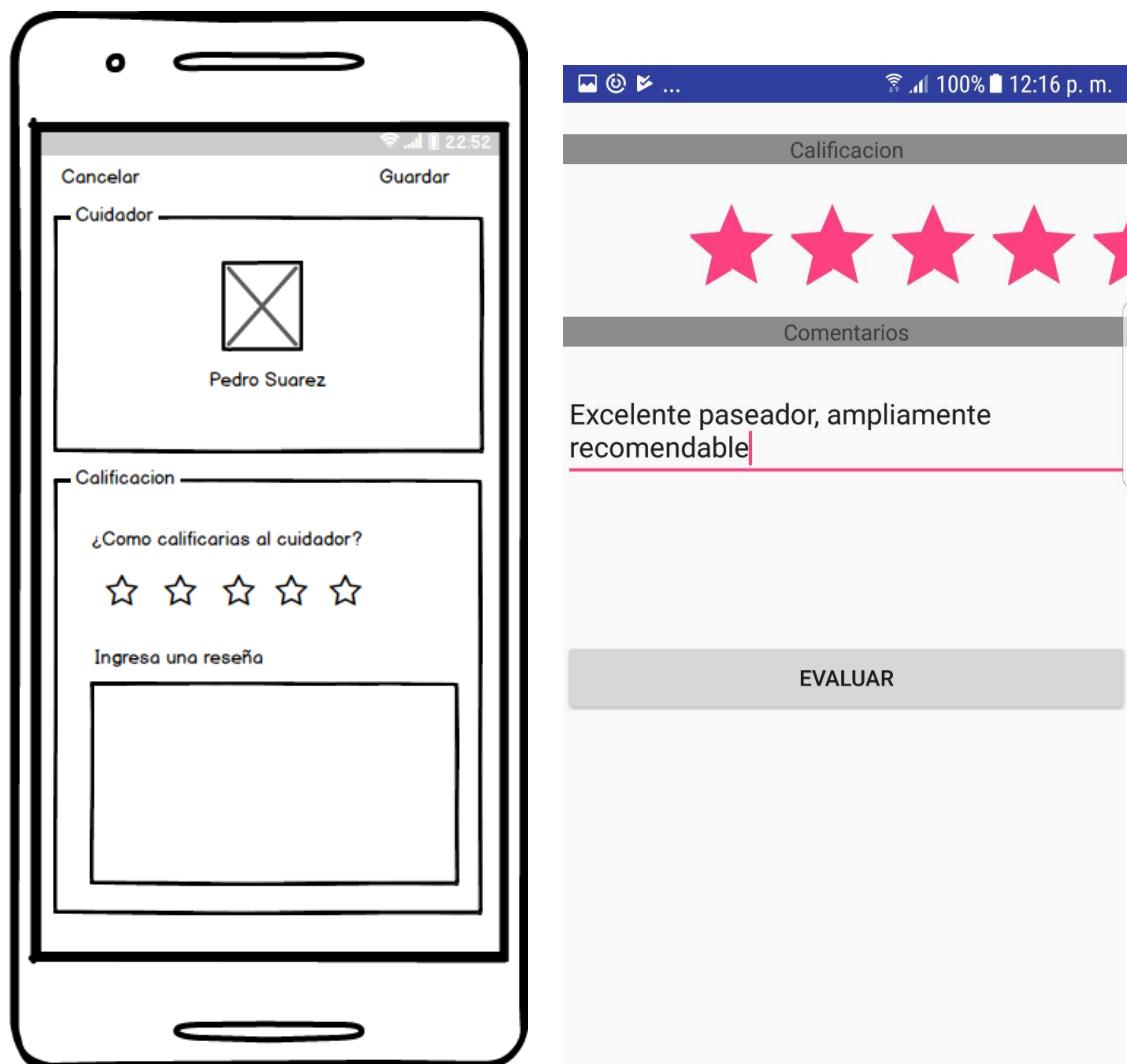
5.9 Mapa de cuidadores/paseadores cercanos



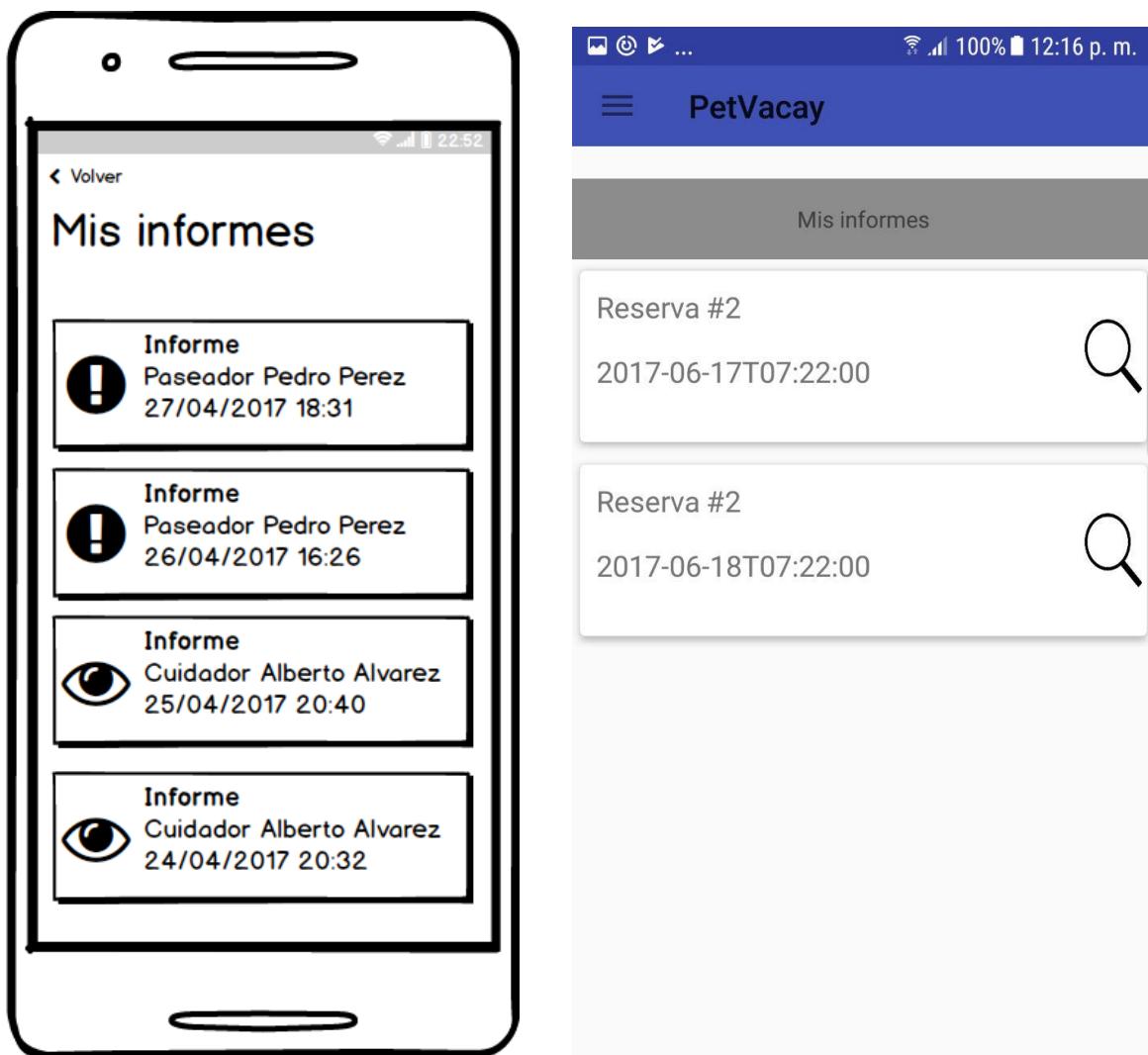
5.10 Mapa de paseador



5.11 Calificación de cuidadores/paseadores



5.12 Feedback con informes



5.13 Menú de pago

The image shows a smartphone displaying a payment menu and a separate payment form.

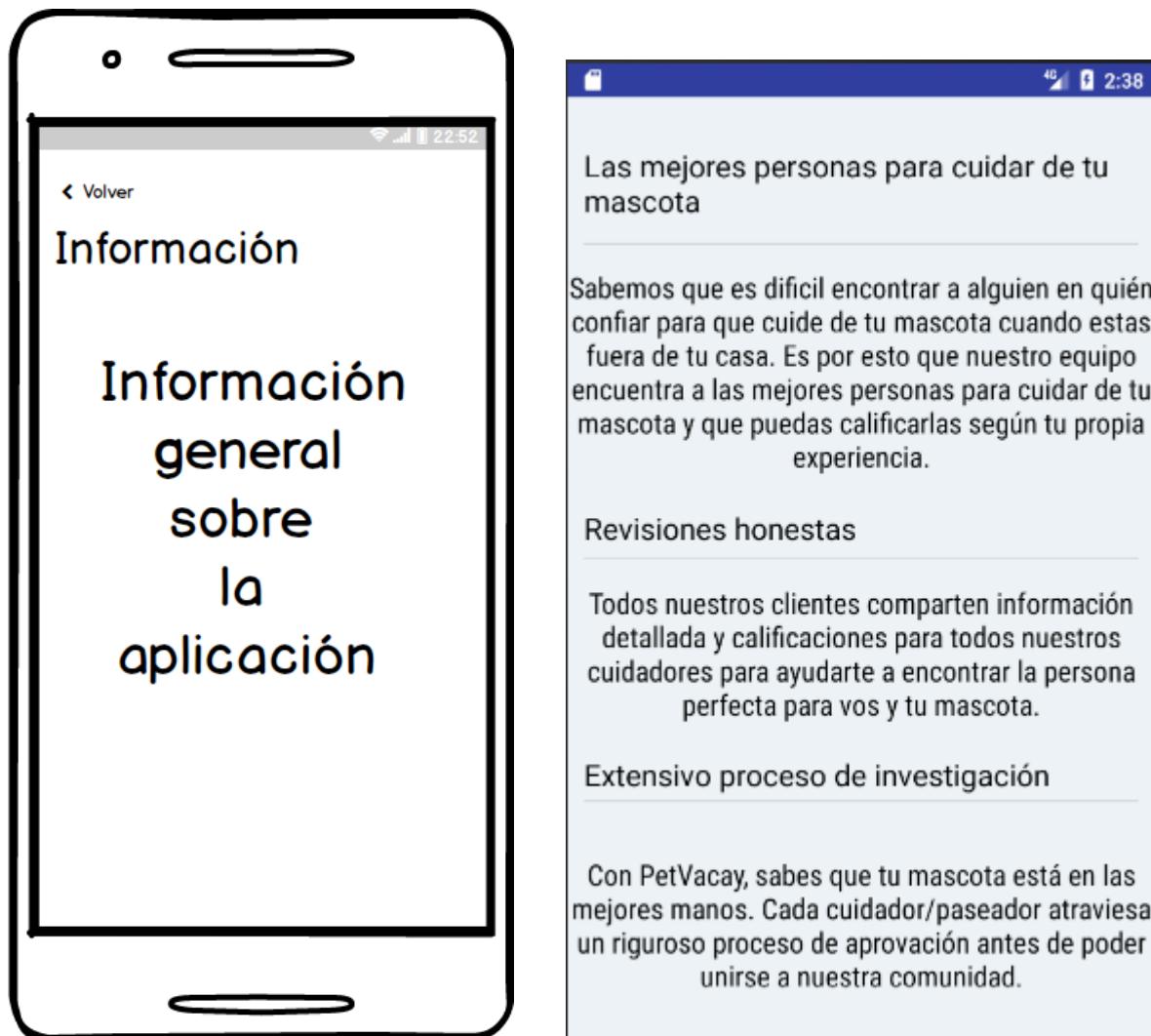
Smartphone Screen (Left):

- Detalle del pago:**
 - Paseo de mascota
 - 27/04/2017
 - 1 hora
 - \$ 200**
- Método de pago:**
 - Tarjeta **VISA**
 - Número de tarjeta **1234-5678-9012-3456**
 - Fecha vencimiento **06/2018**
 - CCV **123**
- Pagar**

Separate Payment Form (Right):

- Monto a pagar: \$110**
- Numero tarjeta de credito**
- Fecha vencimiento**
- CCV**
- Pago de la reserva.**
- PAGAR**

5.14 Información de la aplicación



6. Gestión de Proyecto

6.1 Métricas utilizadas

6.1.1 Story Points

Los *Story Points* representan una medida arbitraria que se les asigna a cada historia de usuario con el objetivo de poder medir el esfuerzo requerido para implementarla.

Si bien se puede utilizar cualquier escala que el equipo decida, generalmente se adhiere a escalas como pueden ser las siguientes

- 1,2,4,8,16
- X Small, Small, Medium, Large, Extra Large
- Serie de Fibonacci (1, 2, 3, 5, 8...,45)

Lo interesante de planificar los *sprints* utilizando *story points*, a diferencia de horas de trabajo, es que el equipo adquiere mayor nivel de abstracción sobre el esfuerzo requerido para completar una determinada historia de usuario.

En nuestro proyecto optamos por una escala de Fibonacci modificada la cuál es la siguiente

Fibonacci modificado: 1,2,3,5,13,40,100

El motivo de su elección es que, con dichos números, como se dijo anteriormente, facilitan y ayudan mucho más a estimar el esfuerzo requerido para completar la historia de usuario en cuestión.

6.1.1.1 Traducción de Story Points a días

La siguiente tabla ilustra la traducción aproximada de los puntos de historia a días ideales, sin contemplarse pequeñas desviaciones que hayan surgido durante el proceso de desarrollo.

Story Points (SP)	Ideal Days (días)
1	1
3	2
5	3
13	6

6.1.2 Velocidad del equipo

Como se especificará con más detalle en el apartado 6.2.4 (Planificación de Sprints), al contar con dos meses para la entrega y no tener referencia de velocidad para el primer sprint, calcularemos la misma en base a la cantidad de iteraciones definidas en dicho apartado y el tiempo que tenemos para el desarrollo, pudiendo cambiarse esta velocidad al finalizar el primer sprint en caso de que no se cumpla con el objetivo fijado para el mismo.

Tenemos dos meses para el desarrollo de nuestra aplicación con cuatro iteraciones (el motivo de esto está explicado en el apartado antes mencionado), entonces si dividimos la cantidad total de story points de todas las historias de usuario por la cantidad de iteraciones planeadas obtendremos la velocidad inicial del equipo.

En este caso:

Velocidad del equipo = total de story points / cantidad de iteraciones

Velocidad del equipo = 112 / 4

Velocidad del equipo = 28 story points/sprint

6.2 Planificación

6.2.1 Etapas del proyecto

El proyecto a realizar cuenta con cuatro etapas fundamentales:

I. Exploración

En esta etapa se registra toda la información relevante al producto a realizar, público al que apunta y su grupo de interesados, se releva sobre el alcance general de la aplicación, así como también sus requerimientos funcionales y no funcionales, y por último se especifica la metodología de trabajo que se utilizará durante el proyecto.

II. Iniciación

En esta etapa se documenta la planificación general del trabajo, el detalle del cronograma junto con el tiempo estimado para cada tarea y se especifica cómo se mitigarán los posibles riesgos que puedan surgir durante el proyecto.

Es también en esta etapa que se realizan los planes de SQA y SCM y se define el *Product Backlog* inicial.

III. Desarrollo

En esta etapa se hace énfasis en el desarrollo de la aplicación móvil, tanto en la parte del *back-end* como en la interfaz de usuario (*front-end*), se documenta además la arquitectura utilizada, así como también las decisiones de diseño tomadas y su justificación.

Se utiliza *refactoring* para mejorar el código y se evalúa además la usabilidad y experiencia de usuario de la aplicación.

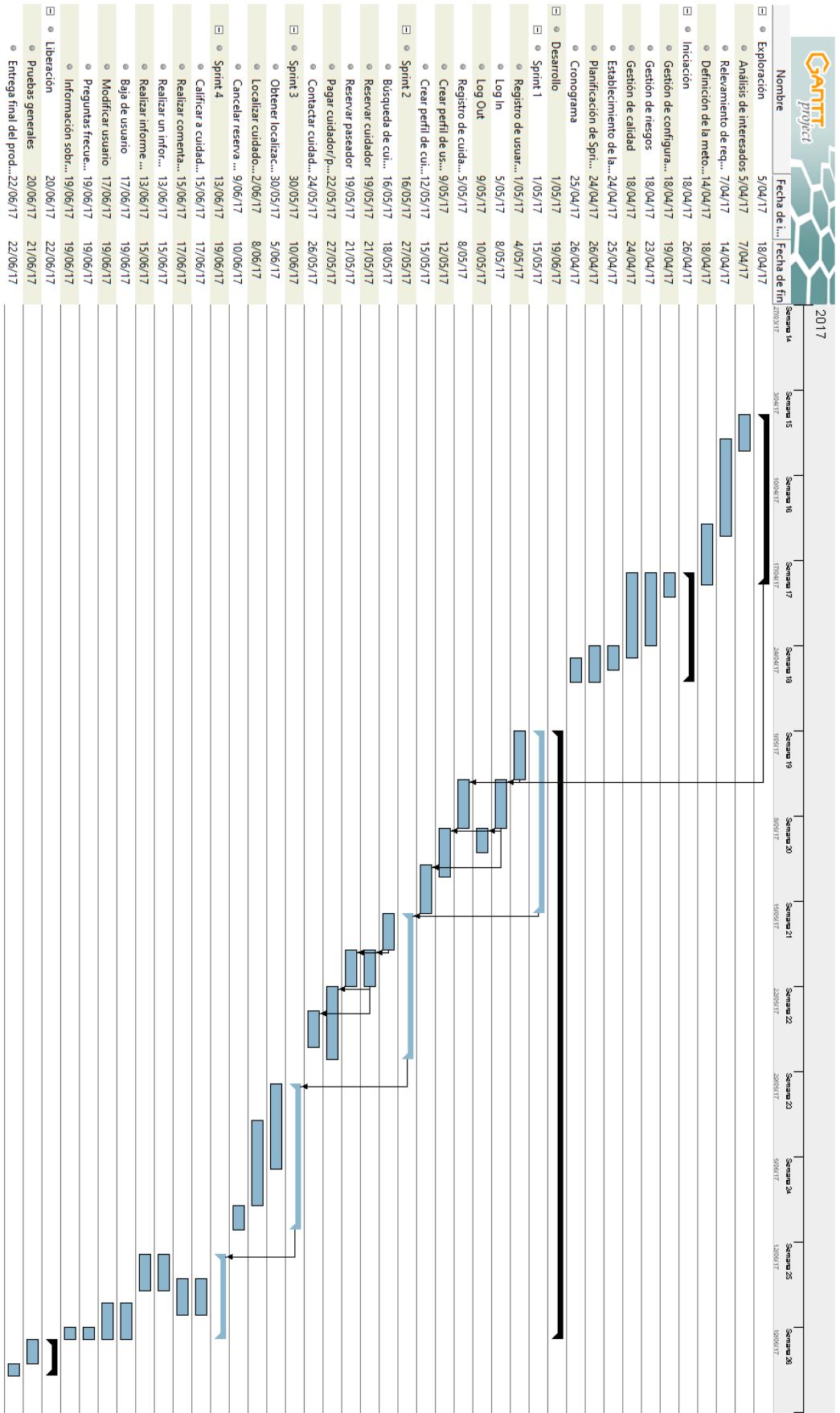
En esta etapa, como se especificó anteriormente, se utiliza Scrum como marco de trabajo.

