TRABAJO PRÁCTICO INTEGRADOR
PET HERO

UNIVERSIDAD TECNOLÓGICA NACIONAL
UNIDAD ACADÉMICA MAR DEL PLATA

MATERIAS
Metodología de Sistemas I y Laboratorio IV

INTEGRANTES
LINES, IGNACIO

[REPOSITORIO GITHUB]

ÍNDICE

INTRODUCCIÓN

Propósito

El objetivo del presente informe es documentar y relevar los aspectos más importantes sobre el desarrollo del trabajo integrador propuesto por las cátedras. En este mismo se requiere realizar una aplicación cuyo modelo de negocio consiste en que personas puedan brindar el servicio del cuidado de mascotas. Dicho cuidado se trata de una estadía corta a cambio de una remuneración.
Con este objetivo en mente, se procede a ahondar sobre las características de dicho sistema.

Ámbito del Sistema

El sistema será utilizado por aquellos usuarios que precisan que sus mascotas sean cuidadas por profesionales y también por aquellos usuarios que deseen prestar el servicio de cuidado de mascotas. Este sistema está basado en web por lo que facilita su uso, acceso y navegación dentro del mismo sin necesidad de que los usuarios descarguen ningún software.
El software a desarrollar tiene como objetivo principal, facilitar la comunicación y negociación entre dueños de mascotas y potenciales cuidadores. Con este sistema, se desea automatizar el cuidado de mascotas. El sistema provee un sistema de perfiles tanto para cuidadores como dueños de mascotas. En los perfiles de dueños de mascotas (Owner) se puede cargar la información y características de las mascotas y además seleccionar un cuidador (Guardian) en un rango de fechas.
En el sitio web, los Owner podrán consultar la información sobre los Guardians, de forma que seleccionen al que consideren más adecuado para sus mascotas. Por su lado, los Guardians podrán seleccionar su disponibilidad y recibir un pago del 50% adelantado para confirmar la reserva de su servicio.
Cómo el sistema busca facilitar la comunicación entre estos dos tipos de usuarios, se propone la implementación de un chat directo para agilizar la generación de reservas.
El sistema debe utilizarse exclusivamente para la contratación de Keepers de mascotas.

Definiciones, acrónimos y abreviatura

- **Owner:** Tipo de perfil de los dueños de mascotas en el sitio web.
- **Guardian:** Tipo de perfil de los cuidadores de mascotas en el sitio web.
- **DAO:** Data Access Object
- MVC: Model View Controller

DESCRIPCIÓN GENERAL DEL SISTEMA

Perspectiva del producto

Se requiere realizar una aplicación cuyo modelo de negocio consiste en que personas puedan brindar el servicio del cuidado de mascotas. Dicho cuidado se trata de una estadía corta a cambio de una remuneración.

Objetivos del sistema

Facilitar la comunicación, negociación y acuerdo entre usuarios que deseen cuidar mascotas y usuarios que deseen que sus mascotas sean cuidadas.

Ámbito de la aplicación

Navegadores Web.

Arquitectura utilizada

La arquitectura utilizada para realizar la aplicación es la que se vio en la materia de Laboratorio IV, que es la arquitectura de MVC con una conexión hacia la base de datos MySql a través de PDO. Está conformado por los siguientes componentes:

- Modelo: este componente contiene las clases de los objetos, que a su vez cada uno tiene sus Getters y Setters, además del constructor.
- View: el componente View contiene las vistas, y contienen código en html que interactúa con el usuario. Manda la información que recibe del usuario al controller o recibe información de este para mostrarsela al usuario.
- Controller: contiene la lógica del manejo de información recibida del usuario para la base de datos y viceversa. Instancia a las clases DAO para pedir la data guardada en la base de datos, y se la devuelve a las vistas. Cada clase tiene su propio controller que lleva a cabo diferentes métodos para diferentes acciones.
- DAO: la capa DAO es la encargada de conectar la base de datos con el resto de la aplicación. Recibe información de lo que necesita el controller y a través de diferentes queries pedidas de la base de datos, asigna la información a un arreglo o un objeto, y la devuelve a la clase controller.
- MySql: por último, en el MySql se guardan los objetos que se van ingresando en el programa en diferentes tablas, que se conectan a través de claves primarias y foráneas. El DAO manda INSERTS y SELECTS para enviar y pedir los datos guardados.