IV. Liberación

En esta etapa se ejecutan todas las pruebas generales de la aplicación asegurándose de que se cumplan todos los requerimientos funcionales sin ningún defecto, y a su vez se terminan redondeando todos los detalles finales de la documentación, dejando todo pronto para la entrega del proyecto.

6.2.2 Cronograma

A continuación, se presenta el cronograma general del trabajo especificando las cuatro etapas del proyecto mencionadas en el apartado anterior junto con los *sprints* a realizarse y la estimación en días para cada tarea.



6.2.3 Detalle de los entregables

Primer entregable - 27/4

El primer entregable contará con toda la documentación relacionada con el alcance y justificación de nuestro producto, así como también la elección y justificación de la metodología de trabajo utilizada, bocetos de la interfaz de usuario esperada, y la planificación del trabajo a realizar incluyendo un cronograma, gestión de riesgos y la gestión de configuración.

Segundo entregable – 22/6

El segundo entregable incluirá todo lo descrito en el primer entregable más todos los detalles de la implementación de la aplicación, incluyendo la arquitectura utilizada, diagramas UML, justificaciones de diseño, aseguramiento de calidad, y la aplicación funcionando.

6.2.4 Planificación de los Sprints

Al seguir un desarrollo ágil, debemos definir tempranamente los *sprints*.

La metodología Scrum propone ir realizando iteraciones cada un período de entre dos a cuatro semanas sobre el producto e ir haciendo revisiones al final de cada una de manera de poder sacar conclusiones y en base a estas continuar con la velocidad del equipo o modificarla en base a los resultados obtenidos.

Dada nuestra forma de desarrollo en trabajos anteriores, creemos que utilizar *sprints* de un corto periodo de tiempo resulta más beneficioso durante el proceso de desarrollo, ya que permite ir evaluando en cada uno lo realizado y tener una idea más concisa sobre el alcance general del producto a entregar y responder a cambios o inconvenientes en caso de que estos surjan.

Para la segunda entrega, como se detalló en el punto anterior, hemos acordado, entre otras cosas, entregar un producto sólido, funcional y de buena calidad. Como contamos con dos meses para el desarrollo (22 de junio del 2017), creemos conveniente establecer los *sprints* cada dos semanas, contando con un total de cuatro *sprints* en los cuales al final de cada uno, se priorizarán y se definirán las historias de usuario a realizarse en el siguiente.

La cantidad de historias de usuario a realizarse en cada sprint dependerán básicamente de la velocidad del equipo (la cual está definida en el apartado 6.1.2) y de los *story points* asignados a cada historia de usuario (especificados en el apartado 4.3.4 para cada una), puede que se modifiquen al re-evaluar al final del sprint la velocidad del equipo.

Al finalizar cada sprint, el equipo hará una revisión general del mismo (llamada *sprint retrospective*) en donde se evaluará si en dicho sprint se cumplió o no con la cantidad de historias de usuario acordadas para el mismo, y en caso contrario, re-estimar las mismas o modificar la velocidad del equipo. Por otro lado, también se planificarán las historias de usuario a realizarse en el siguiente sprint.

6.2.4.1 Cronograma de Sprints

A continuación, se detalla el cronograma de *sprints* propuesto para nuestro proyecto detallando las historias de usuario a realizarse en cada uno según lo evaluado en los puntos 6.1.2 Velocidad del equipo y el anterior.

Sprint 1: 01/05/2017 – 15/05/2017

- Historia de Usuario 01: Log In (Story Points: 5)
- Historia de Usuario 02: Log Out (Story Points: 3)
- Historia de Usuario 03: Registro de usuarios (Story Points: 5)
- Historia de Usuario 04: Registro de cuidadores y paseador (Story Points: 5)
- Historia de Usuario 05: Crear perfil de usuarios (Story Points: 3)
- Historia de Usuario 06: Crear perfil de cuidadores/paseadores (Story Points: 5)

Sprint 2: 16/05/2017 – 29/05/2017

- Historia de Usuario 07: Búsqueda de cuidadores y paseadores aplicando filtros (Story Points: 13)
- Historia de Usuario 15: Reservar cuidador (Story Points: 3)
- Historia de Usuario 16: Reservar paseador (Story Points: 3)
- Historia de Usuario 17: Pagar cuidador/paseador (Story Points: 5)
- Historia de Usuario 18: Contactar cuidador/paseador (Story Points: 5)

Sprint 3: 30/05/2017 – 12/06/2017

- Historia de Usuario 10: Obtener localización por GPS de la mascota (Story Points: 13)
- Historia de Usuario 19: Localizar cuidadores/paseadores cercanos a la ubicación del usuario vía GPS (Story Points: 13)
- Historia de Usuario 22: Cancelar reserva de cuidador/paseador (Story Points: 3)

Sprint 4: 13/06/2017 – 20/06/2017

- Historia de Usuario 08: Calificar a cuidadores y paseadores (Story Points: 5)
- Historia de Usuario 09: Realizar comentarios sobre cuidadores y paseadores (Story Points: 5)
- Historia de Usuario 11: Realizar un informe al terminar un paseo por parte de los paseadores (Story Points: 5)
- Historia de Usuario 12: Realizar informe diariamente por parte de los cuidadores (Story Points: 5)
- Historia de Usuario 13: Baja de usuario (Story Points: 3)
- Historia de Usuario 14: Modificar usuario (Story Points: 3)
- Historia de Usuario 20: Preguntas frecuentes (Story Points: 1)
- Historia de Usuario 21: Información sobre la aplicación (Story Points: 1)

Aclaraciones

Existen ciertas discrepancias entre el cronograma general de trabajo especificado en la parte 6.2.2 y la planificación de *sprints*, esto se da únicamente en la fecha de finalización de cada sprint. La razón de esto es que consideramos que las funcionalidades a implementarse se pueden desarrollar en la cantidad de días asignados en dicho cronograma, dejando uno o dos días para realizar técnicas de *refactoring* sobre el código, pruebas unitarias (como se especificará más adelante en el apartado 10 Gestión de la calidad) y en algunos casos, realizar un *spike* –un *spike* en una metodología ágil consiste en un período dentro del sprint durante el cual no se implementan directamente historias de usuario, pero se investiga sobre algún tema sobre el cuál el equipo no tenga suficiente experiencia y sea provechoso para los *sprints* posteriores - para lograr mayor conocimiento en áreas de desarrollo en las cuales no estamos muy informados como lo es durante los días finales del segundo sprint y antes del comienzo del tercero, el manejo del GPS del dispositivo.

Con respecto a los *story points* asignados a cada sprint, en el primero nos sobraron dos, los cuales pueden utilizarse posteriormente en el segundo y tercer sprint donde la cantidad de *story points* estimada a realizarse supera por una unidad la velocidad.

Nos parece importante ser conservadores en cuanto al tiempo que nos va a llevar cada tarea ya que el equipo no tiene experiencia con Andorid y además no tenemos proyectos anteriores de los cuales podamos obtener métricas para poder estimar de forma más precisa. Creemos que luego del primer sprint vamos a poder realizar mejores estimaciones y vamos a tener una mejor idea de cuál es en realidad la velocidad del equipo.

6.2.5 Sprints Reviews

Para la realización de los *reviews* al finalizar cada *sprint*, se decidió utilizar la herramienta Skype ya que es de fácil acceso y permite un flujo en la conversación bastante alto en caso de que no se pudieran efectuar reuniones en persona.

Otra de las cosas que se decidió fue, no emplear un día entero en realizar el *sprint review* debido a que no se contaba con mucho tiempo para entregar el producto final.

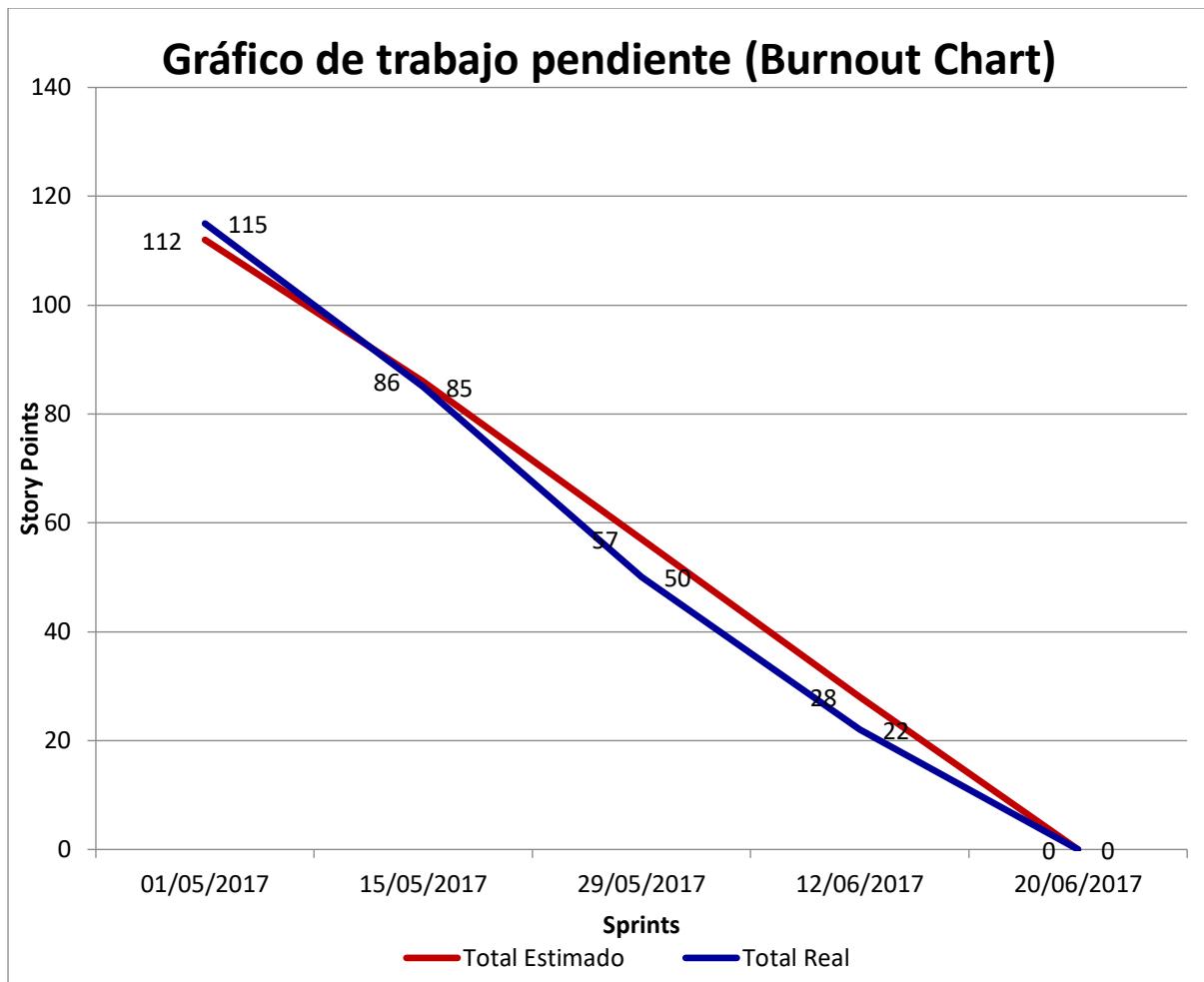
Por este motivo, lo que se hizo fue que al terminar el *sprint* se emplearan aproximadamente dos horas para realizar la *review* y enseguida comenzar con la siguiente de manera de aprovechar la mayor parte del tiempo posible.

6.2.6 Burnout Chart

En esta sección detallaremos como fue el esfuerzo real con respecto al estimado en un principio, así como también las fechas límites a cumplir de los distintos *sprints*.

Historias de Usuario: Tiempo estimado y real	Puntos de historia	Inicio 01/05/2017	15/05/2017	29/05/2017	12/06/2017	Fin 20/06/2017
Historia de usuario #1	5		5			
Tiempo Real		5	5			
Historia de usuario #2	3		3			
Tiempo Real		5	5			
Historia de usuario #3	5		5			
Tiempo Real		5	5			
Historia de usuario #4	5		5			
Tiempo Real		5	5			
Historia de usuario #5	3		3			
Tiempo Real		5	5			
Historia de usuario #6	5		5			
Tiempo Real		5	5			
Historia de usuario #7	13			13		
Tiempo Real		13		13		
Historia de usuario #8	5					5
Tiempo Real		5				5
Historia de usuario #9	5					5
Tiempo Real		3				3
Historia de usuario #10	13				13	
Tiempo Real		13			13	

Historia de usuario #11	5						5
Tiempo Real		3					3
Historia de usuario #12	5						5
Tiempo Real		3					3
Historia de usuario #13	3						3
Tiempo Real		3					3
Historia de usuario #14	3						3
Tiempo Real		3					3
Historia de usuario #15	3				3		
Tiempo Real		3			3		
Historia de usuario #16	3				3		
Tiempo Real		3			3		
Historia de usuario #17	5				5		
Tiempo Real		13			13		
Historia de usuario #18	5				5		
Tiempo Real		3			3		
Historia de usuario #19	13					13	
Tiempo Real		13				13	
Historia de usuario #20	1						1
Tiempo Real		1					1
Historia de usuario #21	1						1
Tiempo Real		1					1
Historia de usuario #22	3					3	
Tiempo Real		2				2	
Total Estimado		112	86	57	28	0	
Total Real		115	85	50	22	0	



Como se puede notar, lo que estimamos en un principio no terminó siendo lo que habíamos planificado ya que algunas historias de usuarios terminaron llevando más tiempo del pensado.

A su vez, nos llevó a realizar un esfuerzo mayor para la realización de las mismas, de manera que se pudiera llegar a la fecha de finalización de cada sprint con las distintas tareas terminadas para el mismo.

Como conclusión si bien hubo un pequeño desvío al comienzo del primer sprint, el equipo supo ajustarse a los tiempos y poder completar el resto de los sprints en los días previstos en un principio, por lo cual consideramos que la velocidad inicial propuesta no fue muy desviada de la realidad, sino que nos pudimos ajustar a ella y nos permitirá tomarla como referencia para futuros proyectos.

6.3 Comunicación

Teniendo en cuenta que los integrantes del equipo realizan varias materias en común por lo que tienen horarios parecidos en la facultad y se ven varias veces a la semana se decidió realizar como mínimo dos reuniones presenciales con el objetivo de poder desarrollar en conjunto y de poder colaborar entre todos con las distintas dificultades que puedan surgir.

El resto de los días en los que no es posible reunirse el equipo decidió trabajar en forma remota a través de Skype.

También se creó un grupo de Whatsapp, el cual permite tener una comunicación fluida entre todos los integrantes.

Se analizará el uso de Slack, ya que la misma es una herramienta que permite mantener conversaciones, intercambiar archivos, fotos, etc. Una de las características principales que posee es la posibilidad de organizar la comunicación en canales, por lo que las conversaciones se pueden organizar según el tema.

A lo largo del desarrollo de la aplicación no resultó necesario la utilización de Slack ya que las demás herramientas fueron suficientes.

Por último, vale la pena destacar que se mantiene una comunicación fluida con el profesor del curso ya sea a nivel presencial en el horario de clase, así como también mediante correo electrónico.

6.4 Herramienta de gestión Trello

Trello^[7] es una herramienta web online para la gestión de proyectos basada en la metodología Kanban, se pueden crear distintas listas o grupos en las cuales se van agregando tareas, representadas como tarjetas virtuales, las cuales se pueden mover de una lista a otra y pueden asignarse a distintos miembros del equipo.

Utilizamos esta herramienta como una forma de facilitar visualmente el trabajo por hacer, el trabajo en proceso y el trabajo completado, así como también los integrantes del equipo asignados para cada tarea, de esta manera se logra una mayor organización sobre el desarrollo de nuestra aplicación.

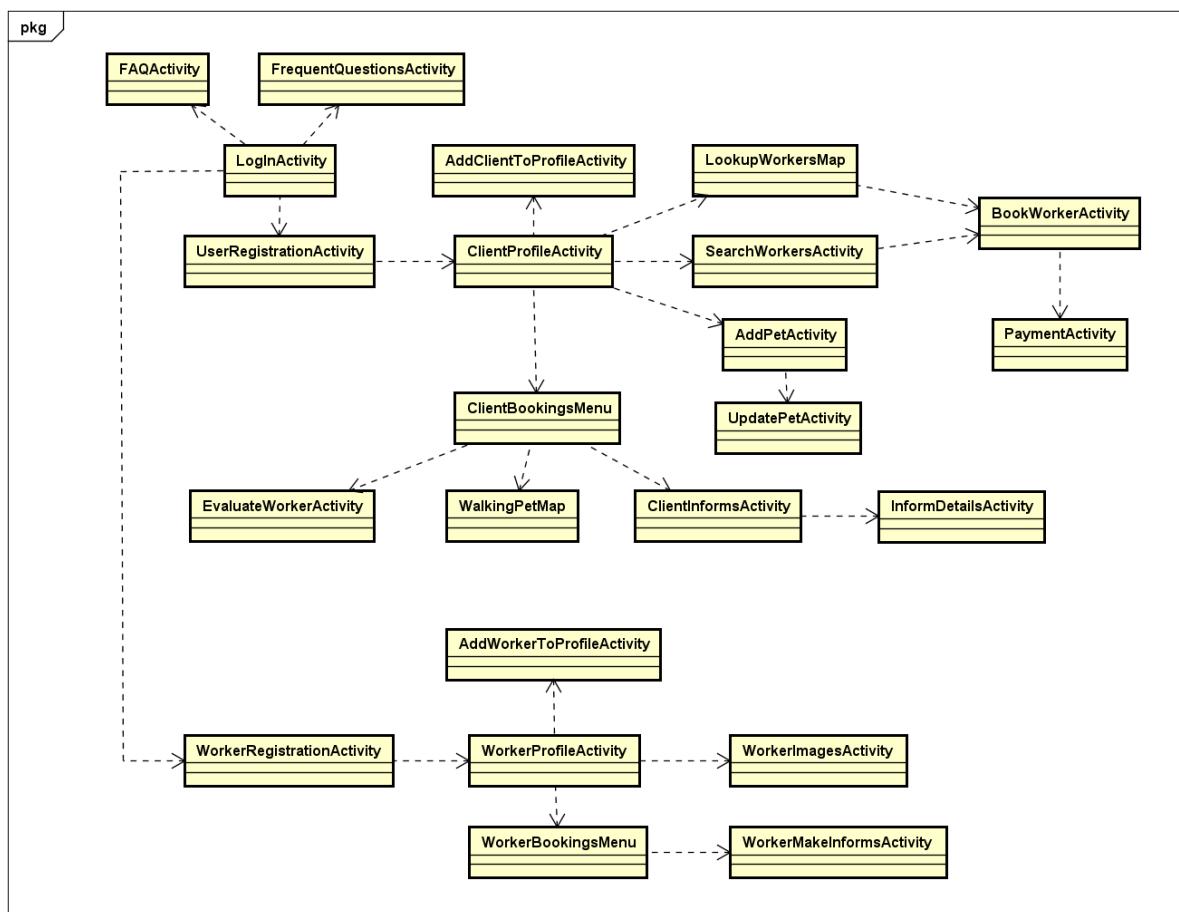
7. Arquitectura y diseño

A continuación, se detalla un primer acercamiento sobre el diseño y la arquitectura de nuestro sistema, estos últimos no son definitivos y muy probablemente cambien durante la ejecución de los *sprints* al ingresar en la etapa de desarrollo una vez cumplido con el primer entregable.

7.1 Diagramas UML

7.1.1 Diagrama de clases del front-end

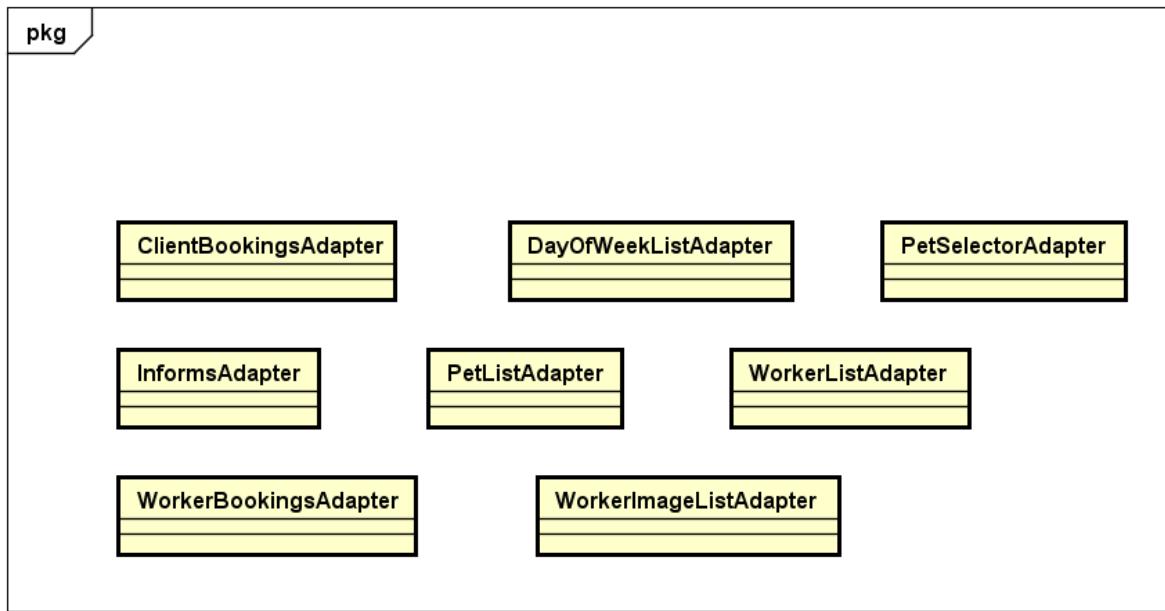
7.1.1.1 com.example.ximenamoure.petvacay



Este paquete es responsable de encapsular todas las *activities* del sistema, cada una es responsable únicamente de cargar su vista y llevar un control de las operaciones internas de las mismas y responder a los eventos del sistema cuando el usuario interactúa con la interfaz de las mismas.

Para simplificar el diagrama se muestran solo las dependencias del flujo de uso de la aplicación y no las que surgen con el *Navigation Drawer* así como también las principales clases de la solución.

7.1.1.2 com.example.ximenamoure.petvacay.Adapters

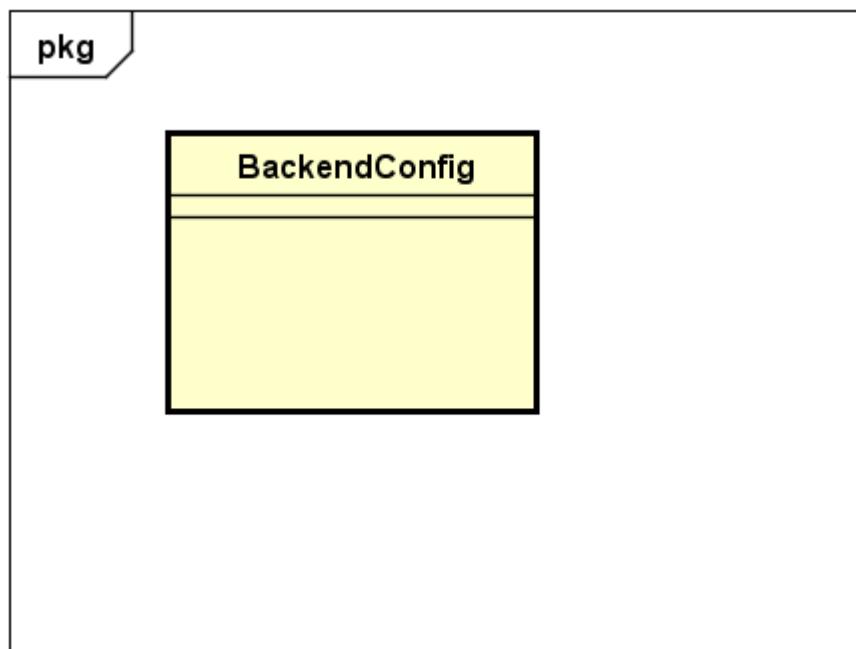


Este paquete tiene como responsabilidad agrupar todos los adaptadores utilizados en el sistema cuya única función de estos es actuar como puente entre los datos de una vista y una AdapterView, creando una vista para cada ítem de la colección conformada por el modelo de datos de la lógica de negocio.

Esto es útil para definir listas con ítems de nuestra lógica de negocio de forma dinámica contando a su vez con una interfaz personalizada.

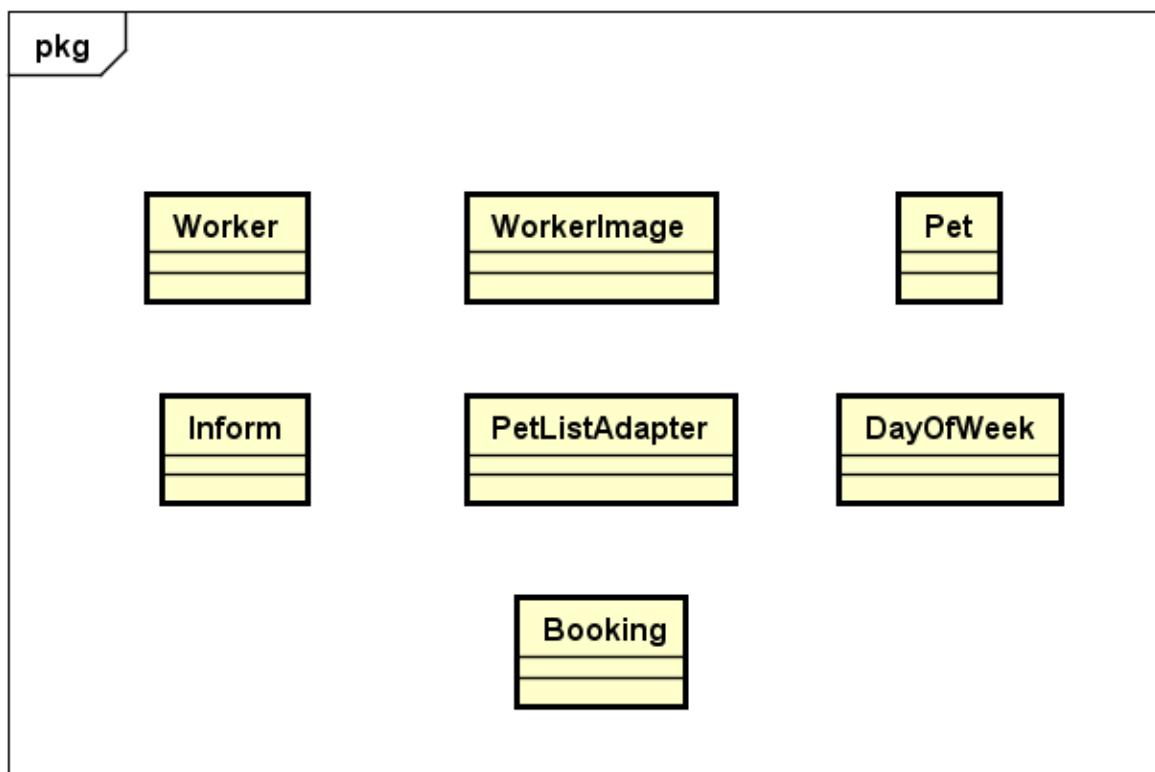
Cada adaptador tiene una dependencia con su modelo requerido en el paquete *com.example.ximenamoure.petvacay.Models*.

7.1.1.3 com.example.ximenamoure.petvacay.BackendConfig



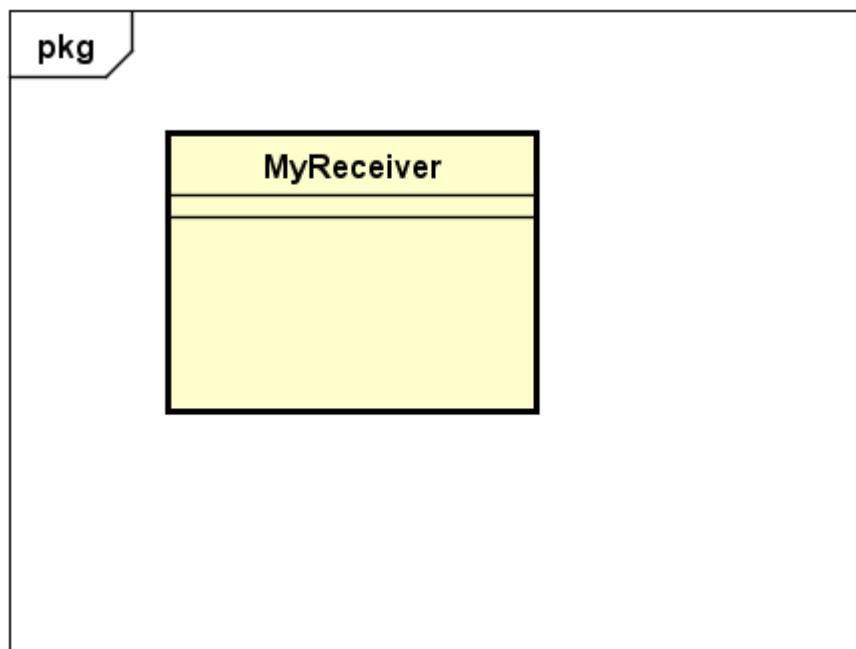
Este paquete sirve para guardar la ruta IP del servidor *backend*, como en nuestro caso el servidor no está *hosteado* en ningún servicio de internet, sino que lo está localmente en una máquina, actúa como archivo de configuración al momento de ejecutar la aplicación.

7.1.1.5 com.example.ximenamoure.petvacay.Models



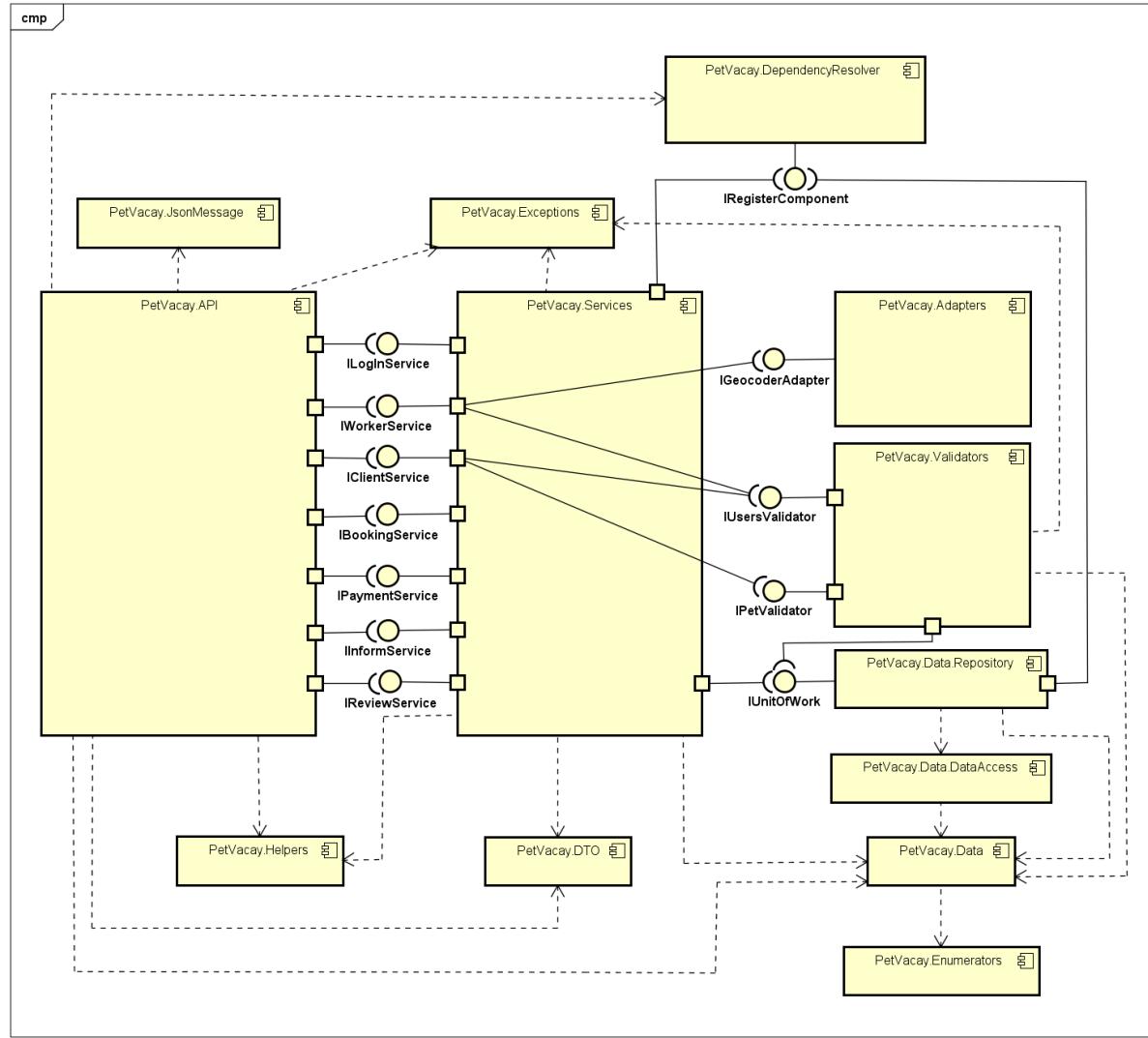
Este paquete tiene como responsabilidad agrupar todos los modelos de datos a usarse posteriormente en las vistas y ser utilizados por los adaptadores definidos anteriormente. Guardan información pertinente a los datos de la entidad que se desean mostrar en la vista.

7.1.1.6 *com.example.ximenamoure.petvacay.Notifications*



La única clase de este paquete *MyReceiver* es responsable de recibir por *broadcast* los intentos que lanza el servicio de alarma provisto por Android cada un cierto período de tiempo, a modo de mostrar una notificación en el celular del trabajador si este tiene alguna reserva que evaluar el día actual.

7.1.2 Diagrama de componentes



Los componentes principales de la solución del lado del servidor son:

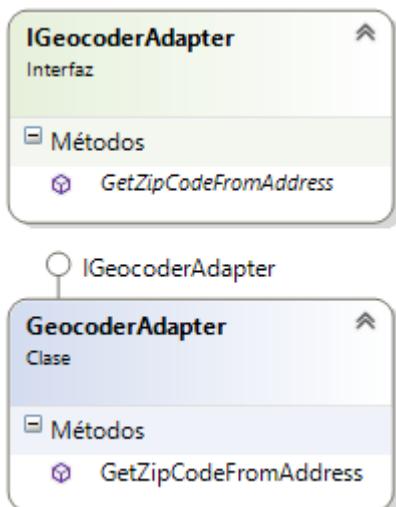
- **PetVacay.API** quien es el encargado de recibir las llamadas a la API invocadas por el cliente (aplicación en Android) y devolver una respuesta con código de error al mismo.
- **PetVacay.Services** quien realiza toda la lógica de negocio y a quien la API delega el procesamiento de sus solicitudes.
- **PetVacay.Data.Repository** quien es el encargado de gestionar todo lo referente al acceso a datos como inserciones, consultas, etc.

7.1.2.1 Mecanismos de comunicación

Todas las comunicaciones entre estos componentes son realizadas mediante interfaces las cuales son expuestas por cada componente y utilizadas por el otro. Cabe mencionar que, para permitir un mayor desacoplamiento, dichas interfaces son inyectadas en el constructor de las clases (inyección de dependencias) que las utilizan a modo de independizarse de su implementación, esto se explicará con más detalle en el apartado 7.2.3.

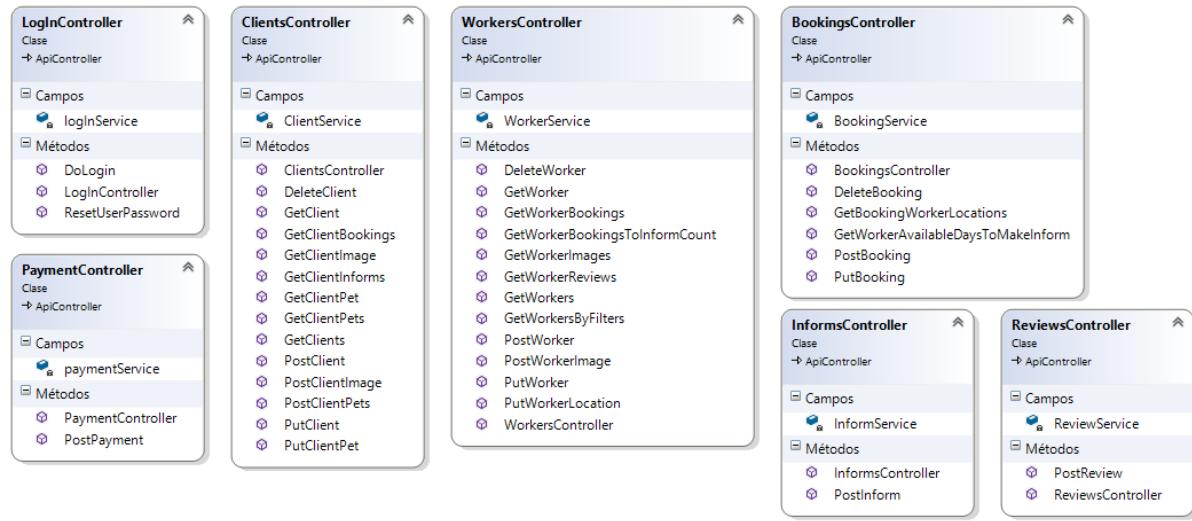
7.1.3 Diagrama de clases

7.1.3.1 PetVacay.Adapters



Este paquete contiene una interfaz que funciona como *Adapter* ya que es la encargada de procesar una solicitud que requiere de la respuesta de un tercero, en este caso la API de Google para el manejo de ubicaciones. Su implementación obtiene a partir de una dirección en formato *string*, el código postal o zip asociado a la misma, en caso de no existir, retorna 0.