Lenguajes utilizados

PHP (lógica), HTML (web), CSS (estilizado), MySQL (base de datos).

DEFINICIÓN DE REQUISITOS DEL SISTEMA

Listado de requisitos funcionales

Como principal requisito, se requiere realizar una aplicación cuyo modelo de negocio consiste en que personas puedan brindar el servicio del cuidado de mascotas. Dicho cuidado se trata de una estadía corta a cambio de una remuneración. A partir de esta premisa podemos detectar los requerimientos funcionales del sistema requeridos para el trabajo integrador.

- El sistema deberá permitir que los usuarios que se registren como Keepers.
- El sistema deberá permitir a los usuarios Guardians tener un perfil en el sitio donde se visualiza que tipo de mascota están dispuestos a cuidar (pequeño, mediano o grande) y la remuneración esperada por la estadía.
- El sistema deberá permitir a los usuarios que se registren como Owner.
- El sistema deberá permitir a los usuarios Owner que visualicen su perfil.
- El sistema deberá permitir a los usuarios Owner la habilidad de generar una review sobre el servicio una vez completado el alojamiento.
- El sistema deberá permitir a los usuarios Guardians, definir los días específicos que cuentan con disponibilidad para el cuidado de perros.
- El sistema debe permitir que un Guardian pueda cuidar más de una mascota por estadía pero mientras que sean de la misma raza en el mismo día.
- El sistema deberá permitir que los usuarios Owner puedan crearle un perfil a cada mascota que poseen (perro o gato).
- El sistema deberá permitir por cada perfil de mascota completar su información con: una foto, raza, tamaño, plan de vacunación (como imagen), observaciones generales de la misma y un video de manera opcional.
- El sistema deberá permitir a un Owner seleccionar un Guardian y generar una reserva en el sistema entre las fechas que requiere el servicio.
- El sistema deberá permitir que el Guardian pueda aceptar o rechazar una reserva.
- El sistema deberá permitir envíar un cupón de pago al Owner con el 50% del costo del total de la estadía.
- El sistema deberá revisar si el pago fue realizado para confirmar la reserva.

- El sistema deberá permitir que los usuarios Guardians puedan cuidar tanto gatos como perros.

- El sistema deberá enviar el cupón de pago para un Owner vía email.

Resumen

RF1 - Ingresando nuevo Owner en la aplicación.
RF2 - Ingresando nueva mascota en la aplicación.
RF3 - Consultando mi listado de mascotas (Owner).
RF4 - Ingresar nuevo Guardian.
RF5 - Un Guardian podrá indicar la disponibilidad de estadías.
RF6 - Consultar listado de Guardians cómo Owner.
RF7 - Consultando disponibilidad de Guardians en un rango de fechas.
RF8 - Generando nueva reserva desde un Owner a un Guardian.
RF9 - Consultando mis reservas programadas e históricas como Guardian.
RF10 - Confirmando reserva como Guardian. Se agrega la validación de la raza por estadía.
RF11 - Ingresando nuevo gato (Owner).
RF11 - Generando nuevo cupón de Pago para un Owner.
RF12 - Simulación de pago de cupón (confirmación de reserva).
RF13 - Chat entre Owner y Guardian.
RF14 - Enviando cupón de pago por mail.