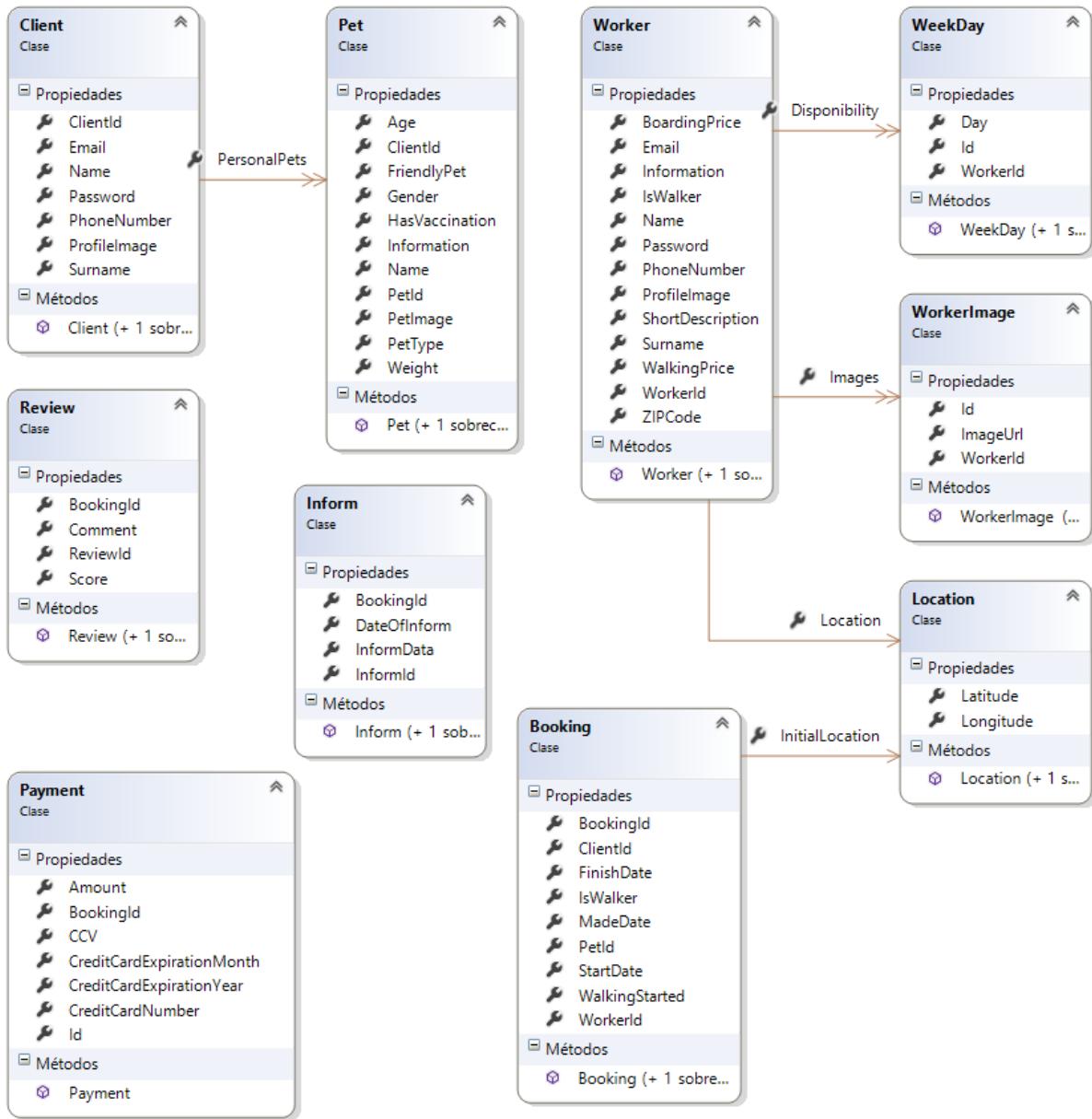
7.1.3.2 PetVacay.API



Este paquete contiene todos los *web services*, los *controllers* con sus *endpoints* requeridos para la comunicación y procesamiento de solicitudes por parte del cliente. Cada método define su *endpoint* y en caso de error al procesar la solicitud devuelve un mensaje acorde al tipo de error surgido. La misma no maneja lógica alguna, esta es delegada a la capa de servicios que se detallará en su respectivo diagrama.

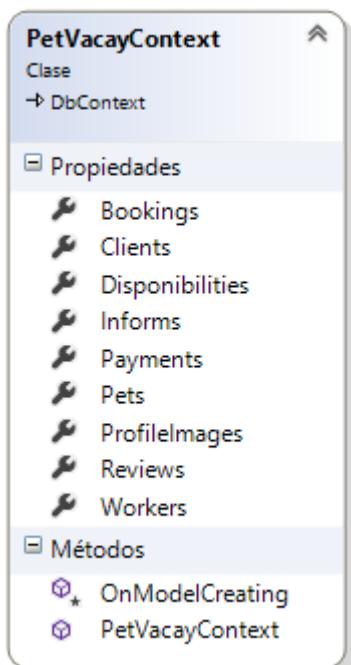
Los nombres de las clases son auto explicativos del rol de los *endpoints* a definirse en las mismas.

7.1.3.3 PetVacay.Data



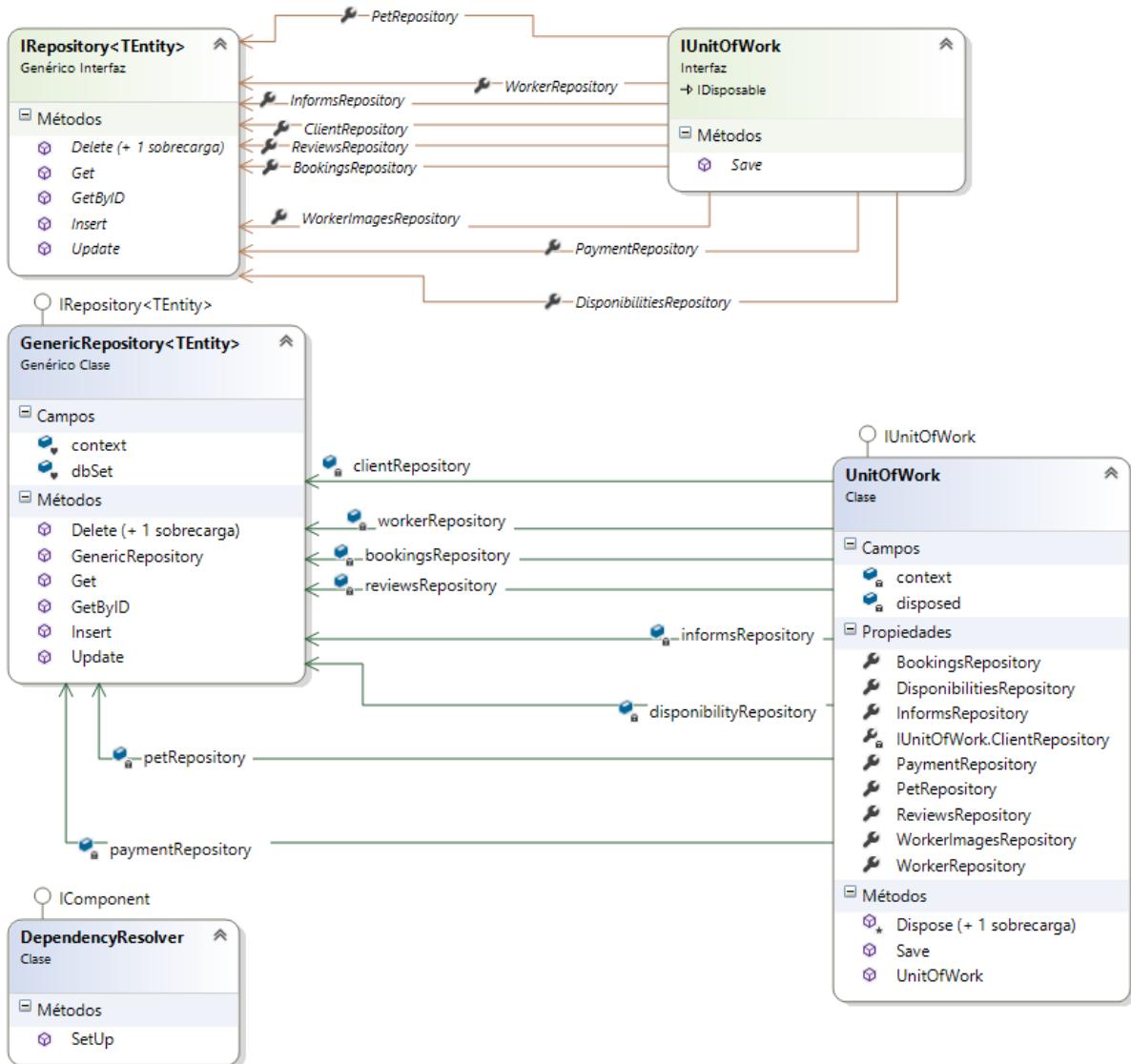
Este paquete tiene como responsabilidad agrupar todas las clases del dominio o entidades a persistirse en la base de datos y que forman parte del contexto del problema.

7.1.3.4 PetVacay.Data.DataAccess



Este paquete simplemente posee la clase que se encarga de mapear las entidades definidas en el paquete anterior a la base de datos, a través del uso de *Entity Framework* se define al estilo *code first* las relaciones entre las entidades, las propiedades de sus columnas y las tablas a tenerse en la base de datos.

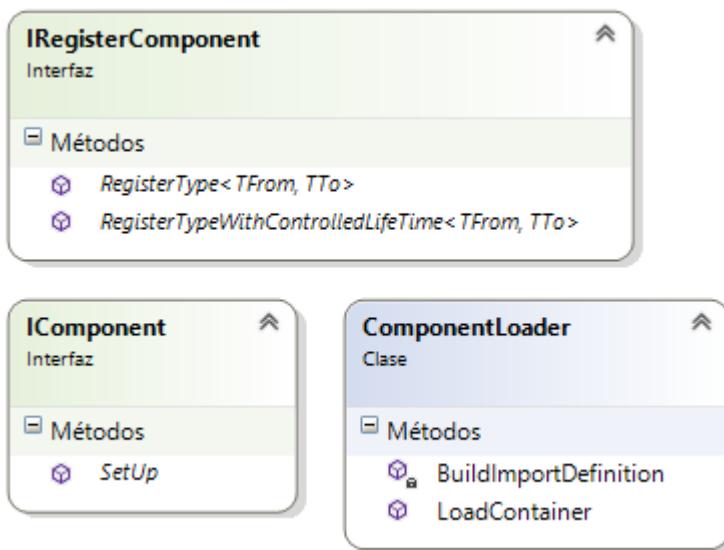
7.1.3.5 PetVacay.Repository



Este paquete por un lado abstracta el acceso a datos a través de una interfaz **IRepository** con los métodos CRUD de manera de permitir mayor independencia del tipo de base de datos a usarse.

Por otro lado, posee una **UnitOfWork** la cual se encarga de proveer todos los repositorios existentes, a modo de poder luego trabajar con ellos desde la capa de servicios ya sea haciendo búsquedas avanzada, inserciones, etc.

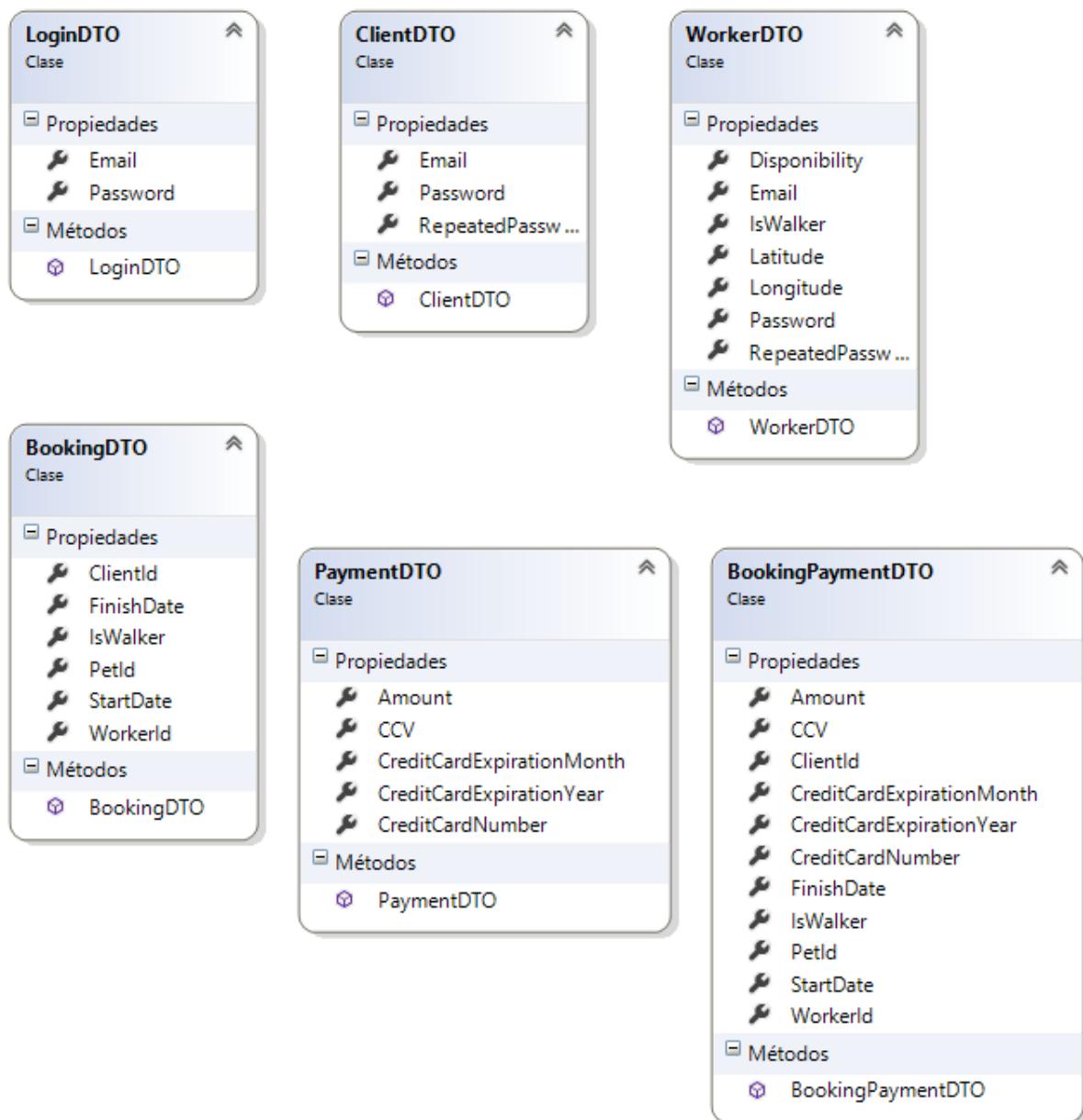
7.1.3.6 PetVacay.DependencyResolver



Este paquete se encarga de resolver las dependencias de cada paquete en el cual se realice una inyección de dependencias.

Simplemente se encarga de proveer la interfaz para registrar las dependencias en las cuales cada paquete registrara su interfaz con su implementación a modo de desacoplarse de la implementación en el paquete donde se haga la inyección.

7.1.3.7 PetVacay.DTO



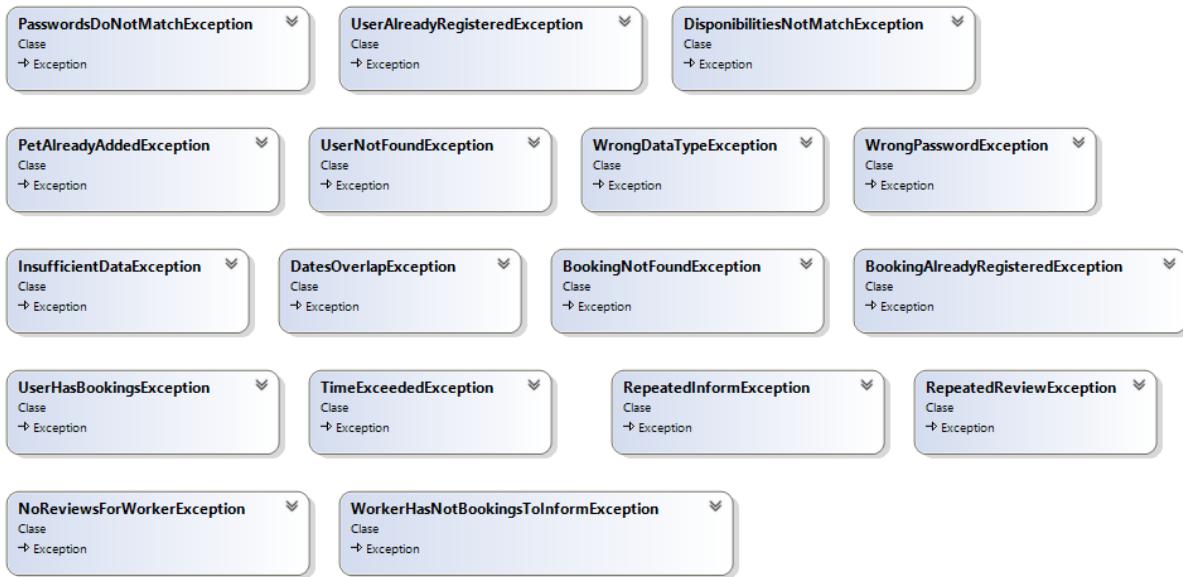
Este paquete agrupa todos los DTOs (Data Transfer Object) del sistema, a modo de evitar que las capas superiores conozcan las entidades del negocio permitiendo un menor impacto de cambio en el futuro.

7.1.3.8 PetVacay.Enumerators



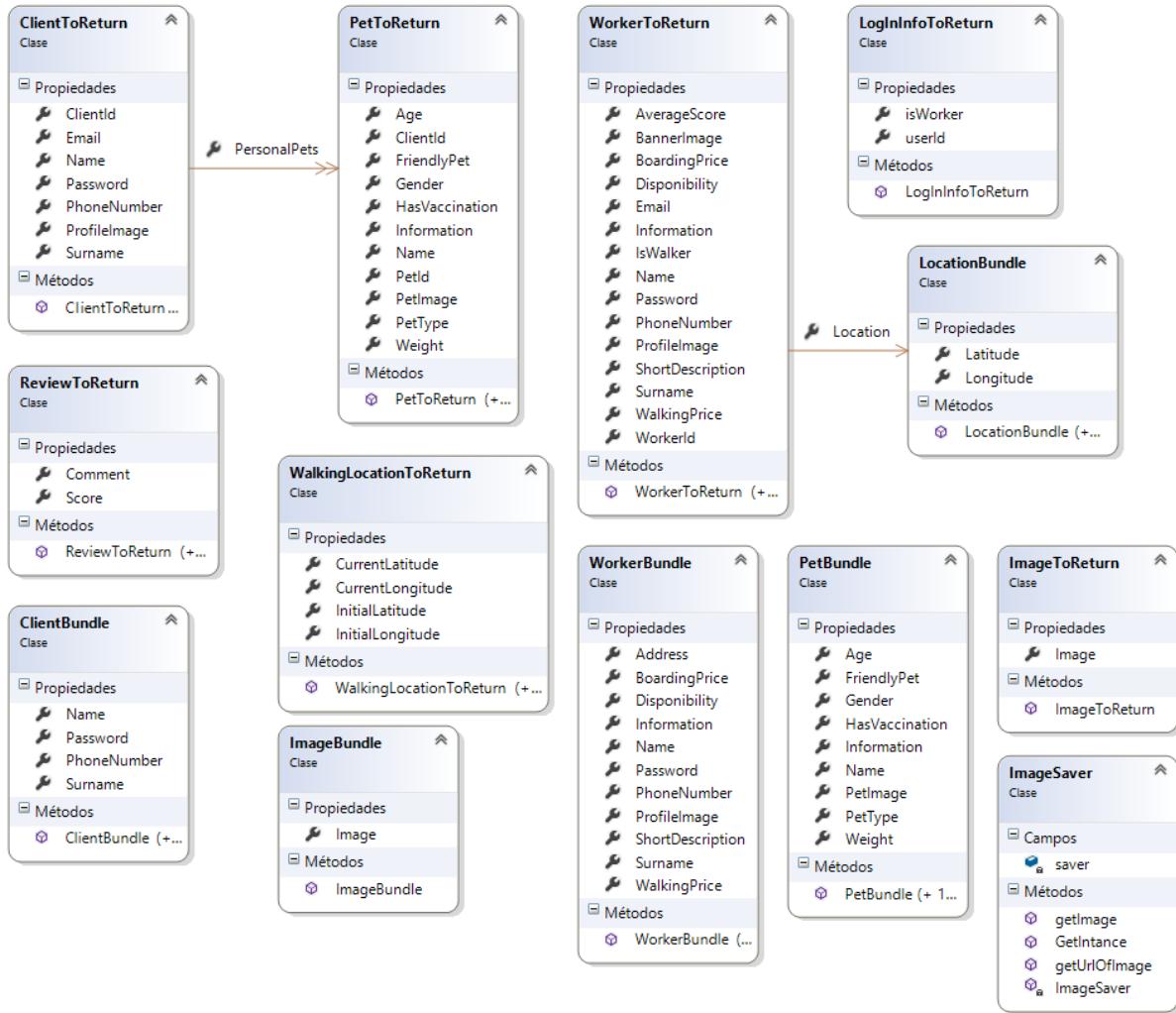
Paquete con el enumerador para los tipos de mascotas soportados actualmente por el sistema, fácilmente expandible.

7.1.3.9 PetVacay.Exceptions



Paquete con todas las excepciones propias creadas para una mejor comunicación entre las distintas capas y a su vez para proporcionarle al usuario final de la aplicación cliente mensajes de acuerdo al tipo de error ocurrido.

7.1.3.10 PetVacay.Helpers

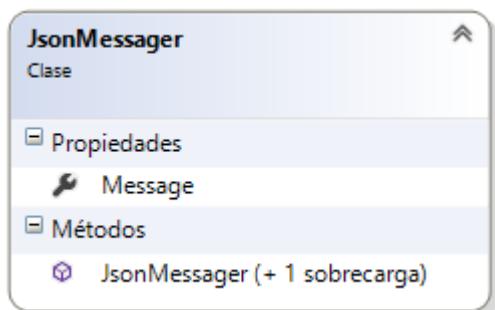


Este paquete se encarga por un lado de definir todas las clases que se devolverán al cliente a través de la API (las cuales varían de las alojadas en la base de datos, por ejemplo, entidades con imágenes guardadas como ruta del *filesystem*).

La clase **ImageSaver** en particular es la que se encarga del procesamiento de las imágenes, dada una dirección en el filesystem del servidor levantar dicha imagen y retornarla en formato de *array* de *bytes*, y a su vez también el proceso inverso, dada una imagen en el formato anterior, guardarla en el filesystem y devolver su ruta.

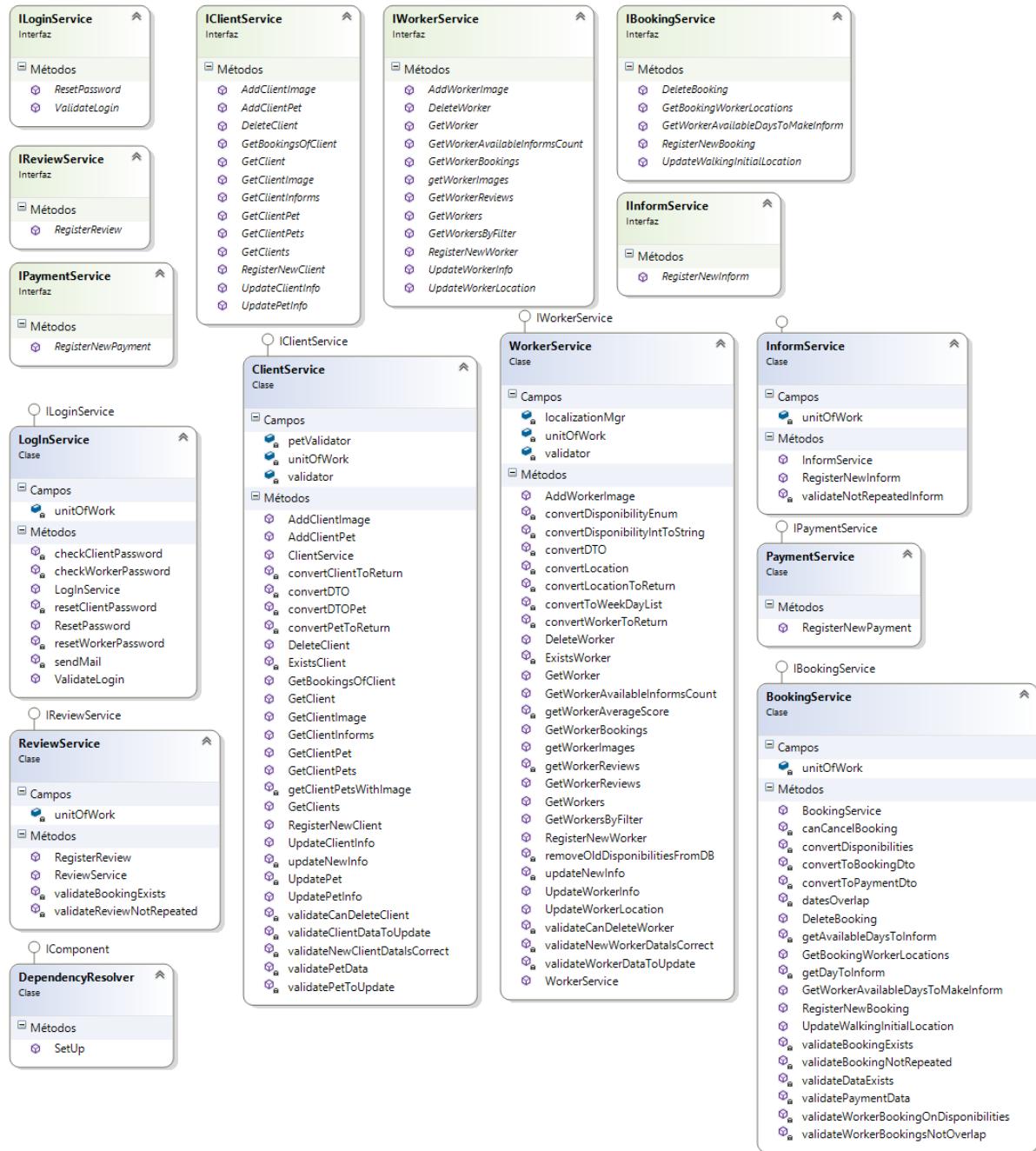
El resto de las clases de tipo *Bundle* sirven como *helper* para poder cumplir con el principio de Clean Code referente a la cantidad de parámetros que es deseable que reciba un método.

7.1.3.11 PetVacay.JsonMessage



La clase **JsonMessenger** de este paquete simplemente se encarga de guardar el mensaje en formato *string* o formato primitivo a un formato que pueda ser serializable a *json* para luego poder ser recibido desde la aplicación cliente.

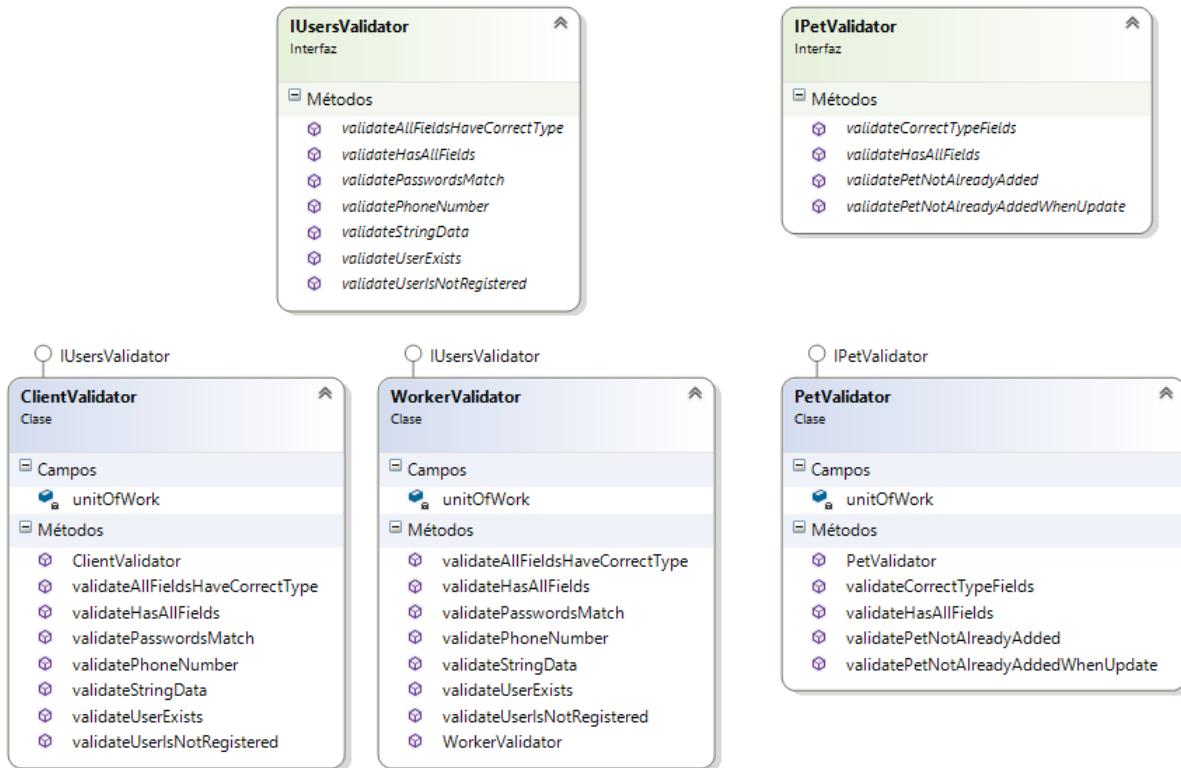
7.1.3.12 PetVacay.Services



Este paquete es uno de los más importantes de toda la solución del backend, el mismo se encarga del procesamiento de las solicitudes de los clientes y es quien agrupa toda la lógica de negocio.

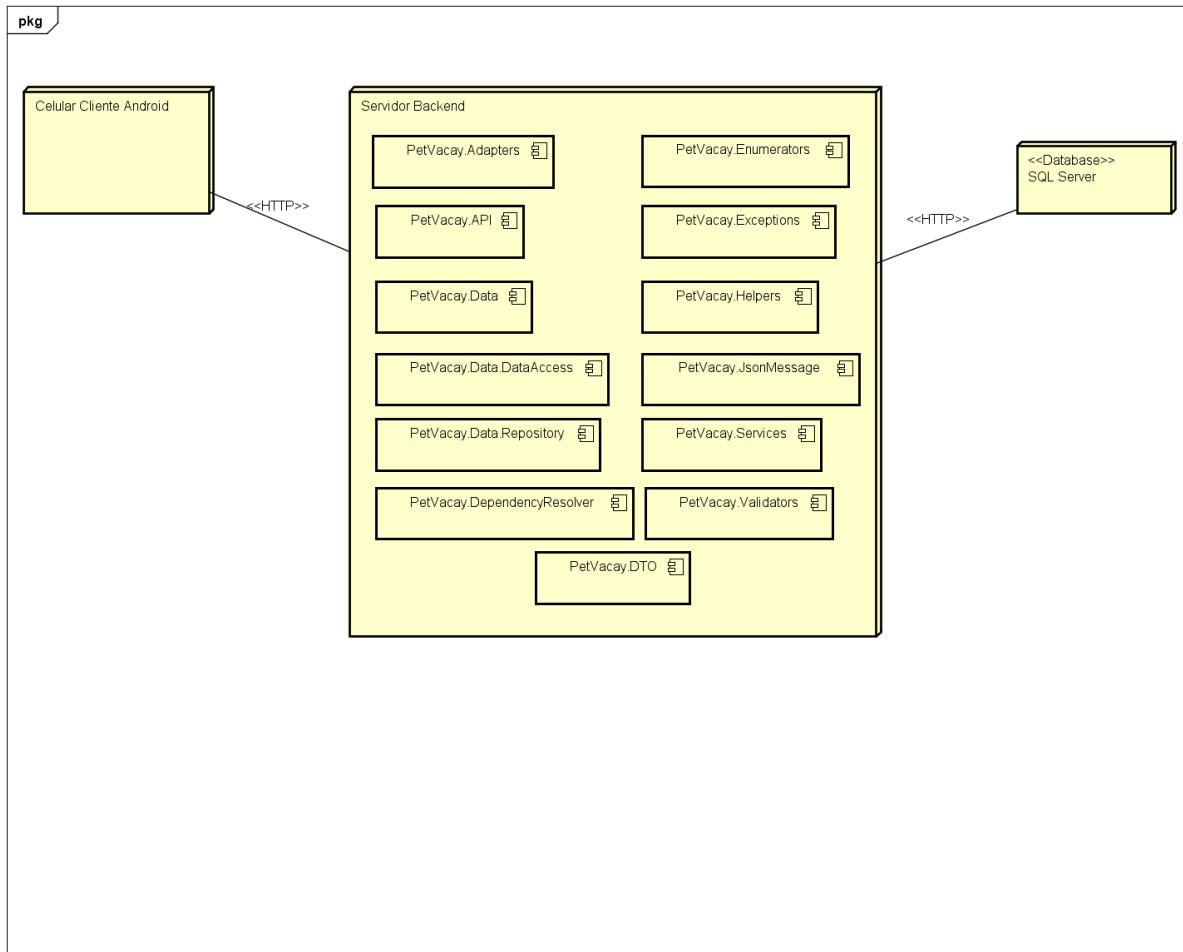
Obviamente que al ser una aplicación en capas, todo lo referido a acceso a datos es delegado a la capa encargada de la misma.

7.1.3.13 PetVacay.Validators



Este paquete contiene clases auxiliares que permiten validar todos los campos de los usuarios, sean clientes o trabajadores, y las mascotas, controlándose el formato de los mismos, el tipo de datos de los mismos, validación de contraseña, existencia de usuarios, etc.

7.1.4 Diagrama de despliegue

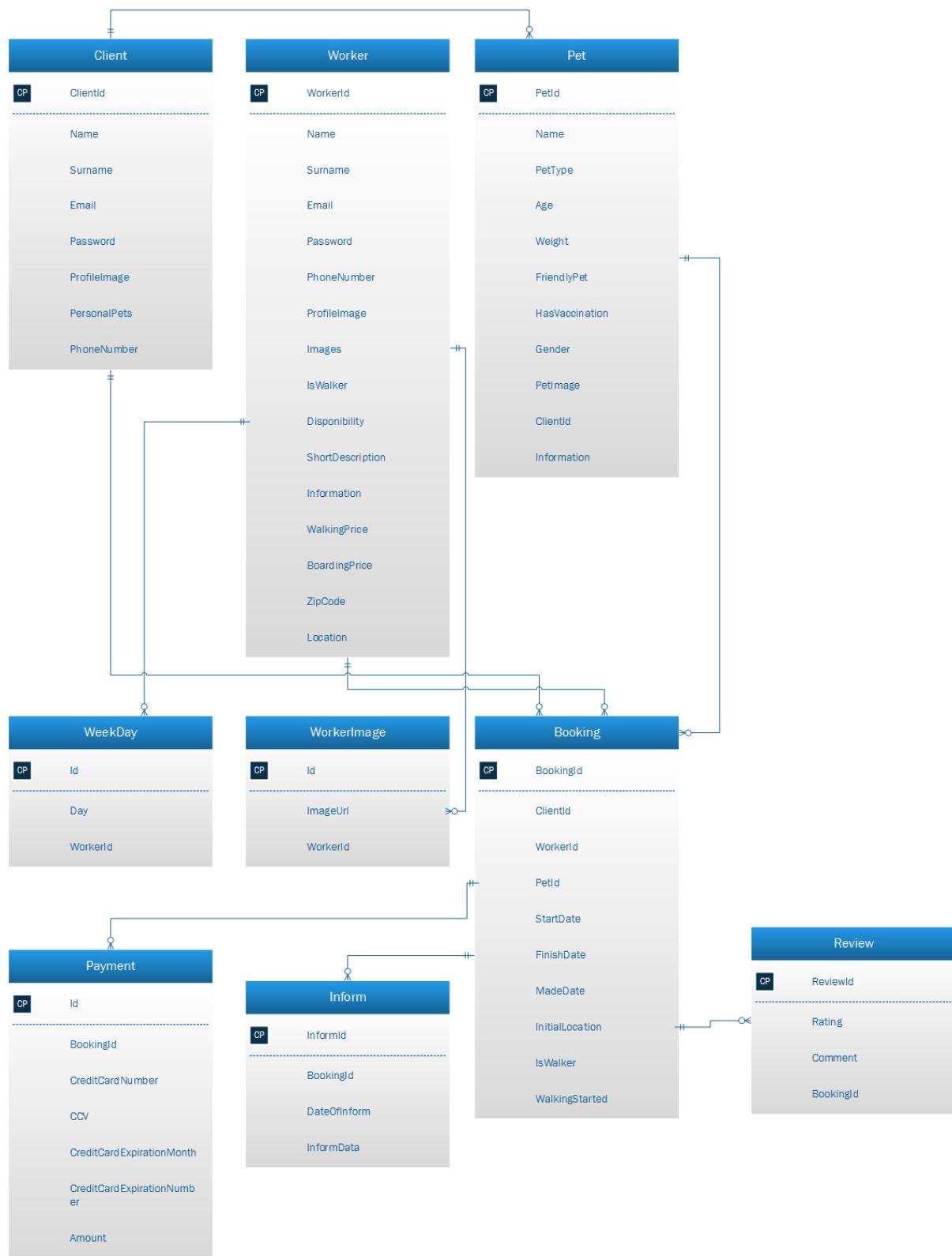


Dada nuestra arquitectura, la solución se puede desplegar en tres ambientes de ejecución distintos.

Por un lado, la aplicación móvil la cual correrá en cualquier celular con sistema operativo Android, quien se comunicará a través de llamadas HTTP al servidor el cual posee toda la lógica para poder contestar la solicitud del cliente.