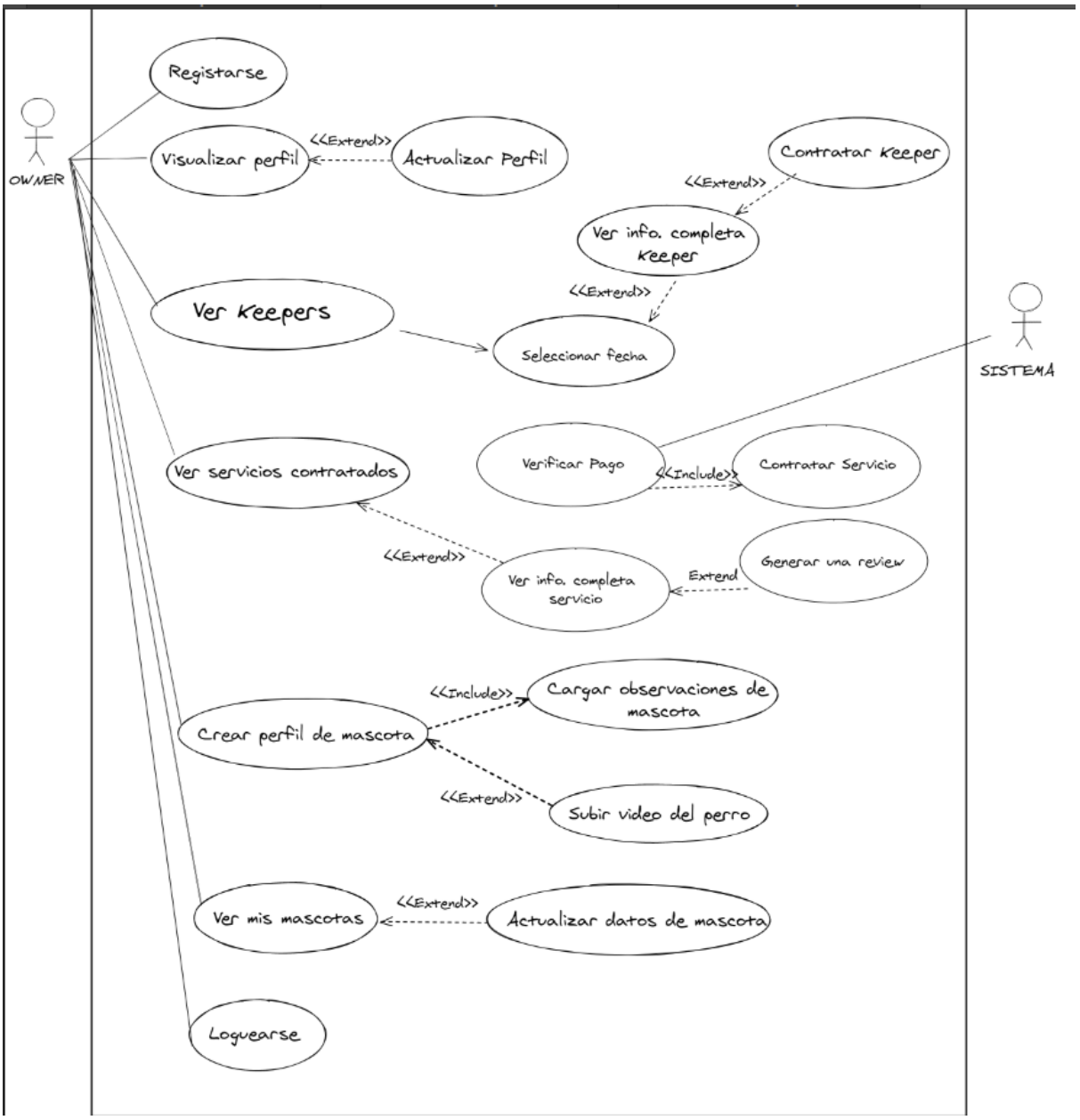
Listado de requisitos no funcionales

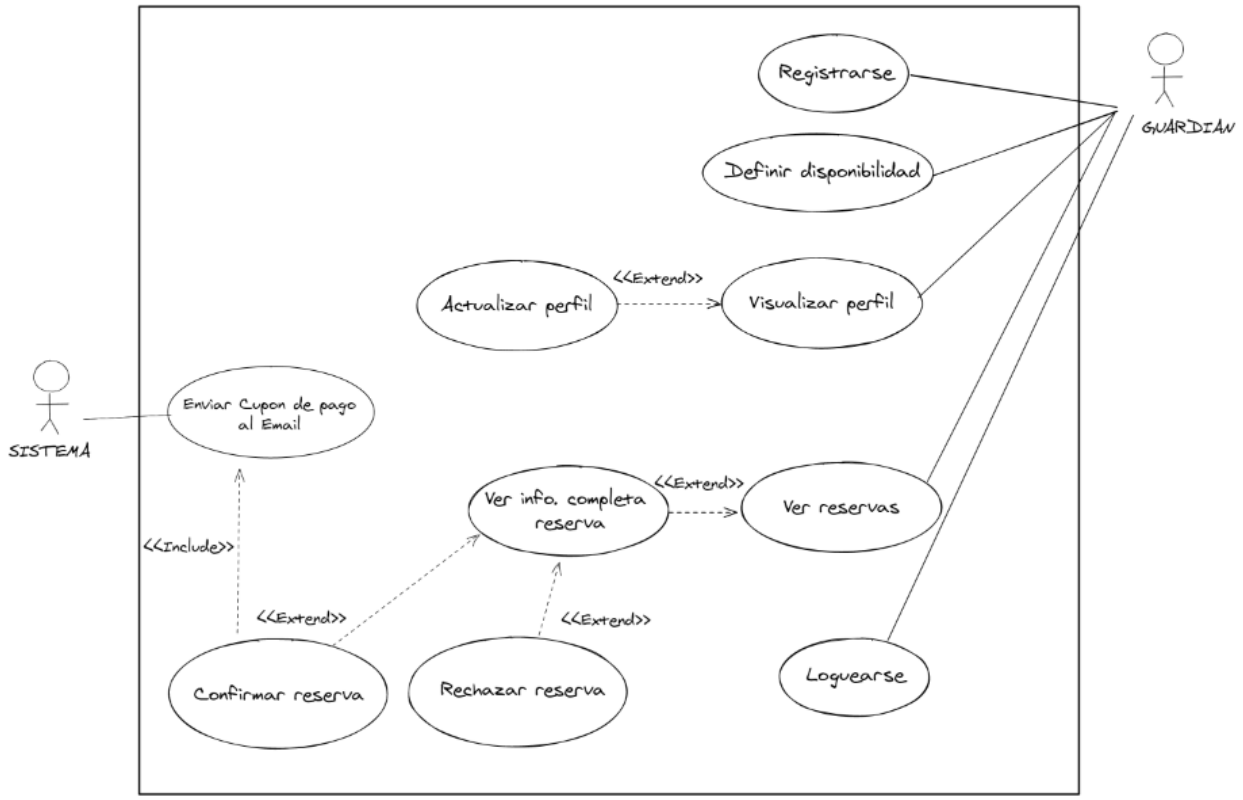
● **Programación en capas de la aplicación** respetando la arquitectura de 3 capas lógicas vista durante la cursada. Esto implica el desarrollo de las clases que representen las entidades del modelo y controladoras de los casos de uso, las vistas y la capa de acceso a datos.

● **Utilización de versionado de código** para el desarrollo.