Por último, la base de datos SQL Server puede estar en otro ambiente de ejecución y el servidor simplemente se comunicará con ella también a través del protocolo HTTP.

7.1.5 Modelo de tablas de base de datos



El modelo de base de datos, dado que se utilizó *Entity Framework* tiene un mapeo casi uno a uno con las entidades del dominio. En la imagen se pueden observar las relaciones entre cada tabla.

7.2 Justificaciones de diseño y arquitectura

A continuación, se detallan las principales decisiones tomadas por el equipo en cuanto al diseño y arquitectura de nuestra solución, listando los patrones arquitectónicos y de diseño utilizados a modo de atacar diversos atributos de calidad que el equipo consideró importantes y de desarrollar un software con buen código a modo de ser mantenible el día de mañana.

7.2.1 Arquitectura en capas - Patrón Layered

La aplicación consta de tres capas principales:

- **Capa de web-services**, quien se encarga de recibir las peticiones de los clientes y parsear las respuestas a devolverse a formato *json*.
- **Capa de servicios**, la cual se encarga de toda la lógica de negocio de la aplicación.
- **Capa de acceso a datos**, como su nombre lo indica, esta capa se encarga de gestionar y manejar los accesos a la base de datos como inserciones, consultas, modificaciones y borrado.

Este tipo de arquitectura permite una mayor respuesta antes los cambios minimizando el impacto de los mismos, ataca el atributo de calidad de la modificabilidad.

7.2.2 Patrón Unit of Work

Este patrón consiste en evitar la duplicación de código previendo de una estructura genérica con métodos CRUD los cuales independientemente del tipo de entidad a guardarse su lógica es la misma.

Por otro lado, se implementa una *Unit of Work* cuya función es agrupar todos los repositorios que conforman la capa de datos permitiendo operar bajo el mismo contexto de base de datos, en nuestro caso *Entity Framework*.

Este patrón aumenta el reuso, la modificabilidad y evita la duplicación de código para funcionalidades idénticas donde se puede abstraer el tipo de dato a manejarse.

7.2.3 Inyección de dependencias

Con el fin de desacoplar y construir un sistema fácilmente modifiable, creemos que este mecanismo es muy importante y ayuda a lograr dicho propósito.

En la capa de *web services*, servicios y acceso a datos se ha implementado inyección de dependencias, a los *web services* se le inyecta la dependencia a la capa de servicios a través de una varias interfaces dependiendo el tipo de *web service*, y a los servicios se le inyecta la dependencia con la capa de acceso a datos a través de la interfaz de la *Unit of Work*.

Por último, se ha dejado la posibilidad a futuro de poder también cambiar de contexto de base de datos, permitiendo cambiar el tipo de implementación del mismo en el constructor de la *Unit of Work*.

7.2.4 Patrón Adapter

Como se mencionó anteriormente, se aplicó el patrón *Adapter* para la obtención del código postal o zip a partir de una dirección en formato *string*, esto lo hace consumiendo un servicio web provisto por la API de Google.

Como es un servicio de terceros puede cambiar a futuro, se quiera o no, no es algo que el equipo de desarrollo pueda prever o esté al alcance del mismo, por lo que se decidió implementar un mecanismo de forma que el impacto de cambio sea el mínimo posible, cambiando únicamente la clase que implementa la interfaz del adaptador.

7.2.5 Patrón Singleton

Hay clases como por ejemplo *ImageSaver* las cuales nos interesa tener una sola instancia de la misma, ya sea porque guarda en estado global a todo el sistema, o simplemente porque es una clase que procesa datos independientemente del estado en que se encuentre la misma.

7.2.6 Uso de Data Transfer Objects (DTOs)

Para el manejo de comunicación entre las distintas capas se utilizan DTOs que sirven para evitar conocer entidades del dominio y también para evitar pasar datos que quizás no le interesan al cliente, algunos *web services* de la API reciben un DTO que luego se convierte y se guarda en la base de datos como su entidad normal.

7.2.7 Arquitectura Restful

El diseño de una API REST da muchas ventajas con respecto a otro tipo de implementaciones de las mismas.

Como elementos que caracterizan a una API REST se tienen los siguientes:

- **Protocolo cliente-servidor sin estado**, cada mensaje HTTP contiene toda la información requerida por ambas partes, esto es fácil de observar en nuestra solución ya que ninguno de los *web-services* guarda estado.
- **Cuatro operaciones bien definidas**, las cuales son POST, PUT, GET y DELETE, esto también se observa en nuestra solución si se fija en las *annotations* de cada método de los *web services* ([*HttpGet*], [*HttpPost*], [*HttpPut*], [*HttpDelete*]), se trató en lo posible de implementar esas cuatro operaciones en cada uno.
- **Manejo de una URI estandarizada para identificar los recursos**, en nuestros *web services* cada URI respeta la convención establecida: /api/recurso/subrecurso/...

Como ventaja de implementar una API REST se tiene que es mucho más escalable que las otras versiones de APIs ya que existe una clara separación entre cliente y servidor.

Por otro lado, se tiene que la API REST es independiente al lenguaje o plataforma en la cual se esté programando, lo cual ofrece una mayor libertad a la hora de programar y a su vez aumenta la modificabilidad si se quiere cambiar de lenguaje o plataforma por algún motivo.

7.2.8 Patrón Strategy

Para poder utilizar distintas implementaciones de métodos, el uso de este patrón se hace muy presente en el módulo de las validaciones (**PetVacayValidators**) donde una interfaz provee los métodos para validar usuarios y luego se tienen implementaciones distintas si es para un cliente o si es para un trabajador.

7.2.9 Análisis de principios SOLID

7.2.9.1 Single Responsibility

Cada módulo de nuestra aplicación tiene una sola responsabilidad en concreto, como se detalló anteriormente en los diagramas de clase, logrando de esta manera tener una alta cohesión en los mismos, por lo tanto, se cumple con este principio.

7.2.9.2 Open/Closed

De la manera con la que se trabajó, el uso de interfaces, la respuesta al cambio y un mínimo impacto en la modificación de algún módulo, se observa que el sistema es abierto a la extensión permitiendo agregar nuevas funcionalidades o distintas implementaciones de las mismas, y se trata a su vez de ser cerrado a la modificación del código ya existente.

7.2.9.3 Liskov Sustitution

En nuestra solución no utilizamos herencia, por lo cual, este principio no aplica evaluarlo.

7.2.9.4 Interface Segregation

Cada interfaz provista por los distintos componentes tiene las operaciones mínimas que requiere el que las necesita utilizar, por lo cual tanto si surge una nueva implementación como si se utiliza dichas interfaces, todos los métodos serán utilizados y en caso de ser implementadas, todos los métodos serán importantes para la clase que los implemente. Por ende, cumple con este principio.

7.2.9.5 Dependency Inversion

Este principio establece que clases de alto nivel no deben depender de clases de bajo nivel ni viceversa, sino de abstracciones.

Esto se nota especialmente cuando se observa la comunicación entre los distintos componentes, cada uno se comunica con una interfaz provista por el otro independizándose de su implementación y de esta manera dependiendo de abstracciones.

Esto también se observa en la vista de módulos donde todas las dependencias a otros se realizan a través de una interfaz y no de la clase concreta.

8. Gestión de Riesgos

En cualquier proyecto de software es esencial la gestión de riesgos. La misma sirve para asegurar que ningún evento pueda impedir que se cumplan con los objetivos del proyecto.

8.1 Identificación de riesgos

Para poder evaluar los riesgos se decidió ponderar los mismos según su impacto y probabilidad de ocurrencia. Con dicha ponderación se decide cuáles son los riesgos a los cuales se les debe prestar mayor atención y controlar de forma continua, así como también cuales son críticos para el proyecto en caso de suceder.

Luego de ponderar los riesgos se elaboró un plan de contingencia y un plan de mitigación para cada uno. Por plan de mitigación nos referimos a acciones a realizar para prevenir que el riesgo se haga efectivo y por plan de contingencia nos referimos a acciones a realizar en caso que el riesgo se haga efectivo.

8.2 Planilla de riegos

Id	Factor de riesgo	Impacto (I)	Probabilidad de ocurrencia (P)	Magnitud M= I x P	Plan de contingencia		Plan de mitigación
1	Experiencia desarrollando en Android nula	4	0.6	2,4	Mayor dedicación. Consultar a expertos. Realización de cursos online y tutoriales sobre el tema.	Trabajo en equipo, apoyo entre los integrantes del mismo. Realización de tutoriales. Realización de prototipos	
2	Errores en estimación de las tareas	3	0.6	1,8	Realizar los ajustes necesarios en el siguiente sprint.	Utilizar <i>poker planning</i> o alternativas conocidas.	
3	Gestión de proyecto - Poca experiencia	4	0,4	1,6		Consultar regularmente con el profesor.	
4	Pérdida de información del repositorio	5	0,2	1	Cambiar de repositorio	Realizar respaldos en otro repositorio	
5	Otra empresa saca una aplicación que cumple las mismas características	5	0,2	1	Evaluar los servicios que brinda la aplicación e intentar hacerla más atractiva para atraer a los usuarios que utilicen la aplicación de la competencia	Estar atentos al mercado constantemente, evaluando la posibilidad de que otra empresa implemente la idea. Hablar con las empresas para buscar colaboración y así no tenerlos como competidores	
6	Fracaso total del proyecto	5	0,2	1	Tercerizar	Buscar promocionar el producto, re evaluar costos de manera que se pueda continuar con el proyecto	

Escala utilizada para medir el impacto:

Impacto:
0 - No tiene impacto en el proyecto
1 - Marginal
2 - Poco importante
3 - Importante (puede retrasar el proyecto)
4 - Crítica (pude detener el proyecto)
5 - Catastrófica (fracaso del proyecto)

Escala utilizada para medir la probabilidad de ocurrencia:

Probabilidad de ocurrencia
0.0 - no probable
0,2 - poco probable
0,4 - probable
0,6 - muy probable
0,8 - altamente probable
1,0 - se convierte en un problema

8.3 Seguimiento

Con el objetivo de lograr que los riesgos más críticos al proyecto no ocurrieran o fueran mitigados a tiempo se decidió monitorearlos de forma constante.

Al término de cada *sprint*, se analiza la situación de cada riesgo y se vuelve a calcular su impacto.

8.3.1 Control de riesgos durante el Sprint 1

Durante este Sprint se prestó mayor atención al riesgo “Experiencia desarrollando en Android nula”. El mismo fue monitoreado a lo largo de todo el Sprint y se aplicaron los siguientes planes de contingencia: Mayor dedicación, Consultar a expertos y realizar tutoriales online.

Igualmente, el riesgo se hizo presente ya que por la falta de experiencia desarrollando en Android, este Sprint llevo más tiempo de lo estimado. Es por esto que se decidió aplicar los planes de mitigación. Como medida principal se decidió realizar más trabajo en equipo y dedicarle más horas de trabajo.

Otro riesgo que se hizo presente fue el llamado “Errores en estimación de las tareas”. El mismo se debió principalmente a la ocurrencia del riesgo mencionado en el párrafo anterior. Es por esto que se decidió no volver a estimar las tareas, sino que se decidió dejar el calendario como fue estimado en un principio y simplemente dedicarle más horas y realizar más trabajo en equipo.

En los siguientes Sprints como ya teníamos más experiencia desarrollando en Android pudimos cumplir con el calendario.

Los demás riesgos fueron monitoreados, pero no se hicieron presentes por lo que no tuvimos que aplicar ningún plan de contingencia.

8.3.2 Control de riesgos durante los Sprints 2, Sprint 3 y Sprint 4

Podemos concluir que el riesgo que más se hizo presente fue el riesgo “Experiencia desarrollando en Android nula”. Es por esto que se aplicaron los planes de contingencia y mitigación descriptos anteriormente.

9. Gestión de configuración

9.1 Gestión de código fuente

9.1.1 Control de versionado

Para el manejo de versiones se utilizará la herramienta GIT ya que, dada la complejidad de la aplicación a desarrollar, creemos que es importante apoyarnos en un sistema que nos permita manejar versiones de nuestro código y a su vez poder dividir el trabajo en distintas *branches* con el objetivo de aumentar la eficiencia del equipo y como ayuda para nosotros mismos para organizar mejor nuestro trabajo.

Como repositorio online se optó por utilizar Bitbucket^[8] debido a que es el único conocido que soporta múltiples repositorios privados en una cuenta de manera gratuita. Consideramos también GitHub^[9] por nuestra experiencia en trabajos anteriores, pero dado que este último no ofrece repositorios privados de manera gratuita, decidimos utilizar Bitbucket.

La estructura del mismo constará inicialmente de dos *branches*:

- **master** donde se hará el *commit* inicial y el *commit* final con el producto terminado, así como también la documentación relativa al mismo y los artefactos necesarios para su puesta en producción.
- **develop** donde se desarrollará la aplicación, al finalizar su desarrollo se hará un *merge* con la rama **master**.

Durante el desarrollo podrán surgir *branches* auxiliares sobre **develop** en caso de dividirse las funcionalidades a implementarse entre los miembros del equipo.

9.1.2 Evidencia de versionado

Como se dijo anteriormente, se trabajó en dos ramas: **master** y **develop**, a continuación, se dejan las direcciones de los repositorios en BitBucket junto con la evidencia del uso del mismo:

9.1.2.1 Repositorio “Ingeniería de software en la práctica”

URL: <https://bitbucket.org/xmoure/ingenieria-de-software-en-la-practica>

Evidencia:

ximena	22a64a0	mejoras ui	v develop	22 minutes ago
ximena	5d5b52d	mejoras ui	v develop	4 hours ago
ximena	425e5cc	mejora ui	v develop	5 hours ago
ximena	7cd108b	cambios ui	v develop	5 hours ago
ismael p	8d1b7f6	Arreglos en la UI	v develop	14 hours ago
Juan Rodríguez	9c1f8cc	Terminada Historia de Usuario 20	v develop	3 days ago
Juan Rodríguez	d3139b9	Comienzo Historia de usuario 20	v develop	3 days ago
ismael p	9967bb7	Notificación de informes de las reservas.	v develop	3 days ago
ismael p	8461df3	Baja de usuarios y reseteo de contraseña enviandole correo.	v develop	4 days ago
ismael p	fd37f4e	Arreglado lo de hoy con dia actual para los informes y validacion del dia...	v develop	5 days ago
ismael p	61eee02	Corregido error cuando seteaba datos	v develop	2017-06-14
ismael p	0d66d62	Calificación y comentarios de trabajadores.	v develop	2017-06-14

9.1.2.2 Repositorio “Ingeniería de software en la practica - BACKEND”

URL: <https://bitbucket.org/xmoure/ingenieria-de-software-en-la-practica-backend>

Evidencia:

Juan Rodríguez	220459d	Pruebas Worker	v develop	2 hours ago
ismael p	e6b5efa	Pruebas	v develop	3 hours ago
Juan Rodríguez	6a221ce	Ultimos detalles	v develop	23 hours ago
Juan Rodríguez	9960165	Últimas revisiones antes de entregar	v develop	2 days ago
ismael p	52fbfcf7	Backend para chequear si tiene reservas que informar este dia.	v develop	3 days ago
ismael p	aed9ffc	Cambio de contraseña mandando mail con la nueva y baja de usuarios...	v develop	4 days ago
ismael p	3705dd3	Arreglado lo de hoy con los informes que muestre el dia de hoy si esta ...	v develop	5 days ago
ismael p	209910f	Backend para las reviews, creacion, obtencion de las reviews de los trab...	v develop	2017-06-14
ximena	d9026df	corrección de un error	v develop	2017-06-13
ximena	6757b2d	cambios en la reserva	v develop	2017-06-13
ximena	afd6659	algunos cambios en la reserva para poder agregar el pago	v develop	2017-06-13
ismael p	c391658	Backend para las reservas terminado con registro de reserva, obtencio...	v develop	2017-06-12
ismael p	13f1900	Registro de informes de los trabajadores.	v develop	2017-06-12

9.2 Gestión de documentación

Para el repositorio de documentos se decidió utilizar Google Drive ya cuenta con Google Docs para el versionado de documentos y permite de forma simultánea a distintos usuarios modificar documentos. Una característica muy importante de Google Docs es que permite realizar comentarios sobre documentos.

Cabe destacar que también se analizó utilizar Dropbox pero debido a las características anteriormente mencionadas se descartó.

10. Gestión de la calidad

Con el objetivo de lograr un producto de calidad se consideró necesario definir distintas actividades que permitieran controlar que tanto el proceso como el producto cumplieran con los objetivos planteados.

Para esto, en primera instancia, se definió que entendemos por calidad y cuáles eran los objetivos en cuanto a calidad del producto a desarrollar, así como los objetivos del proceso.

10.1 Objetivos de calidad del producto

- Lograr un producto que sea fácil de usar para el usuario (cantidad de clicks, intuitiva).
- Lograr un producto atractivo para el usuario.
- Producto compatible con Android.

10.2 Objetivos de calidad del proceso

- Se busca lograr tener un proceso de desarrollo bien definido con el fin de poder recolectar métricas que permitan realizar un mayor seguimiento del proyecto. Las mismas también permiten tomar acciones correctivas a tiempo por lo que se logra una mejora continua.
- Tener control del proyecto durante todas sus fases.
- Evitar re trabajo

10.3 Aseguramiento de la calidad

10.3.1 Calidad del proceso

10.3.1.1 Spring retrospective meeting

Ceremonia realizada al final de cada sprint. La misma consiste en que todos los integrantes del equipo se reúnen y evalúan el sprint que ha finalizado y determinan que se puede cambiar para lograr que el siguiente sprint sea más productivo.

Esta ceremonia se diferencia del sprint review ya que en esta última se analiza lo que el equipo está construyendo mientras que en la retrospective meeting se analiza como el equipo lo está construyendo. Uno de los objetivos de esta ceremonia es realizar mejoras al proceso para que el equipo pueda mejorar la forma en la que trabaja.

Es un mecanismo importante para el equipo ya que le permite continuamente evolucionar y mejorar a lo largo de todo el proyecto.

10.3.2 Uso de estándares de codificación

Con el fin de obtener código que se adapte a los principios básicos de diseño y que sea limpio se definieron estándares a seguir en cuanto a la codificación del mismo.

- El código será escrito totalmente en idioma inglés. Esto incluye comentarios.
- El nombre de las clases debe empezar siempre con mayúscula, es decir, se debe utilizar *UpperCamelCase*.
- Los métodos deben comenzar con minúscula. Se debe utilizar *lowerCamelCase*.
- En las clases primero se deben declarar los atributos, luego los constructores y por último los métodos de la misma.
- Se debe dejar una línea por medio entre cada método y la declaración de los atributos y métodos.
- Usar *try-catch-finally* para el manejo de excepciones.
- Nombre de los métodos debe ser auto explicativo.
- Métodos pequeños que cumplan una sola función.
- Seguir las buenas prácticas de cada lenguaje en particular.
- Se deben seguir los principios de diseño básicos como Abierto-Cerrado y

Responsabilidad única.

Reglas específicas para Android

- Los nombres de las *activities* deben terminar con la palabra *Activity*.
- Los nombres de los *fragments* deben terminar con la palabra *Fragment*

10.3.3 Uso de estándares de documentación

Se definieron los estándares de documentación 302, 303, 306 publicados por Universidad ORT Uruguay.