DIAGRAMA DE CASOS DE USO

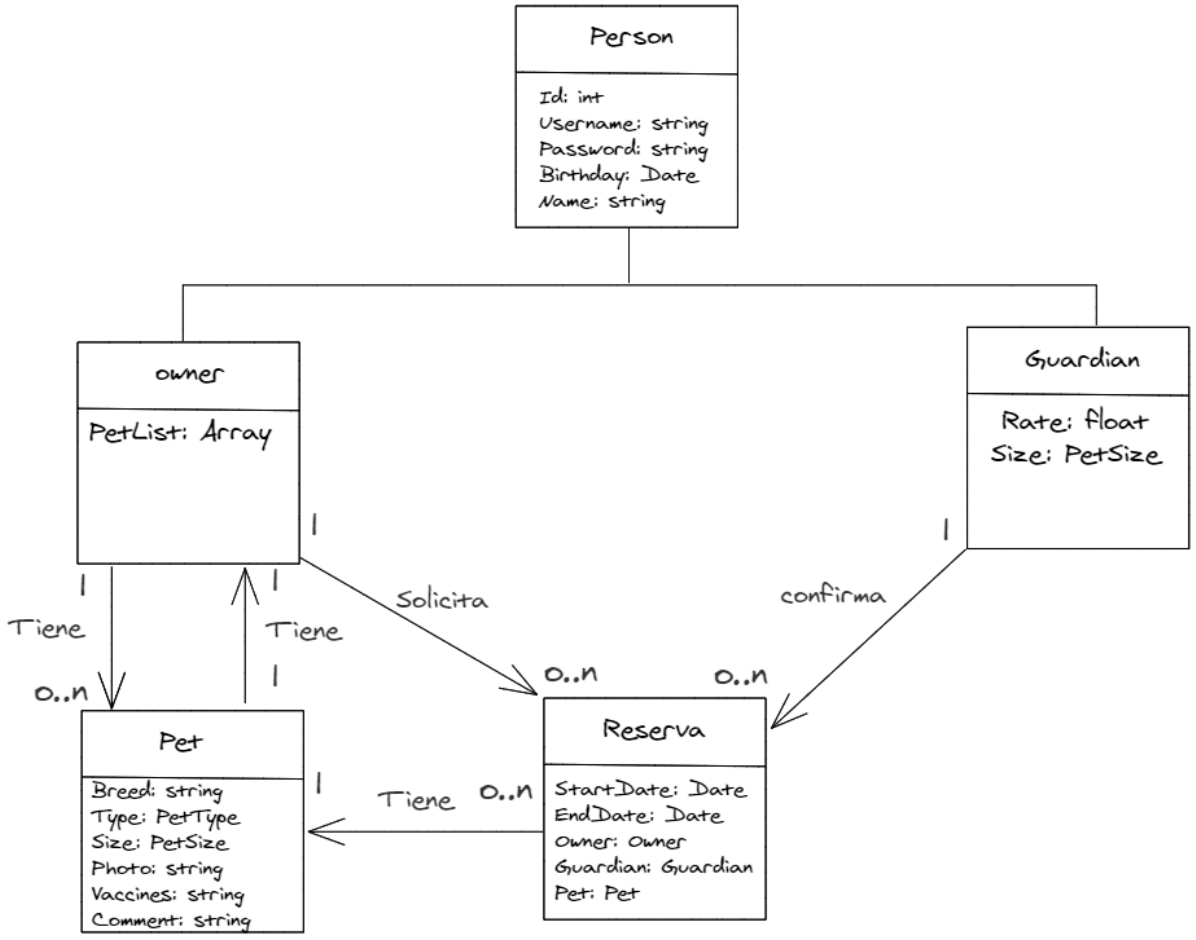
Por cuestiones de legibilidad se decidió realizar dos diagramas de caso de uso, uno para el usuario Owner y otro para el usuario Guardian. Sin embargo es importante destacar que ambos forman parte del mismo sistema y tienen casos en común.
Entre estos casos se destacan: Registrarse, Loguearse, Visualizar Perfil y Actualizar Perfil.
El Sistema es un actor que se encarga de Verificar los pagos y enviar los emails.




Registrarse
GUARDIAN
Definir disponibilidad
Actualizar perfil
<<Extend>>
Visualizar perfil
SISTEMA
Enviar Cupon de pago al Email
Ver info. completa reserva
<<Extend>>
Ver reservas
<<Include>>
<<Extend>>
<<Extend>>
Confirmar reserva
Rechazar reserva
Loguearse

MODELO CONCEPTUAL

En el siguiente modelo se describen los componentes principales del sistema,sus relaciones y los tipos de datos que contienen además de las cardinalidades.
Se observa que Owner y Guardian heredan de la clase Person.
Un Owner puede registrar cero o más mascotas pero una mascota pertenece a un único Owner.
Un Owner puede realizar cero o más reservas pero una reserva pertenece a un único Owner.
Un Guardian puede confirmar cero o más reservas pero una reserva pertenece a un único Guardian.
Una Pet puede tener cero o más reservas pero una reserva pertenece a una única Pet.



ESPECIFICACIÓN DE CASO DE USO

Caso de Uso: “Dueño solicita una reserva a guardián”.

1.Descripción
El caso de uso describe el momento en el que un usuario Owner realiza una reserva con un Guardian.
Actor primario: Usuario tipo Owner.
Actores secundarios: Sistema de Pagos.

2. Escenario de éxito

1. El caso de uso comienza cuando el Owner desea solicitar una reserva a un Keeper.
2. El sistema solicita una fecha en la cual el Owner desee contratar el servicio del Keeper.
3. El usuario ingresa la fecha.
4. El sistema verifica que existan Guardians disponibles en esa fecha.
5. El sistema muestra los Guardians disponibles.
6. El usuario selecciona un Guardian y realiza una reserva con él.
7. Fin de caso de uso.

3. Escenario alternativo

4.1 No existen Guardians disponibles en la fecha seleccionada.
    1. El sistema muestra un mensaje de error.
    2. Fin de caso de uso.

4. Suposiciones

1. El usuario debe haber iniciado sesión en el sistema.

DEUDA DE IMPLEMENTACIÓN

Dadas las circunstancias en las que se desarrolló este trabajo integrador por la falta del resto de integrantes quedan adeudados los siguientes requerimientos funcionales:

RF11 - Generando nuevo cupón de Pago para un Owner.
RF12 - Simulación de pago de cupón (confirmación de reserva).
RF14 - Enviando cupón de pago por mail.

Para generar el cupón de pago se debería implementar una nueva tabla en la base de datos que contenga como Key el ID del Owner (quien realiza el pago) y un ID de reserva. En dicha tabla habría (en primera instancia) un valor booleano que indique si el pago se efectuó o no. Una vez que el Guardian acepta la reserva, el sistema enviaría por email (al del usuario Owner) un link especial con la información del pago creado (esta información estaría como parámetro generado con http_build_query), que una vez clickeado por el usuario le indicaría al sistema que se efectuó exitosamente (podría tratarse de una página web con un mensaje de “Pago efectuado con éxito”).
PHP utiliza la función mail() para enviar un correo electrónico. Esta función requiere tres argumentos obligatorios que especifican la dirección de correo electrónico del destinatario, el asunto del mensaje y el mensaje real, además hay otros dos parámetros opcionales.
Tan pronto como se llame a la función mail, PHP intentará enviar el correo electrónico y devolverá true si tiene éxito o false si falla.

mail(to, subject, message, headers, parameters);

```
<?php
    $to = owner.GetEmail();
    $subject = "Realiza un pago para efectuar la reserva";

    $message = "Efectúa el pago haciendo click en el siguiente link: …";
    $header = "From: pethero@gmail.com \r\n";

    $retval = mail ($to,$subject,$message,$header);

    if( $retval == true ) echo "Mail enviado exitosamente";
?>
```

Una vez realizado el “pago” el sistema indicará en la reserva bajo el estatus de “Pagado”.