10.3.4 Actividades de calidad

Validaciones

El proceso de validación consiste en asegurar que lo que se está construyendo se corresponde con el uso que se le quiere dar al producto.

Al finalizar cada *sprint* el equipo se reunirá y evaluará si esto se está cumpliendo. A su vez cuando el equipo lo considere necesario se podrá evaluar con distintos usuarios mostrándoles prototipos o demos de las funcionalidades desarrolladas.

Producto a evaluar	Técnica	Registro de resultados	Cuando se realiza
Product Backlog	Validación con usuarios finales y profesor del curso	Informe con la validación.	Periódicamente
Producto de software	Validación con usuarios finales	Informe de resultados.	Al tener implementado al menos un 50% de los requerimientos y al finalizar el proyecto antes de la entrega final.

Revisiones

Las revisiones permiten obtener defectos, desviaciones y retroalimentación sobre el software que se está desarrollando para así poder aplicar en caso de ser necesario acciones correctivas.

A lo largo del desarrollo se realizarán actividades de revisión entre pares de código y de documentación.

Producto a evaluar	Técnica	Registro de resultados	Cuando se realiza
Código fuente	Revisión de a pares	Comentarios en el <i>board</i> en Trello	Al finalizar una funcionalidad. Al finalizar cada sprint.
Gestión de riesgos	Revisión de los documentos	Comentarios en el documento	Al finalizar cada sprint

Verificaciones

Consiste en verificar que cada entregable de software cumple con los requerimientos especificados.

Producto a evaluar	Técnica	Registro de resultados	Cuando se realiza
Producto de software	Evaluación del funcionamiento y completitud de las funcionalidades.	Issues en BitBucket o tareas en el <i>board</i> en Trello	Al finalizar cada sprint.
Producto de software	Pruebas unitarias	Tareas en el <i>board</i> en Trello	Al realizar cada funcionalidad.

Pair-Programming

Para obtener un mejor código y con menos defectos se realizará una práctica conocida como programación en pareja (Pair-Programming).

La misma no se realizará a lo largo de todo el desarrollo. Se intentará implementar cuando algún miembro del equipo lo necesite porque está teniendo dificultades desarrollando alguna funcionalidad. También se implementará si el equipo lo considera necesario para el desarrollo de alguna funcionalidad.

Siempre que dos o más integrantes del equipo se puedan reunir en un mismo lugar físico se intentara aplicar esta práctica.

10.3.5 Principales actividades

Practica	Desarrollo	Objetivos/fundamento
Ceremonias de Scrum	Cumplir con las diversas pautas y ceremonias de scrum.	Para la correcta utilización de SCRUM se realizarán las siguientes ceremonias: -Sprint retrospective - Daily meeting -Sprint review -Sprint planning
Pair programming	Escribir el código de a pares en una misma máquina	Mejorar la calidad del código, tener mayor disciplina.
Refactor	Contar con tiempo para analizar el código y realizar mejoras	Obtener código que cumpla con los estándares definidos y que siga los principios de diseño.

Code review	Generar instancias para la revisión de código.	<p>Obtener código que cumpla con los estándares y mejorar la calidad del mismo. También es una forma de aprender nuevas formas de realizar cierta funcionalidad al tener que buscar alternativas más eficientes.</p> <p>Ayuda a que el código sea más mantenible.</p>
Spikes	Realizar tareas de estudio e investigación.	<p>Obtener el conocimiento necesario para desarrollar las distintas funcionalidades. Sobre todo teniendo en cuenta el reto de desarrollar en Android el cual es un lenguaje nuevo para todo el equipo.</p>
Obtención y medición de métricas	Medir y documentar a lo largo del desarrollo las distintas métricas que se establecieron que eran importantes.	<p>Tener un mayor control sobre el proyecto. Tomar medidas correctivas.</p>

10.3.6 Pruebas unitarias

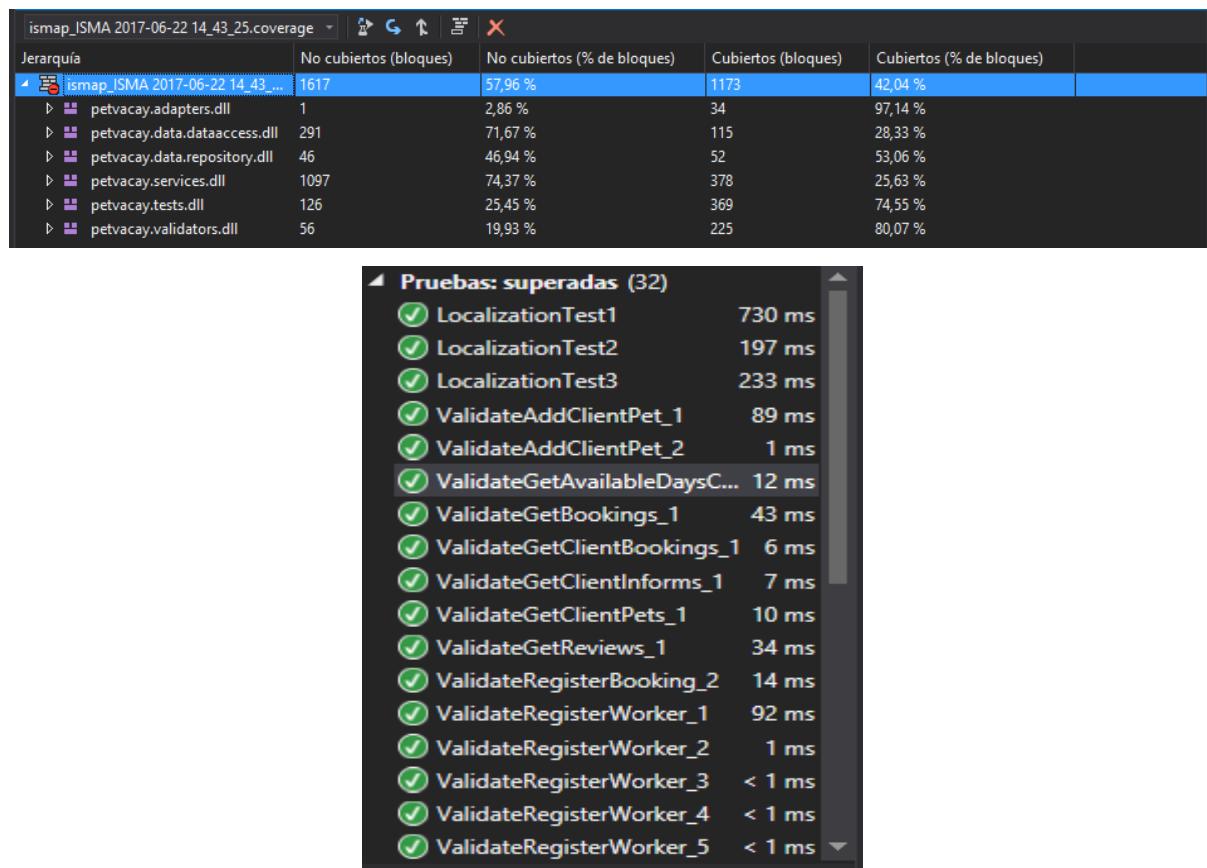
Cada desarrollador es responsable de codificar tanto las funcionalidades como sus pruebas unitarias.

Aclaración:

Se decidió a lo largo del desarrollo que lo mencionado en el párrafo anterior no se iba a realizar. Esto se debió a que principalmente en el primer Sprint tuvimos algunas dificultades en el desarrollo de la aplicación principalmente debido a la falta de Experiencia por lo que se priorizó realizar pruebas funcionales a medida que se desarrollaba y también realizar revisiones o que otro integrante del equipo probara funcionalidades hechas por otro integrante y así encontrar errores.

Sabemos que no es lo mejor en cuanto a obtener un software sin errores, pero priorizamos, por un tema de tiempo, realizar pruebas funcionales.

Cuando se tuvieron todos los requerimientos funcionales establecidos se realizaron algunas pruebas unitarias, pero no se llegó a tener un cien por ciento de cobertura de código.



10.3.7 Pruebas con usuarios

Las pruebas con usuarios serán desarrolladas en un ambiente controlado con los miembros del equipo observando las reacciones de los mismos ante el manejo del sistema.

Luego de que el usuario interactúe con el sistema se le realizarán ciertas preguntas y se le permitirá dar un *feedback* sobre la aplicación.

Aclaración:

Se realizaron pruebas con usuarios al finalizar el tercer Sprint, pero se decidió no parar el desarrollo de la aplicación por un tema de tiempo.

De estas pruebas se obtuvo feedback que nos sirvió para mejorar algunas interfaces.

11. Conclusiones

Este trabajo nos sirvió para obtener bastante experiencia en Android y también para tener una idea de lo que va a ser la tesis final de carrera. Durante el desarrollo de la aplicación fue necesaria mucha investigación propia.

Sabemos que en lo que se refiere a la aplicación se podrían realizar varias mejoras a las interfaces para hacerlas más intuitivas y más vistosas para el usuario. Esto se podría haber hecho en caso de haber tenido más tiempo para el desarrollo.

12.Bibliografía

[1] <<Smartphone OS Market Share, 2016 Q3>>

<http://www.idc.com/promo/smartphone-market-share/os>

[2] <<El país con más cara de perro>>

<http://www.elpais.com.uy/informacion/uruguay-pais-mas-cara-perro.html>

[3] Ian Sommerville, Ingeniería de software, 9na. Edición, 2011.

[4] <<Desarrollo en Cascada Vs. Desarrollo Iterativo e Incremental >>

<http://isecom.blogspot.com.uy/2013/08/desarrollo-en-cascada-vs-desarrollo.html>

[5] <<Metodologías Ágiles>>

<http://comunidad.iebschool.com/metodologiasagiles/general/concepto-metodologias-agiles/>

[5] Mike Cohn, Agile Estimating and Planning, 2006

[6] <<Trello>> <https://trello.com/>

[7] <<Blasmiq Mockups>> <https://balsamiq.com/products/mockups/>

[8] <<Bitbucket>> <https://bitbucket.org/>

[9] <<GitHub>> <https://github.com/>

12. Anexos

12.1 Uso de la herramienta Trello

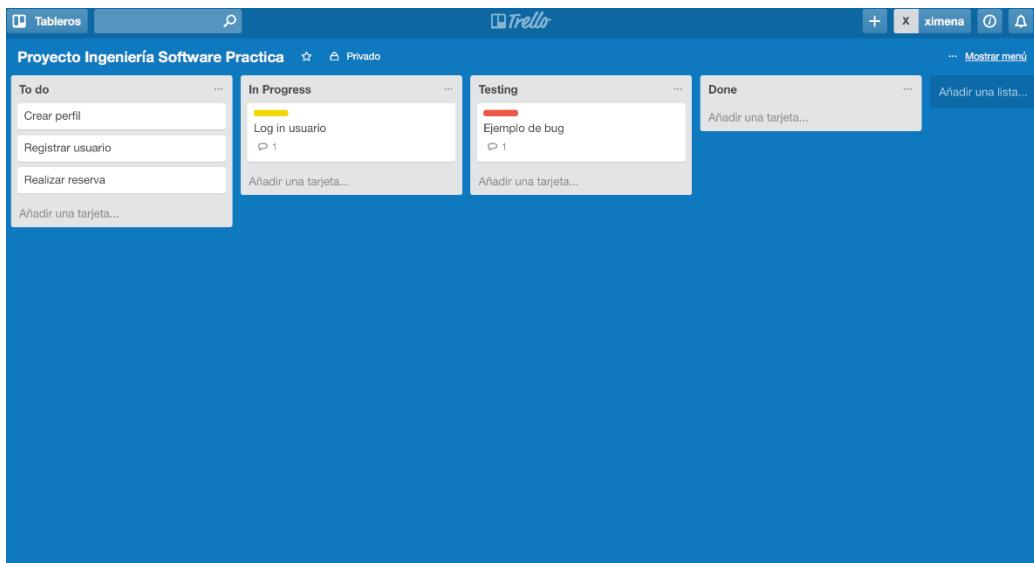
A continuación, se muestra una captura ilustrando la utilización de esta herramienta de gestión, dividiéndose como un tablero *Kanban* en cuatro columnas:

- To Do
- In Progress
- Testing
- Done

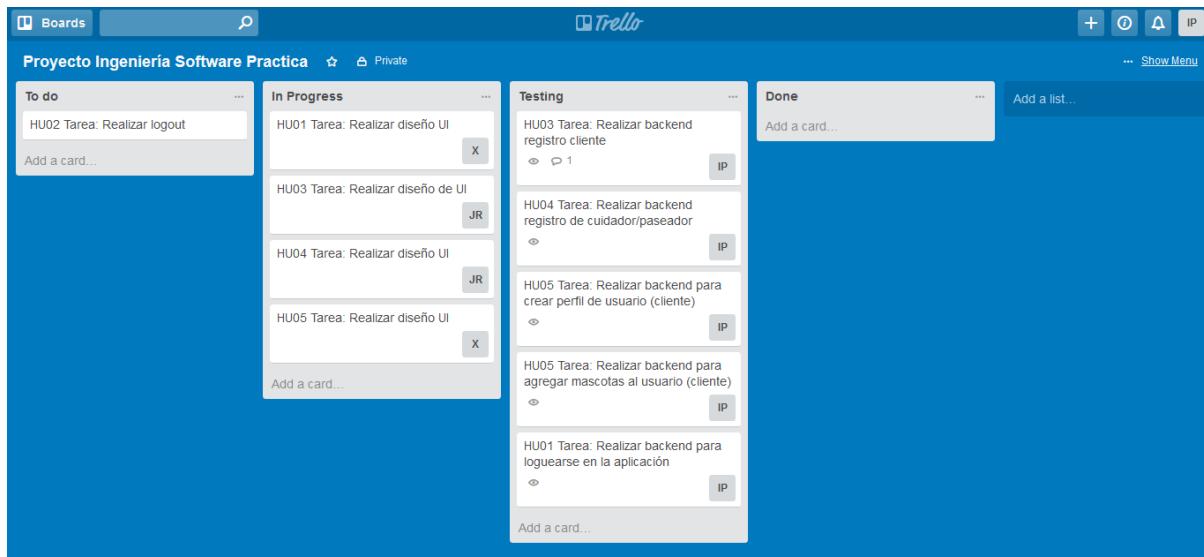
En la columna *To Do* se encuentran especificadas todas las historias de usuario a realizarse durante la etapa de desarrollo, la cual abarca los cuatro sprints, dichas historias de usuario se irán pasando a la columna *In Progress* al momento de comenzar el sprint correspondiente a la misma y asignándose a algún miembro del equipo, al momento de finalizar su implementación se realizarán las pruebas unitarias correspondientes en la columna *Testing* y al culminar con esto se moverán a la columna *Done*.

La herramienta permite crear etiquetas. Para este proyecto se decidió utilizar las siguientes etiquetas:

- Etiqueta de color Rojo: se utiliza para especificar que se encontró un bug.
- Etiqueta color Amarillo: se utiliza para señalar mejoras que se le pueden hacer a una historia de usuario en cuanto a código se refiere. Por ejemplo, luego de un code review.



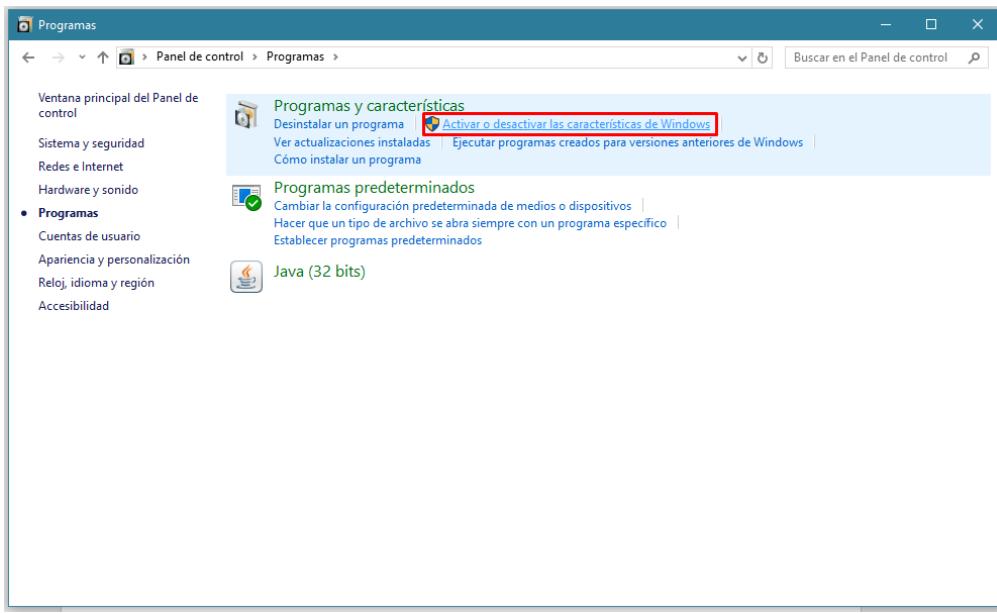
12.1.1 Evidencia de uso de la herramienta Trello



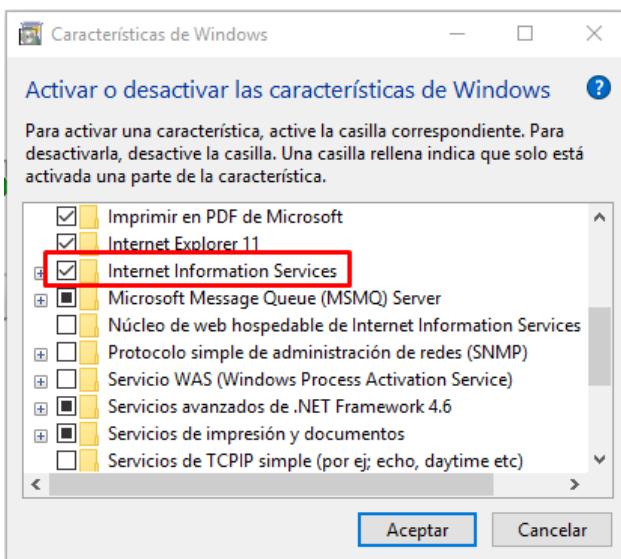
12.2 Manual de instalación

1. Primero que nada, debemos fijarnos que tengamos IIS instalado.

Para esto debemos ir al “Panel de Control”, en la sección “Programas” y seleccionar la opción “Activar o desactivar las características de Windows”.



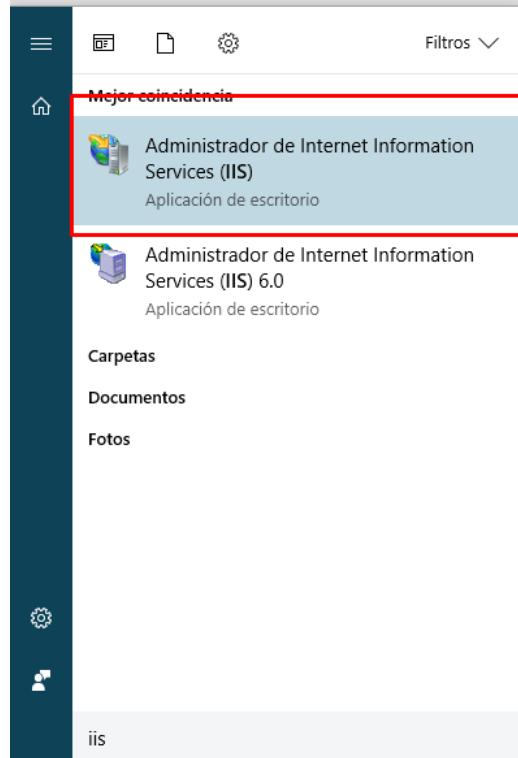
2. Esto nos abrirá una ventana, donde debemos seleccionar todas las características que componen a la opción *Internet Information Services*.



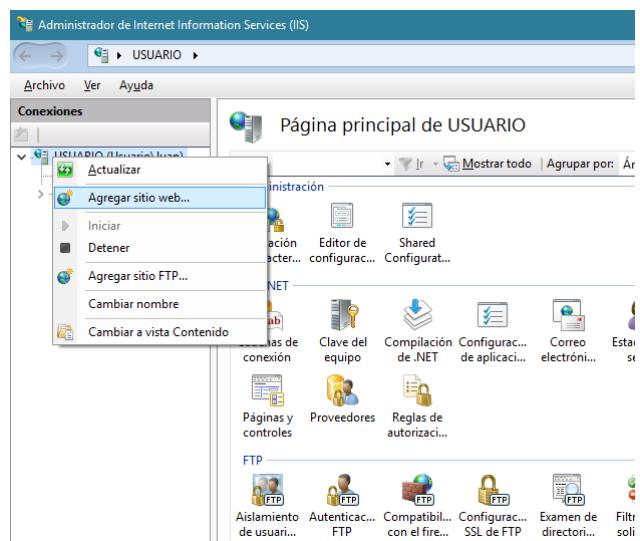
Damos aceptar y se dará pasar a instalar el mismo.

3. El siguiente paso es compilar el *backend* en *release* y copiar la carpeta en que se encuentra el mismo en el directorio “C:\inetpub\wwwroot”.
4. Agregar nuevo sitio desde el IIS Manager

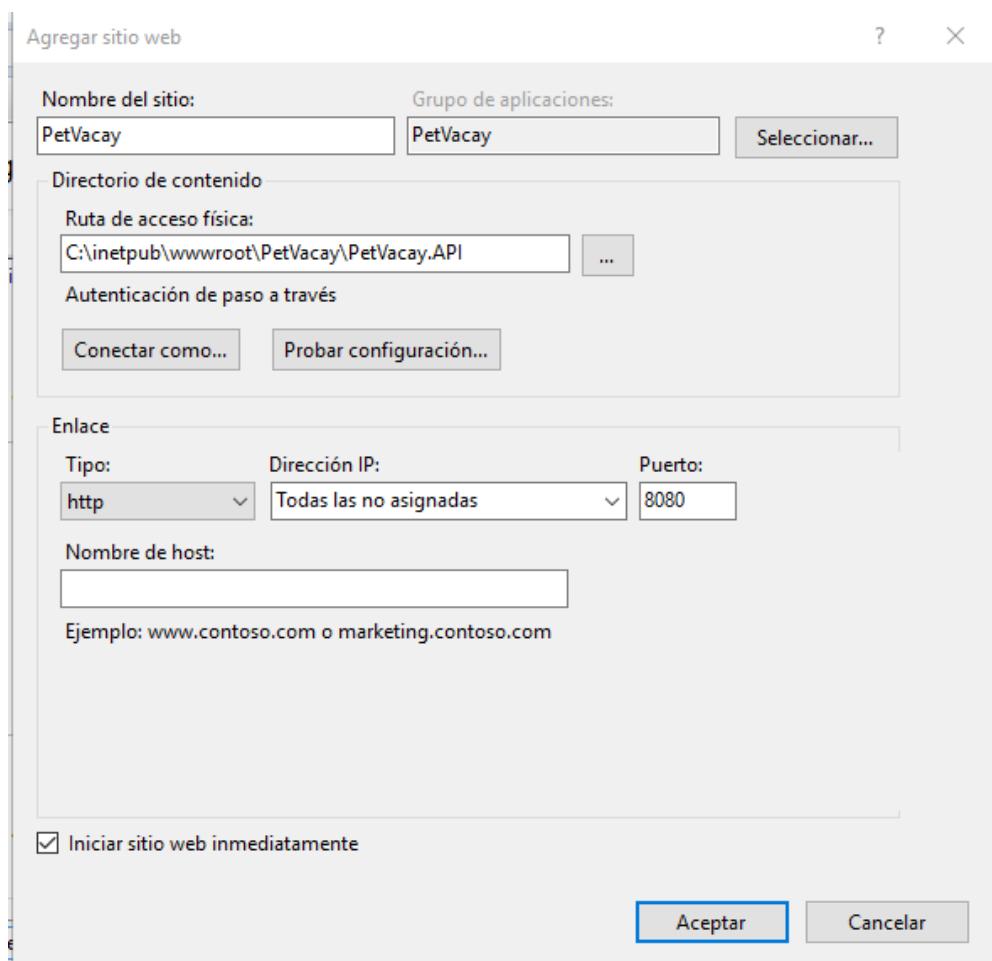
Primero lo buscamos de la siguiente manera:



5. Ahora tenemos que agregar un sitio, para esto debemos hacer click derecho sitios y seleccionamos Agregar sitio web.



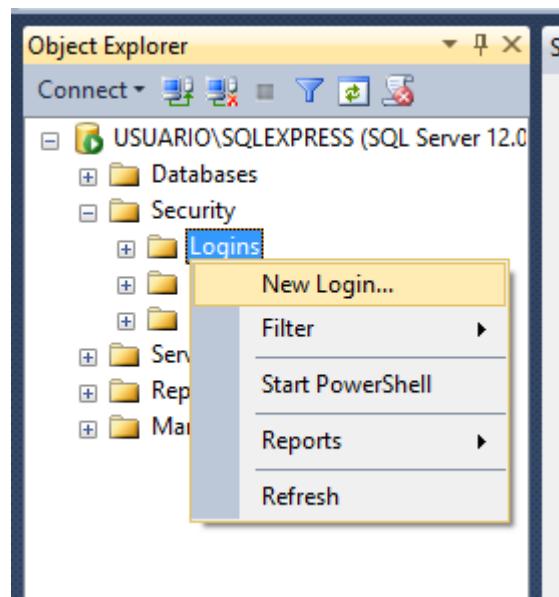
6. Se nos abrirá una pestaña que debemos llenar con los siguientes datos:



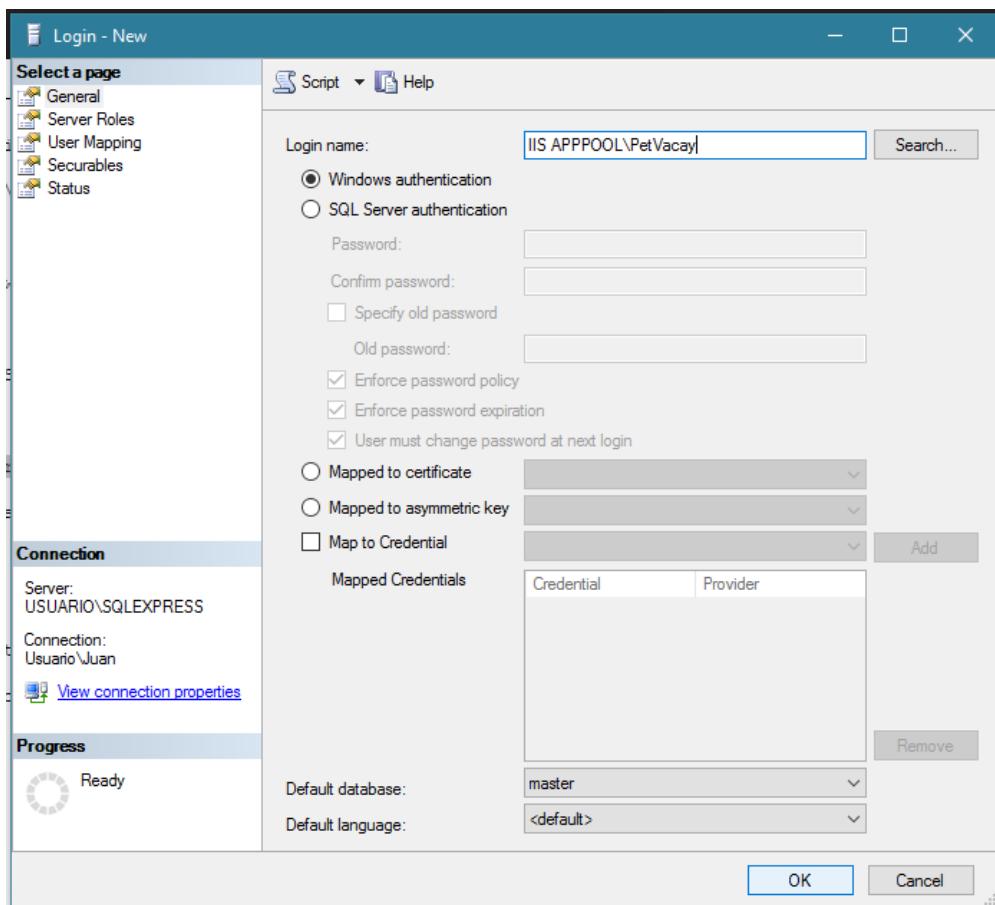
Luego seleccionamos “Aceptar”.

7. Lo siguiente que debemos hacer es asignarle los permisos correspondientes.

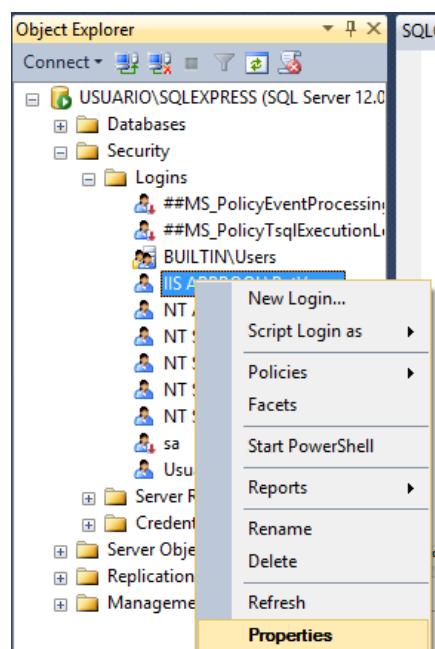
Para esto debemos, hacer lo siguiente:



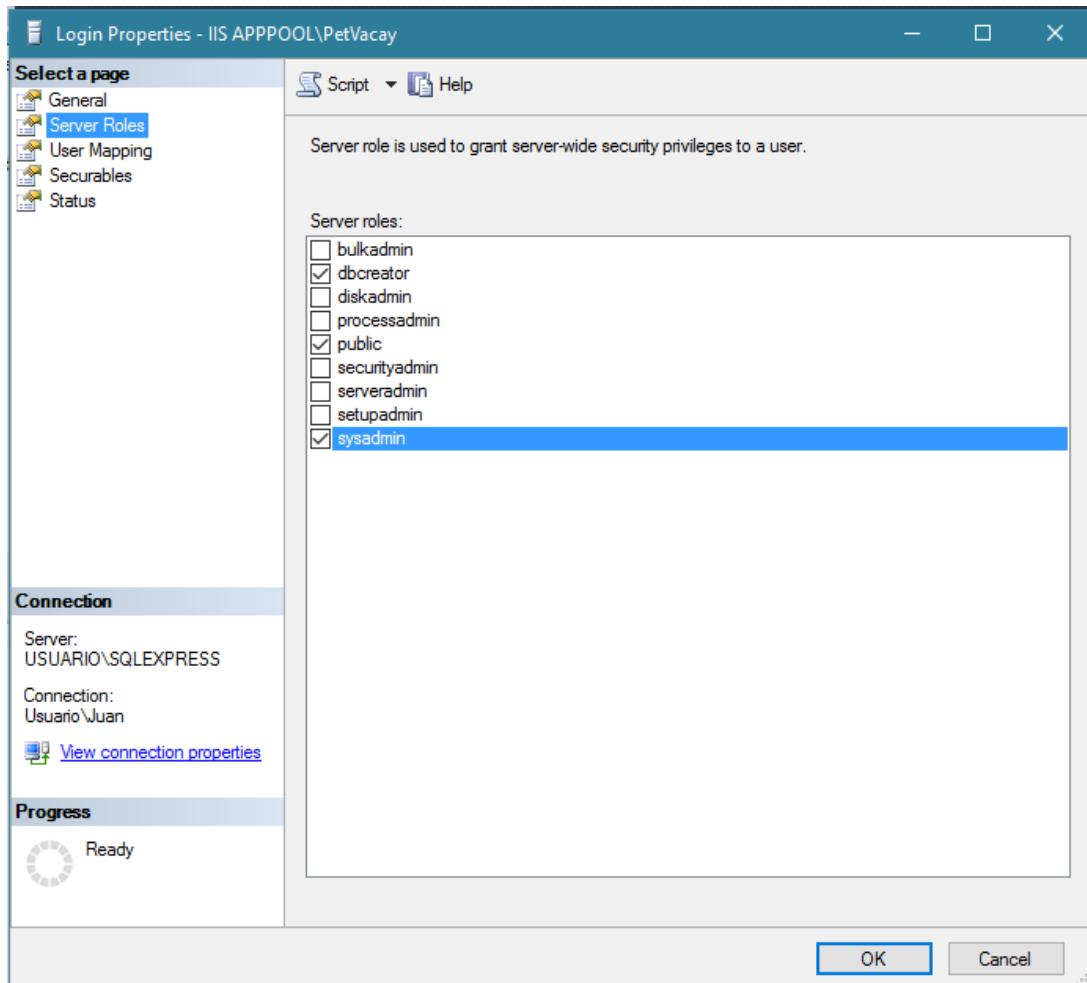
Llenamos con los siguientes datos y damos “Ok”.



8. Hacemos clic derecho en lo que acabamos de crear y seleccionamos “Properties”



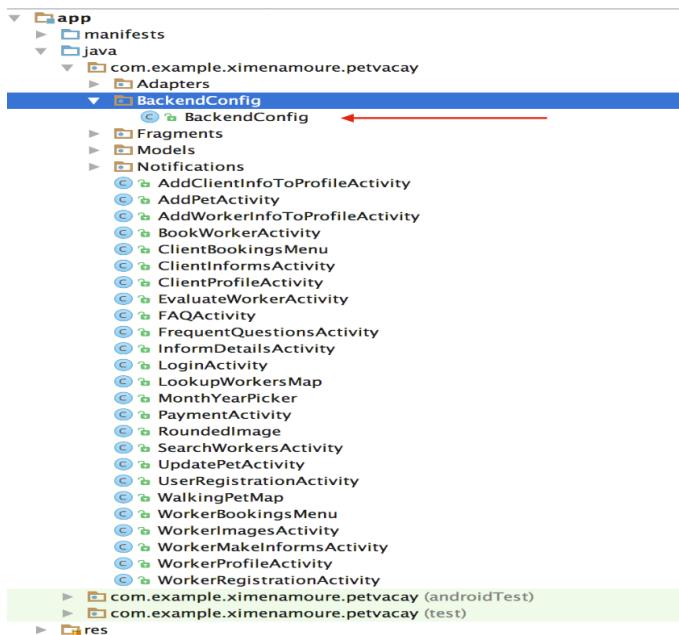
9. Para finalizar seleccionamos la pestaña “Server Roles” y damos los permisos de “dbcreator”, “public” y “sysadmin”.



Luego seleccionamos “Ok”.

10. En Android Studio:

Es necesario cambiar la IP. Para eso vamos a *BackendConfig*



Y cambiamos la IP.

The screenshot shows the code editor for the 'BackendConfig.java' file. The code defines a static class 'BackendConfig' with a static variable 'BackendIP' set to 'http://192.168.1.53:8091/api/'. It also includes a static method 'GetInstance()' which returns an instance of 'BackendConfig'. A red arrow points from the text 'Y cambiamos la IP.' to the line of code 'public static String BackendIP = "http://192.168.1.53:8091/api/";'.

```
BackendConfig.java
package com.example.ximenamoure.petvacay.BackendConfig;

import android.location.LocationManager;

public class BackendConfig {

    private static BackendConfig config;

    public static String BackendIP = "http://192.168.1.53:8091/api/";

    public static String LocationProvider= LocationManager.NETWORK_PROVIDER;

    private BackendConfig() {}

    public static BackendConfig GetInstance()
    {
        if(config == null)
        {
            return new BackendConfig();
        }
        return config;
    }
}
```

11. Si probamos la app en el emulador es necesario cambiar en el BackendConfig el LocationProvider.

Para eso vamos a BackendConfig como en el paso anterior y donde dice

public static String LocationProvider= LocationManager.NETWORK_PROVIDER;

ponemos:

public static String LocationProvider= LocationManager.GPS_PROVIDER;

The screenshot shows the Android Studio interface. On the left, the project structure is visible under the 'app' folder, including 'manifests', 'java', and 'BackendConfig'. The 'BackendConfig.java' file is open in the main editor. The code defines a class 'BackendConfig' with a static variable 'LocationProvider' set to 'LocationManager.NETWORK_PROVIDER'. A red arrow points from the text 'ponemos:' to this line of code. The code also includes a static IP address 'BackendIP' and a static method 'GetInstance()' which returns a new instance of 'BackendConfig' if it's null.

```
BackendConfig.java
package com.example.ximenamoure.petvacay.BackendConfig;

import android.location.LocationManager;

public class BackendConfig {

    private static BackendConfig config;

    public static String BackendIP = "http://192.168.1.53:8091/api/";

    public static String LocationProvider= LocationManager.NETWORK_PROVIDER; ← Red arrow here

    private BackendConfig() {}

    public static BackendConfig GetInstance()
    {
        if(config == null)
        {
            return new BackendConfig();
        }
        return config;
    }
}
```

12. Se debe crear la carpeta para guardar las imágenes “C:\profileImages”, si se desea cambiar esta ruta, se debe ir al backend, “PetVacay.Helpers”, “ImageSaver” y en el método “getUrlOfImage” se cambia la variable “path” por la ruta deseada.

The screenshot shows the 'ImageSaver.cs' file in the 'PetVacay.Helpers' namespace. It contains a method 'getUrlOfImage' that takes a byte array 'profileImage', an employee ID 'eId', and a name prefix 'namePrefix'. The method converts the byte array to a file stream and writes it to the 'C:\profileImages' directory with a .jpg extension. A red arrow points to the line where the path is defined: 'string path = "C:\\profileImages\\\" + namePrefix + eId + ".jpg";'

```
ImageSaver.cs
public string getUrlOfImage(sbyte[] profileImage, string eId, string namePrefix)
{
    byte[] convertedProfileImage = (byte[])profileImage.Cast<byte>();
    string path = "C:\\profileImages\\\" + namePrefix + eId + ".jpg";

    FileStream stream = new FileStream(path, FileMode.Create, FileAccess.Write);
    stream.Write(convertedProfileImage, 0, profileImage.Length);
    stream.Close();

    return path;
}
```